Pontifícia Universidade Católica
do Rio de Janeiro

Felipe Ferreira Quintella

DWeb3D: Um toolkit para facilitar a criação e manipulação de cenas 3D usando X3D

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática da PUC–Rio

Orientador: Prof. Alberto Barbosa Raposo

Rio de Janeiro
Setembro de 2009

Pontifícia Universidade Católica
do Rio de Janeiro

Felipe Ferreira Quintella

DWeb3D: Um toolkit para facilitar a criação e manipulação de cenas 3D usando X3D

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC–Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Ph.D. Alberto Barbosa Raposo
Orientador
Departamento de Informática — PUC–Rio

Ph.D. Bruno Feijó
Departamento de Informática — PUC-Rio

Ph.D. Simone Diniz Junqueira Barbosa
Departamento de Informática — PUC-Rio

Ph.D. Luciano Pereira Soares
Departamento de Informática — PUC-Rio

Ph.D. José Eugênio Leal
Coordenador Setorial do Centro Técnico Científico — PUC–Rio

Rio de Janeiro, 08 de Setembro de 2009

Felipe Ferreira Quintella

Graduou-se em Informática na PUC-Rio em 2006. Trabalhou na CrossWay de 2000 a 2007 com a gerência de um MMORPG, como responsável pela gerência do projeto e da integração de sistemas. De 2007 em diante trabalha na Fundação Getúlio Vargas, como analista do novo sistema acadêmico para os programas de extensão da instituição.

Ficha Catalográfica

Quintella, Felipe

DWeb3D: Um toolkit para facilitar a criação e manipulação de cenas 3D usando X3D / Felipe Ferreira Quintella; orientador: Alberto Barbosa Raposo. — Rio de Janeiro : PUC–Rio, Departamento de Informática, 2009.

v., 80 f: il. ; 29,7 cm

1. Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui referências bibliográficas.

1. Informática – Tese. 2. Grafo de Cena. 3. X3D. 4. Toolkit Gráfico. 5. Realidade Virtual. I. Raposo, Alberto Barbosa. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Agradeço a todos aqueles que amo, pela compreensão, apoio e pelo tempo que tiveram que doar para que eu pudesse desenvolver este trabalho. Sem o apoio destas pessoas este não seria possível.

Agradeço também ao meu orientador, pelas diversas leituras e pelos comentários inteligentes na confecção desta tese.

Resumo

Quintella, Felipe; Raposo, Alberto Barbosa. **DWeb3D: Um toolkit para facilitar a criação e manipulação de cenas 3D usando X3D**. Rio de Janeiro, 2009. 80p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Este trabalho se propõe a estudar os desafios de aplicações 3D na web, analisando o cenário atual deste tipo de aplicação, o porquê da baixa adoção dos métodos existentes, o que funciona e o que não funciona bem. Então é sugerida uma nova abordagem para a construção de mundos 3D e aplicações interativas.

O foco do estudo foi no padrão X3D, por ser uma solução aberta, suportada por um consórcio internacional, madura e em constante crescimento, porém com pequena adoção. Suas qualidades e problemas são discutidos e correlacionados com soluções existentes. Neste processo detecta-se algumas necessidades das aplicações atuais e a complexidade do X3D ao lidar com essas questões. Como uma tentativa de demonstrar que algumas das complexidades do X3D podem ser amenizadas foi desenvolvido o DWeb3D. O DWeb3D é um *toolkit* para facilitar o desenvolvimento de aplicações X3D dinâmicas. Ele foi criado para ser uma forma de demostrar que é possível agilizar o processo de desenvolvimento, dando acesso mais amplo aos desenvolvedores nesta área. O toolkit oferece ferramentas para lidar com a publicação, o sincronismo, a interatividade e o controle de múltiplos usuários, além de possibilitar a persistência do grafo de cena.

Palavras–chave

Grafo de Cena. X3D. Toolkit Gráfico. Realidade Virtual.

Abstract

Quintella, Felipe; Raposo, Alberto Barbosa. **DWeb3D: A toolkit to help the creation and use of 3D scenes using X3D**. Rio de Janeiro, 2009. 80p. MSc Thesis — Department of Computer Science, Pontifícia Universidade Católica do Rio de Janeiro.

This work studies the challenges of 3D applications on the web. It analizes the current scenario of 3D web applications, the reasons of the low adoption of existent solutions, what works well, and what doesn't work. It is then suggested a new approach for the construction of 3D worlds and interactive applications.

The study is focused on the X3D standard because it is open, supported by an international consortium, mature and in constant development, but with a low adoption rate. The X3D qualities and problems are discussed and correlated with other solutions. In this process it was detected some necessities in current applications and the complexity of X3D to deal with these issues. As an attempt to demonstrate that the complexity of X3D in some aspects may be reduced, the DWeb3D toolkit was built.

DWeb3d is a toolkit to help the development of dynamic X3D applications. It was created as a way to demonstrate that it is possible to facilitate the development process, increasing the access to developers in this area. The toolkit provides tools to deal with publishing, synchronism, interactivity, multiple users management and disk persistency.

Keywords

Scene Graph. X3D. Graphic Toolkit. Virtual Reality.

Sumário

Lista de figuras

Lista de tabelas

1
Introdução

O X3D[1] e as tecnologias de que ele deriva se apresentam como uma ótima alternativa para se tornarem o padrão quando o assunto é 3D na web. Porém hoje, vários anos após a sua criação, e mesmo com todas as qualidades que ele possui, sua adoção ainda é muito tímida e não se vê perspectiva de uma mudança a curto prazo desta realidade. Neste trabalho iremos analisar as possíveis causas deste fato e propor métodos que facilitem lidar com esses fatores limitantes do X3D.

1.1
Motivação

Há algum tempo existem diversas iniciativas com o intuito de levar o mundo dos objetos 3D para a web, assim facilitando o seu acesso, popularizando a tecnologia e proporcionando novas formas de interação.

Uma das primeiras tentativas de alcançar este objetivo veio na forma da VRML (`http://www.web3d.org/x3d/vrml/`) (Rik97, Web97). A VRML (Virtual Reality Modeling Language) é um padrão de formato de arquivo para realidade virtual, utilizado tanto para a internet como para ambientes desktop. Por meio desta linguagem, escrita em modo texto, é possível criar objetos (malhas poligonais) tridimensionais se podendo definir cor, transparência, brilho, textura (associando-a a um bitmap). Os objetos podem ser formas básicas, como esferas, cubos, ovóides, hexaedros, cones, cilindros, ou formas criadas pelo próprio programador, como as extrusões. Além dos objetos, também é possível acrescentar interatividade a estes por meio de sensores, podendo assim deslocá-los de posição, acrescentar luz, produzir um som quando o objeto é clicado ou a câmera simplesmente se aproxima dele, e abrir um arquivo ou página da web, ou ainda outra página em VRML, quando o objeto é acionado. Não é necessário um software específico para a criação de arquivos VRML (embora existam), uma vez que os objetos podem ser todos criados em modo texto.

[1] http://www.web3d.org/x3d

A VRML quando foi apresentada gerou uma boa recepção por parte da comunidade web, porém sua adoção foi limitada, e hoje muito do que existia em VRML sumiu ou está desatualizado.

Mais recentemente, no início dos anos 2000, para substituir a VRML e tentar cobrir os seus pontos fracos, surgiu o X3D, que além de poder fazer tudo que a VRML fazia é também extensível. O esforço é coordenado pelo *Web3D Consortium*, que é formado por diversos membros, incluindo corporações. Porém, ainda mais que a VRML, o X3D (`http://www.web3d.org/x3d/`) (Web08, Doi06) tem sido ignorado pela indústria e vem tendo o interesse diminuido pelo comunidade.

Paralelamente a essas tecnologias, existem tentativas de prover conteúdo 3D utilizando Flash e alguns poucos motores gráficos proprietários que possuem opções de exportação para web (através de mecanismos próprios que necessitam de plugins específicos).

De todas as tecnologias vistas, a mais uitlizada atualmente é o Flash (`http://www.adobe.com`) (All02, Hei02) e o Shockwave 3D (`http://www.adobe.com`) (San04, Dun03), porém este apresenta diversas limitações provenientes de sua arquitetura básica desenvolvida para lidar com animações e vídeos, e não com conteúdo 3D interativo. Apesar de seus problemas como o custo elevado, e de não terem sido desenvolvidas especificamente para o 3D, estas ferramentas têm uma aceitação muito superior ao X3D.

Uma fonte de pesquisa sobre a atual popularidade dos formatos pode ser obtida em: *http://www.karmanaut.com/virtuality/zeitgeist/*

O próprio autor no entanto demonstra que apesar dos números parecerem interessantes, eles somente demonstram o número de páginas que fazem referência ao nome da tecnologia, e não a real utilização da mesma.

Seguindo esta ideia realizou-se a seguinte pesquisa no Google: -inurl:htm -inurl:html FORMATO SRC. Onde o FORMATO é substituído pelo formato buscado. Os resultados obtidos são mostrados na Tabela 1.1.

Formato	Resultados
X3D	35.300
W3D (ShockWave 3D)	17.500
SWF (Flash)	15.300.000
U3D (Universal 3D File, padrão ECMA - 363)	6.300
VRML	96.100

Tabela 1.1: Tabela com os resultados da pesquisa no Google por diversos formatos.

Estes números, apesar de não demonstrarem o número real de páginas utilizando o X3D, demonstram o número de páginas fazendo referência ao

assunto e isto é uma indicação direta a popularidade que o assunto tem na internet. Podemos notar alguns fatores interessantes.

- O X3D ainda é menos referenciado que a VRML.
- Os números do X3D e da VRML estão na mesma ordem de grandeza (dezenas de milhares).
- Todos os formatos têm utilização consideravelmente menor que a do flash (swf). Pode-se dizer que os demais formatos pesquisados têm uma presença insignificante quando comparada à do Flash. Mesmo ele não sendo especificamente 3D ele é utilizado para gerar efeitos 3D.

No seguinte trecho de Toni Parisi (Par06), pode-se verificar a importância que o autor dá ao desenvolvimento rápido, que ele acredita ser a principal demanda da indústria.

> 3D em tempo real está se tornando uma importante mídia para a web. As redes de dados e as placas de processamento gráfico agora têm poder o suficiente para permitir experiências 3D na web, incluindo jogos, mundos virtuais, simulações, aplicativos educacionais e aplicativos para treinamento. Desenvolvedores comerciais estão expressando crescente interesse em explorar o 3D em tempo real em aplicações web, para aumentar o valor dos produtos, criar experiências realmente imersivas e entregar informações de uma forma significante.
>
> Muita da infra-estrutura foi montada para permitir o desenvolvimento profissional de aplicações 3D em ambientes, multiplataforma e de código aberto. Padrões como o ISO para o X3D estão agora maduros, funcionais e robustos, são suportados por diversas iniciativas de código fonte aberto. Entretando, essas tecnologias só vão até ai. De fato elas são focadas na transmisão do conteúdo 3D e não na aplicação que o engloba. A indústria está necessitando de conteúdo 3D integrado e não em aplicações que o integrem. A indústria necessita de ambiente de desenvolvimento ágil para criar aplicações cliente-servidor ágeis e altamente interativas para a web.

O trecho acima indica o principal problema abordado neste trabalho: a dificuldade de desenvolvimento de aplicações mais robustas com o X3D que, embora seja um padrão aberto, poderoso, extensível e que possui um consórcio que o mantém e suporta, não provê um ambiente de desenvolvimento rápido.

Esse fator é provavelmente uma das principais causas da sua baixa adoção. Para analisar este fenômeno, é possível traçar um paralelo com o Flash e analisar porque este tem uma melhor aceitação.

Nesse intuito, observa-se o que o Flash tem que o X3D não tem.

1. Interface de edição gráfica (Deh04) [2] completa, bem aceita e com uma linguagem de programação conhecida por vários programadores.

2. Base de programadores grande com conhecimento da linguagem

Uma vez que são os programadores que decidem qual a tecnologia será utilizada para seus projetos, é importante oferecer recursos e tentar estar no ambiente de um grande número de programadores. Para atingir essas metas, podemos sugerir o desenvolvimento de um toolkit ou biblioteca que facilite a vida destes programadores, facilitando o processo de desenvolvimento e prototipagem e desta forma acelerando a aceitação da tecnologia.

1.2 Limitações do X3D

Como na VRML, o X3D propõe-se a descrever num arquivo toda a cena, que por sua vez será interpretada por um browser próprio ou um plugin para um browser existente. Isto à primeira vista parece ser uma boa ideia, porém como o modelo não prevê uma estrutura de comunicação com servidores nem uma forma de interação, a gama de aplicações onde ele pode ser aplicado fica limitada (But07).

Como tradicionalmente proposto na VRML, o X3D também se propõe a fazer o controle do comportamento das entidades existentes no modelo dentro do arquivo de descrição da cena. Com isso, se tem uma facilidade de fazer modelos auto-suficientes, mas a possibilidade de interação e comunicação cliente/servidor fica limitada. Como acontece nas novas aplicações web, também é esperado que no ambiente das aplicações 3D seja oferecido colaboração entre usuários e isto não é trivial de se alcançar no modelo proposto pelo X3D.

Podemos entender o mundo do X3D como diversos mundos 3D estanques onde cada usuário possui o seu e eles não se falam. Se for desejado fazer com que um mundo fale com o outro ou definir um único mundo onde diversos usuários interajam é necessário fazer uso de diversos artifícios complexos e pouco documentados. Pode-se entender um pouco melhor essa organização visualizando o modelo proposto na Figura 1.1.

Outro aspecto que deve ser discutido é a curva de aprendizado necessária para se programar aplicações com o X3D. Além da estrutura em XML ne-

[2] O Macromedia Flash Studio.

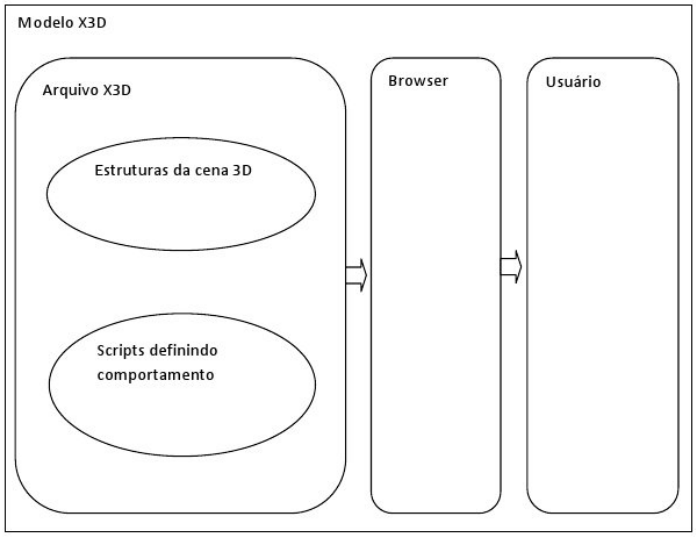

Figura 1.1: Modelo X3D.

cessária para a definição da cena, é também necessário conhecer ECMAScript para a definição das ações. Se for necessária comunicação com outras aplicações, o programador ainda terá de conhecer uma outra linguagem de programação para criar a interface. Todas essas camadas aumentam significativamente a curva de aprendizagem, pois não basta um programador conhecer uma determinada tecnologia bem para que ele possa trabalhar com o X3D, dependendo das necessidades de seu projeto ele pode ter de conhecer e dominar 3 diferentes tecnologias e aprender como fazer as interfaces entre elas.

O X3D também não lida muito bem com cenas muito complexas ou muito grandes. Isto ocorre porque devido ao seu projeto ele precisa passar toda a informação da cena de uma só vez ao carregá-la. Este método torna os tempos de carga para cenas muito grandes inviáveis para o ambiente web, além de tornar muito caro computacionalmente lidar com cenas complexas. Outra limitação detectável é o pouco suporte a simulações físicas complexas. Somente em 2006 o X3D passou a disponibilizar um sub-set para simulações físicas e até hoje seu suporte nos browsers ainda é incompleto (Rit06).

Mais um aspecto que pode ser explorado é menos concreto mas não menos importante, o entusiasmo da comunidade. O X3D é uma tecnologia derivada da VRML, esta por sua vez é uma tecnologia que na época do seu lançamento recebeu um grande destaque e criou uma grande expectativa da comunidade. Por diversos motivos a VRML não conseguiu atender as expectativas iniciais criadas sobre ele, oque criou um sentimento de decepção e um abandono gradativo da tecnologia. O X3D, por ser derivado, herdou da VRML este

desânimo da comunidade e isto é algo difícil de se alterar.

1.3 Necessidades de 3D na web

Para podermos avaliar a real utilidade do X3D temos que delinear funcionalidades que nos seriam necessárias para aplicações no mundo real. Pode-se citar algumas e seus motivos e depois explicar como elas podem ser alcançadas hoje com o que o X3D oferece.

- Criação e manipulação de uma cena 3D (Bou07, Beh04)
  - Esta funcionalidade é obvia para algo que se propõe a levar o mundo 3D para a web.
  - O objetivo é atingido através da própria definição de tags do X3D.
- Criação de eventos que respondam a inputs do usuário (Par06, Dac03).
  - Se é desejado que a aplicação seja interativa, de alguma forma ela precisa ser capaz de responder a eventos e estímulos do usuário, gerando outputs diferenciados de acordo com inputs diferentes.
  - O objetivo é atingido através de alguns tags específicos e da possibilidade de utilização de ECMAScript e extensão do padrão.
- Carga e persistência de dados dessa aplicação (Gre05, Lug05)
  - Se é desejado que o usuário possa interagir com algo, também será necessário em vários casos que ele possa guardar o estado da sua interação para uma outra futura interação.
  - O objetivo pode ser atingido através de complexas rotinas em ECMAScript e webservices que possuam um controle de sessão e estado.
- Interação com a GUI web (Par06, Hua09)
  - Se o X3D foi feito para estar na web e ser visto também através de browsers comuns, é importante que ele possa interagir com o resto do ambiente em que ele está inserido.
  - Pode ser feito através de chamadas em ECMAScript que requerem conhecimento da estrutura interna do X3D.
- Interação entre usuários acessando uma mesma cena (Bou07, Web07)

  - Com a transformação da web num ambiente cada vez mais colaborativo, aplicações na web também podem requerer que mais de um usuário possa alterar o estado da cena, e que as alterações deste estado possam ser visíveis para todos os usuários utilizando a aplicação no momento.
  - É possivel de ser alcançado através de complexas soluções envolvendo ECMAScript, código no servidor e controle de sessões.

- Integração com outras aplicações (Fri07)

  - Como parte de um sistema seria interessante que o X3D pudesse falar com outras aplicações.
  - É possível de ser feito através de ECMAScript e webservices.

Assim sendo, o objetivo desta dissertação é desenvolver algumas ferramentas para X3D e uma aplicação de demonstração para provar que a tecnologia é flexível o suficiente para se apresentar coligada a outras ferramentas, mais conhecidas da base de programadores, desta forma aumentando as chances de adoção do formato.

1.4 Organização da dissertação

O texto desta dissertação está organizado da seguinte forma. No Capítulo 2 detalhamos os objetivos desta dissertação. No Capítulo 3, descrevemos alguns trabalhos relacionados com as limitações do X3D que queremos abordar. O toolkit desenvolvido (DWeb3D) é apresentado no Capítulo 4, e no Capítulo 5 é mostrada uma aplicação de demonstração. Conclusões e trabalhos futuros são apresentados no Capítulo 6.

2
Trabalhos Relacionados

Como discutido anteriormente, algumas das características desejadas são bem atendidas pelo X3D, porém outras necessitam de métodos muito complexos ou muito trabalhosos para se alcançar o objetivo desejado.

Este capítulo analisará trabalhos que de alguma forma se relacionam com as características desejadas para o toolkit proposto. Porém, antes disto, é importante analisar o porquê da proposta de um toolkit e as formas como ele pode ajudar.

2.1
Histórico

Para entender o porquê da baixa adoção do padrão, precisa-se estudar onde ele se limita e o que se espera atualmente de tecnologias 3D na web, porém antes é interessante analisar o histórico de evolução da própria web e para onde esperamos que ela vá.

A internet foi desenvolvida pela ARPA (Advanced Research and Projects Agency) em 1969 sob encomenda dos militares norte americanos, com o objetivo de conectar os departamentos de pesquisa e bases militares. Esta rede foi batizada com o nome de ARPANET. Antes mesmo dela, já existia outra rede que ligava estes departamentos e bases, mas como os EUA estavam em plena guerra fria, e toda a comunicação desta rede passava por um computador central que se encontrava no Pentágono, sua comunicação era extremamente vulnerável.

A ARPANET (Kha89) foi desenvolvida com o intuito de ser descentralizada e portanto não suscetível a um ataque. Com um Back Bone subterrâneo, ela ligava os militares e pesquisadores sem ter um centro definido ou mesmo uma rota única para as informações. Nos anos 1970, as universidades e outras instituições que faziam trabalhos relativos à defesa tiveram permissão para se conectar. Em 1975, existiam aproximadamente 100 sites.

