PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 11/04

Grafo de Cena e Realidade Virtual

Romano José Magacho da Silva
Alberto Barbosa Raposo
Marcelo Gattass

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

PUC RIO - DEPARTAMENTO DE INFORMÁTICA ISSN 0103-9741

Monografias em Ciência da Computação, Nº 11/04
Editor: Carlos J. P. Lucena Abril, 2004

Grafo de Cena e Realidade Virtual

Romano José Magacho da Silva
Alberto Barbosa Raposo
Marcelo Gattass

Grafo de Cena e Realidade Virtual

Romano J. M. da Silva, Alberto B. Raposo, Marcelo Gattass
Tecgraf – Depto. de Informática – PUC-Rio
{romano, abraposo, mgattass}@tecgraf.puc-rio.br
PUC-Rio.Inf.MCC11/04 Abril, 2004

Resumo

Várias aplicações de Realidade Virtual (RV) estão utilizando grafos de cena para representação do ambiente virtual, pois as otimizações existentes nessas bibliotecas aumentam a taxa de quadros por segundo da visualização. O aumento da taxa de quadros por segundo fornece maior sensação de imersão, característica fundamental em um ambiente de RV. A presente monografia apresenta um estudo sobre o conceito de grafo de cena, destacando o *OpenGL Performer* e o *OpenSceneGraph*. Também é feita uma descrição dos principais conceitos e ferramentas de RV, bem como suas relações com os grafos de cena.

Palavras-Chave: Realidade Virtual, Grafo de Cena, Computação Gráfica.

Abstract

Several applications in the field of Virtual Reality (VR) are using scene graphs to represent the virtual environment, since the optimizations offered by these libraries increase the frame rates of the visualization. The increase in the frame rate enhances the immersion sensation, a fundamental characteristic of a VR environment. This text presents a study about the scene graph concept, highlighting the *OpenGL Performer* and the *OpenSceneGraph*. Some of the main concepts and tools of VR are also described, as well as their relations with scene graphs.

Keywords: Virtual Reality, Scene Graph, Computer Graphics.

Introdução

Realidade Virtual (RV) possui várias definições diferentes, porém, é de comum acordo que a essência da realidade virtual está nos ambientes tridimensionais baseados em computadores. Geralmente chamados de "mundo virtual" ou "ambiente virtual", eles são representações do mundo real ou de ambientes conceituais que podem ser investigados através de navegação, interação e atualizações em tempo real.

A construção dos ambientes consiste em representar o mundo real utilizando primitivas que possam reproduzi-lo no ambiente computacional. Existem várias técnicas disponíveis para essa representação, sendo a mais comum a representação por malhas de triângulos.

No ambiente computacional, as bibliotecas gráficas são responsáveis por fazer a ligação entre a aplicação e a placa gráfica. A aplicação envia a malha de triângulos, juntamente com seus atributos, como cor, material, texturas, etc., para a biblioteca gráfica e esta, por sua vez, envia os vértices para a placa gráfica que fará os cálculos necessários para transformar a coordenada 3D dos vértices dos triângulos em *pixels* na tela do computador.

As bibliotecas gráficas mais utilizadas hoje em dia são a OpenGL e a Direct3D. A OpenGL é uma biblioteca gratuita, disponível na maioria dos sistemas operacionais, que possui um mecanismo de extensão que permite que novas funcionalidades das placas gráficas sejam incorporadas à aplicação sem a necessidade de lançamento de uma nova versão da biblioteca. A Direct3D é uma biblioteca proprietária desenvolvida pela empresa Microsoft e distribuída para os usuários do sistema Windows®.

Com o avanço da tecnologia das placas gráficas, as aplicações de computação gráfica e, conseqüentemente, de realidade virtual, estão sendo capazes de produzir efeitos visuais jamais vistos em aplicações em tempo real. Porém, essa tecnologia traz consigo um certo grau de complexidade. As interfaces das bibliotecas gráficas começaram a aumentar em tamanho e complexidade. A fim de se alcançar um efeito de multi-textura (duas texturas aplicadas uma sobre a outra em um triângulo), por exemplo, dezenas de linhas de código devem ser incluídas na aplicação.

Além do aumento da complexidade de desenvolvimento, o tamanho dos modelos empregados para representar o ambiente virtual também aumentou. Para que a placa gráfica não perca tempo de processamento, é necessário que algoritmos de otimizações sejam utilizados pela aplicação. Um exemplo desse tipo de algoritmo é o descarte por campo de visão. Nesse algoritmo, os objetos que estiverem fora do campo de visão do observador são descartados e não precisam ser enviados para a placa gráfica. Caso não houvesse esse algoritmo, os triângulos teriam que passar por todo o *pipeline* de renderização [2] até que fossem descartados no final por não estarem dentro da área visível do monitor.

