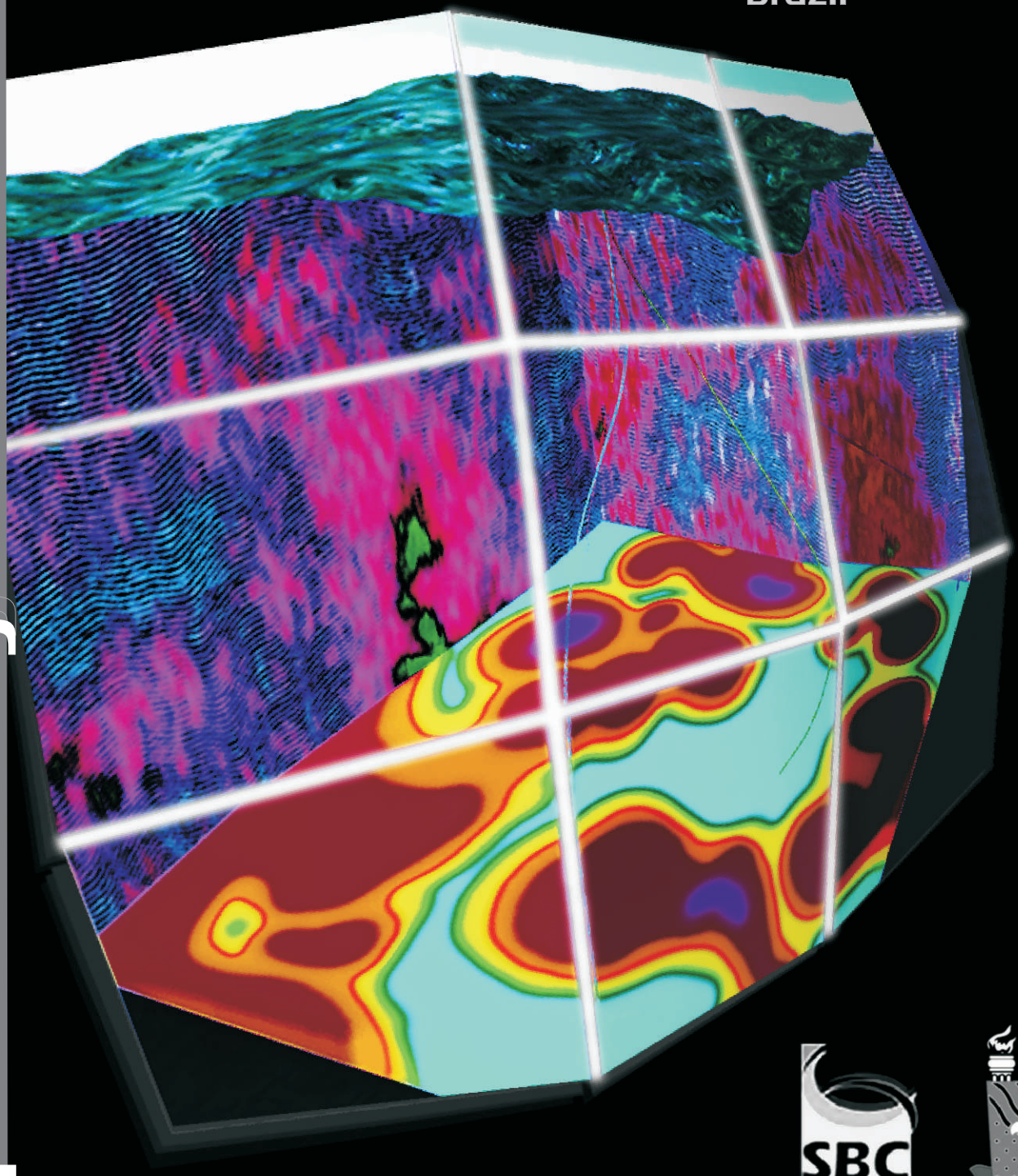
SVR2008

X Symposium on Virtual and Augmented Reality

May 13-16, 2008
João Pessoa - PB
Brazil



proceedings

SBC
Sociedade Brasileira
de Computação

SAPIENTIA ÆDIFICAT

SVR2008

X SYMPOSIUM ON VIRTUAL AND AUGMENTED REALITY

May 13-16, 2008
João Pessoa, PB - Brazil

proceedings

Edition
Sociedade Brasileira de Computação – SBC

Organizers
Alberto Barbosa Raposo
Judith Kelner
Liliane dos Santos Machado
Romero Tori

Realization
Universidade Federal da Paraíba – UFPB

Promotion
Sociedade Brasileira de Computação – SBC
Departamento de Informática – DI/UFPB
Laboratório de Tecnologias para o Ensino Virtual e Estatística - LabTEVE

Proceedings of the X Symposium on Virtual and Augmented Reality – SVR 2008

Organizers

Alberto Barbosa Raposo
Judith Kelner
Liliane dos Santos Machado
Romero Tori

For additional copies of this volume, please contact:
Sociedade Brasileira de Computação
Av. Bento Gonçalves, 9500 - Setor 4 - Prédio 43412 - Sala 219
Bairro Agronomia - CEP 91.509-900 - Porto Alegre - RS, Brasil
Fone/fax: +55 (51) 3308-6835/ +55 (51) 3308-7142
http://www.sbc.org.br

CIP - Biblioteca Setorial de Informática da PUC-Rio

Symposium on Virtual Reality (10.: 2008, João Pessoa, PB)
S989 Proceedings of the X Symposium on Virtual Reality, João Pessoa, PB, May 13-16, 2008 / organizers Alberto Barbosa Raposo, Judith Kelner, Liliane dos Santos Machado [e] Romero Tori. - Porto Alegre: Sociedade Brasileira de Computação, 2008.
xx, 389 p. ; il. (col.)
Inclui referências bibliográficas.

SVR 2008
Realização: Universidade Federal da Paraíba, Departamento de Informática.
ISBN 857669167-1

1. Realidade Virtual – Congressos. I. Raposo, Alberto Barbosa.
II. Kelner, Judith. III. Machado, Liliane dos Santos. IV. Tori, Romero.
V. Universidade Federal da Paraíba. Departamento de Informática.

contents

Supporting Organizations

Sponsoring Organizations

Message from the Program Chairs

The Brazilian Symposium on Virtual and Augmented Reality celebrates its 10th edition in 2008. The first scientific event dedicated to gathering Brazilian researchers in VR – the I Workshop on Virtual Reality – took place in São Carlos, SP, in 1997, under the leadership of Professor Claudio Kirner. In order to celebrate the ten editions and to report part of this history, SVR 2008 counts on Professor Kirner as a keynote speaker and as author of an invited paper about the history and evolution of VR in Brazil.

Currently in its tenth edition, SVR has its space and importance consolidated in the Brazilian scenario. In the last few years, SVR has embraced other technologies related to VR, such as Augmented Reality, and expanded its scope to become an internationally recognized event. The international participation both at the program committee and at the technical sessions has expressively increased. SVR has also been receiving some of the internationally most prominent researchers in the field, such as Profs. Doug Bowman, Don Brutzman and Hiroaki Yano, the international keynote speakers this year. This international insertion of SVR is important not only for the Brazilian scientific community to get in touch with prominent international groups, but also to make our hardworking community internationally visible. Therefore, we continue the effort towards encouraging submissions in English and programming technical sessions entirely presented in English.

The process of full papers' evaluation was improved this year. Each paper was evaluated by 4 reviewers in a double-blind process and, for the first time in SVR, the authors had a rebuttal period, when they could see the reviews and clarify doubts or question the evaluations before the final acceptance decision. This contributed for a more transparent and precise evaluation process. We received 69 full paper submissions, and accepted 24 (34.78%). Responsible for this difficult mission, the program committee was formed by specialists with recognized actuation in VR, AR, and related areas. The program committee increased from 61 (22 international) members in 2007 to 85 (28 international) this year. Among the 24 accepted full papers, 9 were written in Portuguese and 15 in English, being 4 of them from international authors.

While the full papers program has excellence as its main goal, the selection of short papers tried to favor the merit, and not the competition. With a submission calendar separate to that of full papers, we received 49 short paper submissions and selected the 31 that presented quality, merit and adherence to the purpose of the SVR. Each short paper was evaluated by 3 reviewers. Following the innovation introduced last year, papers are presented in thematic sessions that include both full and short papers, valorizing the latter.

Some innovations of the printed proceedings this year are the color pages and the inclusion of abstracts of some of other activities of SVR 2008, such as keynote speeches, minicourses and the pre-symposium.

Finally, we would like to thank the members of the program committee, the ad hoc reviewers, the sponsoring institutions and companies, the important support of the Brazilian Computer Society (SBC) and its Special Committee on VR (CERV), represented by its president, Romero Tori, the organizing committee of SVR 2008, led by Liliane Machado, and the keynote speakers. Very special thanks are addressed to the authors that submitted their work, since they are the reason for the existence of SVR.

The material in these proceedings is very representative of the wide scope of technologies, methodologies, and applications of virtual and augmented environments. We wish you, reader, a profitable and pleasant "immersive" travel through the material presented here. Enjoy!

Alberto Raposo – Program Committee Chair
Judith Kelner – Program Committee Co-chair

João Pessoa, May 2008.

Mensagem dos Coordenadores do Comitê de Programa

O Simpósio Brasileiro de Realidade Virtual e Aumentada comemora sua décima edição em 2008. O primeiro evento científico dedicado a reunir pesquisadores brasileiros da área de RV – o I Workshop em RV – aconteceu em São Carlos, SP, em 1997, sob a liderança do Prof. Claudio Kirner. Para celebrar as dez edições e contar parte dessa história, o SVR 2008 tem como palestrante convidado o Prof. Kirner, que também assina um artigo convidado sobre a história e evolução da RV no Brasil.

Atualmente em sua décima edição, o SVR já consolidou seu espaço e importância no cenário científico brasileiro. Nos últimos anos, o SVR passou a abranger outras tecnologias relacionadas a RV, tais como a realidade aumentada, e também aumentou seu escopo, visando se tornar um evento reconhecido internacionalmente. A participação internacional tanto no comitê de programa quanto nas sessões técnicas tem aumentado significativamente. O SVR também tem recebido alguns dos mais importantes pesquisadores internacionais na área, como os Profs. Doug Bowman, Don Brutzman, e Hiroaki Yano, os palestrantes convidados deste ano. Essa inserção internacional do SVR é importante não só para que a comunidade brasileira entre em contato com reconhecidos grupos internacionais, mas também para aumentar a visibilidade internacional da nossa comunidade. Portanto, continuamos o esforço no sentido de encorajar submissões em inglês e programar sessões técnicas apresentadas em inglês.

O processo de revisão de artigos completos foi melhorado este ano. Cada artigo foi avaliado por 4 revisores e, pela primeira vez, houve um período de *rebuttal*, quando os autores puderam ver as revisões de seus artigos e esclarecer dúvidas ou questionar as revisões antes da decisão final sobre a aceitação. Isso contribuiu para um processo de avaliação ainda mais preciso e transparente. Foram recebidas 69 submissões de artigos completos, dos quais 24 (34,78%) foram aceitos. Responsável por essa difícil missão, o comitê de programa foi composto por especialistas com reconhecida atuação nas áreas relacionadas. O comitê de programa aumentou de 61 nomes em 2007 (22 internacionais) para 85 este ano (28 internacionais). Dentre os 24 artigos completos aceitos, 9 foram escritos em português e 15 em inglês, sendo 4 deles de autores internacionais.

Enquanto a seleção de artigos completos procura a excelência, a seleção de artigos resumidos tende a favorecer o mérito ao invés da competição. Com um calendário de submissão diferente dos artigos completos, foram recebidas 49 submissões de artigos resumidos, dos quais foram selecionados 31 que apresentaram qualidade, mérito e aderência aos propósitos do SVR. Cada artigo foi avaliado por 3 revisores. Seguindo a inovação introduzida ano passado, os artigos serão apresentados em sessões temáticas, incluindo artigos completos e resumidos, valorizando os últimos.

Algumas novidades dos anais este ano são as páginas coloridas e a inclusão de resumos de algumas das outras atividades do SVR, tais como as palestras convidadas, os minicursos e o pré-simpósio.

Finalmente, agradecemos aos membros do comitê de programa, a todos os revisores, às instituições e empresas que apoiaram o evento, ao importante apoio da SBC (Sociedade Brasileira de Computação) e a sua Comissão Especial de RV (CERV), representada por seu coordenador, Romero Tori, ao comitê organizador, liderado pela Profa. Liliane Machado, e aos palestrantes convidados. Nossos agradecimentos especialíssimos também a todos os autores que submeteram seus trabalhos ao SVR, pois eles são a razão da existência do evento.

O material contido nesse volume é representativo do grande escopo de tecnologias, metodologias e aplicações de ambientes virtuais e aumentados. Desejamos a você, leitor, uma proveitosa e agradável viagem "imersiva" através do material aqui apresentado.

Alberto Raposo – Program Committee Chair
Judith Kelner – Program Committee Co-chair

João Pessoa, Maio de 2008.

Message from the Organizing Chairs

The tenth edition of the SVR presents the consolidation of the community of virtual and augmented reality in Brazil: initially started as a workshop, the first national event on virtual reality happened in 1997. Ever since, the community grew and new research groups were formed. In the year of 2000 the SBC Special Committee in Virtual Reality (CERV) was created. In the present year of 2008, the Federal University of Paraíba (UFPB) has the pleasure of hosting the tenth edition of SVR in João Pessoa. João Pessoa is known as the eastern point of Americas, "the place where the sun rises first", and is the second greenest city of the world, intermixed with traces of the Atlantic forest.

In this tenth edition, SVR presents several keynote speeches and mini-courses addressed to people of all levels of knowledge in the area, allowing them learning and updating themselves about recent advances in VR and AR in Brazil and in the world. Besides the pre-symposium, devoted for beginners in the area, the conference will have eight mini-courses about actual subjects in the area, nine technical sessions with peer-revised papers, graduate and undergraduate workshops, as well as projects exhibition. It was programmed three international and one national keynote speeches. For the first time in the history of SVR, a researcher from Japan, a vanguard country of technologies and innovation, will utter a speech. Thus, in addition of Prof. Hiroaki Yano, there will be keynote speeches of professors Doug Bowman and Don Brutzman, both from USA and authors of important books in VR area. In the national keynote speech from Prof. Claudio Kirner, the responsible by first workshop about VR occurred in Brazil, will be presented the history of VR in Brazil and its perspectives. This way, we are sure that the event will reach all participants expectations.

We would like to take advantage of this opportunity to thank the conference sponsors, who made this SVR possible: Petrobras; Secretary of Science, Technology and Environment (SECTMA) of the State of Paraíba; Ministry of Science and Technology through the National Council for Scientific and Technological Development (CNPq); Ministry of Education through the Coordination for the Improvement of High Education Personnel (CAPES), Federal University of Paraíba (UFPB) and Absolut Technologies.

We also thank all reviewers, chairs and co-chairs whose work guaranteed the quality of each activity. Specially, we would like to thank the Brazilian Computer Society (SBC) and its team; the Special Committee in Virtual Reality (CERV); the Federal University of Paraíba, through the Rector's Office; the Center of Exact and Natural Sciences and the Department of Informatics for the support. Particularly, we would like to thank the Laboratory of Technologies for Virtual and Statistics Education (LabTEVE) whose coordinator, Prof. Ronei Marcos de Moraes, and researchers contributed so much with the organization for the success of this conference: you are great!

We hope you have a fruitful conference and an enjoyable stay in João Pessoa!

With best regards,

Liliane S. Machado – General Chair
Romero Tori – General Co-Chair

João Pessoa, May 2008.

Mensagem dos Organizadores

A décima edição do SVR mostra a consolidação da comunidade de realidade virtual e aumentada no Brasil: inicialmente como um workshop, em 1997 ocorreu o primeiro evento de RV no país. Desde então a comunidade cresceu e novos grupos se formaram. No ano de 2000 foi criada oficialmente a Comissão Especial de Realidade Virtual junto à Sociedade Brasileira de Computação. Neste ano de 2008, a Universidade Federal da Paraíba tem o prazer de sediar em João Pessoa a décima edição do SVR. João Pessoa é conhecida como o ponto mais oriental das Américas, "onde o Sol nasce primeiro", e é a segunda cidade mais verde do mundo, entrecortada por porções da mata Atlântica.

Em sua décima edição, o SVR trará palestras e mini-cursos voltados a interessados de todos os níveis, permitindo-lhes conhecer e se atualizar sobre os avanços da Realidade Virtual no Brasil e no mundo. Além do pré-simpósio, voltado aos novatos na área, o evento trará oito mini-cursos com conteúdos atuais, nove sessões técnicas com trabalhos rigorosamente selecionados, workshops de teses, dissertações e trabalhos de iniciação científica, bem como exposições de projetos. Para este ano estão programadas quatro palestras: três internacionais e uma nacional. Pela primeira vez na história do SVR haverá a palestra de um pesquisador do Japão, um país de vanguarda em tecnologia e inovação. Assim, além do Professor Hiroaki Yano, haverá palestras dos professores Doug Bowman e Don Brutzman, ambos dos EUA e autores de importantes livros na área de realidade virtual. Como palestrante nacional teremos o professor Claudio Kirner, responsável pela realização em 1997 do primeiro evento brasileiro de realidade virtual, que discorrerá sobre o histórico e perspectivas da área no Brasil. Sendo assim, temos certeza que o evento alcançará as expectativas de todos os participantes.

Gostaríamos de aproveitar esta oportunidade para agradecer aos patrocinadores que permitiram a realização do evento: à Petrobras; à Secretaria de Ciência, Tecnologia e Meio Ambiente (SECTMA) do Estado da Paraíba; ao Ministério de Ciência e Tecnologia através do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq); ao Ministério da Educação através da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES); à Universidade Federal da Paraíba e à Absolut Technologies.

Agradecemos também a todos aqueles que colaboraram como revisores e coordenadores (chairs e co-chairs), garantindo a qualidade das atividades. Em especial, gostaríamos de agradecer à SBC e sua equipe; à Comissão Especial de Realidade Virtual (CERV); à Universidade Federal da Paraíba, através da sua Reitoria; ao Centro de Ciências Exatas e da Natureza e ao Departamento de Informática pelo apoio ao evento. Particularmente, agradecemos ao Laboratório de Tecnologias para o Ensino Virtual e Estatística (LabTEVE) cujo coordenador, Professor Ronei Marcos de Moraes, e integrantes tanto colaboraram na organização para o sucesso deste evento: vocês são ótimos!

Esperamos que todos tenham um evento proveitoso e uma ótima estadia na cidade de João Pessoa!

Cordialmente,

Liliane S. Machado – General Chair
Romero Tori – General Co-Chair

João Pessoa, Maio de 2008.

SVR 2008 Chairs

Organizing Committee:

General Chair: Liliane S. Machado (UFPB)
General Co-chair: Romero Tori (USP and SENAC/SP)

Program Committee:

Chair: Alberto Raposo (PUC-RJ)
Co-chair: Judith Kelner (UFPE)

Pre-Symposium Chairs:

Robson Siscoutto (UNIC)
Rosa Costa (UERJ)

Technical Minicourses Chairs:

Veronica Teichrieb (UFPE)
Fatima Nunes (UNIVEM)

Workshop of Undergraduate Works Chair:

Alexandre Cardoso (UFU)

Workshop of Thesis and Dissertations Chair:

Edgard Lamounier (UFU)

Steering Committee:

Alexandre Cardoso (UFU)
Claudio Kirner (UNIMEP and UNASP)
Márcio Pinho (PUC-RS)
Romero Tori (USP and SENAC/SP)

Program Committee

Alan Craig	NCSA	USA
Albert Rizzo	University of Southern California	USA
Alberto Raposo	PUC-Rio	Brazil
Alexandre Cardoso	UFU	Brazil
Ana Paula Melchiori	UNICAMP and UNESP-Rio Claro	Brazil
Andre Hinkenjann	Bonn-Rhein-Sieg Univ. of Applied Sciences	Germany
Anselmo Montenegro	UFF	Brazil
Anselmo Paiva	UFMA	Brazil
Antonio Valério Netto	Cientistas Associados	Brazil
Antonio Carlos Sementille	UNIVEM and UNESP	Brazil
Bianchi Meiguins	UFPA	Brazil
Bruno Raffin	INRIA	France
Carla Freitas	UFRGS	Brazil
Carlos Luiz Nunes dos Santos	UFRJ	Brazil
Claudio Kirner	UNIMEP and UNASP	Brazil
Claudio Pinhanez	IBM	USA
Creto Vidal	UFC	Brazil
Daniel Thalmann	EPFL	Switzerland
Daniela Trevisan	UFRGS	Brazil
Dirk Reiners	University of Louisiana at Lafayette	USA
Doug Bowman	Virginia Polytechnic Institute and State Univ.	USA
Edgard Lamounier	UFU	Brazil
Ednaldo Brigante Pizzolato	UFSCar	Brazil
Eduardo Albuquerque	UFG	Brazil
Enio Russo	Petrobras	Brazil
Fatima Nunes	UNIVEM	Brazil
Fernando Osório	UNISINOS	Brazil
Gabriel Zachmann	Clausthal Technical University	Germany
Gernot Goebbels	fleXilution GmbH	Germany
Gerson Cunha	UFRJ	Brazil
Hafid Niniss	NRCD	Japan
Hank Kaczmarsk	UIUC	USA
Ildeberto Rodello	UNIVEM	Brazil
Jaime Sanchez	Universidade do Chile	Chile
Jauvane Oliveira	LNCC	Brazil
Joaquim Cavalcante-Neto	UFC	Brazil
Joaquim Jorge	IST/UTL	Portugal
José Remo F. Brega	UNESP-Bauru	Brazil
Judith Kelner	UFPE	Brazil
Léo Magalhães	UNICAMP	Brazil
Liliane Machado	UFPB	Brazil
Luciana Nedel	UFRGS	Brazil
Luciano Reis	Petrobras	Brazil
Luciano Silva	Universidade Prebiteriana Mackenzie	Brazil
Luciano Soares	PUC-Rio	Brazil
Luiz Gonzaga da Silveira Jr	UNISINOS	Brazil

Manfred Bogen	Fraunhofer IAIS	Germany
Manoel Ribeiro	UFPA	Brazil
Manuel M. Oliveira Neto	UFRGS	Brazil
Marcelo Cohen	PUCRS	Brazil
Marcelo Gattass	PUC-Rio	Brazil
Marcelo de Paiva Guimarães	Fac. Campo Limpo Paulista/Centro Univ.	Brazil
Marcelo da Silva Hounsell	UDESC	Brazil
Marcelo Zuffo	USP	Brazil
Márcio Pinho	PUCRS	Brazil
Marcos Wagner Souza Ribeiro	ULBRA - Itumbiara	Brazil
Maria Ferreira	USP	Brazil
Maria Andréia Rodrigues	UNIFOR	Brazil
Mario Kubo	USP	Brazil
Mark Billinghurst	University of Canterbury	New Zealand
Mark Green	University of Ontario Institute of Technology	Canada
Martin Goebel	fleXilution GmbH	Germany
Masahiko Inami	The University of Electro-Communications	Japan
Michael Moshell	University of Central Florida	USA
Miguel Dias	Microsoft	Portugal
Nadia Magnenat-Thalmann	University of Geneva	Switzerland
Pablo Figueroa	UNIANDES	Colombia
Paul Sharkey	University of Reading	UK
Paulo Bressan	Universidade Prebiteriana Mackenzie	Brazil
Paulo Cezar Carvalho	IMPA	Brazil
Ramesh Raskar	Mitsubishi Electric Research Labs	USA
Roberto Cezar Bianchini	USP	Brazil
Robson Siscoutto	Universidade de Cuiabá	Brazil
Romero Tori	USP and SENAC/SP	Brazil
Ronei Moraes	UFPB	Brazil
Rosa Maria Costa	UERJ	Brazil
Sabine Coquillart	INRIA	France
Sérgio Pellegrino	ITA	Brazil
Selan dos Santos	UFRN	Brazil
Shervin Shirmohammadi	University of Ottawa	Canada
Soraia Musse	PUCRS	Brazil
Tereza Kirner	UNIMEP	Brazil
Terrence Fernando	Salford University	UK
Veronica Teichrieb	UFPE	Brazil
Waldemar Celes	PUC-Rio	Brazil

Reviewers

Alan Craig
Albert Rizzo
Alberto Raposo
Alexandre Cardoso
Ana Paula Melchiori
Anderson Maciel
Andre Hinkenjann
Anselmo Montenegro
Anselmo Paiva
Antonio Valerio Netto
Antonio Carlos Sementille
Aristófanes Silva
Asla Sá
Börje Karlsson
Bianchi Meiguins
Bruno Raffin
Carla Freitas
Carlos Dietrich
Claudio Kirner
Claudio Pinhanez
Creto Vidal
Daniel Thalmann
Daniela Trevisan
Diana Adamatti
Dirk Reiners
Doug Bowman
Edgard Lamounier
Ednaldo Brigante Pizzolato
Eduardo Albuquerque
Enio Russo
Fatima Nunes
Felipe Carvalho
Fernando Osório
Fernando Trebien
Gabriel Zachmann
Gernot Goebbels
Hafid Niniss
Hank Kaczmarsk
Ildeberto Rodello
Jaime Sanchez
Jauvane Oliveira
Joaquim Cavalcante-Neto
Joaquim Jorge
Jose Remo F. Brega
Judith Kelner
Léo Magalhães
Leandro Fernandes
Liliane Machado
Lucas Teixeira
Luciana Nedel
Luciane Fraga
Luciano Reis
Luciano Silva
Luciano Soares
Luiz Gonzaga da Silveira Jr
Maicon Filippsen
Manfred Bogen
Manoel Ribeiro
Manuel Loaiza
Marcelo Cohen
Marcelo Gattass
Marcelo de Paiva Guimarães
Marcelo da Silva Hounsell
Marcio Pinho
Marcos Slomp
Marcos Wagner Souza Ribeiro
Maria Ferreira
Maria Andréia Rodrigues
Mario Kubo
Mark Billinghurst
Mark Green
Michael Moshell
Nadia Magnenat-Thalmann
Pablo Figueroa
Paul Sharkey
Paulo Bressan
Paulo Cezar Carvalho
Rafael Rocha
Renato Silveira
Roberto Cezar Bianchini
Robson Siscoutto
Rodrigo de Toledo
Romero Tori
Ronei Moraes
Rosa Maria Costa
Sabine Coquillart
Sérgio Pellegrino
Selan dos Santos
Shervin Shirmohammadi
Silvano Malfatti
Soraia Musse
Veronica Teichrieb
Vitor Pamplona
Waldemar Celes
Wendel Silva

SBC – Sociedade Brasileira de Computação

Presidente: José Carlos Maldonado (ICMC - USP)
Vice-presidente: Virgílio Augusto Fernandes Almeida (UFMG)

Diretorias

Administrativa: Carla Maria Dal Sasso Freitas (UFRGS)
Finanças: Paulo Cesar Masiero (ICMC - USP)
Eventos e Comissões Especiais: Marcelo Walter (UFPE)
Educação: Edson Norberto Cáceres (UFMS)
Publicações: Karin Breitmann (PUC-Rio)
Planejamento e Programas Especiais: Augusto Sampaio (UFPE)
Secretarias Regionais: Aline dos Santos Andrade (UFBA)
Divulgação e Marketing: Altigran Soares da Silva (UFAM)

Diretorias Extraordinárias

Regulamentação da Profissão: Ricardo de Oliveira Anido (UNICAMP)
Eventos Especiais: Carlos Eduardo Ferreira (USP)
Cooperação com Sociedades Científicas: Taisy Silva Weber (UFRGS)

Conselho

Mandato 2007-2011:

Cláudia Maria Bauzer Medeiros (UNICAMP)
Roberto da Silva Bigonha (UFMG)
Cláudio Leonardo Lucchesi (UNICAMP)
Daltro José Nunes (UFRGS)
André Ponce de Leon F. de Carvalho (ICMC - USP)

Mandato 2005 – 2009:

Ana Carolina Salgado (UFPE)
Jaime Simão Sichman (USP)
Daniel Schwabe (PUC-Rio)
Vera Lúcia Strube de Lima (PUCRS)
Raul Sidnei Wazlawick (UFSC)

Suplentes - Mandato 2007-2009:

Ricardo Augusto da Luz Reis (UFRGS)
Jacques Wainer (UNICAMP)
Marta Lima de Queiroz Mattoso (UFRJ)

CERV – Comissão Especial de Realidade Virtual

Comitê Gestor

Coordenador: Romero Tori (USP e SENAC/SP)
Vice-Coordenador: Edgard Lamounier (UFU)

Membros do Conselho

Representantes dos Professores/Pesquisadores:
Alberto Raposo (PUC-RJ)
Claudio Kirner (UNIMEP e UNASP)
Edgard Lamounier (UFU)
Eduardo Albuquerque (UFG)
Fatima Nunes (Univem)
Jauvane C. de Oliveira (LNCC)
Judith Kelner (UFPE)
Liliane dos Santos Machado (UFPB)
Luciana Nedel (UFRGS)
Marcelo de Paiva (Unasp/Faccamp)
Mario Kubo (INTERLAB/USP)
Robson Augusto Siscouto (UNIC)
Rosa Costa (UERJ)
Veronica Teichrieb (UFPE)

Representantes dos ex-coordenadores:
Alexandre Cardoso (UFU)
Márcio Serolli Pinho (PUC-RS)

Representantes dos Alunos de Pós-Graduação:
Thiago Souto Maior Cordeiro de Farias (UFPE)
João Bernardes (INTERLAB/USP)

Representantes dos Alunos de Graduação:
João Marcelo Xavier Natário Teixeira (UFPE)
João Paulo Lima (UFPE)

Conference Program

	Tuesday May 13th			Wednesday May 14th			Thursday May 15th			Friday May 16th		
8:20-10:00	Pre-Symposium	Minicourse 1	Minicourse 2	Registration			**International Talk** Prof. Hiroaki Yano			**International Talk** Prof. Doug A. Bowman		
10:20-12:00				**International Talk** Prof. Don Brutzman			Technical Session 3		WIC/WTD	Technical Session 5		WIC/WTD
14:00-15:40	Pre-Symposium	Minicourse 3	Minicourse 4	**Invited Minicourse**	Minicourse 5	Technical Session 1	Technical Session 4	Minicourse 5	Minicourse 6	Technical Session 6	Technical Session 7	Minicourse 7
16:00-17:40					Minicourse 6	Technical Session 2	**National Talk** Prof. Claudio Kirner			Technical Session 8	Technical Session 9	
18:00				Opening Session			CERV Meeting					

keynotes

Don Brutzman

X3D Worlds

Abstract

Extensible 3D (X3D) Graphics has steadily progressed since work first began a decade ago. The Web3D Consortium is the public-private partnership of industry, agencies, universities and individuals that has "kept the flame alive" and made X3D what is today. Thanks to steady innovation by Web3D members, new X3D features continue to evolve and grow into great capabilities. Implementation, evaluation and then formal review by the International Standards Organization (ISO) have made X3D an approved standard for real-world use, both on and off the Web. Further collaboration with the World Wide Web Consortium (W3C) has made X3D a "first-class citizen" on the Web, providing excellent (and growing) interoperability with other XML standards.

Bio

Don Brutzman is a computer scientist and Associate Professor working in the Modeling Virtual Environments & Simulation (MOVES) Institute at the Naval Postgraduate School in Monterey, California, USA. He is one of the original participants involved with VRML since this community effort began in 1994. He is a founding member of the non-profit Web3D Consortium, serving on the Board of Directors. He co-chairs the X3D Working Group and leads X3D technical development efforts. He also serves as the Web3D liaison to theWorldWideWeb Consortium (W3C) Advisory Committee. He has been teaching VRML and X3D since 1996.

Hiroaki Yano

Virtual Reality System for Whole Body Interaction

Abstract

In this talk, a virtual reality (VR) system for whole body interaction will be addressed. VR system can realize intuitive interaction environment with virtual objects. Especially, compared with the conventional computer interface techniques, VR systems can provide multi-modal interaction environment using user's whole body. It requires some elemental technologies, a visual display, haptic interface for hands, mouth and feet. Our team has developed many VR devices such as immersive projection displays, haptic interfaces, wearable vibrotactile suit and locomotion interfaces and so on. We will show you brief introduction of such devices. In addition, as an application of this system, a gait rehabilitation system using an immersive projection display and a locomotion interface will be presented.

Bio

Hiroaki Yano is an Associate Professor in the Departament of Intelligent Interaction Technologies at University of Tsukuba in Japan. His research interests include haptics interfaces, locomotion interfaces and cooperative work in VR environments. He received a BE and PhD in Mechanical Engineering from the University of Tsukuba, both with awarded works. He used to be present at the Emerging Technologies track at ACM-SIGGRAPH to demonstrate the innovative devices and works, developed by his research group.

Doug Bowman

VR and beyond - Increasing the impact of VR and 3D interaction research in the real world

Abstract

In the 1990s, virtual reality promised to be the next big thing, a technology that would transform education, design, entertainment, and medicine. Although there have been some notable successes, for the most part immersive VR has not made the impact that many envisioned in those early days. But what does this mean for VR researchers in 2008? In this talk, I will discuss two strategies the VR and 3D user interface (3D UI) communities can use to increase the real-world impact of their work. The first strategy is to use VR techniques and knowledge in non-VR contexts. For example, 3D UIs can be used for interaction with large remote displays, even when they are displaying 2D data. The second strategy is to demonstrate the benefits of immersion - to provide empirical results that quantify gains in performance, satisfaction, or experience due to immersive technology.

Bio

Doug A. Bowman is an Associate Professor of Computer Science at Virginia Tech, where he directs the 3D Interaction Research Group and is a member of the Center for Human-Computer Interaction. His research interests include 3D user interfaces, interaction techniques for virtual environments, the benefits of immersion in VR, and large high-resolution displays. He is a co-author of the book 3D User Interfaces: Theory and Practice, and was awarded a National Science Foundation CAREER grant for his work on domain-specific 3D user interfaces. He and his students have designed, run, and analyzed scores of VR user studies over the past decade. Bowman received his MS and PhD in Computer Science from the Georgia Institute of Technology.

Claudio Kirner

Evolução da Realidade Virtual no Brasil

Abstract

This talk will address the Virtual Reality history in Brazil, pointing out the main factors that contributed to its beginning and consolidation, as well as the steps of its evolving until now. A historical approach of that evolving will be analyzed in the international context, exploring the chronological and technological point of view. We will present details about actions, events, researches and other elements related to the Virtual Reality history in Brazil.

Bio

Claudio Kirner is graduated in Electric Engineering at EESC-USP, M.Sc. in Electronic Engineering at ITA, and D.Sc. in Systems Engineering and Computing at COPPE/UFRJ. He has a post-doc in VR at the University of Colorado at Colorado Springs. Currently he is a Professor at the Universidade Metodista de Piracicaba (UNIMEP) and at the Centro Universitário Adventista de São Paulo (UNASP). Professor Kirner has advised 22 M.Sc. and 7 D.Sc. students at COPPE/UFRJ, USP-SP, UNICAMP, USP-S.Carlos, UFSCar, and UNIMEP. He is author of about 140 scientific papers and author or organizer of 14 books. He coordinated the I Workshop on VR (1997) and the I Workshop on AR (2004) in Brazil.

papers

Invited Paper

Evolução da Realidade Virtual no Brasil

Claudio Kirner
Universidade Metodista de Piracicaba – UNIMEP
Centro Universitário Adventista de São Paulo - UNASP
ckirner@gmail.com

Abstract

This paper shows the Virtual Reality history in Brazil, pointing out the main factors that contributed to its beginning and consolidation, as well as the steps of its evolving until now. A historical approach of that evolving is analyzed in the international context, exploring the chronological and technological point of view. This paper presents details about actions, events, researches and other elements related to the Virtual Reality history in Brazil.

Resumo

Este artigo aborda a história da Realidade Virtual no Brasil, mostrando os fatores que contribuíram para seu surgimento e sua consolidação, bem como as etapas da evolução da área até os dias atuais. A análise dessa evolução foi inserida dentro do contexto internacional, explorando tanto os aspectos cronológicos, quanto os tecnológicos. O artigo apresenta os detalhes das ações, eventos, pesquisas e outros elementos da história da Realidade Virtual no Brasil.

1. Introdução

A Realidade Virtual surgiu no Brasil, na década de 90, dentro do contexto internacional, impulsionada pelo avanço tecnológico, exposição de pesquisadores a novas tecnologias e iniciativas individuais, integrando áreas multidisciplinares, envolvendo: computação gráfica, sistemas distribuídos, computação de alto desempenho, sistemas de tempo real, interação humano computador, periféricos etc.

Essas iniciativas, muitas delas apoiadas por órgãos financiadores, principalmente pelo CNPq, convergiram para a realização do primeiro evento nacional de Realidade Virtual (I Workshop de Realidade Virtual – WRV'97), em 1997, que congregou alguns pesquisadores ativos na área e interessados, dando origem à série SVR (Symposium on Virtual and Augmented Reality).

O segundo evento (WRV'99) consolidou a comunidade de Realidade Virtual no Brasil, estruturando a Comissão Especial de Realidade Virtual - CERV, associada à Sociedade Brasileira de Computação - SBC.

Este trabalho aborda a visão histórica da evolução da Realidade Virtual no Brasil, dentro do contexto internacional, apresentando suas origens, ações, eventos e principais ocorrências antes do WRV'97 e ao longo da realização dos dez primeiros SVR.

Além da abordagem cronológica (*"timeline"*), este trabalho também analisa a evolução da Realidade Virtual e suas variações do ponto de vista tecnológico, mostrando a convergência de tecnologias que contribuíram para a área, em cada fase de sua evolução.

2. Surgimento e Evolução da Realidade Virtual

Existem muitos levantamentos das ocorrências importantes que definiram a história da Realidade Virtual, enfatizando os pesquisadores pioneiros, projetos, publicações, equipamentos, software, empresas, eventos, paradigmas etc. Sherman e Craig [10], por exemplo, apresentam um bom levantamento cronológico da história da Realidade Virtual.

No entanto, outras abordagens podem ser usadas para estudar a evolução da área, integrando, às ocorrências temporais, as várias fases de desenvolvimento tecnológico.

A seguir, serão apresentadas uma abordagem cronológica do surgimento e evolução da Realidade Virtual e uma análise das ocorrências tecnológicas em cada fase.

2.1. Abordagem cronológica

Embora a Realidade Virtual seja uma tecnologia, que firmou-se na década de 90, ela tem suas origens na década de 50, com experiências multimodais baseadas em técnicas cinematográficas. Nesse sentido, Morton Heilig iniciou, em 1956, a construção do Sensorama,

uma máquina que permitia ao usuário fazer um passeio pré-gravado de motocicleta por Manhattan. O usuário podia ver o trajeto, através da projeção de um filme, e sentir sensações sincronizadas com o passeio como: sons, cheiros, vibrações e vento.

Em 1961, os engenheiros da Philco: Comeau e Bryan criaram um capacete ("Head-Mounted Display" – HMD), cujo movimento controlava uma videocâmera remota, permitindo a implementação de telepresença por vídeo.

Apesar de já existirem gráficos desenhados por computador no monitor ("computer graphics"), Ivan Sutherland, em janeiro de 1963, apresentou sua tese de doutorado no MIT, intitulada "Sketchpad, a Man-Machine Graphical Communication System" [11], usando pela primeira vez a computação gráfica interativa, que constituiu-se no marco da criação da Realidade Virtual. Para isto, foi usada uma caneta óptica para realizar interações de seleção e desenho de figuras no monitor, complementando ações do teclado. Neste trabalho, Sutherland fixou a maior parte das palavras chaves da definição de Realidade Virtual, envolvendo: representações virtuais geradas por computador (gráficos no monitor), interação em tempo real (gráficos interativos) e dispositivos especiais (caneta óptica).

Em 1965, Ivan Sutherland publicou o trabalho "The Ultimate Display" [12], no qual estabeleceu os conceitos de um display que poderia ser usado para o usuário ver e interagir com objetos em um mundo virtual, com aparência de real, envolvendo estímulos visuais, sonoros e táteis, de forma a atuar de maneira intuitiva e realista.

Em 1968, Ivan Sutherland finalmente publicou o artigo "A Head-Mounted Three Dimensional Display" [13], descrevendo o desenvolvimento de um capacete (HMD) estereoscópico e rastreável, na Universidade de Harvard. Este capacete era baseado em dois mini-displays CRT, que serviam para projetar as imagens diretamente nos olhos do usuário, e usava uma interface para rastreadores de cabeça mecânicos e ultrassônicos. Este foi outro marco há história da Realidade Virtual, estabelecendo o conceito de imersão.

Cabe destacar que o capacete interativo por vídeo, criado pelos engenheiros da Philco, juntamente com o capacete interativo por computação gráfica de Sutherland, ambos rastreáveis, estabeleceram as bases da Realidade Aumentada. Depois de décadas, vídeo, rastreamento e computação gráfica integrados, interagindo em tempo real, permitiriam o desenvolvimento de aplicações de Realidade Aumentada.

Depois disso, é interessante mencionar algumas ocorrências na evolução da Realidade Virtual:

- 1977: a luva Dataglove é desenvolvida, contribuindo para os aspectos multisensoriais da Realidade Virtual. A luva transformou-se em um produto comercial somente em 1985, lançada pela empresa VPL Research.

- 1981: O simulador Super Cockpit da Força Aérea Americana passou a operar com um capacete de visão óptica, que possibilitava ao piloto uma visão aumentada com informações do avião, como a indicação visual dos mísseis disponíveis para disparo instalados nas asas. Um visor de acrílico permitia a visão direta da cena misturada com a projeção sobreposta das imagens geradas por um display CRT acoplado ao capacete. O custo desse projeto estava na faixa de milhões de dólares. Este é um dos primeiros registros de projetos de Realidade Aumentada.

- 1989a: A empresa Mattel introduziu a luva Powerglove e um sistema de rastreamento para o videogame Nintendo. Esses produtos não fizeram sucesso no ramo de videogame, mas foram adaptados para os primeiros sistemas populares de Realidade Virtual, baseados em microcomputadores PC.

- 1989b: Jaron Lanier, um artista e cientista da computação, cunhou o termo "Realidade Virtual", como alternativa a termos semelhantes como "mundo virtual" e "realidade artificial".

- 1990: O Prof. Thomas Caudell, da Universidade do Novo México, em uma visita à empresa Boeing, cunhou o termo "Realidade Aumentada", em referência a um dispositivo de Realidade Virtual, que apoiava funcionários na montagem de equipamentos eletrônicos de aeronaves.

- 1991a: O primeiro periódico comercial para a comunidade de Realidade Virtual "CyberEdge Journal" foi publicado.

- **1991b**: Foi criado o Rend386, um software livre (gratuito e de código aberto), voltado para o desenvolvimento de aplicações populares de Realidade Virtual, de autoria de Bernie Roehl e Dave Stampe da Universidade de Waterloo.

- 1992a: Surgiu a Realidade Virtual por projeção, implementada na CAVE, como alternativa para o uso de capacete. O projeto foi desenvolvido por Carolina Cruz-Neira, na Universidade de Illinois, em Chicago, e demonstrado no evento SIGGRAPH'92.

- 1992b: A empresa Sense8 Co. passou a comercializar o software para desenvolvimento de aplicações de

Realidade Virtual "WorldToolKit", constituído por uma biblioteca de funções C próprias para Realidade Virtual, aumentando a produtividade e a qualidade das aplicações.

- 1992c: A empresa Silicon Graphics Inc. lançou o Iris Inventor, uma ferramenta de software em C++ para modelagem e visualização 3D, que, mais tarde, forneceu as bases estruturais da linguagem VRML.

- 1993a: Surgiram duas conferências acadêmicas sobre Realidade Virtual: VRAIS'93, realizada em Seattle, e "Research Frontiers in Virtual Reality IEEE Workshop", realizada em San Jose. Em 1995, as duas conferências se juntaram, dando origem ao evento IEEE VRAIS, que depois foi denominado IEEE VR.

- 1993b: Foi realizado o "Workshop on Augmented Reality and Ubiquitous Computing", no MIT, com a presença de vários pesquisadores, dentre os quais: Ronald Azuma, Steve Feiner, Paul Milgram, Myron Krueger, Pierre Welner, Wendy MacKay e Rich Gold, Logo em seguida, em julho de 1993, é publicada uma edição especial da revista "Communications of the ACM" sobre Realidade Aumentada, com o título: "Computer-Augmented Environments: Back to the Real World" [15], chamando a atenção para a área.

- 1994a: A linguagem VRML, elaborada por especialistas da área acadêmica e de empresas, foi liberada com especificação aberta para uso público. Esta foi uma das principais ferramentas para a disseminação da Realidade Virtual na Web.

- 1994b: Paul Milgram et. al. publicaram o artigo "Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum", discutindo a Realidade Misturada, abrangendo a Realidade Aumentada e a Virtualidade Aumentada [8].

- 1995: Ronald Azuma publicou o artigo "A Survey of Augmented Reality" [2], disseminando os conceitos e aplicações de Realidade Aumentada.

- 1999a: O software livre "ARToolKit", uma biblioteca escrita em C e baseada em rastreamento por vídeo, foi liberada para uso, despertando o interesse pela área de Realidade Aumentada pelo mundo.

- 1999b: Foi iniciada a especificação da linguagem X3D, pelo Consórcio Web3D, tendo como base a estrutura do XML e visando definir a sucessora do VRML.

- 2001: Foi publicado o livro de autoria de John Tiffin e Nobuyioshi Terashima, intitulado "HyperReality: Paradigm for the Third Millenium", fixando as bases da evolução da Realidade Aumentada com a incorporação de recursos de Inteligência Artificial [14].

Depois disso, surgiram diversos sistemas e ferramentas, mas é importante salientar a tendência de facilitar o acesso aos recursos de Realidade Virtual e Realidade Aumentada com maior oferta de software livre (gratuito e de código aberto), como o sistema Flux Studio, voltado para autoria visual de aplicações de Realidade Virtual com VRML e X3D. Nesse sentido, também ocorreu a flexibilização de acesso a recursos mais avançados, como o sistema Vizard, que na versão profissional tem um custo da ordem de vários mil dólares, mas cuja versão "light", com quase todas as funcionalidades da ferramenta, pode ser usada para teste por 90 dias ou adquirida por algumas dezenas de dólares.

2.2. Abordagem tecnológica

Na abordagem tecnológica, a evolução da Realidade Virtual é tratada do ponto de vista de vários parâmetros, envolvendo: sistemas; interfaces; inteligência; tipos de interação e tempo.

O ambiente real (físico) e o ambiente computacional são elementos de referência para essa análise. Até o início da década de 90, a tecnologia integrada ao ambiente real e computacional propiciava interações previsíveis, decorrentes da eletrônica e da computação, mas ainda apresentava pouca inteligência, envolvida na interação com o usuário. Nesse sentido, os sistemas, de maneira geral, eram relativamente simples.

Durante a década de 90, os avanços tecnológico, computacional e de telecomunicações permitiram a convergência de fatores para o aparecimento de sistemas com comportamento não determinístico e interação multimodal. Muitos dispositivos tecnológicos tornaram-se inteligentes, apresentando interfaces mais interativas e amigáveis, e os sistemas computacionais ficaram mais presentes e transparentes aos usuários. Isto resultou do uso mais intensivo de: computação ubíqua, sistemas distribuídos, interações multimodais, processamento massivamente paralelo, inteligência artificial etc.

É nesse contexto que a evolução da Realidade Virtual apresenta suas principais variações, de acordo com a Figura 1, mostrando a evolução dos sistemas reais e virtuais, em função de tecnologias, envolvendo inteligência, interfaces, interações e tipos de sistema.

Até o início da década de 90, o ambiente real usava pouca ou nenhuma tecnologia avançada. Os sistemas de Realidade Virtual eram simples e pouco interativos e a tecnologia de rastreamento do usuário era baseada em elementos mecânicos, magnéticos, e ultrassônicos.

Durante a década de 90, o avanço tecnológico permitiu

o desenvolvimento de dispositivos interativos mais avançados, com comportamento determinístico. O rastreamento óptico passou a ser usado, em função dos avanços na área de visão computacional. A Realidade Misturada, abrangendo a Realidade Aumentada e a Virtualidade Aumentada, tornou-se objeto de estudo e desenvolvimento mais intensivo.

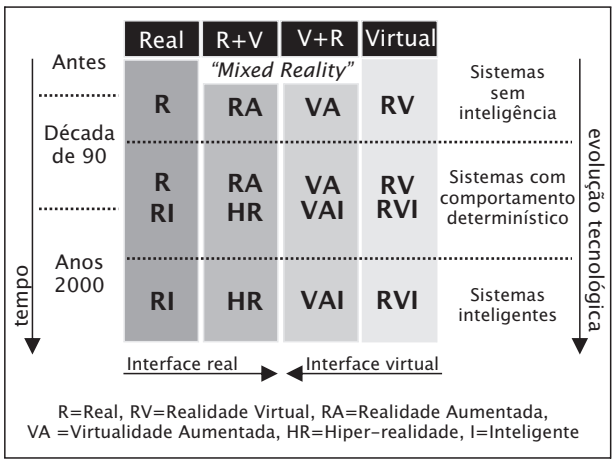


Figura 1. Evolução da Realidade Virtual no contexto tecnológico.

Ao redor dos anos 2000, técnicas de inteligência artificial e interação multimodal, aliadas à computação ubíqua, contribuíram para o desenvolvimento de dispositivos mais avançados e de sistemas de Realidade Virtual, Realidade Aumentada e Virtualidade Aumentada com algum grau de inteligência e comportamentos não determinísticos. Começaram a aparecer os objetos reais inteligentes e os sistemas virtuais inteligentes, que interagem com os usuários de maneira não determinística, reagindo diferentemente, em função de situações variadas.

O diagrama da Figura 1 apresenta também a influência do tipo de interface/interação na classificação dos tipos de sistemas virtuais [3]. Nesses sistemas, o uso de interfaces/interações do ambiente real define os ambientes de Realidade Aumentada e de Hiper-realidade. O uso de interfaces/interações do ambiente virtual define os ambientes de Realidade Virtual e de Virtualidade Aumentada inteligentes e sem inteligência.

3. Início da Realidade Virtual no Brasil

Os primeiros registros de ações na área de Realidade Virtual no Brasil datam do início da década de 90.

Essas ações envolveram: publicações, realização de eventos, defesas de mestrado e doutorado, desenvolvimento de projetos, cooperações internacionais, criação de grupos etc.

O marco escolhido para definir o início efetivo da Realidade Virtual no Brasil foi a realização do primeiro evento da série SVR (Symposium on Virtual and Augmented Reality), realizado na Universidade Federal de São Carlos, em novembro de 1997, sob a denominação de 1º Workshop de Realidade Virtual – WRV'97.

3.1. Primeiras ações na área de Realidade Virtual no Brasil

Antes da realização do WRV'97, algumas ações na área de Realidade Virtual foram realizadas no Brasil. Em seguida, consta um levantamento dessas ações [4], que não pretende ser definitivo, podendo ser complementado com novas informações.

- Outubro de 1992: O Prof. Claudio Kirner (UFSCar) iniciou uma visita de 6 meses na Silicon Graphics – USA, com o apoio do CNPq, onde estudou o software Iris Inventor e fez um contato para seu Pós-Doutorado em Realidade Virtual, na Universidade do Colorado, em Colorado Springs, USA.

- 1993a: O Prof. Claudio Kirner iniciou seus estudos de Pós-Doutorado em Realidade Virtual, com o apoio do CNPq e duração de dois anos, na Universidade do Colorado, em Colorado Springs, USA.

- 1993b: Foi publicado, na UFRGS, o trabalho de pesquisa "Um Estudo sobre Técnicas Aplicadas à Realidade Virtual" de autoria de A. C. Berg, sob a orientação do Prof. A. Laschuck.

- 1994: Foi publicado, no Workshop on Virtual Environments and their Applications and Virtual Prototyping'94, em Portugal, o artigo "Network Requirements for Large Distributed Virtual Environments", de autoria de Regina B. Araújo e Claudio Kirner, da UFSCar.

- 1995a: Foi publicado, na UFRGS, o trabalho de pesquisa "Realidade Virtual: O Estado da Arte", de autoria de G. A. Assis, sob a orientação do Prof. A. Laschuck.

- 1995b: Foi publicado, no INPE, o relatório técnico "Conceitos Básicos de Realidade Virtual", de autoria de Liliane S. Machado.

- 1995c: Foi publicado, no SBRC'95, o artigo "Arquitetura de um Ambiente Virtual Distribuído para aplicações de Larga Escala", de autoria de Regina B. Araújo e Claudio Kirner, da UFSCar.

- 1995d: Foi publicada, no Workshop on Information Technology'95, em Berlim, a Proposta de Convênio Internacional entre Alemanha e Brasil, na Área de Informática, envolvendo o projeto "Virtual

Environment for Shared Interactive Visualization – VESIV", coordenado, no Brasil, pelo Prof. Claudio Kirner (UFSCar).

- 1995e: Foi submetido o projeto PROTEM-CC-FaseIII/CNPq "Ambiente Virtual para Visualização Interativa Compartilhada – AVVIC", coordenado pelo Prof. Claudio Kirner (UFScar).

- 1995f: Foi submetido o projeto RHAE-Informática/CNPq, na área de Realidade Virtual, coordenado pelo Prof. Claudio Kirner (UFSCar).

- 1995g: Foi criado, na UFSCar, o Grupo de Realidade Virtual, em outubro de 1995, sob a coordenação do Prof. Claudio Kirner.

- 1996a: Foi publicada, na JAI/SBC´96, a apostila do mini-curso "Introdução à Realidade Virtual", de autoria de Claudio Kirner (UFSCar) e Márcio. S. Pinho (PUCRS).

- **1996b**: Foi publicado, no SBIE'96, o artigo "Uma Experiência do Uso de VRML no Ensino de Computação Gráfica em Curso de Graduação em Informática", de autoria de Márcio S. Pinho, da PUCRS.

- 1996c: Foi publicado, no SBIE'96, o tutorial "Realidade Virtual como Ferramenta de Informática na Educação", de autoria de Márcio S. Pinho (PUCRS).

- 1996d: Foi publicado, no SIBGRAPI'96, o tutorial "Uma Introdução à Realidade Virtual", de autoria de Márcio S. Pinho (PUCRS) e Claudio Kirner (UFSCar).

- 1996e: Foi publicado, no SBRC'96, o artigo "Especificação e Análise de um Sistema Distribuído de Realidade Virtual", de autoria de Regina B. Araújo e Claudio Kirner, da UFSCar.

- 1996f: Foi publicado, no SEMISH'96, o artigo "Análise dos Requisitos de Rede de um Sistema Distribuído de Realidade Virtual", de autoria de Regina B. Araújo e Claudio Kirner, da UFSCar.

- 1996g: Foi publicada, na VII SEMAC-S.J. Rio Preto, a apostila do mini-curso "Sistemas de Realidade Virtual: Aspectos, Distribuição e Programação na Internet", de autoria de Claudio Kirner, Regina B. Araújo e Juliano Ipólito, da UFSCar.

- 1996h: Foi publicado, no Congresso Petrobras de Informática e Telecomunicações'96, o artigo "Mundos Virtuais", de autoria de Gerson G. Cunha e L. R. F. Moreira, da UFRJ.

- 1996i: Foram publicados, na UFSCar, os Anais do Ciclo de Palestras de Realidade Virtual, editados por Claudio Kirner.

- 1996j: Foi criado, na PUCRS, o Grupo de Realidade Virtual, sob a coordenação do Prof. Márcio S. Pinho.

- 1996k: Foi defendida, na EPUSP, a Tese de Doutorado "Especificação e Avaliação de um Sistema Distribuído de Realidade Virtual", por Regina B. Araújo, sob a orientação do Prof. Claudio Kirner.

- 1997a: Foi publicado, no Workshop on Information Technology'97, o artigo "Virtual Environment for Shared Interactive Visualization", de autoria de Claudio Kirner et.al., da UFSCar.

- 1997b: Foi publicado, no SMC´97, em Orlando-USA, o artigo "Validation of Real-Time Systems using a Virtual Reality Simulation Tool", de autoria de Alexandre M. Gimenez e Tereza G. Kirner, da UFSCar.

- 1997c: Foi publicado, no SMC´97, em Orlando-USA, o artigo "Animation of Deformable 3D Objects in Virtual Reality Systems", de autoria de Bruno O. Schneider e Claudio Kirner, da UFSCar.

- 1997d: Foi publicado, no HICSS'97, no Hawaii-USA, o artigo "The Dimensioning of a Distributed Virtual Reality System", de autoria de Regina B. Araújo e Claudio Kirner, da UFSCar.

- 1997e: Foi publicado, no Workshop de Robótica Inteligente'97, o artigo "Robot Control using Virtual Reality Techniques", de autoria de Márcio S. Pinho (PUCRS).

- 1997f: Foi publicado, no evento Information Technology for Competitiveness'97, o artigo "Experiences of Virtual Reality Lab at PUCRS", de autoria de Márcio S. Pinho (PUCRS).

- 1997g: Foi criado, na COPPE-UFRJ, o Grupo de Realidade Virtual Aplicada, sob a coordenação do Prof. Gerson Cunha.

- 1997h: Foi defendida, no INPE, a Dissertação de Mestrado "A Realidade Virtual em Aplicações Científicas", por Liliane S. Machado, sob orientação do Prof. Luiz A. V. Dias.

O marco escolhido para definir o início efetivo da Realidade Virtual no Brasil foi a realização do primeiro evento da série SVR (Symposium on Virtual and Augmented Reality), realizado na Universidade Federal de São Carlos, em novembro de 1997, sob a denominação de 1º Workshop de Realidade Virtual – WRV'97.

3.2. Primeiro SVR

O primeiro evento da série SVR ocorreu na Universidade Federal de São Carlos - UFSCar, no período de 09 a 12 e novembro de 1997. Ele foi

denominado 1º Workshop de Realidade Virtual – WRV'97 [6], tendo sido coordenado pelo Prof. Claudio Kirner.

O objetivo primordial do WRV'97 foi promover a integração de pesquisadores, profissionais e estudantes, interessados na área de Realidade Virtual. Em termos específicos, buscou-se atingir os seguintes objetivos:

•Difundir a área de Realidade Virtual, através de mini-cursos, palestras, e painéis de discussão;

•Divulgar os trabalhos que estavam sendo realizados no Brasil;

•Fornecer, aos participantes, um panorama de projetos que eram conduzidos em outros países.

As atividades do Workshop incluíram:

•5 mini-cursos nacionais;

•3 mini-cursos internacionais convidados;

•4 palestras convidadas, sendo 3 internacionais;

•5 sessões técnicas para apresentação de 20 trabalhos selecionados;

•1 painel de discussão;

•demonstrações de projetos e aplicações de Realidade Virtual.

Os trabalhos técnicos, mini-cursos, painéis e palestras enfocaram temas relacionados aos diferentes aspectos do desenvolvimento e utilização de sistemas, ambientes e aplicações de Realidade Virtual, incluindo:

• Desenvolvimento de hardware de Realidade Virtual;

• Desenvolvimento de software de Realidade Virtual;

• Interação tridimensional homem-máquina;

• Desenvolvimento de sistemas e de aplicações de Realidade Virtual;

• Aspectos de avaliação em ambientes e sistemas de Realidade Virtual;

• Impactos sociais e econômicos da Realidade Virtual;

• Tendências, a nível nacional e internacional;

• Outros temas relevantes.

O Workshop teve cerca de 250 participantes, incluindo docentes, pesquisadores, alunos de graduação e pós-graduação e alguns profissionais. Esses participantes vieram de diferentes estados, como São Paulo (capital e interior), Rio de Janeiro, Paraná, Minas Gerais, Rio Grande do Norte, Rio Grande do Sul e Santa Catarina.

O evento foi apoiado pela Sociedade Brasileira de Computação – SBC e recebeu patrocínio do CNPq, EMBRAPA, SEBRAE, UFSCar, Papelaria TendLer, Compaq, Silicon Graphics-Brasil e Sense8.

3.3. Criação da Comissão Especial de Realidade Virtual – CERV

Após o primeiro evento nacional de Realidade Virtual (WRV'97), esperou-se dois anos para a realização do próximo evento de Realidade Virtual, visando dar tempo para o desenvolvimento de novos trabalhos na área e o surgimento de novos membros na comunidade, que estava se organizando.

Durante a realização do 2º Workshop Brasileiro de Realidade Virtual (WRV'99), em Marília-SP, no período de 18 a 20 de novembro de 1999, os participantes fizeram uma reunião, no dia 15/11/99, e decidiram criar a Comissão Especial de Realidade Virtual (CERV), vinculada à SBC, formalizando assim a existência de uma comunidade organizada e ativa na área de Realidade Virtual no Brasil. Nesse sentido, os participantes da reunião elaboraram um abaixo-assinado para ser enviado à SBC, com 53 assinaturas, sendo 44 sócios da SBC. O Prof. Claudio Kirner, Coordenador Geral do WRV'99, foi escolhido para coordenar a CERV por dois anos.

No dia 10 de janeiro de 2000, o Prof. Claudio Kirner encaminhou um ofício, com o abaixo-assinado, à Diretoria de Eventos e Comissões Especiais da SBC, solicitando a criação da CERV, a qual foi aprovada em março de 2000 [5].

A partir do WRV'99, os eventos passaram a ocorrer anualmente. Cada Comitê de Programa do evento de um ano atuava como Conselho da CERV até o evento do ano seguinte.

Durante a realização do VI SVR, em Ribeirão Preto-SP, em outubro de 2003, o Regimento da CERV foi apresentado e aprovado na Reunião Plenária, ocorrida no dia 16/10/2003, fazendo com que a Comunidade de Realidade Virtual tivesse regras claras para regular sua organização, ações e eventos. Nesta mesma reunião, o Prof. Claudio Kirner foi reconduzido para mais um ano de mandato, como Coordenador da CERV, e o Prof. Márcio S. Pinho foi eleito para atuar como Vice-Coordenador, com mandato de um ano, e Coordenador, com mandato para o ano subsequente. Foi eleito também o Conselho da CERV, com representantes de professores/pesquisadores, alunos de pós-graduação e alunos de graduação. Metade do conselho foi eleito com mandato defasado. Assim, em cada SVR, passaria a ser eleito o Vice-Coordenador, além de ser renovada metade do conselho.

Em 2007, durante a realização do IX SVR, a Reunião Plenária alterou este dispositivo do Regimento da CERV, estabelecendo o mandato conjunto do

Coordenador e Vice-Coordenador por dois anos, sem possibilidade de reeleição.

Desde sua criação, em 1999, a CERV teve os seguintes Coordenadores: Claudio Kirner (2000-2004); Márcio S. Pinho (2004-2005); Alexandre Cardoso (2005-2006); Romero Tori (2006-2008). Além disso, o Prof. Edgard Lamounier Jr. atuou como Vice-Coordenador, no período de 2007 a 2008, para o ajustamento da nova forma de atuação da Coordenação da CERV.

4. Evolução dos SVR

A série SVR, desde a sua criação, em 1997, evoluiu em nome e conteúdo [5].

O primeiro evento foi realizado em São Carlos – SP, no período de 09 a 12 de novembro de 1997, com a denominação "1º Workshop de Realidade Virtual – WRV'97". O evento foi coordenado pelo Prof. Claudio Kirner, contou com o apoio da SBC e recebeu financiamento do CNPq e de organizações e empresas.

O segundo evento da série foi realizado em Marília – SP, no período de 18 a 20 de novembro de 1999, com a denominação "2º Workshop Brasileiro de Realidade Virtual – WRV'99". O evento foi coordenado pelo Prof. Claudio Kirner, contou com o apoio da SBC e recebeu financiamento do CNPq, da FAPESP e de organizações e empresas.

O terceiro evento foi realizado em Gramado – RS, no período de 16 a 18 de outubro de 2000, com a denominação "3rd Workshop on Virtual Reality – WVR'2000", tendo sido coordenado pelos Profs. Márcio S. Pinho e Ricardo Reis. Este evento e os subsequentes foram promovidos pela SBC. O evento foi realizado concomitantemente com o IHC, tendo sido financiado pelo CNPq, pela FAPERGS e por organizações e empresas.

O quarto evento foi realizado em Florianópolis – SC, no período de 16 a 19 de outubro de 2001, com a denominação "4th SBC Symposium on Virtual Reality – SVR'2001". O evento foi coordenado pelos Profs. Raul S. Wazlawick e Marta C. Rosatelli e recebeu financiamento do CNPq, da CAPES, da FAPESC (antiga FUNCITEC) e de organizações e empresas.

O quinto evento foi realizado em Fortaleza – CE, no período de 07 a 10 de outubro de 2002, com a denominação "V Symposium on Virtual Reality – SVR'2002", estabilizando o nome até 2006. O evento foi coordenado pelo Prof. Creto A. Vidal e fez parte do IMIGRA'2002, um consórcio de eventos realizados conjuntamente, envolvendo IHC, SIBGRAPI, SBMIDIA e SVR. O SVR'2002 recebeu financiamento do CNPq, da CAPES e de organizações e empresas.

O "VI Symposium on Virtual Reality – SVR'2003" foi realizado em Ribeirão Preto – SP, no período de 15 a 18 de outubro de 2003, sob a coordenação dos Profs. César A. C. Teixeira e Claudio Kirner. O evento recebeu financiamento do CNPq, da FAPESP, da CAPES, da FINEP e de organizações e empresas.

O "VII Symposium on Virtual Reality – SVR'2004" foi realizado em São Paulo – SP, no período de 19 a 22 de outubro de 2004, sob a coordenação dos Profs. Romero Tori e Claudio Kirner. O evento recebeu financiamento do CNPq, da FAPESP, da CAPES, da FINEP e de organizações e empresas. Neste SVR, deu-se início ao Pré-Simpósio, um conjunto de palestras de um dia, proferidas por autores de capítulos de um livro preparado especialmente para o evento. O livro deste Pré-Simpósio, intitulado "Realidade Virtual: Conceitos e Tendências", foi editado pelos Profs. Claudio Kirner e Romero Tori. Anteriormente, foi lançado por iniciativa pessoal, em 2002, o livro "Realidade Virtual: Fundamentos e Aplicações", de autoria de Antonio Valério Neto, Liliane S. Machado e Maria C. Oliveira.

O "VIII Symposium on Virtual Reality – SVR'2006" foi realizado em Belém – PA, no período de 02 a 05 de maio de 2006, sob a coordenação dos Profs. Bianchi S. Meiguins e Márcio S. Pinho. Devido à transferência do período do SVR de outubro para maio, não houve SVR no ano de 2005. O evento recebeu financiamento do CNPq e de organizações e empresas. Durante o Pré-Simpósio, foi lançado o livro "Fundamentos e Tecnologia de Realidade Virtual e Aumentada", editado pelos Profs. Romero Tori, Claudio Kirner e Robson Siscoutto, em CD e em versão digital livre para "download" na Internet [7][9].

O nono evento da série teve seu nome alterado para "IX Symposium on Virtual and Augmented Reality – SVR'2007". O evento foi realizado em Petrópolis – RJ, no período de 28 a 31 de maio de 2007, sob a coordenação dos Profs. Jauvane C. de Oliveira e Alexandre Cardoso, tendo recebido financiamento do CNPq, da FAPERJ, da CAPES e de organizações e empresas. Durante o Pré-simpósio, foi lançado o livro "Realidade Virtual e Aumentada: Conceitos, Projeto e Aplicações", editado pelos Profs. Claudio Kirner e Robson Siscoutto. Também foi lançado, durante o SVR'2007, o livro "Tecnologias para o Desenvolvimento de Sistemas de Realidade Virtual e Aumentada", editado pelos Profs. Alexandre Cardoso, Claudio Kirner, Edgard Lamounier Jr. e Judith Kelner.

O "X Symposium on Virtual and Augmented Reality – SVR'2008" estará sendo realizado, por ocasião da

publicação deste trabalho, em João Pessoa – PB, no período de 13 a 16 de maio de 2008, sob a coordenação dos Profs. Liliane S. Machado e Romero Tori.

Além das apresentações de trabalhos em sessões técnicas, as edições do SVR apresentaram outras atividades como: palestras nacionais e internacionais; mini-cursos nacionais e internacionais, produzindo apostilas e livros; painéis e mesas-redondas; exposições de produtos e projetos; reuniões plenárias; workshops de trabalhos de iniciação científica e de dissertações e teses; festivais de mundos virtuais; etc.

5. Workshops de Realidade Virtual e realidade Aumentada

Em decorrência do avanço da tecnologia de Realidade Virtual e Aumentada e da polarização de alguns grupos de pesquisa em temas específicos, surgiram alguns workshops locais [6], realizados no intervalo entre os SVR, com o objetivo de congregar os interessados nos assuntos específicos, disseminar o tema de pesquisa e alavancar grupos de pesquisa mais novos.

Foi o caso do I Workshop de Realidade Aumentada – WRA'2004, realizado em Piracicaba – SP, no período de 12 a 14 de maio de 2004, sob a coordenação do Prof. Claudio Kirner.

Em 2005, ocorreram dois workshops: o WRA'2005, realizado em Piracicaba – SP, no período de 21 a 23 de setembro, sob a coordenação do Prof. Claudio Kirner, e o I Workshop de Aplicações de Realidade Virtual – WARV'2005, realizado em Uberlândia – MG, no período de 22 a 24 de novembro, sob a coordenação do Prof. Alexandre Cardoso.

Em 2006, o III Workshop de Realidade Aumentada – WRA'2006 foi realizado no Rio de Janeiro, no período de 27 a 29 de setembro, sob a coordenação da Profa. Rosa Costa, e o II Workshop de Aplicações de Realidade Virtual – WARV'2006 foi realizado em Recife – PE, no período de 21 a 24 de novembro, sob a coordenação da Profa. Judith Kelner.

Em 2007, os dois workshops foram unidos em um único evento, resultando no IV Workshop de aplicações de Realidade Virtual e Aumentada – WRVA'2007, que foi realizado em Itumbiara – GO, no período de 21 a 24 de novembro, sob a coordenação do Prof. Marcos Wagner.

Em 2008, o WRVA'2008 será realizado, em Bauru, sob a coordenação do Prof. José Remo F. Brega.

6. Evolução da Realidade Virtual no Brasil

Além das ações que precederam o WRV'97, e a realização dos eventos SVR e Workshops relacionados, outras atividades ocorreram, nos últimos anos, envolvendo a área de Realidade Virtual.

Novos grupos de pesquisa se formaram, surgiram empresas distribuidoras de software e hardware de Realidade Virtual, foram instalados Centros de Realidade Virtual em grandes empresas no Brasil, foram instalados equipamentos de grande porte como a Caverna e o "Powerwall" em universidades e empresas, cursos de pós-graduação passaram a formar mestres e doutores na área etc.

6.1. Grupos de pesquisa na área de Realidade Virtual e Aumentada

Apesar de existirem poucos grupos de pesquisa em Realidade Virtual, no Brasil, por ocasião da realização do WRV'97, outros grupos se formaram, ao longo desses anos, constituindo 29 grupos, enumerados em seguida pelo critério geográfico [9]:

1 - Grupo de RV – PUCRS; 2 - Grupo de Computação Gráfica – UFRGS; 3 - Computer Graphics Laboratory – UNISINOS; 4 - Laboratório de RV – UFSC; 5 - Laboratório de RV Aplicada – UDESC; 6 - Núcleo de RV – USP; 7 - Laboratório de Tecnologias Interativas – USP; 8 - Grupo de RV – SENAC-SP; 9 - Pós-Graduação em Eng. Eletrônica e Computação – ITA; 10 - Linha de Pesquisa em Sistemas Inteligentes – UNISANTOS; 11 - Grupo de Multimídia e RV – UNIMEP; 12 - Grupo de RV – UFSCar; 13 - Grupo de RV – UNESP-Bauru; 14 - Grupo de Pesquisa em RV – UNIVEM; 15 – Grupo de RV – UNASP; 16 - Grupo de RV- UFU; 17 - Grupo de Pesquisa em RV nas Neurociências – UERJ; 18 - Scientific Visualization and Virtual Reality – LNCC; 19 - Tecnologia em Computação Gráfica – PUC-Rio; 20 – Grupo de RV Aplicada – UFRJ; 21 - Laboratório de Ambientes Virtuais Interativos – UNIC; 22 - Novas Tecnologias em RV – ULBRA/Itumbiara e CEFET/Goiânia; 23 - Laboratório de Tecnologias para Ensino Virtual – UFPB; 24 - Grupo de Pesquisa em RV e Multimídia – UFPE; 25 - Laboratório de RV – UFRGN; 26 - Grupo de Computação Aplicada – UNIFOR; 27 - Mestrado em Modelagem Computacional de Conhecimento – UFAL; 28 - Laboratório de Computação Gráfica – UFC, 29 - Grupo de Pesquisa em Tecnologias Interativas – UFPA.

Além desses grupos específicos, há grupos em outras áreas de atuação como Robótica, Jogos, Educação, Artes. etc, trabalhando com Realidade Virtual.

6.2. Empresas Distribuidoras e Centros de Realidade Virtual

No final da década de 90 e início dos anos 2000, a Realidade Virtual passou a ser usada em empresas no Brasil, apoiando o desenvolvimento de projetos e produtos.

A empresa Absolut Technologies [1] surgiu como principal distribuidor de equipamentos, software e soluções profissionais de Realidade Virtual para empresas e universidades.

A Petrobras, já em 1998, realizou a Primeira Mostra Petrobras de Realidade Virtual. De acordo com o cronograma de implantação de instalações de Realidade Virtual da Absolut Technologies [1], a Petrobras acelerou a montagem de Centros de Realidade Virtual, a partir de 2001, possuindo atualmente quase duas dezenas de instalações desse tipo distribuídas em vários estados.

A Embraer, por sua vez, investiu em Realidade Virtual, no final da década de 90, tendo inaugurado seu Centro de Realidade Virtual no ano 2000.

A General Motors do Brasil instalou uma Powerwall em 2004.

Entre as universidades, a USP adquiriu uma Caverna ("CAVE") em 1998 [1] e, em 2001, o LSI-USP construiu sua própria Caverna Digital. A universidade de Caxias do Sul instalou uma Caverna para uso na área de Artes, em 2005. A USP-São Carlos instalou uma Caverna em 2006 e o LNCC instalou sua Caverna em 2007.

Eventualmente, alguns equipamentos ou Centros desses tipos podem ter sido instalado no Brasil, sem que tenham sido mencionados aqui.

6.3. Formação de pessoal na área de Realidade Virtual

Com o crescimento da Comunidade Acadêmica e Científica de Realidade Virtual, vários cursos de pós-graduação no Brasil passaram a implantar linhas de pesquisa em Realidade Virtual ou desenvolver projetos relacionados com a área.

Várias universidades estão tendo ou já tiveram atuação significativa em Realidade Virtual nos seus mestrados e/ou doutorados, incluindo: UFRGS, PUCRS, UFSC, USP, UNIMEP, UFSCar, UNIVEM, ITA, UFU, UFRJ, PUC-Rio, UFPB, UFPE e UFC.

7. Tendências da Realidade Virtual no Brasil

Em função da evolução da Realidade Virtual no Brasil, nas universidades e em empresas, e o estado da arte a nível internacional, pode-se identificar algumas tendências.

Assim, como no passado, a Realidade Virtual vem sendo desenvolvida sob duas vertentes: sistemas profissionais de alto custo, em uso nas empresas; e sistemas populares e de médio custo, em universidades. Enquanto as empresas estão focadas em aplicações específicas de Realidade Virtual, as universidades vêm pesquisando seu desenvolvimento e aplicações de maneira mais ampla.

Do ponto de vista do hardware, continua a tendência de uso de "clusters", para aplicações de alto desempenho, e o uso de redes e da Internet para aplicações multiusuários. Os microcomputadores de alto desempenho continuam sendo demandados, devido ao baixo custo, assim como dispositivos especiais mais baratos.

O uso de câmeras de vídeo ("webcam") vem se intensificando, com as aplicações baseadas em visão computacional para a obtenção de rastreamento de baixo custo. O GPS também vem ganhando importância em aplicações voltadas para ambientes abertos ("outdoor").

As pesquisas de Realidade Aumentada estão se ampliando, em decorrência da evolução natural da área de Realidade Virtual, do baixo custo e das facilidades de acesso aos recursos. Nesse sentido, a disponibilidade de "toolkits" e sistemas de autoria, voltados para os desenvolvedores de aplicações ou para os usuários finais, é de fundamental importância para acelerar o avanço dessa área.

Da mesma maneira, os sistemas hápticos estão em crescimento, uma vez que os custos estão baixando e novas equipes estão trabalhando em pesquisas na área, voltadas principalmente para uso em Medicina.

As interfaces, que até agora não acompanharam os avanços de hardware, estão sendo revistas, apontando para uma tendência forte ao uso de telas de toque ("touch screen"), explorando o uso de interfaces tangíveis. Esta tendência está dentro do espaço de atuação de interfaces com Realidade Aumentada, cujo potencial é maior por atuar no espaço tridimensional, em vez do plano da tela.

Por sua vez, as aplicações colaborativas com Realidade Virtual e Realidade Aumentada tendem a crescer, apoiando principalmente áreas como educação e treinamento, com ênfase em educação à distância, e laboratórios virtuais.

Visualização científica e visualização de dados também estão evoluindo com o uso de Realidade Virtual e Aumentada, através de aplicações mais potentes e amigáveis, melhorando a interação e o entendimento do usuário.

Os jogos de entretenimento e de uso educacional estão ganhando mais impulso com a adoção de técnicas e recursos de Realidade Virtual e Aumentada, explorando o realismo e novas formas de interação em tempo real.

8. Conclusões

Apesar da Realidade Virtual e da Realidade Aumentada originarem-se há algumas décadas, suas evoluções foram muito dependentes do avanço tecnológico de hardware de alto desempenho, de software e de dispositivos. Assim, a Realidade Virtual consolidou-se na década de 90, enquanto a Realidade Aumentada pode ser considerada uma tecnologia dos anos 2000, quando técnicas de visão computacional e rastreamento óptico tornaram-se realidade. A disponibilização gratuita e aberta de software, como VRML e ARToolKit, teve um papel importante na popularização da Realidade Virtual e Aumentada.

O primeiro evento de Realidade Virtual no Brasil (WRV'97) surgiu depois de quatro anos da realização dos primeiros eventos dessa área nos Estados Unidos. A série SVR encontra-se entre os eventos mais antigos realizados fora dos Estados Unidos.

A área acadêmica reagiu bem a essa nova tecnologia, constituindo, no período de 10 anos, cerca de 29 grupos de pesquisa, vários deles formando pessoal em nível de mestrado e doutorado em Realidade Virtual e Aumentada.

Por outro lado, a Realidade Virtual foi incorporada rapidamente a empresas de grande porte do Brasil, por volta do ano 2000, para o desenvolvimento de aplicações em: exploração de petróleo; indústria aeronáutica; e indústria automobilística. Alguns tipos de recursos avançados de Realidade Virtual também foram adquiridos por universidades e centros de pesquisa, como é o caso da Caverna ("CAVE").

A área acadêmica também se organizou rapidamente nessa época, constituindo a Comissão Especial de Realidade Virtual – CERV, associada à SBC, ficando responsável pelo evento SVR (Symposium on Virtual and Augmented Reality) e por ações direcionadas para alavancar o desenvolvimento da área no Brasil.

A área de Realidade Virtual no Brasil, ao longo de sua existência, tem recebido apoio de instituições, empresas e órgãos financiadores, mas vale salientar a presença do CNPq, apoiando todos os eventos da série SVR e outras atividades da área, principalmente na fase inicial, quando demonstrou sensibilidade, para estimular novas áreas do conhecimento no Brasil, e percepção do potencial de inovação da Realidade Virtual.

Enfim, por ocasião da realização do X Symposium on Virtual and Augmented Reality – SVR'2008, este registro, dos principais acontecimentos relacionados com a área de Realidade Virtual no Brasil, está sendo feito para homenagear todas as pessoas e instituições que atuaram direta ou indiretamente nessa empreitada nacional.

9. Referências

[1] Absolut Technologies, *Instalações de RV*, Retrieved March 14, 2008, from http://www.abstech.com/layout_todas_instalacoes.php

[2] R. Azuma, "A Survey of Augmented Reality", *Presence: Teleoperators and Virtual Environments*, v .6, n.4, August 1997, pp. 355-385.

[3] C. Kirner and T.G. Kirner, "*Virtual Reality and Augmented Reality Applied to Simulation Visualization*", in A.A.R El Sheikh, A. Al Ajeeli and E.M.O. Abu-Taieh, (Editors), *Simulation and Modeling: Current Technologies and Applications*. 1 ed. Hershey-NY: IGI Publishing, Hershey, PA, v. 1, 2007, p. 391-419.

[4] C. Kirner, *Historia da RV no Brasil: Ações*, Retrieved March 14, 2008, from http://ckirner.com/historiarv/acoes.htm

[5] C. Kirner, *Historia da RV no Brasil: Criação da CERV*, Retrieved March 14, 2008, from http://ckirner.com/historia-rv/cerv.htm

[6] C. Kirner, *Historia da RV no Brasil: Eventos*, Retrieved March 14, 2008, from http://ckirner.com/historiarv/eventos.htm

[7] C. Kirner, *Site de RV no Brasil*, Retrieved March 14, 2008, from http://www.realidadevirtual.com.br

[8] P. Milgram, et. al. "Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum". *Telemanipulator and Telepresence Technologies, SPIE*, V.2351, 1994, pp. 282-292.

[9] SBC, *Página da CERV*, Retrieved March 14, 2008, from http://www.sbc.org.br/cerv/

[10] W.R. Sherman and A.B. Craig, *Understanding Virtual Reality: Interface, Application, and Design*, Morgan Kaufmann Pub., San Francisco, CA, 2003.

[11] I.E. Sutherland, Sketchpad: *A Man-Machine Graphical Communication System*, PhD Thesis, MIT, January 1963, Technical Report No. 574, University of Cambridge, UCAM-CL-TR-574.

[12] I.E. Sutherland, “The ultimate display”, *in Proceedings of IFIPS Congress*, New York City, NY, vol. 2, May 1965, pp. 506-508.

[13] I.E. Sutherland, "A Head-mounted Three-dimensional Display," *in 1968 Fall Joint Computer Conference, AFIPS Conference Proceedings*, vol. 33, 1968, pp. 757-764.

[14] J. Tiffin and N. Terashima (Editors), “*Hyper-reality: Paradigm for the Third Millennium*”, Routledge, 2001.

[15] P. Wellner, W. Mackay and R. Gold (Editors), “Computer augmented environments: Back to the real world”, *Communications of the ACM*, 37(7), July 1993, pp. 24–26.

Session 1: Mixed Reality

Massively Parallel Computing Techniques: a Step Towards Real Time High Definition AR Applications

Thiago S. M. C. de Farias, João Marcelo X. N. Teixeira, Gabriel F. de Almeida, Veronica Teichrieb, Judith Kelner

Grupo de Pesquisa em Realidade Virtual e Multimídia (GRVM)
Centro de Informática (CIn) – Universidade Federal de Pernambuco (UFPE)
Av. Professor Moraes Rego, S/N, Prédio da Positiva, 1º Andar – Recife – PE, CEP 50670-901
{tsmcf}, {jmxnt}, {gfa}, {vt}, {jk}@cin.ufpe.br

Abstract

Single processor technology has been evolving across last decades, but due to physical limitations of chip manufacturing process, the industry is pursuing alternatives to sustain computational power growth, including the creation of multi-core systems. Parallel computing targets problems that are scalable and possibly distributed, dividing the problem into smaller pieces. This approach may be explored to satisfy real time constraints required by augmented reality algoritms. Its use may vary from realistic generation of virtual scenes to 3D reconstructions and image processing to perform optimized tracking. In this paper, the labeling algorithm, commonly used in the augmented reality pipeline, is implemented using a massively parallel paradigm. The implementation is able to provide satisfactory speed up improvements using CUDA, NVIDIA's architecture for GPU programming. The aim of this paper is not to present a new technology, but to show the great improvements that can be obtained by applying it in computer vision and augmented reality applications.

1. Introduction

Single processor technology has been evolving across last decades. For years, programmers have created their applications based on the premise that an enhancement in performance meant a higher clock frequency of operation. Unfortunately, because of physical and architectural bounds, there are limitations on the computational power that can be achieved with a single processor system. Due to physical limitations of chip manufacturing process, chipmakers currently pursue alternatives to sustain computational power growth, including creation of multi-core systems embedded on a chip. One of these implementations can be found in GPU (Graphics Processing Unit) industry. Since nowadays GPUs are seen as high performance units, this capability made parallel processing paradigm stronger. It demands another type of development, and not every application can take advantage of it.

Parallel computing targets problems that are scalable and possibly distributed, dividing the original problem into smaller pieces that will be solved in different places. Several parallel approaches may be used to speed up the solution of a given problem, such as: effective use of multiple cores of a processor (through OpenMP [1], or some third part library); use of a distributed solution (using a middleware, or even a network grid).

In case of taking advantage of processor multiple cores, or using a machine with more than one processor, the solution will scale up to the number of cores in the CPU, or the (number of CPUs) x (cores in each CPU). In spite of this solution being not rare, it is not highly scalable, since the newest powerful mainstream processor is a Quad Core [2], which provides a maximum speed up of 4x, if no synchronization pass is needed in the process.

Scalability is a key point of the network grid solution. The achieved result is optimal if the problem can be split into independent parts. Although, in case there is any communication between the cells of a grid, the network latency will slow down the entire process.

An alternative for making a parallel version of an algorithm is to use a massive parallel computing device as a coprocessor. The GPU can be considered a representative example. Its use for solving problems not related with image generation and rendering, named GPGPU (General Purpose computing on Graphics Processing Unit), is commonly seen for algorithms in the image processing area, computer vision, and several research areas that can make benefit of massively parallel processing. GPUs usually process millions of pixels per second through its several pixel shading pipelines (a common high definition video in 1080p resolution has about 2,073,600 pixels at a frame rate of 30 Hz, which represents 62,208,000 pixels per second).

Although this amazing processing power may be used for general purpose computations, a GPU is not optimized for this function. There are problems intercommunicating the processing parts (implemented in fragment programs). Data can be shared and

optimized for reading (through data transformed into textures), but the result of each processing unit computation is only visible outside the scope of the GPU, since it is written in a texture (using a render to texture technique), as well. The time spent with memory transfers between host and GPU is also an issue and is certainly one of the problems to be solved by next generation GPUs.

The massive parallel computing approach may be explored to satisfy real time constraints required by augmented reality (AR) applications. Its use may vary from realistic generation of virtual scenes to physics simulation. It could also aggregate data from 3D reconstructions and image processing to perform optimized tracking, providing ambient occlusion with virtual objects. In this paper, the labeling algorithm (commonly used in the traditional AR pipeline) is reimplemented using the new massively parallel computing paradigm, delivered by NVIDIA CUDA [3]. Despite the fact that this architecture represents a brand new technology, the implementation is able to provide very satisfactory speed up improvements (the transition between sequential and parallel paradigms will be discussed in Section 4). The authors highlight the possibility of working with images from High Definition (HD) videos (about 1080p). Before the massively processing technology, the idea of processing images with resolutions above 800x600 pixels was nearly unachievable, due to the use of sequential or even ordinary parallel approaches. The aim of this paper is not to present a new technology, but to show the great improvements that can be obtained by applying it in computer vision and AR applications.

The next sections are divided as follows. Section 2 explains both hardware and software infrastructure provided by CUDA. Section 3 presents related works that also use GPGPU for performing tasks done traditionally in a sequential way. Section 4 details the case study chosen, in which the authors' labeling algorithm implementation is described. The basic sequential algorithm is presented and then its correspondence and modifications on the GPGPU side. Section 5 describes our test scenarios and lists the results obtained. At last, Section 6 gives some conclusions about the possibilities of using the GPU as a coprocessor for enhancing AR functions performance, and points some future works in this area.

2. CUDA

The G80 processor, NVIDIA's first implementation of Compute Unified Device Architecture (CUDA), is an attempt of making GPGPU available for the average programmer, and a step into the High Performance Computing (HPC) market. G80 hardware is accessed through a low-level parallel thread execution virtual machine, and a virtual instruction set architecture called PTX. When the application is transferred to the target hardware, PTX code is translated to the device instruction set.

Contrary to ATI's interface to their hardware, named Close To Metal (CTM) [4], that has a very well defined low level interface, NVIDIA preferred to focus on a C-based language, although it still requires a great knowledge of how the GPU works, and how it is organized.

The GeForce 8800, the first G80-based card, has 16 groups of 8 scalar processors each, totalizing 128 processors, which are configured to run at 675 MHz. Since the execution unit uses double pumped frequencies, the resulting clock is 1350 MHz. This architecture enables the GPU to use these groups, called multiprocessors, to process blocks of 64 to 512 threads. These blocks are divided into groups of 32, called warps, and are used by the processor for scheduling, since they do not work on a thread-level context switch. Kernel is a program, executed as blocks of warps, but the arrangement of threads in blocks and blocks into grids of blocks are defined by the programmer, as seen in Figure 1.

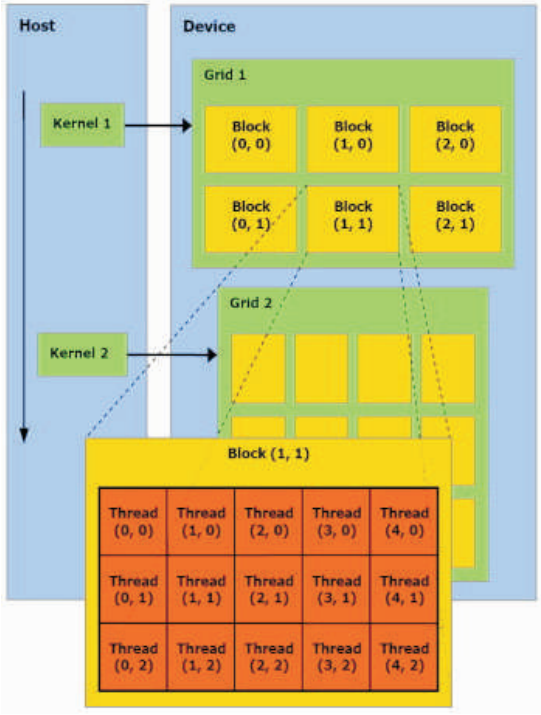


Figure 1. The G80 execution model [3].

Each multiprocessor contains 8 generalized processors (or Scalar Processors - SPs) and 2 Special Function Units (SFUs). The SFUs are used to calculate more complex instructions, like sine, cosine and logarithm. Special instructions are several times slower because of the precision required, but it is possible to execute most instructions faster in a less precise mode. The main explanation is that speed, rather than precision, was a major concern when rendering 3D scenes. An integer multiplication, for example, is processed by SFU and requires 8 cycles, while the lower precision mode of this instruction (24 instead of 32 bits) can be executed by the standard processors in only 2 cycles.

The memory model implemented on the G80, shown in Figure 2, defines the following scope for read/write operations [3]: per-thread for registers, per-thread for local memory, per-block for shared memory, and per-grid for global memory. For constant memory and texture memory, per-grid read-only operations are enabled, and only reads from texture and constant memory are cached. Some of the memory is located on chip, like registers and shared memory, while texture, constant, local, and global memory are implemented in device memory.

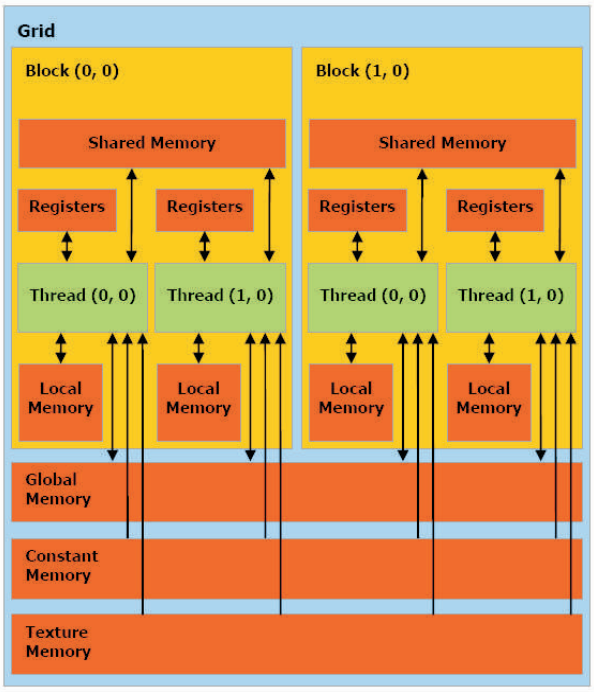


Figure 2. The G80 memory model [3].

This architecture demonstrates that the GPU is designed for highly parallel computation, and high arithmetic intensive; there is no point in using a GPU for any task that is not massively parallel. The G80 compared to an optimized calculator, so the scope of applications that can benefit from GPU execution is not as big as if having a massive multi-core CPU.

There are some guidelines that should be considered when developing G80-based applications, in order not to compromise performance. First of all, there are some hardware limitations that the application needs to stay within boundaries. The maximum number of threads per multiprocessor is 768, or 24 warps; these threads must be organized in a maximum number of 8 blocks per-multiprocessor, and 512 threads per block. As for per-multiprocessor memory limits, there are 8192 32-bit registers, 16 KB of shared memory, 8 KB of cached constants and 8 KB of 1D cached textures.

Memory latency is another major topic, since the cost of memory access depends on its location. Contrary to what can be seen on hierarchical memory architecture, local memory is not faster than shared memory; actually, it is several times slower, and thus cannot be cached. That is because local memory is a partition of the device memory, so it is important to maximize the use of faster, on-chip, shared memory and registers.

Another issue when developing using CUDA is to appropriately feed the G80 with enough threads for execution, as the processor can schedule millions of threads. This way, in order to help developers design efficient applications (in terms of memory and processor usage), NVIDIA released a CUDA Occupancy Calculator; using just a few parameters, as threads per block, registers per thread and shared memory per block, the programmer can know how much they can improve CUDA applications.

NVIDIA constructed a powerful piece of hardware, accompanied by useful software and even a line of products for high demand customers. As CUDA is a hardware platform, it can be expected even more processing power on years to come, for less money; and as the competitors wish to enter this market, it possibly will be seen improvements on ATI's side as well.

3. Related work

GPGPU computation has been widely adopted in several research areas, such as audio and signal processing [5], medicine and biological science [6], image processing, among others.

Focusing in the image processing area, which is directly linked with AR systems, many algorithms have been ported to the GPU platform using fragment programs to have access to the hardware as a co-processing unit. Some examples of image processing techniques

implemented in GPU are found in GpuCV [7], an implementation of the OpenCV [8] library using acceleration in GPU.

Also, there is the GPU_KLT [9], a well known feature tracker based in the original KLT algorithm [10], which is used by most 3D reconstruction systems. This work is also implemented using fragment programs, which suffers from hardware optimization.

Another library focused in parallel algorithms, this time using CUDA, is the CUDDP [11] (CUDA Data Parallel Primitives). CUDDP's objective is to group data parallel algorithm primitives in order to execute tasks like sorting, stream compaction, building data structures such trees and summed-area tables (using a scan algorithm). CUDDP was created initially to wrap up the implementation part of the algorithm described in [12].

All these projects aim to speed up an algorithm or technique, minimizing the contribution of the time spent by the procedure in the entire process. It is important to take advantage of the time that was initially used by the algorithm to do more tasks and achieve better results, spending the same time.

4. Case study

Labeling of a binary image refers to the act of assigning an unique value to pixels belonging to the same connected region [13]. Many vision problems can be posed as labeling problems in which the solution to a problem is a set of labels assigned to image pixels or features. This algorithm is often exploited in marker based AR applications [14], and because of that fact it was chosen as the case study for this work.

In formal mathematical terms, the labeling problem is to assign a label from the label set § to each of the sites in *S*. Edge detection of an image, for example, is to assign a label f_1 from the set § = {edge, non-edge} to site i *S* where elements in *S* index the image pixels. The set $f=\{f_1, \ldots f_m\}$ is called a labeling of the sites in *S* in terms of the labels in § . When each site is assigned to an unique label, $f=f(i)$ can be regarded as a function with domain *S* and image § . Because the support of the function is the whole domain *S*, it is a mapping from *S* to § , that is, f: S → § . A labeling is also called a coloring in mathematical programming.

Since the labeling problem relies on image's global information for associating the connected regions, the processing unit must have access to all pixels inside its search range. In order to have the labeling algorithm implemented in parallel, one should be tempted to divide the problem in two consecutive steps: local labeling processing and further merge of the obtained labels. There is a tradeoff between these two steps: the easier and more parallel the local labeling step is, the more complex the merge becomes. In the approach used for performing this case study, the authors decided to perform a simple local labeling and at the same time create a hybrid (sequential/parallel) merge implementation.

The parallel labeling implementation is generic enough to divide an input image of $m \times n$ pixels in $m_b \times n_b$ blocks, each one able of being processed independently from the others. The variables m_b and n_b represent the division factor along the x and y axes, respectively. As a consequence, the size of each image block should be $\frac{m}{m_b} \times \frac{n}{n_b}$ pixels, as shown in Figure 3.

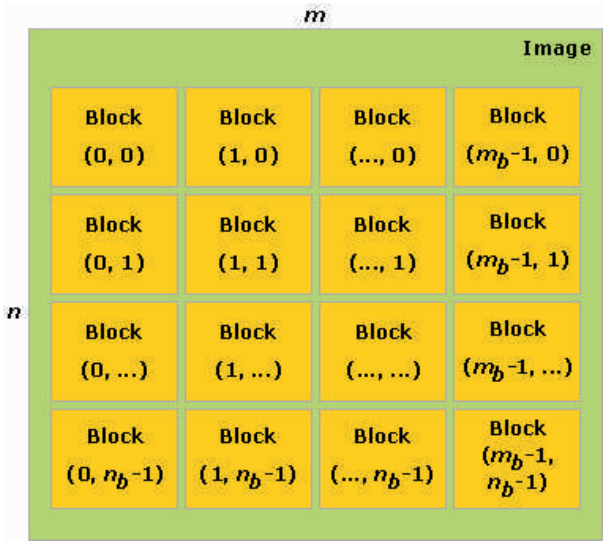


Figure 3. Input image divided in blocks

The local labeling algorithm implemented was the one described in [13]. The method scans a binary image from top to bottom and from left to right, per each line. Conceptually, the operations can be divided into four major steps that are illustrated in Figure 4a to Figure 4d.

In Figure 4a, when an external contour point, say A, is encountered the first time (it does not have any label yet), a complete trace of the contour is made until returning to the first point. A label is then assigned to A and to all points of that contour.

In Figure 4b, when a labeled external contour point A' is encountered, the scan line to find all subsequent black pixels (if they exist) is followed and assigned with

the same label as A'.

In Figure 4c, when an internal contour point, say B, is encountered the first time, B is assigned to the same label as the external contour of the same component. Then the internal contour containing B is traced and the same label as B is also assigned to all contour points.

In Figure 4d, when a labeled internal contour point, say B', is encountered, the scan line to find all subsequent black pixels (if they exist) are followed and the same label as B' is assigned to them.

The next step, the merge process, must wait until all processing blocks conclude their operations. In other words, the processing time of the local labeling phase is limited by the slowest processing block.

The hybrid merge is described as follows. At first, all blocks perform the boundary check at the same time (as shown in Figure 5), and then each one generates a segment of the list of label equivalences.

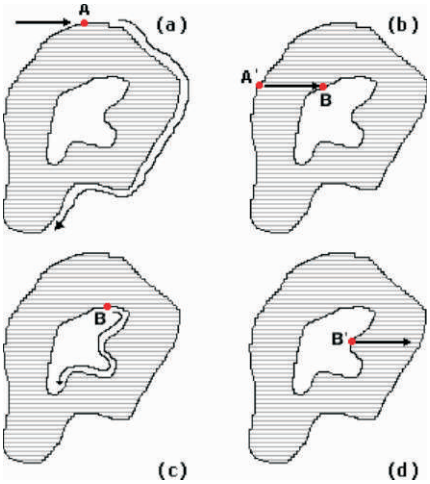


Figure 4. The four major steps in tracing and labeling.

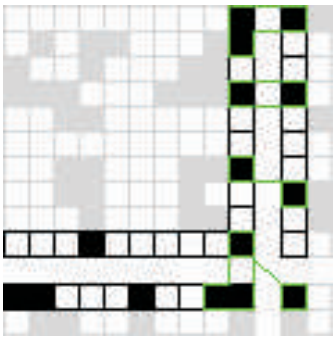


Figure 5. Parallel merge: each block checks its right and bottom boundaries for equivalences.

After that, a sequential loop iterates along the generated list of equivalences, and using a recursive approach the label values are grouped.

5. Results

The algorithm presented in Section 4 was implemented using the sequential approach on the PC platform, and using the massive parallel paradigm on the GPU. The computer used was an AMD Athlon X2 4800 processor with 1GB of RAM, running Windows© XP SP2. The device used as a coprocessor in the parallel approach was a NVIDIA GeFORCE 8800 GTX, with 768MB of GDDR3 memory. The tests were conducted in a non-biased environment.

The authors took for test an image composed by interleaved concentrical 1-pixel rectangles, which is considered a worst case scenario, because the trace function will walk through every rectangle in the image. The picture used has a resolution of 960x720 pixels, and a zoomed part can be seen in Figure 6.

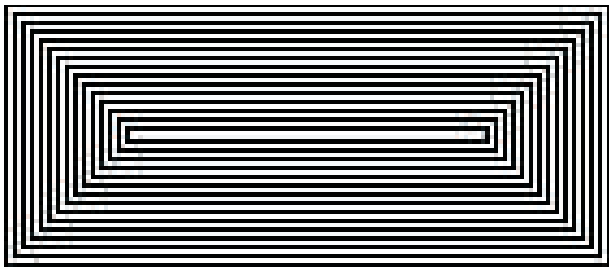


Figure 6. Zoomed central part (140x60) of the HD image.

For the sequential implementation, the average running time was 28.465 milliseconds, measured through 100 iterations, discarding outliers (about 3%). Using the GPU, the average running time was 4.858599 milliseconds, measured on the same terms of the sequential way, which gave us a speed-up of 5.858x, from 35.13 to 205.79 frames per second. The output of the algorithm is shown in Figure 7.

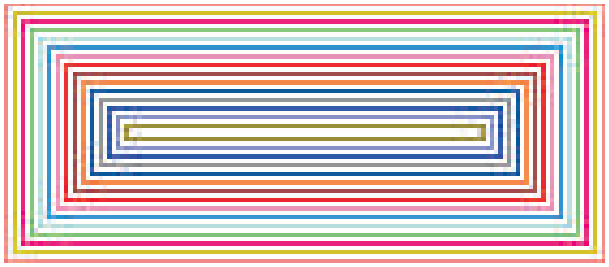


Figure 7. Labeling output. Each color means a diferent label assigned.

Another test scenario was used in order to obtain a performance comparison for the average case. The

adopted input image was the one shown in . It is composed by 3 different square patterns, and resulted from the binarization process. The sequential algorithm was executed in 9.649 milliseconds, while using the CUDA parallel approach it lasted the same amount of time corresponding to the worst case scenario. This represents a speed up of 1.9862x, and the parallel duration can be justified by the fact that since there are less black pixels in the image to work on, there are threads not being used. This way, the processing time of the algorithm in parallel is almost constant.

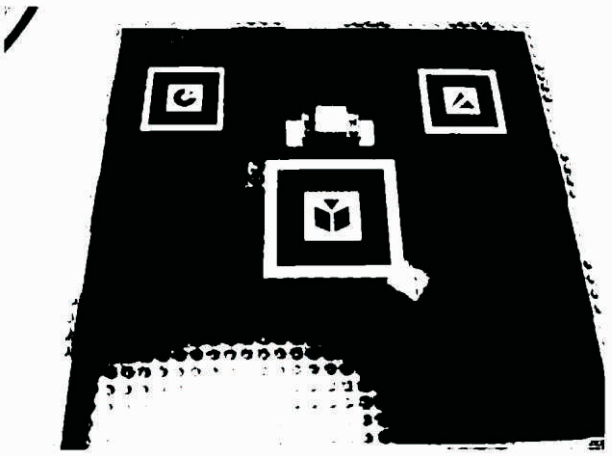


Figure 8. Input image for the second test scenario: average case with real markers.components.

The speed-up could be improved by the addition of a warm-up phase, which consists of a kernel code that accesses the texture and constant data from the device in the CUDA implementation, in order to intensify the use of their respective caches, minimizing the memory access time. Another optimization that could also be done was to convert the expensive operations (especially multiplication) to cheaper ones (mul24 instructions or shift operations, which take less clock cycles to be executed).

6. Conclusions and future work

The labeling algorithm taken as reference in this work is not the best example to explore the device's computational power, since it was created to deal with sequential approaches. However, by simply applying a divide-and-conquer technique, keeping the entire logical structure and adding a posterior merge phase, the authors just got the same algorithm running around 205 frames per second, almost 6 times faster than the sequential implementation on CPU.

Other algorithms included in the AR pipeline, even the simpler ones, like threshold, are better to transcript to the massive parallel approach due to its intrinsic independence of their executing parts. Taking threshold as an example, the data independence is in pixel level, since each pixel is compared to a fixed reference value, which means that the number of threads can be up to the number of pixels in the image.

Future work relative to the labeling algorithm consists basically in optimizing the code to remove the expensive instruction calls. It will increase even more the frame rate achieved.

The use of CUDA or any other massively parallel programming approach is a key to apply to the real time constraints in computer vision and AR offline results. Physics simulation, photorealistic rendering, 3D reconstruction for interaction between real and virtual worlds, and richer inputs and outputs to AR algorithms (High Definition Augmented Reality) are a good starting point to apply that new paradigm.

At this time, the use of a GPU as C coprocessor is only possible with last generation video cards, which have an average cost of US$500.00 (NVIDIA GeFORCE 8800 GTX). The cost-benefit relation is very good, considering the additional 500 GFLOPS (Giga Floating-point Operations per Second) provided by the video card model used to produce the results presented in this paper.

7. References

[1] OpenMP. Available: OpenMP site. URL: http://www.openmp.org/, visited on 30/11/2007.

[2] Intel Quad Core. Available: Intel site. URL: http://www.intel.com/technology/quad-core/index.htm, visited on 30/11/2007.

[3] NVIDIA CUDA Compute Unified Device Architecture Programming Guide. Available: NVIDIA site. URL: http://developer.download.nvidia.com/compute/cuda/1_0/NVIDIA_CUDA_Programming_Guide_1.0.pdf, visited on 30/11/2007.

[4] Close To Metal. Available: Sourceforge site. URL: http://sourceforge.net/projects/amdctm/, visited on 30/11/2007.

[5] N. Röber, M. Spindler, and M. Masuch, "Waveguide-based Room Acoustics through Graphics Hardware", *ICMC*, 2006.

[6] M. Charalambous, P. Trancoso, and A. Stamatakis, "Initial Experiences Porting a Bioinformatics Application to a Graphics Processor", *Panhellenic Conference in Informatics*, 2005.

[7] GpuCV. Available: GpuCV site. URL: https://picoforge.int-evry.fr/cgi-bin/twiki/view/Gpucv/Web/WebHome, visited on

30/11/2007.

[8] OpenCV. Available: Intel site. URL: http://www.intel.com/technology/computing/opencv/, visited on 30/11/2007.

[9] GPU_KLT. Available: Sudipta Sinha site. URL: http://cs.unc.edu/~ssinha/Research/GPU_KLT/, visited on 30/11/2007.

[10]J. Shi, and C. Tomasi, "Good Features to Track", *IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593-600.

[11]CUDDP. Available: GPGPU site. URL: http://www.gpgpu.org/developer/cudpp/, visited on 30/11/2007.

[12]M. Harris, S. Sengupta, and J.D. Owens, *GPU Gems 3*, *Parallel Prefix Sum (Scan) with CUDA*, Addison Wesley, August 2007, pp. 851-876.

[13]F. Chang, C. Chen, and C. Lu, "A Linear-Time Component-Labeling Algorithm Using Contour Tracing Technique", *Comput. Vis. Image Underst.*, Elsevier Science Inc., 2004, pp. 206-220.

[14]M. Fiala, "ARTag Revision 1, a Fiducial Marker System Using Digital Techniques", *Technical Report NRC 47419*, National Research Council Canada, Ottawa, ON, 2004.

Um sistema híbrido para rastreamento baseado em esferas retrorreflexivas e características do objeto rastreado

Lucas Teixeira, Manuel Loaiza, Alberto Raposo, Marcelo Gattass
Tecgraf, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro
{lucas}, {manuel}, {abraposo}, {mgattass}@tecgraf.puc-rio.br

Abstract

The tracking of objects for Virtual and Augmented Reality applications normally considers either the use of markers or markerless techniques, based on characteristics of the object to be tracked. The approach based on markers is more established, but requires the insertion of these "artificial" elements in the tracked object. Markerless techniques, on the other hand, are still in an experimental phase, offering several challenges to be overcome. This paper presents a hybrid system that tries to get the best of both approaches. Invariant properties of retroreflexive spherical markers patterns are used to detect the markers in the object. The inclusion of these markers in known polygonal areas of the tracked objects helps the detection of intrinsic characteristics of them, providing more robustness to the tracking process. A case study using a construction in ruins mock-up is presented.

Resumo

O rastreamento de objetos para aplicações de Realidade Virtual e Aumentada normalmente considera ou o uso de marcadores ou técnicas sem marcadores, baseadas em características intrínsecas do objeto rastreado. A abordagem baseada em marcadores é mais estabelecida, mas tem a desvantagem de exigir a colocação desses elementos "artificiais" no objeto rastreado. Técnicas sem marcadores ainda estão em fase experimental, apresentando muitos desafios técnicos a serem superados. Este artigo apresenta um sistema híbrido que tenta obter o melhor dessas duas abordagens. Propriedades invariantes de padrões de marcadores esféricos retrorreflexivos são usadas para detecção dos mesmos. A inclusão de marcadores em áreas poligonais conhecidas do objeto rastreado ajuda a obter características intrínsecas do mesmo, garantindo maior robustez no rastreamento. É apresentado um caso de estudo utilizando uma maquete de uma construção em ruínas.

1. Introdução

Os sistemas de rastreamento têm sido, por vários anos, ferramentas básicas e muito utilizadas na implementação de aplicações de realidade virtual e aumentada. O objetivo destes sistemas é realizar o rastreamento individual ou em grupo de diferentes objetos ou marcadores dentro de um cenário predefinido. Os marcadores podem variar desde marcadores artificiais que o usuário deseja colocar dentro da cena para representar pontos referenciais bem contrastados com resto de objetos da cena, até características próprias da cena, como cantos, arestas ou texturas. Entre as diversas tecnologias utilizadas para desenvolver estes dispositivos de rastreamento podemos citar a mecânica, magnética, sonora, óptica e híbrida. Entre elas, as tecnologias ópticas têm sido umas das mais utilizadas e desenvolvidas por conta de algumas vantagens que apresentam em relação às outras, tais como: seus sensores não precisam de fio, e são menos susceptíveis a ruído, permitindo o incremento do número de marcadores rastreados simultaneamente no mesmo cenário, sem que isto gere uma sobrecarga para o sistema, especialmente na parte de hardware.

Neste artigo é proposto um sistema de rastreamento óptico para aplicações de realidade aumentada que trabalha com marcadores sintéticos (padrões de marcadores) e pontos característicos próprios da estrutura do objeto real que desejamos rastrear e queremos enriquecer com informação virtual. Este sistema funciona como uma abordagem híbrida na utilização destes dois tipos de pontos referenciais. Inicialmente o sistema rastreia marcadores sintéticos tais como esferas retrorreflexivas colocadas em pontos bem definidos do objeto real. Esses pontos servem para calibrar e se obter uma posição inicial para a câmera, o que permite colocar a informação virtual em coerência com o modelo real do objeto rastreado. Uma vez bem identificados estes pontos, é iniciado um processo de aprimoramento na extração e inclusão de outras características próprias da estrutura do objeto real, tais

como cantos que ficaram distribuídos em áreas em volta dos marcadores sintéticos e cuja posição é bem conhecida no modelo do objeto real. A extração destas novas características será agilizada pelo fato de que onde elas deverão aparecer na imagem são áreas vizinhas às posições onde aparecem os marcadores sintéticos.

Esta nova abordagem tenta mostrar que este tipo de sistema híbrido pode ter um bom desempenho aproveitando as melhores características do rastreamento com padrões sintéticos assim como do rastreamento feito com base nas características próprias da estrutura do objeto real que será alvo do rastreamento.

Esse artigo prossegue organizado da seguinte forma. A seção 2 apresenta trabalhos relacionados. A seção 3 descreve a parte de hardware do sistema. A seção 4 descreve a extração e identificação dos marcadores, respectivamente. A seção 5 mostra como extrair características do objeto a partir da posição dos marcadores. A seção 6 apresenta um caso de estudo com uma aplicação dos algoritmos propostos usando uma maquete de uma construção em ruínas.

2. Trabalhos relacionados

Sistemas de rastreamento óptico utilizados na implementação de aplicações de realidade virtual e aumentada têm como tarefa principal a identificação de objetos, marcas ou estruturas específicas dentro de uma imagem, que são considerados os marcadores de rastreamento. Com esses marcadores de rastreamento, o sistema pode recuperar a geometria projetiva entre a câmera e a posição no mundo dos marcadores que aparecem na cena. O reconhecimento dos marcadores deve ser rápido e não ambíguo. Para tal efeito, diversas abordagens têm sido propostas, não só aprimorando o método de reconhecimento dos marcadores, mas também influenciando a definição de formato, cor, material e outras características visuais dos mesmos. Dentre as tecnologias mais comumente utilizadas podemos citar: marcadores fiduciais [7], marcadores retrorreflexivos [8, 9] e uma tendência chamada de "markerless" que se baseia no reconhecimento específico de características próprias da cena como: cantos, arestas, texturas e outros marcadores criados a partir da combinação das características antes mencionadas, que são adaptadas para serem usadas como marcadores de rastreamento da cena [6, 10].

O processo de rastreamento de marcadores pode ser individual ou grupal (padrões de marcadores). Comumente, a tecnologia utilizada para o pareamento e identificação individual e grupal de padrões de marcadores se origina em pesquisas feitas na área de reconhecimento de padrões e paralelamente utilizadas na área de visão computacional. Muitas vezes técnicas robustas e bem desenvolvidas da área de reconhecimento de padrões são "exportadas" para utilização na área de visão computacional. Em [14] é proposto o uso de duas técnicas utilizadas em reconhecimento de padrões, que juntas são aplicadas na implementação de pareamento de padrões de marcadores em um sistema de rastreamento óptico. Uma dessas técnicas é a teoria sobre as propriedades projetivas não-variantes (*projective invariant*) descritas em [1], que será explorada na implementação do rastreamento híbrido aqui proposto, o qual consistirá no rastreamento de marcadores artificiais e feições características contidas dentro da imagem, que darão robustez a um sistema de realidade aumentada proposto. Outro sistema híbrido, que trabalha com marcadores e feições naturais, é descrito por Kanbara et al [4]. Esses autores usam um sistema de rastreamento mais complexo, com câmeras estéreo e um sensor inercial para auxiliar no rastreamento.

3. Hardware

O sistema óptico desenvolvido é composto basicamente por uma câmera colorida e marcadores dispersos pelo objeto de interesse. Os marcadores precisam estar fixos em relação ao objeto, mas tanto a câmera como o objeto podem se movimentar.

3.1. Marcadores

Os marcadores ideais para esse sistema são os esféricos e que sejam capazes de ser rapidamente identificáveis na imagem.

O uso de marcadores esféricos é comumente usado para sistemas de rastreamento óptico [8, 9] porque a sua área projetada na imagem é comumente uma área circular. Calcular o centro do circulo é equivalente a calcular a projeção do centro da esfera sobre o plano da imagem. Isso porque o centro da esfera será projetado como sendo um ponto do raio que sai do centro da esfera, atravessa sua superfície, e intercepta perpendicularmente o plano da imagem. Esse ponto de intersecção ficará aproximadamente sobre o centro da área circular que representa o marcador. Utilizamos então as coordenadas da imagem do centro desta área circular para posicionar os marcadores dentro da imagem.

Outro ponto importante do marcador é sua fácil identificação na imagem. Para isso, o fenômeno da retrorreflexão é largamente utilizado. A retrorreflexão é a propriedade que alguns materiais compostos têm de,

ao invés de fazer a difusão da luz, refletir a maioria dos raios luminosos na mesma direção em que os raios foram emitidos [5].

Esse tipo de marcador é produzido por poucas empresas no mundo. O uso de fita reflexiva sobre esferas, recurso normalmente utilizado, não tem o mesmo resultado de uma esfera pintada com tinta retrorreflexiva. A solução encontrada para produzir os marcadores foi pintar esferas de isopor com tinta látex branca e, com a tinta ainda úmida, cobrir a superfície com micro-esferas de vidro. Essas micro-esferas de vidro sobre tinta branca geram o fenômeno da retrorreflexão. As micro-esferas usadas são as mesmas usadas na tinta de demarcação de rodovias, sendo um produto barato e relativamente fácil de encontrar.

3.2. Iluminador e lente

Para poder extrair os marcadores da imagem com o menor processamento possível, em nossa aplicação utilizamos a abordagem de iluminação infravermelha passiva, cuja estrutura consiste em usar: marcadores pintados com tinta retrorreflexiva, uma câmera equipada com lente especial sem filtro IR, e um pequeno iluminador formado por 4 leds infravermelhos colocados perpendicularmente à lente da câmera. Isso permite que os marcadores sejam facilmente contrastados com o resto dos objetos na cena. Em especial, este contraste será facilmente visualizado no canal vermelho extraído da imagem.

O ambiente de rastreamento terá a restrição de só poder estar iluminado por luz fluorescente cuja emissão interna de luz infravermelha seja muito baixa e não gere iluminação excessiva que possa criar falsos marcadores na área de rastreamento.

4. Características projetivas invariantes

Uma vez que os marcadores retrorreflexivos são extraídos e identificados, eles podem ser agrupados em padrões específicos previamente definidos e treinados para seu posterior reconhecimento. O método utilizado para definir padrões de formatos e o reconhecimento individualmente destes padrões é a teoria de características projetivas invariantes.

A implementação das propriedades projetivas invariantes tem sido um tema discutido especialmente na área de reconhecimento de padrões [11]. Ela tem sua base na propriedade do cross ratio invariante na projeção em perspectiva. A propriedade de cross ratio define que se temos 4 pontos colineares (A, B, C, D), podemos definir um valor de cross ratio baseado nas distâncias destes pontos de acordo com a seguinte relação:

$$\textbf{Cross ratio}\ (A, B, C, D) = \frac{|AC|/|BC|}{|AD|/|BD|}$$

Esta propriedade do cross ratio é expandida não só para atingir padrões colineares de pontos, mas também padrões com marcadores co-planares. Neste caso, o cross ratio é obtido com base nas áreas dos triângulos gerados usando combinações dos vértices que compõem um padrão co-planar, como mostrado em [13]. Esta variante estendida das propriedades projetivas não-variantes do cross ratio aplicada sobre pontos co-planares foi bem desenvolvida até ficar não-variante a possíveis permutações na etiquetação dos pontos que compõem os padrões co-planares. O trabalho de [12] mostra esta nova abordagem e testa a sua aplicação. O trabalho de [11] mostra outra extensão para gerar propriedades projetivas não-variantes para casos com mais de 5 pontos co-planares. Nos trabalhos acima, o identificador para a projetiva não-variante para um grupo de 5 pontos é definido como um vetor de 5 intervalos, com um valor mínimo e máximo para cada posição do vetor. Este vetor é obtido aplicando a técnica P2-Invariant sobre a amostra de 5 pontos, onde cada posição do vetor é relacionada a cada marcador que compõe o grupo de 5, o que segundo o autor permitirá um pareamento não só grupal, mas também individual para cada marcador que compõe grupo de 5. A única desvantagem é a necessidade de gerar todo o vetor de 5 intervalos para cada candidato de grupo de marcadores que quisesse ser pareado com um padrão específico.

Foi baseado nos resultados dos trabalhos acima descritos, que o algoritmo proposto definiu a configuração do padrão utilizado. Um padrão co-linear de 4 marcadores, mostrados na seção 3. O padrão tem duas características não-variantes à projeção em perspectiva, a co-linearidade e, por conseqüência, o cross ratio.

4.1. Colinearidade

A colinearidade é uma característica invariante à projeção em perspectiva que é explorada na nossa aplicação de rastreamento. Esta característica, além de ter um identificador único extraído com base na teoria das projetivas invariantes, pode gerar um formato específico que pode ser utilizado como um filtro pelo qual vários candidatos de "n" pontos na cena podem ser descartados se não cumprirem esta característica. Neste trabalho, o teste de colinearidade feito sobre padrões formados por 4 marcadores corretamente alinhados tem um custo computacional

aproximadamente de O(n). A idéia por trás do teste é:

1. Para cada grupo de 4 pontos, escolhemos os 2 primeiros pontos. Neste caso, os pontos não precisam estar ordenados.

2. Com os 2 primeiros pontos do conjunto de 4, calculamos a equação da reta que eles podem definir.

3. Para cada um dos 2 pontos restantes, calculamos a distância desses pontos para a reta.

4. Se as distâncias estiverem abaixo de um certo limiar, então o padrão é considerado válido.

5. Inicialmente, este primeiro limiar pode ser bem alto para permitir a aceitação de um bom número de fortes candidatos a padrões colineares. Num segundo teste, os padrões que passarem o primeiro teste de colinearidade são ordenados em relação ao eixo X primeiro e depois ao eixo Y, caso os pontos fiquem muito próximos nas coordenadas em X.

6. Neste segundo teste de colinearidade, padrões que realmente tenham um alto grau de colinearidade serão utilizados para o teste seguinte, que será o cálculo e comparação do valor da sua projetiva invariante.

4.2. Área envolvente

Uma segunda técnica utilizada para diminuir o custo computacional é a geração de uma caixa envolvente em volta da posição (no *frame* atual) onde são perfeitamente reconhecidos os padrões de rastreamento. Uma vez que um padrão específico é encontrado e validado, o sistema cria uma caixa envolvente com um certo dx e dy de largura e comprimento em volta do qual criaremos uma área de predição para que, num próximo *frame,* só aqueles marcadores que caiam dentro dessas respectivas áreas envolventes sejam considerados.

Este tipo de aprimoramento gera uma otimização e autonomia no processo de reconhecimento em tempo real.

5. Extração de características intrínsecas fortes

Características intrínsecas fortes de objetos, do ponto de vista do rastreamento, são características bem definidas, que podem ser usadas como pontos de referência no objeto para dar maior robustez no rastreamento.

No sistema desenvolvido, os marcadores são colocados em áreas poligonais do objeto passíveis de segmentação, para ser possível recuperar as posições dos pixels dos cantos dessas áreas poligonais. O processo de extração dessas características será explicado nas próximas seções.

5.1. Segmentação de regiões conhecidas sob os marcadores

As posições dos marcadores são conhecidas e as invariantes nos fornecem o identificador de cada marcador. Sendo assim, é possível usar um método de segmentação calibrado em pré-processamento. Foi usado o algoritmo Flood-Fill [2], principalmente pelo seu baixo custo computacional. Foi usada a sua versão que considera os 8 vizinhos de um pixel. O algoritmo itera sobre uma lista L de pixels (x,y) que inicialmente tem apenas a coordenada do pixel origem do Flood-Fill, que chamamos de pxFFOr. Nessa iteração o algoritmo retira o primeiro da lista L e visita todos os seus 8 vizinhos. Os vizinhos que atenderem ao critério de semelhança são marcados como pertencentes à região de pxFFOr e inseridos no final da lista L. O algoritmo itera sobre essa lista L até não ter mais nenhum elemento na lista.

O critério de semelhança usado para segmentar as esferas brancas foi o pixel ter a cor entre 155 e 255 em todos os três canais de cor. Apesar desse critério ser bem amplo, podemos fazer isso se colocarmos os marcadores em lugares de bom contraste com o fundo. O critério de semelhança usado para segmentar a forma poligonal sob os marcadores é igual, apenas mudando os intervalo de cor aceito, para de 0 a 100 em todos canais RGB. Os intervalos de cor foram escolhidos empiricamente. Essa parte do sistema pode ser substituída por qualquer técnica de segmentação que se adapte melhor ao objeto de interesse, desde que mantenha o custo computacional bastante reduzido.

Como queremos a segmentar por completo a região sob os marcadores, o resultado esperado é a componente dos marcadores somada com a componente da região em torno dele, que faz parte da região conhecida. Para isso primeiramente aplicamos o algoritmo de segmentação de marcadores, depois substituímos o valor dos pixels dessa região segmentada por um valor que é aceito pelo critério de semelhança da segmentação da região poligonal conhecida. Aplicamos o algoritmo de segmentação final iniciando em pxFFOr novamente. A resultante disso vai ser o que queremos, a soma da região sob o marcador com a da região dele, conforme ilustrado na Figura 1.

5.2. Busca de um pixel do contorno externo da componente conexa gerada pelo algoritmo Flood-Fill

O algoritmo *Flood-Fill* tem como saída uma componente conexa, mas não garante a inexistência de buracos nessa componente. Portanto, a busca pelo contorno mais externo da área segmentada pode falhar

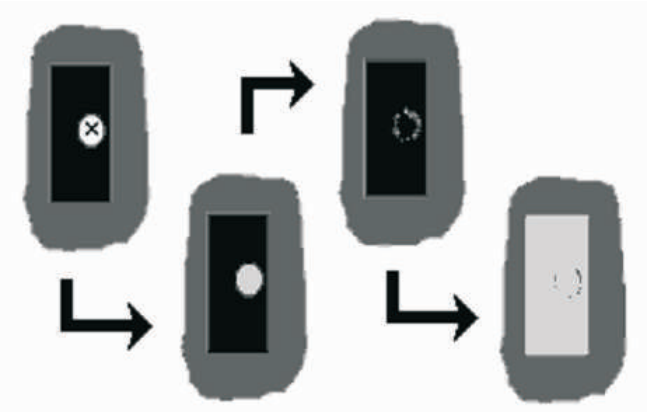


Figura 1. Representação do algoritmo de segmentação utilizado: a região poligonal conhecida em preto, o marcador em branco, o ponto detectado pelo sistema de rastreamento é o X sobre o marcador e em cinza claro os resultados das duas execuções do algoritmo de Flood-Fill.

se buscarmos apenas um pixel que seja vizinho de um pixel que não faça parte da componente conexa, pois esse pixel pode ser vizinho de um buraco interno do segmento, como os pontos indicados pelo quadrado pontilhado na Figura 2.

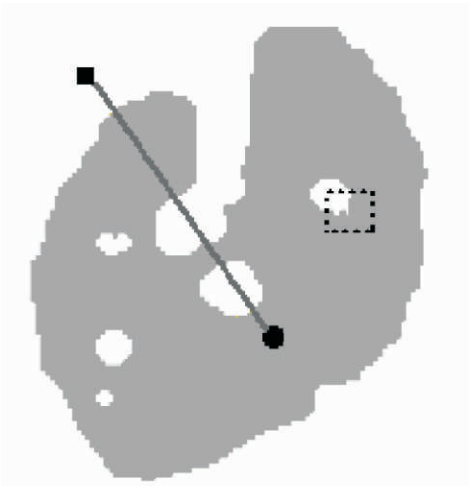


Figura 2. A componente conexa em cinza, o fundo em branco, um ponto qualquer pertencente à componente é o circulo preto, um ponto qualquer não pertencente à componente conexa é o quadrado preto.

Para o sistema, foi desenvolvido um algoritmo de busca do contorno externo de uma região segmentada (a componente conexa, que chamaremos a componente de Cc) na imagem. O primeiro passo do algoritmo é calcular uma caixa envolvente a Cc. Durante a execução do algoritmo de Flood-Fill é extremamente barato manter o valor do Xmin e Ymin e o Xmax e Ymax da Cc. Com esses quatro valores temos o retângulo envolvente alinhado com os eixos X e Y. Como esse retângulo envolve a Cc podemos afirmar que qualquer pixel fora desse retângulo está garantidamente fora da Cc. Pegamos um desses pixels fora do retângulo, que chamaremos de pxExt (quadrado preto na Figura 2). Outro pixel importante para nosso algoritmo é um ponto qualquer pertencente à Cc. Podemos usar o pixel pxFFOr, que também é a posição detectada do marcador, pois ele sempre vai pertencer à Cc uma vez que é o valor desse pixel que usamos de referência para gerar a Cc. Esse pixel vai ser chamado de pxInt (círculo preto na Figura 2). Se traçarmos um segmento de reta entre os pontos pxInt e pxExt, esse segmento de reta pode passar por um ou mais pixels de contorno internos ou externos, mas garantidamente ele passa por um pixel que faz parte do contorno externo, pois o pxExt é externo e para ele chegar a pxInt ele precisa atravessar o contorno externo. Desses vários pixels de contorno precisamos de um que pertença ao contorno externo. Para isso, percorremos o segmento de reta a partir de pxExt em direção a pxInt até a primeira vez que encontremos um pixel pertencente à Cc . Como iniciamos a busca em pxExt que pertence à região externa à Cc então a primeira vez que encontremos um pixel de Cc será o encontro das duas regiões que nos interessa, ou seja, o contorno externo. Para bem definir um contorno, precisamos armazenar tanto o pixel de contorno pertencente à Cc, que chamaremos de pxCin, como o pixel vizinho a ele pertencente à região externa, pois o pxCin pode pertencer a mais de um contorno, caso exista um buraco interno do outro lado do pixel de contorno.

5.3. Critério local para reconhecimento de vértices de retângulos

Normalmente critérios globais são usados para determinar os vértices de um polígono [3], mas critérios dessa natureza são caros por ter que percorrer pelo menos toda área segmentada.

No sistema desenvolvido podemos tirar vantagem do fato que os vértices fazem parte do contorno, de modo que o espaço de busca pode ser bastante reduzido. Já que é possível caminhar sobre o contorno da região na qual procuramos os vértices, é suficiente então determinarmos um critério para, dado um ponto do contorno externo, saber se ele é ou não um dos vértices do retângulo. O critério que propomos supõe que seja conhecida uma estimativa da orientação do retângulo. Com isso, podemos segmentar o retângulo em 4 quadrantes de modo que cada vértice vai estar em um quadrante diferente. Depois rotacionamos o quadrante de modo que a reta A fique alinhada ao eixo y e a reta B com o eixo y. Para os quadrantes 1 e 3 rotacionamos mais 45° no sentido anti-horário e para os 2 e 4

rotacionamos mais 45° no sentido horário. Feito isso temos uma aproximação de parabolóide no espaço, que chamaremos de espaço rotacionado formado pelo eixo x' e y', com ponto de máximo sendo uma boa aproximação do vértice que procuramos (linha tracejada na Figura 3). Como o contorno é discreto, podemos encontrar esse ponto de máximo testando, para cada pixel do contorno, se a reta paralela a x'que passa por esse ponto do contorno, passa em algum outro ponto dentro da região contornada. Para o vértice isso não ocorre. Isso seria caro se tivéssemos que testar, em todos os pontos dessa reta dentro da imagem, mas se a área retangular é razoavelmente bem determinada, o algoritmo Flood-Fill encontrará o retângulo com apenas pequenos problemas de locais como exemplificado na Figura 3 no quadrante 3 pela seta. Em pré-processamento podemos escolher um segmento de reta com tamanho razoável para escapar dos máximos locais analisando algumas imagens do objeto rastreado. Esse tamanho tende a ser bem pequeno, como de 10 a 20 pontos para cada lado do ponto sendo testado. Uma boa otimização desse critério é começar pelos dois extremos do segmento e ir iterando em direção do ponto sendo testado, pois é mais provável que encontre nos extremos desse segmento de reta um ponto pertencente à região interna ao contorno, e então o ponto já pode ser descartado como vértice. Essa forma de decidir se o ponto é vértice ou não deixa uma ambigüidade, pois se percorremos todo o contorno não saberemos se o vértice é o do quadrante 1 ou 3, o mesmo acontecendo com o 2 e 4, já que o módulo da inclinação x' é o mesmo para esses pares quadrantes. Para resolver esse problema, só buscamos no contorno dentro do quadrante correspondente.

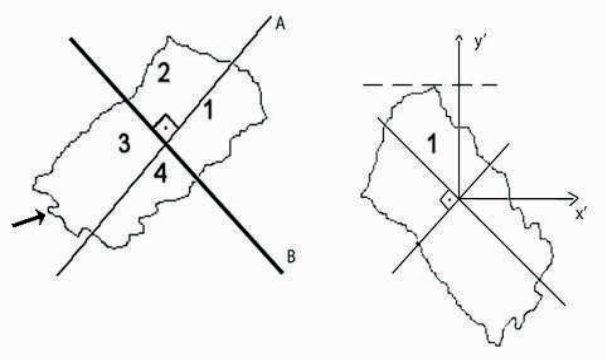


Figura 3. A imagem da esquerda é a área retangular com os eixos A e B que são a orientação do retângulo. A da direita mostra o espaço rotacionado.

6. Caso de estudo

Para testar nossos algoritmos usamos uma maquete (Figura 5) das ruínas do convento São Boaventura (Figura 4), localizado em Itaboraí/RJ, construído em 1660. A realidade aumentada é uma boa forma de criar interatividade e aumentar o interesse pela história. A aplicação final tem como objetivo a reconstrução virtual do convento, em cima da imagem das ruínas.

Figura. 4. Convento de São Boaventura.

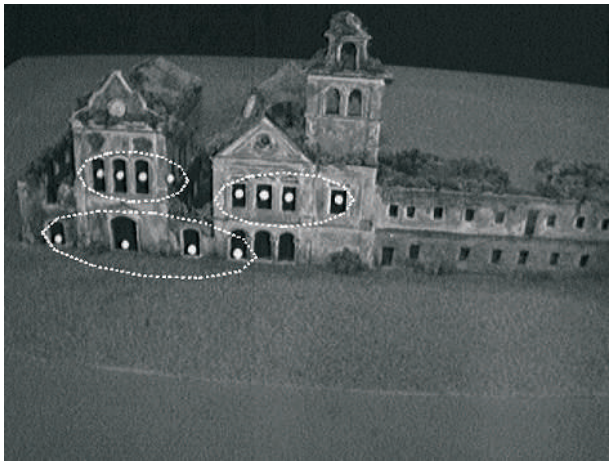


Figura. 5. Maquete do convento de São Boaventura. Em destaque estão 3 padrões colineares de marcadores.

6.1. Proposta da aplicação

A aplicação faz o rastreamento em tempo real de uma maquete com marcadores esféricos e retrorreflexivos presos a ela. Alguns desses marcadores estão em janelas retangulares. Como usamos a técnica de projetivas invariantes, temos um identificador único para cada marcador. Assim, caso o marcador for um que sabemos que está sobre uma janela retangular, extraímos os cantos dessa janela, para adicionar robustez ao rastreamento. Alguns outros critérios podem ser adicionados para decidir se adicionamos esses pontos para calibração como área mínima do retângulo na imagem e se o retângulo está muito cisalhado.

Após o rastreamento, aumentamos a imagem com partes das ruínas que não existem mais. A reconstrução do modelo original do convento está sendo objeto de pesquisa histórica e visitas a conventos similares.

A aplicação, atualmente, é feita apenas para ambientes fechados, dadas as restrições de iluminação explicadas na seção 3.

6.2. Etapas da aplicação

A aplicação foi divida em duas partes. A primeira é o treinamento. Nessa etapa, posicionamos a câmera em vários diferentes pontos de vista em frente à maquete e usamos essas imagens para calibrar o sistema. A segunda parte é o rastreamento, usando imagens capturadas com uma câmera em tempo real e renderizando a imagem aumentada correspondente.

6.2.1. Treinamento – Offline. A primeira etapa no treinamento é fazer um estudo sobre a iluminação e escolher o threshold adequado ao ambiente, para que os marcadores retrorreflexivos sejam bem detectados.

O segundo passo é calibrar as invariantes projetivas. Isso é feito usando as imagens dos vários pontos de vista com os marcadores já detectados e manualmente associando os marcadores correspondentes de cada padrão nos vários pontos de vista da maquete. Em cada imagem, a aplicação calcula o valor da invariante projetiva de cada padrão e armazena o mínimo e máximo para serem usados na etapa on-line.

Por fim, testamos o algoritmo de extração de características intrínsecas fortes, para determinar o tamanho mínimo do segmento usado no critério de busca pelos vértices descrito na seção 5.3 e quais marcadores associados aos padrões terão o retângulo ao redor detectado.

6.2.2 Rastreamento – On-line. O usuário altera o ponto de vista do objeto até que todos os padrões sejam detectados. Após isso, o sistema de rastreamento entra no modo de coerência temporal usando a caixa envolvente para o custo computacional ser adequado para tempo real. O reconhecimento dos padrões retorna a posição deste padrão na imagem e o identificador do marcador. Essa identificação é uma chave dupla composta pelo número do padrão ao qual ele pertence e qual a ordem dele dentro do padrão. Para os padrões que têm algum marcador com a área adjacente para identificar, calculamos a reta que melhor ajusta os posições dos 4 marcadores colineares do padrão pelo método dos mínimos quadrados. Como o padrão é colocado paralelo à base das janelas, com essa reta temos uma boa estimativa da orientação das mesmas. Usamos o algoritmo de segmentação da seção 5.1 para extrair a região conhecida sob o marcador e calculamos um ponto no contorno com o algoritmo da seção 5.2. Então percorremos o contorno e, para cada ponto, é testado o critério da seção 5.3 para saber se aquele ponto é um vértice. Por fim, usamos esses pontos para calcular a posição dos objetos virtuais nas coordenadas da imagem.

6.3 Testes e Resultados

Os testes foram feitos usando um vídeo feito com câmera fireware da Unibrain fire-i, o processador utilizado foi Intel P4 3Ghz. A maquete foi instrumentada com 3 padrões colineares, sendo que 2 deles estão em algumas janelas, dando um total de 7 janelas sendo segmentadas (ver Figura 5).

A tabela 1 mostra os tempos médios obtidos para as diversas etapas do processo de rastreamento proposto. A tabela 2 relaciona o número pontos errados que foram detectados dentro da área envolvente com o tempo médio do cálculo das projetivas invariantes.

Tabela 1. Resultados médios do processo de rastreamento.

Passo	Tempo médio
carregar imagem	1.8 ms
extrair os marcadores e calcular as invariantes no modo de coerência temporal	3.6 ms
calcular os retângulos	2.7 ms por retângulo
total	24.3 ms

Tabela 2. Tempo de cálculo das projetivas invariantes em relação ao número de pontos errados dentro das áreas envolventes.

Número de pontos errados	Tempo médio
0	1.5 ms
1	1.9 ms
2	2.7 ms
3	4.5 ms
4	7.3 ms
5	11.6 ms

O tempo para extrair os marcadc res e calcular as invariantes sem coerência temporal foi de 0.8 a 2.6 s.

O sistema proposto apresentou bons resultados no rastreamento da maquete. O tempo médio é tempo real, sobrando tempo adicional para técnicas de estimação da posição da câmera e renderização de objetos virtuais. O normal foi ter até 3 pixels errados dentro das áreas envolventes, por isso o tempo médio foi baixo. Os valores maiores em geral são decorrentes de movimentos muito bruscos.

A Figura 6 mostra a visualização da maquete com a inserção de janelas virtuais.

7. Conclusões e trabalhos futuros

Este trabalho apresentou um sistema de rastreamento híbrido, usando marcadores retrorreflexivos e características do objeto rastreado.

Um aspecto interessante do sistema é que, como os marcadores são discretos, é possível posicioná-los

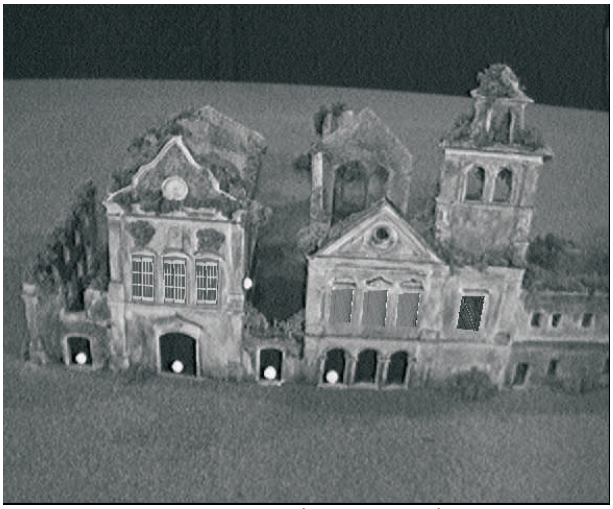


Figura 6 – Imagem aumentada com a renderização de fotos de janelas contemporâneas ao convento.

como foi feito nessa maquete, de forma que não ocupem grandes áreas na imagem, e que não façam oclusão de partes importantes da maquete. Com isso a imagem aumentada passa a poder mostrar mais partes da imagem real, sem prejudicar o rastreamento.

Como trabalho futuro, é necessário estudar uma maneira de reduzir o tempo para estimar as projetivas invariantes sem a coerência temporal e aumentar a robustez para permitir movimentos mais bruscos.

A aplicação apresentada é um protótipo de testes para o desafio de se utilizar realidade aumentada em ambientes abertos, sem o preparo do ambiente exigido quando se usam marcadores fiduciais

8. Agradecimentos

O Tecgraf/PUC-Rio é um grupo financiado principalmente pela Petrobras. O convento São Boaventura se localiza na região do futuro COMPERJ (Complexo Petroquímico do Rio de Janeiro). Agradecemos ao Sr. Roberto Menegon pelo apoio técnico no desenvolvimento das esferas retrorreflexivas.

9. Referências

[1] Suk, T., Flusser, J., “Point-based projective invariants”, *Pattern. Recognition (33)*, pp. 251-261, 2000.

[2] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, “Computer Graphics: Principles and Practice”, *Addison–Wesley,Reading*, pp. 689–693 , 1990.

[3] Claudio Rosito Jung, Rodrigo Schramm, "Rectangle Detection based on a Windowed Hough Transform", *Proc. of SIBGRAPI*, pp. 113-120, 2004.

4] Masayuki Kanbara, Naokazu Yokoya, Haruo Takemura, "A Stereo Vision-Based Augmented Reality System with Marker and Natural Feature Tracking", *VSMM*, p. 455, 2001.

[5] Helio Moreira and Roberto Menegon, “Sinalização Horizontal”, São Paulo, 2003.

[6] David G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 60, 2 , 2004, pp. 91-110.

[7] Fiala, M., "ARTag, a fiducial marker system using digital techniques", *CVPR 2005. IEEE Computer Society Conference on Computer Vision and Pattern Recognition,*2005., vol.2, no., pp. 590-596 vol. 2, 20-25 June 2005.

[8] Vicon Motion Systems - Optical Motion Capture Technology for Science, Engineering and Entertainment Industries. http://www.vicon.com – acessado em 11/2008.

[9] Advanced Realtime Tracking GmbH. http://www.ar-tracking.de, acessado em 11/2008.

[10] Williams, B. and Klein, G. and Reid, I., “Real-time {SLAM}relocalisation”, *Proc. International Conference on Computer Vision*, 2007.

[11] Meer, P., Lenz, R., Ramakrishna, S.: Efficient Invariant Representations. *International Journal of Computer Vision (26)*. pp. 137-152, 1998.

[12] Meer, P., Lenz, R., Ramakrishna, S., “Correspondence of Coplanar Features Through p2 Invariant Representations”, *Applications of Invariance in Computer Vision*, pp. 473-492, Springer - Verlag Press ,1993.

[13] Suk, T., Flusser, J.: “The features for recognition of projectively deformed point sets”, *Proceedings of the IEEE International Conference on Image Processing, pp. 348-351*, IEEE Press, Washington, USA, 1995.

[14] Manuel Loaiza, Alberto Raposo, Marcelo Gattass.: "A Novel Optical Tracking Algorithm for Point-Based Projective Invariant Marker Patterns", /Advances in Visual Computing, *Third International Symposium, ISVC 2007/*, pp. 160-169, Springer, Lake Tahoe, NV, USA, 2007.

Um Componente para Construção de Cenas com Consideração de Profundidade em Ambientes de Realidade Misturada

Silvio R. R. Sanches[1], Ildelberto A. Rodello[2], José R.F. Brega[3], Antonio Carlos Sementille[2],[3]

1 - Laboratório de Tecnologias Interativas (INTERLAB) – Universidade de São Paulo (USP)

2 - Programa de Pós Graduação em Ciência da Computação - Centro Universitário Euripídes de Marília (UNIVEM)

3 - Laboratório de Sistemas de Tempo Real (LSTR) – Universidade Estadual Paulista (Unesp)

silviorrs@gmail.com, rodello@univem.edu.br, remo@fc.unesp.br, semente@univem.edu.br.

Abstract

Most available support tools and software libraries, in spite of satisfactorily estimating the position and orientation in which virtual objects are supposed to be generated, do not usually allow that real elements superimpose on virtual ones. In certain situations, the problem may cause an incoherent relation of depth between the virtual objects and the real elements of the scene. The present paper proposes both the organization and the exhibition in a coherent way of real and virtual elements in Mixed Reality environments, by using the chromakey technique and the OpenGL framebuffer manipulation functions.

1. Introdução

Grande parte das ferramentas para desenvolvimento de sistemas de Realidade Misturada (RM) oferece funções para estimativas de posicionamento e orientação para que se insiram objetos virtuais no ambiente; no entanto, a cena real normalmente é exibida como plano de fundo durante todo o tempo de execução da aplicação, e, mesmo posicionado mais ao fundo, o objeto virtual em nenhum momento é obstruído por algo real.

Segundo Kiyokawa et al. [7], é desejável que, em ambientes misturados, objetos reais e virtuais mutuamente se ocultem (oclusão mútua), reproduzindo noções de profundidade essenciais para obtenção de maior realismo da cena.

Considerando o contexto exposto, apresenta-se, neste artigo, um método para composição de cenas tridimensionais em sistemas de Realidade Misturada, baseados na biblioteca ARToolkit [2], a qual utiliza a técnica do chromakey para extração dos elementos reais.

O método, implementado como um componente multiplataforma para o ARToolkit, utiliza um alphamap (mapa de transparência) do elemento real, que deve ser reproduzido em primeiro plano e realiza operações na imagem no framebuffer do OpenGL, anteriormente à exibição na tela. A utilização das funções desenvolvidas permite a composição de cenas misturadas com consideração de profundidade, possibilitando que elementos reais possam ser exibidos mais próximos do observador e, desse modo, obstruam os objetos virtuais dispostos na cena, se necessário.

A abordagem proposta produz uma solução para a oclusão mútua sem a necessidade de construir-se um mapa de profundidade (distância de cada pixel em relação à câmera) da cena, evitando um custo computacional elevado.

Para que favoreça seu pleno entendimento, este artigo está organizado da forma que segue.

A Seção 2 apresenta alguns trabalhos relacionados. Na Seção 3 abordam-se o problema da composição da cena com consideração de profundidade e os detalhes da solução proposta. A Seção 4 trata da implementação do módulo desenvolvido e uma análise de seu desempenho é realizada na Seção 5.

Finalmente, reservou-se a Seção 6 para as conclusões do presente estudo, e para apresentar as perspectivas de trabalhos futuros.

2. Trabalhos relacionados

Algumas experiências de RM e equipamentos utilizados na indústria de televisão, baseados em hardware especializado, permitem que se obtenha o mapa de profundidade da cena, o que possibilita sua construção respeitando-se a distância dos objetos (reais e virtuais) em relação ao observador (câmera). Entre essas soluções, cabe destacar o Display ELMO, a Zcam, o FlashKey e o Sistema 3DK.

O Display ELMO [7] consiste de um sistema formado por um display do tipo optical see-through equipado com mecanismos de percepção de profundidade. Por meio da comparação do valor obtido com os valores de profundidade da cena virtual, armazenada em um buffer, a visualização correta pode ser representada, em tempo real, inclusive em ambientes não controlados e

dinâmicos. Os resultados são aceitáveis, embora o recorte do elemento real seja grosseiro e visível na imagem final.

A solução desenvolvida pela empresa 3DV Systems para composições de imagens realiza a extração do objeto de primeiro plano, a partir de um plano de fundo arbitrário, utilizando a distância relativa à câmera de vídeo. Trata-se da câmera de vídeo com estimativa de profundidade Zcam [8]. A câmera produz sinais do tipo RGBD, em que D corresponde à distância de cada pixel. O sinal RGBD possibilita a criação de composições de múltiplas camadas de imagens.

O processo consiste em produzir um "muro de luz" que se move de acordo com o campo de visão. Esse sinal é gerado por um pulso de curta duração, que atinge a mesma área do campo de visão em um limite de dez metros. A luz emitida não é visível, para que não interfira na imagem da câmera. Quando refletido, esse sinal retorna à câmera, após passar por um processo de simplificação – o que consiste no bloqueio de algumas luzes de retorno para obter maior precisão nas informações de posicionamento em relação à câmera.

Um método denominado FlashKey [3], desenvolvido para produções de televisão da BBC (British Broadcasting Corporation), de Londres, também pode exibir objetos virtuais e reais com estimativas de profundidade. A pessoa, ou objeto real, que se deseja posicionar em frente ao objeto virtual, é iluminado por uma luz azul intermitente, e a câmera de vídeo reconhece a ação da luz em determinadas áreas da imagem. Para produzir essas luzes, utilizam-se LEDs acionados por um componente especialmente desenvolvido para o controle de sinais do vídeo. Processa-se, então, a imagem da câmera e gera-se um sinal nas áreas do azul brilhante, por meio de um programa de tempo real executando em um computador pessoal convencional. Simultaneamente, executa-se um procedimento para eliminar a luz azul da imagem, utilizando-se um filtro com características de cor e de tempo.

O sistema de estúdio virtual 3DK [9] consiste em um conjunto de câmeras equipadas com sistemas rastreadores, que fornecem informações a respeito da movimentação da câmera. As câmeras produzem sinais de vídeo relativos ao elemento de primeiro plano, enquanto um sistema gerador de imagens, normalmente um supercomputador gráfico, produz sinais que correspondem ao plano de fundo, e, opcionalmente, sinais de máscaras, contendo valores de profundidade. As imagens de primeiro e segundo planos são combinadas, normalmente por um dispositivo de chromakey, e a composição resultante fica disponível a outros componentes, que utilizam as máscaras geradas para produzirem outros efeitos, inclusive a consideração da profundidade para compor a cena. Cada sinal pode também ser utilizado como feedback para o alinhamento da câmera, auxiliando o operador a posicioná-la corretamente.

3. Composição do ambiente misturado

3.1. O problema

As soluções baseadas em hardware, mostradas na Seção 2, permitem a construção de cenas considerando a profundidade de objetos reais e virtuais, porém, são inviáveis para utilização em grande parte das aplicações RM, devido à necessidade da geração do mapa de profundidade da cena, o que exige elevado poder computacional e utilização de equipamentos sofisticados.

Para uma solução em software, a identificação da profundidade dos elementos que compõem o ambiente misturado, bem como métodos que possibilitem a extração do elemento real, para que este possa ser exibido em primeiro plano são possíveis de se implementar. Durante a execução, ambos os elementos devem obstruir uns aos outros, baseados em seus deslocamentos a cada quadro de vídeo.

3.2. A solução proposta

3.2.1. Estrutura. A estrutura do projeto, mostrada na Figura 1, permite a identificação de dois módulos, implementados e utilizados em conjunto com as bibliotecas OpenGL, glut (biblioteca de utilidades para programas OpenGL) e ARToolkit. O módulo "Chromakey" consiste de um conjunto de funções responsável pela extração do elemento de interesse, inclusão do novo plano de fundo, identificação da profundidade dos objetos (reais e virtuais), entre outras funcionalidades necessárias à composição da cena.

"Aplicações" é o módulo no qual se desenvolveram os protótipos utilizados nos testes.

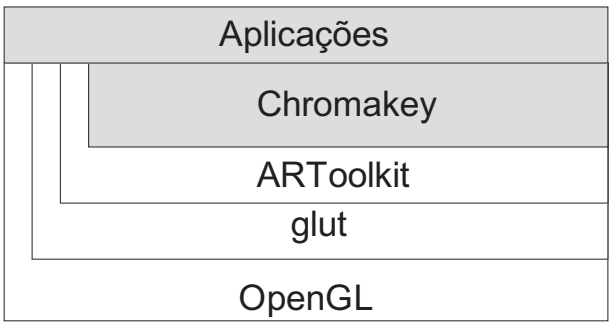


Figura 1. Estrutura do projeto.

3.2.2. Metodologia. Para a elaboração da metodologia utilizada no desenvolvimento do projeto, é necessário o entendimento da forma com que o ARTookit gerencia e exibe a imagem recebida do dispositivo de captura e os objetos virtuais, construídos por meio de alguma biblioteca gráfica. A API OpenGL, responsável pelo gerenciamento da janela na qual a cena é visualizada, utiliza um conjunto de buffers (framebuffer) para construir e exibir a imagem. O ARToolkit, que é baseado no OpenGL, transfere diretamente ao Color Buffer (buffer que contém a imagem a ser exibida na tela) a imagem capturada pela câmera, como mostrado na Figura 2.

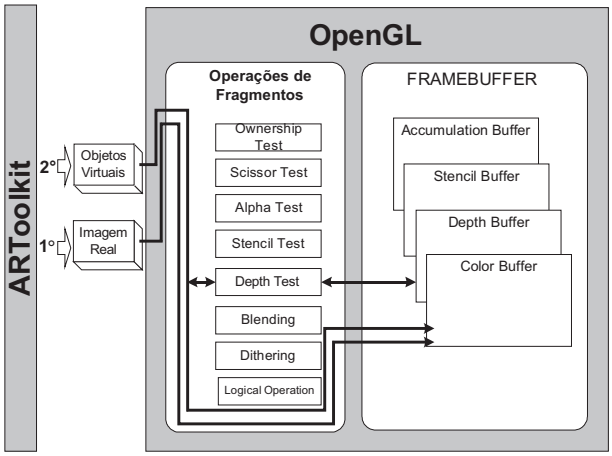


Figura : Método de geração da imagem do ARToolkit

Em relação aos objetos virtuais, gerados a partir de coordenadas tridimensionais, pode-se utilizar suas coordenadas "z" para controlar quais partes estarão visíveis na cena. Para isso, o Depth Test (utilização do buffer de profundidade do framebuffer) deve ser ativado (Figura 2). Os objetos virtuais são organizados, em relação às suas respectivas coordenadas "z", pelo Depth Buffer e, em seguida, copiados para o Color Buffer sobre a imagem real anteriormente carregada. Montada no Color Buffer, a imagem resultante é, então, exibida pelo dispositivo de visualização. Desta forma, mesmo com a movimentação dos marcadores (e, por conseqüência, dos objetos virtuais), faz-se o cálculo da profundidade a cada frame, exibindo-os corretamente. No entanto, a imagem real, capturada pela câmera e imediatamente transferida ao Color Buffer, permanece sempre ao fundo.

Entre as técnicas de recorte de elementos de primeiro plano mais conhecidas, o chromakey [1][4] (utilizado há algum tempo pelas indústrias de televisão e cinema) permite o isolamento de objetos colocados sobre fundo de cor constante, em tempo real, desde que tal objeto não contenha partes na mesma cor desse plano de fundo.

Considerando-se a possibilidade da utilização de planos de fundo constantes, propõe-se um ambiente formado por marcadores associados tanto aos elementos reais quanto aos elementos virtuais e um plano de fundo de cor única (azul ou verde).

O alphamap (mapa de transparência) do elemento real (elemento de interesse) pode ser extraído por meio da técnica do chromakey e o novo plano de fundo pode ser construído substituindo-se os pixels azuis da imagem original. Resultantes desse processo, obtêm-se a imagem do elemento de interesse à frente do novo plano de fundo e o alphamap armazenados. Para a obtenção das coordenadas dos elementos reais – necessárias à composição da cena – a solução utilizada neste trabalho foi a fixação de um marcador no elemento real, que é considerado o elemento de interesse. Desse modo, sua profundidade (coordenada "z") pode ser obtida pelas funções de rastreamento do ARToolkit, e utilizada para construção da cena no framebuffer do OpenGL antes da sua exibição.

De posse dos valores de profundidade de todos os elementos (reais e virtuais) que compõem a cena, pode-se utilizar o alphamap para sobrepor objetos virtuais, se necessário.

4. Implementação

Uma vez definidas a estrutura e a metodologia adotadas no projeto passa-se à fase seguinte: a da implementação das funções responsáveis pelas tarefas descritas na Subseção 3.2.2. Detalhes da elaboração do módulo "Chromakey", como a definição de suas funções, os tipos de planos de fundo e marcadores que podem ser utilizados, são descritos nesta Seção.

4.1. Funções do módulo

Dos módulos mostrados na Figura 1, o "Chromakey" é onde se implementa o conjunto de funções que operam como um componente para a biblioteca ARToolkit. A descrição de cada função implementada no módulo é apresentada na Tabela 1.

4.2. Tipos de planos de fundo

Com a retirada do fundo original, exigida pela abordagem, torna-se necessária a inserção de outra imagem como novo plano de fundo da cena. Foram implementados protótipos, onde o fundo azul (ou verde) é preenchido por um objeto virtual ou é substituído por uma imagem estática ou por uma seqüência de frames (Figuras 3, 4 e 5). O algoritmo de chromakey aplicado no módulo desenvolvido é o de

Berg e Lalioti [4].

Tabela 1. Funções do módulo Chromakey

Nome da função	Descrição
chLoadFile()	Abre o arquivo (imagem ou vídeo) que contem o novo plano de fundo.
chReadFile()	Faz a leitura do arquivo indicado pela função *chLoadFile()*, um *frame* por vez, durante a execução do *loop* principal do programa.
chSwapBackgroundBlue() chSwapBackgroundGreen()	Troca do plano de fundo de cor única pela imagem di arquivo e gera o *alphamap* do elemento de primeiro plano.
chVisibilityandDepth()	Realiza toda rotina para verificação da visibilidade dos marcadores e obtenção da profundidade dos objetos.
chDrawAlphamap()	Aplica o *alphamap* sobre a imagem armazenada no *framebuffer* do OpenGL
chCaptureZ()	Retorna a coordenada "z" do objeto virtual
chGetThresh()	Retorna o valor do treshold para chromakey, calculado pelo algoritmo de Berg e Lalioti [4]

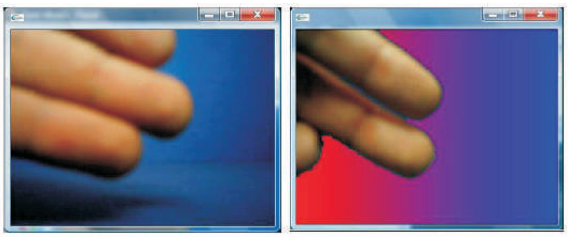


Figura 3: Substituição do fundo por um objeto virtual.

Figura 4: Substituição do fundo por uma imagem estática.

Figura 5: Substituição do fundo por uma sequência de *frames* (vídeo)

4.3. Marcadores "invisíveis"

Comum a sistemas de RM baseados em rastreamento de marcadores, uma limitação encontrada é a permanência destas marcas no ambiente misturado. Muitas vezes, o elemento virtual gerado não obstrui por completo o marcador real a que esteja vinculado. Isso ocorre em aplicações baseadas no ARToolkit e outras bibliotecas semelhantes. O fato de a abordagem proposta não utilizar o plano de fundo original para compor a cena resultante pode ser considerado uma alternativa para contornar esse problema. Para que não seja visualizado no ambiente misturado, o marcador real deve ser retirado, juntamente com o plano de fundo, no momento da aplicação do chromakey. Para isso, esses marcadores devem ser construídos em tons de cores próximos à cor do plano de fundo a ser extraído.

A identificação dos marcadores em cena, por meio do ARToolkit, é feita a partir de uma imagem binária. Por esse motivo, os marcadores "invisíveis" devem ser elaborados em tons de azul ou verde, diferentes o suficiente (ou distantes na escala de cores), para que possam, a partir do threshold do ARToolkit, serem identificados como preto e branco. Ao mesmo tempo, estas tonalidades devem ser parecidas (ou próximas na escala de cores) o suficiente, para que sejam reconhecidas como azul (ou verde) pelo chromakey, fazendo com que os marcadores sejam extraídos no processo. Naturalmente, a qualidade do dispositivo de captura da imagem e a iluminação do ambiente são importantes na utilização desse tipo de recurso.

Na Figura 6(a), mostra-se um dos marcadores, elaborado em preto e branco, e utilizado pelo ARToolkit. Obviamente, devido ao contraste, estas são as cores ideais a serem utilizadas em marcadores; no entanto, as marcas mostradas nas Figuras 6(b) e 6(c) foram testadas e apresentaram resultados satisfatórios nos processos de extração do chromakey e de detecção do ARToolkit, como será mostrado no protótipo da Seção 4.3.

Figura : Marcadores utilizados nos protótipos.

Nesse tipo de abordagem, um detalhe importante a ser observado diz respeito ao o processo de detecção do marcador, que deve ser feito na imagem original. Isso significa que a função arDetectMarker() do ARToolkit deve ser chamada antes da troca do plano de fundo, devido ao marcador não estar mais visível na nova imagem.

4.4. Composições com profundidade utilizando chromakey

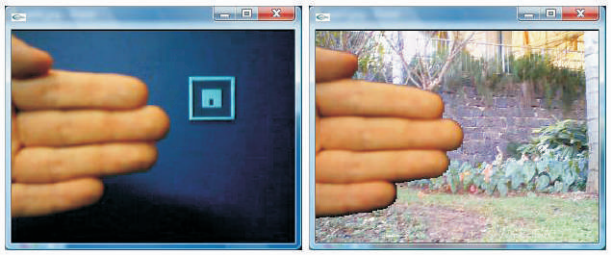
Verificada a viabilidade da utilização do chromakey para a extração dos elementos de primeiro plano, definidos os algoritmos que compõem as funções do módulo "Chromakey", os tipos de planos de fundo que podem ser carregados e os marcadores que podem ser utilizados, é possível desenvolver os protótipos do módulo "Aplicações", que implementa a estrutura descrita na Subseção 3.2.2.

4.4.1 Sobreposição do objeto virtual. As funções da biblioteca ARToolkit não possibilitam o desenvolvimento de aplicações em que os elementos reais sejam exibidos à frente de objetos virtuais. Utilizando-se o módulo desenvolvido neste trabalho, torna-se viável implementar esse tipo de recurso para aplicações executadas em hardware convencional. Descrevendo-se passo a passo este processo:

Inicialmente, o arquivo contendo o novo plano de fundo deve ser carregado, utilizando-se a função chLoadFile(), ainda em fase de inicialização. Por meio da função arVideoGetImage() do ARToolkit, um frame da imagem capturada pela câmera é armazenado em um buffer (Buffer#1). Os marcadores são detectados pela função arDetectMarker(), e o arquivo contendo o novo plano de fundo pode ser lido por meio da função chReadFile().

Utilizando-se o algoritmo de chromakey, implementado na função chSwapBackgroundBlue( ), a imagem é analisada, pixel a pixel, substituindo-se o fundo azul pela imagem contida no arquivo, e em seguida, transferida ao framebuffer. Nesse mesmo processo, o alphamap é gerado no Buffer#2 atribuindo-se o valor zero (0) aos pixels pertencentes ao plano de fundo, e o valor duzentos e cinqüenta e cinco (255) aos pixels pertencentes ao elemento de primeiro plano.

O resultado desse processo é mostrado na Figura 8. Na Figura 7(a) tem-se a cena capturada apenas com um marcador, fixado no fundo azul. Na Figura 7(b), exibe-se a imagem visualizada após a chamada da função chSwapBackgroundBlue().



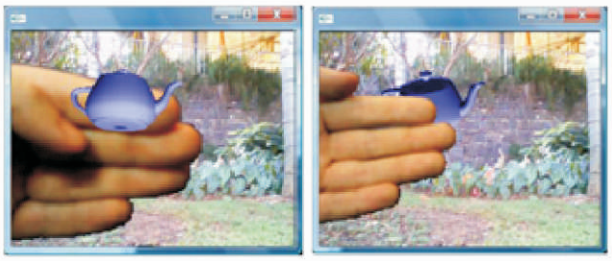
(a) (b)

Figura 7: Substituição do fundo e retirada do marcador do ambiente.

Como pode ser observado, o marcador foi elaborado em tons de azul, como mostrado na Seção 4.2, para que este não seja visualizado na imagem resultante. Em seguida, utilizando-se as coordenadas obtidas pela função arGetTransMat(), os objetos virtuais podem ser gerados e transferidos ao framebuffer.

Como se pode observar na Figura 7(a), o marcador está fixo ao fundo. Portanto, o elemento real (mão) deveria sobrepor o objeto virtual renderizado (chaleira). Para evitar-se o problema da oclusão, o objeto virtual foi transladado. Assim, como mostrado na Figura 8(a), tal objeto é exibido em primeiro plano, obstruindo o elemento real e tornando a cena incoerente em relação à profundidade.

Para sobrepor-se o elemento real ao objeto virtual, a imagem mostrada na Figura 8(a) é recuperada do framebuffer e trazida ao Buffer#1. Nesta imagem, o objeto virtual, gerado inicialmente a partir de coordenadas tridimensionais, já é parte do plano de fundo. O alphamap, armazenado no Buffer#2 é aplicado sobre a imagem, que, em seguida, é escrita novamente no framebuffer. Todo esse processo, que foi descrito de forma simplificada, é realizado pela função chDrawAlphamap(). Na Figura 8(b) apresenta-se o resultado obtido.



(a) (b)

Figura 8: Resultado após a chamada da função chDrawAlphamap().

4.3.2 Consideração da profundidade. Como foi demonstrado, as funções do módulo "Chromakey" permitem realizar a sobreposição do objeto virtual por algo real na cena. No entanto, no decorrer da execução, este elemento real também poderá estar posicionado mais distante ao observador que algum marcador vinculado a um objeto virtual e, portanto, deve ser obstruído por esse objeto. Para que se obtenha coerência na composição da cena, torna-se, então, necessário um método que permita a identificação automática da profundidade dos elementos (reais e virtuais).

Para isso, deve-se obter a localização do elemento real. A solução utilizada, como mencionado na Seção 3.2.2, foi a fixação de um marcador no elemento real (elemento de interesse). Desse modo, sua profundidade pode ser obtida pela função chCaptureZ(), definida dentro da função chVisibilityAndDepth().

Neste protótipo, buscando maior facilidade na implementação, o elemento real foi associado ao primeiro marcador da lista de marcadores da cena (obj_id=0). Foram utilizados três marcadores, definidos em um arquivo de configuração do usuário, que foram mantidos na cena para visualização de suas posições. Como no protótipo anterior, a função chLoadFile() é chamada durante a inicialização.

A imagem da câmera, capturada por meio da função arVideoGetImage(), é armazenada no Buffer#1. Os marcadores são detectados (arDetectMarker()) e o arquivo contendo o novo plano de fundo é lido (chReadFile()). A função chSwapBackgroundBlue() substitui o plano de fundo e armazena o alphamap da imagem no Buffer#2. Após o teste de visibilidade, feito por meio da função chVisibilityAndDepth(), para composição do ambiente baseado na posição dos marcadores, as coordenadas "z" obtidas são comparadas à coordenada "z" do marcador vinculado ao elemento real.

Se existirem objetos virtuais cuja coordenada "z" possua valor maior (mais distante) que a coordenada "z" do marcador real, esses objetos são gerados sobre a imagem do Buffer#1 (elemento real em frente ao novo plano de fundo), e escritos no framebuffer. Se for encontrado, pelo menos, um objeto virtual, a imagem contida no Buffer#1 deve ser recuperada e o alphamap, armazenado no Buffer#2, deve ser inserido sobre a imagem. Caso existam vários objetos virtuais, faz-se o controle de profundidade – nesta etapa realizado somente entre esses objetos – pela utilização do Depth Buffer (buffer de profundidade do framebuffer OpenGL) que é habilitado pela chamada da função e do respectivo parâmetro glEnable(GL_DEPTH_TEST).

Com a aplicação do alphamap, o elemento real será exibido à frente dos objetos virtuais já renderizados. As tarefas de recuperação da imagem do framebuffer e da inserção do alphamap sobre esta imagem são realizadas pela função chDrawAlphamap(). É importante observar que, caso não existam objetos virtuais mais distantes do observador que o elemento real, não há necessidade de utilização do alphamap.

O passo seguinte é a renderização dos objetos virtuais posicionados mais próximos do observador do que o elemento real. Novamente, nenhum controle adicional em relação à profundidade desses objetos virtuais é necessário além da habilitação do *Depth Buffer*. Para a visualização dos resultados, na Figura 9, têm-se o ambiente real e três marcadores em cena. A localização dos objetos virtuais renderizados é obtida pela detecção dos marcadores laterais, e o posicionamento do ator real, pela detecção do marcador ao centro.

Na Figura 10, em que se aplica o *chromakey*, o marcador localizado ao centro (vinculado ao ator real) está posicionado mais próximo do observador que os demais. Portanto, o elemento real é gerado em primeiro plano. Na Figura 11, o marcador vinculado ao ator está posicionado mais distante do observador que o marcador à esquerda e mais próximo que o marcador à direita. Portanto, o objeto virtual à esquerda é gerado em primeiro plano e o elemento real obstrui apenas o objeto virtual à direita (marcador posicionado mais ao fundo). O posicionamento do marcador vinculado ao elemento real mais ao fundo que os demais marcadores faz com que ambos os objetos virtuais sejam visualizados em primeiro plano, como mostrado na Figura 12.

Em todas as situações descritas, exceto na Figura 9, a imagem final é exibida coerentemente em relação à profundidade.

Figura 9: Objetos gerados pelo ARToolkit sobre o fundo real.

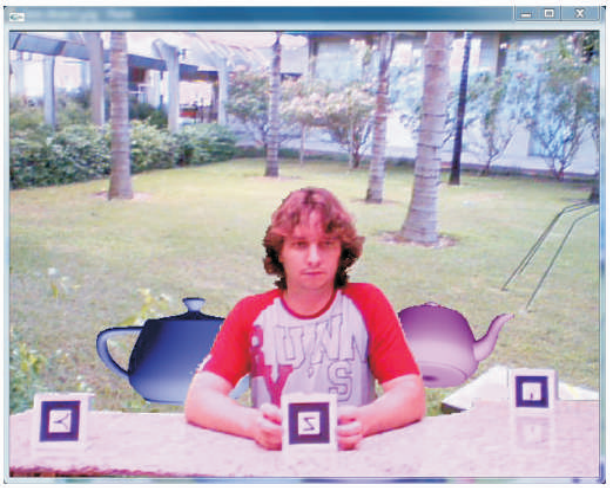


Figura 10: Elemento real em primeiro plano.

Figura 11: Elemento real entre objetos virtuais.

Figura 12: Elemento real posicionado ao fundo.

5. Análise de desempenho

5.1. Ambiente Experimental

Os experimentos foram realizados em um ambiente composto por uma estrutura em formato de caixa, com dimensões de 800 x 500 x 500 mm, com 3 lados adjacentes. O interior da estrutura foi revestido de papel cartão na cor azul. A câmera foi posicionada a uma distância de 450 mm do plano de fundo. Um estúdio para *chromakey* com fundo na cor verde, mostrado na Figura 13 foi utilizado para os testes finais. A câmera, nesse caso, foi posicionada a uma distância de 1,5 m do plano de fundo.

Figura 13: ambiente experimental.

Os demais componentes utilizados foram:

· Hardware: Processador AMD Turion 64 ML-32 1,8 GHz, Placa Gráfica ATI Mobility Radeon Xpress 200, 1,00 GHz de memória RAM e HD Toshiba ATA 80 GB.

· Software: Windows XP Professional SP2, Microsoft Visual Studio 2005 e ARToolkit versão 2.72.1.

5.2. Resultados obtidos

Considerados de tempo real, os sistemas de RM têm restrições quanto ao tempo de resposta. A maior parte das aplicações baseadas no ARToolkit utiliza hardware convencional, de baixo custo e de poder computacional limitado. Por esse motivo, direcionaram-se os testes realizados considerando alguns critérios de desempenho relevantes, a saber: a taxa de frames resultante da execução do protótipo de acordo com o tipo de plano de fundo; a quantidade de polígonos que formam o objeto virtual renderizado; a quantidade de marcadores em cena e o posicionamento dos marcadores.

Em todos os testes, em que o fundo foi substituído por uma imagem estática, utilizou-se o mesmo arquivo de

imagem. O mesmo procedimento foi adotado em relação ao plano de fundo dinâmico. A iluminação do ambiente também não foi alterada no decorrer dos testes.

Os valores mostrados nos gráficos das Subseções seguintes dizem respeito à média da taxa de frames, obtida das execuções do protótipo em questão. Cada protótipo foi executado sessenta vezes – trinta analisando imagens com resolução 320x240 pixels e trinta analisando imagens com resolução 640x480 pixels –, permanecendo em execução durante aproximadamente 20 segundos. A cada dez execuções de um mesmo protótipo, a máquina era reiniciada; em seguida, executava-se um novo protótipo até que as dez execuções de cada protótipo fossem realizadas. Ao final, repetiam-se os passos citados, até que cada protótipo fosse executado sessenta vezes, somando um total de seis baterias de testes. Procurou-se, com esses procedimentos, evitar que instabilidades provocadas por outros softwares influenciassem os resultados da avaliação.

5.2.1. Tipos de planos de fundo. Primeiramente, foram realizados testes para avaliar o desempenho do protótipo, quando o fundo azul (ou verde) é trocado pelos diferentes planos de fundo, mostrados na Seção 4.1. Para tanto, ao aplicar-se o chromakey, o plano de fundo foi substituído por um elemento virtual, uma imagem estática e uma seqüência de frames.

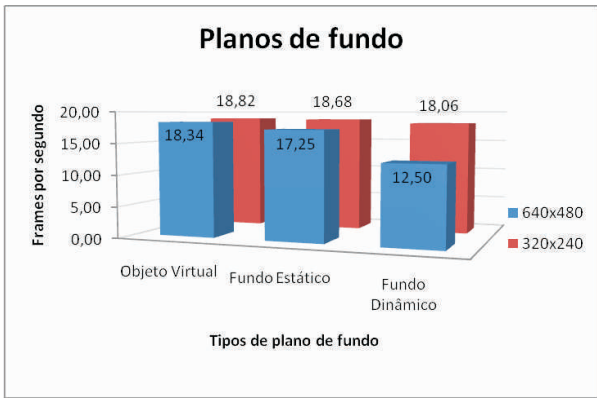


Figura 14. Tipos de plano de fundo.

Como se pode observar, no gráfico mostrado na Figura 14, à medida que aumentava a complexidade do plano de fundo inserido, ocorria uma queda no desempenho do protótipo. Essa queda foi mais acentuada no protótipo utilizando imagem com resolução 640x480, em que um arquivo de filme foi inserido como novo plano de fundo. Em relação à mudança de resolução de 320x240 para 640x480, a queda de desempenho dos protótipos não pode ser considerada significativa, pois a quantidade de pixels analisados foi quadruplicada (de 76800 para 307200).

5.2.2. Quantidade de marcadores na cena. Em aplicações baseadas no ARToolkit, um dado importante a se considerar relaciona-se à quantidade de marcadores visíveis na cena. Havendo regiões quadradas, semelhantes a marcadores, visíveis à câmera, o ARToolkit dispara as rotinas de comparação de padrões, ocasionando um atraso no processamento.

Considerando-se que o algoritmo de chromakey analisa a imagem pixel a pixel, o que aumenta ainda mais o custo computacional, realizaram-se testes para verificar a influência do aumento da quantidade de marcadores, visíveis à câmera, no desempenho do protótipo. Nesses testes, os marcadores foram posicionados para que, durante todo tempo de execução do protótipo, permanecessem visíveis à câmera. Para certificar-se desta visibilidade, objetos virtuais foram renderizados sobre essas marcas.

No gráfico da Figura 16, pode-se observar que a presença de marcadores na cena provoca uma queda acentuada no desempenho do protótipo – o que é verificado comparando-se os valores obtidos das execuções dos protótipos em que nenhum marcador é utilizado, aos valores obtidos das execuções dos protótipos em que apenas um marcador é reconhecido pelo ARToolkit.

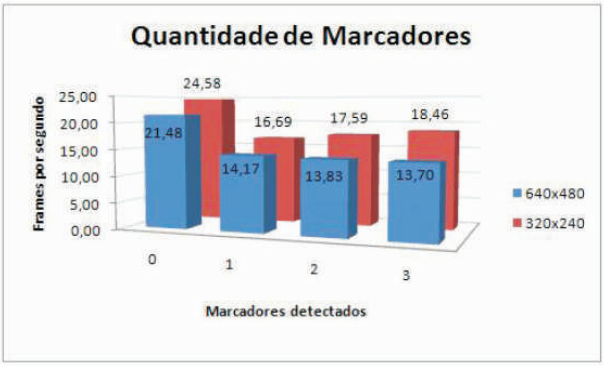


Figura 15. Quantidade de marcadores.

Em relação à presença de dois ou três marcadores, não houve alteração significativa no desempenho, exceto quando a resolução foi modificada de 320x240 pixels para 640x480 pixels. Nesse caso, a queda foi proporcional em todos os protótipos, provavelmente por estar mais relacionada ao aumento da quantidade de pixels analisados do que propriamente ao número de marcadores reconhecidos. É importante lembrar que, embora os testes tenham sido realizados com um máximo de três, um número maior de marcadores

pode ser utilizado nas aplicações.

5.2.3. Posicionamento dos marcadores na cena. Como descrito na Subseção 4.3.2, o protótipo que constrói a cena misturada, considerando a profundidade dos objetos, verifica o valor das coordenadas "z" dos marcadores, e decide pela utilização ou não do *alphamap*. Por esse motivo, o posicionamento dos marcadores em cena pode influenciar o desempenho do protótipo.

Os testes realizados consideraram três possíveis situações de ocorrência quando três marcadores estão visíveis na cena. Definindo-se a localização do marcador vinculado ao elemento real como referência para os testes, é possível identificar três situações: "o marcador está posicionado à frente dos demais", "o marcador está posicionado entre os dois outros marcadores" e "o marcador está posicionado mais ao fundo (mais distante) que os outros marcadores". Os resultados dos testes são mostrados no gráfico da Figura 17.

Como era esperado, o desempenho é afetado quando o elemento real está posicionado à frente de objetos virtuais. Nesse caso, a função chDrawAlphamap() é chamada para se aplique o *alphamap*, afetando o desempenho do protótipo. Obviamente, esse impacto é mais acentuado em imagens maiores (640x480 pixels), pois é maior a quantidade de pixels redesenhados.

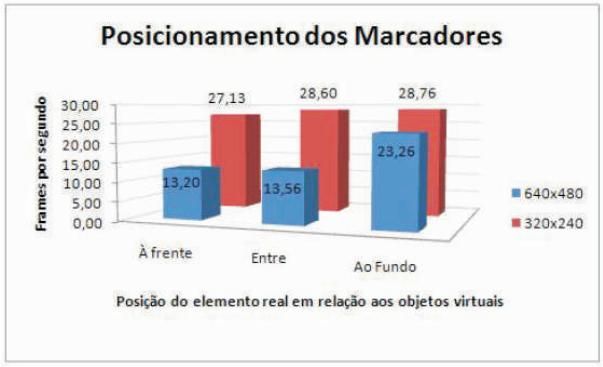


Figura 16. Posicionamento dos marcadores.

6. Conclusões e trabalhos futuros

Neste artigo, apresentou-se um componente multiplataforma para a biblioteca ARToolkit que possibilita a extração de elementos da cena real, com a finalidade de realizar composições com consideração de profundidade, em relação aos objetos reais e virtuais em ambientes de RM.

Buscando o isolamento do elemento real, propôs-se a técnica do *chromakey,* implementada em software, utilizando-se os serviços de captura de vídeo da biblioteca ARToolkit. Para composição da imagem final foram realizadas operações no framebuffer do OpenGL. Os testes realizados indicaram a viabilidade do processo.

Pretende-se, como trabalhos futuros, o estudo de formas de minimização do efeito de "serrilhamento" na silhueta dos elementos de primeiro plano, bem como a adição de sombras, a utilização de múltiplas câmeras e redundância de marcadores para minimizar problemas de oclusão. Além disso, pretende-se estender as funções do módulo para torná-lo compatível com diversos formatos de imagem e vídeo como plano de fundo.

Outra proposta de trabalhos futuros relaciona-se à utilização de modelos virtuais mais complexos. Nesse caso, a biblioteca OSGART [6], que é baseada no ARToolkit poderá vir a ser utilizada. Técnicas para extração de elementos de primeiro plano em imagens arbitrárias, como a implementada na biblioteca OpenCV [5], também podem ser avaliadas.

Um aspecto a ser observado diz respeito ao fato de a utilização do componente estar limitada a aplicações que aceitem planos de fundo de cor única. Por outro lado, tal restrição possibilita a não visualização do marcador no ambiente. Para tanto, as marcas devem ser elaboradas da mesma cor do plano de fundo para que sejam extraídas pelo chromakey.

7. Referências

[1] A. R. Smith, J. F. Blinn, "Blue Screen Matting", *in Proc. of the 23rd annual conference on Computer graphics and interactive techniques*, New York, NY, USA: ACM Press, pp. 259-268, 1996.

[2] ARToolkit "ARToolkit versão 2.72.1". Disponível em: <http://sourceforge.net/projects/artoolkit> Acesso em: 19 out. 2006.

[3] BBC Research, "Projects: Flash Keying". Disponível em: <http://www.bbc.co.uk/rd/projects/ virtual/flash-keying> Acesso em: 7 jan. 2007.

[4] F. van den Bergh, V. Lalioti, "Software Chroma Keying in an Immersive Virtual Environment", *South Africal Computer Journal*, no. 24, pp. 155-162, 1990.