No final da década 1970, a ARPANET tinha crescido tanto que o seu protocolo de comutação de pacotes original, chamado de Network Control Protocol (NCP), tornou-se inadequado. Depois de algumas pesquisas, mudou-

se do NCP para um novo protocolo chamado TCP/IP (Transfer Control Protocol/Internet Protocol) desenvolvido em UNIX. A maior vantagem do TCP/IP era que ele permitia (o que parecia ser na época) o crescimento praticamente ilimitado da rede, além de ser fácil de implementar em uma variedade de plataformas diferentes de hardware de computador.

Atualmente, a internet é composta de aproximadamente 50.000 redes internacionais, sendo que mais ou menos a metade delas nos Estados Unidos. A partir de julho de 1995, havia mais de 6 milhões de computadores permanentemente conectados à Internet. Em 2009 o número estimado é de aproximadamente 60 milhões. (Informações obtidas em http://www.zakon.org/robert/internet/timeline/).

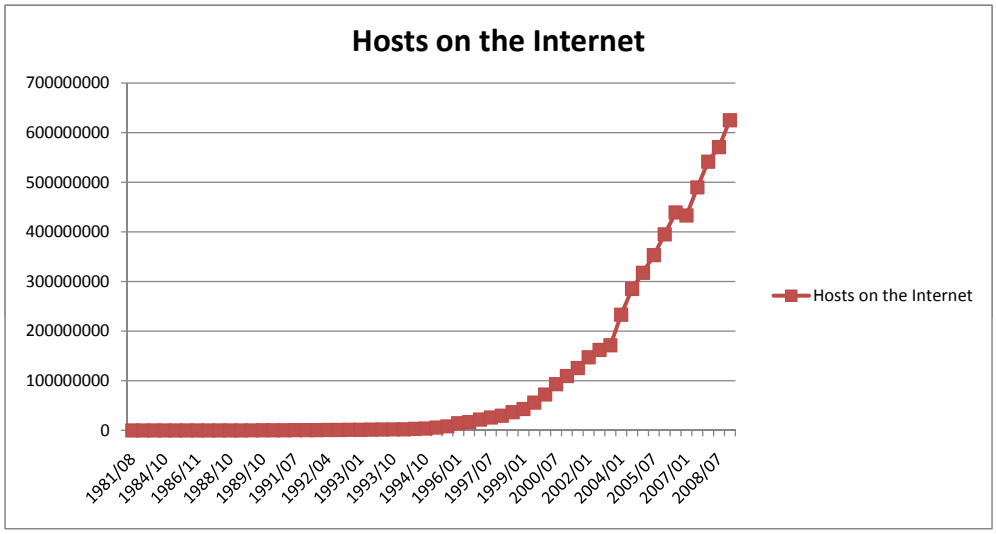

Figura 2.1: Hosts na internet conforme os anos. Fonte: http://www.zakon.org/robert/internet/timeline/

Analisando a Figura 2.1 pode-se notar como o crescimento tem acelerado nos últimos anos, e a expectativa é que continue assim por mais alguns anos. A importância disto para o mundo 3D e para o X3D é que quanto maior e mais desenvolvida a web se torna, mais tecnologias ela engloba e mais soluções são apresentadas através de suas numerosas páginas. Dentro destas soluções estão algumas relacionadas ao mundo 3D.

Com a evolução da web as aplicações estão cada vez mais migrando para funcionar neste novo ambiente. Hoje já é possível encontrar na web aplicações

antes exclusivas dos desktops, como um editor de textos por exemplo. Estas aplicações requerem uma maior capacidade dos browsers que têm evoluído muito. Além disso também necessitam de mais interação e fazer bom uso das qualidades do ambiente, que são a capacidade de distribuição de interatividade de vários usuários num ambiente compartilhado.

Com o crescente conceito de cloud computing (Bos07, Buy09) é esperado que mais e mais aplicações migrem para o ambiente web. E isso não deixa de incluir as aplicações que fazem grande uso do 3D.

Na web, a história de aplicações 3D teve inicio com o VRML como uma forma de descrever cenas 3D para o ambiente das páginas web. Suas aplicações iam desde a demostração de produtos como modelos 3D, a mundos virtuais com bate papo ou aplicações de visualização. No inicio as aplicações eram simples demonstrações de cenas em que o usuário poderia navegar. Com o tempo foi adicionada capacidade de scripts com gatilhos para incluir reações programadas à cena, adicionando assim mais interatividade. Estes scripts permitiram também uma futura expansão.

Após alguns anos de utilização chegou-se à conclusão de que a VRML era limitado em sua forma, e para substituí-lo se desenvolveu o X3D. Este novo padrão utiliza XML como estrutura básica e propõe métodos para extensão. Porém ainda herda a estrutura de definição da VRML e talvez esta seja a sua maior limitação.

2.2 Toolkits e criatividade

Este assunto é largamente explorado em (Gre07) e é de fundamental importância para o desenvolver deste trabalho. Sua importância se dá porque estamos sugerindo que podemos pegar uma tecnologia poderosa porém pouco adotada (X3D) e através da adição de ferramentas disponibilizar um pacote que permita facilitar o processo de desenvolvimento e com isso agradar a um número maior de desenvolvedores.

Como descrito no artigo mencionado acima, para que uma nova área possa se desenvolver, é necessário dar aos desenvolvedores capacidade de testar a sua criatividade através de ciclos de protótipo, testes, análise, protótipo, etc. Este ciclo previsto nos métodos de desenvolvimento incremental possibilita ao desenvolvedor verificar o progresso feito em curtos períodos de desenvolvimento, arriscar mais, pois a perda caso o produto saia ruim é menor, e verificar suas ideias com testes. A verificação de ideias é um dos principais aspectos no conceito de focar na criatividade, pois permite ao desenvolvedor verificar se algo que ele criou funciona sem o grande ônus de um projeto mais complexo.

Existem alguns casos clássicos em que a presença de uma estrutura de abstração das camadas mais baixas foram fatores fundamentais para o sucesso da tecnologia. O exemplo mais conhecido é o da própria web. O advento do HTML possibilitou uma rápida adoção do ambiente como repositório de informações, pois bastava aos desenvolvedores escreverem seus textos de forma estruturada e ligá-los a outros. Outro exemplo conhecido é o próprio flash, pois a sua arquitetura possibilitou a popularização de páginas mais dinâmicas, com animações interativas ou filmes.

Porém, como alcançar essas metas dentro de um projeto com requisitos complexos e que precisam de uma base funcional grande? Parte desta resposta vem da reutilização de código. Se o desenvolvedor puder reutilizar aquilo que ele desenvolveu para seu protótipo inicial, ou melhor ainda reutilizar algo que alguém já fez por ele, o ônus de fazer isso novamente fica eliminado. Porém não basta reutilizar código, é necessário também eliminar camadas desnecessárias em operações de baixo nível que não interessam à aplicação. Neste ponto o toolkit é extremamente útil, pois ele encapsula diversas funções recorrentes e permite ao desenvolvedor focar no que ele realmente deseja a aplicação.

Um último aspecto que vale a pena comentar, é que o uso de toolkits promove boas práticas de programação, pois eles já carregam um conceito de desacoplamento, organização de código e modelagem, que pode ser reaproveitado no código final.

2.3 Aplicações possíveis

Como em qualquer ambiente de programação as aplicações possíveis com o X3D estão dentro de uma gama bastante ampla e são limitadas principalmente pelo que a imaginação dos programadores e as possibilidades de hardware permitem. Porém, é interessante explorar alguns casos mais concretos para verificar como a tecnologia pode responder a problemas da vida real.

2.3.1 Visualização

Aplicações de visualização são aquelas onde o objetivo principal é observar um modelo 3D de diversos ângulos e pontos de vistas diferentes. Podemos citar visualizações de CAD, projetos de avião, plataformas, etc. Nestas aplicações, detalhes como escala correta e possibilidade de navegação na cena são muito importantes, pois elas primariamente se destinam a testar e auxiliar o desenvolvimento de projetos sem a necessidade de construção de maquetes, o que infere numa redução de custos (Figura 2.2). O X3D possibilita a visu-

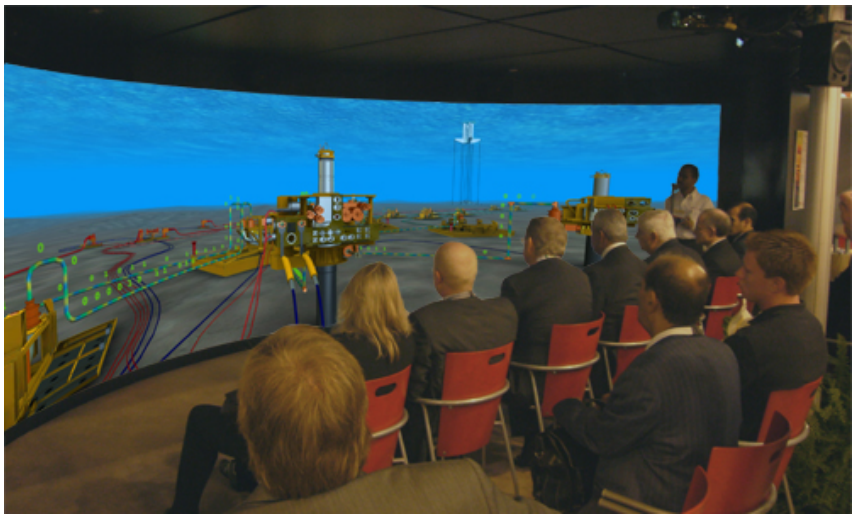

Figura 2.2: Exemplo de um software de visualização que utiliza o X3D. Fonte: http://www.web3d.org/casestudies/detail/fmc-wins-prize-for-best-exhibition-with-octaga-panorama/

alização através de sua estrutura hierárquica de nós. Vale lembrar que essas cenas ainda são limitadas pelo tamanho que o arquivo pode ter na web, uma vez que pelo modelo padrão o X3D deve ser carregado todo de uma só vez.

2.3.2 Simulações

Simulações normalmente são aplicações destinadas a reproduzir algum ambiente, para fins de testes ou de treinamento. Elas podem ou não incluir características de física. Um bom exemplo para aplicações de simulação seria um jogo para treinamento militar, um simulador de voo para treino de pilotos, ou até mesmo um simulador físico onde se verifica onde ocorreriam falhas numa estrutura (Figura 2.3). Com o X3D é possível fazer simulações através da definição de uma cena e a adição de scripts à ela, assim criando respostas a eventos. Porém ele não possui nós específicos para simulação física e o controle sobre a câmera é limitado.

2.3.3 Jogos

A indústria de jogos para computador tem uma ligação muito grande com o universo das aplicações 3D. Eles quase sempre são os primeiros a explorarem novas técnicas e investem bastante recursos com pesquisa. Este fenômeno é em parte devido ao grande retorno financeiro que a indústria de entretenimento digital possui hoje em dia. É possível desenvolver jogos utilizando o X3D, porém a complexidade dos jogos atuais cresceu de forma tamanha que adicionar

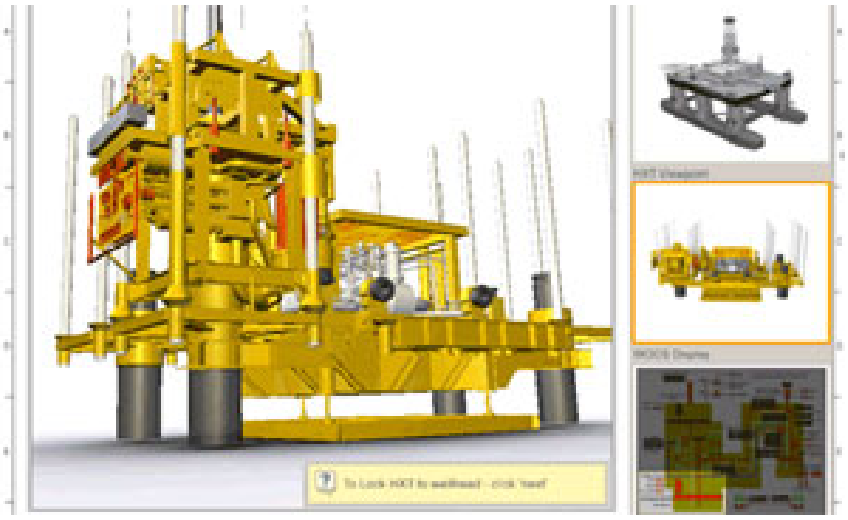

Figura 2.3: Imagem de um sistema de treinamento para situações de risco em plataformas. Fonte: http://www.web3d.org/casestudies/detail/training-and-mission-rehearsal-prior-to-oil-and-gas-subsea-operations/

os efeitos e reações esperadas no X3D é um desafio bastante complexo.

Os jogos podem ser de vários estilos, dentre eles podemos destacar os MMORPGs e os jogos online, por estarem já no ambiente da web. Estes jogos privilegiam a interação entre vários jogadores e novamente isto é um ponto chave que é complicado desenvolver no X3D.

2.4 Metas

Um trabalho sobre o X3D e as tecnologias existentes na web pode ser bastante extenso e cobrir uma gama enorme de assuntos, aspectos e implicações. Após uma análise da realidade existente, faz-se necessário delimitar esse universo no escopo deste trabalho. Para isso determina-se metas enumeradas abaixo:

1. Delinear possíveis causas para a pouca adoção do X3D.
2. Propor uma abordagem para este problema.
3. Desenvolver uma aplicação de demonstração e analisar seus resultados.

Na Seção 1.2, traçamos algumas necessidades de 3D na web e verificamos que algumas não são bem atendidas pelo X3D. Assim, a abordagem proposta passará pelo projeto e implementação de um toolkit de desenvolvimento de aplicações X3D que englobe as seguintes características:

– Colaboração entre usuários;

– Interação com a GUI Web;

– Integração com outras aplicações;

– Persistência de dados.

Além da implementação dos recursos acima, também é objetivo do toolkit prover as abstrações necessárias para que os desenvolvedores de aplicações com o X3D consigam desenvolvê-las de maneira mais rápida e com menor curva de aprendizado.

2.5 Colaboração

Uma das metas almejadas é possibilitar a colaboração e a interação de duas ou mais pessoas numa cena 3D (Jou08). Este tipo de interação é bastante comum em jogos multi-jogadores, pois um jogador afeta a experiência online do próximo. Como esta foi uma das formas pioneiras de colaboração no ambiente da internet, vale a pena discuti-la aqui.

2.5.1 MMOGs

Nos jogos o uso de 3D e da web possibilita a criação de mundos virtuais dinâmicos habitados por "avatares"[1]. Esse gênero é chamado MMOG (Massively Multiplayer Online Game) e surgiu em meados de 1996 com um jogo chamado Meridian 59[2], que apesar de pouco conhecido existe até hoje. Já em 1998 foi lançado o Ultima Online[3] que veio popularizar o gênero alcançando números de jogadores pagantes nunca antes vistos. Porém nenhum desses jogos era realmente 3D na época. O primeiro MMOG totalmente 3D foi o EverQuest[4] e este também foi um sucesso abrindo assim o caminho para centenas de jogos mais modernos (Liu07, Tay05).

Os jogos e as aplicações de visualização criaram o conceito de mundo virtual, ou seja, um ambiente persistente e dinâmico onde diversos avatares convivem e interagem, alterando assim o ambiente e a forma de percepção dessa realidade. No final de 2004 e início de 2005 foram lançados dois produtos que possuem ambientes virtuais de grande expressão, o World of Warcraft[5] e o Second Life[6]. O World of Warcraft tem como grande mérito o sucesso que alcançou, com números não oficiais de assinantes em torno de onze milhões.

[1] Representação virtual de um personagem.
[2] http://meridian59.neardeathstudios.com/
[3] http://www.uo.com
[4] http://www.everquest.com/
[5] http://www.worldofwarcraft.com
[6] http://www.secondlife.com

Ele é hoje, desse gênero, o mais jogado e provavelmente um dos jogos com maior sucesso comercial no mundo. Já o Second Life possui uma proposta bastante diferente, pois permite que seus usuários criem livremente o conteúdo do mundo virtual. Seu grande mérito é ter pela primeira vez trazido o mundo dos negócios para dentro de um mundo virtual, sendo que hoje em dia falar de negócios com 3D na web sempre traz à mente das pessoas o Second Life. Pelo fato dele ser a plataforma mais genérica disponível atualmente, é interessante explorar um pouco melhor o seu funcionamento.

2.5.2 Second Life

Lançado em 2005, ele foi o primeiro produto a propor à comunidade que construísse livremente o conteúdo de seu mundo virtual. Nele uma pessoa entra, cria um avatar e com ele passa a explorar um mundo virtual completamente criado por outras pessoas comuns denominadas residentes dentro do ambiente virtual. Para criar itens não é necessário possuir qualquer tipo de permissão especial, bastando achar uma área de construção livre chamada sandbox. Porém, para se colocar itens de forma permanente é necessário possuir um pedaço virtual de terra, que é necessário comprar dos desenvolvedores do mundo, além de pagar uma taxa mensal de manutenção.

Em seu modelo o Second Life também permite a compra e venda de itens criados pelos residentes, e isso gerou um comércio virtual e uma economia virtual bastante interessante.

Para entender porque o Second Life é hoje um sucesso e movimenta milhões de dólares mensalmente, devemos entender um antigo desejo que havia na comunidade web. As pessoas desejavam, e de certa forma ainda desejam, ter uma representação do mundo real no ambiente virtual que eliminasse as barreiras de distância, mas que mantivesse os paradigmas com os quais elas estão acostumadas, como o modelo de prédios, lojas, endereços, etc. O outro lado da moeda eram os desenvolvedores, que procuravam um ambiente que os permitisse criar seus modelos e cenas e comercializá-los, sem ter que lidar com a camada mais baixa de tecnologia necessária para um ambiente desta complexidade.

Com a combinação de usuários procurando novas possibilidades de mundos virtuais, desenvolvedores procurando formas de desenvolver mais facilmente e empresas procurando novas formas de se promover e abrir novos negócios, surgiu um novo mundo bastante complexo, mas com novas possibilidades dentro da internet.

É importante verificarmos que este produto possui muitas das qualidades

discutidas no artigo (Gre07) e que ele é um grande sucesso hoje em dia. Porém, mesmo sendo um sucesso e sendo a referência em termos de ambientes virtuais, o Second Life não possui uma grande integração com as páginas web e ainda tem muito de sua infra-estrutura controlada pela companhia que o criou, o que limita a possibilidade de expansão e de derivação deste produto em novas propostas.

Recentemente um grupo de desenvolvedores independentes tem tentando desenvolver um servidor alternativo para o Second Life chamado Open Simulator[7]. A iniciativa teve bastante progresso e já é possível conectar-se a um conjunto paralelo de servidores. Porém esse novo ambiente ainda é pequeno, bastante instável e não conta com a capacidade de vendas. Assim sendo ele tem um número de usuários ainda pequeno se comparado com os servidores oficiais.

Mesmo sendo um produto interessante, a ele falta a flexibilidade de um padrão aberto, e a possiblidade de expansão que o X3D possui.

2.5.3 BS Collaborate

Solução da Bitmanagement[8], uma empresa alemã especializada em produtos para o X3D, o BS Collaborate é um produto que permite a visualização de vários usuários numa mesma cena. Permite também que eles conversem utilizando uma interface dentro da cena 3D. Podemos verificar o funcionamento do servidor através do diagrama da Figura 2.4.

O BS Collaborate utiliza um servidor SQL para guardar as informações dos diversos usuários visitando a cena e as utiliza para interagir com os clientes que utilizam um protocolo próprio de troca de mensagens. Com isso ele consegue mostrar as diversas pessoas visitando a cena, as conversas, mas a cena ainda precisa ser a mesma, e essa limitação de alteração da cena durante a visualização é ruim, pois evita que se criem efeitos mais interessantes. Além disso ele possui um outro fator limitante ao uso da comunidade, o custo. Trata-se de um produto comercial e com um custo elevado para desenvolvedores autônomos.

Este produto é interessante por ser uma proposta de colaboração sobre a plataforma X3D, porém ele é fechado, caro e não possui todas as características que se listam aqui como desejadas.