A fim de facilitar o desenvolvimento de aplicações, uma nova forma de representar os objetos pode ser utilizada: são os chamados grafos de cena, projetados para, dentre outras coisas, facilitar o uso de algoritmos de otimização.

Um grafo de cena provê uma camada de abstração sobre a biblioteca gráfica, simplificando tarefas como a de multi-textura citada anteriormente, além de fornecer diversos algoritmos de otimização, como o descarte por campo de visão, entre outros. Um grafo de cena organiza, de forma espacial, os objetos

presentes em um modelo, permitindo que os algoritmos de otimização atuem de forma hierárquica [38].

Várias aplicações de RV estão utilizando grafos de cena para representação do ambiente virtual, pois as otimizações existentes na biblioteca aumentam a taxa de quadros por segundo. O aumento da taxa de quadros por segundo fornece maior sensação de imersão, característica fundamental em um ambiente de RV.

Além da preocupação em aumentar a velocidade de renderização de uma aplicação, os sistemas de RV devem ser capazes de executar em diversas configurações.

Com o desenvolvimento das tecnologias de RV, diversos tipos de dispositivos de interação e de saída foram surgindo. Para interagir com o mundo virtual, o usuário pode sentar em uma cadeira, utilizando um capacete de RV e tendo em sua mão uma luva, ou ele pode estar cercado por paredes de projeção, em uma configuração denominada CAVE, movendo os objetos utilizando um *mouse* sem fio ou um dispositivo óptico ou eletromagnético.

Para facilitar o desenvolvimento de aplicações multi-dispositivos, surgiram algumas bibliotecas de RV que são capazes de fornecer dados como a posição do dispositivo, ou a sua orientação, sem que a aplicação saiba de qual dispositivo este dado está vindo. Elas agem como uma camada de abstração. Além dos dispositivos de entrada, a aplicação pode gerar a saída visual sem saber se ela será apresentada em um monitor ou em uma parede com várias telas de projeção. Algumas camadas de abstração de realidade virtual foram analisadas e, neste trabalho, é apresentada a biblioteca ViRAL (***Vi****rtual* ***R****eality* ***A****bstraction* ***L****ayer*), desenvolvida pelo grupo de realidade virtual do TecGraf/PUC-Rio.

A próxima seção descreve as características de um grafo de cena apresentando vantagens e desvantagens do seu emprego. As bibliotecas de grafo de cena, OpenSceneGraph e OpenGL Performer serão analisadas. Em seguida, é feito um estudo das tecnologias de RV existentes, mostrando seu rápido crescimento e justificando a necessidade do uso de camadas de abstração. Duas camadas de abstração serão analisadas: o VRJuggler e o ViRAL.

Grafos de Cena

Grafos de cenas são ferramentas conceituais para representação de ambientes virtuais tridimensionais nas aplicações de computação gráfica [43]. Um ambiente virtual é uma representação de diversos aspectos do mundo real ou abstrato. Os aspectos que são considerados em uma aplicação de computação gráfica são: posição do objeto, forma, textura da superfície, iluminação, entre outros [14]. Cada um desses aspectos e seus atributos estão bastante detalhados na literatura, porém apresentamos aqui um resumo dos mais importantes:

- Descrição geométrica: A descrição geométrica é qualquer maneira de se representar a forma da entidade que pode ser processada para se obter uma imagem dessa entidade. A maneira mais comum é a representação aproximada por um conjunto de polígonos (mais especificamente por triângulos). O grau de complexidade da descrição geométrica é, em geral, diretamente proporcional à qualidade visual, porém inversamente proporcional à velocidade com que a imagem é gerada;
- Câmera: A câmera é a visão do mundo virtual. Geralmente ela é uma câmera de projeção perspectiva;
- Transformação: O objeto é posicionado no mundo virtual através de uma transformação geométrica. Ela transforma as coordenadas locais do objeto nas coordenadas do mundo virtual. As transformações são muito importantes para definir a hierarquia dos objetos;
- Aparência: Material, textura, transparência, sombra e reflexão estão entre os diversos atributos que definem a aparência de um objeto. Assim como a descrição geométrica, a aparência interfere diretamente na imagem final sendo gerada e na velocidade de geração;
- Comportamento: um objeto pode ser estático ou dinâmico. O objeto dinâmico é aquele que muda de posição, forma ou aparência entre um quadro e outro;
- Iluminação: várias fontes de luz podem ser adicionadas à cena e vários são os modelos de iluminação que podem ser empregados. O mais utilizado é o Gouraud [18] pois o seu cálculo é feito em hardware nas placas gráficas atuais, porém com o advento dos processadores gráficos programáveis, outros modelos podem ser usados, como o Phong [32] e o anisotrópico [34].