[5] J. Bernardes, F. Miranda, D. Calife, L. Trias, R. Tori. Recife PE, Brasil: In A. Cardoso (edit), C. Kirner (edit), E. Lamounier (edit), J. Kelner (edit). "Tecnologias para o desenvolvimento de sistemas de Realidade Virtual e Aumentada", pp. 111-135, 2007.

[6] J. Loosea, R. Grasset, H. Seichter, P. Lamb, "Osgart artoolkit for openscenegraph". Disponível em: <http://www.artoolworks.com/ community/ osgart> Acesso em: 09 fev. 2007.

[7] K. Kiyokawa, Y. Kurata, H. Ohno, "An Optical see-

through display for Mutual Occlusion with a Real-Time Stereovision System." *Computers & Graphics*, v. 25, n. 5, pp. 765-779, 2001.

[8]R. Gvili, A. Kaplan, E. Ofek, G. Yahav, "Depth Keying", *Stereoscopic Displays and Virtual Reality Systems X,* vol. 5006, no. 1, pp. 564-574, 2003.

[9]S. Gibbs, C. Arapis, C. Breiteneder, V. Lalioti, S. Mostafawy, J. Speier, "Virtual Studios: An Overview", *IEEE MultiMedia*, vol. 05, no. 1, pp. 18-35, 1998.

Arquitetura para Suporte à Interação Multimodal em Sistemas de Realidade Aumentada

Carlos Eduardo Ribeiro, Marcio Merino Fernandes
FACEN-Ciência da Computação
Universidade Metodista de Piracicaba
Piracicaba-SP - Brasil
carlos@skiva.com.br, mmfernan@unimep.br

Resumo

A disponibilização de mundos virtuais a um público cada vez maior impulsiona o desenvolvimento de novos mecanismos de interatividade. Um requisito desejável é que sempre que possível, esses novos mecanismos sejam implementados utilizando-se tecnologias (hardware e software) convencionais, de modo a viabilizar o seu uso a um público mais amplo. Assim, o surgimento de novos dispositivos, tais como aqueles para o reconhecimento de fala, viabilizou novas possibilidades de interatividade, algo bastante desejável para o desenvolvimento de sistemas de Realidade Aumentada.

Assim, este artigo apresenta uma arquitetura genérica de hardware e software, visando facilitar a integração de dispositivos de interação multimodais em sistemas de Realidade Aumentada, em particular aqueles baseados no software ARToolKit.

1. Introdução

O avanço tecnológico e o crescimento da indústria fizeram com que a Realidade Virtual (RV) se tornasse viável, em diferentes áreas, permitindo desta forma a criação de hardware e software de baixo custo para aplicações em diferentes tipos de sistemas [4].

Com a disponibilização de mundos virtuais na Internet, os conceitos de navegabilidade e interatividade nos mesmos foram incorporados, permitindo interações até então não realizadas, como passeios em ambientes virtuais, manipulações de robôs etc., ocorrendo por meio de equipamentos convencionais como teclado e mouse, ou equipamentos especializados para esse fim. O surgimento de novos dispositivos, tais como, capacetes, câmeras etc., também tem contribuido para o crescimento da Realidade Aumentada (RA) [4].

Além disso, novas possibilidades de interatividade foram viabilizadas com o amadurecimento da tecnologia de reconhecimento de fala [5], contribuindo de forma significativa com o desenvolvimento de sistemas de interação. A partir do momento em que os mecanismos de reconhecimento de voz passaram a identificar fonemas, sílabas e palavras, e gerar instruções que passaram a ser executadas por computadores ou equipamentos eletro-eletrônicos, se viabiliza um novo conceito de interatividade entre homem e máquina, representando um conjunto de modalidades possíveis para a entrada e saída de informações entre usuários e sistemas.

O objetivo deste trabalho é desenvolver uma arquitetura baseada em hardware e software para sistemas de Realidade Aumentada, permitindo que dispositivos de interação multimodais [6] sejam integrados. Em termos de hardware a arquitetura se apoiará em computadores pessoais, redes e dispositivos de hardware dedicado. Em termos de software, fará uso de sistemas operacionais, software de rede, sistemas de reconhecimento de voz e software para RA comumente empregados, neste caso, em particular o ARToolKit [3], [1], cujo protótipo desenvolvido neste trabalho será denominado de ARToolKit Multimodal.

Os sistemas de computação atuais são baseados em hardware produzido em larga escala, são tipicamente computadores pessoais ou estações de trabalho utilizando os sistemas operacionais MS-Windows® ou Linux. Porém, diversos outros sistemas não se adaptam bem a essas plataformas em virtude de particularidades como, tamanho, consumo de energia, portabilidade etc., exigindo a utilização de dispositivos de hardware dedicado, ou ainda os chamados sistemas embarcados [2].

Com base no conceito de co-design de sistemas compostos por subsistemas baseados em hardware e em software, este trabalho visa propor uma arquitetura englobando essas duas possibilidades, dando heterogeneidade nas alternativas para o projeto. Desse modo, a arquitetura proposta pode ser dividida em dois subsistemas: um oferecendo suporte aos dispositivos de interação baseados em software, e outro àqueles baseados em hardware.

O subsistema para dispositivos baseados em software

deverá se apoiar em componentes de interface já existentes, cabendo ao projetista do sistema preocupar-se apenas com a implementação do mecanismo de interação em si. Deve ser notado que a utilização de interfaces baseadas no reconhecimento de voz ainda apresenta alguns desafios importantes, destacando-se entre eles: o processamento em tempo real, baixo custo, tradução do sinal acústico levando-se em consideração diferentes parâmetros como volume, sotaques de fala dos seres humanos, ambigüidade no significado de palavras, taxas de erro etc. Esse é um fator que deve ser levado em conta no desenvolvimento de protótipos, mas que pode ser atenuado na implementação de sistemas mais robustos.

O subsistema para dispositivos baseados em hardware deverá assumir o uso de dispositivos, desde os mais simples como, teclado, mouse e joysticks, até outros mais complexos e customizados, implementados utilizando tecnologias de circuitos integrados ou FPGAs (Field Programmable Gate Array).

Dessa forma, os subsistemas oferecerão o suporte necessário para o desenvolvimento de uma arquitetura possuindo características de portabilidade, e também, capacidade de extensão no que diz respeito à utilização de novos dispositivos de interface, oferecendo uma plataforma para a execução de sistemas de RA que possa acomodar dispositivos de interface adicionais sem mudanças ou adaptações substanciais.

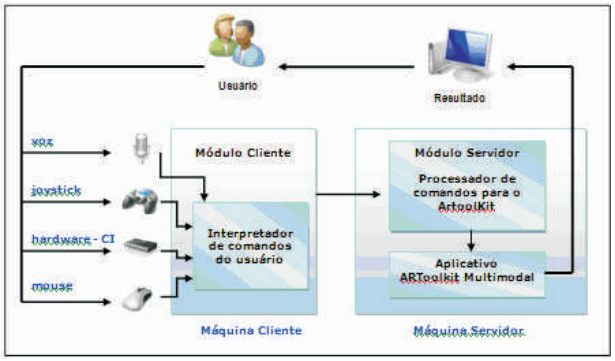


Figura 2. Modelo conceitual.

2. Modelo conceitual

A criação de um sistema de interação para RA que permita a utilização de interfaces multimodais baseadas em dados, voz, e outras formas, e que permita sua configuração e adaptação em diferentes sistemas, conforme suas necessidades, baseia-se em uma estrutura genérica, cuja arquitetura está dividida em dois módulos, os quais foram denominados de módulo cliente e módulo servidor. Sua implementação tem como objetivo realizar o reconhecimento dos sinais enviados pelas interfaces utilizadas pelo usuário, por meio de comando de voz, joystick, mouse ou outro dispositivo de hardware implementado para esta finalidade (Figura 1).

2.1 Módulo Cliente

A função principal do módulo cliente consiste em reconhecer os sinais dos diferentes dispositivos tratando os sinais e enviando a mensagem ao módulo servidor. Esta mensagem, por sua, vez poderá ser emitida para a própria máquina cliente, ou por meio da rede usando socket. Cada dispositivo de interação possui características particulares que deverão ser observadas (Figura 2).

Voz: Todo comando de voz será tratado por um reconhecedor de voz já existente, o qual apenas interpretar á os comandos fornecidos por uma lista de palavras (comandos) já pré-definida no módulo cliente.

Joystick: Cada stick do joystick irá possuir representações distintas no envio da mensagem, desta forma, representando as coordenadas X, Y e Z.

Hardware(CI): Equipamento desenvolvido especificamente para testar a aplicação multimodal, por meio da porta paralela.

Mouse: Será intuitiva na parte da aplicação, habilitando ou desabilitando interações, bem como participando da escolha de elementos de interação.

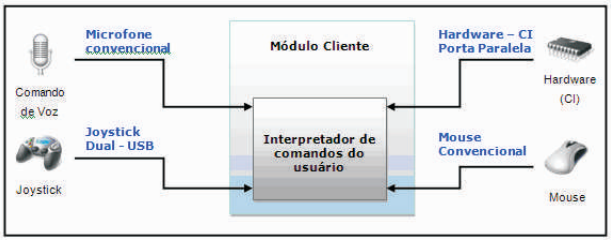


Figura 2. Módulo Cliente

2.2 Módulo Servidor

O módulo servidor tem como objetivo receber e tratar as mensagens enviadas pelo módulo cliente, gerando comandos padronizados ao aplicativo ARTool-Kit Multimodal. As implementações realizadas neste trabalho permitem que outros dispositivos de interação possam ser incorporados à aplicação, desde que sigam as especificações das mensagens enviadas ao ARTool-Kit Multimodal.

O módulo servidor deverá ser executado em uma máquina devidamente instalada e configurada em rede, cujo endereço deverá ser informado ao módulo cliente para que as mensagens sejam direcionadas, fazendo necessário a instalação do ARToolKit Multimodal. A

comunicação entre o módulo cliente e o aplicativo ARToolKit será realizada através de um arquivo de texto, gerado na base do aplicativo, denominado de "comando.dat"(Figura 3).

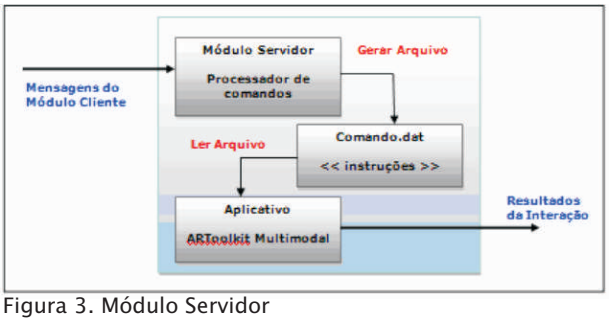


Figura 3. Módulo Servidor

O processador de comandos realizará a análise das mensagens, resolvendo conflitos entre dois ou mais dispositivos ou mensagens ambíguas que venham a interferir na execução do sistema. A interação entre o módulo cliente e o módulo servidor para troca de mensagens será realizada por meio de *socket*, indiferente se os módulos forem instalados na mesma máquina como uma aplicação local ou remota (Figura 4).

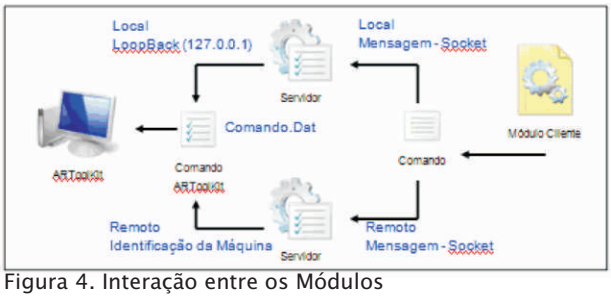


Figura 4. Interação entre os Módulos

3. ARToolKit Multimodal

Uma interface multimodal tem como objetivo receber sinais de diferentes dispositivos como um reconhecedor de comandos de voz, mouse, *joystick* e de hardware customizado. Além disso, o tratamento das informações recebidas deverá ser efetuado, de modo a gerar mensagens correspondentes a serem enviadas via *socket* até o servidor (Figura 5). Assim, o usuário poder á selecionar, um ou mais, dentre vários dispositivos de interação com o ARToolKit Multimodal.

As mensagens de interação geradas pela fusão dos dispositivos que envolvem a multimodalidade, será tratada pelo módulo servidor, eliminando a concorrência e a ambigüidade das mensagens.

4. Estudo de Caso

A arquitetura foi submetida a testes práticos com o objetivo de verificar a sua usabilidade. O primeiro teste com o ARTollKit Multimodal, foi a sua utilização em uma sala de aula para a demonstração de fundamentos de RV e RA, utilizando duas modalidades de interação: voz e *joystick*. O primeiro teste permitiu a análise do reconhecimento de voz em ambientes abertos, mostrando-se eficaz, mas com alta taxa de interferência produzida pelo som ambiente da sala. No mesmo cenário um segundo teste foi realizado, organizando os comandos principais no formato de árvore, cuja seqüência de comandos é iniciada por um précomando, reduzindo o índice de interferência do som ambiente.

Baseado na arquitetura em questão, outros protótipos poderão ser desenvolvidos, tais como: utilização dos recursos de reconhecimento de voz para manipulação de objetos VRML (*Virtual Reality Modeling Language*), criação de jogos com interfaces multimodais, reconhecimentos de movimentos para manipulação de objetos ou cenas etc.

5. Conclusão

O trabalho descrito neste artigo trata de questões teóricas e práticas relacionadas com a utilização de interfaces multimodais em sistemas de realidade aumentada. Nesse sentido, foi proposto uma arquitetura genérica visando padronizar e facilitar a incorporação de variados dispositivos de interface no sistema ARToolkit, denominada ARToolKit Multimodal. A implementação de um protótipo dessa arquitetura, e os correspondentes testes, permitiu que se chegasse a algumas conclusões iniciais. A primeira confirma que o reconhecimento de voz, por ser uma forma de interação bastante natural e eficiente, deve ser melhor explorada em sistemas avançados de computação. Apesar de não ser perfeito, o nível de evolução dessa tecnologia já permite a sua utilização em sistemas que exigem um grau de robustez elevado. Por outro lado, as interfaces multimodais, nas suas mais variadas formas e configurações, não combinam necessariamente com todo tipo de aplicação. é importante que se leve em consideração qual é o objetivo principal da aplicação, e quais os meios mais eficientes e interessantes para torná-la operacionalmente eficiente. Outras observações a serem consideradas dizem respeito a aceitação de uma interface multimodal, a qual é dependente das preferências pessoais de quem a está utilizando, e também com a sua facilidade de adaptação. De qualquer forma, a exploração de novas possibilidades oferecidas pelo uso de interfaces multimodais em sistemas de realidade aumentada mostrou-se bastante interessante, e com alto potencial para alavancar o desenvolvimento de novas classes de aplicações.

6. Referências

[1] ARToolKit. http://www.hitl.washington.edu/artoolkit/, Mar 2008.

[2] A. Berger. Embedded Systems Design: An Introduction to Process, Tools, and Techniques. CPM Books, USA, 2002.

[3] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *IWAR '99: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, Washington, DC, USA, 1999. IEEE Computer Society.

[4] C. Kirner, R. Tori, and R. Siscoutto. Fundamentos e Tecnologia de Realidade Virtual e Aumentada. *Sociedade Brasileira de Computação, 2006. Livro do Pré-Simpósio, VIII Symposium on Virtual Reality*.

[5] M. F. McTear. Spoken dialogue technology: enabling the conversational user interface. *ACM Computing Surveys*, 34(1):90–169, 2002.

[6] S. Oviatt. Advances in robust multimodal interface design. *IEEE Comput. Graph. Appl.*, 23(5):62–68, 2003.

Uma Metodologia para o Desenvolvimento de Aplicações de Realidade Aumentada em Telefones Celulares Utilizando Dispositivos de Sensoriamento

Juan Carlos Zuñiga, Sergio Takeo Kofuji
Programa de Pós-Graduação em Engenharia Elétrica - Escola Politécnica da USP
Grupo de Computação Pervasiva e Alto Desempenho
São Paulo - SP - Brasil
jcz30@pad.lsi.usp.br, sergio.kofuji@poli.usp.br

Abstract

New Handheld AR applications use a mobile phone camera, markers and wireless sensor networks (inserted in the environment) as elements for the capture and data entry into the AR system. These applications show interesting characteristics for: sensing, visualization and interaction in ambient intelligence (AmI). The lack documentation and complexity development these applications produce the need for methodologies and tools for a faster and simple development. This work presents a methodology for development and design of Handheld AR applications which use mobile phones and wireless sensor networks.

Keywords: *Handheld Augmented Reality, Wireless Sensor Networks (WSN), Mobile Phones.*

1. Introdução

O uso de dispositivos móveis ou *handheld devices* na implementação de um sistema de Realidade Aumentada (RA) é definido como *Handheld Augmented Reality* ou simplesmente *Handheld AR* [8]. Aplicações do tipo *Handheld AR* se desenvolveram e evoluíram nos últimos anos em muitas áreas de aplicação como: sistemas de informação, navegação, educação e turismo. Estas aplicações utilizam uma câmera integrada no dispositivo móvel e marcadores inseridos no ambiente, como elementos para o reconhecimento de padrões e ingresso de dados para o sistema de RA, respectivamente. As informações geradas nestas aplicações são apresentadas como imagens virtuais nas telas dos dispositivos móveis.

Já aplicações do tipo *Handheld AR* mais complexas, utilizam sensores como *RFID tags* [2] e Redes de Sensores Sem Fio (RSSF) [7], como fontes de entrada de dados junto ao uso da câmera e dos marcadores. A aplicação apresentada em [7] utiliza o telefone celular como uma interface que permite ao usuário visualizar informações do ambiente que não lhes são perceptíveis a simples vista, como informações da umidade do ambiente, o que apresenta características interessantes para áreas de aplicação como: sensoriamento, visualização e interação em ambientes inteligentes. Porém, a carência de documentação e a complexidade no desenvolvimento de aplicações deste tipo [6], gera a necessidade de metodologias e ferramentas que ajudem a um desenvolvimento rápido, simples e que permitam apresentar as informações geradas pelos sensores como objetos virtuais de forma intuitiva e em função do contexto [1].

Portanto, o objetivo deste artigo é apresentar uma "metodologia" para o projeto e desenvolvimento de aplicações do tipo *Handheld AR* que utilizam telefones celulares e sensores com interfaces de comunicação sem fio. Nossa metodologia abstrai detalhes de implementação relativos à aquisição dos dados gerados pelos sensores, a modelagem das preferências do usuário em um perfil (para prover informações relevantes e modos de interação intuitivos) e a disponibilização destas informações como objetos virtuais, os quais são sobrepostos em um cenário real utilizando o telefone celular como interface de interação com o usuário.

2. *Handheld AR*

A *Handheld AR* é um subconjunto da RA móvel [8] e está baseada no uso de dispositivos móveis do tipo *Handheld* como: telefones celulares, *SmartPhones* e *Personal Digital Assistants* (PDAs). Independentemente do dispositivo móvel a utilizar, uma aplicação do tipo *Handheld AR* passa por um fluxo de processamento composto pela:

2.1 Captura de vídeo

A captura de vídeo depende da plataforma do dispositivo móvel. Na plataforma Windows Mobile 5.0, o DirectShow permite o acesso as câmeras dos Smartphones, Pocket PC e PDAs através da Camera Capture API. Já na grande maioria dos telefones celulares, a captura de vídeo é realizada utilizando APIs específicas do sistema operacional instalado no

dispositivo [3]. Por exemplo: para a plataforma Symbian é utilizada a biblioteca ECam [4]. Outras plataformas utilizam SDKs proprietárias dos fabricantes das câmeras. Finalmente, a plataforma Java provê a Mobile Media API (MMAPI) ou JSR 135, que é um pacote opcional do J2ME destinado a fornecer funcionalidades de multimídia aos aplicativos.

2.2 Rastreamento de marcadores

O *ARToolKitPlus* é uma biblioteca para aplicações de *RA open source* baseada no *ARToolKit*, projetado exclusivamente para a detecção de marcadores e não oferece suporte para a captura de vídeo ou renderização de objetos virtuais. O *ARToolKitPlus* é utilizado para as plataformas Windows Mobile 5.0 (*Smartphones, Pocket PC* e *PDAs*) e consoles de jogos como o Gizmondo. Para telefones celulares que utilizam o sistema operacional *Symbian*, existe uma versão do *ARToolKit* para esta plataforma que é chamada de *ARToolKit for Symbian*. Outros sistemas de reconhecimento de marcadores para dispositivos móveis estão disponíveis no mercado. Mas, a finalidade destes sistemas não tem como objetivo a implementação de objetos virtuais ou aplicações do tipo *Handheld AR*. Alguns destes sistemas são: *QR Code, BeeTagg, ShotCode* e *Semacode*.

2.3 Renderização de Objetos Virtuais

O rápido desenvolvimento dos dispositivos móveis vem criando uma demanda por bibliotecas que tornem possível o desenvolvimento de aplicações que usam recursos gráficos 3D. Nesse contexto, a biblioteca mais popular para essa finalidade é o OpenGL ES, direcionada para sistemas embarcados. Outra opção é o *Klimt*, que cobre algumas das funcionalidades não suportadas por OpenGL ES. O *Klimt* também possui algumas otimizações para a plataforma *Pocket PC*. Já para a plataforma Windows Mobile 5.0 tem se o Direct3D Mobile [3]. Finalmente, a plataforma Java provê uma API para renderização 3D (JSR 184) chamada *Mobile 3D Graphics API for J2ME* (M3G). O M3G é uma biblioteca orientada a objetos. A desvantagem das aplicações feitas em Java é o baixo desempenho. Entretanto, o projeto de telefones celulares com GPU (*Graphics Processing Unit*) integrada, apresenta expectativas na melhora do desempenho para esta tecnologia.

3. Metodologia proposta

O objetivo deste trabalho é desenvolver uma "Metodologia para o Desenvolvimento de Aplicações de Realidade Aumentada em Telefones Celulares", a qual permitirá ajudar e facilitar o projeto e desenvolvimento de aplicações do tipo *Handheld AR* que utilizam telefones celulares e sensores com interfaces de comunicação sem fio. Para atingir este objetivo, este artigo foi desenvolvido em quatro etapas diretamente relacionadas: o estudo, levantamento e definição dos processos que se deve seguir para desenvolver uma aplicação do tipo *Handheld AR*; a modelagem conceitual dos dados do ambiente (gerados pelos sensores) e do usuário (perfil de preferências do usuário); o projeto de implementação e desenvolvimento dos módulos (bibliotecas de software) e serviços para processar e apresentar as informações geradas pela aplicação como objetos virtuais; e a aplicação e avaliação da metodologia proposta em um estudo de caso.

3.1. Arquitetura

Uma aplicação do tipo *Handheld AR* tradicional, propriamente dito, não precisa de suporte para aquisição de dados de sensores. Já aplicações como a apresentada em [7] precisa deste tipo de suporte. Porém, este trabalho não apresenta uma arquitetura clara para o suporte dos dados gerados pelos sensores, sendo a nossa arquitetura proposta uma contribuição para este tipo de aplicações. A arquitetura proposta é apresentada na figura 1 e contempla: a aquisição e processamento de informações obtidas por sensores com interfaces de comunicação sem fio e câmeras embutidas nos dispositivos móveis, o reconhecimento de marcadores e a renderização de objetos virtuais.

A figura 2 apresenta um cenário geral que descreve os ambientes, dispositivos e modos de interação nos quais a metodologia e arquitetura proposta podem ser aplicados. Este cenário esta composto por três perspectivas: ambiente, dispositivos móveis e usuário. A perspectiva do ambiente descreve os lugares, serviços e tecnologias disponíveis para este tipo de aplicações. Já a perspectiva dos dispositivos móveis descreve os tipos de dispositivos contemplados pela arquitetura proposta, em função de características de hardware, software e design. Finalmente, a perspectiva do usuário apresenta os tipos de interfaces a serem utilizadas para facilitar e melhorar a interação, tendo em consideração as preferências e necessidades do usuário.

3.2. Fluxo de Desenvolvimento

Primeiro, determina-se os dados que desejam ser coletados e analisa-se o tipo de sensor mais apropriado, assim como sua alocação no ambiente de forma não intrusiva. Segundo, estabelece-se uma conexão direta entre os sensores e o dispositivo móvel (utilizando uma interface de comunicação sem fio, como por exemplo: Bluetooth, Zig-Bee, etc.) para a transmissão dos dados.

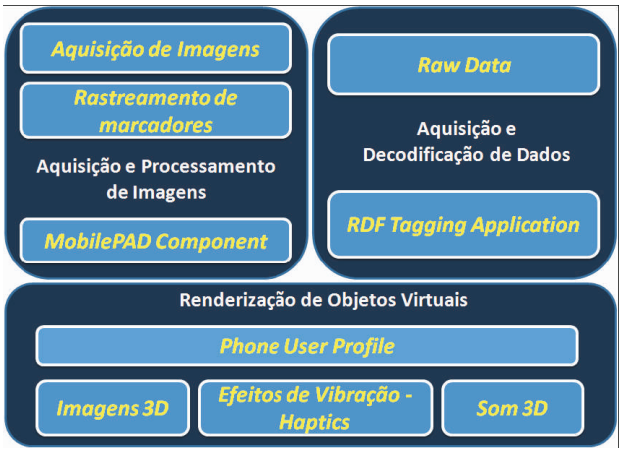


Figura 1. Arquitetura Proposta.

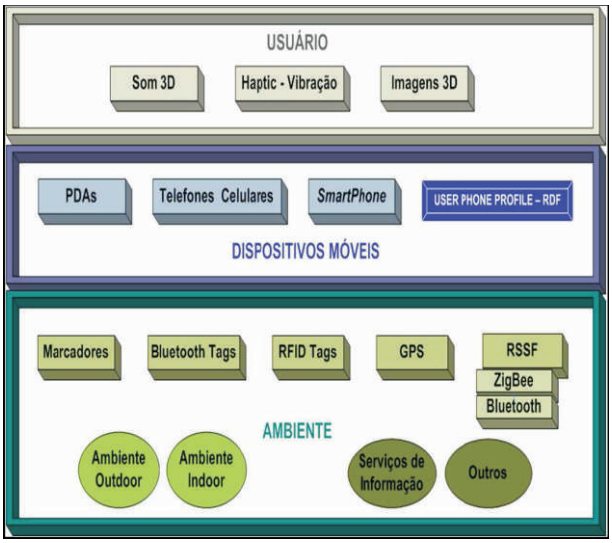


Figura 2. Cenário Geral.

Terceiro, os dados gerados pelos sensores chegam ao dispositivo móvel e precisam ser processados e gerenciados, para facilitar estes processos são utilizadas técnicas da web semântica [9]. Quarto, os dados processados geram informações que são apresentados como objetos virtuais, utilizando as preferências previamente modeladas no perfil do usuário, apresentando-se só informações relevantes através do modo de interação mais adequado. Quinto, se for necessário, utilizando a câmera do dispositivo móvel, descobre-se os marcadores (inseridos no ambiente) e insere-se as imagens virtuais associadas a estes marcadores.

3.3. Estudo de Caso

Por exemplo, um ambiente industrial composto por máquinas, fornos industriais e outros equipamentos que podem atingir temperaturas que possam ferir as pessoas que trabalham neste ambiente. O uso de uma aplicação do tipo *Handheld AR* para evitar acidentes apresenta-se como uma solução simples e prática. O usuário ingressa no ambiente industrial levando no seu bolso um telefone celular (com a aplicação *Handheld AR* instalada), os sensores de temperatura instalados no ambiente informam a temperatura dos fornos e máquinas. Se alguma destas temperaturas representa um risco para o usuário, o sistema executa um efeito de vibração ou som (*stream* de áudio) predeterminado em função do perfil do usuário para informar esta situação. Logo, o usuário pode utilizar a câmera do telefone celular para descobrir os marcadores (inseridos no ambiente) e visualizar maiores informações do que está acontecendo. Para desenvolver este estudo de caso, foi utilizada a plataforma Java para a captura de vídeo e renderização de imagens virtuais, e a ferramenta Semacode[1] para a criação dos marcadores. Como parte da implementação deste trabalho foram desenvolvidas bibliotecas de software encarregadas da aquisição e processamento dos dados gerados pelos sensores, criação dos efeitos de vibração e streams de áudio, a modelagem das preferências do usuário (perfil do usuário), e um componente de software que integra a captura do vídeo, reconhecimento do marcador e a renderização e apresentação dos objetos virtuais. Nota-se que o estudo de caso ainda está em desenvolvimento, fato pelo qual detalhes técnicos das bibliotecas desenvolvidas não são apresentadas no presente texto.

4. Trabalhos Relacionados

Projetos que utilizam sensores com interfaces de comunicação sem fio como RFID tags [2], RSSF [7] e GPS [5], como fonte de entrada de dados na implementação de aplicações do tipo Handheld AR em telefones celulares são ponto de partida do presente trabalho. Porém, a complexidade no desenvolvimento destas aplicações requer de metodologias e ferramentas que facilitem o seu desenvolvimento, sendo este o objetivo de nossa proposta. A geração de um perfil que descreve as preferências do usuário é uma contribuição do presente trabalho por permitir prover informações personalizadas e intuitivas em função do contexto como objetos virtuais (imagens virtuais, efeitos de vibração e reprodução de sons).

5. Conclusões

Com o crescimento no desenvolvimento de aplicações de sensoriamento e interação em ambientes inteligentes, estabelece-se a viabilidade do uso de dispositivos móveis, em especial telefones celulares, para a coleta, processamento e visualização de dados. Portanto, uma metodologia clara e bem definida ajuda e facilita um rápido e simples desenvolvimento deste tipo

[1] Semacode: http://www.semacode.org/

aplicações. O uso de técnicas de RA para a representação das informações geradas pelos sensores (instalados em diversos ambientes), permite incrementar o grau de percepção dos usuários, baseado nas potencialidades do sistema visual humano (imagens virtuais), sentido do tato (efeitos de vibração) e sistema auditivo (reprodução de sons) para analisar, perceber e entender informações. Além disso, a modelagem das preferências do usuário em um perfil, permite prover informações relevantes e modos de interação de forma intuitiva para os usuários.

6. Trabalhos futuros

•Desenvolver estudos de caso de sensoriamento e interação para diversos ambientes (casas, ambientes industriais, veículos, etc.) utilizando RFID tags, Bluetooth tags, GPS e sensores integrados (embutidos) nos telefones celulares.

•Aperfeiçoar as bibliotecas desenvolvidas para aproveitar recursos das placas de aceleração gráfica (GPU) embutidas nos telefones celulares, para incrementar o desempenho das aplicações.

7. Referências

[1] M. Billinghurst and A. Henrysson. Research directions in handheld ar. *The International Journal of Virtual Reality, 5(2)*: 51–58, 2006.

[2] G. Broll, S. Siorpaes, E. Rukzio, M. Paolucci, J. Hamard, M. Wagner, and A. Schmidt. Supporting service interaction in the real world. PERMID'06: Pervasive Mobile Interaction Devices, 2006.

[3] J. P. S. do Monte Lima. Um framework de realidade aumentada para o desenvolvimento de aplicações portáveis para a plataforma pocket pc. *Universidade Federal de Pernambuco (UFPE) - Centro de Informática (CIn)*, 2007.

[4] R. Harrison. Symbian OS C++ for Mobile Phones. *John Willey and Sons*, 2004.

[5] M. Kähäri and D. Murphy. Mara - sensor based augmented reality system for mobile imaging device. *5th IEEE and ACM International Symposium on Mixed and Augmented Reality - Ismar 06*, 2006.

[6] S. G. Neto, T. Farias, V. Teichrieb, and J. Kelner. Criação de aplicações de realidade aumentada em dispositivos móveis baseados em Symbian OS. *IIIWorkshop de Realidade Aumentada*, 2006.

[7] M. Rauhala, A.-S. Gunnarsson, and A. Henrysson. A novel interface to sensor networks using handheld augmented reality. MobileHCI'06: *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, 2006.

[8] D. Wagner and D. Schmalstieg. First steps towards handheld augmented reality. ISWC'03: *Proceedings of the 7th IEEE International Symposium on Wearable Computers*, page 127, 2003.

[9] J. C. Zuñiga, I. Roca, and S. T. Kofuji. Modelagem de dados e sensores para aplicações em intra-vehicle wireless sensor networks. *I Workshop on Pervasive and Ubiquitous Computing (SBAC-PAD)*, 2007.

O Uso de Realidade Aumentada em Sistemas de Manutenção Inteligente

Danúbia Espíndola, Carlos Eduardo
Universidade Federal do Rio Grande do Sul – UFRGS
dmtdbe_furg@yahoo.com.br; cpereira@ece.ufrgs.br

Resumo

Este trabalho propõe uma metodologia para o desenvolvimento de um sistema de Manutenção Inteligente integrado a recursos de Realidade Aumentada (RA). São apresentados conceitos, possíveis métodos para implantação desta metodologia, bem como, soluções em padronização para este propósito. Por fim, são discutidas conclusões e perspectivas futuras.

1. Introdução

A importância da manutenção nas indústrias vem aumentando consideravelmente nas últimas décadas. Isto ocorre devido à alta competitividade que acerca o mercado em busca de qualidade, preço e prazo na produção, fatores estes cruciais em termos de disponibilidade, confiabilidade e segurança de equipamentos e máquinas de produção[6].

Tendo em mente que as tarefas de manutenção necessitam de segurança tanto no procedimento de manutenção, quanto para a própria integridade física do usuário, a Realidade Aumentada surge como um meio de proporcionar ao utilizador uma interação segura, sem necessidade de treinamento real, uma vez que, esta pode trazer para o ambiente real, objetos virtuais, incrementando e aumentando a visão que se tem do mundo real[16].

Além disso, a possibilidade de manuseio desses objetos com as próprias mãos possibilita uma interação atrativa e motivadora com o ambiente. Entretanto, para que os objetos virtuais façam parte do ambiente real e sejam manuseados, devem-se utilizar softwares com capacidade de visão do ambiente real e de posicionamento dos objetos virtuais. Também são necessários dispositivos tecnológicos de interação, tais como, luvas, capacetes ou óculos apropriados para este propósito.

Sendo assim, técnicas tradicionais de manutenção na indústria, tais como, substituição de componentes e/ou reparos realizados apenas no momento em que ocorre a parada, já não são mais suficientes. A busca pelo aprimoramento efetivo do processo de produção, tornou-se uma constante na corrida pelos requisitos fundamentais de custo, tempo e qualidade do produto[20][3].

Tem-se então o conceito conhecido hoje como manutenção inteligente, onde através de técnicas de processamento de sinais é possível prever o comportamento futuro de máquinas e equipamentos, evitando assim paradas e falhas indesejáveis. A manutenção inteligente, busca por soluções embarcadas em hardware e software para a avaliação contínua do nível de degradação, previsão de desempenho e diagnóstico de funcionamento[17][9].

Neste contexto, ferramentas que agreguem aos sistemas de Manutenção Inteligente, recursos de Realidade Aumentada, conferem uma significativa contribuição tanto ao estado da arte, quanto ao meio industrial.

2. Manutenção inteligente e Realidade Aumentada

A Manutenção Inteligente visa permitir uma evolução dos sistemas tradicionais de manutenção corretiva/preventiva para um sistema preditivo. Sistemas preditivos baseiam-se no estado e na condição de uso dos equipamentos.

A manutenção corretiva consiste em substituir ou reparar a peça no instante da falha, isto implica em riscos aos funcionários e paradas inesperadas. A preventiva busca evitar falhas executando manutenções em intervalos de tempo fixo e pré-determinados, porém, podem ocorrer paradas constantes e desnecessárias na linha de produção, bem como, a substituição de peças em boas condições operacionais. A preditiva visa prever quais peças irão falhar através do monitoramento constante das condições de uso dos equipamentos.

De posse dos dados de condição e utilizando-se conceitos de instrumentação, aplicam-se técnicas de processamento de sinais e modelos de degradação que

podem predizer qual e quando o componente de uma máquina provavelmente irá falhar, reduzindo drasticamente o tempo de parada e permitindo a otimização dos processos de manutenção.

A Realidade Aumentada irá unir objetos virtuais com o cenário real de manutenção, produzindo um ambiente único, sobreposto ao ambiente físico. Desta forma a análise, a interação com gráficos e a exploração de aspectos cognitivos, relatados com a compreensão da informação, facilitam as tarefas de manutenção e a tomada de decisão.

Rotinas de manutenção e diagnóstico de erro podem ser melhoradas através da utilização desta tecnologia. Pretende-se, então, desenvolver uma metodologia para implantação de um sistema que possa indicar as instruções de manutenção e fornecer os resultados de diagnóstico através de recursos de RA[14].

Entre os principais desafios na implantação de técnicas de RA, verificou-se a dificuldade na capacidade de sincronizar e alinhar num mesmo sistema de coordenadas, o movimento da câmera, o ambiente do utilizador e os objetos virtuais inseridos nesse ambiente. Ou seja, o alinhamento e a sincronização entre o real e o imaginário.

Outro aspecto, importante, é quanto ao dispositivo de apresentação que podem ser de quatro tipos: baseados em monitor (*Window on the World ou Fish Tank*), baseados em HMD - onde é acoplada uma câmera de vídeo (*Video See Through*), baseados em HMD de lentes semireflexivas (*Optical See Through*) e baseados em projetores - apresentam os objetos virtuais diretamente no mundo real.

Descrito a contextualização do tema, a próxima seção apresenta uma metodologia para implantação de um sistema de manutenção inteligente que integre recursos de RA. Uma pesquisa mais profunda no que tange as duas áreas pode ser obtida nos trabalhos [8][12][2][15].

3. Metodologia

Num primeiro momento, propõe-se uma metodologia para o desenvolvimento de um sistema que monitore um equipamento através de técnicas de manutenção inteligente. Este sistema deverá enviar informações de diagnóstico ao ambiente de RA. A metodologia é descrita conforme a Figura 1.

Nesta figura, os módulos são representados por retângulos e a integração destes através de setas. As setas indicam a necessidade de implementação de algoritmos de integração para soluções descritas na seção 4.

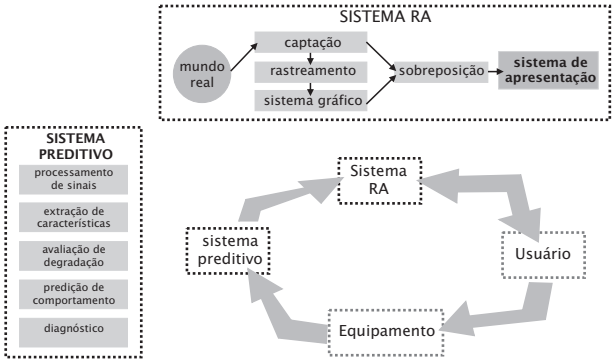


Figura 1. Metodologia proposta.

O módulo *Sistema Preditivo*, é responsável pelo monitoramento do equipamento crítico passível de manutenção. Tendo identificado os componentes críticos, determina-se, então, se é necessário à implantação de sensores e quais tipos de sensores são adequados ao monitoramento do equipamento. O sensor irá depender do tipo de sinal que se pretende analisar, como por exemplo: temperatura, pressão, corrente, torque, tensão entre outros.

De posse dos sinais oriundos dos sensores, passa-se então à etapa de pré-processamento. Nesta, os dados adquiridos devem ser convertidos a um formato específico. Além disso, a correção e alteração dos dados podem eliminar informações desnecessárias ao sistema. Pode-se ainda amostrar e normalizar estes dados para, enfim, passar-se a etapa de processamento de sinais.

Resumidamente, o módulo *Sistema Preditivo* pode ser dividido em cinco etapas: a etapa de processamento de sinais, extração de características, avaliação da degradação, predição de comportamento e diagnóstico. A Tabela 1 apresenta um panorama das técnicas aplicadas a cada uma destas etapas. Essas técnicas são descritas detalhadamente em [5][10].

Etapa	Técnicas
Processamento de sinais	Transformada de Fourier Transformada de Wavelet
Extração de características	Freqüências Fundamentais Análise Tempo-Freqüência Análise de Pacotes Wavelet
Avaliação da degradação	Regressão Logística Reconhecimento Estatístico de Padrões Mapas Auto-Organizaveis
Predição do comportamento	ARMA Redes Neurais Matriz de Predição Lógica Fuzzy
Diagnóstico	Modelo de Markov Redes Baysianas

Tabela 1. Técnicas utilizadas nas etapas da manutenção inteligente.

O módulo *Sistema RA* foi dividido em cinco etapas: Captação, Rastreamento, Sistema gráfico, Sobreposição e Sistema de apresentação. A Captação é responsável por capturar a imagem do mundo real e executar a projeção do mundo 3D em um plano 2D. Pode-se utilizar como dispositivo de captura: câmeras de vídeo e webcams.

A etapa de Rastreamento é responsável por obter a orientação e a posição dos objetos da cena real para que se possam manipular elementos tridimensionais[1]. Com base nesse posicionamento, o Sistema Gráfico se encarrega de gerar a imagem virtual onde cada objeto virtual é modelado no sistema de coordenadas.

O Sistema Gráfico necessita de informação espacial sobre a imagem do mundo real que permita controlar a câmera virtual usada para gerar a imagem dos objetos virtuais. A imagem dos objetos virtuais, captada pela câmera virtual é, finalmente, misturada com a imagem do mundo real pela etapa de Sobreposição para formar a imagem em Realidade Aumentada. Finalmente, o ambiente de RA é apresentado através de um dos dispositivos de apresentação citados na seção anterior.

Com base nesta metodologia, são apresentadas na próxima seção, ferramentas encontradas para estes propósitos, bem como, indicado possíveis soluções para a implementação desta proposta.

4. Ferramentas e soluções

Sendo a proposta, uma integração entre Manutenção Preditiva e Realidade Aumentada, as soluções encontradas na bibliografia que contemplem ambos os temas são raras. Entretanto, soluções individuais a cada um dos módulos são diversas, o que evidencia a importância de uma metodologia que agregue ambas as áreas.

Entre as soluções pesquisadas no âmbito da manutenção inteligente, foram encontrados sistemas, plataformas e padrões desenvolvidos para a indústria, a fim de, facilitar e fornecer os recursos necessários a implantação de sistemas de manutenção inteligente.

Os padrões mais utilizados na indústria são: MIMOSA (*Machinery Information Management Open System Alliance*), OSA-CBM (*Open System Architecture for Condition Based Maintenance*); OSA-EAI (*Open System Architecture for Condition-Based Maintenance*). As plataformas PROMISE, DYNAMITE, TELMA e WATCHDOG AGENT apresentaram importância relevante entre as soluções encontradas.

Entre os sistemas pesquisados citam-se SIMAP, DATC, RIMFDS e IMS. Este último, utiliza o padrão OSA-CBM e possui um componente prognóstico núcleo, chamado *WatchDog Agent*[5]. O IMS foi escolhido como solução ao módulo sistema preditivo desta metodologia. Maiores detalhes podem ser analisados em [5].

O sistema IMS, consiste em algoritmos computacionais de prognóstico embarcados e um conjunto de ferramentas em software para predizer a degradação de dispositivos e sistemas. Este cálculo de degradação é baseado em leituras de múltiplos sensores que medem as propriedades críticas do processo/máquina. Este sistema possui, ainda, rotinas desenvolvidas para cada uma das etapas do módulo sistema preditivo, proposto na metodologia. Uma vez, adquirido o sistema, pode-se agregar modificações nos algoritmos de cada uma das etapas envolvidas.

As soluções encontradas na área de Realidade Aumentada são muitas. Entre os softwares utilizados nesta área citam-se: VRML, Java, ARToolkit entre outros[19][4]. Os hardwares de Realidade Aumentada dependem da aplicação, caso necessite-se interação com as mãos necessita-se de luvas. Para maior imersão utilizam-se HMD, capacetes ou óculos estereoscópicos. O ambiente pode ser apresentado em telas de projeção comuns, monitores ou capacetes. Quando existe movimento no ambiente real são necessários sensores de rastreamento.

Neste estudo, optou-se pela utilização dos seguintes hardwares: um HMD, uma luva, uma câmera, um computador e uma válvula como equipamento a ser monitorado. O HMD para apresentação dos dados de manutenção. A luva como dispositivo de interação para treinamento de manutenção em equipamento virtual. A câmera para atualização de movimento do usuário. Em software, selecionou-se o ARToolKit para o desenvolvimento dos códigos de sobreposição e criação do equipamento virtual (objeto virtual) e o WatchDog para implantação do sistema preditivo. A Figura 2 descreve o layout de hardware do experimento.

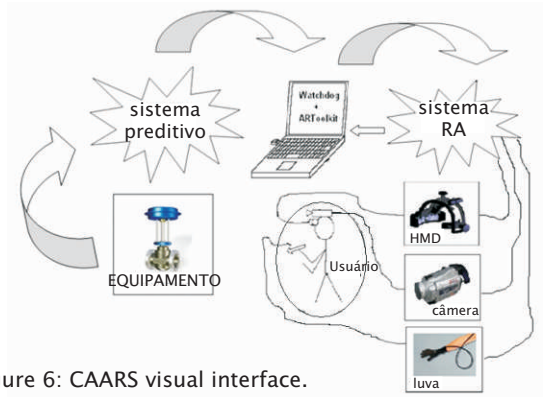


Figure 6: CAARS visual interface.

5. Conclusões e perspectivas

Este artigo descreveu, de forma sucinta, um projeto em andamento para integração de recursos de RA em um sistema de manutenção inteligente. Na sua primeira fase, foi desenvolvida uma metodologia como solução a esta proposta, onde o módulo Sistema Preditivo foi capaz de diagnosticar e interpretar as condições de funcionamento do equipamento.

Em uma segunda etapa, esses dados de diagnóstico, deverão ser transmitidos à interface de RA, onde serão apresentados as instruções de manutenção, bem como, gráficos de simulação virtual 3D. Estes dados irão auxiliar no processo de montagem e desmontagem virtual do equipamento através de uma luva.

Entretanto, esta metodologia é indicada a máquinas e equipamentos de manutenção complexa que geralmente não param nunca e quando isto ocorre implicam em grandes prejuízos por minuto de parada. Logo, é necessário definir quais os componentes críticos da planta, a fim de, definir a viabilidade em termos de custo da implantação desta solução.

Sendo assim, nem sempre é viável economicamente implementar o módulo *Sistema preditivo*, muitas vezes é preferível substituir a peça no instante da falha. Entretanto, o auxílio dos recursos de RA representado pelo módulo Sistema RA, é uma alternativa de baixo custo onde os ganhos de sua utilização representam uma vantagem competitiva em qualquer que seja o sistema de manutenção utilizado.

Por fim, é importante salientar ainda, que a indústria nuclear, *offshore*, aeronáutica e automobilística possui grande interesse nestas soluções. Recentes pesquisas podem ser observadas em [7][13][11][18].

6. Referências

[1] A. Comport et al., "A real-time tracker for markerless augmented reality", *Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR '03)*, (2003)

[2] A. Mathew, et al., "Reducing maintenance cost through effective prediction analysis and process integration" In: *Advances in Vibration Engineering 2006 5(2)*: pp.97-96.

[3] B. Schwald and B. Laval, "An Augmented Reality System for Training and Assistance to Maintenance in the Industrial Context", *Journal of WSCG*, Vol.11, WSCG'2003, (2003)

[4] C. Nakajima and N. Itho, "A Support System for Maintenance Training by Augmented Reality", *Proceedings of the 12th International Conference on Image Analysis and Processing (ICIAP'03)*, (2003)

[5] D. Djurdjanovic et al., "Watchdog Agent, an infotronics-based prognostics approach for product performance degradation assessment and prediction." *Adv Eng Inf 2003*;17(3–4):109–25.

[6] F. Peysson et al., "New Approach to Prognostic Systems Failures" In: *Proceedings of the 17th IFAC World Congress 2007*.

[7] H. Regenbrecht et al., "Augmented Reality Projects in the Automotive and Aerospace Industries", *Published by the IEEE Computer Society, IEEE*, (2005)

[8] J-B. Le´ ger., "A case study of remote diagnosis and e-maintenance information system." I*nvited keynote paper for IMS'2004 "International conference on intelligent maintenance systems,"* Arles, France, 2004.

[9] J-P. Li and G. Thompson, "Modelling of mechanical failures in a virtual reality design environment", In: *IEEE Annual Symposium Reliability and Maintainability,* 2003 pp:507 – 512.

[10] J. Lee and J. Ni, "Infotronics-based intelligent maintenance system and its impacts to closed-loop product life cycle systems." *Invited keynote paper for IMS'2004—International conference on intelligent maintenance systems*, Arles, France, 2004.

[11] J. Weidenhausen et al., "Lessons learned on the way to industrial augmented reality applications, a retrospective on ARVIKA", *Computers & Graphics, 27*, (2003), pp. 887–891.

[12] M. Haringer and H. Regenbrecht, "A pragmatic approach to Augmented Reality Authoring", *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR'02)*, (2002)

[13] M. Macchiarella and D. Vincenzi, "Augmented Reality In *A Learning Paradigm For Flight And Aerospace Maintenance Training", IEEE*, (2004)

[14] M. Roemer et al, "An overview of selected prognostic technologies with reference to an integrated PHM architecture." In: *Proceedings of the IEEE aerospace conference 2005*, Big Sky, United States, 2005.

[15] N. Navab, "Industrial Augmented Reality(IAR): Challenges in Design and Commercialization of Killer Apps", *Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR '03)*, (2003)

[16] N. Zenati et al, "Assistance to Maintenance in Industrid Process Using an Augmented Reality System", *International Conference on Industrial Technology (KIT), IEEE*, (2004)

[17] R. Behringer et al. "A distributed device diagnostics system utilizing augmented reality and 3D audio", *Computers & Graphics, 23*, (1999), pp. 821-825.

Augmented Reality (ISMAR '03), (2003)

[16] N. Zenati et al, "Assistance to Maintenance in Industrid Process Using an Augmented Reality System", *International Conference on Industrial Technology (KIT), IEEE*, (2004)

[17] R. Behringer et al. "A distributed device diagnostics system utilizing augmented reality and 3D audio", *Computers & Graphics, 23*, (1999), pp. 821-825.

[18] T. Pettersen et al., "Augmented Reality for Programming Industrial Robots", *Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR '03)*, (2003)

[19] W. Friedrich, "ARVIKA – Augmented Reality for Development, Production and Service", *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR'02)*, (2002)

[20] W. Lee and J. Park, "Augmented Foam: A Tangible Augmented Reality for Product Design", *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR'05)*, (2005)

Session 2: Distributed and Collaborative Systems

An Infrastructure to Generate Adaptive Virtual Environments with the Management of Client-Server Communication in Real Time

Aquino, M. S.[1,2], Souza, F. F.[2], Frery, A. C.[3], Souza, D. A. C. M.[4], Fujioka, R. C.[4], Vieira, M. M. S.[4]

[1]Departamento de Sistemas e Computação/UFCG – Campina Grande/PB - Brazil

[2]Centro de Informática/UFPE – Recife/PE - Brazil

[3]Instituto de Computação/UFAL – Maceió/AL - Brazil

[4]Curso de Ciências da Computação/UNIPÊ – João Pessoa/PB – Brazil

salerno@dsc.ufcg.edu.br; fdfd@cin.ufpe.br; acfrery@pesquisador.cnpq.br;

Abstract

The Virtual Environments evolution and the arising of new technologies for rendering and visualizing three-dimensional objects have contributed for the development of interfaces adapted to the users. These interfaces allow users navigate and interact with elements nearby of his or her realities. Techniques for the generation of Virtual Environments that adapt to the user's profile have been developed, incorporating procedures for tracking actions and changing the environment in light of this behavior. This paper presents the VEPersonal (Personalized Virtual Environment) system, an infrastructure for managing adaptive Virtual Environments, in real time, according to the user's profile. This work proposes an interface for client-server communication, for managing the actions of the user and updating the environment in real time. With this proposal, an adaptive Virtual Environment can be created according to the user's cognitive development, in a dynamic and personalized manner.

1. Introduction

A Virtual Environment (VE) is an interactive environment, composed of three-dimensional objects and generated in real time by a computer system. Its goal is to simulate a real environment or to build imaginary ones, allowing one or more users to interact through the visualization and manipulation of objects. Such an interaction increases the sense of presence in the environment to users [1].

The simplest interaction with VEs is navigation, i.e. the movement of users in three-dimensional space. This brings new resulting in viewpoints from the scene. In this case, there are no changes in the virtual environment, but only one exploratory tour. Interactions, leading to changes in virtual environments really occur when the user enters the virtual space, sees, explores, manipulates and triggers alterations to virtual objects [1].

The increase in the number of interactive virtual environments and diversity of applications focused on specific environments led to the generation of environments closer to reality.

Research conducted in this area is intended to reduce the problems on accessibility and usability for 3D interactive environments. Such researches have different approaches to certain issues, such as: what changes must be made and in what time; what information on the user is required, and what forms of content adaptation are needed.

Adaptation techniques have been developed to generate customized VE, built according to the preferences of each user, his or her level of knowledge and his or her needs. Some of such techniques are based on aiding to the user while browsing [2, 3, 4], and on the definition of priorities for presentation of content (products more accessed or queried at e-commerce, for example) [5, 6].

According to Chittaro and Ranon [7], the ability to adapt the content, structure and presentation of the environment in accordance with user's characteristics is increasingly seen as a key factor to boost their level of satisfaction. In a virtual shopping mall, for example, the ability to change the layout of stores as the interests of the user (impossible to be done in a real shopping mall, and little effective in a two-dimensional environment) can turn the application more friendly and attractive.

According to Santos [8], such an organization requires certain groupings according to some semantic criteria. For example, an environment to support distance learning, needs to group the contents spatially according with the areas of knowledge to which they belong.

However, despite the progress on VE adaptation, very little has been discussed as far as the presentation of content with different levels of information is concerned. This kind of applications may present an environment with a greater richness of details depending on the knowledge level of the user.

This type of customization can be used in applications that require different levels of information to be presented to different targeted publics. For example, in virtual environments devoted to distance learning, several students with different knowledge can connect remotely and access a VE specific to each profile. Also the simulation environments can generate different levels of complexity to a given situation according to the user's model.

One of the open problems in this kind of application is the adaptability in real time that is the ability of the system to modify the information in the virtual world during a session. Changes to VE are carried out only on the following user's interactions, described by the above-referred works.

In order to solve such limitations (adaptability in real time and different levels of information to different users), section 2 presents an infrastructure for creating and managing adaptive virtual environments that can be updated in real time. The main contribution of this paper, presented in section 3, is a client-server communication Web interface that allows the adaptation of virtual environments.

2. The VEPersonal System

The VEPersonal (Personalized Virtual Environment) is a system responsible for the creation and maintenance of adaptive virtual environments in which objects are accessed via Web for navigation and interaction. Such environments are modified in real time, according to the user's cognitive development [9].

The main beneficiaries of VEPersonal are those responsible for the development and construction of adaptive VEs. The infrastructure created by VEPersonal lets developers generate virtual environments composed of objects with different levels of complexity. Once defined the environments and the business rules for their operations, VEPersonal is able to manage the communication with the user and to update these environments, accordingly [10].

VEPersonal can support various applications such as environments for education or simulation, where the environment can evolve, while the user is performing tasks and acquiring better knowledge of the content.

Bearing in mind that virtual environments must be accessed via Web and used by users with different backgrounds and with different cognitive abilities, some assumptions were made for the configuration of the VEPersonal architecture. They include low-cost environment for use via Web; reduced rate of data transfer between VEPersonal and user's machine; reuse of objects of VE facilitating the construction of new virtual worlds; customization of the environment in accordance with the user profile and the development of his or her knowledge while browsing; and the VE updating in real time to make the environment more friendly and interactive.

In relation to the rate of data transfer, the VEPersonal system was built as a client-server application. Given the amount of information that needs to be manipulated (generation of virtual worlds, monitoring of the users actions and modification of such worlds), this type of application enables the reduction of load on the client machine, since all the more complex operations that demand the manipulation of data or many more processing time are executed on the server-side. The bandwidth is reduced because only the information expected by the running application, such as 3D objects of the user's actual environment and the user's actions, are transferred.

The use of the X3D language [11] eased objects reuse to build VEs. It enables the storage of 3D objects in a Database Management System (DBMS) with support to XML [12], and allows the transportation of such objects in a Web environment as well. With this structure, it's possible to build a framework able to recover only the objects necessary to the VE to be presented to the user.

The customization of virtual worlds could be achieved from the construction of the User's Model and the use of intelligent agents to update such a model. Agents are capable of evaluating the knowledge gained by the user during his or her interaction with the system, by using the information stored in the User Model. From such an evolution they determine the necessary changes to the environment, allowing adaptativity for virtual environments.

Finally, VE updating in real time is performed by an interface that allows objects to be inserted/removed to/from the virtual world without having to reload the whole environment. Thus, the virtual environment can be updated during the user interaction, while reducing the volume of data transferred by the Web.

One of the main components of VEPersonal is its architecture. It favors the management of dynamic virtual environments that are adapted to their users. This architecture facilitates the monitoring of the interaction and VE updating as the characteristics of the users and their knowledge evolve [13].

The architecture, shown in Figure 1, allows storing the characteristics of the users and their behavior in the

User Model (UM), and storing the history of the environment changes in the Environment Model (EM). Such information obtained from the interface through sensors that detect the actions undertaken by the user, are important to the system monitors progress during his or her interaction and creates objects to the virtual world adequately.

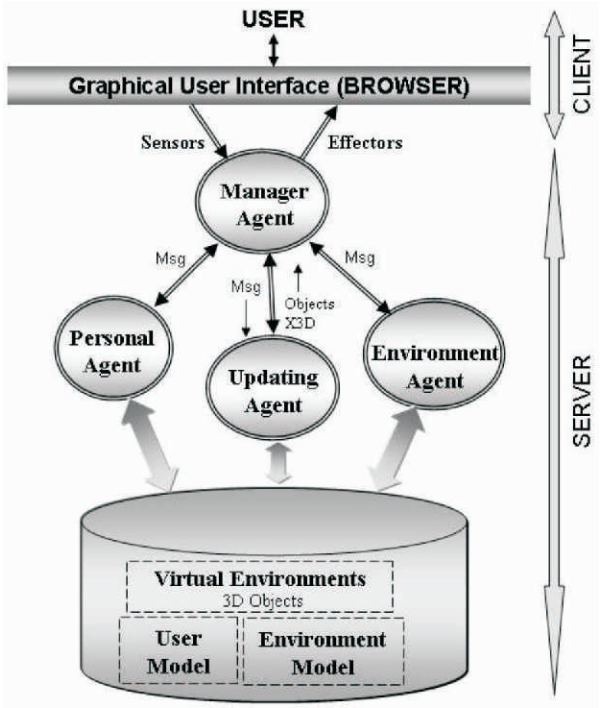


Figure 1. The VEPersonal architecture

The analysis of the information stored in the UM and the EM, and the update of objects in the world are performed by an agent society. Besides the agents proposed in Aquino et al. [13], a Manager Agent has been introduced to control the other agents and to facilitate communication with the interface. Thus, four agents are needed [14]:

• Manager Agent is responsible for monitoring the actions of the user and the current state of the virtual environment. It also determines the tasks to the other agents and coordinates communication between them;

• Personal Agent determines the profile of the user and updates the UM whenever any change of the profile is detected. It has an inference system in its knowledge base that evaluates user's cognitive capacity;

• Environment Agent verifies the current state of virtual environment and stores the changes in EM; and

• Updating Agent is responsible for the update of the virtual world based on the information provided by the other agents.

During the decision-making process, the Manager Agent consults the Environment Agent for the current structure of VE (stored in the EM) and also the Personal Agent to identify the profile of the user (stored in UM). This information is sent to the Updating Agent, which generates queries to the DBMS to recover objects to be used for updating the world. In possession of such objects, the Updating Agent sends them to the Manager Agent that generates the 3D structure of the virtual environment and sends it to the user interface.

The coordination and communication among agents are carried out with JADE - Java Agent DEvelopment framework [15]. It guarantees a standard integration in multiagents environments, which facilitates communication. Messages exchanges between agents and information gathering, necessary for updating the VE, are also facilitated. The Agent Communication Language used in JADE, specified by the Foundation for Intelligent Physical Agents – FIPA [16], is the SL (Semantic Language) [17].

3. Client-Server Communication

VEPersonal uses a Web environment for communicating with the user who is connected remotely. An interface for communication between the user and the system has been developed to achieve the synchronization of information in order to be a dynamic environment.

Due to the amount of information that must be stored, retrieved and processed in a VE, a client-server application becomes more appropriate, decreasing the burden on the client machine.

The interface task is to communicate the actions of the user to the Manager Agent, in addition to receiving objects to update the virtual world. This updating can be by sending a complete X3D environment, insertion of new objects (tags X3D) or removal of objects (by name) that exist in the VE. The interface should dispose means to aid the Manager Agent to carry out such operations.

The Xj3D browser [18] performs the visualization of virtual worlds. This browser is an open source project of the Web3D Consortium [19], developed in Java, which allows viewing X3D and VRML environments. Applications using Xj3D have been developed for simulation, training and education as well as for medical visualization, games and projects involving viewing sensors [20].

Besides that browser, the API Scene Access Interface

(SAI) [21] was used to establish communication between X3D environments and Java. This API allows the insertion, deletion and modification of the objects in the Xj3D browser in real time.

The integration of these features in the Web environment used the Java Web Start technology (JWS) [22] in order to enable the execution of Web applications, managing services and communications protocols. JWS is initiated automatically when it is done the first download of the VEPersonal application.

JWS stores the application locally, thus, all the subsequent initiations are nearly instantaneous, since all of the needed resources are already available.

On the client side, the user accesses the system via the browser and can perform actions such as registration and authentication. When he or she requests the generation of the 3D world, a Java application is initiated to monitor the user actions in the environment and to inform to the server about the execution of an action (e.g. click of the mouse or proximity to objects). This application is also responsible for receiving the virtual world to be loaded or objects to be included and/or excluded in/from such a world (Figure 2).

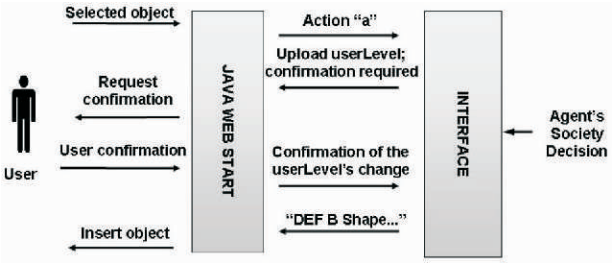


Figure 2. Client–server interface of VEPersonal using Java Web Start

On the server side, when an action is received from the user, the Manager Agent is responsible for determining what should be the actions to be performed. These actions are transformed into messages from the protocol and sent to the others agents. The society of agents takes the necessary decisions and returns to the interface the corresponding updating of the virtual world, if necessary.

The communication interface of VEPersonal was structured in a way that the client-server connection could be performed decoupled. Figure 3 presents the structure of the interface and the project's design patterns used in the communication between the client and the server.

On the client machine, the execution of the Xj3D browser needs various DLL (dynamic-link library) [23].

To make the application decoupled, that is, independent of the location of the library, it was built a Load function that locates and loads automatically to the client's machine the required libraries to the application. In this case, the application can run on any machine because it does not depend on external libraries.

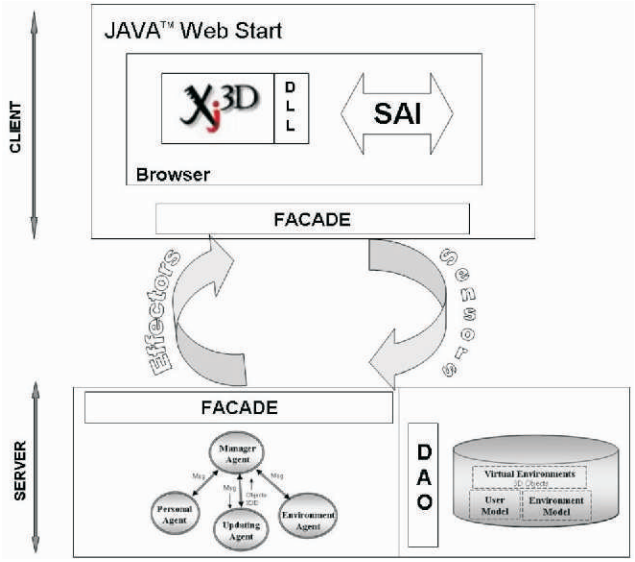


Figure 3. The VEPersonal's interface structure and the used design patterns

The Interface in VEPersonal is responsible for communicating the application (which is running in the browser) with the server. This communication occurs through sensors and effectors. The sensors notify the changes made in the browser, identifying the object and the type of action performed by the user. For the implementation of the sensors it has been defined a method (Listener), which uses the design pattern Observer. This pattern defines a dependence on one-to-many between objects, so that when an object changes its state, all their dependents are automatically notified and updated. In this case, the Listener is used to propagate to the server the changes occurring in the browser.

Effectors are responsible for sending of objects and virtual environments from the server to the browser. Both effectors and sensors use the framework LipeRMI [24] for data transfer between the client and server via Web.

LipeRMI re-implements RMI [25] so that the calls are made over the Internet, minimizing the use of bandwidth for communication (reducing the number of active connections). Once the client is already

connected, this connection should remain open throughout the session for data exchanging between client and server (or until the application allows the connection to be closed). Hence the server will never need to open a connection from server to client, since it may use the existing connection. Therefore, the client is not concerned with firewall control - solving the biggest RMI problem.

By using LipeRMI, communication is defined by events that activate the connections of the client and the server, checking when a connection is initiated and terminated.

The main events performed by the client during communication with the interface are: to create a connection when the user logs on; to send messages to the server to inform users actions; and to get messages sent by the server.

The server on its turn shall wait for a client message and, therefore, remains on waiting state performing the listening event. Once established a connection with the client, the server performs a sending or receiving event.

In order to abstract features of the browser and to decrease the coupling with the server, making the communication independent of the employed technology, the Facade design pattern was adapted. Facade is a structural pattern that aims at hiding the details of a process by creating a Facade class. The client, then, just calls a method of Facade class and this class is responsible for executing this process. A Facade was also developed for the server, specifying the patterns of communication with the interface.

The methods used by the interface to the Facade standard are: openEnvironment, setEnvironment, get ObjectList, addObject, removeObject, getX3dEnvironment, setUserProfile, addVEPersonalEventListener, removeVEpersonalEventListe-ner, exitEnvironment.

The Interface of VEPersonal performs client-server communication with high cohesion and low coupling due to the design patterns used. Such a police clearly allows the alteration or substitution of the employed technologies, whenever occur the launching of new technologies or evolution of the already existing.

4. Case Study

VEPersonal was developed for use in a Web environment. To have access to this environment, an HTML page was built having information on the project.

In this page, the user must fill out a form with personal data, preferences, interests and knowledge level. After this, he or she can authenticate his or her login and password to be admitted to the applications and their underlying Ves.

Our case study focus on an environment designed to teach physics to students of secondary schools. In this context, students can interact with experiments, access information about their contents and answer to tests on the studied contents. Each test is specific to the level of knowledge of the user.

The study environment presents a "foyer" (hall of entry) with three doors. Each one gives access to a room containing mechanical experiments on Kinematics, Dynamic or Static Mechanics (Figure 4).

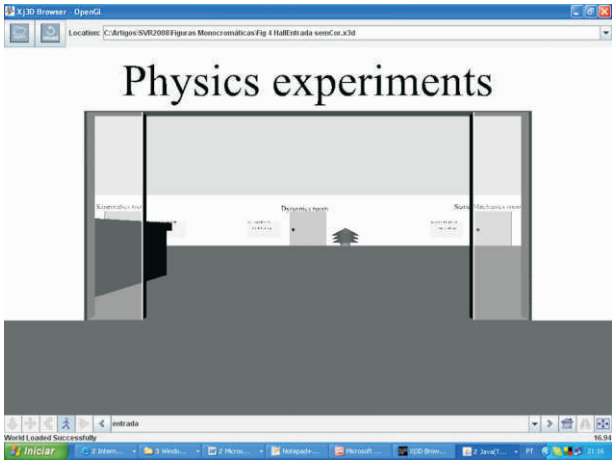


Figure 4. Hall of Entry of the application of Physics Experiments seen by the user

The student is guided to interact with each experiment, running a series of tasks proposed by the system. He or she can browse the rooms and leave them according to his or her convenience. The system is capable of storing the current state of the environment (in the Environment Model) and current information from the user (in the User Model). Such information will be recovered when the student returns to the environment.

In Figure 5 it is shown the student interaction with the Free Fall experiment. The objective of this experiment is to study the acceleration of gravity. It is composed of a structure called VerticalKit and a ball positioned at 1.60m high, plus buttons that allow interaction with the application. The student performs the experiment selecting the button EXECUTE. Right away, the sphere carries out a descending movement, under the effect of the gravity, until arriving to the base.

In this moment, the time spent during the fall is presented to the student (Figure 6). Such a student also

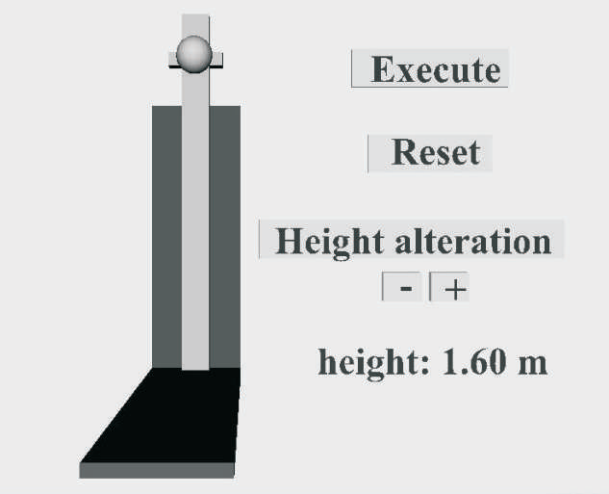


Figure 5. Beginning of the Free Fall experiment.

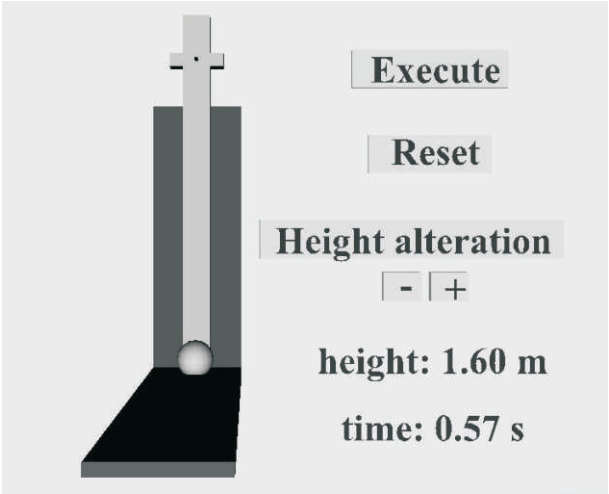


Figure 6. Experiment after execution. corresponding fall is shown.

For each experiment the student can consult the underlying theory and perform interactions. An evaluation test is presented according to the interactions carried out by the user. This test is specific for the knowledge level of the student (according to the profile stored in the User Model). Preliminary tests with user profile levels were accomplished in [10].

The test for a beginner student, presented in Figure 7, is composed of a question and three answer options. The insertion of the test in the environment is determined by agents that receive information from the interface about the actions of the student, for example, the number of times the experiment is executed and the consulted theory.

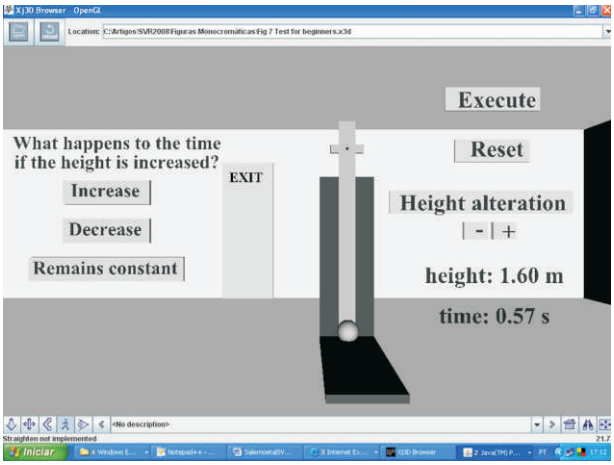


Figure 7. Test for beginners.

In case of an experienced student user, a more elaborate question (with bigger rank of difficulty) is presented (Figure 8).

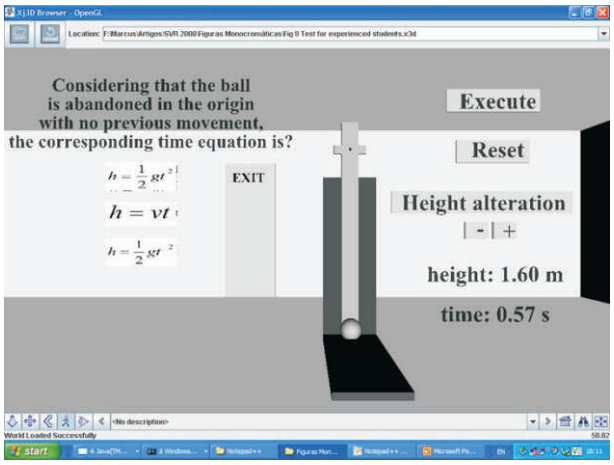


Figure 8. Test for experienced students.

In all rooms that the user will be interacting with an experiment, always will have a door (EXIT) where he or she can leave the environment at any time and to close the session. Also it is permitted to the user to return to the foyer by the same door that he or she entered in the experiment.

Another advantage for using the user profile in this application is the determination of access to other environments. For example, the access to a room that has experiments with the same concept, will only occurs to students that already reached an adequate level of knowledge. Figure 9 describes the vision of a beginner student that does not have permission to enter in another room (because he or she probably does not have the necessary level of knowledge).

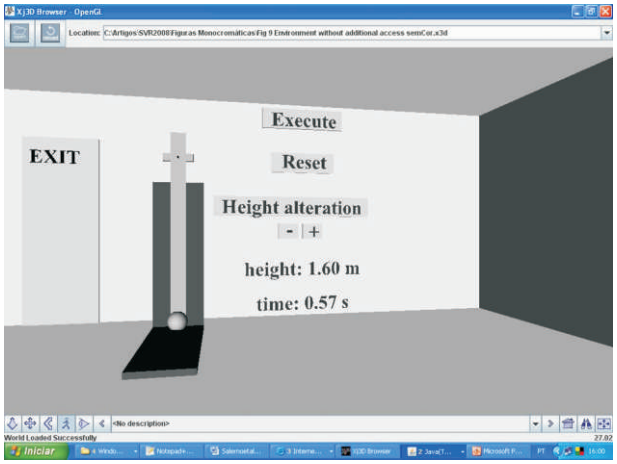


Figure 9. Environment seen by user without additional access.

On the other hand, Figure 10 shows a door to the right that permits the student to enter and to access information about the Newton Laws, for example (because he or she has appropriate knowledge for that). This example is the result of an adaptation to the characteristics of the user that are stored in the User Model.

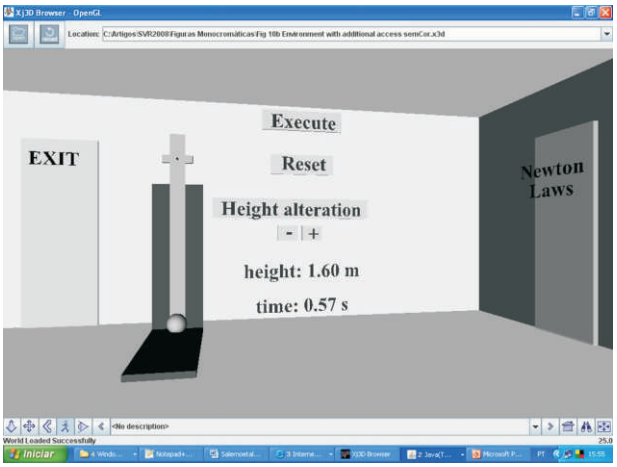


Figure 10. Environment seen by user with additional access.

5. VEPersonal Performance Analysis

The VEPersonal performance was measured taking into account the number of interactions that the user performs in each session and the time of response that the system takes to run an activity and/or to perform the necessary updating.

The interactions are to perform an experiment, to consult information, to answer questions, to enter into a new environment and to come back to an environment already visited. Such interactions can be conducted at any time and in any order, depending only on the user's motivation and his or her style of navigation. The response of the system comes after assessment of the user's action by agents and the recovery from the DBMS of objects that will compose the environment.

Two physics experiments compose the case study application: Free fall and Simple Pendulum. The tests were conducted using a computer with 2,4 Ghz Athlon XP processor and 512 MB of memory for the server. The client was a Notebook HP Pavillion ZE2000 with a 1,4 Ghz Intel Celeron M processor and 1GB of memory.

The current version of VEPersonal permits the access of a user at a time to the system. For the achievement of the tests, were registered initially ten users with their respective profiles. All of the entries were stored in the User Model.

When a user accesses VEPersonal, his or her login and password are verified and after this step, the initial environment foyer ("Hall of Entry") is loaded (see Figure 4). From this stage, the following values have been raised in each session to check the system ´s performance:

· Response time of the system to load a new environment;

· Response time of the system to load an environment already visited; and

· Response time of the system to update an environment due to the change of profile.

Automatic performance tests were generated to evaluate the time spent by VEPersonal to load the environments. It was used the JUnitPerf framework [26] that allows building objects that receive existing tests of the JUnit and add to them the performance evaluation. It was used the TimedTest performance test that runs a test and measures the elapsed time in this action.

The VEPersonal performance tests evaluates the total time expended by the project components altogether (database, agents and browser) to build the user environment.

The application supported the proposed changes, and managed satisfactorily the recovery of the objects from the DBMS, assembly of the environment and its sending to the browser. The values obtained in this assessment are presented in Table 1.

Table 1. Response time from the system to the environment updating.

User	Environment	Response Time to Load Environments (ms)		
		New Environment	Environment Visited Previously	Environment with user's profile change
Marcus	Hall of Entry	747	632	678
Marcus	Free Fall	712	581	673
Marcus	Pendulum	705	579	672
Salerno	Hall of Entry	742	627	642
Salerno	Free Fall	686	584	635
Salerno	Pendulum	685	584	634
Daniel	Hall of Entry	747	629	638
Daniel	Free Fall	703	578	642
Daniel	Pendulum	700	579	641
Abella	Hall of Entry	745	621	646
Abella	Free Fall	680	562	612
Abella	Pendulum	685	564	610
Rodrigo	Hall of Entry	732	628	657
Rodrigo	Free Fall	686	573	639
Rodrigo	Pendulum	684	569	641
Fujioka	Hall of Entry	758	634	642
Fujioka	Free Fall	688	557	628
Fujioka	Pendulum	684	559	629
Average Time		**709,39**	**591,11**	**642,17**

During the tests, it was verified that the average time of response to load a new environment was 709.39 milliseconds (ms). An environment that has been visited spends on average 591.11 ms to be bonded, and to update the environment due to the change of user profile has average time of 642.17 ms.

The tests were initially carried out to measure the performance of the server. As noted in Table 1, considering the preliminary results, it is appropriate to conclude that the interface has performed communication with server satisfactorily, indicating that the services were performed in an acceptable time. In this case, the tests have met the expectations projected for VEPersonal.

6. Conclusions

This work has presented an infrastructure for managing components of real-time adaptive VEs, called VEPersonal. The proposed system is aimed at solving the problems of content adaptation with different levels of detail to be displayed as the user's profile is being modified. It is seen on these systems that the user model and the existence of a knowledge base often represented by an agent, are important in building environments that can adapt to their users. This technique is important, especially when environments for teaching and simulation are considered. That is, there is a learning process during the interaction with the VE. Thus, the techniques of adaptability presented in this paper can increase the motivation and interactivity of the user, as well as to contribute to the development of his or her learning.

The main contribution of this work is the specification and implementation of VEPersonal, highlighting its client-server communication interface that is capable of managing virtual environments on the client machine, updating the VE in real time and generating several information levels according to the user profile. Section 5 shows that these goals were met.

This type of application needed solutions that could integrate the different technologies used, such as Java Web Start to allow access to the browser Xj3D, the protocol LipeRMI to control the transfer of data between the interface and agents, and the use of design patterns as the Listener and Facade, to allow for the standardization of communication. Such an integration has guaranteed the communication between server and the VE, enabling the agents to manage an application remotely without loss of both performance and information.

The user satisfaction evaluation is an ongoing work. Bearing in mind that the development of learning environments requires the assistance of specialists, we are developing an interdisciplinary strategy to tackle this issue. In this case, the interaction rules and user's evaluation in such environments, defined by the specialists, are included in the agents' knowledge base of VEPersonal.

As future work, we can propose to develop the management of simultaneous interactions from users. Agents' communication protocols must include user's identification and the DBMS's connection, in order to recover a specific VE for each user and to update it accordingly.

7. References

[1] C. Kirner, R. Siscoutto, “Fundamentos de Realidade Virtual e Aumentada”. In: Realidade Virtual e Aumentada: Conceitos, Projeto e Aplicações. Eds. Cláudio Kirner e Robson Siscoutto. *Pre-Symposium Book, IX Symposium on Virtual and Augmented Reality*, Petrópolis – RJ, 2007.

[2] J. Rickel, W. Johnson, “Task-Oriented Collaboration with Embodied Agents in Virtual Worlds”. *Embodied Conversational Agents*, MIT Press, Boston, 2000.

[3] J. Rickel, S. Marsella, J. Gratch, R. Hill, D. Traum, W. Swartout, “Toward a New Generation of Virtual Humans for Interactive Experiences”. *IEEE Intelligent Systems, vol. 17, no. 4, Special issue on AI in Interactive Entertainment, 2002*.

[4] R. Chittaro, R. Ranon, L. Ieronutti, “Guiding

Visitors of Web3D Worlds through Automatically Generated Tours". *Proceedings of Web3D 2003: 8th International Conference on 3D Web Technology*, ACM Press, New York, March 2003, pp. 27-38.

[5] E. A. Wernert, A. J. Hanson, "A framework for assisted exploration with collaboration". In *Proceedings of IEEE Visualization'99 (VIS '99)*, pp. 241–248. 1999.

[6] K. Walczak, W. Cellary, "Building database applications of virtual reality with x-vrml". In *Proceedings of the 7th International Conference on 3D Web Technology*, 111–120, 2002.

[7] L. Chittaro, R. Ranon, "New Directions for the Design of Virtual Reality Interfaces to E-Commerce Sites". *Proceedings of AVI 2002: 5th International Conference on Advanced Visual Interfaces*, ACM Press, New York, pp. 308-315, May 2002.

[8] C. T. Santos, "Um ambiente Virtual Inteligente e Adaptativo Baseado em Modelos de Usuário e Conteúdo" (Máster Thesis). *São Leopoldo: UNISINOS*, 2004.

[9] M. S. Aquino, F. F. Souza, A. C. Frery, "VEPersonal – An infrastructure of Virtual Reality Components to Generate Web Adaptive Environments". A*CM International Conference Proceeding Series*; Vol. 125. *Proceedings of the 11th Brazilian Symposium on Multimedia and the Web – WebMedia 2005*, Poços de Caldas-MG, Brazil, December 05 – 07, pp. 1-8, 2005.

[10] M. S. Aquino, "VEPersonal – Uma Infra-estrutura para Geração e Manutenção de Ambientes Virtuais Adaptativos (Phd thesis). *Centro de Informática, Universidade Federal de Pernambuco*. 2007, pp. 180.

[11] X3D International Specification Standards. http://www.web3d.org/x3d. Last access in April 2007.

[12] Extensible Markup Language. http://www.w3.org/XML. Last access in April 2007.

[13] M. S. Aquino, F. F. Souza, A. C. Frery, "A Multi-Agent Architecture for Generating and Monitoring Adaptive Virtual Environments". *5th International Conference on Hybrid Intelligent Systems – HIS'05*, Rio de Janeiro-RJ, Brazil, Publisher: IEEE Computer Society, Washington, DC, USA, 2005 p.515-517.

[14] M. S. Aquino, F. F. Souza, A. C Frery, D. A. C. M. Souza, R. C. Fujioka, "Supporting Adaptive Virtual Environments with Intelligent Agents". *Proceedings of the 7th International Conference on Intelligent Systems Design and Applications, ISDA'07*, Rio de Janeiro-RJ, Brazil, Los Alamitos : IEEE Press, 2007. v. 1, 2007, pp 217-222.

[15] Java Agent DEvelopment framework – JADE. http://jade.tilab.com. Last access in April 2007.

[16] Foundation for Intelligent Physical Agents – FIPA. http://www.fipa.org. Last access in April 2007.

[17] FIPA Communicative Act Library Specification. http://www.fipa.org/specs/fipa00037/SC00037J.html. Last access in April 2007.

[18] Xj3D - Java based X3D Toolkit and X3D Browser. http://www.web3d.org/x3d/xj3d. Last access in April 2007.

[19] Web3D Consortium. Open Standards for Real-Time 3D Communication. http://www.web3d.org. Last access in April 2007.

[20] S. N. Matsuba, "Interview Yumetech – Xj3D. 3d-test – 3D temps réel et interactive". http://www.3d-test.com/interviews/xj3d_1.htm. Last access in April 2007.

[21] Scene Access Interface Tutorial. http://www.xj3d.org /tutorials/general_sai.html. Last access in April 2007.

[22] Java Web Start Technology. Sun Developer Network. http://java.sun.com/products/javawebstart. Last access in April 2007.

[23] Dynamic-link library – DLL. http://msdn2.microsoft.com/en-us/library/ms682589.aspx. Last access in September 2007.

[24] Lipe RMI – A Light Weight Internet Approach for Remote Method Invocation. http://lipermi.sourceforge.net. Last access in September 2007.

[25] Remote Method Invocation – RMI. http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp. Last access in September 2007.

[26] A. Glove, "In pursuit of code quality: Performance testing with JUnitPerf". http://www-128.ibm.com/developerworks/java/library/j-cq11296.html. Last access in September 2007.

IVR: Uma Plataforma de Realidade Virtual Imersiva para Transmissão de Vídeo de Alta Definição e Interatividade

Erick A. G. Melo, Thales P. Ferreira, André F. Palmeira,
Marcelo F. de Sousa, Guido Lemos de Souza Filho

Laboratório de Aplicações de Vídeo Digital - Departamento de Informática
Universidade Federal da Paraíba
Campus I - Cidade Universitária - João Pessoa – PB – BRASIL
{erick}, {thales}, {andre}, {marcelo}, {guido}@lavid.ufpb.br

Resumo

Neste artigo apresentamos uma plataforma de realidade virtual imersiva (IVR), a qual agrupa um conjunto de vídeos de alta definição, gerando um vídeo panorâmico 360º. O vídeo gerado pode ser visualizado em uma caverna 3D, sendo possível a imersão do usuário e a interatividade. A obtenção é feita através de câmeras firewire de alta definição, e os vídeos capturados são comprimidos e transmitidos em tempo real. As câmeras são transportadas por um robô, controlado por uma aplicação web. A disposição das câmeras e o movimento do robô fornecem ao usuário um vídeo panorâmico interativo em tempo real. A transmissão do mesmo é realizada através de uma rede wireless, dando mobilidade ao robô. A plataforma IVR é usada em uma rede gigabit dedicada, a qual provê a transmissão dos vídeos entre os estados do Rio de Janeiro e São Paulo, no Brasil.

1. Introdução

A Realidade Virtual (RV) é usada de várias maneiras e tem por objetivo a imersão do usuário, ou usuários, em um ambiente virtual composto por 3 dimensões (3D) através, comumente, do uso de um computador e dispositivos acessórios. Desenvolvimentos recentes em vídeo e tecnologias de automação robótica têm permitido que aplicações RV gerem melhores ambientes virtuais (AV). A interação e a eficiência da Interface Homem – Máquina (IHM) tem sido uma questão chave na usabilidade e sofisticação dessas aplicações [3].

Neste artigo apresentamos um trabalho que buscou focar nessas questões chaves para oferecer um melhor controle por parte do usuário e melhor qualidade das imagens do ambiente simultaneamente. No lado do cliente, foi desenvolvida uma aplicação a fim de controlar um passeio por um caminho virtual. Para visualizar o AV, é utilizada a primeira CAVE (Cave Automatic Virtual Environment) construída na América Latina, localizada no Laboratório de Sistemas Integrados (LSI) [4], na Universidade de São Paulo (USP). Um servidor de vídeo (streamer) é responsável pelo envio do vídeo a ser exibido na CAVE.

Explorar lugares desconhecidos ou perigosos pode ser evitado fazendo uso de aplicações RV. Ao invés do uso de material humano, robôs com câmeras podem ser utilizados para exploração de tais lugares. Para realizar esta tarefa, um tipo de sistema usaria apenas uma câmera, cabendo única e exclusivamente ao diretor, escolher quais as regiões do espaço que devem ser efetivamente capturadas, limitando de certa forma a experiência do usuário final. Uma melhor aproximação da realidade seria usar várias câmeras, em uma configuração em forma de anel, de forma a permitir que um maior campo de visão seja simultaneamente capturado e exibido em tempo real aos usuários. No sistema proposto, usamos um anel de câmeras, montados em uma plataforma robótica, para capturar uma imagem aproximada de 360º.

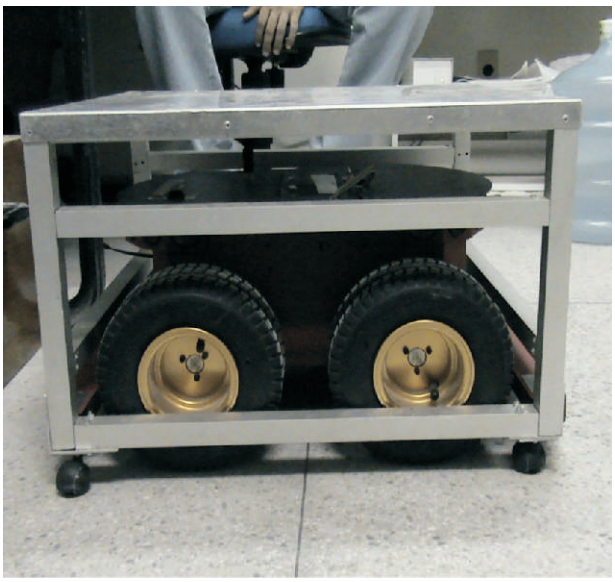


Figura 1. Protótipo do robô.

Figura 2. Protótipo do robô com seis câmeras.

2. Plataforma IVR

A plataforma IVR possibilita ao usuário acessar a aplicação interativa, que vem multiplexada com o vídeo panorâmico, sendo visualizada através da caverna digital, por um navegador web ou até mesmo por uma televisão (TV) digital conectada a um Terminal de Acesso (*Set-top Box*).

O acesso remoto à aplicação pode ser feito através de vários dispositivos, desde um simples controle remoto de TV digital até por um celular via bluetooth, infra-vermelho, ou wi-fi, ou qualquer outro dispositivo que possua tais tecnologias de acesso.

Para possibilitar a execução das aplicações no lado cliente é usado o *middleware* Ginga [1]. Ginga é o nome do *middleware* especificado pelo padrão internacional do sistema brasileiro de TV digital terrestre. As aplicações a serem executadas sob o Ginga são classificadas em duas categorias, dependendo da forma que são escritas. Aplicações procedurais são aquelas escritas usando a linguagem Java, e, declarativas são aquelas escritas usando a linguagem NCL. Similarmente, os ambientes de execução de aplicação Ginga podem ser de dois tipos: para processo declarativo ou aplicações procedurais, e são chamadas Ginga-J [1] e Ginga-NCL [2], respectivamente.

O aglomerado de câmeras de alta definição destinadas à captura dos vídeos é conectado aos laptops acoplados aos robôs, conforme Figura 2, e estes recebem e codificam os vídeos em software ou hardware, os quais serão transmitidos via rede sem fio para um *switch gigabit ethernet* através dos access points. Posteriormente, os fluxos de vídeos serão recebidos e processados pelo Cluster [5] localizado na Universidade de São Paulo, onde será gerado e exibido um vídeo panorâmico de alta definição na caverna digital. Um protótipo resumido dos componentes de hardware da IVR pode ser visualizado através da Figura 3.

O conjunto de câmeras firewire e laptops ficam sobre o robô. Cada laptop possui uma placa pré-802.11n que possibilita taxas de transferências nominais de 300 Mbps e é responsável pela codificação e transmissão de três fluxos de vídeo de alta definição, obtendo então um fluxo total aproximado de 75 Mbps. Os access points também são pré-802.11n e possuem uma interface de rede gigabit ethernet comunicando-se diretamente com o switch gigabit, localizado na Universidade Federal do Rio de Janeiro de onde são transmitidos todos os fluxos de vídeos para a Universidade de São Paulo.

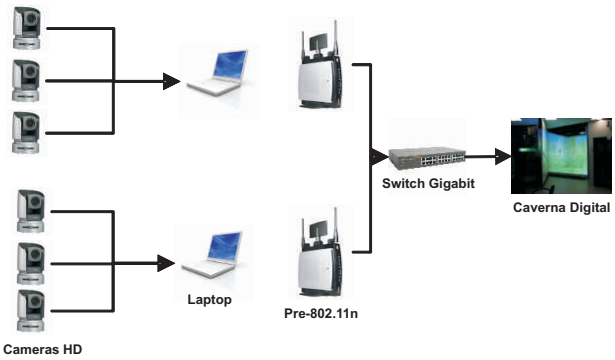


Figura 3. Componentes de hardware da IVR.

2.1 Plataforma robótica

O robô precisa levar consigo todas as câmeras e os laptops que farão a transmissão dos fluxos de vídeo. Para suportar o peso de todos esses itens, projetou-se uma plataforma capaz de transportar os equipamentos. Com isso, o peso não será aplicado diretamente sobre a estrutura do robô, livrando-o de possíveis sobrecargas, conforme a Figura 1.

Após realizados testes de carga com o robô, onde o mesmo conseguiu transportar aproximadamente 120 Kg, foi construída uma plataforma de metalon. A plataforma é constituída de dois compartimentos: o compartimento inferior, dedicado ao transporte dos computadores, com dimensões suficientes para que os computadores possam permanecer bem acomodados, e o compartimento superior, dedicado ao transporte das câmeras, que possui dimensões coerentes com as dimensões do suporte das câmeras. O compartimento inferior foi projetado de modo que ele possa ser desacoplado da plataforma, o que facilita o trabalho de fixação e ajuste da posição das câmeras, possibilitando que a parte inferior da plataforma seja utilizada para outros fins.

2.2 Controle da plataforma robótica

O robô é configurado através de um arquivo *Extensible Markup Language* (XML) [10], o que facilita a sua configuração, assim como também, a adição de novas características ao seu controle remoto. Na Figura 4, temos a Configuração da rede (<config>), que exibe informações como o *Internet Protocol* (IP) do robô e a porta, possibilitando conexões remotas.

```
<config>
  <nome>Galateia</nome>
  <ip>galateia.dca.ufrn.br</ip>
  <porta>7777</porta>
  <descricao>Robo de 4 rodas UFRN</descricao>
</config>
```

Figura 4. Código XML com a configuração da rede.

A Figura 5 nos mostra a tag <video>, que informa a porta pela qual o vídeo será enviado pelo robô e chegará ao cliente que a controla, além de informações sobre a janela de apresentação do vídeo.

```
<video>
  <id>Camera 01</id>
  <portavideo>1234</portavideo>
  <position>
    <x>1</x>
    <y>0</y>
  </position>
  <size>
    <width>250</width>
    <height>200</height>
  </size>
</video>
```

Figura 5. Código XML com a configuração do vídeo.

Por último, a Figura 6 nos mostra um exemplo da tag <button>, que permite associar um botão que aparece no controle remoto a uma ação do robô. Nela, temos um botão que fará o robô andar para frente.

```
<button>
  <type>press</type>
  <caption>FRENTE</caption>
  <key>UP</key>
  <value>10</value>
  <position>
    <x>98</x>
    <y>210</y>
  </position>
  <size>
    <width>53</width>
    <height>53</height>
  </size>
  <icon>up.png</icon>
  <iconpress>upPress.png</iconpress>
  <hint>Move para frente</hint>
</button>
```

Figura 6. Código XML com a configuração de um botão do controle remoto que movimenta o robô.

Os módulos de controle que são executados pelo robô são três:

· Módulo **ControlServer** – é responsável pelo recebimento de conexões de clientes que desejam comandá-lo remotamente.

· Módulo **VideoServer** – é responsável pelo envio de um stream de vídeo da câmera, acoplada a sua estrutura e instalada no computador embarcado.

· Módulo **VirtualServer** – é responsável pela conexão do robô ao mundo virtual. Através desse módulo é feita a representação 3D do mundo do robô no mundo virtual. Esse módulo também é responsável pelo envio de mensagens de movimento para o servidor do mundo virtual.

O módulo VideoServer foi construído apenas para validar a teleoperação do robô e pode ser substituído pelo conjunto de câmeras que ficam sobre ele, conforme a Figura 2.

Visando melhorar o acesso à plataforma robótica, foram adicionadas novas possibilidades ao uso das setas do teclado para informar que direções o robô terá de seguir. Isso pode ser feito via luva de dados usando um protocolo específico, conforme a Figura 7, o qual é utilizado para implementar um driver de dispositivo e depois fazer uso da biblioteca *Java Native Interface* (JNI) [8] para a utilização da mesma em ambiente Java. Esse acesso ao robô também pode ser expandido através de um celular que possua rede sem fio ou pelo navegador web, possibilitando assim outros meios de controle da plataforma robótica. O uso de Java facilita o desenvolvimento, visto que a maioria dos dispositivos citados podem ser acessados por tal linguagem.

```
1)  Abrir uma sessão da porta COM com os
seguintes parâmetros:
    Speed      = 38400 bps
    Bits       = 8
    Parity     = NO
    Stopbit    = 1

2)  Enviar o comando INFO_REQ (0xFFh) para
verificar a presença da luva na porta COM:
    Se presente podemos ler um pacote de
    dados (22bytes):
    Productor          4 bytes
    Glove Name         12 bytes
    Firmware version   4 bytes

3)  Resetar a luva com o comando RESET
(0xFEh):
    Agora a luva está em STAND-BY (ver SDK)

4)  Enviar um comando  START_COMMAND (0xFDh)
para entrar em LOOP MODE

    Em LOOP MODE a luva envia pacotes de
dados para o computador a taxas de 100Hz.
```

Figura 7. Detalhes do protocolo default da luva de dados.

Para a visualização do vídeo panorâmico em tempo real e a movimentação do robô, utilizamos uma interface de controle conforme a Figura 9, que se aproveita de um componente gráfico que possibilita a visualização do vídeo em vários formatos utilizando um navegador web. Tal componente é um player de vídeo chamado VLC [6].

Para inserirmos o VLC na interface do controle remoto fazemos o uso do JVLC [7], que provê uma *Application Programing Interface* (API) para criação de componentes que permitem a adição de um player dentro de um applet Java. Com isso o controle remoto se utiliza de um player flexível e robusto para a exibição dos vídeos.

O protótipo de plataforma robótica utilizado atualmente, sem nenhuma alteração, é o *Pioneer 3AT* da *ActivRobots* [9], conforme a Figura 8.

Figura 8. Plataforma robótica.

2.1 Testes de odometria na plataforma robótica

Odometria é o estudo da posição estimada durante a navegação de veículos com rodas. A Odometria é usada, por exemplo, por alguns robôs com rodas, para estimar (não determinar) a sua posição em relação a uma localização inicial.

O seu conceito também consiste no uso dos dados da rotação das rodas ou rastros para estimar mudanças de posição ao longo do tempo. Esse método é normalmente muito sensível a erro. Rapidez e eficácia na coleta de dados, calibração de equipamento e processamento são requeridos para que a Odometria seja usada efetivamente.

A realização de testes com o robô Pionner 3-AT objetiva corrigir possíveis erros de Odometria, pois é ela que torna possível a atualização da posição do avatar do robô (sua posição virtual) no ambiente 3D. À medida que o robô anda, ele gera um número que representa sua posição atual. Esse valor é utilizado e logo depois transformado pelo ambiente virtual. Porém, o sistema interno do Pioneer 3-AT, responsável pela implementação da Odometria, adiciona erros a esse valor.

Por exemplo, em alguns casos o valor informado continua aumentando mesmo com o robô em repouso. Em outros, depois de algum tempo executando, o valor é informado erroneamente, devido a fatores externos, como derrapagem dos pneus no chão.

Uma das soluções encontradas foi zerar periodicamente o valor gerado pelo robô, de modo a não permitir que ele alcance altos valores e com isso gere erros substanciais. Sempre que o valor é zerado, ele informa ao ambiente virtual que o sistema de Odometria foi zerado para que a posição seja calculada de forma correta.

Figura 9. Protótipo do controle da plataforma robótica.

3. Sistema de streaming de vídeo panorâmico

A plataforma de geração de vídeo panorâmico proposta é baseada na especificação de um sistema distribuído utilizada em [25] para criação de vídeo 3D em tempo real. Como observado na seção 2, a característica fundamental da plataforma é o seu caráter distribuído, por possuir uma coleção de computadores independentes, que aparenta aos usuários um sistema único e consistente [26]. Um computador central, responsável pela maior parte do processamento computacional, recebe, através de uma rede local, dados vindos de dois ou mais computadores que possuem informações inter-dependentes, como os seus *temporal reference* [27], utilizados para a sincronização, os

descritores do ambiente em que se encontra o robô e as tabelas descritoras dos fluxos de dados, utilizadas para possibilitar a futura demultiplexação do agrupamento dos fluxos de vídeo que chegam até o cluster. Logo, para gerenciar tais tarefas, podemos alocar estas responsabilidades para um dos laptops que geram o fluxo de vídeo. Assim, apesar de independentes, cada laptop não enxerga somente a si mesmo, mas trabalha colaborativamente compartilhando os dados necessários para a exibição do vídeo panorâmico. Mais adiante veremos que estes computadores poderão sofrer uma carga extra no processamento de acordo com o tipo de câmeras que eles utilizam. As bibliotecas utilizadas foram divididas em duas partes: (i) API para transmissão e codificação de vídeo e (ii) API de serviços para computação distribuída. A implementação da característica distribuída do sistema é feita utilizando um conjunto de serviços oferecidos pelo CORBA [28]. Como ponto inicial, o serviço de nome é utilizado para localizar os objetos distribuídos, ou seja, todos os conjuntos de dados remotos que podem ser compartilhados. O serviço de tempo oferece o *temporal reference* sincronizados para cada fluxo de vídeo. Por último, o servidor de conexão dos canais de comunicação mantém o estado das conexões e, por sua vez, permite lidar com falhas nas câmeras.

A Figura 10 mostra a arquitetura, juntamente com os serviços utilizados. O nó de construção do vídeo panorâmico identifica os nós de fontes de dados usando o serviço de nome, iniciando remotamente a aquisição dos fluxos. Uma fonte de dados pode ser a geração de um vídeo/áudio codificados, uma aplicação, as tabelas descritoras dos fluxos de dados e textos que descrevem o ambiente atual do robô. Através do servidor de conexão o fluxo de dados é enviado sobre canais de transmissão confiáveis até os mesmos chegarem ao nó de streaming de vídeo em tempo real.

Utilizando um arquivo de configuração, que pode ser modificado dinamicamente, o usuário poderá escolher entre utilizar MPEG-2, MPEG-TS ou MPEG4 para encapsular o vídeo e ainda enviá-lo utilizando o padrão MPEG-TS, RTP, RTSP ou até mesmo *sockets*, no caso de envio para redes locais. Os nós de fonte de dados são inicializados em forma de *daemon* permitindo remotamente o início e o término de cada nó, ou o possível encerramento da aplicação. A pré-sincronização é conseguida através da distribuição dos *temporal reference* pelo nó sincronizador e a inserção dos mesmos nos campos da estrutura do MPEG. Assim, futuramente, os frames podem ser emparelhados de acordo com o seu *temporal reference* e na ordem em que o conjunto de câmeras estão agrupadas.

3.1 Captura dos fluxos de vídeo

Para a captura dos fluxos de vídeos vindo das câmeras firewire de alta definição, duas bibliotecas foram usadas, a libdc1394 [11] e a libraw1394 [12]. Libdc1394 é uma biblioteca que provê uma API completa e de alto nível para desenvolvedores que desejam controlar câmeras baseadas no padrão IEEE 1394, que estão em conformidade com as especificações de câmera digital do mesmo padrão (também conhecida como a IIDC [14] ou especificações da DCAM). Ela fornece uma completa API que inclui detecção de câmeras, comandos de difusão, controle de tráfego de recursos (básico), pleno controle de recursos (incluindo valores absolutos), canais de memória, acionador externo ou via software, suporte a todos os modos de vídeo (incluindo modos de 16 bits e modos 1394B a 800Mb/s), captura de vídeo usando DMA e controle full Format_7. A biblioteca também inclui ferramentas para conversão de vídeo como conversão do espaço e da cor (YUV, RGB, MONO, etc) e o demosaicing do padrão de imagens coloridas Bayer (com oito diferentes algoritmos, incluindo modos de 16 bits).

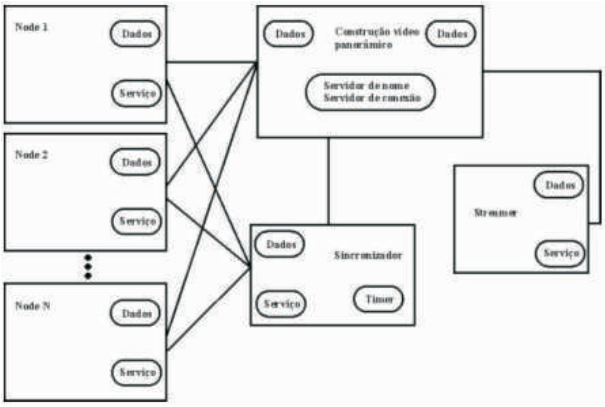


Figura 10. Arquitetura do streaming de vídeo panorâmico.

3.2 Compressão de vídeo

As câmeras geram fluxos de vídeo cru, exigindo assim a codificação dos mesmos para viabilizar a sua transmissão através de redes sem fio.

Como mencionamos, anteriormente, os laptops estão responsáveis pela codificação dos fluxos de vídeo, seja por software ou hardware. A compressão feita por software requer do laptop um grande desempenho do processador e por hardware, menos processamento.

Há no mercado câmeras que utilizam o codec "MPEG HD" que é baseado na compressão MPEG-2 MP@H, permitindo aos usuários gerar um vídeo no formato MPEG-TS a taxa de bits entre 35, 25 e 18 Mbps, fazendo com que a tarefa de codificação não fique sob

a responsabilidade do laptop e sim, da câmera. MPEG-TS (*Transport Stream*) é um protocolo de comunicação para áudio, vídeo, e dados que é especificado no MPEG-2 parte 1 [13]. A sua característica principal é permitir a multiplexação de áudio e vídeo digital e a sincronização da saída. O MPEG-TS oferece recursos para correção de erros no transporte de mídia não confiável, e é usado em aplicações broadcast como DVB (*Digital Video Broadcasting*) e ATSC (*Advanced Television Systems Committee*). Ela é contrastada pelo MPEG-PS (*Program Stream*), desenvolvido para mídias mais confiáveis como as dos DVDs.

Cada *transport stream* é formado por um conjunto de *sub-streams* conhecidos como *elementary streams* (ES). Cada ES pode ser o resultado da compressão em MPEG-2 dos fluxos de áudio, vídeo ou dados representados unicamente no stream, utilizando um identificador de pacote (PID) [29]. Para cada nó de dados da Figura 10 um canal de vídeo, uma trilha de áudio ou dados estará disponível para ser multiplexado. Devido à natureza contínua de cada stream, a multiplexação não seria tão simples de ser feita, considerando as possibilidades da existência de dezenas deles e ainda erros decorrentes da transmissão dos pacotes enviados, degradando a experiência do usuário devido a retransmissões. Logo, é feita a quebra dos streams em pacotes menores, e armazenados em unidades denominadas transport packets contendo 188 bytes. Este nível extra de empacotamento permite um maior suporte para as técnicas de correção de erro [29].

Uma vez que temos um conjunto completo de *transport packets*, podemos inseri-los dentro do transport stream final. Também queremos incluir streams formados exclusivamente por dados, sendo estes gerados através das aplicações e de informação textual. Um tipo de informação textual poderia ser a descrição do ambiente do robô citado anteriormente. Para isto o padrão MPEG provê uma forma bem definida para carregar dados que não têm conformidade com o tipo áudio/vídeo dentro dos *transport packets*.

Para reconstruir os ES em algo que o receptor possa apresentar para o usuário, o MPEG especifica que outra informação deverá ser adicionada. Estes dados são codificados em forma de ES e colocados no transport stream durante o processo de multiplexação. Tais dados se parecem com um simples banco de dados que descreve a estrutura do transport stream. Em um nível mais simples, isto contém um número de tabelas, as quais descrevem um conjunto de áudio, vídeo e dados. Estas tabelas listam cada stream, provendo o PID e o tipo de dados contido no mesmo, conforme a Figura 11.

No fluxo multiplexado MPEG-TS vai inserida uma aplicação interativa, e quem possibilitará o acesso à mesma no lado cliente será o middleware Ginga, e o cluster gráfico na Universidade de São Paulo fica responsável pelo processamento da imagem e a geração do vídeo panorâmico 3D. O usuário poderá interagir com tal aplicação através de um controle remoto padrão de TV ou por um celular que possua wi-fi ou outra tecnologia de rede sem fio (infra-vermelho, bluetooth), conforme Figura 12.

Além da codificação em alta definição também temos que garantir que usuários que não possuam uma conexão Gigabit Ethernet, o que é mais comum nos dias de hoje, possam acessar e interagir na internet com o robô e a aplicação. Para isso, nos utilizamos de três bibliotecas na compressão feita por software: A XviD [15], a libavformat e a libavcodec, sendo estas duas últimas pertencentes a FFmpeg [31].

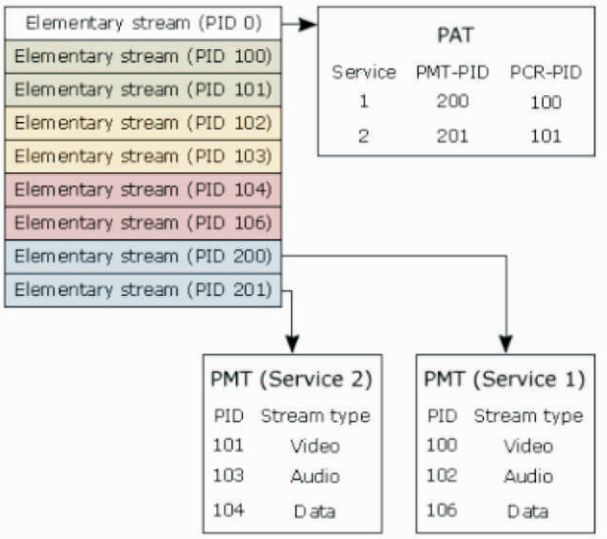


Figura 11. Informações do stream: PID e tipo de dados [29].

XviD é um codec de vídeo que implementa algumas definições do padrão MPEG-4 [16]. XviD possui características do perfil avançado deste padrão como b-frames (imagens bidirecionais: a codificação é feita em duas direções, ou seja, em frames que já foram exibidos e os que ainda vão ser exibidos), compensação de movimento de pixel global e local, máscara de iluminação, quantização trellis, e H.263 [17], MPEG e matrizes de quantização personalizadas.

A libavcodec é uma biblioteca responsável pela codificação e decodificação de áudio e vídeo e a libavformat é um conjunto de *parsers* e geradores de vários formatos de áudio e vídeo. Com elas, é possível também a codificação dos fluxos de vídeos crus vindos

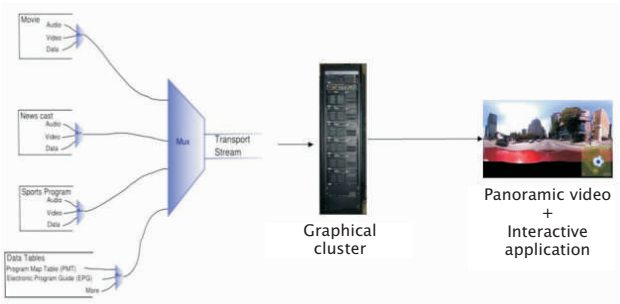


Figura 12. Transport Stream multiplexado com aplicação e processado em um vídeo panorâmico pelo cluster gráfico.

das câmeras, no padrão MPEG-2 [30].

Para a compressão destes fluxos de vídeo, foram implementadas duas versões: uma utiliza a XviD, codificando-os no padrão MPEG-4 e a outra utiliza a libavformat e libavcodec, codificando-os no padrão MPEG-2. Para este último, modificamos algumas características do vídeo comprimido, como por exemplo, diminuindo o tamanho da resolução do vídeo de saída. Para os dois formatos, garantimos uma transmissão de vídeos de ótima qualidade na internet.

A maioria dos players de hoje suportam os formatos de vídeo MPEG2, MPEG-TS e MPEG-4, tornando-se assim mais um requisito importante para a plataforma IVR.

3.3 Transmissão de vídeo

A transmissão do fluxo TS multiplexado é executada por um streamer TS chamado *TSStreamer*. Tal ferramenta pode enviar um arquivo TS através da rede mantendo a taxa correta, quaisquer que sejam os formatos de vídeo, áudio ou dados que o mesmo carregue. Para isso, o streamer usa as informações do PCR (*Program Clock Reference*), que traz a marcação de tempo em milissegundos de cada um dos programas do TS que contenham áudio e vídeo. Usando o *clock* da CPU como referência, o TSStreamer consegue então manter o envio do TS na taxa correta, não importando quais formatos de vídeo tenham sido usados na codificação. Além disso, dado um TS, esta ferramenta pode transmiti-lo na taxa correta, e opcionalmente inserir pacotes nulos.

Para garantir a transmissão dos fluxos de vídeos em MPEG-4 usamos um streamer em *Real Time Streaming Protocol* (RTSP) [18]. O RTSP é um protocolo de aplicação para controle sob a distribuição de dados com propriedades em tempo real. RTSP provê um *framework* extensível que permite controlar a entrega sob demanda de dados em tempo real, como áudio e vídeo. Esse protocolo se destina a controlar sessões de entrega de dados, provendo um significado para escolha de canais de entrega como *User Datagram Protocol* (UDP), *multicast* UDP e *Transmission Control Protocol* (TCP), e provê um meio para escolher mecanismos de entrega baseados em RTP [19]. O streamer utilizado é o Spook [20], um daemon para captura, processamento, codificação, e distribuição de fluxos de áudio e vídeo. Ele pode ser usado por *webcams*, vídeos de vigilância, distribuição de programas de televisão ou vídeos ao vivo, *webcasts*, e até mesmo vídeo conferência.

A entrada de vídeos pode ser alimentada por dispositivos da Video4Linux(1/2) e câmeras *Firewire* IIDC, e a entrada de áudio pode ser alimentada por dispositivos OSS ou por dispositivos *Advanced Linux Sound Architecture* (ALSA) através da camada compatível *Open Sound System* (OSS). MPEG-4, via XviD, e *Joint Photographic Experts Group* (JPEG), via JPEGlib são os codificadores de vídeo suportados.O único formato de áudio comprimido suportado atualmente é o MP2 (MPEG1 camada 2) através de um codificador embutido. MPEG-4 e MP2 RTP streams estão disponíveis via RTSP e JPEG ainda está disponível via *HyperText Transfer Protocol* (HTTP).

3.4 Reprojeção de vídeo

A partir das imagens capturadas das câmeras é possível reconstruir uma imagem panorâmica a partir do método da reprojeção de Kurash [21]. Este método funciona a partir do uso da geometria epipolar. Com o uso dessa geometria é possível reconstruir, a partir de imagens 2D, a geometria 3D de uma cena.

O objetivo da utilização deste método é alcançar a projeção e o "casamento" correto das imagens nas bordas, ou seja, a fusão das imagens das diversas câmeras em apenas uma imagem, própria para projeção em ambientes imersivos do tipo CAVE.

Uma premissa deste método é assumir que o ambiente de multiprojeção possui dimensões físicas conhecidas, o que é perfeitamente passível de ser obtido, seja pela especificação ou através de medições. Este método considerará ainda que as quatro paredes do ambiente de multiprojeção formam um cubo com dimensão L (por exemplo, L = 3m indica quatro telas de 3m x 3m, que é o caso da CAVERNA Digital USP).

Para a construção das imagens foi considerado o ponto de vista da cena V como sendo o centro geométrico do cubo do ambiente imersivo de multiprojeção, conforme mostrado na Figura 13. Com esta consideração inicial, a construção da cena segue o mesmo procedimento para todas as telas. O ponto de vista V fica a uma distância L/2 do plano de projeção e a uma altura de L/2 do piso. O campo de visão horizontal e vertical neste

ponto é de 90º.

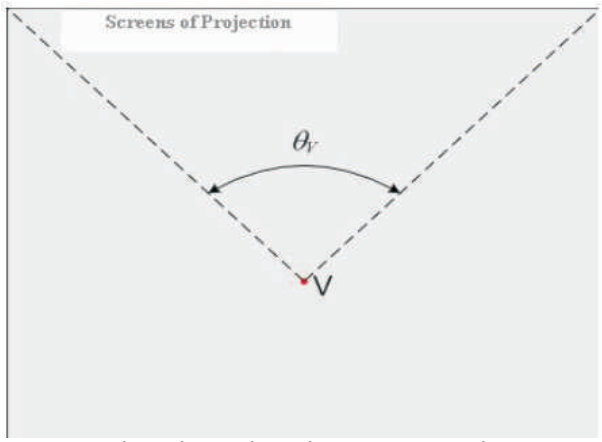


Figura 13. Planta baixa do ambiente imersivo de multiprojeção.

O objetivo do Método da Reprojeção é construir uma imagem I_V vista do ponto V, com campo de visão θ_V, e em direção perpendicular à tela de projeção. Esta imagem é baseada nas imagens capturadas pelo conjunto de câmeras, que são mencionadas como C_0, C_1, ...C_{N-1}, onde N é o número total de câmeras no anel. Dependendo do ângulo relativo da direção das câmeras em relação à direção de V, poderão ser suficientes $M = (N/4) + 1$ ou $M = (N/4) + 2$ câmeras para construir a imagem de cada tela do ambiente imersivo de multiprojeção. No exemplo estudado com $N = 12$, são suficientes quatro ou cinco câmeras do anel para cada tela. A Figura 14 mostra um exemplo para $M = 4$ câmeras e outro exemplo com $M = 5$ câmeras. Observa-se que a projeção das imagens das M câmeras irá contribuir para a síntese da imagem na tela de projeção.

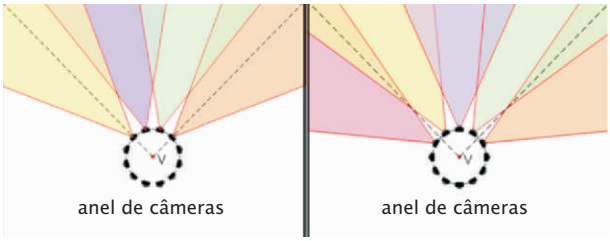


Figura 14. Projeção com 4 e 5 câmeras.

4. Ambiente virtual colaborativo

Para a implementação do ambiente virtual colaborativo, foi utilizado uma nova versão do servidor multi-usuário Vixnu [23], que foi reconstruído seguindo o *framework* HN2N [22]. Este *framework* tem como objetivo possibilitar a construção de sistemas colaborativos distribuídos, que consigam suportar um elevado número de usuários conectados ao mesmo tempo, sem que haja perda na qualidade de serviço (QoS). A idéia principal consiste em separar os clientes (indivíduos) em grupos e organizar esses grupos de maneira hierárquica, conforme a Figura 15.

O HN2N é então uma arquitetura cliente/servidor hierárquico, onde, para cada grupo de clientes, existe um servidor à sua disposição. Numa configuração inicial, existe somente um servidor, representando o primeiro grupo de usuários (Grupo 1). Este servidor (S0) encontra-se no nível 0 da hierarquia do sistema. À medida que mais e mais clientes se conectam ao servidor S0, haverá um momento em que este ficará sobrecarregado, fazendo com que a qualidade do sistema fique comprometida.

Neste momento, um novo servidor é criado, representando o Grupo 2, para ajudar a suportar os usuários. Este servidor (S1), recém criado, será posicionado no nível 1, que se encontra logo abaixo do nível 0 e terá uma conexão com S0. Os clientes do sistema são reagrupados de forma a dividirem-se entre os servidores representados pelos grupos 1 e 2.

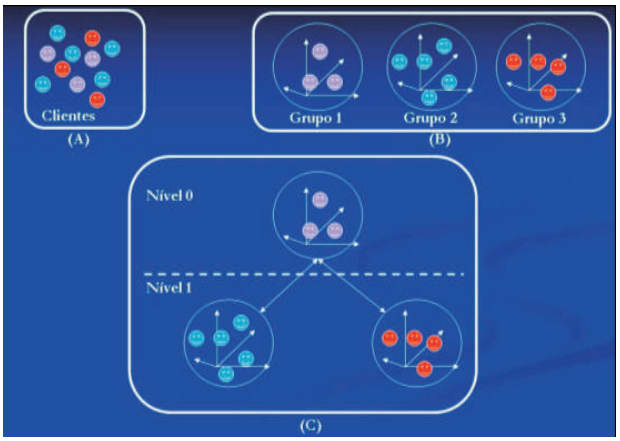


Figura 15. (A) Clientes, (B) Grupo de clientes, (C) Organização hierárquica [22].

O diferencial da arquitetura HN2N em relação as demais que se propõem a resolver o problema de ambientes de larga escala, é a possibilidade de deixar todos os clientes indiretamente interligados, sem que haja perda na qualidade do serviço e possibilitando qualquer cliente se comunicar com qualquer outro que também esteja conectado. Além disso, aplicam-se filtros nas mensagens que vão de um grupo para outro, reduzindo assim o tráfego e conseqüentemente o processamento nos clientes.

O HN2N é o único modelo que resolve o problema do "Tiro na Multidão". Nele, mesmo que o cliente não esteja próximo (no mesmo grupo) de quem deu o tiro

(ou outro evento de alta relevância), essa mensagem será marcada com uma prioridade maior e os servidores não filtrarão essa mensagem já que ela é de interesse de todos do ambiente. Assim, todos estarão interessados em saber se está havendo um tiroteio por perto, coisa que as outras arquiteturas não resolvem.

4.1 Vixnu 2.0

O Vixnu 2.0 é um servidor multi-usuário que fornece suporte à comunicação multimídia entre os avatares de um ambiente virtual, oferecendo aos usuários recursos como comunicação com voz real e voz virtual, provendo maior flexibilidade aos avatares. Deste modo, além de uma representação gráfica, os avatares podem assumir uma representação em áudio de acordo com as preferências do usuário.

Optou-se pela construção de uma nova versão desse servidor, devido a algumas limitações identificadas em sua primeira versão como:

· A interface 3D dos clientes só executa em ambiente Windows e com o navegador Internet Explorer em conjunto com a Microsoft VM;

· Haver replicação de código nos módulos de comunicação (responsáveis pela criação de conexões UDP, TCP);

· Era um sistema que não suportava muitos usuários.

Uma vez implementado, seguindo o framework HN2N, o Vixnu 2.0 herda a característica de suportar grandes números de usuários e aproveita o mesmo sistema de comunicação para envio e recebimento de mensagens, evitando assim a replicação de código. E ao utilizar a API Java3D [24] torna-se possível a execução da interface 3D em multiplataformas.

A arquitetura alto nível do sistema pode ser vista no diagrama ACME da Figura 16, onde temos o componente *Vixnu_Server*, conectados a dois clientes: o *Vixnu_Client_User* (que faz interface com o usuário) e o *Vixnu_Client_Agent* (que se comunica com o robô).

5. Aplicações Interativas

Para a exibição da aplicação que vem multiplexada ao fluxo TS é necessário o middleware Ginga. O middleware é responsável por "esconder" as particularidades do equipamento no qual uma determinada aplicação será executada.

Podemos citar inúmeros exemplos de aplicações que poderiam utilizar a plataforma proposta. Uma delas seria um museu virtual, através do qual o usuário navegaria por ele e controlaria o robô por um passeio em um ambiente real. Ao se deparar com uma determinada obra de arte, o usuário poderia selecionar e acessar informações referentes a tal obra.

Outra aplicação seria para uma torcida virtual. Em uma situação de Copa do Mundo, por exemplo, um grupo de usuários brasileiros poderia estar em um ambiente 3D, como uma caverna digital, para assistir uma partida entre Brasil e Argentina, podendo escolher assisti-la dentro de uma sala de um grupo de torcedores argentinos, também em um ambiente 3D, mas em outro local do mundo. Através da aplicação, os dois grupos poderiam interagir entre si, de qualquer lugar do globo, recebendo não só vídeo, mas também áudio de alta definição, conforme a Figura 17.

Como último exemplo, podemos citar uma aplicação para o comércio eletrônico. Nela, o cliente também estaria dentro de um ambiente 3D, e teria a opção de escolher o seu próprio produto em uma prateleira de supermercado ou shopping através de um robô guiado por controle remoto, podendo visualizar todos os ângulos do produto. Caso o cliente deseje comprá-lo, o robô leva-o até o caixa, fazendo com que o usuário participe remotamente de todas as etapas da compra.

6. Conclusão

Neste artigo fizemos à descrição detalhada dos componentes integrantes da plataforma IVR. Foi mostrado que através dela é possível a criação de um ambiente virtual imersivo 3D de alta qualidade de

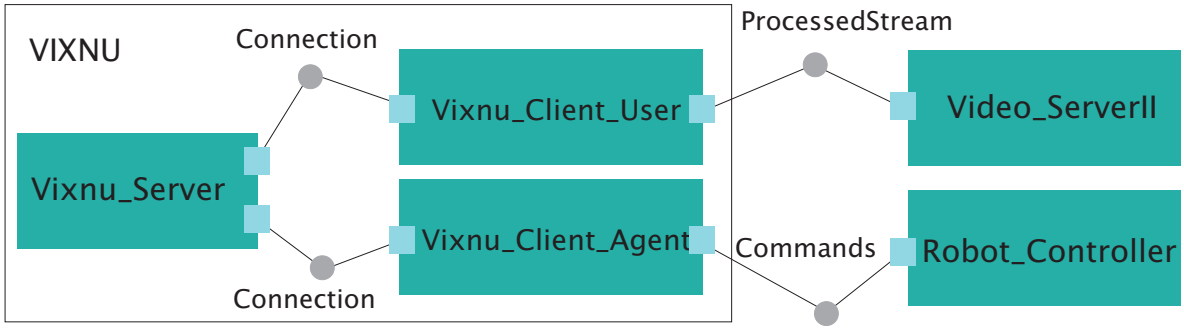


Figura 16. Modelagem em ACME do Vixnu [23].

Figura 17. Vídeo panorâmico com aplicação [32].

imagens em redes de altíssima velocidade e com o uso de aplicações interativas no mesmo, não deixando de lado também a transmissão de conteúdo para web, convergindo assim o acesso à plataforma por vários meios, desde celulares a navegadores web. A execução de tais aplicações interativas tornou-se possível devido ao uso do middleware Ginga, desenvolvido para o Sistema Brasileiro de TV Digital.

7. Referências

[1] G. L. de Souza Filho, L. E. C. Leite, and C. E. C. F. Batista. "Ginga-J: The Procedural Middleware for the Brazilian Digital TV System". *In: Journal of the Brazilian Computer Society*. No. 4, Vol. 13. p.47-56. ISSN: 0104-6500. Porto Alegre, RS, 2007.

[2] Soares, Luiz Fernando Gomes; Rodrigues, Rogério Ferreira; Moreno, Márcio Ferreira. "Ginga-NCL: The Declarative Environment od the Brazilian Digital TV System". In: *Journal of the Brazilian Computer Society*. No. 4, Vol. 13, page 37-46. ISSN: 0104-6500. Porto Alegre, RS, 2007.

[3] J. W. Hiltenbrand. "A development framework for highly-interactive virtual reality applications", 2000.

[4] L. P. Soares, V. H. P. Gomes, L. Nomura, M. Cabral, A. R. Pereira, L. Dulley, R. Lopes, and M. K. Zuffo. "Lab presentation of laboratory of integrated systems – lsi". *IEEE Virtual Reality 2005 Lab Presentation*, 2005.

[5] L. P. Soares and M. K. Zuffo. "JINX: an X3D browser for immersive simulation based on cluster of commodity computers". *SIGGRAPH*, Monterey, California, pp. 79-86 2004.

[6] VideoLAN Project. VLC Media player. Disponível em: http://www.videolan.org/. Acessado em junho de 2007.

[7] JVLC Project. Java Multimedia Library. Disponível em: http://trac.videolan.org/jvlc/. Acessado em junho de 2007.

[8] Java Native Interface: Programmer's Guide and Specification v1.5. Disponível em: http://java.sun.com/docs/books/jni/. Acessado em julho de 2007.

[9] ActivMedia Robots: The Professional Roboticists' Source to jump-start autonomous robot and perception research and university education. Disponível em: http://www.activrobots.com/. Acessado em setembro de 2007.

[10] W3C: World Wide Web Consortium. Disponível em: http://www.w3.org/XML/. Acessado em agosto de 2007.

[11] Libdc1394: Site official the library dc1394. Disponível em: http://damien.douxchamps.net/ieee1394/libdc1394/index.php. Acessado em junho de 2007.

[12] IEEE 1394 for Linux. Disponível em: http://www.linux1394.org/doc/libraw1394/. Acessado em junho de 2007.

[13] ISO/IEC standard 13818-1: Moving Picture Experts Group. Disponível em: http://neuron2.net/library/mpeg2/iso13818-1.pdf. Acessado em agosto de 2007

[14] 1394 Trade Association: Specification IIDC. Disponível em: http://www.1394ta.org/index.shtml. Acessado em setembro de 2007.

[15] XviD: MPEG-4 Video Codec. Disponível em: http://www.xvid.org/Home-of-the-Xvid-Codec.1.0.html. Acessado em agosto de 2007.

[16] MPEG4 Video Codec Overview. 2007. Disponível em: http://www.mpegif.org/mpeg4/. Acessado em agosto de 2007.

[17] ITU H.263 Video Compression. Disponível em: http://www.h263l.com/. Acessado em setembro de 2007.

[18] The Internet Engineering Task Force: Real Time Streaming Protocol. Disponível em: http://tools.ietf.org/html/rfc2326 . Acessado em setembro de 2007.

[19] The Internet Engineering Task Force: A Transport Protocol for Real-Time Applications. Disponível em: http://tools.ietf.org/html/rfc1889. Acessado em setembro de 2007.

[20] Spook: Linux Server Application to Capture Live Video and Audio and Stream it Over an IP Network. Disponível em: http://www.litech.org/spook/. Acessado em agosto de 2007.

[21] C. Kurashima, R. Yang, and A. Lastra, "Combining Approximate Geometry with View-dependent Texture Method – A Hybrid Approach to 3D Video Teleconferencing", Proc. of the 15Th Brazilian Symposium on Computer Graphics and Image Processing – *SIBGRAPI*, Fortaleza, CE, Brazil, pp. 112-119, 2002.

[22] Burlamaqui, Aquiles Medeiros Filgueira ; Oliveira, Marlos Marques ; Gonçalves, Luiz M G ; Souza Filho,

Guido Lemos ; Oliveira, Jauvane . “A Scalable Hierarchical Architecture for large scale multi-user virtual environments”. In: *IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems*,VECIMS, La Corunha, 2006

[23] Burlamaqui, Aquiles Medeiros Filgueira ; Souza Filho, Guido Lemos ; Fernandes, Jorge Henrique Cabral . “Vixnu - Um Servidor Multiusuário para Ambientes Virtuais Reconfiguráveis com Suporte à Comunicação Multimídia”. In: *Workshop de Teses e Dissertações WTD*, Salvador, 2003.

[24] Sun Microsystems. Java3D API. Disponível em: http://java.sun.com/products/javamedia/3D/download.html. Acessado em agosto de 2007.

[25] Edouard Lamboray, Stephan Würmlin e Markus Gross. “Real-Time Streaming of Point-Based 3D Video”. *IEEE Virtual Reality, 2004. Proceedings***,** pp. 4-5, Zurich, 2004.

[26] Tanenbaum, Andrew S., Distributed Operating Systems. 2ª ed. Prentice Hall: EUA, pp. 2, 1995.

[27] Keith Jack. Video Demystified - A Handbook for the Digital Engineer. 4ª ed. Newnes: EUA, pp. 560, 2005.

[28] The common object request broker: Architecture and specification, version 3.0. Object Management Group, July 2002.

[29] Digital TV, DVB and ATSC Tutorials - The Interactive TV Web. Disponível em: http://www.interactivetvweb.org/tutorial/dtv-intro/dtv-intro.shtml. Acessado em agosto de 2007.

[30] Tudor, P.N. “Mpeg-2 Video Compression”, 1995. Disponível em http://www.bbc.co.uk/rd/pubs/papers/paper_14/paper_14.shtml. Acessado em setembro de 2007.

[31] FFmpeg. Disponível em: http://ffmpeg.mplayerhq.hu/. Acessado em outubro de 2007.

[32] Ulrich Neumann, Thomas Pintaric, Albert Rizzo. “Immersive Panoramic Video”. *Proceedings of the 8th ACM International Conference on Multimedia*, pp. 493-494 Los Angeles, 2000.

Desenvolvimento de interface gráfica 3D para sistema de tomada de decisão

Antonio Valerio Netto, Mauro Enrique Munoz, Ricardo Guimarães dos Santos
Cientistas Associados Desenvolvimento Tecnológico Ltda.
{antonio.valerio}, {mnunoz}, {rsantos}@cientistasassociados.com.br

Abstract

The article presents the 3D development of a platform GIS (Geographical Information System) with 2D and 3D integrated interface. This platform is dedicated to decisions based in technical information and context on the distribution nets of urban electric power. This system can benefit the visualization of great amount of information in an interactive way in an environment of crossing of that information. Besides, it allows fast identification of the context where the electric elements of the electric power distribution system are and the spatialization of phenomena physical or natural. It also makes possible the accomplishment of analysis in an interactive virtual environment with possibility of decisions based in volume information and depth.

1. Introdução

Resultado da realização de um estudo de mercado com foco estratégico no produto no médio prazo foram identificados ferramentas que promovam soluções que utilizam SIG (Sistema de Informação Geográfica). Além disso, detectou-se um nicho de mercado não explorado no Brasil e América Latina que é o SIG 3D. Os SIG 3D permitem a criação de interfaces para aplicações que elevam a visualização geográfica para um nível mais alto da visão que um ser humano tem do mundo real.

Com a utilização de softwares de SIG que apresentam os contextos em três dimensões é possível uma interpretação mais ágil e precisa de uma grande quantidade de informações. Isso possibilita uma melhor compreensão das relações espaciais existentes entre os elementos analisados e facilita a visualização de situações complexas, cuja representação só seria possível por meio de um grande volume de mapas ou documentos. No caso das empresas concessionárias de distribuição de energia elétrica, o SIG tradicional representa uma poderosa ferramenta de gestão territorial. Contudo, a incorporação do SIG 3D permite uma melhor interação, sendo de grande valia em situações onde a visualização tridimensional oferece melhores recursos para a tomada de decisão e também em um melhor controle no monitoramento dos ativos aterrados previstos em recentes leis (mapeamentos subterrâneos, etc.).

O objetivo do projeto desenvolvido foi oferecer um novo aplicativo capaz de ajudar as concessionárias de distribuição de energia elétrica a planejar e gerenciar melhor suas operações. Para isto, foi criado um sistema computacional, que mostra imagens em três dimensões, baseado na tecnologia SIG (Sistemas de Informação Geográfica), que cria um ambiente virtual, interativo e georeferenciado na tela do computador dos operadores responsáveis pela gestão do sistema elétrico nacional.

O sistema denominado de ENS3D (Energy Network System 3D), fornece três ambientes de visualização. O primeiro, chamado de diagrama unifilar (DU) ou temático, é uma representação vetorial do sistema elétrico de determinada região. Ele permite uma visão precisa de todos os elementos da rede (postes, transformadores, chaves, cabos e linhas de transmissão), que são expressos por meio de círculos, quadrados e traços. O segundo ambiente (visão 2D) é um mapa de navegação em duas dimensões da cidade, que permite a localização geográfica do que está sendo representado no diagrama temático. O terceiro ambiente de visualização, a visão 3D, é uma maquete tridimensional da cidade. Com ela, o técnico tem uma representação real do lugar em questão. Além de contar com informações como latitude e longitude, ele consegue visualizar detalhes de sua topografia (elevações, depressões, etc) e das construções existentes na área. Os três ambientes são sincronizados e ao movimentar um dos ambientes os outros dois também se deslocam de forma georeferenciada.

A grande vantagem do sistema é que o mesmo facilita a visualização de situações complexas, cuja representação só seria possível por meio de um grande volume de mapas ou documentos. E, ao possibilitar uma melhor compreensão das relações espaciais existentes entre os

elementos do sistema elétrico analisados, permite, de forma interativa, que o usuário faça uma interpretação mais ágil e precisa de uma grande quantidade de informações. Um exemplo de decisão a ser tomada com base nas informações do ENS3D é o fechamento ou abertura das chaves que controlam o fluxo de energia elétrica pela rede. A identificação rápida do contexto espacial onde estão os elementos elétricos do sistema de distribuição de energia na cidade, como nome de ruas ou prédios importantes existentes na região, é essencial para uma decisão sem erros.

O sistema também pode auxiliar as distribuidoras de energia elétrica no trabalho de inspeção da rede e no combate às fraudes. Com a tecnologia contida no aplicativo é possível criar mapas temáticos com a representação dessas fraudes de modo que o gestor pode definir um plano de ação mais eficiente e de menor custo, considerando a distribuição espacial das ocorrências.

Em linhas gerais, o ENS3D é uma plataforma SIG com interfaces 2D e 3D integrados. Esta plataforma é dedicada à tomada de decisões baseadas em informações técnicas e contextuais sobre as redes de distribuição de energia elétrica urbana.

Neste artigo é focado o desenvolvimento da visão 3D do sistema ENS3D. Para isto, nas Seções 2 e 3 são apresentados, a plataforma SIG e o gerenciador da interface gráfica escolhidos para o projeto. Na Seção 4 é relatada a arquitetura do sistema Enviro que serviu de plataforma para o desenvolvimento da visão 3D. Na Seção 5 é detalhado o trabalho de *rendering* das geometrias no ambiente 3D desenvolvido. Por fim, na Seção 6 são relatadas as considerações finais.

2. Plataforma SIG

O Virtual Terrain Project (VTP) [1] tem como meta à criação de ferramentas para a reprodução de qualquer parte do mundo real em um ambiente virtual. A primeira versão desta biblioteca foi lançada em Março de 1997 reunindo esforços de vários desenvolvedores ao redor do mundo sob a constante e ativa participação de Ben Discoe. Foi uma biblioteca originalmente desenvolvida para a plataforma Windows.

A biblioteca VTP possui três módulos principais: vtui, vtlib e vtdata. O módulo vtui deriva funcionalidades e estruturas de dados do gerenciador de janelas wxWidgets. Com o vtlib é possível automatizar a construção e renderização interativa de terrenos 3D a partir de uma base de dados espaciais. O vtdata possui métodos de entrada e saída para dados espaciais, sistemas de coordenadas, tipos de dados abstratos e operações 2D e 2.5D (é uma aproximação bidimensional do 3D, isto é, ele tem características visuais do 3D, mas sua estrutura computacional é bidimensional). A vtdata possui quatorze grupos de classes distintos: Sistemas de coordenadas; Estruturas de construção; Vegetação; Estradas e transporte; Formatos de dados vetoriais e poligonais; Tipos abstratos para modelos matemáticos; Vetores; Mistos; log; Conteúdo; Elevação; Bitmaps; Características abstratas e Geo-codificadores.

O grupo de classes "Sistemas de coordenadas" possui duas classes que foram diretamente utilizadas pelo ENS3D: vtProjection e vtLocalConversion. A vtProjection representa um sistema de coordenadas na terra (CRS), com um sistema de coordenadas projetadas (PCS). Baseada na classe OGRSpatialReference que representa um sistema de referência espacial completo. A vtProjection estende a OGRSpatialReference com uma série de métodos. A vtLocalConversion representa o mapeamento entre coordenadas geográficas ou em algum sistema de projeção para um sistema de coordenadas 3D onde cada unidade do ambiente equivale a 1m do mundo real utilizando a convenção do eixo da mão direita adotado pelo OpenGL.

Os grupos de classes "Estruturas de construção", "Vegetação", "Estradas e transporte" possuem classes de alto nível para a renderização e armazenamento em memória de diversas geometrias dinâmicas e estáticas para o ambiente 3D. Algumas destas classes armazenam vetores com tais geometrias permitindo seu controle individual e dos grupos como um todo. Três classes são capazes de carregar níveis de informação a partir de arquivos com dados vetoriais; Tais classes não foram utilizadas pelo ENS3D que possui seus dados vetoriais armazenados em uma base de dados espaciais.

Classes definidas pela VTP para a manipulação de tipos abstratos de dados foram amplamente utilizadas e estendidas durante a implementação do sistema. O ENS3D utiliza-se da classe vtHeightField e suas extensões para o gerenciamento de qualquer coleção de superfícies associando coordenadas (x,y) a uma altitude. A vtContentManager foi manipulada durante a implementação da renderização dinâmica de modelos 3D, essa classe é capaz de ler arquivos Extensible Markup Language (XML) e carregar modelos 3D associando-os as classes que os representam no sistema, possibilitando seu georeferenciamento e renderização. A vtlib foi amplamente utilizada na implementação do sistema. Esta biblioteca é capaz de produzir e atualizar (rendering) automaticamente um terreno 3D interativo a partir de uma base de dados espaciais.

•Na computação gráfica é necessário que o programador estabeleça relações entre diversos sistemas de coordenadas. Os SIG's são especialmente sofisticados neste sentido, pois se deve considerar os sistemas de coordenadas da cartografia tradicional além dos sistemas gráficos 3D. Ainda não há um padrão bem estabelecido para se nomear estes diferentes sistemas de coordenadas, assim a vtlib usa a seguinte terminologia:

- Coordenadas de Terra: São os sistemas de coordenadas originais dos dados, em pés, metros, graus como aparecem em mapas tradicionais ou SIG's;
- Coordenadas de Mundo ou de Terreno: Cada terreno possui sua origem no nível do mar, no canto inferior esquerdo do ambiente 3D. Este sistema é interpretado diretamente pela vtlib/OpenGL, de forma a aproximar a unidade de medida metro à unidade do sistema de coordenadas tanto quanto possível;
- Coordenadas Locais: Cada característica ou construção digital específica do terreno possui seu próprio sistema de coordenadas e está na origem deste.

Baseada nestas convenções, a vtlib cria em tempo de execução uma classe que realiza transformações lineares para a mudança do sistema de coordenadas de terra para o de mundo. As características possuem métodos próprios que fazem as transformações lineares das coordenadas de mundo para coordenadas locais. As transformações lineares são processos computacionalmente caros, geralmente $O(n^3)$ ou mais, assim, metodologias que utilizem menos transformações são sempre bem vindas, neste sentido, as classes vtGeom e vtMesh são exemplos de classes que dispensam as transformações para coordenadas locais utilizando o sistema de coordenadas do OpenGL que é idêntico ao sistema de coordenadas de mundo.

A criação de um terreno virtual utilizando VTP passa pela construção de um modelo digital de terreno. Trata-se de uma das etapas mais dispendiosas do processo de implementação em um SIG 3D. Foi utilizada a classe vtTerrain, que é responsável pela leitura, renderização e gerenciamento do terreno digital.

A figura 1 ilustra as correlações entre o Sistema de Gerenciamento do Banco de Dados (SGBD), API's *(Application Programming Interface)* e GUI's do sistema desenvolvido. Fisicamente, o sistema pode usar um servidor de dados espaciais e uma estação de trabalho.

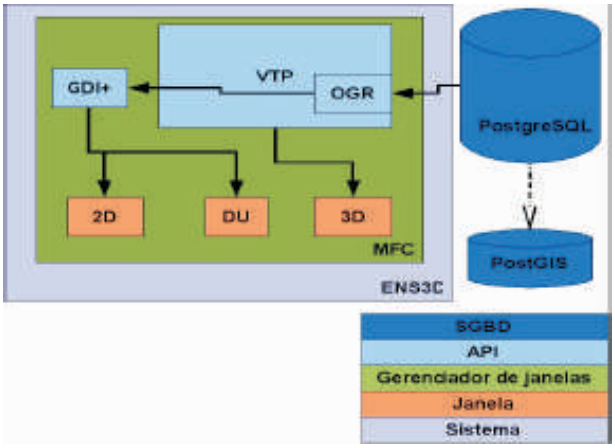


Figura 1. Arquitetura do ENS3D.

O banco de dados utilizado no projeto foi o PostgreSQL. Trata-se de um Sistema de Gerenciamento de Banco de Dados Objeto-Relacional (SGBDOR) baseado no PostgreSQL 4.2, desenvolvido no Berkeley Computer Science Department da Universidade da Califórnia. Pioneiro na implementação de muitos conceitos, ele permite ter suas funcionalidades estendidas pelo usuário que pode agregar: Tipos de dados; Funções; Operadores; Funções agregadas; Métodos de indexação e Linguagens procedurais.

A escolha do PostgreSQL justifica-se pela estabilidade que esse SGBDOR apresenta ao manipular bases de dados grandes. PostGIS é uma extensão para o PostgreSQL que permite o armazenamento de objetos SIG na base de dados. PostGIS suporta GiST-based (padrão para Árvores de Busca Generalizadas e forma genérica de indexação que aceleram buscas em estruturas de dados irregulares), R-Tree (estrutura de dados similar a Árvore-B utilizada por métodos de acesso a dados espaciais com indexação multi-dimensional), e funções para análise e processamento de objetos SIG.

3. Gerenciador da interface gráfica

Inicialmente, foi realizada uma análise de qual biblioteca de programação para GUI (Graphics User Interface) se adequaria melhor ao projeto. Tal estudo visou à escolha de uma tecnologia de desenvolvimento que atenuasse a curva de aprendizado, minimizasse o esforço de programação e a produtividade.

Dentre os aplicativos disponibilizados como exemplo pelo VTP, nota-se que sua grande maioria tem interface desenvolvida utilizando a biblioteca wxWidget, a qual é uma biblioteca de programação para GUI independente de plataforma. Existem várias bibliotecas que podem ser utilizadas no desenvolvimento de

interfaces para softwares que utilizem o VTP. Entre eles, se destacam:

• FLTK – Fast Ligth Toolkit: um conjunto de ferramentas GUI C++ multi-plataforma. Gerando uma GUI funcional com suporte a gráficos 3D via OpenGL. Além disso, inclui o construtor de interfaces chamado FLUID.FLTK;

• SDL – Simple Directmedia Layer: é uma biblioteca multi-plataforma projetada para prover acesso de baixo nível para áudio, teclado, mouse, joystick, hardware 3D via OpenGL e framebuffer 2D. Utilizada por players de MPEG, emuladores e muitos games populares como "Civilization: Call To Power";

• Open Producer: biblioteca multi-plataforma focada no controle de câmeras. A câmera do Open Producer provê projeções, campos de visão, controles de ponto de vista e controle de quadros. Esta biblioteca pode ser usada em um ambiente multitarefa habilitando múltiplas câmeras. Paralelamente, suporta configuração de hardware com múltiplas saídas. Threading, para a sincronização de câmera e controle de taxa de atualização de quadros, sendo estes simplificados pela interface do Open Producer.

• MFC – Microsoft Foundation Class: é uma biblioteca que agrega porções da API Windows em classes C++, formando um framework. Há definições e gerenciadores de classes para a maioria dos objetos Windows, bem como controles comuns pré-definidos.

• OpenGL: É um ambiente de desenvolvimento de aplicações gráficas interativas em duas e três dimensões.Gerenciamento do Banco de Dados (SGBD), API's (Application Programming Interface) e GUI's do sistema desenvolvido. Fisicamente, o sistema pode usar um servidor de dados espaciais e uma estação de trabalho.

Embora seja perfeitamente possível desenvolver toda interface do projeto utilizando OpenGL, tal tarefa demandaria a implementação de itens como botões, janelas, barras de menus e outros itens GUI, o que seria mais adequado com o emprego de outras bibliotecas mais apropriadas (MFC e wxWifgets). No Open Producer e no SDL tais implementações seriam delegadas ao OpenGL. O que não mudaria o quadro. Apesar de muito bem documentado, robusto e

difundido, utilizar o OpenGL para o gerenciamento de janelas não seria uma escolha que minimizaria o esforço de programação e nem maximizaria a produtividade.

No caso das bibliotecas MFC [2] e wxWidgets, as mesmas são gerenciadores de janelas robustos, possuem uma arquitetura semelhante. Em termos de produtividade não se pode afirmar que um seria mais produtivo que o outro. Porém, em termo de disponibilidade de material de referência e exemplos, o MFC, seria a biblioteca que mais atenuaria a curva de aprendizado. Um outro ponto positivo é a sua compatibilidade com a plataforma de desenvolvimento escolhida para o projeto, o Visual C++.

Na Internet, estão disponíveis templates para wxWidgets. Entretanto, as que são estáveis não têm releases livres nem gratuitos. Considerando o conjunto de fatores relatados anteriormente, optou-se pela utilização da biblioteca MFC para este projeto, no que se refere às interfaces.

Outra biblioteca utilizada no projeto foi a Geospatial Data Abstraction Library (GDAL) [3]. Trata-se de uma biblioteca de tradução para dados em formatos geoespaciais, ela apresenta um modelo de dados abstratos únicos que responde às chamadas das aplicações para todos os formatos suportados. A tarefa desempenhada pela GDAL serve de base para a grande maioria dos SIG's, suas funcionalidades são herdadas de algumas bibliotecas, entre elas está a OGR, biblioteca desenvolvida em código aberto C++ que permite a leitura de vários tipos de formatos de arquivos vetoriais, tais como ESRI Shapefiles, S-57, SDTS, PostGIS, Oracle Spatial, e Mapinfo (formatos mid/mif e TAB).

4. Estudo da arquitetura do Enviro

O Enviro é um ambiente executável que permite uma navegação 3D interativa em um terreno virtual. Este sistema possui dois modos de visualização: Earth View e Terrain View. O modo Earth View possibilita: visualizar o planeta a partir do espaço; navegação e zoom; exibição das extensões dos terrenos conhecidos, podendo carregá-los com um clique; mudança da iluminação solar; exibição dos dados sobre pontos e o traçado de arcos geodésicos.

O modo Terrain View permite: a carga de elementos para composição da cena; o vôo ou andar pelo terreno; navegar pelo grafo da cena; alterar configurações de câmera e renderização; alteração do nível de detalhes do terreno; adição de árvores, cercas, elementos de redes elétricas e construções ao terreno; edição e movimentação dos elementos do terreno e a gravação das alterações no terreno.

Em termos de design de software, o Enviro está organizado em quatro sub-projetos do Visual Studio C++ .net 2003: xmlhelper; vtosg; vtdata e mfcEnviro. O xmlhelper implementa um parser XML utilizado

pelos demais módulos do Enviro para a leitura e escrita de arquivos XML. Os arquivos de configuração do terreno utilizado pelo ENS3D e Enviro estão escritos em XML. São eles: conteudo.vtco; terreno.xml; construcoes.vtst e vegetacao.vtst. O processo de carga do terreno se dá com a seguinte seqüência:

· A leitura do arquivo conteudo.vtco gera no Enviro uma lista de modelos 3D, que podem ser utilizados pelo sistema;

· A escolha de um terreno força o sistema a carregar o terreno.xml. Este arquivo contém informações sobre diversas configurações do terreno, inclusive níveis de informação, texturas, animações. Essa leitura indica ao sistema quais arquivos *.vtst devem ser carregados;

· Nos arquivos *.vtst estão definidas coordenadas geográficas que indicam onde cada modelo deve ser plotado e uma referência para este modelo.

Durante a implementação do ENS3D, nenhuma alteração foi necessária neste sub-projeto. No subprojeto vtosg estão implementas as classes que compõem a biblioteca vtlib, responsável pela automação da renderização do terreno. O sub-projeto vtdata define tipos abstratos de dados. O sub-projeto mfcEnviro inter-relaciona e coordena os demais criando instâncias de suas classes. O diagrama da figura 2 é uma simplificação que visa exibir a arquitetura da parte do sistema utilizada nas adaptações para incluir o projeto Enviro no ENS3D.

Nesse diagrama é possível observar que a classe EnviroApp deriva da classe CWinApp. Esta derivação permite ao EnviroApp, funcionalidades básicas de aplicações MFC. Nota-se, adicionalmente, que EnviroApp possui em sua composição um objeto EnviroView, derivado da classe CView. Assim pode-se deduzir que a instância de EnviroApp, possui uma janela gráfica associada, onde podem ser representadas as geometrias geradas e gerenciadas pela instância de EnviroGUI.

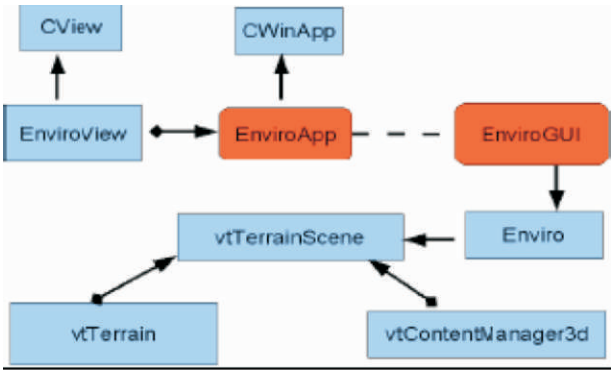


Figura 2. Arquitetura parcial do Enviro.

Por outro lado, EnviroGUI derivado da classe Enviro, que por sua vez, é derivado da vtTerrainScene, possui em sua composição um objeto vtTerrain que gerencia e atualiza (rendering) o terreno virtual, e um objeto vtContentManager3d que atualiza e gerencia geometrias 3D que caracterizam e povoam o terreno virtual. Tal arquitetura tem como principal vantagem a portabilidade, na necessidade de se implementar uma janela gráfica com outro gerenciador de janelas que não o MFC, pode-se gerar a EnviroApp como sendo derivada da classe geradora de aplicativos da biblioteca escolhida.

4.1. Integração entre o ENS3D e o Enviro

Em seu release mais embrionário, o ENS3D foi implementado com base no projeto mfcSimple disponibilizado entre os códigos fonte do VTP. Nesta fase de desenvolvimento foram incluídas classes e as funcionalidades do ambiente 2D e DU ao mfcSimple.

O mfcSimple original possuía apenas uma janela gráfica 3D, nesta fase, os esforços foram concentrados em modelar a interface e implementar as funcionalidades dos ambientes: visão 2D e DU, além de suas barras de ferramentas. Ao identificar as funcionalidades oferecidas pelo Enviro, mudou-se a estratégia. Iniciou-se um processo de agregar ao projeto Enviro as funcionalidades implementadas para os ambientes 2D e DU. Tanto o Enviro, quanto o ENS3D, utilizam a biblioteca de características simples OGR, entretanto, a OGR utilizada pelo ENS3D possui uma particularidade. Ela foi compilada com o intuito de acessar uma base de dados espaciais gerenciada pelo PostgreSQL, tal particularidade, levou ocasionou o primeiro problema da fase de integração. Ocorreram conflitos de nomes das funções disponibilizadas pela OGR. Foi necessária uma adaptação do Enviro para a utilização da OGR.

Com tais alterações concluídas e a nova versão do Enviro compilando corretamente, iniciou-se o processo de transferir para o Enviro, as classes que implementam as funcionalidades 2D e DU. Neste estágio, ocorreu outra dificuldade técnica. Existia tanto no VTP, quanto no Graphics Device Interface (GDI+), a definição de determinadas classes. Essas declarações causaram novamente conflito na compilação. Foi eleita uma classe menos abrangente entre as duas e foi realizado o renomeamento. O GDI+ é um sub-sistema do MS-Windows, devido a isto, alterar seu código poderia trazer comportamentos imprevisíveis ao sistema operacional, já no caso do VTP, tem-se total liberdade para remodelá-lo sem prejuízo do sistema operacional.

Desta forma, foram incorporados ao Enviro às funcionalidades implementadas para as janelas 2D e DU do sistema, nesse ponto, o ENS3D passa a ter todas as funcionalidades do Enviro agregadas, entretanto mudanças na arquitetura desse sistema embrionário ainda seriam necessárias para a interação entre as janelas gráficas. Neste estágio de desenvolvimento, a arquitetura do sistema gerenciava individualmente o acesso à base de dados espaciais, replicando a base em memória de forma redundante. Foram realizadas as integrações dos módulos de visualização e interação do sistema.

A reestruturação do projeto trouxe as seguintes melhorias ao sistema:

• Todo o acesso à base de dados é realizado exclusivamente pela classe *EnergiaDao*. Dessa forma, o controle sobre as conexões com a Base de Dados (BD) passou a ser centralizado por essa classe. Também, apenas essa classe conhece as estruturas da base de dados fazendo com que as alterações na BD não se espalhem pelo restante do código. Outra melhoria foi à criação de um cache interno à classe. Assim, apenas os dados realmente alterados na base são relidos, de forma transparente, pelo módulo;

• Isolamento entre as vistas (classes *ViewDU*, View2D e View3D) e as ferramentas gráficas (classe *Form*). A classe *Frame* passou a centralizar a comunicação entre estes objetos. Sem a dependência explícita entre a *Form* e as vistas pode-se ativar, desativar ou até criar uma nova vista sem modificar o objeto *Form*, tornando o módulo mais extensível;

• Centralização pelo *Frame* das atividades independentes das vistas. Pelo projeto antigo, atividades como a busca de endereços, que não dependiam da forma de visualização do modelo, deveriam ser implementadas separadamente em cada uma da vista ou em cada um dos *Forms*. Com o novo projeto, esta e outras atividades similares são disparados pelo *Form* e implementada pelo *Frame*;

• Unificação dos três *Forms* em uma única classe. Ao eliminar das vistas as atividades independentes da visualização ativa, descobriu-se que se poderia utilizar uma única classe *Form* para realizar todo o trabalho de interfaceamento das ferramentas da interface do módulo;

• Hierarquização das vistas. Com a criação de ViewPlane praticamente toda a renderização e acesso à base dos dados bidimensionais pôde ser implementada em uma única classe fazendo de ViewDu praticamente um caso restrito da View2D. Da mesma forma, a criação da classe ViewBase, utilizada como base para todas as vistas do módulo, permitiu a codificação unificada de atividades como: sincronização entre as vistas, mudança da ativação das vistas e captura do evento para a visualização das informações de um elemento gráfico.

5. Rendering de geometrias no ambiente 3D

Quando o ENS3D é inicializado, efetua-se a leitura do arquivo chamado Sanca.xml que está armazenado em um diretório pré-determinado. Este arquivo possui parâmetros de configuração do terreno como hora, dia, mês e ano. Essas informações são importantes para que o ambiente posicione a fonte de luz solar, os *paths* para arquivos de textura do terreno e do globo celeste. Todos esses parâmetros são utilizados na criação de um objeto da classe SancaTerrain, derivado da classe vtTerrain.

O arquivo Sanca.xml possui um *path* para um arquivo de referência a conteúdo (*.vtco). Este arquivo armazena parâmetros de configuração para o *rendering* de modelos 3D que podem ser agregados a qualquer terreno virtual, além de carregar instâncias destes modelos para um objeto vtContentManager3d que consiste em um vetor de modelos 3D previamente construídos e seus parâmetros de *rendering*.

A grande maioria das geometrias representadas neste ambiente tem como origem dados estáticos, como modelos 3D de elementos elétricos, prédios históricos ou construções importantes. Tais modelos foram modelados utilizando o software 3D Studio Max. Após sua modelagem, estas construções digitais foram georeferenciadas para a compilação de arquivos no formato "*.vtst".

Os arquivos *.vtst são usados pelo VTP para a definição de níveis de informação como, vegetação e construções. Baseados nesses dados, o ENS3D faz apenas a carga dos elementos estáticos do ambiente 3D e não possibilita qualquer alteração dos mesmos.

Com a utilização do Enviro como ponto de partida para a implementação do ENS3D, herdou-se uma série de funcionalidades. Entretanto, o sistema ainda necessitava de adaptações para o total cumprimento dos seus requisitos.

Antes desta mudança na arquitetura do ENS3D, existiam dois requisitos a implementar, o primeiro deles era o *rendering* das linhas que representassem os cabos da rede elétrica dividindo-as em grupos separados por cores que se relacionassem com seus respectivos alimentadores. Três alternativas técnicas foram

propostas para tal *rendering,* a primeira delas representava os alimentadores com uma camada de informação estática, essa alternativa não cumpria a função de representar os alimentadores ao longo do tempo, pois à medida que o ENS3D é utilizado, novas configurações são geradas na rede de distribuição, alterando o desenho dos alimentadores.

A segunda alternativa foi utilizar a classe *vtRoute*. Esta classe busca um modelo previamente carregado em um vetor *vector<vtUtilNode*>* que é formado durante a carga do sistema com a leitura do arquivo *conteudo.vtco* e traça automaticamente os cabos que conectam estes modelos. O **.vtco* utiliza um padrão mnemônico que dispensa explicações sobre seus parâmetros. Esta metodologia é bem elegante, supri o requisito desejado, entretanto possui um demanda um elevado custo computacional. A carga de um único alimentador baixava a velocidade de *rendering* para menos de 0,5 frames por segundo (fps). Tal velocidade tornou inviável a utilização desse método.

Por fim, a terceira metodologia adotada consistiu em utilizar primitivas do OpenGL diretamente. O VTP já possuía a classe MyGeom que é capaz de gerenciar e renderizar geometrias dinamicamente no ambiente do ENS3D. Mas, a MyGeom e a própria *vtlib* tinham de sofrer alterações para a implementação de tal requisito.

A MyGeom deveria ser capaz de recuperar novas configurações da rede elétrica com uma consulta à base de dados espaciais, tal consulta dá-se de forma transparente com a utilização do objeto *EnergiaDao*, que encapsula a base de dados e todas as consultas do sistema. O *EnergiaDao* possui o método polimórfico *getLayer*, uma de suas implementações retorna da base de dados um vetor *LineSe*t, que consiste de um vetor de linhas com ângulos associados a cada vértice. Os pontos do *LineSet* precisam ainda passar por algum processamento pois estão em coordenadas de terra que precisam ser convertidas em coordenadas de mundo. Tal tarefa foi desempenhada por um objeto vtLocalConversion.

Antes do *rendering* ainda existe um detalhe, a base de dados espaciais do ENS3D é bi-dimensional, MyGeom então deveria ser capaz de encontrar a altura do terreno em uma determinada coordenada 2D, para tal, a MyGeom precisou de um ponteiro para o terreno, esse ponteiro recebia sua atribuição a partir da execução do método MyGeom::SetTerreno.

Com esta atribuição, a MyGeom passou a ter um ponteiro para o terreno que por sua vez possui em sua composição, um objeto da classe vtHeigthField3d. Esta classe é derivada da classe *vtHeigthField* que possui o método *vtHeigthField::FindAltitudeOnErath*. Originalmente, este era um método privado, o que impedia que a MyGeom o executasse. Dessa forma, a classe *vtHeigthField* foi redefinida para que suportasse tal chamada simplesmente tornando público esse método. A MyGeom ficou pronta para extrair da base de dados espaciais, as linhas que formam o alimentador dinamicamente. Havia, entretanto, mais um requisito. Linhas paralelas à linha retornada pelo método *EnergiaDao::getLayer* deveriam ser desenhadas conectando cada elemento da rede de distribuição elétrica. Para suprir esse requisito foi implementado na classe MyGeom, o método *MyGeom::Paralelas* que calcula as paralelas tomando como base, o ângulo de rotação de cada elemento elétrico da rede.

Neste estágio, um último pré-requisito faltava para ser implementado. Tratava-se do desenho dinâmico das representações de chaves abertas e fechadas da rede elétrica. A primeira metodologia adotada foi capturar os parâmetros a partir de um arquivo *.vtst, mas como apresentado anteriormente, este tipo de metodologia não propiciava a atualização do estado das chaves em constante mudança durante a execução do sistema. Para a implementação alterou-se mais uma vez a vtlib incluindo na classe *vtTerrain*, o método *vtTerrain::PlantKeys*, que difere do método *vtTerrain::PlantModelAtPoint* no tocante à altura relativa onde a geometria é desenhada. Adicionalmente, foi criado na classe *MyGeom*, o método *MyGeom::Chaves* e os vetores de objetos da classe *vtTransform*. Estes vetores representam os repositórios onde ficam armazenadas todas as instâncias gráficas geradas pelo método *MyGeom::Chaves* permitindo um total gerenciamento destas geometrias.

A utilização dessa técnica conduziu a mais uma alteração na vtlib. Quando um objeto *vector<vtTransform*>* chamava o método *vector<vtTransform*>::clear*, o mesmo realizava indiretamente uma chamada ao método *vtTransform::~vtTransform*. Esse método, originalmente privado, não permitia este acesso, diante disso foi redefinida a classe *vtTransform* para permitir a destruição dos objetos *vtTransform*.

O método *MyGeom::Chaves* verifica se os vetores vector<vtTransform*> estão vazios, caso estejam, faz-se uma chamada ao método *EnergiaDao::getLayer* que retorna um vetor PointSet contendo um vetor de pontos em coordenadas de terra. Estas coordenadas de terra devem sofrer as devidas conversões para coordenadas de terreno utilizando a classe vtLocalConversion. Para encontrar a informação

"altitude" em cada coordenada, deve-se realizar uma chamada ao método *vtHeigthField::FindAltitudeOnErath.* Renderiza-se os cones com o método *vtMesh::CreateConicalSurface,* e armazena-se as referências para estas geometrias no vetor *vector<vtTransform*>*, para posterior destruição. No caso do vetor não estar vazio, o mesmo processo repete-se após a limpeza do vetor *vector<vtTransform*>*. O resultado obtido está ilustrado na figura 3.

As informações para a visão 3D são obtidas por duas estruturas de dados, a primeira delas fornece um vetor de pontos com as coordenadas geográficas das chaves. A segunda estrutura consiste em um vetor de linhas georeferenciadas contendo um ângulo associado a cada vértice da linha. Por intermédio dessas informações, é possível desenhar as chaves e as linhas para os postes conforme figura 4. Onde, os cones simbolizam as chaves, sendo vermelho para chaves abertas, e verde para chaves fechadas. É importante salientar que apenas as chaves e os alimentadores são acessados via DB, as demais informações como terreno, imagens aplicadas como textura, postes, edifícios e edifícios com textura são informações estáticas herdadas das técnicas empregadas no Enviro.

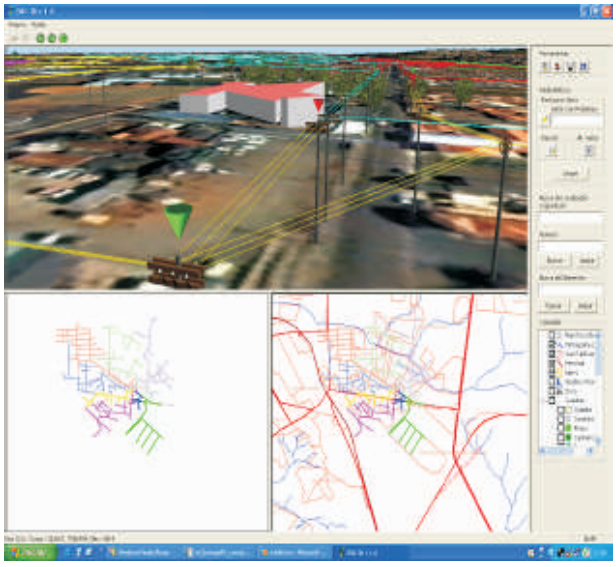


Figura 3. *Rendering* dinâmico das chaves.

6. Considerações finais

No desenvolvimento do projeto proposto foram exploradas várias bibliotecas de código aberto (GDAL, conjunto de bibliotecas do projeto VTP, etc), o que proporcionou uma rica pesquisa e a descoberta de alternativas variadas de desenvolvimento. Contudo, no campo do SIG 3D existem poucos recursos de códigos livres disponibilizados na Internet, principalmente, códigos para o sistema operacional Windows.

O processo de incorporação da plataforma Enviro para a implementação da visão 3D proporcionou uma reutilização de códigos e técnicas. Para tal incorporação, foi necessária uma demanda de tempo para entender a sua estrutura de classes e objetos, além de saber como todos se comportavam. Este estudo proporcionou a integração do projeto ENS3D ao Enviro e a reutilização das técnicas desenvolvida pelo projeto Enviro. Para o entendimento dessas técnicas foi consultado o site da VTP, no qual disponibiliza documentações a respeito do projeto. É valido salientar que no decorrer dessa incorporação ocorrerão diversos problemas, como por exemplo, conflitos de classe, no qual constatou que diferentes bibliotecas estavam utilizam classes com o mesmo nome, mas com funcionalidades diferentes.

Os resultados obtidos pela equipe do projeto tiveram boa repercussão junto à comunidade internacional que desenvolve o VTP. O ENS3D ganhou, inclusive, uma pagina no site da organização [4]. O modelo digital 3D do terreno será disponibilizado, posteriormente, possibilitando que outros desenvolvedores utilizem-no em seus sistemas. Entende-se que o projeto também contribuiu com a idéia da organização mantenedora do VTP que é fornecer um mundo virtual por meio do qual outros usuários e desenvolvedores possam fazer suas pesquisas e aplicações.

O VTP oferece módulos para a visualização estereoscópica por meio do qual pode-se desenvolver uma interface imersiva. No caso do ENS3D, o cliente final não apresentou interesse em utilizar a visualização estérea. Diante disso, não foi focado o desenvolvimento dessa funcionalidade no projeto. O desenvolvimento de uma interface com estereoscopia implicaria em uma nova análise para o desenvolvimento de um layout para o sistema, além dos estudos necessários para a implementação da Interface Homem-Computador (IHC) em ambientes imersivos por meio de equipamentos como luvas e capacetes de realidade virtual.

Com a finalização do projeto ENS3D foi gerado um instalador. Para a criação desse instalador foi necessário fazer um estudo de softwares livres disponíveis que teria essa funcionalidade. Assim foi adotado o ISTool por proporcionar uma interface gráfica facilitando e agilizando o estudo, pois os demais encontrados não possuíam nenhuma interface gráfica. Após a geração do instalador, o mesmo foi executado em uma máquina com a seguinte configuração: processador Pentium 4,

CPU 3.00 Ghz; memória RAM de 3.00GB e placa de vídeo: GeForce 6600 LE 256MB.

É valido ressaltar que o sistema desenvolvido abordou apenas uma cidade (São Carlos/SP) no qual foram implementados alguns edifícios. De modo geral, foi gerada parte da rede elétrica para validar a metodologia desenvolvida. Assim se for utilizada essa metodologia em uma grande região com bastante detalhamento de toda essa região em 3D, será necessário à demanda de grande processamento, dessa forma, será importante adotar outros mecanismo de processamento, com, por exemplo, um cluster de PCs.

Além da aplicação no setor elétrico, o software também pode ser utilizado para outras finalidades como planejamento urbano, otimização do sistema viário, análise de impacto ambiental, planejamento de infra-estrutura e monitoramento de sistemas de água e esgoto. É importante ressaltar que o sistema foi concebido de forma a permitir sua integração com outros aplicativos, como por exemplo, da área de gestão que já existam em empresas privadas ou instituições governamentais.

7. Agradecimentos

Os autores agradece o apoio financeiro da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) por meio do programa de Inovação Tecnológico em Pequenas Empresas (Processos: 02/07862-3 e 03/10954-0). Os autores também agradece o auxílio do CDCC/USP, CPFL, secretaria municipal de planejamento urbano e de habitação do município de São Carlos (SP).

8. Referências

[1] Virtual Terrain Project. Disponível em: http://www.vterrain.org Acesso em: 20/08/2006.

[2] MFC Reference (MFC). MFC Library Reference. Disponível em: http://msdn2.microsoft.com/enus/library/d06h2x6e(VS.80).aspx Acesso em: 15/08/2006.

[3] GDAL. Disponível em: http://www.gdal.org/ Acesso em: 15/08/2006.

[4] ENS3D. Projeto ENS3D junto a comunidade VTP http://vterrain.org/Packages/Uses/ENS3D.html. Acesso em: 10/06/2007.

ARMsg: Mensageiro Digital usando Realidade Aumentada

Jan van der Haas Habib, Alberto Raposo
Tecgraf, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro
janvdh69@gmail.com, abraposo@tecgraf.puc-rio.br

Abstract

This paper introduces the ARMsg, an instant messenger using augmented reality. The ARMsg allows that the user send instant messages accompanied by a 3D avatar to be visualized in the camera of the remote user (receiver).

Resumo

Este artigo apresenta o ARMsg, um mensageiro instantâneo utilizando realidade aumentada. O ARMsg permite que o usuário envie mensagens instantâneas acompanhadas de um avatar (modelo 3D) de sua escolha, para serem visualizadas na imagem da câmera do usuário com quem está se comunicando.

1. Introdução

Ambientes Colaborativos de Realidade Virtual (RV) e Realidade Aumentada (RA) abrem uma nova perspectiva para a colaboração em grupo, ao possibilitar que seus participantes interajam através da simulação de um mundo real ou imaginário ou através da manipulação de objetos virtuais no mundo real.

A operação em tempo real característica da RA, reflete-se no uso freqüente de comunicação síncrona nestes ambientes. Nos casos em que os usuários estão co-localizados, a comunicação é realizada diretamente, sem necessidade do ambiente para intermediá-la. No caso de se ter interlocutores fisicamente distantes, audio-conferência e vídeo-conferência têm sido a solução adotada. Observa-se, porém, que na áudio e na videoconferência, a troca de mensagens não faz uso de objetos virtuais, isto é, nestes casos a Comunicação não é baseada na tecnologia de RA.

Um exemplo clássico sistema de comunicação de RA propriamente dito é apresentado no filme Guerra nas Estrelas de George Lucas. Neste filme, os personagens conversam com interlocutores remotos apresentados como uma imagem virtual de corpo inteiro e em tamanho real. Um sistema de videoconferência RA que segue esta abordagem é apresentado em [3]. Neste sistema, o rosto do interlocutor remoto é apresentado como um objeto virtual sobre um marcador. Vários interlocutores, móveis ou em desktops, podem participar da conferência simultaneamente. Os marcadores funcionam como monitores virtuais, com a vantagem de poderem ser rearrumados livremente na mesa de trabalho e de serem carregados por usuários móveis.

É notável a evolução dos meios de comunicação nas últimas décadas, no entanto ainda não contamos com um método que possibilite o envio de objetos 3D enquanto nos comunicamos. O presente trabalho visa possibilitar que usuários troquem mensagens e enviem avatares que são visualizados, através da RA, pelo interlocutor em cima de um marcador, através da webcam.

Os ambientes colaborativos de RA estão fortemente associados à Cooperação (no sentido do modelo 3C [5]). Na essência da tecnologia RA está a apresentação e manipulação de objetos virtuais no mundo real, provendo o suporte necessário para que a operação conjunta no espaço compartilhado (Cooperação) ocorra. Mais que isto, este espaço compartilhado é o mundo real e a manipulação dos objetos não é necessariamente delimitada por uma tela de monitor nem precisa ficar presa em torno de uma mesa, abrindo novas possibilidades de suporte à Cooperação.

Do ponto de vista da comunicação, já existem algumas soluções de RA, especialmente para situações síncronas. Para usuários co-localizados, um exemplo é o trabalho de Nakashima et al. [6], que apresenta uma aplicação baseada em interação table-top. Para usuários remotamente localizados, há exemplos de sistemas de videoconferências usando recursos de Realidade Mista, como os trabalhos de Barakonyi et al. [2] e Regenbrecht et al. [9]. O primeiro usa marcadores para inserir objetos virtuais em streams de vídeo convencionais de videoconferência (Figura 1). O segundo é, na verdade,

Figura 1. Screenshot do sistema de videoconferência com RA apresentado em [2].

uma aplicação de virtualidade aumentada, inserindo os streams de vídeo em um ambiente virtual.

O trabalho aqui apresentado usa conceitos de sistemas colaborativos, RA e Computação Gráfica para o desenvolvimento do ARMsg, uma espécie de Windows Live Messenger em RA. A idéia é que os usuários conectados possam trocar mensagens cujo conteúdo sejam objetos (gráficos e textos) virtuais, que serão vistos pelo destinários sobre o marcador.

Com o programa desenvolvido, espera-se criar um sistema de comunicação tanto síncrono quanto assíncrono (a mensagem é armazenada em arquivo, para posterior visualização) usando recursos de RA, que poderá agregar características multimídia à mensagem (animação, emoção, áudio, etc.).

2. ARMsgr

O objetivo principal do projeto é desenvolver uma espécie de "Mensageiro Instantâneo em RA", onde o remetente cria um avatar e escreve uma mensagem. O ARMsg possui duas funcionalidades diferentes: remetente e destinatário. O remetente é uma área de edição, onde o usuário escolhe o avatar (dentro de uma lista de objetos previamente desenhados no 3DS Max e exportados para .IVE) e redige a mensagem, gerando o arquivo com a animação e texto a ser transmitido. O destinatário é um visualizador de RA que reproduz a animação (mensagem) gerada sobre o marcador.

Na versão atual, o programa conta com uma biblioteca de modelos 3D que podem ser enviados para o interlocutor. Futuramente, espera-se disponibilizar um repositório de modelos na Internet, para onde os usuários poderão enviar artes 3D próprias, que serão baixadas por todos os usuários ao atualizarem a biblioteca do programa.

O que diferencia este programa das abordagens comuns em comunicação utilizando RA para usuários remotamente localizados, como as apresentadas na seção anterior, é o fato de que no ARMsg, o conteúdo da mensagem é o objeto virtual transmitido entre os usuários. Nas abordagens baseadas em videoconferência, o que é transmitido é o stream de vídeo, sendo os objetos virtuais incluídos localmente. Ou seja, nesses casos, a mensagem em si é o vídeo.

2.1. Arquitetura

A biblioteca OSG [7] é utilizada para construir o grafo de cena e para desenhá-lo, utilizando o OpenGL. A biblioteca ARToolkit [1] é utilizada para capturar a imagem da câmera e fazer o registro espacial, isto é, fazer o reconhecimento de marcadores na imagem da webcam, que serão usados como referência para desenho do objeto virtual. Para que pudesse integrar o grafo de cena do OSG com as funções do ARToolkit, foi necessária a utilização da biblioteca OSGART (OSG

+ ARToolkit) [8]. Um diagrama de como estas bibliotecas se comunicam é mostrado na Figura 2.

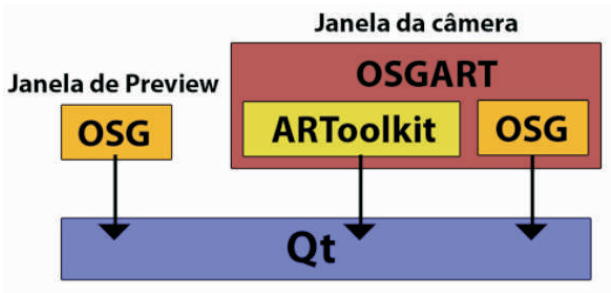


Figura 2. Diagrama de integração entre as bibliotecas.

Resumidamente, o ARToolkit calcula matrizes de trasformação para o marcador reconhecido. O OSG usa nós de transformação para posicionar objetos 3D na cena. O OSGART mapeia as transformações do ARToolkit para os nós de transformação do OSG.

A biblioteca Qt é utilizada para a criação da interface gráfica (GUI).

2.2. Caracterísitcas do sistema

O sistema desenvolvido permite que o usuário envie mensagens instantâneas acompanhadas de um avatar (modelo 3D) de sua escolha, para serem visualizadas na imagem da câmera do usuário com quem está se comunicando.

A janela principal do programa (Figura 3) permite que o usuário escolha um avatar, escreva sua mensagem e altere a fonte, cor e tamanho do texto. O usuário pode visualizar o estado em que sua mensagem está após qualquer modificação através da janela de preview. Também na janela de preview, o usuário pode manipular o avatar, para melhor visualização.

Junto com a janela principal, outra janela se abre quando o usuário executa o programa. Esta é a janela da câmera, onde o usuário visualiza as mensagens por ele recebidas em cima de marcadores. Nela, o usuário pode visualizar as mensagens enviadas pelo usuário a quem está conectado ou apenas visualizar algum arquivo de mensagem do ARMsg (.arm) já salvo em seu PC (Figura 4).

2.3. Módulos desenvolvidos

Durante a implementação do sistema, diversos módulos foram desenvolvidos. Esta seção tem por objetivo explicar o funcionamento dos principais módulos desenvolvidos. Todos os módulos herdam da classe base do Qt, a QObject. Isto é neceessário para que eles possuam a habilidade de enviar sinais indicando que determinadas ações foram realizadas.

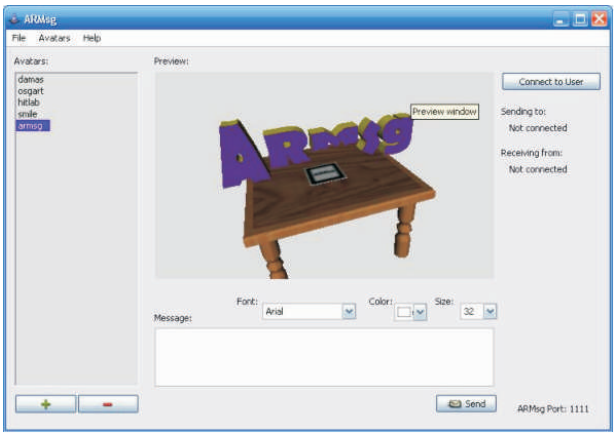


Figura 3. Janela principal do ARMsg.