[7]http://opensimulator.org
[8]http://www.bitmanagement.de/

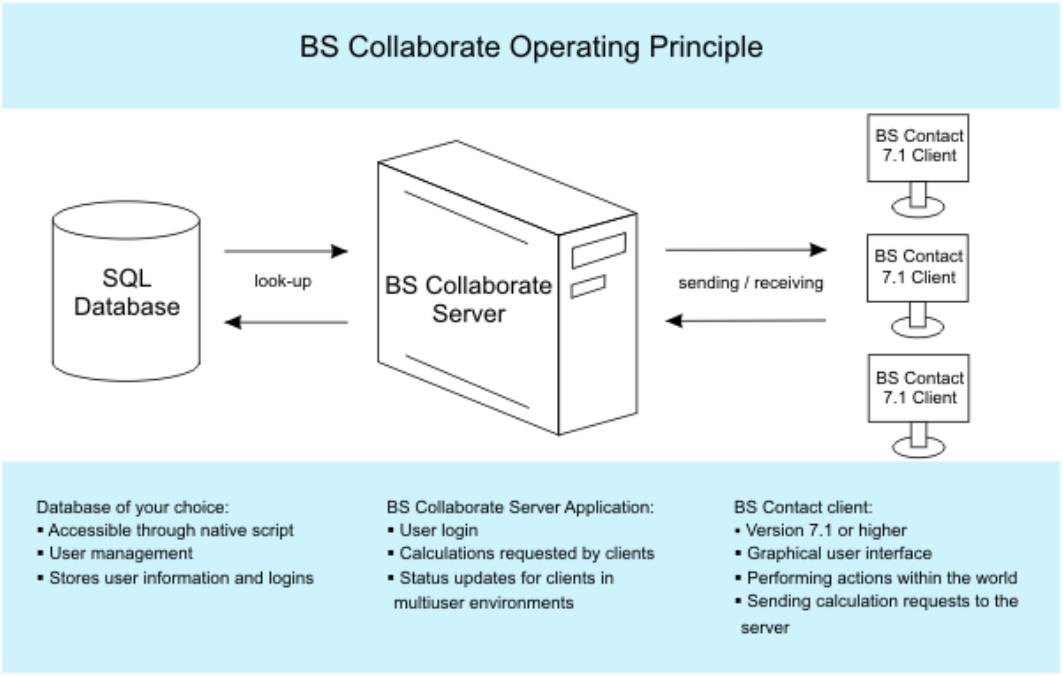

Figura 2.4: Diagrama BsContact Fonte: http://www.bitmanagement.de/

2.5.4 X3D Multi-user Virtual Environment Platform for Collaborative Spatial Design

O artigo (Bou07) nos mostra como um ambiente 3D colaborativo pode ajudar em trabalhos onde o planejamento do espaço é o objetivo final. Sua importância para este trabalho é a análise de como a colaboração pode funcionar em um ambiente X3D, a listagem de motivos pelos quais o X3D foi escolhido, as formas sugeridas de colaboração e, principalmente, mostrar que colaboração é sim um fator importante para a nova geração de aplicações que estão surgindo. O artigo cita as características que ele deseja alcançar em sua aplicação no seguinte trecho:

> *Canais de comunicação: O ambiente deve possibilitar a comunicação entre os colaboradores de forma a facilitar a interação. Isto deve ser feito oferecendo suporte a diversos canais de comunicação como gestos, conversas por voz e texto.
>
> *Características gerais da colaboração: O ambiente deve permitir os usuários colaborar a distância, através da oferta de ferramentas tais como: Manipulação de objetos 3D compartilhados, bloqueio e desbloqueio de objetos compartilhados, assim como administrar os usuários. (...)
>
> *Características da colaboração espacial: (...) ela é útil para representar o mesmo espaço de diversos modos. (Ex: Ponto de

Vista 3D assim como representação da planta baixa em 2D do mesmo ambiente). Além disso os usuários devem ter ferramentas para manipular os móveis além de ferramentas para desenvolver seus próprios modelos e ambientes.

*Presença, percepção e representação dos usuários: A sensação da presença de outra pessoa e a percepção de atividade permite aos usuários estruturar a sua própria atividade, interagindo, comunicando e colaborando de forma progressiva e transparente. O ambiente pode ser populado por diversos usuários ao mesmo tempo, que devem ser representados.

Pode-se notar que as metas propostas são bastante parecidas com as propostas neste trabalho. O artigo também sugere a persistência através de menção de avatares cuja presença deve ser percebida por todos e ser consistente.

2.5.5 An Open Protocol for Wide-area Multi-user X3D

Descrito no artigo (Web07), trata-se de uma proposta de um novo protocolo para sincronização de cenas X3D. Por se tratar de um protocolo para utilização na web e por ser previsto para funcionar até com implementações em JavaScript, ele tenta reduzir ao máximo o tamanho das mensagens de forma a otimizar a comunicação.

O protocolo faz uso de TCP e UDP, priorizando o UDP para atualização de parâmetros com mudanças frequentes e o TCP para funções como carga.

Os desenvolvedores também levaram em conta qual seria o melhor formato para um protocolo destes, mesmo assumindo que um formato binário perde a legibilidade e dificulta o debug, chegaram a conclusão que o menor tamanho e facilidade de decodificação deste formato são tão necessários que os aspectos ruins são aceitáveis e portanto propõem a formatação explícita.

A importância do protocolo é que ele permite a sincronização de entidades e portanto a implementação da persistência através de conversa com o servidor.

2.6 Interação com a GUI Web

Dentro das características desejadas às novas aplicações web, a interação com outros componentes da GUI vem listada. Sua importância se dá porque a tecnologia de páginas web é amplamente aceita e a base do que é a internet hoje em dia, e não integrar significa não poder tirar proveito do ambiente mais importante da web.

2.6.1 VRML e X3D chat worlds

O chat world foi uma das primeiras aplicações a se difundir na web utilizando principalmente o VRML, mas existem algumas versões com X3D. Trata-se de um ambiente de bate-papo onde você pode visitar o mundo virtual e conversar com outras pessoas que estão visitando o mesmo mundo. As versões iniciais eram pouco mais que uma página contendo uma cena 3D e um cliente de chat localizado na mesma página. Não havia colaboração, interação com a cena a não ser a navegação, não havia um avatar ou formas de ver os outros participantes.

Com o tempo versões mais avançadas foram surgindo, o conceito de avatar foi criado e adicionou-se a interação. Podemos citar o ABNet[9] como um cliente desta época que ainda existe hoje. Esta tecnologia específica funciona através de um cliente java e scripts ActiveX[10] que interage com o browser X3D/VRML (BSContact neste caso) e faz a atualização das mensagens necessárias. Ele ainda utiliza uma caixa de diálogo em separado para ver a conversa e a interação se limita a poder ver o avatar de um outro visitante e conversar com ele na caixa de diálogo.

Apesar de ter sido bastante interessante para a época e até terem sido criadas diversas cenas para este conceito (veja alguns exemplos em: http://vrmlworld.net/), esta proposta inclui diversas limitações, como a impossibilidade de se manipular a cena e de criar interações maiores dentro dela. Porém mesmo com as limitações, esta foi uma das formas mais adotadas de uso do VRML, porque ela já inclui um pacote que engloba algumas das funcionalidades desejadas pelos desenvolvedores e que eram difíceis de se alcançar com o X3D puro. Além disso, ela possui algum nível de interação com as páginas web por estar dentro dela e o chat ser um componente da página.

2.7 Integração com outras aplicações

Apesar de normalmente estarem limitadas ao seu browser e ao ambiente virtual 3D, as cenas X3D podem ter de interagir com dados e eventos provenientes de outras aplicações para que possam ser mais interessantes.

[9] http://kimballsoftware.com/abnet/

[10] Tencologia proprietária da Microsoft para criar plugins para o internet explorer.

2.7.1 Vivaty

Este talvez seja o mais novo produto utilizando X3D no mercado. Ele foi desenvolvido pela antiga MediaMachines[11], que hoje mudou de nome para o nome do produto. Sua principal qualidade é a possibilidade de os usuários utilizarem avatares personalizados, e visitar diversas cenas diferentes, inclusive podendo enviar novas cenas para a empresa. Esta, no entanto, não garante que a nova cena será disponibilizada e a interação com a cena ainda é limitada. O produto usa largamente um conceito chamado AJAX3D (Par06). Utilizando técnicas de chamadas assíncronas via scripts consegue-se fazer com que a cena e diversos componentes da UI web interajam.

Este produto faz boa parte de suas operações utilizando o X3D padrão, porém nem tudo, pois necessita de um plugin para o browser específico e usa um protocolo proprietário de comunicação com o servidor. Apesar destas limitações, é um produto interessante pois apresenta uma nova abordagem na forma de oferecer soluções X3D, podendo inclusive se integrar com sites de comunidade famosos como o FaceBook. A empresa também oferece uma aplicação para desenvolver cenas para a plataforma chamada Vivaty Studio[12].

O principal problema desta plataforma é ser proprietária. A companhia controla o conteúdo e não disponibiliza o servidor para terceiros. Isso inviabiliza que desenvolvedores criem novas soluções baseadas na tecnologia da Vivaty[13].

2.8 Persistência de estado

Uma outra funcionalidade necessária é permitir a persistência de estados entre diversas instâncias de uma aplicação X3D. Por estar intimamente relacionada com a colaboração, alguns dos trabalhos apresentados na Seção 3.2 possuem esta funcionalidade. Um exemplo é o trabalho apresentado em (Bou07), que entre outras coisas, explica que a persistência da posição dos avatares é importante, pois cria uma percepção de mundo imersivo onde o observador se coloca dentro da representação virtual.

A persistência em banco de dados também é interessante por permitir que diversos servidores tenham acesso a um ambiente comum de forma simples, além de permitir uma armazenagem mais eficiente do ambiente virtual. Porém não foi possível encontrar nenhum trabalho propondo isto de forma específica para X3D. Desta forma a implementação do DWeb3D foi feita através da

[11]Empresa conhecida no mundo web3D como uma das pioneiras no ramo.
[12]http://www.vivaty.com/downloads/studio/
[13]http://www.vivaty.com

utilização de técnicas genéricas de banco de dados orientado a objetos como descrita em (Gre05). Com essas técnicas é possível armazenar todo o universo de objetos da aplicação, e restaurá-los quando necessário.

2.9 Análise final dos trabalhos

Quando se analisa os trabalhos listados dentro da proposta de estudar as falhas no X3D, verificamos que para alguns pontos existem boas soluções e para outros nem tanto. Para o caso de colaboração existem diversas soluções, porém com a ressalva de que não se pode verificar nenhum produto que permitisse a criação ou alteração de novos ítens na cena de forma dinâmica (isto é, sem precisar editar um arquivo X3D). Para o caso da interface com o GUI web as técnicas ajax atendem bem ao problema, porém só estão aplicadas no Vivaty e através de scripts proprietários. No caso de persistência, existem algumas soluções que atendem até certo ponto ao problema, mas nenhuma verificável como prática, pois são compostas de um conjunto complexo de técnicas. Já no caso de integração com outras aplicações pode-se verificar os protocolos de sincronização com o servidor como sendo uma opção interessante.

Dentre os produtos aqui analisados podemos destacar o sucesso alcançado pela Unity 3D. Este sucesso se deve em grande parte à sua interface intuitiva, seu baixo custo comparativo, e sua grande de gama de funcionalidades, podendo listar dentre elas: simulação física, capacidade de importação de uma grande quantidade de formatos de arquivos 3D, exportação para web, linguagem de scripts fácil e poderosa, animações e suporte a diversos tipos de esqueletos, etc. Além disso o motor possui um editor fácil e poderoso e uma ativa comunidade de desenvolvedores independetes que engloba desde empresas a entusiastas.

Tanto a Unity como o X3D e os outros padrões mencionados têm uma vocação específica. A Unity se apresenta como uma solução focada em jogos, com uma grande gama de ferramentas disponíveis para este fim. Ela possui uma dinamicidade muito grande e muito necessária quando se quer desenvolver uma aplicação com a gama de possibilidades de um jogo. Por outro lado o X3D nasceu focado na web, possui um histórico impressionante e é considerado uma tecnologia madura.

Tentar achar uma declaração definitiva de qual é a vocação de uma tecnologia específica é assunto árduo e controverso. Como ocorre com a maioria das coisas sempre vão existir entusiastas por fazer a mesma coisa de diversas formas e cada método trará benfícios e defeitos. Porém, o X3D se apresenta como uma ótima opção para definição de cenas estáticas, protótipos e pequenos

ambientes 3D. No entanto a ele falta a dinamicidade e a facilidade de interação com outras tecnologias e abordagens. Este trabalho se foca exatamente neste ponto.

O próximo capítulo apresentará o toolkit desenvolvido e como ele aborda as características desejadas.

3
DWeb3D

Supondo que as limitações apresentadas anteriormente sejam o motivo principal para a baixa adoção do X3D, propõe-se desenvolver um conjunto de ferramentas que reduza a curva de aprendizado e o tempo necessário para o desenvolvimento de uma aplicação X3D, ao mesmo tempo que promova a reutilização racional de código.

Seguindo esta linha de raciocínio foi desenvolvido um toolkit para auxiliar o desenvolvimento de aplicações colaborativas utilizando o X3D e uma aplicação que faça uso deste toolkit como prova de conceito. Entende-se como um toolkit um conjunto de bibliotecas de desenvolvimento que lidem com diversos dos desafios encontrados no desenvolvimento de uma aplicação com determinadas características. O objetivo principal é retirar o fardo de lidar com tarefas repetitivas e complexas, dando assim ao desenvolvedor a possibilidade de focar-se no problema que ele deseja resolver, tornando o processo mais criativo. (Gre07).

3.1
Metodologia

Tanto para o estudo como para a aplicação de demonstração foram usadas metodologias bem definidas.

3.1.1
Metodologia para o estudo

Primeiro foram analisados quais são os problemas e como abordá-los. Analisou-se o porquê da baixa adoção do X3D e foram propostos métodos de trabalhos e ferramentas que possam ajudar a mudar esse quadro e tentar medir de alguma forma esta proposta.

Para poder medir o nível de aceitação do padrão foi utilizada uma análise das ocorrências informadas por uma ferramenta de busca. Mesmo não sendo um método muito preciso, ele resulta em valores interessantes, pois a ferramenta indexa tudo que encontra na internet e não é afetada pelo tipo de arquivo.

Assim o sendo, é razoável supor que sua base esteja distribuída igualmente entre todos os formatos.

Destes resultados partiu-se para análise das capacidades do X3D, através de pesquisa na documentação do mesmo e análise dos produtos existentes que o utilizam. O objetivo desta analise foi detectar onde o X3D está sendo utilizado, onde não está e o porquê disto. O resultado foi a lista de limitações detectadas.

Finalmente, com as limitações foi gerada uma proposta que tenta melhorar parte dos problemas. Esta proposta concretizou-se no tookit desenvolvido, que serviu como método de analisar se a proposta pode ou não ajudar no processo de desenvolvimento de aplicações X3D.

3.1.2 Metodologia para o desenvolvimento da aplicação

Como existia somente uma pessoa trabalhando no desenvolvimento da tecnologia, métodos de desenvolvimento mais tradicionais seriam dispendiosos e pouco práticos, portanto foi adotado uma metodologia um pouco diferente que se mostrou eficaz para o caso de uma pessoa como desenvolvedor.

Inicialmente as tecnologias disponíveis foram avaliadas e uma foi selecionada. Os critérios para a avaliação foram:

- Familiaridade: se o desenvolvedor já a conhece, tanto melhor pois não exige uma curva de aprendizado
- Popularidade: é uma tecnologia popular que vários outros desenvolvedores conhecem e utilizariam?
- Robustez: é algo robusto e que permitiria uma modularização razoável?
- Aplicabilidade: a tecnologia possui características que a permitam ser utilizada no ambiente escolhido?

Para o desenvolvimento foi utilizado um método que é uma adaptação das metodologias ágeis. Para isso alterou-se ligeiramente a ideia de ciclos e gestão de tarefas pequenas do scrum (Sch01, Bat08). Os ciclos e as revisões com os usuários foram trocados por metas. Assim sendo, cada meta seria um ciclo para o desenvolvimento incremental. Antes de cada ciclo se determinava uma lista de tarefas necessárias para alcançá-lo e a cada dia uma tarefa era selecionada. Esta lista poderia ser modificada na medida da necessidade. A prática de se definir um construto finalizado ao fim de cada ciclo foi trocada pela construção de um teste da funcionalidade desejada (Figura 3.1).

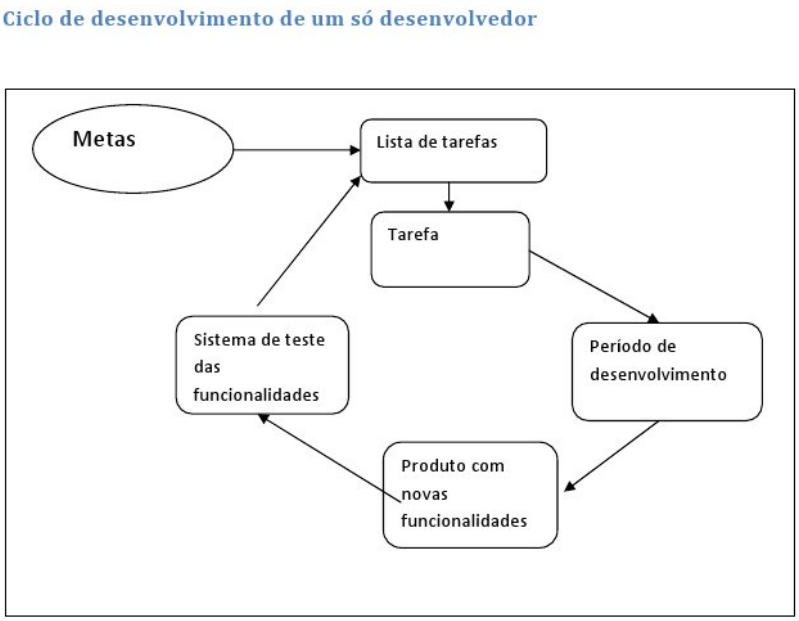

Figura 3.1: Ciclo de desenvolvimento para um só desenvolvedor

3.2 Tecnologias

Para o desenvolvimento do toolkit foi eleita a linguagem C# dentro da plataforma .NET. A escolha foi feita seguindo os critérios de escolha anteriormente listados, sendo:

- Popularidade: é amplamente utilizada e mantida por uma empresa muito grande, além de possuir uma comunidade de desenvolvedores crescente e bastante robusta.
- Robustez: existe há vários anos no mercado e é utilizada nos mais diversos meios, desde de aplicações corporativas a jogos.
- Aplicabilidade: o .NET possui um framework para trabalhar com aplicações web e é bastante utilizado neste meio, além de possuir capacidade de criação de bibliotecas compiladas e pacotes para redistribuição de código.

Como controle de versão foi utilizado o Mercurial[1]. As qualidades que o tornam interessante para um projeto de um só desenvolvedor são a possibilidade de funcionamento sem um servidor centralizado e a possibilidade de sincronização de diversos repositórios, desta forma possibilitando o desenvolvimento em máquinas diferentes e facilitando as cópias de segurança.

[1]http://mercurial.selenic.com/wiki/

Finalmente, para o caso de estudo foi decidido desenvolver um plugin de renderização para a Unity3D[2]. A Unity3D é um motor gráfico desenvolvido por um grupo americano, que se propõe a ser modularizável e extensível. Ela suporta uma grande gama de formatos para importação, possui um IDE[3] bem completo além de outras características interessantes. A escolha foi feita porque a Unity3D possui uma linguagem de scripts baseada em C#, facilitando a integração com o código já existente, possui um plugin para utilização na web e possibilitaria uma interface mais rica do que utilizar um browser X3D existente, uma vez que com estes a interface teria que ser feita em ECMAScript e seria mais complexa.

O X3D possui uma API em ECMAScript já definida chamada SAI[4] que se propõe a disponibilizar o acesso de objetos da cena dentro do browser. O problema de sua utilização é que sua programação fica restrita ao ambiente do cliente de visualização e portanto não é própria para uma interação mais complexa.

3.3 Estrutura do DWeb3D

O toolkit se divide em módulos organizados por função, cada módulo representa uma dll[5]. Isto possibilita um uso mais racional, uma vez que se não forem necessárias todas as funções, não é necessário adicionar à aplicação todas as dlls. O Apêndice A contém alguns gráficos UML representando a estrutura do DWeb3D.

A divisão das dlls é a seguinte (Figura 3.2):

- Model: dll onde está o grafo. É a base de todo o toolkit e necessária para se utilizar as outras partes.
- ObjetSync: dll responsável para sincronização de características dos objetos em diversas instâncias através da rede.
- DWebServer: executável que demonstra como um servidor pode funcionar.
- Unity Render: plugin para possibilitar o uso do X3D com a Unity.
- Util: conjunto de ferramentas utilizado pelo Unity render.
- Unity WebPlayer: plugin para browser que permite a visualização do projeto na Unity.

[2]http://unity3d.com/
[3]Integrated Development Enviroment
[4]Scene Access Interface
[5]Biblioteca dinâmica. Do inglês Dinamic Load Library.