Cada um desses aspectos deve ser inserido em um grafo de cena para representar o ambiente virtual. O grafo de cena é formado, portanto, por nós conectados por arestas compondo um grafo acíclico direcionado. Cada nó possui um conjunto de atributos que podem, ou não, influenciar seus nós conectados. Os nós são organizados de uma maneira hierárquica correspondendo semântica e espacialmente com o mundo modelado.

Os nós podem ser divididos em três categorias: nó raiz, nós intermediários que são chamados de nós internos ou nós de agrupamento e os nós folha que estão localizados no final de um ramo. O nó raiz é o primeiro nó do grafo e todos os outros nós estão ligados a ele direta ou indiretamente. O nós internos possuem várias propriedades, sendo o uso mais comum o de representar transformações 3D (rotação, translação e escala). Os nós folha contêm, geralmente, a representação

geométrica de um objeto (ou dados de áudio, quando o grafo de cena possuir esse recurso).

Construindo um Grafo de Cena

Os nós devem ser conectados formando um grafo acíclico e direcionado. As conexões entre eles devem satisfazer as relações espaciais e semânticas existentes no mundo real. A Figura 1 mostra um exemplo de um grafo de cena representando uma casa. A casa é dividida em quartos e os quartos possuem diversos objetos (nós folha), como cama e cadeira, por exemplo. O grafo foi construído de maneira a manter a noção intuitiva dos relacionamentos entre os objetos citados.

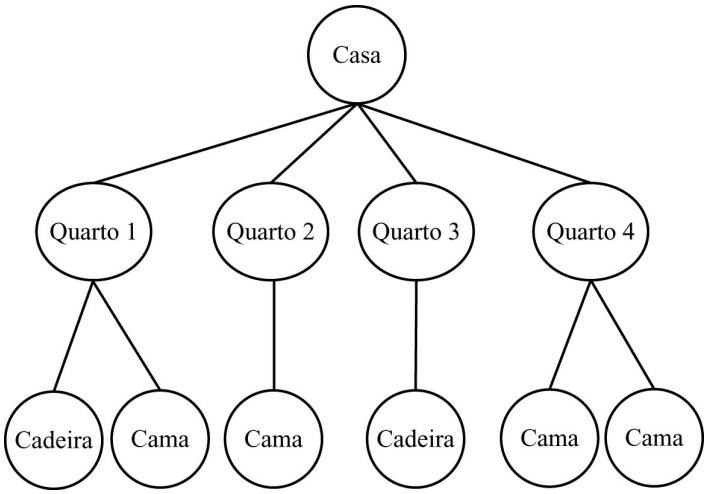

Figura 1: Grafo de cena bem construído de uma casa.

O mesmo grafo poderia ser construído de outra maneira, agrupando-se as cadeiras e camas em nós separados, logo, ao invés de uma casa com quatro quartos, teríamos uma casa e, logo abaixo, dois nós agrupando cadeiras e camas (Figura 2). Esse modelo está semanticamente relacionado com a casa, pois a mesma possui camas e cadeiras, porém os objetos geométricos não estão agrupados espacialmente. Esse resultado indesejado é contrário ao uso eficiente do grafo de cena, como veremos adiante.

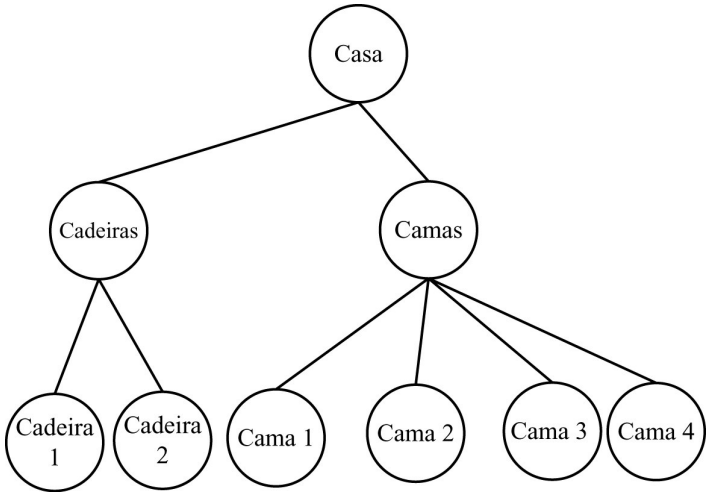

Figura 2: Estratégia de uso não eficiente de um grafo de cena.

Propriedades

Os grafos de cena implementam um princípio chamado de herança de estado. Os nós internos armazenam o estado do sistema, onde estado significa a posição e a orientação dos objetos no ambiente virtual e seus atributos de aparência. A herança de estado é uma propriedade dos grafos de cena que determina que cada nó deve herdar as propriedades de estado de todos os seus ancestrais no grafo até a raiz.