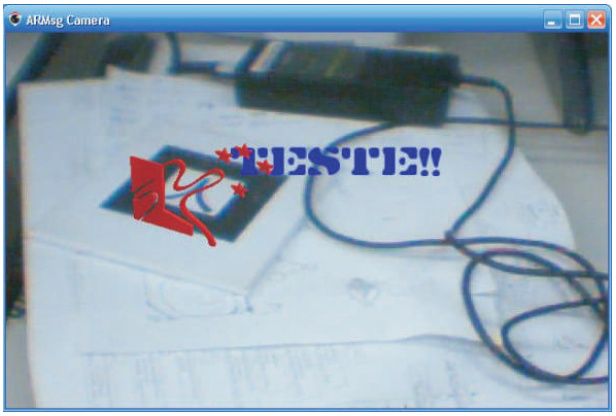


Figura 4. Janela da câmera do ARMsg.

Estes sinais são associados a slots (funções que são chamadas quando o sinal associado a ela é ativado), para que este faça o procedimento necessário.

O módulo AddAvatarDialog herda da classe QDialog do Qt para a criação de uma janela de diálogo onde o usuário digita o nome do modelo que quer adicionar à sua lista de modelos, sem a necessidade da extensão ".ive". Só são aceitos modelos que estiverem contidos na pasta padrão de modelos, ou seja, o usuário deve copiar o modelo que quer carregar para esta pasta e assim pode adicionar este modelo para a lista.

O módulo ConnectTo_Dialog herda da classe QDialog do Qt para a criação de uma caixa de diálogo para que o usuário digite o nome ou endereço IP da máquina com a qual quer se conectar. Caso deixe o espaço em branco e clique em OK, o usuário se conecta a ele mesmo, podendo ver as mensagens enviadas por ele na sua janela de câmera.

O módulo OSGWidget herda da classe QGLWidget do Qt para adaptar os eventos do OSG com os eventos de interface do Qt, permitindo assim usar uma janela do

OSG dentro da interface do Qt, como é o caso da janela de preview. Foi implementado usando o sceneview do OSG. Este módulo não foi escrito pelos autores, mas ele é utilizado no módulo descrito abaixo.

O módulo PreviewOSGWidget é responsável pela janela de preview e herda da classe OSGWidget. Possui todas as informações pertinentes à janela de preview. Contêm variáveis com seus nós de texto e modelo, bem como funções para criar o grafo de cena e atualizar as informações do texto (fonte, cor, tamanho) e do modelo. Ela reimplementa algumas funções da classe OSGWidget para que se adapte ao propósito da janela de preview, como no caso da função paintGL(), que foi alterada para que a cena comece em cima do modelo e não no meio, como é o da OSGWidget.

O módulo AROSGWidget é responsável pela janela da câmera e herda da classe QGLWidget do Qt para adaptar os eventos do OSG com os eventos de interface do Qt. Possui todas as informações pertinentes à janela da câmera. Contêm variáveis com seus nós de texto e modelo, bem como funções para criar o grafo de cena e atualizar as informações do texto (fonte, cor, tamanho) e do modelo. Este módulo é baseado no módulo OSGWidget, mas é implementado usando o osgProducer::Viewer. O viewer possui um loop próprio de execução, portanto este foi passado para a função paintGL() para que o loop updateGL() fique redesenhando a tela constantemente e, no caso, atualizando a posição do observador. O grafo de cena da janela da câmera possui nós com o vídeo e o marcador, bem como o texto e os modelos.

O módulo ARMsg herda da classe QMainWindow do Qt para a criação da janela principal do programa. Este módulo possui as funções de manipulação da interface, a parte de conexão cliente-servidor, a parte de persistência em arquivos, bem como todos os caminhos e inicializações padrões.

Para salvar ou enviar mensagens, o aplicativo salva as informações da mensagem em um arquivo com extensão .arm (Figura 5).

Assim que o aplicativo executa, ele funciona como um servidor conectado à porta padrão do ARMsg (1111). O cliente é usado para escutar o servidor do outro usuário. Ou seja, cada aplicativo possui um cliente e um servidor. Para o usuário 1 receber mensagens do usuário 2, ele deve se conectar ao servidor do usuário 2 e o mesmo procedimento se aplica para o usuário 2.

O programa foi testado, basicamente, utilizando quatro cenários de teste: usuário salva e abre uma mensagem

```
MSG:
Cow parade!!!!!
--end--

FONT:
fonts/Comic Sans MS.ttf

SIZE:
100

COLOR:
1
0.5
1
1

MODEL:

models/vaquinha.ive
```

Figura 5. Formato do arquivo .arm

offline, usuário troca mensagens consigo mesmo, usuário troca mensagens através da internet, usuário troca mensagens através de uma rede local. Em todos os casos, o aplicativo funcionou conforme esperado.

3. Conclusões e trabalhos futuros

O projeto aqui apresentado revela uma possibilidade ainda pouco explorada de utilização da tecnologia RA, abrindo um novo campo de estudo que pode ser explorado por diversas vertentes em trabalhos, incorporando RA nas comunicações via internet e até mesmo em outros campos de entretenimento.

Como trabalho futuro, pretende-se modificar a forma de comunicação entre os usuários, abandonando-se o modelo atual de ligação de máquina com máquina para se adotar um servidor central, que receberá todas as mensagens e as encaminhará ao destinatário final, assim como o popular Windows Live Messenger.

Ainda quanto aos avanços que se pretende na arte, espera-se utilizar a biblioteca Cal3D [4], tornando possível que os avatares sejam animados por esqueleto e que possuam movimentos faciais desenvolvidos com Morph Targets, simulando, por exemplo, a fala.

4. Referências

[1] ARToolkit - http://www.hitl.washington.edu/artoolkit/. Acesso em 26/11/2007.

[2] Barakonyi, I., Frieb, W. and Schmalstieg, D. (2003) Augmented Reality Videoconferencing for Collaborative Work. *Proc. of the 2nd Hungarian Conference on Computer Graphics and Geomtery*, Budapest, Hungary, May 2003.

[3] Billinghurst, M. and Kato, H. (2002) "Collaborative Augmented Reality". *Communications of the ACM*, Vol. 45, No. 7, July, pp.64 – 70

[4] Cal3D - http://home.gna.org/cal3d/ Acesso em 26/11/2007.

[5] Filippo, D., Raposo, A. B., Endler, M., Fuks, H. (2007) "Ambientes Colaborativos de Realidade Virtual e Aumentada". In: C. Kirner e R. Siscoutto (eds.), *Realidade Virtual e Aumentada: Conceitos, Projeto e Aplicações – Livro do Pré-Simpósio IX SVR*, Cap. 9, pp.168-191. Ed. SBC, Porto Alegre, Brasil.

[6] Nakashima, et al. (2007). "A 2D-3D integrated tabletop environment for multi-user collaboration". *Computer Animation and Virtual Worlds, Volume 18*, *Number 1*, February, John Wiley & Sons., pp. 39-56.

[7] OSG - www.openscenegraph.org. Acesso em 26/11/2007.

[8] OSGART - http://www.artoolworks.com/ community/osgart/. Acesso em 26/11/2007.

[9] Regenbrecht, H. et al. (2003) An Augmented Virtuality Approach to 3D Videoconferencing. *Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR '03)*, pp. 290-291.

Laboratórios Remotos com Sistemas Hápticos para Educação à Distância

Thais Pereira[1], Bruno Sales[1], Daniel Souza[1], Liliane Machado[1], Joaquim Gabriel[2], Maria Teresa Restivo[2], Antonio Lopes[2], Ronei Moraes[1]

[1]LabTEVE – Universidade Federal da Paraíba (Brasil)
{tata_burity}, {brunorasales}@gmail.com, liliane@di.ufpb.br, ronei@de.ufpb.br,
[2]Faculdade de Engenharia da Universidade do Porto (Portugal)
{jgabriel}, {trestivo}, {aml}@fe.up.pt

Abstract

Laboratories are an important educational resource to provide experimentation and help in the knowledge construction. Remote laboratories allow experimentation at distance and offer access to education. This work presents the development of an application that integrates haptics to remote laboratories to improve experiments whose comprehension involves mechanical material characterization.

Resumo

Laboratórios são um importante recurso educacional cujo objetivo é oferecer experimentação e auxiliar na construção e fixação de conhecimentos. Os laboratórios remotos permitem experimentação à distância e oferecem acesso à recursos educacionais. Este trabaho apresenta o desenvolvimento de uma aplicação que integra sistemas hápticos a laboratórios remotos para enriquecer a realização de experimentos cuja compreensão envolve a caracterização de materais mecânicos.

1. Introdução

Um dispositivo háptico é uma interface homem-máquina, capaz de estimular o sentido do tato do operador. Tal dispositivo pode ser utilizado para interagir com ambientes reais ou virtuais. Em geral, no primeiro caso, o ser humano utiliza o dispositivo háptico como master, numa aplicação do tipo master/slave com feedback de força (ex. telemanipulação ou teleoperação). No segundo caso, a interação ocorre com uma aplicação computacional (ex. jogos, simuladores) [1]. O uso de tais dispositivos permite aumentar o grau de imersão e envolvimento dos usuários na execução de diferentes tarefas.

Laboratórios remotos são laboratórios reais cujos experimentos podem ser realizados à distância através da web, permitindo a alunos interagir com equipamentos e objetos reais. Em geral, laboratórios remotos não oferecem meios para os alunos sentirem o resultado de suas ações e apenas imagens permitem perceber tais resultados [2].

Apesar da existência de diversos cursos na web com experimentos em laboratório, várias são as discussões a respeito do uso da tecnologia neste contexto. Atualmente, muitos trabalhos discutem esta questão e defendem mesmo um grau idêntico de efetividade no uso dos três tipos de laboratório (real, remoto e virtual) [2,3]. No entanto é muito escassa a literatura que aborda o uso de sistemas hápticos em laboratórios remotos para fins educacionais. Entretanto, sabe-se que na telemedicina, por exemplo, o uso de sistemas hápticos pode melhorar a segurança e efetividade da prática médica [4].

Este trabalho apresenta o uso de sistemas hápticos em laboratórios remotos como recurso para enriquecer experimentos cuja compreensão é facilitada pela percepção de propriedades táteis.

2. Laboratórios Remotos

Os laboratórios remotos são espaços reais que podem ser utilizados remotamente pelos usuários. Neles os alunos podem ver o que ocorre no ambiente real, através de imagens fornecidas por câmeras, e modificá-lo utilizando dispositivos de interação. Em um laboratório remoto a comunicação com os equipamentos reais é estabelecida por sistemas computacionais e componentes elétricos e eletrônicos [5]. As câmeras capturam o ambiente em tempo-real e permitem observar as modificações realizadas à distância. Neste caso, valores medidos pelos componentes eletrônicos também são transmitidos ao aluno.

A Figura 1 representa a interface de usuário de um experimento remoto em que o dispositivo está visível através das suas imagens real (vídeo) e virtual. No experimento, uma barra encastrada em um dos extremos, com sensores de deformação de resistência elétrica colados em uma zona da sua superfície, é atuada por um motor linear em um ponto bem definido do seu

outro extremo livre. Esta experiência foi desenvolvida para possibilitar a alunos das engenharias, particularmente a mecânica, testar remotamente um sistema que permite verificar a Lei de Hooke, adquirir familiaridade com a metodologia usada em laboratório para medição do Módulo de Young de um material e, finalmente, adquirir algum treino de cálculo a partir dos resultados do experimento.

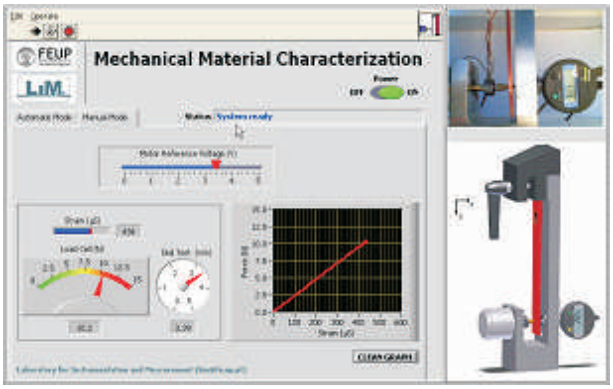


Figura 1 – Interface do usuário para atuação remota em um laboratório real.

3. Laboratórios Remotos com Sistemas Hápticos

A utilização de sistemas hápticos em laboratórios remotos visa melhorar a qualidade da interação, pois permitem ao usuário observar sensações relacionadas à força e às propriedades dos objetos manipulados. O usuário pode manipular remotamente um equipamento através do uso de um dispositivo háptico. Tal qual ocorre em um laboratório presencial, apenas um usuário por vez interage com o experimento. Todo sistema de comunicação ocorre a partir de uma estrutura cliente-servidor na qual o servidor está conectado ao equipamento real e o cliente é executado no computador do usuário.

Com o objetivo de ampliar o realismo em experimentos conduzidos em laboratórios remotos, a aplicação apresentada na Figura 1 foi utilizada para testes iniciais com sistemas hápticos. Nesta aplicação é disponibilizada uma interface bidimensional para atuação de uma barra. A interface apresenta três áreas principais: de dados, de imagem real e de imagem virtual. A imagem virtual representa a imagem de um modelo do equipamento real que está sendo utilizado. A área com a imagem real apresenta a captura em tempo-real do equipamento e permite ao usuário observar os resultados das suas ações. A área de dados, entretanto, é a única região que permitirá a ação do usuário para a realização do experimento. Nesta região o usuário pode aplicar valores variáveis de força sobre a barra (cursor horizontal – entrada de dados) e verificar informações sobre o real valor de carga aplicada, sobre a deformação da barra na zona sensorizada bem como obter o valor da flecha no ponto de aplicação da carga (saídas de dados). Nesta mesma área da interface o usuário pode também observar gráficos com dados da deformação e flecha gerados com a força aplicada (saída gráfica de dados).

Pela natureza deste experimento, apenas um usuário de cada vez pode realizá-lo, tal qual ocorre no experimento real. Entretanto, os resultados podem ser observados e discutidos em grupo.

A nova concepção desta aplicação abordou a integração de modos de interação mais realistas, que permitissem ao usuário explorar mais detalhadamente e utilizar mais intuitivamente o experimento. Nesta nova abordagem, a interface foi concebida para abordar três aspectos:

a) interação intuitiva;

b) exploração detalhada;

c) visualização.

4. Desenvolvimento

Para o desenvolvimento da aplicação utilizando sistemas hápticos, observou-se a necessidade de modificação na aplicação cliente, executada localmente na máquina do usuário. Esta aplicação precisou ser adicionada de rotinas próprias de comunicação com um sistema háptico, definição de pacotes de entrada e saída na comunicação cliente/servidor, modificações da interface e modelagem tridimensional do equipamento real (Figura 2).

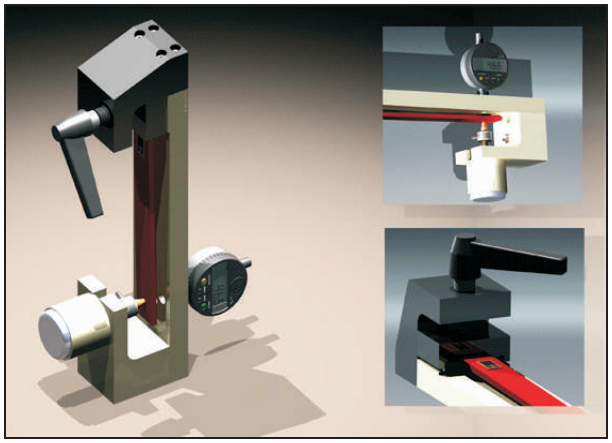


Figura 2 – Modelo virtual do equipamento real.

Dado que o dispositivo háptico utilizado (Figura 3) possui três graus de liberdade em força e o sistema remoto permite apenas atuação segundo uma direção,

foi necessário mapear o espaço de trabalho do dispositivo háptico no espaço de trabalho do sistema remoto (Figura 4). O círculo da direita representa o ponto inicial de atuação da barra. No ambiente do dispositivo háptico, tal corresponde a ter o ponto assinalado à esquerda coincidente com qualquer ponto do plano xy. Se o dispositivo háptico for movimentado para a direita do plano xy, isso corresponderá a uma deflexão da barra.

Figura 3. Dispositivo háptico utilizado pelo usuário.

Para a aplicação servidora ocorreu modificação na forma e no conteúdo do pacote de comunicação. Os pacotes de comunicação foram transmitidos por sockets e precisaram ser adaptados para conter as informações de manipulação ocorridas através do dispositivo háptico. Neste ponto, o volume de informação de manipulação gerado pelo sistema háptico foi de 1000Hz, taxa comum de amostragem para este tipo de dispositivo. Tal fato ocasionou uma sobrecarga na comunicação realizada sob protocolo TCP (transmission control protocol), cujo envio e recebimento de pacotes está associado a uma confirmação de transmissão. Por esta razão, optou-se por utilizar comunicação pelo protocolo UDP (*user datagram protocol*). Apesar deste protocolo não realizar confirmações de envio e recebimento, o volume de pacotes enviados pela rede poderia compensar quaisquer perdas. Entretanto, as solicitações ao servidor chegavam em quantidade maior que a capacidade de resposta deste, pois acompanhavam a taxa de amostragem do dispositivo háptico. Assim, o número de pacotes enviados por segundo passou a ser controlado pela aplicação cliente, eliminando o acúmulo de solicitações no servidor e garantido a continuidade das respostas de força enviadas ao cliente.

As respostas do servidor ocorreram sempre que um pacote do cliente, contendo os deslocamentos do dispositivo, fosse recebido e corresponderam às forças

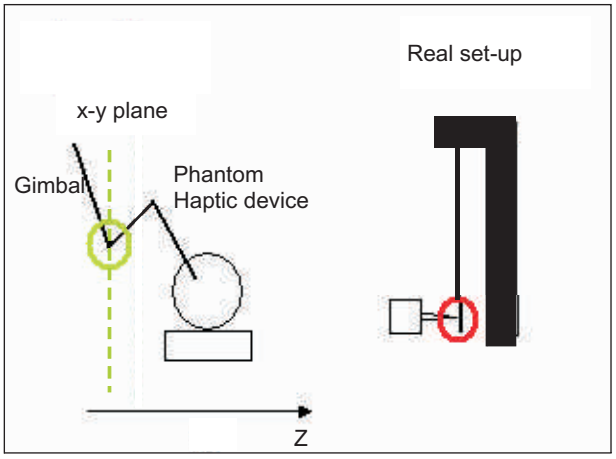


Figura 4 – Relação entre o mundo háptico e o sistema real.

reais medidas no equipamento remoto após a realização do deslocamento recebido. Essa força enviada ao cliente pode também ser visualizada no medidor do equipamento real, cuja imagem é transmitida através da câmera.

A interface da aplicação cliente também precisou ser modificada para conter novas informações e limitar algumas ações locais do usuário. Na nova interface as áreas anteriormente existentes foram mantidas, embora modificadas. A imagem da câmera permaneceu no topo direito da tela, mas a imagem virtual do equipamento é substituída pelo modelo tridimensional e interativo do mesmo (Figura 2). Por sua vez, a área de saída de dados apresenta os valores de flecha, de deformação e de força medidos remotamente no equipamento real.

Os testes da aplicação ocorreram entre uma universidade brasileira (cliente) e uma universidade portuguesa (servidor) utilizando a Internet. Durante os testes, a velocidade média da rede foi de aproximadamente 60Kbps. A taxa média de perda de pacotes foi menor que 1% e o atraso médio entre envio e resposta foi de aproximadamente 0,014 segundos. A resposta visual apresentou atrasos, comuns na aplicação anterior que não utilizava sistemas hápticos. Estes atrasos de vídeo se mantiveram e não dependem do canal de comunicação UDP utilizado para o envio de informações de/para o dispositivo háptico. Apesar disso, o modelo virtual da barra permite que o usuário observe imediatamente a modificação ocorrida, diminuindo o inconveniente de eventuais atrasos na chegada da imagem da barra real.

A Figura 5 apresenta informações da comunicação oferecidas pelo servidor, fornecidas pelo pacote proprietário LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench). Este pacote

permite conectar um sistema computacional a equipamentos reais e está sendo utilizado no experimento apresentado para conectar o servidor ao experimento.

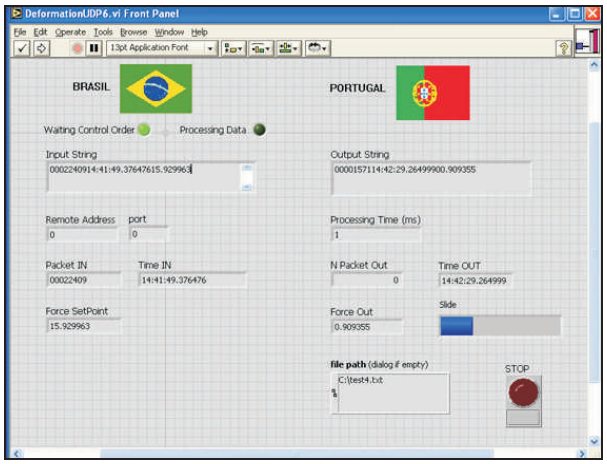


Figura 5 – Janela gráfica com informações sobre a comunicação entre a aplicação cliente (Brasil) e a aplicação servidora (Portugal).

A nova interface do cliente foi construída, seguindo a concepção estabelecida, com o auxílio de três bibliotecas gráficas: a API SDL (*Simple DirectMedia Layer*), responsável pelo gerenciamento do ambiente, eventos de mouse, teclado e o painel principal onde são mostradas as informações, imagens e objetos gráficos; a API OpenGL (*Open Graphics Library*), responsável pelo processamento gráfico da representação virtual do equipamento; e a biblioteca OpenCV (*Open Source Computer Vision Library*), utilizada para a captura das imagens de uma câmera IP (Axis 210) usando o protocolo http, e a exibição destas na interface. A integração desta interface à aplicação de comunicação foi feita em C++ e utilizou a biblioteca *OpenHaptics* (www.sensable.com) para integração do sistema háptico. O dispositivo háptico utilizado pelo cliente foi um *Phantom Omni*.

9. Conclusões

Neste artigo foram apresentados os resultados da concepção e desenvolvimento de uma aplicação de sistemas hápticos para auxiliar o aprendizado através de laboratórios remotos. Os resultados obtidos permitiram verificar a viabilidade de utilização de ferramentas desta natureza no ensino à distância. Atualmente está sendo realizada a validação deste experimento provido de sensação tátil por alunos de engenharia. Tal validação visa também comparar os benefícios advindos da inclusão do retorno de força no experimento anterior.

Para os experimentos foi utilizado um dispositivo háptico 3DOF para retorno de força. Entretanto, para o experimento apresentado, é possível utilizar um dispositivo com apenas 1DOF de retorno de força.

Desenvolvimentos futuros incluem o uso de duas câmeras para captura das imagens para visualização estereoscópica do equipamento real, além da colaboração de vários usuários na realização do experimento.

4. Referências

[1] Netto, A.V.; Machado, L.S.; Oliveira, M.C.F. *Realidade Virtual – Fundamentos e Aplicações*. Visual Books, 2003.

[2] Ma, J. e Nickerson, J. V. "Hands-On, Simulated, and Remote Laboratories: A Comparative Literature Review". *ACM Computing Surveys*, v. 38, n. 3, 2006.

[3] Corter, J. E. et al. "Remote versus hands-on labs: A comparative study". *Proc. 34th ASEE/IEEE Frontiers in Education Conference*. Savannah, GA., 2004.

[4] Dargahi, J. e Najarian, S.; "Advances in tactile sensors design/manufacturing and its impact on robotics applications – a review". *Industrial Robot: An International Journal*, v.32, n.3, pp. 268 – 281, 2005.

[5] Restivo, M. T.; Mendes, J.; Lopes, A.M.; Silva, C.M.; Magalhães, R. and Chouzal, M.F. "E-Teaching Mechanical Material Characteristics". *Proceedings M2D'2006, 5th International Conference on Mechanics and Materials in Design*, INEGI, 2006.

Um Módulo de Rede para a Construção de Aplicações Médicas Colaborativas

Thaís Alves Burity Pereira, Liliane dos Santos Machado
LabTEVE - Universidade Federal da Paraíba,
João Pessoa/PB, Brasil
tata.burity@gmail.com, liliane@di.ufpb.br

Resumo

Sistemas colaborativos representam uma importante ferramenta de ensino, em particular na área médica por viabilizar treinamentos remotos de forma realista, segura e econômica. O presente artigo apresenta um módulo de rede, desenvolvido para ser integrado a um framework livre e aberto para desenvolvimento de simuladores médicos, com o objetivo de possibilitar o desenvolvimento de aplicações colaborativas por meio de interação háptica através da Internet.

1. Introdução

A computação hoje se mostra como uma das mais importantes áreas do conhecimento humano. Este fato decorre, dentre outros fatores, desta ciência possibilitar a multidisciplinaridade, ou seja, a aplicação da tecnologia computacional nas mais diversas áreas, sempre com o propósito de simplificar, automatizar e agilizar tarefas e procedimentos.

A área da educação [1], em particular, principalmente com o advento da Internet e da Realidade Virtual (RV), tem sido bastante favorecida pela computação. A Internet representa hoje o meio de comunicação mais poderoso no mundo, por possibilitar a troca de informações a longas distâncias e com baixo custo, representando um importante meio de disseminação do conhecimento. Já a RV, por meio de dispositivos que atuam sobre os canais sensoriais humanos, possibilita a construção de simuladores capazes de criar modelos dinâmicos do mundo real. Isto permite a realização de experimentos, testes e treinamentos que no mundo real não seriam possíveis ou seriam bastante limitados por representarem situações de risco ou mesmo por exigirem elevados custos ou bastante tempo para serem realizados.

Uma importante área a qual se aplica o uso de simuladores é a medicina. Através de sistemas computacionais de RV é possível, dentre outras coisas, a realização de treinamentos cirúrgicos e outros procedimentos médicos, bem como a realização de práticas de ensino mais interativas, análises e diagnósticos mais precisos.

Nesse contexto, o presente artigo apresenta um módulo de comunicação que faz parte do CyberMed [2] e que se propõe a viabilizar a construção de simuladores de RV voltados para o treinamento médico com a participação de mais de um usuário de forma colaborativa por meio da Internet, através de dispositivos hápticos.

2. Colaboração Háptica

A interação háptica consiste no modo de interação ou participação do usuário em um sistema através de um dispositivo háptico ou de *haptic feedback*, que pode ser de *feedback* tátil ou de *feedback* de força. Os dispositivos hápticos de *feedback* tátil fornecem sensação de toque, provendo informações sobre a geometria da superfície do objeto, sua textura e temperatura. Os dispositivos hápticos de *feedback* de força fornecem sensação de força, provendo informações sobre o peso do objeto e sua maciez ou dureza.

A modelagem física é responsável por definir o comportamento dinâmico do ambiente e dos objetos virtuais e também pelo controle do dispositivo háptico, baseando-se nas leis da física newtoniana [3]. Ela é necessária pois capacitará o computador ao qual o dispositivo háptico está conectado a processar as informações enviadas pelo dispositivo e fazer os cálculos das forças ou variáveis de reação que devem lhe ser enviados.

A interação háptica pode ser incorporada a uma simulação com apenas um usuário ou com múltiplos usuários, o que em geral requer uma rede de comunicação, seja a Internet ou uma rede dedicada, consistindo no que se conhece como *tele-haptics*. No primeiro caso, *haptics* oferecem apenas a sensação de toque, tornando a interação com o ambiente mais realista, uma vez em que o usuário pode perceber a forma geométrica dos objetos que compõem o

ambiente e suas características materiais, como rigidez e textura. No segundo caso, além do toque também são produzidas a sensação de presença [4] e de proximidade [5] entre os usuários que interagem com o mesmo ambiente. A sensação de presença é o que faz o usuário se sentir no ambiente com o qual está interagindo, seja de forma virtual ou remota. A sensação de proximidade provém do fato dos usuários poderem perceber as ações dos demais no ambiente compartilhado, podendo tocar e manipular os mesmos objetos, como numa experiência real.

Contudo, é preciso entender que cada indivíduo tem seu ambiente virtual próprio que consiste numa réplica dos ambientes dos outros indivíduos participantes da simulação – em se tratando de simulações de pequeno porte, que são as que se adéquam ao caso de simuladores médicos. Assim, os efeitos descritos dependem de fatores como alta taxa de atualização gráfica, baixa latência, vasto campo de visão e alto grau de interatividade e comunicação entre os usuários participantes, além da percepção, por parte dos indivíduos, das modificações no ambiente comum devido às ações de um indivíduo em particular [6].

Uma forma simples de compartilhamento de ambientes virtuais por interação háptica pode ser feita localmente pela utilização de mais de um dispositivo háptico em uma mesma máquina, mas esta experiência tem restrições quanto ao número de usuários e com o dispositivo em questão e em geral são pouco documentadas.

A colaboração háptica é uma das três possíveis arquiteturas de ambientes virtuais hápticos compartilhados (*Shared Haptic Virtual Environment* - SHVE) conhecidas atualmente, existindo além dela as simulações estática e cooperativa [7]. Nesse contexto é válido considerar que não existe uma exatidão na literatura sobre as funcionalidades inerentes a cada um dos tipos e assim, em muitas experiências colaboração e cooperação aparecem como sinônimos.

Em uma simulação virtual estática o usuário pode explorar o ambiente virtual através da visão e do toque mas não pode modificá-lo. Em uma simulação colaborativa os usuários interagem com os objetos idênticos ou duplicados de forma alternada, ou seja, um objeto só pode ser manipulado e modificado por um único usuário por vez, o que dá suporte a duas formas de comunicação, denominadas tele-haptics unidirecional ou bidirecional [8]. No modo unidirecional um usuário manipula um objeto e suas ações podem ser sentidas hapticamente pelos demais usuários embora o usuário em ação não receba retorno háptico algum dos usuários "estáticos". No modo bidirecional os usuários sentem as ações dos outros através de modificações no ambiente. Por fim, na simulação cooperativa os usuários interagem com os mesmos objetos simultaneamente e recebem retorno não somente visual, mas também háptico decorrentes das ações dos outros usuários.

Para simulações médicas, a arquitetura que mais se aproxima da forma como os treinamentos e procedimentos acontecem no mundo real é a colaborativa e por isso o trabalho desenvolvido focou-se em seus requisitos.

4. Desenvolvimento

Na simulação colaborativa, devido ao atraso de transmissão em rede, à dificuldade natural de algumas tarefas colaborativas e à falta de conhecimento sobre as habilidades e os comportamentos do usuário, ainda não foi possível desenvolver um modelo universal [9]. Atualmente, para se desenvolver uma aplicação nesse contexto é preciso pré-definir exatamente o que se deseja fazer.

Apesar disso, foi observado que independente da experiência colaborativa em questão, para que esta fosse realizável seriam necessários três elementos (ver Figura 1):

- suporte à utilização de um dispositivo háptico
- suporte à comunicação em rede
- suporte ao compartilhamento de dados por meio de um banco de dados.

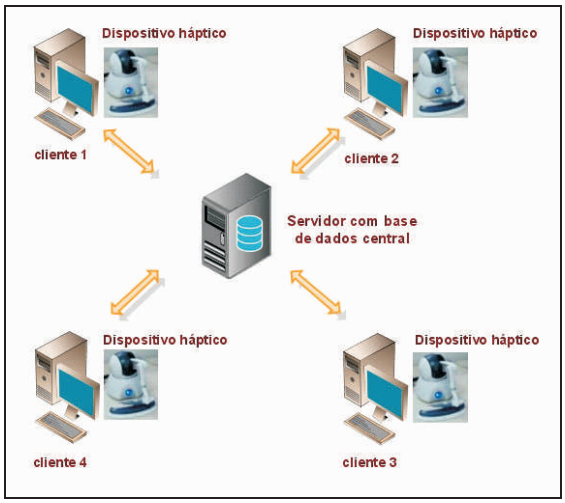


Figura 1 – Arquitetura de um sistema háptico colaborativo.

O trabalho desenvolvido corresponde ao desenvolvimento de um módulo para dar suporte à

comunicação em rede. Ele está inserido no *framework* para a construção de simuladores médicos denominado CyberMed [2], que já oferece suporte a utilização de dispositivos hápticos.

O módulo de comunicação em rede construído consiste em um conjunto de classes em C++ que utilizando *sockets*. Baseando-se na arquitetura cliente/servidor e utilizando os protocolos UDP/IP e TCP/IP, este módulo visa viabilizar a troca de dados de maneira remota para construção de simuladores médicos e outras aplicações que exijam comunicação uni e bidirecional em uma rede. Ele foi construído com base em necessidades verificadas em experimentos realizados em laboratório e em algoritmos conhecidos para implementação de servidores, buscando também oferecer recursos adicionais, para maior mobilidade de utilização. Assim, ele oferece suporte a servidores iterativos e concorrentes *multithread*, com conexão (TCP) e sem conexão (UDP), este último podendo comunicar-se também por *multicast*. Contudo, o módulo foi construído para o desenvolvimento de aplicações de pequeno porte, com até dez usuários.

É importante considerar que para a colaboração háptica o protocolo UDP é mais adequado, por evitar retardos na comunicação, uma vez que a velocidade na transmissão dos dados é essencial para o efeito de realismo na simulação e, além disso, os dispositivos hápticos operam sob alta freqüência. O *Phantom Omni*, por exemplo, o dispositivo utilizado nos experimentos realizados, opera sob a taxa de 1.000 Hz.

As classes de implementação que compõem o módulo são apresentadas nas Figuras 2 e 3, referentes aos protocolos UDP e TCP respectivamente. De acordo com os esquemas, é possível observar que a interface *Socket* define o comportamento de um *socket*, e é implementada pelo *Socket Stream* para comunicação via TCP e pelo *Socket Datagram* para comunicação via UDP. Se for desejável utilizar *sockets* RAW, basta criar uma nova classe que implemente a interface *Socket*. Também é possível analisar que todo cliente ou servidor implementa a interface *Nó*, que define métodos de envio e recepção de dados, já que estes são comuns a todos os nós na rede. Assim, caso seja necessário implementar outro servidor ou cliente para atender necessidades específicas, é possível fazê-lo mantendo a integração entre as partes apenas criando uma nova classe que implemente *Nó*.

Além disso, todo cliente e servidor precisa ter um objeto-socket para se comunicar, bem como um endereço, definido pela classe *Endereço* (que abstrai o uso da estrutura de endereço definida na API de sockets), o que já é feito na implementação dos Servidores UDP e TCP e dos Clientes UDP e TCP. O servidor UDP é iterativo por padrão, portanto, para que ele tenha comportamento concorrente, foi definida a classe *Servidor Concorrente* que contém uma instância de *Servidor UDP*. De maneira similar acontece com o Servidor TCP, embora este tenha duas implementações separadas para servidor iterativo e servidor concorrente. O tratamento de threads é feito pela classe *CybThread*, existente no CyberMed.

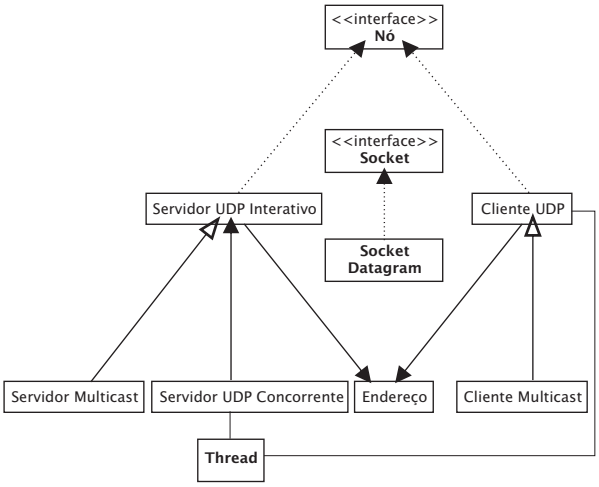


Figura 2 – Esquema de funcionamento da parte do módulo referente ao protocolo UDP.

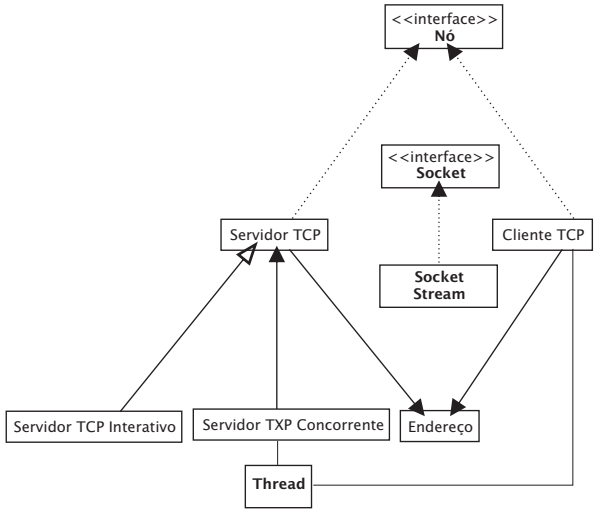


Figura 3 – Esquema de funcionamento da parte do módulo referente ao protocolo TCP.

Utilizando o módulo fica bastante simples construir aplicações colaborativas. Uma aplicação no modelo unidirecional, por exemplo, onde se poderia ter um instrutor orientando vários alunos em um determinado procedimento médico, pode ser rapidamente construída utilizando um servidor

multicast, que envia a posição corrente do dispositivo local, o do instrutor, para os clientes *multicast*, os alunos.

Devido ao fato de que aplicações de RV lidam com objetos e cenas dinâmicos se faz necessário manter um banco de dados no servidor, que deve ser atualizado a cada alteração que os clientes realizam no ambiente virtual compartilhado, a fim de mantê-los sincronizados. Implementações futuras de um módulo para suporte a banco de dados reduziria o esforço envolvido no desenvolvimento de aplicações colaborativas com o CyberMed, visto que a implementação do módulo de rede apresentado não contempla essa funcionalidade.

5. Conclusão

O módulo de rede desenvolvido faz parte do CyberMed, que é um *framework* para a construção de simuladores médicos e que consiste numa ferramenta livre e aberta, ou seja, sem custos para o desenvolvedor. Seguindo as características de modularização do CyberMed [2], também utilizadas para o módulo de rede apresentado, é mantido o baixo acoplamento entre as partes do sistema, gerando ganho em termos de eficiência no desenvolvimento de novas aplicações, bem como um melhor tratamento das dificuldades e erros em cada parte.

O módulo apresentado já foi integrado ao CyberMed e testado com sucesso no desenvolvimento de aplicações com este *framework*. Atualmente estão sendo realizados testes de desempenho deste módulo, bem como seu funcionamento em aplicações que integrem todas as funcionalidades oferecidas pelo CyberMed. Como próxima fase deste trabalho está a elaboração do módulo para uso de banco de dados.

6. Agradecimentos

Agradecemos ao CNPq 506480/2004-6 pelo apoio à realização deste projeto, bem como a concessão de bolsa de iniciação científica à primeira autora.

7. Referências

[1] Valente, J.A. (1993a). Diferentes Usos do Computador na Educação. Em J.A. Valente (Org.), *Computadores e Conhecimento: repensando a educação* (pp.1-23). Campinas, SP: Gráfica da UNICAMP.

[2] Souza, D.F.L.; Cunha, I.L.L.; Souza, L.C.; Moraes, R.M.; Machado, L.S. Development of a VR Simulator for Medical Training Using Free Tools: A Case Study. In: *Proc. of Symposium on Virtual and Augmented Reality (SVR'2007)*, pp. 100-105. SBC. 2007.

[3] Machado, L.S. "A Realidade Virtual no Modelamento e Simulação de Procedimentos Invasivos em Oncologia Pediátrica: Um estudo de Caso de Medula Óssea", *Tese de Doutorado, Escola Politécnica da USP*, São Paulo, 2003.

[4] Sallnäs, E., RASSMUS-GRÖHN, K., SJÖSTRÖM, C. Supporting Presence in Collaborative Environments by Haptic Force Feedback. *ACM Transactions on Computer-Human Interaction, Vol. 7, No. 4*, December 2000.

[5] Basdogan, C., Ho, C., Srinivasan, M. A., Slater, M. An Experimental Study on the Role of Touch in Shared Virtual Environments. *ACM Transactions on Computer-Human Interaction, Vol. 7, No. 4*, December 2000, Pages 443–460.

[6] Durlach, N., Slater, M. "Presence in Shared Virtual Environments and Virtual Togetherness", *Research Laboratory of Electronics Massachusetts Institute of Technology*, Cambridge, MA 02139, 1998.

[7] P.Buttolo, R. Oboe and B. Hannaford, B "Architectures for shared haptic virtual environments". *Computers and Graphics, 21 (4)*, 1997, 421-429.

[8] Shen, X., Zhou, J., Saddik, A., Georganas, N. Architecture and Evaluation of Tele-Haptic Environments. *Proceedings of the Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications*, 2004.

[9] Basdogan, C., Srinivasan, M.A., 2001, "Haptic Rendering In Virtual Environments (PDF)", "Virtual Environments HandBook", pp. 117-134 (online: http://vehand.engr.ucf.edu/).

Session 3: Avatars, Artificial Life, Speech and other Interaction Forms

Improving Collision Detection for Real-Time Video Avatar Interaction

Ricardo Nakamura, Romero Tori
Universidade de São Paulo
Av. Prof. Luciano Gualberto, travessa 3 no. 158 - CEP 05508-970 - São Paulo, SP
55-11-3091-5282
{ricardo.nakamura}, {romero.tori}@poli.usp.br

Figure 1. Views from the test prototype: (a), (b) left and right frames of a stereo pair; (c) depth map (d) color-coded view of the generated octree.

Abstract

In this paper we present a novel technique for the detection of collisions between a video avatar and entities of a virtual environment. This technique is suitable for real-time navigation and interaction of the video avatar within the virtual 3D space, and it is optimized to run on low-cost hardware platforms. It is based on the construction of an octree representation from a dense depth map and thus, takes into account the frame-to-frame posture changes from the video avatar. The results of a multi-threaded implementation are presented and discussed.

1. Introduction

In the field of Augmented Reality (also including mixed reality and augmented virtuality), the use of interfaces that intuitively integrate real and virtual elements is very appealing, and can be considered a good design practice [1]. One way to implement such interfaces is by using video avatars. A video avatar can be defined as a representation of the user, based on his image captured from a video stream and updated at interactive rates. shows a concept illustration of an Augmented Reality application where a video avatar is placed within a virtual scene. Employing the user's image leads to natural communication through gestures and facial expressions in collaborative applications, as noted by Billinghurst and Kato [2] . Interfaces based on video avatars have been explored for entertainment, for instance in the Sony EyeToy games [3]. The self-awareness from observing one's own image in a virtual context can be used for training purposes, as reported by Hämäläinen et al. [4].

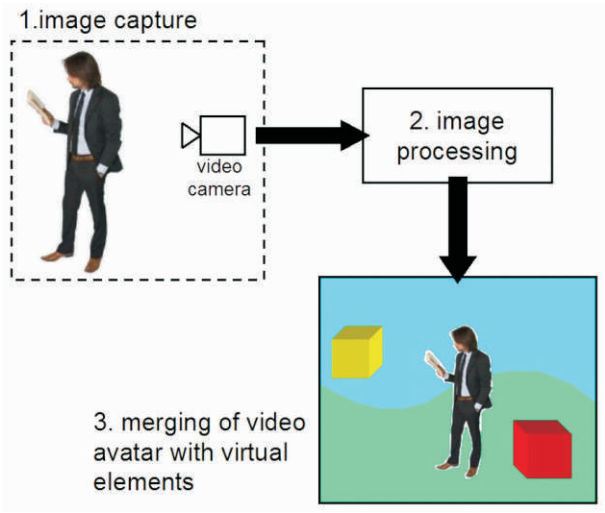


Figure 2. Conceptual augmented reality application with a video avatar.

A video avatar does not have to be capable of interaction to be useful in some contexts. For instance, in a teleconferencing application, a large-scale video avatar may be employed to increase immersion and the sense of "presence" of the remote users [5].

Communication through speech and gestures can be carried out without need for additional interaction capabilities. Likewise, Carranza et al. [6] focus on generating a high-quality video avatar for applications such as free-viewpoint video or 3D television, where dynamic interaction between the video avatar and environment elements is not relevant. However, if the video avatar is meant to be part of the user interface in an Augmented Reality application, that interaction becomes a requirement.

Implementing interaction in a video avatar involves at least detecting when it touches other elements of the augmented reality environment. This, in turn, involves creating a two- or three-dimensional model for the video avatar, which will be used for collision testing. The constraint that the interaction must occur in real time, i.e. at the frame rate of the video stream limits the possible approaches to the problem.

In a previous work [7] we have proposed an architecture for interactive video avatars suitable for home applications. As such, it was designed to run on a personal computer, using a pair of commodity video cameras ("webcams"). For interaction purposes, the video avatar was represented by a set of points in 3D space and an axis-aligned bounding box. Collision detection would be performed first by testing the bounding volumes of interactive objects against the bounding box of the video avatar. In case of a collision at this level, point-in-volume tests with the 3D point set would be performed. This technique showed limitations in terms of performance in complex environments. This was the motivation for finding an alternative, which is presented in this paper. Figure 1 shows the output from one of the test prototypes for our proposed collision detection technique.

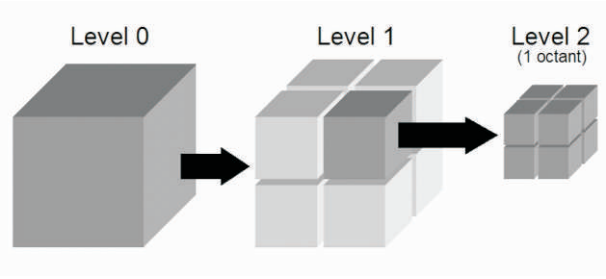


Figure 3. Space partitioning with an octree.

The remainder of this paper is organized as follows: Section 2 presents works in the fields of video avatars and real-time collision detection that are relevant to the discussion of our proposal. Section 3 contains an overview of the video avatar system where the collision detection technique will be employed. Section 4 describes our approach to video avatar collision detection, while Section 5 presents the implementation details. Section 6 contains results obtained from that implementation and the paper closes with a discussion of those results and possibilities of future work in section 7.

2. Related Work

In order to better organize a survey of related works, this section has been divided into two distinct topics. First we present some works that are related to real-time collision detection. Then we present an overview of interactive video avatars and their characteristics, compared to our proposal.

2.1. Real-time Collision Detection

There are several works on collision detection techniques, related to fields such as simulation, virtual reality and computer games. Many of them, like the systems proposed by Cohen et al. and by Chung and Wang , are devoted to high-precision methods, suitable for simulations. However, our interest for this project is in techniques that may return approximate results, but can be executed, on a personal computer, at least as fast as the video frames are captured. These techniques are usually referred to as real-time collision detection, and we adopt this term in the text, although it does not imply the strict restrictions of real-time systems. Ericson presents a survey on collision detection techniques suited for interactive applications, discussing design issues of such systems. Eberly also extensively discusses the efficient implementation of intersection tests between bounding volumes.

Many collision detection techniques are limited to rigid bodies, or articulated assemblies of rigid bodies, whereas the geometry of the video avatar is not know a priori. One possible solution is to match the image captured from the video stream to a humanoid geometric model ,. This approach has two main advantages: it results in a complete geometric model that can be used for very precise collision tests if necessary; furthermore, it is possible to determine what part of the video avatar is colliding, leading to more sophisticated interaction possibilities. However, these techniques are computationally expensive, requiring many computers to run in real time. They also need the input from several cameras placed around the user to correctly match their position to the geometric model. Therefore, they are not suitable for our case.

Different authors have proposed the use of octrees for fast, approximate collision detection [13],[14],[15], even of objects with mutating geometric features. An octree is a hierarchical structure that can be used to create an

approximate volume representation for convex objects. At the first level, the octree represents a finite volume in space. For each subsequent level, each existing node is partitioned into eight octants or cells. Figure 3 shows an example of an octree with 3 levels, or a depth of 3. Although the picture uses cubes for the octants, there is no requirement that the space represented by the octree must be symmetrical. Leaf nodes of the octree correspond to regions of space that contain the represented object. Internal nodes group these leaves into larger bounding volumes, up to the tree's root. Therefore, an internal node indicates a region of space at least partially filled by the represented object.

Collision testing between an octree O and a bounding volume V consists in recursively testing the bounding volumes ON of the octree's nodes for intersection with V, until reaching a leaf node. If an intersection is detected at this level, then a collision has occurred (within the precision of the space partitioning of the octree). In some cases, spheres are used as bounding volumes for the nodes [15]. This can speed up collision testing especially if the octree is not aligned to the coordinate axes. Our technique uses an axis-aligned octree to perform collision detection of the video avatar. This tree is updated on every frame from the video stream, to represent the varying poses of the user as if it was a geometrically mutating object.

2.2. Interactive Video Avatars

There are various approaches to interactive video avatars, depending on the intended application. In general, there is a direct relationship between hardware requirements for the techniques and quality and precision of the results.

Some interactive video avatar applications, particularly in the field of entertainment, employ two-dimensional collision detection. In this case, the video image is acquired from a single video camera. Then, the user's silhouette is detected, using background subtraction or a technique such as optical flow [16]. Then, the silhouette is tested for intersection against the silhouette of other interactive objects.

This approach has two advantages: it involves a single video camera, which eliminates problems of synchronization and inter-camera calibration. Its implementation is also less processor-intensive. Both factors result in lower hardware requirements, which are important for home applications such as games. On the other hand, this technique limits the possibilities of interaction by the user. Since only the silhouette is considered for collision, there is no information about depth – and thus no movement outside of the camera's retinal plane can be used. Examples of systems that employ this type of technique are Sony EyeToy [3] and the "Kick Ass Kung-Fu" system [4]. Our system uses depth information derived from the video streams, so it does not have the above limitation. On the other hand, it requires two video cameras and is relatively more processor-intensive.

Another approach can be taken for video avatar systems based on visual hulls [17], such as the one described by Nguyen et al. [18]. In this case, the estimated visual hull itself can be used as a convex object for collision detection. For this purpose, it can be converted into either a polygonal or volumetric representation. Magnor [19] presents an extensive survey on techniques for both. These representations are usually detailed, convex approximations of the video avatar. However, they require a large number of cameras. They are also computationally expensive to run on a single personal computer interactively, so they are not feasible for our system.

Lastly, there are approaches where video avatar interaction is not based directly on information from the video stream. For instance, Siscoutto and Tori [20] describe an interactive video avatar for teleconference that is capable of navigation within a virtual environment. A simple bounding volume of fixed geometry is used for collision detection against elements of the environment.

Our approach, as discussed in section 2.1, involves the creation of a volumetric representation based on the dense depth map of the video avatar. While there are many works related to video avatars with depth-from-stereo information, the usual approach in these cases is to generate a polygon mesh from the depth map [21],[22], whereas our solution consists in building an octree from the depth map.

3. Overview of the Video Avatar System

This section presents an overview of AVMIX, the video avatar system for which the collision detection technique is proposed. A more complete description of the system can be found in [7]. A very important constraint of our system is that it is aimed at home and educational applications, such as games and low-cost teleconference systems. Thus, it must run on a consumer-level hardware platform consisting of a personal computer and commodity video cameras, or "web cams".

Figure 4 shows a diagram of the AVMIX architecture and the services that are made available to client applications. Subsystems are shown as packages, with

the dependencies between them. The Image Acquisition subsystem is responsible for retrieving frames from the video cameras and performing distortion correction and rectification of the images. The Segmentation subsystem removes the background from the images, so that the user's image stands on an homogeneous background. The Depth Mapping subsystem produces a dense depth map from the segmented video frames, and the Rendering subsystem generates a visual representation of the video avatar.

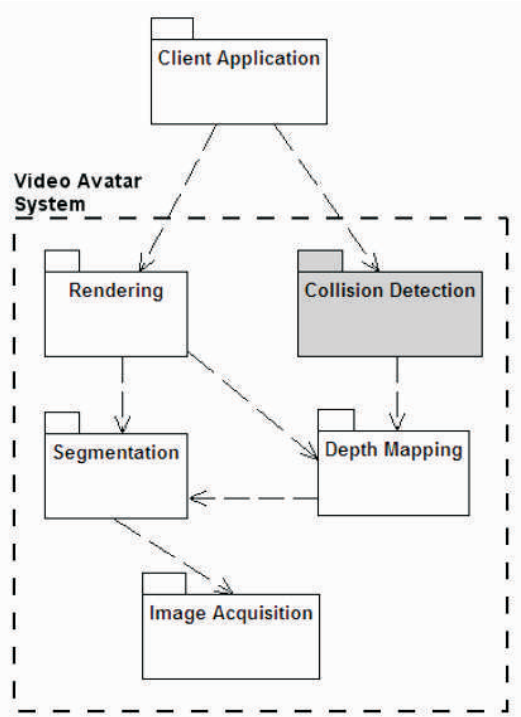


Figure 4. Subsystems that are part of the AVMIX video avatar system.

The collision detection subsystem, which is the subject of this paper, depends on the output of the depth mapping subsystem. This output is a grayscale image corresponding to the dense depth map of the scene containing the video avatar. Each pixel of the image contains the disparity between images. The actual depth, or distance between object and camera, of each pixel in the depth map, can be calculated according to Equation 1, where f is the focal length of the camera, B is the baseline, or distance between the pair of cameras, and w is the disparity.

$$D = \frac{f \quad B}{w} \tag{1}$$

The collision detection subsystem is independent of the implementation of the depth mapping subsystem. Currently, we employ a depth-from-stereo algorithm that uses the sum of squared differences over a fixed size window as a measure of similarity between pixels. More details about the algorithm are presented in [7]. For an extended discussion about depth-from-stereo algorithms, see the survey from Scharstein and Szelinski [23].

4. Collision Detection Technique

As mentioned in section 1, our initial approach to collision detection involved the use of an axis-aligned bounding box for rough testing, and point-in-volume tests for accurate detection [7]. While this solution is relatively simple, the computational cost of the point-in-volume tests is high. A typical point cloud generated from the dense depth map of the video avatar may contain around 104 points. Performing these tests on every frame may be feasible if there are only a few entities of the virtual environment to test for collision. It is possible to spatially filter this set of points, but the risk of missing collisions increases when reducing the number of points.

Our new solution consists in using an octree to represent the video avatar for collision detection. The octree is built for every new frame captured from the cameras, thus it accounts for motion of the user without explicitly matching that motion to a geometric model. Therefore, the new collision detection module has two main components: the octree-building algorithm and the collision-testing algorithm.

4.1. Octree Building

Our algorithm to build the octree consists in calculating the 3D points that make up the video avatar and classifying them into the octree, so that all regions in space containing at least one point are marked as "filled" by the video avatar. Equations (2a), (2b), and (2c), which are based on Chen et al. [22], are used to calculate the 3D points corresponding to the values from the depth map. The values of i and j are the image coordinates of the pixel on the depth map, while D is obtained from Equation 1. The values of α and β are half the horizontal and vertical viewing angles from the cameras, and W and H are the dimensions (width and height) of the video frame.

$$x = D \quad \tan\alpha \quad \frac{i \quad W/2}{W/2} \tag{2a}$$

$$y = D \quad \tan\beta \quad \frac{H/2 \quad j}{H/2} \tag{2b}$$

$$z = D \tag{2c}$$

After the 3D coordinates are calculated, the point is classified into one of the octree's nodes – that is, the corresponding octree leaf node is marked as "full". Figure 5 shows a 2D view of this process. This can be accomplished by recursively testing the point against the bounding volumes of the octree nodes. Our implementation takes a different approach, explained in section 5.

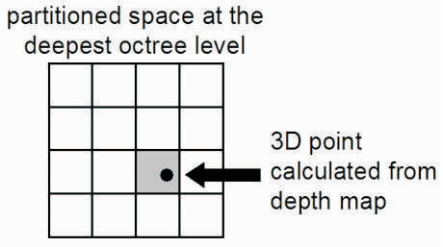


Figure 5. Example of point classification.

4.2. Octree Collision Testing

In AVMIX, objects that may interact with the video avatar are represented, for collision testing purposes, as one of two types of bounding volumes: axis-aligned bounding boxes or spheres.

As discussed in section 2.1, collision testing with the octree can be performed recursively. Starting with a bounding box with dimensions and coordinates matching the entire volume represented by the octree, we perform an intersection test with the other object's bounding volume. If an intersection is detected, tests are repeated for the volumes of each full or partially filled octant. The process continues until an intersection is detected with a full node, or no intersection is found on the children of a partially filled node. The first case indicates that a collision actually happened otherwise no collision is reported. shows a 2D view of the process for a 3-level tree. In case (A), a collision is refuted when the last level of the tree is reached. In case (B), a collision actually happens.

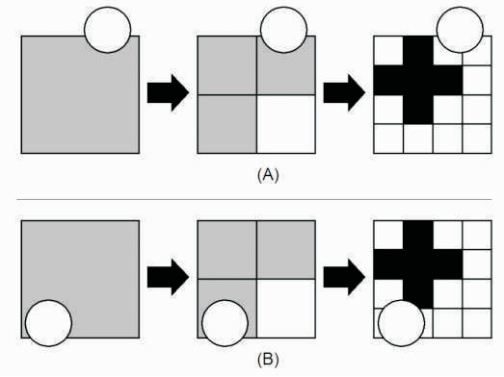


Figure 6. Example of collision testing

5. Implementation

In this section we discuss the implementation details of the collision detection subsystem. We start with the octree representation and construction.

Octrees are usually stored in memory as dynamic data structures, which can be very efficient in terms of space. However, our main interest in using octrees for collision detection is to increase performance. The overhead of memory allocation as the octree is built on every video frame can become prohibitive. Allocating a large number of nodes and managing them would avoid this problem, but it would also remove the benefit of using less memory.

Our approach to the problem is to use a static, array-based representation of the octree. The number of nodes needed to represent a full octree of depth D is given by Equation 2.

$$N_{nodes} = \sum_{i=0}^{D} 8^i \qquad (2)$$

The disadvantage of using an array for the octree is that the total number of nodes – and thus the required storage memory – grows exponentially with tree depth as shown in Table 1. If using 32-bit integers to represent tree nodes, to optimize memory access in a 32-bit processor, the actual memory usage would be at least four times the values listed.

Tabel 1. Number of nodes x octree depth

Octree Depth	Number of Nodes
1	1
2	9
3	73
4	585
5	4681
6	37449
7	299593
8	2396745

From the numbers in Table 1, it can be seen that octrees of depth greater than 6 become prohibitively expensive to implement as arrays, so that becomes a practical limitation for our system. A 6-level octree will partition space into 25 or 32 cells along each axis. If we use such an octree to cover the volume of a cube with 2m sides in space, then each cell will be a cube of 62.5mm sides. While this resolution is not enough to represent individual fingers or other small details, it is probably enough for interaction of a full-body video avatar. Resolution can also be increased if the space

enclosed by the octree is reduced, which may be feasible for specific applications.

The nodes of the deepest level of an octree represented as an array can be very efficiently accessed using a technique known as Morton Location Codes. Ericson [10] describes this in detail, and provides an efficient manner to calculate the location code given a triplet of (x, y, z) coordinates in the octree space.

One important limitation of using Morton Location Codes is that the range of the codes depends on the number of bits used for it. In our implementation, we use 32-bit unsigned integers, which give us a maximum of 10 bits per coordinate. That, in turn, limits the maximum depth of the octree to 10 levels. As discussed before, the practical limit of our implementation is of six levels, so that does not pose a problem for us.

The use of Morton Location Codes on an array-based octree allows us to adopt a different tree-building strategy. The usual approach would be to take each 3D point that is part of the video avatar and recursively test it against the bounding boxes of the octree nodes. For an octree with N levels, that means N operations per point, involving three floating-point comparisons.

Alternatively, we can calculate the Morton Location Code for each point and mark the corresponding octree leaf node as "full". Morton Location Code calculation can be implemented as a small set of integer logical operations to be performed per point. After all points have been classified, we must propagate the "fullness" information upward through the octree, which involves a recursive iteration of all nodes. Table 2 shows the number of macro operations (point classification and node traversal) required by each approach, considering a set of 10000 points, which is the average size of the point set for video avatars in our tests. It can be seen that our approach should be more efficient than simple recursive classification of the points, for octrees of depth between 2 and 6. For greater depths, the exponential increase in the number of nodes to be traversed makes this approach less efficient.

It is very important to note that the operation of marking nodes using the location code involves a calculation that depends on a single value and a blind write to the array. This means that the first phase of the algorithm – classification of the video avatar point set – can be implemented to run in multiple parallel threads with distinct subsets of the point cloud.

The second phase of the algorithm is currently

Tabel 2. Number of operations to build an octree of given depth

	Number of Operations	
Octree Depth	Recursive classification	Morton-propagation
1	10000	10001
2	20000	10009
3	30000	10073
4	40000	10585
5	50000	14681
6	60000	47449
7	70000	309593

implemented in a recursive form, starting from the root of the octree. Each call to the recursive method gathers the values (empty, full or partially filled) of a node's eight children and, based on those, sets the node's own value.

The collision-testing algorithm is implemented in a recursive way, according to the description given in section 4.2. If P is the index of a node in the array-based octree, then its children nodes will be in positions 8P+1 to 8P+8 of the array. Using this equation, it is possible to navigate through the octree. The intersection tests, for bounding boxes and bounding spheres, are based on the implementations given by Ericson [10].

6. Results

To evaluate the performance of our implementation, we ran a series of tests with two prototypes. In the first prototype, the dense depth map is calculated and sent to the collision detection module. The execution time for the construction of the octree is measured, using OpenCV [24] timing functions. Figure 1 shows the prototype running with a synthetic stereo pair. Views (a) and (b) are the segmented input frames, view (c) shows the dense depth map and view (d) shows a color-coded, perspective rendering of the octree.

The average execution times for a set of 80 stereo pairs captured from a video stream were calculated using 1 to 4 threads and a maximum octree depth between 4 and 7 levels. The test platform was a personal computer with an Intel Core 2 Quad Q6600 2.4GHz processor with 4GB of RAM. This way, it is possible to observe the behavior of the system with up to four parallel threads. Figure 7 presents the average execution times in milliseconds for different combinations of number of threads and octree depth. It is possible to observe the exponential asymptotic behavior of the algorithm, due to the traversal of all octree nodes in the second phase. Figure 8 presents the relative performance gains from using more threads in the test platform. We

hypothesize that the non-linear gain from adding the fourth thread is due to the fact that one of the processor cores is already in use by the operating system, thus only three other threads may actually benefit from the additional processor cores. This chart also makes it easy to notice the diminishing benefits of optimizing only the first phase of the octree-building algorithm as the number of nodes of the tree increases and tree traversal becomes the most expensive part of the algorithm.

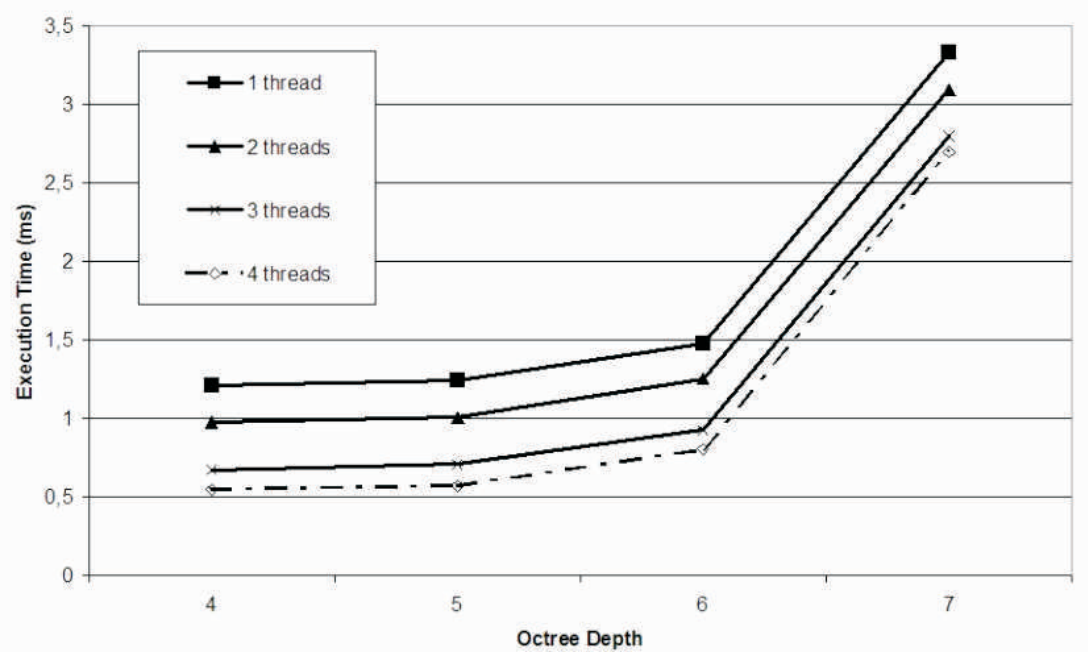


Figure 7. Average execution times for octree construction.

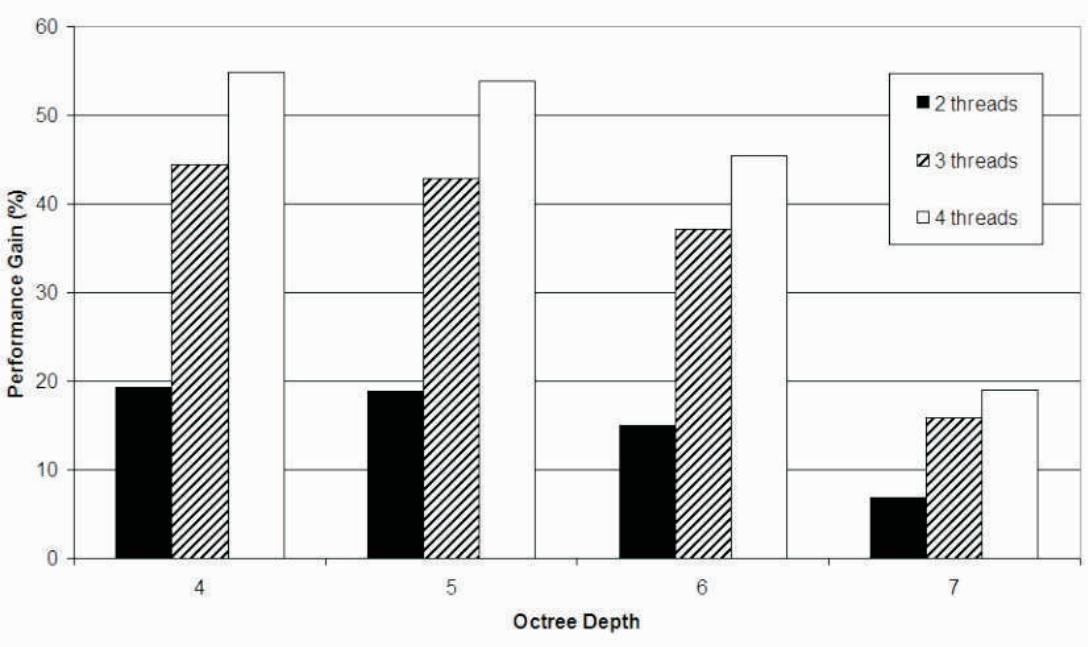


Figure 8. Relative performance gain with multiple threads

The second prototype was developed to gather performance data on the implementation of the collision test algorithms. In this case, the octree is first initialized with a set of points that results in a three-dimensional grid configuration of the leaf nodes, as shown in Figure 9. The octree is set to cover a cubic volume with a side of 2 meters around the origin. Then, a set of one thousand bounding volumes is generated at random positions in the volume of a cube with a side of 2.4 meters around the origin. This allows some of the bounding volumes to be placed outside of the octree volume. These volumes are either axis-aligned cubes with 0.2 meters sides, or spheres with 0.1 meters radius. It is expected that this configuration can be used to measure the average behavior of the algorithm, not being either a best-case or worst-case scenario.

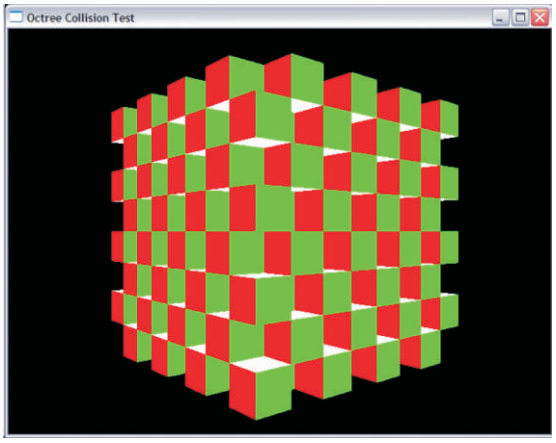


Figure 9. Octree configuration for collision detection performance testing.

The collision tests between the octree and the one thousand bounding volumes were repeated one hundred times for boxes and spheres separately, with execution times measured using OpenCV timing functions. The average execution times were computed, for various octree depths. Figure 10 presents the performance results for collision tests against axis-aligned bounding boxes, while Figure 11 shows the results for tests with bounding spheres. The times shown correspond to the execution of one thousand tests. The fluctuations in the chart of are attributed to precision limitations in the timing function used. The hardware platform for these tests was a computer with a single-core, AMD 64 3200+ processor and 1GB of RAM.

It is interesting to notice that the results for bounding spheres are approximately three orders of magnitude faster than the results for bounding boxes. We believe that this is mostly because of optimizations performed by the compiler, as the complexity of both test algorithms is not significantly different. To test our hypothesis, we ran the same tests with non-optimized builds of the prototype. The results, shown in Figure 12, demonstrate that in this case, performance results are very similar, as expected.

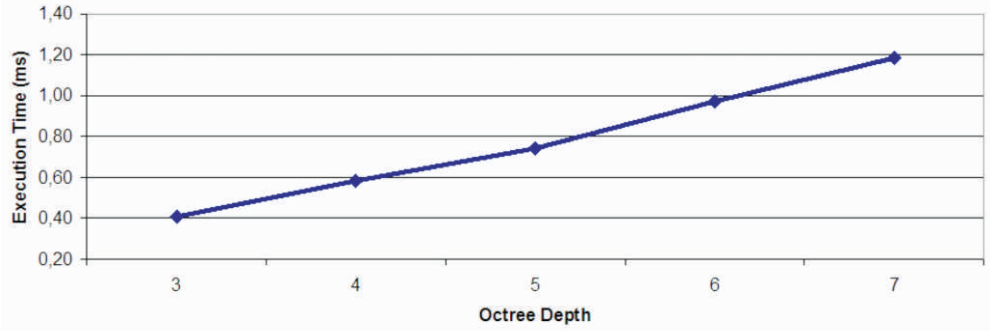


Figure 10. Performance results for tests with axis-aligned bounding boxes.

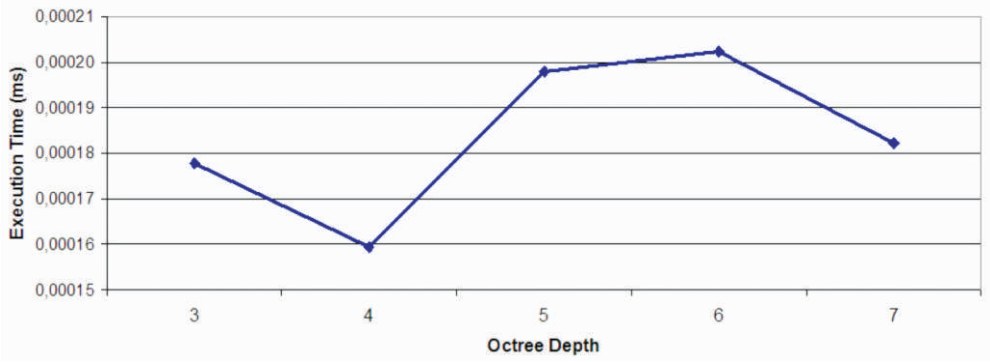


Figure 11. Performance results for tests with spheres.

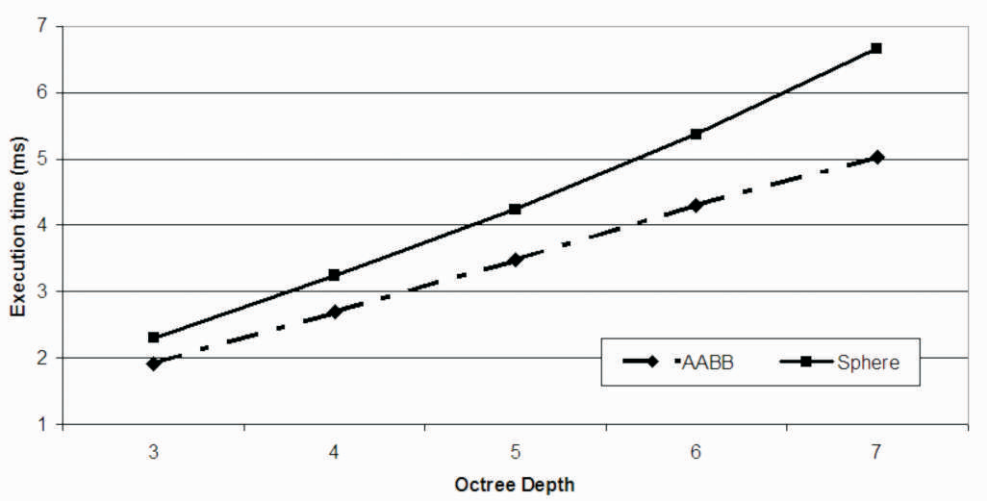


Figure 12. Performance results on non-optimized build.

7. Conclusion

We have presented a technique to perform collision detection between a video avatar and virtual objects in real time. The configurable multi-threaded implementation can take advantage of multi-core processors that are becoming increasingly common in the consumer market. However, tests with the current implementation show that making only the first phase of the octree-building algorithm multi-threaded results in diminishing performance gains for more precise octrees. We also performed tests with the collision detection algorithm, obtaining results that show that it is suitable for real-time use. The results for collision tests of the octree against bounding spheres were particularly fast.

Future work for this project includes implementing a multi-threaded version of the second phase of the octree-building algorithm, as it has become the bottleneck for the current implementation. Another line of research involves evaluating the feasibility of a GPU-based implementation, so that its performance can be compared to the multi-threaded implementation. The results from the bounding sphere collision tests also motivate us to study the performance gains of implementing the video avatar representation as a spherical octree.

Lastly, we also intend to develop complete applications with the AVMIX video avatar system, including augmented reality games and teleconference systems.

8. Acknowledgements

This work has been performed with support from CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil.

9. References

[1] M. Billinghurst, R. Grasset, and J. Looser, "Designing Augmented Reality Interfaces", *Computer Graphics, 39, 1*, ACM Press, New York, USA, 2005, pp. 17-22.

[2] M. Billinghurst and H. Kato, "Collaborative Mixed Reality", *Proceedings of the International Symposium On Mixed Reality*, Springer-Verlag, Berlin, Germany, 1999, pp. 261-284.

[3] Sony EyeToy. http://www.eyetoy.com.

[4] P. Hämäläinen et al., "Martial Arts in Artificial Reality". *Proceedings of SIGCHI Conference On Human Factors In Computing Systems*. ACM Press, New York, USA, 2005, pp. 781-790.

[5] S. Prince et al., "3D-Live: Real Time Interaction for Mixed Reality", *Proceedings of ACM Conference On Computer Supported Cooperative Work*, ACM Press, New York, USA, 2002, pp. 364-371.

[6] J. Carranza et al., "Free-Viewpoint Video of Human Actors", *ACM Transactions on Graphics, 22, 3,* ACM Press, New York, USA, 2003, pp. 569-577.

[7] R. Nakamura and R. Tori, "A Technique for Collision Detection and Real-Time Video Avatar Interaction in Mixed Reality Environments", *Electronic Proceedings of the XI Symposium on Virtual and Augmented Reality*, SBC, Brazil, 2007.

[8] J.D. Cohen et al., "I-COLLIDE: an interactive and exact collision detection system for large-scale environments", *Proceedings of the 1995 symposium on Interactive 3D graphics*, ACM Press, New York, USA, 1995, pp. 189-ff.

[9] K. Chung and W. Wang, "Quick Collision Detection of Polytopes in Virtual Environments", *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, ACM Press, New York, USA, 1996, pp. 125-131.

[10] Ericson, C., *Real-Time Collision Detection*, Morgan Kaufmann, San Francisco, USA, 2005.

[11] Eberly, D. H., *3D Game Engine Design*, Morgan Kaufmann, San Francisco, USA, 2001.

[12] G.K.M. Cheung et al., "Markerless Human Motion Transfer", *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission*, IEEE Computer Society, Washington, DC, USA, 2004, pp. 373-378.

[13] G. Garcia and J.F. Corre, "A New Collision Detection Algorithm Using Octree Models", *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*, IEEE Computer Society, USA, 1989, pp. 93-98.

[14] P.M. Hubbard, "Collision Detection for Interactive Graphics Applications", *IEEE Transactions on Visualization and Graphics,* 1, 3, IEEE Computer Society, Washington, DC, USA, 1995, pp. 218-230.

[15] H.P. Nascimento et al., "Sistema de Detecção de Colisão usando Octrees Esféricas", *Proceedings of the III Workshop Brasileiro de Realidade Virtual*, SBC, Brazil, 2000, pp. 271-272.

[16] B. Neumann, "Optical Flow", ACM SIGGRAPH Computer Graphics, 18, 1, ACM Press, New York, USA, 1984, pp. 17-19.

[17] A. Laurentini, "The Visual Hull Concept for Silhouette-Based Image Understanding", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16, 2, IEEE Computer Society, USA, 1994, pp. 150-162.

[18] T.H.D. Nguyen et al., "Real-Time 3D Human Capture System for Mixed-Reality Art and Entertainment", *IEEE Transactions on Visualization and Computer Graphics*, 11, 6, IEEE Computer Society, USA, 2005, pp. 706-721.

[19] Magnor, M. A., Video-Based Rendering, A K Peters, Wellesley, USA, 2005.

[20] R.A. Siscoutto and R. Tori, "AVTC - Augmented Virtuality Tele-Conferencing". *Proceedings of VII Symposium on Virtual Reality*, SBC, Brazil, 2004, pp. 124-136.

[21] K. Tamagawa et al., "Developing a 2.5-D Video Avatar", *IEEE Signal Processing Magazine*, IEEE, USA, 2001, pp. 35-42.

[22] L. Chen et al., "Using Stereo Camera to Realize Realistic Video Avatar in Virtual Environment", *Proceedings of International Conference On Signal Processing*, IEEE, USA, 2002, pp. 707-710.

[23] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms", *International Journal of Computer Vision*, 47, 1, Springer US, USA, 2002, pp. 7-42.

[24] OpenCV: Open Computer Vision Library. http://sourceforge.net/projects/opencvlibrary/.

Speech Enabled 3D-Browsers: Development Issues and a Software Framework Alternative

Ednaldo Brigante Pizzolato, Diego Daniel Duarte
Departament of Computing
Federal University of São Carlos
São Carlos - SP - Brazil
ednaldo@dc.ufscar.br; duarte.diego@gmail.com

Marcio Merino Fernandes
FACEN-Computer Science
Methodist University of Piracicaba
Piracicaba - SP - Brazil
mmfernan@unimep.br

Abstract

There is a growing trend for advanced 3D visualization systems to employ multimodal input interfaces. One of the most desirable of such modalities is speech recognition, as it is natural to users, and complements very well other input interfaces. However, although well understood, integrating a speech interface into a complex system can be a difficult and time consuming task. In this paper we present our approach to the problem, a software framework requiring minimum additional coding from the application developer. The framework couples voice command definitions with existing interaction code, automating the task of creating a new grammar to be used by the recognizer engine. A new listener component for the Xj3D was created, which makes transparent to the user the integration between the 3D browser and the recognizer. We believe this is a desirable feature for virtual reality system developers, and also to be used as a rapid prototyping tool when experimenting with speech technology.

1. Introduction

In general terms, virtual reality (VR) can be defined as an advanced user interface designed to visualize, moving around, and interact within computer generated 3D spaces [10]. Sophisticated VR applications are now common place in some sectors, such as medicine, advanced training, simulation, and industrial design. Those systems often employ expensive special purpose devices for interaction between users and the virtual world. On the other hand, a basic VR system can be designed with standard components, standing at the core of those a 3D virtual scene, and a browser with visualization and interaction capabilities. Virtual scenes are often designed using VRML (Virtual Reality Modeling Language), or its successor, X3D [6]. Interaction through browsers are typically done by direct manipulation, using a mouse or some other device. Although relatively simple, this setting can be useful for many application areas, and also as a prototyping tool for more advanced systems.

One limitation of the basic system just described is its intrinsic mono modal interaction nature. In other words, the user is limited to a single input modality, which in many cases diverts from the original aim of having interaction with the virtual world in a way as natural as possible. The alternative to that is to provide multimodal interaction capabilities, i.e., multiple input modalities. Probably the first working implementation of this approach was the so-called "Put That There" system, which processed speech in parallel with touch-pad pointing [3]. Since then, many multimodal systems hava been developed [21], and is still a very active research area.

Among the input modalities employed to build multimodal interfaces are speech, natural language, hand gestures, hand postures, and body gestures. Speech is the most natural way to interact among humans, and for this reason it is a very desirable way to interact with a computer system. Some of the advantages of this approach are:

- It provides the user a more natural feeling during the interaction experience;
- It may allow alternative choices on how to interact with the virtual world;
- It enables "hands-free", "eyes-free", or "hands-busy" interaction;
- It enables users to refer to objects not present in the current view of the virtual world.

So, it is no surprise that many virtual reality systems provide speech recognition capabilities as an interaction modality [19]. However, speech recognition is a rather complicated task by itself. Integrating it into another complex system such as a VR can be a burden to the entire project, which may prevent it in many cases. Thus, in this paper we propose a software framework aiming to alleviate this problem. The approach assumes

the adoption of an independent speech recognizer (a common decision in this area), which then requires the VRML/X3D browser to be able to communicate with it. Therefore, our first step was to implement a component inside an Xj3D browser [7] able to listen to the recognizer and understand its codes. However, this solution implies a tight interdependence between the recognizer, the listener component, and the 3D application. Our proposal is to hide, as much as possible, the speech implementation from the VRML/X3D programmer, keeping the additional programming to a minimum. The key for the simplicity is to assign voice commands to mouse events making the virtual world to behave as if the mouse was clicked. The scripts created to function when the mouse is clicked will also function when a specific voice command is given. The remainder of this paper is organized as follows: Section 2 presents some related works that are representative of the main issues involved, followed by a brief presentation about speech recognition in Section 3. Then Section 4 describes the core of the work developed, followed by the onclusions in Section 5.

2. RelatedWork

Researchers have been comparing the use of voiceactivated commands against conventional interface devices (i.e. keyboard and mouse) since early 1980's. The work described in [23] compares the speed and accuracy of speech input versus typed input of commands in a simulated military application over a distributed network. In that work a recognition accuracy of 96.8% was achieved when using voice input, and on average that was 17.5% faster than typed input,with less errors. However, other studies [17, 11] showed that keyboard inputs could be faster than voice inputs (although more error prone). Later on, Karl compared mouse and voice inputs as an interface for a word processor application [14] . He showed that, on average, the task time was 18.67% quicker when speech interface was applied. In another direction, Pausch and Leatherby [22] investigated the improvement voice could bring to systems where mouse and keyboard could also be employed. In general, they concluded that the task completion time could be reduced over 10% when speech and mouse were employed together. On the other hand, Hauptmann pointed out that users may had lost their attention while waiting several seconds for the system's response and assumed that in the future speech could perform much better [12]. More recently, Christian [5] showed that voice control adds approximately 50% to the performance time for certain types of tasks.

The use of speech interfaces with 3D visualization systems is the topic of many research works, either as a conceptual approach, or concerning practical implementation issues, such as in [18]. Some experiences to apply multimodal and spoken language interfaces to a number of electronic reality applications are described in [4]. A bimodal, speech/gesture interface is used to control a 3D display in a virtual environment for structural biology analysis [24]. The integration of speech and other forms of multimodal interaction to the VTK visualization system is described in [15].

The complexity of the speech and natural language input modalities makes it very difficult to build them as an integrated part of a visualization system. Instead, they are typically independent components responsible for acquiring voice commands, interpreting them, and emitting a message to interact with the system. A general framework for building integrated natural-language and multimodal dialog systems is presented in [13]. The framework is based on a distributed component model that allows reuse and extension of existing software modules, as it was demonstrated in the development of the so-called Multiplatform framework. A multimodal interface for navigating in arbitrary virtual VRML worlds is described in [1], an approach employing a context free grammar to control the communication between individual components of the system. In another work a modular approach for integrating multiple input/output modalities in virtual environment is described [2]. The proposed system architecture was applied to case study dealing with molecular visualization, modeling and fitting, showing good results. All those three works are well developed frameworks aimming to facilitate the integration of multiple input modalities into general purpose or virtual reality systems. However, none of those is particularly concerned with the issue aimed by our work, which is to provide developers of 3D interactive applications with a way to integrate a speech interface into their systems with minimum specific knowledge and effort.

In a more general context of using voice input with browser applications, there are some standardization efforts aiming to improve portability and development time. Possibly the most accepted one is VoiceXML, an XML format for specifying interactive voice dialogues [9]. A related standard, called Speech Recognition Grammar Specification defines a syntax for representing dialogue grammars so that developers of systems using speech recognition can specify the words and patterns of words to be listened by the recognizer

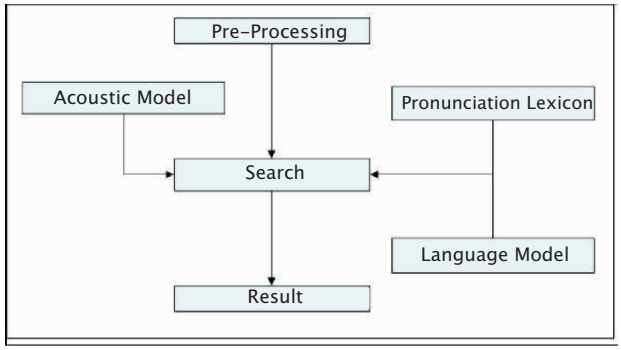


Figure 1. A schematic view of a speech recognizer.

engines [8].

3. Speech Recognition

Speech recognition by machine is the process computers apply to convert speech signals into written information[19]. Researchers have been investigating howto make machines to recognize speech and respond accordingly to it for quite some time. At first glance it was supposed not to be a difficult task. Time showed how difficult the task was. It took almost 40 years of intense researches to bring speech recognition systems to the shelves. And it still has its limitations. Human speech can vary in frequency and time (women has a pattern of voice that is different from men) and the same phonemes can be longer for one person and shorter for another one. Besides, the acoustic evidences of a sound may suggest the utterance of few different phonemes. As the problem is more severe with continuous speech uttered by different people, researchers tried for quite some time to recognize isolated phonemes (and then, words) of one speaker. hen isolated words where tried. Speech recognition by machine had a huge breakthrough when Kai-Fu Lee [16] proposed in 1988 a large vocabulary (1,000 words) speaker independent (many speakers) continuous speech recognizer with excellent performance. The solution (based on Hidden Markov Models) "learned how to guess" (based on statistics) what phonemes were more likely to have been spoken and aided by a lexicon and a grammar could point the most likely phrase a speaker said.

A simplified representation of a generic speech recognizer can be seen in Figure 1. The acoustic models are similar to "hash codes" of the acoustic signals. They are more than that as they statistically represent the variations of a acoustic signal across a large number of speakers. The pronunciation lexicon is the dictionary, i.e., how to connect acoustic signals in order to get a word. An example of a pronunciation lexicon entry is *economics » Ek∗nAmIks*.

The issue is complex but it is worth mentioning that one phoneme is usually made by more than one acoustic evidence. In order to improve recognition performance, companies decided to simplify the process (in two different ways) by reducing the complexity of the system in order to have commercial products. The main dimensions of the problem are:

- Speaker (how many speakers the system is able to cope with);
- Noise (how robust the system is to environment noise);
- Size of the vocabulary (how many words to recognize);
- Type of the lexicon (phonemes or words);
- Type of the speech (continuous speech or isolated word).

The first dimension indicates how well the recognizer behaves for a diversity of speakers. Speaker-dependent systems recognize well for a specific speaker. Commercial speech recognizers cannot target this dimension directly as the systems should be sold to many people. Indirectly, this dimension can be reduced if speaker adaptation is used (the system adapts its internal models to improve recognition for a particular user). In fact, a good dimension to focus on is the size of the vocabulary. As the number of words increases the recognition accuracy decreases. Keeping this dimension down is necessary to obtain good results. This is not a reasonable task when the system has to recognize a speech having no clue what so ever about the subject of the speech (or conversation). However, this is perfectly possible in a controlled conversation. By "controlled conversation" we mean a conversation were the computer controls the flow of the dialog. An example would be a purchase of some piece of clothe. Imagine a human-computer dialogue in a clothes store:

- Computer: What would you like to see?
  expecting shirts, t-shirts, paints
- Person : A t-shirt, please!
- Computer: What size?
  expecting small, medium, large, very large
- Person: Medium!
- Computer: Do you have any color in mind?
  expecting yes/no or colors
- Person: I would like red!

Given the context, this is a normal conversation that any commercial speech recognition system would have with a person. Actually, it is the context that keeps the number of words down (controlling one dimension of

the problem). As this dimension is controlled, this means a hint should be given to the system at any time the recognition is requested. Companies call this a "speech grammar". It is a set of words that are expected to be uttered by a person (at some time of the conversation) in order the computer recognizes the command. Two different systems may have different "speech grammar" syntaxes.

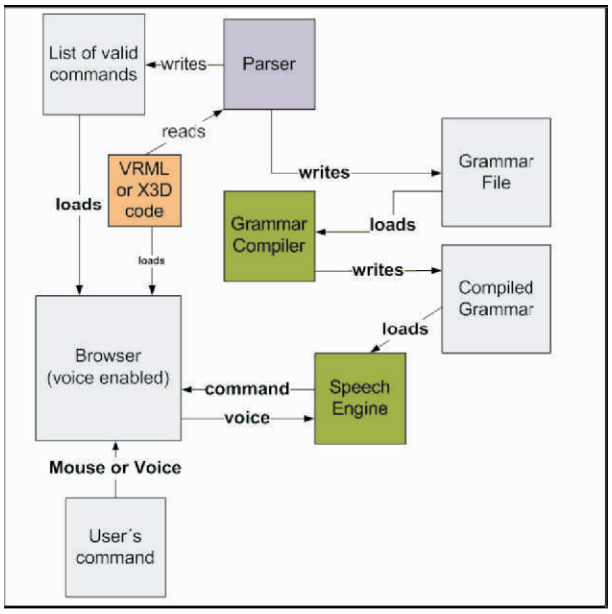


Figure 2. Proposed Software Framework.

4. A Sofware Integration Framework

As already mentioned, the goal of this work is to simplify the task of integrating a speech recognizer into a 3D browser running a VRML or X3D model. In order to build a prototype system, we have adopted the Nuance Speech Recognizer (version 8.0), a commercial product [20]. That was mainly due to its robustness, and availability of several language databases, in particular portuguese, as this is the language used for most of our usability tests. Although not complicated, the standard process of creating a grammar to be recognized by this system is essentially manual, requiring recompilation for every change made. The proposed framework (Figure 2) automates this process. A VRML/x3D programmer will create a program to work as if there was no voice commands. But, he/she will also provide hidden commands that will be useful to make an adapted browser to listen (and respond) to voice commands. All this hidden commands can then be processed by the parser that will interpret all the hidden commands creating a file with a list of valid voice commands. This list (or lookup table) will be used by the browser to trigger a specific script when a valid command is uttered. The parser is also responsible to create a grammar file that will be used for the speech engine. Usually this file needs to be compiled to be further used by the speech engine. Therefore, the programmer only needs to create a command (like "select the sphere") to work as if the mouse was clicked on the sphere and put it in the right place at the code. Given a compiled grammar and a voice input, a speech engine gives back its interpretation of the utterance. What we also had to do was to make the speech engine and the 3D browser to talk (including a listener component inside the browser - see figure 3). With that, the browser can receive a command from the speech engine, checks it against the list of valid commands, and process it by changing the scene. It is also possible to add some pre-recorded voice information to guide users, telling them what command was understood, or that the command was not recognized.

In order to enable the voice part of this bimodal system, the speech recognizer needs to be up and running. In order to function, it needs a valid license and a compiled grammar. To produce a compiled grammar, the grammar compiler needs a valid license as well. Therefore, in aWindows environment a batch file should be built in order to start the programs in a correct order (including the parser). The whole process would be similar in a Unix/Linux environment.

4.1. A new listener component

An important step to build the software framework was to implement a component inside the Xj3D browser in order to listen to the recognizer and understand its codes. The new component architecture of the browser is shown in Figure 3.

The listener waits for a voice signal coming from the speaker. Then, it sends this signal to the speech recognition engine and waits for its response. When the command returns from the speech engine, then the listener verifies if it is a valid one (looking at the list of valid commands previously built by a parser). This list is loaded by the listener when the browser starts. It is worth mentioning once more that this list is based on commands inserted by the programmer into the VRML/X3D code. If the speaker says something wrong, then the recognizer would send back to the listener a specific code informing that it was not possible to recognize that voice input. If it is a valid command, then an event is generated. The type of event to be generated also comes from this list and it is chosen by the programmer. For instance, it can be a touchTime event (when the current time is thrown).

The listener knows the correct object to send the event to. Once the object receives it, it is as if the mouse was clicked. From the programming viewpoint, it is important that the listener inherits properties and characteristics from the Event Model Evaluation component. It is this inheritance that will make scene change possible. In order to illustrate how it works, lets imagine a virtual world with a single cube (Figure 4). The only action this cube can perform (to keep it simple) is to change its color (but we could move it around, enlarge it...). To accomplish it only by using a mouse, the programmer would have to create boxes with different colors and would have to make them sensors (up right corner of figure 4). Then, one would have to click on one of these color boxes to choose a desired color and then would have to click on the object in order to assign the color to the object (to generalize, as it may be possible to have more than one object in the scene).

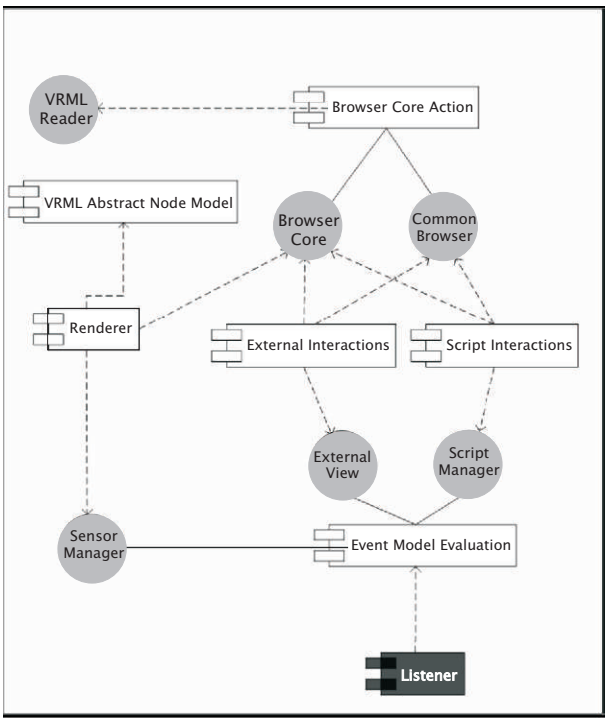


Figure 3. Component view of an Xj3D browser with a speech listener.

To perform the same task using a speech interface, one would have to say "choose red" and then "Paint cube", i.e., two spoken commands to replace the two clicks in the mouse. In order to do that, a VRML programmer (X3D is very similar) would have to insert a few hidden commands inside the code (as comments between #@ and @#), as shown in the example next:

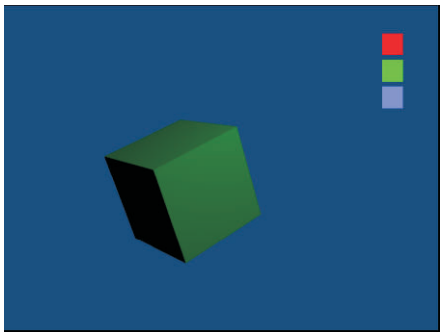


Figure 4. Simplified view of a virtual environment.

```
                #@ touchTime: "Paint cube"
                  command applyColor @#
    }
    Shape
    {
          appearance Appearance
          {
               material DEF Object Material
               {
                    diffuseColor 1 0 0
               }
          }
          geometry Box {}
    }
]
```

Similarly, for each little box with a different color (the sensor boxes) the programmer would have to add a hidden command (also between #@ and @#) like the following one:

```
    children
    [
          .........
          DEF RedSensor TouchSensor
          {
                #@ touchTime: "Choose red"
                    command setRed @#
          }
     .........
]
```

With the new#@touchTime... @#entry inside the RedSensor, the parser can create i) a valid entry for the "speech grammar" file and ii) a valid entry for the list of valid commands. The list of valid commands (used by the proposed new component) is important as it creates a relationship between sensors and voiced commands. The parser creates this relationship as it knows the position of the entry in the VRML code. In the example, "choose red" is the voice command for the RedSensor that act as a touch sensor. SetRed is the command the speech recognizer will send back to the browser if this utterance is recognized and then, the proper actions will be performed by the browser when the listener throws the correct event. As previously discussed, the grammar (used by the speech recognizer) is also very important as it reduces the amount of

words a system has to recognize at a given moment. As an example, a valid grammar file for the example would be:

```
Command
[
   (choose red) {<command "setRed">}
   (choose blue) {<command "setBlue">}
   (choose green) {<command "setGreen">}
   (paint cube) {<command "applyColor">}
]
```

Finally, the original ChangeColor script written for the mouse (without considering speech recognition) should be kept intact, as shown bellow:

```
DEF ChangeCor Script {
   inputOnly SFTime setRed
   inputOnly SFTime setGreen
   inputOnly SFTime setBlue
   inputOnly SFTime applyColor
   outputOnly SFColor setColor
   initializeOnly SFColor color 1.0 0.0 0.0

   url ["ecmascript:
   function setRed(value, timestamp) {
   color[0] = 1;
   color[1] = 0;
   color[2] = 0;
}

function setGreen(value, timestamp) {
   color[0] = 0;
   color[1] = 1;
   color[2] = 0;
}

function setBlue(value, timestamp) {
   color[0] = 0;
   color[1] = 0;
   color[2] = 1;
}

function applyColor(value, timestamp) {
   setColor = color;
}
 "]
}
ROUTE RedSensor.touchTime
   TO ChangeColor.setRed
ROUTE BlueSensor.touchTime
   TO ChangeColor.setBlue
ROUTE GreenSensor.touchTime
   TO ChangeColor.setGreen

ROUTE ObjectSensor.touchTime
   TO ChangeColor.applyColor
ROUTE ChangeColor.setColor
   TO Object.diffuseColor
```

The very same actions would work if we had a virtual environment with more objects (like in figure 5). In this case, one would only have to add the voice commands to select each object (as the other commands we have already shown). One could replace spheres, cubes, cones and cylinders by cars and transform it into a commercial application (some companies like Nissan or Fiat have similar applications at their sites). This framework would allow a programmer to easily add voice commands into such environments and the user would be able to pick a car model and then change its color using either speech or mouse.

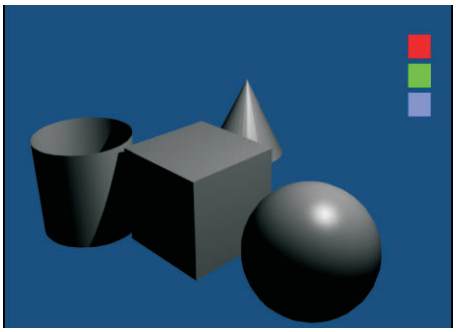


Figure 5. A virtual environment with more objects.

5. Conclusions

We have proposed a new browser architecture (based on the Xj3D) for VRML and X3D programs in order to make programmer's life a bit easier when he/she decides to have a voiced-enabled virtual world. The simple commands are hidden in the VRML/X3D code so as it is compatible with the current browser version being used. Along with a new listener component, a parser has also been developed in order to identify the hidden commands and create the necessary configuration files for the system (speech engine and browser). We have successfully tested our proposal for some simple 3D virtual worlds. From the object selection viewpoint we are currently investigating more complex environments (similar objects in the same scene). We would like to introduce some intelligence into the whole process in order the end user to choose an object by simply saying "the sphere on the left corner" or "the red cube" and so on.

6. References

[1] F. Althoff, H. Stocker, G. McGlaun, and M. K. Lang. A generic approach for interfacing vrml browsers to various input devices and creating customizable 3d applications. In *Proceedings of Web3D 2002 - 7th International Conference on 3D Web Technology*, pages 67–74, Tempe, Arizona, USA, 2002.

[2] R. Arangarasan and G. N. P. Jr. Modular approach of multimodal integration in a virtual environment. In *Proceedings of ICMI m2002 - 4th IEEE International Conference on Multimodal Interfaces*, pages 331–336, 2002.

[3] R. A. Bolt. "Put-That-There": Voice and gesture at the graphics interface. In *SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and*

interactive techniques, pages 262–270, New York, NY, USA, 1980. ACM.

[4] A. Cheyer and L. Julia. Spoken language and multimodal applications for electronic realities. Virtual Reality, 4(2):114–128, 1999.

[5] K. Christian, B. Kules, B. Shneiderman, and A. Youssef. A comparison of voice controlled and mouse controlled web browsing. In *Assets '00: Proceedings of the fourth international ACM conference on Assistive technologies*, pages 72–79, New York, NY, USA, 2000. ACM.

[6] W. Consortium. X3d & related specifications. http://www.web3d.org/x3d, 2007.

[7] W. Consortium. Xj3d - java based x3d toolkit and x3d browser. http://www.web3d.org/x3d/xj3d/, 2007.

[8] W. W. W. Consortium. Speech recognition grammar specification version 1.0. http://www.w3.org/TR/speechgrammar/, 2007.

[9] W. W. W. Consortium. Voice extensible markup language (voicexml) version 2.0. http://www.w3.org/TR/voicexml20/, 2007.

[10] K. S. (Editor). Handbook of Virtual Environments: Design, Implementation, and Applications. CRC, 2002.

[11] R. Haller, H. Mutschler, and M. Voss. Comparison of input devices for correction of typing errors in office systems. In Proceedings of INTERACT '84, First IFIP Conference on Human-Computer Interaction, 1984.

[12] A. G. Hauptmann and A. I. Rudnicky. A comparison of speech and typed input. In HLT '90: Proceedings of the workshop on Speech and Natural Language, pages 219–224, Morristown, NJ, USA, 1990. Association for Computational Linguistics.

[13] G. Herzog, A. Ndiaye, S. Merten, H. Kirchmann, T. Becker, and P. Poller. Large-scale software integration for spoken language and multimodal dialog systems. Natural Language Engineering, 10(3-4):283–305, 2004.

[14] L. Karl, M. Pettey, and B. Shneiderman. Speechactivated versus mouse-activated commands for word processing applications: An empirical evaluation. Internionalal Journal of Man-Machine Studies, 39:667–687, 1993.

[15] A. J. F. Kok and R. van Liere. A multimodal

virtual reality interface for 3d interaction with vtk. Knowledge and Information Systems, 11(3), 2007.

[16] K.-F. Lee, H.-W. Hon, and R. Reddy. An overview of the SPHINX speech recognition system, pages 600–610. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[17] J. Leggett and G. Williams. An empirical investigation of voice as an input modality for computer programming. Int. J. Man-Mach. Stud., 21(6):493–520, 1984.

[18] S. Mcglashan. A speech interface to virtual environments. citeseer.comp.nus.edu.sg/94372.html.

[19] M. F. McTear. Spoken dialogue technology: enabling the conversational user interface. ACM Computing Surveys, 34(1):90–169, 2002.

[20] I. Nuance Communications. Nuance recognizer. http://www.nuance.com/recognizer/, 2007.

[21] S. L. Oviatt. Advances in robust multimodal interface design. IEEE Computer Graphics and Applications, 23(5):62–68, 2003.

[22] R. Pausch and J. H. Leatherby. A study comparing mouseonly input vs. mouse-plus-voice input for a graphical editor. Journal of American Voice Input, 9(2), 1991.

[23] G. K. Poock. Voice recognition boosts command terminal throughput. Speech Technology, 1(2):36–39, 1982.

[24] R. Sharma, M. Zeller, V. Pavlovic, T. S. Huang, Z. Lo, S. M. Chu, Y. Zhao, J. C. Phillips, and K. Schulten. Speech/gesture interface to a visual-computing environment. IEEE Computer Graphics and Applications, 20(2):29–37, 2000.

Using the Hanoi Puzzle to Evaluate the Usability of Speech Interface on Virtual Environments

Ednaldo Brigante Pizzolato, Cleber Leonel C. de Lima
Universidade Federal de São Carlos
Departamento de Computação
São Carlos - SP - Brazil
ednaldo@dc.ufscar.br, heiligritter@gmail.com

Abstract

Advances in software techniques and processing power over the last decades have finally enabled speech recognition to be effectively used as a viable man-machine interface. This paper describes preliminary work on exploring effectiveness of voice commands and mouse interaction to solve the Hanoi problem. Our first results indicate that performance of speech and other kinds of human-computer interaction (mouse and button) are compatible and that speech-enabled virtual reality applications could be used to train impaired people to perform complex tasks today performed only by people with hand abilities.

1. Introduction

Technology constraints have forced the humans to get used to mechanical devices (such as keyboards and mice) in order to interact with machines. These constraints have prevented impaired people from performing certain tasks. The next generation of human-computer interface system applications will probably abandon keyboards and mice, replacing them by more natural means such as speech and gesture recognition. There is no doubt the users would be more comfortable with this interface. In fact, Ogozalek and Praag investigated in 1986 users preferences and concluded that speech was greatly preferred over keyboards as a mean of communication between humans and computers [6]. Many other investigations pointed to the same direction. Speech recognition technology evolved greatly since the early 1950´s but still has some way to go in order to be freely applied as an human-computer interface. Our current research aims at investigating the use of speech recognition in virtual reality applications. In this paper, we present a tool to evaluate the usability of a 3D system using either mouse or voice-enabled commands to solve the Tower of Hanoi puzzle (invented by a French mathematician Edouard Lucas in 1883). A tower of some disks needs to be moved from one peg (the first one) to a third peg using a second peg to help moving the disks and keep the rule of an increasing size stack (a smaller disk is always on top of a larger one). In other words, the objective is to transfer the entire tower from the first peg to the third peg moving only one disk at a time and never a larger one onto a smaller. The problem was picked as it resembles the harbor operations (take on or discharge cargo) which needs some sort of decision making.

2. Previous Work

Researchers have been investigating typed commands against voice-activated commands since early 1980's. For instance, Poock used a simple speech recognizer to compare the speed and accuracy of speech input versus typed input of commands in a simulated military command and control application over a distributed network [8]. Using a small vocabulary (180 words), a recognition accuracy of 96.8% was achieved by novice users when using voice input. The results also showed that voice input was 17.5% faster than typed input, and that typed commands had 183.2% more errors than voiced ones. But other studies [5, 2] showed that keyboard inputs could be faster than voice inputs (however, with more errors). As seen, the early studies in the 1980's seem to be non-conclusive regarding performance. In 1993, Karl et al. compared mouse and voice inputs as an interface for a word processor application [4] . They showed that, on average, the task time was 18.67% quicker when speech interface was applied. Overall, in their experiments, the subjects reacted positively to using speech input and preferred it over the mouse for command activation (confirming Ogozalek and Praag findings of 1986).

In another direction, Pausch and Leatherby [7] investigated the improvement voice could bring to systems where mouse and keyboard could also be employed. In general, they concluded that the task completion time could be reduced over 10% when speech and mouse were employed together.

[1] We thank NUANCE (www.nuance.com) for allowing us to use their speech recognition software to perform the experiments. Special thanks go to Rodrigo. We also thank CNPq for sponsoring the research through UFSCar-PIBIC grant.

Hauptmann and Rudnicky pointed out that users may had lost their attention while waiting several seconds for the system's response and assumed that in the future speech could perform much better [3]. More recently, Christian et al. [1] showed that voice control adds approximately 50% to the performance time for certain types of tasks. Current computer's processing speed and speech recognition accuracy push us to update previous evaluations on this subject. Particularly, the interaction with virtual environments.

3. The HanoiSpeech Toolkit

We have designed a simple 3D environment (HanoiSpeech) that aims only at analyzing the speech inputs and at comparing it with other kinds of inputs (measuring the time to complete the task). The toolkit (written in Java3D) controls speech and mouse inputs and the graphical interface. All speech signals are sent to the speech recognition engine (NUANCE version 8.0) which delivers back a text that most likely matches the utterance. Figure 1 shows the toolkit. There is a horizontal bar where a magnet can go back and forth. Under the magnet are 3 pegs. The leftmost magnet has some disks which need to be moved to the third peg. On the right side of the screen is the command area. People can register into the toolkit and play.

Each play has its total time recorded (along with the number of the movements). If the user chooses to control the environment with the mouse, then he/she would have to click on the peg in order to bring the magnet over it. Then, he/she would have to click on the disk in order to activate the magnet (switch it on). Clicking on another peg makes the magnet moves towards it and so on. If the user chooses to control the process with buttons, then he/she would have to click on the buttons on the right to choose a peg and to activate or deactivate the magnet. By choosing to control the actions through speech, the user would have to say the commands. There are two possible situations for that: a sequence of single commands or a full command. When one chooses to issue a sequence of commands, he/she would have to say commands like "move the magnet to the first peg" or "switch the magnet on". If a full command is chosen, then the user can say "pick up the disk in the first peg and move it to the second peg". The first scenario assumes the computer program can control the movements and has stored the precise positions the magnet would have to go. It would be only a decision problem where the human decides and the computer takes the action.

Another scenario was also investigated where the human had to be responsible for the correct

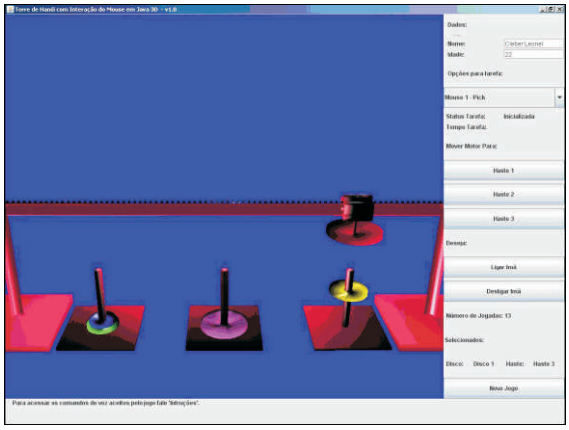


Figure 1. HanoiSpeech GUI.

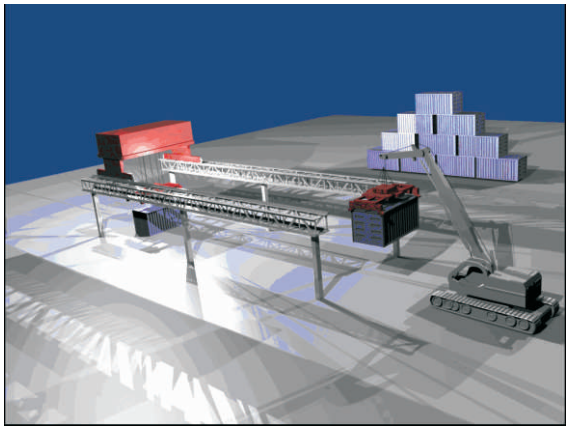


Figure 2. Harbour Scenario.

positioning of the magnet. Written information about the position of the pegs and the current position of the magnet were given to the participants and they had to move the magnet through speech commands (one experiment) or through mouse clicks (another experiment). The speech commands could be general instructions like "move the magnet X units to the right" or fine-tuned instructions like "move the magnet just a bit to the left". As one would assume, it is a more complex task than the one presented in the first scenario as the responsibility to position the magnet correctly is given to the user. This scenario is interesting as it resembles the harbor problem (Figure 2) of picking containers from one place and putting them elsewhere.

Eleven subjects (6 male and 5 female) volunteered for the experiments. None had previous contact with speech enabled systems and all of them had considerable experience with mouse. The experiment was conduct on a Pentium IV

(1 GHz) with 1 GB of RAM running Windows XP Home. The screen had a diagonal size of 17" and the

display resolution was set to 1024 to 768. A simple microphone was used without any noise cancellation capability. Each subject was tested individually in a 12 m² room. As previously stated, the system automatically computes the task time for each experiment and stores it in a data base.

4. Experimental Results

The results of the first scenario (Figure 3) indicate that speech, mouse and buttons can produce similar task times. To complete the first task (mouse only) users took an average time of 2 minutes and 39 seconds. They took around 2 minutes and 34 seconds to perform the same task with buttons and 4 minutes and 40 seconds to perform it through a sequence of speech commands. As it can be seen, one would say speech is not efficient as it takes almost twice the time of the mouse to perform the same task. However, when a full command was employed, the task time came around the figures showed by mouse or buttons (2 minutes and 39 seconds).

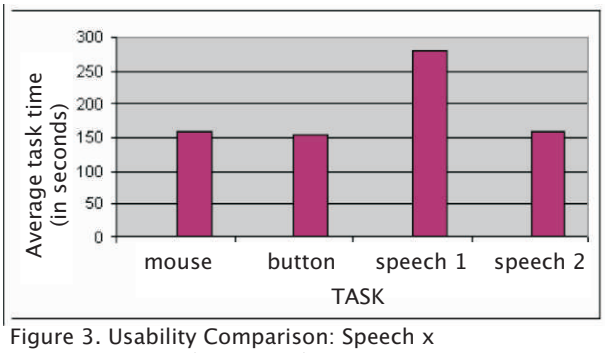


Figure 3. Usability Comparison: Speech x Mouse x Buttons (scenario 1).

The results of the second scenario indicate that people position the magnet better with mouse than with speech commands. The task took around 4 minutes and 28 seconds to be completed with mouse commands and 6 minutes and 14 seconds to be completed with speech. One reason for that is that decimal points imply in more speech commands like "move the magnet just a bit to the right (or left)". Another reason is that users have to calculate how many units the magnet needs to move before issuing the speech command. Such calculation is not necessary when using the mouse. But it shows that even under these circumstances, speech can still be employed as it is about 1.5 mouse time.

In a real world scenario, trained people may match or even outperform the mouse results.

5. Conclusions

The current experimental results show that speech can be employed as an efficient user interface for some tasks. People should be aware that tasks like the ones in scenario 2 may take longer. More scenarios should be investigated in order to better evaluate when mouse is better than speech and why. So far, we verified that the subjects demonstrated satisfaction in using speech to interact with computers and that task time may be very similar to mouse or button based tasks. This is a very good news as: a) people may demand more speech-enabled software applications in order to prevent themselves from getting ill due to automatic repetitive movements; and b) injuries in hands may not bring people out of the market. This should encourage the development of speech-enabled systems and the development of speech-enabled virtual reality systems to train people to these new systems.

6. References

[1] K. Christian, B. Kules, B. Shneiderman, and A. Youssef. A comparison of voice controlled and mouse controlled web browsing. In *Assets '00: Proceedings of the fourth international ACM conference on Assistive technologies*, pages 72–79, New York, NY, USA, 2000. ACM.

[2] R. Haller, H. Mutschler, and M. Voss. Comparison of input devices for correction of typing errors in office systems. In *Proceedings of INTERACT '84, First IFIP Conference on Human-Computer Interaction*, 1984.

[3] A. G. Hauptmann and A. I. Rudnicky. A comparison of speech and typed input. In *HLT '90: Proceedings of the workshop on Speech and Natural Language*, pages 219–224, Morristown, NJ, USA, 1990. Association for Computational Linguistics.

[4] L. Karl, M. Pettey, and B. Shneiderman. Speechactivated versus mouse-activated commands for word processing applications: An empirical evaluation. Internionál Journal of Man-Machine Studies, 39:667–687, 1993.

[5] J. Leggett and G. Williams. An empirical investigation of voice as an input modality for computer programming. *Int. J. Man-Mach. Stud., 21(6)*: 493–520, 1984.

[6] V. Ogozalek and J. V. Praag. Comparison of elderly and younger users on keyboard and voice input computer-based composition tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 205–211, New York, NY, USA, 1986. ACM.

[7] R. Pausch and J. H. Leatherby. A study comparing mouseonly input vs. mouse-plus-voice input for a graphical editor. *Journal of American Voice Input, 9(2)*, 1991.

[8] G. K. Poock. Voice recognition boosts command terminal throughput. *Speech Technology, 1(2)*: 36–39, 1982.

GESKTOP 3D: Explorando novas formas de interação 3D baseadas em gestos

Daniel Tokunaga, Alexandre Vicentim, Marcos Muto, João Bernardes, Romero Tori
Escola Politécnica da Universidade de São Paulo,
Department of Computer Engineering and Digital Systems
{daniel.tokunaga} , {alexandre.vicentim}, {marcos.muto}, {joao.bernardes}, {romero.tori}@poli.usp.br

Abstract

This paper presents the research, development and testing of a 3D virtual desktop with interaction through movements of the user's hand, without any devices except for a webcam.

Resumo

Este artigo apresenta a pesquisa, desenvolvimento e testes de um desktop virtual 3D com interação por movimentos da mão do usuário, sem quaisquer dispositivos exceto por uma webcam.

1. Introdução

Atualmente, pode-se notar uma crescente demanda por formas não tradicionais de interface com o usuário. Na busca por melhorias na interatividade e em interfaces 3D [1], novos dispositivos são criados como, por exemplo, o bem sucedido console de videogame Wii, da Nintendo, que faz o uso de gestos com um bastão para o controle dos jogos.

Nesse contexto, observa-se o potencial do uso de gestos como forma de interação, uma vez que isso pode tornar a interface mais intuitiva e transparente, diminui a curva de aprendizado da aplicação e mantem o foco do usuário em suas tarefas, de modo que a sobrecarga cognitiva provocada pelo uso de dispositivos seja minimizada, ou seja, o usuário não precisa parar suas ações para pensar em como executá-las [2].

Este trabalho apresenta um sistema, denominado Gesktop3D, que consiste em um ambiente de trabalho de organização de arquivos e programas, um "desktop", manipulado através de gestos manuais e com uma interface em três dimensões, de forma que o modo de interação e a interface se aproximem de um ambiente real, sendo possível mover livremente os arquivos dentro do ambiente, inclusive para empilhá-los.

Para o desenvolvimento da aplicação são utilizadas duas ferramentas principais: o enJine, um framework de criação de jogos[3] para o desenvolvimento do Desktop3D, e o HandVu[4], uma API utilizada para rastreamento dos movimentos da mão[5] e reconhecimento de posturas[6].

2. Visão geral do sistema

O Gesktop3D consiste em uma mesa que contém blocos, que representam os arquivos e programas. A interação com o sistema pode ser realizada através de três diferentes modos: pela utilização dos movimentos da mão do usuário (modo gestos), pelo teclado ou pelo mouse.

Para a implementação do modo gestos do sistema foi utilizado a biblioteca HandVu que possibilita o traking da mão e a detecção de posturas[1] possibilitando a interpretação de alguns gestos. As posturas reconhecidas são mostradas na tabela 1.

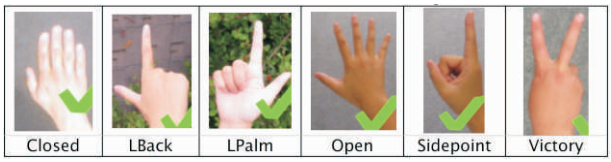


Tabela 1. Posturas reconhecidas pelo HandVu.

Os blocos serão identificados na face superior com uma imagem representando o programa que abre o arquivo (essa imagem será a mesma que é utilizada nos ícones do sistema operacional Windows) e nas laterais pelo nome do arquivo, de forma semelhante a livros no ambiente físico. Os arquivos podem ser movidos dentro do ambiente de trabalho para qualquer lugar do espaço, para isso é necessário selecionar objeto e utilizar as setas do teclado ou o mouse para a movimentação no plano do desktop e a tecla "A" para levantar ou "Z" para abaixar o arquivo ou, no modo de gestos, movimentar a mão no espaço com a postura "LBack"

A fim de melhorar a organização da mesa é possível empilhar arquivos seguindo critérios escolhidos pelo usuário como mesmo tipo de arquivos, músicas, por exemplo, ou dentro de um mesmo contexto como

[1] Estado da mão e do antebraço em um instante de tempo, de modo que nesse estado pode-se definir claramente a posição e inclinação no espaço da mão, bem como o posicionamento dos dedos e da palma da mão [2].

arquivos da monografia que compreendem arquivos de texto, diagramas, imagens e cronograma. A figura 1(a) apresenta o Gesktop3D com alguns arquivos empilhados. Para criar a pilha é preciso selecionar os arquivos que se deseja empilhar e pressionar a tecla "2" e depois a tecla "A", caso se esteja utilizando o modo teclado ou mouse, ou realizar a postura "Open" e levantar a mão, no modo gestos.

Para facilitar a visualização das pilhas e blocos, o ponto de vista (ou câmera) pode ser rotacionado e movimentado no eixo vertical e que, além disso, tem um mecanismo de zoom. Para movimentar a câmera modo de gestos, é necessário realizar a postura "Victory" e movimentar a mão no espaço.

(a) Utilização do Gesktop3D

(b) Visualização de arquivos em pilha.

Figura 1. Gesktop3D.

Também é possível mover as pilhas como os arquivos e há um modo para visualização do conteúdo da pilha, sendo possível selecionar parte da mesma, como mostrado na figura 1(b).

3. Desenvolvimento

A arquitetura da aplicação é mostrada no diagrama de classes e pacotes da figura 2.

DesktopState é a classe principal do desktop, sendo responsável por determinar a interface de interação (teclado e mouse, HandVu ou teclado e HandVu) e aplicar a cada sensor sua respectiva classe de ação (InputAction). Esse pacote também é responsável por iniciar as configurações do sistema e chamar os métodos de criação de objetos na cena. Já os controles de visualização deste ambiente virtual, como movimentação de câmera, funções de zoom in e zoom out, são exercidos pela classe Vision.

Desktop é o pacote que possui as classes de manipulação e visualização da "mesa". Os objetos, contidos na mesa, são representados pela classe DesktopObject. Esta classe possibilita a abstração dos objetos, uma vez que algumas funções de manipulação são aplicáveis a todos os objetos, desta forma é possível encapsular os métodos comuns.

File, filha de DesktopObject, possui as classes de manipulação e visualização de arquivos do desktop. Já Stack, também filha de DesktopObject, contém as classes de controle de pilhas de arquivos do desktop que se caracteriza, basicamente, por uma lista de arquivos que podem ser controlados simultaneamente.

As classes contidas no pacote Hand são responsáveis pela manipulação desses objetos do desktop e a representação visual da mão na mesa.

Por fim, Config é o pacote que concentra as classes de configuração do sistema e de ações sobre o sistema.

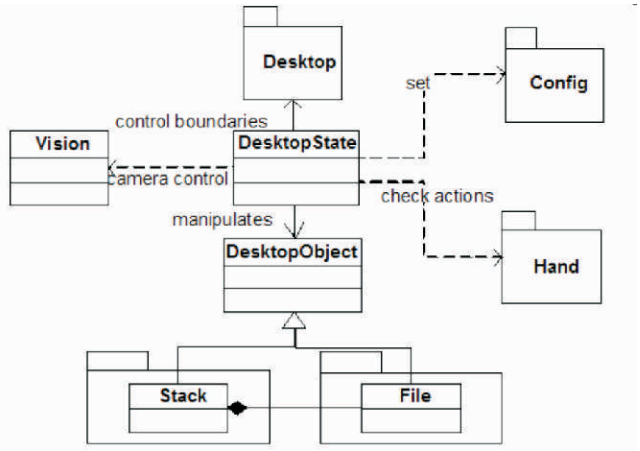


Figura 2. Arquitetura do Gesktop 3D.

4. Testes com usuários

No início do desenvolvimento do projeto foram realizados testes para verificar qual seria o tempo médio para o reconhecimento de uma postura da mão pela primeira vez, para o teste foi utilizado um computador com um processador Core 2 Duo T5500 1.66 GHz com 2GB de memória RAM e uma placa de vídeo onborad de 128MB. Todos os testes foram realizados no mesmo ambiente, em horários próximos e com a mesma condição de iluminação.

Após a conclusão do desenvolvimento, planejou-se um teste de usabilidade para verificar as vantagens e desvantagens do uso do Desktop3D desenvolvido (em comparação com Desktops 2D tradicionais) e a adequação do uso de gestos como forma de interação do sistema.

Para a realização desse teste foram adotados alguns requisitos [7]: utilização de um laboratório especial para testes que possua duas salas, sendo que uma delas será utilizada para os testes e a outra para observação. Essas salas devem estar separadas por um espelho, possibilitando a observação do usuário que está realizando o teste pelo observador que se encontrará na outra sala. Essa separação é utilizada para que o usuário

sinta isolado e para encorajá-lo a realizar comentários em voz alta. Também é necessário filmar a realização do teste e armazenar essa filmagem. Todas as ações realizadas pelo usuário no sistema devem também ser armazenadas em logs.

Para esse teste inicial o público alvo é formado por usuários com idade entre 20 e 30 anos. O cenário inicial do teste possui: 10 arquivos de música pertencentes a 3 bandas diferentes (x1, x2, x3, x4, y1, y2, y3, z1, z2, z3), 3 arquivos com letra de uma das 10 músicas em cada um deles (x1, y2, z3) e outros seis arquivos aleatórios que não podem ser nem de música nem de letras de músicas. As tarefas que devem ser cumpridas nesse projeto são: agrupar todos os arquivos de música, separando-os dos demais arquivos; separar desses arquivos de música que foram agrupados o arquivo x1.musica; agrupar em um local diferente todos os arquivos de música que começam com y, separando-os dos demais; agrupar os arquivos x1.musica e x1.letra; executar o arquivo x1.musica e abrir o arquivo x1.letra; executar todos os arquivos de música que tem o nome iniciado por y; reagrupar os arquivos de música, juntando no mesmo grupo todos os que começam com a mesma letra.

Baseando-se na ISO 9241 [8] e na ISO 9126 [9] os atributos escolhidos para serem avaliados nesse teste são: Operabilidade, medida através da quantidade de erros e do tempo para realização de cada tarefa. Apreendabilidade, medida através da quantidade de erros, do número de passos para realização de uma tarefa e pela quantidade total de tarefas realizadas.

Para realizar essas medições no sistema, definiu-se que erro será toda ação realizada pelo usuário que não leva a atingir o resultado esperado; que a contagem do número de passos será feita levando em consideração a realização das seguintes ações: selecionar e deselecionar objeto, troca da postura da mão, necessidade de reconhecer a mão novamente, empilhar e desempilhar arquivos, arrastar arquivos ou pilhas, alterar modo de visualização, mover a câmera mudando o ângulo de visão, mudança de arquivo no modo de visualização de uma pilha, execução de um arquivo.

O questionário pré-teste contém perguntas para verificar se o usuário realmente se enquadra no público alvo. O questionário pós-teste contém as informações para verificação a satisfação do usuário com o sistema.

Durante a fase de testes também foram tomadas medidas numéricas. Em um dos testes foram capturadas as posições retornadas em diferentes casos. Quando não se exerce uma postura e quando se exerce a postura "closed" sem movimentar a posição. Outra situação é de quando se altera a postura de mão de um estado onde todos os dedos são fechados para a postura "closed". A figura 3 ilustra esta situação descrita.

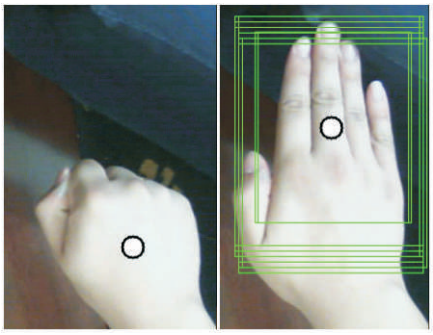


Figura 3 – Alteração da posição da mão devido à alteração da postura.

Outro teste exercido foi o de desempenho frente a diferentes configurações de hardware. Foi calculada a quantidade de quadros por segundo (FPS) de quando se detectava as 6 diferentes posturas em dois diferentes ambientes. O primeiro ambiente possui um processador Core 2 Duo T5500 1.66 GHz com 2GB de memória RAM e uma placa de vídeo on borad de 128MB, o segundo ambiente possui um processador Core 2 Quad Q6600 2.4GHz com 3.25GB de memória RAM.

5. Resultados

Através do teste inicial do tempo médio de reconhecimento de posturas foi possível verificar que o a postura Sidepoint é reconhecida com dificuldade, desta forma, essa postura não foi utilizada no projeto. Também se pôde concluir que o tempo levado para reconhecer o restante das posturas estava satisfatório.

Já nos testes com usuários, pode-se verificar que na maioria das tarefas e para a maioria dos usuários, o número de erros, o tempo para realização das tarefas e o número de passos são maiores quando se usa somente gestos quando comparado com os outros modos de uso (teclado, teclado e mouse, teclado e gestos). Analisando-se o questionário de satisfação, pode-se perceber que em todos os quesitos analisados o uso isolado de gestos apresenta desvantagem sobre os outros métodos de interação, mas trata-se de uma desvantagem pequena. Além dos resultados dos questionários, os comentários dos usuários registrados durante a realização dos testes levam a perceber que ficaram muito satisfeitos em utilizar o aplicativo em todos os modos, inclusive somente com os gestos.

Além de testes com usuários, algumas medidas de desempenho e precisão do sistema também foram

obtidas. Pode-se concluir pelos resultados obtidos que há uma instabilidade na detecção quando se exerce uma postura, enquanto que os valores capturados de quando não se exerce uma postura não variaram mais que 1% da média.

Outro resultado obtido é em relação à mudança da posição decorrida da mudança de postura. Devido ao fato de a ferramenta utilizada para detecção (HandVu) definir a posição da mão como sendo o centro da imagem da mão capturada, alterações consideráveis do formato da mão fazem com que este centro se altere, e assim alterando a posição da mão do usuário, mesmo sem que ele mova a mão.

Embora o planejamento dos testes formais de usabilidade, inclusive das métricas a serem utilizadas, tenha sido parte desse projeto, como descrito acima, a aplicação destes testes em uma população maior é um trabalho futuro. O detalhamento de cada teste, bem como da observação de sua execução, os resultados dos testes e sua análise devem ser por si só objetos de um futuro artigo e, portanto, não fazem parte do escopo desta comunicação.

6. Conclusões

Nesta comunicação apresentamos os primeiros resultados de um ambiente virtual de desktop 3D controlado através de gestos. Essa aplicação utiliza como ferramentas principais um engine didático de jogos, o enJine e o HandVu, aplicação que reconhece gestos e faz o rastreamento dos movimentos da mão do usuário. De modo a atingir os objetivo, ambas as ferramentas foram estudadas a fim de integrá-las usando a JNI, possibilitando o uso do HandVu como uma biblioteca no Java.

Baseado nos resultados encontrados foi possível verificar que o uso de gestos possui grande potencial para a interatividade em ambientes 3D, devido ao fato de que, diferente dos dispositivos usuais atuais como o teclado e mouse, o uso de gestos permite uma movimentação no espaço mais natural. Contudo, devido à dificuldade de reconhecimento da postura em tempo real pelo HandVu, principalmente em plataformas de hardware com menor capacidade de processamento, foi mais adequado utilizar o teclado como auxilio à interação por gestos. Esse tipo de interação pode ser usado, por exemplo, para substituir o uso do touchpad de um computador portátil quando não houver possibilidade de se usar um mouse.

O projeto tem ainda como possíveis trabalhos futuros a projeção da aplicação em uma mesa, utilizando Realidade Aumentada Espacial [10] e a infra-estrutura do Robot ARena [11], de modo a trazer as funcionalidades do desktop virtual para a realidade e tornar o ambiente mais natural para o usuário. Assim, o controle do desktop virtual ocorre pela própria mão do usuário, aumentando sua imersão. A utilização do *framework* formado pela integração do HandVu e do enJine para o desenvolvimento de outras aplicações 3D com interação por gestos traz diversas outras possibilidades de futuros trabalhos.

7. Referências

[1] D. Bowman et al., *3D User Interfaces: Theory and Practice*, Addison Wesley, 2004

[2] V. Pavlovic, R. Sharma and T. Huang, "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, 1997.

[3] enJine Project, available in: http://enjine.dev.java.net/, last accessed: Nov. 2007.

[4] HandVu, available in: http://www.movesinstitute.org/~kolsch/HandVu/HandVu.html, last access: Nov. 2007.

[5] M. Kolsch, "Vision Based Hand Gesture Interface for Wearable Computing and Virtual Environments", 2004.

[6] M. Kolsch and M. Turk, "Fast 2D Hand Tracking with Flock of Features and Multi-Cue Integration", *Proceedings of the Workshop on Computer Vision and Pattern Recognition 2004*, vol. 10, 2004.

[7] C. Lewis, J. RIEMAN, *Task-centered user interface design: a practical introduction*, 1994.

[8] Organização Nacional Para Padronização, *ISO 9241: Usability Standard*, 1998.

[9] Organização Nacional Para Padronização, *ISO 9126: Software Engineering – Product Quality*, 2001.

[10] R. Raskar, and O. Bimber, *Spatial Augmented Reality: Merging Real and Virtual Worlds*, A. K. Peters, 2005.

[11] D. Calife, A. Tomoyose, D. Spinola, J. Bernardes and R. Tori, "Robot ARena: Infrastructure for Applications Involving Spatial Augmented Reality and Robots", *Proceedings of the IX SVR*, SBC, Petrópolis, 2007.

Wearable Augmented Reality System Architecture: for Mobile Assistance and Training

Jayfus Doswell
Juxtopia, LLC - Baltimore. MD, USA
juxtopia@hotmail.com

Abstract

Wearable Augmented Reality (AR) systems have the potential to provide continuous and autonomous instruction and just-in-time assistance anytime, anyplace, and at any-pace. Wearable AR based assistance and instruction provides the advantage of a hands-free, humancomputer interface, flexible mobility, and contextawareness allowing end-users to request immediate assistance on task and develop psychomotor skills while interacting with their natural environment with augmented perceptual cues. These perceptual cues combining multimodal animation, graphics, text, video, and voice along with empirical instructional techniques can elegantly orchestrate a mobile assistance and training tool. The challenge, however, is building a Wearable AR based architecture with capabilities for adapting to various enduser needs and their environments ranging from outdoor environments to the workplace. This paper discusses a novel wearable AR architecture, to implement advance wearable e-training and e-assistance applications.

1. Introduction

Ubiquitous learning with a wearable (AR) system that provides on-demand instructional services requires a well engineered system/software architecture. Target applications generated from the architecture require capabilities for understanding individual learning strengths while tailoring empirically evaluated pedagogical techniques to enhance learning for children and adults, respectively. Additionally, this type of context-aware wearable AR system requires a natural human computer interface allowing end-users to focus on their tasks. Furthermore, in order to significantly impact learning and assist users, a wearable AR system needs to consistently evaluate how the individual performed on a task and measure learning progress while continuously updating information about the learner's profile for the duration of the interaction. Hence, a wearable (AR) system may continually capture and process visual and speech data associated to a given context and for a given individual to, consequently, improve the individual's performance on a specific task. The wearable (AR) system architecture should also be designed as an extendable, flexible, interoperable, reusable, and scalable infrastructure to support the integration of additional components without impacting the core system/software architecture design and to facilitate standard data exchange.

This paper discusses a novel wearable (AR) system architecture, Context-Aware Augmented Reality System (CAARS) architecture that was developed to support an extendable, flexible, interoperable, reusable, and scalable infrastructure for generating context-aware wearable (AR) systems for a variety of industries with capabilities of interoperating with a variety of AR Head Mounted Displays (HMD). Applications for a wearable (AR) system based applications may include, but are not limited to consumer, entertainment, industrial, manufacturing, medical, military.

2. Background

Augmented reality (AR) falls within Milgram's mixed reality continuum as illustrated in Figure 1.

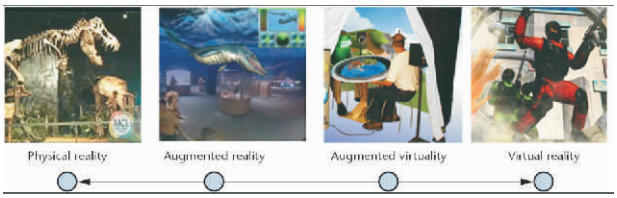


Figure 1. Milgram's Mixed Reality Continuum.

In augmented reality, digital objects are added to the real environment. In augmented virtuality, real objects are added to virtual ones. In virtual environments (or virtual reality), the surrounding environment is completely digital [7][8].

A Wearable or, commonly called, mobile augmented reality system (MARS) combines research in augmented reality (AR) and mobile/pervasive computing in which

a wearable see-through heads-up display and increasingly small computing devices facilitate data communication over wireless networks. The MARS system infrastructure allows users to freely visualize real world objects over which digital annotations may be superimposed. Additionally, a MARS provides flexible mobility and a location independent service without constraining the individual to a specific geographical location. By doing so, MARS technology holds the potential to revolutionize the way in which information is presented and has enormous potential for on-demand, context-aware, and collaborative assistance/training [4]. With this mobile assistance tool, individual and collaborate learning may be enhanced in areas ranging from K-12 history, science, technology, engineering, and math training to medical, manufacturing, military, and security workforce training. Additionally, just-in-time assistance may be delivered for a specific tasks in response to speech or gesture based requests.

MARS technologies are increasingly being used in domains including, but not limited to industrial, medical, and manufacturing areas [1][4][5][6][10].

The main human computer interface (HCI) component of Wearable MARS applications is a see-through digital display worn over a user's eyes (like goggles) that overlay the human visual field with text or computer graphics. This AR feature brings about an interactive experience by supplementing the real world, rather than create an entirely artificial environment as with the case in "Fish Tank" virtual reality. Thus, AR systems combine, and in real-time, align virtual objects with physical ones.

3. CAARS Architecture

Researchers designed and prototyped a context-aware and wearable AR system/software architecture, the *Context Aware Augmented Reality System* (CAARS). Components of the CAARS architecture that were designed and developed include the CAARS Goggles and CAARS Assistance Service.

3.1 CAARS Goggle Architecture & Interface

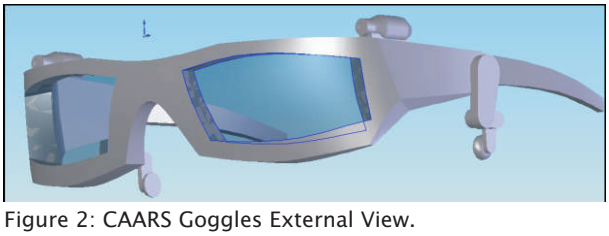


Figure 2: CAARS Goggles External View.

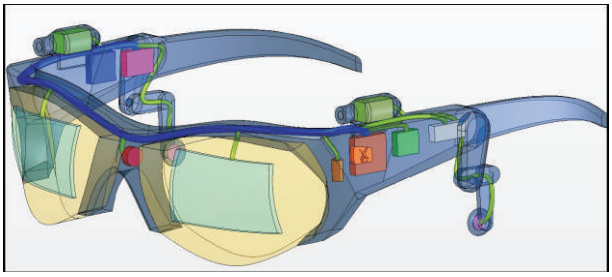


Figure 3: CAARS Goggles Internal View.

The CAARS goggles are depicted as a schematic representation in Figure 2 and Figure 3. The CAARS goggles were designed in a standard computer aided design and drafting (CAD) program. The Goggles were designed as a ruggedized, light-weight, head mounted display (HMD) for comfort and safety and with several major electrical components.

The CAARS Goggles were designed with several major components including, optical see-through display, miniantenna, mini-speakers, mini-microphone, lithium ion batteries, and mini-microphone. Considering the fact that current optical see-through displays are bulky and have awkward structures that minimize its commercial potential, research investigators designed a novel optical see through AR head mounted display that is light-weight, comfortable, and aesthetically pleasing. Research investigators chose the optical see-through displays as opposed to video seethrough display because optical see through displays preserves the real image while reducing degradation. Additionally, the CAARS goggles were designed with two miniature PC compatible cameras to capture visual elements in a learner's environment (e.g., equipment components, etc.). As images are captured by the camera, they are transmitted, wirelessly along with meta-data, in real-time, to the *CAARS Training Service for processing by software agents* for the multimodal delivery of training assistance. The cameras also are used to audit training sessions by recording multiple image frames during a tasks and analyzing corresponding meta-data to measure the result (in terms of productivity) of the training. A wireless antenna was integrated to function as a transceiver in order to transmit and receive digital information to and from the CAARS goggles, respectively. This allows the CAARS goggles to remotely transmit real-world images, position coordinates, head orientation coordinates, and sound from mini-cameras, inertia sensors, and mini-microphone inputsensors, respectively. Speakers were also integrated into the CAARS goggles to facilitate hands free commands of instructional material as well as to sound broadcast alerts and other information

relevant to a training session. The CAARS goggles were also designed with rechargeable lithium ion batteries for potential operation up to 8 hours.

As a result of the CAARS goggle design, learners/users may request various tasks assistance through simple voice commands and, as a result, visually see and hear computer based assistance through the optical see through display and mini-speakers, respectively.

3.2 CAARS Software Service Architecture

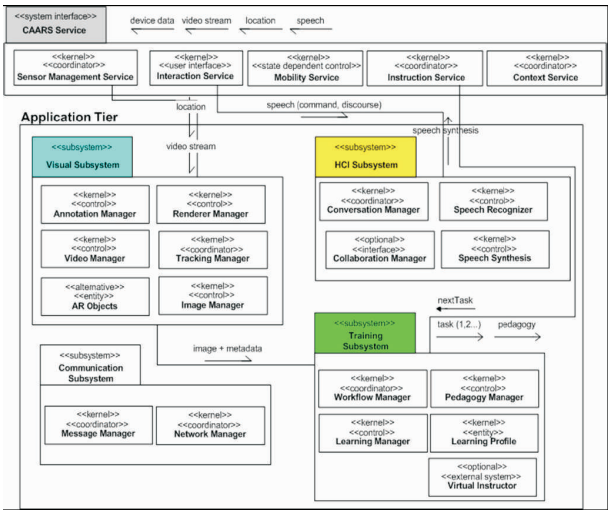


Figure 4: CAARS Software Service Architecture.

Figure 4 illustrates the CAARS high level software services architecture. The wearable AR training service is facilitated by a service oriented architecture made up of several subsystems that deliver visual, human computer interface (HCI), mobility, and training services. These services are high level software interfaces that encapsulate the lower level objects allowing for improved software algorithms to be continually implemented and incorporated into the system.

The Visual subsystem handles the image recognition and analysis to facilitate accurate real-world object recognition (e.g., recognize historical landmark). The HCI subsystem controls speech recognition and speech synthesis services that enable hands free and more natural interaction with the CAARS learning environment. The Training subsystem uses a combination of software agents and pedagogical models to guide learners understand concepts and perform step-by-step procedures for completing tasks. To facilitate this functionality, the training subsystem contains software agents that intelligently control the administration of training scenarios.

4. Auto-manufacturing Implementation

From the CAARS software services architecture, a prototype CAARS based system was developed for demonstrating the effectiveness of this wearable AR intervention for providing training assising to automanufacturing workers on auto production lines. In the auto-manufacturing plant, auto-workers already wear safety goggles. The CAARS goggles replace these safety goggles providing the same human ergonomic and safety features while, at the same time, delivering training and decision support services to the individual auto-worker.

Specifically, using the CAARS goggles, auto-motive line workers would be able to practice and perform various psychomotor tasks (e.g., assembly tasks) without the need to flip through books of schematics or instructions. As such, the CAARS goggles were designed to wirelessly communicate with the CAARS service prototyped from the CAARS software architecture to provide schematics and instruction as a response to verbal requests.

Figure 6: CAARS visual interface.

The CAARS Software Service was built using a combination of open source computer vision and speech services as well as custom components developed in Java and C++. Following standard software engineering methods, the CAARS software service is decoupled from the CAARS goggles. For the visual system subsystem, an

open source image recognition framework using OpenCV, was leveraged to build a custom object recognition and real-world object annotation service. The Human Computer Interface (HCI) subsystem including a speech interface was developed based on Java Speech API to create a speech recognition and synthesis service.

For the CAARS Training Subsystem, a custom software agent subsystem was developed. Figure 6 illustrates the software components that define training workflow and 'classes' that are controlled by the CAARS software service.

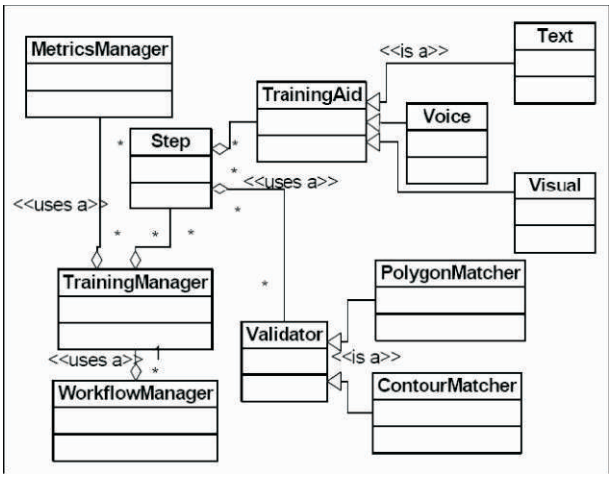


Figure 6: CAARS Training Subsystem.

5. Evaluation

Researchers conducted usability and human factors evaluation on the CAARS goggles and the visual/auditory interfaces for use in training and assistance. Research results suggest that the wearable AR intervention specifically facilitated by the CAARS Goggles and CAARS Software service reduced errors and time on psychomotor tasks. Additionally, research results suggest that the CAARS system accelerates the learning of psychomotor task, especially is the same CAARS Goggles are used for training and work. Furthermore, human factors evaluation of the CAARS goggle prototype indicated a comfortable human computer interface.

6. Conclusion and Future Work

Applications created from the CAARS research prototype have implication in domains including, but not limited to K-12 training, manufacturing training, medical procedures security, video games, workforce training. Continued investigation is planned to extend the CAARS system/software architecture to deliver additional services to improve real-time and just-in-time training and assistance in additional domains. Additionally, researchers plan to conduct longitudinal studies applying the CAARS architecture produced prototypes to provide various training and assistance to other domains (e.g., medicine) to determine the impact of the wearable AR intervention.

Furthermore, researchers plan to evaluate the CAARS software service on a variety of augmented reality head mounted displays.

7. References

[1] Aist, G., Kort, B.,Reilly, R.,Mostow, J., and Picard, R. "Experimentally Augmenting an Intelligent Tutoring System with Human-Supplied Capabilities: Adding Human-Provided Emotional Scaffolding to an Automated Reading Tutor that Listens". *4th IEEE International Conference on Multimodal Interfaces*, pp 483-490, October 2002.

[2] Blake, M.B. and Doswell, J.T. "Context-Aware Augmented Reality System (CAARS)". *SALT New Learning Technologies Conference*, Orlando, FL, Feb. 2006.

[3] Boulanger, P. "Application of Augmented Reality to Industrial Tele-Training". *Proc. of the First Canadian Conference on Computer and Robot Vision (CRV'04)*. Pp. 320-328. 2004.

[4] Doil, F., Schreiber, W., Alt, T., Patron, C.. "Augmented Reality for Manufacturing Planning". In *Proceedings of the Workshop on Virtual Environments. ACM Press*. Zurich, Switzerland. pp: 71 – 76. 2003.

[5] Hollerer, T., et. al.. "User Interface Management Techniques for Collaborative Mobile Augmented Reality". *Computers and Graphics*;25(5): pp799-810. 2001.

[6] Höllerer, T., Feiner, S. "Mobile Augmented Reality". In Karimi, H. and Hammad, A, editors, *Telegeoinformatics: Location based Computing and Services*. Taylor and Francis Books Ltd. London, UK, 2004.

[7] Hughes, C., Stapleton, C.B., et. al. "Mixed Reality in Education Entertainment, and Training. *IEEE Computer Graphics and Applications*. pp. 24-30. November/December 2005.

[8] Milgram, P., Kishino, A.F. "Taxonomy of Mixed RealityVisual Displays," IEICE Trans. *Information and Systems*, vol. E77-D, no. 12, 1994, pp. 1321-1329.

[9] Reiners, D., et. al. "Augmented Reality for Construction Tasks: Doorlock Assembly", *1st International Workshop on Augmented Reality (IWAR'98)*. 1998.

A Dynamic Multi-display System Approach

Gelson C. Reinaldo, Marilena Maule, Márcio Zacarias, Anderson Maciel, Carla M.D.S. Freitas, Luciana P. Nedel

Institute of Informatics – Federal University of Rio Grande do Sul (UFRGS)
Porto Alegre - RS - Brazil

{*gcreinaldo*}, {*mmaule*}, {*mrzacarias*}, {*amaciel*}, {*carla*}, {*nedel*}*@inf.ufrgs.br*

Abstract

Techniques for construction and configuration of tiled display systems have been focused by a number of research groups. Arrays of monitors or projectors in a fixed size matrix NxM can be managed by computer clusters to display a single image with large dimensions and high resolution. In the present work, an array of tablet PCs is used to compose a tiled display with a specific interaction feature due to the mobility of each individual tablet. Tracking ground fixed markers with the tablets web cameras enable the system to change the virtual image region to be displayed by each tile, which allows dynamic exploration of the visualization space.

1. Introduction

In the last few years, computers are quickly multiplying their capacity on data acquisition, storage and processing power. The amount of data to treat is increasing in the same pace. One problem in this context is how to view and interact with these huge amounts of data if there is only low resolution and small conventional displays available. Additionally, computers are really becoming part of our lives.

We are more and more in face of situations where computers are completely pervasive and the interaction space is mixed with the real space, anticipating the future of computing. Among the challenges imposed by such reality, interactive data visualization using tiled displays is getting more and more attention of researchers in several areas of computer science. The central idea of tiled display systems is to use a matrix of ordinary displays as one single high resolution display. The goal is to view a high resolution image through a mosaic of small ones. Figure 1 shows a tiled display composed of four 1280 x 800 pixels each, allowing an image display up to 2560 x 1600 pixels.

However, a greater impact of this technique is in using large-format tiled displays to build-up "information" or "activity" spaces that involve the user, making possible

Figure 1. Arrangements with four tablets Pcs, allowing to display an image of up to 2560 x 1600 pixels.

diverse ways of interacting with multimedia data and information flows. Such environments may prove to be the successors of the desktop metaphor in information technology tasks [1]. A significant amount of previous work explored the area of tiled displays to build large scale devices, to evaluate existent interaction and collaboration techniques, and also to propose new techniques and metaphors. However, the general approach is to use fixed displays with static layout (rectangular is usual).

In this paper we propose, for the first time, a tiled display array built from an arbitrary set of tablet PCs that allows the users to change the tile layout dynamically and in real time as shown in Figure 2. The system proposed minimizes possible space restrictions regarding visualization. It exploits mobility to allow an intuitive and distinct interaction strategy to explore the region to be viewed.

2. Related Work

Several groups have worked on building and configuring tiled displays. Hereld et al. [1] and Tao Ni et al. [3] describe a method to assemble a low cost tiled

Figure 2. Tiled display with an arbitrary arrangement allows viewing different portions of the picture.

display. They use an array of monitors (usually LCDs) or projectors arranged in a fixed tile matrix NxM. The displays are managed by a cluster of computers which can show a single image with large dimension and high resolution encompassing the whole array.

Other authors have investigated new interaction metaphors to allow collaboration on a dynamic, virtual large display. In one of these approaches, tiled displays can be formed by multiple tablet computers. When two single displays are moved close to each other, special sensors detect the orientation and direction of such "touch", and a larger integrated display area is dynamically formed [2]. In order to dynamically enlarge the interaction area for the purpose of shared use, a flexible coupling of displays overcomes the restrictions of display sizes and borders. Hinckley [2] reports the use of synchronous gestures that are distributed patterns of activity that take on a new meaning when they occur together in time, or in a specific sequence in time. For example, the synchronous gesture of joining two mobile devices together can connect the devices in various ways. This creates a collaborative face-to-face workspace with a shared whiteboard application.

ConnecTables [4] form connections between mobile devices. They are wheeled tables with mounted LCDs that can be arranged together so that the top edges of two LCDs meet, forming a collaborative workspace. Each LCD senses the presence of the other one using radiofrequency identification (RFID) tags. Of course, these systems require special hardware, and not simple ones like the tiled displays built based on tablet PCs.

In all methods mentioned above, a fixed array (and a single image) is used, and if a single display is moved that part of the view is lost. In other words, if one changes the layout of the displays arrangement, the image (supposed to be shown in the large display) and the view does not match.

With an extra view area and mobile tiles the total view is flexible to adapt the single tile repositioning. This flexibility of arrangement and display for sure leads to new applications and requires the exploration of new interaction methods and visualization.

3. Design

As stated by Hereld et al.[1] and Tao Ni et al. [3], a cluster of computers allows using each video output to compose a single high resolution image. By partitioning the large image in a way that each computer displays its respective portion, the total image resolution is increased. *Chromium*[1] is a relevant tool regarding this feature. Running Chromium in a cluster, using the client/server model, any OpenGL application executed in the server can have its output partitioned and distributed through the clients. The partitioning and distribution is synchronized and transparent to the application.

Since, in our work we want to build a dynamic display topology, the position and orientation of each display must be captured in real time and passed to Chromium[1] so the image can be segmented dynamically, updating the larger display conformingly.

We developed a system with (i) a cluster of laptops (tablet PCs); (ii) digital web cameras, integrated in each laptop; (iii) a position and orientation capture method based on ARToolKit[2]; (iv) markers for capturing tablets position and orientation; and (v) an example application to show the functionality we want to demonstrate. Figure 3 shows the marker positioned above the tablets, allowing its capture by the cameras.

A complete, real time synchronization through the network is beyond the scope of this work. A simple case where a static picture is shown has been implemented, and it needs no synchronization. In fact, the same program running in each a tablet can calculate the position of the respective computer, and display the corresponding, segmented image portion. It should be noticed that the actual picture size is larger than one single display, and many displays are needed for showing the entire view.

The dynamic tiled display was built in the following way. A planar region, with fixed dimension and position, represents the virtual large screen, where the image will be virtually drawn. The marker is at a fixed position, at a certain distance from the image plane. Each webcam is able to capture the marker image, and obtain the position and orientation of the corresponding tablet using the ARToolKit. Each tablet has a LCD, which is used as a tile of the larger display to show the view of the corresponding synthetic

[1] http://chromium.sourceforge.net
[2] ARToolKit is a software library for building Augmented Reality (AR)

Figure 3. The marker positioned above the tablets, allowing capture by the web cameras.

camera updated based on the marker information. This way, moving a tablet causes the corresponding image in the LCD to be redrawn as we were moving a window over the virtual large picture. Thus, each tile shows different pictures depending on their positions and orientations. Since the tablets have a LCD screen surrounded by a frame, the tiled display formed by the LCDs will show the image with some parts missing, the ones occluded by the LCD frames. However, there is no distortion, just a "natural" discontinuity in the image. An arrangement with a 2x2 tile array increases twice the dimension of the image that can be shown in relation to a single display; the resolution (pixels per inch) remains the same but there are four times visible pixels. Figure 1 shows such an arrangement with four tablets (T_{11}, T_{12}, T_{21}, T_{22}) with 1280x800 pixels each. Tablet T_{11} displays the top left image portion; T_{12} displays the top right; T_{21}, the bottom left, and T_{22} displays the bottom right image part. The size of the virtual image is determined by software, but its visualization depends on the hardware. If we add the dimensions of each LCD screen, we end up with a different value from the actual virtual image size.

4. Implementation

An example application was developed using the ARToolkit and OpenGL APIs. ARToolkit video tracking library is used to handle the signal from the tablets builtin cameras. It allows real time calculation of the camera position and orientation relative to a ground fixed physical marker. This geometric data is actually a transformation matrix that is passed to OpenGL. OpenGL then uses this information to handle a virtual camera placed at a specific position and oriented in such a way that they match the situation of the actual marker relative to the virtual scene.

The same software application runs asynchronously on every tablet. It is a simple application displaying a static plane where a high resolution image is textured. Each instance of the application estimates the virtual camera position and orientation and displays the part of the plane (and the image) corresponding to its own position and orientation.

Actually, as the physical marker represents a fixed coordinate system for the application, each tablet will display a different part of the image depending on its own position in the real world. In practice, if a few tablets are placed on a table surface and the marker on the ceiling, the system works like if a large poster was placed under the table and each tablet was a transparent window on the table surface allowing to see part of the image. If a sufficient number of tablets is placed side by side completely covering the table, the whole image will be displayed as a mosaic. If a tablet is moved, its virtual camera is updated in real time and different portions of the image are intuitively displayed:

• marker distance affects zooming. Moving the tablet up causes a zoom in, and moving it down reflects a zoom out;

• lateral movement (on plane) shows another portion of the picture respective to the tablet movement direction;

• angular movement (on plane – pitch) rotate the canvas in respect to the tablet rotation direction;

• angular movement (row and yaw) skew the image in respect to the tablet rotation;

• tablet overlap causes the same portion of the picture to be drawn on the overlapped areas of both tablets.

Note that the system compensates any move of the tablets so that the virtual image seems stationary.

Figure 4. The picture remains continuous even with the displays overlapping.

Asynchronous decentralized systems like the example described above work without communication and with virtually any number of tiles (displays). On the other hand, a synchronized distributed system is managed by a server which centralizes the communication. In conventional tiled displays it hosts the virtual image and shares portions of the image with the client tiles for display. Chromium is a free implementation with distributed characteristics for client/server communication that can be applied in the context of tiled displays. This rendering system has the advantage of being non-invasive. It means that every OpenGL application, including dynamic 3D scenes, could be used. The present paper is an ongoing work eventually aiming at a full distributed application of dynamic tiled displays. The current Chromium version is able to split a dynamic scene into a set of tiled displays, but does not allow change of display position and orientation on the fly. As a first approach we plan to use the basic functionality of Chromium and extend it to perform dynamic relocation of the partial displays. This will allow for applications to display videos or real time 3D scenes on a dynamic, interactive and collaborative tiled display.

5. Results

The above description was implemented and we conducted some experiments. Results are clearly preliminary because this is a work in progress as mentioned in the previous section.

We experimented the application varying the tablets positions, thus changing the tiled display configuration (see figures 1, 2 and 4). We had some experimental users testing the system. Even without training, they performed intuitive movements to explore the virtual image displayed "under" the tablets.

Important issues are environment illumination and marker positioning. Changes in these parameters required additional camera calibration. Good illumination and fixed marker position reduce the occurrence of problems in marker tracking.

6. Conclusion

In this paper we presented preliminary tests in the construction and evaluation of a tiled display system allowing dynamic rearrangement of its tiles. The results obtained so far allowed us to visualize a static scene, i.e., the objects in the scene do not move and we can run the displays asynchronously.

The next step is to adapt a distributed rendering system (e.g. Chromium) compatible with dynamic virtual images (animations). Furthermore, alternative interaction techniques could be used to allow for and to improve distributed applications.

Interestingly, this work has shown that mobility can establish new interaction metaphors that have revealed new application possibilities. Moreover, this dynamic view resizing flexibility opens a new branch in the area of collaboration. Tasks can be performed on a shared workspace by collaborative users that otherwise could not be accomplished if the same users work individually isolated.

7. Acknowledgments

We greatly acknowledge the support from Microsoft and from HP in this project. Thanks are also due to CNPq for the support to the researchers A. Maciel, L. Nedel e C. Freitas.

8. References

[1] M. Hereld, I. R. Judson, and R. L. Stevens. Tutorial: Introduction to building projection-based tiled display systems. *IEEE Computer Graphics and Applications*, 20(4):22–28, 2000.

[2] K. Hinckley. Synchronous gestures for multiple persons and computers. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 149–158, New York, NY, USA, 2003. ACM.

[3] T. Ni, G. S. Schmidt, O. G. Staadt, R. Ball, and R. May. A survey of large high-resolution display technologies, techniques, and applications. In *VR '06: Proceedings of the IEEE conference on Virtual Reality*, page 31, Washington, DC, USA, 2006. IEEE Computer Society.

[4] P. Tandler, T. Prante, C. Muller-Tomfelde, N. Streitz, and R. Steinmetz. Connectables: dynamic coupling of displays for the flexible creation of shared workspaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology,* pages 11–20, 2001.

Using a General Purpose Virtual Environment for Artificial Life Simulations

Cesar Gomes Miguel, Marcio Lobo Netto
Department of Electronic Engineering
Cognitive Sciences Research Group
University of São Paulo
{fcesargm}, {lobonettg}@lsi.usp.br

Abstract

This short paper describes a general purpose opensource 3D virtual environment called BREVE. We also briefly describe a recent method (NEAT) for the evolution of arbitrary topology artificial neural networks. This method is used in conjunction with BREVE in a simple Artificial Life simulation where a population of virtual organisms controlled by neural networks evolves to accomplish a given task. The role of a good and freely available virtual environment package for such experiments is emphasized using a simple philosophy design: there is no need to reinvent the wheel.

1. Introduction

Artificial Life (ALife) is a multidisciplinary field concerned with building biological models that synthesizes some basic aspects of life [1]. Those models are basically defined by the interaction of many independent sub parts, often constituted of very simple rules that, when observed globally, shows properties of emergent behavior. For instance, a single ant has a very limited number of behaviors but a population of ants, an anthill, acts like a single living organism.

The approach of ALife makes a heavy use of computer simulations and often demands the use of a tool for visualizing the resulting emergent behavior. Moreover, the simulated virtual environment needs to reproduce to some degree of precision the main properties of the real world, like the basic laws of physics and optics. In short, besides modeling each agent's behavior with local rules (how to behave under certain conditions, like the ants), we need a virtual world to place the interacting agents and observe the results, or as in our previous example, the anthill. Most of these models work as an evolvable multi-agent system, in which the agent is an artificial organism living in a virtual environment.

In this work, the local rules of behavior is evolved using NEAT [6] and the simulated virtual environment will be handled by BREVE [2].

1.1 A View Of The Problem

In several works related to ALife the virtual environment is developed and maintained by the authors or by a small group of users, often restricting its usage for the community of researchers interested in the same subject. In addition, it is worth noting that developing a system for the task requires a great amount of work and knowledge in diverse fields such as computer graphics and advanced programming. In cases where the code is released to the public we are often faced with the loss of good and properly documented work.

This short paper is divided in three small sections. First we describe a neuroevolution method, known as NEAT [6]. This method applies a modified genetic algorithm to evolve arbitrary recurrent neural networks using a special type of chromosome encoding. Secondly, we give a short description of BREVE, an advanced open-source software package for general purpose 3D visualization and simulation of multi-agent systems [2]. For the last section, a NEAT Python implementation is used along with BREVE showing how an artificial life simulation where a population of simple neural network-controlled organisms evolve to solve a given problem.

1.2 Related Work

As stated in section 1.1, there have been several artificial life experiments where the virtual environment was specifically designed for the problem. Probably the most well known work is of Karls Sims [5], a virtual world in which artificial creatures evolve to solve a certain type of task, like grab a cube or swimming.

A more recent work investigated the use of a genetic algorithm to evolve the behavior of a virtual robot, controlled by a finite state machine [3]. The environment consisted of an arena enclosed by walls and a robot had to evolve to classify some randomly positioned objects.

Another attempt was a framework for ALife experiments called ALIVE [4], developed in Java 3D and aimed to be used as a virtual laboratory in which different types of virtual environment scenarios could be simulated.

2. Evolving Artificial Neural Networks

Most simulations embracing the use of neural networks as controllers are restricted to fixed topologies and the only characteristic that distinguishes them in a population is their synaptic weights. This is usually represented by a vector of float numbers and can be easily used as a chromosome encoding that single organism. However, the topology restriction implies that the number of possible behaviors is limited to a subset that may not contain the desired one. It is also unrealistic from the biological point of view since the nervous systems in nature has evolved through millions of years, causing a higher adaptation of certain species and eventually reaching human intelligence. A recent model known as NEAT [6] (*NeuroEvolution of Augmented Topologies*) was able to surpass this limitation using a sophisticated genetic algorithm that works with chromosomes of variable length and allows us to evolve real complex neural networks with arbitrary topology.

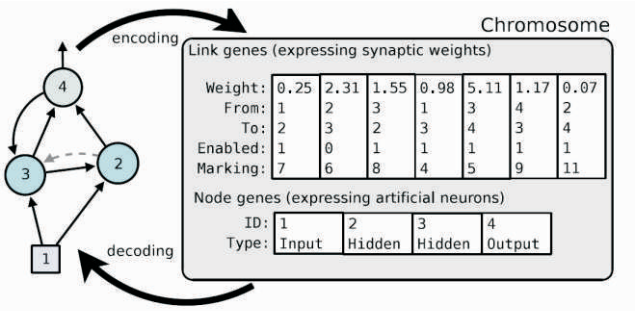


Figure 1. An example of neural network encoding in NEAT.

Basically, the model uses a special and general neural network encoding technique (Figure 1) in which it is possible to apply mutation and crossover genetic operators without destroying the parent's offspring (*a problem known as competing conventions* [6]). The process of adding new genes (that can be a link or node gene) through mutations (and thus augmenting the neural network) is called complexification (Figure 2). It is very likely that with the addition of new structures the offspring will have an inferior fitness value relative to its parents and would probably be taken out the population by selection. A speciation method is applied to ensure that new innovations can be retained in a separated species and given enough time to adapt before competing with others for resources. If after a specified number of generations the species does not improve its average fitness, all its members are replaced by the new offspring.

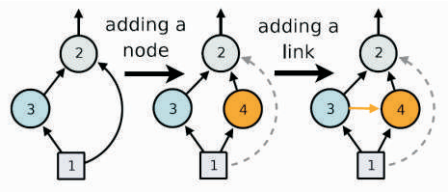


Figure 2. Mutation leads to complexification.

It should also be noted that since a neural network is a particular kind of graph structure (where neurons are the nodes and synapses are the edges), anything ranging from a Bayesian Network to a Finite State Machine can be evolved with NEAT.

Although NEAT has several open-source implementations in different programming languages available on the internet, in this work we use a recent Python implementation that simplifies the work for writing complex simulations[1].

3. The Breve Environment

Specially designed for multi-agent systems simulations, BREVE2 is a rich and open-source 3D virtual environment in which you can write advanced artificial life simulations with little effort [2]. BREVE is written in C++ and includes an OpenGL display engine for real-time visualization (Figure 3). It has full support for writing experiments in the Python programming language, making it easy to script simple control behavior or more advanced methods that requires the usage of OOP paradigm. Among its features we highlight: (1) support for collision detection with arbitrary shapes; (2) allows to build articulated bodies; (3) advanced physical simulation.

There are two basic objects in BREVE, Stationary and Mobile. The former represents immobile objects, such as a floor or obstacles while the latter represents agents which move and behave according to its own controller. At each iteration step the Control class calls the iterate method that is responsible for updating the state of Mobile objects.

Figure 3. Sample Breve screenshots.

[1]NEAT Python: code.google.com/p/neat-python

Optionally you can run BREVE in background mode without a graphical interface, maximizing CPU usage and allowing to control simulations over a Linux network.

4. Simulation

Our interest in ALife is the simulation of open-ended evolution, that is, one with no restrictions regarding the topology and size of the neural network. In a typical experiment setting the individual (agent) perceives and acts in the environment through its sensors and actuators respectively. A simple modeled agent and its perception attributes is depicted at the left of Figure 4. At the right is a typical scenario where the individual has to collect the closest food it is sensing. The virtual environment is created and managed by the BREVE Control class. At each time step the neural network receives as input the value its sensors is reading and the action is taken after activating all the neurons. This process runs in loop for a fixed number of steps and evaluates the agent's fitness based on some criteria. In this simple example there is only one input: the angle between the agent's heading to the closest food. There are two outputs that controls how fast it moves forward and turns to the right or to the left. As is the case with genetic algorithms, the selection method varies according to the experiment where one wishes to study, e.g., gathering food, predator-prey chase, wandering behavior, and so on.

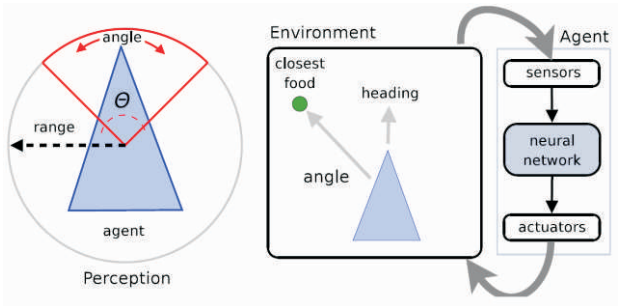


Figure 4. At left: an agent and its perception attributes. Right: how the agent perceives and acts within BREVE.

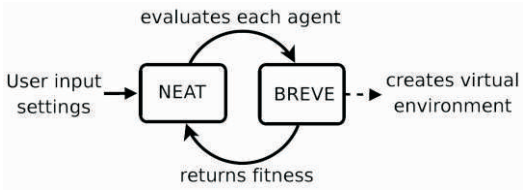


Figure 5. Interaction between NEAT and BREVE.

Apart from the virtual simulation, NEAT manages all the interaction between the genetic algorithm and BREVE as shown in Figure 5. As the input NEAT receives the user's settings: mutation rates, population size, number of epochs, and etc. The first population is created at random and then each individual is passed to BREVE through systems calls.

As soon as the virtual environment is created the individual is allowed to live for a fixed number of time-steps (Figure 4) and its fitness value is calculated based on some defined criteria. When the simulation time ceases, BREVE returns the fitness for that particular individual. The process can also be easily extended to evaluate the whole population at once depending on the experiment's requirement.

5. Future Work and Final Considerations

It was shown how to integrate a powerful neuroevolution method with a general 3D environment simulator, opening several possibilities regarding its use in Artificial Life. The next step will help us to investigate the role of adaptive behavior and learning in respect to the interactions with the environment.

From the technical point of view it would be of great value to have a parallel and distributed version of NEAT so that the evaluation of the population could be carried out in a cluster. In addition, the current version of Breve allows only one camera view at a time, but this restriction will be solved in a future release, increasing the visualizing immersion through the use of several displays. So far BREVE has shown to be very stable in long-time running simulations, becoming well suited for general purpose experiments in Artificial Life and Intelligence, eliminating the need to write a full featured simulator from scratch.

6. References

[1] C. Adami. Introduction To Artificial Life. Springer-Verlag, New York, 1998.

[2] J. Klein. Breve: A 3D simulation environment for the simulation of decentralized systems and artificial life. In *Proceedings of Artificial Life VIII, the 8th International Conference on the Simulation and Synthesis of Living Systems*, 2002.

[3] F. R. Miranda, J. Kogler, M. L. Netto, and E. Del Moral Hernandez. Arena and woxbot: first steps towards virtual world simulations. *XIV Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'01)*, pages 208–215, 2001.

[4] R. P. O. Neves and M. L. Netto. A virtual reality framework for artificial life simulations. *Symposium on Virtual Reality*, 7: 217–227, 2004.

[5] K. Sims. Evolving virtual creatures. In *SIGGRAPH '94 Proceedings*, pages 15–22, July 1994.

[6] K. Stanley and R. Miikkulaien. Evolving neural networks through augmenting topologies. *Evolutionary Computation,* 10: 99–127, 2002.

Session 4: Applications

A Two-User Virtual Environment for Rapid Assembly of Product Models within an Integrated Process Chain

Maxim Foursa, Gerold Wesche, David d'Angelo, Manfred Bogen
Fraunhofer IAIS - Virtual Environments
Sankt Augustin, Germany
{maxim.foursa}, {gerold.wesche}, {david.d-angelo}, {manfred.bogen}@iais.fraunhofer.de

Rainer Herpers
Fachhochschule Bonn Rhein-Sieg
Sankt Augustin, Germany
rainer.herpers@fh-bonn-rhein-sieg.de

Abstract

In this paper we present an ongoing research work dedicated to a Virtual-Reality-based product customization application development. The work is addressing the problem of flexible and quick customization of products from a great number of parts. Our application is an effective instrument that can be simultaneously used by two users for rapid assembly tasks, allowing engineers and designers to work collaboratively. Furthermore, it is directly connected to a manufacturing environment, which is able to produce the product right after customization. In the paper we describe the architecture of the application, our interaction and assembly techniques, and explain how the system can be integrated into a manufacturing environment.

1. Introduction

In this article we describe a Virtual Reality (VR) application that allows to flexibly customize a product based on assembly modeling and schedule immediately the result for production. Current functionality allows two users to work simultaneously in a projection-based display system, called the TwoView display [8].

The 3D user interface for assembly-based modeling needs well designed mappings of user input actions onto the functions provided by computer-aided design systems. An input action is a human activity to enter a command to a computer system. In most cases, the hand is used for such actions, e.g. moving the mouse, hitting a key, or pressing a button. To each of these actions a function is assigned by user interface modules of the software.

On the desktop, the possibilities to design those mappings are very restricted. This is because standard input and output devices together with the WIMP interface (window, icon, menu, pointing) provide well-established standard mappings from hand actions to functionality. In other words, the way a human user applies his hands is predetermined. A consequence of using this standard is that all naturally hand-oriented activities do not map well into the WIMP interface and appear very unnatural to a user, once they are computerized.

Using Virtual Environments (VE), a much larger set of possible mappings from hand inputs to computer functions is available. This includes direct interaction with virtual models, gesture-based interaction and application control. In this paper we describe how we make use of these capabilities for assembly modeling. The capabilities allow developing an effective product customization tool. Its main features are as follows:

• No changes to shape equations are required since the modeling primitives are predefined parts.

• Assembling models from primitives, which is the main task, maps easily and naturally to direct, hand-based interaction. E.g. the user is supported by a context sensitive snapping mechanism.

• Simple transformations and changes of shape attributes of parts, such as radius, height, etc. intuitively map to 3D interaction, as well.

• The information how the parts are connected can be stored independently. This allows the VE application to be integrated into product development processes, since the geometry data will not be modified.

• Encoding possible connections of parts naturally constrains movement of parts in a 3D environment, using snapping techniques. This supports direct and rapid part positioning by hand.

Moreover, we present an XML-based language for the specification of all possible configurations based on a set of predefined parts. Using the VE modeler, a user effectively adds connection information by constructing a virtual product out of these parts. This is stored and, together with the part description it forms a complete product configuration for use in successive steps of the process chain. In section 2 we discuss related work. In

section 3 we describe our approach to product assembly, the Virtual Environment application, including 3D assembly tools and user interface concepts. Section 4 describes the components used for import and export of parts and models, and the process chain in which the modeler is used. Section 5 summarizes our work.

2. Related work

Assembly-based modeling using Virtual Environments can be categorized into two main groups. Most work is centered around Assembly Simulation, which means simulating the process of assembly or manufacturing. This involves packaging issues as well as determining required space regions, by calculating assembly paths for parts, generated by the part itself together with the grabbing robot arm or human hand. The goal is to find a way through space for the part that avoids any collisions and results in a costeffective process. Virtual environments can support these tasks because they provide spatial and direct interaction together with perspective stereo viewing. This is confirmed by Zhang et al. [12], who found that assembly task performance can be improved by visual feedback mechanisms. They used a Responsive Workbench [13] for their experiments.

The VADE system [10] is a fairly complex environment that couples such kind of functionality to a CAD system. It supports one- and two-handed assembly tools and runs within a head-mounted-display (HMD) setup. Zachmann and Rettig [11] present multimodal and spatial interaction for virtual assembly simulation and consider criteria like naturalness, robustness and precision for the development of grasping techniques and gestural input. They also use an HMD.

The other group of contributions, more related to our work, focuses on immersive modeling applications that support the assembly of complex models out of predefined parts. Jung [4] represents the connection sensitivity of part regions using so-called ports. From the rich connection semantics of those the constraints for part connections are derived.

This includes analyzing the geometric relationships among the involved parts and allows to find a solution even for complex port configurations on multiple parts. Compared to Jung, we use a similar, but more intuitive solution, allowing a lightweight implementation. Our algorithms for part connections rely only on geometric properties of socalled handles, which define permitted alignments of connected parts using restrictions of the available degrees of freedom (DOF), see section 3.1.

We explicitly do not make use of the geometry of the parts, opposed to e.g. the VLEGO modeler of Kiyokawa et al. [5], which uses the same sort of toy blocks as in our work. Arising from the properties of those blocks and their studs, VLEGO restricts possible connections of blocks to discrete positions and orientations. Although we use the same sort of blocks in our examples, possible part connections are defined by the properties of additional handles, which are independent of the geometry.

Collaborative object manipulation for two users wearing an HMD has been investigated by Pinho et al. [7]. They present two kinds of collaborative manipulation tools; one is based on the separation of degrees of freedom for two users, whereas the other relies on the composition of user actions. They have observed increased performance and usability in difficult manipulation tasks. Therefore, we expect a similar benefit from our two-user approach. We have already developed basic two-user functionality and we are planning to perform usability tests in our future work. The main difference to our work is that in the work of Pinho et al. the two users apply the cooperative tools simultaneously to one object, whereas the two users in our scenario individually control a part while performing a common assembly task.

Similarly, two users model collaboratively using the SeamlessDesign system of Kiyokawa et al. [6]. They are equipped with See-Through HMDs. Geometric primitives can be combined with constraining primitives interactively. In this way, the constraints are defined in an interactive process, whereas in our work, the handles must be completely defined beforehand. In order to support this process in a 3D environment, these constraining primitives might be useful. Compared to [6] and [7], we use a projection-based virtual environment in which two users perform co-located collaboration. This so-called TwoView display system consists of two active-stereo-capable DLP projectors behind a vertical 3 2:4m screen. Two simultaneously displayed stereo image pairs are separated by polarization filters in front of the projector lenses. The shutter glasses have corresponding filters, so that each user sees his own active stereo projection. Details are explained in the work of Riege et al. [8]. They also introduce collaborative selection and dragging techniques for two users manipulating one object, based on a bent pick ray. The first projection-based two-user display system was proposed by Agrawala et al. [2]. They achieve eparation of the views for the two users by modifying the shutter glasses so that they close both eyes of one user

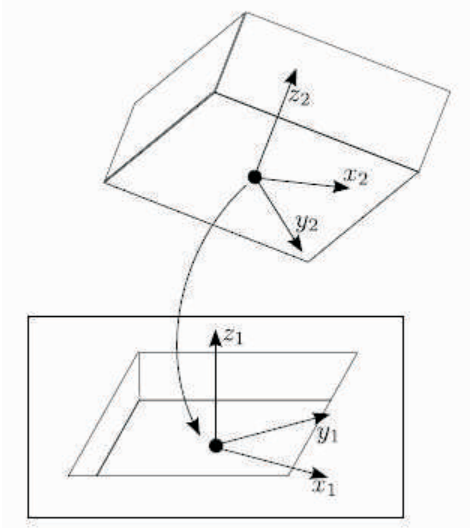


Figure 1. An ExactMatch handle. There are no DOFs for connecting the two parts. However, the geometry would allow three other possibilities. This could be modeled by using three more ExactMatch handles.

whenever a view for the other user is displayed.

3. Interactive 3D Assembly

Our approach to assembly based-modeling avoids the direct transfer of the WIMP interface to the 3D environment. Instead, we have developed 3D interaction widgets, presented in section 3.4. Collaborative two user interaction on the basis of different user profiles is described in section 3.3. We exploit 3D for free hand movement, mapping assembly and disassembly tasks to direct interaction. This is explained in section 3.5. On the other hand, we employ constrained-based interaction, restricting free handperformed placement of parts, so that the resulting model always corresponds to the set of supported configurations, see section 3.1 and 3.2.

3.1. Handles

A handle is an attribute of a part used to specify the geometric arrangement of connections to other parts. The handles employ a discrete set of possibilities for part placement. For describing the geometry of connections, handles use geometric primitives, such as points, vectors, or coordinate systems together with a corresponding connection semantic. Different types of handles use more or less of these geometric connection constraints. All handles must be defined in the local coordinate system of the part.

Handles have two main purposes: first, handles define the set of models possible to construct out of the set of given parts. Optionally, the definition of possible models can be complemented with additional restrictions, such as relations of parts that may connect. For these concepts, see section 4.1.

Secondly, a handle supports a user by guiding him assembling two parts together. In this sense a handle just implements constraint-based assembly. When moving parts together, matching handles are highlighted and the parts snap together in order to give a user feedback about a possible connection. A user has to confirm, and then that connection is stored. How the flow of action occurs for one or two collaborating users is explained in the next section.

Two parts can be connected at a common location if there is at least one pair of matching handles from each part. A handle matches another handle if the two handle types are identical, the handles tags match each other (e.g. there are male and female tags), and the geometric primitives of the handle coincide. In case there are several possibilities to connect two parts, for each configuration there has to be at least one corresponding pair of handles. Therefore, several handles at one particular part always have an 'OR' relationship.

Things get more complicated if a connection of two parts involves several handles on each part. The DOFs for a connection are restricted as soon as a second handle pair must be considered. Furthermore, the connection algorithm must search all possible configurations, depending on the moving parts (see below).

Different assembly configurations require appropriate handle types. Currently, we support the following handle types.

ExactMatch - an ExactMatch handle (see Fig. 1) is a right-handed orthonormal coordinate system. A user defines it by providing a point and two linearly independent vectors. As the name indicates, two handles of this type match if they exactly coincide.

AxisMatch - the AxisMatch handle (see Fig. 2) consists of a point and a vector. Two of these handles match if their points and their vectors coincide. Therefore, one rotational DOF is left. Notice that this DOF does not mean that the two connected parts may rotate against each other after assembly. It simply leaves one angle of the arrangement unspecified.