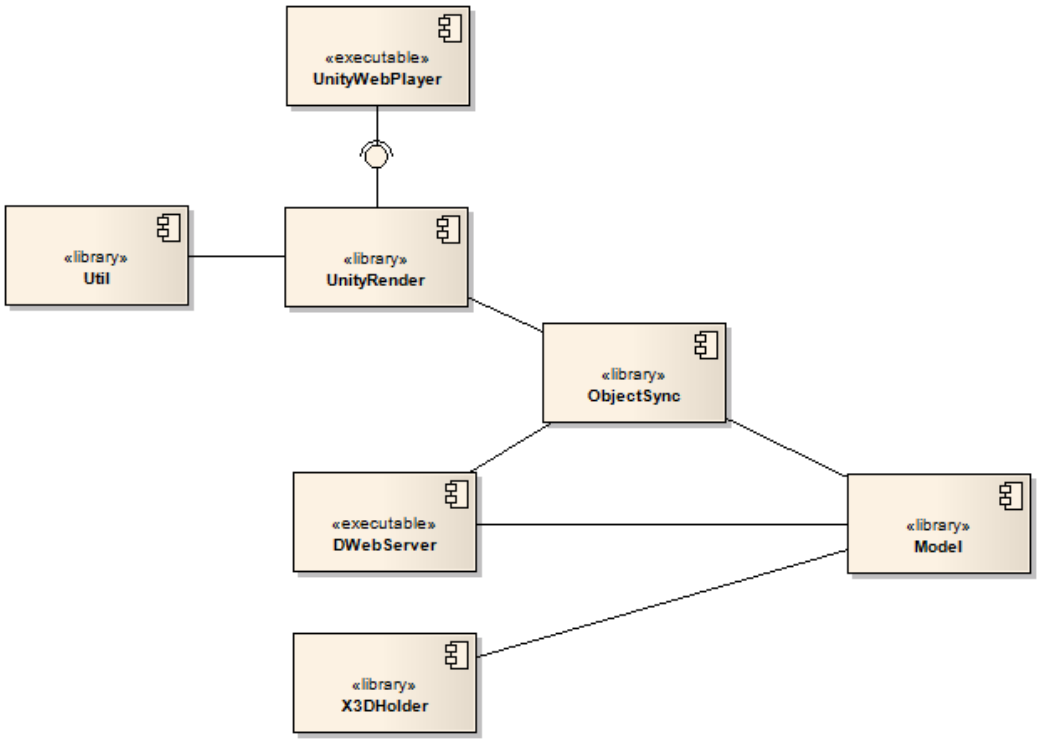

Figura 3.2: Modelo de componentes do DWeb3D.

3.3.1 Model

Este construto é a base do funcionamento do toolkit. Dentro dele está o grafo e todas as suas classes que representam a estrutura do X3D. Também nele estão as classes principais de leitura e escrita de arquivos X3D, como a classe X3DRepresentation ou o X3DHolder.

O X3DRepresentation é a principal classe de manipulação do grafo. Ele é o responsável pela leitura dos arquivos X3D e a escrita do grafo em formato XML. Já o X3DHolder é a classe responsável pela utilização da estrutura num ambiente asp.net.

O grafo em si é uma estrutura mais complexa e por isso será discutido mais a frente.

3.3.2 ObjectSync

Este é o pacote responsável pela sincronização de vários objetos do grafo através da rede. O modelo de classes do ObjectSync é apresentado na Figura 3.3. Para que o ObjectSync possa ser utilizado basta que o objeto alvo seja do tipo ObjectSync ou herde dele e que ele seja registrado no gerente de sincronismo. A interface define quais propriedades serão sincronizadas e

permite disparar o sincronismo a partir de uma única chamada de função.

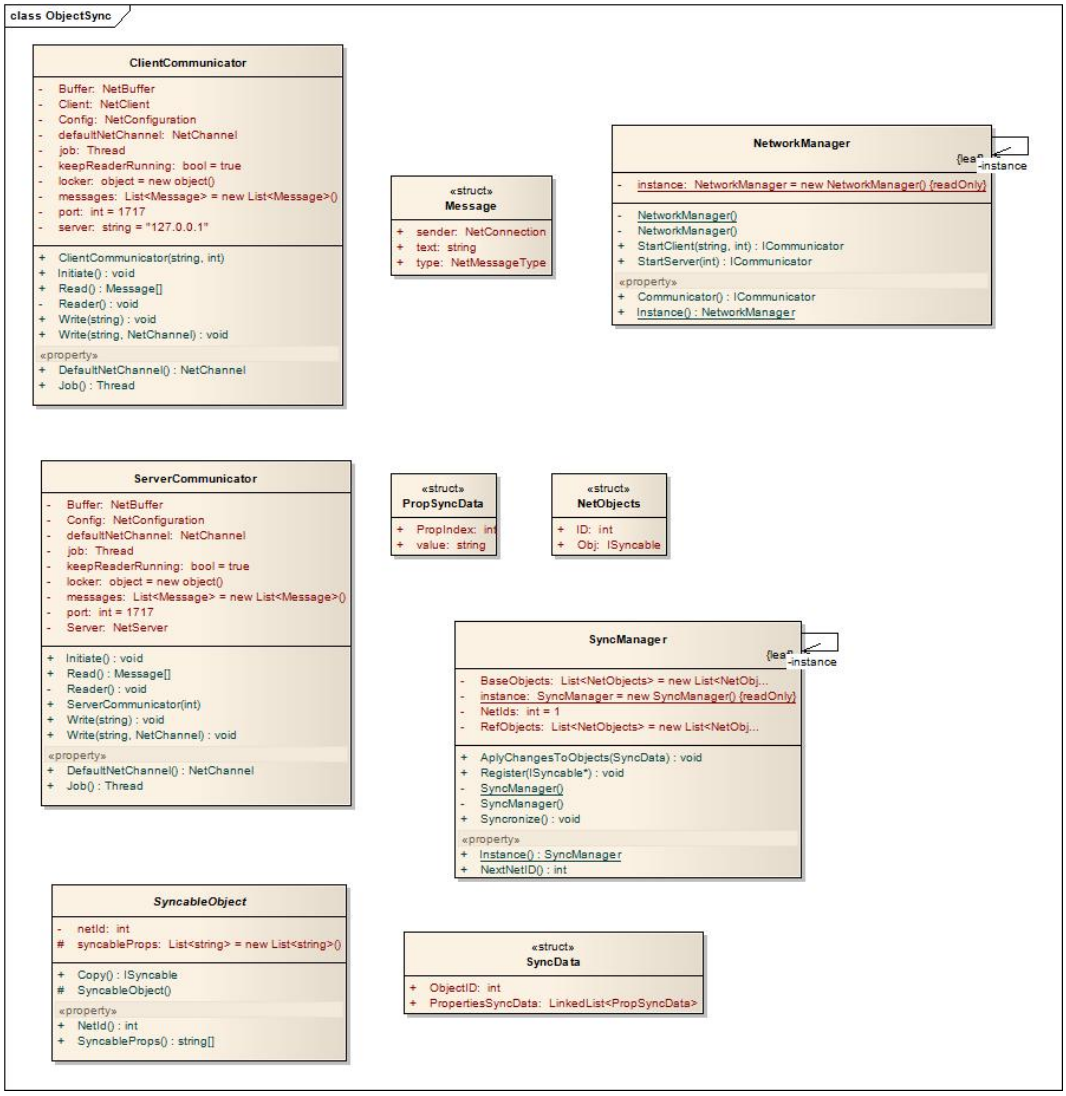

Figura 3.3: Modelo das classes do ObjectSync.

O importante desta biblioteca é que ela lida com toda a camada de rede e torna bastante simples a tarefa de fazer a sincronização.

3.3.3 DWebServer

Dll responsável pela criação de um servidor. Ela somente recebe as sincronizações e repassa para os outros clientes. A principal característica é que ele faz uso das classes contidas no ObjectSync e que por sua vez possuem capacidades de transmitir as sincronizações. Ele mantém uma cópia própria do grafo. Vale lembrar que a sincronização sempre é feita do cliente para o servidor.

3.3.4 Unity Render

É um plugin de redenderização (na forma de uma dll) que permite utilizar o grafo X3D dentro do ambiente da Unity3D. O que ele faz é criar objetos Unity através das respresentações do X3D e mantê-los sincronizados com os objetos do grafo.

3.3.5 Util

Não é utilizado diretamente por desenvolvedores externos. Na verdade este pacote contém código de auxílio interno do toolkit, como conversões e outros códigos reutilizados dentro do toolkit.

3.3.6 Unity WebPlayer

Este é um executável com os scripts desenvolvidos para o DWeb3D já colocados, de forma a servir de demonstração da interação do plugin Unity com o motor gráfico e de como colocar o plugin para funcionar. O WebPlayer poderá ser utilizado como um substituto de um browser X3D por aqueles que desejarem utilizar o plugin.

3.4 O Grafo X3D

Sendo o grafo a parte mais complexa do toolkit, ele merece uma seção para explorá-lo em separado.

O grafo segue uma estrutura hierárquica com um modelo de árvore (Figuras 3.4 e 3.5), sendo que sua raiz é sempre a classe X3DRepresentation. Esta classe contém os métodos de carga e renderização, além de uma lista de nós filhos. Os nós são objetos que herdam do tipo Node. A X3DRepresentation separa os nós do cabeçalho dos nós das cena por suas diferenças funcionais. Todos os nós de cena herdam do objeto SceneNode que possui suas caraterísticas básicas.

Algumas classes são chaves no funcionamento do grafo e merecem uma explicação especial.

3.4.1 SceneNode

Classe base das classes de cena, ela não é utilizada diretamente quando se for manipular o grafo, mas é ela que contém muitas das operações comuns aos

Simple Part 1.jpg

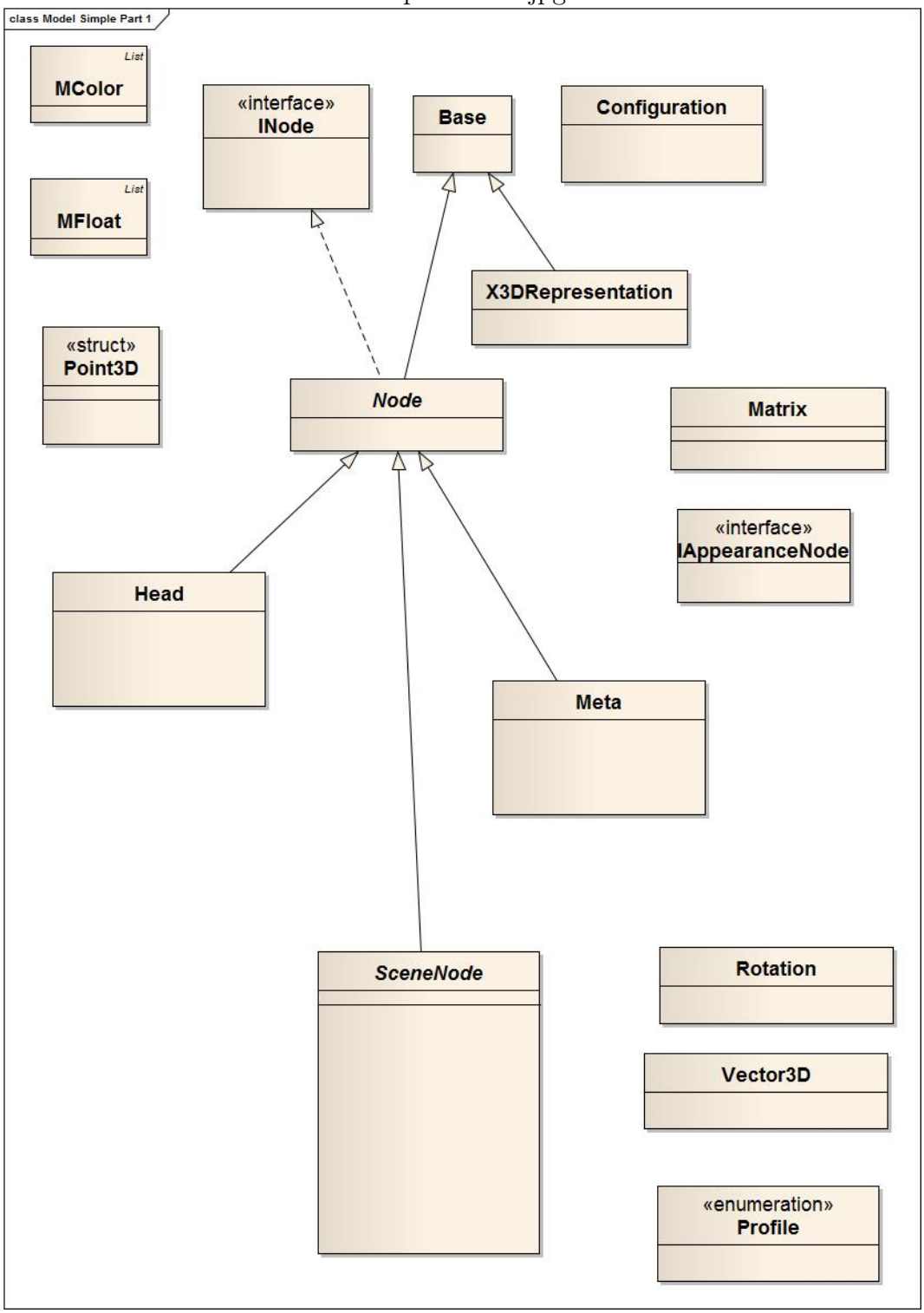

Figura 3.4: Modelo de domínio do Grafo X3D (I).

Simple Part 2.jpg

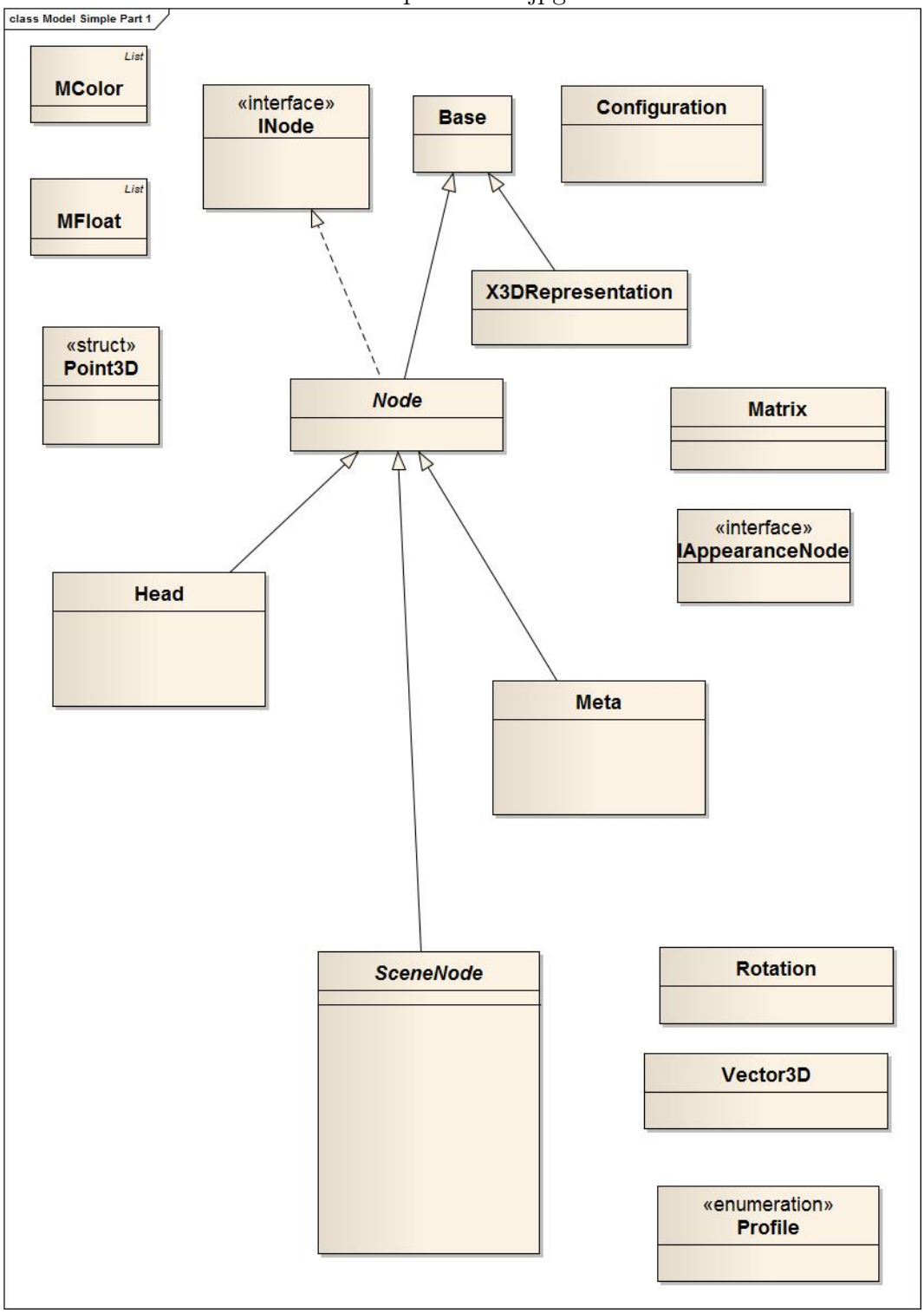

Figura 3.5: Modelo de domínio do Grafo X3D (II).

nós de cena. Por exemplo, buscar filhos, que é implementada como uma função recursiva que busca em profundidade na árvore de filhos. É nela também que está implementado o padrão delegate, ou seja, através dela é delegada a cada classe a responsabilidade por fazer as buscas em seus filhos e por fazer a sua renderização.

A utilização desta estrutura hierárquica possibilitou uma economia considerável de código e por conseguinte fica mais fácil mantê-lo ou estendê-lo.

3.4.2 Group

Esta classe serve para agrupar outros objetos; agrupar e nomear de forma que seja possível encontrar um objeto buscando pelo grupo ao qual ele pertence. A capacidade de nomear é bastante importante porque é assim que se permite uma melhor organização no X3D.

3.4.3 Transform

Trata-se de um grupo com capacidades especiais. Esta classe lida com toda e qualquer transformação necessária no mundo X3D. Por isso ela possui a capacidade de armazenar quaternions para rotações, vetores de translação e vetores de escala. O Util possui métodos para lidar com conversões de matrizes para vetores, de forma que é possível trabalhar também com matrizes. É importante ressaltar que todo objeto vai necessitar estar dentro de uma translação sozinho ou em grupo com outros, pois é a única forma de posicioná-lo na cena.

3.4.4 Appearance

Um appearance pode possuir vários shapes dentro dele. Isto pode parecer sintaticamente estranho, mas é a forma que o X3D organiza as coisas, portanto o Appearance agrupa formas, materiais e texturas, possibilitando definir uma aparência.

3.4.5 ViewPoint

Um ViewPoint determina uma posição estática de uma câmera. Uma cena pode ter vários viewpoints, mas precisa de pelo menos um para definir a posição inicial da câmera. No caso de vários existirem, o primeiro é tido como a viewpoint padrão.

3.4.6 Background

Define as configurações de fundo da cena assim como a iluminação básica desta.

3.5 Implementação das metas no DWeb3D

De forma a ilustrar a adequação do toolkit aos objetivos propostos, deve-se demonstrar como ele pode resolver os problemas apresentados e como sua estrutura foi pensada para estes casos.

3.5.1 Colaboração

Para se alcançar a meta de possibilitar a colaboração alguns fatores são importantes:

- Facilitar a criação de um servidor.
- Possibilitar a sincronização das posições dos objetos entre as visualizações de diversos clientes.
- Possibilitar a criação de novos objetos via interface do cliente.

Para se criar um servidor básico, o código necessário se resume a:

```
public void StartDefaultServer()
{
   // Criando o servidor na porta 1717
    SceneServer s = new SceneServer(1717);
    // Definindo e criando a cena básica que este servidor
    // vai trabalhar
    s.SceneGraph = new X3DRepresentation();
    s.SceneGraph.CreateBasicScene();
    // Definindo que este é o servidor padrão
    // (ele prevê a existência de outros)
    s.IsDefault = true;
    // Adicionando o servidor ao gestor de cenas.
    DefaultServer = s;
    SceneServers.Add(s);
}
```

Pode-se notar que se tratam de quatro operações. Cria-se o objeto, diz-se qual a cena padrão dele (por ser uma cena vazia), define-se que ele é o servidor padrão (podemos ter vários na mesma máquina) e registra-se o servidor no gerente que o vai organizar.

As diversas funções de baixo nível como criar socket, tratar de threads, e criar loops de leitura necessários para um servidor estão englobadas no DWeb3D e, portanto, o desenvolvedor não precisa lidar com elas. Desta forma o tamanho e a complexidade do código produzido são em muito reduzidos.

Para se lidar com a sincronização foi criado o módulo ObjetSync. Para utilizá-los basta que os objetos a serem sincronizados implementem uma interface específica. Esta interface permite definir quais campos queremos que sejam sincronizados no objeto e com isso podemos promover o sincronismo através de uma simples chamada de função como exemplificado na Figura 3.5.1.

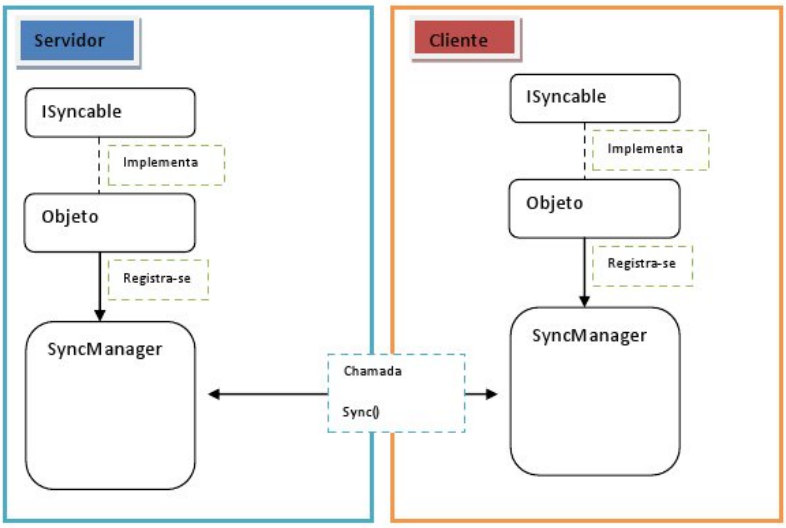

Figura 3.6: Figura exemplificando o processo de sincronização

Um pequeno trecho de código mostrando isto pode ser visto a seguir:

```
// Criação de um objeto transformável
var t = new Transform();
t.LoadFromReader(xmlReader);
```

```
// Definindo o objeto como um ISyncable (O transform é)
ISyncable tmp = t;
// Definindo um netIDválido
tmp.NetId = SyncManager.Instance.NextNetID;
// Registrando o objeto no gerente
SyncManager.Instance.Register(ref tmp );
```

Com pouco código obtém-se o registro do objeto. Porém para o caso dos objetos provenientes do grafo esse passo não é necessário pois eles já se registram na criação. Para sincronizar basta a seguinte linha:

```
SyncManager.Instance.Syncronize()
```

Se o código for proveniente do grafo o registro não é necessário. Mas isto pode ser utilizado para outras classes como por exemplo para disponibilizar um chat.