Analisemos, novamente, o modelo da casa e admitamos que seus objetos estejam modelados em relação à origem do sistema de coordenada. Em cada quarto acrescentamos uma translação que irá posicionar seus quartos corretamente em relação à casa. A casa pode ser ainda rotacionada, por exemplo, a fim de ficar voltada para uma determinada direção. Essa cena é ilustrada na Figura 3. Devido à herança de estado, todas as geometrias identificadas pelos nós raízes herdarão as propriedades dos seus ancestrais e serão posicionadas corretamente. A herança de estado é, portanto, uma ferramenta bastante útil para organização da cena 3D.

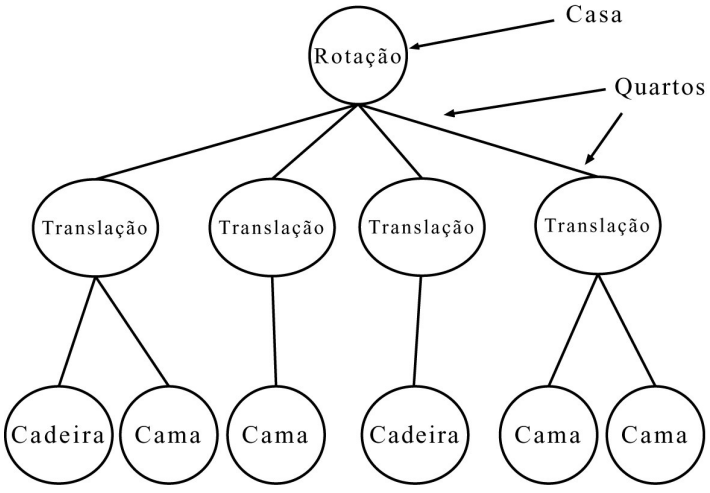

Figura 3: Usando transformações para posicionas os quartos e a casa.

Todos os nós de um grafo de cena podem possuir atributos como material, textura, transparência, entre outros. Todos esses atributos são herdados dos nós ancestrais. Um determinado nó pode sobrescrever um determinado atributo e, sendo assim, toda sua subárvore será modificada. A

Figura 4 ilustra um exemplo de um conjunto de objetos com um determinado material (cor) e alguns nós sobrescrevendo esse atributo.

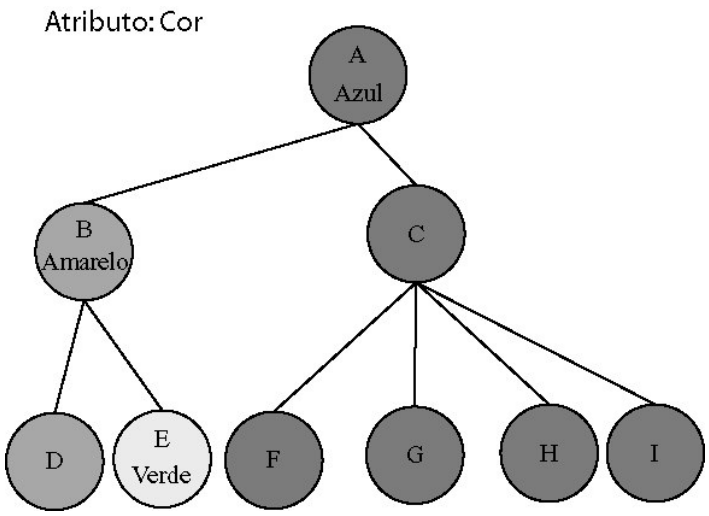

Figura 4: Materiais redefinidos ao longo do grafo. Os nós herdam a cor do nó ancestral.

Otimizações

As aplicações de computação gráfica possuem um *pipeline* de renderização responsável por transformar um vértice em coordenadas do mundo para as coordenadas da tela. Esse *pipeline* é composto por três etapas: *Aplicação*, *Geometria* e *Rasterização*.

Atualmente, as etapas de *Geometria* e *Rasterização* são realizadas em hardware. A primeira é responsável pela transformação e iluminação dos vértices que chegam à placa gráfica. A *Rasterização* recebe os *pixels* com seus valores de cor interpolados de acordo com o algoritmo de Gouraud e o valor da profundidade daquele *pixel*. Com esse valor, um teste de profundidade (Z-buffer [2]) é feito para saber se aquele *pixel* será renderizado ou descartado.

A etapa de *Aplicação* é responsável pelo envio de vértices para a placa gráfica. Essa etapa é totalmente controlada pelo usuário e é nela onde atua o grafo de cena.

Em geral, quanto menos vértices forem enviados para a placa gráfica, melhor será o desempenho geral da aplicação. O desempenho também é influenciado pela troca de estado da aplicação. Se a aplicação modificar várias vezes a textura ativa a ser usada em um modelo, a placa gráfica precisa ser informada. A cada troca de estado na placa gráfica, uma validação deve ser realizada e essa validação é uma etapa demorada. Portanto, é importante que a aplicação agrupe seus polígonos por estado, ao invés de mandá-los em ordem aleatória.