SurfaceMatch - similar to the AxisMatch handle, a SurfaceMatch handle (see Fig. 3) has a point and a vector, forming a surface equation. Therefore, the point and vector have different semantics, compared to the AxisMatch: two positional and one rotational DOF remain when two parts are connected via a

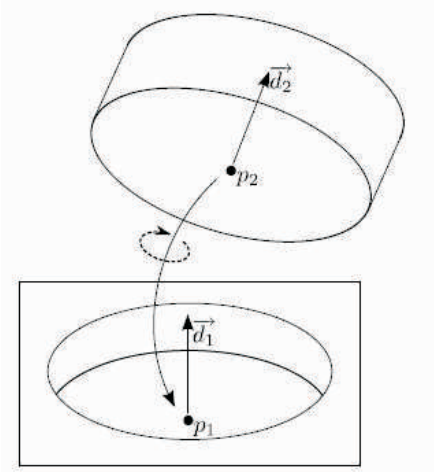


Figure 2. An AxisMatch handle. At which angle the cylindric part is connected is not specified.

SurfaceMatch, i.e. the parts can be moved and rotated against each other.

Note that the DOFs of a handle do not have the semantic of a "joint", i.e. moving parts are currently not supported. This functionality could be added in a future version of our modeler.

3.2. Connecting parts

Assembling two parts requires at least one pair of matching handles. However in most cases the matching process is ambiguous. Imagine surfaces of parts having several locations to which other parts of compatible type could be connected, as shown in Fig. 5. Here, the small piece can be connected in a variety of ways to the large plate, because each stud of the plate and each fitting hole of the piece are instrumented with AxisMatch handles. We implemented a simple user-guided connection algorithm that finds all matching handle pairs and, after the connection has been fixed, marks the involved handles as occupied in order to maintain a consistent state.

As a user is moving a part P through the workspace in order to connect it, we find a counterpart C by checking the bounding boxes. Then, the method proceeds as described by the algorithm in Fig. 4. It uses a snapping distance s between two handles, and connects parts by aligning the handles according to their geometric properties. This algorithm works for situations as shown in Fig. 5: after the first handle pair is connected, the user still has one degree of freedom to move the part. However, this connection method has some shortcomings, most notably its runtime increases with the number of handle pairs. Moreover, it may fix a connection that is geometrically impossible, since there is no collision checking. The advantage is

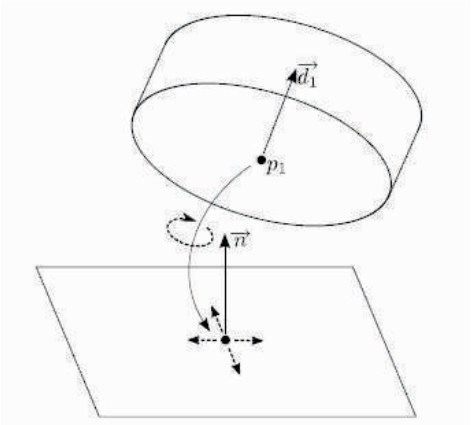


Figure 3. A SurfaceMatch handle. Just the two surfaces are connected. The relative location and orientation of the two parts are not specified.

that it is a lightweight solution completely independent of the geometry. It can thus be applied to any handle-instrumented part sets, and is not restricted to the toy blocks shown in the example.

3.3. Two-user interaction

There are two modes of collaborative interaction for two users. The first scenario supports a customer/product designer relationship and assigns each user corresponding modeling tools. E.g. a customer is assumed to have no knowledge about assembly and 3D interaction, however he has a detailed understanding of how the end product should look like. Therefore the customer cannot create or break up any connections; he can only perform simple state changes like selecting color or aligning the whole scenery for visual inspection.

The other mode equips both users with the full assembly functionality. Locking prevents a simultaneous transformation of the same geometry. However, collaborative assembling and disassembling is supported in a straightforward way. Both users simultaneously move parts until they are connected, as shown in Fig. 6. Another possibility is that a user can select individual parts and hand them over to the other modeler, who connects them to the product prototype. The hand-over process is initiated by positioning the part towards the hand of the other user and pressing a button to issue the transfer of the control over the part. This mode of handling interaction with parts is very intuitive and fits well to a collaborative session. In case the part selection widget is not easily accessible, one user could ask the other user to fetch the desired part for him. This mode is also suitable for the discussion of various modeling possibilities, where one user needs to get control over a part selected by the

Calculate initial handle pair (h_p, h_c)
if (distance $(h_p, h_c) < s$)
  snap P to C so that h_p and h_c are aligned;
  preliminary connect h_p and h_c;
  while (P is attached to hand)
    move P according to its DOFs
  end
  while (secondary handle pairs (h_{p2}, h_{c2}) left $\wedge$
    there are DOFs to move P onto C)
    snap P to C so that h_{p2} and h_{c2} are aligned;
    preliminary connect h_{p2} and h_{p2};
  end
  if (no incompatible handle pairs left)
    fix all preliminary handle connections;
  else
    release all preliminary handle connections;
  end
end

Figure 4. Algorithm for connecting two parts.

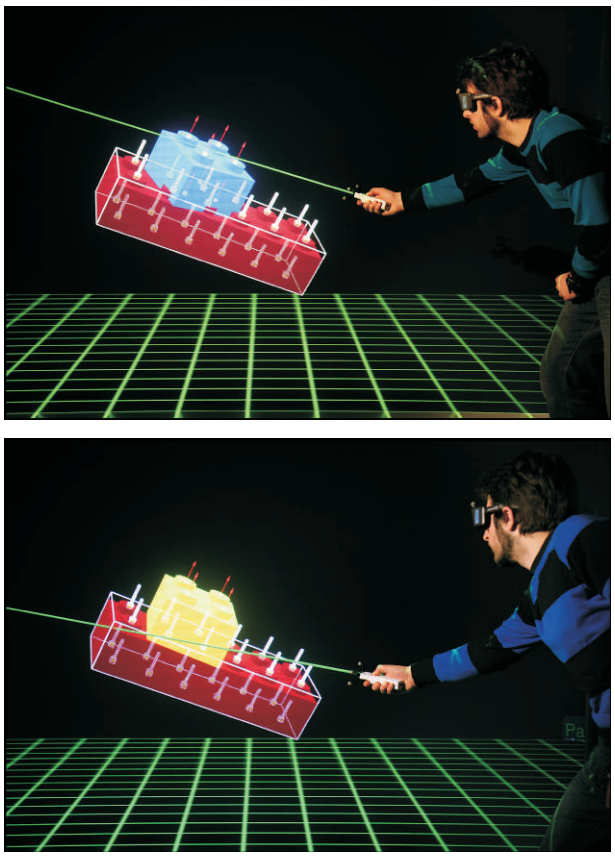


Figure 5. Connecting two parts. The first handle pair matched (top); all handle pairs are connected (bottom).

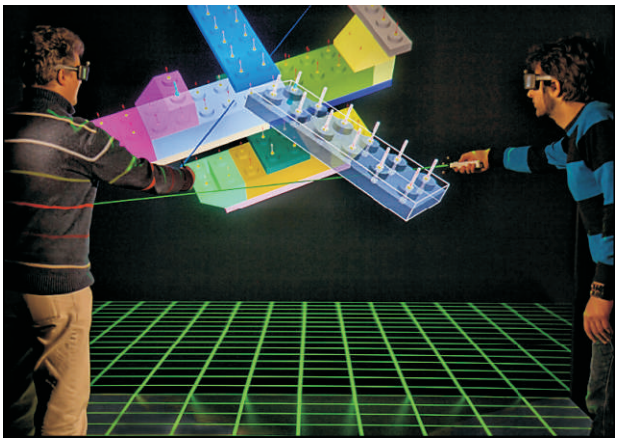


Figure 6. Collaborative assembly environment.

other user in order to demonstrate his ideas. Note that, compared to standard raypick selection of parts, in hand-over mode a user who wants control over the part does not need to perform a ray intersection. Therefore, hand over is a very easy collaborative selection method.

The collaborative modes are defined in user profiles, which assign tools to each user and are loaded upon start of the application.

3.4. User interface

The user interface is fully integrated into the 3D modeling environment, thus there is no distraction of the immersion while using the menu. It contains both adapted 2D widgets, like sliders and scrollbars, a set of 3D widgets and is visible for both users. A semitransparent context menu (see Fig. 7) is used for accessing the functionality of the actual context.

The context menu is displayed with respect to the position and the orientation of a user as well as the objects in the scene. Therefore its accessibility is ensured at all times. 3D widgets can be seen as a combination of geometry and behaviour. This means that system control functionality is moved from a menu directly onto objects, allowing the exploitation of the full degrees of freedom of the 3D workspace. A 3D widget used by our system is the disassemble widget. In most cases selecting an already assembled part means the starting action of an assembly operation, however it is also possible that a user wants to unhinge the part from its actual part group. Instead of selecting the disassemble command from a menu and the successive selection of a part, a user can select the disassemble widget located on top of a part and grab the part immediately afterwards. In this way selection and execution of the command form one fluent action sequence.

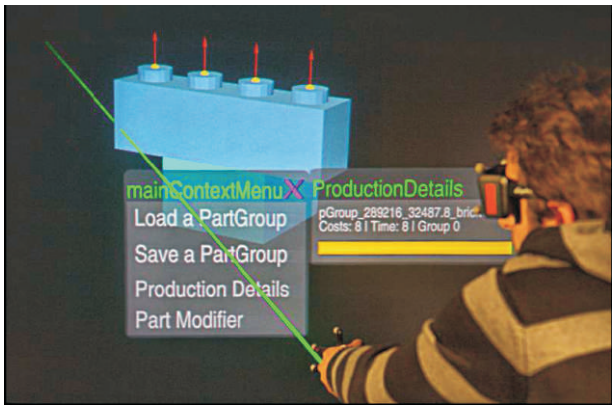


Figure 7. Context menu.

3.5. Assembly tools

The main interaction task in our system is grabbing and moving rigid bodies for assembling or disassembling. This can be performed intuitively in a VE. Each part has a 3D interaction widget assigned which appears depending on the context. The basic steps of the assembly tool are:

1. *Select* a part using a pick ray.

2. *Position* the part using a scaled grab technique, which makes it easier to move parts within a large-scale workspace.

3. *Snap* the part to the desired counterpart.

4. *Transform* the part according to the resulting degrees of freedom determined by the handle connection. The fitting mechanism (described in section 3.2) is used to ensure a valid position after completing the transformation.

The disassembly tool works the following way:

1. *Select* a part supposed to be released. Since the part is connected the system has to decide whether a user wants to move the whole part group or wants to unhinge the part. Therefore, a 3D disassemble widget appears close to the part.

2. *Grab* the widget.

3. *Position* the part to the new position. This can lead to another connection.

Note that the selection step is not necessary in case the part has been handed over in collaborative mode, as explained in the previous section.

4. Integration in a manufacturing environment

The manufacturing environment is controlled by supervisory services which coordinate industrial robots

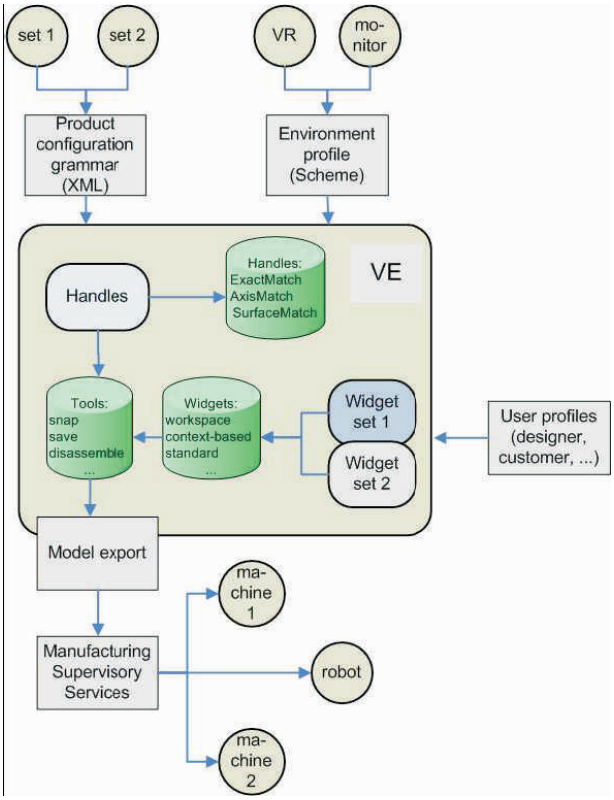


Figure 8. System architecture.

and manufacturing machines. The integration of the immersive assembly framework into the production environment is presented in 8. User interface, handle-based interaction technique and main functionality of the application have been already discussed. Before applying it in a manufacturing plant two profiles describing the working environment and parts to be used should be prepared. The environment profile contains the description of display system and interaction devices, while the part profile describes available parts with links to their geometry and contains connectivity information.

Below we explain how the parts are described, how import and export of models works and how the resulting model is processed after customization.

4.1. Part description

An XML-based language is used to describe all possible connections of parts in a *product configuration grammar*. Potential connections are not described explicitly in this language, since this would result in an immense set of possible part connections even for a relatively small set of parts. Thus possible connections are generated at runtime, out of the grammar, if needed. In Fig. 9 a simple grammar is shown describing possible

```
<ProductConfiguration>
  <PartCollection>
    <Part fileName="cylinder.iv" id="p1" />
    <Part fileName="cubeHole.iv" id="p2" />
  </PartCollection>
  <HandleCollection>
    <AxisMatchHandle id="cylinder_h1" partId="p1"
      x="0.4" y="0.4" z="0.0"
      dX="0.0" dY="0.0" dZ="1.0" />
    <AxisMatchHandle id="cubeHole_h1" partId="p2"
      x="0.4" y="0.4" z="-0.2"
      dX="0.0" dY="0.0" dZ="1.0" />
  </HandleCollection>
  <HandleTagCollection id="topHandles">
    <Handle id="cylinder_h1" />
  </HandleTagCollection>
  <HandleTagCollection id="bottomHandles">
    <Handle id="cube_h1" />
  </HandleTagCollection>
  <CompatibleHandleTagCollection id="CylConnection">
    <CompatibleHandleTag id="topHandles" />
    <CompatibleHandleTag id="bottomHandles" />
  </CompatibleHandleTagCollection>
</ProductConfiguration>
```

Figure 9. Simple Product Configuration Grammar for the configuration shown in Fig. 2.

connection of the two parts in Fig. 2. The grammar provides two methods for describing matching handle connections. The simplest possibility is the explicit specification of a pairwise matching relationship between two handles. In most cases this method is not feasible, because it requires to much effort. A more general approach is the assignment of tags to handles. Each handle can have an arbitrary number of tags assigned. Possible handle connections are described in terms of collections of matching tags.

This tag-based handle-matching technique provides an easy-to-use and powerful functionality for dealing with most of the occurring part connections in an assembly modeling-based system.

4.2. Model import and export

Connected parts are called a part group and a part group can contain an arbitrary number of interconnected parts. The connectivity information of all parts of a part group is stored in an undirected graph called ConnectionGraph. If a new part is added to a part group, edges between the handles of the involved handle pair are inserted into the graph.

Since the ConnectionGraph contains all relevant information about all parts of a part group and its connections, it can be used for the import and export functionality. For exporting a part group, its ConnectionGraph is serialized in an XML-based graph description language. This graph description contains all connectivity information, however it does not contain any information about handles or geometric properties of the parts, but rather unique IDs of the involved parts. When importing a graph description, the system first determines if all needed parts and their handleinformation is already known. In case they are not known, the system dynamically loads their geometry and handle descriptions.

4.3. Process chain

Increasing complexity of manufacturing systems necessitates development of new flexible and adaptive approaches to operation control and product customization. The Eusponsored project INT-MANUS [1] develops a new technology based on such approaches. The core of the technology is so called Smart-Connected-Control Platform (SCCP) [3, 9] - a distributed agent-based system with semantics.

The platform integrates manufacturing machines, robots and operating personnel and it is able to organize and supervise production of complex products. The platform provides interfaces and services that can be used by different complementary systems. Our VR-Application is using the interfaces to schedule production of customized products.

When customization in the VE is finished, the result can be sent to Manufacturing Supervisory Service of the SCCP. The result consists of a list of parts needed for production, material information and assembly graph. The Service will automatically generate instructions for robots, manufacturing machines and operating personnel and will automatically supervise production and assembly processes.

5. Conclusion and future work

In this paper we presented an ongoing research work dedicated to Virtual-Reality-based product customization application development. We described our handle-based interaction technique, presented user interfaces, two-user functionality of the application, and explained how the application is integrated with manufacturing environment.

Our application is a relatively lightweight, but powerful tool, which can be applied at different manufacturing plants in order to facilitate product customization tasks. In our future work we are planning to develop a handle editing tool. Furthermore we will support part modifications using constructive solid geometry. In addition we will conduct usability tests of the application with a stress on two-user-related functionalities.

Acknowledgements

Intelligent Networked Manufacturing System (INTMANUS) is a project partly funded by the European Commission under the joint priority IST-NMP (NMP2-CT-2005-

016550).

6. References

[1] http://www.int-manus.org

[2] M. Agrawala, A.C. Beers, I. McDowall, B. Fröhlich, M. Bolas, and P. Hanrahan. The two-user Responsive Workbench: support for collaboration through individual views of a shared space. SIGGRAPH '97: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 327–332, 1997.

[3] M. Foursa, T. Schlegel, F. Meo, A. Herrmann Praturlon, J. Ibarbia, S. Kopacsi, I. Mezgar, D. Salle, and F. Hasenbrink. INT-MANUS: revolutionary control of production processes. *ISBN:1-59593-364-6, ACM SIGGRAPH 2006*.

[4] B. Jung. Task-Level Assembly Modeling in Virtual Environments. in: *Computational Science and Its Applications ICCSA 2003*, Springer LNCS, Vol. 2669, pp. 721–730, Springer Verlag, 2003.

[5] K. Kiyokawa, H. Takemura, Y. Katayama, H. Iwasa, and N. Yokoya. VLEGO: A Simple Two-handed Modeling Environment Based on Toy Blocks. *Proc. ACM Virtual Reality Software and Technology (VRST)*, 1996.

[6] K. Kiyokawa, H. Takemura, and N. Yokoya. SeamlessDesign: A Face-to-face Collaborative Virtual/Augmented Environment for Rapid Prototyping of Geometrically Constrained 3-D Objects. *Proc. IEEE International Conference on Multimedia Computing and Systems*, pp. 447–453, 1999.

[7] M. Pinho, D. Bowman, and C. Freitas. Cooperative Object Manipulation in Immersive Virtual Environments: Framework and Techniques. *Proc. ACM Virtual Reality Software and Technology (VRST)*, pp. 171–178, Hong Kong, 2002.

[8] K. Riege, G. Wesche, T. Holtk¨amper, and B. Fr¨ohlich. The bent pick ray: An extended pointing technique for multi-user interaction. *Proc. IEEE Symposium on 3D User Interfaces 2006*, pp. 62–65, Alexandria, USA, 2006.

[9] T. Schlegel, A. Srinivasan, M. Foursa, M. Bogen, G. Haidegger, I. Mezgar, J. Canou, D. Salle, F. Meo, J. Agirre Ibarbia, A. Herrmann Praturlon. INT-MANUS: Interactive Production Control in a Distributed Environment. *accepted to International Conference on Human-Computer Interaction (HCII) 2007 conference*, Beijing, China, 22-28 July 2007.

[10] S. Jayaram, U. Jayaram, Y. Wang, H. Tirumali, K.

[11] G. Zachmann and A. Rettig. Natural and Robust Interaction in Virtual Assembly Simulation. *Proc. Eighth ISPE International Conference on Concurrent Engineering: Research and Applications ISPE/CE2001*, July 2001

[12] Y. Zhang, R. Sotudeh, and T. Fernando. The use of visual and auditory feedback for assembly task performance in a virtual environment. *SCCG '05: Proc. 21st spring conference on Computer graphics*, pp. 59–66, Budmerice, Slovakia, 2005.

[13] W. Krueger, B. Froehlich The responsive workbench IEEE Comput. Graph. Appl., 17(4):12–15, 1994.

Developing a Framework for the Automatic Generation and Visualisation Of 3D Urban Areas on Mobile Devices

Christos Gatzidis, Vesna Brujic-Okretic
giCentre, Department of Information Science, City University
{c.gatzidis}, {vesna}@soi.city.ac.uk

Fotis Liarokapis
Department of Creative Computing, Coventry University
f.liarokapis@coventry.ac.uk

Stuart Baker
Alcatel-Lucent Telecom Limited, UK
stuart.e.baker@alcatel-lucent.co.uk

Abstract

Rapid procedural 3D urban modelling has been a constant research issue for a number of years now. We present a complete procedural 3D modelling solution for mobile device usage of the content, called Virtual City Maker, based on scripting algorithms allowing for both the automatic and also semi-automatic creation of photorealistic quality virtual urban content. The content produced can be specially optimized for use with mobile devices but also with navigational tasks in mind. Moreover, a user-centred mobile virtual reality (VR) visualisation and interaction tool operating on PDAs (Personal Digital Assistants) for pedestrian navigation is also discussed supporting the import and display of various navigational file formats equipped with a comprehensive front-end user-friendly graphical user interface providing immersive virtual 3D navigation to a wide range of users.

1. Introduction

Today there is a growing need for computer-based, photorealistic visualizations of 3D urban environments in many areas including environmental planning, engineering, telecommunications, architecture, gaming, 3D city information systems and even homeland security. Therefore, the procedural modelling of virtual cities in 3D is a topic not only computer graphics research but also other fields such as GIS or photogrammetry have focused on for a number of years. Different approaches have been favored with the two main ones being a fully-automatic or a semi-automatic one.

It has been accepted that fully-automatic approaches are categorized according to the input information used. The eldest and still an extremely popular data source for automatic virtual city generation are aerial images ([1], [2], [24]). Some disadvantages of aerial images include the fact that analyzing information of this low level in dense urban contexts can be complex but it has been accepted that they can be a very rich source of various kinds of information. Their main disadvantage listed above can be bypassed by using other special material in combination with them (additional height models, multiple overlap or colour images being some of this material).

Automatic city generation has also been achieved via the use of laser scanning data [3], [4]. This data, unlike aerial images, has the ability to represent the geometry of the surface directly. It has to be noted that the cost involved in acquiring this data is far greater than the one associated with aerial images, often almost prohibitive. However, technological advances since this type of data was first introduced have improved significantly with the cost decreasing progressively. DSMs (laser digital surface models) can offer very reliable information for segmentation and parametric reconstruction because of their nature. Unfortunately, they can also suffer from low density of measured points plus excessive automatic interpretation.

The last method for automatic urban content creation, and one that has been increasingly popular of late, is the use of 2D ground maps/plans [5], [6] from Geographical Information Systems (GIS). These can be very cost-effective in acquisition (often readily available) and are often used not just on their own but in conjunction with one of the previous data sources resulting in a more hybrid approach that can yield more accurate results. 2D ground maps can be an extremely high-level information source, much more so than laser scanning data or aerial images, however completeness and varying quality of the 2D ground maps can often pose inconsistency problems in the automatic reconstruction process.

While the ideal 3D urban modelling system would be a totally automatic one, there are a number of obvious advantages to semi-automatic or user-assisted approaches where essentially an automatic system employs some limited operator input for guidance in a range of the tasks involved in order to offer better control on the resulting scene. Furthermore, the user-

assisted (or interactive) tools can also be utilized as a more intelligent editing application for the preliminary model. For example, a popular approach is to offer suggestions to an automatic urban modelling system which then in turn concludes the modelling task at hand on its own. This can lead to a more efficient method for modeling complex structures. A more conventional approach to interactive modeling is to provide the user with a set of generic models that are then adjusted (using usually image data) altering the model and the viewing parameters [7]. In this approach, the system provides geometric computations but the drawback is that substantial time and effort are required from the user. Some other more recent approaches include the combination of user input with automatic processing introduced at various points with a different degree of control over the result. Offering an approximate building location to extract a building is one of the approaches suggested [8] that can lead to results although there is the important disadvantage of producing a final model very reliant on the automatic analysis. Other semi-automatic urban modelling tools have been described, some including methods for duplicating and/or cloning model meshes that are similar to others [9]. Yet another approach suggests an automatic system constructing topological relations amongst 3D roof points collected by a user for each roof. This method [10] results in a tool that can tackle easily several types of complex roofs. An additional notable interactive urban modelling approach involves the system handling complex building structures by means of constructive solid geometry [11]. This system also uses an image correlation method to fit a primitive to the image. It should be noted however that this last method can be very costly on processing power when modelling more complex urban sites.

One of the areas where 3D urban models are also in need today, apart from the ones mentioned in the beginning of this section, is for urban navigation using mobile devices. This can range to a number of sub-applications including car and pedestrian navigation systems, location-based services and others. It appears that while the methods for automatic and semi-automatic creation of 3D urban models today do cover a lot of ground both in terms of concepts and also in delivery of content, they fail to take into account the considerable limitations and challenges mobile devices have to offer. Constraints in terms of graphics memory, processing power and memory cards render the output of most of these systems unusable on devices such as PDAs or smartphones because of excessive detail. At the same time, care needs to be taken in presenting this content on mobile devices since, for example, typical PDAs are only equipped with screens no larger than a 480x640 resolution display, thus presenting a further challenge. In this paper we propose both a modelling solution (Virtual City Maker) that can deliver urban content efficiently to mobile devices as well as a virtual navigation environment (Virtual Navigator) on a PDA to accompany it so that that the resulting meshes can be put to the test. It should be noted that while the content is specifically optimised for mobile use in mind, for example a varying low-polygon count and adjustable texture image sizes, the results need to be of the highest quality possible rivaling virtual city models seen on desktop machines but also accurately placed in space to accommodate real-world navigational needs.

2. Virtual City Maker Solution

This section will describe in more detail the overall workflow for the Virtual City Maker modelling tool ranging from a description of the framework of the system to an overview of its two modes; the automatic and semi-automatic one with emphasis on some of the concepts, algorithms and interfaces behind them as well as some resulting 3D urban models.

2.1. Overview Of The System

First of all, regarding the input data used (a very important part of the process which affects the results in many ways) the authors of this paper have selected a combination of aerial photographs and also 2D ground maps, at least for the outline of the buildings modelled.

This more hybrid approach combines the strengths of two different techniques for a more efficient result. Aerial images ensure accurate results and the bypassing (to a certain extent) of a generalization process while ground maps provide positional and geographic information. Our work is not only focused on delivering content for mobile devices but also on navigation. Therefore, geo-referenced results and accuracy are key issues with errors in geometry placement only afforded to be marginal making this hybrid data use approach even more justified. Having placed building footprint outlines in a precise manner, these outlines have been extruded to real world heights using, if needed, GIS vector data in the form of the .shp file format. This particular file format operates as a combination of CAD data with added metadata attributes. Some of these attributes include three different heights for each building (from which our system derives the average value, again when the height cannot be derived from shadow detection, more on that on the next section) plus additional useful data

such as an individual ID for each building, its age, shape, type and others.

To conclude the 3D urban creation process terrestrial photographs are also essential to construct a photorealistic 3D virtual city model. These terrestrial photographs are captured manually from different viewpoints with the use of a conventional digital camera photographic technique.

All tasks or methods of the solution are tackled by a unified system with one user-interface (automatic mode). They can also fall into different categories of work which means that they can also be resolved by individual modelling plug-ins (semi-automatic mode) that have been designed to also work as stand-alone applications in their own right.

2.2. The Automated Mode

The key method in the automated reconstruction mode is the detection of building outlines and roofs from aerial images and/or 2D ground maps. The second other important step is the extrusion to real-world heights and that is handled, when needed, by a parser developed to read the GIS .shp files. Figure 1 illustrates the main processes involved in the building and roof detection system.

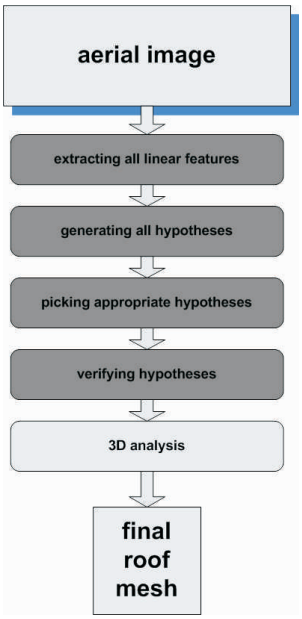


Figure 1. Roof /building detection block diagram.

The philosophy behind this has been to make only those decisions in the 3D analysis stage of the detection process that can be made in as much assurance at each level illustrated as possible. Therefore the hypothesis generation process creates hypotheses that may have somewhat weak evidence, which is where information from additional input data comes into use. Following that, a selection process then picks these based on more global evidence, filtering out the unwanted ones. The hypothesis verification process uses the most of this global information and can therefore make better decisions, resulting in a more accurate output.

To begin with, initial hypotheses are generated for 2D projections of building roofs. The reason behind that is that it is generally accepted that roofs are usually the most distinct features in aerial images from a nadir point of view (such as the ones our system uses) and that focusing on the roofs helps reduce the number of hypotheses to be examined later on [25]. The generation of all roof hypotheses is gathered by the restriction of the building shapes to rectangular form or similar compositions and also of roofs to be flat. Roofs of buildings of this shape should project into a parallelogram-like shape. To showcase a more detailed example of the process we followed here, as Lin [25] suggests, a 90º degree corner projected to an angle, o , given the angle π that one side of the projection makes with the horizontal and the viewing angles ψ and ξ is given by the following equation.

$$o = \pi \tan(\lambda, \omega) \text{ , where}$$

$$\lambda = [\cos^2(\pi + \psi)\cos(\xi)] + \frac{\sin^2(\pi + \psi)}{\cos\xi}$$

and

$$\omega = \sin(\pi + \psi)\cos(\pi + \psi)[\cos(\xi) - \frac{1}{\cos(\xi)}]$$

Figure 2. Parallelogram projection algorithm used.

Any parallelogram found must satisfy the relationships above. Following that, a selection process eliminates or keeps hypotheses based on what detectable evidence there is for them in the aerial image and on the global relationships amongst them all. The basis for the verification of 3D outlines in our work are the shadows buildings cast on the ground in aerial images. It is assumed that the direction of illumination in the image is known and also that the ground surface in the vicinity of the structure is flat. These two assurances allow for the computation of accurate height information. To offer more insight in the algorithms involved in this area, the building height is related to several parameters that can be measured from the image given. With X (pixels/meter) the image resolution in the neighborhood of the building location, H the projected wall height, can be computed (in pixels again) from the building height, B (in meters), and also the viewing angles by the following

algorithm, $H = (B \,.\, X \,.\, \sin\xi)$. The projected shadow width, W, can be calculated from the building height, the viewing angles and the sun angles (the direction of illumination, x , the direction of shadow cast by a vertical line, y , and the sun incidence angle I using the following algorithm as introduced again by Lin [25]).

$$W = \begin{cases} B \cdot X \cdot \tan I \quad when\ \gamma = 0 \\ \dfrac{B \cdot X \cdot \sin(I+\xi)}{\cos i} \quad when \begin{cases} \xi \neq 0 \\ y = x = 270° - \psi \end{cases} \\ \dfrac{B \cdot X \cdot \sin(I-\xi)}{\cos i} \quad when \begin{cases} \xi \neq 0 \\ y = x = 90° - \psi \end{cases} \\ \dfrac{B \cdot X \cdot \sin(\xi - I)}{\cos i} \quad when \begin{cases} \xi \neq 0 \\ y = x + 180° \end{cases} \\ \dfrac{B \cdot X \cdot \sin\xi \cdot |\cos(y+\psi)|}{|\cos(y-x)|} \quad else \end{cases}$$

Figure 3. Shadow width computation algorithm.

However, when surfaces are not flat in the aerial image or are cast on surfaces of nearby buildings, the quality of the 3D information is low. While still usable for other purposes it was decided to have an alternative. Nevertheless then, in this eventuality, the average height value from the GIS .shp data is used. This illustrates in detail why our choice of using a number of different input data can be so useful. The concept behind shadow analysis used in this work derives from the establishment of relationships between shadow casting elements and shadows cast [23].

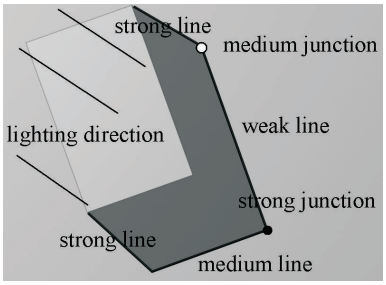


Figure 4. Detecting shadow evidence on image.

Provided the direction of lighting is known, an attempt is made to establish these relationships between shadow casting lines and the actual shadow lines and subsequently between shadow casting vertices or junctions and shadow vertices (illustrated in Figure 4). The ambiguous / non-visible vertical edges of buildings cast visible shadows in a direction parallel to the projection of the illumination on the ground. These shadow lines start at the edge of a building and can be detected fairly easily. There are occasions when these lines are cast in part on the ground and in part on nearby buildings or structures but even then the projection constraint can be justified and they can be detected.

2.3. The Interactive / Semi-Automated Mode

The interactive mode or user-assisted mode of Virtual City Maker has been developed to take advantage of the fact that operator input can not only enhance the automatically-produced results and improve them but also diversify the appeal of the system itself. Often there are cases where users require urban content that is not derived from real-world GIS coordinates but can be fictitious (mobile gaming applications for example). The semi-automatic mode of the application presented here can also cater for that too. In this section, some of the key individual plug-ins of the system that undertake these tasks will be presented.

2.3.1. Creating Buildings. Every virtual city consists of buildings, with these structures being the most important ones for the overall scene. One of the most important plug-ins for interactive or semi-automatic editing and creation of virtual cities, and thus buildings in particular, with the Virtual City Maker tool is the Building Creator script. Its interface is illustrated in Figure 5.

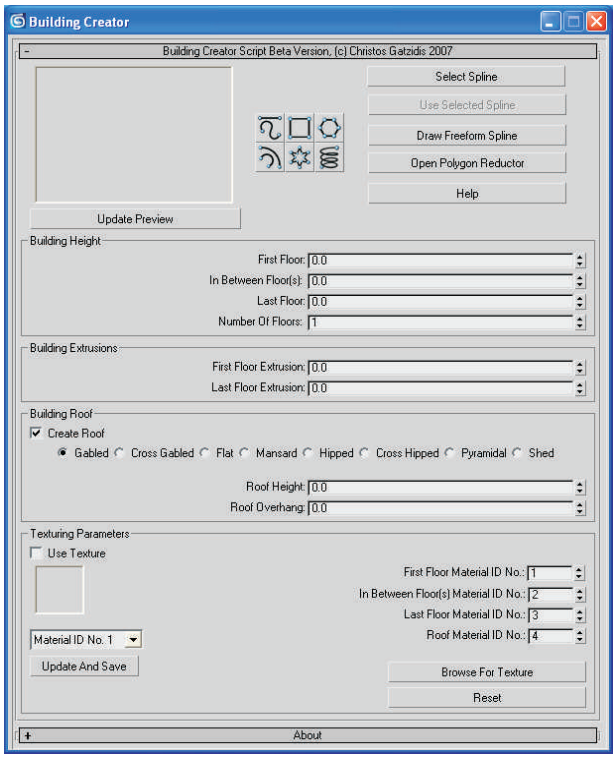


Figure 5. Building Creator script interface.

The version presented here supports the creation of 3D buildings from splines which can either be imported from CAD files such as 2D ground plans or can be freeform ones that the user can draw. Multiple floor segmentation for the building is offered as well as roof modelling with controllable height and overhang based on the user's selection (out of a range of roof shapes such as gabled, cross-gabled, flat, mansard, hipped, cross hiped, pyramidal and shed, all of these illustrated in Figure 6), the ability to create insets for additional geometric detail (such as for example segmentation between floors or creating a ground surface surrounding the building or even a terrace-like shape before the roof) and also allowance for texturing. This final process has been implemented by means of assigning different numerical material IDs to each floor and roof so that when the user selects an image to map onto a surface it will be saved according to this ID selection. This speeds up and simplifies the otherwise tedious process of texturing while at the same time giving the user full control on it.

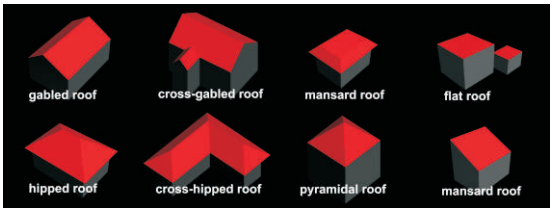


Figure 6. Some of the roof structures produced.

Some other features included are preview windows within the script's interface both for the texture and the model to assist the user with their modelling progress. Also linkage with another plug-in developed, called the Polygon Reductor and described below, is provided for the decreasing of mesh geometrical detail. It is worth-mentioning that a help function is also included for first time users.

2.3.2. Optimising Building Structures. Following that, another very important component script for the overall tool is the previously mentioned Polygon Reductor. This script offers the ability of reducing the polygon or face count for one or multiple buildings already generated while retaining its texture coordinates and also georeferenced positioning (the interface of the plug-in is shown in Figure 9). The algorithm which forms the basis of the Polygon Reductor script is based on a concept called edge decimation [21], [22]. Some concepts known as vertex decimation [20] and triangle decimation [19] are popular alternative approaches to polygon reduction / mesh simplification processes. In this case, i.e. with edge decimation, polygons are removed from the mesh by collapsing or contracting

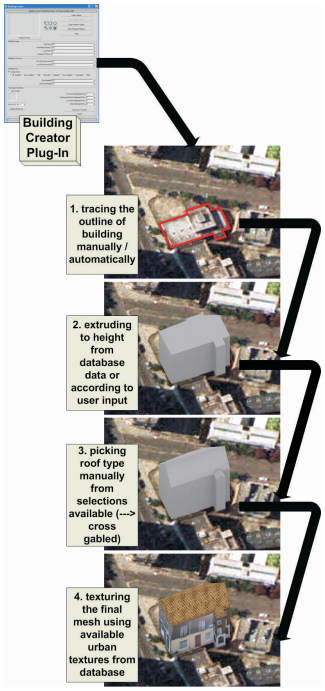


Figure 7. Step-by-step operation Building Creator script.

edges. That effectively means removing two triangles from an area of the mesh's surface thus simplifying it. The two vertices of the collapsed, decimated, edge are merged into one endpoint (Figure 8) and the triangle adjacency lists of the two original vertices are concatenated for the newly formed vertex.

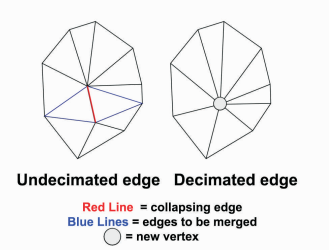


Figure 8. Polygon reduction algorithm used.

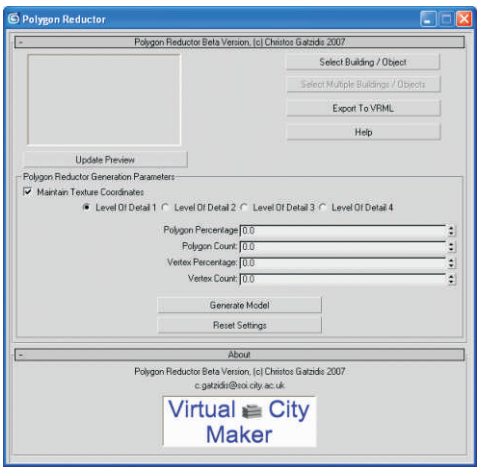


Figure 9. Polygon Reductor Script interface.

It should be noted that this contraction process has been made both progressive (for performance reasons, more on that below) and also reversible. In other words, every time an edge is collapsed and then merged there needs to be enough data kept to split the concatenated list back into the lists for each of the two vertices in question. To achieve that goal it has to be guaranteed that collapsing and then rebuilding an edge is a process that needs to be general enough to work on any random area of each given mesh. There are meshes where not all edges have two adjacent triangles, essentially, meaning with edges that are not continuous in nature. Therefore, to cater for this eventuality, the algorithm used makes a check on all degenerated-bound edges for the two adjacent triangles needed. This tackles one of the most important drawbacks of the edge decimation technique; uncontrollable edge contraction. Edge merging algorithms can often implicitly alter the topology of a model by closing holes in the surface if the importance of all edges is not evaluated.

In more detail Figure 10 illustrates the algorithm we have opted to use which is the fundamental edge decimation one (as described by Ronfard [31], amongst others) both for preprocessing and visualizing the optimized mesh. This approach has the following advantages: a) progressive representations of the original building model B^n with the continuous family of meshes (B^0, B^1, B^{n-1}, B^n) is very space-efficient plus has a smaller storage than the results of standard triangle/vertex approaches b) level-of-detail can be supported via the transformation from the B^i mesh to B^{i+1} by just applying the i-th vertex split/edge collapse operation c) the model can offer view-dependent or selective refinement.

Reconstruction pre-processing:

for (reach approximation index)

{

find and select edge whose decimation displays the smallest error measure;

collapse selected edge into one vertex;

store the collapse record also including one collapsed edge and two collapsed triangles;

}

return (base mesh & collapse records);

Visualization / reconstruction:

using the base mesh perform sequence of vertex split operations using the stored collapsed record;

produce a continuous family of meshes (B^0, B^1, B^{n-1}, B^n);

Figure 10. Progressive polygon red. algorithm used.

Finally, to select the target edges for the edge collapse function introduced by Hoppe [29] which has yielded the best results for our urban meshes compared to other error metrics:

$$E(B) = E_{dist}(B) + E_{spring}(B) + E_{scalar}(B) + E_{disc}(B)$$

where $E_{dist}(B)$ is the distance energy equal to the sum of squared distances from the points $X = \{x_1,...,x_n\}$ to the mesh, $E_{spring}(B)$ places on each edge of the mesh a spring of rest length zero and spring constant *k*, $E_{scalar}(B)$ measures the accuracy of its scalar attributes and $E_{disc}(B)$ measures the geometric accuracy of its discontinuity curves.

There are two ways to decrease the polygon count on a building mesh using this plug-in. The first is automatically, i.e. by selecting one of the four different levels of detail presented with each progressively decreasing polygon count by 25%.

The second is semi-automatically, by decreasing either the vertex or the polygon count/percentage. The resulting output can be previewed within the script, have its settings reset and also exported to the VRML file format for use with our visualization engine.

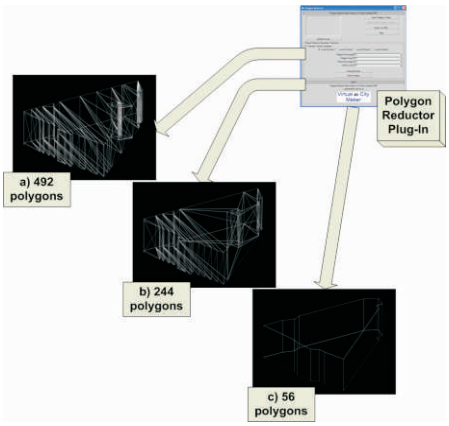


Figure 11. Reducing poly count with Polygon Reductor.

As an example of how crucial it is to reduce the polygon count of a building and also the difference it can make in achieving an acceptable frame rate in real-time visualization, a London building (illustrated in Figure 11) has been generated automatically with our method and is 492 polygons in total. Despite the fact that it is not prohibitively large, trying to visualize it in real-time with tens of similar geometrical structures on a mobile device with a graphics card barely capable of 3D acceleration is impossible. After using the plug-in at hand the face count of this model has been reduced to 244 and finally to 56 polygons. It should be noted here

that while polygon count has decreased to almost a tenth of the original amount, the building's main defining vertices have not been altered. This results in a model with not only the same boundaries as the original but also the same geographical shape.

2.3.3. Different Types Of Shading. Literature suggests [17], [18] that other ways of rendering 3D urban content can also be beneficial for communicating elements of numerous other principles such as cognition, cartography and non-photorealism. Thus an additional plug-in has been incorporated to the Virtual City Maker solution called City Shader. This facilitates the production of cartoon-shaded (via stylistic shaded rendering), clay-rendered and wireframe visualizations with various parameters. Most importantly, these different types of shading can be exported to suitable file formats for mobile use, which involves a number of issues considering mobile devices are not ideal to handle expressive visualizations. For example the clay-rendered model, before exported to VRML file format, had to have all its textures "baked" ensuring lighting and shadows information were kept because this particular file format does not support features such as advanced lighting or radiosity. The algorithm used for some of the results below is a variant of the one presented by Lake [30] which rather than smoothly interpolating shading across a model as in Gouraud shading (another popular approach), finds a transition boundary and shades each side of the boundary with a solid color. The pseudo-code for this process is presented in Figure 13.

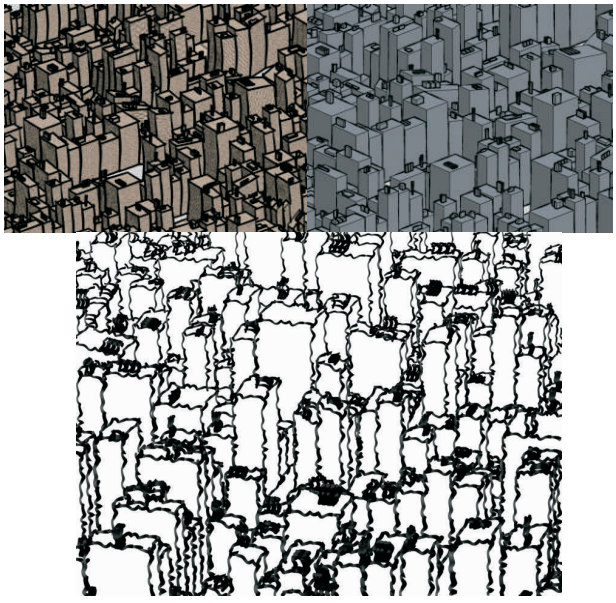


Figure 12. Some expressive urban renders using City Shader.

Pre-processing & visualizing a cartoon-shaded mesh

compute the illuminated diffuse color for each material using

$$S_i = (c_g \text{ x } c_m) + [(c_m \text{ x } i_a) + (d_s \text{ x } d_m)]$$

where S_i is the vertex color, c_g is the coefficient of global ambient light, d_s is the diffuse coefficient of the light source and and c_m and d_m are the ambient and diffuse coefficients of the object's material;

compute the shadowed diffuse color using

$$S_d = (c_g \text{ x } c_m) + (i_a \text{ x } c_m)$$

where i_a is the ambient coefficient of the light source;

for (each material)

{

create a one-dimensional texture map with two texels using the texture functionality

store this one-dimensional texture map

fill the texel (texture pixel) at the $i=1$ end of the texture with S_i and the texel at the $i=1$ end of the texture with S_d;

}

compute the one-dimensional texture coordinate at each vertex of the model using $Max\ \{\overline{V}\cdot\overline{u}, 0\}$, where $\overline{V}$ is the normalized vector from the vertex to the light source location, $\overline{u}$ is the unit normal to the surface at the vertex and $\overline{V}\cdot\overline{u}$, is their vector inner product;

return the rendered mesh with lighting disabled and one-dimensional texture maps enabled;

Figure 13. Pseudo-code for cartoon-shading algorithm.

2.3.4. Enhancing The Urban Environment. The impact of a virtual city both on a navigational and also on an aesthetical level can be enhanced by surrounding world objects, largely demonstrated by the research work in this area ([26],[27],[28]).

The main drawback of using these is that it can be very time consuming to create and place them on a fairly large scale urban model.

A solution to this, tackling foliage meshes, is the Tree Maker plug-in (user-interface illustrated in Figure 14). The Tree Maker script is in many ways representative of all the other work conducted on generation of models other than buildings for the Virtual City Maker solution. It targets performance and also automation.

All tree models generated consist of planes with adjustable height and width. These planes have opacity based bitmaps that create the illusion of a 3D mesh via transparency. They can have 2, 4, 6 or 8 sides, the more sides the better the illusion of a 3D mesh is, with a cost to real-time visualization performance.

An extensive texture library offers the user selection of plants in order to cover a number of different tree types. The user can also select any plane from the virtual city scene as a ground the models will rest upon

and then generate rows (or columns of trees) in both the x and y axis (with controllable distances).

Figure 14 illustrates some types of trees generated by the Tree Maker script and also how they can be placed into an urban scene, all developed by the rest of our solution.

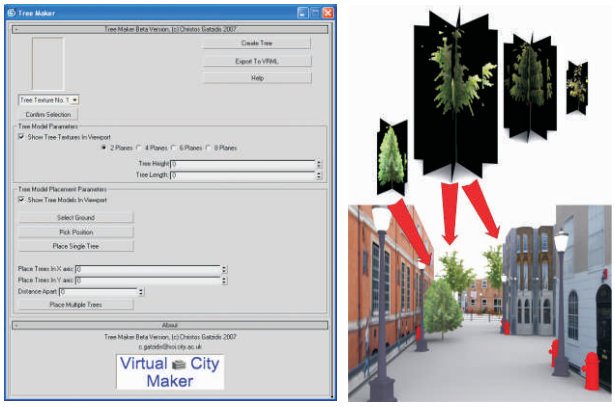


Figure 14. Generating & enriching scene with trees.

2.3.5. Other Plug-Ins. Other plug-ins developed, include tools for the generation of terrain meshes (Terrain Generator) encompassing the urban areas and also tools for producing an environment for the scene (Environment Maker) with a minimum amount of effort from the side of the user and with fully-controllable parameters. The terrain can be created with a different number of segments to provide different LODs (levels of detail) while mountain and hill like shapes can be created and instantiated with appropriate user input (positioning, size and noise).

As for generating the environment for the urban mesh some of the parameters supported include adjusting camera position and sunlight parameters such as lighting according to the time of the day, the shadows as well as the sun system orientation. Effects such as self-illumination, clouds (with user-adjusted size, quality and density) and finally a fog system can also be added.

3. Virtual Navigator Solution

Travel is the major component of navigation and usually involves unconscious cognition whereas wayfinding is the cognitive process of defining a path via an environment using and acquiring spatial knowledge based on both natural and virtual cues [12]. Although a number of mobile navigation systems have been designed, from industry and universities, they do not seem to have the appeal that was expected apart from the GPS in-car navigation systems. However, even these, have selected as the main visualisation environment to be the digital map which provides limited capabilities and they have now started to move into a rough 3D mesh visualisation which does not include photorealism. On the other hand, a few experimental mobile guides have been mainly developed by universities and are mostly based on abstract representations of the real environment without providing a robust solution for both intuitive navigation and wayfinding. In addition, they usually provide standard multimedia interfaces like audio or video operations without taking advantage of the capabilities of VR and user interface (UI) technologies. A brief overview of the most significant mobile VR applications has been recently documented [13], [14].

Part of the problem lies on the provision of meaningful spatial information in a realistic manner, so that inexperienced users do not have to put a great amount of cognitive effort to 'decode' the retrieved information (i.e. 3D map or wayfinding operations). Another significant concern relates to the lack of user-friendly graphical interfaces that will allow for the customisation of the information provided as well as the provision of different navigation tools in real-time. The emphasis on designing for continuous mobile interaction requires addressing the following features [15]: *have a clear start and end; allowing for interaction at any time from the user; simultaneous activities that operate concurrently; time characterisation of pedestrians and device and associative models of information.* To address these issues, a VR pedestrian mobile environment (which was originally developed at City University and now is under continuous development at Coventry University) called Virtual Navigator has been implemented.

Previous user-studies [13] indicated that the use of photorealism in 3D urban maps used for pedestrian navigation is helpful and more attractive than the standard 2D digital map representation. However, a few users reported some problems with the interaction techniques used, mainly because of the limited functionality the interface provided. Another interesting point [14] relates to the provision of choice to the user to accommodate sudden, external factors that may allow them to detour from a default path. In addition, it was positively suggested that the route line should be more distinct, minimising the probability of missing it while moving. To overcome these issues the graphical user interface (a prototype shown in Figure 15) for user-centred navigation and wayfinding was re-designed from scratch and is now divided into four categories including *file*, *routes*, *directions* and *tools*.

Figure 15. Virtual Navigator (manual mode).

The 'file' category contains the necessary functionality to open and close geo-referenced spatial maps represented in 3D. The virtual maps are currently stored in the device by using the wireless connectivity (GPRS or WiFi) so the 3D maps can be downloaded from a web-server. This allows Virtual Navigator to meet one of the most significant requirements of modern navigation systems which is to be operational anywhere and anytime. The 'exit' option permits to exit Virtual Navigation without the need of 'killing' the process from the memory control panel of the PDA.

Next, the 'routes' category allows mobile users to select the type of representation they require for navigation and wayfinding operations. The effectiveness of the wayfinding depends on the number and quality of wayfinding cues provided to pedestrians [12]. Earlier user studies [14] indicated that users prefer different types of wayfinding aids like lines and arrows. In our work, this includes 'arrows', 'lines', 'hotspots' and 'guides'. Arrows, lines and hotspots have been used individually in previous mobile prototypes [13],[14] but not the guides. The purpose of using different categories of routes is to provide a meaningful aid that has a clear start and end, assisting pedestrians using Virtual Navigator, to find their way and not to get lost. The size and colours of the arrows and lines can be customised allowing for personalisation of the cues used.

One of the points that were mentioned in previous user studies [14] is that the addition of recognisable landmarks would provide a clearer cognitive link between the VR environment and the real world scene. To address this effectively, first the 3D map was modelled using more detail such as including trees and lamps which are considered as landmarks from a large number of users. Besides, a number of hotspots that contain different types of functionality were added to the virtual 3D map making it interactive. In the simplest case, these include hyperlinks that link the 3D map with relevant web pages about the environment, but they can also provide links to other multimedia information such as digital pictures, audio and other 3D navigational information. In our work 'guides' are virtual representation of humans, also known as avatars. Their main objective within Virtual Navigator is to guide pedestrians towards the requested destination. Currently, we have designed only one avatar, with male characteristics, which is activated as soon as the user clicks on him. Subsequently, the avatar starts walking with a slow pace, guiding this way the user, from one of the entrances of City University, London (St John Street) to the main entrance (Northampton Square). The 'directions' category refers to the type of audio-visual information that can be provided to the pedestrians. The simultaneous presentation of audio-visual information meets one of the requirements (simultaneous activities that operate concurrently) of design issues in mobile interfaces. In the simplest case-scenario, textual directions provide information on how to navigate from one position to another. Additionally, textual annotations can be used effectively for presenting information about a place or a building (i.e. '…this is the main entrance of City University campus…'). Similarly, auditory directions perform the same action using pre-recorded wav files, but are prone to external noise. Finally, the 'tools' category, allows users to change some of the graphical and navigational properties of Virtual Navigator. These include the speed of navigation, the interaction type as well as the lighting of the scene. The speed of navigation can be customised according to the user needs accordingly (it was found that more experienced users prefer faster mode whereas inexperienced users like a slow pace). By increasing the altitude of the user location, through the 'interaction' menu, and altering the pitch to look directly down, it is possible to switch the view from a ground view into a bird's eye view of the surroundings. This is analogous to the standard map view and can be used for personal orientation and navigation [16].

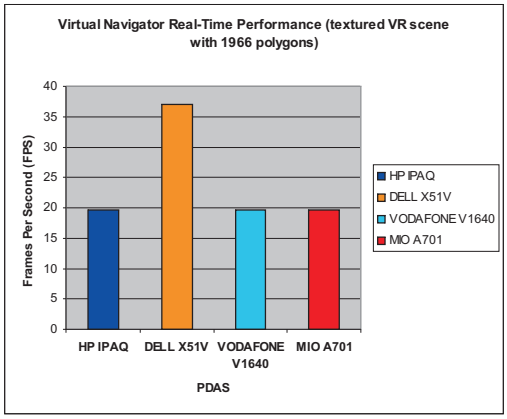


Figure 16. Virtual Navigator real-time performance.

Finally, we have performed a brief experiment to test the Virtual Navigator's real-time performance on a number of PDAs. Results (Figure 16) show that only the PDA (Dell Axim X51v) with a 3D accelerated graphics card can yield relatively high results in the FPS (frames per second) area (37.01 FPS being the highest value). The other three devices performed on the same level with negligible variations since our application switches to software rendering whenever it does not detect a hardware solution (19.60 to 19.62 FPS). In the future, a database with a management system will be designed to hold and maintain all the necessary geo-referenced spatial data into a secure server which will be used in urban navigation. To serve the demands of current user-needs, our server should feed the client, through an optimised network with minimum latency in real-time performance. More advanced routing services will be also developed to provide advanced navigational assistance to mobile users by making searches to the remote database as well as considering the behaviour for different modes of transport.

4. Conclusions And Future Work

While extensive work has been carried out in the methods and approaches behind automatic and semi-automatic creation of 3D virtual cities, to this day no comprehensive solution exists that can provide a complete pipeline from creating photorealistic georeferenced urban content to offering user-assisted enhancement and producing output specifically targeting mobile devices with all the challenges they have to offer. In this paper a novel 3D modelling solution has been presented consisting of a collection of plug-ins, called Virtual City Maker. The approach operates in two modes (automatic and manual) and is reliant upon using high-definition aerial and terrestrial photographic input as well as 2D ground maps and GIS data combined with a user-friendly customized interface. This hybrid method both in terms of input used and in modes results in virtual models of the highest photorealistic standards (fully-textured) and optimized (reduced number of polygons and texture image resolutions and with a number of different LODs). Apart from an accurate recreation and placement of building (including roof structures), other features include the semi-automatic control of supplementary parameters such as terrain, lighting (time of day) and environment (ground, fog, sunlit weather etc.) and also foliage. Furthermore, different types of shading are supported (wireframe, cartoon-rendered, clay-rendered etc.). The 3D urban areas produced have been imported in a mobile VR navigation environment, called Virtual Navigator, specifically created for the purposes of pedestrian navigation, wayfinding and exploration. This engine, operating on a PDA, is capable of reading different navigational file formats and includes a front-end user-centered interface for fully immersive virtual 3D navigation by manipulating 6 degrees of freedom such as observer location in three dimensions (x, y, z) plus orientation (yaw, roll and pitch). Both on the modelling and VR techniques presented above work is continuous and on-going. Future work includes: a) the introduction of a plug-in, which via access to a library of different meshes can interactively enrich the virtual city scene with a number of objects (such as benches, pavements, street lamps etc.) further enhancing the scene considerably with user-assisted input b) the introduction of a new plug-in that can intelligently distinguish between rural areas and more complex urban areas etc. and offer the user with a more "in-between" stage of creating a collection of building meshes c) the introduction of a new plug-in that can semi-automatically model landmarks according to their type (statues, cathedrals etc.) d) integrating semantic details to buildings so that interior details can be modelled too, such as inner walls to provide for indoor navigation e) conduct experiments with the resulting models, the engine and a number of human subjects in order to gain insight on the issues of efficient modelling for mobile navigation and assess some of the usability issues of the interface.

5. Acknowledgements

Work presented in this paper was funded by ALCATEL Lucent Telecom UK Limited.

6. References

[1] H. Mayer, Automatic object extraction from aerial imagery - a survey focusing on buildings, *Computer Vision and Image Understanding, (74)2*, 1999, pp. 138-149.

[2] C. Jaynes, E. Riseman, and A. Hanson, Recognition and reconstruction of buildings from multiple aerial images, *Computer Vision and Image Understanding, (90)1*, 2003, pp. 68-98.

[3] A. Wehr, and U. Lohr, Airborne LASER Scanning - an introduction and overview, *ISPRS Journal of Photogrammetry and Remote Sensing, 54(2)*, 1999, pp. 68-82.

[4] H. Maas, Fast determination of parametric house models from dense airborne laser scanner data, *Proceedings of ISPRS Workshop on Mobile Mapping Technology*, Bangkok, Thailand, 1999.

[5] G. Vosselman, and I. Suveg, Map based building reconstruction from laser data and images, *Proceedings of Third International Workshop On Automatic Extraction Of Man-Made Objects From Aerial And Space images*, Ascona, Switzerland, 2001, pp. 231-239.

[6] C. Brenner, City models - automation in research and practice, *Photogrammetric Week 01*, Wichmann Verlag, 2001, pp. 149-158.

[7] T. Strat, L. Quam, J. Mundy, R. Welty, W. Bremner, M. Horwedel, D. Hackett, and A. Hoogs, The RADIUS Common Development Environment, *Proceedings of the DARPA Image Understanding Workshop*, San Diego, CA, 1992, pp. 215-226.

[8] J. Li, R. Nevatia, and S. Noronha, User Assisted Modeling of Buildings from Aerial Images, *Proceedings of IEEE Conference On Computer Vision And Pattern Recognition*, Ft. Collins, CO, USA, 1999, pp. II:274-279.

[9] Y. Hsieh, SiteCity: A Semi-Automated Site Modeling System, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, USA, 1996, pp. 499-506.

[10] A. Gruen, and H. Dan, A Topology Builder for Automated Building Model Generation, *Proceedings of Automatic Extraction of Man Made Objects from Aerial and Space Images (II)*, Birkhauser, Basel, 1997, pp. 149-160.

[11] E. Gülch, H. Müller, and T. Läbe, Integration of Automatic Processes Into Semi-Automatic Building Extraction, *Proceedings of ISPRS Conference Automatic Extraction Of GIS Objects From Digital Imagery*, Munich, Germany, 1999.

[12] D. Downman, E. Kruijff, J. LaViola, and I. Poupyrev, 3D user interfaces: theory and practice (Boston, Addison Wesley, pp. 183-254, 2005).

[13] C. Gatzidis, F. Liarokapis, and V. Brujic-Okretic, Automatic modelling, generation and visualisation of realistic 3D virtual cities for mobile navigation, *Proceedings of the 9th International Conference on Virtual Reality*, Laval, France, 2007, pp. 225-234.

[14] F. Liarokapis, V. Brujic-Okretic, and S. Papakonstantinou, Exploring urban environments using virtual and augmented reality, *Journal of Virtual Reality and Broadcasting, Digital Peer Publishing, 3(5)*, 2006, pp. 1-13.

[15] A. Dix, J. Finlay, G. Abowd, and R. Beale, *Human-computer interaction* (Harlow, Prentice Hall, 2004).

[16] D. Mountain, and F. Liarokapis, Interacting with virtual reality scenes on mobile devices, *Proceedings of the 7th International Conference on Human Computer Interaction with Mobile Devices & Services*, Salzburg, Austria, 2005, pp. 331-332.

[17] J. Schumann, T. Strothotte, S. Laser, and A. Raab, Assessing the effect of non-photorealistic rendered images in CAD, *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*, Vancouver, British Columbia, Canada, 1996, pp. 35-41.

[18] J. Dollner, and M. Walther, Real-Time Expressive Rendering of City Models, *Proceedings of the Seventh International Conference on Information Visualization (IV'03)*, Seattle, WA, USA, 2003, pp. 245-250.

[19] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen, Decimation of triangle meshes, *Proceedings of SIGGRAPH '92: annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1992, pp. 65-70.

[20] B. Koh, and T. Chen, Progressive VRML Browser, Proceedings of IEEE *International Workshop on Multimedia Signal Processing*, Copenhagen, Denmark, 1999, pp. 71-76.

[21] M. Garland, and P. Heckbert, Surface Simplification Using Quadric Error Metrics, *Proceedings of SIGGRAPH '97: annual conference on Computer graphics and interactive techniques*, Los Angeles, USA, 1997, pp. 209-216.

[22] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, Mesh optimization, *Proceedings of SIGGRAPH '93: annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1993, pp. 19-26.

[23] C. Lin, A. Huertas, and R. Nevatia, Detection of Buildings using Perceptual Grouping and Shadows, *Proceedings of IEEE Computer Vision and Pattern Recognition Conference*, Seattle, WA, USA, 1994, pp. 62-69.

[24] R. Nevatia, and K. Price, Automatic and interactive modeling of buildings in urban environments from aerial images, *Proceedings of IEEE International Conference On Image Processing*, Rochester, New York, USA 2002, pp. 525-528.

[25] C. Lin, and R. Nevatia, Building Detection and Description from a Single Intensity Image, *Computer Vision And Image Understanding Journal*, 72(2), 1998, pp. 101-121.

[26] A. Reche-Martinez, I. Martin, and G. Drettakis, Volumetric reconstruction and interactive rendering of trees from photographs, *ACM Transactions on Graphics (TOG)*, 23(3), 2004, pp. 720-727.

[27] L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S. B. Kang, Image-based plant modeling, *ACM Transactions on Graphics* (TOG), 25(3), 2006, pp. 599-604.

[28] I. Shlyakhter, M. Rozenoer, J. Dorsey, and S. Teller, Reconstructing 3D Tree Models from Instrumented Photographs, *IEEE Computer Graphics and Applications,* 21(3), 2001, pp. 53-61.

[29] H. Hoppe, Progressive Meshes, *Proceedings of SIGGRAPH '96: annual conference on Computer graphics and interactive techniques*, New Orleans, Lousiana, USA, 1996, pp. 99-108.

[30] A. Lake, C. Marshall, M. Harris, and M. Blackstein, Stylized rendering techniques for scalable real-time 3D animation, *Proceedings of NPAR 2000: First International Symposium on Non Photorealistic Animation and Rendering,* Annecy, France, 2000, pp. 13-20.

[31] R. Ronfard, and J. Rossignac, Full-range approximation of triangulated polyhedra, *Computer Graphics Forum Journal*, *15(3)*, 1996, pp. 67-76.

Simplification Using Quadric Error Metrics, *Proceedings of SIGGRAPH '97: annual conference on Computer graphics and interactive techniques*, Los Angeles, USA, 1997, pp. 209-216.

An Application of Augmented Reality in Architectural Education for Understanding Structural Behavior through Models

Claudia Susie C. Rodrigues[1], Paulo F. N. Rodrigues[2], Claudia M. L. Werner[1]

[1]Federal University of Rio de Janeiro
COPPE - System Eng. and Computer Science - Rio de Janeiro, RJ, Brazil
{susie}, {werner}@cos.ufrj.br

[2]Federal University of Rio de Janeiro
FAU – Faculty of Architecture and Urbanism - Department of Structures
Rio de Janeiro, RJ, Brazil
pfnr@fau.ufrj.br

Abstract

One of the most difficult pedagogical problems in architectural education is the interface between structural and architectural design issues. As a matter of fact, the student is usually concerned with the form conception and sometimes the structural understanding does not reach a desirable level. The introduction of this subject at the Faculty of Architecture and Urbanism at Federal University of Rio de Janeiro (FAU/UFRJ) occurs with the study of modeling of structural systems for the simulation and testing through physical models in a qualitative analysis, which seems to be a helpful tool for the comprehension of engineering and design basic concepts. The approach discussed in this paper presents an application of augmented reality, developed in a collaboration with COPPE/UFRJ - System Engineering and Computer Science, in order to motivate and introduce the students into structure studies in a more pleasant manner and suitable to the architect's profile.

1. Introduction

The study of structures is an essential part of architectural education. It seems to be a hard task for students and teachers due to the mathematical terms inherent to engineering concepts. Particularly, taking the architecture student's visual intuition into account, the basic behavior of the structural elements may be more comprehensive by means of a qualitative analysis through the observation of the deformed configurations and properties of the structural systems. The qualitative structural analysis applies its concepts in a nonmathematical approach [1, 2, 3, 4, 5].

The use of physical reduced models made of relatively flexible materials, such as silicon, rubber, cardboard and polystyrene, in order to simulate the deflections of the structural members, was originally presented in the sixties by Salvadori & Tempel [4], being their illustrated teacher's manual on why buildings stand up the first one to use the concept of a visual approach [6]. The structural analysis executed with the help of models to enhance the visualization of the displacements due to the action of loads, provides an easy and efficient manner of understanding the basic engineering concepts, such as tension, compression, bending, buckling and torsion, employing the student's feeling and intuition [7].

The evolution of multimedia made the development of computer-aided tools for the interactive learning environment possible. The application of such tools is an important step to fulfill the needs of architecture students. Many contributions by researchers on this subject, applying some virtual reality techniques, were published during the last two decades [8, 9, 10, 11, 12].

An excellent tool for the architectural education with the aim of bypassing the structural systems teaching and learning difficulties that take place in the first years of architecture and engineering courses is the introduction of computer teaching software employing the Augmented Reality (AR) concepts [13, 14].

This paper presents an application of augmented reality using the ARToolkit library [15, 16] for understanding structural behavior of qualitative models to enhance learning for novice students in architecture and engineering.

The remainder of this paper is organized as follows: Section 2 briefly describes the methodology applied to the Modeling of Structural Systems course at Federal University of Rio de Janeiro. Section 3 details the Augmented Reality Application of this work. Finally, in section 4 conclusions and directions for future work are presented.

2. Modeling of Structural Systems

The study of structures is an important part of a student of architecture's education. Therefore, the

capacity to engage with information held in structures is vital to the study of architecture. In the Faculty of Architecture and Urbanism at Federal University of Rio de Janeiro (FAU/UFRJ), the introduction of this subject occurs with a mandatory discipline of modeling of structural systems for the simulation and testing of physical models, which seems to be a helpful approach for the comprehension of the engineering and design basic concepts. At this stage, the most important characteristics and properties of basic elements along with the structural compositions are thoroughly studied. The main purposes are to allow the students to acquire the needed sensibility and motivate them, awakening the interest for the study of structural analysis in the following years. The most complete comprehension of basic concepts is considered as a prerequisite before getting into the analysis modules.

The methodology applied during the course period includes the use of intuition in the learning process by means of examples of structures found in nature, of the observation of structures from the past and present, their successes and failures, and of the responses to certain loads of reduced structural models made by students and teachers in the classroom. According to the summary of the discipline, the students are obliged to build these physical models and interact with them, mainly, by manipulation and observation. The intention is to show the pupils the structure's functionality in architecture, its relevance in the design and building processes, introducing the structural systems knowledge in a qualitative way, including their properties and behavior when subjected to determined loads. Then it's possible to facilitate the understanding of basic concepts, such as tension, compression, buckling, torsion, among others, taking advantage of the fact that architectural students are visual learners. The subject becomes more attractive and fascinating, turning into a background for the structural analysis, without being superficial. As a matter of fact, the qualitative experimental analysis is an easy way of guiding the intuition through visualization and feeling.

The initiative of employing the augmented reality techniques has been well accepted by undergraduates, who have begun to provide interesting suggestions for improvements. The main advantage of the augmented reality environment over other computer based tools is that it enables the user to interact with the real world, enhancing his or her perception of this environment, through virtual information. This distinguishing feature fits very well with the purposes of the discipline of modeling of structural systems.

3. Augmented Reality Application

The purpose of this paper is to present a development of an AR application, employing the ARToolKit library in order to improve the teaching and learning environment for the architecture undergraduates in a scientific and academic cooperation between FAU/UFRJ and COPPE - System Engineering and Computer Science. By adding virtual elements which are associated to the student's models, the application shows useful information, such as values, directions and positions for the support reactions, stressed regions and deformed configurations. The obligatory manipulation of the reduced models only allows the visualization of the form and the deformed configurations of the structure. The present AR application provides this extra information, not separated from the model, as usual. This contribution is very important and helpful to the learning process and means a complement to the student's knowledge.

Two models were necessary to be built. The first one is a simply supported beam, and the second a continuous beam with two equal spans. The reduced models were built of PVC, which is a relatively flexible material, in order to simulate the deflections of the structure.

The first step towards developing the implementation was an environment preproduction. It was important to prepare this environment with the objective of simulating the scenery of utilization in classroom due to the nature of the application.

The webcam is used in order to capture images containing physical markers printed on a cardboard. The ARToolkit identifies the marker and places a virtual object in the card position. The marker was printed on a rigid cardboard in such a way that it could resist handling and huge manipulation. When the marker is damaged by folding or smashing, it becomes difficult for the ARToolKit program to recognize it. The marker was drawn in an adequate size in order to fit to the camera distance.

The program was implemented to provide the user with three kinds of virtual visualization on the screen:

- Support reactions due to the applied load P;
- Relative distance of load application (a/L);
- Stressed regions;
- Deformed configuration.

The two types of models used by the program must be identified during the calibration stage. It also allows the visualization of support reactions and the perception of stressed regions.

The applied load and the support reactions of the simply supported beam model, as well as its deformed configuration, are presented in figure 1.

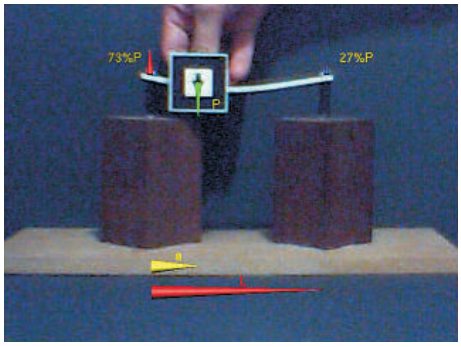


Figure 1. Support reactions of the simply supported beam model.

With the aim of better understanding the structural behavior of a beam subjected to a single concentrated load, the student pressures the structure in the vertical direction using his fingers. While using the program, the webcam captures the image of the student pressing the model, with the marker in his finger. As the user moves the card, the tool shows virtual information overlaid to the beam representing the support reactions, the load "P" and its position "a" in relation to the span "L" of the beam. The load "P" and the support reactions change their lengths due to the intensity of external force applied by the finger. Whenever the point of application of the load is modified by the user, the support reaction values also change according to the equilibrium equations. The program takes this into account by changing the image size of the forces in a certain dimensional scale.

Figure 2 shows the applied load, the support reactions and the deformed configuration of the continuous beam model.

In the stressed regions mode, the procedure is the same presented before, by pressing the structure, making use of the marker. The tension and compression regions are displayed and represented by letters "T" and "C", respectively (figure 3).

The application of this tool does allow the architectural student to develop the capacity for qualitative observation and to cultivate his feeling, perception, and intuition for structural behavior, enhanced by the following conclusions:

- The greater the applied load "P", the greater the displacements.
- As the load "P" increases, the support reactions increase proportionally.
- The sum of the support reactions is equal to the applied load.
- The nearer the applied load is to the support, the greater its reaction.
- In the two-span beam, when the force is applied only to one of the spans, the other span tends to uplift, and its far end support reaction, distant from the central support, changes its direction.
- The deformed configuration and the stressed regions (where tension or compression occurs) are easily visible.

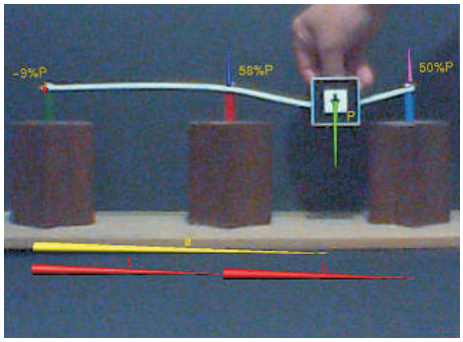


Figure 2. Support reactions of the continuous beam model,

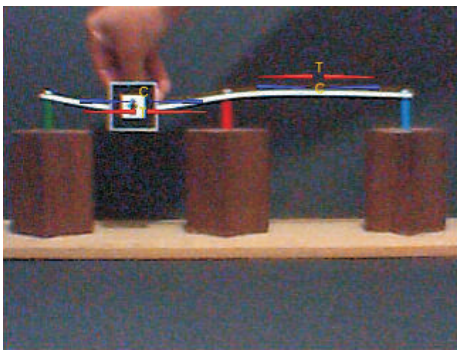


Figure 3. Stressed regions in the continuous beam model.

The main difficulty found was the instability of the virtual objects, because of the limitation of the ARToolkit tracking algorithm. This limitation occurs due to the light of the environment, the camera type used, the angle and the distance between camera and the markers, among others. In this work, two types of cameras were used: SoC PC-Camera and Creative WebCam NX.

4. Conclusions

The introduction to structural analysis at the Faculty of Architecture and Urbanism of Federal University of Rio de Janeiro (FAU/UFRJ), using reduced models

built of flexible materials, has been enhanced with the employment of Augmented Reality techniques.

Furthermore, this procedure provided the students with a better perception of the displacement tendency of the beams, in a qualitative manner, and also made possible a gain of knowledge of the structures responses, when subjected to a concentrated load at different points.

This paper presented the development of an Augmented Reality (AR) application, using the ARToolKit library with the aim of improving the teaching and learning environment for the architecture students in a scientific and academic cooperation between FAU/UFRJ and COPPE/UFRJ - System Engineering and Computer Science.

The student's structural understanding used to be acquired by the interaction with models, theoretical classes and intuition. Now, with the AR application, it is reached through a concrete, reinforced and consolidated way. Additionally, most students felt that the use of this computer-aided tools was a valuable increment to the architecture course's curriculum.

Finally, a new version is under development. It will focus on the refinement and optimization of the program code, which will result in the tool upgrade supporting more complex frame models, including the ones that will be created by the students. Besides, a case study, as a formal experiment with the students, is planned to be conducted at FAU/UFRJ.

5. References

[1] Salvadori, M., and Hel*Proc. IWAR '99,* San Francisco, CA, USA, Oct. 20-21, 1ler, R., *Structure in Architecture*, Prentice Hall, 3rd edition, New Jersey, 1963.

[2] Polillo, A., *Considerations on Structural Education in Architecture Courses (in Portuguese)*, Sedegra-Rio, Rio de Janeiro, 1968.

[3] Gordon, J. E., *Structures or Why Things Don't Fall Down*, Da Capo Press, London, 1978.

[4] Salvadori, M. G., and Tempel, M., *Architecture and Engineering: An Illustrated Teacher's Manual On Why Buildings Stand Up*, Salvadori Educational Center On The Built Environment (SECBE), 3rd edition, New York, 1983.

[5] Hilson, B., *Basic Structural Behaviour – Understanding Structures From Models*, Thomas Telford, London, 1993.

[6] S. Abdelmawla, M. Elnimeiri, and R. Krawczyk, "Structural Gizmos", *Proceedings of ACADIA00 conference: Association for Computer Aided Design in Architecture*, Washington, DC, USA, Oct., 2000, pp. 115-121.

[7] P. F. N. Rodrigues, and A. S. Hermida, "Modeling Basic Structural Elements for the Qualitative Structural Behavior Analysis" (in Portuguese), *Revista de Ciência e Tecnologia*, Facet, Rio de Janeiro, Brazil, Vol. 6, No. 1, Jun., 2006, pp. 5-13.

[8] M. Piccolotto, and O. Rio, "Structural Design Education With Computers", *Proceedings of ACADIA 95: Computing in Design, University of Washington*, Seattle, USA, Oct., 1995, pp. 285-299.

[9] S. Vassigh, "Structures E-Book. ACADIA Quarterly". V. 18, No. 3, *Association for Computer Aided Design in Architecture*, Salt Lake City, Utah, USA, Oct., 1999, pp. 14-15.

[10] S. G. Guidera, "Exploring the Architecture of Structure: Integrating Structures into Design Studio Using Object-oriented", *CAD 2003 ASEE Annual Conference & Exposition: Staying in Tune with Engineering Education,* Nashville, TN, USA, Jun., 2003, pp. 22-25.

[11] I. L. Kim, O. L. Weck, W. Nadir, P. Young, and D. Wallace, "Innovative Modern Engineering Design and Rapid Prototyping Course: A Rewarding CAD/CAE/CAM Experience for Undergraduates", *American Society of Engineering Education (ASEE) 2004 Annual Conference & Exposition,* Salt Lake City, Utah, USA, Jun. 20-23, 2004.

[12] S. G. Millard, "Computer Aided Learning (CAL) and Computer Aided Assessment (CAA) in Civil Engineering", *Proceedings of the 3rd International CDIO Conference*, MIT, Cambridge, Massachusetts, USA, Jun. 11-14, 2007.

[13] R. T. Azuma, "A Survey of Augmented Reality", *Presence: Teleoperators and Virtual Environments*, V. 6, No. 4, 1997, pp. 355-385.

[14] R. T. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent Advances in Augmented Reality", *IEEE Computer Graphics and Applications*, V. 21, No. 6, Nov./Dec., 2001, pp. 34-47.

[15] H. Kato, "ARToolkit", Human Interface Technology Laboratory, University of Washington, Available in: <http://www.hitl.washington.edu/artoolkit>, Acessed in: May 02, 2007.

[16] H. Kato, and M. Billinghurst, "Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System", 999, pp.85-94.

Um Estudo de Caso sobre a Engenharia de Sistemas Baseada em Componentes para Realidade Virtual

Eduardo Filgueiras Damasceno[1,2],Luiz Fernando Braga Lopes[1,2,4], Fábio Montanha Ramos[1,2] , José Barbosa Dias Jr[1,5] , Alexandre Cardoso[1]

[1]Universidade Federal de Uberlândia – Minas Gerais - Brasil
Programa de Pós-Graduação em Engenharia Elétrica

[2]Universidade Paranaense – Paraná – Brasil

[3]Centro Federal de Educação Tecnológica de Rio Verde – Goiás - Brasil

[4]Centro Universitário de Maringá – Paraná – Brasil
Universidade de Rio Verde – Goiás - Brasil

edamasceno@cefetrv.edu.br, lfbraga@wnet.com.br, fa_montanha@yahoo.com.br, juniorddos@yahoo.com.br

Abstract

This paper shows a study case of Component Based Software Engineering and a comparison of 3Dlibraries, toolkits and frameworks for Virtual Reality Deployment .

Key-Words: *Component Based Software Engineering, Virtual Reality.*

1. Introdução

No processo de desenvolvimento de interfaces com o usuário tridimensionais (3DUI – 3D User Interface), com um enfoque para as interfaces para realidade virtual e aumentada, diversas técnicas tentam sistematizar um processo de desenvolvimento em função do reuso de suas partes para atenuar o tempo de desenvolvimento.

No intuito de atenuar esta complexidade e promover o reuso do software como disciplina dentro do foco de desenvolvimento artístico (gráficos) e computacional, o uso de um processo de desenvolvimento baseado em componentes acelera o trabalho e aumenta a qualidade do mesmo.

2. O Desenvolvimento de Interfaces Tridimensionais

A Figura 1 mostra uma visão geral da hierarquia das principais ferramentas de desenvolvimento para Ambientes Virtuais (AV). Esta hierarquia é composta pelo Sistema Operacional como Windows, Linux, entre outros. Bibliotecas de comunicação e sincronização (utilizadas em aplicações distribuídas), bibliotecas como OpenGL [15] e DirectX[18], e pacotes gráficos (coleção de classes que oferecem um conjunto de serviços) como GLScene[2], Ogre [17], ArToolKit [18], Java3D [19], e pacotes de desenvolvimento que são ferramentas direcionadas a confecção de AV ou expressa diretamente em uma linguagem de programação como Java, C++ ou Delphi. Nota-se que existe uma vasta gama de opções disponíveis para cada um dos graus da hierarquia, e que estas opções podem ser combinadas para criar um AV [14].

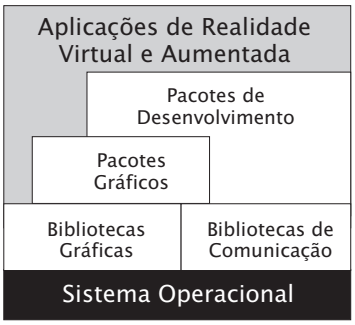


Figura 1: Hierarquia das ferramentas de desenvolvimento de AV.

3. Desenvolvimento Baseado em Componentes

Atualmente, o desenvolvimento de interfaces tridimensionais interativas com o usuário está cada vez mais complexo e quanto mais tecnologias tanto de linguagem de programação, bem como de bibliotecas e mecanismos gráficos aparecem para atenuar o esforço computacional [1].

Após o processo de modelagem arquitetônico do Ambientes de Realidade Virtual (ARV) faz-se necessária uma análise crítica sobre qual tecnologia de desenvolvimento e qual componente de software 3D deve-se usar para a implementação.

Uma vez estabelecida essa arquitetura, deve-se escolher os componentes mais viáveis e bibliotecas preferencialmente reutilizável ou se construir um novo componente próprio [12].

No desenvolvimento de ARVs deparam-se os programadores com tecnologias que se homogeneízam e que muitas vezes se confundem que são: a) Bibliotecas API, b) ToolKits ou pacotes de desenvolvimento e; c) Frameworks ou arcabouços de software.

3.1 Biblioteca API

Biblioteca API (*Application Programming Interface*) é a definição mais antiga e a tecnologia mais usada de uma coleção de subprogramas que contêm implementações de rotinas úteis que podem ser reutilizadas em outras aplicações. Existem dois tipos de Bibliotecas API, as Procedurais e as Classes de Bibliotecas.

As bibliotecas API não têm uma interface de contato definida, podem ser acessadas em qualquer situação ou pelo Sistema Operacional (SO), mas devido às suas funcionalidades estarem associadas à arquitetura do SO não são facilmente substituíveis.

3.2 ToolKits

Os ToolKits ou Software Development Kits (SDK), como sua definição por tradução diz, são um conjunto de ferramentas (rotinas, subprogramas, utilitários e outros programas complementares) que ajudam no desenvolvimento de um software final que comummente são disponibilizados pelas empresas que fornecem outro produto associado a este.

3.3 Framework ou Arcabouço de Software

Um Framework ou Arcabouço de Software é uma estrutura de suporte definida em que um outro projeto de software pode ser projetado, organizado e desenvolvido a partir dele [12].

Os Frameworks são projetados com a intenção de facilitar o desenvolvimento do software, habilitando projetistas e programadores a gastarem mais tempo determinando as exigências do software do que com detalhes técnicos tediosos, de baixo nível da programação de um sistema.

4. Estudo de Caso

Independente da opção a ser escolhida a composição integrada de uma arquitetura padrão deverá ser testada minuciosamente. No processo de desenvolvimento de um componente são observados alguns fatores como já ditos, a qualificação, a adaptação e a composição do mesmo.

De forma geral os componentes re-usáveis para ARV deverão possuir algumas características como: a) independência de dispositivos; b) abstração do sistema de projeção; c) grafos de cena ou hierarquia de cena; d) processo de renderização; e) suporte à navegação, seleção e manipulação de objetos; e f) suporte a sistemas distribuídos. Mas sem perder a simplicidade e flexibilidade de seu uso.

Características como realismo de cena, aparência dos objetos renderizados e o comportamento (interação com o usuário) também são quesitos importantes para um componente de ARV. De acordo com os estudos realizados por [11] diversos fatores contribuem para fazer escolha da arquitetura, dos quais emergem: a) Familiaridade que o programador tem com a ferramenta; b) Custo da ferramenta; c) Arquitetura de hardware; e d) Arquitetura de software Agregam-se a esta análise os relatos de [9] que são: i) Tempo de Desenvolvimento; ii) Composição do Componente; e iii) Flexibilidade da aplicação.

4.1 Componente GLScene

O GLScene é um componente Open-Source, licenciada pela Mozilla Public License (MPL), que possui uma grande quantidade de componentes de software, que permitem a descrição de objetos e renderização de cenas em 3D, possuindo as características de reutilização apresentada até aqui.

O GLScene possui programação orientada a objetos [2], no qual utiliza conjuntos de classes para criação do ambiente genérico 3D, aliados a tecnologia RAD (Rapid Application Development).

4.2 Componente WorldUp

Este Componente utiliza a tecnologia ActiveX, que segundo [4] é uma forma proprietária da Microsoft de desenvolver a integração entre aplicações encapsulando os detalhes de implementação, e é possível construir aplicações 3D e de RV utilizando o WorldToolKit da Sense8, conforme [3]. Também suporta vários dispositivos convencionais, mas os não-convencionais são utilizados agregando outros SDK como é o caso do Virtual Hand SDK [5], e o Microsoft Speech SDK [6].

4.3 VR Juggler

Diferente dos componentes anteriores, este é um Framework e não um componente que pode ser adicionado em uma ferramenta RAD. Desta forma este Framework fornece uma camada de abstração muito maior que os anteriores não dependendo de outros SDK para acoplamento de dispositivos não convencionais e de sistemas de projeção. Esta última, é sua melhor qualidade pois é possível a visualização do ambiente em Monitor, em Salas de Projeção tipo Caverna (CAVE) , e nos óculos tipo HMD [8].

4.4 Open SceneGraph (OSG)

De acordo com [10] este Framework é um conjunto de funcionalidades construídas sobre o OpenGL e o OpenAL [16] liberando estes da necessidade de implementação e otimização de chamadas gráficas de baixo nível e acesso aos mecanismos de som. Sua maior vantagem é de possuir a programação orientada a objetos e uma estrutura de grafo de cena para visualização dos objetos virtuais.

4.5 TDx9 DirectX Component

Este é um componente de software baseado no processo Microsoft COM [7], o qual encapsula de forma visual o uso do DirectX. É um componente disponível que pode ser agregado a diversas ferramentas RAD como o C++ Builder, Delphi, Visual Basic entre outros que aceite o registro de ActiveX [6].

4.6 Coin3D Library

Este é um conjunto de bibliotecas independentes que podem trabalhar em conjunto ou não para a visualização de cenas e objetos 3D [13]. É baseado em OpenGL e possui interface aplicada de classes em C++, possui um plugin para visualização do ambiente virtual que pode ser ativado por qualquer ambiente de desenvolvimento de software.

5. Análises e Discussão sobre os componentes apresentados

Para uma análise qualitativa das tecnologias apresentados para o desenvolvimento rápido de ambiente de realidade virtual os seguintes requisitos foram analisados: a) Realismo da Cena; b) Comportamento e Interação; c) Suporte a Navegação no Ambiente; d)Número de Linhas de Código por Tempo de Desenvolvimento e; e) Composição do Componente.

Neste contexto tem-se quanto ao realismo de cenas, os componentes baseado em OpenGL possui um grau maior em comparação com os baseados em DirectX.

Quanto ao Comportamento e Interação destaca-se o VRJuggler e o OpenSceneGraph por terem uma abstração maior de dispositivos, apesar do tempo de desenvolvimento e o número de linhas de código a serem implementados para tal é muito maior que o encontrado nos componentes TDx9 e no GLScene que são componente reutilizáveis.

No desempenho dos componentes TDx9 em comparação com o GLScene, o tempo de renderização do GLScene foi menor do que o do TDx9, visto que o último, como dito anteriormente possui um interface com o DirectX, mas quando é usado componentes de áudio o TDx9 obteve maior desempenho.

Quanto à composição dos componentes observa-se que tanto para o VRJuggler e o Open SceneGraph são do tipo caixa branca e que podem ser adicionadas funcionalidades sem alterar o código fonte do núcleo principal das bibliotecas. E para o WorldUp e Coin3D estes não dispõe da mesma característica sendo o código fechado tipo caixa preta. Para o componente TDx9 é possível a alteração de suas funcionalidades, entretanto com um custo de aquisição do código-fonte.

Notavelmente para o GLScene não é necessária a aquisição do código, pois o mesmo é de licença MPL. A utilização dos princípios propostos neste trabalho tornou-se necessária, em vista do crescimento geométrico do hiperespaço e da falta de flexibilidade na organização da estrutura de arquiteturas, caracterizada pela sua apresentação fixa, com critérios adotados pelo projetista do sistema de forma extremamente centralizada.

6. Referências

[1] GAMMA E., HELM R., JOHNSON R. and VLISSIDES J.: “Design Patterns : Elements of Reusable Object- Oriented Software”. *Addison-Wesley*, Reading, 1995.

[2] RIEDER, R.; DUÍLIO, G. “Development of a micro world for the education of the Fundamental Mathematics, using OpenGL and Delphi”, In: *X Congreso Iberoamericano de Educación Superior en Computación - CIESC*, 2002.

[3] FACIT Visual Simulation, “3D Real-Time GUI Software Environment” 2007, disponível em http://www.facit.co.uk/worldup.htm.

[4] MICROSOFT, “Introduction to ActiveX Controls”, 2002, disponível em http://msdn.microsoft.com/library//components/activex/intro.asp.

[5] VRLOGIC, “VirtualHand SDK/Studio”, 2005, disponível em http://vrlogic.com/virtual_hand_sdk.htm

[6] MICROSOFT, “MS Speech API Developer´s Guide”, 2000, disponível em www.microsoft.com/speech/.

[7] _____, “COM: Component Object Model Technologies”, 1999, disponível em http://www.microsoft.com/com/default.mspx.

[8]VRJuggler. “Open Source Virtual Reality Tools”, 2003. Disponível em http://www.vrjuggler.org/.

[9] OSG, “Introduction to the OpenSceneGraph”, 2004, disponível em http://openscenegraph.sourceforge.net

[10] BARROS, L. M., GONZAGA, L., RAPOSO, A. B. “Open Scene Graph: conceitos básicos e aplicações em

realidade virtual", *Tutorial on IX Symposium on Virtual and Augmented Reality (SVR)*. 2007.

[11] SMITH, P.S.; e DUKE, D.J., "Binding Virtual Environments to Toolkit Capabilities" in *EUROGRAPHICS 2000*. *Vol 19, Nro. 3*.

[12] PRESSMAN, R. in "Software Engineering: A Practitioner's Approach", 8th Ed. 2006.

[13] COIN3D, "3D Graphics Development Tools", disponível em http://doc.coin3d.org/Coin/index.html, 2005.

[14] GUIMARÃES, Marcelo de Paiva; GNECCO, Bruno Barbieri; DAMAZIO, Rodrigo. *Realidade Virtual e Aumentada: Conceitos, Projeto e Aplicações*. Kirner & Siscoutto. cp. 6, pg 108 -128. 2007. Petrópolis – RJ.

[15] Opengl - The Industry's Foundation for High Performance Graphics from games to virtual reality, mobile phones to supercomputers. 2008, disponível em http://www.opengl.org/.

[16] OpenAL - Cross-Platform 3D Audio. 2008, disponível em http://www.openal.org/.

[17] OGRE3D. Open Graphics Rendering Engine 3D. www.ogre3d.org - Acessado em: 10/03/2007 Online: 2007. [18] Kato H., Billinghurst M., Blanding B., May R., *ARToolKit, Technical Report*, Hiroshima City University, 1999.

[18] MICROSOFT, Introducing DirectX 9.0, 2006, on-line http://msdn.microsoft.com/archive//directx/intro/dx9intro.asp.

[19] SELMAN D., JAVA 3D Programming, Manning, 2002.

A Proposal for a Child Literacy Development Game Using Augmented Reality

Hugo R. da Silva Filho, Luís G. R. Alves, Marizete S. Santos
Universidade Federal Rural de Pernambuco
{hugo.ead}, {gugadin}, {maraufrpe}@gmail.com

Abstract

This paper introduces a recently started research project that aims for the development of a series of games based on Augmented Reality that will serve as entertainment and also as an educational tool. The main focus of this document is on the first game of the series which is currently being developed. Some information on Child Development and correlated works are also presented as they serve as a base to this project.

1. Introduction

Computers have become part of the daily life of an increasing number of people all over the world in many ways: at work as a tool helping to get a variety of tasks done, for fun at home as an entertainment option or at school, helping with research and other activities. Another important observation is that computers are being introduced in children's life much earlier than it was a few years ago [9]. The constant evolution of Information Technology has forced many institutions and industries to update their processes and the way they deal with the surrounding world and markets, in order to keep themselves competitive. Education is no different. It needs to be frequently taken to a next level of sophistication since educators recognize the need of exploring new technologies potential in the search for new ways of teaching. The evolution of the teaching and learning process is necessary due to the new abilities needed in the newly formed information society. A motivation to the use of games and entertainment tools is the ludic approach of it. It is also known that children implicitly learn and develop a variety of abilities by simply playing around. Two distinct goals (both very important) can be achieved by this ludic approach: entertainment and traditional school education [9, 6].

This paper does not suggest a solution to illiteracy; it describes part of a recently started research on Child Development and Augmented Reality (AR) that involves an investigatory work to find ways of using AR and games as tools to aid the development of some abilities in children. In the case of the first game of the series the focus is on child literacy process and reading and writing skills.

2. Child development and games

A lot of research has been done on Child Development and there are many theories and studies that try to understand and describe it. One of the most accepted theories is the Theory of Cognitive Development by Jean Piaget.

According to Piaget's theory Child Development can be summarized in four main periods: the Sensorimotor Period (years 0-2), Preoperational Period (years 2-6), the Concrete Operational Period (years 6-12) and the Formal Operational Period (from 12 to adult age) [10].

Literacy development starts from the early ages, but it is mostly noticed and accelerated around years 5 and 7. This means the kid is walking between the Preoperational and the Concrete Operational periods, in which they appreciate symbolic plays and games with rules.

Symbolic plays are activities like story telling, daily routine reproduction or simulation of feelings or imitation.

Games with rules are based on a set of rules that must be applied to specific situations. The player's non-compliance is usually somehow punished. During this transition between stages of development children start developing the logical thought in opposition to the intuitive thought that rule their acts earlier in life. Games have the capacity of retaining the children's attention and can also create the sensation that learning is fun, providing to the player the opportunity to develop abilities like processing events, raise the understanding of contexts and situations and make logical inferences to solve problems [8].

3. Augmented Reality

Augmented Reality is the process of overlapping virtual

three-dimensional objects, generated by computer, with a real environment, through some technological device. It allows the user to see the real world around added of virtual objects to it as complementary devices. This causes the user to have the idea that both virtual and real content coexist in a determined environment [7, 4].

6. Correlated Work

There are some researches and works done in education and Augmented Reality but we have no records of its use in games related to literacy, reading or writing development. At least two important and motivating works in Augmented Reality shall be mentioned.

The first one is by Brett E. Shelton and uses AR to teach astronomy concepts of "earth-sun relationships" in an application that lets students manipulate the virtual planets while viewing their respective information [11].

The second AR work is one of the HITlabs' interface projects called Augmented Tangible Molecular Models. This project helps students to understand complex molecular structures and features a Magic Book with animations [1].

Regarding the reading and writing processes, the game Pegue as Letras, shown in Figure 1, has been an important reference.

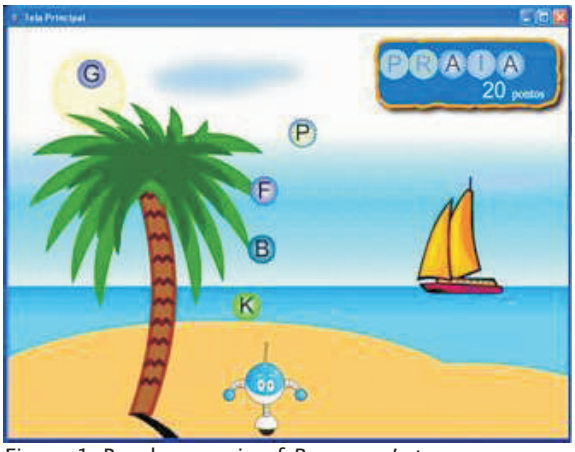


Figure 1. Beach scenario of *Pegue as Letras*.

Although Pegue as Letras employs the formation of words through letter catching, its focus is on younger children starting from the age of 3. All the player has to do is look at the grayed word, located in a box on the corner of the screen, and move the robot around to catch the letters that are part of it. There is an option that activates the easymode, in which the correct letters blink to indicate they are part of the current word. This is very useful to let younger children play even when they have not started recognizing the difference between capital letters [5].

5. Tools and Integration

This section describes the tools used in the research, the most relevant features of them and also exemplifies and demonstrates their integrated use.

5.1. OGRE

OGRE (Object-Oriented Graphics Rendering Engine) is a 3D engine written in C++. Its main goal is providing to developers the capability of creating applications with hardware accelerated graphics more easily. This is achieved by abstracting lowest level underlying technologies such as OpenGL and Direct3D [2].

Although people commonly misunderstand it, OGRE is not a game engine. It does not provide game related features such as collision detection, physics simulation and networking, among others. Because of this focus only on graphic's features, it needs be used along other libraries (like sound and physics libraries) in order to comeet the needs of game development.

OGRE was the chosen engine because of its design-led architecture that makes it useful in many different kinds of graphical applications, from simulations to games. Another strong feature that was important was the cost and the license. It is free of charge and runs under the GNU Lesser Public License (LGPL).

5.2. ARToolKit

ARToolKit is a software library that helps on the development of Augmented Reality applications by providing facilities to deal with 3D objects, multiple camera position and orientation, marker patterns tracking, rendering, and interaction [3].

ARToolKit provides the video capture facility that is necessary to do markers tracking from webcam data. The process of markers tracking demands some processing time, consequently, it decreases the frame rating. Nevertheless, using a small number of markers at a time can guarantee that the software will run with satisfactory frame rates even on lower configuration machines.

The performance of the library is directly related to the hardware used to run it; it varies according the type of video card, the amount of RAM memory and the capacity of the processor, for example. This had a great importance when it came to the decision on what to implement, considering that many Brazilian schools have outdated equipment. In this sense, the team decided to use a short number of markers (around

three at the most), and simple light weight 3D models.

5.3. Integrating OGRE and ARToolKit

The use of ARToolKit and OGRE is complementary. The lack of common entities or interfaces to rule the relationship between these technologies does not generate the need of any special implementation or significative coding effort. The ARToolKit library has been used to do video capture and to track the makers while OGRE has been used to place the 3D models in the scene (e.g. Figure 2).

```
if ((gARTImage = arVideoGetImage()) != NULL)
{
Ogre::TexturePtrvT = NULL;
vT = TextureManager::getSingleton().getByName("VideoTexture");
HardwarePixelBuffeSharedPtr mDynBufferPtr = NULL;
mDynBufferPtr= vidTextPtr->getBuffer(0.0);
PixelBox PixelBox(320,240,1,PF_X8R8G8B8,gARTImage);
mDynTexBufferPtr->blitFromMemory(pPixelBox);
}
arVideoCapNext();
```

Figure 2. Sample code that uses OGRE and ARToolKit.

Concerning the positioning operations applied to the 3D models there has been no need of changes or calculations for the coordinates provided by ARToolKit to the OGRE objects because they both use the same right handed coordinate system.

6. Game Description

This section briefly describes the game's user interface and its controls, as well as the complementary Word List Editor software.

The goal to be achieved by the player is: given a random sequence of letters, the player must put them together to form existing words. To meet this goal the player must decide which letters are going to be used and which ones shall be discarded.

6.1. User Interface

The real world image captured by the video camera will be overlapped with the following interface elements: a letter throwing mechanism, positioned at the center and "deep" into the screen; a notebook 3D model, positioned at a marker; a trash box, located at another marker; a scoreboard, positioned on the top-right corner of the screen; and the virtual letters objects that are thrown to the player.

6.2. Word List Editor

The Word List Editor is a tool used to update the list of words in the game's dictionary. The database basically has two entities: Categories and words. Categories represent the idea of a particular vocabulary such as words related to a beach or to a forest environment. The list of words is stored in a Database but there are plans to implement export and import functionalities to support the data exchange through XML files. This feature will allow educators to trade vocabularies and consequently adapt and improve their dictionaries.

6.3. Playing the Game

There are basically two actions in the game: picking up the current thrown letter to use it in a word or discarding it. These actions are represented by a notebook model and a trash box model and each one of them is located over one marker.

The game uses two markers: one to locate the trash box model (that holds the discarded letters) and the other one to locate the player's notebook (that holds the letters chosen by the player). The markers must be placed side to side, within a distance that allows the player to comfortably place one hand on top each one of them. When the game starts the player must have both markers hidden underneath his hands. The throwing mechanism will start throwing the letters toward the center of the screen, in a parabola route going from deep inside the monitor towards the player.

If the player wants to discard the letter, the hand over the trash box must be raised, when this happens the trash box model will appear and the letter will change its route and go into the trash. This letter won't be thrown again, except in another match.

If the player wants to use the letter, the hand over the notebook marker must be raised, when this happens the notebook model will appear and the letter will change its route and go into it. This letter will be thrown again so the player can use it more than once.

When the player takes neither action, the letter goes back to the throwing mechanism, and sooner or later will be thrown again.

7. Conclusions

The use of Augmented Reality for educational purposes is a recent tendency. Nevertheless, some research products are already being used in fields like chemistry and biology in high school classes. Augmented Reality software can interact with users in interesting ways and this can be explored to bring to the game a better gameplay (experiences and sensations of the players during the game).

Concerning the tools and libraries that are being used, both OGRE and ARToolKit have been responding well to all expectations and fulfilling the requirements

of the application development. OGRE and ARToolKit have proved to work well together, although they have no common entities or correlated interfaces.

The proposed game is being developed on Criatronics' labs since January 2008. A beta version is expected to be released later on June 2008 and, after all needed corrections and improvements, a variety of data will be collected through controlled experiments to gather information from children and teachers.

Finally, the expectation is to acquire enough information from participating children and teachers to evaluate and verify whether they like playing the game, if they like the AR experience in this kind of game and other questions that serve as guideline to the project.

8. Acknowledgements

The authors wish to acknowledge the Secretaria de Educação à Distância (SEED)/MEC for the financial support that made it possible for us to work on this research. We would also like to thank the Núcleo de Educação à Distância of UFRPE for the complementary support.

9. References

[1] HITLab Projects Homepage. Available: http://www.hitl.washington.edu.

[2] (2008) OGRE 3D: Open source graphics engine. Available: http://www.ogre3d.org/.

[3] ARToolKit Home Page on HITLab (2008) [Online]. Available: http://www.hitl.washington.edu/artoolkit/.

[4] R. T. Azuma. A survey on augmented reality. Available: http://www.cs.unc.edu/azuma/ARpresence.pdf.

[5] H. da Silva Filho. Jogos educativos de computador na educação infantil. Available: http://www.newhorizons.org/strategies/technology.

[6] J. Lopes. A introdução da informática no ambiente escolar. Available: http://www.clubedoprofessor.com.br/artigos.

[7] P. Milgram. Augmented reality: A class of displays on the reality-virtuality continuum. *Telemanipulator and Telepresence Technologies*, 2351(7), January 1994.

[8] P. Moratori. Por que utilizar jogos educativos no processo de ensino aprendizagem? Available: http://www.nce.ufrj.br/ginape/publicacoes.

[9] V. B. Oliveira. *O Brincar e a crianc¸a do nascimento aos seis anos*. Editora Vozes, Petr´opolis, RJ, 2000.

[10] J. Piaget. *A Psicologia da Criança*. Bertrand Brasil, Rio de Janeiro, RJ, 1998.

[11] B. E. Shelton. Augmented reality and education: Current projects and the potential for classroom learning. Available: http://www.newhorizons.org/strategies/technology.

A Mixed Reality-Based Exercise Program for Upper-Limb Post-Stroke Rehabilitation

Gilda Aparecida de Assis,
Ana Grasielle Dionísio Correa,
Universidade de São Paulo
{gildaaa}, {anagrasi}@lsi.usp.br

Cicero José Nunes Vaz
Hospital Israelita Albert Einstein
cicerov@usp.br

Roseli de Deus Lopes
Universidade de São Paulo
roseli@lsi.usp.br

Abstract

This paper describes a proposal of exercises for post-stroke rehabilitation of the upper-limb movements using mixed reality combined with mental practice. Physiotherapists induce patients' mental practices so that they execute the assigned tasks in the mixed reality environment. In the mixed reality environment, patients can visualize themselves and their surroundings, just like in a mirror. However, a virtual upper-limb is superimposed over their real paralyzed upper-limb. The set of proposed exercises include low complexity movements as shoulder infection, shoulder abduction, shoulder adduction, fist extension, and hand grasping as well as high complexity movements like reach and grasp a ball, drag it to the desired position and release it. We be lieve that the proposed system can be used to verify the effectiveness of visual stimuli for post-stroke rehabili tation. The system will allow post-stroke patients and their caregivers to perform the rehabilitation exercises at home. Also, this system will provide physiotherapy students the possibility to repeat medical training.

1. Introduction

In Brazil, about 167,199 people 40 years old and up, had a stroke during 2005 [1].

Each year, in the United States, approximately 700,000 people sustain a stroke. Stroke eects are a leading cause of physical disability. Because of this, there is a great variety of therapeutic interventions to improve motor recovery following strokes. At this time, existing physical and occupational therapy interventions are the foundation of post-stroke treatments [6].

Upper-limb hemiparesis is one of the most debilitating effects of stroke, and it is the primary impairment underlying functional disability following stroke [9].

During the subacute phase, at most one year post-stroke, patients learn or relearn competencies necessary to perform daily life activities. Frequent practice of skills enhances motor learning and skill acquisition.

However, most of the rehabilitation treatments are stopped one year after post-stroke, during the chronic phase [9].

Several researchers have shown in both animal and human studies that important variables in re-learning motor skills and in changing the underlying neural architecture are the quantity, duration and intensity of training sessions. Focal ischemic lesions in monkeys, similar to what occurs in a stroke, usually result in a loss of cortical territory in the motor area adjacent to the infarcted region. However, three-four weeks of intensive, repetitive, hand training prevented this loss and in some instances led to an expansion of this cortical region [4].

Mental practice, also known as "imagery", is a rehabilitation tool that can stimulate motor image [7].

In this paper, we propose a set of exercises to be used in a mixed reality environment that promote the recovery of upper-limb motor skills in chronic stroke patients, who had a stroke more than one year before, and who exhibit upper-limb hemiparesis. The exercises are intended to provide chronic stroke patients the visuospatial information to induce a third person perspective motor image.

The rest of the paper is organized as follows: Section 2 briefly presents related work. Section 3 reviews Mental Practice. The design and results of the experiments are then discussed in Section 4. Preliminary conclusions and future works are summarized in Section 5.

2. Related work

A related work was presented in [6] and [4]. The researchers provided a virtual reality training of the hemiparetic hand of post-stroke patients. The paper [4] presents the virtual reality-based exercise program and preliminary studies performed on users with no hand deficits and with a user who had suffered a stroke but had nearly normal hand function. On the other hand,

the paper [6] presents a study performed on eight subjects, 6 males and 2 females, with hemiparesis resulting from a stroke, using the virtual reality training system proposed in [4]. The conclusion of this work was that computerized exercise systems may be a way to maximize both the patients' and the clinicians' time because they can provide the intensity of practice that appears to be needed to effect neural reorganization and functional post-stroke changes. The system used two instrumented gloves. The subjects sat in front of the computer screen and interacted with the virtual world wearing one of the two instrumented gloves. Four virtual reality exercise programs were developed. Each exercise took the form of a simple game where the patient performed a number of trials of a particular task. If a certain performance target was reached, then the patient achieved the goal. These target-based exercises required an initial test to evaluate three parameters of range, speed and fractionation of the patients' movements.

In [2] was examined the potential of using computer-enhanced mental practice in the rehabilitation of upper extremity function after cerebrovascular accident. The treatment protocol consisted of one daily session, 3 days a week, for 8 consecutive weeks. Patients used the visualization prototype, which consists of the following components: a retro-projected horizontal screen incorporated in a wooden table; an LCD projector with parallax correction; a mirror that reflects the projector beam onto the horizontal screen; two movement tracker sensors; and a personal computer equipped with a graphics accelerator. The training procedure consisted of the following steps. First, therapist shows the patient how to perform the movement with the unaffected arm. When the patient performs the task, the system registers the movement and generates its mirrored 3-D simulation. Next, the patient is asked to mentally rehearse the same movement using the paretic limb. System shows the 3-D simulation superimposed over the (unseen) parectic limb, so that the patient can observe a model of the movement to be imagined.

3. Mental Practice

Mental practice, also called symbolic rehearsal or motor rehearsal, is a training technique in which the patients repeatedly "`rehearse"' a motor act in working memory, without producing any overt motor output [3]. Mental practice when combined with physical practice accelerates motor learning and improves subsequent physical performance. Because of its positive effects, mental practice has also been suggested to be a viable tool for improving motor learning and performance in rehabilitation settings [9]. Motor image may be defined as a dynamic state during which representations of a given motor act are internally rehearsed in working memory without any motor output. This type of phenomenal experience imply that the subject feels himself performing a given action. Studies indicate that motor image belongs to the same category of processes as those which are involved in programming and preparing actual actions, with the difference that in the latter case, execution would be blocked at some level of the cortico-spinal flow [7].

A mental practice protocol to increase the function and use of the more affected limb in chronic stroke patients is described in [8]. These mental practice interventions were provided on audiotape, recorded by a psychologist. The tape consisted of relaxation in the opening 5 minutes, followed by suggestions for internal, cognitive polysensory images related to using the affected arm in daily living activities. After the interventions, mental practice subjects showed increase in use and quality of the movement of the more affected arm for daily life activities, as observed by themselves and their caregivers.

Motor image may be experienced in two ways. The first person perspective which is supposed to rely on motor-kinesthetic information processing, and the third person perspective which would rely more on visuospatial processing [7].

4. Exercises Overview

We designed a set of six exercises to be performed by the virtual arm. This approach was suggested by a physiotherapist.

The finite state automata, shown in Figure 1, describes the six exercises available in our system. S1 indicates the initial state where the animation has not started yet. There are five low complexity exercises and one high complexity exercise. Low complexity exercises are S2, S3, S4, S5 and S6. All of them are final states. State S2 is the shoulder abduction, S3 is grasping, state S4 indicates shoulder exion, S5 defines fingers extension and S6 designates shoulder adduction.

A high complexity exercise of reach-grasp-release is available. If the target is put in front of the patient, the reach-grasp-release exercise consists in the following sequence of states: S1-S4-S3-S5-S6. Otherwise, if the target is put beside the patient's aected arm, the sequence is S1-S2-S3-S5-S6. The automata set of input symbols consists of the type of exercise to be performed and the target's relative position to the patient's plegic arm. Incoming arcs activate S2, S3, S4,

S5 and S6 when an intervention occured and sleep time reached 300 ms.

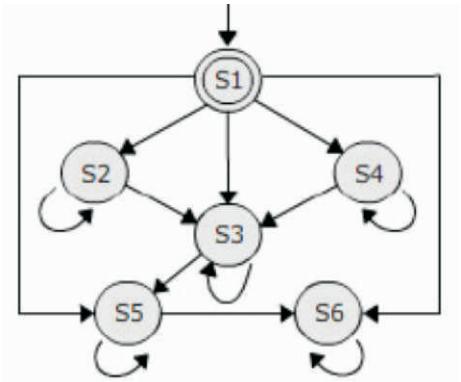


Figure 1. Finite state automata of the proposed system.

As cited in [5] main upper-limb movements for rehabilitation include shoulder abduction, shoulder adduction, shoulder flexion, fingers' extension and hand grasping.

Shoulder abduction is a movement which draws a limb away from the sagittal plane of the body. It ranges from 0 -180 degrees.

Shoulder adduction moves the arm in front of the body, with backhand facing down. It ranges from 0 to 40 degrees.

The shoulder flexion is gained moving the arm in the forward direction, putting the backhand parallel to the body's sagittal plane. This movement ranges from 0 to 180 degrees.

The fingers' extension opens simultaneouly all the fingers. Metacarpophalangeal joints bend from 0 to 90 degrees.

The four movements, which we selected for our simulation, are presented in the Figure 2. Figure 2(a) shows shoulder abduction, Figure 2(b) presents shoulder adduction, Figure 2(c) is an example of fingers' extension and Figure 2 (d) illustrates shoulder flexion movement. The grasping movement closes simultaneouly all the fingers. Interphalangeal joints range from 0 to 100 degrees.

5. Results

To date, we have implemented all proximal low complexity exercises: shoulder adduction, shoulder abduction and shoulder flexion.

We chose Cal3D format for our 3D model of the human arm. Cal3D format stores meshes and skeleton (bones and joints). We attached a marker on the patient's shoulder joint plegic arm. We built our system

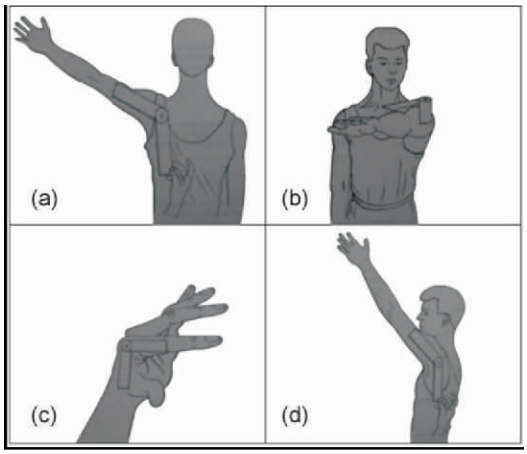


Figure 2. Low Complexity Exercises

using the computer vision ARToolkit library to track the marker and OpenGl library for graphics.

Figure 3, Figure 4 and Figure 5 present different angles' screenshots of the our simulation.

Figure 3. Shoulder Adduction Exercise Simulation, which ranges from 0 to 40 degrees.

Figure 4. Shoulder Abduction Exercise Simulation. Angles are 0, 60 and 120 degrees.

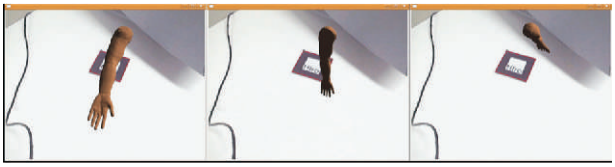


Figure 5. Shoulder Flexion Exercise Simulation. Angles are 0, 90 and 130 degrees.

The method effectiveness will be evaluated using two treatment protocols during a time period. The subjects will be randomly assigned in two groups: control interventation group and virtual interventation group. Patients in the control group will receive the same motor and mental practice therapy as those in the NeuroR group. We will apply pretreatment and posttreatment measures using Action Research Arm Test (ARA) and Jebsen Test of Hand Function in both groups. The Action Research Arm Test is an outcome

measure designed specically for use with patients with stroke. Jebsen Test of Hand Function is a timed test developed to assess hand function and nger dexterity in both the dominant and nondominant hands.

6. Preliminary Conclusions and Future Work

We believe that our system will provide a method to investigate the eciency of a mental practice protocol combined with visual feedback in motor rehabilitation of chronic stroke patients. Therefore, the system could be used to promote upper-limb rehabilitation for additional treatment past the traditional period of patient hospitalization and rehabilitation.

Also, this system will allow the rehabilitation stations to be set up in a patient's home or locations other than the rehabilitation center. The simulation of the exercises will help post-stroke patients and their care givers to perform the rehabilitation exercises in different settings and they will visualize the simulation for the training of motor tasks before performing it.

In the same way, our system will allow to physiotherapist' students learn about how to perform the main upper-limb motor rehabilitation exercises.

Currently we are implementing the algorithm to remove the real paralyzed upper-limb from the image and blend the resulting real image with the virtual arm. Also, a physical therapist is going to evaluate the system.

7. References

[1] M. da Sade. Datasus, 2007. Also available as http://tabnet.datasus.gov.br/tabdata/pacto2006/BR/Brasil_Pacto2006_Series_BR.xls; acessed on Oct/2007.

[2] A. Gaggioli, A. Meneghini, I. Pozzato, G. Greggio, F. Morganti, and G. Riva. Computer-enhanced mental practice in upper-limb rehabilitation after cerebrovascular accident: a case series study. In *6th International Workshop on Virtual Rehabilitation*, pages 151–154. IEEE, 2007.

[3] A. Gaggioli, F. Morganti, R. Walker, A. Meneghini, M. Alcaniz, J. A. Lozano, J. Montesa, J. A. Gil, and G. Riva. Training with computer-supported motor imagery in post-stroke. *CyberPsychology Behavior*, 7(3):327{332, 2004.

[4] D. Jack, R. Boian, A. Merians, S. V. Adamovich, M. Tremaine, M. Recce, G. C. Burdea, and H. Poizner. A virtual reality-based exercise program for stroke rehabilitation. In *ASSETS'00*, pages 56-63. ACM, ACM, November 2000.

[5] A. P. Marques. Manual de Goniometria. 1997.

[6] A. S. Merians, H. Poizner, R. Boian, G. Burdea, and S. Adamovich. Sensorimotor training in a virtual reality environment: Does it improve functional recovery poststroke? *Neurorehabilitation and Neural Repair*, 20(2): 252-267, 2006.

[7] S. J. Page, P. Levine, and A. C. Leonard. The neurophysiological basis of motor imagery. *Behavioural Brain Research*, 77: 45-52, 1996.

[8] S. J. Page, P. Levine, and A. C. Leonard. Effects of mental practice on aected limb use and function in chronic stroke. *American Congress of Rehabilitation Medicine and the American Academy of Physical Medicine and Rehabilitation*, 86: 399-402, march 2005.

[9] S. J. Page, P. Levine, S. A. Sisto, and M. V. Johnston. Mental practice combined with physical practice for upper-limb motor deficit in subacute stroke. *Physical Therapy*, 81(8): 1455-1462, august 2001.

Visualização Tridimensional de Baixo Custo para o Desenvolvimento de Aplicações em Medicina

Alysson Diniz dos Santos, Liliane dos Santos Machado
LabTEVE - Universidade Federal da Paraçba, João Pessoa/PB, Brasil
alyssondiniz@yahoo.com.br, liliane@di.ufpb.br

Abstract

This paper deals with the uses of stereopsis based on anaglyph method for applied visualization in Virtual Reality environments. Thanks to the financial resources and high computing performance demanded by some 3D visualization methods, we have tried to build and integrate a class to a simulation system development package that allows the colored anaglyph utilization. This way, the library that contains the paper's class, permits the development of simulation applications, that can be executed in popular computational platforms, with colored tridimensional visualization of the scenes. This paper still presents the implementations details and the relations between the achieved results.

Resumo

Sabe-se que o uso da estereoscopia traz benefícios ao processo de visualização de dados tridimensionais, recurso largamente utilizado em sistemas baseados em Realidade Virtual. Devido aos recursos financeiros e desempenho computacional exigidos por alguns métodos de visualização tridimensional, procurou-se conceber e integrar a um pacote de desenvolvimento de sistemas de simulação e treinamento médico uma classe que permitisse o uso do método de anaglifos coloridos. Dessa forma, o conjunto de bibliotecas ao qual esta classe foi integrada permite desenvolver aplicações de simulação que podem ser executadas em plataformas computacionais populares com visualização tridimensional colorida das imagens. Este artigo também apresenta aspectos de implementação da classe e a análise comparativa dos resultados obtidos.

1. Introdução

Devido à grande demanda de funcionalidades exigidas na elaboração dos sistemas de Realidade Virtual(RV), a utilização de um conjunto de bibliotecas que forneça funcionalidades que possam viabilizar a criação de ambientes de RV facilita e torna mais eficiente o processo de criação de um ambiente de RV [1, 2].

O CyberMed foi concebido como um conjunto de bibliotecas voltadas para o desenvolvimento de aplicações de simulação baseadas em RV para a área médica [3]. Para tanto, o sistema CyberMed oferece ao seu usuário funcionalidades que viabilizam a criação e manipulação de um ambiente virtual adequado para propiciar ensino e treinamento. Dentre essas funcionalidades estão inclusas a interação háptica, a detecção de colisão, a deformação dos modelos tridimensionais, a avaliação do treinamento do usuário e a visualização tridimensional das cenas gráficas [4, 5].

Sendo a visualização, por muitas vezes, o primeiro contato que o usuário tem com o ambiente de RV, a mesma assume um papel importante para que o sistema consiga obter o resultado desejado. Nesse contexto, o sistema CyberMed pode gerar uma cena gráfica que, ao ser observada em conjunto com óculos adequados, oferecerá a sensação de profundidade ao usuário que observa a cena. Esse efeito é conseguido com a utilização de estereoscopia. A estereoscopia pode ser entendida como a fusão, feita no cérebro, de duas imagens bidimensionais resultantes da projeção planar de uma cena tridimensional [5].

Um dos principais problemas na implantação de estereoscopia em aplicações de RV é o custo financeiro atrelado aos dispositivos de visualização do par estéreo. A técnica que utiliza óculos polarizados exige o uso de dois projetores e filtros para polarizar a luz de forma adequada. Por sua vez, a técnica de luz intermitente demanda um projetor ou monitor de alta freqüência, uma placa de vídeo específica e óculos obturadores, cujos custos podem tornar pouco viável seu uso em aplicações nas quais se deseja baixo custo financeiro. Neste sentido, apesar da existência de HMDs de preços acessíveis, estes são individuais e não permitem a visualização em grupos de trabalho. Por esses fatores, uma solução viável para visualização estereoscópica que pudesse ser utilizada em grupos e em plataformas

populares é a técnica de anaglifo. Nesta técnica não são exigidos projetores ou equipamentos especiais, sendo necessário apenas o uso de óculos com filtros coloridos que podem ser confeccionados pelo próprio usuário [6].

O objetivo deste trabalho foi criar uma classe para implementar técnicas que proporcionem ao desenvolvedor de aplicações com o CyberMed a possibilidade de oferecer imersão através da utilização de cenas estereoscópicas, sem a necessidade de utilizar recursos computacionais específicos. O baixo custo computacional e financeiro foi o principal motivo para a escolha do método de visualização estereoscópica por anaglifos.

2. Técnicas de Estereoscopia

Para a observação estereoscópica de uma cena deve-se primeiro escolher o método de geração do par estéreo. Nesta etapa, uma imagem da cena precisa ser desenhada duas vezes com uma pequena alteração na posição horizontal do observador. Essa diferença, que gera duas imagens semelhantes às geradas pelos olhos humanos, faz com que um único ponto da cena original seja colocado em dois pontos diferentes no par de imagens formado. A diferença na posição desse ponto para as duas imagens, denominada paralaxe, é o que nos vai dar a sensação de profundidade para esse ponto. No entanto, caso a paralaxe seja calculada de forma errada, o observador perderá a visão tridimensional para aquele ponto [7].

As classes de visualização implementadas no sistema CyberMed utiliza o método *off-axis* para a geração do par estéreo. Esse método consiste na criação de dois centros de visualização através de deslocamentos do centro de projeção original ao longo do eixo horizontal. Em comparação com os outros métodos existentes - o *on-axis* (imagens geradas por translações horizontais do objeto), ou o das rotações (imagens geradas por rotações do objeto em torno do próprio eixo) – o método *off-axis* é o que gera o par estéreo de forma mais correta, minimizando as perdas de informação e as disparidades que podem ser geradas pela paralaxe [8].

2.1 Método Anaglifo

O método de anaglifo utiliza a técnica de filtragem por cores. Para tanto, ele utiliza apenas as componentes de determinadas cores para as imagens da direita e da esquerda. A captação de certas componentes coloridas é possível com a aplicação de máscaras de cores adequadas na cena que está sendo tratada. Após fundidas as duas imagens do par estéreo a filtragem adequada da cena fica a cargo dos óculos de filtros coloridos [5].

Observa-se que, de acordo com a máscara de cores aplicada nas imagens, o resultado visual da cena final poderá apresentar diferentes características. O primeiro método de anaglifo implementado no CyberMed aplicou uma máscara vermelha na imagem da direita e uma máscara azul na imagem da esquerda. É nesse método, denominado anaglifo verdadeiro, que são observados os melhores resultados no tocante à sensação de profundidade em cenas com visualização estereoscópica geradas por anaglifos. Contudo, observando-se o fato de que apenas as componentes de cor azul e vermelha estão disponíveis, a cena gerada tende a ser exibida em um tom magenta e é visualizada em uma escala monocromática [9].

2.2 Anaglifo Colorido

O método de anaglifo colorido utiliza-se da aplicação da máscara vermelha à imagem da esquerda e da máscara ciano (balanceamento proporcional entre azul e verde) à imagem da direita do par estéreo. Com a camada de cor verde sendo utilizada, a imagem resultante da cena permite mais combinações de cores na fusão do par estéreo. Este fato permite que a coloração da cena tridimensional aproxime-se da coloração original da mesma [9].

Por se tratar de uma aplicação com foco na simulação de ambientes reais, torna-se de vital importância que o usuário possa observar no modelo estereoscópico cores próximas às do modelo real. Tal fato facilitará ao usuário reconhecer e distinguir partes diferentes apenas pela observação da aplicação. Neste ponto a CybViewColorAnaglyph torna o sistema mais eficiente, pois além de oferecer uma visualização de melhor qualidade, torna a visualização acessível, dado o baixo custo do método e dos óculos utilizados, proporcionando assim maiores níveis de imersão e aprendizado.

3. Implementação

No sistema CyberMed a classe Cyb3DWorld foi concebida como a classe geral da visualização do sistema. Ela contém implementações para as funções básicas de um sistema de RV, tais como a criação de janelas gráficas, a carga e exibição de modelos tridimensionais e o tratamento das operações sobre a janela, tais quais redimensionamento, interação por mouse, teclado e iluminação, deixando apenas a exibição da cena a cargo das subclasses da visualização do sistema. Neste ponto, é importante ressaltar que, caso seja necessária a inclusão ou a remodelagem de qualquer funcionalidade da Cyb3DWorld, é necessária

apenas a criação ou redeclaração da nova funcionalidade na classe que está sendo criada pelo usuário, dessa forma o programador tem liberdade para utilizar as implementações padrão ou utilizar versões próprias.

De acordo com o padrão estabelecido pelo sistema CyberMed, a classe para uso de anaglifos coloridos desenvolvida, denominada CybViewColorAnaglyph, herda métodos da Cyb3DWorld, podendo acessar todos os procedimentos previstos pela classe mãe para a criação, exibição e tratamento de uma cena gráfica. Dessa forma, a classe CybViewColorAnaglyph implementa apenas o método de visualização que cria e exibe simultaneamente o par de imagens estéreo do método de anaglifo colorido (Figura 1).

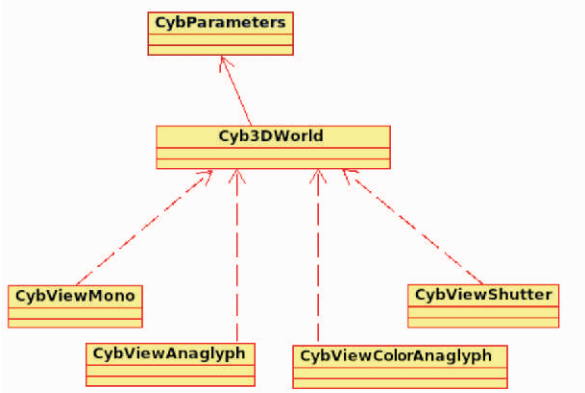


Figura 1. Diagrama de classes de visualização do sistema CyberMed com a nova classe de visualização por anaglifos coloridos.

Para a exibição da cena gráfica, o método *display* da CybViewColorAnaglyph trata do posicionamento das câmeras de forma a obter o par de imagens estéreo, da definição da projeção – que, no caso, é uma projeção em perspectiva, definida pela função glFrustum da OpenGL – e do desenho das malhas, seja para as malhas dos objetos da cena ou dos objetos interadores.

O acesso à cor dos objetos gráficos da cena é fornecido através da CybParameters, – classe que armazena todos os parâmetros do sistema – é através dela que a CybViewColorAnaglyph acessa o parâmetro bidimensional denominado *color* que armazena a cor de cada objeto gráfico criado na cena.

Um laço é criado e percorre cada camada da cena, capturando a cor de cada objeto e redesenhando de acordo com as posições das câmeras e dos centros de visualização as duas cópias do objeto que formarão a imagem em anaglifo. É importante lembrar que, de acordo com a máscara definida antes da criação de cada imagem, uma delas só fica com as componentes vermelhas da cor e a outra com as componentes azul e verde.

A utilização da classe CybViewColorAnaglyph por usuários do CyberMed se dá de forma semelhante às outras classes de visualização, nas quais é necessário que o usuário crie um objeto da classe de visualização desejada no corpo da sua classe principal e dessa forma estará ativando o modo de visualização desejado.

4. Resultados

O sistema CyberMed já possuía três classes de visualização integradas: a CybViewMono que exibe cenas monoscópicas, a CybViewShutter que exibe cenas estereoscópicas a serem visualizadas com óculos obturadores e a CybViewAnaglyph que exibe cenas com o método de anaglifo verdadeiro. Como pode-se observar no diagrama de classes de visualização do sistema (Figura 1) a classe CybViewColorAnaglyph foi posicionada ao lado das outras classes de visualização do sistema e abaixo da classe mãe da visualização, a Cyb3DWorld.

Para a execução dos testes foi utilizado um conjunto de cinco modelos de estruturas do coração, que representam as válvulas, veias e artérias cardíacas. Inicialmente foi gerada a visualização monoscópica, como pode ser observado na Figura 2. Posteriormente, foi utilizada a classe CybViewAnaglyph e a classe CybViewColorAnaglyph para gerar a visualização tridimensional dos mesmos modelos. A observação a olho nu dos resultados, já permite observar a perda da informação de cor com o método de anaglifo verdadeiro (Figura 3), ao passo que o método de anaglifo colorido possibilita uma razoável manutenção das cores da cena (Figura 4). Neste caso, a perda das cores torna evidente o comprometimento da compreensão de estruturas anatômicas dentro de uma aplicação na área médica com o método de anaglifo verdadeiro e ressalta a necessidade da preservação das cores na visualização.

Ressalta-se ainda que, uma vez que o método de anaglifo pode ser impresso, a utilização de óculos com filtros coloridos possibilita a observação tridimensional das Figuras 4 e 5 deste artigo.

5. Conclusão

Com a conclusão da implementação da classe de anaglifos coloridos, a mesma amplificou a sensação de imersão do usuário nas cenas geradas com o sistema CyberMed, atendendo aos requisitos de baixo custo dos equipamentos utilizados. Tal fato torna-se relevante quando se pretende que a aplicação final gerada pelo CyberMed possa ser utilizada por grupos de alunos e em instituições inseridas em diferentes realidades

em instituições inseridas em diferentes realidades sociais. Assim, a existência de uma classe que permita o uso de cores no processo de visualização estereoscópica por anaglifo, permite expandir as possibilidades de uso do CyberMed para aplicações em plataformas populares.

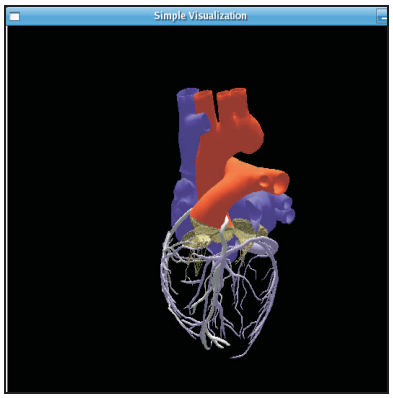


Figura 2. Visualização monoscópica gerada com o CyberMed.

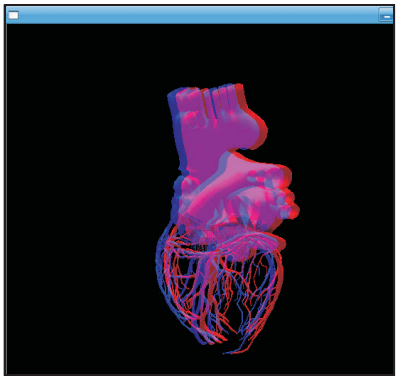


Figura 3. Visualização pelo método de anaglifos verdadeiros gerada com o CyberMed.

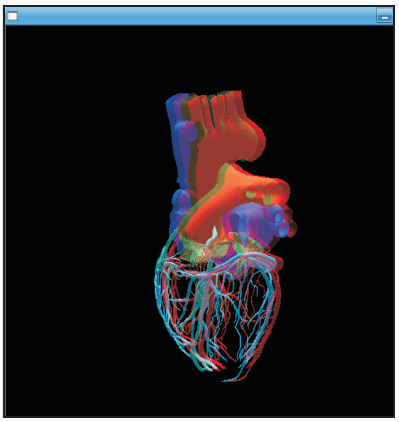


Figura 4. Visualização pelo método de anaglifos coloridos gerada com o CyberMed.

O sistema CyberMed foi totalmente desenvolvido com a linguagem de programação C++ e a API OpenGL. O sistema foi projetado para plataformas Linux e atualmente encontra-se concluído em sua primeira versão e disponível para download em www.sourceforge.net/projects/cybermed. Ressaltamos que a versão disponível para download ainda não possui a classe referida nesse artigo, pois a mesma faz parte do processo de atualizações do CyberMed para a próxima versão. Além disso há projetos em vigor que estão utilizando a biblioteca CyberMed para a criação de ambientes virtuais para o ensino de medicina.

6. Agradecimentos

Alysson Diniz dos Santos é bolsista CNPq. Este projeto é financiado pelo CNPq, através do processo CT-Info 506480/2004-6 e pela FINEP através do convênio 01-04-1054-000.

7. Referências

[1] A. Bowman, R.P. McMahan, "Virtual Reality: How Much Immersion is Enough?". *Computer Innovative Tech. for Computer Professionals*, v.40, n.7, pp. 36-43, Julho 2007.

[2] M. Siegel, S. Nagata. "Just Enough Reality: Comfortable 3-D Viewing via Microstereopsis". *IEEE Trans. Circuits And Systems For Video Technology*, v.10, n.3, pp. 387- 96, 2000.

[3] D.F.L. Souza, et al., "Development of a VR Simulator for Medical Training Using Free Tools: A Case Study". *Proc. .SVR'2007*, pp. 100-105, 2007.

[4] W.R. Sherman, A.B. Craig, "Understanding Virtual Reality : Interface, Application and Design", *Morgan Kaufmann*, 1ª Edição, San Francisco, 2003.

[5] L.S. Machado, R.M. Moraes, "Cenários 3D Interativos com Software Livre". *Revista de Informática Teórica e Aplicada, UFRGS*, vol. 12, no. 2, pp. 91-112, Outubro 2005.

[6] D.F. McAllister, "Stereo and 3-D Display Technologies". *Encyclopedia of Imaging Science and Technology, John Wiley & Sons*, pp. 1327-1344, 2002.

[7] G.C. Burdea e P. Coiffet, "Virtual Reality Technology", *Wiley-Interscience*, 2ª Edição, Nova Jersey, 2003.

[8] L.F. Hodges, "Tutorial: Time-Multiplexed Stereoscopic CG". *IEEE CG&A* vol. 12, no. 2, pp. 20-30, 1992.

[9] J.M. Zelle, C. Figura, "Simple, Low-Cost Stereographics: VR for Everyone" , ACM SIGCSE Bulletin, *Proc. 35th SIGCSE Technichal Symp. on Computer Science Education SIGCSE '04,* vol. 36, pp. 348-352, Março 2004.

Session 5: 3D User Interaction

AR X-Ray : Portable Projector-based Augmented Exploration of Buildings

Fábio R. de Miranda, Romero Tori, Cláudio E. S. Bueno, Lucas P. Trias
Centro Universitário Senac
LPAI - Laboratório de Pesquisa em Ambientes Interativos
Av. Eng. Eusébio Stevaux, 823 São Paulo – SP
{fabio.rmiranda}, {romero.tori}, {cesbueno}@sp.senac.br, lucas.trias@senacsp.edu.br

Abstract

This paper presents the design and implementation of AR X-Ray, an augmented reality tool that allows visual exploration of internal details of buildings (such as pipes and ducts) through the use of a portable projector that works as a "magic lantern" that projects a Virtual X-Ray over real walls. Results and details regarding solutions for technical problems are discussed. Some of the addressed issues are projector tracking, proper registration of synthetic information over real objects and the development of a shader that manipulates the transparency of textures to allow the user to view what is behind objects.

Keywords: *Spatial Augmented Reality, projector-camera systems, ,Interactive Technologies, tracking*

1. Introduction

Spatial Augmented Reality (SAR) [1] has great potential for applications that require augmentation of the real space without forcing the user to wear goggles or any other uncomfortable equipment. Among several ways of creating spatial augmentation, the use of projectors to paint "light textures" onto physical objects is one of the most promising. Works such as those of Raskar et. al [2] and Pinhanez [3] demonstrated the viability of projector-based augmented reality as well as some interesting ideas for applications. Although having some limitations, as the need of appropriate illumination, adequate reflectance property of the surfaces receiving the projections and user tracking if the projection represents 3D perspectives instead of simple 2d textures, this technique can still be used in a vast amount of innovative SAR applications. "AR X-Ray", an interactive projector-based spatial-augmented tool for visual exploration of architectonic structures, is one of these solutions. With "AR X-Ray" it is possible to virtually explore internal layers of real walls by moving a projector and controlling the depth of a virtual ray that projects internal layers of the structure directly over its surface, producing an effect that resembles an x-ray vision (Fig. 1). This project met some challenges related with registration, tracking, real-time texture manipulation and interface design. In this paper we show how those difficulties were solved and present the first complete functional prototype, its architecture, some initial tests results and next steps.

Figure 1. User holds a projector-camera system that delivers the augmentation with proper registration.

The main contributions of this work are: an innovative use of projector-based augmented reality to produce a virtual x-ray tool that frees the user of wearing special devices; the demonstration of a low-cost alternative for the 3D tracking of a mobile projector; the registration of 3D geometry over real objects in a mobile projector through a projective transformation; the development of a prototype as a proof of concept and the documentation of implementation aspects involved in doing it; a "testbed" for further experiments and studies on interface design of interactive spatial augmented applications.

This article is organized as follows: in section 2 we review relevant previous work related to this project, in section 3 the project is explained in detail to include some of its design goals and implementation techniques, in section 4 the results obtained are presented. The conclusions of this work are presented in section 5 and future work is detailed in section 6.

2. Related work

2.1 Projector-based augmented reality

The most important technology behind this project is known as spatial augmented reality, more specifically the one based on the use of projectors casting images directly over the surfaces of real objects. Bimber and Raskar [1] present some pioneer projects in this area. Among these Shader Lamps [2] and "Being There" [4] were responsible for inspiring us to develop AR X-ray. The idea of "Shader Lamps" is to project 2D textures onto the surfaces of real objects. These objects need to have preferably homogeneous white surfaces. The method developed is sufficiently general to address any kind of rigid 3D surfaces, using one or more casually aligned projectors. The authors of "Shader Lamps" also have developed techniques for tracking users and projectors through the use of magnetic trackers. "Being There" is another example of spatial augmented reality based on projectors. This project constructed a human-size environment composed of walls made of Styrofoam blocks. Textures of bricks, wood and even transparent windows are projected over these walls creating a convincing environment in which the user can walk-through. Since the viewer position is tracked it is possible to look outside or see the next room through virtual windows.

2.2 Handheld projectors

The imminent widespread commercial availability of mobile, lightweight projectors either in standalone form or integrated in other devices such as cell phones makes them a promising platform for mobile applications in entertainment, multimedia and support of collaborative work.

A pioneering work that showcased the possibilities of mobile projectors was iLamps [5], in which a user carries a projector-camera system that was made location-aware through the usage of fiducial markers in environmental objects that allowed the system to display contextual information and emulate mouse-like interactions by moving the projector in relation to the fiducials. In iLamps the registration between projectors and augmented objects was solved by homographies (plane-to-plane mappings between projector image coordinates and surfaces projected on).

Recently, the VisiCon project [6] demonstrated the use of handheld projectors in the context of a multiplayer game where players control a real robot on the floor that senses commands embedded in the projector light through photosensors on its back. VisiCon explores accelerometers to compute the inclination of the projector based on the direction of gravity and transform the projected image so that it remains rectangular as the user moves it. The work by Dao et al [7] recently demonstrated the use of tilt sensors and 3D digital compasses to allow the user to move the projector and still keep the projected image rectangular on flat surfaces. *Interaction* paradigms for mobile projectors have been explored in the work by Cao et. al [8] where a projector was tracked in 3D-space with the goal of revealing more content of a bigger virtual screen accordingly, a paradigm the authors call flashlight metaphor and that bears similarity to our work , the biggest difference is that in AR X-Ray registration matches a 3D model with its real counterpart, and not a virtual screen. The work of Blask et al [9] focused on interaction paradigms and used a technique similar to the flashlight where movement parallel to the projection screen was used to pan an image and movement perpendicular to it could zoom in and out. That work used a common projector and motion tracking devices at the wrists of users to simulate a wristwatch projector.

As real mobile handheld projectors are just beginning to hit the market with offers from manufactures such as Siemens, Mitsubishi and Toshiba, such devices are still basically miniatures of conventional meeting room projectors and lack many of the features that would be essential to support most of the interactive applications currently proposed by the research community. Thus it is common to do protopyping by augmenting the current projectors with cameras and extra sensors (such as motion and position trackers, for instance) and assume that future projectors will have those features.

2.3 Interactive Tools for Virtual X-Ray Vision

This project intended to develop a set of tools for giving users the impression of seeing through solid objects [10]. As stated by the authors of that project they "consider x-ray vision to be the act of visualizing some target through at least one layer of occluding structure". Their approach differs from ours as they use video see-through solution instead of spatial augmented reality. Based on a tracked user this project offers three main tools: Tunnel Tool, Room Selector Tool, Room in Miniature Tool and Room Slicer Tool.

The Tunnel Tool is a " Frustum extending from the user's position out along the direction of view", creating the effect of looking through a virtual tunnel that cuts and penetrates the objects and walls. This approach is similar to that adopted in our project. The Room Selector Tool allows the user to control a 3D cursor along the direction of view for selecting rooms inside the real environment. A selected room is shown from the user's viewpoint with walls that would occlude the selected room shown in wireframe, and the user can control which objects inside the selected room will be shown by grouping them in layers. The Room in Miniature Tool is a third person view of a selected room. Finally, the Room Slicer Tool works in conjunction with the Room in Miniature Tool, allowing a third person exploration sliced planes of the room.

2.4 X-Ray Window

This tool, presented in White et al [11] is a prototype developed at NASA's Johnson Space Center aimed at giving crewmembers of International Space Station (ISS) a kind of X-Ray vision. Wearing special augmented reality glasses the user can "see through" the walls of the module or through the station. The main problem in this project is to select, convert, and combine information from various databases.

2.5 Augmented reality in architectural applications

Presented in Webster et al. [12], this early work bears strong similarities with the purpose of AR X-Ray, though based on rather different implementation details. It demonstrated the visualization of structural elements registered over the real parts of buildings with a Head Mounted Device tracked by ultrasonic sensors. . It can be considered an initial attempt at X-Ray vision since structural details could be visualized in wireframe (such as beams inside walls or ceilings and walls behind furniture, for instance).

2.6 Toolglass and magic lenses

This early work [13] dealt with several issues related to see-through user interfaces in the context of traditional desktop applications. One the ideas proposed in this work was that of a Magic Lens – a kind of semitransparent widget manipulated by the user with his hand or a second mouse cursor - that was able to reveal contextual information and suppress some details while enhancing others related to objects that are seen through it.

2.7 Using AR to Visualise Architecture Designs in an Outdoor Environment

This work, presented in Thomas et al [14] is worth mentioning because it is an application of augmented reality to architecture. It presented ways of visualizing outdoor modifications or extensions to buildings through the use of a HMD. This is an early work and tracking was based on a digital compass and low-precision GPS that counted on the user proactively indicating his position or moving his head to compensate for registration errors.

3. The AR X-Ray project

3.1 General vision

The main goal of the AR X-Ray prototype, which is illustrated in Figure 1, is allowing the users move a handheld projector freely and visualize internal details of walls and buildings. The user should also be able to control through an interface the power of the simulated X-Ray vision, thus revealing inner details of walls or even seeing through them.

The medium-term objective of this project is to create an innovative spatial augmented reality infrastructure, based on projections over real walls, in order to allow the visualization of internal structures in a way that simulates an idealized portable X-ray tool. This infrastructure is intended for future use as a platform for testing new interaction design approaches in spatial augmented reality as well as experiments on new applications like education, art and entertainment.

The first problem faced by researchers working on augmenting reality tools aimed to give users some kind of "viewing-inside-and/or-through-objects" power is the "Superman's X-Ray Vision" problem. This problem is related with the paradox of showing too much information about occluded structures, and getting the user confused, against the option of showing only the targeted information, and thus depriving the user of important visual depth and orientation references [15]. In our "Virtual X-Ray" this problem was solved by taking the making the user manipulate a virtual "cutting tool" that simulated the penetration of a visualization "frustum" on the object, producing similar results as if the object was itself rendered in computer graphics and all the action was taking place in a computer graphics environment. Some design alternatives of interaction and visualization, as well as new applications of this technological infrastructure, will be studied as a second part of this project, whose focus will be on design issues.

A known problem with immersive augmented reality, like that provided by optical or video see-through approach, is the difficulty for users to have correct distance perception in immersive virtual environments

[16][17][18]. Other problems related with see-through displays are insufficient brightness, resolution and field of view, besides fatigue and strain [19]. Our X-ray approach avoids these problems by using spatial augmented reality and thus eliminating the need of goggles or other types of potentially cumbersome immersive apparatus.

Figure 2 – Projector-camera system used in AR X-Ray.

AR X-Ray is conceived to be used with mobile projector-camera system with some form of input interface is available. In a way similar to [5, 6, 7, 8], this work used a common projector with appended apparatus in order to simulate a mobile intelligent projector. The projector used in AR X-Ray can be seen in Figure 2, it is a Benq MP611 with a 800x600 resolution and 2400 ANSI lumens. It was incremented with a 640x480 webcam rigidly attached to it and a Nintendo Wii controller to be used as user input. Currently it needs to be tethered to a computer equipped with a GPU (*graphics processing unit*) through VGA and USB cables and also needs to be plugged to an AC power outlet, such tethers limit the range where the tool can be used but does not compromise the overall prototyping work.

This work follows the basic idea of Shader Lamps [2] that was based on maintaining in software an augmented 3D representation of the real scene and projecting in such a way that the rendered images of the virtual objects match their real counterparts. In this approach, in order to augment a real object one has to augment the corresponding object in the virtual scene and register the projection so that real and virtual objects match. This is done through the use of properly positioned ARTag [20] markers and one webcam attached to the projector, and will be discussed further.

3.2 Application architecture

3.2.1. Application modules. The AR X-Ray system can be seen in a general level as a system that converts the position of the projector and the desired zoom level into a rendering of the internal structure of objects that is superimposed on such real objects with the help of a projector. It was developed in Java and C and is comprised of modules specifically suited to tracking, interaction, calibration, registration, scene management (in Java3D) and composition of textures.

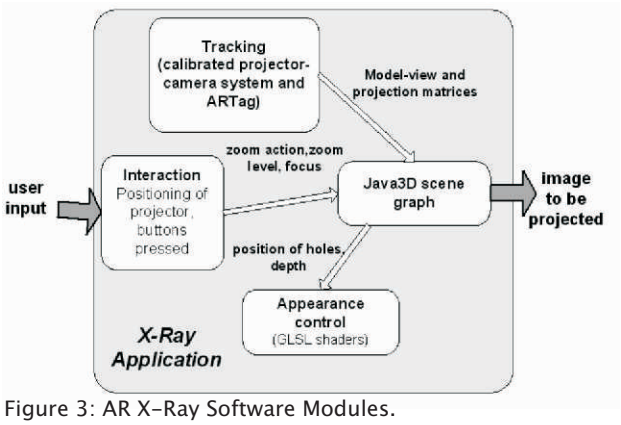


Figure 3: AR X-Ray Software Modules.

The *tracking* module is responsible for determining the position of the projector in space and is based on ARTag. This module is coded in C and is able to estimate the position of the projector in relation to a set of fiducial markers and sends such estimate to the main application (which runs the scene graph) through sockets.

The *Java 3D Scene Graph* module outputs model-view and projection matrices (as used in OpenGL) so that a virtual camera can be positioned in a way the resulting synthesized images get correctly registered. The choice of Java3D for this module was due to the previous experience of the group with it and the perspective of reusing animation and interaction code available in some of its libraries.

The interaction module involves allowing the user to control the radius and depth of the hole that is seen on surfaces. This is done through an Wii controller attached to the project whose buttons number one and two are mapped to mouse keys via the GlovePIE utility [12].

The module responsible for *appearance control* changes the transparency of 3D models through the use of a pixel shader, and the Java3D scene graph is the major integrator that depends on all other modules in order to produce the image to be projected, and will be further detailed in subsection 3.3.3.

3.3. Technical problems

3.3.1. Registration and projection calibration.

Registration is the most important problem that must be overcome when devising an application aimed at enhancing real objects with synthetic augmentations that must match their geometry well enough to give the users the perception that such augmentations are part of the real object. Our approach is based on the one that was outlined in [2] but is adapted to use a mobile projector, though the basic math is the same. The projector is seen as the dual of camera: instead of light rays coming from real objects through an optical center and inciding on a projection plane, the light rays are seen as being emitted by the projection plane and passing through an optical center before inciding on the real object.

From the technical standpoint, the main advantage of the projector operating as the dual of a camera is that the same well known techniques that can be used to estimate the parameters relevant to how an image forms in a camera can be used to represent the optics of the projector - for a thorough explanation of this image formation process, we recommend the work of Faugeras [22]. These parameters are represented in equations (I) and (II) – equation (I) shows that the points in image plane represented by $\mathbf{m}_i$ (in homogeneous coordinates) are the result of the transformation of the 3D points $\mathbf{M}_i$ by a perspective projection matrix.

$$\mathbf{m}_i = \mathbf{P} \cdot \mathbf{M}_i$$

Where $\mathbf{m}_i$ and $\mathbf{M}_i$ are

$$\mathbf{m}_i = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \qquad \mathbf{M}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{1}$$

$$\mathbf{P} = \begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{r_1} & t_x \\ \mathbf{r_2} & t_y \\ \mathbf{r_3} & t_z \\ \mathbf{0}_3^T & 1 \end{bmatrix} \tag{2}$$

Some of the parameters in the projection matrix are called *extrinsic camera parameters*, and are related to the transformation of position in world coordinates between the camera or projector and a reference frame, such parameters are comprised of three angles that characterize the rotation of a camera around world axes and a translation vector, represented by $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$ and t_x, t_y and t_z in equation (II). The elements $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ are each one a row vector that when stacked result in a common 3D-rotation matrix such as the one used commonly in graphical APIs such as OpenGL or Java3D, and t_x, t_y and t_z are scalar values.

The remaining parameters are *intrinsic* camera or projector parameters, because they determine changes in the image formation plane and are the same regardless of the camera position or orientation. These parameters are the position of the image plane center of coordinates (u_0, v_0) and the parameters α_u and α_v that are related to the size of the focal distance and the size (in continuous values) of the pixels in the image forming sensor. These parameters are related to the focal distance f (the distance from the optical center at the plane where the image is formed is) through the relations $\alpha_u = -f \cdot k_u$ and $\alpha_v = -f \cdot k_v$.

We can conclude then that in order to properly register through projection a point in image plane $\mathbf{m}_i$ over its corresponding 3D point $\mathbf{M}_i$ it is necessary to know 11 parameters, of which 6 are extrinsic and 5 are intrinsic. In this procedure, we are mostly interested in the intrinsic parameters for the extrinsic will be later supplied by tracking the projector. This is a difference between our work and the original Shaderlamps[2] work because the latter estimated the full projection matrix since the projector would be kept immobile and the extrinsic parameters would not change.

In AR X-Ray, a small utility was built to allow the determination of pairs of correspondences between 3D points in world coordinates Mi and their respective image points mi. Such utility allows the user to move a crosshair in image plane until it corresponds to a number of given known points in a calibration board. Such utility can be seen in Figure 4. After 3 sets of 37 correspondences are determined, a free implementation [23] of Zhang's camera calibration algorithm [24] was used to estimate the intrinsic parameters (and extrinsic also, but those were discarded because they change as the projector moves).

Once the intrinsic parameters of the projector are known, tracking is used to determine its extrinsic parameters and compose the proper projection matrix that will achieve correct registration.

3.3.2. Tracking. Tracking is fundamental in the AR X-Ray application to allow the system to project the proper augmentation images over real objects. Usually in projector-based applications when tracking is need commercial magnetic [2] or infrared optical trackers [8] are used. Such types of tracking have good precision and are relatively immune to interferences of projection (such as the projector light), but are usually costly. We

Figure 4. Correspondences between images in projector image coordinates and real world coordinates used in projector calibration.

decided to investigate a more affordable alternative.

The most common kind of tracking employed in augmented reality is fiducial marker tracking, made very popular by ARToolkit. Despite its popularity, ARToolkit has a series of known shortcomings, such as low robustness due to occlusion of markers, instability of tracked poses, sensitivity to variations of illumination, and a steep increase in computational cost as the number of markers one wishes to track grows.

In order to address such shortcomings, the AR research community proposed two tracking libraries that feature ID-based tracking and more robust feature detection, immunity to occlusion and lighting variation and pose tracking algorithms, ARTag [25] and ARToolkitPlus [26]. We chose ARTag because it operated exceptionally under moderate occlusion and both in dark environments as well as in front of projector light too, this is illustrated in Figure 5, where a standard ARTag demo exhibits successful registration even when markers are lit by a projected texture. Such result is known and documented in literature [27, 28].

Figure 5. Successful ARTag registration with markers in front of projected light as seen by webcam.

ARTag is more robust than ARToolkit, but it still exhibits some oscillation and instability in pose tracking for single markers. In order to circumvent this problem one is advised to use multi-markers, a resource that allows the programmer to use a series of markers and describe the known spatial transformations between them to the system. This allows the overall pose to be computed by an optimization procedure that takes into account all markers that are seen by the camera and their positions, and also allows the pose estimation to work even if some markers are not seen.

In AR X-Ray, one goal was to allow the user to move the projector with some freedom inside our laboratory. One problem that needed to be solved was that the field of view and resolution of the webcam only allowed it to track reliably up until a distance of about 1.5 meters from the wall, as the user got further, the markers (that had around 10 centimeters) were no longer detectable.

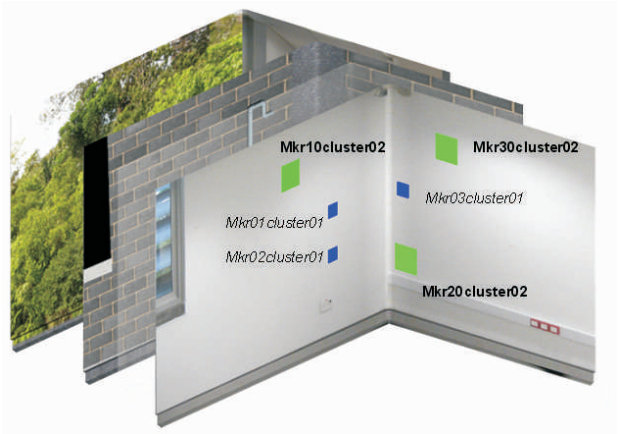


Figure 6. Exploded view of 3D content to be registered over real building.

This problem was solved with an approach first described in a proposal to use ARTag in spacecraft docking [20], in which markers of variable size are used to compose several multi-marker detectors. Larger markers are used when the camera is further and as it approaches sets of smaller markers that can allow a finer positioning, the larger markers are ignored.

In our system the placement of the sets of markers are indicated in the 3D model that represents the scene to represent where the markers are in respect to the scene. A schematic view of such model (where layers of walls are exploded to allow a better visualization) is presented in Figure 6. This model is exported as a VRML file and objects that have names of the form *MkrXXclusterYY* (where XX is the ARTag ID of the marker and YY indicates the multi-marker set it

belongs to) are not treated as common scene objects, instead they are understood as markers and a multi-marker file that can be understood by ARTag describing them is created. The same markers can be seen in Figure 7, where (a), (b) and (c) are markers that allow the detection from larger distances and (d), (e) and (f) are bigger resolution multi-markers.

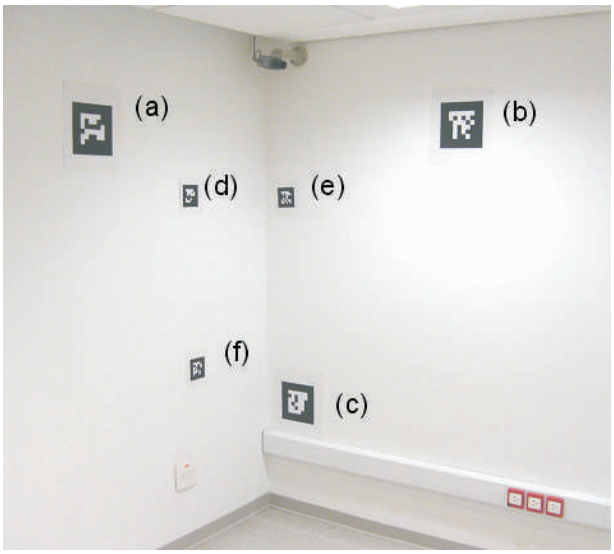


Figure – ARTag markers of different sizes to allow proper tracking at different distances.

Ideally multi-marker setups should be placed with a small error between individual fiducials, usually one large object (such as poster or a board) is printed with all the markers fixed, thus assuring proper low error levels and avoiding compromising the detection.

In AR X-Ray, it was unpractical to print all markers in a single board or sheet due to the dimensions involved (a sheet of approximately 9 m2 would be necessary) and it would occlude the objects to be augmented. Due to these factors, the placement of markers in the physical world was made with the help of a simple application that used ARTag and a webcam that displayed interactively the transformation of a marker being placed in relation to a reference marker and what the goal transformation was based on the data that came from the 3D content file.

One step that is necessary to achieve good precision when tracking fiducial markers is the calibration of the camera to find its intrinsic parameters. Fortunately it is a more automatic process than calibrating the projector, in our project this was done using the GML Camera Calibration Toolbox [29].

The process of estimating the projector position given an image of the scene can be summarized as follows:

i. Find out all markers visible at the scene via their ARTag IDs

ii. Discover the highest-precision set of markers that is visible (this varies with distance, if the user is closer, the smallest markers will be used)

iii. Use ARTag to obtain the transformation between marker set and camera (this transformation is in camera coordinates). The coordinates of the projector in the camera frame is known by design, since they are rigidly attached to each other

iv. Use the inverse of the transformation obtained in (iii) to convert the projector coordinates to marker coordinates

v. Use the information of the VRML file to convert the marker coordinates to room coordinates and thus obtain the extrinsic parameters of the projector

3.3.3. Java3D scene graph. Since Java3D is a scene graph based application programming interface (API), defining such a graph is a significant part of application development, once the main transformations and relations are encoded in it.

AR X-Ray's scene graph is represented in Fig. 8, and as usual there is a common division in a viewing branch and another branch where most of geometry content (the models of walls' layers, in our case), is kept.

The geometry content is below the box called *Content BranchGroup* in the figure, and also holds the set of objects that is responsible for the X-ray effect, by triggering the shaders that run on the GPU through the use of special *ShaderAppearance* nodes that hold references to the scene geometry (represented by *HoleAppearance* in the scene graph).

In our application the 3D models were authored in modelling packages and exported as VRML files (a format that is very well supported under 3D APIs and authoring tools alike and that is able to store materials, texture and animations). One problem that we faced is that there is no way to link shaders to geometry in VRML, and a workaround had to be adopted by the application during content loading time. Such workaround consisted in scanning the scene graph that is being formed as the VRML files are loaded, and replacing any regular Java3D's Apperance graph nodes by our custom *HoleAppearanc*e nodes.

Each *HoleAppearance* is associated with a specific layer of geometry and all the holes in a given scene area controlled by a HoleController class, which is the interface between the user events and the scene graph

and holes position and radius processing.

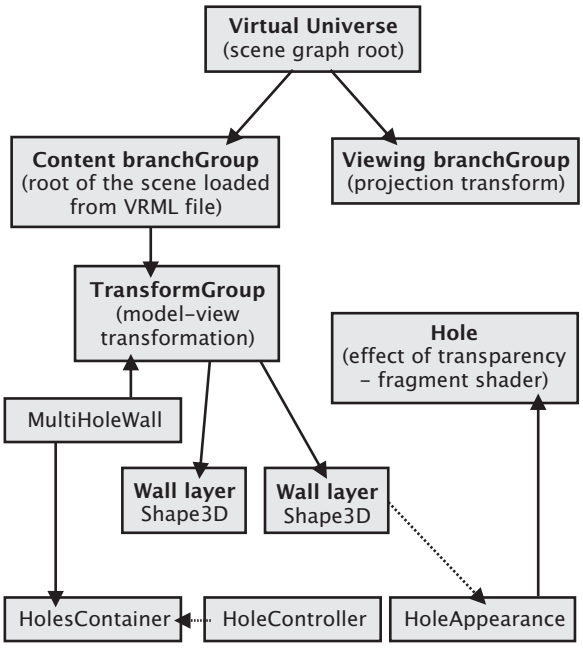


Figure 8. Scene graph of the X-Ray application.

3.3.4. X-Ray shaders. The goal of this project is devising an application that can be used to show to the user the internal structure of objects such as a wall, in our case. After the registration process, the user points at positions in a multi-layer wall and is then presented by the augmenting elements that reveal the innards of the wall.

Giving the user the ability to have the sensation that he is drilling through several layers of wall, varying the depth at his will and controlling the radius of the holes being dug are desired functionalities for the application.

The first approach tried in order to obtain the desired X-Ray effect was to use the texture of the 3D models as a canvas where the holes could be drawn in the alpha channel of such textures. This is a technique that was very fast to implement, with the problems to solve being only the necessity of converting an intersection point obtained between a pick ray controlled by the user and the wall's 3D meshes from 3D coordinates into texture coordinates (*in pixels*). Once this is done, the following step was drawing the intended holes at the texture's alpha channel.

However, the performance of the prototypes was very unsatisfactory in our test system (a 3.2 GHz Pentium 4 CPU with GeForce 6600 GT acceleration card). When using texture images with resolution that provides a photorealistic effect the time spent to refresh the

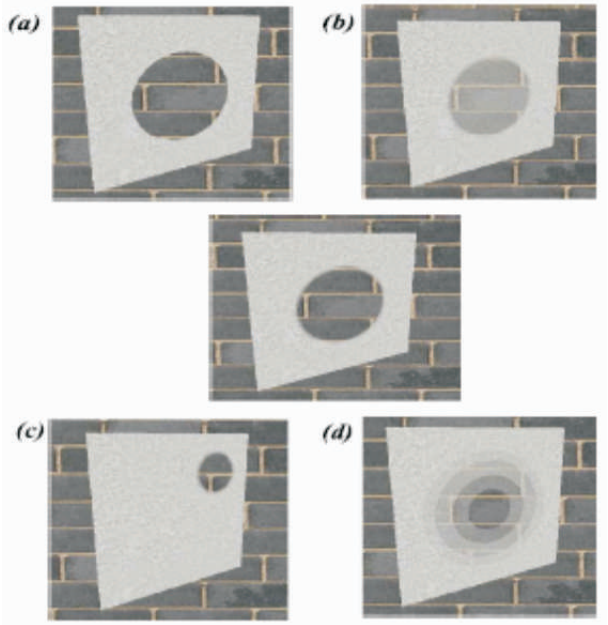


Figure 9. Variation of parameters that determine the hole shader.

screen leads the user to miss the sensation of interactivity. We believe that this is due to (i) the relatively slow operation of iterating through a high resolution image and (ii) inefficiencies in Java3D that sometimes implied in round the tripping whole texture between the graphics card and our application once per frame – we were unable to obtain reference to a texture stored in graphics memory. Such implementation bottlenecks are becoming increasingly less relevant as new and more efficient image I/O and referencing techniques are adopted as Java3D progresses (we used versions 1.4 and 1.5 in our tests).

Considering that the system must be highly interactive to give the user the impression of perfurating the wall, this low performance was an unacceptable problem. Moreover, all the processing of the coordinates of texture and image, picking of pointer position and the changes in the alpha channel of the image are made at the central processing unit, what turns it in a bottleneck in the system.

In order to decentralize the processing, the graphics processing unit (GPU) was used instead of the CPU to draw the hole in the alpha channel. This was done through the use of a fragment, or shader program, which is a routine or piece of code that is compiled in the application startup and executed directly on the graphics processor. A fragment shader executes once for every pixel of the rendered image, in real-time, and offloads processing burden from the CPU helping us assure interactive rates. The shader that was developed

was implemented in OpenGL Shading Language (GLSL) language, and accepted as parameters the variables *radius*, *position*, *attenuation zone* and *target alpha*.

In it is possible to see the variation of the parameters that control the shader – in the center it is possible to see a set of default parameters – a default radius is used, the hole is drawn at the center of the texture coordinates, the alpha value in the hole is 0.0 In the same , the example (a) shows the result of the shader without any attenuation zone for the alpha channel going from 0.0 to 1.0; and in the example (d) we can see the effect of a very large attenuation zone, and in example (b) the alpha at the hole is held at 0.5. In example (c) the texture coordinates of where the hole is drawn is varied, in a way similar to what happens when the user points at a specific place at a wall, the radius of the hole is also varied in this example.

One point worth noticing is that these parameters are passed from Java3D to the OpenGL GLSL shader, and since the shader is coded in a different, purpose-specific programming language, it can be ported over to other 3D graphic APIs easily. OpenGL is an obvious example, but we can also mention Ogre3D and Xith3D, among many others.

4. Results

The prototype (shown in operation in Figure 10) runs at approximately 20 frames per second in our test system (Pentium 4 3.2 GHz equipped with a GeForce 6600 GT accelerator). It tracks projector position well when projector is relatively stable, but even with the multi-markers, there is a perceptible oscillation in pose estimation. This oscillation is smaller when the projector-camera system is very close to the wall and sees all the markers of the smallers sets, but in these conditions where tracking is better the projector image is very small (as shown in Figure 14).

The USB camera that was used has a small delay that causes the system to give an impression of lagging when the projector is moved moderately fast even when running at a rate of 20Hz, but when movements are slow the overall impression is more convincing.

One of the biggest problems is that projector focus had to be adjusted in the range of distances from the wall in which the system was tested (from 1 to 4 meters) in order to avoid a blurred image in the augmentations. The drawback of this is that the intrinsic parameters depend on focus, so we had to keep two sets of intrinsic parameters, one obtained when the projector was close to the wall and other when the projector was at 4.0 meters and switch between them accordingly.

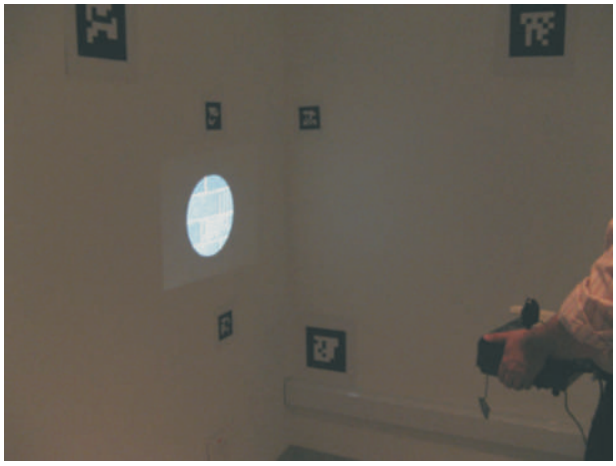


Figure 10. Properly registered brick textures projected over real wall.

5. Conclusions

Some of our initial results reveal that, even with the use of common projectors the concepts presented here are promising and compelling to users. Some contributions we'd like to point out are:

- To our knowledge, it is one of the first attempts of registering a mobile projector by estimating the projection matrix. The original author of *Shaderlamps* [2] obtained such result with a fixed projector, usually works in literature based on mobile projectors only estimate plane to plane mappings in the form of homographies [17].
- A solution to achieving the X-ray effect through the manipulation of textures and we've registered a result of a frustrated implementation using the CPU and a successful implementation based on fragment shaders. That allowed the division of the AR X-Ray computational burden between the graphics card and the processor.
- The descrition of fiducial optical tracking based on ARTag in conditions with difficult illumination (low environmental lights and bright projector light).

6. Future work

The stage of AR X-Ray reported in this paper layed out an infrastructure that can be explored in order to push forward in several directions.

The most immediate direction is address some of the shortcomings of the implementation such as pursuing a more stable tracking through possibly a better pose estimation algorithm (which could involve dropping ARTag since it is not open source) and through the use

of more fiducials, better cameras and even more than one camera.

One direction we would like to investigate is a means of automatically refocusing the projector as the user gets closer to the objects that need to be augmented. This could be done using the tracking information and a mechanic actuator (such as a servomotor) in the projector.

Other natural direction of expansion is the identification of activities (related to work or entertainment, for example) where such approach would be useful. We believe that certain activities in maintenance could be enhanced by this tool – if we find efficient and automatic ways to register the projections, information from buildings' blueprints could be used to help workers locate plumbing pipes and power lines in a convenient way, for example. Some manufacturers produce tiny projectors that might help in such mobile applications.

Visual enhancements that can increase the sensation of augmentation also can be performed – instead of a series of thin flat layers we can study the use of a conjunction of vertex and pixel shaders to provide volumetric rendering or the equivalent of CSG operations.

7.Acknowledgements

Our thanks go to the reviewers that were kind in helping us improve this work, specially for suggesting reference [13].

We'd also like to thank the interns Newton Nakamato and Mark Hodgkin for providing help that was important for this work to be done, and also the Diretoria de Pesquisa do Centro Universitário Senac-SP, which funded this research.

8. References

[1] Bimber, Oliver and Raskar, Ramesh. Spatial Augmented Reality: Merging Real and Virtual Worlds. A K Peters, Ltd, 2005.

[2] Ramesh Raskar, Greg Welch, Kok-lim Low Deepak Bandyopadhyay. Shader Lamps: Animating Real Objects with Image-Based Illumination. 12th Eurographics Workshop on Rendering (EGWR) London, June 25-27, 2001, University College London (UCL).

[3] Pinhanez, Claudio. The Everywhere Displays Projector: A Device to Create . Proc. of Ubiquitous Computing 2001 (Ubicomp"01), Atlanta, Georgia, September 2001.

[4] Low, Kok-Lim; Welch, Greg; Lastra, Anselmo and Fuchs, Henry. Life-sized projector-based dioramas. VRST '01: *Proceedings of the ACM symposium on Virtual reality software and technology* . ACM Press, New York, 2001.

[5] Raskar, Ramesh; van Baar, Jeroen; Beardsley, Paul ; Willwacher, Thomas; Rao, Srinivas; Forlines, Clifton. iLamps: Geometrically Aware and Self-Configuring Projectors. *Proceedings of ACM SIGGRAPH 2003.*

[6] Hosoi, Kazuhiro; Dao, Vinh Ninh; Mori, Akihiro; Sugimoto, Masanori. VisiCon: a robot control interface for visualizing manipulation using a handheld projector. *Proceedings of the international conference on Advances in computer entertainment technology - ACE '07.*

[7] Vinh Ninh Dao, Kazuhiro Hosoi, Masanori Sugimoto . A semi-automatic realtime calibration technique for a handheld projector. *Proceedings of the 2007 ACM symposium on Virtual reality software and technology.*

[8] Xiang Cao, Clifton Forlines and Ravin Balakrishnan. Multi-user interaction using handheld projectors. *Proceedings of the 20th annual ACM symposium on User interface software and technology*.

[9] Blask, G.; Coriand, F. ; Feiner, S. Exploring Interaction with a Simulated Wrist-Worn Projection Display Proceedings. *Ninth IEEE International Symposium on Wearable Computers*, October 2005.

[10] Bane, Ryan; Höllerer, Tobias. Interactive Tools for Virtual X-Ray Vision in Mobile Augmented Reality. *Proc. ISMAR 2004 (IEEE/ACM Intl. Symp. on Mixed and Augmented Reality)*, Arlington, VA, Nov. 2-5, 2004.

[11] White, William W. X-Ray Window: Portable Visualization on the International Space Station. *International Conference on Computer Graphics and Interactive Techniques -SIGGRAPH '04*: ACM SIGGRAPH 2004 Sketches.

[12] Webster, A., S. Feiner, B. Maclntyre, B. Massie, and T. Krueger, Augmented reality in architectural construction, inspection and renovation, in *ASCE Third Congress on Computing in Civil Engineering*. 1996: Anaheim, CA.

[13] Bier, Eric; Stone, Maureen; Buxton, William; DeRose, Tony. Toolglass and Magic Lenses: The See-Through Interface. *InterfaceProceedings of ACM SIGGRAPH 1993*.

[14] Thomas, B. H.; Piekarski, W.; Gunther, B. Using augmented reality to visualise architecture designs in an outdoor environment. *International Journal of Design Computing: Special Issue on Design Computing on the Net (DCNet'99)*, November 1999. University of Sydney.

[15] Hinckley, K., R. Pausch, J. Goble, and N. Kassell, 1994: Passive Real-World Interface Props for Neurosurgical Visualization, *In Proc. ACM CHI '94 (Conference on Human Factors in Computing Systems)*,

452-458.

[16] Interrante, Victoria; Anderson, Lee; Ries, Brian. Distance Perception in Immersive Virtual Environments,Revisited. *Proc. of IEEE Virtual Reality 2006* March 25 -29,Alexandria,Virginia,USA.

[17] Swan II, J. Edward;Livingston,Mark A.; Smallman, Harvey S.;Brown, Dennis; Baillot, Yohan; Gabbard, Joseph L.;Hix, Deborah; A Perceptual Matching Technique for Depth Judgments in Optical, See-Through Augmented Reality.

[18] Livingston,Mark A.; Swan II, J. Edward; Gabbard, Joseph L.;Höllerer, Tobias H.; Hix, Deborah; Julier, Simon J.; Baillot, Yohan; Brown, Dennis. Resolving Multiple Occluded Layers in Augmented Reality. *Proceedings of The 2nd International Symposium on Mixed and Augmented Reality (ISMAR '03)*, October 7–10, 2003, Tokyo, Japan, pages 56–65.

[19] Tracking and Projection Displays Session. *Proceedings of the IEEE Virtual Reality Conference (VR'06)*, pages 19-27.

[20] Mark Fiala. ARTag Fiducial Marker System Applied to Vision Based Spacecraft Docking. *IEEE IROS 2005 Workshop on Robot Vision for Space Applications.*

[21] GlovePIE - Glove Programmable Input Emulator. http://carl.kenner.googlepages.com/glovepie_download.

[22] Faugeras, Olivier. Three-dimensional Computer Vision: a geometric viewpoint . MIT Press, 1993.

[23] Gattass, Marcelo. Zhang Camera Calibration Step by Step. www.tecgraf.puc-rio.br/~mgattass/calibration/.

[24] Zhang, Z. A flexible new technique for camera calibration. *Transactions on Pattern Analysis and Machine Intelligence.* Volume 22, Issue 11, Nov 2000 Page(s): 1330 - 13.

[25] M. Fiala, Artag, a fiducial marker system using digital techniques. CVPR 05.

[26] D. Wagner and D. Schmalstieg. ARToolKitPlus for Pose Tracking on Mobile Devices. *Computer Vision Winter Workshop 2007.*

[27] M. Fiala. Comparing ARTag and ARToolKitPlus Fiducial Marker Systems. HAVE 2005. *IEEE International Workshop in Haptic Audio Visual Environments and their Applications, 2005.*

[28] M. Fiala. Fiducial marker systems for augmented reality: Comparison between artag and artoolkit. In *MIRAGE 2005, INRIA Rocquencourt, France, MAr 2005.*

[29] GML Camera Calibration Toolbox. http://research.graphicon.ru/calibration.

[30] Hua, Hong; Gao, Chunyu; Ahuja, Narendra. Calibration of a head-mounted Projective Display for Augmented Reality Systems. *Proceedings of the International Symposium on Mixed and Augmented Reality* (ISMAR'02), 2002.

Incorporating User Preference to Represent Information for Manual Work Supported by Augmented Reality

Johannes Tümler, Christian Scharf, Rüdiger Mecke, Michael Schenk
Fraunhofer Institute for Factory Operation and Automation IFF Magdeburg
{johannes.tuemler},{christian.scharf},{ruediger.mecke},{michael.schenk} @iff.fraunhofer.de

Georg Paul
Otto-von-Guericke University Magdeburg
paul@iti.cs.uni-magdeburg.de

Abstract

Augmented reality (AR) enhances sensory perception with situationally relevant virtual information. In national research and development projects, prototypes have demonstrated that AR is a suitable technology for supporting industrial work processes by providing information "on necessity". Up to now, studies have given user-oriented aspects (e.g. acceptance, ergonomics) little consideration though. This paper describes a lab experiment in which user preferences regarding the size and color of AR overlays were researched and draws pertinent conclusions for further development.

1. Motivation and Questions

The provision of information plays a fundamental role in industrial manufacturing. Work steps must be described, information on spatial positions of components must be accurately communicated to workers and warnings must be immediately recognizable as such. That is why augmented reality (AR) technology has been a particular focus of research in Germany for several years [27]. AR presents situationally relevant information in a user's field of view, e.g. through a head mounted display (HMD). While previous studies have demonstrated this technology's great potential [6, 1, 26], they have also identified open questions, for example with regard to user-related factors like ergonomics and acceptance of hardware and software [25]. Thus, for example, what type of overlays most effectively supports manual service and assembly activities in industry has still not yet been clearly established. Sandtorstr. 22, 39106 Magdeburg, Germany School of Computer Science, Department of Technical and Business Information Systems, 39106 Magdeburg, Germany The research being conducted in this field covers various realms. One question deals with the color and size in which guide objects (e.g. arrows, icons, text, etc.) ought to be overlaid in an HMD and whether there are different user preferences that ought to be incorporated.

2. Today's Head Mounted Displays

Since Ivan Sutherland developed the first HMD in 1965 [24], the technology has advanced tremendously. Today's AR compatible HMD can be differentiated according to three visualization techniques: Video see-through (VST), optical see-through (OST) and projection seethrough (PST) (see Figure 1).