Apesar de o ObjectSync não lidar com a criação de novos objetos via interface do cliente, é factível criar um objeto que gerencie a criação de novos objetos através de uma propriedade sincronizável.

3.5.2 Persistência e carga

A persistência é feita também através dos mecanismos que permitem a colaboração, porém com um novo mecanismo, o X3DHolder. Este mecanismo utiliza um "truque"; como ele é o responsável pela exposição do objeto em um ambiente web, ele registra um cookie que identifica quem é o usuário. Ele também lê o cookie do usuário caso ele exista e, junto à classe UserManager, detecta quem é o usuário no servidor e permite que este recupere as características de sua última visita.

3.5.3 Interação com outras aplicações

Para que a aplicação X3D possa interagir com outras aplicações é necessário exisitr um mecanismo que permita expor os eventos ocorridos dentro da cena a outras aplicações. Para isto dois caminhos foram identificados:

1- Programar utilizando ECMAScript chamadas a mecanismos externos como webservices e programar nesses webservices a lógica desejada.

2- Programar um visualizador X3D que permita a adição de scripts a eventos do grafo (acesso a um nó, modificação de alguma propriedade, etc) ou até programação de gatilhos como clicar em um certo nó.

O DWeb3D decidiu seguir pelo segundo caminho e para isso foi desenvolvido o UnityRender. O que este módulo faz é converter o grafo X3D num grafo da Unity 3D de forma que seja possível publicar na web um visualizador (a Unity 3D possui essa ferramenta). Este visualizador pode receber mais plugins em formato de scripts em C# que podem vir ou do grafo ou de adições no projeto do visualizador. A Figura 3.7 ilustra o seu funcionamento.

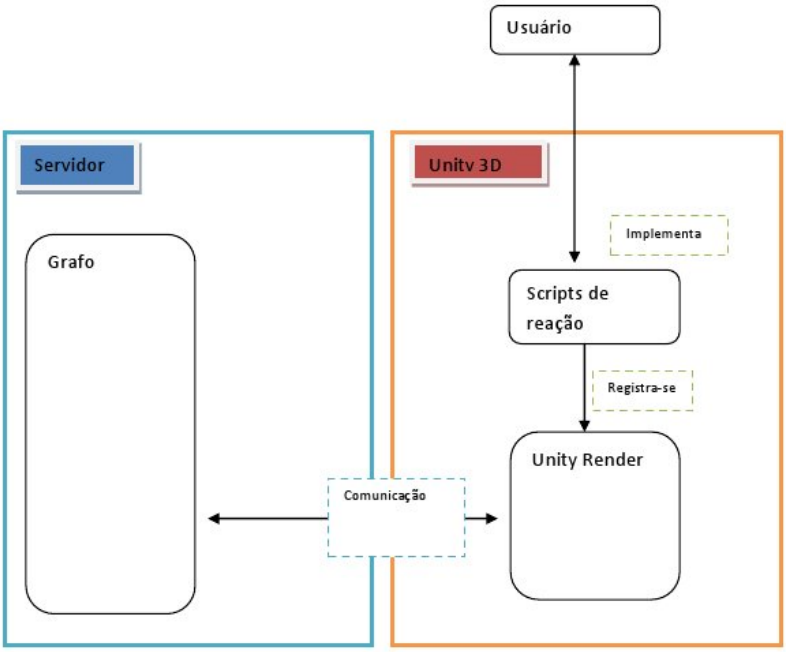

Figura 3.7: Esquema ilustrando o funcionamento do renderizador.

Com esse plugin a interação é alcançada, pois os scripts podem alterar o grafo, que por sua vez se sincroniza com o objetSync e altera o grafo (e outros objetos) no servidor e este por sua vez pode possuir código programado para interagir com ainda mais outras aplicações.

A Figura 3.8 ilustra o esquema de sincronismo proposto.

3.5.4 Interação com a GUI web

Apesar de também ser uma forma de interação com outras aplicações, o GUI web tem um papel especial por ser o ambiente onde a cena X3D se hospeda. Por isso ele possui uma forma especial de interação, que mesmo sendo parecida com a forma mencionada anteriormente utiliza, na parte web, um mecanismo especial e merece ser destrinchado a seguir.

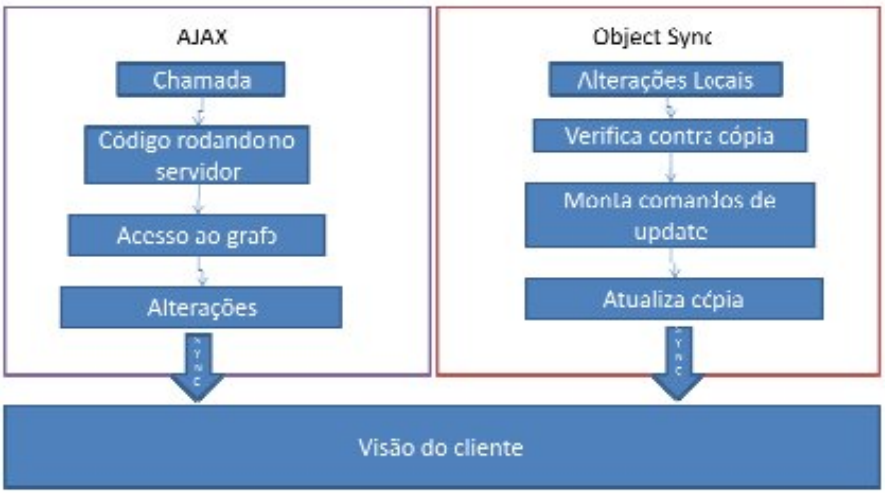

Figura 3.8: Esquema de sincronismo.

Atualmente um dos campos em maior expansão na web são as aplicações RIA[6] (Rou04, Par08), que são aplicações que possibilitam uma interface mais rica com o usuário. O progresso dos browsers e dos computadores pessoais tem permitido que muitas das aplicações antes feitas para rodar em desktop migrem para a web, e com essas aplicações os usuários passaram a demandar funcionalidades que são comuns a elas, como arrastar e soltar, seleção com o mouse, etc. As aplicações muito comumente alcançam esta meta através de aplicações de técnicas Ajax[7] (Rio08, Pud07, Dah08). Assim sendo, também o DWeb3D vai fazer uso de técnicas Ajax para alcançar uma interação rica com a aplicação 3D.

Ajax é o uso metodológico de tecnologias como JavaScript e XML, providas por navegadores, para tornar páginas mais interativas com o usuário, utilizando-se de solicitações assíncronas de informações. Ajax não é somente um novo modelo, é também uma iniciativa na construção de aplicações web mais dinâmicas e criativas. Ajax não é uma única tecnologia, são realmente várias tecnologias conhecidas trabalhando juntas, cada uma fazendo sua parte, oferecendo novas funcionalidades. Ajax incorpora em seu modelo:

- Apresentação baseada em padrões, usando XHTML e CSS.
- Exposição e interação dinâmica usando o DOM.
- Intercâmbio e manipulação de dados usando XML e XSLT.

[6]Rich Internet Application
[7]Asynchronous Javascript And XML

– Recuperação assíncrona de dados usando o objeto XMLHttpRequest.

– JavaScript unindo todas elas em conjunto.

Ajax3D trata de se aplicar a mesma ideia do Ajax tradicional para objetos X3D. Uma vez que o X3D pode receber scripts em ECMAScript e expõe suas características para o domínio do Browser, é possível através do ECMAScript e de técnicas de chamadas assíncronas interagir com o X3D. Com essa ideia foi desenvolvido o projeto Ajax3D que disponibiliza uma pequena biblioteca com as funções básicas para esta interação.

O funcionamento da interação Ajax / cena 3D pode ser observado na Figura 3.9. O importante é notar que sempre o servidor é atualizado, garantindo assim não só a interação com este usuário, mas também a alteração do grafo, que por sua vez possibilita a alteração da cena.

Figura 3.9: Ciclo de interação Ajax / cena 3d.

O papel do Ajax no DWeb3D é permitir que os componentes do GUI web se comuniquem de forma transparente com a cena. Se fosse necessário forçar um *refresh* a cena seria totalmente recarregada. Este tipo de comportamento tornaria impraticável a integração pois a cena deve ter um funcionamento contínuo e uniforme.

3.6 Testes executados durante o desenvolvimento

Durante o processo de desenvolvimento diversos testes foram escritos para verificar o funcionamento correto dos algoritmos propostos. Eles visavam verificar a conectividade, interação web, sincronismo e a geração e leitura do grafo. Os testes foram:

- Geração do grafo: Foi testada a criação de um grafo através da criação manual de classes seguindo a hierarquia de um arquivo XML e foi pedida a renderização do xml em seguida. Se o resultado fosse condizente com o arquivo de origem o teste estava ok.
- Leitura de um grafo: Foi pedido que a classe geradora lesse um arquivo XML e gerasse um grafo. Depois foi pedido a esse grafo que gerasse o XML. Se o resultado fosse condizente com o arquivo original o teste estaria ok.
- Conectividade: Foi gerado um servidor de exemplo e uma classe cliente. Em seguida de máquinas diferentes foram executadas várias instâncias do cliente verificando se o servidor acusava conexão.
- Sincronismo: Foi gerado um grafo padrão tanto num servidor quanto num cliente. O grafo foi registrado como sincronizável e em seguida várias características no grafo do cliente foram alteradas. O servidor ficava imprimindo em tela as alterações e os outros clientes conectados faziam o mesmo.
- Interação web: Foi gerado um cliente que mostrava um grafo com um cubo. Foi programada uma página .Net que alterava através de chamada ajax o tamanho do cubo. Ao clicar num botão da página o cubo deveria mudar de tamanho.

4
Casos de estudo

Como forma de demonstrar a utilidade do DWeb3D, foi desenvolvida uma aplicação que faz uso de suas funcionalidades em pequenos testes pontuais. Neste capítulo essas funcionalidades são demonstradas e explora-se como isto pode ser aplicado a um escopo mais amplo.

4.1
Aplicação de demonstração

Como forma de organizar os testes e as demonstrações foram criados diversos pequenos projetos de testes, cada um focado em uma funcionalidade específica. Estas funcionalidades foram pensadas de forma a exemplificar possíveis usos do DWeb3D.

O importante aqui é demonstrar que a complexidade dessas funcionalidades ficaram escondidas no código do toolkit.

4.1.1
Grafo de Cena

Por ser a base de todo o toolkit uma discussão mais aprofundada sobre o modelo do grafo de cena é importante. Esta discussão se coloca antes das outras por ser mais básica e ajudar a compreensão das demais.

As principais funções disponíveis no pacote Model, que encapsula o grafo e a estrutura do X3D são:

- Carga de um arquivo X3D.
- Exposição da estrutura da cena.
- Renderização da estrutura alterada novamente para um arquivo.

A carga é obtida através de uma função recursiva que lê cada item do arquivo XML do X3D e o transforma num nó equivalente, respeitando a hierarquia. O processo é feito de acordo com o descrito na Figura 4.1.

Para demonstrar o funcionamento da carga do modelo podemos observar o seguinte pedaço de código:

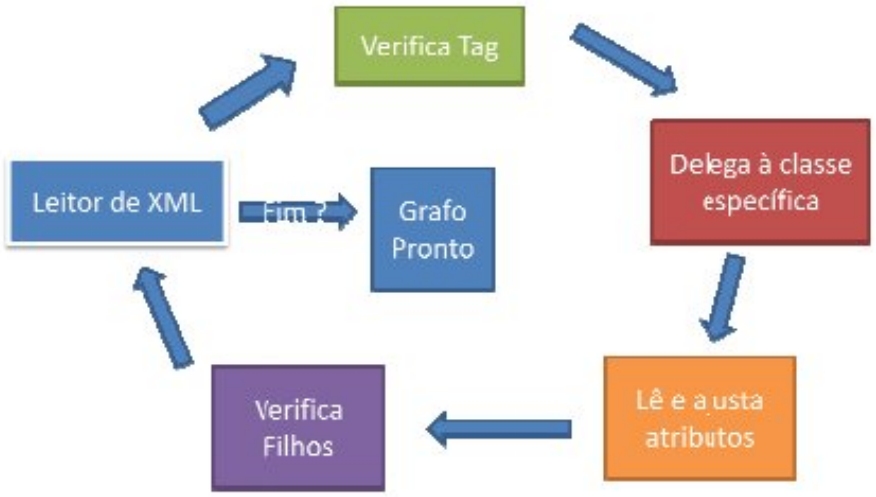

Figura 4.1: Processo de carga do modelo.

```
public static void  TestRead()
{
 // Criando o objeto para armazenar as classes
    X3DRepresentation xrep = new X3DRepresentation();
    // Carregando do arquivo
    xrep.LoadFromFile("..\\Samples\\BeckyRoadOverpass.x3d");
    // Renderizando para a tela
    Console.WriteLine(xrep.RendertoX3D());

}
```

Pode-se notar que o modelo encapsula todo o processo de carga e renderização necessário para lidar com um arquivo X3D em código .NET. No final temos um grafo hierárquico com todos os nós da cena. A renderização por sua vez é feita através de uma função recursiva em profundidade como ilustrada na Figura 4.2.

A Figura 4.3 ilustra o resultado da renderização feita pelo modelo. Ela serve como forma de mostrar a corretude da carga e da renderização.

4.1.2 Colaboração

Como feito nos capítulos anteriores, aqui também a organização se volta aos principais objetivos a serem alcançados com o DWeb3D. Para demostrar

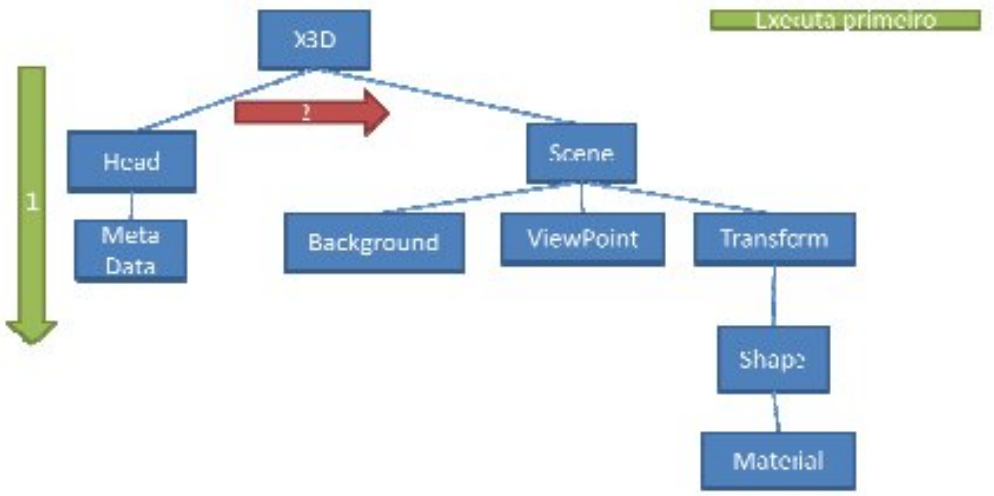

Figura 4.2: Ordem de renderização do grafo.

o quão simples é codificar uma aplicação que possa ser colaborativa foi desenvolvida uma demonstração, lembrando que o objetivo não é desenvolver uma aplicação completa, mas mostrar que é possível cortar etapas e agilizar o processo de desenvolvimento através do uso do toolkit. O conceito geral é ter algo como o ilustrado na Figura 4.4.

A primeira tarefa a ser feita é definir um servidor. Ele deve ser capaz de aceitar vários usuários e sincronizar as mensagens provenientes deles. Para isto foi definida uma cena simples que será carregada no servidor e nos dois clientes. Em seguida um dos clientes irá modificar um objeto e sincronizar. Espera-se que o segundo cliente seja capaz de ver as modificações.

O código para o servidor é o mesmo do exemplo no Capítulo 4. Para os clientes o código se parece um pouco com o do servidor e é descrito a seguir:

```
public void StartClient()
{
  // Criando o cliente na porta 1717
  Client c = new Client(1717, "127.0.0.1"');
  // Definindo e criando a cena básica que este cliente
  // vai trabalhar
  c.SceneGraph = new X3DRepresentation();
  c.SceneGraph.CreateBasicScene();
  // registrando o cliente
```

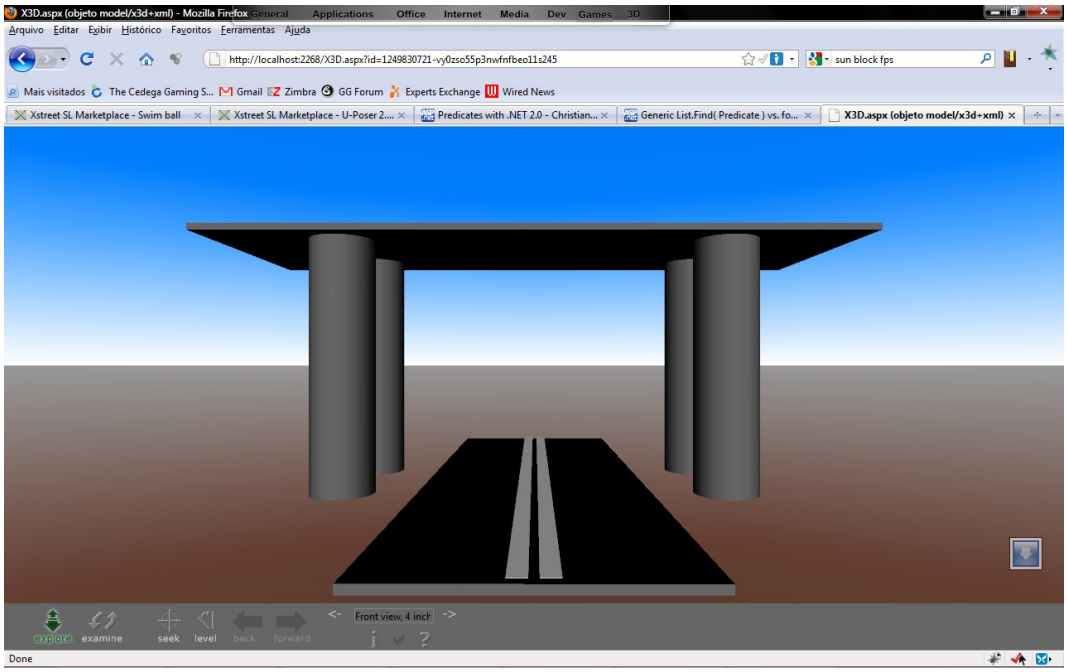

Figura 4.3: Renderização com o Flux 3D do resultado do modelo.

```
  ClientManager.Add(c);
}
```

Para o cliente a principal diferença é que ele precisa saber quem é o seu servidor. Uma vez conectados, cliente e servidor ficam quase invisíveis na aplicação. Desta forma, basta fazer as alterações necessárias e sincronizar. O único detalhe que deve ser notado é que a sincronização normalmente parte do cliente.