Uma boa implementação de um grafo de cena deve ser capaz de otimizar o número de vértices enviados para a placa gráfica, assim como o número de trocas de estado. Para isso, algumas técnicas devem ser empregadas.

Todos os nós do grafo de cena possuem um atributo denominado volume envolvente. Esse atributo é um volume simples, geralmente uma caixa alinhada ou uma esfera, que engloba o conteúdo de todos os nós abaixo do nó em questão. O volume envolvente é utilizado em um grafo de cena para verificação de visibilidade e descarte. As técnicas empregadas para geração dos volumes envolventes estão detalhadas em [38].

As técnicas mais empregadas de descarte em um grafo de cena são:

- Descarte por volume de visão: o grafo de cena testa a interseção o volume envolvente do nó com o volume de visão do observador. Se o volume envolvente estiver completamente fora do campo de visão, o nó e toda a sua subárvore são descartados. Se o mesmo estiver completamente dentro do campo de visão, o nó e toda a sua subárvore são percorridos e caso a interseção seja parcial, o teste é refeito durante o percurso da subárvore;
- Descarte por oclusão: os algoritmos de descarte por oclusão têm por objetivo evitar a renderização de primitivas que estejam ocultas por outras partes da cena. A idéia por trás dos algoritmos de oclusão é realizar algum pré-processamento ou alguns testes durante a renderização para evitar que dados sejam enviados desnecessariamente para a placa. Os algoritmos mais conhecidos são o *Shadow Frusta* [21], *Hierarchical Z-Buffer (HZB)* [19] e *Hierarchical Occlusion Maps* [45]. O HZB está implementado em hardware nas placas gráficas atuais.
- Descarte de objetos pequenos: o volume envolvente dos nós é projetado na tela. Caso ele ocupe menos que um valor limite em *pixels*, esse objeto é descartado. A idéia desse descarte é a de que objetos muito pequenos em cenas complexas não influenciam muito na imagem final.

Com o descarte, o grafo de cena evita o envio de objetos que não estão visíveis para a placa gráfica, dessa forma reduzindo a quantidade de vértices a serem renderizados. É interessante notar ainda a característica hierárquica desses algoritmos. Se um nó for descartado, toda a sua subárvore será descartada. Por esse motivo, uma cena organizada espacialmente será muito mais eficiente do que uma cena que foi montada aleatoriamente.

Outra otimização que reduz a quantidade de vértices é chamada de Níveis de Detalhe (*LOD – Level of Detail*) [26]. A idéia por trás dessa técnica é a de que objetos muito distantes do observador podem ser renderizados com menor qualidade visual (menor número de polígonos) do que objetos mais próximos (Figura 5). Os níveis de detalhe podem ser estáticos, e portanto gerados uma única vez em alguma etapa de pré-processamento, ou dinâmicos, sendo gerados conforme a distância do observador ao objeto.

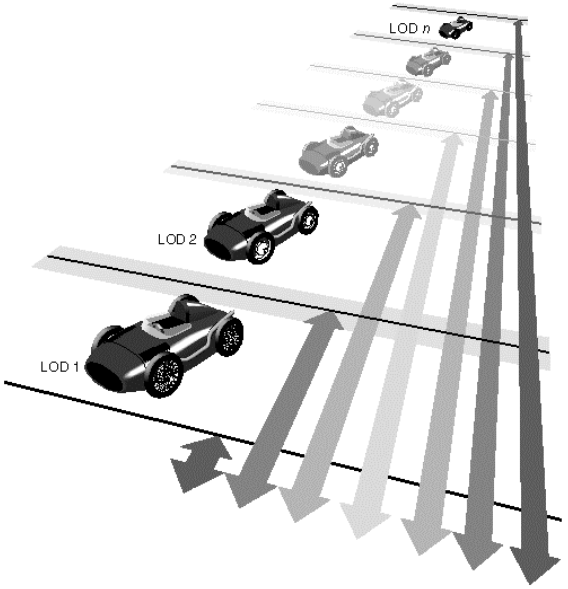

Figura 5: Níveis de Detalhes: objetos mais distantes são vistos com menos detalhe [44].

A fim de suportar a otimização de troca de estados, o grafo de cena deve ser capaz de armazenar todos os atributos que serão utilizados para renderizar antes de efetivamente enviar os vértices para placa de vídeo. De posse dessa lista de atributos, ele ordena os objetos geométricos por estado e depois os envia para a placa.