In the case of VST, a video camera worn by the user (usually attached to HMD) records the user's environment. This is processed by a computer (mixer) to generate a new image that can be enriched with additional virtual overlays. A monitor then displays the new image in the user's field of view. The properties of the camera and the HMD (e.g. resolution, field of view, etc.) as well as the camera's positioning on a user restrict the user's view of the real world. This generally impairs a user's hand-eye coordination [2, 15] and consequently should not be employed in industrial environments in which optimal perception or interaction with the environment is essential. OST circumvents this problem by granting users an almost normal view of the environment: A user of an OST HMD receives additional virtual elements solely through a semi-transparent mirror, while simultaneously being able to view the real world. The complete image (virtual information plus the image of the real world) is generated in the user's head. However, this type of visualization technology gives rise to new challenges pertaining to the correctly positioned linkage of the virtual overlay with the real world [28]. The third visualization technology, projection see-through, is comparatively new and has been scarcely addressed in the literature. While VST produces the final image on a computer and OST in a user's eye, PST does this in the real world by projecting virtual overlays directly onto the environment. Problems arise here too though. Objects that are always close and reflect but do not mirror light must be available as projection surfaces. This is not always the case in industrial applications since metals and finishes in particular can make a projected image difficult to discern. Figure 1 presents

*Sandtorstr. 22, 39106 Magdeburg, Germany
*School of Computer Science, Department of Technical and Business Information Systems, 39106 Magdeburg, Germany

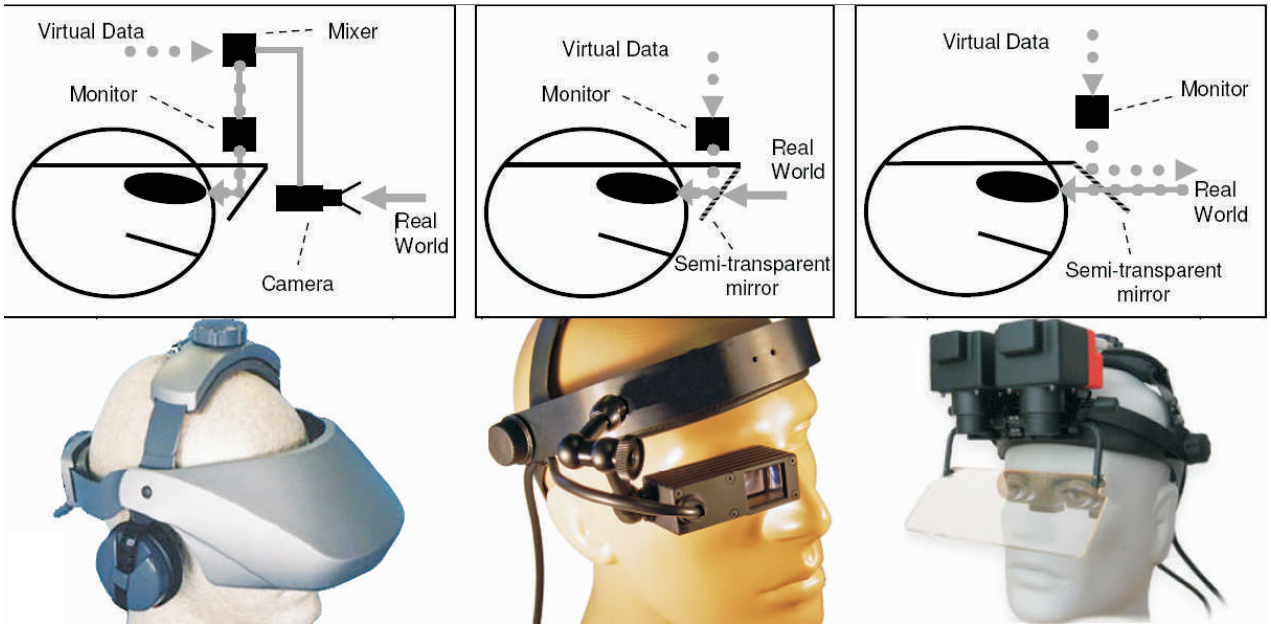


Figure 1. See-through categories: Video see-through (left), optical see-through (middle) and projection see-through (right); example devices: left: 5DT HMD 800-40 (VST), middle: LitEye LE-750 (OST), right: NVIS P-50 (PST).

the different HMD.

OST displays are predominantly suited for use in industry. A research of OST displays revealed differences chiefly with respect to the following criteria (based on [14]):

Handling

- Physical dimensions (e.g. weight, size, mobility, headgear used),
- Ruggedness (e.g. material, shock resistance, dust and water protection class),
- Connectivity (e.g. VGA, S-Video) and
- Power supply (e.g. battery, USB, mains connection).

Visualization

- Display technology (e.g. retinal scan display, OLED LCD, etc.),
- See-through level (currently between 3 % and 99 %),
- Capability to display color (currently 24-bit color, gray scales, monochrome),
- Resolution (currently between 320 x 240 and 1280 x 1024 pixels per color) and
- Luminance (currently between 3.5 and 2800 cd/m2)

Only very few of the OST HMD researched can be acquired in the civilian (non-military) sector and only OST devices can be purchased from four different vendors without difficulty. This has made it impossible to test a wide range of different devices or select the "optimal" display for industrial use.

Two exceptional HMD studies that have however not been pursued further deserve mention: Apart from conventional displays based on raster graphics, research has been done on HMD that approximate the physiology of the human eye decidedly better than current HMD [4]. The principle is based on beaming a higher density of pixels onto that part of the eye with sharper vision than the rest. This is achieved with an image-generating beam of light with a spiral form that narrows at its center. A second remarkable study describes how the phenomenon of focusing between HMD image level and real world could be eliminated [11]. The beams of light from a laser retinal display are initially parallelized by a lens and then a second lens focuses their focal length (nodal imaging), making it unnecessary for the eye to accommodate itself when viewing an image regardless of the distance to the real world it is focusing on. Should these or similarly radical changes in HMD development become the standard in the future, present visualization concepts would potentially have to be fully revised.

3. Presentation of Information

Augmented reality presents situationally relevant informationin a user's field of vision. The term "information" is the subject of heated debate in various disciplines [16, 17]. As employed here, it is understood as data transmitted from a sender to a

recipient for a specific purpose. To be transmitted, information must first be converted into symbolism comprehensible to the recipient. In the case of augmented reality, this means converting information into visual stimuli regardless of the reason it is being transmitted. Today's HMD technology makes it possible to expose users to these stimuli as raster graphics, the computer being able use geometric bodies, photos/videos or text to represent these visual stimuli (Figure 2). While geometric bodies can be clearly specified by mathematical functions and thus scaled as desired (vector description), this is not the case for photos/videos since their informational content is described by the color of pixels arranged in a raster (raster graphic). As a result, geometric forms being displayed can usually only be approximated mathematically. Along with such description, Figure 2 specifies text as an independent representation class, which, while it can be described with vectors or by raster graphics, is however imaged differently by a computer and processed differently by people [30].

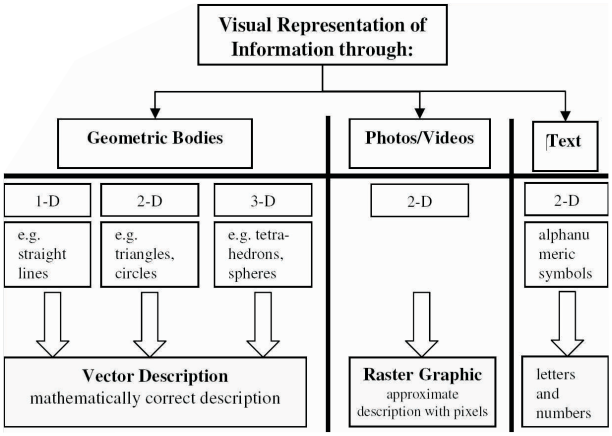


Figure 2. Options for the visual representation of information.

Depending on the information being displayed, a decision has to be made about the form of representation that will be used to present the information by AR. If, for example, information is intended to be an instruction to retrieve an item in a warehouse, then this information could be described purely textually as a part number. On the other hand, a 3-D arrow could point directly to the retrieval point or a photo of the part to be retrieved could be overlaid. At any rate, the user must interpret the representation of information provided in the HMD. Since any interpretation is always affected by subjective influences, the expected result (user understands instruction as intended) is not guaranteed one hundred percent. Hence, information must be represented in such a way that misinterpretation is as unlikely as possible. In the process, various conditions have to be factored in whenever an OST display is employed:

- The user's primary task is not interacting with the AR system or obtaining information from the HMD overlay but rather producing a technical product.
- The representation in the HMD may not conceal too much of the surrounding real world.
- The user may not be overloaded by too many overlaid elements, i.e. too much information may not be displayed all at once.
- The color of the overlaid information should trigger rapid absorption of information and/or steer attention and not stress the user.

As the next section points out, there have hardly been any studies of users, which incorporate these conditions.

4. State of Research

Since Boeing coined the term "augmented reality" in the early 1990s [12] technological and technical issues have constituted the chief focus of research and development. User issues (e.g. acceptance, technology's influence on humans) have hardly received consideration though [9]. This generally includes the design of user interfaces both on the hardware and on the software side [18, 3, 8].

Different individuals prefer different types of assistance [23]. Hence, it must be possible to overlay some information "on demand", whereas critical warnings for example must be overlaid in the AR display "on necessity". It is imperative to categorize the urgency of information and to determine the extent to which the individual categories ought to be overlaid on demand or on necessity. Questions addressing when and why (i.e. in what context of use) what information is overlaid also play a role [22, 20]. The amount of informational content necessary in a particular context is still unclear, yet could for example be dynamically adapted depending on the distance to the target object [5]. One aspect that has received little consideration thus far is research on types of representation for AR displays [29]. What type of representation best enables understanding and effectively perceiving information with the aid of an HMD and produces optimal results has scarcely been documented [21, 13]. While it is known that the type of representation of information strongly influences completion times [14, 22], why this is so or how influences are generated has however not

been sufficiently researched. Various options exist to distinguish guide objects [7], which deserve to be evaluated in an industrial context. One potential influence could be different colors of light and different kinds of light (e.g. generated by laser or halogen), which have different effects on users [11].

Augmented reality greatly needs research in the field of information visualization. Fundamental, open questions ask, for example, what size and what color overlaid guide objects ought to have to support manual industrial processes.

5. Methodology

Issues fundamental to AR visualization of information remain open. It is assumed that the dimensions and coloration of guide objects influence acceptance of an overall system [14]. To determine whether different user preferences exist regarding the dimensions and coloration of guide objects for manual processes, an experimental test of manual activities supported by AR was conducted at the Fraunhofer IFF. The test setup was designed so that various boxes were arranged on a table next to a wall that was colored in light blue. The test persons involved were located approximately 50 cm away from them. Figure 3 presents a diagram of the test setup. A Sony Glasstron HMD, which provides both an optical see-through (OST) and a video see-through (VST) mode, was used in the tests. It displays colors with a resolution of 800 x 600 pixels and a field of view (FOV) of approximately 42 degrees (h) x 31 degrees (v). In the VST mode, the image from the camera (Trust WB-6200p, FOV 63 degrees (h) x 47 degrees (v), resolution in test 640x480 pixels) mounted on the HMD was overlaid in the HMD. An optical, marker-based tracking system from metaio was used to determine user position (tracking).

Altogether 16 male and 6 female test persons between 21 and 29 years old (Ø 25.8) were selected. They were both students and technical staffers. The 22 test persons were chosen randomly and participated in the test voluntarily. Their job was to specifically position, stack and turn the boxes in the test setup guided by the AR overlays displayed in their HMD (Figure 4). Instructions were overlaid as arrows, rectangles and, to some extent, text, which at first had a fixed preset initial size and color. During the approximately twenty minute test run, the test subjects were given four opportunities to alter the size and color of the represented objects by using their mouse scroll wheel. The Wizard of OZ method [10] was applied to ensure that once one work step had been completed the next would be initiated. Halfway through the test, the overlay was switched from VST to OST on the HMD controller in order to additionally test whether size and color selection varies between VST and OST. A mixed order setup (changing the order of VST and OST usage between the different test subjects) was not involved due to the fact that the user was able to change the size twice per VST / OST session. Each test subject had to execute a total of sixty-two work steps.

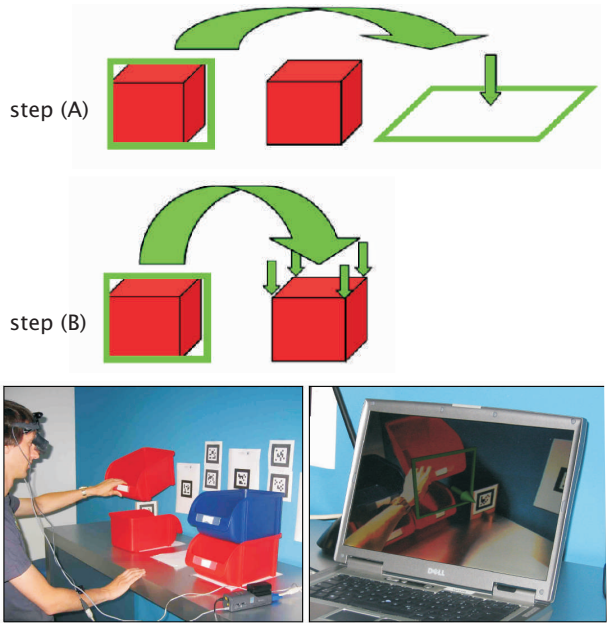


Figure 4. 3-D arrows, 3-D rectangles, etc. describe the test job (top); retrieving and stacking boxes according to the instructions in the HMD (bottom left), laptop for supervision of the test (bottom right) of approximately the same size.

The following hypotheses were formulated before the start of the test:

Existing studies report that certain size settings of virtual objects are more practical and better suited than others [14]. Hence, hypothesis 1 ensued:

H1: All test subjects tend to prefer virtual objects of approximately the same size.

Ultimately, even when presented by a display unit (e.g. HMD), every visually perceived object is reproduced on the retina in a certain size. This led to the formulation of hypothesis 2:

H2: No fundamental differences between VST and OST mode are discernible.

The colored overlayed objects could easily be recognized on the wall in the background, no matter what color was selected for the objects. The color green has a calming effect on humans, which allowed

inferring hypothesis 3:

H3: Green is preferred for guide objects; other colors are rarely selected.

These hypotheses were confirmed or refuted in the course of the test.

6. Findings

6.1 Object Size and Color

During the lab test, every test subject took advantage of the option to scale the size of the virtual objects. 15 of 22 participants retained the size they set at the start even as the test progressed. Some test subjects worked with very small representations. Others tended to prefer very large guide objects (see Table 1 and Figure 5). Consequently, H1 was refuted. A similarly organized follow up test is intended to determine the extent to which fixed preset object sizes influence completion speed and quality in comparison with manually adjustable object sizes.

Table 1. Summary of the results of the size selection test.

	VST	OST
Objects subsequently scaled up	6	2
Objects subsequently scaled down	1	3
Coverage at the end of the test segment	min = 0,24% max = 4,05% Ø = 0,74 σ = 0,83	min = 0,30% max = 2,89% Ø = 0,82 σ = 0,60

When using VST, 16 of 22 participants (approximately 75 %) preferred coverage of 0.24 % to 0.65 %, which corresponds to coverage between 2.71 and 4.39 degrees (diagonal) of the FOV. Since the HMD's FOV rather than the camera's FOV was crucial for the OST test, the results inevitably differed. When using OST, the same number of test subjects preferred coverage of 0.30 % to 0.83 % of the FOV, which approximately corresponds to 3.0 to 4.95 degrees of the diagonal FOV (see Table 2). Thus, H2 was supported.

Particularly conspicuous was that a wider range of preferred sizes was selected for OST than for VST. Thus, manual size selection appears to be far more important for OST than for VST. ought to be overlaid on demand or on necessity. Questions addressing when and why (i.e. in what context of use) what information is overlaid also play a role [22, 20].

Table 2. Summary of the results of the size selection test.

	VST	OST
FOV coverage preferred by the majority of users (75%)	min = 0,24% ≙ 2,71 deg. max = 0,65% ≙ 4,39 deg.	min = 0,30% ≙ 3,0 deg. max = 0,83% ≙ 4,95 deg.

Generalizing these findings, the incorporation of user preference allows drawing the conclusion that, depending on the degree (diagonal) of the HMD's FOV, the majority of users' individual guide objects may take up between 0.05 degrees to 0.10 degrees. Since the object sizes selected in the OST test exhibited a noticeably broader statistical spread, users should be provided the option of setting sizes that suit them

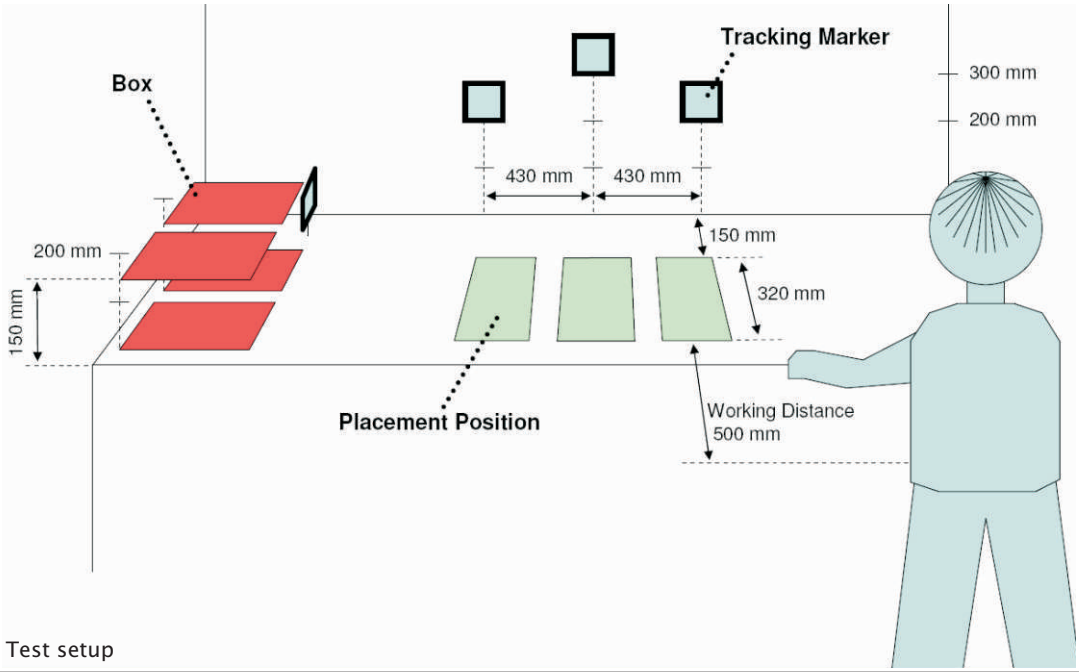


Figure 3. Test setup

individually depending on the application.

All 22 test participants found it helpful to be able to not only scale the size of overlaid objects but also alter their color. Since the color green was predominantly selected throughout the entire test procedure, H3 was confirmed. The reason test participants gave for their preferred color selection was its excellent perceptibility against the real background. Users hardly utilized a subsequent manual color adjustment (see Table 3), which could however potentially be necessary against backgrounds containing widely varying, richly contrasting colors.

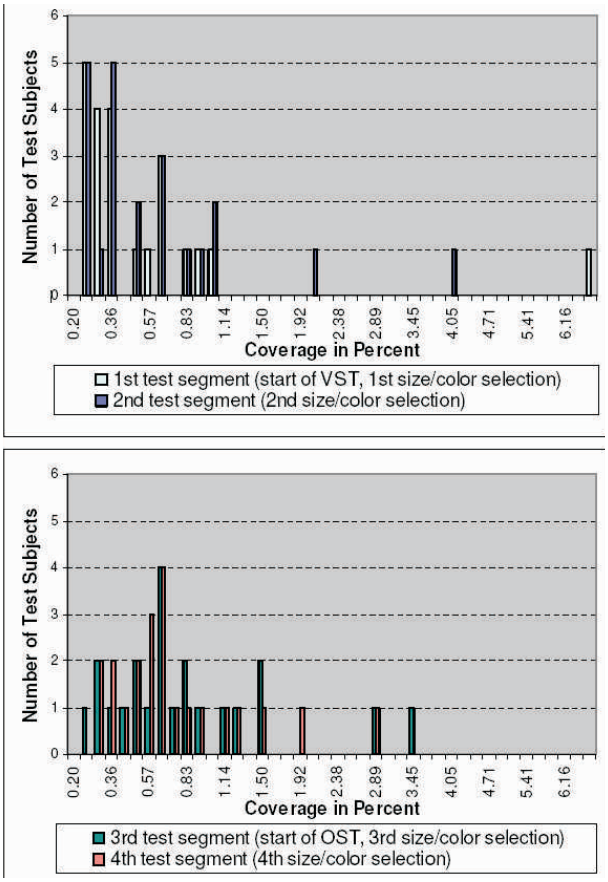


Figure 5. Evaluation of the object coverage test. top: VST procedure with two opportunities to change size. bottom: OST procedure with two opportunities to change size.

Table 3. Summary of the results of the color selection test.

	VST	OST
Color setting after the end of the test segment	63,3% green 22,7% blue 9% cyan 4,5% yellow	82% green 9% cyan 9% yellow
Color subsequently altered	3	1

6.2 Miscellaneous Observations

During the test conducted with VST, nearly every test subject criticized the impairment of hand-eye coordination. This conforms to already existing studies [2, 15]. Approximately 60 % of the test subjects noted noticeably improved hand-eye coordination with OST. This supports the initially formulated proposition to only employ OST as far as possible to provide AR support to manual industrial processes. The average work time with VST was 6.2 minutes, with OST only 4.6 minutes. 90 % of the test subjects who described working with VST as unpleasant worked faster with OST. Improved hand-eye coordination on the one hand and the acclimatization effect experienced when using what is initially a new medium for the majority of the users on the other hand are presumed to be reasons.

During the course of work, certain overlaid help objects were repeatedly not always clearly identifiable depending on the perspective. Thus, the points of some guide arrows and the direction they specified were no longer clearly recognizable under certain circumstances. The misinterpretations that occurred indicate other forms should be utilized. These should always be clearly interpretable from every perspective without users having to change their position or move their heads and the meaning of all the objects overlaid during the work sequence should remain uniform. This supposition was underpinned by intentionally providing the test subjects different representation options to mark objects during the test (i.e. a target position was initially marked with a frame, then with an arrow and finally with four arrows). This produced misinterpretations on the part of nearly every test subject.

Generally, when ambiguities exist, users ought to be given the option to overlay additional text. However, this would have to be as brief a possible and ideally limited to a single word designating an action (e.g. grasp, turn, etc.). One possibility would be user-controlled, manual overlaying or hiding of additional captions. Some test participants even expressed the desire to be able to manually overlay or hide every AR overlay.

The test records do not reveal any differences between male and female test participants or between individuals who wear glasses and individuals who do not.

7. Summary and Discussion

Since it can provide information based on situation and

context in a user's field of vision, mobile AR technology is suited to support manual work activities in industrial environments [19]. Nonetheless, mobile AR technology has hardly been and is hardly being used productively. The reasons are complex and, in the opinion of these authors, are primarily rooted in user-related issues, e.g. acceptance of overall AR systems (hardware, software, HMI, etc.). This study is intended to contribute to increased acceptance of such mobile AR systems by, among other things, analyzing the size and color with which head mounted displays should overlay virtual objects. Differences between individual users were detected and further tests allowing users to manually set size and color selection were proposed.

The test subjects' manual selection of different sizes of representation suggests that the size of representation significantly influences the unambiguous perceptibility and correct interpretation of information and consequently also the execution time of individual work steps. Further research at the Fraunhofer IFF is intended to substantiate this. Tests on a realistic scenario are envisioned.

Apart from the potential to display directions for manual activities in an assembly process for example, augmented reality is excellently suited to provide users navigation information (route guidance to a specific location). The extent to which the findings obtained from the test described above also apply to navigation deserves research.

Currently ongoing tests have coupled head mounted displays with an eye tracker in order to investigate the extent to which the absorption of information from an HMD differs from "normal" absorption without an HMD. Initial findings indicate differences do in fact exist. These differences will play an important role in future studies and be researched intensively.

It will be essential to incorporate other user-oriented factors (see, for example, section 4) before it will be possible to effectively realize the great potential benefits of AR in industrial applications.

8. References

[1] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47, 2001.

[2] F. A. Biocca and J. P. Rolland. Virtual eyes can rearrange your body: Adaptation to visual displacement in seethrough, head-mounted displays. *Presence: Teleoper. Virtual Environ.*, 7(3):262–277, 1998.

[3] H. Brau, C. Ullmann, M. Duthweiler, and H. Schulze. Gestaltung von augmented reality applikationen für kommissionieraufgaben. In *Zustandserkennung und Systemgestaltung 6. Berliner Werkstatt Mensch-Maschine- Systeme 13. bis 15. Oktober 2005* (ZMMS Spektrum Band 19), volume 22 of Fortschritt-Berichte, Düsseldorf, Germany, 2005. VDI Verlag GmbH.

[4] R. Buecher. Wearable mobile display based on the human physiology to set new standards for human machine interfaces. In *The 22nd Digital Avionics Systems Conference*, volume 2, pages 9.E.4.1–9.E.4.4, 2003.

[5] S. DiVerdi, T. H¨ollerer, and R. Schreyer. Level of detail interfaces. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality ISMAR04*, pages 300–301, Arlington, VA, 2004.

[6] W. Friedrich, editor. ARVIKA Augmented Reality für Entwicklung, Produktion und Service. *Publicis Corporarte Publishing Verlag, Erlangen*, Germany, 1 edition, 2004.

[7] J. L. Gabbard, J. Edward II Swan, D. Hix, R. S. Schulman, J. Lucas, and D. Gupta. An empirical user-based study of text drawing styles and outdoor background textures for augmented reality. In *Proceedings IEEE Virtual Reality 2005*, pages 11–18. IEEE Computer Society, 2005.

[8] T. Hollerer, S. Feiner, T. Terauchi, G. Rashid, and D. Hallaway. *Exploring mars: Developing indoor and outdoor user interfaces to a mobile augmented reality system*. pages 779– 785, 1999.

[9] J. Edward II Swan and J. L. Gabbard. Survey of user-based experimentation in augmented reality. In *Proceedings 1st International Conference on Virtual Reality*, Las Vegas, USA, 2005.

[10]J. Kelley. An empirical methodology for writing userfriendly natural language computer applications. In *Proceedings of ACM SIG-CHI '83 Human Factors in Computing systems*, pages 193–196, New York, USA, 1983. ACM.

[11] M. Menozzi. Human factors in "augmented reality": Nutzbarkeit der knotenpunktabbildung im "virtual retinal display". *Technical report, Institut für Hygiene und Arbeitsphysiologie*, Eidgenössische Technische Hochschule, Zürich, Austria, 2001.

[12] D. Mizell. Boeing's wire bundle assembley project. In *W. Barfield and T. Caudell, editors, Fundamentals of Wearable Computers and Augmented Reality*, pages 447–467. Mahwah: Lawrence Earlbaum Associates, 2001.

[13] H.-G. Nusseck. Wahrnehmung virtueller welten - warum die virtuelle welt anders ist als die realität. In *9th IFF Wissenschaftstage*, page 311. Fraunhofer IFF Magdeburg, Germany, 2006.

[14] O. Oehme. Ergonomische Untersuchungen von kopfbasierten Displays für Anwendungen der erweiterten

Realität in Produktion und Service. *PhD thesis, RWTH Aachen*, Germany, 2004. Schriftenreihe Rationalisierung und Humanisierung, Bd. 61. Aachen, Germany: Shaker Verlag.

[15] M. Park, L. Schmidt, C. Schlick, and H. Luczak. Design and evaluation of an augmented reality welding helmet. In *Human Factors and Ergonomics in Manufacturing*, volume 17 - 4, pages 317–330. Wiley Periodicals, Inc., A Wiley Company, 2007.

[16] P. Rechenberg. Zum informationsbegriff der informationstheorie. *Zeitschrift Informatik-Spektrum*, 26(5):317–326, 2003.

[17] P. Rechenberg and G. Pomberger. Informatik-Handbuch. *Hanser Fachbuch*, 3 edition, 2002.

[18] J. P. Rolland and H. Fuchs. Optical versus video see-through head-mounted displays in medical visualization. *Presence: Teleoper. Virtual Environ.*, 9(3):287–309, 2000.

[19] S. Sauer and J. T¨umler. Ein assistenzsystem zur verbesserung von montageprozessen. In (in print) *Forschung vernetzen - Innovationen beschleunigen, wissenschaftliches Kolloquium*. Fraunhofer IFF Magdeburg, Germany, 2007.

[20] L. Schmidt. Benutzerzentrierte ar-systemgestaltung. In M. Schenk, editor, *Virtual Reality und Augmented Reality zum Planen, Testen und Betreiben technischer Systeme - 8. IFF-Wissenschaftstage 22.-24. Juni*, pages 103–112, Magdeburg, Germany, 2005. Fraunhofer Institut f¨ur Fabrikbetrieb und - atisierung IFF.

[21] L. Schmidt, S.Wiedenmaier, O. Oehme, and H. Luczak. Benutzerzentrierte gestaltung von augmented reality in der produktion. In *C. Stary, editor, Mensch und Computer 2005: Kunst und Wissenschaft - Grenz¨uberschreitungen der interaktiven ART*, pages 51–60. Oldenbourg Verlag, München, Germany, 2005.

[22] B. Schwerdtfeger, T. Frimor, D. Pustka, and G. Klinker. Mobile information presentation schemes for logistics applications. In *16th International Conference on Artificial Reality and Telexistence (ICAT 2006)*, Hangzhou, VR China, 2006. Zhejiang University of Technology.

[23] S. P. Smith and J. Hart. Evaluating distributed cognitive resources for wayfinding in a desktop virtual environment. In *VR '06: Proceedings of the IEEE Virtual Reality Conference (VR 2006)*, page 115, Washington, DC, USA, 2006. IEEE Computer Society.

[24] I. E. Sutherland. The ultimate display. In *Proceedings of the IFIP Congress 2*, pages 506–508, 1965.

[25] A. Tang, C. Owen, F. Biocca, and W. Mou. Comparative effectiveness of augmented reality in object assembly. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 73–80, New York, NY, USA, 2003. ACM Press.

[26] A. Tang, C. Owen, F. Biocca, and W. Mou. Performance evaluation of augmented reality for directed assembly. In *S. K. Ong and A. Y. C. Nee, editors, Virtual and augmented reality applications in manufacturing*, pages 301– 322, Berlin, Germany, 2004. Springer Verlag.

[27] J. Tümler and R. Mecke. Mobile augmented reality für die werkerassistenz. In *Forschung vernetzen - Innovationen beschleunigen, wissenschaftliches Kolloquium*, pages 40– 50. Fraunhofer IFF Magdeburg, Germany, 2006.

[28] J. T¨umler, R. Mecke, and J. Xu. See-through kalibrierverfahren f˙ur mobile augmented reality assistenzsysteme. In *J. Gausemeier and M. Grafe, editors, Augmented und Virtual Reality in der Produktentstehung, volume 6*, pages 233–247. Heinz Nixdorf Institute, University of Paderborn, Germany, HNI-Verlagsschriftenreihe, 2007.

[29] S. Wiedenmaier. Unterst¨utzung manueller Montage durch Augmented Reality-Technologien. *PhD thesis, RWTH Aachen*, Germany, 2003. Schriftenreihe Rationalisierung und Humanisierung, Bd. 58. Aachen, Germany: Shaker Verlag.

[30] W.Winn. Encoding and retrieval of information in maps and diagrams. In *IEEE Transactions on Professional Communication*, volume 33, 3, pages 103–107, 1990.

The Image-Based Data Glove

Vitor F. Pamplona, Leandro A. F. Fernandes, Jo.ao L. Prauchner, Luciana P. Nedel, Manuel M. Oliveira

Instituto de Informática - PPGC
Universidade Federal do Rio Grande do Sul
Porto Alegre - RS - Brazil
{vfpamplona}, {laffernandes}, {jlprauchner}, {nedel}, {oliveira}@inf.ufrgs.br

Abstract

Data gloves are devices equipped with sensors that capture the movements of the hand of the user in order to select or manipulate objects in a virtual world. Data gloves were introduced three decades ago and since then have been used in many 3D interaction techniques. However, good data gloves are too expensive and only a few of them can perceive the full set of hand movements. In this paper we describe the design of an image-based data glove (IBDG) prototype suitable for finger sensible applications, like virtual objects manipulation and interaction approaches. The proposed device uses a camera to track visual markers at finger tips, and a software module to compute the position of each finger tip and its joints in real-time. To evaluate our concept, we have built a prototype and tested it with 15 volunteers. We also discuss how to improve the engineering of the prototype, how to turn it into a low cost interaction device, as well as other relevant issues about this original concept.

1. Introduction

Data gloves were introduced by Sandin et al. [7]. Since then, gloves are highly used devices in virtual reality environments, having been mapped to many selection and manipulation techniques, as [16, 18, 21]. Among all commercial solutions, there are simple models capable of giving pitch and roll information of the hand of the user and of measuring finger exure [1, 8]. However, there are special cases [10, 15] where a glove needs to map the full movement of hands and fingers. This includes pitch, yaw, roll and XYZ-translations of the hand and fingers per joint exion and adduction. Unfortunately, this set of features are restricted only to the most expensive data gloves [13, 17, 24].

Concerned about the high cost of the most complete commercial solutions, we propose a new input device: the Image-Based Data Glove (IBDG). Figure 1 illustrates the overall idea of the IBDG. By attaching a camera to the hand of the user and a visual marker to each finger tip, we use computer vision techniques to estimate the relative position of the finger tips. Once we have information about the tips, we apply inverse kinematics techniques [12] in order to estimate the position of each finger joint and recreate the movements of the fingers of the user in a virtual world. Adding a motion tracker device, we can also map pitch, yaw, roll and XYZ-translations of the hand of the user, (almost) recreating all the gesture and posture performed by the hand of the user.

Figure 1. The Image-Based Data Glove prototype: a camera attached to the user's hand tracks his/her fingers using real-time computer vision techniques. Information about finger movements are estimated using inverse kinematics techniques.

The main contributions of this paper are the design of the image-based data glove, whose features include:

• To perform continuous real-time tracking of finger tips positions; and

• The reproduction of exion and adduction of the fingers through inverse kinematics techniques.

We also describe the engineering of the first prototype of the IBDG and present a detailed discussion on how to improve it and make it a low cost and easy to build input device.

The remaining of the paper is organized as follows: Section 2 discusses the history and some existing commercial and non-commercial data gloves. Section 3

present the concepts behind the proposed device and describes the construction of a first prototype. Section 4 describes the testbed application developed to evaluate the data glove prototype, and presents the evaluation criteria and parameters for the experiments. Section 5 presents an analysis of precision and usability of our current implementation and Section 6 makes a discussion on how to improve the IBDG prototype. Finally, Section 7 concludes the paper with some observations and directions for future work.

2. Related Work

Several glove based devices have been developed in last few decades and an in-depth discussion about many of these devices can be found in [20].

De Fanti and Sandin [7] developed the Syre's Glove at University of Chicago. This glove detects movements of fingers by using, along each finger, exible tubes with a light source at one extremity and a photosensitive cell at the other. As users bent their fingers, the variation of light is sensed and correlated to the finger bending.

Zimmerman developed the VPL Data Glove [24], which was an improvement over existing devices and techniques. It provides real-time tracking of the hand position and orientation, as well as monitoring fingers adduction and exion. The VPL Data Glove consists of a Lycra glove with optical fibers attached along the backs of the fingers. Finger exion bends the fiber, which changes the attenuation of transmitted light. This attenuation is sent to a processor which determines joint angles. A magnetic tracking device is attached in the back of the hand in order to capture position and orientation of the palm. Despite its commercialization and widespread use, the VPL Data Glove is not accurate enough for complex gesture cognition.

Another commercial glove developed in the 1980s is the Dexterous HandMaster [20]. It consists of an exoskeletonlike device worn on the fingers and hand. It uses hall-effects sensors as potentiometers at the joints to measure exion and adduction of the fingers and the thumb. Its speed (200 samples per second) and accuracy make it suitable for fine work or clinical analysis of hand function and impairment. One drawback of this device is the fact that it is hard to put on and take off, besides the need of adjustments to fit the hand properly.

The Mattel toy company manufactured a well-known glove peripheral for the Nintendo video game, the Power Glove [20]. It was inspired in VPL Data Glove, but with many modifications that allowed it to be used with a slow hardware and sold for an affordable price. While VPL Data Glove can detect yaw, pitch and roll, uses fiber optics sensors to detect finger exure and has a resolution of 256 positions (8 bits) per 5 fingers, the Power Glove can only detect roll, and it uses sensors coated with conductive ink yielding a resolution of 4 positions (2 bits) per 4 fingers. This allowed the Power Glove to store all the finger exure information in a single byte. Acoustic trackers that accurately locate the hand in space with respect to a companion unit located in the top of television monitor are mounted in the back of the hand. The trackers also provide four bits of roll orientation for the hand. It works with several Nintendo games, some of them especially designed for the Power Glove.

Currently, several data gloves with a broad range of features are available for prices ranging from $60 to $16,000. The cheapest device is the P5 Glove [22], developed by extinct company Essential Reality. It consists of a support to which flexible sticks are connected. These sticks are attached to the fingers. The user moves its hand in front of a receptor "tower" containing two infrared sensors. Sensors detect visible LEDs on the glove (there are eight altogether), and convert each of them to a position $(x; y; z)$ for the hand, and a spatial orientation in terms of pitch, yaw, and roll. The device provides six degrees of tracking at a 60Hz refresh rate. Although the P5 Glove is a low-cost device, its optical tracker suffers from line of sight limitations.

The Pinch Glove, developed by Fakespace [9], consists of flexible cloth gloves augmented with conductive cloth sewn into the tips of each of the fingers. When two or more pieces of conductive cloth come into contact with another one, a signal is sent back to the host computer indicating which fingers are being "pinched". It is distinct from whole-hand glove input devices, which report continuous joint angle information used for gesture or posture recognition.

Because of this difference, the Pinch Glove is suited only for applications involving gestures with any combination of two to ten fingers, all touching another one.

The 5DT Data Glove, developed by Fifth Dimension Technologies [1], is a Lycra Glove with fiber optics flex sensors attached along the fingers to generate finger-bend data. It is capable of measuring finger flexion with 8 bits of resolution per finger. Newer versions incorporate a tracking sensor attached in the back of the glove, measuring pitch and roll of the hand. Its sampling rate (200Hz) makes it suitable for real-time applications. The DG5-VHand, manufactured

by DGTech [8] is very similar to the 5DT Data Glove. Differences consist in the resolution of the finger sensors (10 bit) and the sampling rate of 100 MHz. Besides, the control board is connected to the PC and can be detached from the glove, making it possible to attach it to other parts of the body of the user.

The Fingertracking is a commercial video tracking system developed by A.R.T. GmbH [2]. Its performance is about 20 Hz when only three fingers are tracked and 12 Hz when five fingers are tracked. Fingertracking is less invasive than our approach, but costly and hardly depending of a specific environment set up. It uses four cameras fixed in the environment and this is not always possible or desirable. On the other hand, our proposal intended to be low-cost and independent of the environment setting (a single camera is attached to the user's hand).

Finally, the ShapeHand, developed by Measurand [17], and the CyberGlove II, by Immersion [13], are the most sophisticated and expensive glove-based solutions. They are wireless, which provides more comfort and freedom of motion, are capable of measuring fingers flexion and adduction, and able to track hand movements with six degrees of freedom. They can be easily integrated with full-body motion capture systems. Despite the large set of possibilities, only the most expensive gloves satisfy special cases where joint flexion and adduction information for each finger are needed. In this paper, we propose a computer-vision-based solution for fingers gestures input, which is explained in the next sessions.

3. Image-Based Data Glove

The image-based data glove is a visual tracking system that estimates the 3D position of finger tips and, in turn, uses this information to estimate the flexion and adduction of the fingers using inverse kinematics. Figure 1 shows the current IBDG prototype, where a camera is used to track visual markers attached to each finger of the user's hand. In order to build such a system, three main problems need to be solved:

1. The data glove (hardware) assembling (Section 3.1);

2. The tracking system (Section 3.2); and

3. The virtual hand representation and animation with inverse kinematics (Section 3.3).

These problems were solved by integrating a simple hardware design with three well known libraries: ARToolkitPlus [23] to make the real-time tracking of the markers, V-ART [11] to create the virtual environment and load the hand model, and Blender inverse kinematics module [3] to compute flexion and adduction of the fingers based on their tip positions. In the next sub-sections we describe each part of this integration.

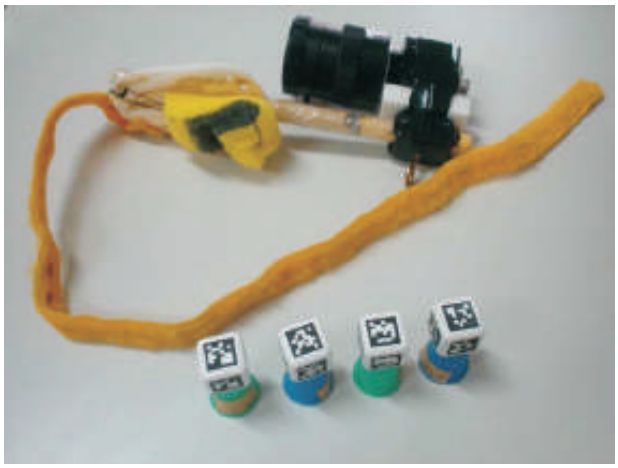


Figure 2. The Image-based Data Glove Prototype. The camera support (a wooden stick) is attached in the hand palm of the user with a Velcro strip (yellow). Thimbles are placed in fingers tips, while the camera is located in the other extremity of the stick, 23 cm far from the fingers tips.

Figure 3. Dices with ARToolkitPlus patterns. Since the thumb is not taken into account in the current data glove prototype, there are only four dices.

3.1 Prototype engineering

The IBDG hardware is composed by a marker attached to each finger tip and a firewire camera attached to the user's palm hand. Figure 1 shows an user wearing our current IBDG prototype (without the camera connecting cable), while Figure 2 shows the prototype disassembled. In our current prototype, the markers are glued to 15 mm dice attached thimbles (see Figure 3). At the faces of each dice, we have square visual patterns with 10 mm edge. The edges of the visual patterns are 5 mm smaller than the edges of the dices because we must guarantee a white margin around the pattern (a common requirement of many tracking systems).

In order to avoid nger misclassication, we assign a different visual pattern to each one of them and

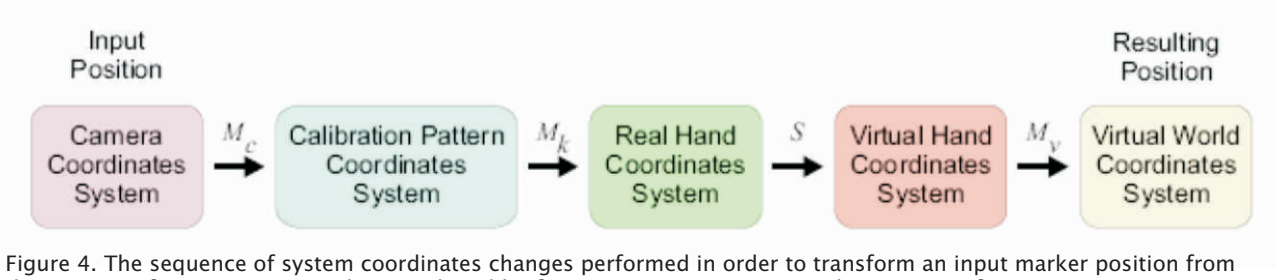


Figure 4. The sequence of system coordinates changes performed in order to transform an input marker position from the camera reference system to the virtual world reference system. M_c, M_k, S and M_v are transformation matrices described in Equation 2.

replicate the pattern of the finger to all the faces of its dice. By gluing the same pattern to all the faces of a dice, we guarantee that the camera will always see at least one marker per finger, unless it is occluded by other finger. Fortunately, the occlusion problem is easily handled by analyzing the path of the fingers along time.

One should notice that we do not track the thumb, but the remaining four fingers. It is because the field of view of the current camera in use has issues in getting some thumb positions. This is, in fact, a limitation of our current prototype and not of the IBDG concept. Section 6 discusses some possible solutions for this drawback. All fingers need to be in the camera's field of view and all markers must be in focus all the time. In order to satisfy these requirements, we placed our current camera some centimeters behind the user's palm (Figure 1). By doing so, the camera can see the markers from closed hand to opened hand gestures, fingers do not touch the lenses, and markers are always in focus.

Our current prototype uses a Flea camera from Pointgrey Research [19]. The Flea model is a compact IEEE- 1394 based digital camera with 30 x 31 mm size that is able to capture high quality images (1024 x 768 pixels) at 30 fps, being an ideal choice for demanding imaging applications. We are also using 3.5-10.5 mm Computar Varifocal Lenses [5]. The camera and the lenses set together have 115 g weight. In order to avoid both radial and tangential distortions introduced by the lenses, we estimate the intrinsic parameters of the camera and its distortion coefficients using the Camera Calibration Toolbox for MATLAB [4] and compensate such distortions using a lookup table. Notice that the estimation of the intrinsic parameters must be performed once.

3.2 Tracking system

The tracking system is responsible for estimating the 3D position and orientation of the visual patterns and, from them computing the position of the finger tips.

The tracking system we chosen for the prototype is the ARToolkitPlus library [23] and the pattern images used (Figure 3) were retrieved from the BCH Id-encoded patterns collection of this library. We are using ARToolKitPlus since we have verified it is more accurate and precise to estimate the position and orientation of tracked patterns than other well known tracking solutions, as the ARToolkit library [14]. Additionally, ARToolkitPlus is more robust than ARToolKit against false identification of patterns due to a CRC algorithm to restore damaged pattern images instead of performing just a comparison, as the simpler version does [6].

It is important to notice that the tracking library retrieves the position and orientation of the pattern that is glued over the dices. Also, the retrieved information is related to the camera reference system. However, we want to estimate the location of the markers (i.e., the center of the dice) in the coordinate system of the virtual world. Therefore, first we need to estimate the markers position in the camera reference system and then perform the related coordinates transformations in order to place them correctly into the virtual environment.

The position P of a marker expressed in the camera coordinate system is computed as

$$P = Q - d\,N \qquad (1)$$

where Q is the position of the center of the pattern; N is the normal vector of the pattern identified by the ARToolkitPlus; and d is half the size of the edges of the dice. With the markers position, we need to estimate its location in the virtual world. Therefore, some system coordinates transformations must be performed, which are illustrated by Figures 4 and 5 and summarized in Equation 2

$$P^1 = M_v\,S\,M_k\,M_c\,P \qquad (2)$$

where P0 is the position at the virtual world. M_c is the matrix that models the camera position and orientation relative to a calibration pattern and M_k is the reference

system of the calibration pattern relative to the real hand reference system (M_h in Figure 5). By assuming that the real and the virtual hand reference systems are the same, up to a scale factor, we can dene a scale matrix S that gives the relation between the real and the virtual world. Although there exist hands of different sizes, we did not experience any kind of problem by empirically setting S just as an approximation for each user. Once in the virtual hand reference system, we transform the marker to the virtual world using the M_v transformation matrix (Equation 2).

The coordinate system transformation just described depends on the use of a calibration pattern (see the second stage of the pipeline shown in Figure 4) and the execution of a calibration section, which defines the relation between the camera and the real hand reference system. In such calibration, the user must perform a hand gesture (see Figure 5) where the relation between the reference system of the calibration pattern (M_k) and the reference system of the real hand (M_h) is known (e.g., M_k can be approximated using a ruler). By doing so, the computation of the M_c is performed using:

$$M_c = M_p^{-1} \quad (3)$$

where M_p is the reference system of the camera relative to the pattern. Fortunately, M_p is retrieved by ARToolkitPlus, when the pattern is identified. It is important to notice that the calibration procedure needs to be executed only once for each user, before he/she starts to use the IBDG in a real task.

3.3 Hand model

Information about the finger joints is not retrieved by the tracking system. Therefore, being able to reproduce exion and adduction movements are of paramount importance to make the representation of the gesture of the user more realistic. We solved this problem by using an inverse kinematics tree [12] independently for each finger. The root node is on the base of the finger and the action node is on the center of the dice. Inverse kinematics is solved every frame and updates joints transformations from the prior computed pose using the current markers position. For the virtual hand representation and animation, we used the hand model from V-ART toolkit [11]. Since VART does not implement inverse kinematics techniques yet, we created a wrapper that integrates the Blender Kinematics Module [3] to its structure.

4. Evaluating the Device

In order to evaluate the proposed device, we have built a testbed application (Section 4.1) and performed some tests (Section 4.2) involving 15 subjects. Our population was heterogeneous, consisting of 2 women and 13 men, of which 13 right handed and 2 left handed, aging from 21 to 38 years old. Each user tested the prototype imitating 6 hand gestures. The position of each tracked marker was recorded automatically by the application, while data about the users and their satisfaction was collected through a simple questionnaire.

4.1 Testbed application

The testbed application was written in the C++ using OpenGL. We used V-ART toolkit to load and display the hand model, Blender Kinematics Module 1.8 to compute joints transformations, and ARToolKitPlus 2.1 to track the visual markers. Figure 6 shows the application task window, where two virtual hand models are displayed side-byside. In the left side is the test goal, a specified gesture of the hand that the user must imitate, and in the right side is the virtual hand that represents the actual tracked gesture of the user.

The goals of this testbed application are: (i) to verify the quality of the finger tips tracking; (ii) to verify the gesture reproduction; and (iii) to collect some impressions of the users while using our first IBDG prototype. Although users can freely move their arms during the tests, at this moment we are not concerned about the hand position and orientation tracking. The study of the combination of fingers and hand tracking is not covered by this paper.

4.2 Tasks and procedures

A pre-test form was answered by the users, which provided us with the following information about the subjects: 66.67% of the users have already had experiences using data gloves and all of them are used to perform activities that demand greater motor coordination of the fingers. When the application starts, the calibration section begins and the current user must align his hand with the calibration pattern in order to get the correct position and orientation of the camera with respect to the reference system of the hand (Section 3). After that, he/she presses the space bar key and the training section begins. During the training section the user is free to perform any movement with his/her hand and see what happens in the system. When he/she becomes familiarized with the device the real tasks begin. Figure 6 shows a task screen, when two virtual hand models are displayed side-by-side. The hand on the left is the pose that the user must imitate, and the one on the right is the virtual hand controlled by him/her. When the user believes

Figure 6. Testbed application showing a task executed by the user. The right hand corresponds to the hand of the user when using the data glove. The left hand means the computer controlled hand holding a pose which must be imitated by the user.

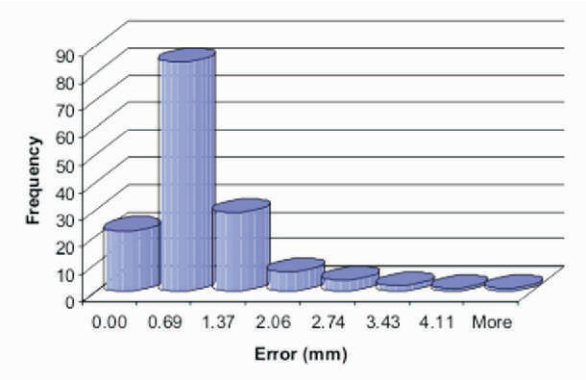


Figure 7. Observed error histogram of tracked markers positions. The mean observed error was 0.6 mm. These results show that the tracking scheme is precise.

that the pose of his/her hand is equivalent to the left one, he/she presses a key and holds his/her hand in that pose for five seconds and then goes to the next task. Notice that is the user who decides when the pose of its virtual hand is equivalent to the suggested pose. The order of the tasks is previously shufed by the application.

A pos-test questionnaire was answered by the users. It contains questions about the comfort, the easiness and the precision of the device. The collected information is discussed in Section 5.

5. Results

Using the data collected from the tests described above we measure the precision (Section 5.1) and the usability (Section 5.2) of the proposed device.

5.1 Precision analysis

In order to measure the precision of the IBDG prototype in an usual situation, we logged the tracked markers position while users hold their hands in a suggested pose (Section 4.2). For each hand pose of each volunteer, the collected data of each finger marker position was used to compute the mean marker position. Then, we estimated the mean position variation along time as the mean distance between the mean position and the observed markers positions plus a confidence interval. The observed error for a marker position is computed as:

$$e_i = \bar{d}_i + 2{:}33\ \sigma_{\bar{d}_i} \qquad (4)$$

where d_i is the mean distance between the mean marker position and the observed data; 2.33 is a Student's-t variable with n -1 degrees of freedom, such that the probability of a measure d belongs to the confidence interval is 99%; n is the observations number; and $\sigma_{\bar{d}_i}$ is the standard deviation. By computing the histogram of the observed error (Figure 7) we notice that the most frequent error is smaller than 2.06 mm and the mean observed error is 0.6 mm.

This data prove the tracking precision of the proposed device, since the error is a fraction of a millimeter.

5.2 Usability analysis

The mean latency time of the system is about 237 ms. It is bounded by the imaging process step (i.e., ARToolKit-Plus procedures). The system response can be improved by using a faster computer. All measurements were made on a 2.4 GHz PC with 2 Gb of memory. We verify that special care must be taken in order that hands of different sizes can be handled by the first prototype of the IBDG. For instance, the focus interval of our current lenses configuration can not handle hands longer than 21.5 cm (i.e., measured from middle finger tip to the beginning of the wrist). Actually, we had to reject two volunteers because their hands were too big for the current prototype.

Regarding comfort, the volunteers pointed out that our current prototype can be classified as acceptable, with the value 2.73, where 5 means comfortable and 1 means uncomfortable. Also, they classify the precision of the system as 2.87, where 5 mean precise and 1 means imprecise. However, we beleave that those concepts can be further improved. We can achieve better precision by re-calibrating the degrees of freedom of the inverse kinematics joints structure. Also, the comfort can be enhanced by replacing the current camera for a lighter one (the prototype weights 195 g, without the firewire cable, where about 58.97% of the weight came from the camera and the lenses set) and by changing the dices by smaller ones. All tested webcams have less weigh than the official camera and the newest webcams have a minor size and weigh.

6. Discussion

Illumination changes. ARToolkitPlus uses binary visual patterns. Therefore, the input image must be converted to black and white before the detection procedure. Fortunately, the library performs an automatic threshold selection before the color conversion. As a result, we did not experience any kind of problem regarding to changes of the lighting conditions nor the projection of shadows over the visual patterns. For dark places, one could use an infrared enabled camera.

Occlusion problem. Occlusions happen in two situations: (i) when the hand is completely open and the camera can not see the patterns; or (ii) when a finger is in front of other. In the current prototype, when an occlusion happens, the fingers position and orientation is set to the last valid captured values to that finger. We have achieved good results with this naive solution.

A better solution for handling occlusions is the use two or more cameras like in Fingertracking system [2]. However, it is costly then our proposed single camera approach (our intention is to build a low-cost device) and the cameras must be fixed in the environment. Notice that the dependence of the environment setting is not always possible or desirable.

Another solution is to predict the fingers position and orientation by analyzing the path of the fingers along time and DOF limits. Since the fingers perform only exion and adduction it is easy to predict where the finger tips are going from one frame to another. We believe that this solution is suitable to be incorporated in our system.

Reproducibility. When asked about the reproducibility of the device, 80% of the 15 volunteers (all students of computer science) said that, given the technical specifications, they can build their own IBDG.

Performance. The performance of the IBDG is bounded by the frame rate of the camera. In our first prototype the camera captures 30 frames per second, however we experienced a lower frame rate (about 13 fps on a 2.4 GHz PC with 2 Gb of memory). The bottleneck of our implementation is the image processing procedures of the ARToolkitPlus. We notice that, by setting a fixed threshold for the color conversion and reducing the resolution of the captured image from 1024 x 768 to 640 x 480 pixels, the performance is greatly improved (about 23 fps) but the identification of the patterns is compromised at gazing angles.

Improving comfort. By replacing the current camera by a lighter one (e.g., a micro camera or a webcam) the comfort can be improved. A wireless or bluetooh enabled camera and a lighter support would help. Some cameras with large field of view can be positioned closer to the users hand, diminishing the weight as a lever. The comfort can also be improved by using the patterns glued directly on thimbles, for instance using five planar plastic faces attached on it. In our observations, the dices remove the natural interaction of the glove, making some poses (e.g., closed hand) difficult to reproduce. Another idea is the use of a latex cleaning glove and glue the patterns on each glove finger. Special care must be taken when placing the patterns in order to keep their planarity, due to some ARToolkitPlus assumptions, and orthogonality among the five sides of this new marker. The orthogonality is required for the Equation 1 to detect the center of the cube and the real position of the finger tip.

Allowing thumb tracking. The thumb moves differently from others fingers but it is possible to track it using a camera with a large field of view.

Reducing the prototype cost. The Flea camera is the most expensive component of our system and can possibly be replaced by a modern webcam.

7. Conclusions and Future Work

In this paper we proposed the idea and describe the engineering of our prototype of the Image-Based Data Glove. The information about finger joints is estimated by inverse kinematics techniques, so all gestures of an human hand can be mapped to virtual environments. The IBDG uses a single camera per hand, is proper for continuous tracking of finger tips position and can reproduce both flexion and adduction of finger joints. Tests were performed with 15 users imitating 6 hand gestures. From the results, we conclude that the current prototype must be improved in order to be more comfortable. However, it is precise and it can be easily replicated. Regarding future work we intend to investigate a different assembly, with lighter materials, stronger structures and small markers.

We believe that the IBDG idea is promising. By investigating a different assembly to the prototype we intend to build a low-cost finger tracking solution comparable (or even better) than existing commercial ones.

8. Acknowledgment

This work was partially sponsored by CNPq-Brazil (477344/2003-8) and Petrobras (502009/2003-9). We would like to thank the volunteers that test the IBDG prototype; our colleagues at UFRGS Computer

Graphics Group by encourage us and provide useful ideas to the conclusion this project; in special to Renato Oliveira by the hand model; and Andréia Schneider, Bruno Schneider, Cleber Ughini, Dalton S. dos Reis and Leonardo G. Fischer by teach us how to use the V-ART library. The authors would like to thank Microsoft Brazil for additional support, and the anonymous reviewers for their comments and insightful suggestions.

[1] 5DT. 5DT data glove. http://www.5dt.com/products/pdataglovemri.html, 2005.

[2] A.R.T. GmbH. Fingertracking. http://artracking.eu/Fingertracking.54.0.html, 2007.

[3] Blender Foundation. Blender. http://www.blender.org, 2007.

[4] J.-Y. Bouguet. Camera calibration toolbox for Matlab. http://www.vision.caltech.edu/bouguetj/calib doc, 2007.

[5] CBC (AMERICA) Corp. http://www.cbcamerica.com, 2006.

[6] W. Daniel and S. Dieter. ARToolKitPlus for pose tracking on mobile devices. In *Proceedings of the 12th Computer Vision Winter Workshop*, Feb 2007.

[7] T. A. DeFanti and D. J. Sandin. Final report to the national endowment of the arts. *Technical Report US NEA R60-34-163*, University of Illinois at Chicago Circle, 1977.

[8] DGTech Engineering Solutions. DG5-VHand data glove. http://www.dg-tech.it/vhand/eng, 2007.

[9] Fakespace Systems Inc. Pinch glove. http://www.fakespace.com/pinch.htm, 2007.

[10] S. S. Fels and G. E. Hinton. Glove-TalkII: an adaptive gesture-to-formant interface. In *Proceedings of the Computer Human Interaction*, pages 456.463, May 1995.

[11] C. M. D. S. Freitas and L. P. Nedel. V-ART: virtual articulations for virtual reality. http://www.codeplex.com/vart, 2007.

[12] M. Girard and A. A. Maciejewski. Computational modeling for the computer animation of legged gures. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniqu*es, pages 263.270. ACM Press, July 1985.

[13] Immersion Corporation. 3D interaction products. http://www.immersion.com/3d/products/cy-ber

glove.php, 2007.

[14] H. Kato. ARToolKit home page. http://www.hitl.washington.edu/artoolkit, 2007.

[15] F. Kuester, M. A. Duchaineau, B. Hamann, K. I. Joy, and A. E. Uva. 3DIVS: 3-dimensional immersive virtual sculpting. In *Proceedings of the 1999 Workshop on New Paradigms in Information Visualization and Manipulation*, pages 92.96. ACM Press, 1999.

[16] J. J. LaViola Jr. A survey of hand posture and gesture recognition techniques and technology. *Technical Report CS99- 11*, Department oc Computer Science, Brown University, 1999.

[17] Measurand. ShapeHand. http://www.measurand.com/products/ShapeHand.html, 2007.

[18] L. P. Nedel, C. M. D. S. Freitas, L. J. Jacob, and M. Pimenta. Testing the use of egocentric interactive techniques in immersive virtual environments. In *Proceedings of the INTERACT 2003, Ninth IFIP TC13 International Conference on Human-Computer Interaction*, pages 471.478. IOS Press, September 2003.

[19] Pointgrey Research. Flea. http://www.ptgrey.com/products/ea/index.asp, 2006.

[20] D. J. Sturman and D. Zeltzer. A survey of glove-based input. *Computer Graphics and Applications*, 14(1):30.39, January 1994.

[21] C. S. Ughini, F. R. Blanco, F. M. Pinto, C. M. Freitas, and L. P. Nedel. EyeScope: a 3D interaction technique for accurate object selection in immersive environments. In *Proceedings of the SBC Symposium on Virtual Reality 2006*, pages 77.88, May 2006.

[22] Virtual Realities. P5 glove: virtual reality glove. http://www.vrealities.com/P5.html, 2007.

[23] D. Wagner. ARToolKitPlus home page. http://studierstube. icg.tu-graz.ac.at/handheld ar, 2007.

[24] T. G. Zimmerman, J. Lanier, C. Blanchard, S. Bryson, and Y. Harvill. A hand gesture interface device. In *Proceedings of the SIGCHI/GI conference on human factors in computing systems and graphics interface*, pages 189.192. ACM Press, 1987.

Direct 3D Manipulation Using Vision-Based Recognition of Uninstrumented Hands

Siniša Kolarić, Alberto Raposo, Marcelo Gattass
Tecgraf - Computer Graphics Technology Group - Department of Informatics
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro - RJ, Brazil
{skolaric}, {abraposo}, {mgattass}@tecgraf.puc-rio.br

Abstract

We describe a prototype of an interactive Mixed Reality application for direct spatial manipulation, based on the vision-based recognition of uninstrumented hands. In our application, we dynamically integrate both of the user's hands into the virtual environment, effectively creating a logical 2 × 3 DoF manipulation device. Using this logical device, we were able to implement, in conjuction with hand gestures, a number of fundamental 3D manipulation operations like select, deselect, translate, rotate, and scale.

1. Introduction

Most current 3D editors and viewers still operate within the WIMP (Windows, Icons, Menus and Pointing device) paradigm, thus utilizing the mouse and related devices, as well as various graphical metaphors, to manipulate 3D geometric objects. For example, to rotate an object, the user usually has to select that object with the mouse, then select the ROTATE tool from the menu or toolbar, and finally move (drag) the mouse with a mouse button pressed and the corresponding graphical manipulator widget activated. The planar (2 DoF) movement of the mouse thus gets translated into a 3D angular range constrained to a rotation plane, causing the object to rotate in the associated global coordinate system.

While the WIMP approach is a tried and familiar one, it coerces the user into performing intrinsically threedimensional operations like TRANSLATE, ROTATE or SCALE by means of a proxy planar mouse operation, which is dissimilar to the way humans actually perform these operations in the physical world, doing free-form hand movements. Given the current state-of-art in computing (especially computer vision), would it be possible to push the technological envelope just a little bit, and build a practical computer-vision based system that is capable of performing 3D manipulation operations in the most natural way, that is, using a user's own hands directly?

In this paper we describe how we built a prototype of such system, capable of manipulating 3D virtual objects by means of a small set of manipulating operations, utilizing a standard PC, a stereo pair of low-resolution cameras and a software application based on computer vision techniques. Using this technology, we integrated our hands into the virtual workspace, thus in effect creating a logical 2 × 3 = 6 DoF input device, capable to manipulate 3D objects in our virtual environment.

Our exposition is structured as follows: Section 1 (current) explains the motivation for this work. Section 2 gives an overview of the related work. Section 3 describes the user's workplace, and explains how we defined the operations for manipulating 3D geometric objects using uninstrumented hands. Section 4 describes technical details of how we utilized computer vision (CV) techniques to determine positions of certain hand features, and to recognize hand gestures, as well as describes experimental results. Section 5 discusses experimental results and future work.

2. Related work

We split our survey of related work into two principal parts: Section 2.1 gives a survey of the work done in the field of direct 3D manipulation, and Section 2.2 gives an overview of the relevant computer-vision techniques for hand recognition.

2.1. Direct 3D manipulation

The expression "direct manipulation", coined by Shneiderman [43], describes an interaction modality where the user can modify computational objects using actions that correspond at least loosely to the physical world. For the purposes of this exposition, we define "direct 3D manipulation" as a special direct manipulation type which:

• deals with manipulation of virtual 3D geometric objects,

• uses free-form hand movements for spatial input [16], and

• has a minimal (or equal to zero) spatial displacement between the user's physical hand (and of its virtual representation) and the manipulated virtual 3D object.

Systems capable to manipulate geometry include Sutherland's Sketchpad [45], and Clark's "Designing surfaces in 3-D" [8]. Bolt's "Put-that-there" system [3] uses a 6-DoF tracker and hand gestures, together with speech input, to manipulate simple shapes on a large, wall-sized screen.

Sachs et al present "3-Draw" [37], a 3D computer-aided design tool capable of drawing complex free-form shapes, using two 6-DoF sensors (one sensor to control 3D drawing and editing tools, and the other sensor to control an object's position and orientation). Krueger's VIDEODESK [22] uses overhead video cameras to track 2D hand positions, and is capable of manipulating splines by modifying their control points using index fingers and thumbs of both hands. Butterworth et al present 3DM [7], an interactive surface-modeling system that uses one single bat (3D mouse) with 6 DoF. Shaw and Green present THRED (Two-Handed Refining EDitor) [40], a free-form sketching system that uses two three-button bats, one for each hand (the dominant hand picks and manipulates 3D objects, and the less-dominant hand sets context). The JDCAD system [24] by Liang and Green uses a 6-DoF bat for spatial input. Deering's HoloSketch [12] is a VR system for 3D geometry creation and manipulation that uses head-tracked stereo glasses and a 3D mouse/wand ("one-fingered data glove") augmented by an offset digitizer rod, effectively making it a six-axis wand. Mapes and Moshell present PolyShop [26], which uses two ChordGloves (datagloves which have electric contacts on fingertips and on the palm) for bimanual spatial input. Mine's CHIMP (Chapel Hill Immersive Modeling Program) [28] uses two separate bats, one for each hand. User can perform a unimanual operation for translations and rotations, and a bimanual symmetric movement for scaling. Cutler et al present [9] two-handed direct manipulation on a "Responsive Workbench" [23], using pinch gloves, stylus providing single dinstinguished point of action, and a 6 DoF tracker attached to stereo shutter glasses for head tracking. Nishino et al [30] present a system using one- and two-handed gestures (deform, grasp, point, scale, rotation) to model and manipulate 3D objects, using data gloves. Schkolne, Pruett and Schr¨oder present "Surface Drawing" [39], [38] where shapes, drawn using wired datagloves, "float" in the space above Responsive Workbench. The FingARtips technique [6] by Buchmann et al tracks hand gestures by using image processing software and finger- and hand-based fiducial markers, and allows users to interact with virtual content using natural hand gestures. Bettio et al [2] present a system where the user stands in front of a large stereo display, and manipulates the model using optically tracked unmarked hands.

In the field of 3D interaction techniques, Strauss and Carey [44] describe graphical manipulators, like trackball, one-axis scale, jack, handle box, and one-axis translate. Mine [27] discusses virtual environment interaction including movement (specifiying direction and speed), selection (local and at-a-distance), manipulation (change in position, orientation and center of rotation) and scaling (center of scaling and scaling center; uniform and non-uniform scaling). Poupyrev et al showcase the Go-Go technique [33], for non-linear mapping for direct manipulation in Ves. Bowman and Hodges [4] evaluate techniques for grabbing and manipulating remote objects in virtual environments. Mine et al [29] address the lack of haptic feedback in VEs and propose to use the sense of proprioception. Zhai [48] reviews the usability of various 6-DoF input devices. Dachselt, Hinz and Huebner give a classification of 3D widgets [10] [11].

For hand postures and hand gestures, Quek [35] [34] and Pavlovic et al [32] give reviews in the context of HCI. For two-handed gesture, Guiard [14] gives a theoretical framework for the study of asymmetry in the context of human bimanual action.

2.2. Computer vision for hand recognition

In our work, we are interested in non-contact tracking of *uninstrumented* (i.e. unmarked, bared) human hands, and for this we use passive computer vision techniques. Therefore, we aren't considering more traditional methods like using colored gloves or attaching rectangular patterns onto hands, or using cybergloves. Also, we aren't considering active computer vision techniques, which use light projectors.

2.2.1. Vision–based hand detection.

Vision-based hand detection is a process that returns a negative answer if no hand has been detected in an image, and a positive answer otherwise. If detected, most hand detection methods will also return the location (for example, a bounding rectangle) of the hand in the picture. In our work, we use hand detection to initialize hand tracking (see Section 2.2.2 for an overview of hand tracking literature, and Section 4.4 for our hand tracking implementation). Detecting human hand in an arbitrary image is a difficult problem. Because of this, many past approaches adopted various constraints on experimental setups which made the tracking problem

easier, like constant backgrounds, markers, non-existence of other skin-colored objects in view, and colored gloves.

The object detection method that has taken the CV community by storm, and which we adopt in our work, was due to Viola and Jones [47]. This method is robust, fast and reliable, and can be trained to recognize any kind of object. It uses AdaBoost [13] learning algorithm to combine weak classifiers (those with at least (50 +)% of guessing probability, where is arbitrarily small) into strong, arbitrarily accurate classifiers. The application of Viola-Jones method to detecting specifically hands has been researched by K¨olsch and Turk in [21] and Ong and Bowden in [31].

2.2.2. Vision-based hand tracking. *Tracking* is the process of estimating the position of a tracked object, taking its previous position into consideration. Since the hand is an articulate 3D structure which gets projected onto the 2D image plane, we can choose to track the hand either in its original 3D space (in this case we say that we employ model-based hand tracking), or in the 2D projection image plane (appearance-based tracking). We can also talk about hybrid tracking, a recent mode of tracking which combines elements of model- and appearancebased tracking.

• Appearance-based hand tracking — various tracking methods in this class differ in what cues they use for tracking—some, for example, use just one cue like skin color or hand motion, and other use a combination of cues (for example, skin color and motion). For example, in Camshift [5] just the hand's color is being used as a cue; in CONDENSATION [17], hand contours + hand motion, and in ICONDENSATION [18] hand contours + hand motion + skin color; in "Flock of Features" [20] the combination skin color + KLT features is being used [41], [46] (see Section 4.4 for more on KLT features).

• Model-based hand tracking — here the backprojection of a predefined 3D parametric hand model is being matched against the input video frame. At each frame, extracted features are being compared with the current 3D model, and the matching error computed; if the error is too large, the 3D model is adjusted in the attempt to decrease the error—if the error is still too big, we repeat the model adjustment, otherwise we found a good matching and the tracking was successful. Examples include the classic DigitEyes system [36], where a 27-DoF hand model is being tracked.

• Hybrid hand tracking — here elements of both the model-based and appearance-based tracking are combined in an effort to get the best of two worlds. Shimada et al in [42] and Athitsos and Sclaroff in [1] synthesize a large number of 2D views of a software 3D hand model, and tag each of these views by the corresponding, exact hand pose vector. After this preprocessing step, appearance-based matching methods are used to process real images.

2.2.3. Human skin color modeling and detection. Human skin color is an important cue that can be useful for both hand detection and hand tracking; it can be used either standalone (i.e. on its own) for both detection and tracking, or as a helper method, i.e. as a means to increase the robustness of other detection and tracking methods. In their recent work [19], Kakumanu et al give a detailed and comprehensive survey of human skin color modeling approaches, as well as of human skin detection methods.

2.2.4. Stereo 3D reconstruction. By expression stereo 3D reconstruction we refer to the process and techniques for determining 3D position and 3D structure from pairs of correspondences in the left and the right image of the input stereo stream. For an overview of triangulation methods, see Hartley and Sturm [15].

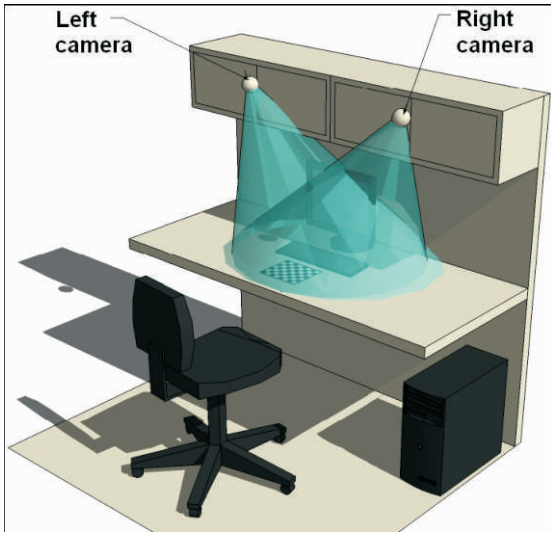


Figure 1. The user's workplace.

3. Direct 3D manipulation using uninstrumented hands

3.1. User's workplace

The workplace (Figure 1), as we define it, consists of a standard office cubicle equipped with a personal computer with two cameras (i.e. a stereo pair)

connected. Cameras are fixed at the top of the cubicle and are directed down, at a certain angle, relative to the surface of the desktop. A stereo pair of cameras enables us, due to the phenomenon called stereo disparity, to estimate 3D positions of various hand features, thus offering us a way to integrate our hands into the VE.

Accordingly, we define the workspace as the intersection of the two visual cones defined by the respective cameras' fields of view—the user must move his hands in this working space, in order for the system to register hand movements and gestures. If a hand exits the workspace, the system stops tracking the hand.

3.2. Defining hand postures

Figure 2 depicts the three hand postures we use in our application. Note that the image depicts the right hand only, but both the left and the right hand can assume these postures. (The left hand assumes postures which are simply mirrored relative to the vertical axis.) Also please note that the hand postures shown in the image are inclined at an angle, which approximates the natural hand inclination when it is being tracked in the workspace. We now define the following hand postures (also called hand states), from which all the manipulation operations are being defined (see Section 3.3), as follows:

1. `HAND_POSTURE_OPEN` — flat palm with all fingers spread apart

2. `HAND_POSTURE_POINTING` — all fingers closed, except the index finger

3. `HAND_POSTURE_FIST` — all fingers closed

As a matter of convenience, we defined one more posture, `HAND_POSTURE_UNKNOWN` , which designates any hand posture that is not recognized by the system.

3.3. Defining direct 3D manipulation operations

Using the hand postures defined above, we now define direct 3D manipulation operations. Manipulation operations can be either one-handed or two-handed:

1. `OP_SELECT` (one-handed) — selects an object. Based on the posture `HAND_POSTURE_POINTING`.

2. `OP_DESELECT` (one-handed) — deselects an object. Based on the posture `HAND_POSTURE_POINTING`.

3. `OP_TRANSLATE` (one-handed) — translates

(moves) selected objects. Based on the posture

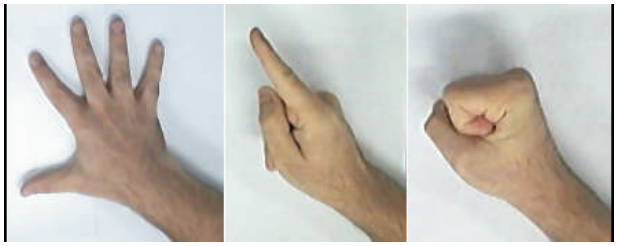


Figure 2. Three hand postures utilized by the system: `HAND_POSTURE_OPEN` (left), `HAND_POSTURE_POINTING` (middle) and `HAND_POSTURE_FIST` (right).

5. `OP_SCALE` (two-handed) — scales objects. Based on two `HAND_POSTURE_FIST` hand postures.

Therefore, we have two two-handed spatial operations: `OP_ROTATE` and `OP_SCALE` — these use both hands at the same time. The remaining three spatial operations (`OP_SELECT, OP_DESELECT` and `OP_TRANSLATE`) are one-handed — must be performed by just one hand, either just by the left or just by the right hand. We will now describe each of these operations in detail. 3.3.1. Selecting and deselecting objects.

Operation `OP_SELECT` selects an object in the scene, while operation `OP_DESELECT` deselects an (already selected) 3D object. In order to (de)select a 3D object, user extends the index finger of one and exactly one hand (thus changing that hand's state into `HAND_POSTURE_POINTING`), and moves the hand into the object. (We emphasized "one and exactly one" because two tracked hands in state `HAND_POSTURE_POINTING` perform the operation `OP_ROTATE` , see Section 3.3.3.)

As soon as the application detects that the tracked hand's centroid entered the interior of the object, while the hand is in state `HAND_POSTURE_POINTING` , the objects gets (de)selected. The user can now move her hand out of the object; the object stays (de)selected.

In WIMP terms, operation `OP_SELECT` is equivalent to moving the mouse pointer onto a screen object (for example, onto an icon) and then pressing the mouse button, thus selecting the icon. Similarly, operation `OP_DESELECT` is equivalent to moving the mouse pointer onto an already selected screen object (for example, onto an already selected icon) and then pressing the mouse button, which deselects the icon.

3.3.2. Translating objects. Operation `OP_TRANSLATE` translates (moves) all the currently selected objects (Figure 3). For this to work, one and exactly one hand must be in the

HAND_POSTURE_FIST state (We say "exactly one" because if both tracked hands are in the HAND_POSTURE_FIST state, we will be performing the OP_SCALE operation, see Section 3.3.4.).

The operation initiates at the moment the user changes the state of one (and exactly one) of her hands into HAND_POSTURE_FIST. The position of that hand's centroid is then the starting point of the translation vector. User moves the hand about, and the selected objects move along too. When the user decides the translation is just right, she changes the hand's state into HAND_POSTURE_OPEN thus terminating the OP_TRANSLATE operation.

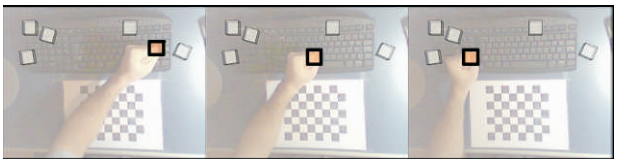


Figure 3. OP_TRANSLATE operation, based on one HAND_POSTURE_FIST posture.

3.3.3. Rotating objects. Operation OP_ROTATE rotates all the currently selected objects (Figure 4). At the moment when the application detects that both hands have their index finger extended (thus entering into the HAND_POSTURE_POINTING state), the hands' respective positions get memorized and defined as two points *A, B* of the initial rotation axis *AB*, with rotation center *C* at the middle point between those two points:

$$C = \frac{AB}{2}.$$

The user can now move her hands about (all the while both hands stay in the HAND_POSTURE_POINTING state), and the selected objects rotate too around C. When the user decides to stop the rotation, she changes both hands' state into HAND_POSTURE_OPEN thus terminating the OP_ROTATE operation.

3.3.4. Scaling objects. Operation OP_SCALE scales all the currently selected objects (Figure 5). At the moment both hands enter into the HAND_POSTURE_FIST state, the centroids of both hands get memorized and defined as two points A,B of the initial scaling axis AB, with center C at the middle point between those two points:

$$C = \frac{AB}{2}$$

Now the user can move her hands (all the while both hands stay in the HAND_POSTURE_FIST state), and the selected objects scale in real-time around *C*, in the directions defined by three axes inferred from *AB*.

Figure 4. The two-handed OP_TRANSLATE operation is based on two HAND_POSTURE_POINTING postures. An example of CCW rotation is shown.

When the user decides to stop the scaling, she changes one (or both) hand's state into HAND_POSTURE_OPEN thus terminating the OP_SCALE operation.

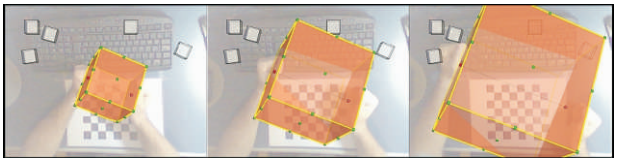


Figure 5. The two-handed OP_SCALE operation is based on two HAND_POSTURE_FIST postures.

4. Vision-based hand recognition

In this section we describe how we used computer vision techniques in order to detect, track and recognize hands and their gestures in the workspace defined in Section 3.1.

4.1. Calibrating the stereo rig

Before anything, the stereo rig must be calibrated i.e. its parameters (intrinsic and extrinsic) determined. By knowing these camera parameters, CV techniques we adopted can reconstruct 3D position of our hands in the workspace, using any triangulation method, for example the Hartley- Sturm method [15].

The set K of intrinsic parameters for a single camera includes two focal lengths (f_x, f_y), the principal point (o_x, o_y) and four distortion parameters (k_1, k_2, k_3, k_4):

$$K = \{(f_x, f_y), (o_x, o_y), (k_1, k_2, k_3, k_4)\}$$

We determined these intrinsic parameters using the Zhang's method [49]. For the calibration pattern we used a 8 × 7 checkerboard pattern.

Having determined two sets K_L, K_R of intrinsic parameters (for the left and right camera), we can proceed to determining extrinsic parameters (orientation and distance of a camera relative to the pattern). For this, we now fix the 8 × 7 checkerboard on the desk surface, and orient cameras so that they both have the pattern in their field of view. Since we already know both cameras' intrinsic parameters, we can now use a function from the OpenCV1 library to compute just the extrinsic parameters (rotation matrices

R_L and R_R, and translation vectors $\vec{T_L}$ and $\vec{T_R}$) of both cameras, relative to the pattern we've just fixed on the desk's surface. The extrinsic parameters R and $\vec{T}$ of the stereo rig are then simply

$$R = R_R R_L^{\tau} \qquad \vec{T} = \vec{T_L} - R^{\tau}\vec{T_R}$$

These parameters R and $\vec{T}$ now completely determine the geometry of our stereo rig and allow us to perform absolute, Euclidean 3D reconstruction of the hand's 3D position in the workspace shown in Figure 1.

4.2. Hand detection

With the stereo rig calibrated, we can now proceed to detecting hands in the stereo input video stream. Here detection serves for the purpose of initiating the process of tracking, described in Section 4.4 below.

For this end, we define two "detection areas" (left and right), one for each of both hands in the application client area. By definition, if a hand is not being tracked, its "detection area" gets shown on the application screen as a red rectangle, at the predetermined location and with a predetermined size. If the user now moves her hand into the corresponding detection area, and puts her hand into the predefined posture (we chose HAND POSTURE OPEN as the tracking initialization posture), the system will detect the hand, output the corresponding bounding rectangle and start tracking the hand within this bounding rectangle.

As we've already mentioned, we chose the Viola-Jones method as our detection method, which requires training and validation using four sets of samples:

• Two positive sample sets:

  – **Positive training set A** - for this set we moved our right hand in posture `HAND_POSTURE_OPEN` randomly in the workspace, approximately under the natural inclination (see Figure 2 left), under our lab's standard lighting conditions, and took 1000 photos containing the hand. Note that "approximately natural inclination" indicates that we included a number of shots of hands rotated to a degree relative to all three axes, in order to increase the robustness of the classifier.

  – **Positive validation set B** - under the same conditions as above, we took additional 100 photos to be used as validation images after the training is complete.

• Two negative sample sets:

  – **Negative training set C -** for the negatives, we created a set of 1000 images that do not contain hands in posture `HAND_POSTURE_OPEN` , using a public image database (CMU VASC Image Database).

  – **Negative validation set D -** we created an additional negative validation set containing 100 images, using the same image database.

We then used OpenCV facilities to train boosted cascades of weak classifiers in the following way:

1. we manually marked bounding rectangles for hands in the positive training samples, and saved the list of bounding rectangles in a file F.

2. before training, we set the required false positives threshold to be 10- 6.

3. we ran the training tool on file F and on the two training sets, the positive A and negative C. After the training has completed, we obtained a 15-stage classifier for posture `HAND_POSTURE_OPEN` with detection rate of approximately 98%.

4. we successfully tested the classifier using sets B and D.

5. we built the trained classifier into our prototype application. An OpenCV function loads the classifier; other function detects hands in posture `HAND_POSTURE_OPEN` in the current video frame. Note that we trained the classifier with our right hand; for the left hand, before the recognition stage we mirror the left side of the application area in order to be able to use the same classifier to recognize the left hand.

4.3. Hand segmentation based on human skin color

The hand detection method described above not only gives an answer whether there is or isn't a hand in an image, but also provides the bounding rectangle of the image region containing the hand. Considering this region of interest (ROI) only, we now make use of the characteristic hue of human skin to determine the pixels belonging to a hand. The reason we perform this segmentation is to increase the hand tracking robustness— see Section 4.4.

To this end, we used color histograms — both in the detection stage (using HSV color space), and for the learning (using normalized RGB histogram) of the color of the hand that has just been detected, i.e. we perform color learning immediately after the hand has been detected (pre-tracking stage).

4.4. Hand tracking

After a hand has been detected in an image, and hand pixels color-segmented using the properties of the

* www.intel.com/technology/computing/opencv/

human skin, we start tracking it. For tracking we chose the method proposed by K¨olsch in [20], which in turn is based on Kanade-Lucas-Tomasi (KLT) features, also called "good features to track". We chose this method because it is robust and can track hands in complex, cluttered environments.

KLT features are based on the early work done in [25], and then developed further in [46] and [41]. To increase robustness, the "Flocks of Features" approach to tracking by K¨olsch adds two additional properties to simple KLT tracking:

• tracked KLT features never exceed a predetermined maximum distance from the median of all tracked KLT features, and

• tracked KLT features can never be closer to each other than a predetermined minimum distance.

Differently from the application showcased by Kölsch in [20], which is able to track only one hand using just one (monocular) camera, our application: 1) implements four fully independent object trackers (four due to each camera tracking up to two hands), and 2) uses stereo disparity for 3D reconstruction of the hand's position in 3D workspace.

We now clarify what is meant by "tracking a hand". After a hand has been detected as explained in Section 4.2 in both cameras' views, and hand pixels color-segmented, up to N (for example, 100) KLT features are being collocated on the hand (i.e. on the blob defined by the detected hand's pixels). By averaging in each frame the 2D positions of all of these N features, we obtain a mean (average) position P of the hand being tracked. Therefore the 2D position P is the output of the tracking routine. Since we can have up to two active (tracked) hands, and each hand gives rise to one triangulated 3D position, we can have up to $2 \times 3 = 6$ DoF at our disposal to implement spatial manipulation operations.

3D reconstruction (triangulation). Finally, with the two corresponding 2D points u, u0 tracked (point u in the left camera view, point u0 it the right camera view, we can compute, in real time, the global 3D position x of a hand (either the left or the right hand) in the workspace. For this we use the triangulation method by Hartley and Sturm [15], a fast, non-iterative method that always finds the global optimum under the assumption of Gaussian noise present in the input images.

4.5. Hand posture recognition

The last step in the CV pipeline is the hand posture recognition, which enables us to implement a simple static gesture recognition. For hand posture recognition, we again use the Viola-Jones method. For this we repeated the training process explained in Section 4.2, only with positive samples containing other postures besides `HAND_POSTURE_OPEN`.

4.6. Tests

The software application was developed utilizing the C++ language, OpenCV computer vision library and OpenGL graphics library. All experiments were done on a personal computer equipped with an 2.66 GHz dual core processor, 2 GB RAM, and two web cameras connected to USB 2.0 ports grabbing 30 color frames per second at the resolution of 320×240 pixels.

Using the system described, we achieved trackingrelated latencies from 7 to 30ms with just one hand tracked (i.e. with two trackers active), and up to 60ms with both hands tracked (i.e. with all four trackers active). Taking the application as a whole, i.e. taking all the other system processes into consideration, we achived frame rates from 8 to 15 fps.

We'll now assess qualitatively estimation accuracy for a hand's position. Since the difference between hand's estimated position and ground truth is difficult to measure for an uninstrumented hand, we give here the figures demonstrating the hand's trajectory in space, from which we can deduce visually the amount of noise present in estimated positions. We trace three simple figures in space with the right hand: a line, a circle and an "eight" figure (Figure 6).

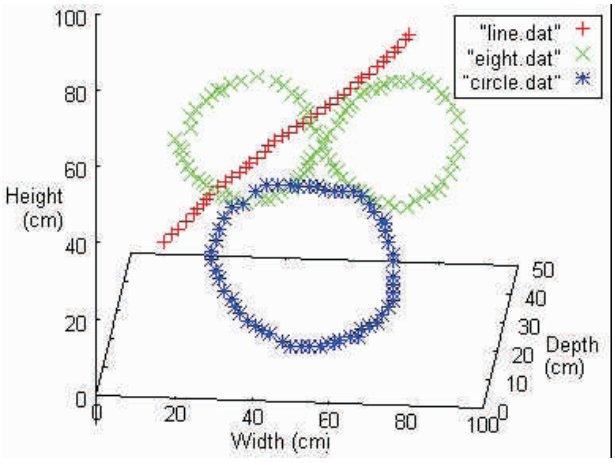


Figure 6. 3D plot of estimated hand positions, obtained by tracing a line, a circle and an "eight" in the workspace.

5. Discussion and further work

We presented an application where the user can employ intuitive manipulative hand gestures in order to manipulate 3D virtual objects. Considering the good

latencies and fps rates we achieved, our system fulfills the requirements of an interactive system.

Since we chose a robust and fast method for hand detection, we are not limited to physical setups as shown in Figure 1 — the system can perform the detection, for example, in outdoors environment too. The same holds for the tracking method we chose — due to its robustness, we can track hands in much more complex and more cluttered environments.

Due to these properties, future work includes expanding the system to work in diverse environments, and not just in the highly controlled environment shown in Figure 1. Further, we would like to increase the expressiveness of direct manipulation by including fingers into the manipulation operations. This entails tracking not only the 2-DoF point (global hand position) in the image plane, but some type of model-based tracking capable to track fingers as well. Finally, we would like to augment the set of manipulation operations by adding more complex, physically based manipulation and deformation operations, like "attract", "repel", "cut", "shear" and similar.

Acknowledgments

This work was performed at Tecgraf/PUC-Rio, a laboratory mainly funded by PETROBRAS.

6. References

[1] V. Athitsos and S. Sclaroff. 3d hand pose estimation by finding appearance-based matches in a large database of training views. *Technical Report BU-CS-TR-2001-021*, Computer Science Department, Boston University, Boston, USA, 2001.

[2] F. Bettio, A. Giachetti, E. Gobbetti, F. Marton, and G. Pintore. A practical vision based approach to unencumbered direct spatial manipulation in virtual worlds. In *Eurographics Italian Chapter Conference*, Conference held in Trento, Italy, February 2007. Eurographics Association.

[3] R. A. Bolt. put-that-there: Voice and gesture at the graphics interface. In SIGGRAPH '80: *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 262–270, New York, NY, USA, 1980. ACM.

[4] D. A. Bowman and L. F. Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Symposium on Interactive 3D Graphics*, pages 35–38, 182, 1997.

[5] G. R. Bradski. Computer vision face tracking for use in a perceptual user interface. Intel Technology Journal, (Q2):15, 1998.

[6] V. Buchmann, S. Violich, M. Billinghurst, and A. Cockburn. Fingartips: gesture based direct manipulation in augmented reality. In *GRAPHITE*, pages 212–221, 2004.

[7] J. Butterworth, A. Davidson, S. Hench, and M. T. Olano. 3dm: a three dimensional modeler using a head-mounted display. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 135–138, New York, NY, USA, 1992. ACM.

[8] J. H. Clark. Designing surfaces in 3-d. Commun. ACM, 19(8):454–460, 1976.

[9] L. D. Cutler, B. Froehlich, and P. Hanrahan. Two-handed direct manipulation on the responsive workbench. In *Symposium on Interactive 3D Graphics*, pages 107–114, 191, 1997.

[10] R. Dachselt and M. Hinz. Three-dimensional widgets revisited - towards future standardization. In *Proceedings of the Workshop 'New Directions in 3D User interfaces'*, 2005.

[11] R. Dachselt and A. Huebner. Virtual environments: Three dimensional menus: A survey and taxonomy. *Comput. Graph.,* 31(1) :53–65, 2007.

[12] M. F. Deering. Holosketch: a virtual reality sketching/ animation tool. *ACM Trans. Comput.-Hum. Interact.,* 2(3): 220–238, 1995.

[13] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[14] Y. Guiard. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model, 1987.

[15] R. Hartley and P. Sturm. *Triangulation. 68(2)*: 146–157, November 1997.

[16] K. Hinckley, R. Pausch, J. C. Goble, and N. F. Kassell. A survey of design issues in spatial input. In *UIST '94: Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 213–222, New York, NY, USA, 1994. ACM.

[17] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision,* 29(1):5–28, 1998.

[18] M. Isard and A. Blake. ICONDENSATION: Unifying lowlevel and high-level tracking in a stochastic framework. *Lecture Notes in Computer Science*, 1406:893–908, 1998.

[19] P. Kakumanu, S. Makrogiannis, and N. Bourbakis. A survey of skin-color modeling and detection methods. *Pattern Recogn.*, 40(3):1106–1122, 2007.

[20] M. Kolsch and M. Turk. Fast 2d hand tracking with flocks of features and multi-cue integration. In *CVPRW'04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04)* Volume 10, page 158, Washington, DC, USA, 2004. IEEE Computer Society.

[21] M. Kölsch and M. Turk. Robust hand detection. In *FGR*, pages 614–619, 2004.

[22] M. Krueger. *Artificial Reality II*. Addison-Wesley: Reading, MA, 1991., second edition, 1991.

[23] W. Krueger and B. Froehlich. The responsive workbench [virtual work environment]. *Computer Graphics and Applications, IEEE,* 14(3):12–15, May 1994.

[24] J. Liang and M. Green. Jdcad: a highly interactive 3d modeling system. *Computers & Graphics*, 18(4):499–506, 1994.

[25] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision, 1981.

[26] D. P. Mapes and J. M. Moshell. A two-handed interface for object manipulation in virtual environments. *Presence*, pages 403–416, 1995.

[27] M. Mine. Virtual environment interaction techniques. Technical Report TR93-005, UNC Chapel Hill, Dept of Computer Science, North Carolina, USA, 1995.

[28] M. R. Mine. Working in a virtual world: Interaction techniques used in the chapel hill immersive modeling program. *Technical Report TR96-029*, 12, 1996.

[29] M. R. Mine, F. P. Brooks, Jr., and C. H. Sequin. Moving objects in space: Exploiting proprioception in virtualenvironment interaction. *Computer Graphics*, 31(Annual Conference Series):19–26, 1997.

[30] H. Nishino, K. Utsumiya, and K. Korida. 3d object modeling using spatial and pictographic gestures. In VRST, pages 51– 58, 1998.

[31] E. Ong and R. Bowden. A boosted classifier tree for hand shape detection, 2004.

[32] V. Pavlovic, R. Sharma, and T. Huang. Visual interpretation of hand gestures for human-computer interaction: A review, 1997.

[[33] I. Poupyrev, M. Billinghurst, S. Weghorst, and T. Ichikawa. The go-go interaction technique: non-linear mapping for direct manipulation in vr. In *UIST '96: Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 79–80, New York, NY, USA, 1996. ACM. [34] F. Quek. Eyes in the interface. IVC, 13(6):511–525, August 1995.

[35] F. K. H. Quek. Toward a vision-based hand gesture interface. In *VRST '94: Proceedings of the conference on Virtual reality software and technology*, pages 17–31, River Edge, NJ, USA, 1994. World Scientific Publishing Co., Inc.

[36] J. M. Rehg and T. Kanade. Visual tracking of high DOF articulated structures: an application to human hand tracking. In *ECCV (2)*, pages 35–46, 1994.

[37] E. Sachs, D. Stoops, and A. Roberts. 3-draw: a three dimensional computer aided design tool. *Systems, Man and Cybernetics, 1989. Conference Proceedings., IEEE International Conference on*, pages 1194–1196 vol.3, 14-17 Nov 1989.

[38] S. Schkolne, M. Pruett, and P. Schr¨oder. Surface drawing: creating organic 3d shapes with the hand and tangible tools. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 261–268, New York, NY, USA, 2001. ACM.

[39] S. Schkolne and P. Schroder. Surface drawing. *Technical Report CS-TR-99-03*, Pasadena, CA, USA, 1999.

[40] C. Shaw and M. Green. THRED: A two-handed design system. Multimedia Systems, 5(2):126–139, 1997.

[41] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[42] N. Shimada, K. Kimura, and Y. Shirai. Real-time 3-d hand posture estimation based on 2-d appearance retrieval using monocular camera. ratfg-rts, 00:0023, 2001.

[43] B. Shneiderman. Direct manipulation. a step beyond programming languages. *IEEE Transactions on Computers*, 16(8):57–69, August 1983.

[44] P. S. Strauss and R. Carey. An object-oriented 3d graphics toolkit. *SIGGRAPH Comput. Graph.*, 26(2):341–349, 1992.

[45] I. Sutherland. Sketchpad: A man-machine graphical communication system. *PhD thesis, Massachusetts Institute of Technology*, January 1963.

[46] C. Tomasi and T. Kanade. Detection and tracking of point features. *Technical Report CMU-CS-90-166*, Carnegie Mellon University, USA, Apr. 1991.

[47] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2002.

[48] S. Zhai. User performance in relation to 3d input device design. *SIGGRAPH Comput. Graph.*, 32(4):50–54, 1998.

[49] Z. Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, 2000.

Session 6: Simulation and Computer Graphics

Illumination Techniques for Photorealistic Rendering in Augmented Reality

Saulo A. Pessoa[1], Eduardo L. Apolinário[1], Guilherme de S. Moura[1], João Paulo S. do M. Lima[1], Márcio A. S. Bueno[1,2], Veronica Teichrieb[1], Judith Kelner[1]

[1]Virtual Reality and Multimedia Research Group (GRVM)
Informatics Center (CIn) – Federal University of Pernambuco (UFPE)
{sap}, {ela}, {gsm}, {jpsml}, {masb}, {vt}, {jk}@cin.ufpe.br,
Science and Technology Center (CCT) – Catholic University of Pernambuco (UNICAP)
masb@unicap.br

Abstract

This paper presents a combined solution to realistically insert virtual objects into real scenes. The solution proposed focuses on the illumination problem, which was divided in three parts: estimating the radiance reaching objects' surface, through an Image Based Lighting technique; self shadowing, through an Ambient Occlusion technique; and lighting effects due to lens properties, through Bloom and Exposure Control techniques. These techniques were implemented in a single solution which obtained good visual results. As interactive performance was achieved, the solution presented is suitable to insert virtual objects into Augmented Reality applications in a photorealistic way.

1. Introduction

There is a branch in Computer Graphics rendering techniques that aims the most realistic presentation of synthetic objects and scenes. This research area is called *Photorealistic Rendering* (PR) and comprehends the production of an image that is indistinguishable from a photograph of a scene [1]. In order to accomplish this result, there are some problems that must be solved, such as tone-reproduction, illumination, among others. Although there are many methods that try to solve these problems, many are designed to run offline. Nevertheless, there is a demand for real-time techniques that can achieve similar results, and some studies managed to accomplish such feat.

The topic mentioned above has an important relation with *Augmented Reality* (AR), since one of the latter's challenges is to mix synthetic objects with real scenes seamlessly. This way, user could interact more naturally with the interface. There are many possible applications which could use this feature to increase user experience; a virtual museum or an architectural software, for instance.

This paper presents a solution which integrates some illumination techniques in order to provide a real-time realistic result for synthetic objects rendering within real scenes. This solution comprises three distinct approaches to improve scene realism, all of them regarding the illumination problem. Some techniques were chosen to tackle these topics. The first technique is *Image Based Lighting* (IBL), where the scene images are used to estimate the incoming light intensity over the objects surface [2]. The second technique is *Ambient Occlusion* (AO) [3], which compensates the lack of shadow generation in the IBL technique. Finally, the third approach adopted to improve scene realism is lighting effects, where effects caused by direct illumination are simulated, such as Bloom and Glare [4].

In sequence, Section 2 presents a background about studies made on PR. Section 3 presents some related work to the use of illumination techniques for PR and AR. In Section 4, the approach adopted in this work is detailed. Section 5 shows the results obtained with the implementation of the mentioned techniques. Finally, Section 6 draws some conclusions about the presented work and points out some improvements for future versions.

2. Background

In Computer Graphics, many methods have been developed to increase the realism in synthetic images. In this context, the problem of PR was the most researched topic. This problem involves the synthesis, from scene description input data, of a whole image that seems like a real photo. Lately, new and more specific problems arose. One of them is the *Photorealistic Rendering of Synthetic Objects into Real Scenes* (PR-SORS), which was firstly demanded by the cinematographic industry. This industry, since last decades and until now, has the necessity of inserting seamlessly digital characters into its movies. Since lots of research was already done in this area, when watching the most recent movies, it can be noticed that good results were

already achieved merging any kind of digital entities (characters, vehicles, environments etc.) with real ones. Another problem that has come out is Real-Time *Photorealistic Rendering of Synthetic Objects into Real Scenes* (RPR-SORS). This problem is relatively new and appears in the AR context [5]. Coarsely, RPR-SORS is just a problem of PR-SORS with some particularities. Since AR deals with interactive applications, the main difference between RPR-SORS and PR-SORS is that in the former everything must execute in real-time.

The quality of the results in both, PR-SORS and RPR-SORS will be determined by the way three major topics are handled: shape, appearance and behavior of the virtual objects [6]. If one of these topics is not properly dealt with, the virtual objects can appear highlighted, permitting users to easily notice that they are not real.

The shape of a virtual object can reveal it when its proportions are not coherent with real objects proportions. A common example in which a virtual object is revealed by its shape can be noticed in cartoons, where characters members' proportions are intentionally distorted to transmit a comic feeling.

The appearance of an object is determined by its albedo (material's reflectance) and by the amount of light reaching its surface. The albedo can either be modeled utilizing simple ways, such as using textures, or more sophisticated ones, using Bidirectional Reflectance Distribution Functions (BRDFs) [7]. The amount of light reaching the object's surface plays an important role in the final result, and that is the reason why many of the researchers' efforts were in finding ways to estimate it [8, 9]. At the beginning, this estimation used to be performed inferring the position of the real light sources and creating virtual point light sources (trying to match its intensity and color) where the real ones were. Although it could seem a good solution, virtual objects should be illuminated not only by real primary light sources, but also by every real object that surrounds them. Therefore, this technique does not generate good results. Individually, the amount of light reflected by objects surrounding the scene is insignificant, but the sum of each contribution can noticeably alter the final result.

Finally, the behavior of the virtual objects can reveal that they are fake. Behavior is directly related to objects movements and how they interact. The complexity of an object's behavior can vary drastically depending on its nature. Deformable objects tend to have more complex behaviors than rigid ones. For example, a tree has a much more complex behavior than a statue.

Then, behavior, differently from shape and appearance, is not always relevant to virtual objects. It is possible to observe that with characters these three topics are important, as seen in Figure 1.

Figure 1. A character and its: (a) shape; (b) shape and appearance; (c) shape, appearance and behavior [10].

Keeping these three topics in mind, in order to obtain photorealistic results they must be coherently contemplated. The coherence between what happens in the real world and what is done in simulation is the key to achieve good results.

As stated before, the RPR-SORS problem was divided in three parts, so that it could be more easily solved. This paper will focus on the appearance topic, more precisely on the illumination problem. Since illumination is an issue applicable to any kind of synthetic object, this is one of the most important problems to be solved.

3. Related work

A number of studies have been conducted to address the problem of how to make a virtual object appear correctly inserted in a physical world. The general consensus is that while visual realism may not be related to productivity in virtual environments, it does increase the virtual world's sense of presence [11]. Certain realistic rendering effects, such as shadows, have been studied in particular to show their importance in determining the presence of virtual objects [12]. However, research about perceptually based rendering has shown that there are certain visual effects that humans are insensitive to [13]. Clearly, there is still a need for further studies about the effects of realism on user's perception of *Virtual Reality* (VR) and AR; on the other hand, there is a significant amount of evidence suggesting enhanced realism gives the user a greater sense of presence of virtual objects in the real world.

In the past few years, there has been a significant amount of research about advanced lighting and rendering techniques which are applicable to AR. In 1998, Debevec [8] raised interest in the concept of IBL, using an image of a light probe to capture environment

illumination and then calculating shading based on this data in offline renderers. This concept adapted the much older technique of sphere mapping, originally introduced in 1976 [14]. A group of developers at ATI implemented an interactive application that uses the technique proposed by Debevec, being able to compare Low Dynamic Range (LDR) to *High Dynamic Range* (HDR) scenes [15]. Kawase implemented a series of effects caused by direct illumination, looking for an approximation of the visual effect in the real world [4]. Besides that, Spencer et al. studied the physical effect of light rays in the human eye, and Kakimoto et al. implemented a shader taking into account physical properties of light rays [16, 17].

Environment maps have been directly applied in various AR systems. State et al. demonstrated a video of a silvered sphere morphing into a teapot, environment mapped by the captured video of the sphere [18]. In a more recent work, Agusanto et al. presented the seamless integration of virtual objects in an AR environment using IBL techniques and HDR environment illumination maps [5]. In their work filtered maps are blended together at run time to produce various material properties. Kanbara and Yokoya took a different approach, dynamically acquiring environment map data to control a distribution of point lights around the virtual geometry for diffuse lighting [19]. Both groups produced shaded virtual objects in real-time, but are limited either by inability to respond to dynamic changes in lighting, or inability to simulate many different material types.

Global illumination techniques have also been applied to PR-SORS. Fournier et al. proposed an early system in 1993 [20], which intended to create a detailed scene model and use a global illumination algorithm to add virtual objects and lights. Due to complex lighting calculations, each frame took minutes to render. The technique was improved by Loscos et al. in 2000 [21], who achieved a considerable speedup, so that rendering times went down to seconds per frame. Gibson and Murta [22] kept real-time rendering at high priority and replaced a global illumination solution with a number of fast approximations for simulating lighting and shadows. The shift away from an accurate rendering solution limits their system to a subset of actual light transport calculations – specifically, the authors do not address light transport from virtual objects to the surrounding physical environment.

Finally, a different approach to the problem of illuminating the physical world taking into account virtual objects has been proposed by Raskar et al. [23] and Bimber et al. [24]. Both groups provide a consistent lighting situation between the real and the virtual objects to achieve a convincing AR application. The work presented by these projects is most applicable to optical see-through AR systems.

4. Proposed approach

The approach adopted in this work deals with three major illumination topics: estimation of radiance over virtual objects' surface, self shadows, and lighting effects that happen due to lens characteristics. The techniques chose were: IBL, AO, Bloom, and Exposure Control. These will be explained in detail next.

4.1. Image Based Lighting

IBL is a technique used to illuminate virtual scenes under a complex environment illumination. The illumination that comes from the environment is represented by an Irradiance Environment Map (IEM). An IEM is an omnidirectional measurement of the radiance reaching a given point in the space, i.e., it contains the amount of light coming from every direction that passes through a given point. In practice, the IEM is sampled using a direction to obtain the radiance coming from this direction. A usual way to obtain this map is by shooting a reflective sphere (the light probe) in different exposures and combining these photos into a single HDR image. This Light Probe Image (LPI) stores the physical values of radiance that can be observed in the real world, differently from tradition digital images which represent colors that should appear on the monitor. There are many formats of IEMs, such as spherical (a LPI) and cubic, which are shown in Figure 2.

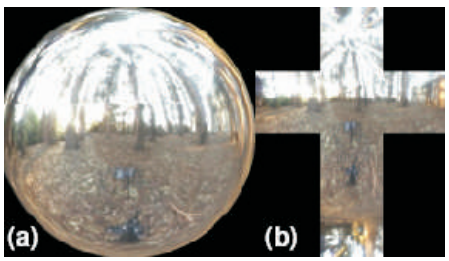


Figure 2. IEMs: (a) spherical, (b) cubic [25].

As in the IBL technique the light source is actually a real world photograph, the results obtained with this technique are more realistic than that ones that use point and directional light sources. Although, because in this work just one light probe is used to illuminate the scene, the results obtained are just an approximation of what actually happens in the real world. With just one light probe, points far away from where the light probe was acquired will be wrongly illuminated. Although, this error is tiny and does not

impact the final results.

There are only two steps required by this technique. In the first one, the IEM is convolved, generating a new image called Diffuse Irradiance Environment Map (DIEM). The second one just applies the DIEM to the virtual objects that must be illuminated.

4.1.1. Convolving the IEM. There are two ways to perform this convolution. The first one involves a brute force convolution that takes too much time (it may last for hours), though it is easier to implement. The second one is more complicated to implement, although it takes much less time than the first one. This work will focus on the second technique due to its better performance. Even using the second technique, it could spend more time than a real-time application allows, so King [26] proposed a method that performs this convolution in Graphics Processing Unit (GPU). Using GPUs, the algorithm runs fast enough to be fully executed at every frame of the application, so that it could convolve a new IEM to each frame without losing an interactive frame rate.

The convolution of the IEM (the lighting function) is performed in three steps: project the lighting function and the reflection function into a suitable base, obtaining some coefficients; multiply the lighting coefficients by the reflection coefficients (this is equivalent to filtering); rebuild the resulting function from the new coefficients. At the end, the resulting function is the DIEM. The base used is spherical harmonics, which is appropriated because its domain is the set of every direction into a 3D space (represented by), the same IEM's domain. Ramamoorthi, in [9], shows that the first 9 coefficients are enough to generate a DIEM. With the base functions in hands, the projection of the lighting function is performed using the discrete spherical harmonic transformation described by King. In this work's approach the way that King samples the IEM was changed to avoid the difficulty involved in evaluating the solid angle subtended by each texel (direction). A uniform spaced sampling scheme was used so that the discrete spherical harmonics transformation became

$$L_l^m = \sum_\Omega L(\omega) y_l^m(\omega) \frac{4\pi}{n}, \quad (1)$$

where L_l^m is a coefficient that will be generated, $0 \le l \le 2$ (because just the first 9 coefficients are needed), $-l \le m \le l$, is the set of every direction in the 3D space, ω is a direction used to sample the IEM and a spherical harmonic base function L is the lighting function (IEM), y_l^m is a spherical harmonic base function, and is the number of samples. In order to generate the evenly spaced directions ω, a mapping scheme where uniformly spaced 2D coordinates are mapped into two angles was used. These two angles represent a direction in spherical coordinates and the mapping is performed using the following expression:

$$(x, y) \rightarrow (2\cos(\sqrt{1-x}), 2\pi y) \rightarrow (\theta, \phi), \quad (2)$$

where (x, y) are evenly spaced 2D coordinates over the unit square, and (θ, ϕ) represents the direction in a 3D space.

The second step in the convolution process consists in multiplying (filtering) the coefficients obtained from the lighting function by the coefficients obtained from the reflection function. After this multiplication, new coefficients are generated.

To finish the convolution of the IEM, the coefficients generated from the last step are converted back into the spatial representation. This is performed using the discrete inverse spherical transformation

$$\tilde{L}(\omega) = \sum_{l=0}^{2} \sum_{m=-l}^{l} L_l^m y_l^m(\omega) \quad (3)$$

where $\tilde{L}$ is the reconstructed function. The tilde indicates that it is just an approximation (remember that just 9 coefficients were used) of the real function. At the end of this step the DIEM is done, so now it can be applied to the virtual objects.

4.1.2. Applying the DIEM. This is the simplest step and involves just the application of the generated DIEM to the virtual objects. To apply appropriately this map to the objects it is important to understand what it exactly is. With a DIEM, differently from the IEM, the texel with coordinate ω stores the total radiance reaching a point with normal ω. In practice, with a direction ω the amount of light that reaches the surface's point with normal ω can be sampled, sotexture mapping is performed according to the virtual objects' normal vectors. At the end, the texel sampled is multiplied by the albedo information and this result is the final color.

In this work's approach the IEM itself was used to simulate specular reflections, i.e., it was used as a *Specular Irradiance Environment Map* (SIEM). In order to obtain this effect the IEM is also applied to the virtual objects using an environment mapping technique. Using this idea, diffuse, glossy and completely specular objects were simulated.

4.2. Ambient Occlusion

Before defining AO, it is worth to define what ambient

light is and which role it plays in the interaction with the objects in a scene. Ambient light is a uniform light,

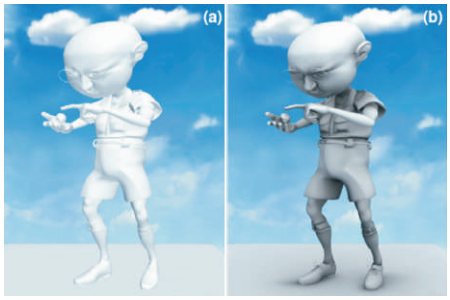


Figure 3. (a) environment lighting, (b) soft shadows added due to ambient occlusion [28].

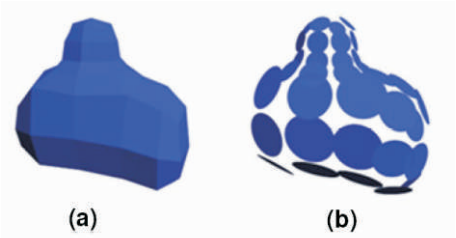


Figure 4. DAO algorithm: (a) original mesh, (b) calculated surfels from this mesh [28].

p

$$AO(p,n) = \frac{1}{\pi}\int_{\psi} V(p,\omega) \times \max(n.\omega, 0)\, d\omega \quad (4)$$

n p $V(p,\omega)$

ω

n ω

n

$$\left(1 - \frac{1}{\sqrt{\frac{A}{r^2}+1}}\right) \times \text{clmp}(\cos\theta_E) \times \text{clmp}(4\cos\theta_R) \quad (5)$$

θ_E θ_R

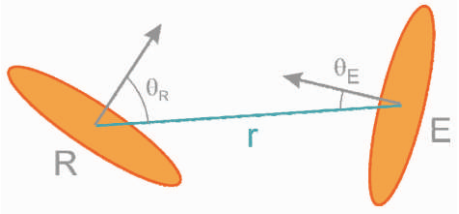


Figure 5. Receiver R receives shadow from emitter E with r as the distance between the centers of the two surfels.

4.2.3. Precomputation step. The amount of precomputation necessary for this method is small. It is enough to calculate for each vertex the correspondent surfel information (position, area, and normal) and to store this information in some textures.

Besides that, if no other information is used during the algorithm execution, the complexity is $O(n^2)$, because it is necessary to test each surfel against all others. It is easy to see that this algorithm scales very bad. To solve this problem some authors [28, 32] show ways of using a data structure to accelerate the calculation of accessibility. Basically, a hierarchy of surfels is created and, during algorithm runtime, this hierarchy is traversed. If the emitter surfel is far away, there is no need to take all its children into account. This simple addition allows the algorithm to run with a complexity of $O(n \log n)$. But, the first version of this work, due to time constraints, implements only the $O(n^2)$ algorithm.

4.2.4. Runtime. Basically, the implemented version of the algorithm is executed in just one pass, but Bunnel suggests a two passes algorithm. In the first one, for a receiving surfel, the occlusion of every other surfel is summed up and the result is subtracted from one. This pass gives a first approximation on the ambient occlusion.

So, in the second pass, again, for a given receiver, the occlusion is calculated, but this time the occlusion value for each surfel is modulated by its own accessibility value from first pass. With this step, any double shadowed surfel is corrected. However, it is easy to see that tripled shadowed surfels will still be too dark.

4.3. Lighting effects

In the real world, when human eye or a camera lens is directed towards a light source, there is a series of effects that affect the final scene visualized. These effects are currently not present in most computer graphics applications, but are slowly being implemented in some recent studies and commercial games [4, 33].

The observed effects were divided in order to solve each isolated problem. For example, when an object is observed against a light source, it seems that the light "bleeds" through the object's silhouette. That particular effect is called Bloom [4]. It can also be noticed that depending on some situations bright light sources or reflections cause light scattering, mainly in human eyes. It is perceived as a set of radial streaks around the light source. This phenomenon is called Glare, and it is widely known that it is caused by diffraction and scattering at obstacles close to or inside the eye. Rays from a light source are first diffracted by the eyelashes and sometimes by the edge of the eyelids, so that concentric streaks appear surrounding the light point [17]. Both effects just mentioned might happen also when the observer looks to a reflexive surface, though with some differences. This effect is named *Dazzling Reflection* [4]. Using an HDR image, it is possible to compute illumination and light reflection more accurately. In addition, depending on the amount of light (luminance) present on the scene, human eye can adapt the pupil to filter the incoming luminosity. This is called Exposure Control, a phenomenon reproduced by real cameras [4].

These effects are relevant to an AR application because they increase the scene realism, being many techniques already implemented in real-time. It is undesirable to render a scene containing a virtual object unaffected by lighting effects as real objects are.

In the presented solution, *Glare* and *Dazzling Reflection* are still under development and do not have effective results. For the other effects (*Bloom* and *Exposure Control*), an HDR image is used in order to provide relevant information about the scene illumination. Each implementation details are presented in the following subsections.

4.3.1. Bloom. The Bloom effect occurs because, when entering the human eye, the incoming ray is scattered, reaching the retina in many points [16]. These points, which would normally be stimulated only by light reflected by the object, end up being stimulated also by the light emitted directly by the light source, causing the characteristic glow.

The exact implementation of the human eye phenomenon is computationally complex, yet possible. For real-time applications, a simplification is preferred, but still capable of producing a similar result.

To simulate this effect, the most used technique is the application of successive Gaussian filters on the brightest points of the image. The first step is isolating the brightest areas of the image; this can be done by a simple threshold. Then, the first Gaussian filter is passed through this binary image, to blur it horizontally. Afterwards, a second Gaussian filter is passed vertically on the result of the first filter. This process is repeated to spread the light bloom. The result image is then added to the original scene to produce the desired effect. Figure 6 shows an example of Bloom application. Note that in Figure 6b the sphere support is obfuscated by the light source.

4.3.2. Exposure Control. The problem of sensibly mapping the results of a physics-based image synthesis algorithm to the limited range of displayable values is

Figure 6. Image: (a) without Bloom, (b) with Bloom [15].

commonly referred to as tone mapping [34]. Recent algorithms try to analyze the local luminance, in order to reproduce the phenomenon that happens in human eyes, producing more realistic results.

This technique is used to simulate the Exposure Control effect, as shown in Figure 7. This effect is naturally produced by the human eye when contracting or expanding the pupil, or by a camera when varying the lens aperture or exposure time.

Figure 7. Different exposure levels [4].

In order to accomplish the mentioned effect, it is necessary to calculate the overall scene luminance, according to Reinhard's algorithm [35]. It is possible to read the luminance of the whole scene in a single pass and store the result in a texture.

This value is then passed to the next step to calculate a scaled luminance for each pixel of the image. This scaled luminance is finally compressed in order to blend high and low values within a displayable range. This final value is then multiplied by the original pixel color to achieve the final result.

5. Results

Results shown in this section were obtained using a PC that has an AMD Athlon 64 3200+ processor, 1024MB of RAM and a NVidia GeForce 7900 GTX graphics card with 512MB of memory. The screen resolution was GH pixels. In order to avoid dealing with specifics details of graphics API, the OGRE [36] was used.

5.1. Image Based Lighting

Figure 8 shows a white ogre head being illuminated under two different environments: a forest background, and a cathedral environment. It is worth to say that in these results only the IEM was used to illuminate the virtual objects, i.e., no synthetic point light sources. In both images, II samples were used during the convolution step and the resulting DIEM's resolution is GG texels. The respective DIEM is shown at the top left corner of each image. As seen in Figure 8, these values provide a reasonable trade-off between good visual results and interactive frames per second (FPS), since in both images 118 FPS was achieved.

Figure 8. (a) forest background and with specular highlights, (b) cathedral environment without specular highlights.

An important IBL's technique parameter is the amount of samples performed. It can alter drastically performance and visual quality. Figure 9 shows how sampling rate alters visual quality and FPS. It was obtained using a scene with 5968 triangles and the generated DIEM's resolution is 128x128 texels.

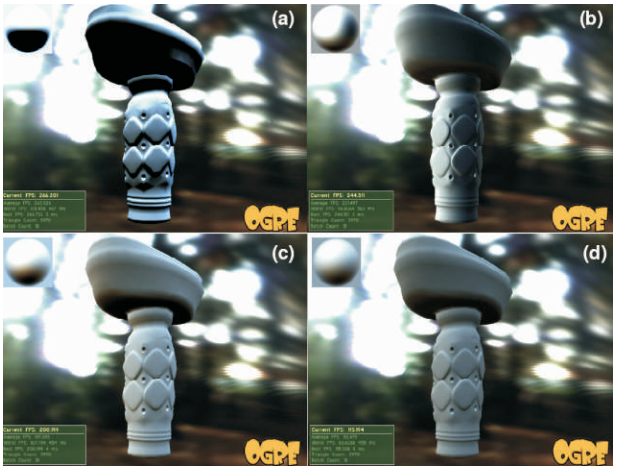


Figure 9. IBL sampling rate: (a) 2x2 samples at 266 FPS, (b) 8x8 samples at 244 FPS, (c) 16x16 samples at 200 FPS, and (d) 32x32 samples at 115 FPS.

Actually, 32x32 is the biggest amount of samples that was succeeded due to GPU program size constraints. With more than 32x32 samples the application was still running, although the visual results were unpredictable.

5.2. Ambient Occlusion

In Figure 10 it is possible to see three different results using the DAO technique explained earlier. The small red square at the top left of each image is a texture in which its red channel stores the values of

accessibility calculated by the algorithm. As this technique is a vertex based one, each texel corresponds to the accessibility of the related vertex, and the higher the value in the texture, the greater is its accessibility.

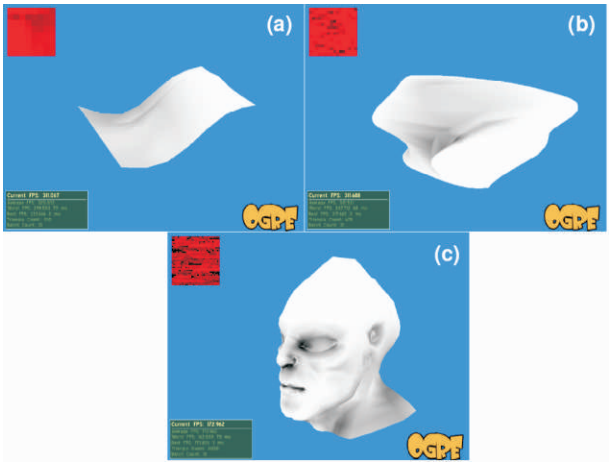


Figure 10. DAO results: (a) 64 vertices at 320 FPS, (b) 256 vertices at 312 FPS, and (c) 1024 vertices at 172 FPS.

As can be seen in Figure 10, the performance was kept almost constant when dealing with meshes of 64 and 256 vertices. The impact is not so soft when the mesh has 1024 vertices, but even so, the performance hit is not enough in order to prevent the use of this technique in a real-time application.

5.3. Lighting effects

Regarding lighting effects, the first results obtained were concerning Bloom and Exposure Control. Figure 11 shows the textured ogre head being dazzled in an HDR environment. The bright points of the scene were extracted to produce the Bloom effect observed (Figure 11a). One 512x512 and two 256x256 temporary textures were used to archive the thresholded and blurred images.

Figure 11. (a) ogre head rendered with Bloom, (b) without Bloom.

Figure 12 illustrates the Exposure Control effect. It can be observed that when the camera is directed towards the window, the whole scene is darkened, because the amount of incoming light increases. 64x64 samples were used in order to calculate the average luminance.

Figure 12. Exposure Control in a kitchen environment.

The example application presented good interactive rates, running steady at 130 FPS. *Glare* and *Dazzling Reflection* were not yet implemented by the authors.

5.4. Integrated techniques

The previous explained techniques have been put together in order to produce the final results (as shown in Figure 13). Since the methods do not have reciprocal dependence, any combination of them can be done. This flexibility can be used to adjust the application to obtain either better visual results or better performance.

The visual contribution given by each technique can be noticed in Figure 13. First, the DIEM is used to simulate the diffuse illumination and the IEM itself (SIEM) is utilized to produce the specular highlights. Then, AO is used to enhance re-entrances' darkness so that saliencies perception is increased, as can be noticed on the gap between lips and the ocular and nasal cavities. Finally, Bloom and Exposure Control techniques are utilized to enhance the illumination effect, respectively exemplified by the light coming from the background surpassing the silhouette of the top of the head and the darker appearance in the second image since its IEM is brighter.

Figure 13. Results combining the three techniques under two different environments. In both images 49 FPS is achieved.

6. Conclusions and future work

This paper presented a solution to the illumination problem in the RPR-SORS context. This solution deals with three major illumination topics: estimation of radiance over virtual objects' surface, self shadows, and lighting effects that happen due to lens characteristics. The techniques described in this paper are independent and consequently they can be easily put together. These

techniques were implemented using GPU resources so that they run with better performance. This approach showed to be suitable to AR applications that need a photorealistic look.

The IBL technique shows some problems when dealing with spherical maps, so the experimentation with different kinds of maps is also desirable.

Techniques are being developed to capture HDR image data in real-time [37], and graphics hardware is capable of rendering using HDR image data. Combining these two tendencies paves the way to create real-time realistic results. The implementation of more lighting effects, such as Glare and Dazzling Reflection, is also planned.

The AO technique has lots of improvements to be done, such as indirect illumination. It is also very important to verify if this technique is one of the best ones to be applied in an AR context. Besides that, an investigation about other shadow generation techniques, such as Shadow Fields [38], is planned.

Another very important part of this work is the study and implementation of new techniques for seamless integration between virtual and real objects in the AR context.

7. References

[1] J. Ferwerda, "Three Varieties of Realism in Computer Graphics", *SPIE Human Vision and Electronic Imaging*, 2003, pp. 290-297.

[2] P. Debevec, "Image-based Lighting", *ACM SIGGRAPH Courses,* ACM, New York-USA, 2005, p. 3.

[3] S. Zhukov, A. Inoes, and G. Kronin, "An Ambient Light Illumination Model", *Rendering Techniques*, Springer-Verlag Wien, New York-USA, 1998, pp. 45-56.

[4] M. Kawase, "Practical Implementation of High Dynamic Range Rendering", *GDC 2004*. Available: Musaki Kawase's home page. URL: http://www.daionet.gr.jp/~masa/rthdribl/index.html, visited on 29th October, 2007.

[5] K. Agusanto, L. Li, Z. Chuangui, and N.W. Sing, "Photorealistic Rendering for Augmented Reality Using Environment Illumination", *ISMAR, IEEE Computer Society*, Los Alamitos, 2003, pp. 208-216.

[6] P. Heckbert, "Class Notes – So What is Computer Graphics?", *Carnegie Mellon University*, 1997, pp. 8-11. Available: Paul Heckbert's home page. URL: http://www.cs.cmu.edu/afs/cs/academic/class/15462/web.97f/notes/intro.pdf, visited on Oct. 2007.

[7] S. Rusinkiewicz, "A Survey of BRDF Representation for Computer Graphics", 1997. Avaliable: Szymon Rusinkiewicz's home page. URL: http://www.cs.princeton.edu/~smr/cs348c-97/surveypaper.html, visited on Nov. 2007.

[8] P. Debevec, "Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography", *SIGGRAPH*, ACM, New York-USA, 1998, pp. 189-198.

[9] R. Ramamoorthi, and P. Hanrahan, "An Efficient Representation for Irradiance Environment Maps", *SIGGRAPH*, ACM, New York-USA, 2001, pp. 497-500.

[10] 3D Total Tutorials. Available: 3D Total site. URL: http://www.3dtotal.com/team/Tutorials/discmage/discmage1.asp, visited on 30th Oct. 2007.

[11] R. Welch, T. Blackmon, A. Liu, B. Mellers, and L. Stark, "The Effects of Pictorial Realism, Delay of Visual Feedback, and Observer Interactivity on the Subject Sense of Presence", *Presence: Teleoperators and Virtual Environments*, 1996, pp. 263–273.

[12] N. Sugano, H. Kato, and K. Tachibana, "The Effects of Shadow Representation of Virtual Objects in Augmented Reality", *ISMAR, IEEE Computer Society*, Washington-USA, 2003, pp. 76–83.

[13] J. Ferwerda, S. Pattaniak, P. Shirley, and D. Greenberg, "A Model of Visual Masking for Computer Graphics", *SIGGRAPH*, ACM, New York-USA, 1997, pp. 143-152.

[14] J.F. Blinn, and M.E. Newell, "Texture and Reflection in Computer Generated Images", *SIGGRAPH*, ACM, New York-USA, 1976, pp. 266-266.

[15] ATI Developer. Available: AMD Developer Central site. URL: http://ati.de/developer/index.html, visited on 13th Sep. 2007.

[16] G. Spencer, P. Shirley, K. Zimmermann, and D.P. Greenberg, "Physically-Based Glare Effects for Digital Images", *SIGGRAPH*, ACM, New York-USA, 1995, pp. 325-334.

[17] M. Kakimoto, K. Matsuoka, T. Nishita, T. Naemura, and H. Harashima, "Glare Generation Based on Wave Optics", *Computer Graphics and Applications, IEEE Computer Society*, Washington-USA, 2004, pp. 133-142.

[18] A. State, G. Hirota, D.T. Chen, W.F. Garrett, and M.A. Livingston, "Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking", *SIGGRAPH*, ACM Press, New York-USA, 1996, pp. 429-438.

[19] M. Kanbara, and N. Yokoya, "Geometric and Photometric Registration for Real-Time Augmented Reality", *ISMAR, IEEE Computer Society*, Washington-USA, 2002, pp. 279.

[20] A. Fournier, A.S. Gunawan, and C. Romanzin, "Common Illumination between Real and Computer

Generated Scenes", *Graphics Interface*, Toronto, 1993, pp. 254-262.

[21] C. Loscos, G. Drettakis, and L. Robert, "Interactive Virtual Relighting of Real Scenes", IEEE Transactions on Visualization and Computer Graphics, *IEEE Computer Society*, Washington-USA, 2000, pp. 289-305.

[22] S. Gibson, and A. Murta, "Interactive Rendering with Real-World Illumination", *Eurographics Workshop on Rendering*, Springer-Verlag Wien, New York-USA, 2000, pp. 365-376.

[23] R. Raskar, G. Welch, and K.-L. Low, "Shader Lamps: Animating Real Objects with Image-Based Illumination", *Eurographics Workshop on Rendering*, Springer-Verlag, London-UK, 2001, pp. 89-102.

[24] O. Bimber, A. Grundhöfer, G. Wetzstein, and S. Knödel, "Consistent illumination within optical see-through augmented environments", *ISMAR, IEEE Computer Society*, Washinton-USA, 2003, pp. 198.

[25] Light Probe Image Gallery. Available: Paul Debevec's home page. URL: http://www.debevec.org/Probes/, visited on 6th Nov. 2007.

[26] G. King, "Chapter 10: Real-Time Computation of Dynamic Irradiance Environment Maps", *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, Addison-Wesley Professional, USA, 2005, pp. 167-176.

[27] J. Kontkanen, and T. Aila, "Ambient Occlusion for Animated Characters", *Eurographics Workshop on Rendering*, Springer-Verlag Wien, New York-USA, 2006, pp. 343-348.

[28] M. Bunnell, "Chapter 14: Dynamic Ambient Occlusion and Indirect Lighting", *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, Addison-Wesley Professional, USA, 2005, pp 223-233.

[29] M. Malmer, F. Malmer, U. Assarson, and N. Holzschuch, "Fast Precomputed Ambient Occlusion for Proximity Shadows", *Tech. Rep. RR-5779*, INRIA, 2005.

[30] M. Sattler, R. Sarlette, G. Zachmann, and R. Klein, "Hardware-Accelerated Ambient Occlusion Computation", *International Fall Workshop Vision, Modeling, and Visualization*, 2004, pp. 119-135.

[31] P. Shanmugam, and O. Arikan, "Hardware Accelerated Ambient Occlusion Techniques on GPUs", *Symposium on Interactive 3D Graphics and Games*, ACM, New York-USA, 2007, pp. 73-80.

[32] K. Hegeman, S. Premoze, M. Ashikhmin and G. Drettakis, "Approximate Ambient Occlusion For Trees", *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, New York-USA, 2006, pp. 87-92.

[33] G. McTaggart, C. Green, and J. Mitchell, "High Dynamic Range Rendering in Valve's Source Engine", *ACM SIGGRAPH Courses*, ACM, New York-USA, 2006, pp. 7.

[34] M. Ashikhmin, "A Tone Mapping Algorithm for High Contrast Images", *Eurographics Workshop on Rendering, Eurographics Association*, Aire-la-Ville-Switzerland, 2002, pp. 145-156.

[35] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, "Photographic Tone Reproduction for Digital Images", *Annual Conference on Computer Graphics and interactive Techniques*, ACM, New York-USA, 2002, pp. 267-276.

[36] OGRE. Available: OGRE's home page. URL: http://www.ogre3d.org/, visited on 7th Oct. 2007.

[37] J. Waese, and P. Debevec, "A Real-Time High Dynamic Range Light Probe", *annual conference on Computer graphics and interactive techniques*, ACM/Addison-Wesley Publishing, 2002, pp. 247.

[38] K. Zhou, Y. Hu, S. Lin, B. Guo, and H-Y Shum, "Precomputed Shadow Fields for Dynamic Scenes", *ACM Trans. Graph.*, ACM, New York-USA, 2005, pp. 1196-1201.

Virtual Modeling and Numerical Simulation of Aneurysms and Stenoses

Rodrigo L. S. Silva, Eduardo Camargo, Pablo Javier Blanco, Márcio Pivello, Raúl A. Feijóo

National Laboratory for Scientific Computation
Hemodynamic Modeling Laboratory
Petrópolis, Rio de Janeiro, Brazil

{rodrigo}, {camargo}, {pjblanco}, {pivello}, {feij}@lncc.br

Abstract

This paper presents a first approach to model and simulate hemodynamic pathologies, based on geometrical singularities, such as Aneurysms and Stenoses. The techniques described in this work allow the specialist to add these pathologies based on parameterized geometries or over real arterial geometries, usually obtained via segmentation techniques from medical images. The relative position of the pathology on the artery can be interactively changed with the aim of obtaining a better understanding of the disease.

1. Introduction

The increase of cardiovascular diseases in the last years has motivated the development of, for instance, computational techniques in order to aid their treatment. The complexity of these surgical interventions is notably growing in the last years due the fast advances in modern medicine. Some cardiovascular diseases, such as arterioscleroses and cerebral aneurysms, are reported to depend on hemodynamic factors, particularly on the characteristics of the wall shear stress induced by blood flow [1]. With the purpose of joining tools to aid the treatment of these diseases a computational environment called HeMoLab was created [6]. HeMoLab is a computer system that allows the creation of patient-oriented models of the human cardiovascular system and also the numerical simulation within a unified framework. The 3D modeling of the human cardiovascular system (HCS) aims to study with a high level of detail the hemodynamic features of the blood flow and how those features are modified when, for instance, a surgical planning is devised over such system. Any part of the arterial network can be represented in details by using patient-specific data (such as medical images) of the arterial district of interest. Particular cases, like the presence of geometric singularities (aneurysm, stenoses, bifurcations and so on) can be simulated in order to study local changes in the blood flow structure and used to retrieve a more complete quantitative information. A complete simulation is composed by the reconstruction of the geometrical data through medical imaging systems (Magnetic Resonance Imaging and Computed Tomography), determination of boundary conditions and mechanical properties of the blood and arterial walls of the selected area and finally the computation of an approximate solution of the problem.

This paper presents an interactive system to create and simulate virtual aneurysms and stenoses over parameterized geometrical structures and patient-based arteries. This interactivity allows the user to simulate several conditions and analyze the numerical results graphically using visualization techniques such as stream-lines, stream-tubes, warping techniques for the velocity field, etc. The tool developed in this paper was integrated into the HeMoLab system that, in turn, is supported by the ParaView architecture [11]. ParaView is a point-and-click 3D scientific visualization system that allows for most of the common visualization techniques (isocontouring, volume rendering) on structured and unstructured grids. Its implementation uses distributed memory parallelism, and is focused on visual data analysis of large scientific datasets. It is an open-source, multi-platform visualization application, and supports distributed computation models to process large datasets. It has an open, flexible, and intuitive user interface and an extensible architecture based on the Visualization Toolkit (VTK) [19].

The remainder of this work is organized as follows: Section 2 presents a brief survey of some works related to computer modelling of the cardiovascular system. In Section 3, the algorithm for the generation of aneurysms and stenoses is detailed, whereas mathematical modelling and numerical approximation of the physical phenomena involved are described in Sections 4 and 5, respectively. Results obtained after simulating the flow inside a patient-specific artery modified by the tools presented here are shown in Section 6, and some concluding remarks are listed in

Section 7.

2. Related Works

Cardiovascular diseases are one of the leading causes of death, morbidity, and invalidity in the world and several works have been conducted in order to prevent, treat and/or diagnose those pathologies. Some work oriented to the treatment, visualization, simulation and diagnosis of stenoses and aneurysms can be viewed in [9, 10, 12, 14].

For instance, in [7] a training system developed with medical purposes was proposed. The system enables the instructor to generate specific cases for analysis, allowing to gain insight not only in the basic feature of searching and stenosis evaluation processes, but also about the importance of the correct viewpoint of acquisition within the environment.

The authors of [8] propose an algorithm that decomposes the patterns of 3D unsteady blood flow into behavioral components to reduce the visual complexity while retaining the structure and information of the original data. The key point of the algorithm is to enable the visualization of large simulated data sets avoiding visual clutter and ambiguity.

This paper differs from these previous approaches by adding a friendly interface to enable the interactive positioning of the aneurysm or the selection of the cut area in the case of a stenosis. Also, algorithms to join two or more meshes with a different resolution were implemented. This feature allows the user to study diverse cases in an interactive and intuitive manner.

3. Modeling Aneurysms and Stenoses

The main feature of the tool proposed in this paper is the ability to easily merge two meshes so as to create new geometries maintaining the correct topology. A friendly interface was designed to enable an interactive positioning of the geometrical region to be added or removed in the case of aneurysm or stenosis, respectively. Basically, the system provides a sphere widget (Figure 1.a) to be added on a given region to create an aneurysm, or use this sphere to remove a portion of the arterial district to create a stenosis.

In Figure 1.b, the geometry of the widget was added to, in this case, a parameterized carotid bifurcation to create a virtual aneurysm. The same widget can be translated to another position as can be seen in 1.c, where it is employed to simulate the presence of a stenosis.

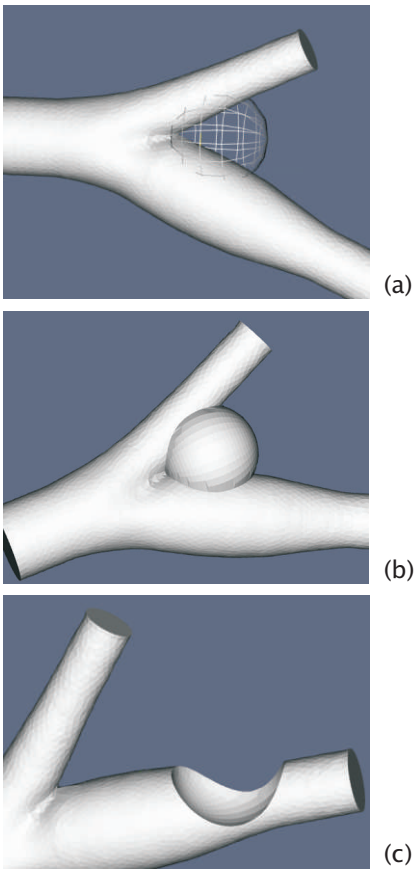


Figure 1. In (a), sphere widget used to model aneurysms and stenoses. In (b) a virtual aneurysm and in (c) a virtual stenosis. Both examples without geometry smoothing.

Finally, Figure 2 shows the result of the altered geometry presented in Figure 1.c after some geometry smoothing.

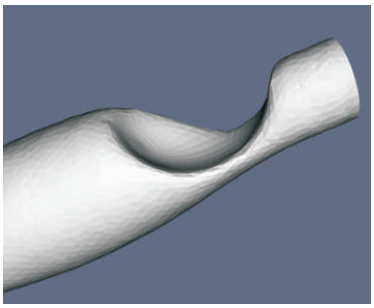


Figure 2. Filtered representation of the Figure 1.c.

3.1 Merging surfaces

One of the major problems of creating the pathologies mentioned above was to merge different meshes while preserving the consistency of the topology, connectivity and point sharing. To solve this problem

the following steps were adopted:

• set the sphere widget in the right position;

• clip the main surface with the widget;

• find boundary triangles on the clipped area;

• find the intersection between the clipped surface and the sphere widget;

• merge the two surfaces discarding the unused triangles on the sphere and the ones over the original geometry.

The resolution used in the geometrical representation of the sphere is also an issue. The ideal case is when the resolution of the sphere is the same as the resolution of the main surface. In this situation, joining the two meshes is easier, because the number of points to be joined in the clipped surface and the sphere is similar. Even when the resolution is similar, the number of the points in both surfaces generally is not the same. To solve this problem, an algorithm to sew the two meshes was developed. Basically, two points of each geometrical entity are used as seeds. The first point is chosen arbitrarily and the second point is found by computing the smallest Euclidean distance between the first point and all of the points of the other board. Based on these two points, a direction is selected and the sewing process starts, as illustrated in Figure 3.a. Since the two surfaces have different numbers of points in the region of intersection, one single point of one surface can be linked by edges with many other points of the other surface (Figure 3.b). As said, a criterion of proximity based on Euclidean distance was used to perform that linking.

3.2 Removing malformed elements

The generation of high quality meshes is a fundamenta step towards the resolution of a problem via an approximate technique such as the finite element method. The techniques described above generate a surface with the right topology but do not guarantee the quality of the mesh. Some special algorithms were developed to handle malformed triangles, e.g., needle-like triangles (Figure 4.a), or tiny triangles that have a much smaller area than the average element area in the surrounding region. (Figure 4.b).

Basically a good algorithm should remove those malformed elements maintaining the right connectivity with the surrounding elements. A needle triangle has always one neighbor that must be also removed, unless its smaller edge is located on a non-shared boundary. In any other case, this neighbor shares the smaller edge with the needle triangle as illustrated in Figure 4.c. A small triangle, when removed, forces the removal of all surrounding triangles (usually needles) as pictured in 4.d.

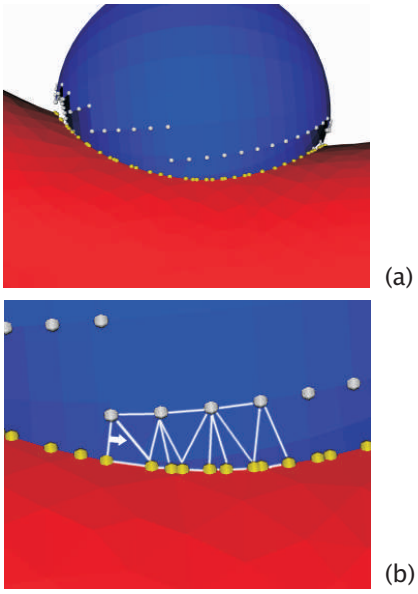


Figure 3. In (a) the points of the surface (yellow) and the points of the sphere (gray). The white arrow is over the edge linking the two seed points in (b) where the beginning of the sewing process is illustrated. The white arrow also shows the sewing direction.

These algorithms were implemented in ParaView and can be interactively used through an user-friendly interface. Needles are identified by a user-set threshold angle that defines the minimum angle allowed in an element. All the elements that have angles bellow the threshold are then highlighted, and the user can take them out from the mesh. Tiny elements are also defined by a threshold value, which represents now a percentage of the average area in the mesh. Elements bellow the threshold are highlighted and, again, the user can erase them. Details and examples about the algorithms described above can be found in [13].

4. Mathematical Model of the coupled 3D–1D blood flow simulation

Blood flow simulation is performed with a coupled 3D – 1D model of the arterial system, which consists in embedding a 3D model of an artery in a simplified 1D model of the arterial system. Inlets and outlets of the 3D model are coupled with their respective counterparts in the 1D model, such that no 3D boundary condition is necessary. The main advantage of using 3D–1D coupled models is that they behave as a unique system, avoiding the need for measurements that would be required when employing standalone 3D

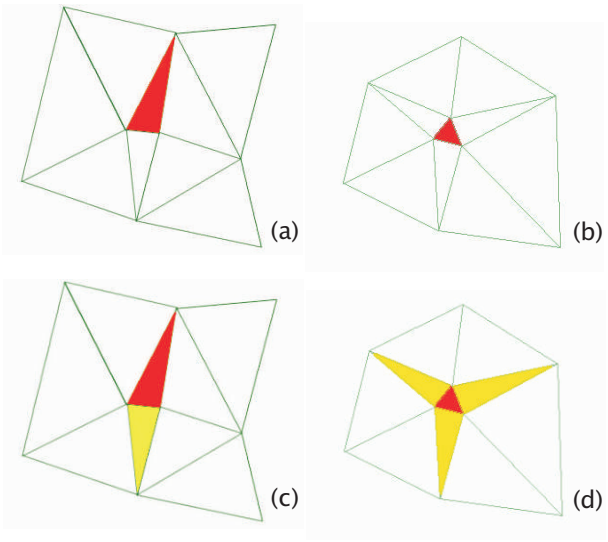


Figure 4. Red Triangles represent malformed elements like needle (a) and small triangles (b). In (b) and (c) the neighbors in yellow are triangles marked to be removed.

models. The boundary conditions over the inlets-outlets of the 3D region are accounted for the own solution of the entire problem. In other words, it can be understood in the following way: the 1D model feeds the 3D model with certain conditions and viceversa. On the other hand, the boundary conditions needed are the heart inflow boundary condition that represents the heart ejection and the Windkessel terminals, which are used to simulate the peripheral beds not included in the model. Therefore, once the 1D model is calibrated and validated, any artery which is present in the model may be replaced by a detailed 3D counterpart that is able to yield richer qualitative and quantitative results about the flow patterns in that region. This is particularly useful when studying athologies such as aneurysm, stenoses, etc. which are phenomena based on local factors.The governing equations were derived based on a variational formulation for the coupling of kinematically incompatible models, in this case 3D-1D flow models in compliant vessels [3]. The associated Euler equations for a Newtonian fluid when coupling a 1D domain 1D with a 3D region 3D through a coupling interface □c, and considering the ALE (Arbitrary Lagrangian Eulerian) formulation over 3D, are the following:

$$\rho A\frac{\partial \bar{u}}{\partial t}+\rho A\bar{u}\frac{\partial \bar{u}}{\partial z}=-A\frac{\partial \bar{p}}{\partial z}-8\pi\mu\bar{u}+f^{z} \quad \text{in } \Omega_{1D}\times(0,T) \qquad (1)$$

$$\rho\left.\frac{\partial \mathbf{u}}{\partial t}\right|_{Y}+\rho\nabla\mathbf{u}\,(\mathbf{u}-\mathbf{w})=-\nabla p+\mu\Delta\mathbf{u}+\mathbf{f} \quad \text{in } \Omega_{3D}\times(0,T) \qquad (2)$$

$$\frac{\partial A}{\partial t}+\frac{\partial(A\bar{u})}{\partial z}=0 \quad \text{in } \Omega_{1D}\times(0,T) \qquad (3)$$

$$\nabla\cdot\mathbf{u}=0 \quad \text{in } \Omega_{3D}\times(0,T) \qquad (4)$$

$$(-p\mathbf{I}+2\mu\varepsilon(\mathbf{u}))\,\mathbf{n}_1=-\bar{p}\mathbf{n}_1 \quad \text{on } \Gamma_c\times(0,T) \qquad (5)$$

$$A_c\bar{u}=\int_{\Gamma_c}\mathbf{u}\cdot\mathbf{n}_1\,d\Gamma \quad \text{on } \Gamma_c\times(0,T) \qquad (6)$$

Where n_1 is the unit outward normal to domain Ω_{1D} over the coupling interface Γ_c. In equations 1 and 3, which represent the 1D model, $\bar{u}$, $\bar{p}$ are the mean velocity and pressure values, ρ is the blood density, μ is the dynamic viscosity, A denotes the cross sectional area, $Q = A\bar{u}$ is the flow rate and z is the axial coordinate. Equations 2 and 4 represent the 3D model, **u** is the blood velocity, **w** is the domain velocity of change consistent with the ALE framework, while p is the blood pressure. Equation 5 stands for the continuity of the traction vector at Γ_c (the coupling interface between the 3D and 1D models), while expression 6 is the counterpart of the mass conservation. The wall movement is modelled according to the independent ring model [4], and its equations are stated bellow:

$$\bar{p}=\bar{p}_0+\frac{E\pi R_0h_0}{A}\left(\sqrt{\frac{A}{A_0}}-1\right)+\frac{k\pi R_0h_0}{A}\frac{1}{2\sqrt{A_0A}}\frac{\mathrm{d}A}{\mathrm{d}t} \quad \text{in } \Omega_{1D}\times(0,T) \qquad (7)$$

$$p=p_0+\frac{Eh}{R_0^2}\zeta+\frac{kh}{R_0^2}\frac{\mathrm{d}\zeta}{\mathrm{d}t} \quad \text{in } \Gamma_w\times(0,T) \qquad (8)$$

This is a rather simplified model of the structural behavior of the arterial wall, but serves to take into account the arterial compliance. The deformation of the domain Ω_{3D} is accounted for through a Laplacian problem, as stated bellow:

$$\nabla^2\mathbf{d}=0 \quad \text{in} \quad \Omega_{3D}\times(0,T) \qquad (9)$$

Since it is a small amplitude movement, no remeshing is performed. Instead, equation 9 is used in order to extend the wall movement to the interior of Ω_{3D}, and $\mathbf{d}|_{\Gamma_W} = \zeta\mathbf{n}$ is the wall displacement, where ζ is the scalar field that denotes the displacement of the wall in the normal direction, given by $\mathbf{n}$, that is obtained from equation 8. Finally, it is $\mathbf{w} = \frac{\delta \mathbf{d}}{\delta t}$. Refer to [3] for a further theoretical account about the coupling of 3D-1D blood flow models.

5 Numerical Approximation

In this section the numerical aspects of this work are briefly described. In this sense, details concerning the numerical techniques applied, the setting of the model regarding boundary conditions, the 3D geometry and the 1D arterial network topology are given.

5.1 Geometry Segmentation and Mesh Generation

The first step to obtain a 3D geometry of the artery is to apply an image segmentation technique to it. The output of this process is a 3D geometry representing the chosen region, which is then used as an input to a mesh generation software. Image Segmentation consists in identifying and isolating a region of interest in an image data set, be it 2D or 3D. In this paper a voxel grow approach was used, which consists in finding the neighbors of a given point based on its neighborhood connectivity and on its color level (or level, for short). Starting from a manually handled seed point and a bandwidth for the level, each neighbor of the point, if its level is within this bandwidth, is added to the output data set. The process is repeated recursively to each point added to this data set until the whole image has been analyzed. Also, multiple seeds with multiple bandwidths may be used. In this case, the seed point was chosen within the artery to be analyzed and the external contour of the output data set determined the geometry of its 3D model, which was used as input for the mesh generation process. For more information about the image segmentation process and other related topics refer to [5] and references therein. The surface obtained after processing the image must be properly manipulated to generate a high-quality mesh. This manipulation involves smoothing, addition of nodes, removing needlelike elements among other classical techniques. Once the triangulation of the surface is satisfactory, the volume mesh generation is performed using a Delaunay technique. The algorithms used for both surface and volume mesh generation are detailed in [17, 18].

5.2 Numerical Approximation of the Blood Flow Model

The numerical approximation of the problem is quite classical and we refer again to [3] for further details. The time discretization is performed by means of a single step finite difference method corresponding to a classical scheme for both 1D and 3D parts. The spatial discretization is carried out through the finite element method. Variables $Q, A, \bar{p}$ of the 1D model are discretized with $\mathbf{P}1$ finite elements, while $\mathbf{u}, p$ in the 3D model are discretized with $\mathbf{P}_1^B$ - $\mathbf{P}_1$ finite elements. The index B stands for bubble functions for the velocity field according to the mini element formulation [2]. The domain displacement, $\mathbf{d}$, is also approximated with $\mathbf{P}_1$ finite elements, and the reference velocity $\mathbf{w}$ is computed from the displacement by a backward Euler difference scheme. For both 1D and 3D parts, stabilization terms must be included in order to avoid the non–physical oscillating solutions present in standard Galerkin approximations. For the 1D model these terms are incorporated along the characteristics lines and correspond to a Galerkin Least Squares formulation [16]. For the 3D model the stabilization terms correspond to the Streamline Upwind Petrov Galerkin technique with a suitable stabilization parameter [15]. In all cases, nonlinearities are treated with Picard iterations.

6 Results

In order to show the capabilities of the tool described above, we present the results of the flow simulation inside a segment of the right interior carotid artery in two situations: (i): when an aneurysm was added with the tool described before and (ii) when the healthy artery was used in the simulation, for comparison purposes. It was used an SGI Altix 3700 BX2 workstation. This machine consists of 32 Itanium2 1.5 GHz processors with 64 GBRAM. The operational system is SUSE Linux Enterprise Server 9. The numerical solver is written in FORTRAN90 and the HeMo-Lab system is developed in C++, using the VTK library. Firstly, Figure 5 shows the interface of the HeMolab system and, specifically, of the tool developed. Figure 6.a shows the geometry for the two cases, while in Figure 7 it is shown the position within the 1D arterial network of the coupling locations (red segment between the green and blue points), Figure 6.b shows details of the mesh in the region of the aneurysm, where it can be seen the smoothness in the transition between the two merged geometries. The simulation was carried out within two cardiac periods of T = 0.8sec each starting from the at-rest situation.

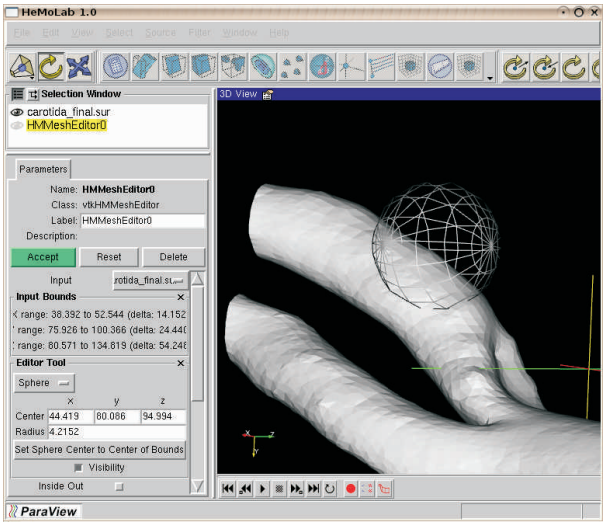


Figure 5. HeMoLab interface. The sphere is interactively positioned over the surface.

Although some time-varying transition should be expected before reaching the time-periodic regime, the main characteristics of the flow patterns are present even when starting the simulation from the at-rest state. The blood flow characteristics are here put into evidence by visualizing the velocity profiles and also some well-known indicators such as the OSI (measures the oscillatory behavior of stresses along a cardiac cycle) and the WSS (measures the mean value of stresses along a cardiac cycle) indexes.

Figures 8.a and 8.d show the flow pattern along the artery for each case at $t = 0.105sec$, with details in the region of the aneurysm. Notice that in the region downstream the aneurysm, both cases led to similar flow patterns and velocity magnitudes. However, after the aneurysm, the velocity is such that higher stresses take place. Also, a natural reduction in the velocity magnitude can be seen inside the aneurysm (see Figures 8.b and 8.e), where more complex flow patterns develop. This region within the aneurysm is also the place where higher values of OSI take place (see Figures 8.c and 8.f): these values were about two times higher in case with aneurysm when compared to the healthy case, while the peak in WSS occurs at the inlet and outlet of the aneurysm where, as it is well-known, the arterial wall is more affected by the blood flow.

Finally, Figure 9 shows the evolution profile of an isosurface of velocity magnitude 20 cm/sec. Here it is possible to appreciate the recirculation pattern of the flow when entering in the aneurysm region.

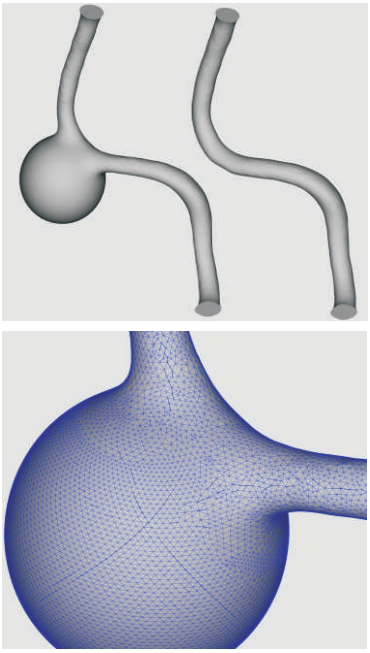


Figure 6. In (a) Arterial geometry, with and without aneurysm and in (b) mesh details on the region affected by the aneurysm.

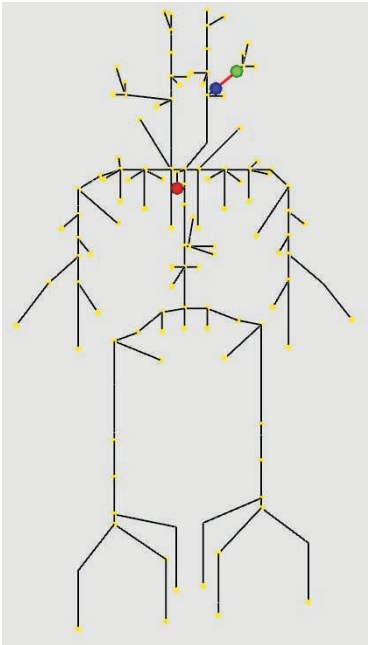


Figure 7. Positioning and coupling points of the 3D arterial district considered within the 1D model.

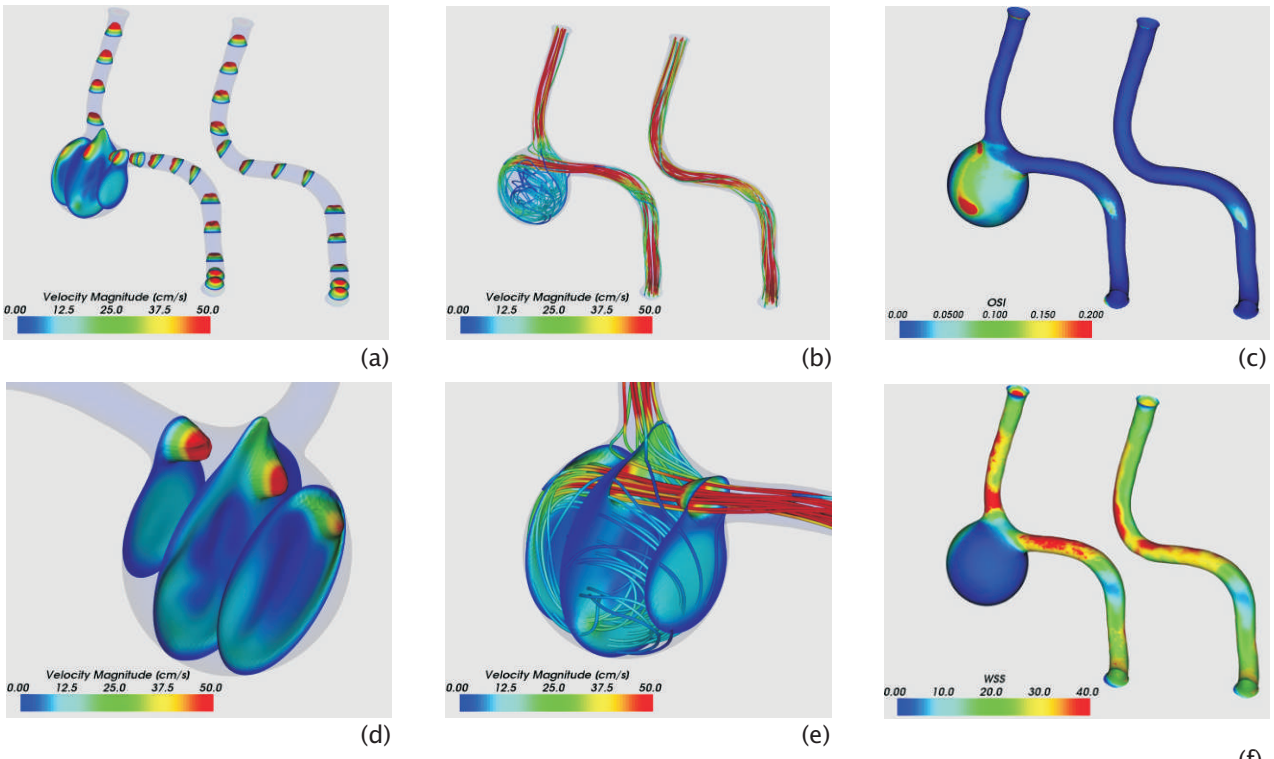


Figure 8. (a) and (d) Velocity profiles and in (b) and (e) stream lines along the vessels for T=0.105s. © OSI and (f) WSS distribution over the arteries with and without aneurysm.

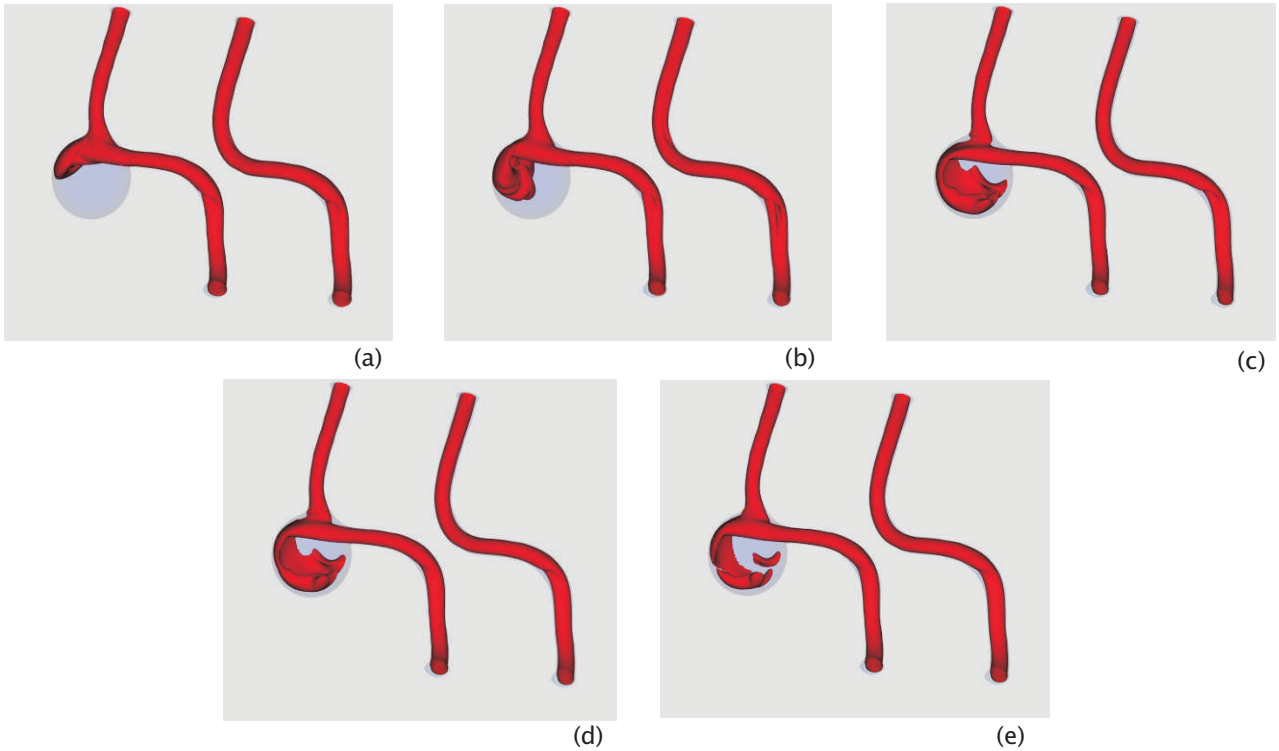


Figure 9. Time evolution of an iso surface of velocity magnitude equal to 20 cm/s. From left to right, top to bottom: t = 0.05 sec, t = 0.0625 sec, t = 0.0925 sec, t = 0.0995 sec and t = 0.105 sec.

7. Conclusions

In this work, it has been presented a first approach model and simulate pathologies arising in cardiovascular modeling such as Aneurysms and Stenoses. The geometrical models can be interactively constructed through the manipulation of spherical widgets over a given geometry representing an arterial district of interest.

After the reconstruction of the geometrical singularity that represents the pathology, the system allows in a straightforward way the computation of an approximate solution through the numerical simulation. The numerical results have been then analyzed using several visualization techniques such as stream lines, isosurfaces, arbitrary slices, warping techniques, etc. From the obtained numerical evidences it can be said that this kind of tool permits the the characterization, through diverse numerical indicators, of the altered physical phenomena that occur when different anomalies are present in the cardiovascular system.

Although the present model allows only spherical widgets, this is the first step to model more general and real situations such as elliptic-like perturbations or even freeform perturbations.

8. References

[1] A.M. Malek, S.L. Alper, and S. Izumo. Hemodynamic shear stress and its role in atherosclerosis. *JAMA*, 282(21):2035– 2042, December 1999.

[2] D. Arnold, F. Brezzi, and M. Fortin. A stable finite element for the stokes equations. Calcolo, 21(1), 1984.

[3] P. Blanco, R. Feij´oo, and S. Urquiza. A unified variational approach for coupling 3D–1D models and its blood flow applications. *Comp. Meth. Appl. Mech. Engrg.*, 196:4391– 4410, 2007.

[4] Y. Kivity and R. Collins. Nonlinear fluid–shell interactions: application to blood flow in large arteries. *Int. Sym. Discrete Meth. Engrg*., pages 476–488, 1974.

[5] I. Larrabide. Processamento de imagens via derivada topológica e suas aplicações na modelagem e simulação do sistema cardiovascular humano. *PhD thesis, National Laboratory for Scientific Computation*, 2007.

[6] I. Larrabide and R. Feij´oo. HeMoLab: Laboratório de Modelagem em Hemodinâmica. *Technical Report 13/2006, National Laboratory for Scientific Computing, 2006*.

[7] I. Lebar Bajec, P. Trunk, D. Oseli, and N. Zimic. *Virtual coronary cineangiography. Computers in Biology and Medicine, 33(3)*: 293–302, 2003.

[8] D. Lee, D. J. Valentino, G. R. Duckwiler, andW. J. Karplus. Simulation and virtual-reality visualization of blood hemodynamics: the virtual aneurysm. In *Proceedings of SPIE*, volume 4319, pages 465–470, May 2001.

[9] M.R. Harreld. Modeling, Simulation and VR Visualization of Brain Aneurysm Blood Flow. *Technical Report 960018*, UCLA Computer Science Department, 1996.

[10] M. Oshima. A New Approach to Cerebral Hemodynamics: Patient-Specific Modeling and Numerical Simulation of Blood Flow and Arterial Wall Interaction. *Bulletin for The International Association for Computational Mechanics*, 14:4–9, 2004.

[11] ParaView. http://www.paraview.org.

[12] Y. Sato, T. Araki, and M. Hanayama. A Viewpoint Determination System for Stenosis Diagnosis and Quantification in Coronary Angiographic Image Acquisition. In *IEEE Transaction Medical Imaging*, volume 17, pages 121–137, 1998.

[13] R. Silva, P. Blanco, M. Pivello, and R. Feijóo. Algoritmos para a Homogeneização de Malhas Triangulares. Technical Report 30/2007, National Laboratory for Scientific Computing, 2007.

[14] N. Stergiopulos, D.F. Young, and T.R. Rogge. Computer Simulation of Arterial Flow with Applications to Arterial and Aortic Stenoses. In *Journal of Biomechanics*, volume 25, pages 1477–1488, 1992.

[15] T.J.R. Hughes, L.P. Franca, and M. Mallet. A new finite element formulation for computational fluid dynamics: Vi. convergence analysis of the generalized supg formulation for linear time-dependent multi-dimensional advectivediffusive systems. *Comput. Methods Appl. Mech. Eng*., 63(1):97–112, 1987.

[16] S. Urquiza, P. Blanco, M. Vénere, and R. Feijó. Multidimensional modeling for the carotid blood flow. Comput. Meth. Appl. Mech. Engrg., 195:4002–4017, 2006.

[17] M. Vénere. Procedimientos para la generación de mallas tridimensionales de elementos finitos. *Revista internacional de Métodos Numéricos para cálculo y diseño en ingenieria*, 12:3–16, 1996.

[18] M. Vénere. Optimización de la calidad de mallas de elementos finitos mediante cambios localizados de topologia. *Revista internacional de Métodos Numéricos para cálculo y diseño en ingenieria,* 13:3–13, 1997.

[19] VTK - Visualization Toolkit. http://www.vtk.org.

Simulando Reações Flexíveis em Movimentos Capturados

Rubens F. Nunes, Creto A. Vidal, Joaquim B. Cavalcante-Neto,
Universidade Federal do Ceará
{rubens}, {cvidal}, {joaquim}@lia.ufc.br

Victor B. Zordan
University of California, Riverside, USA
vbz@cs.ucr.edu

Abstract

Combining physically-based simulation and animation obtained by motion-captured data is a new technique used to adapt the motions captured in studio to the application's needs. In this paper, the focus is on adapting motion-captured sequences to flexible reactions of a virtual human to perturbations, without losing motion realism. The proposed technique follows some of the ideas presented in the literature, but uses a new way of computing feedforward controllers through an auxiliary dynamic simulation. This new computation is simple, general and efficient, and allows the computation of the feedforward controllers in run time. These claims are demonstrated through experiments described in the example section and in the accompanying videos.

1. Introdução

1.1. Descrição do problema

A animação realista de personagens virtuais é um dos fatores determinantes do sentimento de imersão em ambientes virtuais. O interesse por esse tipo de animação tem aumentado tanto nas indústrias de jogos e entretenimento como em outras diversas áreas, tais como biomecânica, robótica e reabilitação, e grandes avanços ocorreram durante as últimas décadas [10]. Mais especificamente, o cuidado em relação à forma como os personagens interagem entre si e com o ambiente é indispensável para a credibilidade da animação e tem um impacto direto na qualidade da percepção do espectador. Por exemplo, na animação de uma luta entre dois personagens, espera-se que o personagem atingido por um golpe exiba uma reação compatível com o golpe desferido por seu oponente.

A utilização de movimentos, capturados em estúdio, executados por atores reais está bastante difundida na indústria da computação gráfica. Entretanto, apesar do maior realismo apresentado pelos dados capturados, ainda é um desafio adaptar essas animações, de forma fácil, às possíveis interações com o ambiente sem introduzir artefatos que destruam o realismo. Por sua vez, o emprego de simulação física permite gerar mais facilmente respostas interativas a perturbações do ambiente, através da aplicação de forças de colisão ao modelo. No entanto, a dificuldade de se projetar controladores para modelos complexos, tais como humanóides, que sejam capazes de produzir movimentos com qualidade semelhante à obtida por captura de movimentos, limita sua aplicabilidade.

A combinação das duas abordagens – movimentos capturados e simulação física – busca aproveitar o que essas duas técnicas têm de melhor. Zordan e Hodgins [22] introduziram controladores capazes de seguir movimentos capturados. Em cada passo da simulação, controladores do tipo PD (Proportional Derivative) calculam torques para corrigir o erro entre o estado atual, obtido a partir da simulação (feedback), e o estado desejado, determinado a partir dos dados capturados. Esses controladores simulam os músculos responsáveis pelo movimento nas juntas (articulações) e seus ganhos (k_s e k_d) correspondem às rigidezes desses músculos. Vale salientar, no entanto, que atribuir valores a esses ganhos é uma tarefa difícil, já que a atribuição de valores elevados leva o modelo a reagir de maneira rígida e inflexível às perturbações do ambiente; enquanto que a atribuição de valores baixos impossibilita seguir com precisão o movimento capturado.

Diante da inexistência de valores intermediários capazes de satisfazer simultaneamente essas duas exigências, Zordan e Hodgins [22] mantêm altos os valores dos ganhos para que o modelo siga o movimento capturado com precisão, e os reduzem temporariamente apenas na ocorrência de um impacto. Contudo, o uso predominante de ganhos altos não é realístico em sistemas biológicos [20].

Em síntese, o problema abordado neste artigo é: "Como simular reações flexíveis a perturbações no ambiente enquanto o personagem virtual segue fielmente movimentos capturados, isto é, sem perder as

sutilezas e o estilo que conferem realismo ao movimento?".

1.2. Solução proposta

De acordo com [20], sistemas biológicos usam controle feedforward (controle de laço aberto, ou seja, que não tem acesso ao estado atual do sistema) na realização da maior parte do trabalho, e usam controle feedback de baixos ganhos apenas na correção de pequenos possíveis erros em relação à trajetória desejada. Yin et al. [20] mostram que, se um termo feedforward for adicionado à equação de controle, controladores feedback de baixos ganhos podem ser capazes de seguir fielmente movimentos capturados. Assim, quando um impacto ocorre, o modelo reage de maneira flexível, sem que seja necessário atualizar os ganhos, como ocorre no método proposto em [22].

A solução proposta neste trabalho toma como base o trabalho de Yin et al. [20] e substitui o método de cálculo do termo feedforward por um novo método, mais simples, mais geral e mais eficiente, que permite a obtenção do termo feedforward em tempo de execução. Assim, ao invés de usar dinâmica inversa como em [20], calcula-se o termo feedforward através de uma simulação física auxiliar do modelo a qual usa controladores feedback de altos ganhos e é executada simultaneamente à simulação física principal. Os impactos inesperados são levados em conta apenas na simulação física principal que, por sua vez, usa controladores feedback de baixos ganhos. Em cada passo da simulação, os termos (torques) feedforward, obtidos pela simulação auxiliar, são incorporados ao conjunto de torques resultantes dos controladores feedback de baixos ganhos e aplicados ao modelo durante a simulação principal (Figura 1). O método proposto também permite a simulação de reações a impactos esperados como, por exemplo, a defesa de um soco. Para esses casos, onde o modelo deve reagir deliberadamente de maneira rígida, o impacto deve ser considerado tanto na simulação principal quanto na simulação auxiliar.

1.3. Organização do artigo

O restante deste artigo está organizado do seguinte modo: a Seção 2 contém uma discussão dos trabalhos relacionados; a Seção 3 consiste da explicação detalhada do novo método proposto para calcular o termo feedforward; a Seção 4 contém descrições de aspectos importantes de implementação; a Seção 5 consiste da apresentação e análise dos testes realizados; a Seção 6 é dedicada às discussões sobre as limitações do método proposto e sobre possíveis melhorias; e a Seção 7 contém as considerações finais sobre o trabalho.

Figura 1. Movimento capturado original (direita); Simulação auxiliar (centro) calcula o termo feedforward usando feedback de altos ganhos; Simulação principal (esquerda) combina feedforward com feedback de baixos ganhos.

2. Trabalhos relacionados

Vários pesquisadores, utilizando simulação física, se empenharam em projetar controladores adequados a movimentos específicos desejados. Hodgins et al. [5], utilizando máquinas de estados finitos, projetaram manualmente uma série de controladores para atletas humanos, os quais geram movimentos tais como corrida, ciclismo e salto sobre cavalo. Alguns autores propuseram representações de controladores que podem ser projetadas automaticamente para modelos menos complexos, utilizando otimização estocástica [12, 16, 17].

Gerar transições entre controladores também não é uma tarefa fácil. Wooten e Hodgins [18] projetaram manualmente controladores parametrizados para as seguintes ações de humanos virtuais: salto, manobras aéreas acrobáticas, aterrissagem e equilíbrio; e permitiram que transições entre esses controladores fossem realizadas. Faloutsos et al. [2] propuseram um framework de composição mais geral que permite a adição de novos controladores. Um controlador supervisor de nível superior é responsável por realizar automaticamente as transições, baseando-se em pré-condições apropriadas para o uso de cada controlador individual.

Neff e Fiume [11] consideram o efeito de forças externas para obter automaticamente os parâmetros dos controladores feedback PD, através de uma

formulação antagônica equivalente, tornando os controles de tensão e de posição independentes. Entretanto, uma contribuição adicional é exigida para seguir dados capturados usando essa formulação.

Modificar movimentos capturados para usá-los em novas situações é um desafio. Alguns autores sugeriram construir grafos de movimentos [6, 8]. De posse de um repositório, uma busca por quadros semelhantes é realizada, de acordo com uma métrica de similaridade. Transições são geradas entre esses quadros, interligando todo o repositório, para gerar o grafo.

Uma direção de pesquisa existente é desenvolver métodos adequados de pré-processamento do grafo, possibilitando que caminhos de movimentos desejados possam ser encontrados em tempo real. Lee e Lee [9] usaram programação dinâmica para simular lutadores de boxe capazes de atingir alvos específicos interativamente, percorrendo adequadamente um grafo de movimentos em tempo real. Reações a impactos foram simuladas cinematicamente.

Alguns trabalhos combinam as duas abordagens. Shapiro et al. [14] se basearam em [2] para criar um framework capaz de gerenciar controladores tanto dinâmicos como cinemáticos. Zordan et al. [24] usaram simulação física para produzir transições entre movimentos capturados a fim de simular reações a impactos.

Assim como o trabalho de Zordan e Hodgins [22], discutido na introdução, alguns outros trabalhos também propuseram estratégias para tratar o problema específico abordado neste artigo.

Pollard e Zordan [13] usaram uma abordagem semelhante à usada em [20] aplicada ao controle de uma mão. Um termo feedforward é pré-computado por dinâmica inversa para compensar a influência do movimento do braço e da gravidade.

Yin et al. [21] usam uma variação simplificada da técnica "feedback error learning", que é específica para movimentos cíclicos (e.g. um ciclo de um caminhar), e não tratam movimentos mais gerais como os utilizados neste trabalho.

Para seguir os movimentos capturados, Wrotek et al. [19] propuseram aplicar torques feedback externos diretamente em todos os corpos, baseados no erro em relação às coordenadas do mundo, ao invés das coordenadas de cada junta, e relataram que assim os ganhos são definidos de maneira mais fácil e mais flexível. Entretanto, para tratar reações a impactos, os ganhos tiveram que ser atualizados, usando a mesma estratégia apresentada em [22].

Allen et al. [1] apresentaram um método automático para calcular os ganhos, levando em conta informações de tempo (sincronização) dadas pelo usuário. Porém, ao seguir dados capturados, o método mantém altos ganhos e, no momento do impacto, também exige a atualização dos ganhos das juntas que devem ser influenciadas. Além disso, os testes realizados não envolvem movimentos atléticos.

Kry e Pai [7] apresentaram uma nova técnica em que, junto com o movimento, forças de contato são também capturadas. Essas informações são usadas para estimar a tensão nas juntas. Em contraste, o método proposto não exige acesso a essas informações.

3. Cálculo do termo feedforward

Para permitir que controladores *feedback* de baixos ganhos sejam capazes de seguir fielmente movimentos capturados, um termo *feedforward* adequado deve ser adicionado à equação de controle. O cálculo do termo *feedforward* é baseado na idéia de que, para movimentos bem treinados, sistemas biológicos já possuem informações que permitem o pré-processamento dos torques necessários à realização desses movimentos e usam controle *feedback* apenas para corrigir pequenos possíveis erros em relação à trajetória desejada.

Portanto, devido à boa qualidade do controle, os movimentos bem treinados são realizados com baixa tensão muscular (o que corresponde a baixos ganhos). Por outro lado, movimentos nunca antes treinados exigem que os músculos estejam mais rígidos para possibilitar a correção de maiores erros decorrentes do controle *feedforward* de baixa qualidade.

O método proposto neste trabalho estima o controle *feedforward* presente nos sistemas biológicos, através de uma simulação física auxiliar executada simultaneamente à simulação principal. A simulação auxiliar usa controladores *feedback* de altos ganhos para produzir torques que fazem o humano virtual seguir fielmente os movimentos capturados. Esse processamento corresponde ao pré-processamento feito por sistemas biológicos ao realizar movimentos bem treinados. Os termos **feedforward** utilizados na simulação principal são exatamente os torques produzidos na simulação auxiliar. Assim, os torques aplicados no modelo durante a simulação principal são compostos pela soma dos termos *feedforward* com os torques produzidos pelos controladores feedback de baixos ganhos utilizados na simulação principal.

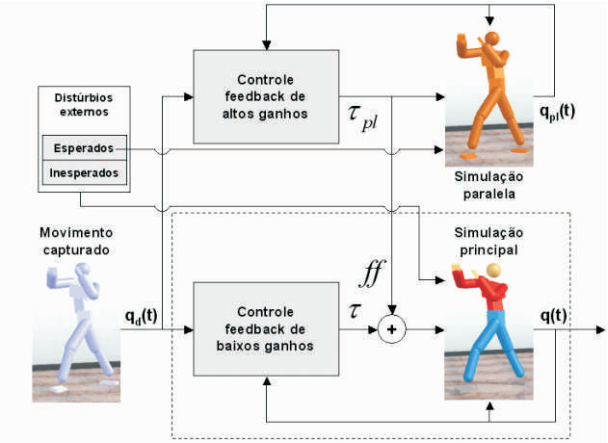


Figura 2. Esquema geral do método.

O esquema geral mostrado na Figura 2 ilustra os passos do método proposto: 1) Obtenção do estado desejado, qd(t), do modelo a partir do movimento capturado; 2) Cálculo dos torques, pl, pelos controladores *feedback* de altos ganhos da simulação auxiliar, para corrigir o erro entre o estado atual, qpl(t), e o estado desejado, qd(t); 3) Cálculo dos torques, , pelos controladores *feedback* de baixos ganhos da simulação principal, para corrigir o erro entre o estado atual, q(t), e o estado desejado, qd(t); 4) Aplicação dos torques, pl, ao modelo da simulação auxiliar e repasse de pl (torques *feedforward,* ff) à simulação principal; 5) Aplicação da soma + ff ao modelo da simulação principal; 6) Integração no tempo das duas simulações, considerando os distúrbios externos, e atualização dos estados atuais do modelo.

As duas simulações são atualizadas simultaneamente, o que permite calcular o termo feedforward em tempo de execução e dispensar, portanto, qualquer fase de pré-processamento, evitando, por exemplo, o grande armazenamento dos dados pré-computados. Apesar de o cálculo do termo feedforward em tempo de execução exigir um custo computacional adicional (maior do que o acesso direto aos dados armazenados pré-computados em [20]), esse custo adicional não compromete o desempenho e não é um problema no método proposto.

3.1. Simulação de reações

As reações que se pretende simular são de dois tipos: as flexíveis, que ocorrem em decorrência dos distúrbios inesperados; e as rígidas, que ocorrem em decorrência dos distúrbios esperados.

Na simulação principal, o controle *feedforward* recebido é adequado para seguir fielmente os movimentos capturados, mas, por não depender do estado atual do modelo principal, não é capaz de corrigir sua trajetória, mesmo para pequenos distúrbios externos. O controle feedback assume essa função e, através do uso de baixos ganhos, permite que as reações ocorram de maneira flexível.