Como estamos comparando a eficiência do DWeb3D, mostra-se a seguir como seria o código de criação de um servidor (somente a parte de rede) sem o toolkit. Como pode-se notar ele é mais complexo.

```
public void Initiate()
{
    // Cria uma configuração para o servidor
    Config = new NetConfiguration("ObjectSync");
    Config.MaxConnections = 128;
    Config.Port = port;

    // Cria o servidor e ouve as conexões
    Server = new NetServer(Config);
    Server.SetMessageTypeEnabled
     (NetMessageType.ConnectionApproval, true);
    Server.Start();
```

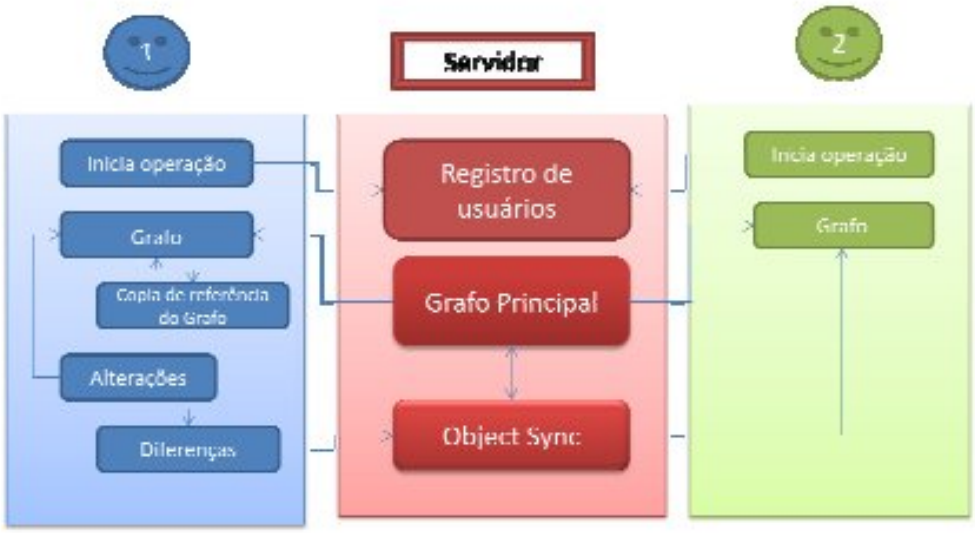

Figura 4.4: Modelo de dois usuários visualizando uma cena.

```
        //Define o canal padrão
        DefaultNetChannel = NetChannel.ReliableInOrder1;

        // Cria um buffer
        Buffer = Server.CreateBuffer();

        // Cria uma thread
        ThreadStart job = new ThreadStart(Reader);
        this.Job = new Thread(job);
        this.Job.Start();

    }
```

O código acima somente cuida da inicialização do servidor. No exemplo usando o DWeb3D, existem classes que gerenciam mais de um servidor, e gerenciam a troca de mensagens, além de cuidar de fazê-las transparente. O código equivalente sem o uso do toolkit seria portanto ainda mais complexo que o trecho acima listado.

Para finalizar, foi criado um objeto fora do grafo para que se pudesse simular uma operação de conversa entre dois usuários, e o código a seguir mostra como isso funciona.

```
// Criação de um objeto chat
var chat = new Chat();

// Definindo o objeto como um ISyncable
// (ele tem que implementar a interface)
ISyncable tmp = chat;
// Definindo um netIDválido
tmp.NetId = SyncManager.Instance.NextNetID;
// Registrando o objeto no gerente
SyncManager.Instance.Register(ref tmp );

// Eviando uma mensagem
chat.EnviarMensagem("Olá");
```

Neste código a função enviar mensagem seta uma variável e depois sincroniza. O servidor replica essa mensagem que os outros clientes poderão ler.

Para ilustrar como essa troca de mensagens seria feita sem o DWeb3D no Apêndice C está um pequeno trecho de código, responsável pelo envio e recebimento de mensagens de uma aplicação de chat.

Verifica-se que o código que têm aproximadamente 50 linhas é bastante extenso e mais complexo de ser entendido que o gerado com utilizando o DWeb3D.

4.1.3 Persistência e carga

Para se obter a persistência basta que se utilize o X3DHolder como forma de expor o código para o visualizador. Como descrito no capítulo anterior, ele será o responsável pelo registro e recuperação do cookie, e nesse aspecto não há muito o que se demonstrar, a não ser como expor o X3DHolder. Isto é feito através do pequeno pedaço de código.

```
<%@ Register NameSpace="DWeb3D.X3DHolder"
TagPrefix="X3DH" Assembly="X3DHolder"  %>

<X3DH:X3DHolder ID="XHolder" runat="server" WIDTH="300" HEIGHT="300" >
</X3DH:X3DHolder>
```

Neste aspecto a aplicação de demonstração ilustra como utilizar o segundo nível de persistência, que é a persistência em disco. Para isto, o servidor ganhou um método Save() como visto no código abaixo:

```
public void StartDefaultServer()
{
   // Criando o servidor na porta 1717
    SceneServer s = new SceneServer(1717);
    // Definindo e criando a cena básica que este servidor
    // vai trabalhar
    s.SceneGraph = new X3DRepresentation();
    s.SceneGraph.CreateBasicScene();
    // Salvando para o disco
    s.Save();
}
```

O código acima cria um servidor, define uma cena padrão e chama o método para salvá-la. Este método vai verificar a existência de uma base de dados (arquivo) padrão no disco, caso ela não exista será criada. Após isto será chamada a biblioteca db4o (Gre05) que se encarregará de lidar com as complexidades de armazenamento de objetos em disco.

Sem o uso do toolkit seria necessário lidar com a inicialização do db4o, verificação de existência da base e configurações da biblioteca, além de ter de buscar o nó correto no grafo, para que o salvamento possa ser feito com sucesso.

4.1.4 Interação com outras aplicações

Um exemplo de como podemos fazer a aplicação interagir com outras é o servidor, pois através dele a cena X3D está interagindo com o código .NET num servidor que por si só já é outra aplicação. Porém, um outro exemplo foi desenvolvido na forma do Unity3D Render. Trata-se de código que pode ser chamado no motor gráfico e assim o grafo é sincronizado lá. Desta forma podemos adicionar comportamentos específicos através de scripts atrelados aos nós na Unity3D. O exemplo abaixo demonstra como se utilizar o script.

```
public class Creator : MonoBehaviour {

// Criando a representação
X3DRepresentation xrep = new X3DRepresentation();
```

```
void Start () {

  // Carregando o arquivo
  xrep.LoadFromFile("..\\Exemplo.x3d");

  // Inicializando o renderizador
  RenderManager.Instance.InitializeGraphsRender(xrep.Scene);
  xrep.Scene.RenderToScreen();

}

// Chamado uma vez por frame.
void Update () {
xrep.Scene.RenderToScreen();
}
}
```

Para demostrar o funcionamento do plugin, uma cena bastante simples será utilizada. Trata-se de uma cena com somente um cilindro, um cubo, um viewport e algumas transformações. Sua descrição pode ser encontrada no Apêndice B, sua imagem renderizada com o Flux 3D[1] pode ser vista na Figura 4.5.

Com a implementação feita podemos acompanhar o processo de renderização da Unity3D nas Figuras 4.6 e 4.7. Nelas podemos ver o antes e o depois da renderização.

Na Figura 4.7 pode-se notar que os nós criados são selecionáveis e que eles apresentam as características dos nós da cena X3D. O processo que ocorreu foi uma conversão do modelo do grafo X3D com o grafo da Unity. Finalizando, outra característica interessante do motor gráfico é que ele permite atrelar scripts a nós. O próprio script de criação visto anteriormente é atrelado a um nó especial chamado criador e utilizando esta técnica é possível adicionar scripts que respondam a eventos específicos do nó como um click ou um usuário passar pelo nó.

Para se obter este mesmo resultado sem a utilização do DWeb3D seria necessário escrever um método de conversão do X3D no grafo da Unity. Este método teria que conter: um leitor de xml, um interpretador, uma função que transformasse essa estrutura em algo que o programa pudesse entender e uma

[1]Visualizador X3D desenvolvido pela Media Machines. http://www.mediamachines.com
.

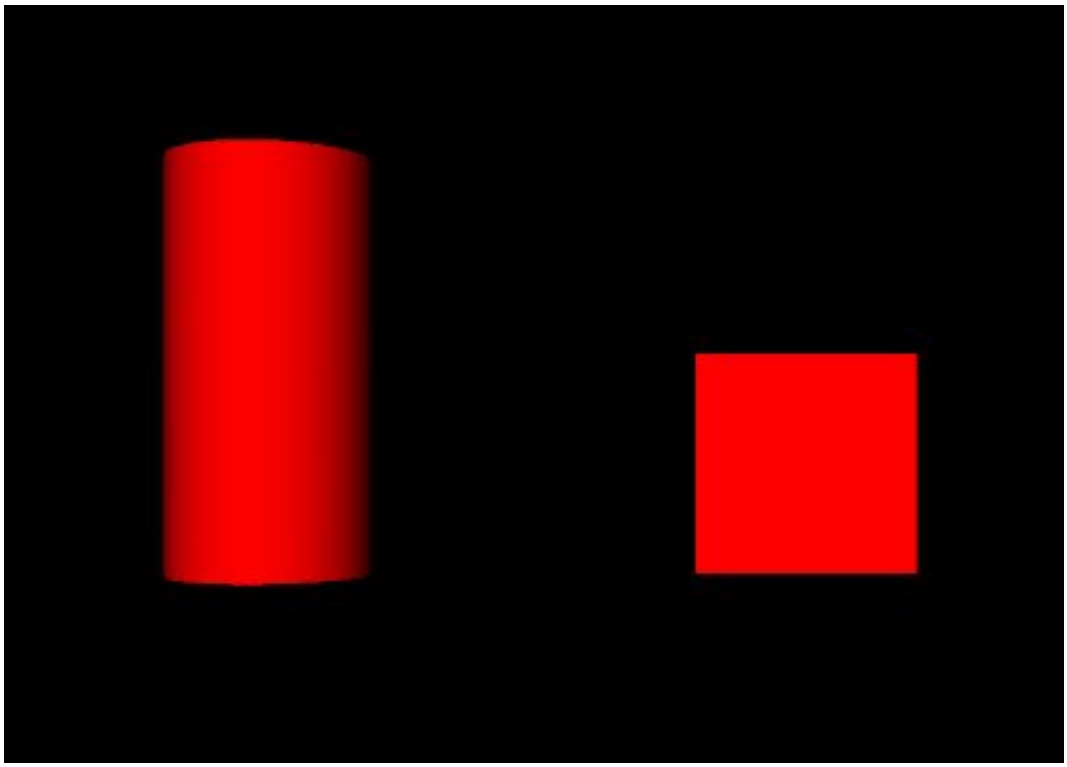

Figura 4.5: Imagem da cena básica renderizada pelo Flux.

função que transformasse a estrutura que o progama entende na estrutura da Unity. Por todo este processo ser muito extenso, será exemplificada a seguir apenas a função que faz uma parte desse processo, que é a conversão de um cubo. Lembre-se que este é apenas uma dentre muitas geometrias que teriam que ser convertidas, e que a conversão não passa só por geometrias, mas também por materiais, transformações, etc.

O trecho no Apêndice D é apenas uma pequena parte do código utilizado para a renderização. Para se refazer o trabalho do toolkit seriam necessárias muito mais linhas de código e uma complexidade bem grande.

O DWeb3D abordou o problema da interação com outras aplicações focando na integração com o Unity3D. Para uma possível interação com um outro motor gráfico, o DWeb3D implementa um esquema de plugins de renderização. Isto significa que, bastando rescrever o código de renderização, é possível aproveitar todos os outros mecanismos do toolkit como carga, manipulação do grafo, etc.

4.1.5 Interação com o GUI Web

O GUI web não deixa de ser uma outra aplicação, então neste caso estamos lidando com um caso mais específico de integração. Porém, sendo o GUI web o ambiente que irá hospedar a cena 3D, a integração é ainda mais

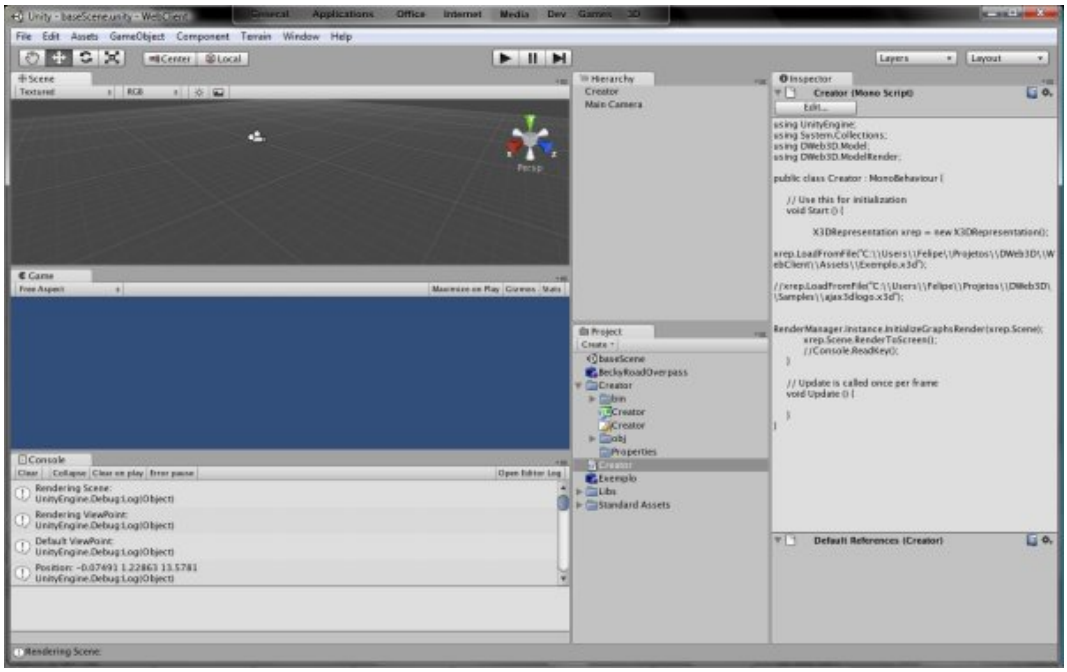

Figura 4.6: A Unity3D antes de iniciar a renderização.

importante. Sua integração pode ser feita via chamadas Ajax comuns no .Net que executam código no servidor. Este código estará no mesmo ambiente do servidor X3D, e através do gerente SceneServers pode ter acesso ao grafo do servidor e alterá-lo afetando assim a cena.

Para exemplificar esta possibilidade vamos verificar o seguinte código:

```
using System;
using DWeb3D.Model;

namespace TestWebApp
{
    public partial class _Default : System.Web.UI.Page
    {
       // Criando o objeto de armazenamento
        private X3DRepresentation x3dm;

        protected void Page_Load(object sender, EventArgs e)
        {
            x3dm = new X3DRepresentation();
            x3dm.LoadFromFile("..\\Samples\\Exemplo.x3d");
            XHolder.Representation = x3dm;
        }
```

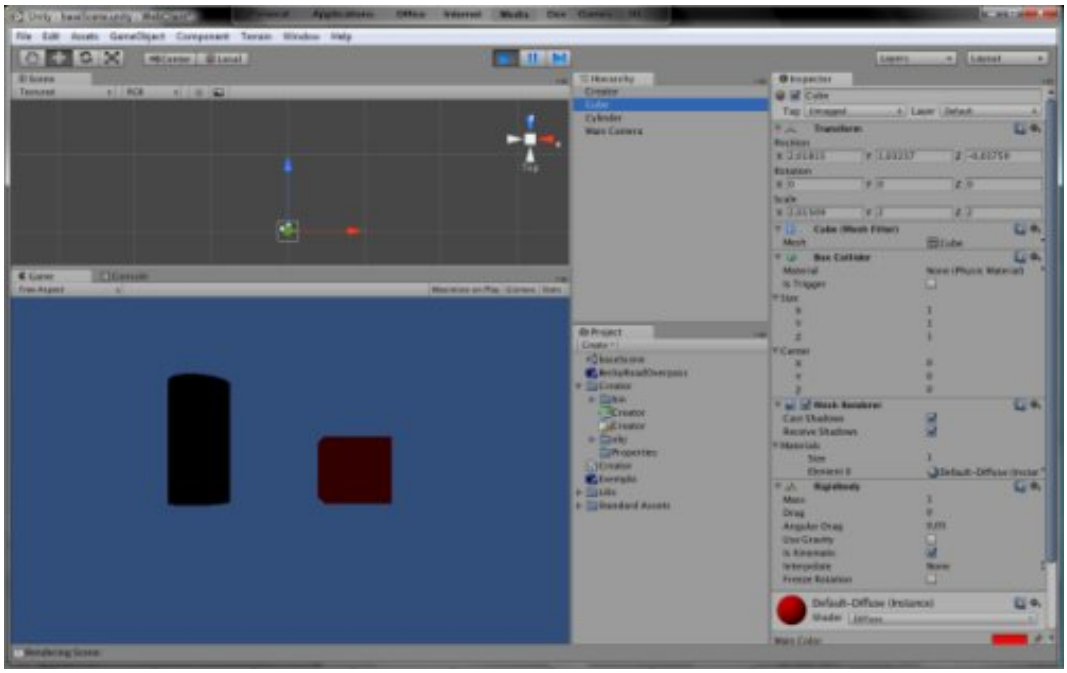

Figura 4.7: A Unity 3D depois de renderizado.

```
// Código chamado pelo botão para aumentar o box
        protected void BtAumentarCaixa_Click(object sender, EventArgs e)
        {
            ((Box)XHolder.Representation.Scene.Nodes[1]).Size
             += new Vector3D(2, 2, 2);
        }
    }
}
```

Quando o usuário clica no botão aumentar caixa o código vai no local de armazenagem (sem refresh devido ao Ajax) e soma 2 em todos valores. O antes e o depois dessa interação podem ser vistos nas Figuras 4.8 e 4.9 respectivamente.

Para se obter os mesmos resultados sem o DWeb3D, a forma mais simples seria desenvolver um código em ECMAScript que via API de browser (que muda de acordo com o cliente renderizando a cena) acessasse o nó e fizesse a modificação. O problema é que ECMAScript é complexo de depurar e os browsers mudam um pouco a API de acordo com a implementação.

4.2 Análise dos resultados obtidos

Pudemos verificar nos exemplos deste capítulo como a utilização do DWeb3D reduz a quantidade de código necessário para se obter alguns re-

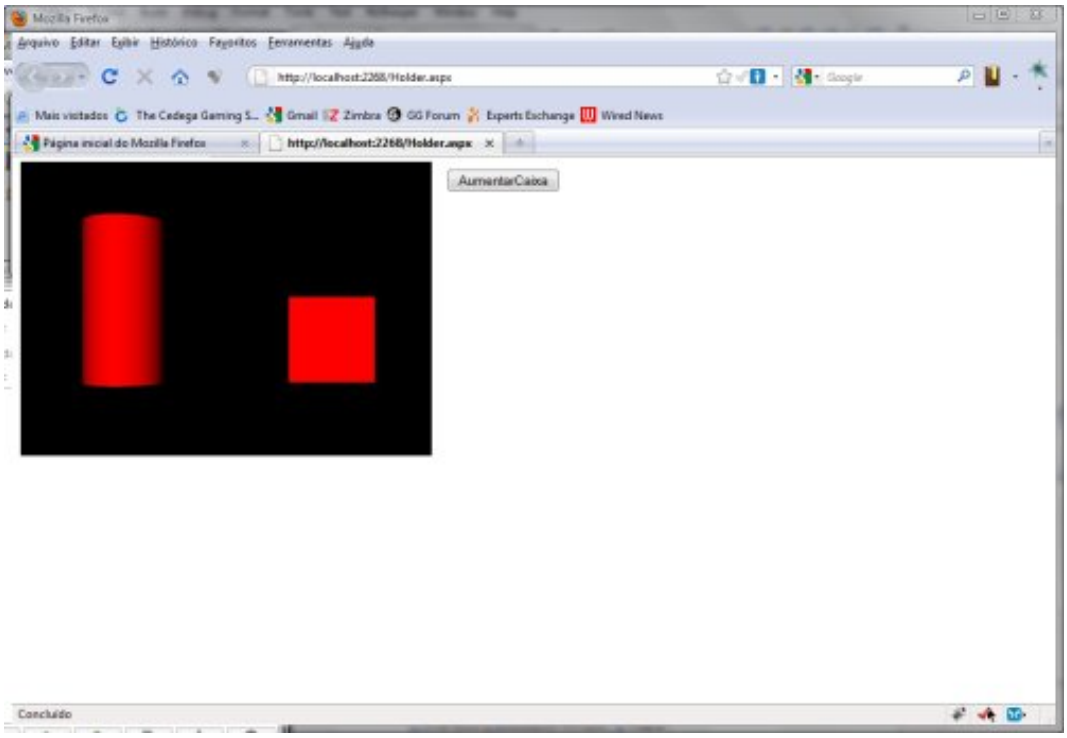

Figura 4.8: Figura simples antes do clique.

sultados. Esta redução de código não só facilita o processo de desenvolvimento inicial, mas também traz uma série de vantagens para aqueles que precisam desenvolver estes códigos. Por terem um custo menor no desenvolvimento, o risco das aplicações diminui. Assim sendo, o desenvolvedor pode arriscar mais pois o custo de um erro será menor. Outro aspecto positivo é a possibilidade de liberar os desenvolvedores a se focar nos objetivos de suas aplicações. Isto ocorre porque se os desenvolvedores não precisarem lidar com as camadas básicas, que são trabalhosas e complexas, ele poderá se focar nas funcionalidades desejadas em seu software.

A aplicação de testes nos mostrou, mesmo que de forma empírica, que este toolkit pode facilitar o processo de desenvolvimento de aplicações que tenham como requisito uma das funcionalidades propostas.

O DWeb3D pode se apresentar como uma ferramenta útil para facilitar o desenvolvimento de aplicações X3D complexas e se adequa a princípios citados anteriormente, onde um toolkit como estes, ao facilitar a vida do desenvolvedor, facilita os processos de desenvolvimento de forma generalizada e assim promove a utilização da tecnologia.