Vantagens

Uma vantagem direta que o grafo de cena traz com todas as otimizações que ele implementa é a melhoria do desempenho da aplicação. Mas essa não é a única vantagem, como mostramos na lista a seguir:

- **Produtividade**: Os grafos de cena diminuem o trabalho de se desenvolver aplicações gráficas de alto desempenho. O grafo de cena gerencia toda a parte gráfica, reduzindo as várias linhas de código que seriam necessárias para implementar a mesma funcionalidade utilizando uma interface de programação baixo nível, como a OpenGL. Além disso, vários conceitos avançados de programação orientada a objetos, como o uso de padrões de projeto [16], tornam o grafo de cena uma aplicação flexível e de fácil reuso. Além disso, um grafo de cena geralmente é acompanhado de outras bibliotecas responsáveis por gerência de janelas, carregamento de modelos 3D e imagens. Tudo isso faz com que o usuário possa desenvolver uma aplicação com poucas linhas de código;
- **Portabilidade**: Os grafos de cena encapsulam todas as tarefas de baixo nível necessárias para renderizar a cena e ler e escrever arquivos, reduzindo, ou até mesmo extinguindo, a quantidade de código que é específica de alguma plataforma a ser inserido na aplicação. Sendo assim, se o grafo de cena for portável, imediatamente toda a aplicação será portável, sendo necessário apenas uma nova compilação ao se mudar de plataforma;

- **Escalabilidade**: os grafos de cena são feitos para funcionar em configurações simples baseadas em computadores de mesa e placas gráficas aceleradores convencionais ou em hardware complexos, tais como *clusters* de máquinas gráficas, ou sistemas multiprocessados/*multipipe*. O desenvolvedor não se preocupa com a configuração em que ele irá rodar a aplicação, podendo estar focado exclusivamente no seu desenvolvimento.

Ferramentas disponíveis

Várias bibliotecas de grafo de cena existem há algum tempo. A lista a seguir não é, de maneira alguma, completa, mas apresenta os mais utilizados:

- SGI OpenGL Performer (http://www.sgi.com/products/performer)
- OpenSceneGraph (http://www.openscenegraph.org)
- OpenSG (http://www.opensg.org)
- Open Inventor (http://oss.sgi.com/projects/inventor)
- PLIB (http://plib.sf.net)
- SGL (http://sgl.sf.net)
- OpenRM (http://openrm.sf.net)

Apenas as duas primeiras da lista serão detalhadas a seguir.

SGI OpenGL Performer

O OpenGL Performer, da empresa Silicon Graphics, é uma interface de programação para o desenvolvimento de aplicações gráficas 3D em tempo real. Ela funciona em plataformas IRIX, Linux e Windows baseada na biblioteca gráfica OpenGL [29]. Além do grafo de cena, ela possui uma série de outros módulos que permitem o melhor desempenho da aplicação. A Figura 6 ilustra a organização das bibliotecas que compõem a interface.

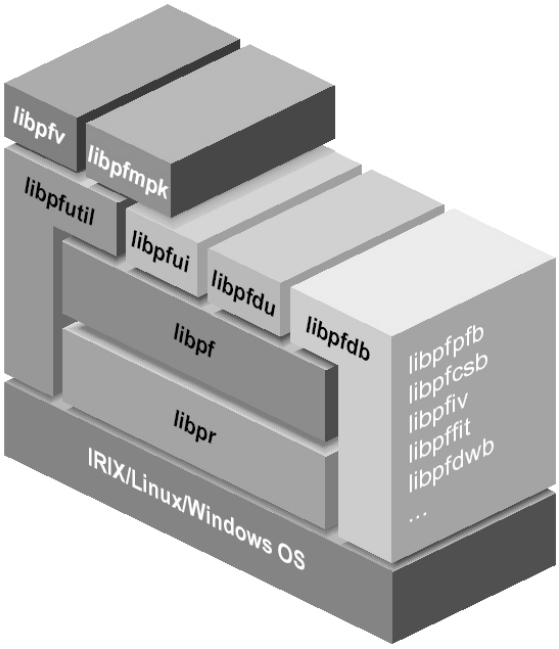

Figura 6: Hierarquia de módulos do OpenGL Performer [30].

A biblioteca *libpr* é a biblioteca base e a responsável pelo desempenho de renderização. Ela é uma biblioteca desenvolvida em C e com uma interface C++, orientada a objeto, que fornece funções de renderização otimizadas, controle de troca de estados eficiente e outras funcionalidades gráficas independentes de aplicação (rotinas de matemática, detecção de colisão, criação e gerência de janelas, etc).

Acima da *libpr* está localizada a *libpf*. Essa biblioteca fornece um ambiente de visualização em tempo real através de um grafo de cena, além de um sistema de renderização multiprocessado. Esse sistema multiprocessado aproveita o máximo das arquiteturas SGI.

As outras bibliotecas são responsáveis por leitura de arquivo (*libpfdb*), criação de manipuladores e componentes de interface com usuário (*libpfui*) e configuração de renderização multiprocessada e multicanal, além de interface com usuário (*libpfutil*).