É importante relembrar que o método proposto calcula os torques *feedforward* usando controladores *feedback* de altos ganhos, que têm acesso ao estado atual da simulação auxiliar (vide Figura 2). Assim, se a simulação auxiliar levar em conta os distúrbios externos inesperados, os torques, pl, produzidos para corrigir a trajetória do modelo auxiliar encapsularão esses distúrbios e os conduzirão à simulação principal, já que os torques, pl, são enviados à simulação principal como torques *feedforward* a serem aplicados ao modelo principal. No entanto, como os torques, pl, são calculados por controladores de altos ganhos, as reações obtidas nas duas simulações (simulação auxiliar e simulação principal) serão rígidas e inflexíveis, o que é indesejável no caso de distúrbios externos inesperados. Para evitar esse efeito indesejável, é importante ter o cuidado de desconsiderar os distúrbios externos inesperados durante a simulação auxiliar.

Vale salientar que, além de impactos inesperados, existem também os impactos esperados, para os quais a reação natural é rígida. Por exemplo, quando alguém dá um soco, mantém os músculos do braço rígidos e, da mesma forma, quando alguém se defende de um soco com o braço, deve mantê-lo firme para impedir que o soco o atinja.

O controle *feedback* ligado ao modelo principal a ser simulado é um controle de baixos ganhos, que, portanto, é adequado a reações flexíveis. Assim, para simular reações rígidas seria necessário aumentar deliberadamente os valores dos ganhos no momento do impacto. Porém, essa estratégia suscita uma série de questões difíceis de serem tratadas, como, por exemplo:

1. Quais juntas deveriam ter seus ganhos aumentados?
2. De quanto seria o aumento de cada ganho?
3. Em que instante os valores dos ganhos deveriam retornar aos baixos valores originais?

Além disso, na prática, há situações em que tanto impactos inesperados quanto impactos esperados ocorrem simultaneamente. Realizar esses ajustes e tratar situações como essa seria trabalhoso.

O método aqui proposto permite que impactos esperados pelo modelo resultem em reações propositalmente rígidas e inflexíveis, sem necessidade de modificação dos ganhos, além de tratar eficazmente e de forma automática situações com impactos

esperados ocorrem simultaneamente. Realizar esses ajustes e tratar situações como essa seria trabalhoso.

O método aqui proposto permite que impactos esperados pelo modelo resultem em reações propositalmente rígidas e inflexíveis, sem necessidade de modificação dos ganhos, além de tratar eficazmente e de forma automática situações com impactos simultâneos. Para isso, é suficiente considerar os impactos esperados tanto na simulação principal quanto na simulação auxiliar. Na Figura 2, pode-se observar que todos os distúrbios externos são considerados na simulação principal, enquanto que na simulação paralela, apenas os esperados são levados em conta.

3.2. Controle feedback

Controladores *feedback* PD (*Proportional Derivative*) não-lineares são usados tanto na simulação auxiliar quanto na simulação principal para corrigir o erro entre o estado atual e o estado desejado do modelo, através da seguinte expressão:

$$\tau = I \cdot \left[k_s f(\theta_e)(\theta_d - \theta) - k_d(\dot{\theta}_d - \dot{\theta})\right] \quad (1)$$

onde θ e θ_d são os ângulos atual e desejado da junta, k_s e k_d são as velocidades atual e desejada da junta, k_s e k_d são os ganhos dos termos proporcional e derivativo, I é o momento de inércia da cadeia de corpos afetados pela junta, $\theta e = (\theta_d - \theta)$ e $f(\theta_e)$ é o fator não-linear definido em função de θe. τ é limitado para evitar instabilidades na simulação dinâmica.

Os valores de θ_d usados nas duas simulações são obtidos diretamente a partir dos movimentos capturados. Os valores de $\dot{\theta}$ podem ser estimados por diferenças finitas também a partir dos dados capturados. Entretanto, na simulação auxiliar, devido ao *feedback* de altos ganhos, esses valores podem ser simplesmente zerados sem comprometer a qualidade com que o modelo auxiliar segue os movimentos capturados. Já na simulação principal, devido ao *feedback* de baixos ganhos, a informação de velocidade desejada é importante e o método proposto permite que essa informação seja convenientemente obtida diretamente a partir da simulação auxiliar, já que o modelo auxiliar segue fielmente os movimentos capturados. Portanto, os valores de $\dot{\theta}_d$ adotados na simulação principal são as velocidades $\dot{\theta}$ da simulação auxiliar, que são mais compatíveis do que possíveis estimativas por diferenças finitas a partir dos dados capturados.

O uso do fator de inércia, introduzido por Zordan e Hodgins [22], permite que apenas um par de ganhos (k_s e k_d) seja especificado para todo o modelo, eliminando o processo trabalhoso de especificar ganhos para cada junta. Allen et al. [1] descrevem os detalhes do cálculo desse fator, o qual deve ser atualizado em cada passo da simulação. Um par de ganhos (k_s e k_d) para cada simulação deve ser definido pelo animador.

Observações acerca da influência da razão kd:ks sobre os movimentos levaram às seguintes conclusões: 1) se k_d:k_s for baixa, o modelo tende a oscilar nas proximidades da configuração desejada; e 2) se k_d:k_s for alta, o modelo tende a se mover vagarosamente, como se estivesse na água. Considerando essas observações, o ideal é que a razão kd:ks seja grande nas proximidades da configuração desejada, para evitar as oscilações; e pequena caso esteja longe da configuração desejada, para evitar que o movimento seja retardado de forma não natural.

Fattal e Lischinski [3] usam controladores PD não-lineares, em que a razão é atualizada modificando-se o termo k_d. No modelo aqui proposto, que também usa controladores PD não-lineares, o valor de k_d:k_s é atualizado modificando-se o termo k_s. Essa estratégia é mais intuitiva, uma vez que quando o modelo recebe um impacto, o que implica no afastamento da configuração desejada, a tendência é ele se tornar mais tenso, correspondendo ao aumento dos ganhos. Assim, o aumento de ks reduz a razão k_d:k_s, como desejado.

Outra observação interessante é que, após receber um impacto, sistemas biológicos apresentam um pequeno atraso na reação devido ao tempo gasto no fluxo de informação ao longo dos neurônios e, principalmente, através das sinapses entre eles [20]. O modelo aqui proposto inclui esse atraso na definição do fator $f(\theta_e)$,

$$f(\theta_e) = \begin{cases} 1, & \text{se } |\theta_e| \le \lambda \\ \dfrac{|\theta_e|}{\lambda}, & \text{se } |\theta_e| \le M \cdot \lambda \\ M, & \text{caso contrário} \end{cases} \quad (2)$$

onde λ corresponde ao atraso na reação e M é o valor máximo permitido para o fator . λ e é medido em radianos, deve ser positivo e M deve ser maior do que 1. Analisando a função, temos que: $f(\theta_e) \geq 1$; $f(\theta_e)$ é linear, para $\leq |\theta_e| \leq$ M.; $|\theta_e| / = 1$, quando $|\theta_e| = \lambda$; e $|\theta_e| / =$ M, quando $|\theta_e| =$ M.. Em todos os testes realizados, os valores usados foram $= 0.1$ e M = 5.

3.3. Vantagens em relação ao uso de dinâmica inversa

Yin et al. [20] calculam o termo *feedforward* usando dinâmica inversa em uma fase de pré-processamento. No cálculo de dinâmica inversa, as acelerações são estimadas pela derivada segunda de *splines* cúbicas, geradas a partir dos dados capturados. Os valores resultantes da dinâmica inversa juntamente com os torques *feedback* de baixos ganhos, produzidos em tempo de execução, são usados na simulação física do modelo.

As vantagens do método proposto podem ser enumeradas como a seguir:

• Cálculo mais simples do termo *feedforward*. O cálculo feito pelo modelo proposto é exatamente igual ao cálculo dos torques *feedback*, só que em um modelo auxiliar.

• Consistência com o modelo principal. Os torques *feedforward* são mais compatíveis com o modelo principal, já que o modelo auxiliar é idêntico ao modelo principal e a técnica de simulação adotada é a mesma do modelo principal. Em contraste, o modo de cálculo por dinâmica inversa usa uma abordagem completamente distinta da simulação principal. Por exemplo, na dinâmica inversa, o movimento capturado é considerado nos cálculos, de forma indireta, através das acelerações que são estimadas a partir dele.

• Rastreamento de trajetórias mais gerais. Dinâmica inversa é apropriada para transformar trajetórias de movimentos realistas e detalhados (e.g. movimentos capturados) em funções apropriadas de força. Entretanto, quando a trajetória não corresponde a um movimento que respeita as leis da física ou é esparsa (e.g. keyframes), é complicado achar forças adequadas capazes de seguir uma aproximação da trajetória desejada. Por outro lado, controladores feedback são usados eficientemente para produzir torques apropriados para trajetórias mais gerais (inclusive trajetórias que não obedecem à física ou que são esparsas), tais como transições entre movimentos [18, 2, 22, 21] ou *keyframes* (controle de pose) [17, 21, 1]. Portanto, como o método proposto usa controladores *feedback* para obter os torques *feedforward*, ele é adequado e diretamente aplicável também a movimentos cinemáticos gerais, e não apenas a movimentos capturados.

• Custo computacional mais baixo. Em [20], o cálculo do termo *feedforward* por dinâmica inversa possui um custo computacional mais alto e não é realizado em tempo de execução. O método proposto realiza os cálculos em tempo de execução, permitindo tratar casos em que as trajetórias não podem ser previstas, tais como aquelas determinadas por controladores de mais alto nível, que dependem de sensores ou de impactos inesperados, para definir suas ações [2]. Essas trajetórias não podem ser pré-processadas com o intuito de obter torques *feedforward*, e, portanto, dinâmica inversa como usada em [20] não se aplica a esses casos.

• Tratamento eficiente em grafos de movimentos [6]. Reações também podem ser simuladas enquanto o modelo percorre grafos de movimentos. O uso de dinâmica inversa nesses casos requer que os torques *feedforward* sejam pré-computados para todos os movimentos capturados e para todas as transições possíveis, o que exige uma longa fase de pré-processamento e muita memória para armazenar os torques *feedforward*. Por sua vez, o método proposto calcula os torques *feedforward* em tempo de execução também nesses casos.

• Tratamento de reações a impactos previstos. O método proposto trata essas reações de forma bastante simples e direta, simplesmente considerando esses impactos nas duas simulações – simulação principal e simulação auxiliar. Em contrapartida, Yin et al. [20] não fornecem um tratamento especial para esse tipo de impacto. Além disso, aplicar a idéia proposta de considerar os impactos previstos a fim de que eles influenciem o termo *feedforward* resultante, usando dinâmica inversa, requer que os impactos sejam pré-processados e é trabalhoso incluí-los nos locais e instantes desejados.

4. Implementação

Um ambiente computacional de testes foi desenvolvido para examinar as reações do modelo, enquanto ele segue os movimentos capturados. Nesse ambiente, com o auxílio do teclado ou do mouse, o usuário pode tanto atirar objetos quanto aplicar diretamente forças ou torques externos nos corpos selecionados do modelo, em tempo de execução. O usuário também pode escolher se a influência externa inserida será ou não esperada pelo modelo.

O ambiente computacional de testes foi desenvolvido em C++, usando: o motor dinâmico aberto ODE (*Open Dynamics Engine*) [15] para realizar a simulação física e a detecção de colisão; a biblioteca FOX toolkit [4] para criar a interface gráfica; e a biblioteca OpenGL para renderizar os quadros da animação.

As animações do modelo exibidas no ambiente são obtidas diretamente da simulação física principal, sem

nenhum pós-processamento, e apresentam desempenho de tempo interativo. Em um PC com processador AMD Athlon de 1.8 GHz, 512 mb de memória RAM e placa de vídeo NVIDIA GeForce FX 5200 de 256 mb de memória, a taxa média de amostragem é de 10 fps, quando renderizando um quadro a cada 1/30 s de simulação. O intervalo de tempo considerado para cada iteração da simulação é de 0.0005 s, e um quadro é renderizado a cada 67 iterações (67*0.0005 s 1/30 s). Isso significa que para animar um segundo de simulação são necessários 3 segundos do tempo real.

Os movimentos capturados utilizados nos testes foram obtidos usando captura óptica em uma taxa de 120 fps. As posições dos marcadores ópticos foram convertidas nas posições e orientações globais correspondentes dos corpos do modelo [23], permitindo o cálculo direto dos ângulos das juntas. Para gerar os quadros intermediários necessários à simulação, os ângulos das juntas foram calculados usando interpolação linear esférica (slerp).

A realização correta de mudanças de coordenadas é fundamental em situações tais como a combinação dos torques feedback e feedforward na simulação principal ou a consideração dos impactos esperados na simulação auxiliar.

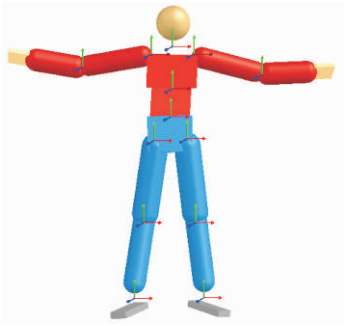


Figura 3. O modelo usado possui 39 DOF internos. Cada junta indicada possui 3 DOF.

5. Resultados

O modelo humanóide usado nos testes (Figura 3) é composto por 16 corpos rígidos e 13 juntas esféricas (*"ball-and-socket joints"*), totalizando 39 graus de liberdade internos e 6 graus de liberdade globais da "raiz" do modelo (pélvis). Os pulsos são fixos.

A flexibilidade do modelo e a qualidade com que o modelo segue os movimentos capturados dependem dos valores atribuídos aos ganhos (ks e kd) das duas simulações – simulação principal e simulação auxiliar. Os valores dos ganhos da simulação auxiliar devem ser definidos antes, pois os ganhos da simulação principal não interferem nessa escolha. Apesar de definida manualmente (i.e., por tentativa e erro) pelo animador, a escolha desses parâmetros não foi trabalhosa e, além disso, os valores escolhidos foram mantidos fixos em todos os testes. As relações entre os ganhos usados, nas duas simulações, foram: ksprincipal = 0.05*ksauxiliar e kdprincipal = kdauxiliar.

Para demonstrar a eficácia do método proposto, três testes foram realizados. Os vídeos acompanhando o artigo permitem visualizar e perceber melhor os movimentos e a qualidade do método.

No primeiro teste, vários objetos são lançados contra o modelo, enquanto ele segue os movimentos capturados. O modelo reage aos distúrbios externos e volta a seguir os movimentos capturados de maneira bastante natural. A Figura 4 mostra um exemplo em que o modelo reage a um impacto ocorrido no braço e, em seguida, a outro na cabeça. O modelo auxiliar não recebe os impactos e pode ser usado como referência para comparação. O movimento capturado original é mostrado nos vídeos acompanhantes.

O segundo teste compara a capacidade dos controladores feedback de baixos ganhos de seguir os movimentos capturados: sem usar o termo feedforward (Figura 5a); e usando o termo feedforward proposto, obtido a partir da simulação auxiliar (Figura 5b). O termo feedforward calculado usando o método proposto é eficaz e permite que os controladores feedback de baixos ganhos sejam capazes de seguir fielmente os movimentos capturados.

O terceiro teste mostra que o método proposto pode tratar facilmente distúrbios externos que são previstos pelo modelo, os quais devem implicar em reações rígidas e inflexíveis. A Figura 6a ilustra uma situação em que o modelo recebe um impacto inesperado na parte de trás da cabeça e, ao mesmo tempo, um impacto esperado na perna. O impacto esperado na perna é considerado também no modelo auxiliar, como ilustrado na figura. Os ganhos não são alterados em nenhum momento. Na Figura 6b, os mesmos dois impactos são considerados inesperados, para efeito de comparação. No primeiro caso, a reação ao impacto na perna ocorre de maneira rígida, como se espera, enquanto que, no segundo caso, o joelho se mostra flexível.

O método proposto se mostrou bastante eficaz e natural na reação aos impactos em todas as partes do modelo, inclusive quando mais de um impacto ocorrem simultaneamente ou vários impactos ocorrem em seqüência. Além disso, o método permitiu tratar, de

maneira simples e eficaz, distúrbios externos previstos.

6. Discussão e perspectivas

Para manter o equilíbrio do modelo nos testes realizados, forças e torques externos foram aplicados à "raiz" do modelo (pélvis) e aos pés (apenas quando em contato com o chão), para forçá-los a seguir os movimentos capturados. Essas forças e torques foram geradas por controladores *feedback* adicionais de altos ganhos, que foram usados nas duas simulações. Devido ao alto realismo presente em movimentos capturados, essa estratégia foi eficaz. Embora essa estratégia não seja fisicamente correta, o método proposto independe da estratégia de equilíbrio usada e, portanto, permite que outras estratégias realistas sejam empregadas. Entretanto, manter o equilíbrio durante movimentos atléticos, como os que foram apresentados neste trabalho, permanece um problema sem solução satisfatória. Yin et al. [21] propõem uma estratégia de equilíbrio promissora, porém específica, para movimentos em que os pés de apoio são trocados de modo periódico e uniforme (e.g. caminhar, correr).

Tanto limites nas juntas do modelo quanto contatos entre corpos pertencentes a um mesmo modelo apresentaram algumas instabilidades na simulação. Devido a essas instabilidades, limites nas juntas e contatos entre corpos pertencentes ao mesmo modelo não foram tratados nos testes realizados. Entretanto, essa é uma limitação da implementação, e não do método proposto. A superação dessas deficiências está sendo estudada e será apresentada no futuro.

É conveniente testar o método proposto de forma mais geral, acoplando-o, por exemplo, a sistemas capazes de realizar simples transições entre movimentos capturados ou de gerar e percorrer grafos de movimentos [6].

Seria interessante definir um critério geral para determinar se um impacto deve ou não ser esperado (percebido antecipadamente) pelo modelo. As forças de contato com o chão, por exemplo, já devem ser esperadas pelo modelo e, por isso, consideradas na simulação auxiliar.

Outra perspectiva interessante de utilização do método proposto seria permitir o modelo reagir antecipadamente para evitar que os impactos ocorram em partes mais vulneráveis do corpo, como em [25].

7. Conclusão

Este trabalho abordou o problema de simulação, de modo fácil e automático, de reações realistas a perturbações externas em movimentos capturados. Ao treinar um tipo de movimento, sistemas biológicos armazenam informações mais abstratas, e não explicitamente os torques feedforward. Em muitos casos, o movimento desejado específico a ser realizado é determinado (imaginado) com detalhes pouco tempo antes de ser executado, baseando-se muitas vezes no feedback obtido do ambiente, através dos sentidos. Conseqüentemente, para esses casos, o pré-processamento feedforward da ação correspondente, usando as informações armazenadas no treinamento, também ocorre pouco tempo antes da execução da ação. Portanto, calcular o termo feedforward em tempo de execução mantém o método proposto mais próximo de sistemas biológicos do que pré-processar todas as ações possíveis que o modelo possa executar.

8. Agradecimentos

Os autores agradecem ao CNPq e à CAPES pelo apoio financeiro para a realização deste trabalho.

9. Referências

[1] Allen, B., Chu, D., Shapiro, A. & Faloutsos, P., "On the beat! Timing and tension for dynamic characters". *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, San Diego, CA, USA, 2007, pp. 239-247.

[2] Faloutsos, P., van de Panne, M. & Terzopoulos, D., "Composable controllers for physics-based character animation", *Proceedings of ACM SIGGRAPH*, Los Angeles, CA, USA, 2001, pp. 251-260.

[3] Fattal, R. & Lischinski, D., "Pose Controlled Physically Based Motion", *Computer Graphics Forum,* 2006, 25(4), 777-787.

[4] FOX toolkit, http://www.fox-toolkit.org/, 2008.

[5] Hodgins, J.K., Wooten, W.L., Brogan, D.C. & O'Brien, J.F., "Animating human athletics", *Proceedings of ACM SIGGRAPH*, Los Angeles, CA, USA, 1995, pp. 71-78.

[6] Kovar, L., Gleicher, M. & Pighin, F., "Motion graphs", *ACM Transactions on Graphics*, 2002, 21(3), pp. 473-482.

[7] Kry, P.G. & Pai, D.K., "Interaction capture and synthesis", *ACM Transactions on Graphics*, 2006, 25(3), pp. 872-880.

[8] Lee, J., Chai, J., Reitsma, P.S.A., Hodgins, J.K. & Pollard, N.S., "Interactive control of avatars animated with human motion data", *ACM Transactions on Graphics*, 2002, 21(3), pp. 491-500.

[10] Magnenat-Thalmann, N. & Thalmann, D., "Virtual humans: thirty years of research, what next?", *The Visual Computer,* 2005, 21(12), pp. 997-1015.

[11] Neff, M. & Fiume, E., "Modeling tension and

relaxation for computer animation", *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, San Antonio, Texas, USA, 2002, pp. 81-88.

[12] Ngo, J.T. & Marks, J., "Spacetime Constraints Revisited", *Proceedings of ACM SIGGRAPH, Anaheim*, USA, 1993, pp. 343-350.

[13] Pollard, N.S. & Zordan, V.B., "Physically based grasping control from example", *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, NY, USA, 2005, pp. 311-318.

[14] Shapiro, A., Pighin, F. & Faloutsos, P., "Hybrid control for interactive character animation", *Proceedings of IEEE Pacific Conference on Computer Graphics and Applications*, Canmore, Alberta, CAN, 2003, pp. 455-461.

[15] Smith, R., Open dynamics engine, http://www.ode.org/, 2008.

[16] Van de Panne, M. & Fiume, E, "Sensor-actuator networks", *Proceedings of ACM SIGGRAPH*, Anaheim, USA, 1993, pp. 335-342.

[17] Van de Panne, M., "Parameterized gait synthesis", *IEEE Computer Graphics and Applications*, 1996, 16(2), pp. 40-49.

[18] Wooten, W.L. & Hodgins, J.K., "Simulation of leaping, tumbling, landing, and balancing humans", *Proceedings of IEEE International Conference on Robotics and Automation*, San Francisco, USA, 2000, pp. 656-662.

[19] Wrotek, P., Jenkins, O.C. & McGuire, M., "Dynamo: Dynamic data-driven character control with adjustable balance". *Proceedings of ACM SIGGRAPH Video Games Symposium*, Boston, USA, 2006, pp. 61-70.

[20] Yin, K., Cline, M.B. & Pai, D.K., "Motion perturbation based on simple neuromotor control models", *Proceedings of IEEE Pacific Conference on Computer Graphics and Applications*, Canmore, Alberta, CAN, 2003, pp. 445-449.

[21] Yin, K., Loken, K. & van de Panne, M., "SIMBICON: Simple biped locomotion control", *ACM Transactions on Graphics*, 2007, 26(3), article 105.

[22] Zordan, V.B. & Hodgins, J.K., "Motion capture-driven simulations that hit and react", *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, San Antonio, Texas, USA, 2002, pp. 89-96.

[23] Zordan, V.B. & Horst, N.V.D., "Mapping optical motion capture data to skeletal motion using a physical model", *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, San Diego, California, USA, 2003, pp. 245-250.

[24] Zordan, V.B., Majkowska, A., Chiu, B. & Fast, M., "Dynamic Response for Motion Capture Animation", *ACM Transactions on Graphics*, 2005, 24(3), pp. 697-701.

[25] Zordan, V.B., Macchietto, A., Medina, J., Soriano, M., Wu, C., Metoyer, R. & Rose, R., "Anticipation from Example", *Proceedings of ACM Virtual Reality Software and Technology,* Newport Beach, CA, USA, 2007, pp. 81-84.

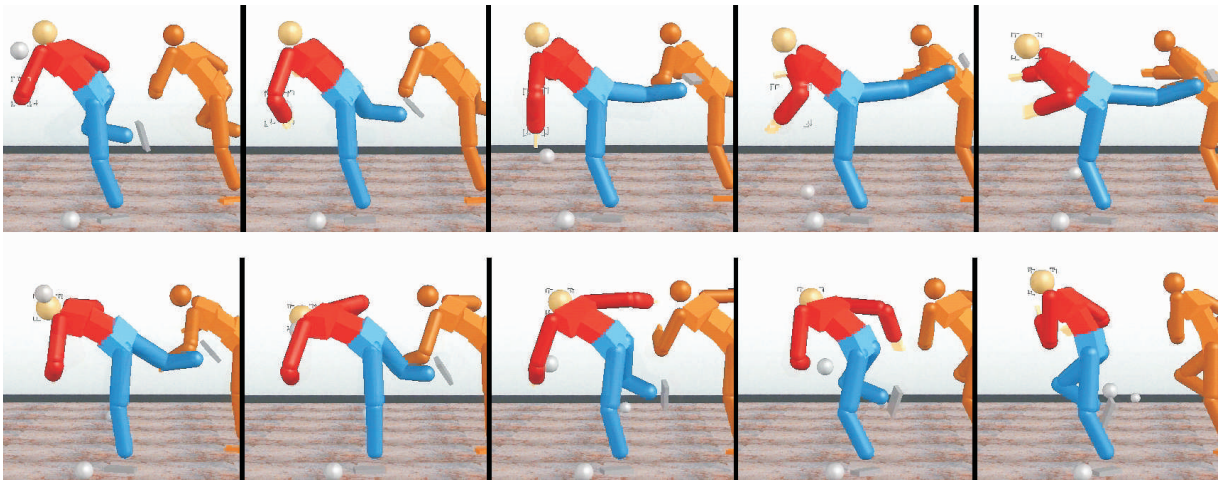


Figura 4. O modelo é atingido inesperadamente no braço e, em seguida, na cabeça, reagindo flexivelmente aos impactos e voltando a seguir o movimento capturado de maneira natural.

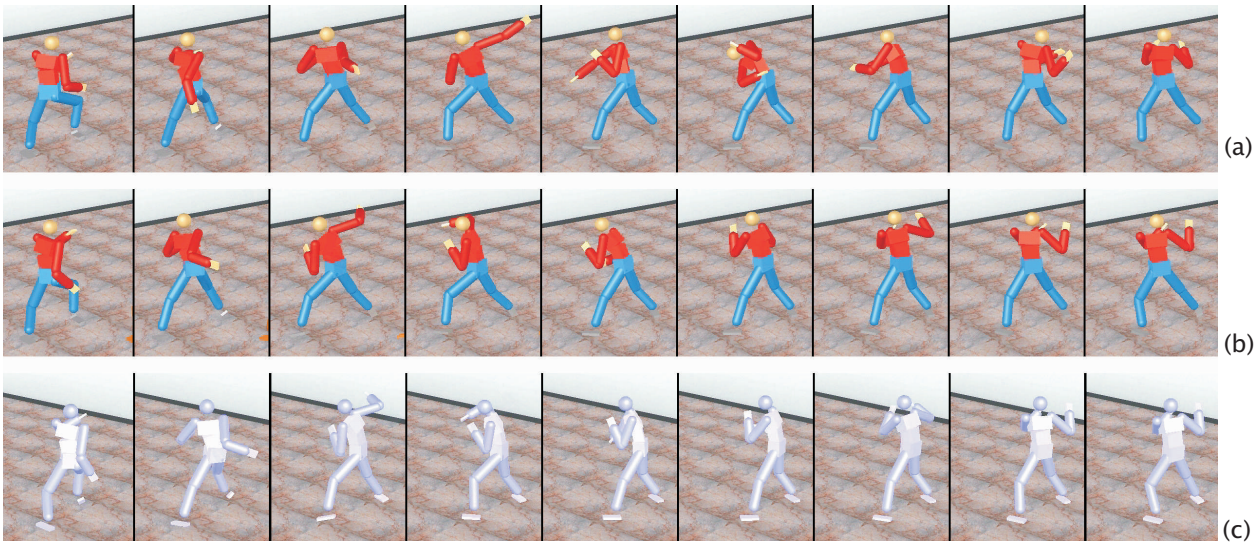


Figura 5. (a) Controle feedback de baixos ganhos sem o termo feedforward; (b) controle feedback de baixos ganhos com o termo feedforward; (c) movimento capturado original.

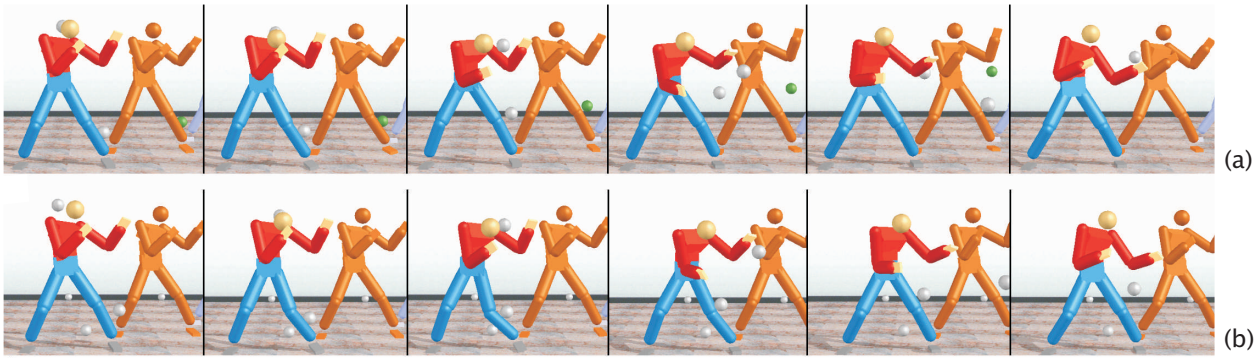


Figura 6. O modelo recebe, ao mesmo tempo, um impacto inesperado na parte de trás da cabeça e um impacto (a – esperado; b – inesperado) na perna.

ReActive – Engine Reativo de Física

Gabriel F. de Almeida, Artur L. dos Santos, Ronaldo F. dos Anjos Filho, Felipe B. Breyer, Mozart W. S. Almeida, Rodrigo C. de Farias, Veronica Teichrieb, Judith Kelner
Grupo de Pesquisa em Realidade Virtual e Multimídia (GRVM)
Centro de Informática (CIn) – Universidade Federal de Pernambuco (UFPE)
{gfa}, {als3}, {mwsa}, {rcf3}, {vt}, {jk}@cin.ufpe.br, {rfdaf}, {felipe}@gprt.ufpe.br

Abstract

Among last generation games, physics simulation engines are a mandatory requisite for realism, and soon will be used on applications other than 3D virtual worlds. This paper describes the ReActive project, an engine that simulates physics using a reactivity approach instead of an iterative one, and provides an abstraction layer for third-party physics engines, diminishing code size to be maintained. It is also possible to extend functionalities and define custom bodies behavior on a simulation with this library. To specify the ReActive engine, the authors present a study related to the main physics simulation and graphics engines, and some prototypes implemented as proof-of-concept, foreseeing the use of physics simulation on collaborative environments.

Resumo

Engines de simulação física vêm ganhando projeção com os jogos de última geração, e em breve poderão ser incorporados a outros tipos de aplicações, na busca de comportamentos mais realistas além dos ambientes virtuais 3D. Este artigo descreve o projeto ReActive, um engine de simulação física que engloba elementos inovadores, como o conceito de reatividade em lugar de iteratividade para aplicações que executam simulação física, além de uma camada de abstração para engines físicos de terceiros, diminuindo a quantidade de código a ser mantido. Também é possível estender as funcionalidades disponíveis e definir comportamentos específicos para corpos numa simulação, tudo através da biblioteca. Para especificar o ReActive, apresenta-se o estudo realizado dos principais engines de simulação física e motores gráficos, além de protótipos implementados como prova de conceito, prevendo o uso de engines físicos em ambientes colaborativos.

1. Introdução

Este artigo descreve o projeto ReActive, um *engine* (motor) que inclui, entre outros elementos inovadores, o conceito de reatividade para o desenvolvimento de aplicações que executam simulação física. *Engines* de simulação física vêm ganhando projeção entre vários gêneros de *software*, principalmente nos jogos de última geração, e em breve poderão ser incorporados a outros tipos de aplicações, principalmente as que buscam por comportamentos mais realistas não só em ambientes virtuais 3D, mas também em interfaces de interação.

A utilização desses *engines* é possível pela constante evolução do poder de processamento das CPUs e das alternativas que foram criadas para execução de código de simulação física, utilizando *hardware* especializado [1] . É importante considerar também que há em curso uma mudança de paradigma no desenvolvimento dos processadores de uso doméstico. A indústria vem se preocupando em criar mais poder de processamento paralelo, tendo em vista os limites físicos para o crescimento da velocidade dos processadores [2]. Desta forma, é possível prever que elementos como simulação física e inteligência artificial, que tinham seu desenvolvimento restrito por não dispor de tempo de processamento suficiente para execução em aplicações de tempo real, devem avançar bastante.

Com a possibilidade de executar cálculos físicos em tempo real, e com maior fidelidade do que era anteriormente possível, a quantidade de *engines* físicos cresceu através de bibliotecas de código aberto ou fechado, comerciais ou gratuitas. E, assim como a variedade, o poder de escolha dos desenvolvedores de *software* que incorporam comportamento físico também aumentou. Cada um desses *engines* apresenta soluções próprias e é difícil migrar o código de uma biblioteca para outra (ou até mesmo entre diferentes versões da mesma biblioteca), por isso a escolha por algum deles tem que ser baseada nas necessidades do aplicativo a ser desenvolvido.

É neste espaço que o ReActive surge como uma solução para prover ao programador uma interface extensível e compatível com diversos *engines* de simulação física, além do modelo reativo de atualização. Estas características se traduzem em menor quantidade de código a ser mantido, que também é uma das metas do projeto ReActive.

Na próxima seção, será fornecida uma breve visão de alguns conceitos ligados ao desenvolvimento do ReActive. Na seção 3, são apresentados os trabalhos relacionados, como *engines* gráficos e físicos. A seção 4 apresenta a proposta do ReActive. A seção 5 enumera características incorporadas ao ReActive através da análise de outros *engines*, enquanto a seção 6 descreve os testes e protótipos para provas de conceito que foram implementadas. Na seção 7 são apresentados alguns cenários para utilização do ReActive, e a seção 8 apresenta as conclusões e trabalhos futuros.

2. Conceitos relacionados

Pode-se definir um *engine* físico como um sistema computacional desenvolvido com a finalidade de simular em máquina a Física Clássica seguindo o modelo Newtoniano, ou seja, utilizando os conceitos básicos de massa, velocidade, atrito e resistência do ar . Com base neste conceito, um *engine* físico é responsável por realizar todas as simulações relacionadas ao movimento de partículas e corpos, além das suas inter-relações, como colisão e campos de força. Tais *engines* geralmente são componentes de um sistema mais abrangente que, tirando proveito das informações resultantes na simulação física, apresentam uma visualização realística dos resultados ao usuário.

O grau de realismo que um *engine* físico pode oferecer está diretamente relacionado com o nível de precisão matemática da sua implementação. A partir deste conceito é possível dividir então os *engines* físicos em dois grupos: simuladores de alta precisão e simuladores em tempo real. Os *engines* físicos de alta precisão envolvem simulações que requerem maior poder de processamento para realizar cálculos físicos bastante precisos e próximos da realidade, sendo utilizados principalmente no meio acadêmico, entre físicos e astrônomos [4]. Tais *engines*, por serem partes de sistemas que requerem alto poder de processamento, não possibilitam a realização de simulações em tempo real, diferenciando-se da classe de *engines* físicos *real-time* [3]. Como exemplo clássico de simulações *real-time* tem-se os jogos eletrônicos, que precisam de tais *engines* para dar uma aparência mais realista ao produto final, mas com pouco impacto no desempenho do jogo.

Entretanto, o interesse no realismo dos jogos vem crescendo, sendo uma conseqüência do avanço no poder de processamento dos computadores domésticos. Este avanço tem permitido aos desenvolvedores de jogos utilizarem técnicas modernas de simulação física, que em conjunto com o ambiente gráfico surpreendem no grau de realismo [5].

Como tais *engines* geralmente são componentes de um sistema mais complexo, é importante que tais *engines* físicos ofereçam interfaces bem estruturadas, fáceis de usar corretamente, pois aumentam a qualidade do sistema por prezarem pela qualidade de *software* [6]. Assim, diversos *engines* utilizam, por exemplo, o conceito de *Factory* para a instanciação dos corpos físicos, oferecendo então um sistema de gerenciamento de recursos seguro e eficiente.

A existência de um ponto de entrada ou configuração do *engine* físico favorece o uso de *Singleton*, já que geralmente se deseja uma única instância do gerenciador. O uso de Adapters para compatibilizar interfaces diferentes também é comum nos *engines* físicos, já que tais componentes são utilizados em diferentes tipos de sistemas.

3. Trabalhos relacionados

Antes de iniciar o desenvolvimento do projeto ReActive, foi preciso fazer um levantamento dos principais *engines* físicos, gráficos e binders (bibliotecas de acoplamento entre objetos físicos e gráficos) existentes na literatura, a fim de coletar características que estes possuem em comum. Nas subseções seguintes serão descritos alguns dos *softwares* estudados para a concepção do ReActive.

3.1. *Engines* físicos

Dentre os vários *engines* físicos, a biblioteca Bullet [7] provê a detecção de colisão e a dinâmica de corpos rígidos, e é dividida em dois módulos, sendo um para cada funcionalidade mencionada. Esta divisão é refletida na estrutura de diretórios e subdiretórios, para melhor representar os sub-módulos desta biblioteca.

Uma característica interessante dessa divisão é que o módulo de detecção de colisão pode ser usado separadamente do módulo responsável pela dinâmica, ou seja, os desenvolvedores podem integrar ao Bullet o sistema de detecção de colisão mais adequado para a sua aplicação.

A biblioteca ODE [8] (*Open Dynamics Engine*) é um simulador de corpos rígidos articulados. Uma estrutura articulada pode ser representada por corpos rígidos

conectados através de juntas. Como exemplo, pode-se citar um veículo, com as rodas conectadas ao chassi; outro exemplo seria uma criatura com pernas conectadas ao corpo. O ODE foi desenvolvido para ser usado em simulações em tempo real, onde, tanto os objetos quanto o cenário são móveis.

Como o Bullet, a biblioteca ODE oferece, também, um sistema de colisão embutido que pode ser facilmente substituído. Este sistema de colisão permite rápida identificação de objetos que potencialmente podem colidir via o conceito de espaços.

Ao contrário dos outros *engine*s que primam pela velocidade da simulação, Newton [9] é um *engine* físico que possui maior precisão nos cálculos de suas simulações. Gratuito, mas de código fechado, ele foi desenvolvido para simulação realística de corpos rígidos em jogos ou em outras aplicações de tempo real, além de possuir um solver determinístico. O Newton é bastante difundido, sendo uma escolha popular em diversas comunidades.

Por último, o AGEIA PhysX [1] é um poderoso *middleware* de física, baseado em *software*, para a criação de ambientes físicos dinâmicos. O PhysX, apesar de gratuito, possui o código fechado e suporta as principais plataformas para jogos e aplicações gráficas, tais como PS3, XBOX, PC, entre outras.

Pela abrangência de plataformas, facilidade de programação e diversas outras *features*, o uso deste *engine* é bastante difundido, sendo usado em diversas aplicações e jogos comerciais. A placa AGEIA PhysX foi a primeira PPU (Physics Processing Unit) dedicada para computadores desktop, construída para melhorar a física contida em aplicações que usam este *engine*.

3.2. *Engines* gráficos

A análise de *engines* gráficos foi realizada principalmente por causa do uso deste tipo de biblioteca em conjunto com *engines* físicos, onde o primeiro realiza a tarefa de exibição, e o segundo, de simulação. Além disso, *engines* gráficos comumente inserem uma camada de abstração sobre outras bibliotecas de exibição, tarefa que o ReActive quer executar com *engines* físicos.

O OGRE (*Object-Oriented Graphics Rendering Engine*) [10], que é um *engine* gráfico 3D, orientado a cena, abstrai o uso das bibliotecas Direct3D e OpenGl, provendo uma interface de programação baseada em world objects e outras classes de fácil entendimento.

A arquitetura deste *engine* é baseada em *plugins* e fornece ao programador uma interface para implementação dos mesmos, permitindo adicionar novas funcionalidades como captura e reprodução de vídeo e áudio, por exemplo.

Como wrapper, o OGRE oferece uma camada de abstração entre o programador e as bibliotecas que ele abstrai, distanciando em diversos níveis o primeiro das minúcias das segundas, não permitindo acesso ao código destas, e gerenciando de forma automática a expansão e aceleração oferecidas pela placa de vídeo.

O Horde3D [11] é um *engine* de renderização *open source* (LGPL) que, diferentemente do OGRE, concentra-se em efeitos visuais encontrados nos jogos atualmente. Além de ser leve e possuir uma interface simples e intuitiva, o *engine* é adequado para a renderização de uma grande quantidade de objetos animados. Horde tem o apoio da Universidade de Augsburg, na Alemanha e, da mesma forma que o OGRE, abstrai o uso de OpenGl em suas aplicações.

O *Crystal Space* [12], assim como o Horde, foi desenvolvido para ser um *engine* para construção de jogos 3D em tempo real, além de ser independente de plataforma. Sua arquitetura é baseada em plugins, possuindo diversos destes já implementados, conferindo abstração ao programador e funcionalidades comumente usadas em games, tais como renderização gráfica 3D, reprodução de som, simulação de objetos físicos (utilizando ODE ou Bullet) e união (*bind*) destes com objetos gráficos, dentre outras.

3.3. Outros tipos de *engines*

Outra característica importante do ReActive, e que beneficiará desenvolvedores, é o fato de eles não precisarem mais se preocupar em sincronizar os objetos gráficos de sua aplicação com as entidades físicas que estão sendo simuladas.

Tal característica é encontrada no NxOgre [13], uma biblioteca que une (bind) o *engine* físico PhysX e o *engine* gráfico OGRE, a fim de agilizar o desenvolvimento de aplicações físicas, abstraindo alguns conceitos e reduzindo sensivelmente o código.

A principal vantagem do NxOgre reside no fato do programador não precisar sincronizar manualmente os objetos gráficos e físicos, sendo necessária apenas a criação de um objeto que os abstrai, enquanto a sincronização fica a cargo da biblioteca.

O PAL (Physics Abstraction Layer) [14] foi desenvolvido como uma camada de abstração para o uso de diversos *engines* físicos, permitindo assim ao programador maior flexibilidade sobre qual *engine* usar em sua aplicação, ou até mesmo trocá-lo por outro caso seja necessário. Com arquitetura baseada em *plugins*, pode-se facilmente adicionar ao PAL qualquer outro

novo *engine* físico que seja desenvolvido futuramente.

4. ReActive

Esta seção descreve os principais conceitos incorporados no ReActive para suprir as necessidades dos seus usuários. A sua arquitetura, componentes e infra-estrutura serão descritas para exemplificar o seu funcionamento. No momento atual, o ReActive está em fase de implementação dos elementos descritos a seguir, mas vários protótipos foram implementados como prova de conceito e estão descritos na seção 6.

O projeto ReActive tem como principal característica a adoção de um modelo reativo para a atualização dos objetos, ou seja, a posição dos objetos visuais são retornados ao usuário apenas quando seus objetos físicos correspondentes sofrem algum tipo de modificação. Esta abordagem é diferente da abordagem iterativa, que atualiza todas as posições de todos os objetos a cada ciclo de execução. Nesta primeira versão, apenas elementos compostos de corpos sólidos e juntas serão contemplados no projeto, sendo extensível para a adição de outros elementos como corpos flexíveis e fluidos, em uma futura versão.

Sempre que possível, se fará uso do padrão de *factories*, centralizando a construção automática de objetos da cena e controlando os identificadores de cada instância, eliminando um ponto de falha do controle do usuário. Porém, caso ache necessário, o programador terá a liberdade de criar e nomear seus próprios objetos de cena, tomando para si a responsabilidade de manter a consistência dos dados e gerenciar as instâncias criadas.

A robustez é um dos fatores mais relevantes do ReActive e, para isso, algumas regras foram definidas: evitar o uso de referências cíclicas e passagem de *strings* como parâmetros, apesar dessas práticas terem sido observadas em outros *engines*.

Durante a especificação do ReActive, também foi verificada a necessidade de um modo de *debug* visual, que exponha elementos físicos abstratos, como vetores de força ou eixos de rotação, com primitivas gráficas para que a sintonia fina das cenas seja mais intuitiva. Este *debugger* ainda não foi implementado, mas foi definido como um requisito importante para o ReActive em futuras versões.

O ReActive será distribuído em forma de biblioteca para ser incorporada e utilizada como suporte a programação, reduzindo ao mínimo possível a queda de performance da aplicação e trazendo benefícios na produtividade da equipe para diminuir os custos dos projetos.

4.1. Arquitetura

A arquitetura do ReActive é orientada a plugins para os *engines* de simulação física, permitindo a extensão do projeto para suportar novas tecnologias. A camada de abstração sobre os *engines* físicos obedecerá a um conjunto de interfaces, simplificando a implementação de sistemas de detecção de colisão e possibilitando a atribuição de comportamentos específicos para os objetos.

A entidade *ReActiveManager* é o ponto de entrada da biblioteca, como pode ser visto na Figura 1, que fornece os serviços de gerenciamento de materiais, *scripts*, comportamentos (*behaviors*) e *core plugins*, sendo o último a entidade principal com a qual o usuário mais irá interagir.

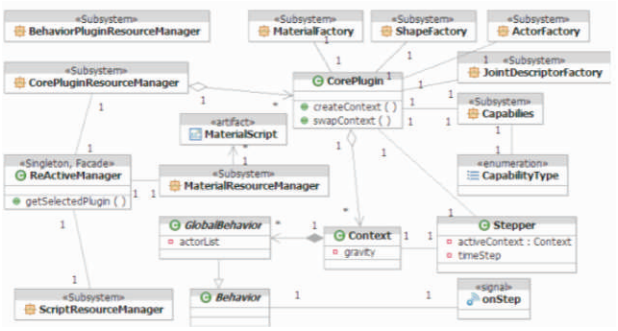


Figura 1. Diagrama de classe do ReActive.

Os *core plugins* são entidades que representam as abstrações sobre os *plugins* dos *engines* físicos. A partir deles é possível ter acesso às funcionalidades particulares de uma biblioteca de simulação física e ao *Stepper*, entidade responsável pela atualização do contexto da cena.

contexto da cena.

O contexto representa uma cena modelada, com atores estáticos e dinâmicos, juntas e *behaviors* globais, de juntas e de atores. É pelo contexto que os *behaviors* conseguem atuar na cena.

Os *behaviors* são adições aos comportamentos físicos já existentes na simulação, podendo ser usados, por exemplo, para simular a presença de fluidos como água ou petróleo em uma cena. Existem três tipos de *behaviors*: globais, de juntas e de atores. Cada um possui características específicas; os globais, como o próprio nome diz, agem sobre todos os atores do contexto; os de atores desempenham suas tarefas (e.g., um campo de força que um determinado ator não pode atravessar) sobre um ator específico ou uma região do espaço 3D para ativar o comportamento determinado; e os *behaviors* de juntas são usados para definir conexões

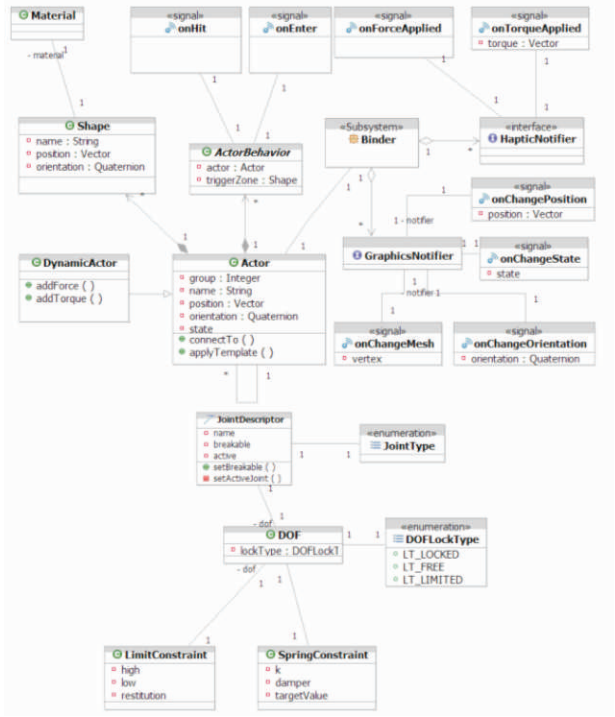


Figura 2. Entidades relacionadas com Actor

quebráveis, que muitos *engines* físicos implementam.

Um ator é uma combinação de primitivas e modelos que formam uma entidade física única, como um carro ou uma bicicleta. Podem ser de dois tipos: dinâmicos e estáticos.

Os atores estáticos possuem propriedades como posição e orientação, além da função de se conectar a outro ator através de juntas, e uma função de aplicação de *behavior* de ator (*applyTemplate*). Como pode ser visto na Figura 2, os *behaviors* de atores definem dois eventos de colisão: *onHit* e *onEnter*, sendo ativado o primeiro quando o ator colide com algo e o segundo quando a *trigger zone* do ator colide com outro *behavior* ou ator.

Os atores dinâmicos podem ser representados como corpos rígidos, diferindo de atores estáticos apenas pelas propriedades dinâmicas de um corpo rígido e funções de acúmulo de força e torque.

Além disso, pode-se associar um *script* a um ator. *Scripts* são descrições de alto nível das propriedades dos atores, como materiais envolvidos, *behaviors* associados e características do corpo rígido.

dos atores, como materiais envolvidos, *behaviors* associados e características do corpo rígido.

As juntas só podem ser criadas via descritores de juntas (*JointDescriptors*), que possuem informações sobre o tipo

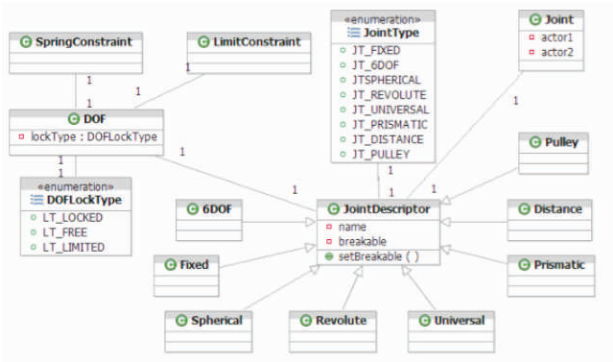


Figura 3. Tipos de juntas.

da junta, quais atores serão conectados, se é quebrável e graus de liberdade da junta. Isso significa que o modelo de juntas do ReActive leva em consideração a possibilidade de criação de uma junta restringindo-se os graus de liberdade da mesma.

Os tipos possíveis para a criação de juntas são: *Fixed*, *6DOF* (*six degrees of freedom*), *Spherical*, *Revolute*, *Universal*, *Prismatic*, *Distance* e *Pulley*, que podem ser vistos na Figura 3.

Em conjunto com os *JointDescriptors*, existe a entidade DOF, que representa o tipo de restrição que será aplicado a um determinado grau de liberdade da junta, e nessa versão do Reactive serão disponibilizados apenas dois tipos de restrição: restrição de limite e de mola. Uma restrição de limite, como o próprio nome diz, restringe os movimentos possíveis de uma junta, já as restrições de mola adicionam um efeito de mola numa junta.

A entidade binder é a responsável pela função de *middleware* da biblioteca, fornecendo a abstração da representação de ator e retorno de cálculos de posição para as aplicações que utilizam a biblioteca. Nessa versão do ReActive serão oferecidos dois notificadores: háptico e gráfico. O notificador háptico é associado a eventos, como a força e o torque de colisões calculados e enviados para *feedback*. O notificador gráfico também trata mudanças de posição, orientação ou mudanças de estado como eventos.

As formas disponíveis para uso são: malha triangular (*triangle mesh*), cone, cilindro, cápsula, *heightfield*, esfera, formas convexas (*convex hull*), plano e cubo. Todas as formas possuem uma referência a uma classe Material e descendem da classe Shape. A única exceção é a malha de triângulos (*TriangleMesh*) que, além das características citadas, possui também uma referência para objetos do tipo *TriangleMeshData*.

5. Análise e levantamento de características de similares

Esta seção apresenta características de *engines* físicos que foram estudados.

5.1. Metodologia

A principal proposta do ReActive é abstração, seja em relação à qual *engine* físico está sendo utilizado ou à como os objetos físicos irão atualizar os objetos gráficos da aplicação. Por este motivo, foi preciso definir metodologias diferentes de análise para os diferentes tipos de *engines* estudados.

Para análise dos engines físicos foram levados em consideração os pontos semelhantes de cada um deles, dando ênfase em quais as formas dos objetos físicos e quais os tipos de juntas disponibilizadas.

Já com relação aos *engines* gráficos, apesar de não possuírem ligação com o ReActive quanto as funcionalidades, foram observados os pontos fortes e fracos de cada um deles, como também detalhes de suas arquiteturas que poderiam ser úteis para o desenvolvimento do ReActive. Os mesmos critérios foram usados para o estudo do binder NxOgre.

5.2. Levantamento de características

As formas geométricas encontradas em comum para todos os *engines* físicos analisados foram *box*, *sphere*, *capsule* e *convex*. Formas do tipo *plane*, *heightfield* e *triangle mesh* foram encontradas em todos os *engines* com exceção do Newton, mas tanto este quanto o Bullet oferecem as formas *cone* e *cylinder*, enquanto o PhysX oferece ainda *wheel shapes*. Todas as formas citadas serão implementadas no ReActive, mesmo que originalmente algum *engine* específico não a suporte.

Com relação a juntas, apenas os tipos *Ball-and-socket* e *revolute* são oferecidas por todos os *engines* analisados, enquanto o tipo prismatic só não é oferecido pelo Bullet e o tipo 6DOF só não é oferecido pelo ODE. Do mesmo modo que as formas geométricas, serão implementados para o ReActive todos os tipos de juntas encontrados nos *engines* analisados, mesmo que não exista implementação destas em algum deles.

Após a análise das características comuns dos *engines* físicos, foi feita uma análise, tanto destes como também do binder e dos *engines* gráficos, a respeito de arquitetura e padrões de projeto utilizados em suas implementações.

Em relação a padrões de projeto foi observado que a maior parte destes *engines* utiliza os padrões *Singleton* e *Factory* [15], provendo maior segurança à aplicação e facilidade de implementação aos desenvolvedores. Já com relação à arquitetura, foram observados três conceitos interessantes a serem utilizados na implementação do ReActive: Integration Library (Horde), *plugins* (OGRE) e *blueprints* (NxOgre).

A Integration Library oferece o conceito de node attachments. Um *attachment* é uma classe derivada que pode ser anexada aos *scene nodes* do Horde, contendo dados específicos da aplicação, como uma propriedade física, por exemplo. Essa classe é gerenciada pelo engine e, quando a cena é atualizada, pode-se saber se houve transformações nos nós pais ou filhos. No caso do ReActive, essa integração seria no sentido inverso, atrelando entidades gráficas às entidades físicas, e o gerenciamento delas seria executado pelo ReActive.

O OGRE, por sua vez, possui uma interface de programação para plugins, oferecendo aos desenvolvedores a possibilidade de estender o *engine* com *features* não suportadas por ele anteriormente, o que pode ser uma característica interessante a ser adicionada ao ReActive.

Finalizando, o conceito de *blueprints*, oferecido pelo NxOgre, define objetos "descritores" que possuem informações estruturais dos objetos que eles descrevem. Tais informações são utilizadas por *factories* específicas para a criação das entidades físicas gerenciadas pelo PhysX, como também a qual objeto gráfico do OGRE esta entidade está relacionada. Este conceito trará maior segurança e facilidade para os desenvolvedores que utilizarem o ReActive em suas aplicações.

Além das características supracitadas, uma documentação simples e atualizada também é importante, e foi um ponto bastante discutido pela equipe ao longo da implementação dos protótipos e análise de similares.

6. Prototipagem e testes

A simulação física é peça fundamental em muitos jogos recentes, sendo conseqüência direta da busca constante pelo aumento no grau de realismo em tais aplicativos. Entretanto, a inserção de tais *engines* em um contexto mais complexo nem sempre ocorre de maneira trivial. Um jogo *multiplayer*, por exemplo, traz a dificuldade adicional de se implementar uma interface entre o *engine* físico e a camada de comunicação. Dependendo da forma de implementação, a quantidade de objetos físicos instanciados pode comprometer o desempenho do jogo e da rede envolvida.

Tendo em vista tais dificuldades de integração, uma alternativa para minimizar estes problemas é a criação de uma interface única entre o *engine* físico e a camada

de comunicação. Desta forma, evita-se a realização de duas integrações separadas – uma com o *loop* do jogo e outra com a camada de comunicação – para ter-se apenas uma integração. O ReActive propõe então o suporte a uma camada de abstração de rede, possibilitando a implementação colaborativa de forma mais simples e otimizada, utilizando técnicas eficientes de sincronização.

Para a implementação eficiente da camada de comunicação, é preciso observar principalmente dois fatores do ambiente no qual o sistema estará contextualizado. O primeiro se refere à largura de banda disponível, pois para algumas das técnicas de sincronização, a taxa de transmissão é proporcional ao número de objetos ou instâncias físicas presentes na simulação. Assim, para simulações complexas, a largura de banda pode alcançar o nível de saturação. O segundo fator está relacionado à latência da rede, problema bastante freqüente nos jogos *multiplayer*, em que os jogadores geralmente estão conectados por diversos tipos de enlaces, como ISDN, xDSL e WiFi. Uma rede com alta latência reduz a performance do sistema, podendo inviabilizar o seu uso.

Para contornar tais problemas, é preciso implementar técnicas de sincronização que busquem aperfeiçoar o desempenho do sistema e da própria rede . Em relação às simulações físicas, três técnicas são bastante utilizadas: sincronização por objetos remotos, *lockstep* e *bucket synchronization*.

A sincronização por chamada a objetos remotos é uma das técnicas mais utilizadas em jogos e sistemas com poucas instâncias de objetos. Utiliza a idéia de centralizar todos os cálculos importantes do sistema em uma máquina, enquanto as outras receberão os estados de todas as instâncias de objetos do programa. Desta forma, as máquinas que recebem as informações não precisam realizar os cálculos que a máquina central executa, pois já recebe os resultados prontos.

Entretanto, uma grande desvantagem neste tipo de sincronização é a necessidade de largura de banda suficiente para transmitir todo o estado atual (dados das instâncias dos objetos) do sistema. Um sistema que trabalha com muitas instâncias necessitará de uma rede com alta largura de banda, o que muitas vezes não corresponde à realidade.

Para resolver o problema citado anteriormente, podem-se utilizar técnicas que utilizam o conceito de simulação simultânea . Ao invés de passar o estado de cada objeto remoto, executa-se exatamente a mesma simulação em cada máquina, passando para cada uma um conjunto idêntico de comandos ou eventos que serão calculados ao mesmo tempo nas diferentes máquinas. Esta forma de sincronização é complexa para ser implementada pelos requisitos de consistência de dados entre as máquinas, mas traz benefícios para a rede, já que é bem menos custoso enviar eventos do que todos os estados dos objetos remotos envolvidos na simulação.

A técnica de sincronização por objetos remotos foi implementada e testada num aplicativo demo do projeto VisGas, em que se exibe a simulação de *risers*, estruturas utilizadas para extração de petróleo e gás entre o solo oceânico e a superfície. Utilizando a arquitetura cliente-servidor, criou-se um sistema no qual o servidor realiza todos os cálculos da simulação física, e envia os estados de orientação e posição dos 600 objetos (cápsulas e esferas) aos clientes. O *engine* físico (neste caso o PhysX) é utilizado apenas no servidor. No lado cliente, ocorre somente o funcionamento do *engine* gráfico OGRE, responsável pela renderização gráfica e exibição na camada de apresentação.

A visualização ocorre a taxas elevadas, podendo exceder a 200 *frames* por segundo, dependendo das configurações da máquina usada. Entretanto, para evitar altas taxas de transmissão, o envio dos estados é limitado em 40 Hz. Como a visão humana é capaz de perceber diferenças de imagem no mínimo em cerca de 1/30 segundo, não ocorrem descontinuidades perceptíveis ao olho humano, já que a cada 1/40 segundos um novo quadro é processado e exibido. Entretanto, mesmo limitando o número de envios por segundo, a taxa de transmissão alcança valores em torno de 2,54 Mbps, como visto na Figura 4.

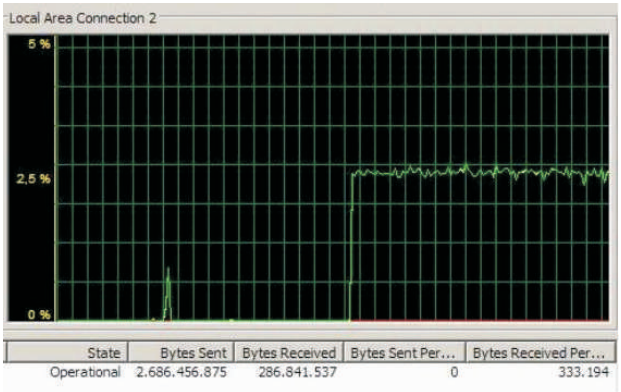


Figura 4. Taxa de transmissão do protótipo.

Na técnica de *Lockstep* , cada usuário espera pelos eventos dos outros participantes do sistema. Enquanto o usuário não receber todos os eventos dos outros, o mesmo não pode passar para o próximo *frame*. Isto é valido para todos os usuários do sistema.

A grande vantagem dessa técnica, ilustrada na Figura 5, é a garantia de pequenas taxas de transmissão, já que apenas eventos são repassados. Para isto, é preciso que todas as partes formem um sistema determinístico, ou seja: se um evento A ocorre no usuário 1 e o mesmo vai do estado X para o estado Y, o usuário 2 também deverá processar o evento A e passar do estado X para o estado Y.

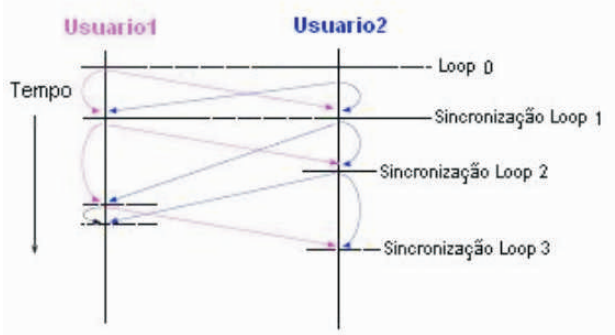


Figura 5. *Lockstep* synchronization.

A técnica de *Lockstep* pode ser utilizada em um modelo cliente-servidor e também num modelo *peer-to-peer* [17]. O modelo cliente-servidor necessita que todos os clientes enviem ao servidor todos os eventos, que então serão repassados para os demais clientes. Já em *peer-to-peer*, cada usuário envia seus eventos diretamente para todos. Esta forma é ainda mais vantajosa em redes que suportam multicast.

Entretanto, independente da arquitetura escolhida ser cliente-servidor ou *peer-to-peer*, existem alguns problemas sérios ao se utilizar *Lockstep*. Um deles é quando uma das partes da rede sofre de alta latência, pois o tempo de resposta final para cada *loop* corresponde ao tempo de resposta da parte da rede que estiver mais lenta.

Em *Bucket Synchronization* [18], similarmente ao *Lockstep*, cada usuário espera pelos eventos dos outros participantes do sistema. Entretanto, a diferença entre as duas técnicas está na existência de um *buffer* dos eventos recebidos em cada uma das partes do sistema. O próprio nome da técnica (bucket – balde em inglês) se refere a este *buffer*. Quando um evento chega, é armazenado no *buffer* e será calculado e renderizado apenas em *frames* futuros. Como exemplo, quando os eventos do *loop* 1000 chegarem, os mesmos só serão executados no *loop* 1002. Assim, caso aconteça um atraso em uma das partes, a simulação continuará acontecendo já que existe um *buffer* de eventos que fornecem as informações para que a simulação ocorra.

A técnica *Bucket Synchronization* foi utilizada no famoso jogo Age Of Empires, da Microsoft [18]. Seus desenvolvedores realizaram uma pesquisa com usuários em relação à percepção dos eventos sob diferentes níveis de latência. Ao observarem que um intervalo de tempo em torno de 100 a 200 milissegundos não era perceptível aos usuários e a taxa de atualização do jogo deveria ser de 20 Hz, definiram um *buffer* para dois eventos, já que não seria perceptível para o usuário caso o evento ocorresse 100 milissegundos depois.

As técnicas de sincronização orientadas a eventos requerem que a simulação ocorra da mesma forma em todas as suas partes. É preciso então ter a garantia de determinismo, em que todo o código envolvido deve realizar as mesmas operações ou seqüências de instruções caso seja chamado na mesma ordem várias vezes. Sendo assim, se o mesmo conjunto de instruções for executado n vezes, os n estados finais do programa serão idênticos a cada execução.

Os problemas para alcançar estas garantias surgem quando o sistema utiliza bibliotecas externas. Muitas vezes tais bibliotecas são complexas, oferecendo uma gama de serviços difíceis de serem testados quanto ao determinismo. É preciso então realizar testes de todas as funcionalidades utilizadas em tais bibliotecas, analisando a arquitetura da máquina usada para testes, sistema operacional, e versões das bibliotecas.

Um sistema colaborativo de simulação em tempo real, que utiliza técnicas de sincronização como *lockstep* e *timebucket*, precisa então dispor de *engines* físicos que garantam o determinismo do sistema.

Em relação ao Reactive, para garantir tais requisitos, serão realizados testes com os *engines* físicos envolvidos. Testes com o engine PhysX já foram realizados, tanto em simulação por *hardware* (usando a PPU da AGEIA) como por *software*.

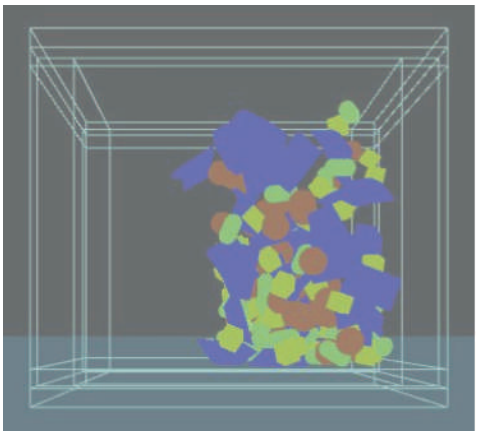


Figura 6. Cena física de testes.

Para facilitar o processo de testes de colisões entre instâncias físicas, criou-se uma cena física com uma

caixa oca, visualizada na Figura 6. Na sua região interna, estão inseridos corpos rígidos, como caixas, esferas, cápsulas e pirâmides, confinados a um subespaço. Definindo movimentos que atuem sobre a caixa, causam-se diversos tipos de colisões entre os corpos. Capturando as posições de tais corpos rígidos durante uma variação de tempo e executando a mesma simulação repetidamente, torna-se possível detectar a existência ou não de variações nos resultados. No teste realizado, a caixa executava um movimento harmônico simples (MHS) [19].

O levantamento dos dados ocorreu da seguinte forma: a cada 60 ciclos do *loop* da simulação, eram enviadas para um arquivo de saída as matrizes de orientação e posição de todas as formas físicas na cena. Depois de duas execuções em máquinas diferentes, os arquivos foram comparados. Se o resultado corresponder a arquivos idênticos, significa que há uma alta probabilidade dos tipos de objetos e tipos de colisões agirem de forma determinística. Caso contrário, significa que o sistema não é determinístico.

Após duas séries de testes de 5, 10 e 15 minutos, a simulação em *software* resultou em arquivos idênticos para todas as saídas. Para garantir um resultado mais preciso, outra bateria de testes foi realizada, mas desta vez com intervalos de tempo de 8 horas. Novamente, as saídas resultaram em arquivos idênticos. Tais resultados indicam alta probabilidade do PhysX se comportar de maneira determinística, pelo menos em relação aos corpos rígidos e suas colisões.

Já com a simulação em *hardware*, utilizando a mesma cena dos testes em *software*, os resultados são diferentes. Simulando várias vezes na mesma máquina, as saídas não são semelhantes. Entretanto, simplificando o ambiente para apenas um cubo, a simulação se comporta de maneira determinística. Adicionando mais um cubo, a simulação começa a apresentar um comportamento aleatório. A partir destas observações, infere-se que a implementação do sistema de colisão em *hardware* não processa os dados de maneira estática, podendo gerar resultados completamente diferentes para entradas iguais. Já a simulação em *software* demonstrou resultados idênticos em todos os testes realizados.

7. Aplicações

Diante das características apresentadas, aplicativos como simuladores de vôo, sistemas dedicados ao treinamento de mão de obra em engenharia são bons exemplos de programas que podem explorar as inovações oferecidas pelo ReActive, se encaixando neste perfil o projeto VisGas. Projetos de novos jogos que buscam elevar o grau de realismo também poderão ser beneficiados pelo uso do ReActive nas suas implementações.

Outro fator importante no ReActive será a integração direta entre os *engines* físicos e uma camada de comunicação para sistemas colaborativos. O ReActive propõe o suporte à uma camada de abstração de rede, possibilitando a colaboração entre aplicações de simulação física de forma mais simples e otimizada. Assim, jogos com suporte ao modo *multiplayer* e aplicativos para treinamentos e demonstrações poderão utilizar o ReActive para executar simulação física através da rede, para usuários que se encontram em diferentes localidades.

O conceito de reatividade também favorece aplicativos que usam esse modelo de programação, como interfaces de interação, sejam elas 3D ou 2D. O uso de simulação física para interfaces gráficas com o usuário ainda não é difundido, mas devido ao crescente poder de processamento, pode ser num futuro próximo. Janelas poderiam ter suas animações regidas por simulações físicas simplificadas, como o deslizamento de um corpo rígido influenciado pela ação da gravidade. Botões poderiam ter propriedades como densidade e coeficientes de atrito, interagindo não só com a interface gráfica, mas também com o sistema de visualização de um aplicativo 3D.

8. Conclusão e trabalhos futuros

Dentre várias contribuições, o ReActive se destaca principalmente pelo modelo de programação simples, que objetiva reduzir a quantidade de código fonte a ser escrita, facilitando sua manutenção e reduzindo o tempo de desenvolvimento.

O conceito de reatividade, que abstrai a existência de um ciclo fixo de operações para um paradigma no qual se baseia nas interações e conseqüentes respostas (reações) entre os objetos físicos envolvidos, define uma forma intuitiva e simples de se desenvolver um sistema de simulação física em tempo real.

A escolha de um engine físico para determinada cena na simulação é mais um fator positivo para o uso do ReActive, sendo uma camada de abstração para a manipulação das cenas ou contextos físicos. Como cada engine físico disponibiliza uma interface diferente para o seu uso, implementar um código que possibilite alternar entre diferentes bibliotecas é um processo complexo e demorado. Com o ReActive, será possível a troca do engine ativado em uma cena física de forma simples, escolhendo o mais adequado para cada tipo de

simulação.

Por causa da extensão do projeto, alguns requisitos ainda precisam ser definidos, mas já estão previstos, como o visual debugger e a especificação do protocolo de colaboração, contemplando o problema de não-determinismo de alguns *engines* de simulação física.

Agradecimentos

Os autores agradecem ao CNPq, que parcialmente financiou este projeto.

9. Referências

[1] AGEIA PhysX. Disponível em: AGEIA site. URL: http://www.ageia.com/physx/index.html, visitado em outubro de 2007.

[2] El-Rewini, H., e M. Abd-El-Barr, *Advanced Computer Architecture and Parallel Processing*, Wiley-Interscience, Hoboken EUA, 2005.

[3] Eberly, D.H., *Game Physics*, Elsevier Science, New York EUA, 2003.

[4] Brandt, D., J.R. Hiller, e M.J. Moloney, *Modern Physics Simulations*, Wiley-Interscience, Hoboken EUA, 1995.

[5] Crysis. Disponível em: EA Crysis site. URL: http://www.ea.com/crysis, visitado em novembro de 2007.

[6] R. Hilmer, D. Wu, "Design Patterns for Interactive Physics", *Painel da Game Developers Conference*, 2001. Disponível em: Pseudo Interactive site. URL: http://www.pseudointeractive.com/files/DesignPatternsForInteractivePhysicsGDC2001.ppt, visitado em novembro de 2007.

[7] Bullet Physics Library. Disponível em: Bullet site. URL: http://www.bulletphysics.com, visitado em outubro de 2007.

[8] Open Dynamics Engine. Disponível em: ODE site. URL: http://www.ode.org, visitado em outubro de 2007.

[9] Newton Game Dynamics. Disponível em: Newton site. URL: http://www.newtondynamics.com, visitado em outubro de 2007.

[10] OGRE 3D. Disponível em: OGRE site. URL: http://www.ogre3d.org, visitado em outubro de 2007.

[11] Horde 3D. Disponível em: Horde site. URL: http://www.nextgen-engine.net, visitado em outubro de 2007.

[12] Crystal Space. Disponível em: Crystal Space site. URL: http://www.crystalspace3d.org, visitado em outubro de 2007.

[13] NxOgre. Disponível em: NxOgre site. URL: http://www.nxogre.org, visitado em outubro de 2007.

[14] PAL – Physics Abstraction Layer. Disponível em: Adrian Boeing site. URL: http://www.adrianboeing.com/pal/index.html, visitado em outubro de 2007

[15] Freeman, E., E. Freeman, B. Bates, e K. Sierra, *Head First Design Patterns*, O'Reilly Media, Sebastopol EUA, 2004.

[16] S. Jamin, "Class Notes – Networking Multiplayer Games", *University of Michigan*, 2007. Disponível em: Computer Game Design and Implementation Course site.

Session 7: Development and Applications

Aplicação da Tecnologia Sun SPOT para Realidade Virtual

Aristóteles Fernandes Bandeira de Oliveira, Tiago Cruz de França, Alisson V. Brito
Centro Federal de Educação Tecnológica da Paraíba – CEFET-PB
{aristotelesfboliveira}, {jesus.senhor}@gmail.com, alisson@cetefpb.edu.br

Abstract

This paper presents the applicability of an experimental technology called Sun SPOTs (Small Programmable Objects Technology) to Virtual Reality. Such technology is compound of an ARM processor, sensors and I/O ports running Java software according the Java 2 Micro Edition (J2ME). This work investigates the Sun SPOT potential to work as a powerful interaction human-machine device for Virtual Reality. Actually we are applying them to model the movement of real objects and create their respective virtual representation, considering mainly their dynamic.

Resumo

*Os dispositivos de hardware são de gande importanca à Ralidade Virtual. Neste, apresentamos o uso da tecnologia Sun SPOT (*Small Programmable Objects Technolgy*) para interação do Mundo Real como Virtual. Os SPOTs são equipamentos com um processador embutido, executando sistemas embarcados, desenvolvidos com tecnologia Java 2 Micro Edition (J2ME), sobre um tipo de máquina virtual denominada* Squawk Virtual Machine. *Os SPOTs também possuem diversos sensores como o acelerômetro, e se comunicam usando redes ZIG BEE. Através de uma aplicação de movimentação do mouse com uso do acelerômetro do SPOT em dois eixos, apresentamos o uso deste equipamento para interação Real-Virtual.*

1. Introdução

A Realidade Virtual não é uma tecnologia recém descoberta. Apesar de existir há mais de duas décadas, manteve-se inacessível para muitos, devido ao alto custo de implantação e manutenção. Contudo, os recentes avanços tecnológicos e a alta acessibilidade à informática vêm transformando a Realidade Virtual de uma mera possibilidade em uma verdadeira "Realidade". Atualmente, é possível encontrar softwares e hardwares com custo acessível à maioria das empresas, utilizando a tecnologia "RV" para o desenvolvimento de sistemas que permitem simular situações reais em um computador, possibilitando ao usuário a sensação de ser outra pessoa ou estar em outro lugar [1].

O acelerado desenvolvimento tecnológico decorrido nas últimas duas décadas provocou uma série de grandes mudanças em todo o mundo, tanto no ponto de vista empresarial e econômico, como também político e social. Uma das principais mudanças deu-se na internacionalização do mercado, com o rompimento das barreiras comerciais físicas entre as nações ao redor do planeta, possibilitando o acesso de empresas de qualquer parte do mundo a mercados dos mais diversos lugares. Essa forma de comercialização eletrônica trouxe tanto pontos positivos, como o aumento da distribuição de produtos comerciais com uma redução de tempo e gastos na produção, quanto pontos negativos, pois se faz necessária mão-de-obra especializada e melhor qualificada.

Buscando novas formas de obter melhorias organizacionais e modernizar o sistema produtivo, algumas empresas estão optando pela utilização de sistemas desenvolvidos utilizando um software baseado em Realidade Virtual das mais diversas maneiras, possibilitando, dentre outras coisas, a simulação de equipamentos, treinamento de funcionários, validações de protótipos e planejamento de produção [7].

O avanço tecnológico na área da Realidade Virtual vem elevando a qualidade de dispositivos de hardware, como capacete e óculos de visualização e luvas mais leves e com mais recursos, chamando a atenção de várias áreas empresariais, aumentando a leva de usuários e de sistemas no mundo todo. Existe também uma grande quantidade de softwares de desenvolvimento de sistemas em Realidade Virtual disponíveis, com variadas ferramentas de programação e para várias plataformas. Hoje, graças à alta capacidade de hardware de processamento de dados e vídeo, é possível construir e explorar ambientes de Realidade Virtual apenas com:

um computador pessoal, ferramentas adequadas, experiência e criatividade.

2. A Tecnologia Sun SPOT

Sun SPOT (*Sun Small Programmable Object Technology*) é uma plataforma do tamanho da palma da mão (como pode ser visto na Figura 1) e acionada por software Java capaz de feitos incríveis, como equipar dirigíveis e mísseis, controlar mãos e carros robóticos e monitorar um sistema de testes industriais [2].

Baseado em uma CPU ARM de 32 bits e em um rádio de 11 canais de 2,4 GHz, um dispositivo Sun SPOT permite que os desenvolvedores compilem aplicativos transdutores sem fio com software Java e usem IDEs familiares, como a IDE *NetBeans*, para escrever os códigos. Os desenvolvedores podem escrever seus códigos em Java, carregá-lo no dispositivo do Sun SPOT e executá-lo, podendo desconectá-lo do computador. O dispositivo acionado à bateria inclui um acelerômetro de três eixos, sensores de temperatura e de luminosidade, cinco pinos de entrada e saída de uso geral, oito LEDs de três cores, quatro pinos de saída de alta tensão e uma interface USB. O seu sistema funciona independente de um computador, funcionando sobre o *Squawk Virtual Machine* (VM), uma pequena máquina virtual na Java 2 Platform Micro Edition (plataforma J2ME), que permite que aplicativos sem fio sejam executados diretamente na CPU sem depender de nenhum sistema operacional [3].

Figura 1: imagem de um Sun SPOT.

3. Sun SPOT e a Realidade Virtual

O termo Realidade Virtual é bastante abrangente, e mais definições similares vêm sendo dadas por diversos autores. Com uso da Realidade Virtual é possível manter um dos maiores níveis de interação do mundo real com o virtual, criando uma ampla comunicação entre o homem e o computador (Interação Homem-Máquina). Com o uso de tecnologias apropriadas é possível aperfeiçoar tais interações [7].

Um grande desafio da Realidade Virtual se encontra nos dispositivos de hardware, que são utilizados para gerar o Mundo Virtual e permitem a interação com o mesmo. O uso do Sun SPOT é uma forma de obtenção de interações de entrada/saída do Mundo Virtual com o Real, através de mecanismos e adereços pertencentes e/ou conectados e interligados ao Sun SPOT, como acelerômetros, câmera de vídeos, e dispositivos de entrada e saída de som.

O acelerômetro é um dispositivo conectado ao Sun SPOT capaz de detectar e medir os movimentos do mesmo em três diferentes eixos (X, Y, Z). Ele também é capaz de medir a orientação do aparelho com relação à gravidade. O acelerômetro consiste em um sensor mecânico microeletrônico, que é alterado de sua posição quando uma aceleração linear é aplicada, causando um balanço elétrico que é lido pelo conversor analógico-digital da placa, e fazendo assim com que o sensor detecte que houve uma alteração no movimento do Sun SPOT.

Os Sun SPOTs se comunicam entre si usando uma rede *ZigBee* [4], que é uma tecnologia de redes sem fio que permite uma comunicação confiável, com pouco consumo de energia e baixas taxas de transmissão. Esta tecnologia utiliza mecanismos de conexão e desconexão de um dispositivo em uma rede, identifica e armazena em uma tabela os dispositivos vizinhos, fornecendo identificação e manutenção do encaminhamento.

Desta forma é possível estabelecer redes em malha, conhecidas como redes Mesh [8], capaz de encaminhar, através de múltiplos saltos, mensagens ao seu destino. Redes Mesh utilizam protocolo de roteamento que fazem várias varreduras das possíveis rotas, considerando a mais rápida e com menos perda de pacotes [5].

Várias funcionalidades se tornam possíveis com o uso de redes e acelerômetros, como simular movimentos do Mundo Real em uma representação Virtual da mesma, onde um corpo que é acelerado com a mesma intensidade e direção em que o Sun SPOT é movimentado – essa comunicação Real-Virtual da variação aceleração-direção pode dar-se por meio da rede ou de transmissão USB.

Este pequeno exemplo citado acima demonstra apenas uma das infinitas possibilidades do uso das capacidades do Sun SPOT na Realidade Virtual, tornando-o uma ótima ferramenta para a utilização no campo Virtual.

4. Uso do Sun SPOT em aplicações de Realidade Virtual

A utilização do acelerômetro, em conjunto com outros diversos dispositivos presentes no Sun SPOT permitem uma infinidade de possibilidades relativas à Realidade

Virtual. Duas delas (e que já estão sendo implementadas por nós) são a modelagem de movimentos de objetos reais para o Mundo Virtual, e o uso do Sun SPOT como um periférico de interação homem-máquina tridimensional [6] (um mouse 3D).

O principal desejo da modelagem de objetos tridimensionais é a manipulação dos mesmos através do usuário, utilizando-se de equipamentos e periféricos como controles. Através do acelerômetro do Sun SPOT, o usuário é capaz de movimentar o Objeto Virtual de acordo com a sua vontade, seja utilizando o Sun SPOT como controle, seja utilizando o Sun SPOT conectado a um Objeto Real, onde a sua movimentação ativa a movimentação do Objeto Virtual.

O mouse é considerado um dos principais periféricos de interação Homem-Máquina, e sua utilização é de crucial importância para o uso da maioria dos programas. Há dois tipos de mouses – os que se movem através de mecanismos acionados por uma esfera, que é rolada através do movimento do mouse, e o mouse óptico, que é ativado através de sensores ópticos, não tendo peças móveis. Ambos os modelos trabalham com o movimento em cima de uma superfície. Utilizando-se do Sun SPOT, o usuário pode mover o cursor na tela, como se estivesse usando um mouse. A diferença fica por conta da superfície, que se torna completamente opcional. O movimento do cursor baseia-se na movimentação do Sun SPOT, através do acelerômetro, o que permite ao usuário controlar o cursor no ar.

5. Desenvolvimento de um Mouse-3D utilizando a tecnologia Sun SPOT

A viabilização de um Mouse-3D utilizando Sun SPOTs foi feita através do desenvolvimento de duas aplicações separadas. Uma das aplicações é executada num computador pessoal convencional, com um dos SPOTs conectados à ele, trabalhando como *gateway*, chamada de Aplicação Desktop. A outra aplicação pode ser executada em qualquer Sun SPOT e é responsável por captar qualquer movimentação no dispositivo através dos sensores de movimento.

Numa freqüência de 1KHz, os SPOTs checam cada acelerômetro por alguma variação significativa no movimento. Se ocorrer, o mesmo envia um pacote para a Aplicação Desktop contendo o eixo onde ocorreu o movimento (X ou Y), juntamente com a intensidade da aceleração detectada.

A Aplicação Desktop trabalha recebendo todos pacotes do SPOT e efetuando as devidas movimentações do ponteiro do mouse nas direções e intensidades detectadas e enviadas pelo SPOT através do pacote ZigBee.

A classe Java utilizada para movimentação do Mouse foi a java.awt.Robot. Esta pode ser utilizada para efetuar qualquer operação do mouse e teclado de forma automatizada. Muito utilizada para automatização de tarefas de entrada e saída.

Através de parâmetros em ambas as aplicações, atributos das aplicações podem ser modificados e adequados à cada aplicação, tais como, a sensibilidade do Mouse 3D, a velocidade do ponteiro do mouse na Aplicação Desktop e a freqüência de captação e envio de pacotes pelos SPOTs.

6. Referências

[1] Netto, A.V.; Machado, L.S.; Oliveira, M.C.F. "Realidade Virtual: Fundamentos e Aplicações". Ed. Visual Books, Florianópolis/SC, 2002.

[2] Project Sun SPOT. http://www.sunspotworld.com/.

[3] The Squawk Virtual Machine. http://research.sun.com/projects/squawk/.

[4] ZigBee Alliance – Wireless Control that Simply Works. http://www.zigbee.org.

[5] Rua, D.; Martins, N.; Reis, P.; Sousa, J. P. Interface USB para recolha de dados de sensores remotos utilizando *ZigBee e IEEE 802*.15.4. http://www.deetc.isel.ipl.pt/jetc05/JETC05.

[6] J. Rehg, T. Kanade. "Visual tracking of high dof articulated structures: an application to human hand tracking." In *Third European Conference on Computer Vision (ECCV'94)*, pag. 35-46, Springer-Verlag, Stockholm, Suécia, maio de 1994.

[7] Valerio Netto, A., Machado, L. dos S., Oliveira, M. C. F. *"Realidade Virtual: Fundamentos e Aplicações"*. Editora Visual Books, 2002.

[8] Anggelou, G. 2008. Wireless Mesh Networking: With 802.16, 802.11, and ZigBEE. *McGraw-Hill Professional*, 1.ed. ISBN: 0071482563.

Integração de Linguagens de Programação para Uso de Dispositivos Não-Convencionais: Possível Solução para Construir Aplicações com Baixo Custo

Cléber Gimenez Corrêa, Fátima de L. S. Nunes, Adriano Bezerra
Centro Universitário Eurípides de Marília – UNIVEM
Laboratório de Aplicações de Informática em Saúde - LApIS
Marília, SP, Brasil
correacleber@yahoo.com.br, fatima@univem.edu.br, adrianobezerra1@yahoo.com.br

Abstract

This paper presents a detailed technical description about the inclusion of non-conventional devices in Virtual Reality (VR) applications, integrating programming languages, as Java and C, aiming at obtaining realism during the interaction in order to ensure the quality of the training. Thus, it is possible to integrate equipments, whose manufacturers offer drivers in C language, to applications built in Java language, contributing to the implementation of projects using free technologies.

Keywords: *Integration of languages, Medical training, Nonconventional devices, Virtual Reality.*

Resumo

Este artigo apresenta uma descrição técnica detalhada sobre a inclusão de dispositivos nãoconvencionais em aplicações de Realidade Virtual (RV), integrando linguagens de programação, como Java e C, visando à obtenção de realismo durante a interação, para garantir a qualidade do treinamento. Desta forma, é possível integrar equipamentos, cujos fabricantes disponibilizam drivers na linguagem C, a aplicações construídas em linguagem Java, contribuindo para a implementação de projetos utilizando tecnologias gratuitas.

Palavras-chave: *Integração de linguagens, Treinamento médico, Dispositivos não-convencionais, Realidade Virtual.*

1. Introdução

Em aplicações de RV, principalmente aquelas voltadas ao treinamento médico, é desejável, entre outras características, precisão e respostas em tempo real, para que a interação torne-se mais próxima da ação realizada em situações reais [1].

Neste contexto, as exigências para imprimir realidade aos treinamentos tornam indispensáveis a inclusão de dispositivos de entrada e saída, responsáveis pela comunicação entre o usuário e o sistema computacional, classificados geralmente em convencionais (teclado, mouse, entre outros), e não-convencionais (luva de dados, equipamento háptico, entre outros). Estes últimos podem permitir um maior grau de realismo durante a interação. Como exemplo desse realismo, pode ser citada a luva de dados, pois segundo [2], o reconhecimento de gesto é uma forma de interação eficiente e altamente intuitiva para Ambientes Virtuais (AVs).

Estes dispositivos geralmente possuem drivers e bibliotecas com funções prontas, implementadas pelos fabricantes, que, em geral utilizam linguagens de programação com funções em baixo nível. Entretanto, percebe-se que diversos projetos de RV estão sendo construídos em linguagens de mais alto nível, como a linguagem Java, com a finalidade de aproveitar características intrínsecas dessas ferramentas, como gratuidade, disponibilização de classes diversas para a criação e manipulação de AVs, que contribuem para o baixo custo e o aumento de produtividade, respectivamente, durante o projeto, implementação e documentação do software.

O ViMeT (*Virtual Medical Training*) é um exemplo de aplicação que utiliza tecnologia Java, consistindo em um *framework* para geração de aplicações voltadas ao treinamento médico, fornecendo funcionalidades como: deformação, criação dinâmica de AVs, detecção de colisão, estereoscopia e interação por meio de diversos tipos de dispositivos [3].

Para viabilizar a integração de dispositivos não-convencionais em aplicações implementadas em linguagem Java, quando o fabricante não disponibiliza *drivers* nesta linguagem, duas estratégias são possíveis: construção de *drivers* próprios, ou integração de linguagens de programação, possibilitando o uso do driver fornecido pelo fabricante. A primeira abordagem envolve alguns problemas, como a necessidade de abranger diversos tipos de portas de entrada para conexão dos equipamentos, programação em baixo nível, tempo de resposta e grande quantidade de tempo

dedicada ao desenvolvimento e aos testes, que poderia ser empregada no aperfeiçoamento da aplicação. A segunda abordagem elimina alguns desses problemas e, adicionalmente, permite a utilização de um *driver* que já foi especificamente desenvolvido pelo fabricante, sendo devidamente testado e aprovado para uso.

No entanto, a adoção da segunda abordagem exige também o domínio da programação das linguagens envolvidas e uma pesquisa intensa sobre funcionalidades de ambas, a fim de que a integração omita detalhes técnicos do usuário final e, ao mesmo tempo, proporcione uma interação adequada. Entretanto, na literatura da área, este assunto é escasso, motivo pelo qual este artigo é apresentado tendo o objetivo de auxiliar desenvolvedores a realizar tal tarefa, oferecendo detalhes que podem ser úteis na solução de problemas encontrados durante a implementação.

2. Metodologia

No presente trabalho, dois dispositivos não-convencionais estão sendo utilizados. O primeiro é uma luva de dados *5DT DataGlove 5 Ultra*, fabricada pela 5DT (*Fifth Dimension Technologies*), e que possui cinco sensores para captar a flexão dos dedos, conforme a Figura 1a [4].

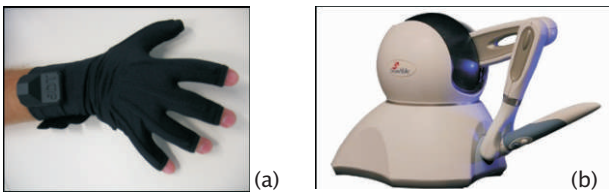


Figura 1. (a) *DataGlove 5 Ultra* e (b) *Phantom Omni* [5]

O Ambiente Virtual (AV) gerado com o auxílio do *framework* já existente, (Figura 2), é constituído por três objetos: um instrumento médico, um órgão humano e uma mão virtual. Definiu-se que a luva representa a ação de segurar e soltar o objeto que simula o órgão durante o exame de punção.

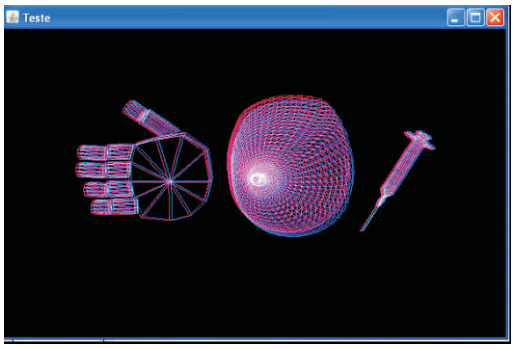


Figura 2. Aplicação gerada.

O segundo dispositivo é um equipamento háptico *Phantom Omni*, desenvolvido pela *SensAble Technologies* e apresentado na Figura 1b, o qual permite um retorno de força, e detecta informações de rotação e translação nos três eixos [5]. No AV, o dispositivo háptico está relacionado com o instrumento médico utilizado para simular a coleta de material em um exame de biópsia.

As informações captadas por estes dispositivos devem ser transferidas para a aplicação e determinar o comportamento de objetos no AV. No caso da luva de dados, os movimentos resultam em alterações na mão virtual, e a posição dos dedos é modificada. Já no caso do dispositivo háptico, os movimentos são reproduzidos pelo instrumento médico virtual. A força aplicada também é calculada pela aplicação e, quando é detectada uma colisão entre os objetos que representam o instrumento médico e o órgão humano, uma deformação deste último é gerada.

Para integrar os recursos provenientes das linguagens de programação C e Java, foi utilizada uma interface de programação nativa, denominada de JNI (*Java Native Interface*), que permite a interoperação entre o código Java e bibliotecas e programas desenvolvidos em outras linguagens, como C, C++, *Assembly* [6]. Para isso, a linguagem Java utiliza os arquivos jni.h e jni_md.h, sendo que o primeiro deve ser incluído no cabeçalho do programa em C e é também utilizado no arquivo de cabeçalho a ser gerado.

Para se trabalhar com métodos nativos, deve-se colocar a palavra-chave *native* na definição dos métodos da classe em Java, indicando à máquina virtual que estes métodos serão executados em uma outra linguagem (código nativo). Além disso, deve estar presente o método *System.loadLibrary*, que recebe como parâmetro a biblioteca de ligação a ser gerada, no caso de sistemas operacionais Windows, as denominadas DLLs (*Dynamic-Link Libraries*) [6].

A Figura 3 apresenta parte do código implementado, com a palavra-chave *native* indicando que o método "NomeMetodo" será executado em outra linguagem, a palavra "ArquivoDLL" representa a biblioteca de ligação a ser carregada e a nomenclatura *short[]* indica que um vetor do tipo *short* será retornado.

```
public class ClasseNativa {
    public native short[] NomeMetodo();
    static
    {
        System.loadLibrary("ArquivoDLL");
    }
…
```

Figura 3. Código exemplo de uma classe.

Em seguida, é necessário gerar um arquivo de cabeçalho com a extensão *.h*, o que pode ser realizado com o uso da ferramenta *javah*, que acompanha o JDK (*Java Development Kit*) [6], a partir do arquivo *.class*, criado na compilação da classe [5]. Pode-se notar a assinatura do método em código nativo, presente no arquivo de cabeçalho (Figura 4), e no programa em C (Figura 5), que contém o prefixo Java, o nome do pacote, o nome da classe, e o nome do método, separados por *underline*.

```
/*DO NOT EDIT THIS FILE - it is machine
generated*/
#include <jni.h>
/*Header for class Pacote_ClasseNativa*/

#ifndef_Included_Pacote_ClasseNativa
#define_Included_Pacote_ClasseNativa
#ifdef__cplusplus
extern "C" {
#endif
/*
    *Class:       Pacote_ClasseNativa
    *Method:      NomeMetodo
    *Signature:   ()[S
*/
JNIEXPORT jshortArray JNICALL
Java_Pacote_ClasseNativa_NomeMetodo
(JNIEnv*, jobject);
#ifdef__cplusplus
}
#endif
#endif
```

Figura 4. Exemplo de arquivo de cabeçalho.

As *strings JNIEXPORT* e *JNICALL*, definidas no arquivo *jni.h*, são usadas para especificar as chamadas e as ligações entre os métodos JNI e as funções nativas. JNIEnv é o ponteiro da interface JNI, que aponta para uma matriz de ponteiros, que por sua vez apontam para as funções JNI. O tipo *jobject* indica que um método não-estático está sendo usado, e este obtém uma referência para o objeto, como um argumento *this* implícito.

Os tipos de dados e *arrays* da linguagem Java possuem tipos correspondentes na linguagem C. No exemplo, o tipo Java *short[]* corresponde ao tipo C *jshortArray*, indicando um vetor do tipo *short* [7]. Em seguida, o programa em linguagem C é elaborado contendo as funções que serão chamadas pelos métodos escritos em Java. No cabeçalho deste programa devem ser invocados os arquivos de cabeçalho e *jni.h*, que serão utilizados pelo compilador para gerar a biblioteca.

E assim, por último é preciso gerar uma biblioteca de ligação, procedimento realizado com o auxílio de compiladores [6], tais como: *Microsoft Visual C++*, *Borland C++ Compiler*, *Bloodshed Dev-C++*, sendo que os dois últimos são disponibilizados gratuitamente.

```
#include <jni.h>
#include "NativeGlove.h"
…
JNIEXPORT jshortArray JNICALL
Java_Pacote_ClasseNativa_NomeMetodo
(JNIEnv *env, jobject obj)
{
   //Função para obter valores dos
sensores
}
```

Figura 5. Exemplo de código nativo.

3. Resultados

Os procedimentos da seção anterior foram seguidos para realizar a integração entre a aplicação codificada em Java e um programa em C com funções fornecidas pelo fabricante da luva de dados, as quais permitem iniciar e terminar a conexão com a luva, além de capturar valores dos sensores.

Desta forma, construiu-se a classe denominada *NativeGlove*, composta de métodos para acessar as funções do código nativo escrito em C, e o método especial *System.loadLibrary* para carregar a biblioteca de ligação, no caso, o arquivo *5DTGlove.dll*. Em seguida, gerou-se um arquivo de cabeçalho (*NativeGlove.h*), e um programa com funções em C foi desenvolvido para acessar a luva de dados. Um compilador é necessário com o objetivo de gerar a DLL no ambiente Windows XP. Optou-se pelo aplicativo *Microsoft Visual C++ 6.0* para testes iniciais por uma questão de familiaridade com o aplicativo; entretanto, o intuito é utilizar compiladores gratuitos. Os valores dos sensores são enviados para a aplicação, e os dedos da mão virtual se movem de acordo com os movimentos do usuário e são apresentados no monitor e vídeo. O diagrama da Figura 6 define a interação do usuário com o sistema, a qual ocorre por meio da luva de dados e de um monitor comum, e a integração entre as linguagens C (Funções da luva de dados), e Java (Geração da aplicação a partir do *framework*), bem como o arquivo *5DTGlove.dll*, que é a biblioteca de ligação.

Os procedimentos para a integração da luva de dados foram realizados também com o dispositivo háptico. No entanto, funções de retorno de força e de obtenção informações de translação e rotação deverão ainda ser implementadas a fim de obter-se uma avaliação mais adequada. A Figura 7 apresenta os movimentos do objeto mão no AV em momentos distintos, conforme a flexão dos dedos do usuário.

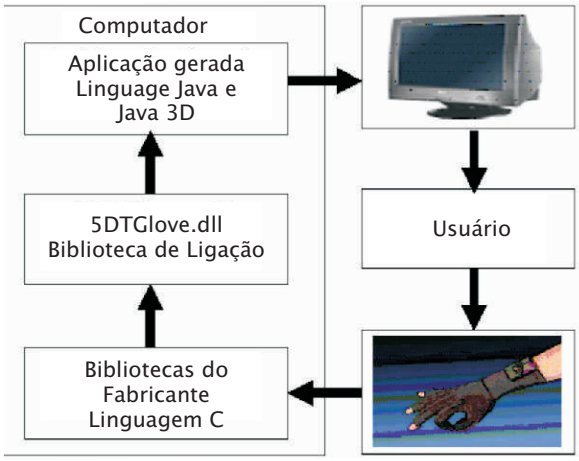


Figura 6. Diagrama de interação e integração.

Figura 7. Movimentos da mão virtual.

4. Conclusão e discussões

Este artigo apresentou uma forma de integrar a aplicações escritas em Java, dispositivos cujos *drivers* são disponibilizados em linguagem C. Seguindo os passos aqui descritos, é possível a adição de tais equipamentos de forma prática e sem permuta de tecnologias. Verifica-se que esta forma de integração pode ser generalizada, considerando a portabilidade da linguagem Java, e pode contemplar, inclusive, *drivers* fornecidos para outros Sistemas Operacionais.

Além do realismo proporcionado pelo uso desses dispositivos em aplicações de treinamento médico, salienta-se que é possível atender as necessidades do usuário sem demandar tempo do projeto para a implementação de novos *drivers*, e ainda manter o custo reduzido, visto que a aquisição de equipamentos não-convencionais, por si só já representa uma elevação no custo de aplicações.

Até o presente momento, o tempo de resposta da aplicação está em um nível satisfatório, mensurado empiricamente, isto é, o usuário não percebe atrasos nas respostas às interações, mesmo com a integração entre as linguagens. Entretanto, uma avaliação com protocolo mais adequado será realizada para mensurar este tempo de resposta e a quantidade de quadros por segundo, e conseqüentemente, avaliar o desempenho da aplicação, pois a interação não pode ser prejudicada e comprometer o treinamento.

A opinião de médicos e estudantes da área também será coletada, levando em consideração a experiência com equipamentos de RV e procedimentos de biópsia, facilidade de uso e conforto com os dispositivos, e uso da intuição durante a interação.

5. Referências

[1] F. L. S. Nunes et al., "Aplicações Médicas usando Realidade Virtual e Realidade Aumentada", In *Livro do 9º SVR*, Petrópolis, RJ, Brasil, 2007, pp. 234-235.

[2] J. Eisenstein et al., "Device Independence and Extensibility in Gesture Recognition", In *Proceedings of the IEEE Virtual Reality*, 2003, pp.207-214.

[3] A. C. M. T. G. Oliveira et al. "Virtual Reality Framework for Medical Training: Implementation of a deformation class using Java". In *Proceedings of ACM Siggraph International Conference on Virtual Reality and its applications in industry, Hong Kong, Nova York: ACM Press*, 2006, pp. 347-351.

[4] 5DT - Fifth Dimension Technologies, Disponível em: http://www.5dt.com/hardware.html, Acesso em: Março 2007.

[5] SensAble Technologies. Disponível em: http://www.sensable.com, Acesso em: Março 2007.

[6] G. Cornell, C. S. Horstmann. "Métodos Nativos". *Core Java 2, Volume II, Recursos Avançados*. Pearson Education do Brasil, São Paulo, 2003, pp. 755-785.

[7] JNI, "Java Native Interface Specification", Disponível em: http://java.sun.com/j2se/1.4.2/docs/guide/jni/spec/jniTOC.html, Acesso em: Março 2007.

Infra-estrutura de Baixo Custo para Visualização 3D Estereoscópica Destinada a Aplicações Biológicas

Paulo Roberto Trenhago[1], Selan Rodrigues dos Santos[2], Jauvane Cavalvante de Oliveira[1]

[1]Laboratório Nacional de Computação Científica
Laboratório ACiMA, Petrópolis, RJ, Brazil
{trenhago}, {jauvane}@lncc.br

[2]Universidade Federal do Rio Grande do Norte
Departamento de Informática e Matemática Aplicada, Natal, RN, Brazil
selan@dimap.ufrn.br

Resumo

Este trabalho descreve a montagem de um sistema de visualização 3D com suporte a estereoscopia em cores de baixo custo em tela de grande dimensão, destinado a aplicações em Biologia. No experimento foram empregados dois filtros de polarização da luz, dois projetores de imagens, um espelho de metal polido e uma tela de projeção de 1,85m x 2,45m.

Palavras-chave: *Visualização Científica, estereoscopia em cores, sistemas virtuais imersivos.*

1. Introdução

Um rápido desenvolvimento das Ciências Biológicas ocorre devidos aos constantes avanços em áreas interrelacionadas como Física, Matemática, Química, Eletrˆonica, Mecânica e Computação. Neste cenário, a Ciência da Computação desempenha um papel fundamental para integrar, organizar, analisar e disponibilizar grandes volumes de dados gerados pelas redes de pesquisas biológicas mundiais. Por exemplo, a construção, análise e visualização de modelos de redes biológicas [2, 4, 6], seqüenciamento e análise de DNA e mecânica do sistema cardiovascular [10], são tarefas impraticáveis sem o uso de sistemas computacionais de grande porte e pesquisadores com formação multidisciplinar.

Recentemente, novos campos de pesquisa em Biologia estão surgindo e requerem o uso de sistemas de visualização 3D e 4D (três dimensões espaciais + 1 dimensão temporal) como dinâmica molecular, *docking* [1] e dobramento de proteínas; desenvolvimento embrionário [11]; zoologia comparada; reconstrução virtual de fósseis e paleoambientes (ambientes antigos); evolução de caracteres morfométricos e/ou alterações fenotípicas provocados por agentes ecotoxicológicos.

Ambientes de Realidade Virtual (RV) imersíveis de grande escala como CAVEs (*Cave Automatic Virtual Environment* [3]) e RAVEs (*Reconfigurable Virtual Environment*), possuem um custo elevado se comparado a soluções baseadas em RV de mesa, ou mesmo às que utilizam dispositivos não convencionais de imersão como *face/head mounted displays*. Além disso, algumas soluções propostas que utilizam estereoscopia baseada em anaglifos prejudicam ou inviabilizam muitas das aplicações em Ciências da Vida. Estes fatores nos motivaram a projetar e construir um sistema de visualização estereoscópico passivo com suporte total a cores em telas de grande dimensão utilizando dois projetores e uma placa de vídeo com duas saídas.

O sistema de visualização em desenvolvimento apresenta estereoscopia em cores a um custo inferior ao de uma projeção equivalente realizada com estereoscopia ativa, a uma resolução de aproximadamente 10 pontos por polegada de imagem projetada.

2. Materiais e métodos

2.1 Projeto físico

O espaço destinado a instalação do sistema de visualização foi estruturado de maneira que não houvesse entrada de luz externa e possui as seguintes dimensões: 5m × 6m (base) e 3m (altura). Um diagrama de instalação simplificado, representado pela Figura 1, ilustra a disposição dos principais componentes do sistema. Para a renderização das imagens, foi utilizado um computador com processador Intel Pentium D, 2Gbytes de memória RAM, disco rigido de 160 Gbytes, placa de rede Gigabit Ethernet, sistema operacional Window XP Professional e dois monitores LCD Samsung de 17 polegadas, com suporte a uma resolução de 1024 × 768 pixels a 85Hz. A placa de vídeo empregada possui o chipset NVIDIA 8800 GTX com 768 Mbytes de memória e barramento PCIExpress.

Para a projeção das imagens foram utilizados dois projetores NEC LT245, com resolução 1024×768 pixels a 85Hz, montados em um suporte a uma altura de 1, 2m do piso. As projeções são realizadas em tela Stewart Disney Black de 2, 46m × 1, 85m (altura por largura), para projeção por trás, suportada por uma

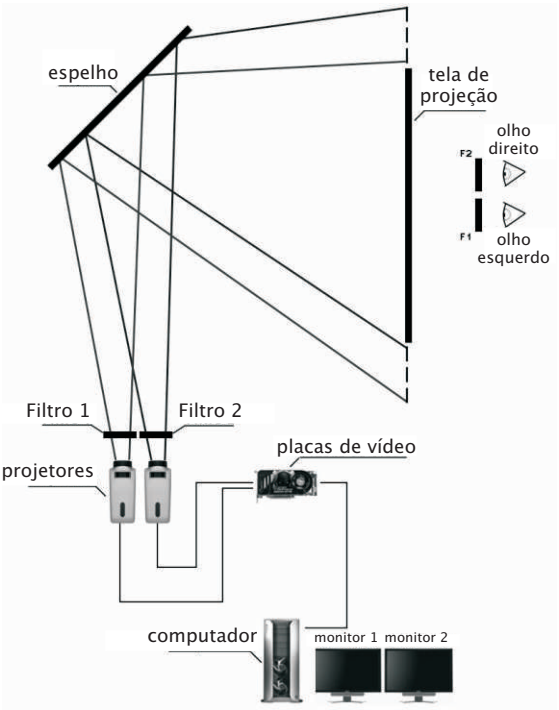


Figura 1: Diagrama de instalação do sistema de visualização.

armação de tubos PVC. Devido à restrição de espaço, utilizamos um espelho de metal polido First Surface Mirror para ampliação da distância para uma correta projeção das imagens em toda a área da tela de projeção.

2.2 Sistema de estereoscopia

Para gerar a estereoscopia é preciso sintetizar uma imagem para o olho esquerdo e outra para o olho direito. A imagem correspondente ao olho direito deve sensibilizar o olho direito mas não pode ser percebida pelo olho esquerdo e vice-versa. O efeito final obtido através desta técnica é a ilusão de que os objetos estão presentes no espaço 3D e não estão restritos ao plano de projeção.

Existem diversos mecanismos para enviar seletivamente imagens aos olhos correspondentes (e.g. filtros de cores, filtros de luz polarizada, oclusão controlada). Entretanto, para obtermos estereoscopia passiva com suporte total a cores e que permite girar a cabeça sem prejudicar a visualização, foram empregados filtros de polarização de luz circular. Nas imagens destinadas ao olho direito utiliza-se filtro de luz com polarização circular no sentido horário, enquanto que para o olho esquerdo utiliza-se um filtro de polarização no sentido oposto. Um par de filtros foi posicionado na frente dos projetores e outro par correspondente foi utilizado na confecção de óculos, conforme ilustrado na Figura 1.

2.3 Alinhamento dos projetores

Os projetores foram dispostos na vertical, um ao lado do outro, em dois planos paralelos e suas projeções ajustadas para tamanhos iguais (veja Figura 2-a).

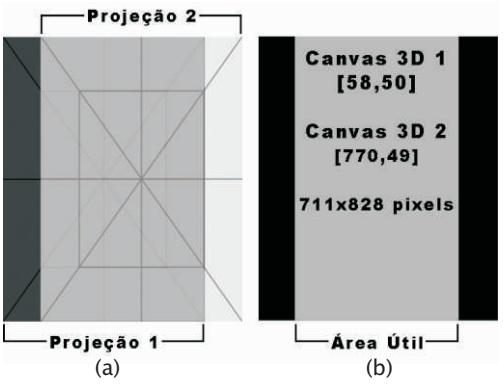


Figura 2: Processo de ajuste de projeção: (a) imagem correspondente ao alinhamento físico dos projetores; (b) área útil resultante do alinhamento e coordenadas de tela.

A disposição dos projetores de outro modo que não seja em planos paralelos provoca distorções nos objetos renderizados. Portanto, o correto alinhamento destes projetores é crítico na montagem do sistema. Ainda em relação ao alinhamento, cuidados devem ser tomados para que os eixos X de cada projeção (representado com uma linha horizontal na Figura 2-a) estejam alinhados. O fato dos projetores estarem dispostos na vertical resulta em uma projeção de imagem rotacionada 90 no sentido anti-horário. Por conseguinte, é necessário inverter as imagens diretamente na placa de vídeo. O resultado final é um área de exibição de 1536 × 1024 pixels (768 × 1024 pixels para cada projetor), com um novo sistemas de coordenadas de tela, conforme ilustrado na Figura 2-b.

2.4. Determinação da área útil e resolução

A área útil disponível para a renderização corresponde à área de intersecção das projeções 1 e 2 (Figura 2-b). Para a determinação da área útil e sua resolução, desenvolvemos um programa em Java [9] que, de forma interativa, permite o alinhamentos das imagens projetadas. O software gera duas imagens de referência, cada uma correspondendo a um dos olhos. Estas imagens — um retângulo em vermelho e outro em verde, ambos com linhas guias pretas — são projetadas sobre um fundo preto. O fundo preto corresponde a uma janela do javax.swing sem bordas, título e botões com controle do tamanho e posição das imagens de

referência.

O ajuste é conseguido quando uma imagem amarela pura (combinação aditiva das cores vermelha e verde na mesma proporção) e somente guias pretas ficaram visíveis. Através do software foi gerado um relatório com as cordenadas Canvas1 em [58, 50, 711, 828], Canvas2 em [770, 49, 711,828] que correspondem às coordenadas iniciais em pixel, largura e altura respectivas aos canvas (Figura 2-b). O endereçamento e cada imagem ao respectivo projetor é feito simplesmente pelas cordenadas de tela.

2.5. Aplicação de teste

Desenvolvemos uma aplicação para carregar e renderizar em 3D um modelo no formato OBJ do Maya de uma larva de *Leptobrachella mjobergi* [5], um animal que habita as profundezas dos oceanos. Optamos por utilizar a linguagem Java pela facilidade de manipulação de janelas, pela API 3D e pelo suporte a estereoscopia.

Na Listagem 1, as variáveis *sCanvas1* e *sCanvas2*, linhas 2, são as janelas de suporte aos canvas 3D c1 e c2, respectivamente. O construtor da classe *CaveTeste*, linha 8, configura o *JFrame* principal, *sCanvas1* e o *sCanvas2*, para exibirem janelas sem molduras e sem botões, define a cor de fundo para preto e ajusta as coordenadas das janelas. Linhas 23 e 26 definem os *canvas* 3D para exibirem imagens separadas para cada olho e as linhas 33 e 35 definem a posição dos olhos do observador em relação a cabeça, dados determinados experimentalmente.

Listagem 1 Fragmento de código da aplicação

```
1  public class TesteCave extends JFrame {
2     private JFrame sCanvas1, sCanvas2;
3     private Canvas3D c1 = new Canvas3D(
4          SimpleUniverse.getPreferredConfiguration());
5     private Canvas3D c2 = new Canvas3D(
6          SimpleUniverse.getPreferredConfiguration());
7
8     public TesteCave() {
9        setSize(1536, 1024);
10       setUndecorated(true);
11       setBackground(Color.black);
12       // ...
13       sCanvas1 = new JFrame();
14       sCanvas1.setUndecorated(true);
15       sCanvas1.setSize(711, 828);
16       sCanvas1.setLocation(58, 50);
17        // ...
18       sCanvas2.setSize(711, 828);
19       sCanvas2.setLocation(770, 49);
20    }
21    public void inicio() {
22       c1.setSize(711, 828);
23       c1.setMonoscopicViewPolicy(View.LEFT_EYE_VIEW);
24       sCanvas1.add(c1);
25        // ...
26       c2.setMonoscopicViewPolicy(View.RIGHT_EYE_VIEW);
27       scene = createSceneGraph(0);
28       u = new SimpleUniverse(c1);
29       View view0 = u.getViewer().getView();
30       View view = new View();
31       PhysicalBody myBod = view0.getPhysicalBody();
32       myBod.setLeftEyePosition(
33             new Point3d(-.0045, 0.0, 0.0));
34       myBod.setRightEyePosition(
35             new Point3d(+.0045, 0.0, 0.0));
36       view.setPhysicalBody(myBod);
37       view.setPhysicalEnvironment(
38             view0.getPhysicalEnvironment());
39       view.attachViewPlatform(
40             u.getViewingPlatform().getViewPlatform());
41       view.addCanvas3D(c2);
42        // ...
43    }
44    public static void main(String[] args) {
45       new TesteCave();
46    }
47 }
```

3. Resultado

Concluídas as tarefas de montagem da parte física e o alinhamento dos projetores, a determinação das coordenadas e resolução da tela de projeção foi realizada. O software para testes preliminares foi desenvolvido e o resultado da renderização estereoscopica foi bastante satisfatório. A Figura 3, representa a saída das imagens projetadas na tela pelos dois projetores.

A resolução alcançada foi de 10 pontos por polegadas permitindo gerar imagens de boa qualidade e bom desempenho da estereoscopia (confira a Figura 4). Outra vantagem desta infra-estrutura é a possibilidade de utilizar os recursos de estereoscopia nativos do Java 3D, OpenScene-Graph [7] e VTK (*Visualization Toolkit*) [8], permitindo um rápido desenvolvimento das aplicações de visualização.

4. Discussão

O sistema permitiu visualizar as estruturas da larva de *Leptobrachella mjobergi* em um tamanho que ressalta detalhes que facilmente poderiam passar desapercebidos em sistemas de renderização de menor porte. A infra-estrutura de visualização desenvolvida poderá, por exemplo, permitir o desenvolvimento de sistemas para a comparação geométrica de estruturas e/ou organismos de espécies relacionadas evolutivamente através de sobreposição 3D e uso adequado de transparências. Como resultado, um novo modelo geométrico representativo de uma provável direção evolutiva poderia ser gerado através de uma extrapolação morfométrica.

Para finalizar, acreditamos que a renderização estereosc ópica colorida em ambientes semi-imersíveis de modelos gerados a partir de animais e plantas, seja por cortes histológicos seriados ou equipamentos de aquisição de imagem, pode ser uma ferramenta valiosa aos biológos nos estudos de evolução, fisiologia, paleontologia, embriologia.

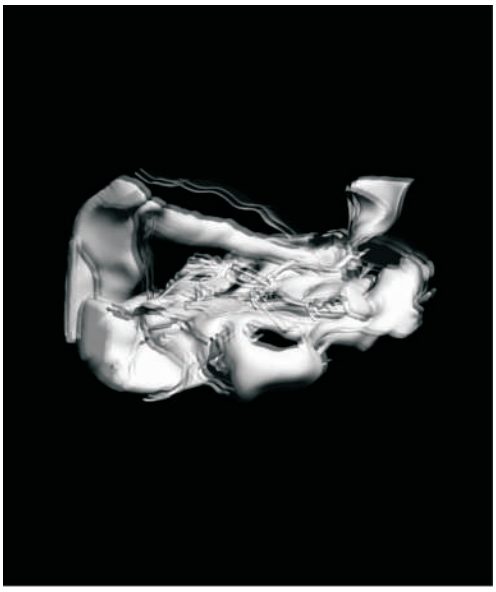


Figura 3: Resultado: representação de uma projeção estereoscópica das estruturas internas da larva de Leptobrachella mjobergi.

Figura 4: Infra-estrutura de visualização exibindo resultados parcias.

5. Conclusão

O trabalho descrito neste artigo trata de questões teóricas e práticas relacionadas com a utilização de interfaces multimodais em sistemas de realidade aumentada. Nesse sentido, foi proposto uma arquitetura genérica visando padronizar e facilitar a incorporação de variados dispositivos de interface no sistema ARToolkit, denominada ARToolKit Multimodal. A implementação de um protótipo dessa arquitetura, e os correspondentes testes, permitiu que se chegasse a algumas conclusões iniciais. A primeira confirma que o reconhecimento de voz, por ser uma forma de interação bastante natural e eficiente, deve ser melhor explorada em sistemas avançados de computação. Apesar de não ser perfeito, o nível de evolução dessa tecnologia já permite a sua utilização em sistemas que exigem um grau de robustez elevado. Por outro lado, as interfaces multimodais, nas suas mais variadas formas e configurações, não combinam necessariamente com todo tipo de aplicação. é importante que se leve em consideração qual é o objetivo principal da aplicação, e quais os meios mais eficientes e interessantes para torná-la operacionalmente eficiente. Outras observações a serem consideradas dizem respeito a aceitação de uma interface multimodal, a qual é dependente das preferências pessoais de quem a está utilizando, e também com a sua facilidade de adaptação. De qualquer forma, a exploração de novas possibilidades oferecidas pelo uso de interfaces multimodais em sistemas de realidade aumentada mostrou-se bastante interessante, e com alto potencial para alavancar o desenvolvimento de novas classes de aplicações.

6. Referências

[1] A. Anderson and Z. Weng. VRDD: Applying virtual reality visualization to protein docking and design. *JOURNAL OF MOLECULAR GRAPHICS MODELLING*, 17(3-4):180, 1999.

[2] J. Collado-Vides and R. Hofestadt. Gene regulation and metabolism: postgenomic computational approaches. *The MIT Press*, 2002.

[3] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The CAVE: audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):64–72, June 1992.

[4] A. Goldbeter. Cell to Cell Signalling: From Experiments To Theoretical Models. *Academic Press, Inc*, San Diego, CA 92101, 1989.

[5] A. Hass, S. Hertwig, and I. Das. Extreme tadpoles: The morphology of the fossorial megophryid larva, leptobrachella mjobergi. *ZOOLOGY*, 1(109):26–42, Setembro 2006.

[6] B. H. Junker and F. Schreiber. Analysis of Biological Networks. *Wiley-Interscience*, Germany, 2008.

[7] P. Martz. OpenSceneGraph Quick Start Guide. *Skew Matrix Software LLC*, 2007.

[8] W. Schroeder, K. Martin, and B. Lorensen. The Visualization Toolkit An Object-Oriented Approach To 3D Graphics. *Kitware, Inc*. publishers, 4th edition, 2006.

[9] A. Terrazas, J. Ostuni, and M. Barlow. *Java Media APIs: Cross-Platform Imaging, Media and Visualization*. Sams White Book, 2002.

[10] S. A. Urquiza, P. J. Blanco, M. J. Vénere, and R. A. Feijóo. Multidimensional modelling for the carotid artery blood flow. *Computer Methods in Applied Mechanics and Engineering*, 1(195):4002–4017, 2006.

[11] K. E. Willmore, L. Leamy, and B. Hallgrímsson. The effects of developmental and functional interactions on mouse cranial variability through late ontogeny. *Evolution and Development*, *6(8)*: 550–567, Novembro/Dezembro 2006.

Interação com equipamentos convencionais e não convencionais em treinamento médico

Adriano Bezerra, Fátima de L. S. Nunes, Cléber Gimenez Corrêa
Centro Universitário Eurípides de Marília – UNIVEM
Laboratório de Aplicações de Informática em Saúde - LApIS
Marília, SP, Brasil
adrianobezerra1@yahoo.com.br, fatima@univem.edu.br, correacleber@yahoo.com.br

Abstract

This paper presents the insertion of conventional and non-conventional devices in Virtual Reality applications in order to build a comparison about usability, considering a framework for medical training. The process of insertion of a dataglove is presented, by using C and Java language, besides a way for simulating the use of a dataglove by using keyboard. At the end, a comparison between the two equipments is presented, considering aspects like realism, usability and cost.

Keywords: *Non-Conventional Device, Conventional Device, Interaction, Medical Training, Virtual Reality.*

Resumo

O presente artigo discute a interação por meio de dispositivos convencionais e não convencionais a fim de fazer uma comparação de suas usabilidades em aplicações de treinamento médico virtual. É apresentado o processo de inclusão de uma luva de dados utilizando as linguagens Java e C e apresentada uma forma de simular o uso da luva via teclado convencional. Ao final, uma comparação entre os dois equipamentos é apresentada, considerando os fatores realismo, usabilidade e custo.

Palavras-chave: *Equipamentos Não-Convencionais, Equipamentos Convencionais, Interação, Treinamento Médico, Realidade Virtual.*

1. Introdução

O uso da Realidade Virtual (VR) em ferramentas de treinamento médico tem recebido maior atenção durante as últimas décadas [1], considerando principalmente a inclusão de equipamentos que podem trazer um grau de realismo elevado por meio de interação mais natural do usuário com o Ambiente Virtual (AV). Esses equipamentos visam a fornecer apoio multi-sensorial para atender a demanda em tempo real [2].

A interação em AVs é um processo pelo qual o usuário navega, ou seja, se movimenta pelo ambiente, observando sua composição, podendo selecionar, escolher um ou mais entre diversos objetos virtuais; manipular, modificar as características do objeto ou objetos selecionados (posição, rotação, escala), além de controlar o sistema, alterando o seu estado ou o modo de interação [3].

Visto que determinados treinamentos médicos são executados em animais e pacientes reais, podendo causar experiências mal sucedidas e aumento de riscos aos pacientes, as aplicações de RV para esta área oferecem vantagens, tais como: disponibilidade do Ambiente Virtual (AV), inexistência de riscos a pacientes, médicos e estudantes, e diversidade de casos clínicos [4].

Nessas aplicações existe a necessidade de executar procedimentos precisos usando dispositivos de entrada e saída, que são responsáveis pela comunicação entre o usuário e o sistema de computação, classificados geralmente em convencionais (teclado, mouse, monitor, entre outros), e não-convencionais (luva de dados, equipamento háptico, capacete estereoscópico, entre outros).

Esse artigo faz uma comparação quanto à usabilidade na interação do usuário em aplicações de treinamento médico. Para possibilitar maior maleabilidade a formas de interação, um módulo de interação está sendo construído em um *framework* de RV orientado a objetos para treinamento médico, em desenvolvimento [5]. Após a inclusão de uma luva de dados, foram tecidas análises do resultado obtido e uma comparação entre o uso da luva e do teclado, utilizando uma aplicação para treinamento médico implementada usando o *framework* citado. Esse artigo está estruturado da seguinte forma: a seção 2 diz respeito à metodologia, apresentando os materiais e os métodos utilizados no trabalho. A seção 3 descreve os resultados obtidos e as discussões. A seção 4 apresenta a conclusão do artigo.

2. Características dos Dispositivos

No presente trabalho, dois equipamentos estão sendo utilizados. O primeiro é um teclado, equipamento convencional com baixa precisão. O segundo é uma luva de dados *5DT DataGlove 5 Ultra*, fabricada pela 5DT (i) [6]. A luva e o teclado representam a ação de segurar e soltar o órgão humano durante o exame de punção. A Figura 1 mostra a flexão dos dedos e o grau de liberdade que a luva proporciona. A Figura 2 representa o posicionamento necessário dos dedos no teclado, conforme descrito na próxima seção.

Figura 1. Flexão dos dedos proporcionada pelo equipamento *DataGlove 5 Ultra* [6].

Figura 2. Representação dos dedos da mão esquerda nas teclas a serem pressionadas.

A luva de dados possui cinco sensores de fibra ótica localizados na região dos dedos, que permitem a captação de informação sobre a flexão dos dedos [6].

3. Metodologia

A Figura 3 apresenta em exemplo do AV construído com auxílio do framework ViMeT [5], contendo na cena um órgão humano virtual (no caso, uma mama virtual), uma mão virtual que representa a mão de um especialista, e uma seringa virtual que representa o equipamento utilizado para um exame de punção.

Os equipamentos citados na seção anterior captam as informações e transferem-nas para a aplicação, determinando o comportamento dos objetos no AV.

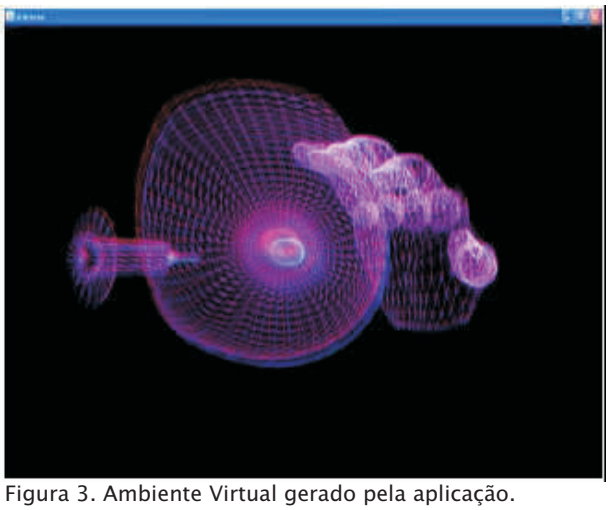


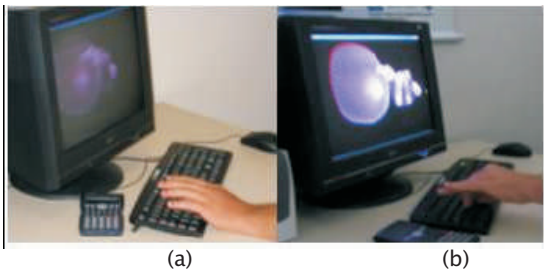
Figura 3. Ambiente Virtual gerado pela aplicação.

Com o teclado, teclas específicas (A, S, D, F e V), ao serem pressionadas, indicam que a mão do usuário está sendo fechada para segurar o objeto que representa o órgão humano. A liberação destas teclas indica que a mão está sendo aberta e o órgão humano virtual deve ser solto. A seleção das teclas levou em consideração o posicionamento da mão do usuário no teclado, de forma a simular da melhor maneira possível o ato de abrir e fechar a mão. O teclado também permite controlar o sistema, alterando, por exemplo, o modo de interação, de navegação pelo AV para manipulação da seringa virtual com o equipamento mouse. As informações provenientes da luva de dados a respeito da flexão dos dedos são transferidas ao sistema fazendo com que os dedos pertencentes a uma mão virtual no AV se movam corretamente e no ângulo desejado. Da mesma forma que o teclado, a flexão captada indica o ato de segurar e soltar o órgão humano pelo usuário.

A inclusão dos equipamentos foi implementada em linguagem Java juntamente com a API (*Application Programming Interface*) Java 3D, disponibilizadas gratuitamente [7]. Para a luva está sendo adotada, além da linguagem Java e sua API Java3D, a linguagem de programação C, a fim de permitir o uso do driver nativo fornecido pelo fabricante.

A integração das linguagens C e Java ocorre por meio do recurso JNI (*Java Native Interface*), que compõem o JDK (*Java Development Kit*), e permitem a um programa em Java, invocar funções escritas em C, com a criação de bibliotecas de ligação, as chamadas DLLs (*Dynamic-Link Libraries*) [8].

No teclado é possível distinguir teclas que poderão atingir dois estados: liberada ou pressionada, como indicado na Figura 4.



(a) (b)

Figura 4. Interação via teclado: (a) teclas soltas; (b) teclas pressionadas.

Com a luva de dados não são distingüidos estados, pois ela tem um grau de liberdade angular na qual o objeto virtual reflete igualmente a flexão dos dedos no AV, propiciando um ganho significativo no realismo, como é mostrado na Figura 5. Portanto, os valores dos sensores são enviados à aplicação, e os dedos da mão virtual se movem de acordo com os movimentos do usuário, sendo apresentados na tela do monitor.

Para a implementação da simulação via teclado é utilizada uma interface disponível na linguagem Java, denominada *KeyListener*, que define os eventos do teclado e permite a utilização de diversos métodos, tais como: *keyPressed* (indica o pressionamento de uma determinada tecla), *keyReleased* (indica a liberação de uma determinada tecla), *keyTyped* (indica o pressionamento de uma tecla que não seja de ação), *getKeyChar* (retorna uma string com o nome da tecla).

A implementação da luva de dados é complexa porque envolve integração entre linguagens, verificação de portas de conexão (USB, por exemplo), inicialização e encerramento da conexão com o dispositivo, leitura de sensores. Um vetor é usado para armazenar os valores dos cinco sensores que compõem a luva de dados empregada neste projeto.

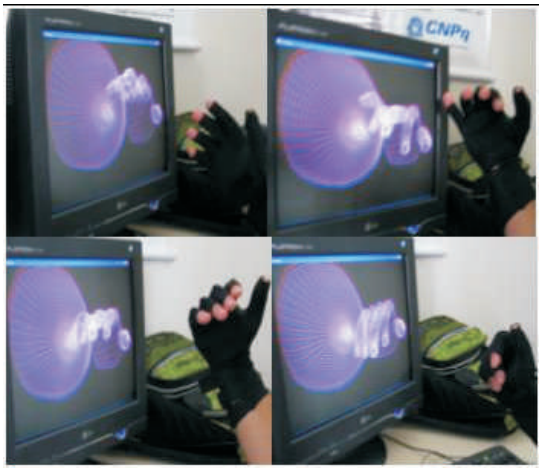


Figura 5. Seqüência da flexão gradativa dos dedos no AVusando a luva de dados.

4. Resultados

A combinação de equipamentos em um sistema tende a facilitar a escolha dos usuários, que podem levar em consideração os recursos disponíveis, os custos do sistema e o grau de realismo desejado para atingir seus objetivos. Neste contexto, a escolha de equipamentos é muito importante, uma vez que envolve diversas questões, como disponibilidade financeira e grau de realismo proporcionado. Equipamentos convencionais como teclado são mais baratos que os não convencionais como luva de dados. Entretanto, esta última oferece um grau de realismo mais elevado em uma simulação, pois se assemelha a objetos do mundo real. Neste trabalho, tem-se como resultado o funcionamento pleno da interação usando os dispositivos citados. Verifica-se, como esperado, que os equipamentos convencionais não produzem a sensação de realismo necessária para a aplicação.

Em relação ao teclado, uma proposta a ser estudada seria fazer uma implementação vinculando o tempo em que a tecla permanece pressionada com o ângulo de rotação de cada dedo.

Com os testes realizados com a luva foi possível observar que o tempo de resposta para alcançar os movimentos desejados na mão virtual, é amplamente superior quando comparado ao do teclado, devido principalmente à maior naturalidade dos movimentos.

Apesar de apresentar um realismo elevado e um tempo de resposta pequeno, a quantidade de sensores disponíveis pela luva utilizada ainda limita a representação de determinados movimentos como, por exemplo, os movimentos das falanges da ponta dos dedos.

Outro ponto a ser observado é que a modelagem e o posicionamento da mão virtual no AV devem ser cuidadosos. É necessária a construção de uma hierarquia de tal forma que a movimentação de algumas partes acarrete a movimentação de outras partes. Isso deve ocorrer, por exemplo, quando se translada a palma da mão, acarretando movimento nos dedos. Por outro lado, os dedos devem ser hierarquicamente separados uns dos outros a fim de que o movimento de um não exerça influência nos seus vizinhos.

5. Conclusões e discussões

É possível verificar que a luva de dados tem um ganho significativo em se tratando de realismo na aplicação, pois além da pessoa ter liberdade e naturalidade para se movimentar com a luva, ela também simula com clareza no AV os movimentos da mão. Com o teclado o usuário fica preso ao equipamento, sem liberdade para

se movimentar, visto que é um equipamento de base fixa. Mesmo que seja utilizado um teclado sem fio, é necessário o seu apoio em uma base, o que dificulta os movimentos do usuário. Além disso, o posicionamento das teclas dificulta a obtenção de realismo.

Apesar da luva ser melhor no sentido de realismo, ela tem um custo mais elevado quando comparado ao do teclado. Por este motivo, em um framework destinado a auxiliar a implementação de ferramentas para a área médica, é interessante disponibilizar a interação via teclado.

Isto viabiliza a demonstração e até o uso da aplicação, mesmo se o realismo fica prejudicado. Assim, na ausência da luva, as aplicações geradas funcionam corretamente com o teclado, sendo possível simular todos os passos inerentes aos procedimentos no exame de biópsia.

A fim de obter dados mais precisos sobre a utilização dos dispositivos apresentados, uma avaliação será feita com profissionais da área médica a fim de verificar a usabilidade e a transparência de informações tal qual o tempo de resposta e o desempenho no AV. Essas informações serão coletadas e utilizadas para aperfeiçoar as aplicações.

6. Referências

[1] Kiss, B., Szijártó, G., Benedek, B., Simon, L., Csukly G., Takács, B., CyberTherapy: *Applications of Virtual Reality and Digital Humans in Clinical Psychology, Második Magyar Számítógépes Grafika és Geometria Konferencia*, Budapest, 2003.

[2] Kirner, C., Siscoutto, R., Realidade Virtual e Aumentada: Conceitos, Projeto e Aplicações, *Livro do Pré Simpósio, IX Symposium on Virtual and Augmented Reality*, pp.9, 2007.

[3] Bowman, D. A., La Viola Jr, J. J., Kruijiff, E., Poupyrev, I., *An Introduction to 3-D User Interface Design. Presence: Teleoperators and Virtual Environments*, pp. 96-108, 2001.

[4] Machado, L. S., Moraes, R. M., Souza, D. F. L., Valdeck, M. C. O. SITEG – Sistema Interativo de Treinamento em Exame Ginecológico, *Proc. 8th SVR Symp. on Virt. and Aug. Reality*, Belém, PA, Brasil, CD-ROM, 2006.

[5] Oliveira, A. C. M. T. G., Pavarini, L., Nunes, F. L. S., Botega L. C., Rossato, D. J., Bezerra, A., Virtual Reality Framework for Medical Training: Implementation of a deformation class using Java. In: *Proceeding of ACM Siggraph International Conference on Virtual-Reality Continuum and its Applications in Industry, Hong Kong, Nova Iorque: ACM Press*, pp. 347-351, 2006.

[6] 5DT Fifth Dimension Technologies. Disponível em: <http://www.5dt.com/hardware.html> . Acesso em: Janeiro 2008.

[7] Sun – Java 3D API Tutorial. Internet site address: <http://java.sun.com/developer/onlineTraining/java3D, acessado em 10/02/2008.

[8] Sun – Java Native Interface Specification, Disponível em: <http://java.sun.com/j2se/1.4.2/docs/guide/jni/spec/jniTOC.html>, acessado em 10/02/2008.

Integração de um Framework de Realidade Virtual com um Sistema de simulação de objetos 3D para aplicações de treinamento médico

Ana Cláudia M. T. G. de Oliveira[1,2], Sérgio R. Delfino[1], Fátima L. S. Nunes[1]

[1]LApIS – Laboratório de Aplicações de Informática em Saúde
Centro Universitário Eurípides de Marília (UNIVEM)
Marília – SP – Brasil

[2]FATEC –Faculdade de Tecnologia de Garça
Garça - SP - Brasil

{anatiessi}, {srdelfino}@gmail.com, fatima@univem.edu.br

Abstract

Tools to simulate medical procedures may be built aiming at training students and professionals to execute actions that require accuracy and efficiency, thus decreasing the treatment cost at medium term. The Virtual Reality applications in such area make use of virtual objects that simulate human organs and medical tools. Nevertheless, in order to reach the realism desired in these applications it is interesting to use objects built considering characteristics obtained from real objects. This paper describes an integration of a three-dimensional object simulation system with a Framework used to develop medical applications focusing on the virtual training of biopsy exams.

Keywords: *3D Simulation, Deformation, Framework, Java3D Stereoscopy, Virtual Reality*

Resumo

Ferramentas que simulam procedimentos médicos podem ser construídas com a finalidade de treinar estudantes e profissionais para a execução de ações que requerem precisão e eficácia, diminuindo o custo de treinamento em médio prazo. As aplicações de Realidade Virtual para essa área utilizam em sua grande maioria objetos modelados que simulam órgãos humanos e instrumentos médicos. Porém para alcançar o realismo desejado nessas aplicações é interessante a utilização de objetos construídos a partir de características obtidas de objetos reais. Neste artigo é descrita a integração de um sistema de simulação de objetos tridimensionais a partir da extração de características de imagens médicas a um Framework de RV para desenvolvimento de aplicações médicas, tendo como foco o treinamento virtual de exames de biópsia.

Palavras-chave: *Deformação, Estereoscopia, Framework, Java3D, Realidade Virtual, Simulação 3D.*

1. Introdução

Nos últimos anos tem-se observado um aumento na quantidade e qualidade de aplicações de Realidade Virtual (RV) para a área médica em centros de pesquisa ao redor do mundo. Neste contexto, têm surgido também ferramentas que permitem o reúso de código, tais como bibliotecas e Frameworks, com o propósito de aumentar a produtividade na implementação de sistemas de RV especificamente para a área médica. No Brasil, essa situação também tem sido percebida – vários grupos de pesquisa já vêm apresentando aplicações empregadas no dia-a-dia da prática médica e ferramentas para auxiliarem a implementação de novas aplicações. Essas últimas, de forma geral, oferecem funcionalidades comuns à maioria das aplicações de treinamento médico como métodos de deformação, detecção de colisão, estereoscopia, criação de ambientes virtuais dinâmicos e acesso a dispositivos convencionais e não-convencionais. Exemplos dessas ferramentas e bibliotecas desenvolvidas no Brasil são apresentados em [1, 2, 3, 4 e 5].

Um dos desafios para construir ambientes virtuais para treinamento médico é a disponibilização de casos para estudos, que devem ser em grande quantidade, com características variadas e reais. Dentro deste desafio, destaca-se a necessidade de construir objetos que simulem órgãos humanos de maneira realística. Há duas abordagens básicas para suprir esta necessidade: modelagem de objetos usando aplicativos próprios para esta finalidade e simulação/reconstrução de objetos tridimensionais (3D) a partir de imagens reais, adquiridas por equipamentos médicos.

A questão que se depara aqui é que ambas as estratégias têm vantagens e desvantagens e as duas exigem conhecimento específico da aplicação e dos objetos a serem representados. Isto significa que não é possível utilizar modelagens genéricas ou aplicar métodos de

reconstrução genéricos sem considerar os objetivos da aplicação.

Desta forma, tem-se um impasse: as ferramentas e bibliotecas que auxiliam na construção de aplicações são genéricas dentro do seu domínio e aquelas destinadas a propiciarem objetos 3D são específicas para a finalidade de cada aplicação. Em algum momento do desenvolvimento, há a necessidade de integrarem-se as ferramentas.

Este trabalho apresenta a integração de ferramentas neste sentido, considerando um *Framework* para simular aplicações de treinamento médico e um sistema de simulação que utilizou técnicas de processamento de imagem e técnicas de RV para construir objetos 3D a partir de medidas extraídas de imagens bidimensionais provenientes de mamografia, para mais detalhes sobre a arquitetura, características, desempenho e testes realizados nestas ferramentas [6, 7]. Consiste em uma evolução de um protótipo de uma ferramenta para simular o exame de punção mamária [8]. Neste protótipo eram utilizados objetos modelados fixos, sem informações a respeito de pacientes e com procedimentos precários de deformação. Neste artigo são apresentadas algumas das melhorias realizadas no referido projeto. Após a integração foi considerado como estudo de caso um sistema que visa a simular o exame de biópsia mamária. Com isso, usuários podem treinar virtualmente os procedimentos para a realização desse exame a partir de dados extraídos de casos reais.

2. Metodologia

Para o desenvolvimento do Framework foi utilizada a metodologia proposta em [9] e utilizou-se para implementação a linguagem de programação Java e sua API Java 3D [10]. Para que haja um reúso eficiente deve haver uma documentação que auxilie o desenvolvimento de novas aplicações a partir do Framework. Para o Framework foi construído um manual de instanciação e a documentação de cada classe, no formato javadoc [11]. Além do Framework foi desenvolvida também uma ferramenta de instanciação automática [12], cuja interface é apresentada na Figura 1. Na Figura 2 é apresentada uma aplicação desenvolvida utilizando a ferramenta de

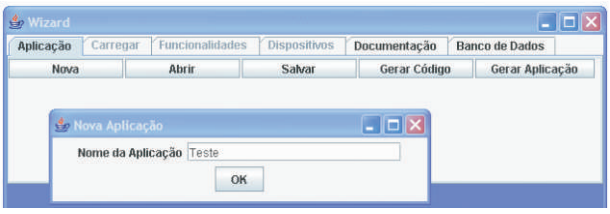


Figura 1. Interface da ferramenta de instanciação.

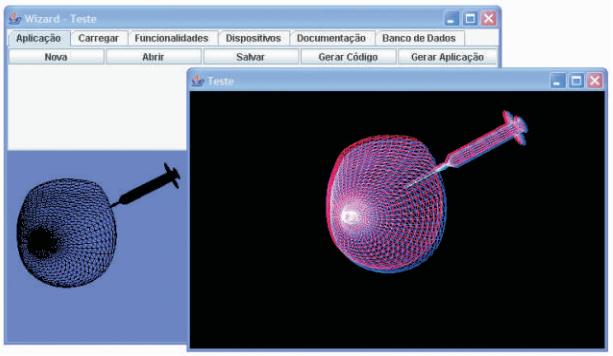


Figura 2. Aplicação gerada pela ferramenta de instanciação.

instanciação.

As aplicações desenvolvidas para a realização de testes do *Framework* utilizaram objetos sintetizados, porém verifica-se que a integração de ferramentas que criam objetos tridimensionais a partir de imagens reais oferece maior personalização às aplicações.

O sistema de geração de casos reais está dividido em módulos: Obtenção das imagens, responsável por obter as imagens mamográficas bidimensionais; Pré-processamento, que prepara a imagem para a análise e mensuração de características e estruturas da mama; Segmentação da área da mama nas imagens bidimensionais; Segmentação do nódulo; Extração de medidas da mama e do nódulo (altura, largura e localização) e Geração do modelo 3D. Neste sistema é possível o usuário escolher o número de vértices e arestas a malha.

A integração consiste em utilizar os objetos modelados pelo sistema de simulação, definindo-se formatos e parâmetros apropriados. Para isso foram realizados os seguintes passos: (1) definição do formato comum às duas ferramentas - foi definido usar arquivos com a extensão .obj, que permitem representar objetos 3D a partir de primitivas; (2) inserção dos objetos em um banco de dados usado pela ferramenta de instanciação; (3) inserção dos objetos modelados no AV usando a ferramenta de instanciação; (4) estudo e definição de parâmetros adequados para os objetos simulados e (5) criação do Ambiente Virtual a partir da escolha de funcionalidades desejadas pelo usuário e disponibilizadas pelo *Framework*. A Figura 3 apresenta o objeto 3D carregado na ferramenta de instanciação. Um exemplo completo com todos os passos executados é apresentado na próxima seção.

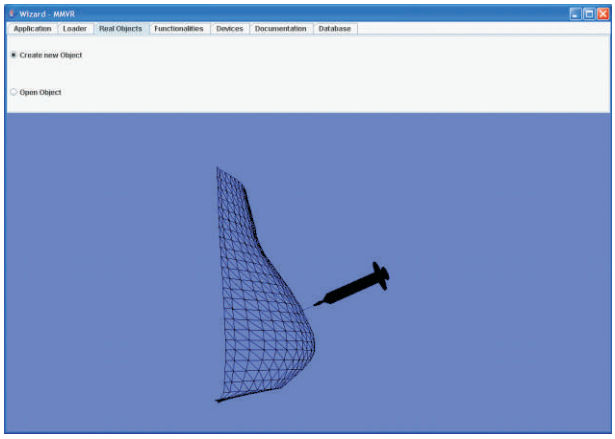


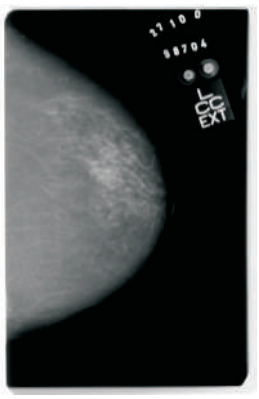
Figura 3. Objeto gerado pelo sistema de simulação 3D carregado na ferramenta de instanciação automática.

3. Resultados e discussões

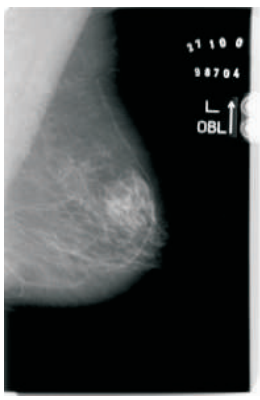
A fim de verificar a integração do sistema de simulação 3D ao Framework foi realizado um estudo de caso que será apresentado a seguir. As imagens reais que foram utilizadas são apresentadas na Figura 4(a) e Figura 4(b). Estas imagens passaram por etapas de pré-processamento e processamento citadas a seguir.

Na Figura 5(a) é apresentado o resultado das etapas de segmentação que visa a identificar a área da mama e de um nódulo presentes em ambas as visões das imagens bidimensionais. Na Figura 5(b) são mostrados os objetos 3D modelados a partir das medidas extraídas das imagens segmentadas.

O sistema de simulação 3D possui sua própria interface gráfica, conforme a Figura 6, porém no *Framework* esta interface não é necessária e na ferramenta de instanciação é utilizado somente o resultado do sistema que consiste no objeto modelado a partir de imagens médicas [6].



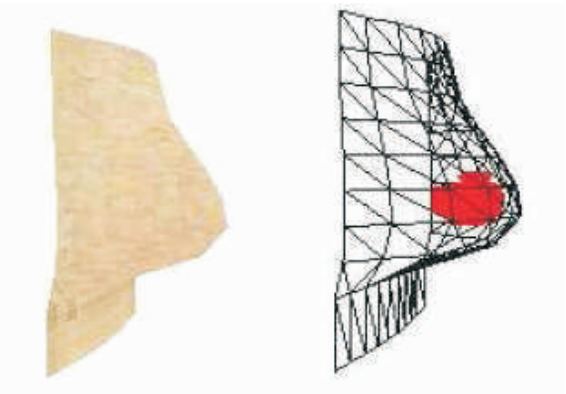
(a)



(b)

Figura 4. exemplo de imagens bidimensionais usadas: (a) imagem mamográfica – visão Crânio Caudal e (b) imagem mamográfica – visão Médio Lateral .

(a)



(b)

Figura 5. Etapas do sistema de simulação de objetos a partir de imagens reais: (a) Etapa da Segmentação da mama e nódulo, (b) Objetos 3D modelados a partir de imagens reais.

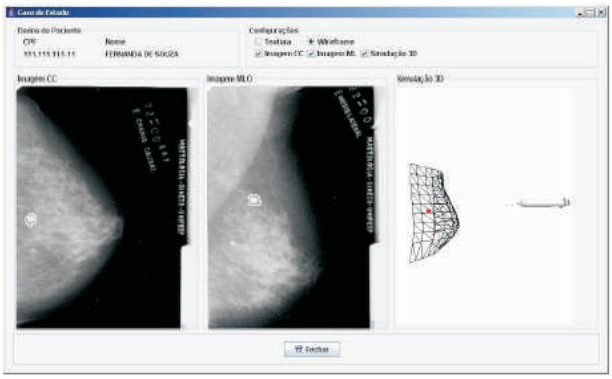


Figura 6. Interface Gráfica do sistema de simulação 3D.

A integração do sistema de simulação 3D consiste na disponibilização dos objetos gerados a partir do sistema para serem utilizadas no *Framework*. Porém não é uma tarefa trivial, pois os objetos gerados a partir do referido sistema foram obtidos por algoritmos próprios implementados no sistema e não com a utilização de ferramentas de modelagem. Esta diferença acarretou alguns problemas cujas soluções são apresentadas a seguir.

Quando o objeto é inserido no AV por meio da ferramenta de instanciação, ele fica bem menor do que aquele disponibilizado na interface do sistema de simulação. Outra questão pertinente é o posicionamento do objeto no AV, pois originalmente ocorre uma rotação no eixo Z. (Figura 7). Na ferramenta de instanciação é possível acertar o

posicionamento do objeto, por meio de alterações nos parâmetros de escala, translação e rotação (Figura 8). No entanto, após este reposicionamento as funcionalidades do *Framework* também precisam sofrer alterações nos parâmetros.

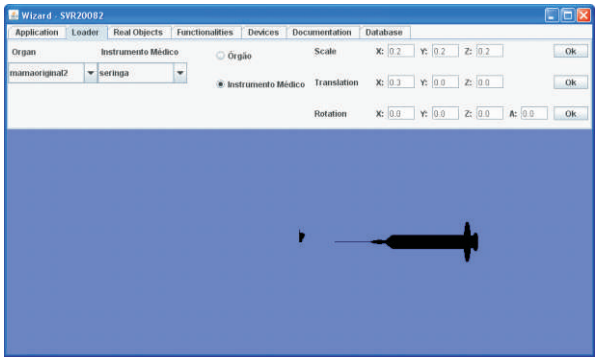


Figura 7.Aplicação desenvolvida por meio da *ferramenta de instanciação* utilizando o objeto do sistema de simulação 3D.

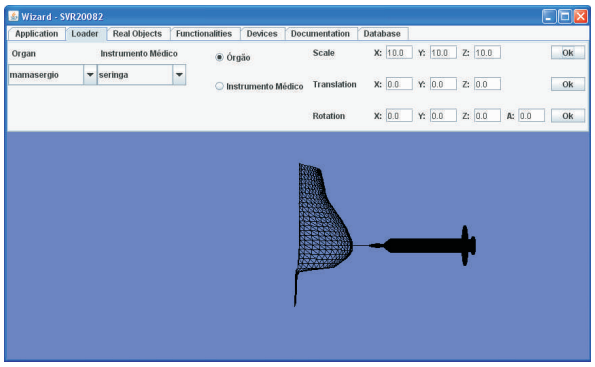


Figura 8. Resultado da alteração da escala e do reposicionamento do objeto.

Na Figura 9 é apresentada uma aplicação desenvolvida por meio da ferramenta de instanciação utilizando o objeto gerado pelo sistema de simulação 3D na posição original. Para a deformação houve a necessidade de alterar os parâmetros referentes à força e à massa [13]. Como o objeto representa somente a superfície da mama é possível visualizar o prolongamento dos vértices causado pela deformação.

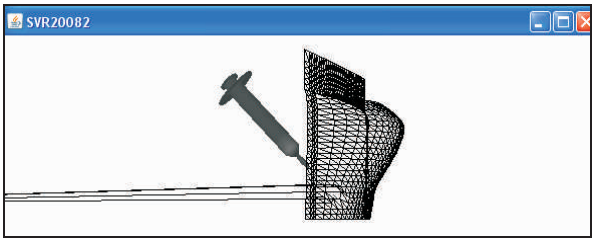
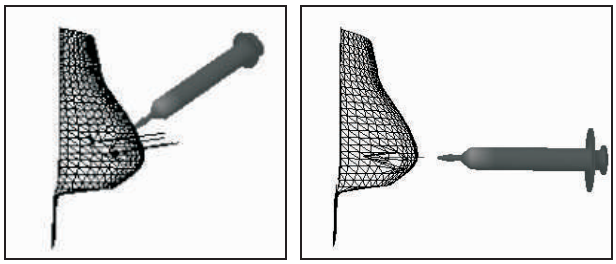


Figura 9. Aplicação gerada pela ferramenta de instanciação com problemas devido a parâmetros referentes às funcionalidades do Framework.

Na Figura 10 (a) é apresentada uma aplicação gerada pela ferramenta de instanciação utilizando o objeto reposicionado e sem alterações nos valores padrões das funcionalidades do Framework, podendo ser observado que a deformação ocorre em relevo e não em profundidade. Após correção nos parâmetros das funcionalidades de deformação e detecção de colisão, obtém-se a aplicação apresentada na Figura 10 (b).



(a) (b)

Figura 10. (a) Aplicação gerada pela ferramenta de instanciação com parâmetros default. (b) Aplicação gerada pela ferramenta de instanciação com alteração nos parâmetros.

A única funcionalidade que tem o seu funcionamento comprometido e a estereoscopia, tornando-se assim uma limitação e também foco de pesquisa. Estes parâmetros devem ser corrigidos para que as aplicações possuam características mais próximas das obtidas nos procedimentos médicos reais.

4. Conclusões e trabalhos futuros

O presente trabalho traz como contribuição o processo de integração de um Framework de RV para desenvolvimento de aplicações de treinamento médico com um sistema de simulação 3D. Esta integração proporciona um maior realismo ao treinamento médico (aqui especificamente ao procedimento de biópsia mamária), permitindo aos profissionais da área médica treinarem os procedimentos referentes a esse exame, sem a necessidade de cobaias ou cadáveres reduzindo o custo e aumentando a qualidade do treinamento.

A integração do sistema de simulação proporciona um avanço para o Framework, pois no desenvolvimento de aplicações a partir dele eram somente utilizados objetos modelados que simulam órgãos humanos sem nenhuma informação real. Com a integração o usuário desenvolver aplicações a partir de imagens reais. A utilização destas imagens abre um novo foco para a pesquisa que consiste no estudo dos parâmetros utilizados na estereoscopia.

Salienta-se que o processo de integração aqui desenvolvido para biópsia mamária pode ser aplicado a imagens provenientes de qualquer órgão humano ou

modalidade médica. O requisito básico é que o sistema de criação dos objetos 3D considere os formatos reconhecidos pelo *Framework*, que podem ser obtidos na documentação deste [11]. Assim, é possível implementar aplicações para treinamento de exames de biópsia quaisquer com rapidez e flexibilidade, usando o *Framework* citado, e com objetos com características reais.

As limitações que se pretendem resolver em trabalhos futuros são: o estudo dos parâmetros das funcionalidades (deformação, detecção de colisão e estereoscopia) e testes com objetos modelados a partir do sistema de simulação 3D que simulem outros órgãos humanos.

5. Referências

[1] SOUZA, D.F.L.; CUNHA, I.L.L.; SOUZA, L.C.; MORAES, R.M.; MACHADO, L.S. (2007) Development of a VR Simulator for Medical Training Using Free Tools: A Case Study. In: *Proc. of Symposium on Virtual and Augmented Reality (SVR'2007)*, pp. 100-105.

[2] RODRIGUES, M. A. F.; SILVA, Wendel Bezerra; BARBOSA NETO, Milton Escóssia et al. Um Sistema de Realidade Virtual para Tratamento Ortodôntico. In: *VIII Simposium on Virtual Reality*, 2006, Belém, Pará, Brazil. p. 433-444.

[3] VILLAMIL, Marta B.; NEDEL, Luciana P.; FREITAS, Carla M.D.S.; MACQ, Benoit. An Anatomy-based Approach to Simulate the Temporomandibular Joint Movement. In: Computer Animation and Social Agents (CASA), 2007, Hasselt. *Proceedings of the 20th International Conference on Computer Animation and Social Agents*. Hasselt : Hasselt University, 2007. v. 1. p. 41-47.

[4] TREVISAN, Daniela G.; NICOLAS, Vincent; MACQ, Benoit; NEDEL, Luciana P. MedicalStudio: a medical component-based framework. In: *Workshop de Imagens Médicas (WIM)*, 2007, Recife. Anais do Workshop de Imagens Médicas (em CD), 2007.

[5] BALANIUK, Remis ; COSTA, Ivan Ferreita da ; MELO, J. . Cosmetic Breast Surgery Simulation. In: *Proceedings of the VIII Symposium on Virtual Reality - SVR2006*, 2006, Belém.

[6] DELFINO, S. R.; NUNES, F. L. S. Utilizando a tecnologia java para geração dinâmica de estudos de caso para treinamento médico. In: *Anais do VI Workshop de Informática Médica*. Vila Velha-ES: 2006.

[7] OLIVEIRA, Ana Cláudia Melo Tiessi Gomes de, NUNES, F. L. S., BEZERRA, A. Concepção e Implementação de um Framework para simulação de exames de punção usando Realidade Virtual In: *Proceedings of IX Symposium on Virtual and Augmented Reality*, 2007, Petrópolis. p.253 - 262.

[8] LIMA, L., NUNES F. L. S. et al. Virtual Reality for medical training: a prototype to simulate breast aspiration exam In: *Proceedings of ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*, p. 328-331, 2004.

[9] BOSCH, J.; et al.. Framework Problems and Experiences. In: *FAYAD, M.; JOHNSON, R.; SCHMIDT D. Building Application Frameworks: Object-Oriented Foundations of Framework Design*. Nova Iorque : John Willey and Sons, 1999. p. 55-82.

[10] SUN. (2006a).The JavaTM Tutorial. Disponível em: <http://java.sun.com/docs/books/tutorial>. Acesso em: jun. 2006.

[11] OLIVEIRA, A. C. M. T. G. Documentação. Disponível em: <http://galileu.fundanet.br/bcc_bsi/bcc/lapis/projetos/ana/docum.php>.

[12] MARQUES F. L. S. N.; OLIVEIRA A. C. M. T. G. de; ROSSATO, D. J.. ViMeTWizard: Uma ferramenta para instanciação de um framework de Realidade Virtual para treinamento médico. In: *XXXIII Conferencia Latinoamericana de Informática*, 2007, San José.

[13] PAVARINI, L. et al.. DefApliMed – Sistema de Deformação para Aplicações Médicas, com base no método Massa-Mola, utilizando a API Java 3D . In: *Proceedings of the VIII SYMPOSIUM ON VIRTUAL REALITY, 2006*, Belen. …, 2006. CD-ROM.

O uso da Realidade Virtual não-imersiva para o auxílio ao tratamento da aviofobia pelos profissionais da psicologia

Danielly Cristine de Medeiros, Wender Antônio da Silva, Marcos Wagner S. Ribeiro, Manfred Bogen
ILES/ULBRA Itumbiara-GO
Grupo de Realidade Virtual de Goiás
{dc.medeiros}, {wender_silva}, {marcos_wagner}@yahoo.com.br

Nadabe Cardoso O. Alves Fortes
Universidade Estadual de Goiás
Grupo de Realidade Virtual de Goiás
nadabef@gmail.com

Sheister Oliveira Andrade
Agência Nacional de Aviação Civil
Cód. ANAC: 686659
sheister_andrade@yahoo.com.br

Edgard A. Lamounier Júnior, Alexandre Cardoso, Ezequiel Roberto Zorzal
Faculdade de Engenharia Elétrica
UFU – Uberlândia-MG
{lamounier}, {alexandre}, {ezorzal}@gmail.com

Abstract

Phobia is an unjustified fear, a constant anxiety, an intense and unrealistic feeling that manifests when a phobic person is faced with a situation that triggers the phobia. This article presents the application of Virtual Reality as a means to help overcome flight phobia. The virtual reality environment has three levels of interaction and challenges for the Phobic patient.

Resumo

A fobia é um sentimento de medo injustificável, ansiedade persistente, intensa e irrealística, sentimento o qual surge quando o fóbico se depara com a situação causadora de sua fobia. Este artigo apresenta uma aplicação em Realidade Virtual para auxiliar o tratamento de superação de aviofobia, o ambiente em Realidade Virtual possui três níveis de interação e de dificuldades para o paciente fóbico.

1. Introdução

A fobia é um sentimento desproporcional de medo, que se intromete persistentemente no campo da consciência e se mantém ali, independentemente do reconhecimento de seu caráter absurdo. A característica essencial da fobia consiste num temor patológico que escapa à razão e resiste a qualquer espécie de objeção, temor este dirigido a um objeto (ou situação) específico [2].

Sabe-se que a fobia não é causa de nada, mas sim conseqüência; é o reflexo de alguma angústia interna, e em princípio deve ser tratada como tal, ou seja, deve-se tratar a angústia que causou e não a fobia em si. Tratando-se e curando a angústia estará curada a fobia.

São várias as situações fóbicas freqüentes: injeções ou pequenas cirurgias, comer ou beber na frente das pessoas, hospitais, viajagem sozinho de carro ou ônibus, e viagens de avião [6].

Existem diversos tipos de tratamentos para fobias, um entre esses diversos tipos de tratamentos que é importante ressaltar, trata-se da terapia comportamental, a qual serve como base para os tratamentos por meio da realidade virtual. A terapia comportamental pode ser definida como um modo científico e particular de aplicação sistemática dos princípios da aprendizagem à mudança do comportamento, no sentido de promover formas mais adaptativas e positivas de interação. De maneira geral o objetivo da terapia é eliminar o sofrimento das pessoas por meio da alteração do comportamento problema propiciando novas condições de aprendizagem e, conseqüentemente, favorecendo um estilo de vida mais gratificante para a pessoa que solicita este tipo de ajuda [7]. Assim, o presente estudo abrange o uso da Realidade Virtual no auxílio ao tratamento de aviofobia.

1.1 Aviofobia

Aviofobia trata-se de fobia de avião [4]. Assim, fobia de avião não é um sentimento apenas de pessoas que nunca viajaram de avião, ela afeta indiscriminadamente, passageiros novatos e veteranos. Trata-se de um medo na maioria das vezes inexplicável, o qual se manifesta em diferentes formas [2]. O medo de voar é muito freqüente. Algumas pessoas têm medo de acidentes aéreos, mas mesmo assim, não evitam o vôo. Outras desenvolvem uma fobia, porque evitam o vôo. Não conseguem sequer entrar num avião [5].

1.2 Realidade Virtual

A realidade virtual iniciou-se após a Segunda Guerra Mundial, para treinamento da Força Aérea dos Estados Unidos, onde havia simulação de vôo para pilotos sem experiência [8].

A realidade virtual é considerada uma interface avançada para aplicações computacionais, que permite ao usuário a movimentação (navegação) e interação em tempo real, em um ambiente tridimensional, podendo fazer uso de dispositivos multisensoriais, para a atuação ou feedback [9].

Assim pode-se entender que realidade virtual permite a criação de uma interface homem – máquina natural e poderosa, possibilitando ao usuário interação, navegação e imersão em um ambiente tridimensional sintético, gerado pelo computador por meio de canais multisensoriais dos cinco sentidos (visão, audição, tato, olfato e paladar) [4].

2. Trabalhos Relacionados

Já existem alguns trabalhos usando a Realidade Virtual para o auxílio ao tratamento de fobias.

No artigo denominado *"Using Augmented Reality to Treat Phobias"* [1], é demonstrada a criação de um protótipo para o tratamento de desordens psicológicas (fobias de aranhas e baratas). De acordo com os autores, notaram-se resultados positivos, sendo suas aplicações úteis como uma ferramenta terapêutica para várias outras desordens psicológicas, a figura 1 ilustra a aplicação.

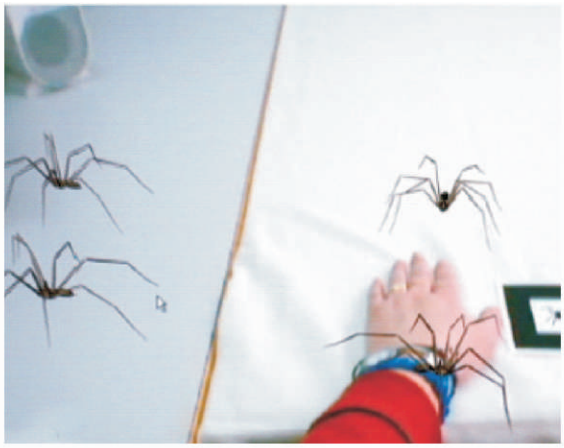


Figura 1. Ambiente 1 – Realidade Aumentada [1]

No trabalho *"VESUP: O Uso de Ambientes Virtuais no Tratamento de Fobias Urbanas"* [10] os autores descrevem um sistema composto de três ambientes: elevador convencional, elevador panorâmico e túnel, conforme figura 2 a seguir. Os ambientes desenvolvidos são voltados, especialmente, para o diagnóstico e

Figura 2. Imagem do túnel [10].

tratamento de pessoas com fobias associadas à vida em grandes centros urbanos.

Outro interessante trabalho intitulado: *"Sistema de Realidade Virtual para Tratamento de Fobia"* [3], onde foram criados cinco cenários: Túnel, metrô, aeroporto/avião, elevador e ônibus, e em sua maioria, foram feitos pensando em transtornos, agorafóbicos, sendo útil para outras formas de fobias.

3. Implementação do Sistema

Foi modelado um ambiente de realidade virtual não-imersiva. A este sistema foram adicionados recursos de som e vídeo, assim, o sistema apresenta características mais fortes de imersão ao usuário/paciente que estiver sendo exposto ao ambiente. Desta forma, para criar sensação de realismo foram adicionados sons reais da turbina do avião, assim, temos o som peculiar de cada etapa: da decolagem e pouso, sons como se estivesse ocorrendo turbulências no vôo, e vídeos nas janelas do avião, onde e demonstrado os eventos que ocorrem nestas situações. Como ferramenta de modelagem, foi ultilizado em grande parte do trabalho o Flux Studio e em alguns momentos o 3DS Max, juntamente com o VRML. Justifica-se a escolha do Flux Studio como ferramenta de modelagem pelo fato do mesmo não gerar códigos tão extensos e por possuir uma grande diversidade de funcionalidades técnicas, e também por nos oferecer uma grande gama de recursos, assim facilitando o processo de modelagem. O ambiente virtual modelado possui objetos como: a pista, aeronave, visão da parte interna e várias visões externas (vídeos) da aeronave por meio das janelas do avião. A figura 3, a seguir é uma representação gráfica do ambiente no Flux Studio.

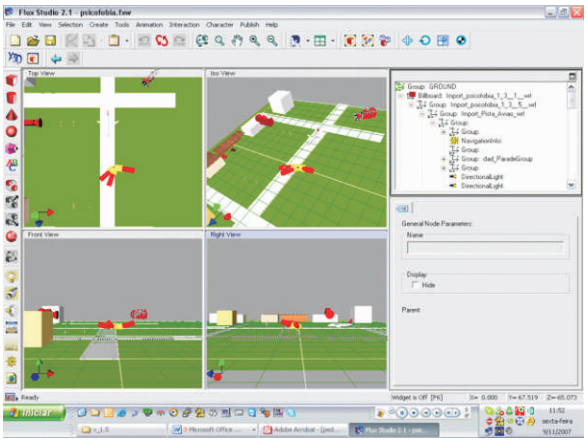


Figura 3. Ambiente gráfico Flux Studio.

3.1. Funcionamento do sistema

O protótipo de software desenvolvido é um sistema de Realidade Virtual não-imersiva, que possibilita ao profissional de psicologia e usuários, treinar e testar todos os níveis de fobia, no tratamento da aviofobia. Desta forma o software propõe três, níveis: 1- O paciente é exposto até o momento em que ele vai adentrar a aeronave, 2- O paciente adentra a aeronave, 3- O paciente segue durante a decolagem na aeronave. Cada etapa da simulação é responsável em auxiliar o psicólogo a medir o grau de fobia do paciente. Desta forma podemos verificar que: - Nível 1: permite estudar e avaliar o paciente, observando suas reações diante a aeronave. A avaliação é realizada pelo profissional, o qual ao perceber a evolução do paciente, permite que ele passe ou não para o próximo nível. - Nível 2: É observado as reações do paciente dentro da aeronave. Onde o paciente sempre terá na sua visão frontal, um corredor onde ele caminha para sua devida poltrona, com o auxilio de uma rota pré-definida e/ou utilizando-se da interação existente no VRML, onde se pode navegar com mouse ou teclado. - Nível 3: Neste nível o paciente já fica sentado ao lado da janela e começa a ver por meio da mesma um ambiente (vídeo), de forma que ele perceba que o avião irá decolar. Desta forma o profissional começa a observar as reações que o paciente tem durante esse vôo, nessa viagem que ele faz pelo mundo virtual.

Os níveis foram definidos de acordo com sua funcionalidade, o mesmo obedece a uma seqüência de realização do procedimento real. Uma vez que o objetivo do produto final é oferecer um instrumento de auxilio no tratamento de aviofobia. E importante destacar que a implementação de cada etapa do sistema vem sendo acompanhada por profissional da área de psicologia.

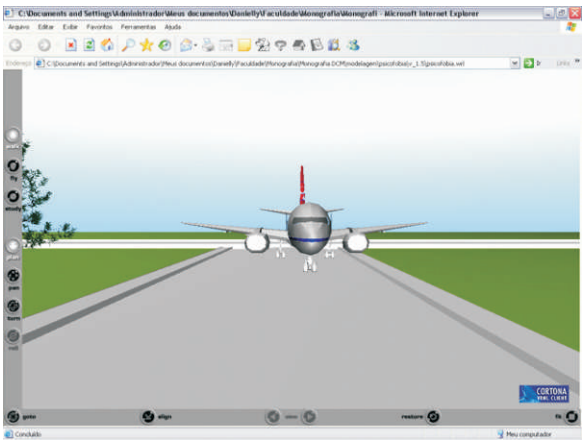


Figura 4. Visão inicial do protótipo.

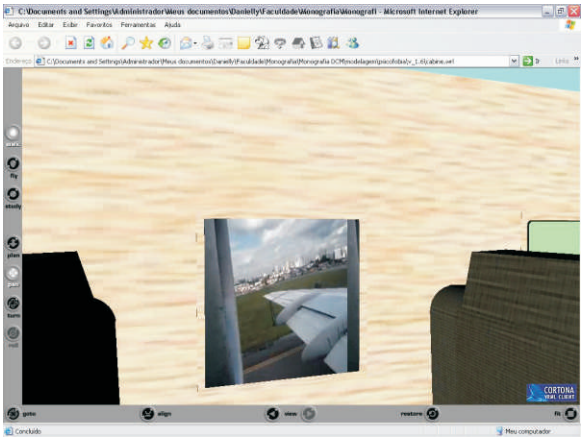


Figura 5. Vista janela (Vídeo).

As figuras 4 e 5 abaixo apresentam parte do sistema em que o usuário navega. Na figura 4 temos toda parte externa, esta é a visão inicial que o paciente/usuário possui para adentrar no ambiente (aeronave). Na figura 5, tem-se a janela do avião, onde, vídeos representam de modo real uma decolagem e uma aterrissagem.

4. Avaliação

O protótipo de software construído foi apresentado a profissionais da área de psicologia, para que os mesmos em parceria pudessem avaliar e ajudar na avaliação da interface e da usabilidade com um grupo de indivíduos. Foram realizadas avaliações, por meio de questionário baseado ISO Norm 9241 Usability, foi possível verificar que o software está dentro das normas ISO de usabilidade. Porém com este teste não foi possível ainda concluirmos sobre a viabilidade do software como instrumento de auxílio para tratamento de fobias, especificamente aviofobia. No entanto a avaliação dos profissionais da psicologia foi positiva, gerando boa discussão e muitas idéias.

5. Conclusão

Constatou-se que a Realidade Virtual pode trazer grandes facilidades na área médica e psicoterapêuticos, proporcionando aos psicólogos, uma ferramenta que o auxilia no tratamento de doenças fóbicas, como a aviofobia abordada neste trabalho. E quanto ao protótipo construído, concluí-se que a criação de um protótipo de software deste nível, pode ser um instrumento de auxílio no tratamento de aviofobia, e ao funcionar corretamente, torna-se um grande diferencial para as clínicas especializadas em tratamentos de fobias, pois acaba com os altos custos relacionados ao deslocamento do paciente até o ambiente que neste caso trata-se de um aeroporto, e a locação de uma aeronave para que o paciente seja exposto

6. Referências

[1] ALCAÑIZ, Mariano; MONSERRAT, Carlos, Using Augmented Reality to Treat Phobias. Disponível em: <http://www.computer.org/publications/dlib>. Acesso em: 15 mai. 2007, 14h30min.

[2] BALLONE, Geraldo J. Medos, fobias e outros bichos. Disponível em: <http://gballone.sites.uol.com.br>. Acesso em: 10 mai. 2007, 14h00min.

[3] MEDEIROS, Gustavo Adolfo de. Sistema de Realidade Virtual para Tratamento de Fobia. *Dissertação de mestrado em Ciências da Computação*, Universidade Federal do Rio de Janeiro, setembro de 2006.

[4] CECILIA, Sandra. Fobias. Disponível em: <http://www.spiner.com.br/modules. php>. Acesso em: 13 mai. 2007, 01h27min.

[5] LUTUFO NETO, F., "Programa saúde da Família", Disponível em: <http://idssaude.uol.com.br/.asp>. Acesso em 03 mar 2007, 17h48min.

[6] SANT'ANA, Vânia L. P. A Psicoterapia Analítico-Comportamental. Disponível em: <http://www.dpi.uem.br/vi-semanapsi>. Acesso em: 08 mai. 2007 18h36min.

[7] THEOPHILO, Roque. Introdução ao Glossário de Fobias. Disponível em: <http://www.psicologia.org.br>. Acesso em: 12 mai. 2007 21h40min.

[8] KIRNER, C. Dr. Sistemas de Realidade Virtual. Disponível em: <http://www.dc.ufscar.br/~grv/tutrv/tutrv.htm>. Acesso em: 04 jul 2007 01h42min.

[9] MACHADO, Liliane dos Santos, MORAES, Ronei Marcos de. Realidade virtual aplicada à medicina. Organizado por: TORI, Romero; KIRNER, Cláudio; SISCOUTTO, Robson. Fundamentos e tecnologia de realidade virtual e aumentada. *Belém, VIII Symposium on Virtual and Augmented Reality*, Maio de 2006.

[10] WAUKE, Ana Paula T, COSTA, Rosa Maria E. M., CARVALHO, Luis Alfredo V. de. VESUP: O Uso de Ambientes Virtuais no Tratamento de Fobias Urbanas. Disponível em: <http://telemedicina.unifesp.br/pub/SBIS/CBIS2004/trabalhos/arquivos/585.pdf>. Acesso em: 16 mai. 2007, 18h03min.

Proposta de Arquitetura Baseada em VRML e PHP para e-commerce

Leandro Silva Campos, Marcos Wagner S. Ribeiro
ILES/ULBRA Itumbiara-GO - Grupo de Realidade Virtual de Goiás
{leandrocamposbr}, {marcos_wagner}@hotmail.com

Jucélio Costa de Araújo
CEFET Morrinhos-GO
juccaraujo@yahoo.com.br

Wender Antônio da Silva
ILES/ULBRA Itumbiara-GO - Grupo de Realidade Virtual de Goiás
wender_silva@hotmail.com

Eliane Raimann
CEFET Jataí-GO
elianeraimann@yahoo.com.br

Abstract

This paper presents a proposal of architecture for development of virtual 3D shops, based in Non-Immersive Virtual Reality technologies, aiming at to offer more realistics interfaces in e-commerce applications, and collaborating to supplant the difficulties imposed for the actual scenario based in websites. The presentation of the proposed architecture is made by means of the development of a virtual shop prototype, which implements the interaction among a 3D environment modeled in VRML and a 2D environment constructed in HTML and PHP.

Keywords: *E-commerce; PHP; Virtual Reality; VRML.*

Resumo

Este trabalho apresenta uma proposta de arquitetura para o desenvolvimento de lojas virtuais 3D, baseada em Realidade Virtual não-imersiva, a qual visa oferecer interfaces mais realistas nas aplicações de e-commerce, colaborando para suplantar as dificuldades impostas pelo cenário atual baseado em websites. A apresentação da proposta de arquitetura é feita por meio de um protótipo de loja virtual, a qual implementa a interação entre um ambiente virtual 3D modelado em VRML e um ambiente 2D construído em HTML e PHP.

Palavras-chave: *E-commerce; PHP; Realidade Virtual; VRML.*

1. Introdução

O entendimento de que a Economia Digital baseada na Internet será a base do desenvolvimento sustentável e a principal fonte de geração de riqueza das nações no século XXI [1], tem feito com que diversas organizações governamentais e não-governamentais envidem esforços no sentido de fortalecer e consolidar o comércio eletrônico (*e-commerce*) em todos os seus segmentos.

Esta pesquisa justifica-se devido ao fato de que o uso da Realidade Virtual em e-commerce pode oferecer interfaces mais realistas nas aplicações, colaborando para suplantar as dificuldades impostas pelo cenário atual de lojas virtuais baseadas em websites. Segundo [10], o modelo atual de *e-commerce* é uma mera adequação do modelo de website comum, acrescido de alguns poucos recursos. O ambiente resultante não lembra em nada o ambiente das lojas reais. Segundo [3], esta distância no contexto visual não satisfaz as necessidades emocionais e sociais das pessoas. Além disso, devido ao fato de não ser um ambiente envolvente, torna-se difícil manter o cliente na loja virtual por muito tempo.

Em aplicações de realidade virtual o usuário não é relegado ao papel de mero observador passivo, mas torna-se um "participante" efetivo, um membro ativo ambiente virtual, com o qual pode interagir e obter respostas diferenciadas, em tempo real, conforme suas ações no ambiente.

1.1. Realidade Virtual

Realidade virtual (RV) pode ser definida como sendo a forma mais avançada de interface entre o usuário e o computador até agora disponível, com aplicações na maioria das áreas do conhecimento [8, 9].

A Realidade Virtual é comumente classificada em Imersiva e Não-imersiva. A primeira utiliza dispositivos não convencionais, como capacete de visualização ou luvas especiais para interagir como ambiente virtual. Na segunda são utilizados dispositivos convencionais, como mouse, teclado e monitor de vídeo.

Apesar de proporcionar experiências distintas, para [9], o importante é que haja por parte do usuário a impressão de estar atuando dentro do ambiente virtual, apontando, pegando, manipulando e executando outras ações sobre os objetos virtuais, em tempo-real.

Esta pesquisa trata especificamente de Realidade

virtual, apontando, pegando, manipulando e executando outras ações sobre os objetos virtuais, em tempo-real.

Esta pesquisa trata especificamente de Realidade Virtual não-imersiva, devido, principalmente, ao fato de que a maior parte da audiência de aplicações de e-commerce está concentrada em usuários domésticos. Por essa razão, a utilização de hardware especial não pode, a princípio, ser um requisito para a utilização do ambiente virtual.

1.2. Comércio Eletrônico

O termo Comércio Eletrônico, ou e-commerce designa o comércio de bens e serviços negociados por meio da internet e está inserido em um contexto maior, designado e-business, o qual contempla todos tipos de negócios realizados por meio da internet. Assim, o e-commerce é considerado uma das ferramentas do e-business. Em países como os Estados Unidos, em que a Internet se iniciou mais cedo, a explosão do e-commerce é mais evidenciada [1].

O Brasil ocupa uma posição importante no ranking dos 10 países com maior número de usuários de internet [1], o que demonstra o potencial do país para a exploração do e-commerce.

2. Trabalhos relacionados

Estudos específicos sobre a aplicação de Realidade Virtual Não-Imersiva em e-commerce foram realizados por vários pesquisadores, dentre os quais, destacam-se:

- Mass (1999) [5], *VRCommerce — electronic commerce in virtual reality*: Apresenta uma proposta para a construção de shoppings virtuais 3D. Cada loja é carregada e disponibilizada à medida que o usuário navega pelo shopping.
- Chittaro (2000) [3], *Adding Adaptive Features to Virtual Reality Interfaces for E-commerce*: Propõe a construção de interfaces personalizadas para cada consumidor, baseando-se na idade, sexo, itens visualizados na loja e itens comprados. A partir destas informações são mapeados os possíveis interesses futuros do consumidor para criar uma interface adaptada ao seu estilo de compra.

Tabela 1. Principais características dos trabalhados avaliados

Trabalho	Tecnologia		Visualização		
	RV	Apoio	Ambiente	Produtos	Compra
Mass (1999)	VRML	Java	3D	3D	3D
Chittaro (2000)	VRML	Java	3D	3D	n/a
Chittaro (2002)	VRML	Java	3D	3D	n/a
Nguyen (2003)	WildTangent	Java	3D	3D	3D
Esta pesquisa	**VRML**	**PHP**	**3D**	**Fotografia**	**2D**

3. Arquitetura

3.1. Tecnologias de Apoio

Para a escolha das tecnologias a serem usada no desenvolvimento do protótipo desta pesquisa, foram observados os seguintes critérios:

a) Devido à natureza acadêmica e experimental desta pesquisa, as tecnologias utilizadas deveriam: ser de distribuição gratuita, de código aberto ou prover uma versão para uso não-comercial; possuir ampla referência em pesquisas acadêmicas; ser independente de plataforma.

b) Devido ao fato de tratar-se de um protótipo para e-commerce, o conjunto de tecnologias deveria: possibilitar obter dinamicamente informações de um banco de dados em tempo real; oferecer recursos para que a o ambiente virtual possa ser visualizado em browsers padrão de mercado; a exigência de largura de banda de conexão com a Internet não deveria ser restritiva para usuários residenciais.