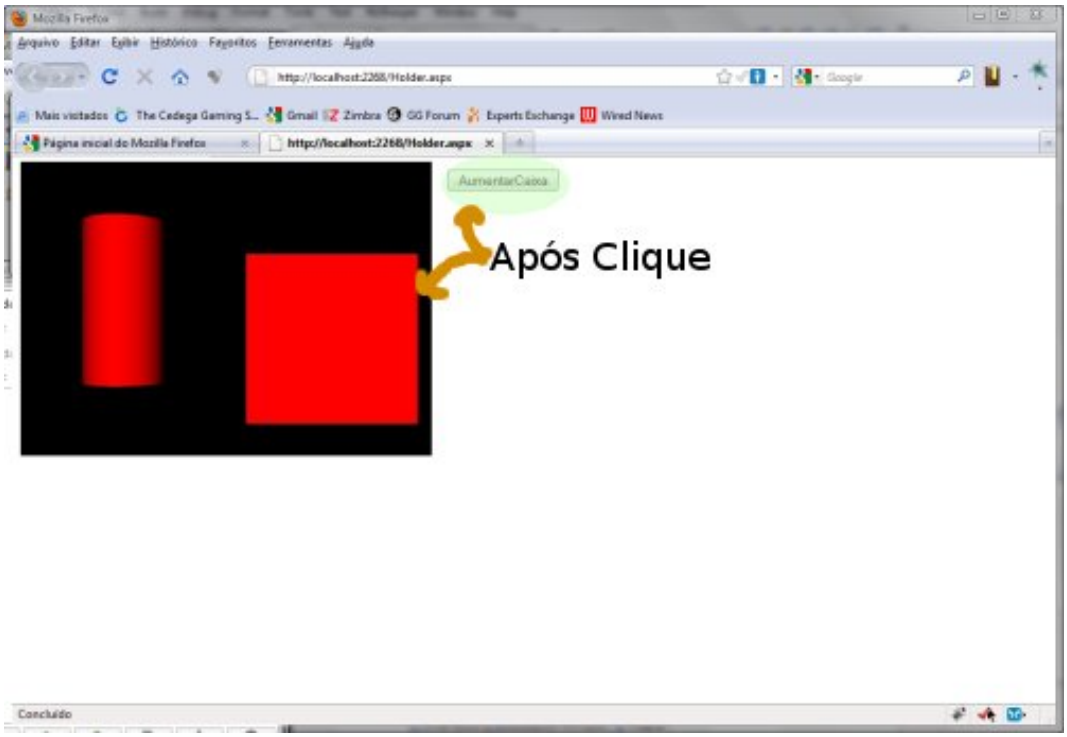

Figura 4.9: Figura simples após o clique

5
Conclusões

Esta dissertação apresentou uma proposta de toolkit para agilizar o desenvolvimento de aplicações X3D. Foram abordados alguns pontos fracos detectados no X3D, especialmente na comparação direta com outras tecnologias 3D na web. Os aspectos abordados foram: colaboração, persistência, interação com outras aplicações e interação com o GUI web.

Para a colaboração foi desenvolvido um dispositivo de cliente / servidor que possibilita a troca de mensagens e sincronização da cena. Para persistência foi criado um *server control* que gerencia os cookies e os usuários, possibilitando a persistência da posição da câmera entre diversos acessos dos usuários. Ainda em persistência disponibiliza-se um mecanismo para persistência em disco da cena. Para a interação com outras aplicações reutiliza-se o mecanismo de troca de mensagens de forma que seja possível atribuir respostas programáticas a eventos da cena. Finalmente, para interação com o GUI web, foram utilizados métodos de programação Ajax do próprio .NET, que permitem acesso ao grafo da cena.

O desenvolvimento do toolkit foi motivado pela ideia de que ao facilitar a vida dos programadores, descomplicando o processo de desenvolvimento, pode-se acelerar a aceitação de uma tecnologia. Porém, não é possível afirmar que um toolkit como o DWeb3D vai mudar sozinho a realidade do X3D, e este trabalho não se propõe a isto. De fato seria interessante ter resposta de outros desenvolvedores que pudessem testar e avaliar a ferramenta, o que não foi possível dentro do escopo deste estudo. Outro ponto importante na avaliação do DWeb3D seria o desenvolvimento de uma aplicação mais complexa, de uso real.

Um ponto que poderia ser melhorado no futuro é adicionar ao DWeb3D a possibilidade de comunicação síncrona. Atualmente o toolkit só trabalha com comunicação assíncrona, o que obriga o cliente fazer pedidos de sincronismos periódicos, mesmo que ele não tenha alterado nada. Ainda no âmbito da comunicação talvez seja interessante um mecanismo encapsulado no toolkit que permita a transferência de objetos completos via rede. Hoje ele sincroniza propriedades de objetos, mas é necessário haver uma cópia dele no cliente

e no servidor. Para atendermos esta demanda com as funcionalidades hoje implementadas foi criado um processo onde um objeto gerente recebe uma mensagem que indica qual objeto ele deve criar. Isto funciona, mas pressupõe que o gerente conhece todos os tipos de objetos necessários.

Ainda como trabalho futuro, pretende-se desenvolver um site para publicar o toolkit. Se ele deve ser útil para a comunidade, ele deve estar disponível de forma gratuita na internet e preferencialmente ser OpenSource. Ainda dentro desse espírito deve-se desenvolver e publicar uma documentação de uso completa que permita a outros desenvolvedores entender o DWeb3D em detalhes. Também pretende-se estender a aplicação de demonstração criando um pequeno mundo virtual visitável que possa servir de exemplo para outros desenvolvedores.

Dentro desta linha de pesquisa seria interessante um estudo sobre as aplicabilidades de aplicações 3D interativas e seu pontencial de impacto na forma como a interação ocorre na web. Outro estudo interessante seriam formas de codificar informação de cenas 3D e interações de forma a otimizar a experiência online que de acordo com todas as suas limitações ainda é bem única.

Dentre as lições aprendidas com este trabalho, notei o quão difícil é achar dados confiáveis no mundo que é a internet e estatísticas consisas de utilização de qualquer coisa na web são raras. Nota-se também o porquê de muitas decisões de arquitetura quando se tem de lidar com o mundo tão diversificado como os browsers, visalizadores e computadores clientes. É que alguns tipos de interação são complicados, como por exemplo coisas que precisem de um tempo de resposta muito curto, pois as diversas camadas de aplicação existentes criam uma sobrecarga considerável.

O tipo de abordagem utilizada tem limitações claras ao definir a necessidade do plugin da Unity3D, pois outros visualizadores não serão capazes de utilizar o DWeb3D. Esta decisão se deu para simplificar o processo e permitir trabalhar com um ambiente um pouco mais controlado, pois os recursos de um projeto como este são bastante limitados.

Outro aspecto a ser explorado é que o DWeb3D se propõe a instrumentar o X3D para fazer funções que muitos defendem não ser sua vocação. Estes defendem ter o X3D um aspecto muito mais voltado para cenas não multi-usuário, para visualização de produtos, definição de cenas com interações simples; outras ferramentas seriam naturalmente mais interativas como a Unity3D e outros motores gráficos que aparecem no mercado com opções de exportar para web. É fato que ferramentas mais novas vêm melhor preparadas para as demandas correntes do mercado, porém não se deve ignorar as

qualidades de um padrão bem estabelecido como X3D e nem subestimar o que pode ser alcançado com ele. Além disso o DWeb3D ao se utilizar da Unity3D, agrega ao X3D muitas das qualidades da Unity3D.

Uma forma interessante de verificar as qualidades do toolkit seria compará-lo a outros browsers. Tendo em vista essa comparação escolhi um dos browsers mais conhecidos no mercado atualmente, o BSContact. Trata-se de um browser X3D completo que implementa a SAI e é utilizado em várias aplicações existentes. Apesar dele possuir uma versão de um servidor multi-usuário, a programação ainda tem de ser feita basicamente utilizando ECMAScript e o controle sobre o servidor é limitado, ele basicamente sincroniza os diversos clientes sobre uma cena porém esta pode ser muito pouco modificada. A grande vantagem da abordagem do DWeb3D é exatamente permitir a modificação da cena 3D.

Referências Bibliográficas

[All02] ALLAIRE, J.. **Macromedia Flash MX next-generation rich client**. Whitepaper, Adobe, 2002. http://www.adobe.com/devnet/flash/whitepapers/richclient.pdf. 1.1

[Bat08] BATES, C. D.; YATES, S.. **Scrum down: a software engineer and a sociologist explore the implementation of an agile method**. In: CHASE 2008: PROCEEDINGS OF THE 2008 INTERNATIONAL WORKSHOP ON COOPERATIVE AND HUMAN ASPECTS OF SOFTWARE ENGINEERING, p. 13–16, New York, NY, USA, 2008. ACM. 3.1.2

[Beh04] BEHR, J.; DÄHNE, P. ; ROTH, M.. **Utilizing X3D for immersive environments**. In: WEB3D '04: PROCEEDINGS OF THE NINTH INTERNATIONAL CONFERENCE ON 3D WEB TECHNOLOGY, p. 71–78, New York, NY, USA, 2004. ACM. 1.3

[Bos07] BOSS, G.; MALLADI, P.; QUAN, D.; LEGREGNI, L. ; HALL, H.. **Cloud Computing**. Whitepaper, High Performance On Demand Solutions (HiPODS), 2007. 2.1

[Bou07] BOURAS, C.; TEGOS, C.; TRIGLIANOS, V. ; TSIATSOS, T.. **X3D Multi-user Virtual Environment Platform for Collaborative Spatial Design**. In: ICDCSW '07: PROCEEDINGS OF THE 27TH INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS WORKSHOPS, p. 40, Washington, DC, USA, 2007. IEEE Computer Society. 1.3, 2.5.4, 2.8

[But07] BUTTUSSI, F.; CHITTARO, L. ; COPPO, M.. **Using Web3D technologies for visualization and search of signs in an international sign language dictionary**. In: WEB3D '07: PROCEEDINGS OF THE TWELFTH INTERNATIONAL CONFERENCE ON 3D WEB TECHNOLOGY, p. 61–70, New York, NY, USA, 2007. ACM. 1.2

[Buy09] BUYYA, R.; YEO, C. S.; VENUGOPAL, S.; BROBERG, J. ; BRANDIC, I.. **Cloud computing and emerging IT platforms: Vision, hype,**

and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst., 25(6):599–616, 2009. 2.1

[Dac03] DACHSELT, R.; RUKZIO, E.. **Behavior3D: an XML-based framework for 3D graphics behavior**. In: WEB3D '03: PROCEEDINGS OF THE EIGHTH INTERNATIONAL CONFERENCE ON 3D WEB TECHNOLOGY, p. 101–ff, New York, NY, USA, 2003. ACM. 1.3

[Dah08] DAHLAN, A. A.; NISHIMURA, T.. **Implementation of asynchronous predictive fetch to improve the performance of Ajax-enabled web applications**. In: IIWAS '08: PROCEEDINGS OF THE 10TH INTERNATIONAL CONFERENCE ON INFORMATION INTEGRATION AND WEB-BASED APPLICATIONS AND SERVICES, p. 345–350, New York, NY, USA, 2008. ACM. 3.5.4

[Deh04] DEHAAN, J.. **Flash MX 2004 - guia autorizado Macromedia.** Elsevier, 2004. 1

[Doi06] DOI, Y.; KAGAWA, K.. **An X3D generator plug-in for Eclipse in a Web-based Educational System for Programming.** In: PROCEEDINGS OF WORLD CONFERENCE ON EDUCATIONAL MULTIMEDIA, HYPERMEDIA AND TELECOMMUNICATIONS, p. 2523–2528, Chesapeake, VA: AACE, 2006. EDMEDIA. 1.1

[Dun03] DUNN, T. L.; WARDHANI, A.. **A 3D robot simulation for education**. In: GRAPHITE '03: PROCEEDINGS OF THE 1ST INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES IN AUSTRALASIA AND SOUTH EAST ASIA, p. 277–278, New York, NY, USA, 2003. ACM. 1.1

[Fri07] FRINCU, M. E.; PETCU, D.. **Remote Control for Graphic Applications**. In: SYMBOLIC AND NUMERIC ALGORITHMS FOR SCIENTIFIC COMPUTING, INTERNATIONAL SYMPOSIUM ON, p. 304–309, Los Alamitos, CA, USA, 2007. IEEE Computer Society. 1.3

[Gre05] GREHAN, R.. **Complex Object Structures, Persistence, and db4o**. Whitepaper, DBO, 2005. http://www.odbms.org/download/006.01/Grehan/Complex/Object/ Structures/May/2005.pdf. 1.3, 2.8, 4.1.3

[Gre07] GREENBERG, S.. **Toolkits and interface creativity**. Multimedia Tools Appl., 32(2):139–159, 2007. 2.2, 2.5.2, 3

[Hei02] HEINS, T.; HIMES, F.. **Creating Learning Objects With Macromedia Flash MX**. Whitepaper, Adobe, 2002. http://www.mystery-productions.com/hyper/flashmxlo.pdf. 1.1

[Hua09] HUANG, J.; CHENG, B.. **Interactive Visualization for 3D Pipelines Using Ajax3D**. In: NETWORKING AND DIGITAL SOCIETY, INTERNATIONAL CONFERENCE ON, p. 21–24, Los Alamitos, CA, USA, 2009. IEEE Computer Society. 1.3

[Jou08] JOURDAIN, S.; FOREST, J.; MOUTON, C.; NOUAILHAS, B.; MONIOT, G.; KOLB, F.; CHABRIDON, S.; SIMATIC, M.; ABID, Z. ; MALLET, L.. **ShareX3D, a scientific collaborative 3D viewer over HTTP**. In: WEB3D '08: PROCEEDINGS OF THE 13TH INTERNATIONAL SYMPOSIUM ON 3D WEB TECHNOLOGY, p. 35–41, New York, NY, USA, 2008. ACM. 2.5

[Kha89] KHANNA, A.; ZINKY, J.. **The revised ARPANET routing metric**. SIGCOMM Comput. Commun. Rev., 19(4):45–56, 1989. 2.1

[Liu07] LIU, S.; LI, J. ; WANG, X.. **Local Reputation for P2P MMOG Design**. In: PDCAT '07: PROCEEDINGS OF THE EIGHTH INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED COMPUTING, APPLICATIONS AND TECHNOLOGIES, p. 523–528, Washington, DC, USA, 2007. IEEE Computer Society. 2.5.1

[Lug05] LUGMAYR, A.; KALLI, S.. **Using Metadata-based SVG and X3D Graphics in Interactive TV** . Springer London, 2005. 1.3

[Par06] PARISI, T.. **Ajax3D: The Open Platform for Rich 3D Web Applications**. Whitepaper, Media Machines, Inc, 2006. 1.1, 1.3, 2.7.1

[Par08] PARK, Y. S.; LEE, J. H.; CHOI, H. R.; KIM, H. S.; JUNG, J. U. ; PARK, J. Y.. **Development of an RIA-based user interface for promotion of effectiveness in marine transportation**. In: ACS'08: PROCEEDINGS OF THE 8TH CONFERENCE ON APPLIED COMPUTER SCIENCE, p. 366–372, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS). 3.5.4

[Pud07] PUDER, A.. **A cross-language framework for developing AJAX applications**. In: PPPJ '07: PROCEEDINGS OF THE 5TH INTERNATIONAL SYMPOSIUM ON PRINCIPLES AND PRACTICE OF PROGRAMMING IN JAVA, p. 105–112, New York, NY, USA, 2007. ACM. 3.5.4

[Rik97] RIKK CAREY, G. B.. **Annotated VRML 97 Reference Manual**. DevPress, 1997. 1.1

[Rio08] RIORDAN, R.. **Head First Ajax**. O'Reilly Media, 2008. 3.5.4

[Rit06] TURKOWSKI, R.. **Web3D Consortium X3D Revision to add Physics, Particle Systems, UI Enhancements, Realistic Motion**. Whitepaper, Media Machines, Inc. Disponível em: `http://www.web3d.org/images/uploads/pdfs/Web3D_Consortium-X3D_Revision_1.pdf` Acesso em 30/12/2009, 2006. 1.2

[Rou04] O'ROURKE, C.. **A Look at Rich Internet Applications**. Oracle Magazine, 2004. 3.5.4

[San04] SANTOS, R. J. D.; BATTAIOLA, A. L. ; DUBIELA, R. P.. **Aspectos Fundamentais da Criação de jogos em Shockwave 3D**. WJogos / SBGames 2004, Simpósio Brasileiro de Jogos de Computador e Entretenimento Digital, 2004. 1.1

[Sch01] SCHWABER, K.. **Agile Software Development with SCRUM**. Prentice Hall, 2001. 3.1.2

[Tay05] TAY, V.. **Massively Multiplayer Online Game (MMOG) - A Proposed Approach for Military Application**. In: CW '05: PROCEEDINGS OF THE 2005 INTERNATIONAL CONFERENCE ON CYBERWORLDS, p. 396–400, Washington, DC, USA, 2005. IEEE Computer Society. 2.5.1

[Web97] CONSORTIUM, W.. **Virtual Reality Modeling Language**. Disponível em: `http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/` Acesso em: 10 julho 2009, 1997. 1.1

[Web07] WEBER, J. C.; PARISI, T.. **An Open Protocol for Wide-area Multi-user X3D**. Proceedings of the twelfth international conference on 3D Web technology, p. 133–136, 2007. 1.3, 2.5.5

[Web08] CONSORTIUM, W.. **Extensible 3D (X3D)**. Disponível em: `http://www.web3d.org/x3d/specifications/ISO-IEC-19775-1.2-X3D-AbstractSpecification/` Acesso em: 11 julho 2009, 2008. 1.1

A
Apêndice A - Estrutura do DWeb3D

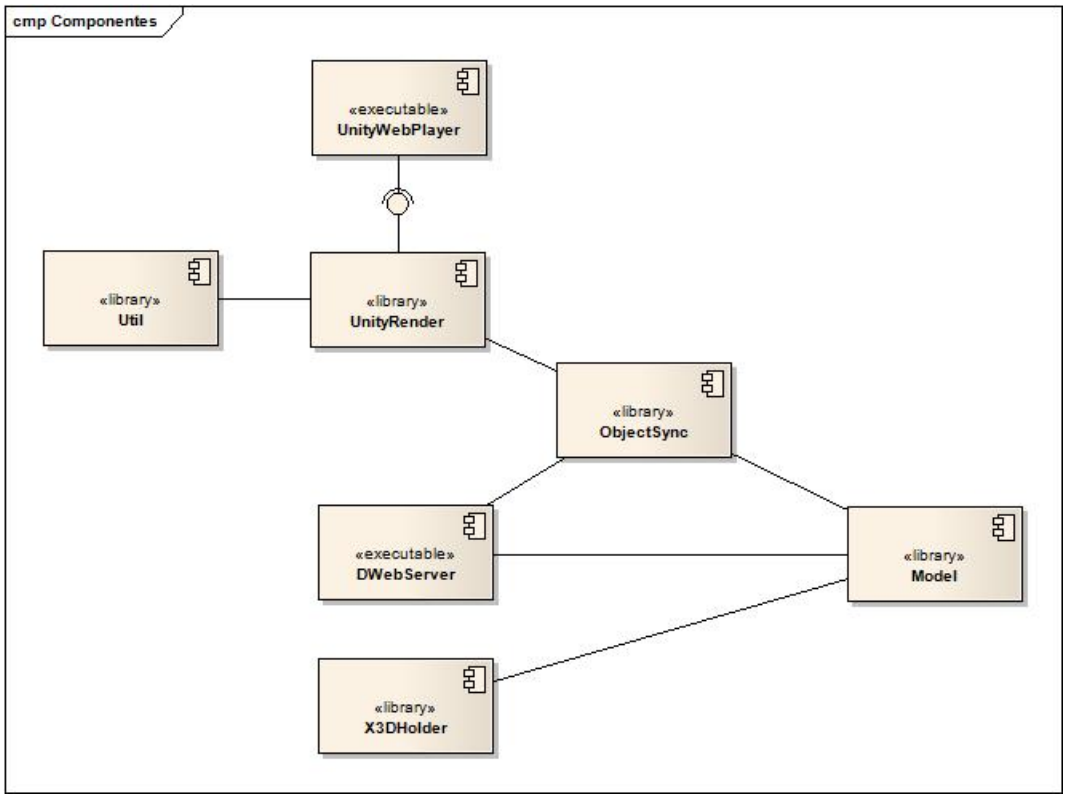

Figura A.1: Componentes.