O grafo de cena contém a informação que define o mundo virtual. Ele inclui descrição da geometria e a sua aparência, informação espacial, tais como transformação, animação, níveis de detalhe, efeitos especiais entre outros. O OpenGL Performer atua no grafo de cena para realizar a renderização e o descarte. Para isso, entram em ação dois conceitos: *channel* e *pipe* (Figura 7).

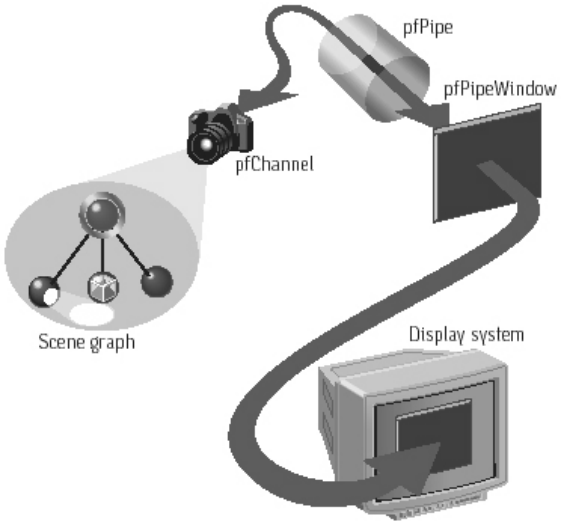

Figura 7: O caminho do grafo de cena [30].

Um *channel* é equivalente a uma câmera movendo-se dentro da cena. Cada *channel* está associado a uma área da configuração final de visualização. O *pipe* é o coração de todo processamento feito pelo OpenGL Performer. Ele renderiza cada *channel* na sua área de projeção na tela. Nos equipamentos da SGI, cada *pipe* pode ser associado a um processador gráfico. Essas máquinas são chamadas *multi-piped*.

O OpenGL Performer é uma excelente solução de grafo de cena e renderização em tempo real. Porém, ele é uma solução comercial de alto custo, desenvolvida especialmente para os equipamentos SGI. Em outros equipamentos, boa parte das otimizações que são extremamente importantes, como o conceito de *multi-pipe*, não está presente.

OpenSceneGraph

O *OpenSceneGraph* é uma interface de programação construída sobre a biblioteca gráfica OpenGL [29]. Ele é responsável pela gerência do grafo de cena e otimizações gráficas. Ele é multi-plataforma, gratuito e o seu código é aberto. A Figura 8 ilustra os módulos da interface.

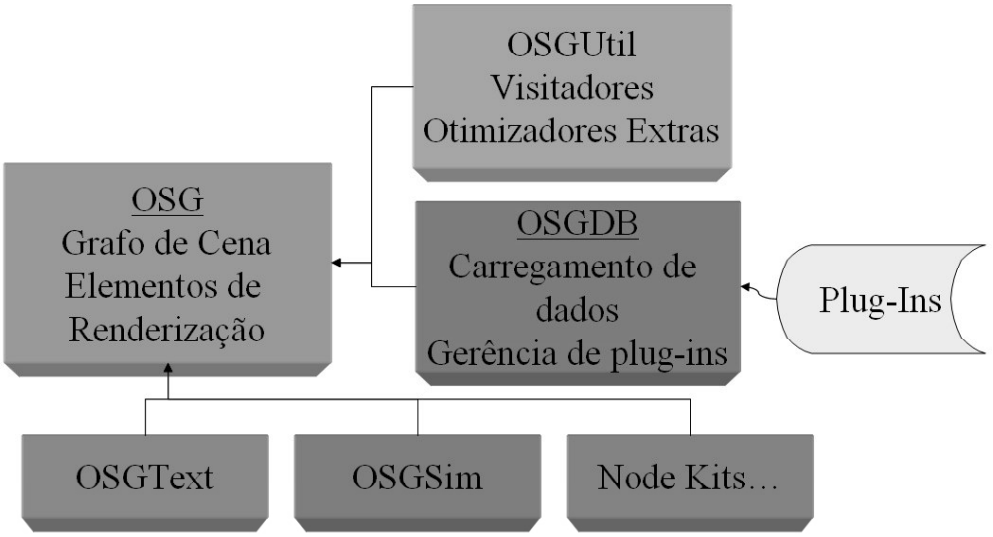

Figura 8: Módulos da interface de programação *OpenSceneGraph*.

O módulo *osg* é responsável pelo gerenciamento do grafo de cena. Ele possui as otimizações básicas, como descarte hierárquico por campo de visão e por oclusão, níveis de detalhe e gerência de troca de estados. O módulo *osgUtil* possui a câmera da cena e outras rotinas de otimização, como a remoção de transformações estáticas, fusão de nós do mesmo estado, partição espacial da cena para melhor descarte, geração de geometrias otimizadas para renderização, e outras.