A partir destes requisitos, as tecnologias escolhidas para a construção do protótipo foram:

- HTML para apresentação do website [2].
- Linguagem PHP para os scripts server-side [7].
- Banco de dados MySQL[11].
- VRML como tecnologia de Realidade virtual.

3.2. Arquitetura do Sistema

O propósito deste trabalho é a elaboração de uma arquitetura para a construção de lojas virtuais baseadas na interação entre o usuário e ambientes 3D e 2D integrados de forma transparente ao usuário, conforme ilustra a Fig. 1.

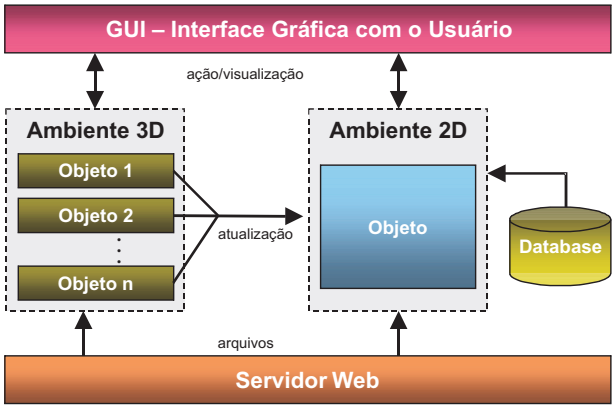


Figura 1. Arquitetura proposta para o sistema.

Resumidamente, destacam-se os seguintes pontos-chave:

- A interface gráfica com o usuário (GUI) é provida por um web browser padrão de mercado.
- A visualização e interação do usuário com os ambientes 2D e 3D ocorrerá de forma integrada e simultânea.
- Os ambientes 2D e 3D não existem isoladamente, pois cada um presume a existência do outro.
- A navegação no ambiente 3D é realizada utilizando-se o mouse e o teclado para se locomover e selecionar objetos. A navegação no ambiente 2D se dá como em um website de comércio eletrônico convencional, uma vez que é baseada em HTML.
- O ambiente 2D é atualizado dinamicamente, apresentando com os dados de um único objeto de cada vez, conforme a seleção do usuário no ambiente 3D.
- Todas as informações sobre o objeto selecionado são carregadas de um banco de dados em tempo real.
- Um ou mais servidores web devem prover as solicitações de arquivos para ambos os ambientes.

4. Funcionamento

Assim que o usuário navega até a URL da loja, o *web browser* carrega um frameset HTML, o qual contém as instruções de composição dos três frames principais do protótipo: "topo", "VRML" e "produto".

O *frame* "topo" contém a página "topo.html", na qual são exibidos o logotipo da loja e demais links de um site de e-commerce convencional.

O *frame* "VRML" carrega a página "vrml.html", a qual contém as instruções de carregamento do plugin e do código VRML do ambiente 3D, cujo carregamento se dará automaticamente.

No *frame* "produto" é carregada a página "produto.php", cujo conteúdo é construído dinamicamente, a medida que ocorre a interação do usuário com o ambiente 3D. Ao clicar em um objeto no cenário 3D, o mecanismo VRML se encarregará de instruir o navegador a carregar a página PHP correspondente ao produto selecionado. Os dados sobre o produto são obtidos de um banco de dados MySQL.

A Figura 2 representa o modelo de implementação do website do protótipo.

Toda a interação do usuário com o ambiente virtual se dá por meio do navegador web, no qual estão visíveis

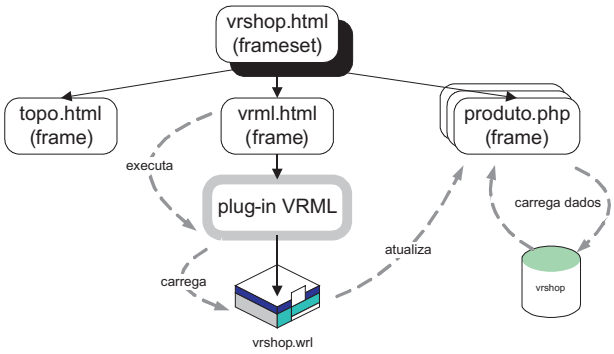


Fig. 2. Modelo de implementação do website do protótipo.

os ambientes 3D e 2D da loja virtual. A visualização de ambos os ambientes se dá simultaneamente, compartilhando a área de visualização do navegador (Figura 3).

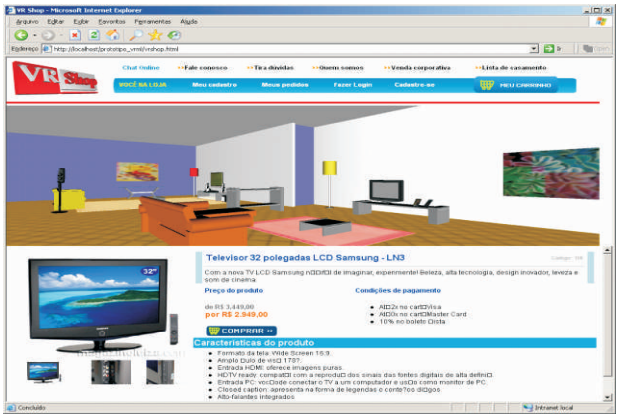


Fig. 3. Visualização dos ambientes 3D e 2D após o usuário clicar no modelo 3D do televisor no ambiente virtual.

5. Conclusões e trabalhos futuros

O protótipo resultante desta pesquisa demonstrou-se eficaz considerando-se o objetivo para o qual foi criado. Tanto os aspectos visuais quanto funcionais do protótipo foram considerados satisfatórios na medida em que satisfazem os requisitos de integração dos ambientes 3D e 2D em uma única interface, de modo transparente ao usuário.

A interface 3D mostrou-se atrativa o suficiente para despertar o interesse do usuário e prender sua atenção durante todo o processo de compra.

A utilização do módulo 2D para a continuidade do processo de compra, baseado no modelo de lojas convencionais de comércio eletrônico mostrou-se importante por ser familiar ao usuário de Internet que já utiliza lojas virtuais convencionais.

Entende-se que a implementação do conjunto

VRML/PHP é uma das maiores contribuições do trabalho, pois não foram encontradas referências a este modelo de arquitetura nas pesquisas realizadas.

Os seguintes trabalhos futuros são pretendidos como continuidade desta pesquisa:

- Implementar na arquitetura um mecanismo de interação que permita ao usuário pesquisar itens no ambiente 2D e ser levado ao respectivo produto no ambiente 3D.
- Elaborar arquitetura que possibilite a montagem de ambientes personalizáveis pelo próprio usuário a partir dos objetos disponíveis na loja.

Também se pretende submeter o protótipo à apreciação de um grupo usuários do sistema de e-commerce convencional, visando comprovar sua eficácia como tecnologia de suporte ao processo de compras pela Internet.

6. Referências

[1] CAMARA-E.NET. *Câmara Brasileira de Comércio Eletrônico*. Disponível em <http://www.camara-e.net>. Acesso em abr. 2007.

[2] CAMPOS, Leandro S. (2004). *HTML Rápido e Prático*. Goiânia: Terra, 2004. 152p.

[3] CHITTARO, Luca; RANON, Roberto. (2000). *Adding Adaptive Features to Virtual Reality Interfaces for E-Commerce*. In: Proceedings of AH2000: International Conference on Adaptive Hypermidia and Adaptive Web-based Systems, Lecture Notes in Computer Science 1892, Sprinces-Verlag, Berlin, 2000, p.85-96.

[4] CHITTARO, Luca; RANON, Roberto. (2002). *New Directions for the Design of Virtual Reality Interfaces to E-Commerce Sites*. In: AVI 2002: 5th International Conference on Advanced Visual Interfaces, ACM Press. p.308-305.

[5] MASS, Yosi; HERZBERG, Amir (1999). *VRCommerce - Electronic Commerce in Virtual Reality*. In: ACM Conference on Electronic Commerce 1999: 103-109.

[6] NGUYEN, Thanh Huu. (2003). *Virtual Reality for E-Commerce. Building as Interactive 3D Online Storefront*. Dissertação - Faculty of Art, Design and Communication - RMIT University, Melbourne Australia, 2003. 86f.

[7] PHP Usage (2007). *Usage Stats for PHP*. Disponível em <http://www.php.net/usage.php>

Acesso em set. 2007.[1] [8] RIBEIRO, Marcos Wagner de Souza. (2006). *Arquitetura para Distribuição de Ambientes Virtuais Multidisciplinares.* Tese (Doutorado em Ciências) – Faculdade de Engenharia Elétrica – UFU, Uberlândia, 2006. 176f.

[9] TORI, Romero; KIRNER, Cláudio (2006). *Fundamentos de Realidade Virtual*. In: TORI, Romero; KIRNER, Cláudio; SISCOUTO, Robson (Ed.). Fundamentos e Tecnologia de Realidade Virtual e Aumentada. Livro do pré-simpósio SVR 2006. Belém: Editora SBC, 2006, cap.1-2, p.2-38.

[10] TROYER, Olga de. et al. (2006). *Developing Semantic VR-shops for e-commerce*. Vrije Universiteit Brussel - Belgium, 2006. 40p.

[11] WELLING, Luke; THOMSON, Laura (2002). *PHP e MySQL – Desenvolvimento Web*. Rio de janeiro: Campus, 2003. 676p.

Vendas no e-commerce por meio de uma aplicação utilizando Realidade Aumentada para a visualização de acessórios da linha de vestuário

Diego Fernandes Medeiros
ILES/ULBRA Itumbiara-GO
Grupo de Realidade Virtual de Goiás
diegonline_si@yahoo.com.br

Hugo Xavier Rocha
ILES/ULBRA Itumbiara-GO
Grupo de Realidade Virtual de Goiás
hxrocha@hotmail.com

Alexandre Cardoso
Faculdade de Engenharia Elétrica
UFU – Uberlândia-MG
alexandre@gmail.com

Wender Antônio da Silva
ILES/ULBRA Itumbiara-GO
Grupo de Realidade Virtual de Goiás
wender_silva@yahoo.com.br

Jucélio Costa de Araújo
ILES/ULBRA Itumbiara-GO
Grupo de Realidade Virtual de Goiás
juccaraujo@yahoo.com.br

Edgard A. Lamounier Júnior
Faculdade de Engenharia Elétrica
UFU – Uberlândia-MG
lamounier@gmail.com

Marcos Wagner S. Ribeiro
ILES/ULBRA Itumbiara-GO
Grupo de Realidade Virtual de Goiás
marcos_wagner@yahoo.com.br

Roger Amandio Luz
ILES/ULBRA Itumbiara-GO
Grupo de Realidade Virtual de Goiás
luzroger@gmail.com

Abstract

The present article constitutes objective to ahead determine one better boarding in e-commerce of the visualization of accessories of the clothes line. Presenting, in this way, a system of augmented reality with implantation to the transactions of the electronic commerce, in order to corroborate with the satisfaction of consumers being determined the visualization of same next to itens or the products chosen, thus materialize a more attractive way for such processes.

Resumo

O presente artigo constitui objetivo de determinar uma melhor abordagem no e-commerce diante da visualização de acessórios da linha de vestuário. Apresentando, deste modo, um sistema de realidade aumentada com implantação às transações do comércio eletrônico, a fim de corroborar com a satisfação de consumidores determinando a visualização dos mesmos junto aos itens ou produtos escolhido, concretizando assim um modo mais atrativo para tais processos.

1. Introdução

O e-commerce dispõe de uma enorme gama de produtos que podem ser postos a venda, dentre estes, especificamente para os itens da linha de vestuário, identifica-se que, para a compra dos mesmos alguns fatores a fim de garantir melhor satisfação do cliente não são abordados e devem ser ressaltados. Nas transações do comércio eletrônico é instituída somente a opção de visualização da foto. Por ser um meio de vendas on-line e pelo fato da presença não física neste tipo de comercialização, o consumidor não possui a oportunidade de experimentar-se com o item desejado. Assim, este trabalho aplica a utilização da Realidade Aumentada (RA) que se agrega dentro do contexto de Realidade Virtual (RV) enfatizando tal oportunidade.

Diferentemente da realidade virtual, que transporta o usuário para o ambiente virtual, a realidade aumentada mantém o usuário no seu ambiente físico e transporta o ambiente virtual para o espaço do usuário, permitindo a interação com o mundo virtual, de maneira mais natural e sem necessidades de treinamento ou adaptação [1].

1.1.Realidade Aumentada

A Realidade Aumentada é o enriquecimento do ambiente real com os objetos virtuais, usando algum dispositivo tecnológico, funcionando em tempo real [6].

A Realidade Aumentada é uma particularização de realidade misturada, quando o ambiente principal é real ou há predominância do real [1].

2. Trabalhos Relacionados

Diante de pesquisas, vários trabalhos realizados implantam a aplicação da Realidade Aumentada no comércio em geral. O artigo *"Usando Realidade Virtual e Aumentada na Visualização da Simulação de Sistemas de Automação industrial"* [2] demonstra a utilização de sistemas de RV e RA na simulação de sistemas de

automação industrial, como meio para treinamento e melhor aprendizagem.

Figura 1. Visualização da simulação utilizando RV

O trabalho intitulado *"Visualização auxiliada por Realidade Aumentada voltada para o acompanhamento de construções arquitetônicas"* [3] enfatiza a criação de uma interface para a interação onde seja possível comparar e avaliar durante a execução no próprio canteiro de obras por meio da tecnologia de RA construções arquitetônicas, sendo utilizado um tablet PC. Já o artigo *"Realidade Virtual em aplicações do comércio eletrônico"* [4], enfatiza a idéia do uso da RV as transações do comércio eletrônico, demonstrando que a introdução de um novo paradigma de interfaces de comunicação com o usuário pode tornar mais atrativa o ato da compra.

3. O ambiente histórico

A finalidade do trabalho é "esboçar" um sistema de Realidade Aumentada nas transações do comércio eletrônico (e-commerce) no deverá ser implantado na internet. Sendo assim a arquitetura do sistema proposto pode ser analisada conforme figura 2.

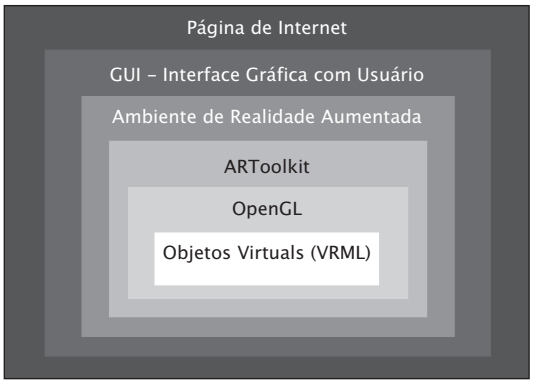


Figura 2. Arquitetura do sistema proposto.

O sistema é abordado dentro de uma tecnologia estabelecida na internet "comércio eletrônico";

• O sistema engloba um ambiente de Realidade Aumentada do qual apresentará objetos virtuais na cena real;

• Os objetos virtuais foram modelados na ferramenta Flux Studio e exportados para ambientes de formato VRML;

• O ambiente de Realidade Aumentada utiliza o recurso de software ARToolKit afim de determinar a posição e orientação da câmera com relação a marcadores.

• O usuário navega na página e desfruta do ambiente de Realidade Aumentada tendo como apoio as instruções para utilização de tal oportunidade, ou seja, é determinada uma interface agradável que auxilia o usuário para uso do sistema;

• O usuário interage com os objetos virtuais utilizando marcadores.

3.1 Sistema de Realidade Aumentada

O sistema de Realidade Aumentada desenvolvido foi o sistema de visão por vídeo baseado em monitor, neste a imagem é capturada pela webcam que por sua vez transmite ao monitor a cena real com a agregação de objetos virtuais.

Para um sistema de Realidade Aumentada faz-se necessário o uso do software ARToolKit ou similar (sabe-se que existem vários softwares de projeção de Realidade Aumentada, optou-se pelo ARToolKit por ser uma biblioteca gratuita e de código aberto), sendo utilizado neste estudo a versão 2.72.1. Assim, destaca-se que o ARToolkit usa técnicas de visão computacional para calcular a posição no espaço real da câmera e sua orientação em relação aos cartões marcadores, permitindo ao programador sobrepor objetos virtuais aos cartões [5], conforme demonstrado na figura 3.

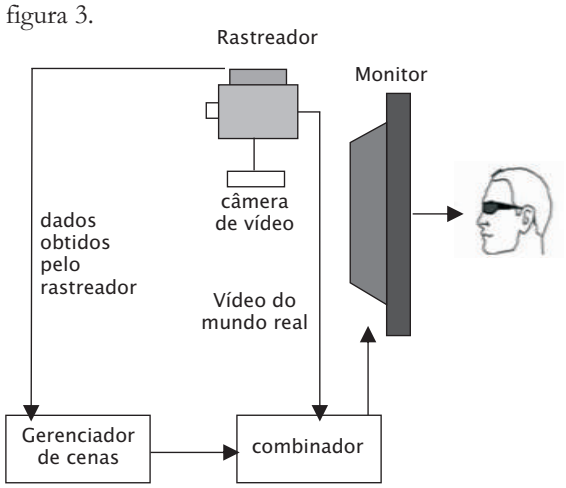


Figura 3. Sistema de Visão baseado em Monitor [5].

3.2 Página de Internet

Para aplicar a idéia abordada faz-se necessário a simulação de uma transação do comércio eletrônico, ou seja, foi desenvolvida uma página de internet da qual destaca a venda de alguns acessórios da linha de vestuário. Como se trata de um software protótipo, os produtos postos para visualização foram limitados, é

interessante ressaltar que o protótipo é um conceito.

4.Implementação do Sistema

Para o desenvolvimento do protótipo apresentado neste trabalho, foram escolhidos três itens que se adequam ao que o sistema de Realidade Aumentada utilizado pode proporcionar. A figura 4 demonstra o código fonte da modelagem do objeto virtual "Chapéu".

```
#VRML V2.0 utf8
## Vizthumbnail
Thumb_chapeu_wrl650311196258953.jpg
Transform {
 children [
  Shape {
   appearance Appearance {
    material DEF Shiny_Black Material {
     ambientIntensity 0.200
     shininess 0.100
     diffuseColor .07059 .07059 .07059
    }
   }
```

Figura 4. Codificação do software.

Cada objeto modelado pelo software Flux Studio 2.1 aparecerá conforme a apresentação à webcam pelo seu marcador, abstraindo-se assim os conceitos e as implementações do software ARToolkit. Como o sistema de Realidade Aumentada é baseado no ARToolkit versão 2.72.1, necessitou-se que cada objeto virtual: óculos, chapéu e a cartola de extensão *".wrl"* fossem salvos no diretório ARToolKit\bin\wrl. Ainda neste diretório precisou-se também criar os arquivos com extensão *".dat"* que faz associação da placa com o objeto virtual, determinado ainda os seus parâmetros, conforme demonstra a figura 5.

```
chapeu.wrl
0.0 0.0 50.0          # Translation
180.90 90.0 0.0 0.0   # Rotation
1800 1800 1800        # Scale
```

Figura 5. Parâmetros do objeto "Chapéu".

Para que a figura chapéu seja evidenciada no lugar correto foram caracterizados os parâmetros de translação, rotação e escala. Com relação aos marcadores foram feitos seus cadastramentos executando o programa *"mk_patt.exe"* encontrado na pasta *"bin"*. Após o cadastramento, os marcadores que são salvos também na pasta *"bin"* com os respectivos nomes foram transportados para o diretório ARToolKit\bin\Data. Na pasta *"data"* foi alterado o arquivo *"object_data_vrml"* a fim de adequar a quantidade de marcadores e adicionar os parâmetros referentes às novas placas.

Após a conclusão do sistema de Realidade Aumentada, foi iniciada a etapa da elaboração de uma página de internet, tendo nesta, uma tarefa com vários processos.

A finalidade do trabalho visa estabelecer o conceito da visualização dos acessórios estabelecidos no e-commerce por meio da Realidade Aumentada, sendo necessário determinar além da opção de visualização de fotos dos acessórios, um link para a visualização em Realidade Aumentada e para a impressão do marcador, conforme demonstrado na figura 6.

Figura 6. Botões de interação.

Cabe ressaltar que o sistema deverá ser instalado no computador do cliente. Assim, o ARToolkit para download sempre será atualizado conforme a entrada de novos produtos disponíveis para comercialização. Neste caso, para a visualização de novos produtos cadastrados, o consumidor deverá atualizar seu software, ou seja, baixar e instalar atualizações.

Diante do funcionamento do sistema, o consumidor primeiramente navega pelo site tendo várias opções para sua interatividade. Inicialmente são apresentados ao usuário, na página principal do site, os itens para visualização em Realidade Aumentada postos à venda, conforme figura 7.

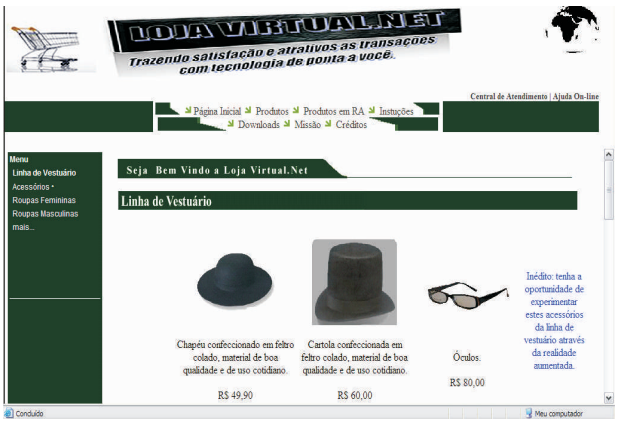


Figura 7. Página Inicial do site.

Assim, ao se clicar em cada um destes itens abrirá uma página que determinará o produto juntamente com as informações e botões das interações.

O próximo passo visa à impressão dos marcadores que são para os produtos: óculos, cartola e chapéu.

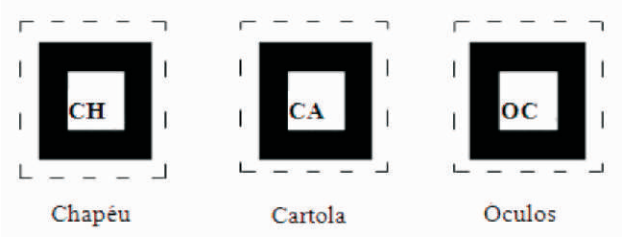


Figura 8. Marcadores de cada objeto.

A medida dos marcadores perante verificação após sua impressão foi de 3 x 3,2 cm. Estes devem ser pequenos, pois serão anexados na testa. Após a verificação e execução do sistema de Realidade Aumentada, observe na figura 9, a visualização do chapéu em Realidade Aumentada, em uma transação do comércio eletrônico, onde o usuário necessita simplesmente anexar o marcador em sua testa e se observar diante do monitor, como se estivesse de frente ao espelho.

Figura 9. Visualização em RA do produto "Chapéu".

5. Conclusões

Conclui-se que a Realidade Virtual e Aumentada são tecnologias das quais se encaixam em várias áreas, gerando extrema eficiência em suas determinadas finalidades. Sendo assim, do mesmo modo que em outros ramos, sua aplicação no contexto do comércio eletrônico englobou pontos importantes sendo fundamental para o possível progresso desta tecnologia disponibilizada na internet, procedendo a vantagens e benefícios às partes envolvidas. Ainda, destaca-se que, este trabalho é um protótipo, e está em fase de conceito, assim, entende-se que o mesmo deverá ser melhor trabalhado e submetido a testes rígidos de operabilidade e de interatividade com os usuários.

Ainda, entende-se que há entraves em relação aos procedimentos de operacionalização, tais como a falta de uma interface mais amigável, a atualização em precisar re-instalar a biblioteca do ARToolkit e a instalação de bibliotecas OpenGL para o funcionamento correto entre outras.

6. Referências

[1] TORY, Romero. KIRNER, Claudio. SISCOUTTO, Robson. Fundamentos e Tecnologia de Realidade Virtual e Aumentada, 2006, *VII Symposium on Virtual Reality*.

[2] BUCCOOLI, A. B. Arthur. ZORZAL, R. Ezequiel. KIRNER, Claudio. Usando Realidade Virtual e Aumentada na Visualização da Simulação de Sistemas de Automação industrial. In: *SVR2006 – VIII Symposium on Virtual Reality*, Belém, 2006.

3] FERNANDES, A. Gabriel. CUNHA, Gerson. Visualização auxiliada por Realidade Aumentada voltada para o acompanhamento de construções arquitetônicas. In: *WRA2006 - III Wokshop de Realidade Aumentada*, Rio de Janeiro, 2006.

[4] KINIZ, Mauricio; KNEASEL, Frank Juergen. *Realidade Virtual em Aplicações do Comércio Eletrônico Dissertação 2003*. Instituto Catarinense de Pós Graduação. Disponível em: <http://www.icpg.com.br/artigos/rev02-14.pdf> Acesso em: 03.08.2007.

[5] CARDOSO, Alexandre; LAMONIER JR, Edgard. Realidade Virtual – Uma Abordagem Prática. *Minicursos – VII Symposium on Virtual Reality* – São Paulo, 2004.

[6] KIRNER, Claudio. SISCOUTTO, Robson. Realidade Virtual e Aumentada: Conceitos, Projetos e Aplicações. *2007, IX Symposium on Virtual and Augmented Reality*.

Realidade Virtual no Apoio ao Ensino de Geometria Descritiva

Marcelle de Sá Guimarães,
Universidade Federal Fluminense
Rio das Ostras – RJ – Brasil
marcelle@ic.uff.br

Karla Bastos Guedes, Ivan Onofre Silva
Universidade Federal Fluminense
Niterói – RJ – Brasil
{karlaguedes}, {ivan_onofre}@vm.uff.br

Stefan Machado Seixas
Universidade Federal Fluminense
Rio das Ostras – RJ – Brasil
stefan_seixas@yahoo.com.br

Hugo Guilherme Andrade da Silva
Universidade Federal Fluminense
Rio das Ostras – RJ – Brasil
hugo_guilherme_rio@yahoo.com.br,

Resumo

Este trabalho apresenta o sistema VirtualGD *que está sendo construído com o objetivo de apoiar o processo de ensino aprendizado de Geometria Descritiva. Com este sistema, baseado em técnicas de realidade virtual, esperamos contribuir para o desenvolvimento da capacidade de visualização espacial dos alunos e, dessa forma, facilitar a compreensão dos conceitos envolvidos na disciplina. Ao mesmo tempo, os professores vão dispor de um recurso didático alternativo e moderno para dar suporte às suas atividades acadêmicas.*

1. Introdução

A geometria descritiva é uma disciplina inicial dos cursos de engenharia e arquitetura, que trata da representação de entes espaciais tais como pontos, retas, planos ou poliedros em quadros de representação bidimensionais (planos), com o objetivo de resolver problemas tridimensionais a partir da geometria plana.

A geometria descritiva é ministrada, em geral, através de aulas expositivas, com uso do quadro negro, onde, com o auxílio de materiais específicos (réguas, esquadros compassos etc.), o professor faz os desenhos que são copiados pelos alunos. Observa-se, no entanto, com essa prática, que um grande número de alunos apresenta sérias dificuldades em relação à capacidade de visualização espacial, sendo esse um dos principais fatores para o insucesso na aprendizagem dessa disciplina.

Dessa forma, percebemos a necessidade de buscar o uso de recursos mais modernos, que possam minimizar esse problema pois, apesar dos crescentes avanços, observados hoje em dia, em relação ao uso de novas tecnologias na educação, percebe-se, ainda, em relação ao ensino de geometria descritiva, uma carência de recursos didáticos alternativos que apóiem o aprendizado da disciplina.

Nos últimos anos, a tecnologia de realidade virtual vem sendo utilizada com sucesso em diferentes áreas do conhecimento, tais como educação, entretenimento, medicina, e outras. Os sistemas educacionais virtuais vêm sendo explorados, principalmente, pelo seu alto poder motivador, especialmente quando existe a possibilidade de interação direta do aluno com os objetos de estudo.

No tocante ao ensino-aprendizado de disciplinas gráficas, a realidade virtual também vem ganhando, gradativamente, mais notoriedade entre os pesquisadores, especialmente no que se refere ao potencial dessa ferramenta para o desenvolvimento de habilidades de visualização espacial.

Entretanto, observamos que os trabalhos publicados nos eventos técnico-científicos da área encontram-se, ainda, ao nível de pesquisas acadêmicas, não estando portanto disponíveis para utilização. Por outro lado, encontramos alguns produtos comerciais disponíveis no mercado, porém de alto custo e nem sempre perfeitamente adequados às nossas necessidades educacionais.

Assim sendo, este trabalho busca suprir essa lacuna, mediante o desenvolvimento de um sistema, utilizando técnicas de realidade virtual, que possa apoiar professores e alunos no ensino aprendizado da geometria descritiva.

1.1. Trabalhos Relacionados

Em [8], os autores apontam o potencial dos sistemas estereoscópicos tridimensionais para o desenvolvimento da capacidade de visualização espacial. Em continuidade a esse estudo, os autores apresentam técnicas de realidade virtual a serem empregadas no projeto de uma ferramenta 3D para desenvolvimento

de habilidades de visualização espacial em cursos de geometria descritiva [7]. Nesse projeto, assim como em [6], é sugerida a utilização de um sistema de visualização estereoscópica utilizando projeções polarizadas, onde os alunos assistiriam às aulas utilizando óculos de polarização passivos. Esses projetos, entretanto, ainda não foram implementados.

Outro pesquisador da área apresenta um sistema tutorial gráfico, disponibilizado na Internet, para o ensino de geometria descritiva [2, 4]. O sistema é composto de conteúdos escritos, acompanhados de animações ilustrativas. O sistema, porém, não é voltado para o uso em sala de aula, e não explora recursos de estereoscopia ou interatividade em tempo real. Um estudo mais recente do autor, vem fazendo uso de técnicas de realidade aumentada para a visualização e aprendizado de superfícies poliédricas [3].

2. Objetivo

A proposta deste trabalho é construir um ambiente virtual tridimensional, envolvendo os conceitos básicos iniciais da geometria descritiva, e que possa apoiar o ensino-aprendizado dessa disciplina.

O sistema seráutilizado em sala de aula, pelo professor, para estimular os processos de visualização espacial do aluno, objetivando uma melhor compreensão dos conceitos envolvidos. Por outro lado, o professor terá a seu dispor uma ferramenta didática alternativa e moderna para apoiar a prática de suas atividades acadêmicas.

3. Fundamentação teórica

A geometria descritiva foi desenvolvida no século XVI pelo matemático francês Gaspar Monge, que propôs a projeção cilíndrica ortogonal de entes espaciais em dois planos de projeção perpendiculares entre si.

Esses planos são então sobrepostos e vistos de frente, sendo representados apenas pela linha que denota a interseção entre eles. Essa representação é chamada de épura (Figura 1).

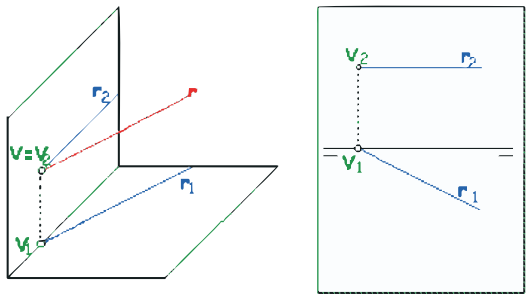


Figura 1. Representação de uma reta horizontal no espaço e na épura.

4. Metodologia

O sistema que estamos propondo, intitulado VirtualGD, será constituído de módulos independentes, contemplando, cada um, conteúdos específicos, que visam cobrir, no total, o estudo do ponto, o estudo da reta e o estudo do plano. A cada aula, o professor poderá dispor dos módulos que julgar mais convenientes.

Os módulos são todos interativos, respondendo às solicitações do usuário em tempo real. Os conceitos são ilustrados e animados de forma gráfica, através de representações tridimensionais. Os objetos de estudo podem ser manipulados, proporcionando uma melhor visualização dos mesmos por diversos ângulos.

O sistema será manipulado pelo professor, projetado em uma tela branca e visualizado pelos alunos mediante o uso de óculos de visão estereoscópica. Um sistema estereoscópico vai contribuir para a obtenção de representações mais realistas, proporcionando maior clareza sobre a profundidade da cena e a posição dos objetos no espaço. Dessa forma estaremos estimulando a capacidade de abstração mental do aluno, e facilitando a assimilação das relações entre os entes espaciais envolvidos.

Corroborando o ponto de vista de [1], a visualização do VirtualGD será feita com o uso de óculos anaglíficos por serem menor custo e de mais fácil implementação. As disciplinas de geometria descritiva são oferecidas, usualmente, para turmas com grande audiência, o que inviabilizaria a utilização de outros sistemas de projeção estereoscópica mais complexos.

O projeto está sendo desenvolvido em X3D [9], através do uso da ferramenta de autoria FluxStudio [5], prevendo uma futura disponibilização do mesmo na Internet.

5. O sistema VirtualGD

A seguir, apresentamos uma pequena ilustração do módulo "retas em posições especiais" do sistema VirtualGD.

O cenário inicial desse módulo apresenta os planos horizontal e vertical de projeção, além de um menu de retas a serem projetadas.

Se o usuário selecionar, por exemplo, "reta frontal", ela aparece no espaço de projeção e a opção "projetar" é disponibilizada (Figura 2).

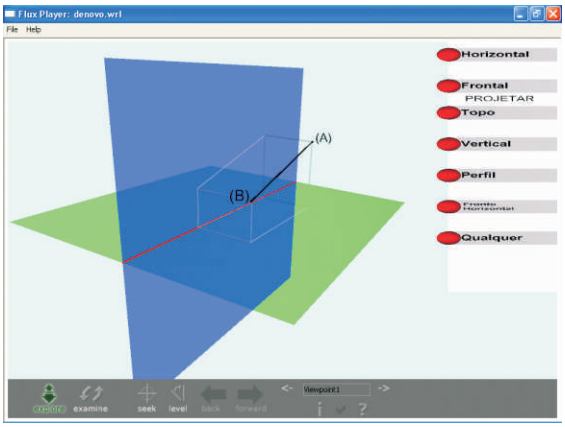


Figura 2. VirtualGD: Representação espacial da reta frontal.

O usuário pode manipular livremente esses planos de projeção para melhor compreender a posição dessa reta no espaço (Figura 3).

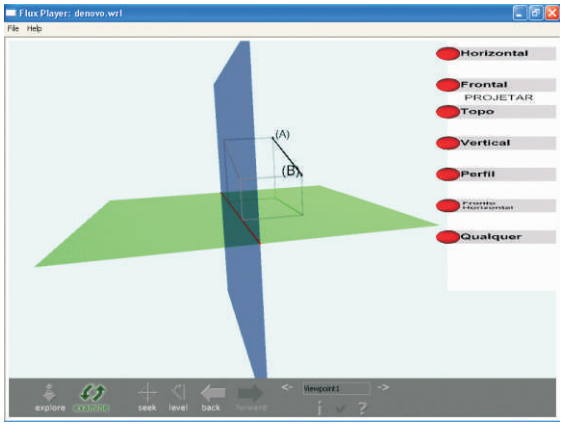


Figura 3. VirtualGD: Objeto de estudo manipulado pelo usuário.

Selecionando a opção "projetar", as projeções horizontal e vertical da reta são apresentadas, e o sistema disponibiliza o botão "épura" (Figura

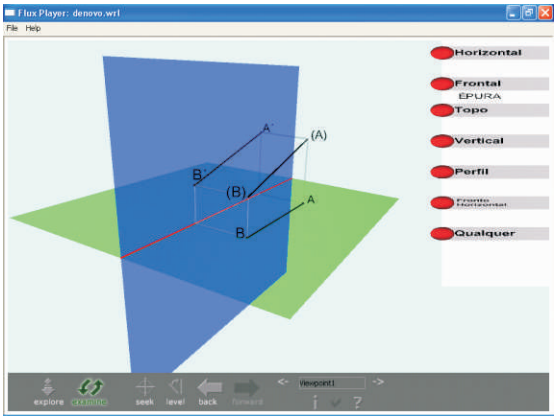


Figura 4. VirtualGD: Projeções horizontal e vertical da reta (A) (B)

Ao escolher "épura", o sistema apaga a reta do espaço e inicia uma animação que ilustra o processo de obtenção da épura (Figura 5).

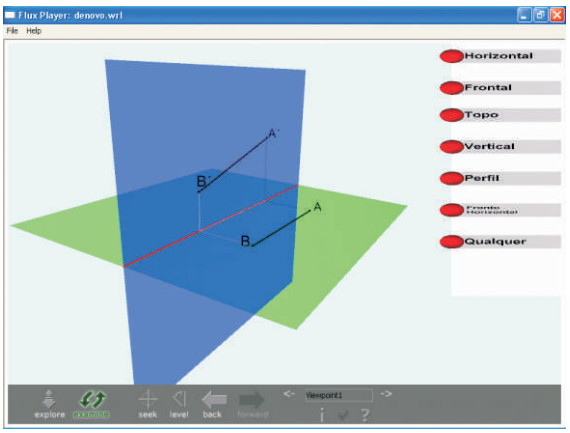


Figura 5 (a).

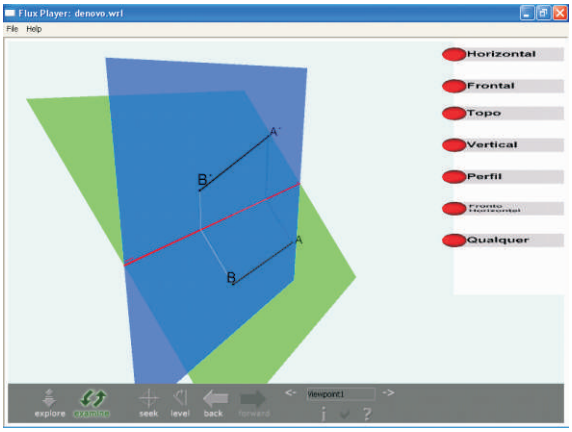


Figura 5 (b).

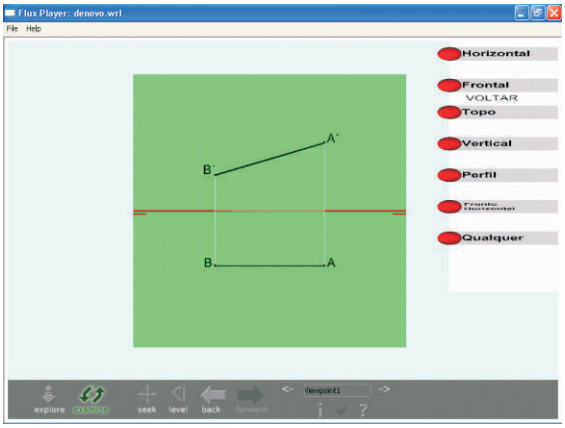


Figura 5 (c).

6. Resultados Iniciais

O sistema encontra-se em fase de desenvolvimento, já tendo sido implementados os módulos de "projeções", "projeções mongeanas", "estudo do ponto" e "retas em

"projeções mongeanas", "estudo do ponto" e "retas em posições especiais".

Inicialmente esses módulos foram testados pelos alunos de geometria descritiva de uma instituição de ensino superior que procuravam os professores e monitores da disciplina para tirar dúvidas ou elucidar conceitos fora da sala de aula.

Os resultados preliminares foram bastante satisfatórios, uma vez que os alunos conseguiram compreender os conceitos apresentados no sistema, afirmaram ter assimilado melhor o posicionamento espacial dos entes envolvidos, e foram capazes de dirimir suas dúvidas, preenchendo as lacunas deixadas pela explanação teórica em sala de aula.

Os monitores e professores, por seu lado, se sentiram à vontade com a utilização dos módulos, que foram considerados amigáveis e intuitivos.

7. Próximos Passos

Os próximos passos deste trabalho compreendem a utilização do sistema em sala de aula, para a devida avaliação dos professores e alunos da disciplina, além da conclusão dos demais módulos propostos.

Posteriormente, estaremos disponibilizando o VirtualGD via Internet, através do sistema de apoio educacional oferecido pela instituição de ensino onde o sistema está sendo desenvolvido. Dessa forma, os módulos poderão ser utilizados não só pelos professores, mas, também, por alunos e monitores, em horários extraclasse, para revisão e aprofundamento dos conteúdos vistos em sala de aula.

8. Referências

[1] L. C. Botega, F. L. S. Nunes e F. R. Montanha, "Implementação de estereoscopia de baixo custo para aplicações em ferramentas de realidade virtual para treinamento médico", *II Jornada do Conhecimento e da Tecnologia*, Marília, 2005.

[2] A. J. R. de Lima, C. J. Haguenauer, G. G. Cunha e L. G. R. de Lima, "Espaço GD – Uma experiência Semipresencial de Ensino de Geometria Descritiva", *VII International Conference on Graphics*, Curitiba, 2007.

[3] A. J. R. de Lima e C. J. Haguenauer, "Visualização das seções cônicas da Geometria Descritivas através de Realidade Aumentada", *Exhibit and Products Demo no IX Symposium on Virtual and Augmented Reality*, Petrópolis, 2007.

[4] A. J. R. de Lima, "Espaço GD", http://acd.ufrj.br/gd/index.htm, consultado em janeiro de 2008.

[5] MEDIAMACHINES, "FluxStudio/FluxPlayer", http://www.mediamachines.com/, consultado em fevereiro de 2008.

[6] E. T. Santos, "Uma Proposta para Uso de Sistemas Estereoscópicos Modernos no Ensino de Geometria Descritiva e Desenho Técnico", *Anais do III Congresso Internacional de Engenharia Gráfica nas artes e no Desenho*, Ouro Preto, 2000.

[7] R. D. Seabra e E. T. Santos, "Utilização de Técnicas de Realidade Virtual no Projeto de uma Ferramenta 3D para Desenvolvimento da Habilidade de Visualização Espacial", *Educação Gráfica*, n. 9, pp. 111-122, Bauru, 2005.

[8] R. D. Seabra e E. T. Santos, "Proposta de Desenvolvimento da Habilidade de Visualização Espacial através de Sistemas Estereoscópicos", *Primer Encuentro Internacional de Professores e Investigadores Del Área de Sistemas de Representación*, Rosário, 2004.

[9] WEB3D CONSORTIUM, "X3D Documentation", http://www.web3d.org/x3d/, consultado em fevereiro de 2008.

Uma Estrutura para a Representação de Ambientes Reais Através de Ambientes Virtuais Dispostos na Internet

Thaíse Kelly de Lima Costa, Liliane dos Santos Machado
LabTEVE - Universidade Federal da Paraíba,
João Pessoa/PB, Brasil
thaisekelly@yahoo.com.br, liliane@di.ufpb.br

Abstract

The representation of real environments through virtual environments allows people to visualize and know distant places. Recently, 2D models have been used as a start point for 3D environments and 2D interfaces have been developed to provide and facilitate the navigation. However, the interfaces are exclusively developed to each application. In this work we present a structure able to associate interfaces and virtual environments that are not dependent of the environment. To validate the structure a case study is presented for a Virtual Campus with access to several services.

1. Introdução

O uso de modelos computacionais tridimensionais permite a visualização de lugares que poderiam ser de difícil acesso para algumas pessoas, devido, por exemplo, a distância geográfica, a dificuldade física de locomoção, ou a dificuldade de localização pela falta de conhecimento do lugar. Estes modelos são conhecidos como maquetes tridimensionais virtuais. A incorporação de técnicas de Realidade Virtual (RV) vem transformar essas "maquetes virtuais" em Ambientes Virtuais (AV) 3D capazes de mapear as ações do visitante e de interagir com ele em tempo real [1].

Recentemente, vem surgindo uma nova proposta que consiste na integração de modelo bidimensional como porta de entrada para os modelos tridimensionais [2]. Este fato possibilita a distribuição contextualizada de Ambientes Virtuais (AV) 3D em espaços bidimensionais.

Os ambientes virtuais podem estar associados a interfaces que buscam de alguma forma auxiliar o usuário no ambiente virtual. Estas interfaces costumam estar atreladas ao AV para o qual foram construídas. Dessa forma, o desenvolvimento de uma estrutura que permita a adição de AVs a uma interface, padronizaria a integração destes ambientes.

Este trabalho visa mostrar uma estrutura capaz de associar uma interface a um ambiente virtual de forma padronizada. Assim, a utilização da estrutura torna a interface independente do ambiente virtual, sendo possível a sua utilização por outros ambientes que se interesse pela forma de apresentação ao usuário. Como estudo de caso, será realizada a associação da interface a um Campus Virtual que tem por objetivo oferecer serviços variados, disponibilizando não apenas um ambiente universitário para visita [3], ou acesso serviços apenas de caráter educacional [4], mas também possibilitar a disponibilização de outros tipos de serviços, como administrativos e sociais.

A fim de uma melhor apresentação, dividiremos o trabalho em seções abordando: estrutura e interface, que esclarece de que forma um ambiente pode ser associado e como é apresentado; modelagem, que expõe o processo de construção do Campus Virtual; e serviços, que mostrará como o campus trabalha a questão da integração de informações.

2. Estrutura e Interface

A idéia da interface é apresentar, inicialmente, um modelo bidimensional como forma de acesso inicial aos ambientes virtuais 3D. Quando o ambiente corrente é um modelo 2D (imagem), a interface apresenta ao usuário um menu de navegação e uma lista de sub-regiões que proverão acesso a seus ambientes tridimensionais. Já quando o ambiente corrente é um AV 3D, a interface apresenta uma imagem detalhada da região e uma lista dos objetos virtuais que se associam a algum tipo de serviço. Este esquema é apresentado na figura 1.

Esta interface é associada aos ambientes seguindo um conjunto de regras que padroniza a ligação, de forma que possa ser utilizada para visualização de outros ambientes virtuais. Para isto, foram criados dois tipos de conjunto de informações: o indexCv3d e o cv3d. O primeiro tipo contém informações para a ligação de um modelo 2D com a interface e atualização desta, já o segundo é responsável pela associação do ambiente 3D à interface, incluindo dados a respeito do ambiente e

dos objetos virtuais associados a serviços.

Figura 1. a) Esquema da interface quando associada a um modelo bidimensional; b): Esquema da interface quando associada a um ambiente virtual 3D.

A estrutura do indexCv3d é apresentada na figura 2. De acordo com a figura, observa-se que na primeira linha são fornecidas três informações: (1) 'nomeAV', que indica o nome do AV associado a imagem; (2) 'imagem', que especifica a fotografia global da região que será representada virtualmente; (3) 'limiteEscala', que representa o zoom máximo para esta imagem, sendo usado para o limite de aproximação na interface.

A partir da segunda linha são apresentadas as informações a respeito das sub-regiões da imagem, sendo estas: (1) 'tag', que indica o nome da sub-região, sendo usada na criação da lista de sub-regiões; (2) 'imagemContorno', que representa uma fotografia de borda utilizada como textura para o destaque do local; (3) 'translação XY', que indica o valor X e Y que será necessário transladar a imagem de contorno para adequá-la a posição da região na imagem original; e (4) 'arquivo', que consiste em um arquivo que poderá ser do tipo cv3d (caso a região proporcione uma ligação a um AV 3D), ou um arquivo de linguagem de modelagem de cenários (caso a região propicie a chamada a uma aplicação, ou página web).

<nomeAV>	imagem	limiteEscala	
<tag1>	imagemContorno1	translacaoXY1	arquivo1
<tag2>	imagemContorno2	translacaoXY2	arquivo2
⋮	⋮	⋮	⋮
<tagN>	imagemContornoN	translacaoXYN	arquivoN

Figura 2. Estrutura do arquivo com extensão indexCv3d.

A estrutura do cv3d é apresentada na figura 3. Observando esta figura tem-se: (1) 'tagGeral', que indica o nome do AV 3D, que é apresentado na interface após o carregamento do ambiente; (2) 'ambiente3D', que representa o arquivo do AV 3D que será carregado pela interface; (3) 'imagemRegiao', que representa uma imagem mais detalhada da região, sendo utilizada para apresentação na interface.

A partir da segunda linha constam informações sobre os objetos da cena que possuem algum tipo de serviço que poderá ser disponibilizado, sendo elas: (1) 'tag', que indica o nome do objeto virtual, sendo usado para criação da lista dos objetos na interface; (2) 'arquivoObjeto', corresponde a um arquivo de linguagem de modelagem de cenário que contém o objeto virtual com sua chamada ao serviço ou ambiente; (3) 'arquivoCv3d', que corresponde ao arquivo do tipo cv3d, necessário apenas quando o objeto virtual está relacionado a outro ambiente 3D, e não diretamente a uma aplicação ou página web.

<tagGeral>	ambiente3D	imagemRegiao
<tag1>	arquivoObjeto1	arquivoCv3d1
<tag2>	arquivoObjeto2	
⋮	⋮	⋮
<tagN>	arquivoObjetoN	arquivoCv3dN

Figura 3. Estrutura do arquivo com extensão cv3d.

No cv3d, os objetos virtuais com disponibilidade de acesso a algum tipo de serviço estão localizados separadamente ao arquivo principal do ambiente tridimensional (como pode ser visualizado na figura acima), a fim de que sejam integrados diretamente a cena principal. Isto se faz necessário para contornar o fato de não ser possível a verificação de eventos em objetos integrados a cena principal de forma indireta (quando os objetos estão incluídos em sub-ambientes que são atribuídos a cena principal por meio do nó "Inline" da linguagem de modelagem de cenários) através da utilização de linguagem externa para acesso a cena X3D.

3. Modelagem

Para a construção de um ambiente virtual 3D, questões a respeito do custo computacional do modelo também

são avaliadas. Sabe-se que tal custo pode influenciar na velocidade de exibição e navegação da cena gráfica e, conseqüentemente, desestimular o usuário quanto ao seu uso. Dessa forma, a criação do Campus Virtual para a UFPB também tenta considerar a relação entre custo e desempenho do ambiente 3D.

A criação do modelo tridimensional é baseada em uma forma hierárquica de organização, de maneira a ajudar na compreensão das dependências existentes entre as entidades que compõem a cena gráfica, como também na redução do custo de armazenamento do modelo através do reaproveitamento destas entidades. Esta forma de organização se assemelha a uma grafo direcionado tendo o modelo final como raiz, que se ramifica até atingir as entidades básicas. No grafo, cada nó pode ter mais de um pai, ou seja, uma entidade pode participar da formação de mais de um objeto evitando cópias desnecessárias na formação da cena gráfica.

As entidades básicas contêm diversos atributos que descrevem sua geometria, aparência e transformações. Estes atributos podem ser trabalhados a fim de ajudar na qualidade dos modelos, sem comprometer fortemente o custo computacional. Por exemplo, ao construir uma parede de tijolos, ao invés de modelar uma geometria composta por vários paralelepípedos, pode-se modelar uma única caixa fazendo as transformações geométricas necessárias e aplicando uma textura de tijolos, obtendo uma entidade com qualidade e de menor custo computacional em relação à primeira.

A construção dos modelos da cena gráfica é realizada através do uso de ferramentas de modelagem tridimensional, neste caso o Blender (http://www.blender.org). Tais ferramentas permitem a formação de objetos básicos que posteriormente são unidos através do uso de uma linguagem de descrição de cenas 3D para a formação do AV completo, sendo utilizada no projeto a linguagem de descrição de cenários X3D (http://www.web3d.org/x3d).

4. Serviços

Além de um modelo tridimensional do campus com possibilidade de passeio, o projeto fornece serviços e informações para seus usuários. Eles são oferecidos por meio da associação de objetos às aplicações. Por exemplo, um objeto "calculadora estatística" pode levar o aluno a uma ferramenta de ensino; um objeto "porta da sala de um professor" pode conduzir a uma página com as informações referentes ao professor. A figura 4 apresenta algumas aplicações que podem ser disponibilizadas pelo ambiente.

A integração dos serviços requer o uso de uma linguagem de programação externa, neste caso Java, que deve ser suportada pela linguagem de modelagem de cenários utilizada (explicada na seção 3.4). A comunicação (linguagem de modelagem de cenários e Java) é realizada através de um protocolo para manipulação do grafo de cena (SAI – *Scene Access Interface*), que permite o relacionamento com as entidades gráficas.

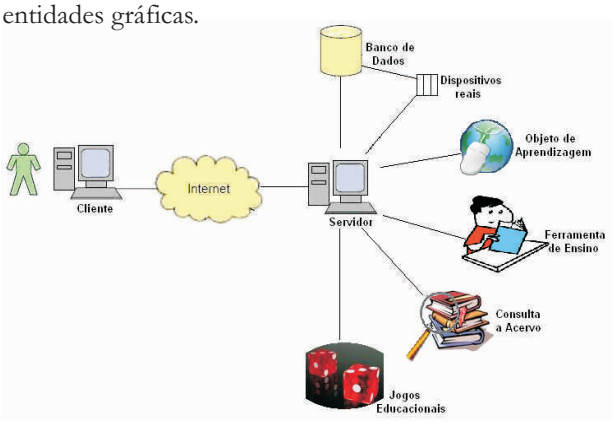


Figura 4. Serviços que podem ser oferecidos com aplicação do estudo de caso do Campus Virtual.

O sistema proposto é baseado em um modelo cliente/servidor. Este modelo é utilizado pois permite a centralização dos dados em máquinas servidoras com possibilidade de acesso por vários clientes. Através deste modelo, as requisições do cliente são enviadas a um servidor que se responsabilizará por responder os pedidos e enviar o cenário virtual solicitado. A figura 5 apresenta o modelo de comunicação.

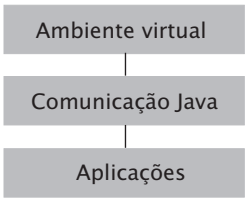


Figura 5. Modelo de comunicação.

O Ambiente Virtual mostrado na figura representa os arquivos do modelo tridimensional do ambiente. O nível de comunicação Java possibilita a associação do modelo às aplicações.

5. Resultados

A estrutura apresentada está sendo utilizada para a exibição de um ambiente virtual para campus universitário que pode disponibilizar acesso a serviços de aspectos educacionais, administrativos e sociais. A figura 6 apresenta um esquema dessa primeira parte. Em (a) é apresentado o AV com a imagem aérea da

universidade e em (b) é mostrado o menu com uma lista indicando as sub-regiões do Campus Virtual; já em (c) é apresentado um exemplo da associação destes dois elementos (imagem + indicador) para auxiliar o usuário a localizar suas necessidades.

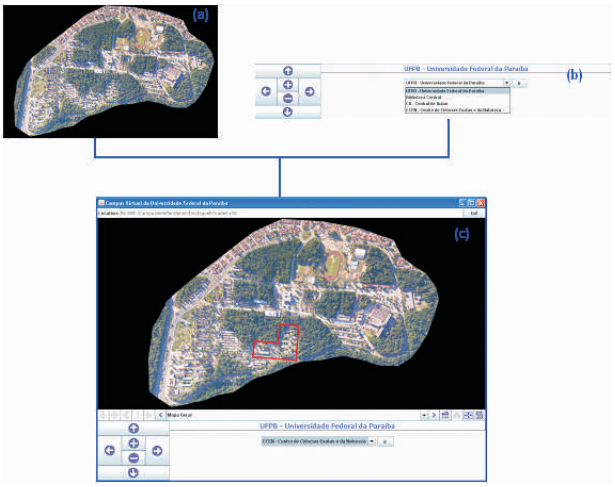


Figura 6. (a) AV com aplicação da imagem aérea; (b) menu com lista das sub-regiões do Campus Virtual; (c) exemplo do funcionamento da interface inicial.

Quando o usuário seleciona a região de interesse e ativa o botão "Ir" o ambiente tridimensional da área é carregado no navegador e, então, o usuário pode passear livremente pelo cenário tridimensional por meio da utilização do mouse. A figura 7 apresenta o esquema da interface durante a apresentação do ambiente 3D de uma região do Campus Virtual da UFPB.

6. Conclusão

Este trabalho apresentou uma estrutura para ser utilizada na criação de uma interface hábil para associação padronizada de ambientes virtuais. Assim este trabalho apresenta como principal contribuição a produção de uma estrutura e interface adaptáveis ao desenvolvimento de AVs e, com a aplicação do estudo de caso, a disponibilização de um Campus Virtual com possibilidades de serviços e informações variadas, expandindo o foco para o qual os Campus Virtuais geralmente são desenvolvidos Como estudo de caso, foi desenvolvido de um ambiente virtual de um campus universitário, com capacidade de agregação a atividades variadas, disposto no modelo de interface sugerido.

Este projeto abre novas oportunidades de pesquisas que contribuirão para o aperfeiçoamento deste Campus, como: produção de atividades colaborativas, a fim de que os estudantes possam aperfeiçoar seus conhecimentos através de tarefas que exijam participação de outros estudantes; construção de um sistema de distribuição consistente, a fim de que todos os clientes possam ter cópias fiéis mesmo após alterações realizadas por outros clientes; e desenvolvimento de um sistema de controle de acesso, a fim de regular a utilização de determinados serviços por pessoas autorizadas. Este controle pode estar ou não associado à estrutura, visto que ele pode ocorrer a partir de uma aplicação Java, ou estar previsto na estrutura, sendo esta inclusão um dos focos de trabalhos futuros.

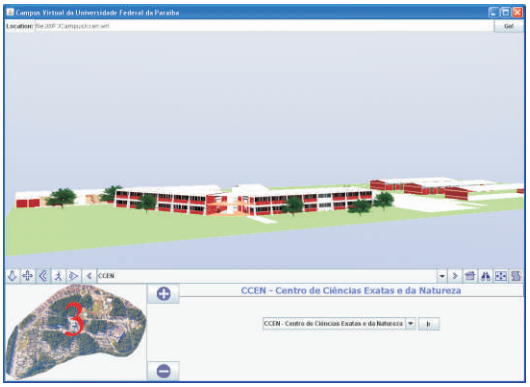


Figura 7. Esquematização da interface durante a apresentação do ambiente tridimensional.

7. Agradecimentos

Agradecemos ao CNPq (processo 485437/2007-4) pelo apoio ao projeto. Agradecemos também a CAPES pela concessão de bolsa à primeira autora do trabalho e ao professor Paulo Rosa, do Departamento de Geociências da Universidade Federal da Paraíba, pela imagem aérea da UFPB utilizada neste trabalho.

8. Referências

[1] Burdea, G.C., Coiffet, P. (2003). "Virtual Reality Technology". 2nd. Edition, John Wiley & Sons, New York.

[2] Brutzman, D. (2006). "Web3D Past, Present and Future: X3D Earth". Disponível em: http://www.web3d2006.org/slides/X3dEarth.BrutzmanKeynoteWeb3dSymposium.2006April19.pdf. Acesso em: fevereiro/2008.

[3] Sarisakal, M. N., Ceylan, K. G. (2003). "A Virtual Reality for Avcilar Campus of Instanbul University using VRML". In: J. of Electrical & Electronics Eng.3(2):977-981.

[4] Sourin, A. (2004). "Nanyang Technological University Virtual Campus". *IEEE CG&A*. Vol. 24(6):6-8.

Session 8: VR Development and X3D

Virtual Presence Mixing 3D Video and Interactive Scenes

Felipe Carvalho, Peter Dam, Luciano Soares, Alberto Raposo, Eduardo Corseuil
Tecgraf – Computer Graphics Technology Group / Department of Computer Science
PUC-Rio – Pontifical Catholic University of Rio de Janeiro, Brazil
{kamel},{peter},{lpsoares},{abraposo},{thadeu}@tecgraf.puc-rio.br

Ismael dos Santos
Petrobras Research Centre - CENPES, Rio de Janeiro, Brazil
ismaelh@petrobras.com.br

Abstract

High quality non-interactive 3D Videos are largely used to document projects, present innovative products and ideas, and to explain complex concepts. On the other hand, in spite of its highly interactive characteristic, virtual reality (VR) suffers due to the non-photorealistic image quality. This research presents an approach for combining 3D rendered videos with interactive scenes. The goal is to overcome the lack of realism in VR models and improve presence and expectation in simulations.

1. Introduction

Professional videos and visualization tools are used in several applications like training, design review among other areas. Our main focus of research is to propose an architecture to organize a visualization process pipeline mixing video and geometric data. It is clear that not always the user wants to navigate in a 3D scene, many times he just wants to animate through an already defined path, and at some places play with some object to test some possibilities.

Videos are a common technology quite useful to document projects, present products and teach science. It is very used in presentation rooms in several companies being simple to deal and easy to anyone playback. High quality VR interactive content allows users to chose different viewpoints and manipulate the environment, but it needs special software and often more advanced user interface devices.

Each one has its advantages and disadvantages [1]. Videos are simple and have high quality images, but follow a linear timeline and lack interaction. Interactive tools, on the other hand, are not photorealistic. If we intend to interact in large models, such as CAD data, for instance, we have to face an even more complex problem since these data sets are quite detailed and take too much processing to render an image in real-time. We propose rendering complex scenes in video and interact just in simpler and located scenes. In order to interact with the scenes, several features were implemented allowing the user to manage the scene, like moving objects and cameras, start audio clips, launch animations, among other possibilities.

The screenplay in our approach will be defined as a graph, due to the possibility of following different paths to reach one or more intermediate stages and endings, constantly changing from interactive scenes to videos, it is even possible to produce a cyclic graph. We are calling this solution as "interactive video" and this term will be used across the document. One big difference in this system is the non-linearity. It is possible to skip among the videos and interactive scenes, letting the user decide what he wants to do next.

This document is organized into the following sections. Section 2 presents related works about the use of videos and interactive scenes. Section 3 shows the ideas of presence and expectation that are the key elements that lead us to start this research. Section 4 explains how the interactive video is produced. Section 5 discusses the main algorithms and techniques applied in the research. Section 6 present the results achieved. Conclusions and future work follow in section 7.

2. Related Work

Some applications are available to couple dynamic content, like animations to interactive scenes. VRML [2] for instance is a common environment to interact in virtual reality scenes, but video visualization is limited, usually achieve by playing a short video in a polygonal surface. 3D multimedia tools like Acrobat 3D [3] and Director [4] allow this mix between different contents. They can be quite useful to document projects but do not cover the visualization of complex models the industry needs, such as massive industry CAD models.

The idea of using 3D movies and interactive scenes is much explored in the game industry. Many of best-selling video games like Gran Tourismo [5], Final Fantasy [6] and Halo [7] use off-line videos with several special effects and complexity together with the interactive game in lower quality to achieve the game

expectative of higher quality and maintain high frame rates. Nevertheless, the application of these resources in industry is not explored. It is possible to find very professional videos and powerful interactive tools to visualize, but they are not really coupled.

Since the industry uses stereo projection system to visualize their 3D models, it is important to cover this technology. Stereoscopic videos or simply 3D movies are not easy to produce and visualize. Some players are stereo capable, but using an external solution will lead to a loosely coupled solution, and then we decided to reuse an internal development capable to playback stereo videos [8]. The stereoscopic interaction in 3D scenes can be achieved in some tools, most of them commercial like SmartPlant Review [10] and EON Reality [11], but the same problem arises: using an external tool will reduce the control over the solution flow. We decided to support interactive scenes across OpenSceneGraph [9], that is free available and supports our basic needs to produce graphical primitives and interaction. Several features were implemented in the scene graph in order to achieve the interactive behaviors desired for this research.

3. Presence and Expectation

Augmenting the involvement during a virtual experience by holding the attention of the users in conscientious or unconscientiously ways can guarantee the success of VR experiences. Presence is closely related to the concept of tele-presence, which term was coined by Marvin Minsky [12].

The present work is under the subjective definitions of presence following the works of Winter and Nunez [13, 14, 15]. Winter pointed two psychological states as the base to feel presence: immersion and engagement [13]. The meaning of immersion is related with the feeling of "being involved". Engagement is resulted by the attention on a set of events and stimulus.

Besides the subjective meaning of immersion in Winter, a rational definition can be noticed in Slater [16]. Here immersion is defined in the sense of something that is measurable and is related with the technology used to create the sensation of presence.

Nunez [14] redefined presence using the concepts of realism and immersion. The amount of user information necessary during the experience is defined as immersion, and the concept of realism is related with the user's expectations. These definitions, pushed the meaning of presence to a different perspective, where an inference process resulted from the information in the environment is also important. The experience here is not only a process of decoding sensorial stimulus, but also something that is being constructed based on expectations. Something is considered realistic if there is a match with a previous expectation. In [15] some experiments were made, following these ideas related with expectation, and positive results were noticed. The experiments aimed the creation of expectations in the subjects using materials before the virtual experience. Better levels of presence were noticed in the group of subjects that read the materials before the experience.

Given the positives results using the idea of creating expectations to conduct a virtual experience and to induce presence, it would be valid to search attractive and rich ways to induce presence during virtual experiences. Photorealistic videos plus 3D interactions would be a different approach from the traditional ones. Using these videos in the sense of creating expectations to better conduct the 3D interaction scenes would increase user's attention.

4. Composing Presentation

The compositing process is divided into two parts, the macro-level and the micro-level. This process consists of a connection of a large variety of elements, to produce the desired scenario flow.

4.1. Macro-Level Visualization

The macro-level contains the play list of videos and interactive scenes and their order to be interpreted by the player. The interactive scenes and videos are presented as nodes in the graph with one entry point and exit points. Nodes are associated to each other by links. A module tool was developed to enable users to connect the nodes as they want (Figure 1) in a simple graphical interface.

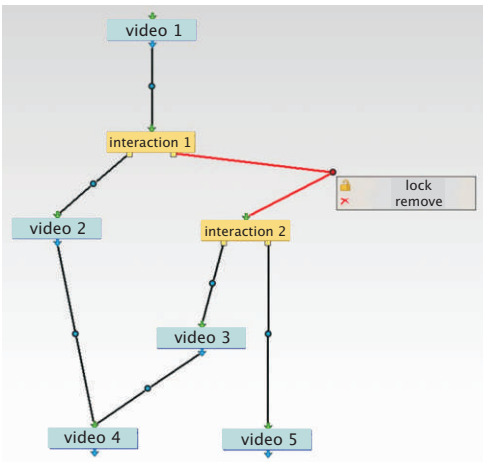


Figure 1. Macro-Level Example Scenario Flow.

This tool allows the interactive video designer to create nodes and to link them according to the desired screenplay. One node must be defined as the starting one. Although more than one ending may exist, we always need to define a starting node.

A video node represents a pre-rendered video. This type of node allows infinite connections into the entry point, but only one connection can exist in the exit point. This means that a video node can only lead to just another node. Note that several videos can be put in sequence, working pretty similar to traditional video edition tools.

An interaction node represents a real-time interaction simulation, thus allowing the spectator to interfere with the content of that scene. An interaction node allows multiple connections into the entry point, as well as multiple exit points connections. The nodes can be organized in such a way that the spectator feels as if he has the power to change the video. The interactions can lead to different exit points, although, must be programmed previously in the micro-level.

The macro-level information can be stored in a XML file. It is possible to define each node configurations, how they are connected, and which one is the starting node. Additional information such as the stereo configuration (mono, anaglyph, active etc), and the links positions are also stored in the macro-level XML file. A macro-level file example is presented in Code 1.

```
<Graph>
  <Stereo ConfigFile=”settings.xml”/>
    <Nodes>
      <Interaction Name=”SC01’StartNode=”true” File=”M1dxm” Pos=”10 10”/>
      <Interaction Name=”SC02’ File=”MV01e.xml” Pos=” 10 20”/>
      <Video Name=”Vid01’ File=”VideoPlayerParams_01xml” Pos=”1030”/>
      <Video Name=”Vid02’ File=”VideoPlayerParams_01xml” Pos=”1040”/>
    </Nodes>
    <Links>
      <Link SourceNode=SC01’ Pipe=”0” TargetNode=”Vid01”Pos=”10 15”/>
      <Link SourceNode=Vid01’ TargetNode=”Sc02”Pos=”10 25”/>
      <Link SourceNode=SC02’ Pipe=”0” TargetNode=”Vid02”Pos=”10 35”/>
    </Links>
</Graph>
```

Code 1. Macro-level example.

4.2. Micro-Level Visualization

The micro-level defines the internal behavior of videos and interactive scenes. Video micro-levels defines the video file to be read, and specify the start time and time length, since we could just playback part of the video and not the whole content of a video file.

Interactive scenes are an event-action approach to describe behaviors. The set of behaviors will be responsible for the interaction flow during the virtual experience. There are several options, and then an editing module was produced for the Autodesk 3DStudio Max, to easily produce interactive micro-level scenes.

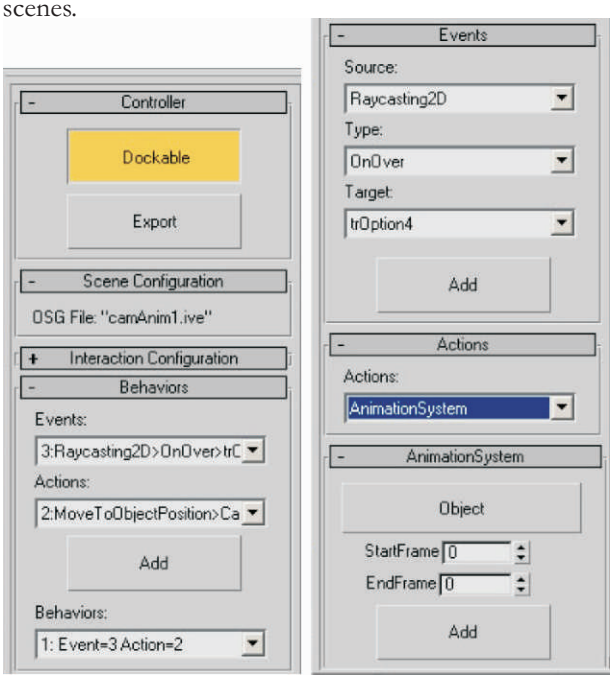


Figure 2. 3DS Max Plugin.

Once our scene is modeled in the 3D modeling application, we must program the behaviors using techniques defined in the system. A scripted plug-in was created for 3DS Max allowing to create events and actions based on pre-programmed techniques and associating these to objects.

With this part programmed, the plug-in exports a micro level XML that holds the configuration and a model file in the OSG IVE file format . The micro-level files are referenced by the macro-level. The micro-level XML is much more extensive than the macro-level, due to the large amount of events and actions. A micro-level file example for an interactive scene is presented in Code 2.

```
<Scene>
 <SceneGraphicNameFile=”./ivefiles/camAnim1ive”/>
 <InteractionTasks>
   <Selection DescriptionFile=”SelectionByRaycasting.xml”/>
 </InteractionTasks>
 <Events>
   <OnOver Id=”1” Source=”Raycasting2D” Target=”trOption1”/>
   <OnClick Id=”2” Source=”Raycasting2D” Target=”plOption4”/>
   <OnStartScene Id=”3” Source=”Scene”/>
   <Finished Id=”4” Source=”MoveToObjectPosition” Target=”CamOP1’/>
   <Finished Id=”5” Source=”AnimationSystem” Target=”trOption4”/>
 </Events>
 <Actions>
   <AnimationSystemId=”1” Object=”trOption1” StartFrame=”10”  EndFrame=”50”/>
   <AudioSystemId=”2” Object=”CurrentView” Audio=”hover2.wav” Loop=”false”/>
   <MoveToObjectPositionId=”3” Source=”CurView” Target=”Cam1” Frames=”100”/>
   <Exit Id=”4” Pipe=”0”/>
 </Actions>
 <Behaviours>
```

```
    <Behaviour ID="1" Event="1" Action="1"/>
    <Behaviour ID="2" Event="3" Action="2"/>
    <Behaviour ID="3" Event="4" Action="3"/>
    <Behaviour ID="4" Event="2" Action="4"/>
    <Behaviour ID="5" Event="5" Action="5"/>
  </Behaviours>
</Scene>
```

Code 2. Micro-level example.

Due the long list of events and actions, we are not going to publish all the possibilities. Here we are just presenting the basic ones. Although, more complex nodes and connections exists, achieving very sophisticated effects.

5. System Architecture

The macro-level is interpreted and used as reference to connect the micro-level nodes. As the start node is detected, the player runs the node interpreter depending whether it is a video or an interactive scene.

Videos are going to be played using the start and length parameters. The interactive scene micro-level is interpreted and a listener mechanism is created to wait for events and trigger the required actions. Devices and interactions techniques are plug-ins of the system. When a new interaction technique is developed it is directly ported to the solution. The interpreter is also a plug-in, responsible to send parameters for the interaction techniques when an action needs to be executed. The interpreter is also responsible to start and finish every plug-in used during the interaction.

The plug-ins exchange messages between them across a message manager that handles the message transport. The messages can carry basic types and pointers to the scene graph nodes (Figure 3).

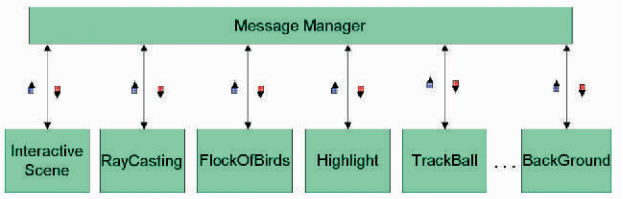


Figure 3. Message Transports.

The message manager is used to control communications of the macro-level (Figure 4). The GManager plug-in start and finish the interactive scenes and the videos. Having all interactive and non-interactive possibilities coupled increases the flexibility of the solution enabling the user, for instance, to cancel a video and skip to the next interactive video node. The structure of these message managers is formed by a message collector which at every frame search for messages by asking all current plugins.by asking all current plugins.

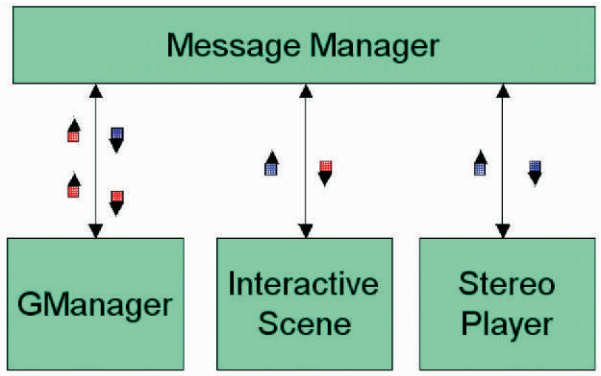


Figure 4. Main Message communication.

The interpreting structure created to handle the interaction flow is composed by a sequence of small trees (Figure 5). Each tree has an event source as root node. On the second level there are types of events that the source emits during the interaction. On the third level there are targets and finally on the last level there are the actions to be executed.

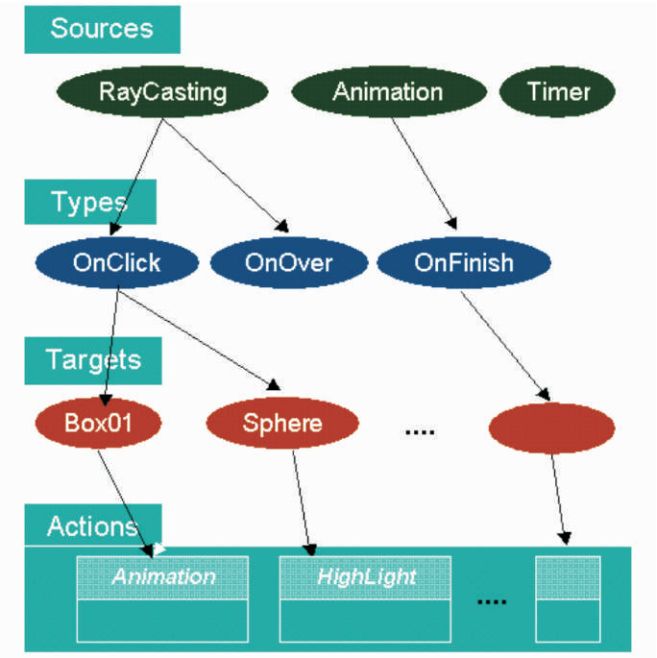


Figure 5. Interactional Flow tree.

The scene description file of the micro level contains 5 basic sections:

1) SceneGraphFile: Contains the OpenSceneGraph file exported by the 3dsmax. In this file there is information about geometry, materials, light, camera, and animation;

2) InteractionTasks: This section acts like an include file directive founded on common programming languages. The included XML files contains

descriptions of interactive tasks using events, actions and behaviors too. Inside the file, that included an interaction task XML file, a label references that task and it is considered as an event source. Example:

```
<Selection
  DescriptionFile="SelectionByRaycasting.xml"/>
```

The label "Selection" is considered as the event source for all the events created inside the file SelectionByRaycasting.xml;

3) Events: Contains the list of events. Each one contains the source where it was created, a type and a target object. Events can be triggered inside the techniques by programming hard code, or can be created inside a XML file being triggered by others events;

4) Actions: Contains the list of actions. Each one has the name of the technique and a list of parameters to be used in the execution of the action;

5) Behaviors: Contains a list of associations between actions and events.

We have created actions and events related with object animation, 3D sound, interpolation of position, path following, object following, object highlight, 2D and 3D text, 3D icons, particles, background image, video stereo player, timers, trackball interaction, treatment of devices such as trackers (optical and a flock of birds), keyboard, mouse, and custom high level interaction tasks (selection, manipulation and navigation) by the association of atomic actions and events.

6. Results

Since we are from the computation area, and not designers, we were initially not completely sure how many interaction resources were important to produce interactive videos. We tested my cases like the one shown in Figure 6 and we defined and implemented a list of features enabling the video designer to produce interactive videos.

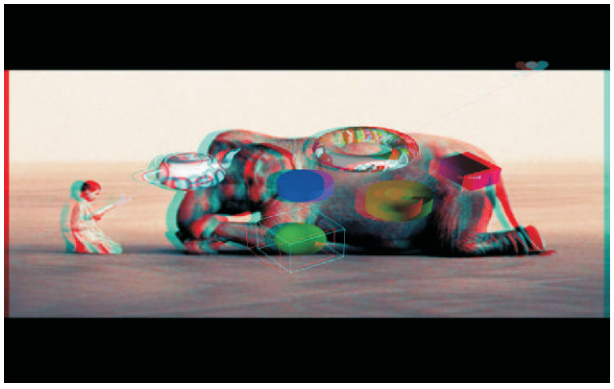


Figure 6. Interactive Video Test Scene.

This solution is being tested in VR facilities. The users evaluated are used to just see high quality videos (Figure 7), and our first observations indicate they get quite impressed when they can change the virtual environment.

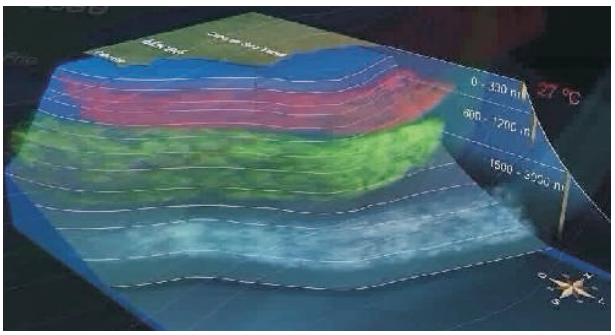


Figure 7. Video Scene.

People producing the interactive scene can see and test all the 3D behaviors inside 3DStudio Max. It is possible because a special player (Figure 8) was developed to assist this creation. This player opens in one of the 3DS views allowing the developer to navigate and test as if he was in an immersive environment.

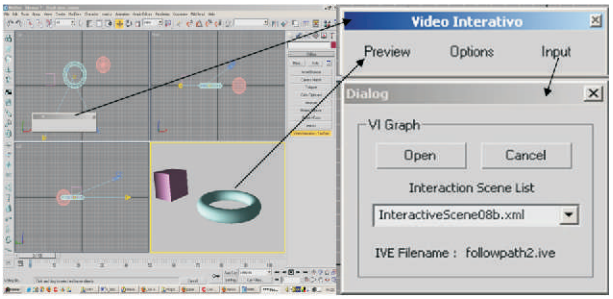


Figure 8. Special Player for 3DS Max.

7. Conclusions

This solution can impact the way people use virtual reality, from videos to real VR applications, increasing presence and not frustrating the audience since the VR quality is not the same as that of professional videos.

Since most of VR facilities in industry are based on planar projections, normal videos can be applied. However, if different projection surfaces are used, such as CAVEs, videos must be produced to fit these needs anyway. For instance, for a six-sided CAVE, it is necessary 6 videos, each one with the correct perspective for each face.

As a future work, we intend to include in the 3D editor dummy icon nodes, to inform the action attached to the scene. We are still studying ways to simplify the micro-level programming for designers, because this might be the toughest part of producing an interactive

video, since it needs a logical sequence to work correctly. We intend to use questionnaires to verify the levels of presence during the virtual experience. This method would be a more formal evaluation.

8. References

[1] J.M. Sullivan, Rescalable Real-Time Interactive Computer Animations Multimedia Tools for Communicating Math., J. Borwein, M.H. Morales, K. Polthier, and J.F. Rodrigues, eds., pp. 311-314, Springer, 2002.

[2] Web3D Consortium, VRML97 Functional specification, ISO/IEC 14772-1:1997. http://www.web3d.org/x3d/specifications/vrml/.

[3] Acrobat 3D, http://www.adobe.com/products/acrobat3d.

[4] Adobe Macromedia Director, http://www.adobe.com/products/director/.

[5] Grand Turismo Play Station game, http://www.us.playstation.com/PSone/Games/Gran_Turismo/OGS/.

[6] Final Fantasy game, http://www.square-enix.com/na/title/ff/.

[7] Halo XBox game http://www.bungie.net/.

[8] Raposo, A.B., Szenberg, F., Gattass, M., Celes. Visão Estereoscópica, Realidade Virtual, Realidade Aumentada e Colaboração. In: A. M. S. Andrade, A. T. Martins, R. J. A. Macêdo (eds.), Anais do XXIV Congresso da Sociedade Brasileira de Computação, Vol. 2, XXIII JAI - Livro Texto, Cap. 7, p.289 - 331. SBC, Brazil, 2004.

[9] Burns, D. and Osfield, R. 2004. Open Scene Graph A: Introduction, B: Examples and Applications. In *Proceedings of the IEEE Virtual Reality 2004 (Vr'04) - Volume 00* (March 27 - 31, 2004). VR. IEEE Computer Society, Washington, DC

[10] Integraph SmartPlant Review, http://www.intergraph.com/smartplant/review/.

[11] Eon Reality, http://www.eonreality.com/.

[12] Minsky, M., *Telepresence*, Omni 45-52, June, 1980.

[13] Witmer, B.G., Singer, M.J., Measuring Presence in Virtual Environments: A Presence Questionnaire, *Presence, v. 7*, n. 3, 1998.

[14] Nunez, D., How is presence in non-immersive, non-realistic virtual environments possible?, *Proceedings of the 3rd International Conference on Computer Graphics, Virtual Reality, Visualization and Interaction in Africa* , 2004.

[15] Nunez, D., Blake, E.H., The thematic baseline technique as means of improving the sensitivity of selfrepor scales, *Presented at Presence 2003*.

EnCIMA: A Graphics Engine for the Development of Multimedia and Virtual Reality Applications

Silvano Maneck Malfatti[1], Selan Rodrigues dos Santos[3], Luciane Machado Fraga[1], Claudia Marcela Justel[2], Jauvane Cavalvante de Oliveira[1]

[1]Laboratório Nacional de Computação Científica - Laboratório ACiMA, Petrópolis, RJ, Brazil
{malfatti}, {lmfraga}, {jauvane}@lncc.br,

[2]Instituto Militar de Engenharia - Seção de Engenharia de Computação SE/8, Rio de Janeiro, RJ, Brazil
cjustel@de9.ime.eb.br

[3]Universidade Federal do Rio Grande do Norte
Departamento de Informática e Matemática Aplicada, Natal, RN, Brazil
selan@dimap.ufrn.br

Abstract

This work presents the features of a flexible realtime 3D graphics engine aimed at the development of multimedia applications and collaborative virtual environments. The engine, called EnCIMA (Engine for Collaborative and Immersive Multimedia Applications), enables a quick development process of applications by providing a high level interface, which has been implemented using the C++ objectoriented programming paradigm. The main features of the proposed engine are the support to scene management, ability to load static and animated 3D models, particle system effects, network connection management to support collaboration, and collision detection. In addition, the engine supports several specialized interaction devices such as 3D mice, haptic devices, 3D motion trackers, data-gloves, and joysticks with and without force feedback. The engine also enables the developer to choose the way the scene should be rendered to, i.e. using standard display devices, stereoscopy, or even several simultaneous projection for spatially immersive devices.

1. Introduction

Virtual reality (VR) applications aims to enable user to experience the sense of presence - casually defined as the feeling of being "in" a virtual environment - which is realized through factors such as immersion, involvement, and interaction [8, 3]. Allowing the user to experience a virtual environment (VE) through simultaneous and coordinated sensory stimuli is the basic mechanism VR uses to afford presence. This is accomplished with special interaction devices (e.g. spatially immersive devices, head/face mounted displays, data gloves, haptic devices, trackers, etc.). Consequently, the process of designing a VR application is not an easy task, specially if the application is built directly on a graphics application programming interface (API). This situation requires the developer to be able to handle all the resources necessary to make the application work as intended.

To make things easier, in the early 90s the concept of "engine" was put forward in the game industry. In general terms, an engine is a middleware, i.e. a group of library functions that abstract most of the low level implementation details by providing the developer with a high level programming interface [6]. These functions are organized in groups with specific functionality such as resource management, networked communication, 2D and 3D rendering, collision detection and response, audio rendering, physics simulation, and scene management [4]. As a result, the developers are able to generate applications that are independent of third-party APIs, provided that the programming is done through the high-level engine's functionality. In addition, this approach supports platform-independent VR applications and speeds up the overall development process.

The success of this idea was responsible for making some VR application in areas like medical simulation, military training, education, art and entertainment, to adopt game engines as their basic development toolkit [13]. However, the use of game engines to develop VR applications has a major limitation: because engines usually supports only traditional interaction devices- i.e mice, keyboards, and joysticks - the type of virtual environment that can be generated is restricted to desktop VR [15].

According to Maia [9], another issue is that the resources of a game engine are targeted at supporting game related features. To obtain a general purpose game engine it would be necessary to re-design it entirely.

For this reason, there have been an effort towards developing engines specialized for VR applications. The work done by Pinho [12] showed that VR engines should provide special features such as support to a wide range of special interaction devices, various 3D model loaders, schemes for management and optimization of 3D graphics environment, multi-platform support, and multi-screen display capability.

This paper presents a VR engine aimed at assisting the development of VR applications, which supports the aforementioned features in addition to the graphics resources available in tools used in game design. Our motivation was to design an engine that would afford a quick development process of applications with various types of interaction devices, and to provide a more engaging environment with animated characters, better lighting, particle systems, terrain modelling, collision detection, etc. It is worth mentioning that our engine also supports specialize display devices such as CAVE-like systems.

This work is organized as follows, Section 2 presents some background on the usual components of an engine architecture. In Section 3 we present a few examples of similar graphics engines, focusing on their features, whereas in Section 4 we delve into the EnCIMA's architecture, exploring some of its features and functions. Section 5 describes two test applications developed with our engine. Finally, in Section 6 we present some conclusions and future work.

2. Graphics Engine Architecture

A graphics engine is a key component in a VR application, being responsible for performing important tasks such as accessing input devices, resource management, updating components of the virtual environment, rendering the 3D scene and presenting the final result through display devices [3].

A traditional way of designing an engine architecture is to organize its functionality in layers. Maia in [7] defined a generic structure for engines, which is composed of three layers, as shown in Figure 1. These layers are organized into modules with a strict hierarchical dependency.

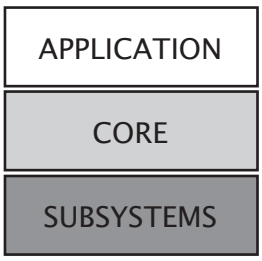


Figure 1. Generic architecture of a game engine.

2.1 The Sub-systems Layer

The sub-systems layer is composed of several modules that offer specific services to the core which, in turn, transforms them in high-level actions for the application. Therefore, every module of this layer is characterized by the application domain it is target for, the tasks assigned to the module, and the technology employed to execute these tasks.

Consider, for instance, the Input module which is responsible for recognizing the interaction devices plugged into the application. The input module's main attribution is to receive user action entered through devices like mouse, keyboard, joystick, data gloves, and position tracking systems. For that purpose, the input module must provide functions that recognize buttons being pushed, requests for updating cursor position, understand tracker orientation and positioning in terms of VE's coordinate system, etc.

For this reason, every module in the sub-systems layer represents and interface between the core and APIs or low level driver that can have direct access to the underlying operational system. Figure 2 shows an example of a subsystems layer composed of three modules with specific responsibilities.

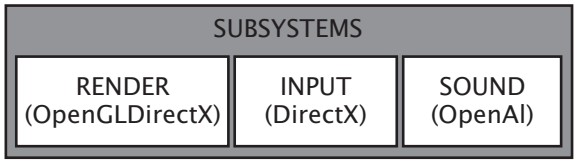


Figure 2. Example of a graphics engine's sub-systems layer.

2.2 The Core Layer

The core is responsible for providing the link between the application layer and the engine's available sub-systems. Johnston [5] defines the core as the engine's central component or server, responsible for invoking the appropriate functions with the right parameters in response to events generated by the user or the environment. Therefore, all modules from the sub-systems layer must register with the core, so that the core is able to initiate each registered module and coordinate the interaction and data exchange among 2 the engine's components.

Similarly to the sub-systems layer, the core is also organized into several modules, each of which having specific responsibilities that are fundamental for the engine's proper functioning. These responsibilities may include access to the local file system, resource management, mathematics core library, activity log, and the specification of the manipulation interface between

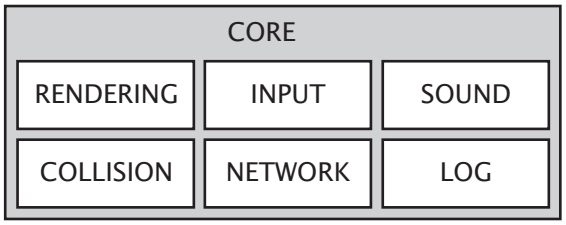


Figure 3. Example of a core layer comprised by six modules.

objects from the VE and the associated devices. Figure 3 presents an example of a core with six modules.

2.3 The Application Layer

The application is the target software that has been built on the functionality offered by the core. It is the application designer's responsibility to specify how the VE is supposed to look like, load and position all 3D models, set up sensors to trigger the appropriated animation and visual effects, as well as to define the VE's response based on either the engine's state machine or user interaction.

The usual strategy employed by VR application developers when they want to create a new VE is to derive through inheritance the engine's classes that offer support to scene resources. In doing so, the developer has access to highlevel methods that encapsulate the functionality available within each of the various core's modules.

3. Related Work

In this section we review three academic VR engines, focusing on their main features and underlining motivation. We wanted to understand the functional aspects of the selected engines and how easily they can be employed to build VR applications.

3.1 enJine

The enJine is an open source graphics engine developed entirely on Java and Java3D API. Consequently, one of the main features is the platform independency and the objectoriented paradigm, both inherent to Java applications [10]. The underlining motivation for enJine was to provide a tool with educational purposes, supporting the teaching of game programming, VR applications, and other computer related concepts such as software engineering and computer graphics.

The VR aspects of the enJine benefited from the Java 3D inherent features which affords, with a few lines of code, access to unconventional display devices such as head/face mounted displays, stereo rendering, and multi-screen projection. Despite the existence of comprehensive documentation and the facility in developing VR application, we believe enJine to have one noticeable limitation: it only supports the so-called traditional interaction devices, i.e. mice, keyboards, and joysticks.

3.2 SmallVR

SmallVR is a toolkit that supports the development of VR applications [12]. The conception of SmallVR was driven by the necessity of an educational tool that could be used in the practical lessons of Virtual Reality modules. SmallVR strongest advantage is to afford a fast development process of VR applications. This is accomplished because SmallVR provides all the basic core functionality, therefore the students do not need to program neither graphics device control nor functions to load 3D object models.

The third-party libraries used by SmallVR architecture are GLUT and OpenGL [14]. As a result, it is possible for the developer to keep utilizing the basic structure of GLUT programs, and still add objects and program actions without having to surrender the application's execution control.

Other features of SmallVR are the support to scene graph, loaders for popular 3D model formats such as Wavefront's OBJ and Autodesk's 3DS, the implementation of rendering acceleration algorithms such as view frustum culling, collision detection, the existence of drivers to support motion tracking devices and head/face mounted displays.

Although SmallVR offers a range of valuable features and proper documentation, the toolkit does not provide other important graphics resources such as support for animated 3D models, shadow casting, particle systems, as well as audio rendering - all these features are left to the developer to program.

3.3 CGE

The CGE, CRAb Graphics Engine, is one of the most complete academic VR engines available. CGE was developed based on the CRAbGE framework, proposed by Maia [7].

CGE is an open source, platform-independent engine whose rendering system is done with the OpenGL API. One of the hallmark features of the CGE is a highly customizable interface, via scripts and plugins. In terms of graphics 3 resources, CGE provides several scene rendering acceleration techniques such as hierarchical frustum culling, levelof-detail (LOD), billboards, and particle systems.

Furthermore, CGE also supports collision detection, spatialized sound, the loading of static and animated

3D models such as 3DS and MD2, and accepts some types of special interaction devices like positioning tracker and dataglove. However, CGE does not handle simultaneous and coordinated renderings for multi-projection spatially immersive devices, like the CAVE system, nor provide support for haptic devices.

4. The EnCIMA Engine

The EnCIMA engine was designed with the purpose of enabling the developer to get his application running as quick and simple as possible, The engine offers a easy to use object-oriented interface designed to minimize the effort required to render 3D scenes, in such a high level that the application becomes independent of third-party 3D graphics rendering API (e.g. Direct3D or OpenGL). For that reason, the developer does not need to have previous or specific knowledge on how to program a given API nor the interaction with special input/output devices.

In terms of graphics resources, EnCIMA provides the loading of both static and animated 3D models, the procedural generation or loading of terrain meshes, environment effects such as skydome, sprites, static and animated billboards, and particle system effects.

The engine also provides high-level functions to control 2D and 3D audio playback, and access to spatialized sound that can be affected by a series of dynamic simulation effects such as Doppler, environment volume, attenuation, and movement of the sound's source location.

Additionally, the engine supports 3D and 2D input devices. Figure 4, for instance, shows a variety of devices handled by the engine: 3D mice, data gloves, tracking systems, joysticks (with or without force feedback), and Sensable's Phantom.

The EnCIMA's architecture (show in Figure 5) follows a multi-layer design that integrates the tradition components mentioned in Section 2: sub-systems, core, and application. In the coming subsections we describe in more details each of the EnCIMA's three layers.

4.1 The Sub-systems Layer

The sub-systems layer integrates well-known APIs, and proprietary drivers that communicate directly with the Windows-based operational system. EnCIMA renders using OpenGL API, whereas the audio is handled by DirectX. The graphics user interface, interaction devices driver, and network connection are all handled by nativeWindows API.

4.2 The Core Layer

The core layer is composed by a set of manager

Figure 4. Example of devices supported by the EnCIMA engine: data glove, 3D mouse, and 3D tracker.

modules. Each manager is defined as a class whose main attribution is to communicate with the sub-systems and offer services to the application layer. Figure 5 presents the seven modules that form the EnCIMA's core layer. The next subsections describe the functionality of all seven modules.

4.2.1 Graphics Manager. The graphics manager is responsible for the allocation of graphics resources, the loading of several image formats (including support to transparency), texturing, and the ability to capture screenshots. The singleton graphics manager is available to all classes that require any image related action, for instance texture loading, heads up displays (HUDs), terrain generation, and particle systems (e.g. fire, smoke, etc.). The graphics manager is an important module because it manages and shares graphics resources, assigning them unique identifiers. All the subsequent requests for the same graphics resource are sent to the graphics manager, which answers back with a reference to the requested resource. This simple approach avoids memory waste and optimize its usage and access. The code shown in Code 1 presents the LoadImage method of the GraphicsManager that is responsible for the image loading process.

The graphics manager employs a reference counting scheme, which decrements the counting when a resource is released and increases it when the resource is requested. When the counting reaches zero the graphics resource is finally freed.

4.2.2 Time Manager. The time manager controls all timers used in animation and special effects. Its main task is to guarantee that timers have a consistent temporal behavior in machines with various processing capacities. The main advantage in centralizing time management is that the low level function that retrieves the system time is called only once every frame and used to update all registered timers.

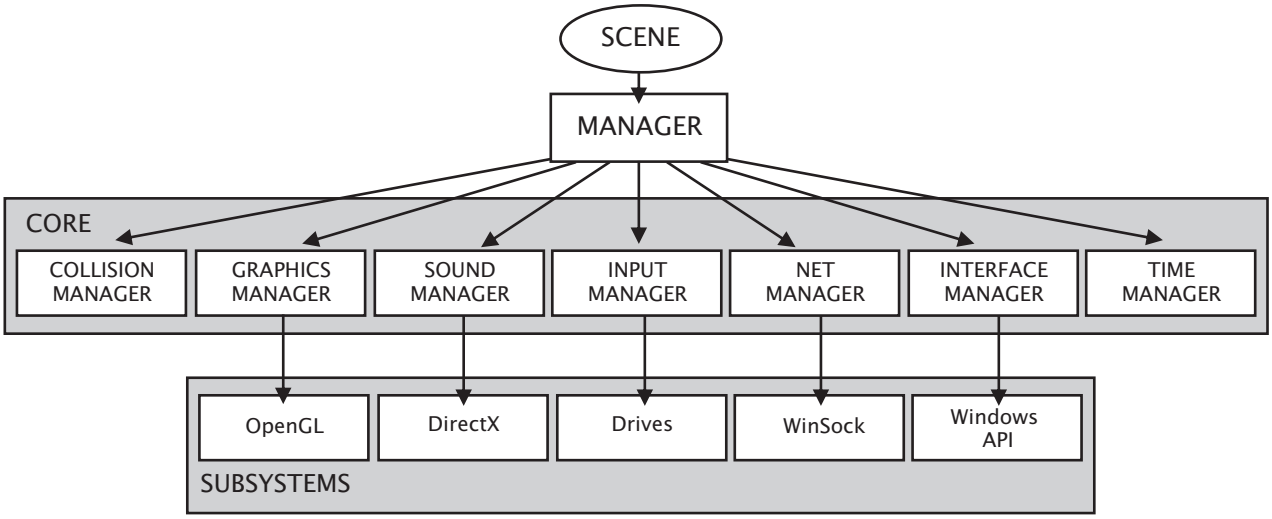


Figure 5: The EnCIMA's architecture.

```
CVRImage*
CVRGraphicsManager::LoadImage ( char* fileName,
                                bool transparency ) {

  // Verify wether the image has already been loaded.
  for (int iIndex=0;index<(int)vImages.size();iIndex++){
    for (strcmp(fileName,vImages[iIndex]->GetImageName())
           == 0) {
         vImages[ iIndex ]->iReferences++;
         return vImages[ iIndex ];
    }
  }

  // Not loaded yet, thus we create a new one.
  CVRImage* pNewImage - new CVRImage;

  if ( pNewImage ) {
      pNewImage->LoadImage( fileName, transparency );
      if (pNewImage->GetImageData() ) {
          vImages.push_back( pNewImage );
          return pNewImage
      }
  }

  // Image not found.
  delete pNewImage;
  return NULL;
}
```

Code 1: GraphicsManager's method to load images.

```
void SceneTest::Init() {

   // Creates a 2D bitmap font.
   fAngle = 0.0f;
   Font   - CreateFont2D( "EnCIMA", 30, true);
   Font->SetColor( 1.f, 0.f, 0.f);
   Font->SetPosX( 50.f);
   Font->SetPosY( 50.f );

   //Initialize the tracker.
   pManager->cInputManager.cTracker.Init();

   //Get the angle with Y axis from sensor 1.
   fAngle=pManager->cInputManager.cTracker.GetAngleY(1);
}
```

Code 2: Establishing communication with 3D positioning tracker through the input manager.

4.2.3 Input Manager. The input manager communicates with every input device supported by the engine. This module detects, initiates, and uniquely identifies all input devices plugged into the application. The Code 2 snippet shows the usage of the input manager to initiate and obtain the angles from one of the 3D positioning tracker's sensors.

4.2.4 Sound Manager. The sound manager coordinates the loading of 2D and 3D sounds, allowing the application to set up various sound parameters such as volume, 3D position, area of influence, and attenuation.

4.2.5 Network Manager. The net manager is responsible for establishing network connections between servers and clients, for collaborative applications. The data exchange follows a multicast-based client-server model. This means that every server is responsible for both managing groups of client applications and delivering modifications to all the participants of a given group.

4.2.6 Interface Manager. The interface manager is responsible for handling every detail related to the display device utilized by the application. For traditional displays, it is possible to set window size, position, background color, stereoscopic rendering, graphics resolution, and fullscreen mode.

4.2.7 Collision Manager. Collision detection avoids virtual objects to penetrate one another. A pre-stage in collision detection is to identify the moment that objects get close enough to be considered for collision tests [16]. The collision manager is responsible for identifying potential colliding objects and perform intersection tests for collision determination.

EnCIMA applies the sphere-trees technique to perform ollision detection. This technique approximates the geometry of objects with an hierarchy of spheres. Every object in a scene is involved in a sphere, therefore sphere intersection tests are done to check for collision among objects. The advantage of working through an hierarchy of spheres is that an entire sub-tree (of spheres) can be readily eliminated from the collision detection process by simply testing if the root sphere (the one that bounds all its children) does not intersect with the target object's sphere.

The sphere primitive is often the preferable choice of bounding volume, the reasons being: i) it is procedurally and mathematically simple to construct a sphere, ii) spheres are invariant under rotations, and iii) the calculation of intersection between spheres is simple an computationally inexpensive.

We have employed the Sphere-Tree Construction Toolkit [2] to generate the sphere-tree of the objects in a scene. The sphere-tree generation is done in a pre-processing step, in which the created trees are stored in files. Later, the engine loads and positions them on the corresponding object's surface. Figure 6 illustrates an example of the sphere-tree corresponding to a scalpel model.

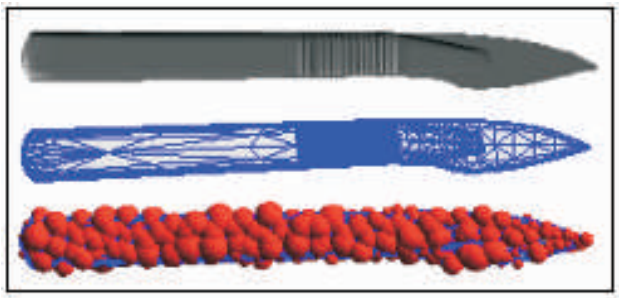


Figure 6. Sphere-tree of a scalpel model, generated with the Sphere-Tree Construction Toolkit.

4.3 The Application Layer

The application layer is the starting point for any VR application developed on EnCIMA. Through this layer an application has access to all the engine's functionality and resources. The class Scene, shown on the top of Figure 5, is the realization of this layer. This class contains a high-level manager (Manager) that has a reference to every manager located in the core.

This centric approach facilitated the implementation of the Scene class where several high-level functions are available to the developer. The only requirement to start off an application is to derive through inheritance the Scene class and override the `Init()` and `Execute()` methods. The `Init()` method, as the name implies, initiates the application's objects and resources and is called only once in a run, whereas the `Execute()` is called every frame and is responsible for the application loop that contains the appropriate sequence of actions. The class CVRScene handles all the other details, such as resource allocation, the rendering of 2D and 3D objects, and the release of system resources when the application finishes.

Other methods available in the Scene class are:

- `CreateFont2D()`: creates bitmap fonts to be used as HUDs.
- `CreateBillboard()`: creates a textured polygon whose front-face is always facing the virtual camera. This resource is specially useful in particle systems.
- `CreateMd2()`: loads a animated object in Md2 format. This alleviates the developer's burden of having to program animation, since this can be done in a prestage, using specialized modelling and animation software.
- `CreateObj()`: loads an static object in Wavefront's OBJ format. The object can be represented by a mesh of triangles or n-sided polygons, associated to material or texture. This is specially useful when loading predesigned scenarios.
- `CreateBmpTerrain()`: generates a heightfieldbased terrain from a greyscale bitmap image.
- `LoadSound()`: loads a 3D sound that can be positioned anywhere within the VE.
- `LoadMusic()`: loads a 2D sound to be played regardless of the user's location within the VE.
- `CreateSkyBox()`: creates a background for the environment based on a texturized cube that encapsulates the entire VE.
- `CreateSkyDome()`: creates a background for the environment based on a texturized sphere the surrounds the entire VE.

In addition to the above mentioned functions, the `Scene`'s high-level manager grants access to all the core's modules, thereby allowing direct use of important underlying behavior when needed. Tables 1 summarizes a comparison of functionalities provided by EnCIMA and the reviewed engines of Section 3.

5. Case Study

In this section we present two case studies in which the EnCIMA engine has been successfully applied in order to generate a VR application. The goal of the first

VR ENGINES

Feature	enJine	SmallVR	CGE	EnCima
Network	–	–	•	•
Collision detection	•	•	•	•
Static models	•	•	•	•
Animated models	•	–	•	•
Terrain generation	–	–	–	•
Particle system	–	–	•	•
Standard Input device	•	•	•	•
3D sound	–	–	•	•
Tracker support	–	•	•	•
Haptic device	–	–	–	•
Dataglove	–	•	•	•
3D Mouse	–	–	–	•
Culling	•	•	•	•
Partition Environment	•	–	•	–
Portability	•	•	•	–
Multi-screen projection	–	•	–	•

Obs. '•' means feature supported.

Table 1: Comparison of engine's functionality.

application (shown in Figure 7) was to support navigation and exploration of an oil platform's installations. Furthermore, the application simulates two types of situations: the execution of evacuation and rescue procedures in case of a serious accident, and a training scenario for firefighters.

The environment for this application is represented, basically, by a static 3D model of the oil platform in conjunction with a skybox. The code fragment presented in Code 3 shows the `Init()` method that instantiates the application resources. To create the simulation's graphical interface the developer only needs to declare reference variables for the objects needed and invoke the methods from both the `Scene` class and the core's modules to allocate resources.

The second application, called AVIDHa (Atlas Virtual Interativo Distribuído Háptico), is a human body 3D atlas for anatomy study purpose. The system provides high definition models of human body parts with photo-realistic textures, as shown in Figure 8.

The application allows the anatomy student to fly through and inside the human body. The flight control is done with either 3D mouse or joystick. The student may also choose to investigate each system separately, change organ's opacity to exam internal parts, or capture screenshots for later examination.

In terms of performance for the AVIDHa application, the engine delivered a refresh rate of 30 frames per second, for a 3D models with 5.3 million polygons and 87.8 MBytes of texture, running on a Intel Pentium D (3.0 GHz e 2.0 Gbytes of RAM) with a NVIDIA GeForce 8800 GTX.

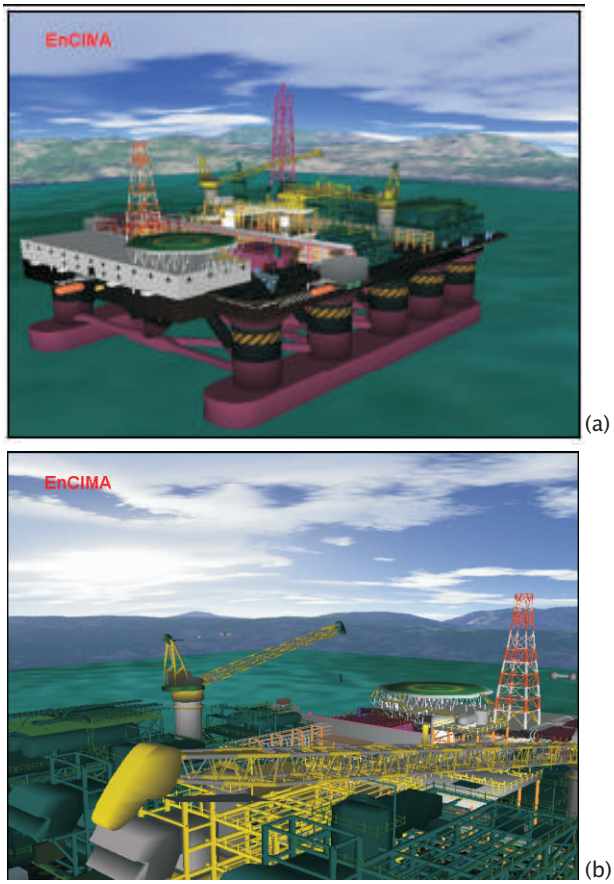


Figure 7. The simulation of an oil platform: (a) overview, and (b) detail.

In the next development phase of this application, we intend to enable students to use a phantom device to feel the organ's contours and haptic texture, as well as to provide network communication through a client-server model so that the teacher would be able to control the students' actions.

6. Conclusion and Future Work

An overall subjective evaluation of the applications described in the case studies section showed us that the use of EnCIMA engine speeded up the overall development process, even in situations that require the use of special interaction devices, such as 3D mice, phantom, or CAVE.

The EnCIMA's flexible API offers the basic elements needed by all visualization and simulation applications, without requiring the user to have any knowledge on OpenGL or DirectX libraries.

The next step is to work on the sub-systems layer in order to extend its support to other platforms besides Windows. This extension would not affect any of the previously created applications, since EnCIMA's

```
void SceneTest::Init() {

   // Creates a 2D bitmap font.
   fAngle = 0.0f;
   Font   - CreateFont2D( "EnCIMA", 30, true);
   Font->SetColor( 1.f, 0.f, 0.f);
   Font->SetPosX( 50.f);
   Font->SetPosY( 50.f );

   // Set the camera's position.
   pManager->cView.SetPosition( -500, 90, 0 );
   pManager->SetPosX( 50.f );
   pManager->SetPosY( 50.f );

   // Loads the platform model stored as an obj file.
   obj = this->CreateObj("pt40t.obj","objects\\",true);

   // Loads sound
   CVRListener* list - GetListener();
   CVRSound* sound = LoadSound("\\Sounds\\SANDSTRM.wav");
   sound->SetRepeat( -1);
   sound->PlaySound( );
   sound->SetVolume( -800 );
   sound->SetMaxMin( 300, 100);
   sound->SetPosition( -160, 30, 0);

   // Creates a skybox
   skybox = CreateSkyBox( true );
   skybox->SetCenterBox( 0.f, 0.f, 0.f );
   skybox->SetBoxSize( 3000, 1000, 3000 );
   skybox->SetFrontTexture( "\\Images\\front.bmp" );
   skybox->SetBackTexture( "\\Images\\back.bmp" );
   skybox->SetLeftTexture( "\\Images\\left.bmp" );
   skybox->SetRightTexture( "\\Images\\right.bmp" );
   skybox->SetDownTexture( "\\Images\\down.bmp" );
   skybox->SetUpTexture( "\\Images\\up.bmp" );
   skybox->SetDrawGround( true );
   skybox->SetFloorHeight( 400 );
}
```

Code 3. Initializing the oil platform simulation application.

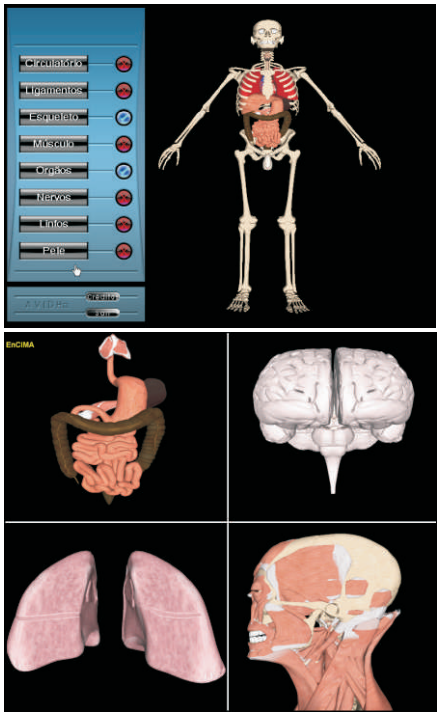


Figure 8. The AVIDHa application: (a) the interface is composed of user controls rendered as HUDs, and a 3D rendering space in which the human body model is shown; in (b) we show a detailed view of 3D models of human body parts.

application layer provides a high-level, platform-independent API.

Finally, we will develop two new modules: one module to provide navigation aids (such as maps, signs, and automatic camera control) that would help users to perform wayfinding in complex virtual environments, and; another to afford object deformation, which is important in providing a consistent visual and haptic feedback.

7. Acknowledgement

The authors acknowledge financial support from Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ, grant number 170.332/2006), Rede Nacional de Ensino e Pesquisa (RNP Giga, number 2439), PCI/LNCC/MCT, and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, grant number 305106/2004-0).

8. References

[1] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira. VR Juggler: A virtual platform for virtual 8 reality application development. In *VR '01: Proceedings of the Virtual Reality 2001 Conference (VR'01)*, page 89, Washington, DC, USA, 2001. IEEE Computer Society.

[2] G. Bradshaw and C. O'Sullivan. Adaptive medial-axis approximation for sphere-tree construction. *ACM Transactions on Graphics,* 23(1):1–26, 2004.

[3] G. C. Burdea and P. Coiffet. Virtual Reality Technology. *Wiley-Interscience*, second edition, 2003.

[4] K. C. Finney. 3D Game Programming All in One. Course Technology PTR Game Development Series, Andr´e LaMoth ed. *Thomson Course Technology PTR*, first edition, April 2004.

[5] D. Johnston. 3d game engines as a new reality. In *Proceedings of the 4th Annual CM316 Conference on Multimedia Systems*, pages 1–8, Southampton University, UK, November 2004. http://mms.ecs.soton.ac.uk/mms2004.

[6] M. Lewis and J. Jacobson. Game engines in scientific research - introduction. *Communications of the ACM*, 45(1):27–31, 2002.

[7] J. G. R. Maia. CRAbGE: Uma arquitetura para motores gráfico, flexíveis, expansível e portável para aplicações de realidade virtual. *Master's thesis*, Departamento de Computação, Universidade Federal do Ceará, Março 2005.

[8] J. G. R. Maia, J. B. Cavalcante Neto, H. O. O. Gomes, and C. A. Vidal. CRAbRender: Um sistema de renderização para aplicações de rv. In *Proceedings of the VII Symposium on Virtual Reality, SVR2004*, volume 1, pages 380–382, São Paulo, SP, 2004.

SBC/Faculdades Senac.

[9] J. G. R. Maia, J. B. Cavalcante Neto, and C. A. Vidal. CRAbGE: Um motor gráfico, customizável, expansível e portável para aplicações de realidade virtual. In *Proceedings of the VI Symposium on Virtual Reality, SVR2003*, volume 1, pages 3–14, Ribeirão Preto, SP, 2003. SBC/Faculdades COC.

[10] R. Nakamura, J. Bernardes, and R. Tori. enJine: Architecture and application of an open-source didactic game engine. In *B. Feijó, A. Neves, E. Clua, L. Freire, G. Ramalho, and M. Walter, editors, Digital Proceedings of the V Brazilian Symposium on Computer Games and Digital Entertainment*, pages 1–7, Last access: 22/11/2007, November 2006. http://www.cin.ufpe.br/˜sbgames/proceedings/files/enjine.pdf.

[11] W. Paper. Physics, gameplay and the physics processing unit. Technical report, AGEIA Technolgies Inc., 82, Pioner Way, Mountain View, CA, USA, 2005.

[12] M. S. Pinho. SmallVR: Uma ferramenta orientada a objetos para o desenvolvimento de aplicações de realidade virtual. In *Proceedings of the V Symposium on Virtual Reality, SVR2002*, volume 1, pages 329–340, Fortaleza, CE, 2002. Porto Alegre: SBC.

[13] W. R. Sherman and A. B. Craig. Understanding Virtual Reality: Interface, Application, and Design. Morgan Kaufman Publishers, first edition, 2003.

[14] D. Shreiner, M. Woo, J. Neider, T. Davis, and A.R.B. OpenGL. OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL(R), Version 2. *Addison-Wesley Professional*, fifth edition, August 2005.

[15] B. Stang. Game engines: Features and possibilities. *Technical report, Institute of Informatics and Mathematical Modeling at The Technical University of Denmark*, 2003.

[16] G. van den Bergen. Collision Detection in Interactive 3D Environments. The Morgan Kaufmann Series in Interactive 3D Technology. Morgan Kaufmann, harcover edition, October 2003.

Telepresence: A tool for distance education

Federico Builes, Pedro Esteban Vicente Duarte, Juan Esteban Pemberthy, Juliana Restrepo, Helmuth Trefftz
Eafit University
Virtual Reality Laboratory
{fbuilesv}, {pesteban}, {jpembert}, {jrestre}, {htrefftz}@eafit.edu.co

Abstract

Telepresence is a tool for distance education created by the Virtual Reality Laboratory of the Eafit University[1]. We give a technical overview of the software, its capabilities and its current uses. We also try to address some of the issues we experienced in a recent series of tests we did with the tool in El Peñol, Antioquia in Colombia, using the software to teach Spatial Geometry to teachers of the region.

1. Introduction

The people living in developing countries have limited access to education due to several factors such as the social situation of the region, the distance to the main cities and prohibitive costs; Colombia is not an exception to this. Several technical-based solutions currently exist to this problem, such as web pages, shared Powerpoint presentations, e-mail or video-conferencing software.

Thanks to the Internet, every day more and more children and adults are gaining access to vast resources of information that they didn't have before, but we still think that the situation could be improved.

The solution presented in this paper is a software called Telepresence. This is a tool for synchronous distance education that provides students and teachers a way to communicate and interact in a virtual reality environment. We provide audio, video and presentation capabilities, in addition to what we think is the cornerstone of the system: 3D manipulable objects that serve as a medium of interaction for both sides. Each one of these components will be described in the following sections.

Our tool was created in response to the needs of our country, where many remote towns have problems getting teachers to travel to those areas. We realized we could provide a partial solution to this problem with two strategies:

1. Have the teacher teach his classes from his house or home town while the students are connected from the remote location.

2. Train teachers from those regions using the tool, making it a teacher-to-teacher tool, and in that way, the students wouldn't have to see the class through a computer every single day.

Another issue that we adress with the tool is the transmission of abstract knowledge, such as 3D mathematics, physics and geometry. Illustrating some key concepts like 3D functions, forces in physics and the figures in spatial geometry can be hard for teachers when they only have access to a whiteboard. Our idea is to let them rely on the software to display all the 3D contents of their courses. In that way, it will be easier for teachers to demonstrate 3D contents in their classes and students will grasp the concepts with much more ease. This last approach has worked very well in the experiences that we have conducted, as described in [12].

2. Related Work

The NICE project [7] was created as an immersive learning environment for children, implemented using CAVEs. In NICE, several children could collaborate with other remotely-located children to, for example, construct and cultivate simple virtual ecosystems. Children could communicate via voice, but each participant could only see a 3D avatar of the other, lacking any facial expressions. Our tool adds video in order to alleviate this deficiency.

AVALON [11] was previously created by our laboratory to explore the use of Networked Virtual Environments as means to support distance education. Users moved freely inside virtual worlds that were created to support specific contents of an Environmental Awareness course. Users could communicate with voice and their avatars. The instructor could project slides in a whiteboard, located in a specific part of the virtual world. Each participant was represented by a 3D avatar, but no video was transmitted. In order to compensate

for the lack of video, students could choose among a set of expressions in order to communicate their status to the instructor. These expressions were then displayed on top of the corresponding students avatar.

MUVEES[3] provides different education oriented virtual worlds that vary in contents according to topics being studied. Avatars represent users and there is some basic support for artificial facial expressions. Video-conference is still not present on this system.

Lalioti describes use of multimedia systems within virtual reality environments in [8]. Some of our reasoning is based on this and other works by the same author [5, 6].

3. Software Architecture

• This section serves as a general overview of the system. Telepresence is divided into four main components:

• Audio streaming

• Video streaming

• Shared Slides/Presentations

• 3D Contents

We think that 3D contents are worth taking a deeper look, so we've separated that into its own section.

The whole application was written using Java, the Java Media Framework (JMF) for the A/V streaming, Java3D for the 3D contents, Swing for the GUI and our own wrappers for the presentations (which can be Powerpoint or PDF slides).

The following subsections cover the technical specifications of each one of the system's components:

3.1 Audio

• Point-to-point audio is transmitted using existing technologies that sit on top of the Java Media Framework. Since not all the regions of the country have easy access to a stable Internet connection, we've divided the audio and video quality into four different segments:

• High

• Normal

• Low

• Audio-Only

We consider that even if the available Internet bandwidth is limited, audio is a must in all the outgoing and incoming communications. Since the experience without audio can be comparable to one watching a slideshow on the Internet, we feel that it's the base of the system, and it must be available under every single circumstance, even if video has to be disabled for bandwidth issues.

All the audio data is encoded using the GSM codecs provided by the Java Media Framework.

Working with this framework provides us an easy way to provide inter-platform compatibility, so all the communications work in Linux, Solaris and Windows if the base system is Java-capable.

3.2 Video

• Video is an important part of the software since it provides the students a real feeling of Telepresence (when used in combination with audio). As specified in the "Audio" section, the software provides different levels of QoS for media streaming that can be adjusted according to the available Internet bandwidth. We use four levels for video quality:

• High: Video is transmitted as a JPEG stream which provides an impeccable video for the users.

• Normal: The streams are encoded using H.263. This provides a decent video quality without consuming many resources.

• Low: These are JPEG frames glued together, giving the user a visualization of the other person, even if it's not a real time streaming.

• Audio-Only: Audio only doesn't provide any video functionality. To be used only under severely restricted connections.

All these functionalities are implemented over RTP as defined in [10]. The RTP layer provided by the Java Media Framework makes it easy to switch between different implementations "on the fly".

3.3 Presentations

In any given moment of a class, the teacher can switch between Virtual Reality and the Slides modes. The Virtual Reality mode is used for the 3D contents while the Slides tab provide an easy way to share Powerpoint or PDF presentations.

Since a reliable Internet connection is not available under all circumstances, we've chosen to use animation-less slides (encoded as single GIF images), to make them usable under limited contexts.

We've written a tool to transform the PPS or PDF files to a bundle presentation of GIF files that contains only the sequencing information and the slides themselves. This way, only the needed file is loaded at a time, providing a low latency system for both sides of the connection.

4. 3D Contents

As mentioned in the Introduction, 3D contents are the base of the application. A 3D content is basically a 3D *miniprogram* that runs inside the big application. These contents are created individually using Java3D to provide a form of interaction between the teacher and the students.

The idea of using them originally was making it simpler for the students to understand abstract concepts like multivariable calculus functions or spatial geometry forms. Using a 3D content means that the teacher doesn't have to draw a complex 3D image on a 2D surface (board), instead, the software creates the image for him. In the end, it makes it easier for the instructor to illustrate the functions (or forms, concepts etc) and it gives a much more pleasant experience for the students who don't have to wrap their brains around 3D objects in an awkward way.

The courses are divided into several blocks, each one containing one or more 3D contents, and possibly, some slides. Currently, the work of the programmer is to create the 3D part of the course, programming a Java3D scene which then is parsed and executed by a ContentManager from the application.

All a programmer has to do to get his/her 3D miniprogram into the application is adhere to a public Java Interface, and the application takes care of everything else. Being able to create the contents directly in Java3D gives a lot of possibilities to the programmers to represent information, but at the same time, it makes it hard for teachers to create new contents since they would need to know how to program in Java to start rolling their own courses. Right now every teacher that is creating a course has a CS student to help him/her in the creation of his contents, but this is something that should be changed in the future to make it easier for non-programmers to create their own courses using a specialized tool, even if it takes away some of the current freedoms that programmers have.

5. Experiences

During 2007 the project started some pilot tests in "El Peñol, Antioquia" to try out the software with mathematics teachers of the region. The first steps for this experience were trying to setup the Internet connections between the school where the courses were going to be taught and our University campus. We found some troubles when we didn't predict the closed ports of the incoming connections in the remote school. Since they used a different ISP, we had to go through a lot of hoops just to get things working, but after a lot of paperwork, we finally did it.

El Peñol is a town that is approximately 60km away from Medelln, where the main Eafit University branch is located. This is where the professor was teaching his class every week.

The difference between the Internet connection of both places is significative: while the University has access to a 20+ Mbps connection, the town's school only had 96Kbps, divided across 20 computers. This was a notable problem in all of our tests regarding the video streaming.

The students for this test were math teachers from the region as mentioned earlier, and the teacher in charge of the classes was the professor Pedro Vicente Esteban Duarte, who taught a course focused on Spatial Geometry using a framework called "Teaching for Understanding", created by researchers of the Project Zero at the Hardvard Graduate School of Education[13].

The technological environment for the "Leon XIII" school in "El Peñol" were regular computers with 1.6Ghz processor, 512MB RAM and the already mentioned, 96kbps shared Internet connection. This also presented an issue when dictating some of the most computer processor/video intensive courses as the 3D Plotter.

Each side of the connection had access to a microphone and a web-camera. Aditionally, the students in the "Leon XIII" could see the professor in Medellin using a projector. During the 3 months that the experience lasted, students were instructed in different topics, ranging from the basic geometry concepts (What is a point and where is it used) to some more advanced concepts (functions in the plane, functions in the space and how do we see them in the real world).

During the middle of the project, the Internet connection in the remote location got worse, so we had to stop using the video streaming. During a heavy rain period we could not teach any classes because the whole location would be out of electricity or Internet. During the final period, everything worked fine again.

Leaving aside the non-technical part of the experience, it's worth mentioning some things that we didn't consider before going on with the tests:

• Internet Stability: We never considered this to be much of a problem until we got to the place.

• Weather Conditions: We didn't think this could affect

us, but for more than week the region had no electricity.

• Preparation of the Teachers: For some teachers it was the first time using a computer, so we had to spend several weeks teaching them how to do basic word processing and e-mail. In the end, these teachers were able to start the application (Telepresence) by themselves, needing no external help, which is a big advance by itself.

The Spatial Geometry course was arranged before starting the course by two programmers with the Math teacher, and they got to implement around eight different 3D contents while the teacher made a set of Powerpoint slides for each 2 hours class. This way, students got to see both parts of the picture, theoretical and practical examples where they could interact with the objects.

The course was built gradually, starting from the point and its physical representations to the creation of big objects in different civilizations of the world. All the real world examples were a great help for the students. In the end, they were able to teach these same courses to their students, and each one of them applied it to their different areas of knowledge (ranging from basic geometry to the mathematics in the country fields, in biology and even on their houses). After this, we found the project to be really succesful and it will be used next year in two more locations.

More information about previous experiences with the tool can be found at [12] and [4].

The tool itself is released under the GNU GPL 2 and can be downloaded from [2].

6. Problems

The project has experienced several issues since its inception. The most notable one as mentioned several times before, was the availability of good, reliable Internet connections. Developing countries as Colombia barely have Internet coverage outside the main cities, and there's almost no decent, stable connection in any town in the country which makes a virtual presence experience much harder to achieve.

The Internet problem brings a serious problem: The quality of audio and video streaming. In some of our recent tests (as related in the "Experiences" section) we had to fully disable video because the client's connection (students) couldn't keep up with the audio and video streaming at the same time. Even if the students gave good reviews of the application, and the project managers were satisfied, we feel that this was not a full VR experience, and some advances should be made in this area. Since most of our A/V streaming solution was homegrown, we tried to use more professional and tested applications like Skype and GnomeMeeting (the idea was a mention in [9]). Using Skype, we were able to overcome some issues on the audio part of the program, but the video was still too laggy to be really usable.

Another issue that we had is the unicast point-to-point connection. With the current methods, a teacher can be anywhere dictating his class while a whole classroom is in another part of the globe, seeing him/her and interacting with him/her through a video-beam, speakers and only one computer. The idea behind the project would be to use less students but each one with his own computer, camera and microphone.

The issue that arises with this, is again, bandwidth. Even if we can implement a solution for multi-point video and audio, the bandwidth consumption is too high to think of it as a real solution. We think a possible answer to this problem could be using image avatars instead of video, since this is the part that consumes most of the connection.

And lastly, one more issue the project experienced is the lack of computer education in the participants of the project. Many of the teachers involved in the Telepresence experience had not touched a computer in their life, so they were still afraid and confused when it came to interact with the PC. The good news is that after the project many of them feel comfortable using a computer for their daily work, so this should not be an issue in a near future.

7. Conclusions

Our most important conclusion is that a lot of work still has to be done. We've reached a point where we feel proud about our work, and we believe in this methodology as something that will be big in the future, but we still have a lot of room to improve.

One of our biggest concerns is the unfamiliarity of the users with computers as a tool for everyday teaching, but the experience shows us that this is not a show stopper since the users will learn to teach and use computers as one more tool of their repertoire.

Another point we bring to discussion is the availability of Internet in remote regions. We still don't find good connections (or even connections for all that matters), so this is a point that has to be taken into account whenever we're deploying the software in a remote location. As we showed, poor Internet connections also implications on the quality of the services that we're

trying to provide.

A weak connection means weak streaming, which is worse than no streaming at all in the case of video, so this has to be taken into consideration too.

8. Future Work

As pointed out in the "3D Contents" section, the tool currently only uses contents that were written using Java3D. A specialized tool to create these contents is needed to facilitate this task for teachers creating their own courses.

Another improvement that was mentioned in the "Experiences" section is replacing the Java Media Framework for something newer. This API hasn't been maintained in three or fours years and it's starting to lack in respect to new technologies as H.264 for video or MPEG-4 for audio and video. This is what we're currently aiming to: the creation of our own library for video and audio streaming.

Although the tool is released under an open source license, we need to get more people using it, reporting any bugs they may encounter and helping us improve it.

The application could also be improved to provide multipoint streaming capabilities. This means that all the students wouldn't need to be together anymore, they could be on separate parts of the city, town or campus receiving their classes. The problem with this would be the bandwidth capabilities needed to provide everyone of a different video stream. This could be a possibility when the classes are being dictated in the same local area network , e.g. same university campus.

Finally, with the upcoming launch of the OLPC's XO computer, a light version of the program could be developed to make use of the million of kids who'll have ready access to Internet-connected computers in a nearby future.

9. Acknowledgements

The work reported in this paper has been funded by Eafit University, Colciencias and the "Secretaría de Educación" in Antioquia, Colombia. We'd like to thank all the teachers from eastern part of Antioquia for taking the time to participate in this experience. We would also like to thank the people of the Virtual Reality Lab. in Eafit University for their help with testing the application, and finally: Lina Escobar, Andres Agudelo, Andres Quiroz and Elizabeth Rincon for their original work on the Telepresence project.

10. References

[1] http://arcadia.eafit.edu.co.

[2] http://code.google.com/p/telepresencia.

[3] Jim X. Chen, Yonggao Yang, and Bowen Loftin. Muvees: a pc-based multi-user virtual environment for learning. *IEEE Virtual Reality*, 2003.

[4] Agudelo et. al. International conference on education (iadat-e. *Telepresence for Distance Education: Lessons Learned*, 2004.

[5] M. Goebel F. Hasenbrink, H. Tramberend and V. Lalioti. Immersive telepresence in responsive virtual environments. *NEC Research Symposium*, Yokohama, Japan, 1998.

[6] M. Goebbel G. Goebbels, V. Lalioti. On collaboration in distributed virtual environments. *HC2000 - 3rd International Conference in Human and Computer*, University of Aizu, Japan, 2000.

[7] A. Johnson, M. Roussos, J. Leigh, C. Barnes, C. Vasilakis, and T. Moher. The nice project: Learning together in a virtual world. *VRAIS*, 1998.

[8] V. Lalioti. Multimedia in virtual reality. *1st Panhellenic Conference on New Information Technology*, Athens, Hellas, 1998.

[9] Thies Pfeiffer, Matthias Weber, and Bernhard Jung. Ubiquitous Virtual Reality: Accessing Shared Virtual Environments through Videoconferencing Technology. In Louise Lever and Marc McDerby, editors, *Theory and Practice of Computer Graphics 2005*, Eurographics UK Chapter Proceedings, pages 209–216. Eurographics Association, 2005.

[10] Schulzrinne, Casner, Frederick, and Jacobson. RTP: A transport protocol for real-time applications. *Internet-Draft ietf-avt-rtp-new-01.txt (work inprogress)*, 1998.

[11] H. Trefftz, C. Correa, M. A. Gonzalez, G. Imbeau, J. Restrepo, M. I. Velez, and C. Trefftz. Distance education and distributed virtual environments. *The Virtual Campus: Trends for Higher Education and Training*, 1997.

[12] Helmuth Trefftz and Juliana Restrepo. Telepresence support for synchronous distance education. *ACM Symposium on Virtual Reality Software and Technology (VRST),* 2005.

[13] M. S. Wiske. Teaching for understanding: Linking research with practice. 1997.

Um Ambiente Virtual Colaborativo e Telecomandável Baseado em X3D

Bruno Rafael de Araújo Sales, Liliane dos Santos Machado
LabTEVE - Universidade Federal da Paraíba,
João Pessoa/PB, Brasil
brunorasales@gmail.com, liliane@di.ufpb.br

Resumo

Este trabalho trata de um sistema de realidade virtual (RV) colaborativo baseado no novo padrão Extensible 3D (X3D). Uma estrutura é definida a fim de demonstrar como é possível o desenvolvimento de ambientes desse tipo utilizando o X3D. Deste modo, é apresentada a união de uma aplicação colaborativa com um sistema de armazenamento de dados que pode dar suporte à interligação do meio virtual com o real a fim de possibilitar experimentos sejam realizados por usuários de forma conjunta utilizando dispositivos localizados remotamente. Assim, o sistema pode vir a auxiliar professores e alunos em suas atividades laboratoriais visto que estes podem compartilhar e colaborar num mesmo ambiente.

1. Introdução

A capacidade de unir pessoas geograficamente distribuídas oferecida pela Internet serve como base para o desenvolvimento de aplicações e serviços colaborativos em diversos campos de atuação. Neste contexto estão inseridos os Ambientes Virtuais Colaborativos (AVC), de maneira a possibilitar seus participantes a interagirem através da simulação de um mundo real ou imaginário ou até mesmo da manipulação de objetos virtuais no mundo real [3].

O que diferencia um AVC de um Ambiente Virtual (AV) de uso local é que no primeiro caso existe a possibilidade de interação e cooperação entre usuários distantes no decorrer da execução de uma determinada tarefa [3]. Um dos problemas encontrados quando utilizados esses ambientes é a garantia de consistência das informações, isto é, garantir que todos os usuários estão presentes em um mundo com mesmas configurações independente do momento em que a conexão foi estabelecida.

A interligação de um AVC com um sistema de armazenamento de dados permite que variáveis de status dos componentes da cena sejam guardadas com suas configurações atuais. Deste modo, é possível ser feito um trabalho contínuo dentro do AVC, ou seja, mesmo após desconectar o usuário terá a garantia de que a cena irá permanecer da maneira que foi deixada. Isso possibilita o usuário a dar continuidade a tarefas inacabadas em um momento futuro, partindo do ponto onde parou.

Além disso, o armazenamento das configurações atuais da cena pode dar suporte à interligação de AVs com ambientes reais com intuito de permitir o acesso remoto a recursos e dispositivos fisicamente existentes [10,11]. Por meio desse acesso, pessoas situadas em locais distantes podem manter controle sobre um ambiente real somente manipulando o AV relacionado. Dessa maneira, professores e estudantes podem fazer uso de estruturas laboratoriais nos chamados laboratórios remotos [8].

Este artigo pretende apresentar uma estrutura usada no desenvolvimento de um AVC que permite a colaboração entre usuários na execução de tarefas e realização de experimentos. Também é apresentada a possibilidade de interligação do AVC a um ambiente real a fim de gerenciar dispositivos fisicamente existentes por meio de telecomandos.

2. Ambientes Virtuais Colaborativos

Um AVC possui como característica fundamental o fato de englobar além da visualização e navegação em uma cena 3D, a possibilidade de usuários poderem colaborar na execução de tarefas. O aspecto colaborativo é inserido nos AV a fim de permitir que usuários possam estar presentes em um mesmo mundo, realizar tarefas em conjunto, entre outras coisas.

Em ambientes deste tipo a sensação de presença é bem maior do que em aplicações que utilizam tecnologias 2D. O sentimento de "estar junto" ou "trabalhar junto" com outros membros aparece com maior afinco em ambientes 3D pois permite às pessoas identificarem-se visualmente tornando a interação remota e a colaboração mais natural [3].

Entretanto, o que realmente promove a colaboração e o

envolvimento em um ambiente 3D é a seqüência de atividades que requer participação e interação entre estudantes de diferentes localidades. A competição amigável, discussões sobre temas propostos e o conhecimento prévio de cada estudante, faz com que a colaboração traga benefícios no aprendizado de todos [9]. Cada participante do ambiente contribui com os seus conhecimentos, acrescentando novas idéias e informações a outros estudantes [5, 6].

A inserção do fator colaborativo em um AV acarreta a análise de alguns pontos antes não observados. Aspectos como a quantidade de usuários, a complexidade da cena, a capacidade da rede utilizada e como será feita a interação entre usuário e AV e usuários entre si, devem ser observados no desenvolvimento de um AVC [3]. Um nmero elevado de usuários exige mais processamento computacional. Observando também o número de tarefas que poderão ser realizadas por cada usuário. A complexidade do mundo está relacionada ao número de objetos presentes na cena e também do nível de detalhamento do ambiente. A capacidade da rede influencia o funcionamento de um AVC, pois a comunicação e a troca de informações entre os usuários é intensa. Redes de baixa velocidade podem comprometer a sensação de tempo real da aplicação.

Sob o ponto de vista da interação, os usuários podem interagir por meio de ações ou mensagens de texto ou voz, sendo que uma abordagem muito comum é utilização de avatares (representações do usuário através de entidades virtuais) para fazer a representação visual dos usuários no AV.

3. Estrutura do AVC

Esta sessão visa expor uma estrutura proposta para desenvolvimento de Ambientes Virtuais Colaborativos e Telecomandáveis. A estrutura a ser apresentada está dividida em camadas referentes às diversas necessidades que um ambiente desse tipo possui. Como se trata de um sistema cliente/servidor algumas camadas da estrutura do cliente diferem da estrutura do servidor.

O servidor está subdividido em: Gerenciamento, Transporte e Armazenamento. O cliente, por sua vez, é composto por: Gerenciamento, Cena 3D e Transporte. A camada de Gerenciamento do servidor trata de receber informações sobre alterações na cena feitas por clientes e decidir o que fazer com essas informações (a quem retransmitir, enviar para armazenamento, etc). A camada de Armazenamento é responsável por receber dados da camada de gerenciamento e adicioná-los ou alterá-los no banco de dados. Bem como recuperar do banco de dados, dados solicitados pela camada de gerenciamento.

No cliente, a camada de Gerenciamento tem a tarefa de receber dados da camada de transporte (provenientes do servidor) e enviá-los a cena em forma de eventos. Cena 3D diz respeito ao ambiente virtual no qual o cliente está inserido. As camadas de transporte do cliente e do servidor se assemelham. Ambas possuem os mesmos objetivos: promover a troca de informações entre as estações. A Figura 1 ilustra as estruturas de cliente e servidor.

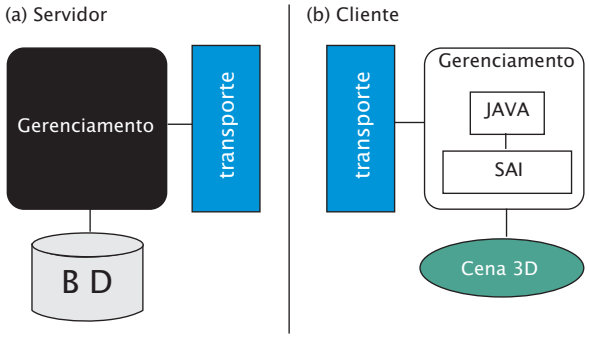


Figura 1: Estrutura de camadas do Servidor e do Cliente.

4. X3D e *Scene Access Interface* (SAI)

Desenvolvido pelo grupo Web3D, o X3D é "um padrão aberto para distribuir conteúdos de realidade virtual em 3D, em especial pela Internet." [1]. Um arquivo X3D contém não só dados geométricos, mas também descrições de comportamentos dos objetos e da cena. O X3D surgiu com o intuito de substituir o padrão existente (VRML – *Virtual Reality Modeling Language*), corrigindo alguns aspectos indesejáveis que este contém e incorporando avanços de dispositivos e técnicas. O X3D oferece suporte a uma série de dispositivos, métodos e técnicas, sendo possível adicionar funcionalidades às suas aplicações. Essa extensibilidade (adição de funcionalidades) e o conjunto de serviços (funcionalidades já disponíveis) do X3D são definidos por *Profiles* e *components* [1, 7].

Apesar de suas diversas características, o X3D não é uma linguagem de propósito geral. Por esta razão uma API chamada SAI (*Scene Access Interface*) está sendo desenvolvida com o objetivo de permitir integrar cenas X3D e aplicações desenvolvidas em outras linguagens, como Java, C ou linguagens de *script*. A SAI visa possibilitar o acesso a uma cena X3D através de uma aplicação externa. Assim, passa a ser possível inserir objetos, removê-los, notificar eventos e realizar modificações que afetam a cena e, conseqüentemente, elementos relacionados e externos a ela. Ou seja, a cena

pode ser totalmente controlada de forma indireta através de um programa escrito em linguagens de programação ou script [4, 7].

5. Desenvolvimento

O desenvolvimento do AVC baseou-se na estrutura definida para cliente e servidor. Foram abordadas questões sobre consistências dos mundos, colaboração efetiva dos usuários, armazenamento do status da cena, telemanipulação, sensação de presença e tempo real. Através desta estrutura, os objetos virtuais passam a poder representar equipamentos reais localizados remotamente de modo que seja viabilizada a cooperação entre estudantes situados em locais distantes na realização de experimentos em laboratórios remotos. Uma ilustração de como as ações executadas no ambiente virtual chegam ao ambiente real é mostrada na Figura 2.

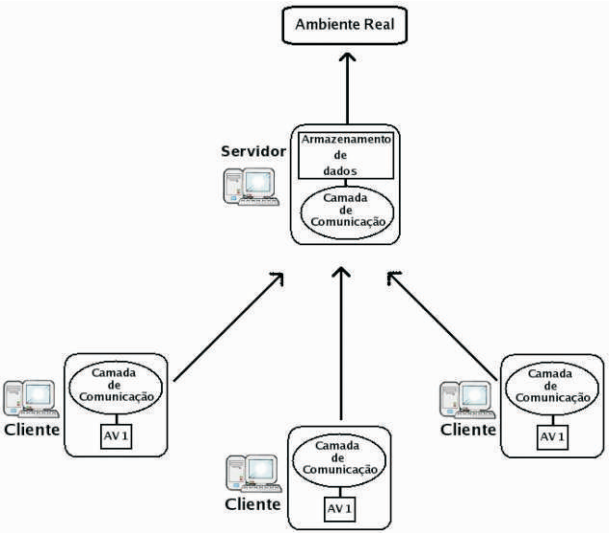


Figura 2. Esquema de como ações executadas na aplicação do cliente chegam na aplicação do servidor são interpretadas e repassadas para o ambiente real.

O X3D se inseriu no desenvolvimento do sistema proposto como padrão para modelagem do AV. As vantagens do X3D sobre o VRML influenciaram essa decisão, visto que haverá necessidade de integrar dispositivos específicos ao AV e permitir seus controles. Neste caso, a SAI permite esta integração no X3D.

O modelo de comunicação adotado foi o cliente/servidor. Nesse modelo os dados são armazenados em computadores denominados servidores. Além disso, toda informação trocada por clientes passa antes pelo servidor. Este gerencia para quem irá reenviar as mensagens que chegaram a ele. O protocolo de comunicação utilizado foi o UDP (*User Datagram Protocol*), devido à sua maior rapidez na entrega dos dados e não necessidade de confirmação de mensagens [2].

O sistema permite ainda que usuários estejam conectados a mundos diferentes ou até mesmo partes diferentes do mesmo mundo virtual. Assim, as mensagens que chegam ao servidor provenientes de um determinado cliente, só são repassadas para os demais que se interessem por essa mensagem, ou seja, os que estiverem inseridos no mesmo ambiente. Isto possibilita o carregamento gradativo das cenas, isto é, ao invés de toda a cena ser armazenada localmente, apenas a parte na qual o usuário está navegando é requisitada. Tal abordagem evita a sobrecarga de memória principal da máquina cliente, além de diminuir o tempo de espera do cliente para iniciar a navegação. Desse modo, o cliente recebe do servidor o ambiente em que deseja se inserir e também as configurações atuais deste ambiente, ou seja, o status de componentes da cena. Essas configurações ficam armazenadas no banco de dados. Para cada cena estão armazenadas as configurações de componentes que podem sofrer algum tipo de alteração. Por exemplo, o status de uma lâmpada informando se ela se encontra ligada ou desligada. Com estas configurações armazenadas, garante-se que os usuários que escolherem uma mesma cena para navegar estejam inseridos em um mesmo contexto, com o status dos componentes iguais.

6. Resultados

Os resultados obtidos consolidaram o funcionamento do sistema e foi possível verificar a colaboração entre os usuários em tempo real. A aplicação utilizada para testes tratou um ambiente bastante simples, representando uma sala de aula composta por cadeiras que podem ser rotacionadas e um interruptor que apaga ou acende a luz. Neste caso, o servidor não possui uma cena gráfica e apenas escuta conexões de clientes, gerencia as informações recebidas e se comunica com o banco de dados.

Assim que a cena é exibida, o cliente pode navegar livremente além de fazer alterações desejadas e visualizar alterações feitas por outros usuários presentes na mesma cena. A Figura 3 exibe a cena utilizada para testes mostrando as que modificações podem ser feitas nas suas configurações. Nesta cena, o usuário é capaz de acender ou apagar uma luz clicando em um suposto interruptor e também rotacionar as cadeiras.

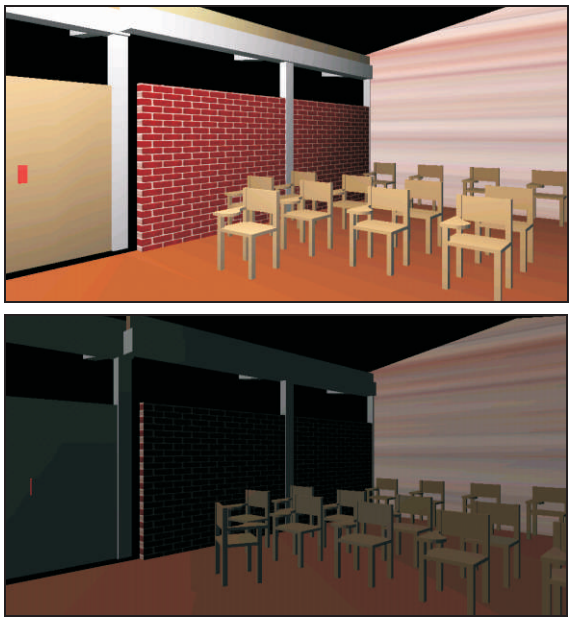


Figura 3. a) Cena original não modificada e b) Cena com a luz apagada e uma cadeira rotacionada.

Foram realizados testes com cinco clientes, localizados geograficamente distantes, comunicando-se através da Internet. Os clientes possuíam uma conexão com velocidade de 300kbps. Os atrasos não foram perceptíveis para os usuários e as alterações feitas por um cliente foram observadas quase que de imediato pelos demais. Adicionalmente foi realizado um teste composto por quatro clientes, dois conectados a um ambiente e dois conectados a outro. As modificações individuais de cada cliente não afetaram os clientes conectados a outro ambiente e apenas foram reenviadas para os que estavam inseridos na mesma cena do cliente que realizou modificações, confirmando assim o funcionamento correto do sistema.

7. Conclusão

A colaboração em AV é de fato algo que pode dar suporte ao processo de ensino e aprendizado de estudantes e professores. Além do mais, fazer com que as tarefas ou ações feitas sejam armazenadas em um banco de dados por meio de variáveis de status dá ao usuário a sensação de dinamismo e tempo contínuo. Desta maneira, tarefas inacabadas podem ser concluídas em um momento futuro. Considerando ainda a possibilidade de interligação do AV com um ambiente real, conclui-se que fazer uso desses AVCs direcionados a experimentos laboratoriais beneficia ainda mais o processo de construção do conhecimento devido à possibilidade de realização de experimentos através dos laboratórios remotos.

Como trabalho futuro sugere-se o melhor tratamento no caso em que dois clientes tentam modificar um mesmo objeto simultaneamente. Além disso, também devem ser feitos testes de rede a fim de medir com precisão os atrasos das mensagens, mesmo tendo sido observado que tais atrasos não comprometeram o bom funcionamento do sistema.

8. Agradecimentos

Este trabalho tem apoio do CNPq através do processo 485437/2007-4.

9. Referências

[1] A. Cardoso, C. Kirner, E. Lamounier, J. Kelner, “Tecnologias para o Desenvolvimento de Sistemas de Realidade Virtual e Aumentada”. Ed.UFPE, Recife. 2007.

[2] A. S. Tanenbaum, “Redes de Computadores”. Editora Campus, 4ª ed. 2003.

[3] L. C. A. Rinaldi, et al. “Fundamentos e Tecnologia de Realidade Virtual e Aumentada”, cap 5. SBC. 2006.

[4] Web3D, “SAI Tutorial”. Online: www.xj3d.org/tutorials/general_sai.html. Acesso em: outubro/2007.

[5] N. D. Blas, C. Poggi, “European virtual classrooms: building effective “virtual” educational experiences”. Virtual Reality, v. 11, n.2-3, pp: 129-143. 2007.

[6] B. Lok, et al. “Applying virtual reality in medical communication education”. Virtual Reality, v. 10, n.3, pp: 185-195. 2006.

[7] Web3D, “X3D Public Specifications”. Online: www.web3d.org. Acesso em: outubro/2007.

[8] L. S. Machado, et al. “Improving Interaction in Remote Labs Using Haptic Devices”. Em: Proc. of REV International Conference, Porto/Portugal. 2007.

[9] Youngblut, C., "Educational Uses of Virtual Reality Technology". Technical Report IDA Document D-2128, Institute for Defense Analyses, Alexandria, VA, 1998.

[10] Monferrer, A., Bonyuet, D., “Cooperative Robot Teleoperation through VR Interfaces”. *Proc. Sixth Int. Conf.on Information Visualization* (IV'02), 2002.

[11] Müller, D., “MARVEL- Mechatronics Training in Real and Vritual Environments. Concepts, Pratices, and Recommendations”, Marvel. 2005.

A Framework to Generate HLA compliant Visualization Federates through Web Services and X3D

Regina B. Araujo[1], Azzedine Boukerche[2], Ednaldo Pizzolato[1], Fabio M. Iwasaki[1]
[1]CS, Federal University of S. Carlos, SP, Brazil
{regina}, {ednaldo}, {fmiwasaki}@ dc.ufscar.br
[2]SITE, University of Ottawa, Ottawa, Canada
boukerch @ site.uottawa.ca

Abstract

This paper presents an open standard and web-based framework that generates HLA compliant simulation visualization interfaces with support to control and management functionalities. Differently from other existing approaches, our framework can provide greater flexibility for customization and deployment of HLA-based simulations in different hardware and software platforms.

1. Introduction

The modeling and visualization of simulations, particularly those aimed at training situations, such as emergency actions by Special Forces (e.g., police and fire fighters) are not trivial to build, manage and control. By having control over a simulation, in real-time, a specialist user, such as a fire fighter commander, can add and/or remove existing objects in the simulation, as well as change existing object properties, making the application a flexible environment to train human capabilities on different equipments and situations. Management, on the other hand, can provide important statistics on the performance of humans and equipments in different simulation runs. This paper presents an open standard, web3D-based framework, which can generate visualization interfaces to HLA (High Level Architecture) compliant simulations, with support to control and management functionalities.

Similar approaches to 3D visualization of HLA-based simulations were proposed using Java3D, most of them with their own protocol for communication. However, since the time they were presented, Web Services technology became a better alternative. Web Services use XML formatted messages to exchange data using primarily HTTP and HTTPS and are less prone to security-related problems. Moreover the solutions mentioned above do not deal with simulation control and management but only visualization.

For visualization, instead of using Java3D and having the 3D content being compiled along with the Java application or applet, X3D allows the content to be written in a text file that can be changed without the need to recompile the application. Web Services and X3D standards can provide greater flexibility for the customization of the functionalities of an HLA-based visualization application. Moreover, they can provide simulation deployment in different hardware and software platforms.

The paper is organized as follows: Section 2 gives a brief review on HLA and web services showing how they relate to our solution. X3D is briefly reviewed on section 3. Our framework for simulation visualization, control and management is described in Section 4, followed by Conclusions and References.

2. High Level Architecture and Web Services

The framework presented in this paper is based on the IEEE standard High Level Architecture (HLA) for distributed simulations [1]. HLA-based simulations can interoperate and have its components reused. An HLA compliant simulation is referred to as a **federate**, and a **federation** is defined as a set of federates working together. The HLA has three main components: the HLA rules, the HLA federate interface specification and the HLA object model template (OMT). The HLA rules define the responsibilities federates and federations must hold. The interface specification defines the standard services and interfaces of an underlying software architecture, the **Runtime Infrastructure (RTI)**, which supports the data exchange among federates. These interfaces are arranged in 7 service groups: federation management, declaration management, object management, ownership management, time management, data distribution management and support services.

Web Services can be used to implement an architecture that meet the requirements of Service Oriented Architecture, such as interoperability between different systems, clear and unambiguous description language

and retrieval of services. Web Services and HLA can be used to integrate heterogeneous simulations and international efforts have been made towards the provision of standardized Web Service interfaces independent of the RTI implementation (HLA WSDL API) [2][3][4]. This approach eliminates the need for customized software to route the calls from Web Services and a vendor specific RTI implementation. Even though the underlying work is still the same, it is transparent to federates and reusable by other simulations. Next section describes our solution for the creation of HLA-based simulation visualization for control and management that uses web services and X3D for greater flexibility.

3. HLA Compliant Simulation Visualization

There are three techniques for simulation visualization (with varying degree of integration between the simulation control and its visualization): post-processing, tracking and steering [5]. In post-processing technique, the resulting images from the visualization are generated after the simulation ends. In tracking, it is possible to view the current internal state of an executing simulation but not to change it. In the steering technique, the user has control over the simulation parameters and can change them at run time – this is the technique supported by our framework. In a training simulation, the user's inputs are continuously changing his/her corresponding virtual environment states. To interact with a simulation, it is necessary to define a strategy to further refine its visualization [6]. The visualization of a training simulation can be more or less effective depending on its specification and the user's perception [7]. The user can have a richer experience if the visualization specification can be changed during the simulation execution or customized for the user's level of expertise. For this reason, this framework uses the X3D standard [8], VRML successor, to render the simulated environment and user interface. Next section presents how X3D is used in our framework.

3.1 Using Extensible 3D for Visualization

Extensible 3D (X3D) is an ISO standard for real-time 3D graphics that defines a file format using XML and a run-time environment to be embedded in applications. The use of X3D features in our framework allows a wide variety of customization of the visualization client interface. X3D base components are sets of predefined nodes with some functionality. Those nodes are used to describe 3D objects and their relationships in a scene. Unlike Java3D or vendor-specific visualization software for HLA that must be compiled with the application, X3D is interpreted dynamically from human-readable text files. Also, many X3D content authoring software are being released with the support of well known modeling tools like Blender, for example. In order to create an X3D scene from a HLA simulation, the following base nodes are used to generate a basic X3D file:

- Prototype node: a new node type defined in terms of built-in or other prototype nodes by using the PROTO statement.
- Script node: a node used to program a behavior in the scene.
- Sensor node: a node that detects user inputs.
- Transform node: a node to position an object in the space.
- Inline node: a node that embeds a scene from another X3D file.

A mapping between the HLA object and interaction classes to X3D prototype nodes is performed and described in [9]. The prototype is named after the object or interaction class from the HLA/Federation Object Model - FOM (whose main function is to specify, in a common standardized format, the nature of the data exchanged among federates). Its attributes or parameters are linked from the prototype interface fields into a script node fields. Within the script node, function stubs are generated for the attributes or parameters converted to fields. Each prototype can link to other X3D files localized through URL using the inline node. By linking other files, the scene graph they define can be reused, such as the geometric models, animation scripts, user interface etc. Thus, for the same simulation, the visualization federate can have multiple interfaces with varying graphics level of details and specific interface for different users. The description of the framework is presented in the next section.

4. A Framework to Generate Vizualization Federates

Our framework generates visualization federates from a FOM and establishes communication with the RTI through Web Services. The federate is a web application and is ran within the context of a web browser. The application architecture is shown in Figure 1, and is divided into three layers:

- The communication layer uses the HLA WSDL API to exchange data with a federation through the RTI services;

- The application layer is the link between the communication and the presentation layers (Figure 2) and is generated by the framework based on a FOM;
- The presentation layer embeds the 3D browser into the federate and presents the data from the federation.

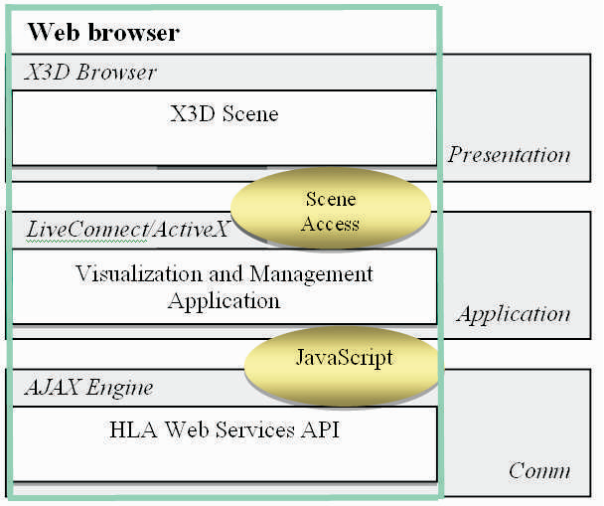


Figure 1. Application Architecture Overview.

This layered architecture supports different X3D browsers (such as Flux Player [10] or the Xj3D toolkit [11]) and different HLA APIs by defining an implementation-specific middleware for web-based or standalone federate deployment. By embedding the visualization application into a web page, the simulation data can be presented in two ways: In the X3D scene; and using HTML document elements such as tables.

Also, when the application is embedded in an HTML document, the AJAX [12] techniques are used to make Web Service calls and exchange data within the application layers. The application written in Java is an applet that exchanges function calls with JavaScript using the LiveConnect or ActiveX technologies.

4.1 Application Layer Functionalities Description

The framework defines functions stubs that must be implemented to attend to federation-specific requirements a federate must meet, such as subscription and publication of data, synchronization points and time management policies, for example. Some functionality for object discovery and update are default for all classes but this can be changed: when a new object instance is discovered from the federation and notified to the Federate Ambassador (responsible for handling all outgoing information passed from the user simulation to the RTI), the corresponding prototype node is instanced into the scene and its handle and reference are kept in the application layer. The handle and reference are used to find and update the object attributes either in the federation or in the scene.

Interactions can be sent by triggering sensor nodes and object attributes can be changed through script nodes. Listener for field change events from the script nodes are generated in the application layer. Those listeners are callback functions that will request the desired service to the RTI Ambassador to send interactions or update object attributes.

A performance evaluation of the prototype was carried out in two different scenarios (Core 2 Duo 2.4GHz processor with 4GB RAM with a GeForce 7950 GT KO 512MB; Pentium 4 HT 3.2GHz with 2GB RAM with a GeForce 6800 XT 256MB). The simulation was hosted locally for both machines, receiving constant updates on position and other attributes of around 50 simulation-driven objects. In both platforms, the frame rate was kept over 15 FPS with low polygon count models. Further data analysis is needed though the update rates of data sent by each federate varies depending on the simulation model.

Performance will depend mainly on browser implementation, and to some extent, it will depend also on the number of scripts being performed. In case performance falls under acceptable response time, the 3D models being used can be replaced by less complex models. More complex simulations are under implementation to be tested in a LAN environment.

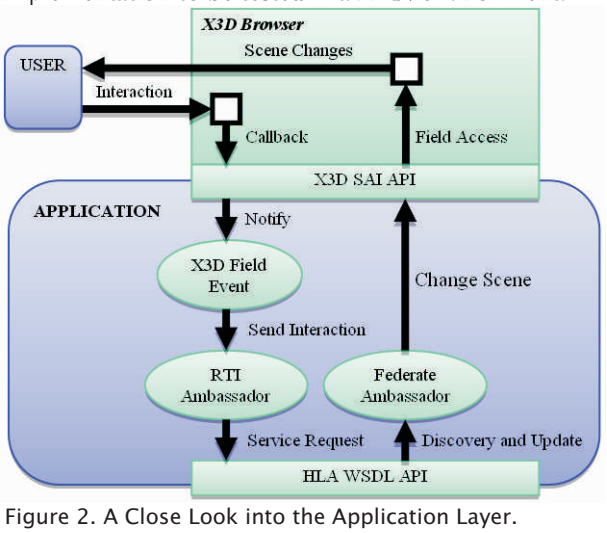


Figure 2. A Close Look into the Application Layer.

5. Conclusions

This paper presents an HLA-based framework that generates HLA-based visualization interfaces with support to simulation control and management. Differently from other existing approaches, our framework can provide greater flexibility for customization and deployment of HLA-based simulations in different hardware and software platforms, by making use of web services and X3D technologies. A performance evaluation of the prototype was made in two different platforms, achieving a 15 fps rate in a scenario with low complexity geometric models at constant update. The tests show that our solution can be a powerful tool for the generation of visualization federates, in which control and management can be done at simulation run time. A graphic tool is being built to make it easier to use our framework to build visualization federates.

6. Acknowledgements

This work was supported by São Paulo State Research Support Foundation, FAPESP, under Process 2006/00741-7 and CAPES.

7. References

[1] IEEE: "IEEE 1516, High Level Architecture (HLA)", www.ieee.org, March 2001.

[2] Doug Barry: “Web Services and Service Oriented Architectures”, www.service-architecture.com.

[3] J.M. Pullen et al., "Using Web Services to Integrate Heterogeneous Simulations in a Grid Environment," *Proc. Workshop HLA-Based Distributed Simulation on the Grid, LNCS 3038*, Springer-Verlag, 2004, pp. 835-847.

[4] Björn Möller, Clarence Dahlin: “A first look at the HLA Evolved Web Service API”, *Proceedings of the 2006 European Simulation Interoperability Workshop*, June 2006.

[5] Marshall, R., “Visualization Methods and Simulation Steering for a 3D Turbulence Model of Lake Erie”, *Proceedings of the 1990 symposium on Interactive 3D graphics*, pp.89-97, 1990.

[6] Johnson, C, Parker, S. G., Hansen, C., Kindlmann, G. L., Livnat, Y., “Interactive Simulation and Visualization”, *Computer*, vol.32, no.12, pp.59-65, December 1999.

[7] van Wijk, J.J., Eindhoven, T.U., “The Value of Visualization”, vis, p. 11, *16th IEEE Visualization 2005 (VIS 2005)*, 2005.

[8] Web 3D Consortium: “ISO/IEC 19775:2004, X3D Abstract”, www.web3d.org, November 2005.

[9] Araujo, R.B., Iwasaki, F.M., Pizzolato, E., Boukerche, A. Creating HLA Compliant Simulations for Visualization in Heterogeneous Devices. *Submitted to the 13th Intl. Symposium on 3D Web Technology*, LA, USA.

[10] Media Machines, www.mediamachines.com, last access in September 2007.

[11] The Xj3D Project, www.xj3d.org, last access in September 2007.

[12] Garret, J.J., “Ajax: A New Approach to Web Applications”, www.adaptivepath.com/publications/essays/archives/000385.php, February 2005.

Session 9: Educating People, Bots and Monkeys

Simulação Virtual da Evolução de Estratégias e do Controle Inteligente em Sistemas Multi-Robóticos

Gustavo Pessin, Fernando Osório
Unisinos – PIPCA - PPG em Computação Aplicada
Av. Unisinos, 950
São Leopoldo, RS - Brasil
{pessin}, {fosorio}@gmail.com

Soraia Musse
PUC-RS – FACIN - Faculdade de Informática
Av. Ipiranga, 6681
Porto Alegre, RS - Brasil
soraia.musse@pucrs.br

Resumo

O objetivo deste artigo é detalhar o projeto e o desenvolvimento de um sistema multi-agente que opera em um ambiente virtual de simulação realística. Neste sistema, uma equipe de agentes autônomos trabalha cooperativamente a fim de realizar com sucesso a identificação e o combate de incêndios em áreas florestais, sem intervenção humana. O ambiente suporta uma série de características fundamentais para a simulação realística da operação, como terrenos irregulares, processos naturais e restrições físicas na criação e uso de robôs móveis. A operação multi-agente depende essencialmente de duas etapas: planejamento e ação. No planejamento, usamos Algoritmos Genéticos para evoluir estratégias de posicionamento de atuação dos robôs bombeiros. Para a ação foram criados robôs de combate fisicamente simulados, sendo as informações sensoriais de cada robô (e.g. GPS, bússola, sonar) usadas na entrada de uma Rede Neural Artificial (RNA). Esta RNA que controla os atuadores do veículo permite navegação com desvio de obstáculos. Os resultados das simulações demonstram que a Rede Neural controla satisfatoriamente os robôs móveis, que o uso de Algoritmos Genéticos configura a estratégia de combate ao incêndio de modo satisfatório e que o sistema multi-agente proposto pode vir a ter um papel muito importante no planejamento e execução de operações reais de combate a incêndios florestais.

1. Introdução

Com a evolução das pesquisas em robótica, cada vez mais os robôs estão se tornando complexos em termos físicos. A grande variedade de estudos em morfologia robótica tem desenvolvido variações de robôs dotados de diversos meios de locomoção (e.g. pernas, rodas, esteiras). Em paralelo a este desenvolvimento temos a evolução constante de uma gama extremamente grande de sensores (e.g. sistemas de visualização, posicionamento, detecção de obstáculos). O desenvolvimento de algoritmos e técnicas para coordenar estes conjuntos físicos em um ambiente dinâmico é um desafio extremamente complexo [3]. Dotar robôs autônomos de capacidade de raciocínio inteligente e de interação com o meio em que estão inseridos é uma área de pesquisa que tem atraído a atenção de um grande número de pesquisadores [2]. O uso de ambientes de realidade virtual torna a pesquisa em robótica muito mais ágil do que a pesquisa usando robôs reais.

No trabalho [18] propomos a idéia de um ambiente de simulação virtual para a identificação e combate de incêndios florestais, com agentes controlados por regras. Neste artigo apresentamos a evolução do ambiente, que foi remodelado usando a biblioteca de simulação física Open Dynamics Engine (ODE) e a aplicação de técnicas inteligentes nos agentes.

Existem diversas áreas onde a habilidade de um único agente não é suficiente ou eficiente para a realização de uma tarefa, em alguns destes casos, como patrulhamento, vigilância, resgate ou exploração o mais indicado é a aplicação de sistemas multi-robóticos. Sistemas multi-robóticos são sistemas onde robôs autônomos trabalham cooperativamente a fim de cumprir uma missão, podendo existir interação entre os robôs ou não [4].

Um problema fundamental da robótica é a navegação. O deslocamento de um lugar para outro depende de três aspectos fundamentais: localização, orientação e controle motor. Para conhecer tanto sua localização como sua orientação, um robô móvel deve possuir sensores próprios (e.g. GPS, bússola). Para o controle motor, deve possuir um número adequado de motores (um veículo autônomo, por exemplo, usualmente possui um motor angular, para giro das rodas e um motor linear, para tração). Normalmente sensores e atuadores são sujeitos a erros e interferências, assim o controle das ações de um robô deve sempre levar em conta a imprecisão dos sensores e motores envolvidos. Um

[1] Código-fonte disponível em http://pessin.googlepages.com

sistema robusto deve permitir que, mesmo com sensores e atuadores imprecisos, o agente cumpra o seu objetivo. Uma técnica de Aprendizado de Máquina indicada para este controle é a de Redes Neurais Artificiais, dada sua capacidade de aprendizado a partir de exemplos e a respectiva generalização e adaptação das saídas. É uma técnica muito utilizada no controle de navegação de sistemas reativos [34, 43].

Na tarefa de controle de incêndio, uma das questões mais importantes tem relação com a formação do posicionamento dos agentes para atuação no combate. De acordo com as capacidades de atuação de cada agente, as condições climáticas (vento, chuva), a topografia e a vegetação, diferentes formações (Figura 1) podem ser sugeridas. Estas formações, quando sugeridas por um especialista, podem não levar em conta um número muito grande de variáveis, assim, a definição do posicionamento de formação poderia fazer uso de técnicas de Aprendizado de Máquina (*Machine Learning* - ML). Uma técnica de ML indicada para estes casos é a de Algoritmos Genéticos (*Genetic Algorithms* - GA) [7, 8], que são algoritmos de otimização global que empregam estratégia de busca paralela e estruturada, embora aleatória, direcionada a busca de pontos de aptidão [9], permitem assim a realização de busca multi-critério em um espaço multidimensional e por serem não supervisionados, não necessitam de nenhuma base de informação de antemão, e se corretamente utilizados são capazes de escapar de mínimos locais [26].

Figura 1. Formações para uma equipe de quatro agentes (da esquerda para a direita: linha, coluna, circular ou ferradura) [1].

Para que seja possível a implementação física real, o sistema multi-agente que propomos deve ser projetado, desenvolvido e testado anteriormente em ambientes de simulação realísticos. Ambientes com terrenos 3D gerados com a biblioteca de programação Open Scene Graph (OSG) [28] junto com a biblioteca de programação Demeter [29] permitem combinar o mapa de elevação juntamente com uma determinada distribuição de vegetação, criando assim um terreno bastante realístico [30]. Além disso, o uso da biblioteca de programação Open Dynamics Engine (ODE) [27] permite implementar atributos físicos (e.g. atrito, fricção, gravidade, colisão) tornando o sistema ainda mais realístico.

Neste trabalho, um grupo de agentes autônomos trabalha cooperativamente a fim de realizar com sucesso a identificação e o combate de incêndios em áreas florestais, sem intervenção humana. São simulados diferentes tipos de agentes, com níveis hierárquicos diferentes, assim, cada etapa do processo conta com agentes específicos que possuem um conjunto de ações determinadas. Em nossos estudos e pesquisas, pelo que pudemos constatar, não existe um ambiente de simulação aberto que combine todas as características requisitadas para este trabalho, que são: *(i)* Realismo físico de modelagem robótica; *(ii)* Realismo físico de interação de agentes em terrenos irregulares; *(iii)* Facilidades de comunicação entre os agentes; (iv) Capacidade de aplicação de métodos de controle (e.g. Redes Neurais Artificiais); (v) Capacidade de aplicação de métodos de evolução de estratégias (e.g. Algoritmos Genéticos); (vi) Simulação de processos naturais (como a propagação do fogo).

Nossas principais metas neste projeto são: (i) Recolher informações sobre dados florestais, tipos de vegetação, topografia, e comportamento de incêndios para criar o ambiente virtual mais realista possível; *(ii)* Simular incêndios em florestas, reproduzindo de forma bastante realista o ambiente e a propagação dos focos de incêndio; *(iii)* Pesquisar ferramentas e técnicas de combate à incêndios florestais utilizadas por bombeiros; (iv) Implementar a simulação de agentes móveis autônomos colaborativos capazes de formar uma brigada de combate a incêndios; (v) Estudar métodos de aprendizado de máquina e suas vantagens para o modelo; e (vi) Estudar a robustez das ações dos agentes pela leitura de dados de sensores sujeitos a erros. Como resultado principal, o ambiente virtual deve permitir ajudar no projeto e validação dos robôs reais a serem construídos (tanto morfologia como técnicas de controle) e na validação da operação multi-agente de identificação e combate de incêndios. O sucesso desta tarefa envolve o uso de uma variedade de tecnologias de diferentes campos, assim, a construção deste sistema é um bom estudo para avaliar eficiência de arquitetura multi-agente, estratégia cooperativa, fusão de sensores e modelagem robótica. A tarefa em si poderá ser generalizada para outras atividades práticas como acidentes nucleares ou desastres ambientais. Com técnicas de aprendizado de máquina esperamos obter agentes que suportem melhor um ambiente dinâmico, fazendo assim o sistema mais flexível e autônomo possível.

Neste artigo apresentamos na Seção 2 uma pequena conceituação teórica de agentes e sistemas multi-agentes. A Seção 3 apresenta conceitos de Aprendizado

de Máquina e das técnicas de IA escolhidas. Na Seção 4 detalhamos características de robótica móvel e conceitos de simulação e modelagem, bem como as bibliotecas selecionadas para o desenvolvimento do trabalho. Na Seção 5 descrevemos técnicas e operações reais de identificação e combate a incêndios florestais. Na Seção 6 descrevemos o ambiente desenvolvido, a operação multi-agente de identificação e combate, o desenvolvimento e a aplicação da Rede Neural Artificial e o desenvolvimento e a aplicação do Algoritmo Genético. Finalizamos apresentando a conclusão do trabalho realizado.

2. Agentes e Sistemas Multi-Agentes

Uma das primeiras e mais simples definições de agentes inteligentes é de que são sistemas computacionais que habitam um dado ambiente, sentem e agem autonomamente nesse ambiente [25]. Segundo [40] um agente de software pode ser visto como um sistema dinâmico, onde a percepção e a ação constituem processos simultâneos e inseparáveis. Em termos gerais, ambiente é onde um agente ou um conjunto de agentes está inserido, pode ser físico (como ambientes onde estão inseridos robôs), de software ou de realidade virtual (onde se faz simulação do ambiente físico) [38].

De acordo com [40] o projeto de um agente de software envolve a definição de três componentes que são rigorosamente interconectados e mutuamente interdependentes: *(i)* Definição do ambiente para atuação; *(ii)* Estabelecimento de comportamentos e tarefas a serem realizadas; *(iii)* Determinação da morfologia do agente. A concretização do projeto de um agente robótico deve priorizar quatro aspectos principais [41]: *(i)* Autonomia: Deve ser capaz de operar com o mínimo possível de intervenção, supervisão e instrução humana; *(ii)* Auto-suficiência: Deve ser capaz de operar por tempo prolongado, ou, pelo menos, suficiente para realizar a tarefa e buscar recarga de energia; *(iii)* Localização: Deve ser capaz de adquirir informação sobre o ambiente somente através de seus próprios sensores; (iv) Corporificação: Deve ser desenvolvida como um sistema físico ou computacional, porém a corporificação não se dá necessariamente pela materialização, como a apresentada em animais ou robôs físicos, mas por uma relação dinâmica com o ambiente, ou seja, a corporificação pode ser realizada em ambientes de simulação computacionais, desde que estes ambientes sejam realísticos do ponto de vista físico [41].

Sistemas multi-agentes são sistemas constituídos de múltiplos agentes que interagem ou trabalham em conjunto de forma a realizar um determinado conjunto de tarefas ou objetivos. Um sistema multi-agente pode ser visto como uma rede, fracamente acoplada, de solucionadores de problemas que trabalham em conjunto para resolver problemas que vão além da sua capacidade individual. Estes solucionadores de problemas são essencialmente autônomos e, muitas vezes, heterogêneos em sua natureza [36]. Um sistema multi-agente deve ter alguma forma de controle. A utilização de ambientes dinâmicos, robôs com mau funcionamento, múltiplas regras e restrições em comportamentos individuais adicionam complexidade ao problema de controle [4].

Cooperação pode ser entendida como a capacidade que os agentes têm de trabalhar em conjunto de forma a concluírem tarefas de interesse comum [31]. Para que exista cooperação, deve existir alguma forma de interação entre os agentes, ou com humanos, através de alguma forma de comunicação [19]. A cooperação não depende obrigatoriamente de comunicação entre todos os agentes, visto que, como no mundo real, um coordenador (unidade de controle) pode delegar tarefas (enviar mensagens) para um grupo de pessoas (agentes) que não tenham comunicação direta entre si. Neste caso, as pessoas podem trabalham em prol do mesmo objetivo sem ter conhecimento do que as outras pessoas estão fazendo. A morfologia de um robô deve, ao incluir a capacidade de comunicação, possuir um módulo de controle de comunicações, assim, deve possuir componentes de percepção (recepção de mensagens) e de ação (envio de mensagens). O módulo de comunicação está diretamente ligado ao módulo de controle do agente [31].

3. Aprendizado de Máquina

Aprendizado de Máquina (Machine Learning - ML) é uma área da Inteligência Artificial que tem como objetivo desenvolver técnicas computacionais de aprendizado e de aquisição de conhecimentos [5]. Essas técnicas devem exibir um comportamento inteligente e realizar tarefas complexas com um nível de competência equivalente ou superior ao de um especialista humano [6]. Para a construção do sistema proposto neste artigo utilizamos Redes Neurais Artificiais e Algoritmos Genéticos, assim, descrevemos sucintamente suas características a seguir.

3.1. Redes Neurais Artificiais

Redes Neurais Artificiais (RNA) são sistemas paralelos distribuídos, compostos por unidades de processamento simples que calculam determinadas funções matemáticas, normalmente não-lineares [9].

Seus atributos básicos podem ser divididos em arquitetura e neurodinâmica. A arquitetura determina a estrutura da rede, ou seja, o número de neurônios e sua interconectividade e a neurodinâmica, por sua vez, define as propriedades funcionais da rede, ou seja, como ela aprende, recupera, associa e compara novas informações com o conhecimento já armazenado [37]. Matematicamente, RNAs são aproximadores universais, que realizam mapeamentos em espaços de funções multi-variáveis [32]. A capacidade de aprender e generalizar das RNAs é um dos seus maiores atrativos. O processamento da informação em uma RNA é feito por meio de estruturas neurais artificiais [5], sendo que esta estrutura, bem como o próprio neurônio artificial são uma analogia biológica ao funcionamento do cérebro.

O algoritmo Backpropagation [42] é um algoritmo supervisionado de aprendizado de RNA. A base de treinamento é um conjunto de dados que deve apresentar, para cada entrada, a saída prevista do sistema. Este tipo de aprendizado ocorre em várias épocas, sendo que cada época representa a apresentação do conjunto inteiro de dados à RNA para o ajuste dos pesos. O treinamento de uma RNA deve envolver várias rodadas de simulação, iniciando os pesos de forma aleatória. Outra questão importante é o grau de generalização, sendo medido usualmente através do uso de uma base de validação usada em paralelo a base de treinamento.

O Stuttgart Neural Network Simulator (SNNS) [10] é um simulador de Redes Neurais Artificiais criado no Institute for Parallel and Distributed High Performance Systems (IPVR) da Universidade de Stuttgart. O SNNS possui um grande número de algoritmos de aprendizado (e.g. *Backpropagation, Quickprop, RProp*). Um aplicativo do pacote SNNS, o SNNS2C, permite a conversão de uma RNA em código em C, que pode ser inserido em uma outra aplicação.

3.2. Algoritmos Genéticos

Algoritmos Genéticos (*Genetic Algorithms* - GA) [7, 8] são técnicas de otimização global que empregam estratégia de busca paralela e estruturada, embora aleatória, direcionada a busca de pontos de aptidão [9]. Devido a ser uma técnica estocástica, é classificada como não supervisionado, visto que não necessita de nenhuma base de informação de antemão. Os GA utilizam procedimentos iterativos que simulam o processo de evolução de uma população de possíveis soluções de um determinado problema. Durante o processo evolutivo, cada individuo da população é avaliado através de uma função de aptidão (*fitness*) que permite mensurar o quanto cada indivíduo está apto a resolver o problema. Nos indivíduos selecionados, é feito crossover e mutação, gerando descendentes para a próxima geração [9,12,13]. O crossover e a mutação transformam uma população através de sucessivas gerações e possibilitam assim que o GA percorra os melhores pontos do espaço de busca até chegar a um resultado satisfatório [11].

A GAlib (http://lancet.mit.edu/ga) é uma biblioteca de software livre desenvolvida por Matthew Wall do Massachusetts Institute of Technology (MIT) em C++ que contém um conjunto bastante amplo de funções relacionadas a programação de Algoritmos Genéticos. É uma das mais completas e eficientes bibliotecas de software para a simulação de Algoritmos Genéticos, permitindo experiências com diferentes funções objetivo (fitness), representações genéticas, operadores genéticos e métodos de seleção.

4. Robótica Móvel

Um robô móvel é um dispositivo mecânico montado sobre uma base não fixa que age sob o controle de um sistema computacional, equipado com sensores e atuadores que o permitem interagir com o ambiente [34,39]. A interação com o ambiente se dá através de ciclos de percepção-ação que consistem em três passos fundamentais: *(i)* Obtenção de informação através de sensores; *(ii)* Processamento das informações para seleção de ação; e, *(iii)* Execução da ação através do acionamento dos atuadores. Esse conjunto de operações, em uma análise superficial, pode parecer simples, porém o controle robusto de sistemas robóticos tem complicações físicas, mecânicas, eletrônicas e computacionais que tornam a criação de um conjunto de regras uma tarefa árdua e sujeita a erros. Sensores são os mecanismos de percepção de um robô e realizam medições físicas (e.g. contato, distância, orientação, temperatura), provêm sinais ou dados crus que precisam ser interpretados pelo "cérebro" do robô. A interpretação destes sinais deve ser a única maneira de um robô autônomo entender o ambiente que o cerca para poder realizar as mudanças de ação necessárias [35]. Atuadores são os mecanismos de ação de um robô (e.g. motores, pistões, braços manipuladores), controlados por circuitos eletrônicos que recebem valores de ação, valores estes que devem ser calculados pelo "cérebro" do robô e devem estar de acordo com especificações de fabricantes [35].

4.1. Simulação e Modelagem

Experimentos em robótica móvel podem ser realizados de duas formas: diretos em um robô real ou em um robô simulado em um ambiente virtual realístico [20].