Simple Part 1.jpg

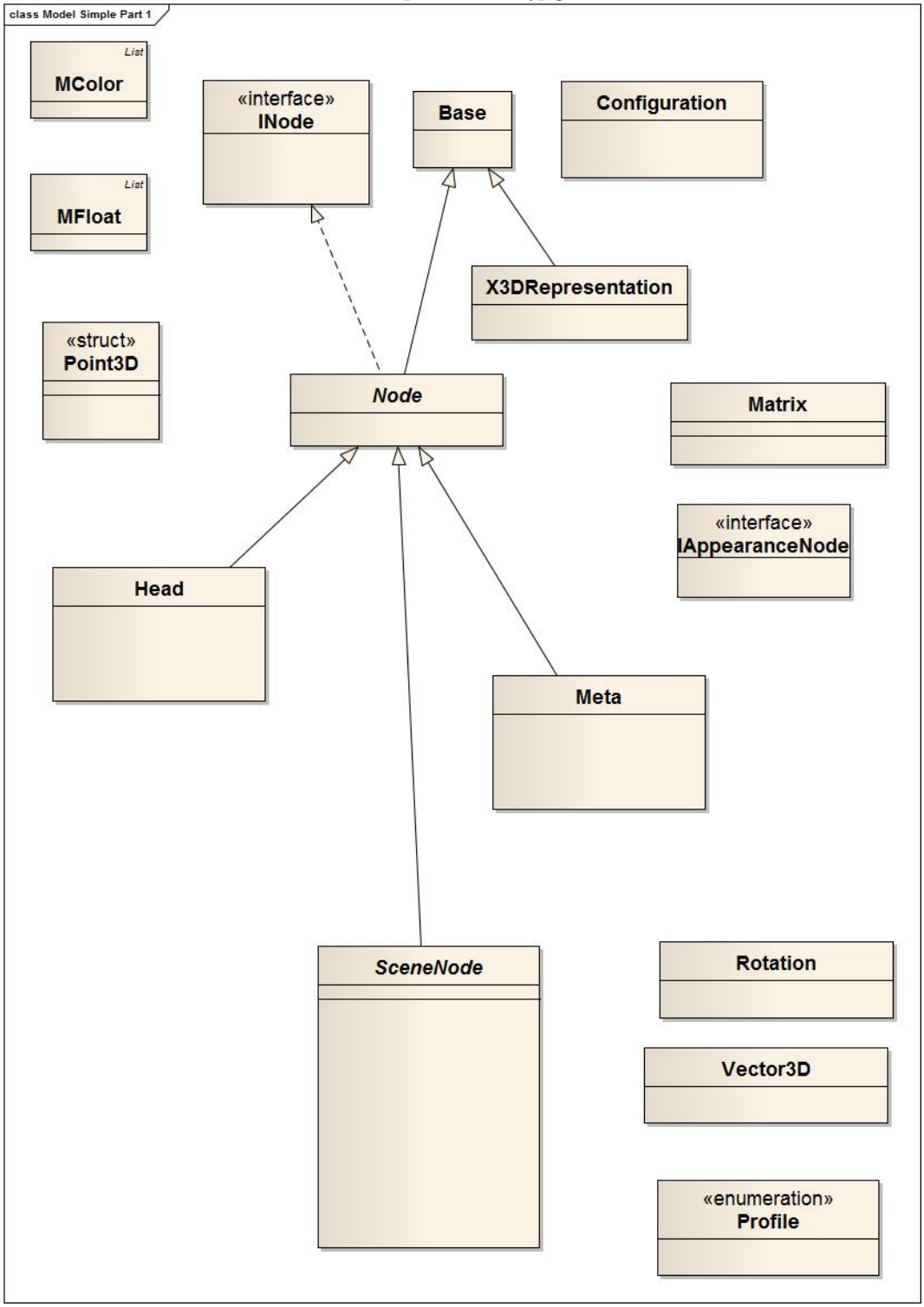

Figura A.2: Modelo de classes.

Simple Part 2.jpg

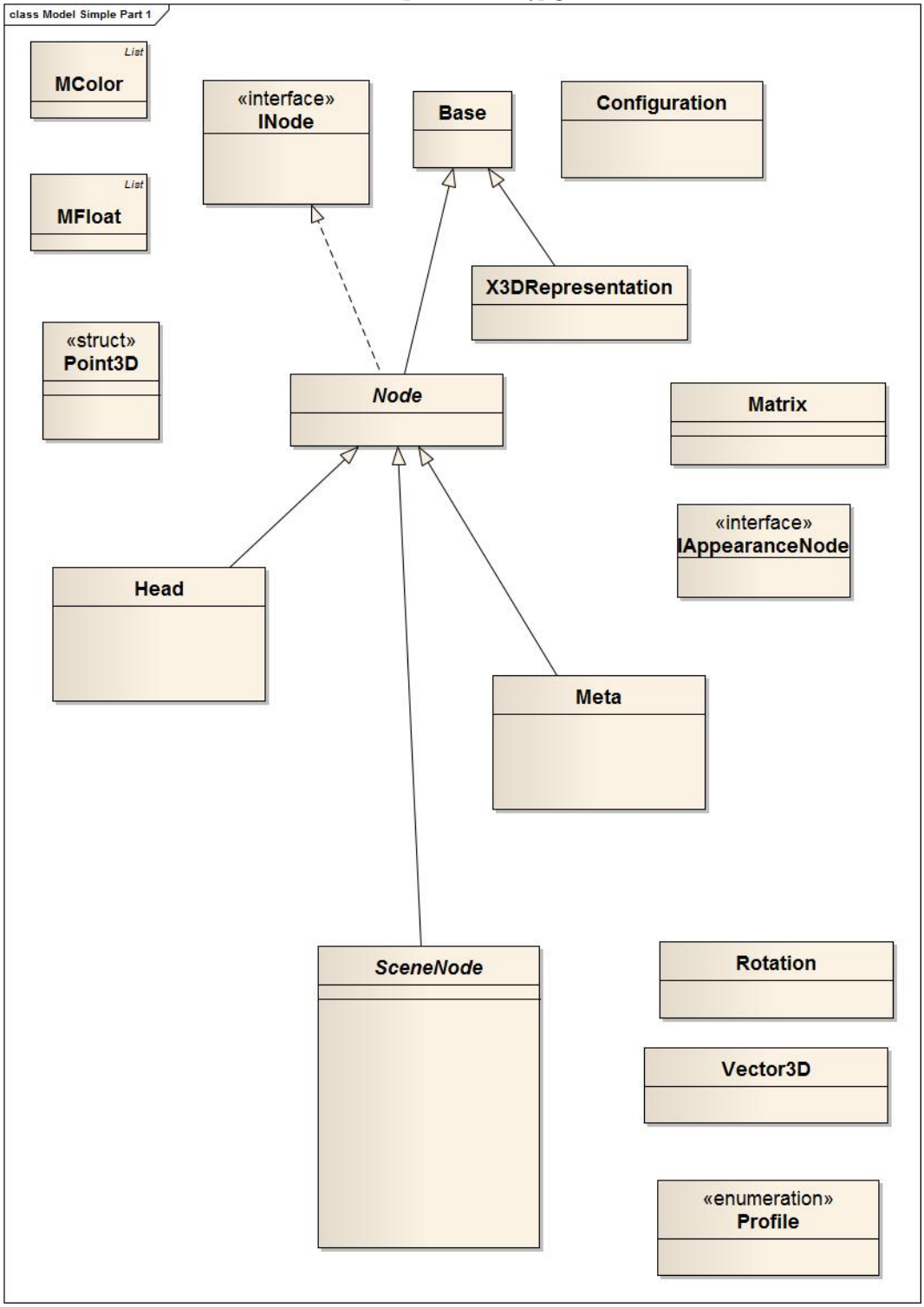

Figura A.3: Modelo de classes.

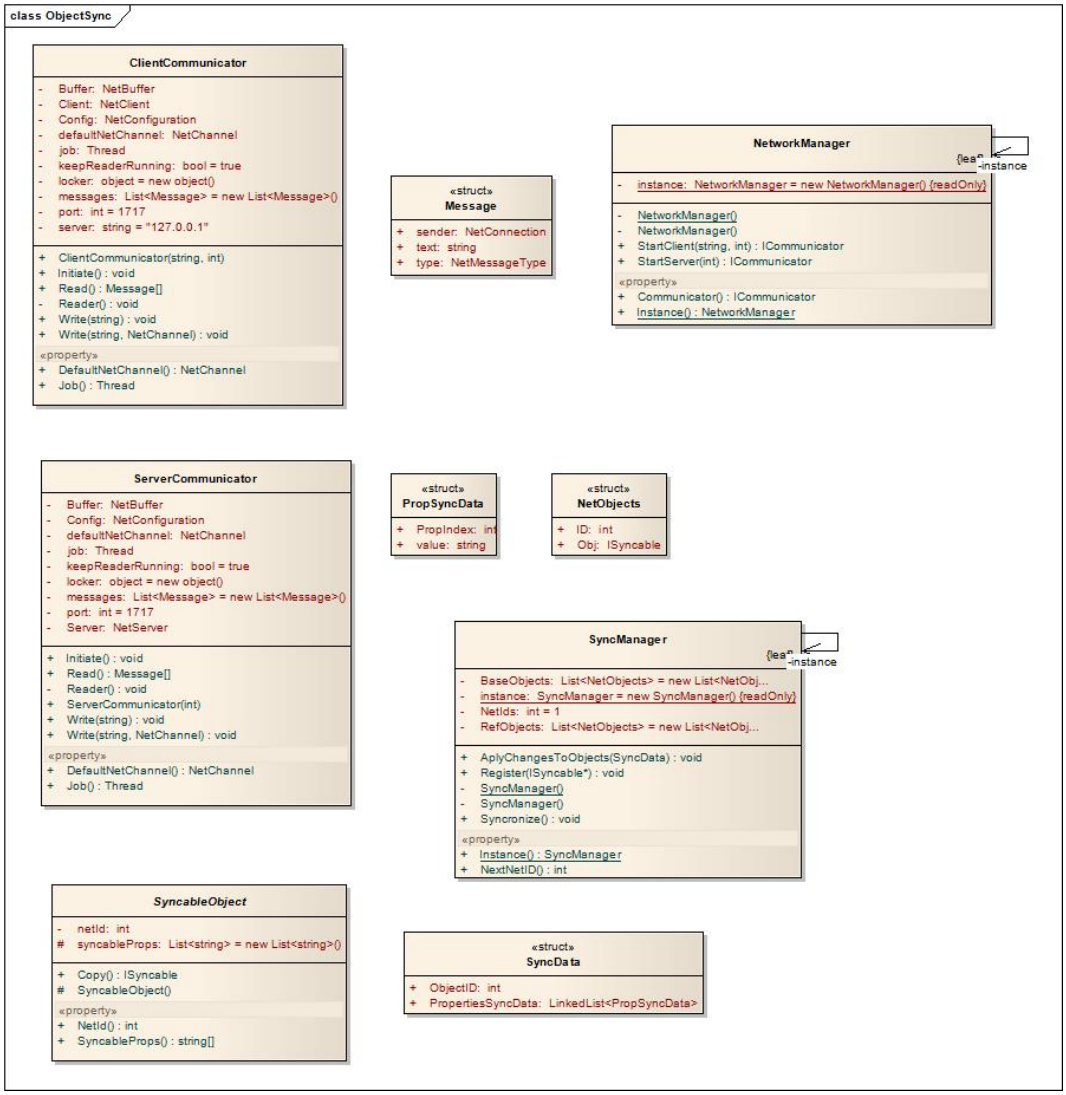

Figura A.4: Classes ObjetSync.

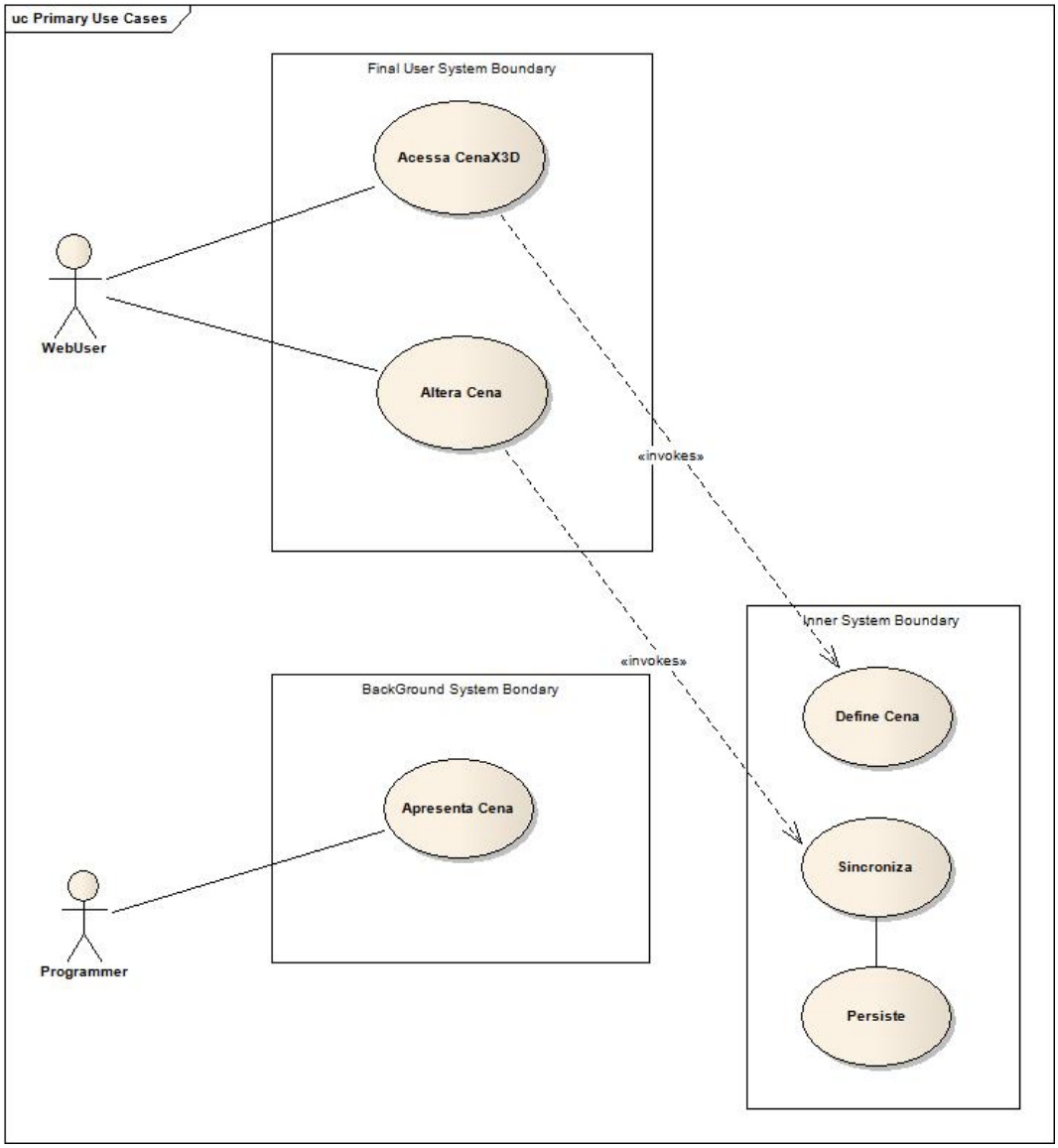

Figura A.5: Casos de uso principais.

B
Apêndice B - Conteúdo do exemplo contedo um cena X3D.

Conteúdo do arquivo Exemplo.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN"
   "http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D profile='Immersive' >
<head>
 <meta name='Vizthumbnail' content='Thumb_Exemplo_x3d5142461251326236.jpg'/>
 <meta name='ExportTime' content='19:37:16'/>
 <meta name='ExportDate' content='8/26/2009'/>
 <meta name='VivatyStudioVersion' content='709'/>
 <meta name='VivatyStudioSource' content='exemplo.fxw'/>
</head>
<Scene>
<Viewpoint DEF='Viewpoint1'
 description='Viewpoint1'
 jump='true'
 fieldOfView='0.785'
 position='-.07491 1.22863 13.5781'
 orientation='0 0 1 0'/>
<Transform DEF='dad_Box1'
 translation='2.01815 1.03237 -.03759'>
 <Shape DEF='Box1'
  containerField='children'>
  <Appearance
   containerField='appearance'>
   <Material DEF='Red'
    containerField='material'
    ambientIntensity='0.200'
    shininess='0.200'
    diffuseColor='1 0 0'/>
  </Appearance>
```

```
  <Box DEF='GeoBox1'
   containerField='geometry'
   size='2.01509 2 2'/>
 </Shape>
</Transform>
<Transform DEF='dad_Cylinder1'
 translation='-3.04034 1.94938 .0928'>
 <Shape DEF='Cylinder1'
  containerField='children'>
  <Appearance
   containerField='appearance'>
   <Material
    containerField='material'
    USE='Red'/>
  </Appearance>
  <Cylinder DEF='GeoCylinder1'
   containerField='geometry'
   height='4.000'
   radius='1.538'/>
 </Shape>
</Transform>
</Scene>
</X3D>
```

C
Apêndice C - Código para um chat padrão sem o uso do Toolkit

```
while ( this.clientSocket.Connected )
{
    //Descobre o tipo do comando.

    byte [] buffer = new byte [4];
    int readBytes = this.networkStream.Read(buffer , 0 , 4);
    if ( readBytes == 0 )
      break;
    CommandType cmdType =
         (CommandType)( BitConverter.ToInt32(buffer , 0) );

    //Lê o tamanhão do IP

    buffer = new byte [4];
    readBytes = this.networkStream.Read(buffer , 0 , 4);
    if ( readBytes == 0 )
    break;
    int senderIPSize = BitConverter.ToInt32(buffer , 0);

    //Lê o IP do cliente

    buffer = new byte [senderIPSize];
    readBytes =
        this.networkStream.Read(buffer , 0 , senderIPSize);
    if ( readBytes == 0 )
      break;
    IPAddress senderIP = IPAddress.Parse(
             System.Text.Encoding.ASCII.GetString(buffer));

    //Descobre o tamanho do nome do cliente.
```

```
buffer = new byte [4];
readBytes = this.networkStream.Read(buffer , 0 , 4);
if ( readBytes == 0 )
  break;
int senderNameSize = BitConverter.ToInt32(buffer , 0);

//Lê o nome do cliente

buffer = new byte [senderNameSize];
readBytes = this.networkStream.Read(buffer, 0, senderNameSize);
if ( readBytes == 0 )
  break;
string senderName =
    System.Text.Encoding.Unicode.GetString(buffer);

//Descobre o tamanho do nome do destinatário

string cmdTarget = "";
buffer = new byte [4];
readBytes = this.networkStream.Read(buffer , 0 , 4);
if ( readBytes == 0 )
   break;
int ipSize = BitConverter.ToInt32(buffer , 0);

//Lê o comando do destinatário

buffer = new byte [ipSize];
readBytes = this.networkStream.Read(buffer , 0 , ipSize);
if ( readBytes == 0 )
  break;
cmdTarget = System.Text.Encoding.ASCII.GetString(buffer);

//Descobre o tamamanho dos meta dados

string cmdMetaData = "";
buffer = new byte [4];
readBytes = this.networkStream.Read(buffer , 0 , 4);
if ( readBytes == 0 )
  break;
```

```
    int metaDataSize = BitConverter.ToInt32(buffer , 0);

    //Lê os meta dados

    buffer = new byte [metaDataSize];
    readBytes = this.networkStream.Read(buffer , 0 , metaDataSize);
    if ( readBytes == 0 )
         break;
    cmdMetaData = System.Text.Encoding.Unicode.GetString(buffer);

    Command cmd = new Command(cmdType,
                        IPAddress.Parse(cmdTarget), cmdMetaData);
    cmd.SenderIP = senderIP;
    cmd.SenderName = senderName;
    this.OnCommandReceived(new CommandEventArgs(cmd));
    }
    this.OnServerDisconnected(new ServerEventArgs(this.clientSocket));
    this.Disconnect();
}
```

D
Apêndice D - Trecho de código para transformação do grafo .NET em arquivo X3D

```
// Função para converter um cubo
private void RenderBox(Box box, Transf transform, Appearance appearance)
{
// Criando um objeto unity do tipo cubo
  GameObject cube = GameObject.CreatePrimitive(PrimitiveType.Cube);
  // definindo que ele vai se comportar como um Rigidbody
  // (Serve para colisão e simulações além de animação)
  cube.AddComponent("Rigidbody");
  //Translação
  if (transform != null && transform.Translation != null)
     // Definindo a posição da translação
     //(no unity cada objeto tem a sua)
     cube.transform.position =
       new Vector3(currentTransform.Translation.Coords.X,
         currentTransform.Translation.Coords.Y,
         currentTransform.Translation.Coords.Z);

  // Definindo a rotação
  if (transform != null && transform.Rotation != null)
    cube.transform.Rotate(new Vector3(currentTransform.Rotation.X,
      currentTransform.Rotation.Y,
      currentTransform.Rotation.Z),
      currentTransform.Rotation.A);

  //Materiais
  foreach (var material in materials)
  {

    if (appearance != null)
    {
```

```
        if (material.Appearance.ToString() ==
            appearance.ToString())
        {
          //Definindo valores do material
           cube.renderer.material.color =
             new Color(material.M.DiffuseColor.R,
               material.M.DiffuseColor.G,
               material.M.DiffuseColor.B);
         }
     }else
     {
        // se não tem um específico utilizar o default
        cube.renderer.material.color
          = new Color(material.M.DiffuseColor.R,
            material.M.DiffuseColor.G,
            material.M.DiffuseColor.B);
      }
    }
    // Aplicando a escala
    cube.transform.localScale = new Vector3(box.Size.Coords.X,
      box.Size.Coords.Y, box.Size.Coords.Z);
    // Definindo para não usar grafidade
    cube.rigidbody.useGravity = false;
    cube.rigidbody.isKinematic = true;

}
```