Usualmente, experimentos em robótica móvel utilizando um robô real demandam enorme despendimento de tempo e de recursos financeiros. Para que seja possível a implementação física real, o sistema multi-agente que propomos deve ser projetado, desenvolvido e testado anteriormente em ambientes de simulação realísticos. A simulação de siste

Usualmente, experimentos em robótica móvel utilizando um robô real demandam enorme despendimento de tempo e de recursos financeiros. Para que seja possível a implementação física real, o sistema multi-agente que propomos deve ser projetado, desenvolvido e testado anteriormente em ambientes de simulação realísticos. A simulação de sistemas robóticos é especialmente necessária para robôs caros, grandes, ou frágeis [3], sendo uma ferramenta extremamente poderosa para agilizar o ciclo de desenvolvimento de sistemas de controle robóticos eliminando desperdício de recursos, tanto financeiros como computacionais. Para que uma simulação seja útil, entretanto, ele deve capturar características importantes do mundo físico, onde o termo importantes tem relação ao problema em questão [3]. No caso deste trabalho, é fundamental que existam restrições físicas no modelo e que exista a possibilidade de trabalho em um terreno irregular, provido de obstáculos. Para o desenvolvimento deste trabalho foram pesquisadas algumas ferramentas de simulação, como o Microsoft Robotics Studio (msdn.microsoft.com/robotics), o Webots (www.cyberbotics.com), o Khepera Simulator (diwww.epfl.ch/lami/team/michel/khep-sim), o Mission Simulation Facility (ase.arc.nasa.gov/msf), o JUICE (www.natew.com/juice) e o Simulator BOB (tu-harburg.de/ti6/mitarbeiter/pst/Sim). No entanto, optamos pela criação de um simulador próprio que reproduzisse o mundo real em um ambiente virtual 3D com realismo físico. Como optamos por desenvolver nosso próprio ambiente, todas as bibliotecas de programação selecionadas para o desenvolvimento do ambiente e dos agentes robóticos são software livre, multi-plataforma e em linguagem C/C++. Uma pequena descrição de cada uma das bibliotecas utilizadas é fornecida a seguir: *(i)* Open Dynamics Engine (ODE) [27] é uma biblioteca desenvolvida para a simulação física de corpos rígidos articulados [33]. Uma estrutura articulada é criada quando corpos rígidos de vários tipos são conectados por algum tipo de articulação, como, por exemplo, um veículo terrestre que tem a conexão de rodas em um chassi. A ODE foi projetada para ser utilizada de modo interativo em simulações de tempo real e é especialmente indicada para a simulação de objetos móveis em ambientes dinâmicos. A ODE não tem como objetivo realizar simulações de outras dinâmicas além da dinâmica de corpos rígidos (e.g. partículas, roupas, ondas, fluidos, corpos flexíveis, fraturas). A ODE possui contatos rígidos, assim, quando dois corpos colidem não há penetração. O sistema de detecção de colisão é nativo e suporta diversas primitivas (e.g. esfera, caixa, cilindro, plano, raio). A utilização da ODE no nosso ambiente éfundamental por fornecer restrições físicas, principalmente na definição da morfologia dos robôs; *(ii) Open Scene Graph* (OSG) [28] é uma biblioteca para desenvolvimento de aplicações gráficas 3D de alta performance. Baseada no conceito de grafos de cena, provê ao desenvolvedor um ambiente orientado a objeto sobre a *OpenGL* (www.opengl.org), liberando este da necessidade de implementação e otimização de chamadas gráficas de baixo nível; *(iii) DrawStuff* [27] é um ambiente de visualização de objetos 3D que tem o propósito de permitir a demonstração visual da ODE sendo uma biblioteca bastante simples e rápida para utilização; (iv) *Demeter* [29] é uma biblioteca desenvolvida para renderizar terrenos 3D, desenvolvida para ter rápida performance e boa qualidade visual, pode renderizar grandes terrenos em tempo-real sem necessidade de hardware especial, depende da *Simple DirectMedia Layer* (www.libsdl.org) para realizar o tratamento das texturas do terreno e da *Geospatial Data Abstraction Library (*www.gdal.org) para carregar arquivos de elevação.

5. Incêndios em Ambientes Naturais

A fim de melhor entender como proceder no combate a incêndios florestais e assim planejar as estratégias a serem implementadas nos agentes autônomos, foi realizado um estudo sobre técnicas reais de operação. Este estudo teve como base os trabalhos de [14, 15, 16, 21, 22]. Para a implementação da propagação do fogo obtivemos de [17] medições reais de velocidades. O estudo dos modelos de florestas e resíduos florestais é de grande importância para o aprimoramento dos modelos de simulação a serem implementados em ambientes virtuais [18].

Os combustíveis florestais, que são produto da cobertura vegetal e sua dinâmica, são importantes parâmetros nos processos de ignição e propagação dos incêndios. O conhecimento das características básicas dos combustíveis (e.g. tipo, quantidade, continuidade, arranjo) é muito útil na previsão do comportamento dos incêndios [23]. Quanto aos tipos de combustíveis florestais, sugere-se os seguintes modelos [24]: Modelos 1, 2 e 3 são do tipo herbáceo, sendo pasto fino, contínuo, e contínuo espesso respectivamente, os

incêndios propagam-se com grande velocidade; Modelos 4, 5, 6 e 7 são do tipo arbustivo, indo de árvores jovens até mato denso; Modelos 8, 9 e 10 são do tipo manta morta; Modelos 11 e 12 são do tipo resíduos lenhosos. Cada um destes modelos possui velocidade de propagação de fogo específica.

O estudo das técnicas reais de operação permite que se possa planejar melhor o formato da operação e as estratégias a serem implementadas no sistema. O conjunto das técnicas listadas a seguir é de consenso geral para aplicações em monitoramento e combate a incêndios. A operação de combate ou supressão de um incêndio envolve seis etapas distintas [21]: *(i)* Detecção; *(ii)* Comunicação; *(iii)* Mobilização; *(iv)* Deslocamento; *(v)* Planejamento; e *(vi)* Combate. Existem quatro métodos de combate ao fogo nos incêndios florestais [21]: *(i)* Método direto: usado quando a intensidade do fogo permite uma aproximação suficiente da brigada à linha de fogo, são usadas as seguintes técnicas e materiais: água (bombas costais, baldes ou moto-bombas); terra (pás); ou batidas (abafadores); *(ii)* Método paralelo ou intermediário: usado quando não é possível o método direto e a intensidade do fogo não é muito grande, consiste em limpar, com ferramentas manuais, uma estreita faixa, próxima ao fogo, para deter o seu avanço e possibilitar o ataque direto; *(iii)* Método indireto: usado em incêndios de intensidade muito grande, consiste em abrir aceiros com equipamento pesado (e.g. trator, motoniveladora); *(iv)* Método aéreo: usado nos incêndios de copa, de grande intensidade e área e em locais de difícil acesso às brigadas de incêndio, são usados aviões e helicópteros.

A rapidez e a eficiência na detecção e monitoramento dos incêndios florestais são fundamentais para a viabilização do controle do fogo, redução dos custos nas operações de combate e atenuação dos danos. Para países de grande extensão territorial, como o Brasil, o monitoramento dos incêndios florestais através de imagens de satélites é o meio mais eficiente e de baixo custo quando comparado com os demais meios de detecção [15]. O lançamento em 1972 do primeiro satélite Landsat possibilitou detectar alterações nas áreas florestais do espaço. Desde então, as imagens termais e de infravermelho têm sido usadas na detecção de incêndios e estudos de mapeamento, permitindo que áreas queimadas e não queimadas sejam detectadas através do contraste entre os gradientes térmicos [16].

6. Implementação do Protótipo

A operação multi-agente depende essencialmente de duas etapas, planejamento e ação, assim, foram implementados protótipos específicos para cada etapa. Ambos foram desenvolvidos em C++. O protótipo onde é realizada a ação dos robôs bombeiros, controlados por uma Rede Neural Artificial pode ser visto na Figura 2. Este protótipo tridimensional usa a biblioteca OSG, responsável pela saída gráfica do protótipo, a biblioteca Demeter, responsável pelo terreno irregular e a biblioteca ODE, responsável pelo realismo físico, tanto da morfologia robótica como da colisão entre os objetos presentes no ambiente (e.g. robôs, árvores, inclinação de terreno). O uso da biblioteca ODE permite que os robôs simulados fisicamente respeitem questões como gravidade, inércia e atrito. Por exemplo, mantendo uma força f constante nos motores lineares (torque) um veículo terá velocidade v em regiões planas, em regiões de declive terá v maior e em aclives terá v menor. O protótipo onde é realizado o planejamento usa um Algoritmo Genético. Este não necessita obrigatoriamente de visualização, porém está implementado com a possibilidade de uma saída gráfica 2D desenvolvida com SDL, como mostra a Figura 4(b). A integração entre os protótipos se dá através de um arquivo texto, após realizar a evolução, o protótipo responsável cria um arquivo com as posições de atuação que é lido na inicialização do protótipo de ação no combate.

Figura 2. Ambiente de simulação desenvolvido com OSG, ODE e Demeter.

Para a simulação da vegetação e correta propagação do fogo, existe uma matriz oculta sob o terreno. Esta matriz possui, para cada área do terreno, o tipo de vegetação presente, assim, considerando orientação do vento, intensidade do vento e tipo de vegetação de uma área podemos construir a simulação de propagação do fogo. A velocidade de propagação respeita dados do modelo retirados de [17]. Quanto ao vento, tanto a sua intensidade como a sua orientação podem ser geradas aleatoriamente ou configuradas a partir de dados parametrizados pelo usuário. O tempo de permanência do fogo em uma área é relacionado diretamente ao tipo da vegetação presente e se comporta baseado nos valores de tipo de vegetação, inclinação do terreno, intensidade e orientação do vento. Desta forma a propagação do fogo busca simular de modo bastante realístico a forma como o fogo se propagaria em um ambiente real. O comportamento do fogo é idêntico nos dois protótipos. Ambos os protótipos possuem um mapa que simula a integração das informações de vegetação, topografia e comportamento de fogo detalhados na Seção 5. A criação dos mapas teve como base cartas topográficas e o mapa de modelos de combustíveis florestais, que podem ser vistos em [18].

A comunicação entre os agentes robóticos existe para que alertem os atrasos na navegação, assim, o raio de atuação é recalculado a cada n fragmentos de tempo. A simulação desta comunicação faz uso de um blackboard. Além do alerta de atrasos, o sistema de comunicação simula a troca de mensagens entre o agente monitor e os agentes de combate. Existem casos de comunicação um para um e de um para todos. A fila usada como função de blackboard armazena as seguintes informações: indicador de remetente, indicador de destinatário, timestamp, e tipo da mensagem (e.g. aviso de incêndio, aviso de fim de incêndio, negociação de times, posição do incêndio).

No ambiente desenvolvido, cada árvore existe como um modelo OSG e como um cilindro ODE. Os sensores identificam os cilindros (Figura 3(a)) que são posicionados junto às árvores. O ambiente permite que, através de parâmetros, possamos habilitar ou não a exibição dos objetos físicos ao invés de apenas a sua representação gráfica.

6.1. Operação de Identificação e Combate

Usamos o ambiente para simular a seguinte operação: um agente monitor (satélite) monitora o terreno da área florestal, ao identificar uma área com foco de incêndio, ativa o módulo de evolução de estratégias (detalhado na Subseção 6.3). Após obter as coordenadas de atuação através do GA, o agente monitor envia mensagens aos agentes de combate informando suas posições de atuação (nesta implementação simulamos esta operação através de um arquivo texto). O comportamento dos agentes de combate é reativo, deslocando-se em direção a posição de seu objetivo específico desviando de obstáculos. O método de combate de incêndio simulado é o método indireto. Os agentes de combate simulados são motoniveladoras que tem como finalidade cercar o foco de incêndio e criar um aceiro (área livre de vegetação onde o fogo se extingue pela falta de combustível). Esta operação pode ser entendida com a Figura 4(a). Quanto ao controle de posicionamento, um sensor do tipo GPS é simulado em cada robô. Em experimentos realizados com um GPS Garmin Etrex (www.garmin.com) obtivemos um erro médio de 18,6 metros. Considerando que cada agente possui seu próprio GPS, o tratamento deste erro é crucial na criação dos aceiros [18]. Tratamos esta informação da seguinte maneira: o erro médio deste sensor durante o deslocamento é usado somado a distância de criação do aceiro e também somado ao final da área de criação.

6.2. Morfologia dos Robôs Móveis

Os robôs móveis foram desenvolvidos com a biblioteca de simulação de corpos rígidos articulados ODE. As Figuras 3(a) e 3(b) apresentam o veículo desenvolvido. Dada a existência de restrições físicas, a única maneira de controlar este veículo é com a aplicação de forças em seus dois motores simulados, que são: um motor angular (para o giro do volante) e um motor linear (para o torque). Além do GPS, responsável pela obtenção da localização, cada robô possui também uma bússola, necessária para a obtenção da orientação do veículo.

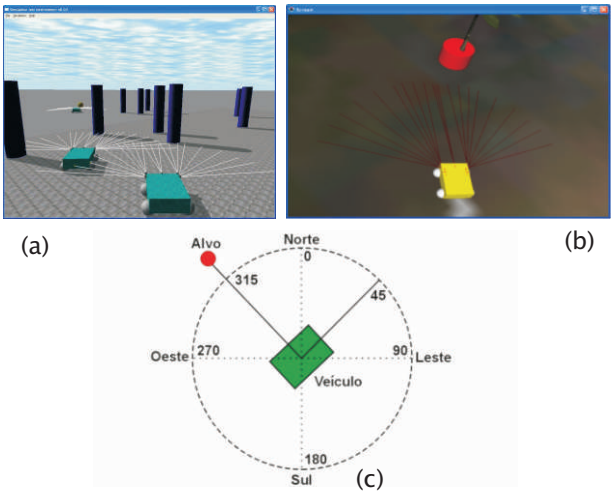


Figura 3. Sensores desenvolvidos: (a) e (b) Sensores de distância e (c) Bússola e GPS, responsáveis pela orientação, localização e obtenção do azimute.

O azimute (ângulo para o alvo) é obtido a partir da posição atual (GPS) e da posição do alvo (recebida por mensagem), como mostra a Figura 3(c). Os sensores de distância são sonares simulados, apresentando as características de capacidade de medir distâncias entre 15cm e 11m, como o Polaroid 6500 (www.senscomp.com).

6.3. Evolução de Estratégias de Posicionamento

O mecanismo de planejamento usa um GA para definir as posições inicias e finais de atuação de cada robô no combate ao incêndio. Considerando que os agentes de combate são motoniveladoras que tem como finalidade criar um aceiro, necessitamos que o GA retorne as seguintes informações: ângulo inicial e final e raio inicial e final para cada robô, ambos em relação ao ponto inicial do foco de incêndio. Esta operação pode ser entendida com a Figura 4. O genoma desenvolvido para as simulações iniciais pode ser visto na Tabela 1. Neste, estão presentes informações de todo o grupo dos agentes envolvidos, assim, o tamanho do genoma é dependente da quantidade de agentes no sistema. Realizamos simulações considerando a existência de 4 agentes de combate.

Gene	Função	Valor mínimo	Valor máximo
0	Posição inicial do agente 0	0,000°	360,000°
1	Posição final do agente 0 e inicial do agente 1	0,000°	360,000°
2	Posição final do agente 1 e inicial do agente 2	0,000°	360,000°
3	Posição final do agente 2 e inicial do agente 3	0,000°	360,000°
4	Posição final do agente 3	0,000°	360,000°
5	Raio inicial do agente 0	10,000m	100,000m
6	Raio final do agente 0 e inicial do agente 1	10,000m	100,000m
7	Raio final do agente 1 e inicial do agente 2	10,000m	100,000m
8	Raio final do agente 2 e inicial do agente 3	10,000m	100,000m
9	Raio final do agente 3	10,000m	100,000m

Tabela 1. Genoma dos indivíduos.

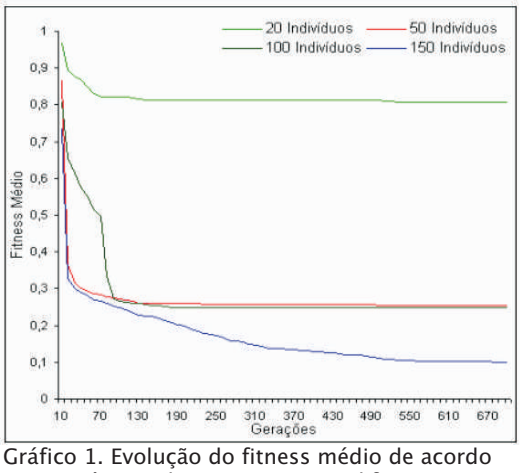


Gráfico 1. Evolução do fitness médio de acordo com o número de gerações, para diferentes quantidades de indivíduos.

O fitness desenvolvido acumula os seguintes valores finais de cada simulação: *(i)* Total de área queimada: busca minimizar a área queimada; *(ii)* Total de área com aceiro: busca minimizar a área de trabalho dos robôs, e; *(iii)* Erro médio absoluto: busca minimizar a diferença entre a média geral de aceiros úteis em relação ao aceiro útil de cada indivíduo, assim, o tamanho das áreas de trabalho tende a se equalizar. Buscamos, na simulação, minimizar o valor do *fitness*, o Gráfico 1 mostra os resultados das simulações com diversos tamanhos de população. Podemos ver o *fitness* diminuindo com o passar das gerações.

A simulação com 150 indivíduos foi a que chegou ao menor *fitness,* podemos ver o resultado satisfatório desta simulação nas Figuras 4(a) e 4(b). O cromossomo resultante das 700 gerações com 150 indivíduos (Figuras 4(a) e 4(b)) apresentou, na última geração os seguintes valores:

225,7 | 199,2 | 174,1 | 155,2 | 136,3 | 27,3 | 30,4 | 33,9 | 38,0 | 35,9

As posições de atuação são calculadas aplicando no cromossomo acima as Equações 1 e 2.

$$x_f = x_a + r_i \times \cos(a_i) \quad (1)$$

$$y_f = y_a + r_i \times \mathrm{sen}(a_i) \quad (2)$$

Onde (x_f, y_f) é a coordenada da posição final dos robôs, (x_a, y_a) é a coordenada da posição inicial do incêndio, r_i é o raio (gene 5 a 9) e a_i é o ângulo (gene 0 a 4). O raio, bem como o ângulo, são específicos para cada operação de cada robô (coordenada inicial e final de criação de aceiro, como mostra a Tabela 1).

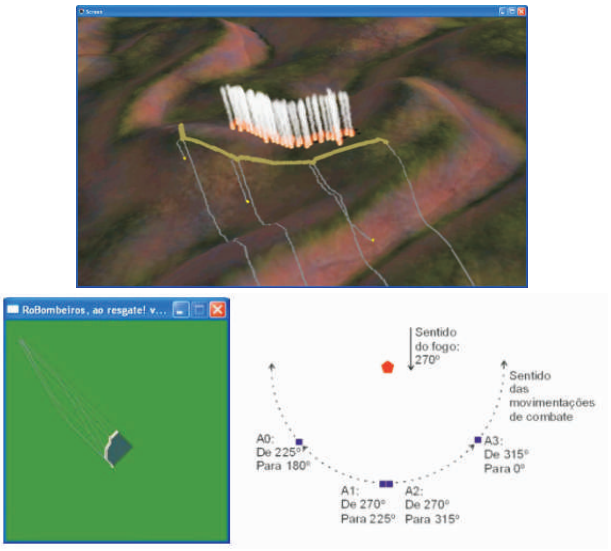


Figura 4. (a) Resultado do planejamento usando GA com 150 indivíduos e 700 gerações, (b) Mesmo resultado em visualização 2D e (c) Modelo teórico de simulação.

6.4 Controle de Navegação

Quando um robô recebe a coordenada da posição do incêndio do agente monitor, ele deve realizar o deslocamento de sua posição atual até a posição de atuação. Para realizar esta navegação, desviando de obstáculos, o agente usa uma RNA do tipo Backpropagation. As entradas da RNA são obtidas pelos sensores presentes no robô e as saídas são as forças a serem aplicadas nos motores. A RNA possui 7 entradas: orientação do veículo e orientação para o alvo (obtidos através de bússola e GPS simulados) mais as proximidades de obstáculos medidas por 5 sensores do tipo sonar. As saídas são: força a aplicar no motor linear (torque, de 0.0 a 6.0) e força a aplicar no motor angular (giro da barra da direção, de –1.5 a 1.5).

Figura 5. Trajetos realizados com simulações usando a RNA: (a) 2,5% de ocupação, (b) e (c) 10% de ocupação.

A base de dados para treinamento da RNA foi obtida a partir de um sistema de regras simples, rodamos 32 simulações com diferentes pontos de início e destino para obter dados específicos de navegação e de desvios de obstáculos, obtendo um total de 4.985 registros. Esta base de dados foi dividida em 70% para treino e 30% para teste, e, usando o SNNS foram geradas seis Redes Neurais Artificiais com 4, 9, 18, 24, 30 e 36 neurônios na camada oculta. Foi realizado o treino de cada RNA e análise de 5.000, 10.000, 20.000, 40.000, 60.000, 80.000 e 100.000 ciclos. A RNA que apresentou o menor Mean Absolute Error (MAE) tanto no giro como na velocidade foi a com 24 neurônios na camada oculta, com 80.000 ciclos de treino. Esta RNA foi convertida em um programa C e utilizada nos robôs. Após a implementação da RNA no controle de navegação com desvio de obstáculos nos robôs móveis, realizamos uma série de simulações com o ambiente contendo 2 robôs, com pontos de início e destino aleatórios e com uma quantidade parametrizada de árvores. Os resultados das simulações podem ser vistos na Tabela 2 e na Figura 5.

Quantidade de Simulações de Navegação	Área Ocupada com Árvores	Resultados Satisfatórios com RNA
50	10,00%	42 (84,00%)
50	5,00%	49 (98,00%)
50	2,50%	49 (98,00%)
50	0,625%	50 (100,0%)

Tabela 2. Resultado das simulações.

7. Conclusão

O objetivo deste artigo foi detalhar o projeto e o desenvolvimento de um sistema multi-agente que opera em um ambiente virtual de simulação realística. Neste sistema, uma equipe de agentes autônomos trabalha cooperativamente a fim de realizar com sucesso a identificação e o combate de incêndios em áreas florestais, sem intervenção humana. O ambiente suporta uma série de características fundamentais para a simulação realística da operação, como terrenos irregulares, processos naturais e restrições físicas na criação e uso de robôs móveis. A operação multi-agente depende essencialmente de duas etapas: planejamento e ação. No planejamento, usamos Algoritmos Genéticos para evoluir estratégias de posicionamento de atuação dos robôs bombeiros. Para a ação foram criados robôs de combate fisicamente simulados, sendo as informações sensoriais de cada robô (e.g. GPS, bússola, sonar) usadas na entrada de uma Rede Neural Artificial (RNA). Esta RNA que controla os atuadores do veículo permite navegação com desvio de obstáculos. Os resultados das simulações demonstram que a Rede Neural controla satisfatoriamente os robôs móveis, que

o uso de Algoritmos Genéticos configura a estratégia de combate ao incêndio de modo satisfatório e que o sistema multi-agente proposto pode vir a ter um papel muito importante no planejamento e execução de operações reais de combate a incêndios florestais.

8. Referências

[1] Balch, T., Arkin R. C., "Motor schema-based formation control for multiagent robot teams", *IEEE International Conference on Multiagent Systems* (ICMAS), 1995.

[2] Dudek, G., Jenkin, M., *Computational Principles of Mobile Robotics*, Cambridge, London, UK: MIT Press, 2000.

[3] GO, J. et. al., "Accurate and flexible simulation for dynamic, vision-centric robots", *International Joint Conference on Autonomous Agents*, 2004.

[4] Osagie, P., "Distributed Control for Networked Autonomous Vehicles", *Dissertação de Mestrado*, KTH-CSC, Royal Institute of Technology, Sweden, 2006.

[5] Rezende, S.O., *Sistemas inteligentes: fundamentos e aplicações*, Ed. Manole, São Paulo, 2003.

[6] Nikolopoulos, C., *Expert Systems - Introduction to First and Second Generation and Hybrid Knowledge Based Systems*. New York, USA: Marcel Dekker Inc. Press, 1997.

[7] Holland, J., *Adaptation in Natural and Artificial Systems*, Michigan Press, 1975.

[8] Mitchell, M., *An Introduction to Genetic Algorithms*, Cambridge, MA, The MIT Press, 1996.

[9] Carvalho, A. C. F., Braga, A. P., Ludermir, T. B., *Sistemas inteligentes: Fundamentos e aplicações,* Cap. Computação Evolutiva, p. 225-248, 2003.

[10] Stuttgart Neural Network Simulator (SNNS), em www-ra.informatik.uni-tuebingen.de/SNNS/, outubro de 2007.

[11] Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[12] Lacerda, E. G. M., Carvalho, A. C. P. L. F. "Introdução aos algoritmos genéticos", *XVIII Jornada de Atualização em Informática*, XIX CSBC, v. 2, p. 51-126, 1999.

[13] Nolfi, S., Floreano, D., *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*, The MIT Press, 2000.

[14] Antunes, M. A. H., "Uso de satélites para detecção de queimadas e para avaliação do risco de fogo", *Ação Ambiental*, 12:24-27, 2000.

[15] Batista, A.C., "Detecção de incêndios florestais por satélite", *Floresta*, Curitiba, Paraná, n 34, 237-241, 2004.

[16] Remmel, T. K., Perera, A. H. "Fire mapping in a northern boreal forest assessing AVHRR/NDVI methods", *Forest Ecology and Management 152*, 2001.

[17] Koproski, L.P., "O fogo e seus efeitos sobre a heperto e a mastofauna terrestre no parque nacional de Ilha Grande", *Dissertação de mestrado*, UFPR, 2005.

[18] Pessin, G. et. al. "Simulação Virtual de Agentes Autônomos para a Identificação e Controle de Incêndios em Reservas Naturais", *IX Symposium on Virtual and Augmented Reality* (SVR), v. 1, p. 236-245, 2007.

[19] Wooldridge, M.; Jennings, N., "Intelligent agents: Theory and practice", *Knowledge Engineering Review*, 1995.

[20] Pfeifer, R., Scheier, C., *Understanding Intelligence*, Cambridge, Massachusetts, USA: The MIT Press, 1999.

[21] Laboratório de Incêndios Florestais, "Pesquisas e projetos em prevenção e combate de incêndios florestais", UFPR, www.floresta.ufpr.br/~firelab, setembro 2006.

[22] Centro de previsão do tempo e estudos climáticos - Instituto nacional de pesquisas espaciais (CPTEC/INPE), www.cptec.inpe.br/queimadas, Acesso em outubro 2006.

[23] Castro, F. X.; Tudela, A.; Sebastia, M. T. "Modeling moisture content in shrubs to predict fire risk in catalonia", *Agricultural and Forest Meteorology*, Spain, v. 116, 2001.

[24] Ministério da Agricultura do Brasil, *Guia método-lógico para elaboração de plano municipal/intermunicipal de defesa da floresta contra incêndios*, 2006.

[25] Maes, P. "Intelligent software: easing the burdens that computers put on people", *IEEE Expert*, v. 11, 1996.

[26] Heinen, M. R., Osório, F. S., "Algoritmos genéticos aplicados em roteamento de veículos" *Hífen*, 2006.

[27] *Open Dynamics Engine*, Disponível em: <www.ode.org>. Acesso em jun. 2007.

[28] *Open Scene Graph*, OSG Community, Disponível em: <http://www.openscenegraph.com>. Acesso em jun. 2007.

[29] *Demeter Terrain Engine*, Disponível em: <http://www.tbgsoftware.com/>. Acesso em jun. 2007.

[30] Osório, F. S., et al. "Increasing Reality in Virtual Reality Applications through Physical and Behavioural Simulation", *Proc. of Virtual Concept Conference, 2006.*

[31] Reis, L. P., "Coordenação em sistemas multi-agente: aplicações na gestão universitária e futebol robótico", *Tese de Doutorado*, Universidade do Porto, Portugal, 2003.

[32] Hornik, K., Stinchcombe, M., White, H. "Multilayer feedforward networks are universal approximators", *Neural Networks*, v. 2, p. 359–366, 1989.

[33] Smith, R., *Open Dynamics Engine*, *User Guide,* 2006.

[34] Marchi, J., "Navegação de robôs móveis autônomos: Estudo e implementação de abordagens", *Dissertação de Mestrado*, Universidade Federal de Santa Catarina, 2001.

[35] JPL/NASA. Jet Propulsion Laboratory, Disponível em: <http://wwwrobotics.jpl.nasa.gov>. Acesso em maio 2007.

[36] Silveira, R. A., "Introdução a Sistemas Multi-agente", Universidade Federal de Santa Catarina (UFSC), 2006.

[37] Kartalopoulos, S. V., "Understanding Neural Networks and Fuzzy Logic - Basic Concepts and Applications", *IEEE Press Understanding Science and Technology Series*, 1996.

[38] Garcia, A. C. B., Sichman, J. S., *Sistemas inteligentes: Fundamentos e aplicações*. In: Barueri, SP, Brasil: Manole, Cap. Agentes e Sistemas Multi-Agentes, p. 269–306, 2003.

[39] Bekey, G. A, *Autonomous Robots: From Biological Inspiration to Implementation and Control*, Cambridge, Massachusetts, USA: The MIT Press, 2005.

[40] Pfeifer, R., Scheier, C., "From perception to action: The right direction?" *Proceedings of the From Perception to Action Conference, IEEE Press*, p. 1–11, 1994.

[41] Pfeifer, R., Iida, F., Bongard, J. "New robotics: Design principles for intelligent systems", *Artificial Life*, 2005.

[42] Rumelhart, D., Mcclelland, J., *Parallel Distributed Processing*, Cambridge: MIT Press, 1986.

[43] Osório, F. S. "INSS: Un Système Hybride Neuro-Symbolique pour l'Apprentissage Automatique Constructif", *Tese de Doutorado*, Institut National Polytechique de Grenoble (INPG), Grenoble, França, 1998.

Transmissão de Características Genéticas na Geração de Personagens Virtuais

Roberto C. Cavalcante Vieira, Creto Augusto Vidal, Joaquim B. Cavalcante-Neto
Department of Computing, Federal University of Ceará - Fortaleza, CE, Brazil
{roberto}, {cvidal}, {joaquimb}@lia.ufc.br

Abstract

Nowadays, virtual reality, computer games and many other applications are using more and more sophisticated models of human characters. Many of these applications, such as life-simulation computer games (The Sims), internet-based virtual worlds (Second Life) and animation movies, besides the complexity of the human models, sometimes require simulation of kinship and interaction between isolated populations with well defined ethnic characteristics. Another difficult task is to generate models physically similar to a given population or family automatically. In this paper, diploid reproduction is mimicked to produce character models, which inherit traits from two parent models. The meshes of all models are constructed based on control parameters that are distributed as genes among a group of chromosomes. Thus, the technique consists of distributing pre-selected characteristics, represented as control parameters, over a pre-determined number of chromosome pairs for both parents; followed by a simulated generation of the father's and the mother's gametes; which are randomly combined in a simulated fecundation.

1. Introdução

O povoamento de ambientes virtuais por personagens é necessário em inúmeras aplicações de realidade virtual e computação gráfica. A necessidade de gerar modelos realistas tem tornado essa tarefa cada vez mais complexa.

Em muitas dessas aplicações, como jogos de simulação do cotidiano (ex: The Sims - http://thesims.ea.com/), mundos virtuais na internet (ex: Second Life - http://secondlife.com/) e filmes de animação, é necessário simular relações de parentesco e interação entre populações isoladas com características étnicas bem definidas.

Em aplicações, como jogos, é necessário povoar ambientes dinamicamente, à medida que a simulação evolui junto com a população, e, às vezes, é interessante gerar populações distintas que migram e geram mistura de características étnicas.

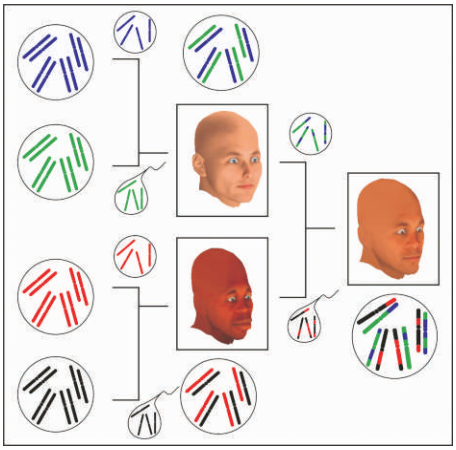


Figura 1. Transmissão de características aos avatares descendentes.

O problema tratado neste artigo é o da transferência automática das características físicas de populações para seus descendentes, para utilização em sistemas de realidade virtual, simulando interações entre famílias ou populações etnicamente distintas. A solução proposta para esse problema é ilustrada na Figura 1. Essa solução tenta simular de forma algorítmica o processo de reprodução humano, enfatizando quatro processos aleatórios: três, que ocorrem na geração de gametas através da meiose (o crossover e os dois alinhamentos aleatórios de cromossomos) e um que ocorre durante a fecundação.

As seções restantes estão estruturadas da seguinte maneira. Na Seção 2, apresentam-se as propostas de solução encontradas na literatura. Na Seção 3, é feita uma breve exposição dos conceitos da biologia da reprodução de seres diplóides utilizados na solução proposta neste trabalho. Na Seção 4, mapeiam-se os conceitos apresentados na Seção 3 para o processo de geração de personagens por reprodução simulada. Na Seção 5, um estudo de casos ilustra o processo geral

descrito na Seção 4 focalizando o processo na geração de faces de personagens virtuais. Na Seção 6, apresentam-se as conclusões sobre o trabalho.

2. Trabalhos Relacionados

O problema da diversidade na geração de personagens virtuais não é novidade, e os trabalhos mais relevantes dessa área são apresentados brevemente nesta seção. No entanto, não há nenhum trabalho envolvendo modelos 3D de personagens virtuais com herança de características de ancestrais, permitindo a simulação de grupos ou de famílias.

Em 1998, DeCarlo e seus co-autores [5] usaram técnicas variacionais para sintetizar faces com algumas distâncias características consistentes com medidas antropométricas. Eles descreveram um sistema que gera automaticamente modelos de faces para a construção de avatares a serem utilizados em aplicações de realidade virtual. Embora haja crítica na literatura sobre a plausibilidade das faces geradas [6], esse trabalho foi um dos pioneiros na utilização de técnicas antropométricas [10] para a geração de faces, e serviu de inspiração para vários outros trabalhos. Blanz e Vetter [3] foram os primeiros a estudar o espaço de faces. Eles usam a técnica de PCA (Principal Component Analysis) para gerar um modelo linear passível de mudanças morfológicas a partir de um banco de dados de faces obtidas por scanner 3D. Blanz e Vetter conseguiram gerar grande variabilidade de modelos bastante realistas. Praun e os seus co-autores [11] utilizaram técnicas de parametrização para combinar características de dois ou mais modelos com topologias distintas, também gerando modelos intermediários. Esta técnica requer esforço humano na definição dos parâmetros. Bui e seus co-autores [4] também utilizaram a técnica de morphing, porém, para animações faciais. Kähler e seus co-autores [7-8] construíram um modelo genérico com camadas anatômicas (pele, músculos e ossos) associados com landmarks antropométricos. Com base em informações antropométricas de indivíduos de diferentes idades, foi possível simular mudanças faciais em diferentes fases da vida. Em 2003, Allen e seus co-autores [1], basearam-se no trabalho de Blanz e Vetter, expandiram a técnica para todo o corpo humano. Combinando modelos com diferentes alturas e pesos. Seo e Thalmann [12] desenvolveram um modelo base adaptável, construído com patches de Bézier. Os dados utilizados para orientar a adaptação do modelo básico são obtidos a partir da combinação de medidas de modelos disponíveis armazenados em uma base de dados construída com o uso de scanner 3D. Mais recentemente, Golovinskiy e seus co-autores [6] criaram uma técnica estatística para analisar e sintetizar pequenas características faciais 3D tais como rugas e poros. O objetivo é conferir maior realismo visual nos personagens usados em jogos, realidade virtual e visão computacional.

Nenhum desses trabalhos, apesar dos bons resultados obtidos, utiliza informações biológicas ou teoria reprodutiva para gerar os modelos, o que seria uma maneira natural para gerar personagens virtuais descendentes ou pertencentes a grupos fisicamente distintos.

3. Reprodução - Conceitos

Nesta seção, é feita uma revisão bastante concisa de conceitos biológicos sobre a reprodução de seres diplóides, com o único objetivo de preparar o leitor para compreender o mapeamento direto desses conceitos na forma algorítmica descrita na Seção 4. Os três conceitos importantes no contexto deste trabalho são: a armazenagem de informações genéticas, a geração de gametas e a fecundação.

3.1. Armazenagem de informações genéticas

As características genéticas dos seres vivos estão codificadas nos cromossomos. Nos seres chamados diplóides, o número total de características genéticas é distribuído em n subconjuntos, cada subconjunto é armazenado em um par de cromossomos ditos homólogos (um cromossomo registrando as características oriundas do pai e o outro as características oriundas da mãe). Cada característica armazenada em um cromossomo é chamada gene. Genes correspondentes nos cromossomos homólogos (genes que ocupam a mesma posição no cromossomo) são chamados de genes alelos.

Todas as células de um ser diplóide adulto originam-se de uma única célula, o óvulo fecundado. Essa célula-ovo ou zigoto, por meio de divisões celulares sucessivas, propicia a formação de organismos complexos. Essas inúmeras células podem ser agrupadas de duas maneiras diferentes: as células somáticas (com 2n cromossomos), que em conjunto formam o corpo do indivíduo, e os gametas (com n cromossomos), destinados à perpetuação das espécies. Nos seres humanos, as células somáticas possuem 23 pares de cromossomos (2n = 46 cromossomos) e os gametas (óvulos e espermatozóides) possuem 23 cromossomos (n = 23 cromossomos).

3.2. Meiose e geração de gametas

Uma célula germinativa é uma célula somática especializada que se divide por meiose para gerar

quatro gametas (células haplóides). A meiose ocorre em oito fases distintas, precedida de uma fase de duplicação dos cromossomos, chamada Intérfase: Prófase I, Metáfase I, Anáfase I, Telófase I, Prófase II, Metáfase II, Anáfase II, Telófase II (Figura 2). Em algumas dessas fases ocorrem processos de natureza aleatória que são responsáveis pela diversidade dos descendentes.

O primeiro desses processos aleatórios ocorre na Prófase I. Nele, pedaços correspondentes de alguns dos cromossomos homólogos duplicados são trocados entre si (crossover). Assim, o novo par de cromossomos homólogos duplicados fica diferente do par duplicado da célula original (vide Figura 3). Nesse processo, quantos pedaços são trocados e quais são seus tamanhos é aleatório. Também não é determinístico, caso o crossover ocorra, em quais dos n pares de cromossomos homólogos o crossover ocorrerá.

O segundo e o terceiro processos aleatórios ocorrem respectivamente na Metáfase I e na Metáfase II, durante o alinhamento dos n pares de cromossomos homólogos. Não é possível saber a priori quais são os n cromossomos que ficarão em uma célula ou na outra após a divisão celular (vide Figura 2).

Em síntese, para a geração dos gametas, cada cromossomo separa-se do seu homólogo, formando células haplóides. Esse processo leva a um fenômeno chamado segregação independente dos homólogos. Isso significa que a geração dos gametas ocorre por uma combinação independente dos cromossomos homólogos, possibilitando a geração de uma variedade enorme de gametas.

3.3. Fecundação

Cada gameta gerado por dois indivíduos diplóides possui apenas um dos alelos de cada gene. A fecundação reúne os alelos dos gametas para formar um ser diplóide. A combinação de alelos poderá ocorrer por dominância, por recessividade ou por dominância incompleta. No caso de heterozigotos, o gene dominante, levará a um fenótipo da sua característica, apesar de também possuir um gene recessivo. Existem casos onde a situação de dominância e recessividade não ocorre por completo. São os casos de dominância incompleta, onde os indivíduos heterozigotos apresentam características intermediarias entre os fenótipos dos homozigotos.

4. Reprodução – Algoritmo

Nesta seção, os conceitos sobre reprodução de seres diplóides são encapsulados em uma forma algorítmica

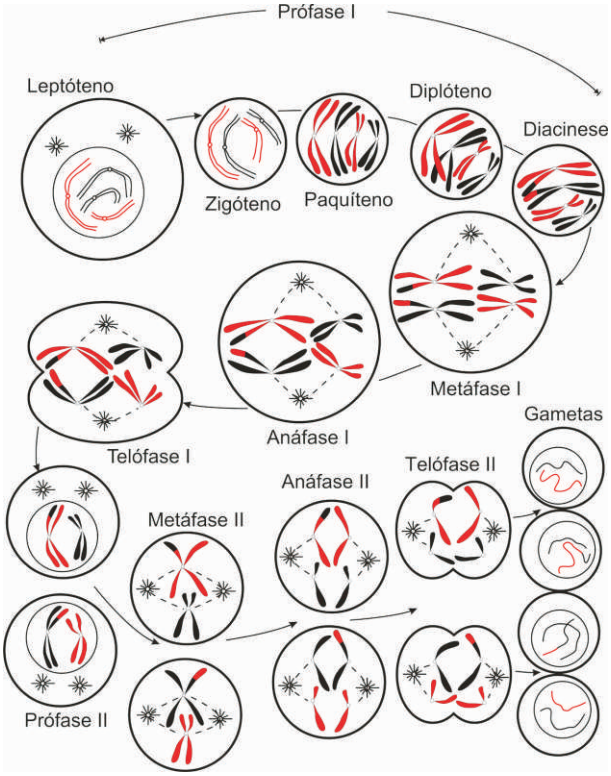


Figura 2. Geração de gametas.

que permite a simulação computacional.

4.1. Identificação e armazenagem dos genes

O gene a ser armazenado em uma estrutura de dados adequada ao algoritmo de reprodução corresponde a uma característica que é usada na reconstrução do personagem virtual. Cada uma dessas características, tais como: altura do personagem, forma e tamanho dos membros, largura dos ombros, forma e tamanho da cabeça, forma dos olhos, forma e tamanho do nariz, da boca, das orelhas, do queixo, e assim por diante; pode ser vista como informação escalar de alto-nível que, respeitando limites antropométricos, controla uma subestrutura hierárquica que constrói parte do personagem.

Após a identificação do conjunto de genes suficientes para reconstrução do personagem virtual, os genes desse conjunto são distribuídos em subconjuntos denominados cromossomos. Nos seres humanos, por exemplo, o total de genes é distribuído em 23 pares de cromossomos. Analogamente, considerando n o número de características genéticas, haverá n pares de genes alelos, distribuídos em m pares de cromossomos homólogos.

A estrutura de dados adequada ao algoritmo de reprodução deste trabalho é descrita como se segue:

· C: Conjunto de pares c_i de cromossomos homólogos

$$C = \{c_1, c_2, \cdots, c_m\}; \quad (1)$$

· c_i: i-ésimo par de cromossomos homólogos

$$c_i(c_i^M, c_i^F) \quad (2)$$

· c_i^M : i-ésimo cromossomo oriundo do ancestral masculino

$$c_i^M = \{g_{i1}^M, g_{i2}^M, \cdots, g_{in_i}^M\} \quad (3)$$

· c_i^F : i-ésimo cromossomo oriundo do ancestral feminino

$$c_i^F = \{g_{i1}^F, g_{i2}^F, \cdots, g_{in_i}^F\}, \quad (4)$$

· g_{ij}^G : j-ésimo gene do i-ésimo cromossomo oriundo do ancestral de gênero G (M, masculino e F, feminino).

O número de genes em cada cromossomo satisfaz a

$$n = \sum_{i=1}^{m} n_i \ . \quad (5)$$

A escolha de m influencia a variabilidade dos descendentes, por contribuir para o aumento de possibilidades de combinações na Metáfase I e II.

Essa estrutura de dados é instanciada para cada personagem. Assim, para começar a reprodução é preciso que já tenham sido instanciadas as estruturas de dados de dois personagens virtuais que serão pai e mãe de uma geração de personagens virtuais descendentes.

4.2. Geração de gametas

Denominando-se *Pai.C* e *Mãe.C* as estruturas de dados dos personagens virtuais geradores, quatro gametas do pai e quatro da mãe são gerados cada vez que se aplica o processo de meiose a *Pai.C* e a *Mãe.C* . É importante salientar que, probabilisticamente, os gametas gerados em uma dada aplicação do processo de meiose sobre essas estruturas de dados são diferentes dos obtidos em meioses anteriores. Assim, aplicando-se várias vezes o processo de meiose sobre *Pai.C* e *Mãe.C* é possível gerar um reservatório de gametas provenientes do pai e da mãe.

Os processos meióticos, relevantes ao tratamento algorítmico da reprodução são descritos a seguir.

4.2.1. Duplicação dos cromossomos. Na verdade esse é um processo que ocorre antes da meiose propriamente dita. A estrutura de dados <gerador>.C é duplicada, fazendo com que a forma duplicada de cada par de cromossomos homólogos tenha a seguinte estrutura:

$$^{d}c_i(c_i^M c_i^M, c_i^F c_i^F) \quad (6)$$

A réplica do cromossomo original c_i^G (G = M ou F) chama-se cromátide irmã e c_i^M estão emparelhadas com as cromátides c_i^F (Figura 2).

4.2.2. Crossover. A Figura 3 ilustra este processo. Inicialmente, os cromossomos homólogos $_1c^M$ e $_1c^F$ com suas respectivas cromátides irmãs $_2c^M$ e $_2c^F$ estão emparelhados (conectores de linha contínua na Figura 3(a)). Em seguida, as cromátides homólogas trocam segmentos (conectores tracejados na Figura 3(a)). O resultado dessas trocas de segmentos está ilustrado na Figura 3(b). Se não houver nenhuma troca de segmentos, as cromátides irmãs permanecerão idênticas. Entretanto, se houver crossover, os quatro cromossomos agora são distintos.

Para fins de simulação do crossover, em um par de cromossomos duplicados $^{d}c_i(c_i^M c_i^M, c_i^F c_i^F)$, cada cromossomo c_i^M troca genes com seu homólogo c_i^F. No i-ésimo par de cromossomos, tem ni genes, o número de trocas possíveis é dado por

$$\sum_{k=0}^{n_i} \begin{bmatrix} n_i \\ k \end{bmatrix} = 2^{n_i} \quad (7)$$

Porém, como os cromossomos que efetuaram trocas de genes são complementares, o número de pares distintos que podem ser gerados pela troca de genes é $2^{n_i - 1}$. Assim, após o crossover, a estrutura do cromossomo duplicado é dada por

$$^{d}c_i(_1c_i^M \ _2c_i^M, _1c_i^F \ _2c_i^F) \quad (8)$$

onde $(_1c_i^M, _1c_i^F)$ e $(_2c_i^M, _2c_i^F)$ pertencem ao universo de $2^{n_i - 1}$ pares possíveis.

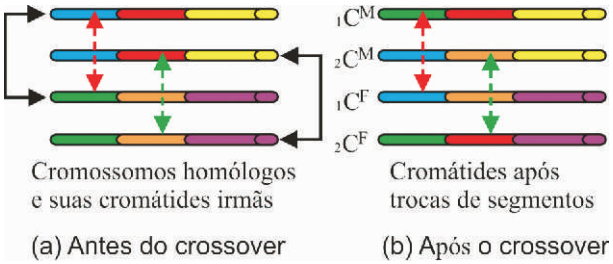


Figura 3. Crossover entre cromossomos homólogos.

4.2.3. Alinhamento de cromossomos – Metáfase I.

Após a simulação do crossover, os *m* pares de cromossomos duplicados (equação (8)) são alinhados (vide Metáfase I na Figura 2). Para essa simulação, é conveniente reescrever a equação (8) como

$$ {}^{d}c_i({}^{d}c_i^{M}, {}^{d}c_i^{F}) \qquad (9) $$

Considerando-se a aleatoriedade de distribuição dos pares em relação à estrutura básica de alinhamento dos m pares de cromossomos, ilustrada na Figura 4, onde os *m* cromossomos ${}^{d}c_i^{M}$ e os *m* cromossomos ${}^{d}c_i^{F}$ estão em lados opostos do plano pontilhado que representa a divisão celular, as duas células geradas no final desse processo pertencem a um universo de 2^{m-1} pares distintos de células.

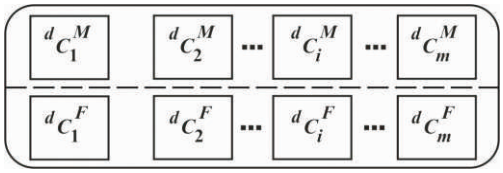


Figura 4. Alinhamento dos cromossomos na Metáfase I.

4.2.4. Alinhamento de cromossomos – Metáfase II.

As duas células geradas ao final da Telófase I são divididas mais uma vez e geram quatro gametas. Em cada uma das duas células geradas na Fase I da meiose, há um conjunto de *m* pares de cromátides irmãs que foram modificadas durante o processo de crossover (não são mais idênticas), isto é, o i-ésimo par pode ser $({}_1c_i^{M}\ {}_2c_i^{M})$ ou $({}_1c_i^{F}\ {}_2c_i^{F})$. Na Metáfase II, esses *m* pares serão alinhados para serem, em seguida, separados em duas novas células. O processo de simulação é idêntico ao da Metáfase I e a aleatoriedade do alinhamento está ilustrada na Figura 5.

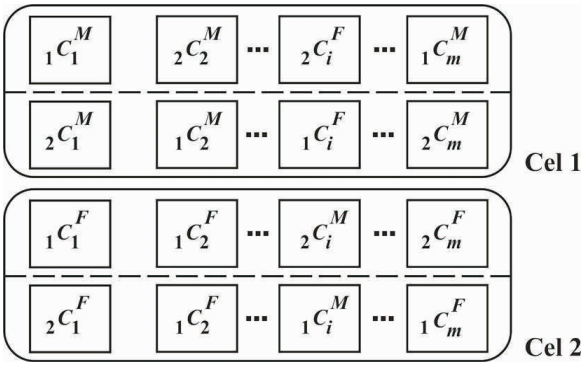


Figura 5. Alinhamento dos cromossomos na Metáfase II.

Novamente, o par de células geradas a partir da Célula 1 (Figura 5) pertence a um universo de 2^{m-1} pares distintos. Analogamente, o novo par de células geradas a partir da Célula 2 (Figura 5) pertence a um diferente universo de 2^{m-1} pares distintos.

As quatro células geradas são gametas. Cada uma delas tem m cromossomos (células haplóides) ao invés de *m* pares de cromossomos como nas células somáticas diplóides.

As quatro células geradas são gametas. Cada uma delas tem *m* cromossomos (células haplóides) ao invés de *m* pares de cromossomos como nas células somáticas diplóides.

4.3. Fecundação dos personagens virtuais

Esse processo aleatório é bastante simples e consiste em escolher aleatoriamente um gameta do reservatório de gametas paterno e um gameta do reservatório de gametas materno e copiar essas informações haplóides na estrutura diplóide do descendente. Dessa forma, supondo-se que o reservatório paterno contenha π gametas e o materno contenha μ gametas, pode-se gerar até π x μ personagens virtuais descendentes de primeira geração. As características genéticas desses descendentes pertencem a um universo de probabilidades que incorpora a aleatoriedade dos crossovers dos *m* pares duplicados de cromossomos, e a aleatoriedade dos alinhamentos da Metáfase I e II dos ancestrais. Nos seres humanos isso corresponde a um universo de 10^{600} [13].

5. Estudo de Caso: Geração de Malhas Faciais

O estudo de caso apresentado nesta seção ilustra o processo geral, descrito na Seção 4, através da geração das faces da prole de dois personagens geradores. Foram feitas duas simulações: a primeira simulou o processo aplicado a diferentes grupos étnicos resultando na árvore genealógica ilustrada na Figura 18; e a segunda utilizou modelos caricaturados resultando na árvore genealógica da Figura 19.

5.1. Considerações sobre as malhas faciais

Os modelos iniciais da primeira simulação foram gerados com o software Poser 6.0 (www.e-frontier.com) e importados para o sistema desenvolvido no presente trabalho. Os modelos caricaturados foram construídos exagerando as medidas antropométricas a partir de um modelo base importado do Poser 6.0. É importante mencionar, no entanto, que as malhas faciais podem ser geradas por qualquer outro software e importadas para o sistema, contanto que exista associação ponto-a-

ponto entre os modelos. Essa associação funciona como no trabalho de Blanz e Vetter [3], portanto a técnica poderia fazer uso de bases de dados obtidas por scanner 3D como fizeram esses autores.

As características associadas aos genes e distribuídas entre os cromossomos foram as medidas faciais, definidas pelos *landmarks* mostrados na Figura 6 [9].

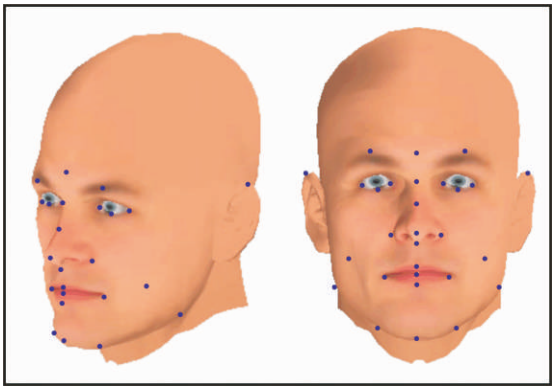


Figura 6. *Landmarks* antropométricos

A malha de referência possui 2076 vértices e ajusta-se de acordo com a combinação resultante dos genes dos ascendentes para gerar automaticamente a malha da prole. As malhas iniciais da simulação podem ser construídas por ajustes manuais ou podem ser selecionadas a partir de uma base de dados de modelos previamente gerados. Se o usuário desejar, também é possível modificar medidas das malhas pertencentes ao banco de dados, utilizando barras de rolagem alterando suas características (silhueta do nariz, largura do nariz, etc.) Isto é possível, usando zonas de influência relativas aos *landmarks* utilizados nas medidas (Genes), conforme descrito nas figuras 7 e 8.

Essas zonas de influência são representadas por esferas que ajustam a vizinhança de um determinado *landmark*, isto é, os pontos que estão no interior da esfera recebem uma influência relativa à sua distância em relação ao *landmark*.

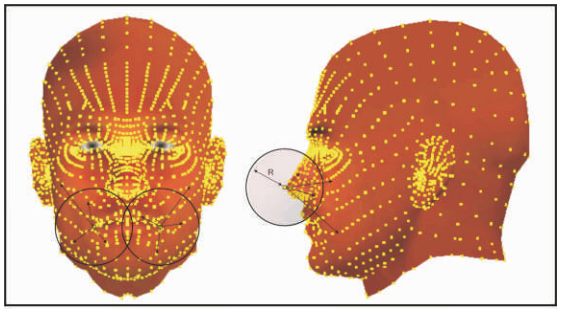


Figura 7. Esferas de influência dos cantos da boca e da ponta do nariz.

```
for every vtx[j] do
begin
    dist = distance between vtx[j] and landmark
    if dist <= S_Radius then
    begin
        vtx[j].X:= vtx[j].X + v[0]*(S_Radius - dist)/S_Radius;
        vtx[j].Y:= vtx[j].Y + v[1]*(S_Radius - dist)/S_Radius;
        vtx[j].Z:= vtx[j].Z + v[2]*(S_Radius - dist)/S_Radius;
    end
end

* V = translation vector
* S_Radius = Sphere_Radius
* vtx = vertex
```

Figura 8. Aplicação das zonas de influência.

Após a definição das malhas do casal, é feita a distribuição de características da face em cromossomos fictícios (Figura 9) para posterior simulação do processo reprodutivo gerando possíveis variações de heranças genéticas na face resultante.

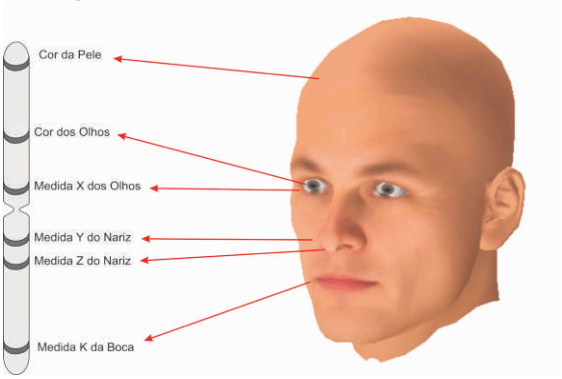


Figura 9. Distribuição de características em cromossomos.

5.2. Definição das Características dos Cromossomos

Neste estudo de caso, a partir dos *Landmarks* antropométricos, dezoito medidas tratadas como genes foram distribuídas em cromossomos (Figura 10).

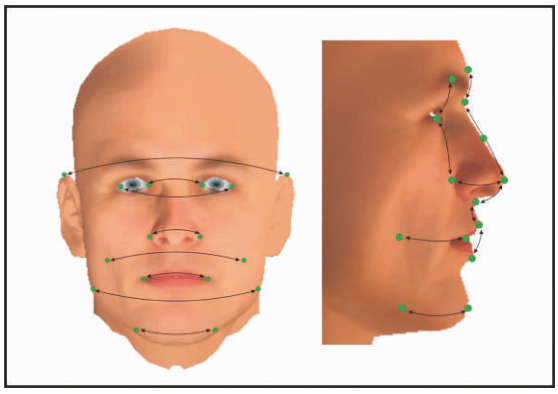


Figura 10. Medidas tratadas como genes.

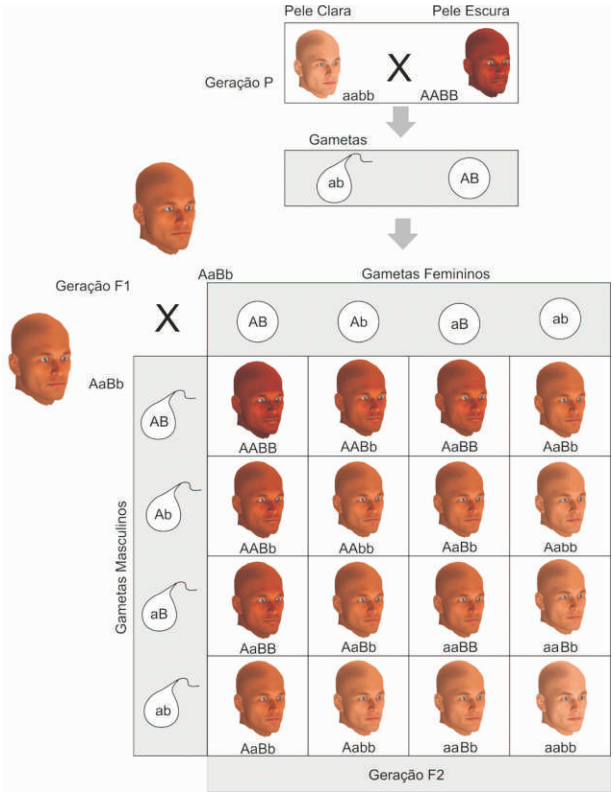


Figura 11. Combinações de genes para cor de pele.

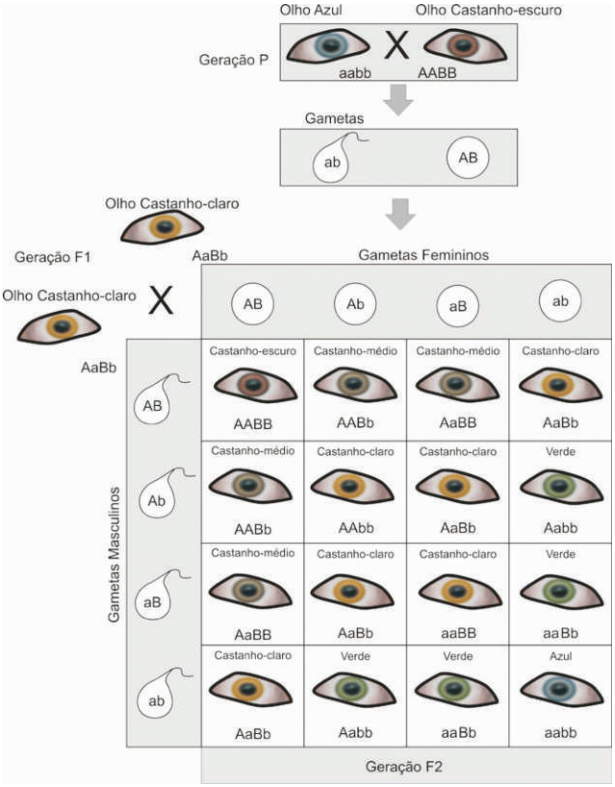


Figura 12. Combinações de genes dos olhos.

Também foram tratadas características da cor dos olhos, conforme descrito por Amabis e seus co-autores [2], e da cor da pele. Para a cor da pele foram definidos dois genes que geram as possíveis combinações conforme a Figura 11. Da mesma forma foram utilizados dois genes para a cor dos olhos combinando-se conforme a Figura 12.

Os modelos iniciais (não gerados pela simulação) são definidos como homozigotos, ou seja, os pares de cromossomos possuem as mesmas medidas. A partir das gerações seguintes os modelos passam a ser heterozigotos, com a combinação dos genes dos modelos que os geraram.

As diferentes formas de distribuição das características nos cromossomos geram resultados variados, uma vez que as características contidas em cromossomos distintos segregam-se de forma independente, e genes contidos em um mesmo cromossomo podem ser permutados por crossover.

O Crossover é um dos fatores importantes que afetam a variabilidade dos descendentes. Assim, é possível jogar com este fator, modificando as probabilidades ou através da limitação da quantidade de trocas que podem ocorrer entre os cromossomos.

A fim de tratar o fator relativo ao gênero, em vez de criar cromossomos para a determinação do sexo, foi definido um gene específico para isso. Se o gene tem valor “X”, o modelo contém características do sexo feminino, se tiver o valor “Y”, contém características do sexo masculino. Assim, um indivíduo com "XX" é do sexo feminino e um, com “XY” é masculino. Ao gerar a malha de um descendente do sexo feminino, as características herdadas do pai são transformadas em formas femininas para combinar com as medidas dos genes correspondentes da mãe. Analogamente, nos descendentes do sexo masculino, as características herdadas da mãe são transformadas em formas masculinas para combinar com as características herdadas do pai. Estas transformações são efetuadas como descrito na Figura 13, assumindo um modelo com formas geométricas que corresponde à média masculina e feminina de uma população, assim como fez Blanz e Vetter [3] com seus modelos obtidos por scanner 3D.

Neste trabalho os modelos que corresponderiam à média foram definidos sem o rigor de uma amostragem real, para efeito de estudo de caso. Modelos com características masculinas e femininas obtidas com esta abordagem são ilustrados na Figura 14.

```
// Male to Female
for every vtx[j] do
begin
    vtx[j].X:= vtx[j].X - (Avg_Male_x[j] - Avg_Female_x[j])
    vtx[j].Y:= vtx[j].Y - (Avg_Male_y[j] - Avg_Female_y[j])
    vtx[j].Z:= vtx[j].Z - (Avg_Male_z[j] - Avg_Female_z[j])
end

// Female to Male
for every vtx[j] do
begin
    vtx[j].X:= vtx[j].X - (Avg_Female_x[j] - Avg_Male_x[j])
    vtx[j].Y:= vtx[j].Y - (Avg_Female_y[j] - Avg_Male _y[j])
    vtx[j].Z:= vtx[j].Z - (Avg_Female_z[j] - Avg_Male _z[j])
end

* vtx = vertex
* Avg_Male = Average_Male
* Average_Female = Average_Female
```

Figura 13. Adaptação da malha ao gênero.

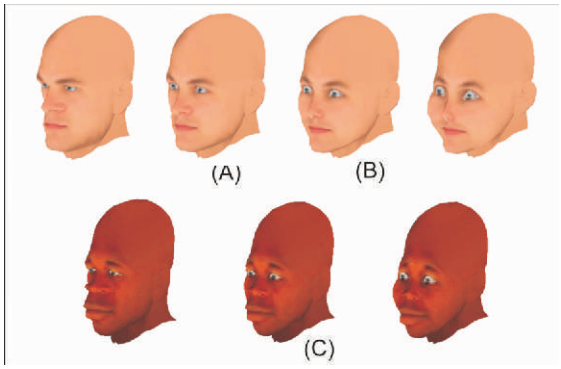


Figura 14. Modificações relativas ao sexo.
(A) Média geométrica masculina utilizada.
(B) Média geométrica feminina utilizada.
(C) Modelo, alterando suas características para mais masculino ou feminino.

As medidas resultantes são geradas a partir da combinação de pares de genes. Na primeira simulação foi usado o conceito de dominância e recessividade das medidas, definida de forma aleatória. Na segunda, foi usada dominância incompleta entre os alelos, gerando um peso aleatório para as medidas a serem combinadas. Conseqüentemente, para uma dada característica, o personagem descendente terá semelhança geométrica mais próxima do ancestral com gene de maior peso.

5.3. Geração de gametas

Nos exemplos simulados neste estudo de caso, as características genéticas foram distribuídas em cinco cromossomos, $C = \{c_1, c_2, c_3, c_4, c_5\}$ (Figura 15), com a seguinte estrutura:

$$C = \{c_1, c_2, c_3, c_4, c_5\} = \left\{ \begin{pmatrix} a^M & a^F \\ b^M & b^F \\ c^M & c^F \\ d^M & d^F \end{pmatrix}, \ldots, \begin{pmatrix} r^M & r^F \\ s^M & s^F \\ t^M & t^F \\ u^M & u^F \end{pmatrix} \right\}$$

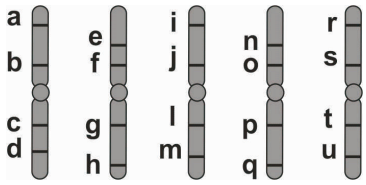


Figura 15. Distribuição dos genes nos Cromossomos.

Na estrutura cromossômica, genes são associados a medidas antropométricas com zonas de influência distintas. Para garantir uma alta variedade de gametas, foi permitido ocorrer crossover entre qualquer gene no cromossomo, ou seja, segmentos de um cromossomo contendo apenas um gene podem ser permutados.

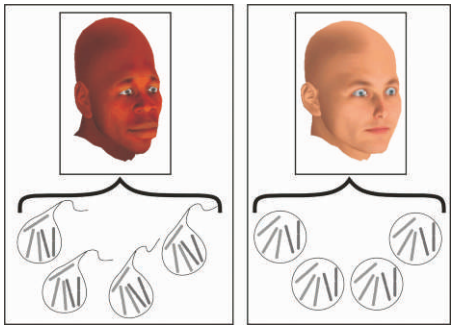


Figura 16. Geração de quatro gametas por interação da simulação.

Neste estudo, ao invés de gerar dois grandes reservatórios de gametas, um paterno e outro materno; em cada processo de meiose aplicado às estruturas *Pai.C* e *Mãe.C* um gameta de cada é escolhido aleatoriamente e fecundado (Figura 17).

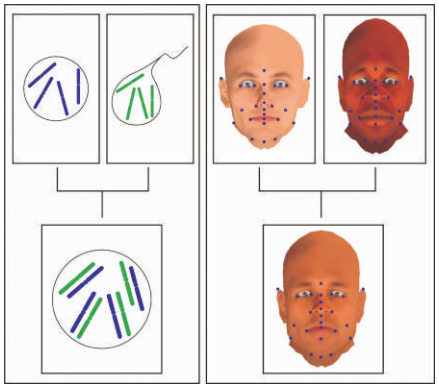


Figura 17. Representação da fecundação utilizando medidas antropométricas.

5.4. Geração dos modelos da prole

O processo consiste em construir uma estrutura cromossômica de um descendente pela fecundação de um gameta feminino por um masculino. Esta estrutura cromossômica é então utilizada para gerar a malha dos descendentes (Figura 14), seguindo os passos:

· Definição da cor da pele pela combinação dos genes herdados, conforme Figura 11;

· Definição da cor dos olhos pela combinação dos genes herdados, conforme Figura 12;

· Definição do sexo pela combinação dos "genes" X e Y e transformação das características relativas ao gênereo, conforme Figuras 13 e 14 para que sejam feitas as combinações de genes nas etapas seguintes;

· Ajuste das medidas ao modelo, utilizando as zonas de influência descritas nas figuras 7 e 8, para cada gene, que pode ocorrer por dominância completa ou incompleta.

Neste estudo de caso, para cada casal analisado, foram realizadas diversas fecundações. A Figura 18 ilustra a árvore genealógica, simulando parentescos de indivíduos etnicamente distintos, até a terceira geração (Avós, pais e filhos). A segunda árvore genealógica (Figura 19) simula parentescos de indivíduos caricaturados com características faciais e cores de pele e olhos bem distintas, até a quarta geração.

6. Conclusões

Este trabalho apresenta uma solução para o problema de transmissão de características herdadas dos modelos ascendentes bem como de simulação de relações de parentesco e interação entre populações isoladas com características étnicas bem definidas. A solução proposta baseia-se no automatismo e grande variabilidade que ocorre na reprodução de seres diplóides. O processo necessita de uma descrição geométrica subjacente que se adapte segundo as informações genéticas armazenadas em uma estrutura diplóide de cromossomos artificiais. O modelo geométrico utilizado neste estudo foi composto por malhas com 2076 vértices e associações ponto-a-ponto. O controle da adaptação do modelo usou pontos definidos por landmarks antropométricos cujos subagrupamentos definiram as características genéticas armazenadas como genes nos pares de cromossomos homólogos artificiais. Através do processo de meiose simulada, aplicado sobre a estrutura cromossômica de dois personagens geradores, gametas foram gerados para posteriormente serem usados na fecundação simulada. No estudo de caso foi apresentado o processo reprodutivo aplicado à geração de faces, no entanto, a teoria é geral e pode ser aplicada ao modelo completo (cabeça e corpo) de um personagem, devendo apenas adaptar a manipulação geométrica às características das diversas partes do corpo. Este método garante o automatismo e grande variabilidade existentes em alguns dos métodos encontrados na literatura. No entanto, nenhum outro método é capaz de gerar personagens com características semelhantes herdadas dos seus ascendentes, simulando grupos de uma mesma família ou etnia, bem como permitindo o cruzamento entre indivíduos de grupos distintos, assim como ocorre na realidade. Com esta abordagem, é possível povoar ambientes dinamicamente à medida que a simulação evolui, gerando populações distintas, podendo ocorrer migração, com mistura de características étnicas e interações entre famílias em um ambiente virtual dinâmico, em tempo real. É importante assinalar que o método proposto é muito mais versátil do que a média ponderada das malhas dos progenitores semelhante ao que acontece com técnicas morphing. Além disso, embora use analogias biológicas de reprodução, este modelo é diferente de algoritmos genéticos e tem objetivos diferentes. Algoritmos genéticos são utilizados para convergir para um determinado resultado, removendo modelos indesejados durante o processo. O método utilizado neste estudo não deseja convergir para resultados específicos, mas gerar qualquer combinação possível de características geométricas a partir das informações herdadas dos ascendentes, como ocorre no processo reprodutivo biológico.

7. Referências

[1] Allen, B.; Curless, B.; Popovic, Z. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003),* 2003 July, 22(3): 587-594.

[2] Amabis, J. M.; Martho, G. R. *Biologia das populações*.Vol.3, 1st ed., Editora Moderna, São Paulo, São Paulo, Brasil, 1995.

[3] Blanz, V.; Vetter T. A morphable model for the synthesis of 3D faces. In: *Warren Waggenspack editor. 26th annual conference on Computer graphics and interactive techniques*; 1999 Aug 8-13; Los Angeles, CA, USA. New York: ACM Press/Addison-Wesley Publishing Co.; 1999. p. 187-194.

[4] Bui, T.D.; Poel, M.; Heylen, D.; Nijholt, A. Automatic Face Morphing for Transferring Facial Animation. In: *M.H. Hamza editor. Computer Graphics and Imaging*; 2003 Aug 13-15; Honolulu, Hawaii, USA. Calgary, AB, Canada: ACTA Press; 2003. p. 19-23.

[5] DeCarlo, D. ; Metaxas, D.; Stone M.. An anthropometric face model using variational techniques. In: Stephen N. Spencer editor. *25th annual conference on Computer graphics and interactive techniques*; 1998 Jul 19-24; Orlando, FL, USA. New York: ACM Press/Addison-Wesley Publishing Co.; 1998. p. 67-74.

[6] Golovinskiy, A.; Matusik, W.; Pfister, H.; Rusinkiewicz, S.; Funkhouser, T. A Statistical Model for Synthesis of Detailed Facial Geometry. *ACM Transactions on Graphics (Proc. SIGGRAPH 2006)*, 2006 Julay, 25(3): 1025-1034.

[7] Kähler, K.; Haber, J.; Seidel, H.. Reanimating the Dead: Reconstruction of Expressive Faces from Skull Data. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003)*, 2003 July, 22(3): 554-561.

[8] Kähler, K.; Haber, J.; Yamauchi, H.; Seidel, H. Head shop: generating animated head models with anatomical structure. In: *Michael F. Cohen and Nancy Pollard editors. ACM Siggraph/Eurographics Symposium on Computer Animation*; 2002 Jul 21-22; San Antonio, Texas, USA. New York, NY, USA: ACM; 2002. p. 55-63.

[9] Kolar, J. C.; Salter, E. M. Craniofacial Anthropometry – Practical Measurement of the Head and Face for Clinical, Surgical and Research Use. Charles C. Thomas Publisher, Ltd. Springfield, Illinois, USA, 1997.

[10] Noh, J.; Neumann, U. A Survey of facial modeling and animation techniques, *USC Technical Report 99-705*, 1998.

[11] Praun, E.; Sweldens, W.; Schroder, P. Consistent Mesh Parameterizations. In: *Stephen N. Spencer editor. 28th annual conference on Computer graphics and interactive techniques*; 2001 Aug 12-17; Los Angeles, CA, USA. New York: ACM Press/Addison-Wesley Publishing Co.; 2001. p. 179-184.

[12] Seo, H.; Magnenat-Thalmann, N. An automatic modeling of human bodies from sizing parameters. In*: Randy Pausch and Gary Bishop editors. Symposium on interactive 3D Graphics*; 2003 Apr 27-30; Monterey, CA, USA. New York, NY, USA: ACM; 2003. p. 19-26.

[13] Starr, C.; Taggart, R. *Biology: The unity and diversity of life*, 7th ed., Wadsworth Publishing Company, Belmont, California, USA, 1995.

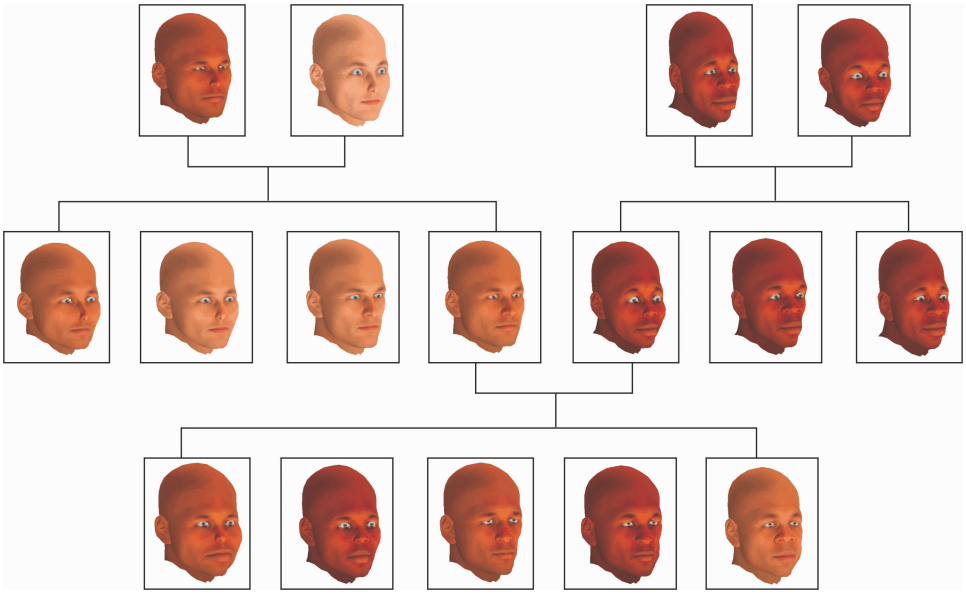


Figura 18. Árvore genealógica incluindo parentesco de modelos etnicamente diferentes.

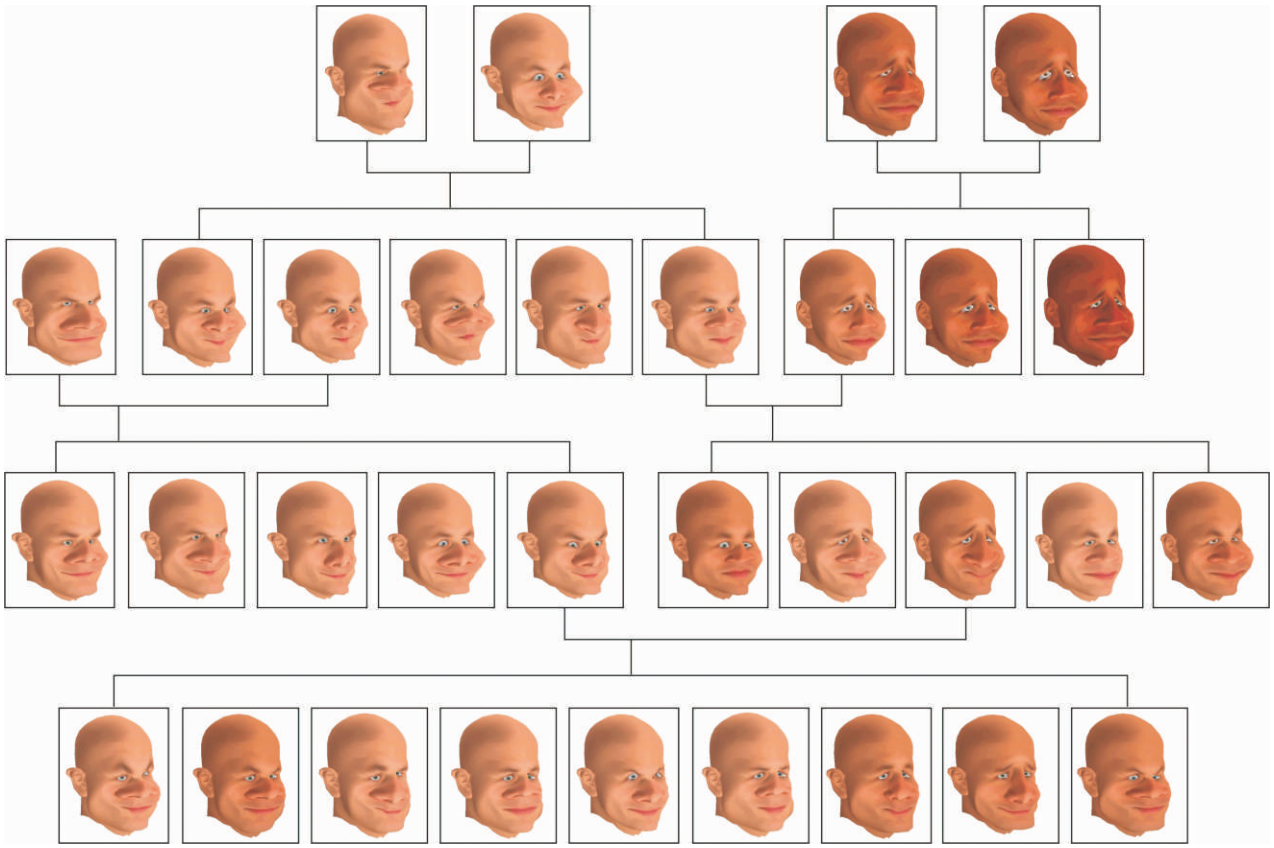


Figura 19. Árvore genealógica de modelos caricaturados.

Macacos-Prego (Cebus apella) Resolvem Problemas de Discriminação Simples em Ambiente Virtual

Carlos de Sousa Brito Neto, Manoel Ribeiro Filho, Olavo de Faria Galvão
Universidade Federal do Pará
carlosmcp@gmail.com, mrf@ufpa.br, olavo@pq.cnpq.br

Abstract

Virtual reality knowhow was applied to build an ecological environment to investigate the function of the number of correct vs incorrect stimuli in learning simple discriminations in four capuchin monkeys. They worked individually in an experimental chamber with a touchscreen displaying fruits on a forest background in a 5-stages fruits collect game. In successive stages fruits of the same type, two, and three types of fruits were presented, and finally the correct fruit was shifted. Touching correct fruit was rewarded by a food pellet. Touching other fruits erased the fruits for 3 seconds and same screen was displayed again. Two color contrasts between fruits and background were used whith similar results. This VR environment represents a second step in a larger project to investigate pre-requisites of cognitive behavior in Cebus apella which will involve other kinds of interaction beyond touchscreen.

Keywords: *Virtual reality environment, simple discrimination, Cebus apella.*

1. Introdução

As pesquisas de aprendizagem com animais utilizando estímulos visuais apresentados através de monitores carecem de um toque de realismo, dinamismo e naturalidade. Situações experimentais visualmente mais próximas do ambiente natural do animal poderiam, em princípio produzir interação mais produtiva e maior facilidade de aprendizagem de relações entre estímulos mais complexas, como relações abstratas de igualdade, simetria, e outras. A solução para obter ambientes interativos com manutenção do controle experimental foi buscada na realidade virtual. Uma tarefa que nos ambientes padrões com tentativas discretas de escolha entre um estímulo positivo (S+) e um ou mais negativos (S-), que gera uma diminuição gradual no número de erros, foi reprogramada em um ambiente virtual relativamente complexo para os padrões da experimentação em aprendizagem animal, para verificar a aprendizagem de discriminações simples com diversos S+ e S- apresentados simultaneamente.

Na literatura da área de aprendizagem complexa com primatas, se explora a capacidade dos primatas de detectar estímulos na tela de um monitor, e de mudar seu comportamento de escolhê-los em função de recompensas consistentemente dadas para uns e não para outros estímulos. No entanto, os estímulos são em geral formas estáticas em fundo uniforme. Assim, criamos um ambiente em que fosse possível inserir problemas de discriminação simples (apresentação do estímulo e posteriormente, a seleção do correto) em um ambiente tridimensional e com uma novidade em relação à literatura consultada: o uso de estímulos realísticos sobre um fundo complexo, rico em detalhes, com diversas formas irregulares e estímulos. Foram também usadas cores como dica adicional para a solução da tarefa de discriminação.

Ao longo dos anos, diversos trabalhos de discriminação simples foram realizados com a ajuda de aparatos, dispositivos eletrônicos e processos manuais, para mostrar os estímulos aos sujeitos. Hoje, com a inserção do computador nesse campo de pesquisa, estamos caminhando para novas possibilidades ou, podemos dizer, para um novo paradigma na forma de se realizar experiências com sujeitos não-humanos. Inicialmente, a contribuição do computador para pesquisa de cognição comparativa foi apenas para automatizar o processo de liberação das pelotas de reforço ou armazenar o resultados das experiências realizadas [11]. Os trabalhos com o uso do computador em que houvesse a interação do sujeito com o computador só vieram um pouco mais tarde com a sua utilização para exibir estímulos visuais no monitor. Barros et al. [2], por exemplo, mostravam figuras que poderiam ser selecionadas através de um toque no monitor com tela sensível. Primeiramente, era apresentada uma figura em tons de cinza, em contraste a uma superfície preta; quando o macaco tocava a figura, que servia de modelo, esta desaparecia e eram mostradas duas figuras, uma igual e outra diferente da figura modelo. Tocar na figura igual à modelo era recompensado com uma pelota de alimento.

Quando o sujeito errava, era apresenta uma tela escura sem nenhum elemento por alguns segundos (ver também Galvão et al. [6].

Outros tipos de experiências que envolviam interação com computador também foram realizadas, como em Leighty e Fragaszy [10], que procuraram elucidar como quatro macacos-prego (Cebus apella), adquiriam habilidade para usar um joystick (Figura 1a). A tarefa consistia de controlar um cursor na forma círculo branco, em contraste com um fundo preto que era envolto em uma área limite, também de cor branca. Eles ainda realizaram testes invertendo a posição dos eixos do joystick para descobrir se eles se adaptariam a esta nova condição. Em um estudo sobre discriminação de cores, Galvão et al. [17] apresentaram a macacos-prego 16 quadrados, sendo 15 de uma cor e um de outra, sobre fundo preto, e recompensavam a escolha do quadrado de cor diferente (Figura 1b). Iversen e Matsuzawa [8] realizaram uma experiência para investigar como primatas se guiavam em tarefas de interceptação. Eles tinham que capturar um alvo em movimento no monitor, um círculo também, que se movia de forma predizível e deveria ser guiado até a um alvo, representado por um quadrado branco ou um retângulo azul, sendo todos estes estímulos apresentados contra um cenário de superfície negra.

Recentemente os primeiros trabalhos com três dimensões emergiram. Experiências com primatas que fazem uso de realidade virtual podem ser vistas nos trabalhos de Ribeiro Filho et al. [13], e Astur e Washburn et al. [4], cujo trabalho possui uma semelhança muito grande com o trabalho de Leighty and Fragaszy [10], em que se desenvolveu um labirinto 3D como mostra a Figura 1d, o qual continha uma esfera e, como tarefa, os sujeitos deveriam também guiá-la, usando um joystick, pelos corredores de um labirinto virtual. Em Ribeiro Filho et al. [13], uma bandeja virtual continha várias peças diferentes e o sujeito deveria selecionar a peça igual à peça apresentada previamente, sozinha, na bandeja. O desempenho no ambiente virtual foi comparado com o desempenho no procedimento real, selecionando as peças em uma bandeja (ver Figura 1c).

Os trabalhos citados mostram exemplos de como os estudos comportamentais estão incorporando composições de estímulos cada vez mais reais. Neste trabalho, uso da discriminação simples em um ambiente virtual, foi uma opção que visava proporcionar um ambiente interativo com mais chances de sucesso, principalmente considerando-se que os quatro sujeitos tinham experiência com discriminações simples com apenas um S+ em cada tela e com fundo uniforme,

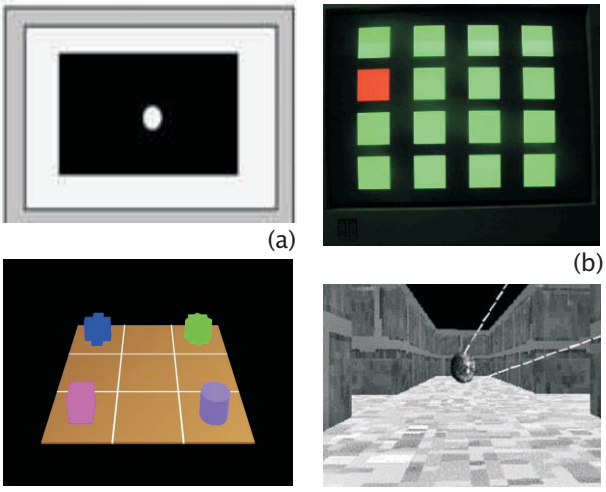


Figura 1. Tipos de composição de tela utilizados em estudos comportamentais com macacos. a) Leighty et al., b) Galvão et al., c) Ribeiro Filho et al., d) Washburn et al.

bem diferente dos demais trabalhos citados que buscam outros aspectos, como a discriminação condicional [13], a predição de alvos [8] e a habilidade motora [4].

Uma dificuldade com o uso de estímulos coloridos com o macaco-prego são as variações individuais na visão de cores. Jacobs [7] mostrou que os machos e um terço das fêmeas da espécie *Cebus apella* têm visão dicromática, não distinguindo verde e vermelho, e dois terços das fêmeas seriam tricromatas. Galvão et al [17] confirmaram, em um estudo de discriminação entre pares de cores, a dificuldade dos macacos-prego em discriminar pares verde-amarelo, vermelho-verde, e vermelho-amarelo.

Resolvemos, então, variar as cores dos objetos do ambiente com intuito de investigar o comportamento dos sujeitos diante de dois casos de contraste. No primeiro, foi usado o default do monitor RGB, e desenhadas as frutas no padrão humano, banana amarela, maçã vermelha e mamão amarelo-esverdeado (ver Figura 2). No segundo, RG, foi eliminado o azul e as frutas, a banana e o mamão ficaram mais "vermelhos" (ver Figura 13). Nos objetos do fundo, como as árvores e os arbustos, foi mantida a uma iluminação verde.

2. Descrição do ambiente virtual

2.1. Tecnologias usadas e técnicas empregadas

Os sujeitos da nossa experiência, primatas da família Cebidae, espécie *Cebus apella* e primatas de uma forma geral, vivem em ambientes bastante arborizados. Logo,

inserir elementos que lembrassem o seu meio natural seria uma forma de manter o interesse do sujeito pela tarefa contida no ambiente. O ambiente de realidade virtual aqui utilizado é classificado como não imersivo, pois o usuário não possui a sensação de estar dentro do ambiente virtual [12]. Para sua construção, foi utilizada a API Java 3D 1.5.1 [9] responsável pela renderização do cenário e dos objetos nele inseridos. Para a construção das interfaces gráficas do pesquisador, utilizamos a API Swing, encontrada no JDK 1.6.03 do Java. Os objetos foram construídos com a ajuda do software 3D Studio Max [1], o qual permite a manipulação direta com os objetos 3D em construção e uma maior definição na sua forma e aparência.

Figura 2. O ambiente com os três tipos de frutas utilizados para a realização da discriminação simples. Tela da etapa 1, em que uma das frutas aparece menor, como que distante.

As texturas empregadas no chão, fundo e frutas foram trabalhadas no Photoshop [14], melhorando o acabamento de cor e sombra das imagens, já que as frutas devem passar sensação visual de realismo. Também foi empregada a técnica de billboard, do Java3D que permite um objeto 2D ter um comportamento de um objeto 3D. Observando o ambiente na Figura 2, podemos notar que todas as árvores são idênticas, tendo apenas como diferença os estímulos, as frutas e suas posições. O motivo pelo qual não criamos as árvores respectivas a suas frutas, como usar uma bananeira para a banana, por exemplo, foi o de garantir que o sujeito se guiasse pelas formas das frutas exclusivamente.

Esse é um dos elementos de um sistema de realidade virtual que mantém o sujeito totalmente envolvido com a tarefa que está sendo realizada. Nesta investigação, queríamos levar a experiência do sujeito de seu mundo real para o mundo virtual. No mundo real, a seleção do comportamento por reforço descreve os mecanismos pelos quais uma mudança comportamental é orientada ao ambiente. Isto é, o que é selecionado pelo sujeito ou a sua ação é sempre uma relação comportamento-ambiente [5]. No mundo virtual, esta idéia deve ser mantida e, para que isto ocorra, deve haver um feedback constante das inter-ações do sujeito com o mundo virtual. Aqui, quando o sujeito selecionar a fruta certa, através do monitor sensível ao toque, o sistema de controle do ambiente virtual automaticamente a remove, além de prover a liberação de uma pelota de fruta. Caso contrário todas as frutas são retiradas por 3 segundos da tela, como forma de punição, e em seguida, reapresentadas ao sujeito.

Aqui estamos utilizando dois tipos de interação: navegação e seleção [12]. A seleção já foi brevemente explicada. Como dito, o sujeito pode "colher" as frutas que estão sobre as copas das árvores. A navegação é encadeada pela ação de seleção: o sujeito, ao recolher todas as frutas corretas da copa da árvore, habilita a passagem para o próximo estágio. Neste momento, o sujeito navega passivamente pelo ambiente até as árvores seguintes. Seus erros de escolha são armazenados em uma simples estrutura de dados em memória primária para serem posteriormente armazenados em um banco de dados. O banco utilizado foi o SQLite [15].

2.2. Modelagem UML do sistema

Aqui descreveremos o diagrama de casos de uso. O sistema é composto de dois atores: o sujeito e o pesquisador, como mostra a Figura 3.

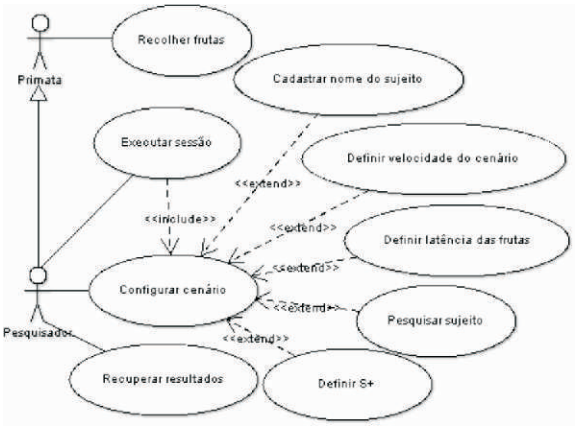


Figura 3. Diagrama de casos de uso.

O sujeito está associado a apenas um caso de uso, que lhe permite recolher as frutas que se encontram nas árvores do cenário. Ao ator pesquisador, é permitido também fazer uso das ações do sujeito, sendo possível

executar uma sessão, apresentação do ambiente virtual ao sujeito, e recolher as frutas do ambiente. Ainda, este pode definir a configuração do ambiente virtual, determinando qual o tipo de iluminação irão receber as frutas e as árvores, determinar qual será a fruta S+ (banana, maça ou o mamão), determinar a velocidade em que o ambiente caminhará entre uma etapa e outra, assim como a velocidade do reaparecimento das frutas após a retirada da tela e cadastrar ainda o nome do sujeito que utilizará o ambiente virtual. É possível também recuperar todos os dados coletados pelo sistema durante a sessão com o sujeito. A interface que possibilita ao pesquisador realizar tais ações é mostrada na Figura 4.

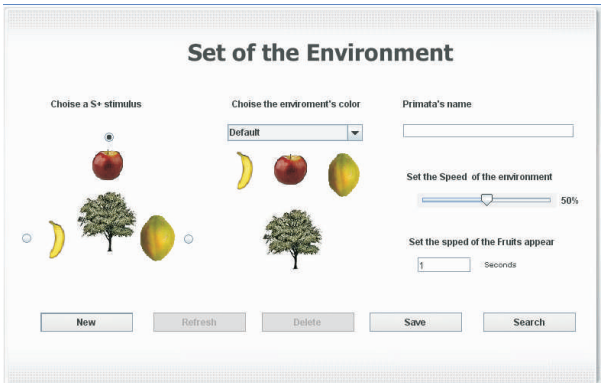


Figura 4. Interface do pesquisador.

Com relação à sua estrutura interna, o diagrama de classes ilustra a utilização de software por quatro principais classes, vide Figura 5. A classe resultado é utilizada pela classe ambiente para salvar os erros da sessão, a qual contém o resultados de uma ou vários ambientes definidos pelo pesquisador. A classe ambiente, por sua vez, só pode ser instanciada pela classe sessão, a qual usa as configurações criadas pela classe configuração Ambiente para construir o ambiente virtual.

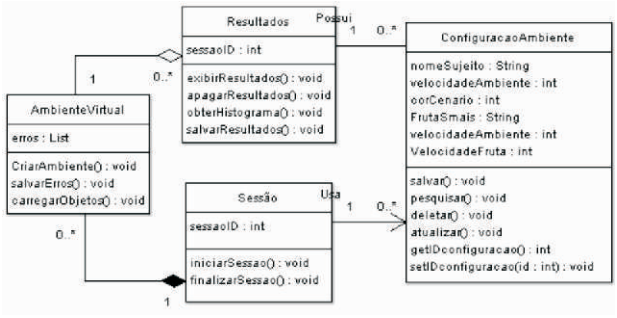


Figura 5. Diagrama de classes.

O diagrama de instalação é o mapeamento entre os componentes software e hardware [3]. Aqui, o sistema é composto de três nós: o monitor sensível ao toque, o qual permite a interação e está conectado a um computador, responsável por reproduzir o ambiente virtual em conjunto com o software JPrimatas e que por sua vez, está conectado via sua porta paralela ao dispensador de pelotas, que é acionado pelo software. O diagrama de instalação que ilustra esta topologia pode ser visto na figura 6.

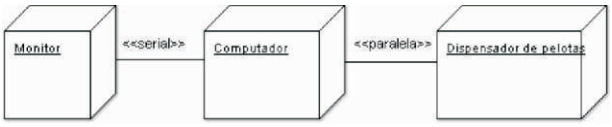


Figura 6. Diagrama de instalação.

3. Método

3.1. Sujeitos

Quatro sujeitos participaram, Eva (fêmea adulta), Guga e Adam (machos adultos), e Negão (macho jovem), todos da espécie *Cebus apella* ou macaco-prego. Todos os sujeitos descritos aqui já possuíam experiência em laboratório com o uso de computador para resolver problemas de discriminação simples entre estímulos apresentados em uma tela sensível ao toque. Os animais recebiam sua alimentação regular após a realização das experiências.

3.2. Aparato

Foi utilizada uma câmara transparente, de arestas de alumínio e paredes de acrílico, com dimensão de 0,60 x 0,60 x 0,60 m, a qual possuía uma janela lateral com grades de ferro que permitia ao sujeito o acesso a um monitor ECS colorido 14" sensível ao toque . Este aparato é mostrado na Figura 7. Um monitor LCD 17", também sensível ao toque (Elo Entuitive) foi também usado para permitir filmagem com melhor imagem.

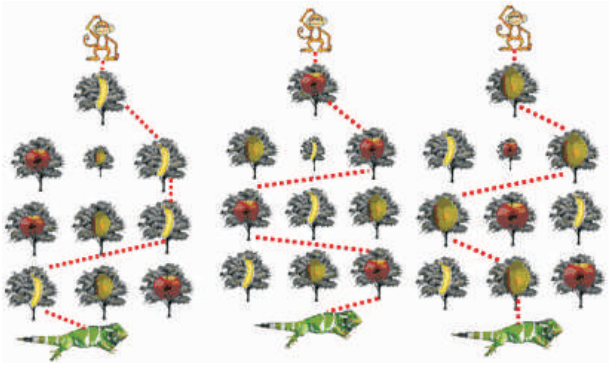
Figura 7. Câmara utilizada na experiência. À esquerda o monitor com a tela sensível, a direita o sujeito dentro da câmara experimental, olhando a tela através da janela e tocando a fruta.

3.3. Procedimento geral

Com cada sujeito, realizamos um total de 12 sessões. Uma sessão consistia de uma tarefa com cinco etapas; Etapa Zero: havia uma única árvore com quatro frutas de um tipo, o S+ (Figura 9). Etapas 1, 2, e 3: havia um grupo de três árvores, cada árvore com 3 frutas de um tipo, sendo S+ a do mesmo tipo das frutas da Etapa Zero (Figura 2); Etapa 4: um lagarto em movimento (Figuras 11 e 12). A transição entre as etapas ocorria após a coleta da última fruta correta da etapa, na forma de "zoom in".

Toques nas frutas S+ eram "acertos", e eram recompensados com uma pelota de ração sabor de fruta. Na Etapa 4, tocar o lagarto produzia uma pelota de ração e o fim da sessão (ver Figura 12).

Em seguida o experimentador programava uma nova sessão com outra fruta como S+ e definia o padrão das cores (default: RGB, ou R-G: sem o azul) para a nova sessão.



(a) (b) (c)

Figura 8. Disposição (centro, esquerda ou direita) das frutas nas etapas sucessivas que compunham uma sessão para: a) banana b) maça c) mamão. A linha de cima representa a Etapa 0, com um único tipo de fruta. As linhas abaixo representam as demais Etapas, mostrando a disposição dos tipos de fruta. A linha pontilhada indica o S+ em cada sessão. Cada sujeito passou por sessões com banana, maçã e mamão como S+ por quatro vezes.

Figura 9. Primeira etapa com o estímulo S+, no caso banana. Padrão de cores RGB.

Após terem recolhido todas as frutas iguais da primeira tela, os primatas navegavam passivamente pelo ambiente até um grupo de três árvores, cada qual com três frutas em sua copa. Uma destas árvores continha as bananas novamente, os S+ nessa etapa, e as demais com maçã e mamão, sendo o mamão em tamanho menor, como se distante. Tocar na maçã apagava todas as frutas por três segundos, tocar no mamão (em tamanho pequeno) não tinha efeito. Novamente o sujeito deveria recolher todas as bananas para completar a tarefa para novamente navegar passivamente ao próximo grupo de árvores, agora todas as três frutas em tamanho igual, portanto com 3 S+, as bananas, e 3 S-, as maçãs e os mamões. Assim se repetia por mais duas vezes com a disposição das frutas diferente da etapa anterior. Após o primata ter explorado todas as etapas, a etapa final era apresentada, na qual um lagarto em movimento aparecia na tela (Figuras 11 e 12).

Figura 10. Visão interna da câmara. Acima aparecem as caixas em que caem as pelotas de ração com sabor de fruta.

A escolha do lagarto alude ao fato de que primatas da espécie *Cebus apella*, além de se alimentarem de frutas, raízes, brotos e flores também se alimentam de pequenos animais. Logo, buscamos examinar o comportamento do primata perante a dinâmica de elementos no ambiente virtual. Iversen and Matsuzawa [8], já citados, realizaram um trabalho similar, mas com menos elementos, com os alvos e fundo de textura uniforme e alto contraste, enquanto neste trabalho os alvos, e principalmente o fundo, são imagens complexas no que diz respeito ao número de detalhes e cor, de tal modo que os alvos ficam como que camuflados no ambiente virtual. Completada a sessão o monitor era afastado e o experimentador programava a sessão seguinte, com outro S+, alternando banana, maçã e mamão, consecutivamente.

Figura 11. O lagarto, que surge na última etapa.

Figura 12. Um sujeito tocando o lagarto na Etapa 4, para receber uma recompensa e encerrar a sessão.

Figura 13. Cenário com vermelho nos alvos e fundo verde.

Primeiro, procuramos investigar se os sujeitos se adaptariam ao ambiente 3D e conseguiriam identificar os estímulos diante de um ambiente teoricamente de difícil compreensão aos sujeitos, pela diferença em relação aos ambientes que eles estavam acostumados e de outras pesquisas aqui citadas em que não há muita variedade de forma, cor ou brilho. Assim, seria um passo importante estabelecer o contato entre o sujeito e o ambiente virtual. A fruta que aparecia na primeira tela seria S+ naquela e nas três telas seguintes. Na primeira sessão usamos a banana como S+, a fruta a ser discriminada do fundo na etapa zero e do fundo e das demais frutas que se encontram nas demais árvores nas etapas 1 a 3 (ver Figura 10). Como citado anteriormente, o ambiente virtual foi criado como um jogo, com etapas que são superadas de acordo com o sucesso das tarefas impostas ao sujeito, no caso tocar todas as frutas corretas (veja na Figura 11 um dos sujeitos olhando o ambiente virtual).

A ordem das frutas corretas (S+) de cada sessão variou. Cada fruta foi a primeira a ser S+ em cada grupo de três sessões. Banana, maçã, mamão; Maçã, mamão, banana; Mamão, banana, maçã. No último grupo de três sessões repetiu-se a ordem do primeiro.

4. Resultados

Todos os primatas conseguiram identificar e tocar as imagens das frutas presentes no ambiente virtual mesmo sendo estímulos com características irregulares em sua textura, cor e forma. A figura da banana, mais que as outras duas, maçã e mamão, parece se destacar bem contra o cenário do ambiente virtual. O mais impressionante foi a forma como eles interagiram com os estímulos, tocando com precisão em cima da imagem da fruta praticamente desde o início da exposição na primeira sessão. Os quatro sujeitos mostraram diminuição no número de erros ao longo das sessões.

Com exceção do sujeito Eva, em uma sessão, todos os sujeitos finalizaram a sua tarefa. Contudo, a sessão que não foi concluída foi a última de uma série seguida, podendo ser atribuída a interrupção da interação à saciação das pelotas.

Houveram 13 sessões do Guga e 8 da Eva, 8 do Negão, e 9 do Adam completamente sem erros.

Todos os sujeitos apresentaram a maioria dos erros nas primeiras três sessões. Eva foi a que teve queda no numero de erros mais suave, mas terminou com uma sessão perfeita, sem erros.

Não houve diferença visível entre as sessões RGB e RB. O único efeito encontrado foi o da queda no número de erros ao longo das sessões.

Será descrito em maior detalhe o desempenho da participante Eva, o desempenho de todos os quatro participantes foi resumido em gráficos mostrando o

número de erros por sessão, na seqüência em que as sessões ocorreram.

Na primeira sessão, o primeiro sujeito, Eva, conseguiu coletar todas as frutas da Etapa Zero em 30 segundos. Esse tempo deve-se ao fato de haver pausas durante a mastigação da pelota. Quando a Eva terminou de coletar todas as frutas e a tela da Etapa 1 foi apresentada, através de um efeito de zoom a partir da Etapa Zero, ela selecionou a banana no início da tarefa sem hesitar muito. Em seguida, na Etapa 2, ela cometeu alguns erros, tocando-a diretamente na maçã e, apesar do erro, novamente verificamos que o sujeito estava reconhecendo o estímulo na tela do monitor. Finalmente ela veio a tocar na banana que estava bem no canto no lado direito do monitor, e o sujeito passou a tocar na área da árvore com as bananas e conseguiu completar a tarefa. Na Etapa 3, com a disposição das frutas diferente da anterior, Eva tocou as frutas que estavam à direta do monitor, tendência provocada possivelmente pela localização da banana no estágio anterior, cometendo 11 erros. Entretanto, voltou a tocar nas bananas e finalmente chegou à Etapa 4, com o lagarto movimentando-se debaixo da árvores. Sua posição era mudada a cada 1 segundo, dando a idéia de movimento. O sujeito demorou para tocar o lagarto apenas na primeira vez. Eventualmente Eva tocou o lagarto e finalizou a tarefa. Nós observamos que o que pode ter facilitado a seleção do animal foi o contraste com uma parte do cenário que é mais escura. O lagarto foi de difícil reconhecimento possivelmente por possuir a mesma cor do cenário, como que camuflado. Isso nos levou mais tarde a aumentar a velocidade da movimentação do lagarto, para que chamasse mais a atenção dos primatas, o que realmente aconteceu.

Finalizada a primeira sessão, programou-se uma nova. A segunda sessão foi configurada para a maçã ser o S+. Na primeira etapa que contém apenas uma árvore, o sujeito conseguiu rapidamente tocar nas maçãs. Na

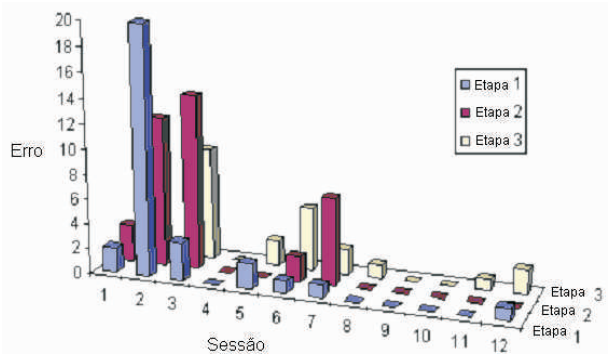


Figura 14. Desempenho de Guga. Ver legenda da Fig. 15.

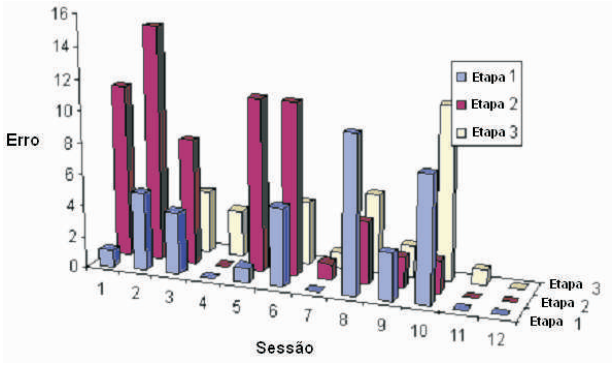


Figura 15. Desempenho de Eva. Número de erros em cada sessão. As sessões ocorreram na seqüência da esquerda para a direita e da frente para trás. Cada grupo de três sessões tinha uma fruta como alvo. Grupos 1, 4, 7, 10: Banana, 2, 5, 8, 11: Maçã; 3, 6, 9, 12: Mamão. Nas sessões 7, 8, 9 foi usado o padrão de cores RG, nas demais RGB. Ver texto.

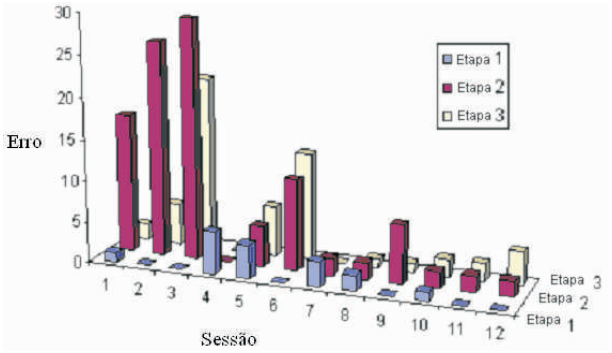


Figura 16. Desempenho de Negão. Ver legenda da Fig. 8.

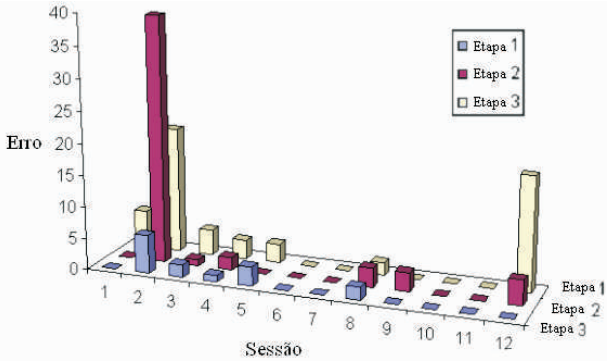


Figura 17. Desempenho de Adam. As sessões ocorreram na seqüência da esquerda para a direita e da frente para trás. Cada grupo de três sessões tinha uma fruta como alvo. Grupos 1, 4, 7, 10: Banana, 2, 5, 8, 11: Maçã; 3, 6, 9, 12: Mamão. Nas sessões 4 a 9 foi usado o padrão de cores RG, nas demais RGB. Ver texto.

Etapa 1, novamente, ela selecionou a fruta S+ como ocorrido na primeira sessão. Em seguida, cometeu cinco erros até alcançar o estágio seguinte. Com o mamão, Eva se saiu melhor, apesar da fruta se camuflar

entre as folhas por sua cor ser a mesma da folhagem e do fundo do cenário.

5. Conclusão

Trabalhos anteriores confirmaram que macacos podem interagir com ambientes virtuais. Nós conseguimos verificar que eles ainda mantém a interação com a complexidade aumentada em uma tarefa ainda não utilizada antes, de discriminação simples com diversos estímulos positivos e negativos simultaneamente.

Os resultados confirmaram que os macacos-prego com habilidades para resolver problemas de discriminação simples em um ambiente 2D simples podem transferir esse repertório para um ambiente 3D complexo, assim como outras espécies de macacos [16]. Com exceção de uma, todas as sessões terminaram após a coleta de todas as frutas e o lagarto. Os macacos conseguiram identificar os estímulos e finalizar a sua tarefa, indicando que poderemos inserir novos problemas em um ambiente virtual, explorando outros tipos de interação e navegação em um ambiente virtual e estudar outros processos comportamentais em condições experimentais que mimetizam com muito maior riqueza as condições naturais de interação comportamento-ambiente do que os ambientes visuais comumente usados em estudos comportamentais com primatas. A velocidade de aprendizagem foi acima da que se verifica nos estudos com tentativas discretas, e um só S+ por tentativa, com estímulos simples sobre fundo uniforme.

A perspectiva da aplicação da realidade virtual no estudo experimental da aprendizagem de relações entre estímulos é promissora, particularmente com a adição de novas formas de interação além do toque em tela sensível, como a possível detecção da direção do olhar e outras.

6. Agradecimentos

À FADESP, pela bolsa de mestrado ao primeiro autor. Ao CNPq, pela Bolsa de Produtividade ao terceiro autor. Ao CNPq, FINEP e NIH pelo financiamento da Escola Experimental de Primatas, UFPA.

7. Referências

[1] 3D Studio Max disponível em <http://www4.discreet.com/3dsmax>.

[2] R.S. Barros, O.F. Galvão, and W. V. McIlvane, "Generalized identity matching-to-sample in *Cebus apella*". *The Psychological Record*, 2002, pp. 441-460.

[3] Bezerra, E., Princípios e Análise e projetos de sistemas com UML. Campus, Rio de Janeiro, 2002.

[4] D. A. Washburn, R. S. Astur, "Exploration of virtual mazes by rhesus monkeys (*Macaca mulatta*)". *Animal Cognition*, 2003, pp. 161–168.

[5] Donahoe, J. W., & D. C. Palmer, Learning and complex behavior, Allyn & Bacon, Boston, 1994.

[6] O. F. Galvão, R. S. Barros, S. B.Lima, C.M.Lavratti, J.R. Santos, A.L.F. Brino, W. V. Dube, and W. J. McIlvane, "Extent and Limits of the Matching Concept in *Cebus apella*: A Matter of Experimental Control?", *The Psychological Record*, 2005, pp. 219-232.

[7] G. H. Jacobs, and J. F. Deegan, "Cone pigment variations in four genera of new world monkeys". *Vision Research*, 2003, pp. 227–236.

[8] I. Iversen, T. Matsuzawa, "Development of interception of moving targets by chimpanzees (*Pan troglodytes*) in an automated task". *Animal Cognition*, 2003, pp. 169–183.

[9] Java3D. The Java 3D API, https://java3d.dev.java.net/, Acessado em 23 de Novembro, 2007.

[10] K. A Leighty, and D. M. Fragaszy, "Joystick acquisition in tufted capuchins (*Cebus apella*)". *Animal Cognition*, 2003, pp. 141-148.

[11] K.A. Leighty, and D. M. Fragaszy, "Primates in cyberspace: using interactive computer tasks to study perception and action in non-human animals", *Animal Cognition*, 2003, pp. 137-139.

[12] Kirner, C. R., "Fundamentos de Realidade Virtual", VIII Symposium on Virtual Reality, SBC, 2006, pp.2-21

[13] M.R. Filho, A. A. Leal, and O.F. Galvão, "An experimental session for Primates using VR Desktop which imitates a procedure of matching objects to sample", IX Symposium on Virtual Reality, 2007, Petrópolis, 2007, pp. 303-305

[14] Photoshop, http://www.adobe.com/br/products/photoshop/photoshop/ acessado em 23 de Novembro de 2007.

[15] SQLite, http://www.sqlite.org/, Acessado em 23 de Novembro, 2007.

[16] N. Sato, H. Sakata, Y. Tanaka, M. Taira, "Navigation in virtual environment by the macaque monkey" *Behavioural Brain Research*, 2004, pp. 287–291.

[17] O.F. Galvão, P. R. K. Goulart, S. Makiama, K. L. S. Marques, A.R. Fonseca, "Color vision in *Cebus apella*: Assessment of color-discrimination using capabilities of regular CRT monitor and color formatting tool of Windows XP", 2007, Em preparação.

Uma ferramenta de auxilio ao ensino de história por meio da reconstituição de ambientes históricos utilizando tecnologias de Realidade Virtual não-imersiva

Rodrigo Vasconcelos Arruda[1], Marcos Wagner S. Ribeiro[1], Alexandre Cardoso[2], Wender Antônio da Silva[1], Ezequiel Roberto Zorzal[2], Edgard A. Lamounier Júnior[2], Nadabe Cardoso O. Alves Fortes[3]

ILES/ULBRA Itumbiara-GO - Grupo de Realidade Virtual de Goiás
rodrigoitb@gmail.com, {marcos_wagner}, {wender_silva}@yahoo.com.br,
Faculdade de Engenharia Elétrica - UFU – Uberlândia-MG
{alexandre}, {ezorzal}, {lamounier}@gmail.com
Universidade Estadual de Goiás
Grupo de Realidade Virtual de Goiás
nadabef@gmail.com

Abstract

This article presents the development of a prototype of software using techniques of Virtual Reality not immersive, with the case study the historic center of the city of Itumbiara - Goiás. The goal is to rebuild virtually the Cathedral Santa Rita de Cássia and the Republic Square in the beginning of its foundation, exposing the motivations and tools used, with the aim to rebuild a historic environment, which is no longer exists. Thus it is expected to contribute and provide an environment geared to the teaching of history and especially for the rescue of the historical and cultural city.

Resumo

Este artigo apresenta o desenvolvimento de um protótipo de software utilizando técnicas de Realidade Virtual não-imersiva, tendo como estudo de caso o centro histórico da cidade de Itumbiara – Goiás. O objetivo é reconstituir tridimensionalmente a Catedral Santa Rita de Cássia e a Praça da República nos primórdios de sua fundação, expondo as motivações e ferramentas utilizadas, tendo como intuito a reconstituição de um ambiente histórico, que já não existe mais. Deste modo espera-se contribuir e constituir com um ambiente voltado para o ensino de História e principalmente para o resgate histórico-cultural da cidade.

1. Introdução

No mundo contemporâneo, onde a tecnologia se encontra em basicamente todos os setores da sociedade, vê-se a necessidade de impulsionar sua utilização como apoio aos métodos tradicionais de ensino de modo atrativo e eficiente.

Nesse contexto, o uso da computação como ferramenta de apoio à educação tem se tornado cada vez mais indispensável para a melhoria da qualidade e dinamismo do processo de ensino.

Dentre as mais variadas técnicas computacionais e suas diversas possibilidades de aplicação, podemos destacar a Realidade Virtual (RV), uma área que vem cada vez mais conquistando espaço e adeptos.

Alguns são os fatores que tornam a RV uma técnica viável e interessante de se utilizar no ensino e no resgate histórico-cultural, dentre as quais destacamos: pelos métodos tradicionais de ensino, o leitor tem uma limitação visual dos objetos em estudo por meio de imagens bidimensionais e textos, enquanto utilizando-se de ambientes tridimensionais o usuário tem a possibilidade de ter múltiplas visões dos objetos que o compõe, permitindo-lhe uma análise mais detalhada e melhor compreensão, além de poder interagir com esse mundo. Além disso, pesquisadores apontam como principais vantagens na utilização de técnicas de RV para fins educativos a motivação, poderio de ilustração de características e processos, permite a visualização de detalhes de objetos [1].

O presente artigo tem como objetivo a apresentação de um protótipo de software desenvolvido com técnicas de RV que possa auxiliar no resgate e ensino da história, assim, pode ser utilizado por pessoas diversas, professores e alunos no processo de abstração dos conteúdos de história, onde basicamente será reconstituído um ambiente regional, o qual, muitas pessoas desconhecem, pois não está nos livros de História tradicionais usados nas escolas.

1.1.Realidade Virtual

Realidade virtual é uma interface avançada para aplicações computacionais, que permite ao usuário

navegar e interagir, em tempo real, com um ambiente tridimensional gerado por computador, usando dispositivos multisensoriais [2]. A Realidade Virtual é categorizada como não imersiva, quando o usuário é transportado parcialmente ao mundo virtual, por meio de uma janela (monitor ou projeção, por exemplo), mas continua a sentir-se predominantemente no mundo real [3].

2. Trabalhos Relacionados

Diversos são os trabalhos existentes que ressaltam a importância da Realidade Virtual nas mais variadas áreas do conhecimento, das quais cita-se o ensino de redes de computadores [4], utilização da Realidade Aumentada em museus para visualização e compreensão de coleções artísticas, técnicas ou biológicas [5], o ensino da fotossíntese usando da Realidade Aumentada [6], o uso da RV no ensino de Matemática fundamental [7], um jogo RPG feito em Realidade Virtual abordando conteúdos multidisciplinares [8], a RV na reconstrução de ambientes históricos [9], dentre outros.

Em todos os sistemas avaliados, conclui-se que a Realidade Virtual é uma ferramenta de grande importância para o auxílio na educação e principalmente na reconstituição ou visualização de ambientes complexos, abstratos ou que não existem mais, atuando em diversas áreas do saber, motivando e tornando o aluno mais ativo nas aulas, proporcionando uma melhor absorção do conteúdo ministrado. Neste sentido é ideal também para demonstrar a História de um determinado lugar ou a sua reconstrução.

3. O ambiente histórico

O ambiente histórico escolhido para a realização deste trabalho é o centro histórico da cidade de Itumbiara nos primórdios de sua fundação, motivado por fatores como: pouco conhecimento dos cidadãos itumbiarenses com relação à sua história, enorme riqueza cultural da cidade pouco explorada, falta de uma ferramenta que possibilite alcançar um maior número de pessoas. O uso da Realidade Virtual na realização deste trabalho visa estimular e motivar o usuário a conhecer toda essa riqueza histórica e cultural descrita em poucos livros.

4. Métodos aplicados

Inicialmente foi realizado um estudo das características arquitetônicas do ambiente em questão, por meio do material disponibilizado na Casa da Cultura de Itumbiara-Goiás que conta em seu acervo com fotografias e objetos originais da época, obras de arte como quadros e uma maquete, trabalhos produzidos por artistas plásticos da cidade que vivenciaram a época. Posteriormente foi realizada a modelagem dos objetos e a implementação do protótipo, que são tratados a seguir.

4.1. Implementação

A modelagem foi feita utilizando-se o 3ds Max. A escolha desta ferramenta deve-se a ela ser uma das mais utilizadas na área de computação gráfica atualmente, graças à sua diversidade de funcionalidades técnicas bem como apoio didático de fácil acesso e que oferece uma grande gama de recursos, facilitando o processo de modelagem. Outro fator importante para a escolha do 3ds Max é que ele permite importar e exportar objetos para os principais formatos tridimensionais existentes, neste caso optou-se pelo formato VRML. A modelagem foi dividida em 3 fases: modelagem da Catedral, a modelagem do Coreto e a modelagem dos demais objetos que formam a praça (banco, canteiros, etc). A figura 1 abaixo é uma foto original da época.

Figura 1 – Antiga Catedral Sª Rita de Cássia.

De seu projeto original, praticamente nada sobrou da Catedral. Hoje Itumbiara conta com uma igreja moderna. A representação tridimensional da Catedral é mostrada pela figura 2 a seguir.

Figura 2 – Catedral Santa Rita de Cássia em 3D.

A figura 3 abaixo, é uma representação em maquete do

cenário da Praça, confeccionada por um artista plástico da cidade.

Figura 3 – Maquete da Praça.

A figura 4 mostra a imagem do objeto tridimensional, modelado no 3ds Max, que representa o Coreto.

Figura 4 – Coreto em 3D.

Feito a modelagem dos objetos utilizando o 3ds Max, estes foram exportados para a linguagem VRML (Virtual Reality Modeling Language), que é uma linguagem de programação destinada a criação de ambientes virtuais 3D.

Arquivos que simulam ambientes tridimensionais em VRML são, na verdade, uma descrição textual na forma de textos ASCII, que são interpretados por um plug-in VRML e visualizados diretamente no navegador de internet [1].

O fato de o código VRML ser interpretado por um plug-in, merecendo destaque o Cortona e o Cosmo Player, lhe traz vantagens em relação a outras linguagens e bibliotecas, como o código funcionar independente de plataforma e navegador utilizado, dependendo apenas da instalação do plug-in adequado e a facilidade de sua disponibilização na Internet, pois gera códigos muitos pequenos.

5. Funcionamento do Sistema

O protótipo é visualizado por meio do browser de Internet (Criou-se uma home page para acesso ao sistema e ao conteúdo em RV). Para isso faz-se necessário a instalação de um plug-in interpretador de VRML, dentre os quais podemos destacar o Cortona, que vem constantemente sendo aprimorado pelo seu fabricante. O conteúdo da página conta a história da cidade de Itumbiara, no estado de Goiás, desde a época de sua fundação até os tempos atuais, bem como a sua importância no cenário goiano. Dentro do ambiente virtual tridimensional, o usuário pode navegar livremente pelo cenário, utilizando simplesmente as setas do teclado (←,↑,↓,→). No interior da Catedral ele tem a possibilidade visualizar os detalhes arquitetônicos da época, as imagens sacras, o altar entre outros. Da mesma forma, o lado exterior conta com uma bela paisagem que era a praça, com seus jardins e canteiros. Dentro do coreto, no andar superior, o ponto de vista do usuário fica ainda mais belo, vendo todos os cantos da praça com uma visão mais elevada.

6. Análise dos Resultados

O protótipo foi apresentado a uma amostra de 30 alunos de 5ª a 8ª série do ensino fundamental de Itumbiara. Estes alunos passaram previamente por um rápido treinamento sobre as funções do protótipo, bem como a finalidade e os objetivos que o mesmo pretende alcançar, que é o resgate de parte da história e da cultura de Itumbiara. Foram-lhes apresentados alguns materiais como fotos, quadros e maquetes para que pudessem ter um senso crítico maior quanto a real necessidade e contribuição do protótipo de software.

Após a utilização do protótipo, solicitou-se ao avaliados que respondessem um questionário, elaborado com base em preceitos da ISONORM NBR 9241-11 que define a usabilidade e explica como identificar a informação necessária a ser considerada na especificação ou avaliação de usabilidade de um computador em termos de medidas de desempenho e satisfação do usuário.

O protótipo foi avaliado quanto a sua interface, navegação, comandos, facilidade e finalidade.

De modo geral, todos os usuários que experimentaram o protótipo disseram que o mesmo possui uma Interface Gráfica agradável, que motiva-os e torna o conhecimento do conteúdo mais atraente.

A Interface Gráfica foi avaliada como agradável por todos os usuários, sendo que 40% a avaliaram como boa e 60% a avaliaram como ótima. Isso pode ser explicado em parte pela simplicidade que o protótipo propõe, mas também por ser uma inovação tecnológica do ponto de vista de conhecido histórico e cultural.

Outro ponto avaliado a favor do protótipo foi o quesito

de navegação, sendo avaliado como bom por 67% dos avaliados. Apesar de praticamente todos os usuários que o experimentaram terem declarado que tinham tido apenas contatos básicos com o computador, estes não apresentaram grandes dificuldades para navegar pelo ambiente virtual, limitando-se as dificuldades de coordenação motora ao operar o mouse.

Em relação aos comandos de navegação do protótipo, é um ponto que merece mais atenção, uma vez que foi o que teve a pior avaliação dos usuários, ou seja, 47% avaliaram como regular e 13% como baixa.

Isso pode ser justificado devido ao fato de estes comandos serem pré-definidos pelo Cortona, plug-in que interpreta o código VRML, e que foi a escolha para utilização na apresentação.

Nessa análise pode-se perceber que os comandos de navegação do Cortona são simples para as pessoas que já tem afinidade com a Realidade Virtual, mas não para usuários em geral, pois não possui características tão intuitivas quanto o esperado.

Portanto, existe a necessidade de se implementar comandos próprios que sejam mais intuitivos para o público alvo pretendido.

De modo geral, apesar de algumas limitações encontradas pelos usuários, a grande maioria (60%) avaliou como boa e outros (27%) avaliaram como ótima a facilidade de utilizar o protótipo.

Parte dos usuários (13%) avaliou a facilidade de navegação como regular, fazendo algumas observações e sugerindo algumas soluções para que se chegue ao produto ideal.

Por fim, pode-se tirar de conclusão que os usuários de forma geral avaliaram o protótipo como uma boa iniciativa e que atende à sua finalidade, que é trazer o conhecimento e conseqüentemente o resgate histórico da cidade. Sua finalidade foi avaliada por 40% dos usuários como ótima e 60% como boa.

7. Conclusões e trabalhos futuros

O protótipo mostrou-se bastante eficaz, uma vez que os objetivos propostos, que era a criação de uma ferramenta que auxiliasse no resgate histórico da cidade foram alcançados, mas ainda não pode ser considerado eficiente e efetivo, sendo necessário para isto a implementação de algumas ações que serão propostas para trabalhos futuros, tais como: um maior nível de interação do usuário com o ambiente e seus objetos, a implementação de um ambiente virtual distribuído que permita os usuários interagirem entre si, aplicar o software em um ambiente imersivo e modelar toda a cidade de Itumbiara.

8. Referências

[1] A. Cardoso, et. al, Tecnologias para o desenvolvimento de sistemas de realidade virtual e aumentada, Ed. Universitária da UFPE, Recife, 2007.

[2] C. Kirner, R. A. Siscoutto, "Fundamentos de Realidade Virtual e Aumentada", *Realidade Virtual e Aumentada: Conceitos, Projeto e Aplicações*, pp. 9-21, maio 2007.

[3] R. Tori, C. Kirner, "Fundamentos de Realidade Virtual", *Fundamentos e tecnologia de Realidade Virtual e Aumentada*, pp. 9-28, maio 2006.

[4] E. B. Hassan, "Laboratórios Virtuais 3D para Ensino de Redes de Computadores", *XIV Simpósio Brasileiro de Informática na Educação*, novembro 2003.

[5] F. Braga, "Realidade Aumentada em Museus: As Batalhas do Museu Nacional de Belas Artes", fevereiro 2007.

[6] M. W. Ribeiro, W. A. Silva, "O ensino do fotossíntese usando Realidade Aumentada", *IX Symposium on Virtual and Augmented Reality*, maio 2007.

[7] S. F. Sampaio, et. al, "Associando Realidade Virtual ao Ensino de Matemática fundamental", *IX Symposium on Virtual and Augmented Reality*, maio 2007.

[8] M. A. Tobaldini, M. Martinelli, J. D. Brancher, "Arquiteturas Históricas no Ambiente de um Jogo de RPG Educacional Computadorizado", novembro 2006.

[9] A. Gonçalves, "Reconstrução de Ambientes Históricos Utilizando VRML: o caso do Fórum Flaviano de Conimbriga", maio 2004.

Um Sistema de Realidade Virtual Desktop para o Ensino de História

Manoel Ribeiro, Ricardo Damasceno, Felipe Reis, Fabrício Silva, Messias Nascimento

Universidade Federal do Pará

Instituto de Tecnologia, Laboratório de Realidade Virtual.

mrf@ufpa.br ricardordm@yahoo.com.br, felipevr@ibest.com.br, Commander.fabricio@gmail.com, messiasjan@yahoo.com.br

Abstract

This paper presents a system of Desktop Virtual Reality of aid in teaching history. The system allows the user to feel immersed, learning while navigates and interacts in the virtual environment. The case study is the city of Belém of Pará of the XIX century. A virtual city of Belém was generated through the use of a framework developed for generation of virtual cities. The user controls avatars to interact with the virtual environment.

Resumo

Este trabalho apresenta um sistema de Realidade Virtual Desktop de auxilio ao ensino de história. O sistema possibilita que o aprendiz se sinta imerso, aprendendo enquanto navega e interage no ambiente virtual. O estudo de caso é a cidade de Belém do Pará do século XIX. A cidade virtual de Belém foi gerada através do uso de um framework desenvolvido para geração de cidades virtuais. O usuário controla avatares para interagir com o ambiente virtual.

1. Introdução

Aplicações computacionais aplicadas à educação vêm crescendo nos últimos anos, entre estas, algumas técnicas têm-se destacado como multimídia, interação baseada em vídeo, realidade aumentada, realidade virtual e jogos eletrônicos. A interação com o usuário, característica importante das últimas quatro técnicas citadas, é fundamental em aplicações ao auxilio à educação.

Esse artigo apresenta um sistema de Realidade Virtual Desktop de auxilio ao ensino de história, tendo como estudo de caso a cidade de Belém do Pará do século XIX. Além disso, apresenta um framework para geração de cidades virtuais, que foi utilizado para a criação da cidade de Belém. O ambiente virtual desenvolvido é caracterizado por um avatar, que vestido com roupas da época, sob o controle do usuário, percorre a cidade, com a tarefa de localizar, e obter informações sobre diversos prédios importantes da história da cidade de Belém.

2. Criação de cidades virtuais

A geração de cidades virtuais tem sido objeto de estudo e dentre as inúmeras técnicas encontradas na literatura faremos uma breve discussão de algumas técnicas que serviram de base para o *framework* desenvolvido.

O trabalho proposto por [1] apresenta um *framework* genérico para a criação semi-automática de ambientes urbanos complexos. A criação semi-automática é feita através de mapas topográficos, imagens, texturas e textos, que definem o tipo das zonas residências. Estes resultam na construção do terreno, em parâmetros e restrições, e numa malha planar topológica e na divisão dos lotes. O sistema utiliza um repositório de prédios modelados. Os prédios são selecionados desse repositório e colocados em posições apropriadas dentro dos lotes. O resultado é a geração de cidades úteis para simulação virtual de humanos, mas não é eficiente para a geração de cidades reais, porque o repositório não possui modelos de prédios de cidades reais.

O trabalho proposto por [2] apresenta uma metodologia para a modelagem e renderização de cenas de modelos arquitetônicos baseados em fotografias. Os resultados apresentados mostram que o método consegue gerar arquiteturas realistas.

O trabalho proposto por [3] mostra a construção 3D de um bairro da cidade de Nicósia, onde foi feita uma análise da estrutura dos edifícios da área a ser modelada para descobrir as características e estilo dos edifícios. As construções residenciais da área seguem alguns estilos arquitetônicos bem definidos, assim foi criada uma biblioteca de componentes 3D como portas, janelas, arcos, etc. O modelo 3D é então construído automaticamente usando blocos apropriados da biblioteca de componentes 3D. Além do método de geração automática, são criadas representações 3D precisas de *landmarks*. Esse método se mostra eficaz para geração de cenas realistas, mas o uso desses

componentes 3D aumenta o número de polígonos da cena.

A figura 1 apresenta o *framework* desenvolvido, onde se podem notar características de [1] como o uso de mapas topográficos para a construção do terreno e um editor de cenário; de [2] usa a técnica que a partir de fotografias são geradas texturas, sendo que no *framework* também se gera texturas a partir de pinturas e desenhos; de [3] para a criação dos modelos 3D são gerados *landmarks*, que são prédios importantes da cidade e os outros prédios (residências, lojas, etc.) usam estilos arquitetônicos da época.

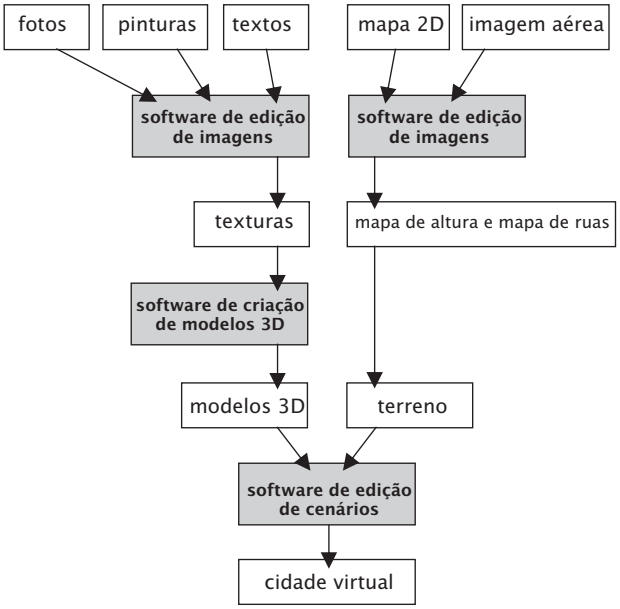


Figura 1. Diagrama representando o *framework* para geração de cidades virtuais.

3. Processo de criação da cidade virtual de belém

Uma fonte importante para a caracterização da cidade de Belém no ano de 1840 foi obtida com informações coletadas de pinturas, livros, mapas, como o mostrado na figura 2, e fotos fornecidos pela biblioteca virtual encontrada em [4]. Imagens de satélites obtidas usando o programa Google Earth[8], comprovam que o mapa da figura 2 está correto e mostra que as áreas próximo ao litoral foram aterradas e como conseqüência praias e uma parte do rio transformaram-se em ruas conforme mostra a figura 3.

Com as fontes obtidas foi possível representar a cidade e algumas características interessantes como: as ruas, texturas e arquitetura das casas, floresta que cercava a cidade e portos dentro do ambiente virtual.

Figura 2. Mapa Topografico de Belém do Seculo XIX.

Figura 3. Cidade de Belém obtida pelo software Google Earth.

3.1. Criação do terreno

A técnica para a construção do terreno consiste em utilizar um heightmap (mapa de altura) para gerar a geometria do terreno. Para a textura do terreno foi utilizada a técnica de splatting descrita em [7]. Esta técnica consiste em utilizar uma imagem, com canal alpha, do tamanho proporcional ao do terreno, para representar as regiões com a textura relacionada.

3.2. Criação de prédios 3D

Para a criação dos prédios 3D utilizamos a técnica de modelagem baseadas em fotos ou desenhos antigos [2]. O principal papel dos desenhos é construir o contorno dos modelos, dando o aspecto geral do prédio, como na figura 4.

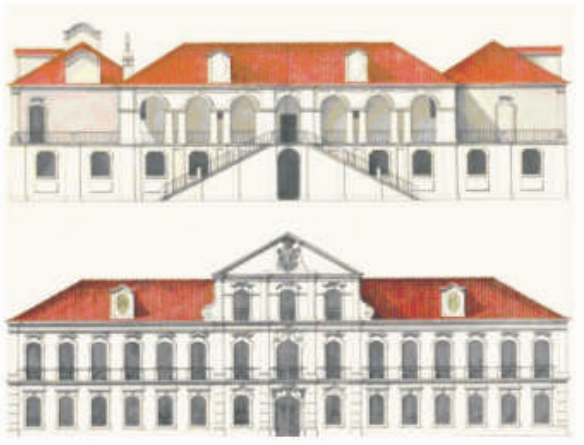


Figura 4. Desenhos do Palácio.

A figura 5 mostra o modelo criado usando a técnica descrita.

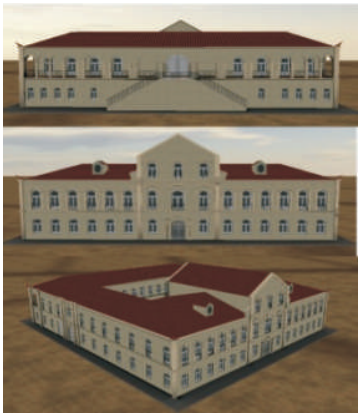


Figura 5. Modelo Criado.

As fotos podem ser usadas tanto na criação dos modelos 3D quanto na criação das texturas. Para serem usadas na modelagem, as fotos são editadas para o perfeito alinhamento das laterais e do chão e também para construir o contorno dos modelos. A vantagem, nesse caso, é que a foto serve tanto para modelagem quanto para textura.

4. Aplicação

A aplicação desenvolvida permite ao usuário controlar um avatar no ambiente virtual permitindo a navegação e a visualização do ambiente. No controle do avatar o usuário poderá movimentá-lo e guiá-lo a diversos locais da cidade virtual de Belém.

4.1. Implementação

Para a implementação foi utilizado a linguagem orientada a objetos C++. Para o motor gráfico foi utilizado o Ogre3D [5] que é um motor gráfico de código aberto que funciona na maioria das plataformas. Foi utilizada também a API de implementação de colisão e aplicação de física no ambiente PhysX[6].

4.2. Caracterização dos Personagens

A representação da população que vivia na cidade de Belém no século XIX atribui ao ambiente uma fidelidade maior quando se deseja ensinar história. Desta forma, este trabalho não se concentra apenas na caracterização da cidade, mas também em alguns habitantes dela.

Dois avatares em especial foram criados dentro do ambiente: cabanos e soldados.

Cabanos eram os homens de uma classe explorada, que viviam em cabanas (por isto eles eram chamados assim) e geralmente possuíam um vestuário marcado pela simplicidade.

Os soldados são caracterizados por vestuário típico da época obtido em [4] e eram utilizados pela classe dominante para manter sua posição política, econômica e social. A figura 6 mostra os modelos 3D do cabano à esquerda e do soldado a direita.

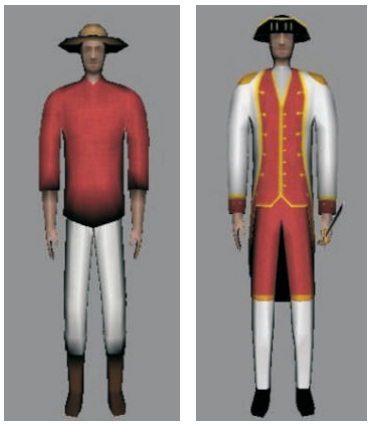


Figura 6. Cabano à esquerda e soldado à direita.

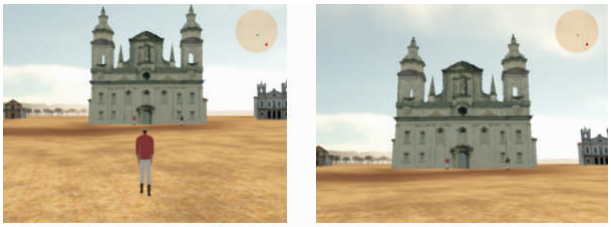
4.3. Ambiente virtual

No ambiente virtual o processo de exploração é feito pelo usuário utilizando um avatar representativo de um personagem do século XIX que morava em Belém.

Ele possui a vestimenta da época e também suas características físicas correspondente a informações retratadas em livros e imagens. Ele é movimentado dentro do ambiente utilizando-se as teclas WASD do teclado.

Há dois tipos de visões: visões de primeira e terceira pessoa ao avatar. A visão em primeira pessoa permite somente a visualização do ambiente. A visão em terceira pessoa permite o controle do avatar, contudo não permite uma visualização total sobre o ambiente.

A figura 7 mostra a câmera em terceira pessoa à esquerda e em primeira pessoa à direita.



(a) Terceira Pessoa (b) Primeira Pessoa

Figura 7. Câmeras no avatar.

A exploração do ambiente com o avatar é livre nos limites da cidade, sendo que através de um mecanismo de colisão o avatar é impedido de ultrapassar estes limites. Além disto, ele também colide com as casas,

árvores, outros avatares e prédios.

O usuário controlando o avatar através da cidade conhece alguns prédios e localidades importantes.

São seis objetivos que o usuário deve seguir. Cada um desses objetivos é listado em uma tela chamada de inventário, que é acessado pela tecla "barra de espaço".

Para auxiliar na localização do próximo objetivo é utilizado um *"mini-map"*, situado no canto superior direito da tela, que mostra o ponto correspondente do objetivo (circulo verde) e o ponto atual do avatar (símbolo vermelho), conforme mostra a figura 8.

Figura 8. Avatar em frente a um landmark (Trem de guerra – continuação da igreja das Mercês) e o *mini-map*.

Quando o avatar é levado para o local do *landmark* correspondente ao objetivo, aparece na tela uma indicação para pressionar o botão "Enter". Feito isto, um texto descritivo sobre aquela localidade é mostrado, conforme mostra a figura 9.

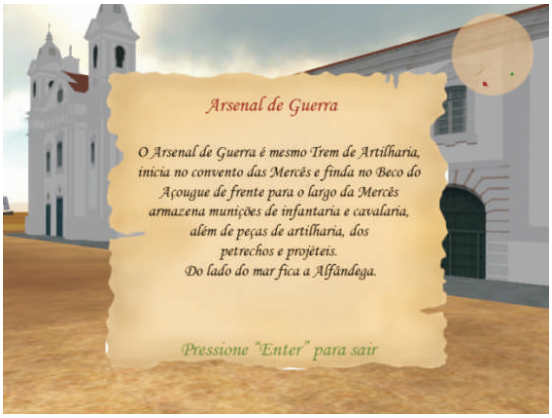


Figura 9. Texto descritivo sobre o *Landmark*.

5. Conclusões

Apresentou-se neste trabalho um sistema de RV desktop para o ensino de história. Esse sistema tem como uma de suas premissas a geração de ambientes virtuais onde acontecem os fenômenos históricos, ou seja, nas cidades. Para isso apresentou-se um framework de geração de cidades, através do qual foi gerada a cidade de Belém do inicio do séc. XIX. O sistema também determina que o usuário interaja com o ambiente através de avatares. As imagens apresentadas neste artigo, assim como as animações disponibilizadas mostram que a cidade gerada é realista. É mostrado também o usuário controlando o avatar, a procura dos objetivos.

Como trabalho futuro, pretende-se evoluir para a construção de um jogo eletrônico educacional, onde o usuário, através de avatares, disputará objetivos com avatares controlados por computador, usando inteligência computacional.

O sistema necessita ser testado por alunos e professores do ensino médio, o que se pretende fazer.

6. Agradecimentos

Agradecemos ao nosso financiador FINEP – pela chamada pública MCT/FINEP/MEC – Jogos Eletrônicos Educacionais 02/2006. Agradecemos também a CAPES pela bolsa de mestrado e a UFPA pela bolsa PIBIC/UFPA pela bolsa de iniciação científica.

7. Referências

[1] L. G. Silveira and S. R. Musse, Real-time Generation of Populated Virtual Cities, *VRST06, Limassol*, Cyprus, pages 155-164, November 2006.

[2] P. Debev, C. Taylor and J. Malij, Modeling and rendering Architeture from Photograps: a hybrid geometry – and image-based approach, In *Proceedings of ACM SIGGRAPH 96*, 9 pages, 1996.

[3] M. Dikaiakou, A. Efthymiou and Y. Chrysanthou, Modeling the Walled City of Nicosia ,In D. Arnold, A. Chalmers, and F. Niccolucci, editors, VAST ,pages 57-65, Brighton, U.K., 2003.

[4] Forúm Landi, Disponível em: http://www.forumlandi.com.br/.

[5] Junker. Gregory, Pro OGRE 3D Programming, Apress, th edition, 2006.

[6] PHYSX, 2006. Disponível em: http://www.ageia.com/ [Acessado em 11 de fevereiro de 2008.

[7] T. Polack, Focus On 3D Terrain Programming (Game Development), Cource Technology, 2003.

[8] Google Earth, Disponível em: http://earth.google.com/intl/pt/.

Ambiente Virtual Na Região do Rio Acará do Século XIX

Manoel Ribeiro, Pebertli Barata, Messias Nascimento, Fabrício Silva
Universidade Federal do Pará,
Instituto de Tecnologia, Laboratório de Realidade Virtual.
mrf@ufpa.br, alhobarata@gmail.com, messiasjan@yahoo.com.br, commander.fabricio@gmail.com

Abstract

This paper presents a virtual environment in the field. The Acará-Açú farm, located in Acará river, the Santa Cruz site , located in Itapicuru stream, that is tributary of the Acará river. The environment has two avatars: a capataz and a slave. This is the first version, a desktop VR environment, of that will be an educational game that intend to help in the study of one the episodes of the revolution of Cabanagem, the battles of the Acará river.

Resumo

Apresenta-se um ambiente virtual no campo. A Fazenda Acará-Açú, localizada no rio Acará, o Sitio Santa Cruz, localizado no igarapé Itapicuru, que é afluente do rio Acará, o rio Acará, e o igarapé Itapicuru com suas margens. O ambiente possui dois avatares: um capataz e um escravo. Essa é a primeira versão, no momento um ambiente de RV desktop, do que será um jogo educativo que pretende auxiliar no estudo de um dos episódios da revolução da Cabanagem, as batalhas do rio Acará.

1. Introdução

As tecnologias de jogos de computador e realidade virtual têm muitos pontos em comum. Em [1], afirma-se que "conhecer a tecnologia de desenvolvimento de jogos é interessante para profissionais e pesquisadores da área de RV e RA, mesmo que esses só pretendam desenvolver aplicações sérias". Apresenta-se um ambiente de realidade virtual desktop, implementado com o uso do Ogre3d [2], que é uma tecnologia usualmente utilizada para desenvolvimento de jogos eletrônicos, mas que também é utilizada para implementar sistemas de realidade virtual desktop. O ambiente virtual apresentado é constituído pela fazenda Acará-Açú, às margens do rio Acará e pelo sitio Santa Cruz, às margens do igarapé Itapicuru, que é afluente do Acará[3]. A equipe do projeto fez uma viagem de barco no trecho entre a Fazenda e o sitio, veja mapa da figura 1, visitou ambas obtendo fotografias e filmagens, com ênfase para a vegetação, para serem usadas como modelos do ambiente. É importante ressaltar que as margens do rio e do igarapé não sofreram grandes modificações desde o início do século XIX, pois é uma região onde o único meio de locomoção continua sendo os rios, não havendo estradas rodoviárias.

A motivação da criação do ambiente virtual na região do rio Acará do século XIX, é que pretende-se usar esse ambiente para o desenvolvimento de um jogo eletrônico com estilo de estratégia, que servirá de auxílio no estudo do início do conflito armado da revolução da Cabanagem [3], que ocorreu em outubro de 1834. A fazenda Acará-Açú pertencia a Antônio Clemente Malcher, que foi o primeiro presidente cabano. A principal atividade econômica da fazenda era a venda de açúcar, e para a sua produção a fazenda possuía canaviais, engenho de açúcar, senzala e escravos. O sítio Santa Cruz pertencia à família Vinagre, cujo filho mais velho, Francisco Vinagre foi o segundo presidente cabano. A principal atividade econômica do sítio era a venda de farinha de mandioca, e a sua produção era feita por membros da família Vinagre, e o sítio possuía plantação de mandioca e casa para fazer farinha.

Figura 1. Mapa da região modelada.

A versão atual ainda não tem características de um jogo educativo, como definido em [4], no entanto possui características de um sistema de RV desktop [5], onde o usuário controla avatares (escravo e capataz) com controle de colisão e *picking*, e navega pelo ambiente virtual. Sob o comando do usuário, o capataz ordena ao escravo realizar tarefas como colher cana para levar ao engenho e carregar açúcar do engenho para a margem do rio.

2.O ambiente virtual

A seguir comentam-se alguns detalhes da construção dos terrenos, vegetação e dependências da fazenda e sítio. O terreno da fazenda Acará-açu foi modelado manualmente, usando poucas faces nas áreas sem depressão. O modificador *MultiRes*, do 3DSmax[6], foi utilizado para simplificar essas áreas. A senzala da fazenda, por sua vez, foi construída usando como referência relatos de historiadores, descrevendo senzalas como tendo grossas paredes e janelas e portas largas [7]. Sua modelagem concentrou-se na edição de texturas, sendo criadas para representar paredes com reboco de barro e palha para o teto, grades de ferro para janelas e porta. Para criação dessas grades foi utilizada uma técnica muito comum em games, que consiste em usar imagens em formato PNG ou TGA, com canal *alpha*, ou seja, que permitem áreas transparentes. Dessa maneira, evita-se modelar cada barra da grade, o que aumenta significativamente a quantidade de faces. Os espaços vazios entre as barras da grade são representados pelas áreas transparentes da imagem. A figura 2 mostra a janela da senzala vista de dentro, cuja textura representa grades de ferro.

Figura 2. Grade de Ferro, da senzala, representada por textura.

A técnica de texturas com canal *alpha* também foi usada na construção do moinho do engenho de açúcar da fazenda. Árvores e arbustos também foram modelados com o uso dessa técnica.

A modelagem dos personagens também recebeu um estudo especial, especialmente para que as roupas dos avatares tivessem coerência com a época. A figura 3 mostra um dos avatares utilizados no jogo.

Figura 3. O avatar escravo.

A figura 4 mostra o escravo e o capataz em frente da plantação de cana-de-açúcar, e a figura 5, mostra o escravo e o capataz em frente ao engenho.

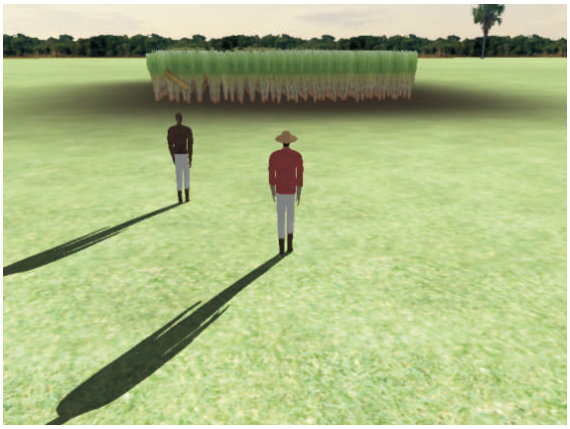


Figura 4. Capataz e escravo no canavial.

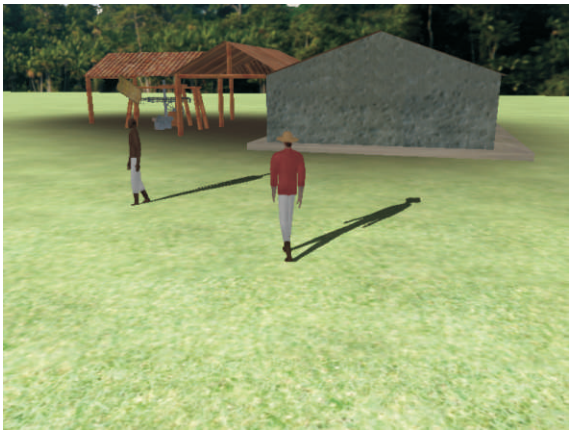


Figura 5. Capataz e escravo no engenho de açúcar.

O sítio foi modelado usando técnicas semelhantes. As figura 6 e 7 mostram um lavrador no sítio.

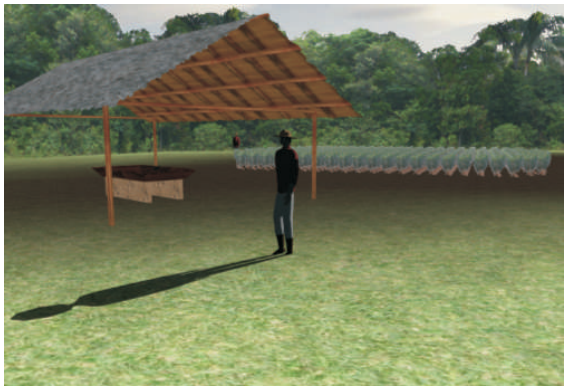


Figura 6. Lavrador próximo a casa de farinha e plantação de mandioca.

Figura 7. Lavrador em frente ao igarapé Itapicuru.

Para a modelagem das águas do rio foi utilizada a técnica do *bump mapping*[8], um efeito que cria a ilusão de relevo em uma determinada textura, permitindo assim uma superfície com ondulações e reflexões necessárias para uma representação mais real da água, sem grandes custos computacionais. A textura do *bump mapping* se move dando a impressão do movimento da água. Esta técnica permite a redução dos polígonos de uma geometria e se concentra no material aplicado.

3. Interações e navegação

A aplicação foi implementado em linguagem C++ com o auxílio do motor gráfico *3D Ogre3D* [2]. Há ainda o suporte da *API Physx* [9], usada para implementação de física e colisão. O uso destas ferramentas, visa a distribuição multiplataforma da aplicação.

A interação do usuário com os avatares é uma característica bem peculiar, usando elementos de jogos populares de simulação e estratégia para formar o estilo interativo.

Um primeiro aspecto está na utilização da câmera, que é centralizada no avatar, porém admite uma liberdade de movimentação mais atrativa que a maioria dos jogos deste estilo, com liberdade de alteração da distancia de visão e movimentação livre em uma semi-esfera imaginária em torno do jogador principal.

O usuário controla a animação de avatares com o teclado.

O usuário controla um avatar chamado capataz, que quando se aproxima dos avatares escravos , ordena a estes para realizar tarefas típicas da época. Quando o capataz fica a menos de 2 metros do escravo, este fica esperando ordens que são feitas usando o teclado. Foram implementadas duas atividades. Tecla *E* para buscar cana e Tecla *F* para fazer o açúcar. A figura 8 ilustra este recurso do jogo. As animações submetidas mostram esse processo.

O usuário pode se deslocar através de animações da fazenda para o sítio, usando a tecla *f9* e do sítio para a fazenda usando a tecla *f10*. Ao chegar ao seu destino assume o controle dos avatares para realizar as tarefas. Este recurso do jogo reproduz a real navegação no rio, passando por trechos existentes na região. As figuras 9 e 10 mostram imagens dessas viagens. As animações submetidas mostram esses deslocamentos.

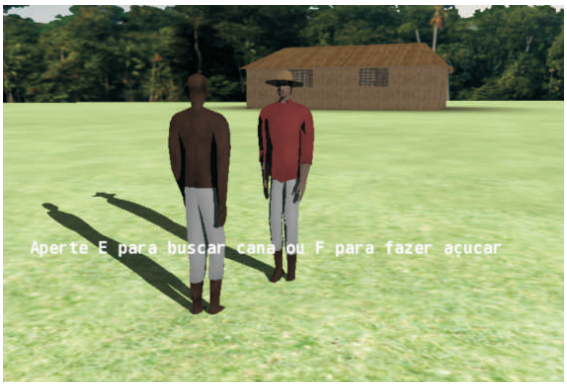


Figura 8. Capataz dá ordens ao escravo.

A figura 9 mostra uma tela da animação de navegação no rio.

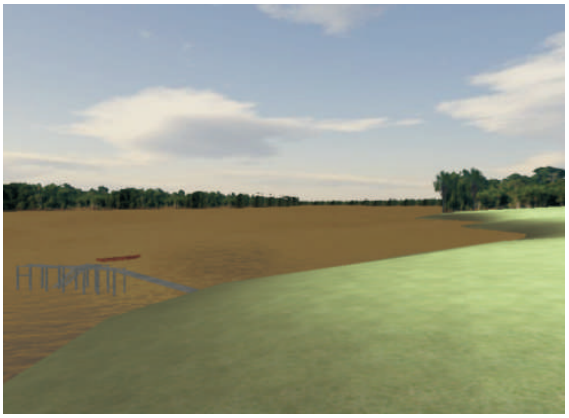


Figura 9. Usuário saindo da Fazenda Acará-Açu.

Figura 10. Encontro das águas do rio com o igarapé.

4. Conclusão

Foi apresentada uma aplicação de RV desktop, com algumas características de Games, ambientada no campo, mas especificamente da região do rio Acará, localizado no estado do Pará, no início do século XIX. As imagens das figuras e as animações submetidas mostram um ambiente imersivo e com alta interação usuário-computador.

Como atividade futura, pretende-se utilizar a base do sistema apresentado para a construção de um jogo eletrônico educativo no estilo de estratégia.

5. Agradecimentos

Agradecemos ao nosso financiador FINEP – pela chamada pública MCT/FINEP/MEC – Jogos Eletrônicos Educacionais 02/2006. Agradecemos também a UFPA pela bolsa PIBIC/UFPA de iniciação científica.

6. Referências

[1] Kirner, Cláudio; Siscouto, Robson, et al., Realidade Virtual e Aumentada, Conceitos, Projetos e Aplicações. *Livro do Pré-Simpósio, SVR2007*, pag. 193, 2007.

[2] Junker, Gregory, Pro OGRE 3D Programming, Apress; 1th edition, 2006.

[3 Raiol, Domingos Antônio, *MOTINS POLITICOS, ou História dos Principais Acontecimentos Políticos da Província do Pará desde o Ano de 1821 até 1835*; Editora da UFPA, 1970.

[4} Raquel M. Müller1, et all, Laguna – Aprendendo sobre a guerra do Paraguai com um jogo educativo, anais do XXVII congresso brasileiro da SBC, WIE, *XIII worshop sobre informática na escola*, pags. 256 a 263, 2007.

[5] Oh, Ji-Young e Stverzlinger, Wolfgang, “A system for desktop conceptual 3D design”. *Virtual Reality. Springer-Verlag London Limited*, pag 198-211, 2004.

[6] Murdock, Kelly L., 3DS Max 8 Bible, Wiley, 2006.

[7] Freire, Gilberto, *Casa Grande e Senzala*, Circulo do Livro, 1989.

[8] Blinn, James F., Simulation of Wrinkled Surfaces, Computer Graphics, Vol. 12 (3), pp. 286-292 *SIGGRAPH-ACM*, august 1978.

[9] PHYSX. Disponível em: http://www.ageia.com/ [Acessado em 11 de fevereiro de 2008.

minicourses

X3D

Don Brutzman (Naval Postgraduate School, USA)

Extensible 3D (X3D) graphics is a collection of open-standards that define a system that integrates network-enabled 3D graphics and multimedia. X3D applications are real-time, interactive, animated systems that can run stand-alone or in networked virtual environments. This tutorial will focus on a commonly used subset of the complete functionality that is encoded in XML. X3D has three encodings XML (.x3d), Classic VRML (.x3dv), and Compressed Binary (.x3db). During the tutorial, the participants will learn hands-on how to build an X3D world, while getting a detailed understanding of the capabilities of X3D.

Specific topics include animation using interpolators and sequencers, scripting, prototypes for extensibility, and a software-visualization case study. We will use the new cross-platform X3D-Edit authoring tool. Participants will also be given the latest X3D Software Development Kit (SDK) which contains a wide variety of free + commercial plug-ins, authoring tools, and content.

Reference: X3D: Extensible 3D Graphics for Web Authors by Don Brutzman and Leonard Daly, Morgan Kaufmann Publishers, April 2007, 468 pages.

Projeto e Implementação de Ambientes Virtuais Distribuídos

Ildeberto A. Rodello, Antonio C. Sementille, Fatima L.S.N. Marques (Centro Universitário Eurípides de Marília - UNIVEM) and José Remo F. Brega (Unesp/Bauru)

Este minicurso tem como objetivo principal apresentar e discutir os principais conceitos e requisitos relacionados ao projeto de Ambientes Virtuais Distribuídos abrangendo uma discussão sobre suas principais características, seus principais componentes, a taxonomia e os modelos de comunicação. Tem ainda como objetivo apresentar os aspectos para o desenvolvimento tais como protocolos, modelos de particionamento e plataformas de implementação, apresentando exemplos e discutindo resultados.

High Perfomance Computing: CUDA as a Supporting Technology for Next Generation Augmented Reality Applications

Thiago Farias, João M. Teixeira, Pedro Leite, Gabriel Almeida, Veronica Teichrieb, Judith Kelner (Virtual Reality and Multimedia Research Group, Computer Science Center, Federal University of Pernambuco)

The main purpose of this minicourse is presenting the potential of GPGPU (General Purpose Graphics Processing Unit) technology for real time markerless augmented reality related processing. CUDA (Compute Unified Device Architecture) is a GPGPU technology that allows programmers to use the C programming language to code algorithms for execution on the GPU. CUDA has been developed by Nvidia and to use this architecture requires an Nvidia GPU and special stream processing drivers. GPU computing with CUDA is a new approach to computing where hundreds of on-chip processors simultaneously communicate and cooperate to solve complex computing problems. Applications that require mathematically intensive computing on large amounts of data are ideal targets for GPU computing. The goal of this minicourse is to present this new technology exploring a theoretical approach, due to hardware availability requirements. The CUDA architecture will be depicted, together with an optimized programming model for obtaining better results using the parallel approach. Some case studies, mainly related to tracking algorithms, will also be shown in order to demonstrate the performance improvement in comparison to sequential approaches.

Desenvolvimento Rápido de Aplicações de RV Utilizando SW Livre

Liliane S. Machado, Ronei M. Moraes, Daniel F.L. Souza and Leandro C. Souza
(LabTEVE - Universidade Federal da Paraíba)

O desenvolvimento de sistemas de realidade virtual pode englobar diferentes funcionalidades. Neste sentido, o aprendizado de técnicas específicas, bem como o aprendizado de bibliotecas de dispositivos e a integração de todas as tarefas demandam tempo e podem tornar complexo o processo de desenvolvimento. O objetivo do curso é apresentar e explorar um conjunto de bibliotecas livres que permitem o rápido desenvolvimento de aplicações baseadas em RV. Para tanto, será utilizado o CyberMed, um conjunto de bibliotecas registradas e disponíveis em licença GPL. Pretende-se apresentar o conceito de desenvolvimento das bibliotecas, suas funcionalidades disponíveis e modos de expandi-las. Os conceitos serão mesclados com exemplos práticos que incluirão o desenvolvimento passo-a-passo de uma aplicação completa envolvendo visualização estéreo, interação através de dispositivos convencionais e hápticos, uso de modelos deformáveis e integração de métodos de avaliação online do usuário.

OpenSceneGraph: Conceitos Básicos e Aplicações em Realidade Virtual e Aumentada com ARToolKit

Luiz Gonzaga da Silveira Jr. (UNISINOS) and Alberto Raposo (Tecgraf/PUC-Rio)

Largamente utilizadas no desenvolvimento produtivo de aplicações gráficas e em realidade virtual, algumas toolkits têm preenchido um espaço deixado pela OpenGL no que concerne ao nível de abstração da programação, manipulação de modelos externos, representação hierárquica de cena, efeitos especias, LOD, HDR, estereoscopia, dentre outros. Open Inventor e Performer iniciaram essa corrida, seguidas pelo Java 3D. Essas tecnologias foram expostas em conferências como SVR, SIBGRAPI, ACM Siggraph e Eurographics. Mais recentemente apareceram OpenSG, Coin3D e OpenSceneGraph, com arquiteturas mais modernas para o desenvolvimento de aplicações de alto desempenho. Como ocorreu com as outras tecnologias anos atrás, estas toolkits estão em evidência atualmente nas conferências da área e correlatas. Acreditamos que OpenSceneGraph, por ter uma grande aceitação nas comunidades internacionais de computação e realidade virtual, e já ter alguma aceitação na comunidade nacional, e Artoolkit igualmente na comunidade de realidade aumentada, merecem uma divulgação na conferência desse ano. Este minicurso tem como objetivo básico introduzir a toolkit OpenSceneGraph (www.openscenegraph.org), destacando os benefícios do seu emprego no desenvolvimento de aplicações realidade virtual e aumentada, através da sinergia com a ARToolKit. Vamos mostrar como integrar OpenSceneGraph e ARToolKit e aplicá-las em conjunto no desenvolvimento de aplicações visual e funcionalmente ricas em realidade virtual, aumentada e misturada.

RV para a Área Médica: Requisitos, Dispositivos e Implementação

Fatima L.S. Nunes (UNIVEM – Centro Universitário Eurípides de Marília) and Rosa M. E. M. Costa (Universidade Estadual do Rio de Janeiro)

A área da saúde é uma das áreas com maior potencial para ser beneficiada pelas aplicações de Realidade Virtual. No entanto, construir aplicações para esta área exige conhecimento específico proveniente de várias outras áreas de conhecimento. Este minicurso tem o objetivo de transmitir os aspectos teóricos e práticos para a construção de aplicações médicas usando Realidade Virtual, sendo abordados aspectos de requisitos, desempenho, dispositivos físicos e implementação. Um objetivo secundário é contribuir com o desenvolvimento desta área no país, por meio das seguintes ações: suscitar o interesse de estudantes e pesquisadores para a área; abordar aspectos de tecnologias e de implementação a partir de experiências estrangeiras e nacionais e apresentar áreas carentes que podem se tornar oportunidades de negócios.

Realidade Virtual e Realidade Aumentada Aplicadas à Visualização da Informação

Alexandre Cardoso, Ezequiel R. Zorzal (Universidade Federal de Uberlândia) and Claudio Kirner (UNIMEP / UNASP)

O Minicurso pretende apresentar os fundamentos de visualização da informação e a utilização de Realidade Virtual e Aumentada no processo de visualização, destacando aplicações potenciais, limitações, algumas diretivas, fundamentos e exemplos. Casos de uso práticos serão apresentados e as dificuldades de desenvolvimento serão consideradas e relatadas.

3D User Interaction

Gerold Wesche and Maxim Foursa (Fraunhofer Institute for Intelligent Analysis and Information Systems, Germany)

The objective of this minicourse is to give an overview of 3D User Interaction Techniques on the basis of state-of-the-art technologies supported by multiple application examples. Areas influencing 3D User Interaction, such as 3D display systems, 3D input devices and others will be also considered. It is expected that attendees of the course will have the opportunity to get a comprehensive overview of existing approaches and future trends and will be able to immediately apply some ideas in their everyday work.

pre-symposium

Fundamentos de Realidade Virtual e Realidade Aumentada

Robson A. Siscoutto (UNIC) e Claudio Kirner (UNIMEP)

Realidade Virtual e Realidade Aumentada são duas áreas relacionadas com as novas gerações de interface do usuário, facilitando e potencializando as interações do usuário com as aplicações computacionais. Esta palestra apresenta os conceitos de realidade virtual e aumentada, mostrando os aspectos de tecnologia, projeto e aplicação.

Dispositivos de Entrada e Saída

Liliane S. Machado (UFPB) e Alexandre Cardoso (UFU)

A utilização de dispositivos específicos para entrada e saída de informações em sistemas de Realidade Virtual (RV) visa aumentar os níveis de imersão do usuário e prover modos mais intuitivos de interação. Desta forma, dispositivos de entrada procuram captar movimentos e ações do usuário para alimentar o sistema de RV que retornará, através dos dispositivos de saída, o resultado do processamento desta interação na forma de estímulos a pelo menos um dos cinco sentidos humanos. Esta palestra apresentará, separados nas duas categorias, alguns dos dispositivos mais utilizados em sistemas de RV, expondo a finalidade e as funcionalidades de cada um.

Técnicas de Interação

Judith Kelner (UFPE) e Veronica Teichrieb (UFPE)

Interação é um método que permite a um usuário realizar tarefas por meio de uma interface. Uma técnica de interação inclui tanto componentes de hardware (dispositivos de entrada/saída) quanto de software (mapeamento da entrada em ação e do retorno do sistema em saída). Interação 3D é crucial em sistemas de RV, assim como de RA. Algumas técnicas de interação 3D foram, originalmente, desenvolvidas para ambientes virtuais. No entanto, a maioria pode ser adequadamente utilizada em ambientes de RA, com poucas ou sem adaptações. Técnicas específicas para RA também foram concebidas, a fim de aproveitar características inerentes deste tipo de interface, tais como a possibilidade de interação natural do usuário tanto com objetos virtuais quanto reais durante a utilização da aplicação, além da mobilidade.

VRML e X3D

Alexandre Cardoso (UFU)

As tecnologias para desenvolvimento de soluções de Realidade Virtual podem ser dispendiosas, a ponto de inviabilizar propostas de solução e de melhorias. VRML e X3D são tecnologias utilizadas para concepção de ambientes virtuais interativos, compatíveis com a Web e gratuitas. A apresentação elucidará os aspectos introdutórios sobre como utilizar VRML e X3D, abrangendo as principais características, limitações, vantagens e desvantagens. Não são necessários conhecimentos prévios de programação e/ou domínio de técnicas de Realidade Virtual.

JAVA 3D

José Remo F. Brega (UNESP), Ildeberto A. Rodello (UNIVEM) e Antonio C. Sementille (UNIVEM)

Usando o paradigma de orientação a objetos e a estrutura de grafo de cena, a API Java 3D oferece abstrações que permitem a criação de ambientes virtuais com as potencialidades oferecidas pela linguagem de programação Java, tais como portabilidade e heterogeneidade. A API oferece recursos para a criação desde formas tridimensionais simples até ambientes mais complexos, com suporte para interação multimodal e distribuição. Possui classes específicas tanto para o tratamento de eventos com dispositivos convencionais quanto não convencionais, além de oferecer a possibilidade do desenvolvimento para um dispositivo específico. Nesse contexto, pretende-se apresentar a API Java 3D, discutindo desde a estrutura básica de um programa, até a implementação de ambientes mais complexos e elaborados. Serão apresentados e discutidos as principais classes e métodos para a criação de formas e suporte para diversos dispositivos de entrada e saída, ilustrando a forma de implementação por meio de diversos exemplos.

ARToolKit

Claudio Kirner (UNIMEP) e Rafael Santin (UNIMEP)

ARToolKit é um software bastante popular indicado para o desenvolvimento de aplicações de Realidade Aumentada (RA). Ele constitui-se em uma biblioteca em C/C++, gratuita e de código aberto, cujas funções são usadas em programas aplicativos desenvolvidos pelo usuário. Além disso, é possível reconfigurar as aplicações, através da edição de alguns arquivos e pastas, gerando novas aplicações. No sentido de facilitar o desenvolvimento de aplicações de RA, foi gerado um sistema de autoria com várias funcionalidades pré-programadas, funcionando em rede e com capacidade de configuração em dois níveis. Essas características permitem que usuários não programadores possam elaborar aplicações de RA individuais e colaborativas. Este palestra apresenta uma visão do ARToolKit e descreve o Sistema de Autoria Colaborativa de Realidade Aumentada para o Usuário Final.

Jogos e Entretenimento com Realidade Virtual e Realidade Aumentada

Romero Tori (USP/SENAC - SP)

Esta palestra abordará os principais conceitos relacionados a jogos e interatividade e apresentará uma visão geral da evolução tecnológica dos jogos de computador. Será traçado um paralelo dessa tecnologia com a de realidade virtual, com destaque para os motores de jogos (*game engines*) e sua utilização no desenvolvimento de jogos e ambientes virtuais. Por fim serão mostradas as possibilidades de inovação no design de jogos computacionais e de sistemas digitais de entretenimento, trazidas pelas interfaces tangíveis e pela Realidade Aumentada.

Aplicações Médicas com Realidade Virtual e Realidade Aumentada

Fatima L. S. Nunes (UNIVEM) e Rosa M. E. M. Costa (UERJ)

A área de saúde constitui um dos campos mais beneficiados pelas aplicações de Realidade Virtual (RV) e Realidade Aumentada (RA). Ferramentas para simulação de procedimentos, treinamentos virtuais e reabilitação estão entre as carências desta área. Independente da finalidade da aplicação, os requisitos para construí-la muitas vezes podem ir além daqueles exigidos para aplicações destinadas a outras áreas. Características técnicas (detecção de colisão com precisão, tempo de resposta, deformação realística, estereoscopia e interação), aspectos humanos e considerações éticas podem levar ao sucesso ou fracasso de uma aplicação.

Esta palestra apresenta uma visão dos requisitos importantes para construir aplicações de RV e RA para a área de saúde, motivando uma reflexão e oferecendo subsídios para o planejamento de ferramentas para esta área.

Realidade Virtual na Educação

Alexandre Cardoso (UFU) e Edgard Lamounier (UFU)

A Educação pode ser vista como um processo de descoberta, exploração e de observação. Tais elementos são de fundamental importância para a construção do conhecimento. Neste contexto, a possibilidade de simular situações e experimentos que de maneira real não seriam possíveis ou inviáveis, propicia grande avanço no processo educacional. Neste contexto, esta palestra visa apresentar e discutir Realidade Virtual como uma significativa ferramenta de suporte ao ensino-aprendizagem. É avaliado como a mesma possibilita a realização de experiências com o conhecimento, de forma mais intuitiva e interativa, suportando a exploração de ambientes, processos ou objetos, de uma forma totalmente inovadora, quando comparada com métodos tradicionais (quadro-negro, livros, filmes etc.).

credits

1; 10 Illumination Techniques for Photorealistic Rendering in Augmented Reality
S. A. Pessoa, E. L. Apolinário, G. S. Moura, J. P. S. M. Lima, M. A. S. Bueno, V. Teichrieb, J. Kelner

2 ARMsg: Mensageiro Digital usando Realidade Aumentada (short paper)
Jan Habib, Alberto Raposo

3 The Image-Based Data Glove
Vitor F. Pamplona, Leandro A. F. Fernandes, João L. Prauchner, Luciana P. Nedel, Manuel M. Oliveira

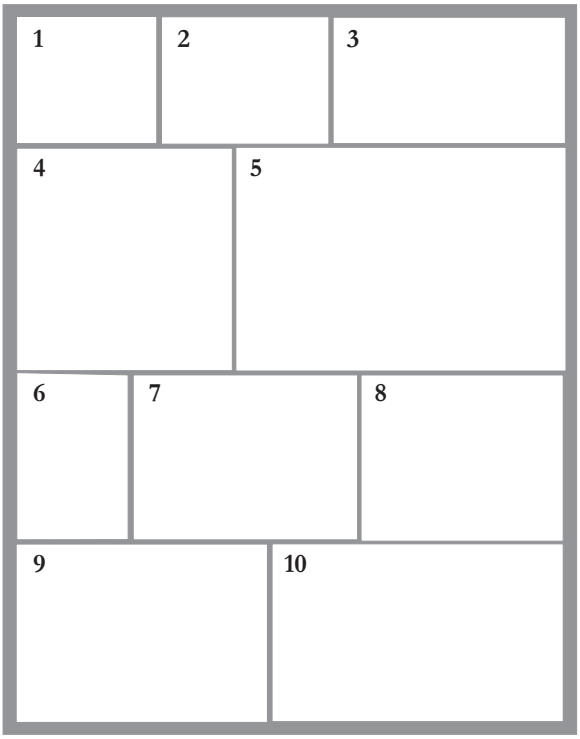
4 Virtual Modeling and Numerical Simulation of Aneurysms and Stenoses
Rodrigo L. S. Silva, Eduardo Camargo, Pablo Javier Blanco, Márcio Pivello, Raúl A. Feijóo

5 A Two-User Virtual Environment for Rapid Assembly of Product Models within an Integrated Process Chain
Maxim Foursa, Gerold Wesche, David d'Angelo, Manfred Bogen Rainer Herpers

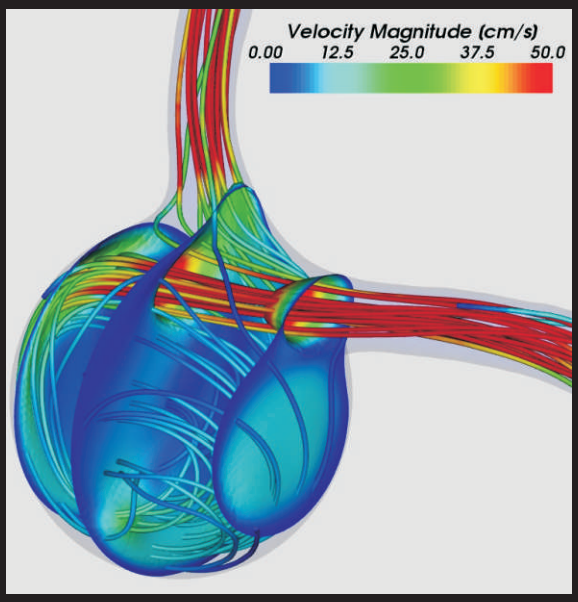
6 O uso da Realidade Virtual não-imersiva para o auxílio ao tratamento da aviofobia pelos profissionais da psicologia (short paper)
D. Medeiros, N. Fortes, E. Lamounier, W. A.. Silva, S. Andrade, A. Cardoso, M. W. S. Ribeiro, E. R. Zorzal

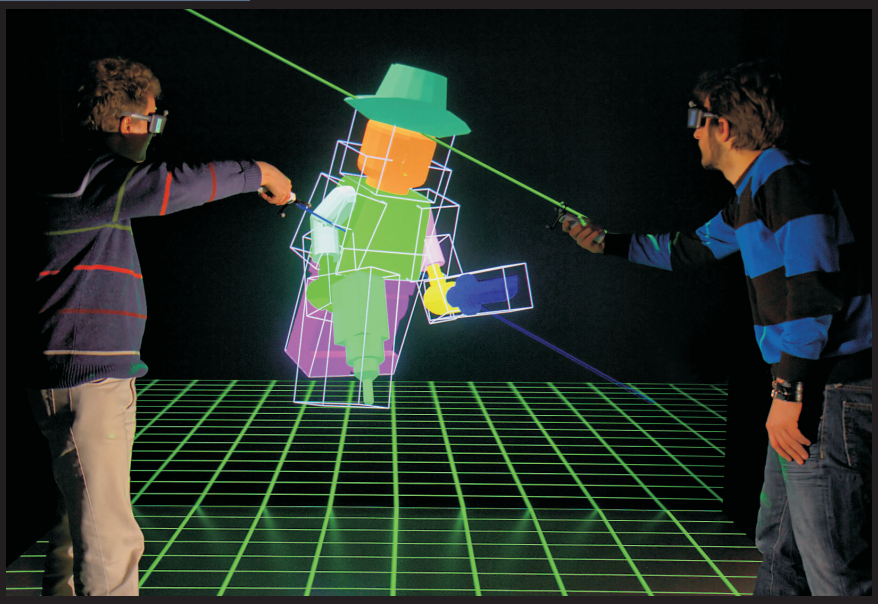
7; 8 EnCIMA: A Graphics Engine for the Development of Multimedia and Virtual Reality Applications
Silvano M. Malfatti, Selan R. dos Santos, Luciane M. Fraga, Claudia M. Justel, Jauvane C. de Oliveira

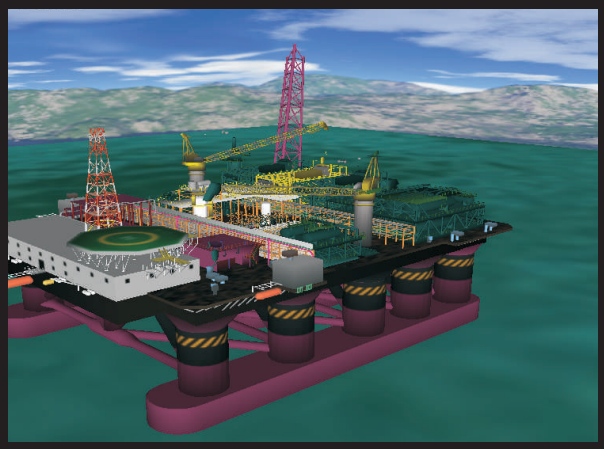
9 Macacos-Prego (Cebus apella) Resolvem Problemas de Discriminação Simples em Ambiente Virtual
Carlos de Sousa Brito Neto, Manoel Ribeiro Filho, Olavo de Faria Galvão

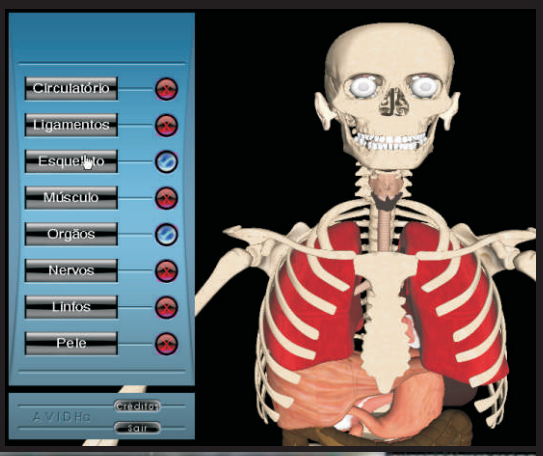


Cover Design: Aline Silveira Cavalcanti
Cover Image: Aline Silveira Cavalcanti
Graphic Design: Aline Silveira Cavalcanti









ISBN 857669167-1