Os **Visitadores** existentes no módulo *osgUtil* são responsáveis pelo percurso no grafo. Os **Visitadores** implementam o padrão de projeto de mesmo nome [16]. A Figura 9 ilustra o diagrama de classes que implementam esse padrão. Existem, geralmente, 2 visitadores padrão: o visitador de atualização e o de descarte.

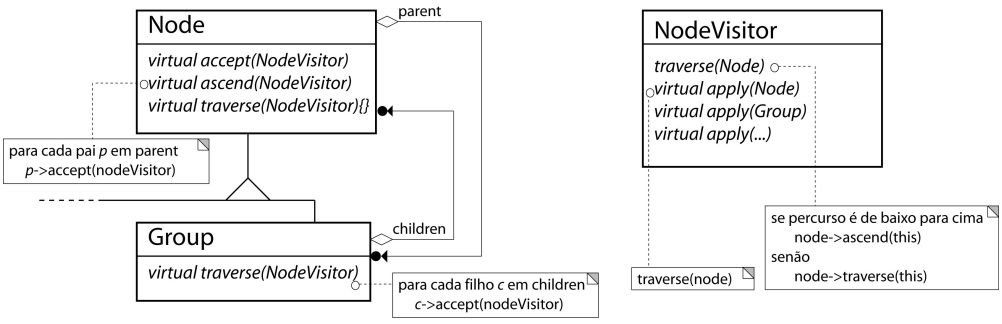

Figura 9: Padrão de projeto visitadores utilizado para percorrer o grafo de cena.

O visitador de atualização é responsável por atualizar os objetos dinâmicos do grafo. O visitador de descarte percorre o grafo e elimina todos os objetos passíveis de exclusão e retorna uma lista de objetos renderizáveis. A etapa de renderização é feita com o resultado obtido pelo visitador de descarte. Durante a renderização a troca de estado é feita por demanda (*lazy-state change*).

O módulo *osgDB* é responsável por implementar uma cadeia de responsabilidade [16] para o carregamento de arquivos de modelos tridimensionais e de imagens. Com isso, o usuário não precisa se preocupar com a leitura em baixo nível dos arquivos, tornando sua aplicação mais portável. Os

leitores de arquivo são implementados na forma de *plugins*, tornando o *osgDB* bastante flexível e extensível. Os módulos *osgText* e *osgSim* são extensões do módulo *osg* para suportar a renderização de texto e outros nós não-específicos, como, por exemplo, pontos de luz.

O *OpenSceneGraph* possui a vantagem de ser uma interface de programação disponível gratuitamente e de código aberto. Apesar de não possuir muita documentação disponível, a quantidade de exemplos e o próprio código fonte auxiliam a sua compreensão.

Por ser orientado a objetos, ele é bastante extensível, permitindo ao usuário a criação de novas funcionalidades sem a necessidade de modificação do seu código fonte. A comunidade existente em torno do OpenSceneGraph é bastante numerosa e existem diversas outras bibliotecas que criaram uma interface para ele, como é o caso do motor de física Vortex da CMLabs [31] e a biblioteca de terrenos Demeter [13].

Realidade Virtual

Com aplicação na maioria das áreas do conhecimento, entre elas a medicina, mecânica, treinamento militar, ergonomia, jogos e entretenimento, e com um grande investimento das indústrias na produção de hardware, software e dispositivos de entrada e saída, a realidade virtual vem experimentando um desenvolvimento acelerado nos últimos anos e indicando perspectivas bastante promissoras para os diversos segmentos vinculados com a área.

Os primeiros esboços de equipamentos de realidade virtual foram criados por Ivan Shutherland em meados dos anos 60, no *Massachussetts Institute of Technology* (MIT), estando entre eles o primeiro capacete de realidade virtual (*HMD – head mounted display*) e um dispositivo de interação chamado *Sketchpad* [40], resultado final de sua tese de doutorado em 1963. Apesar do pioneirismo de Sutherland, o termo Realidade Virtual (RV) acabou sendo creditado a Jaron Lanier, fundador da VPL Research Inc., que o cunhou, no início dos anos 80, para diferenciar as simulações tradicionais feitas por computador de simulações envolvendo múltiplos usuários em um ambiente compartilhado [4]. Pesquisas como a de Myron Krueger, em meados da década de 70, já utilizavam o termo realidade artificial, e William Gibson utilizou o termo espaço cibernético (*cyberspace*) em 1984, no seu romance de ficção científica Neuromancer [17], para designar uma representação gráfica de dados abstraídos dos bancos de dados de todos os computadores do sistema humano.

Não existe na academia ou na indústria, um consenso quanto à definição formal de realidade virtual. Em [6] encontramos realidade virtual descrita como sendo uma forma das pessoas visualizarem, manipularem e interagirem com computadores e dados extremamente complexos. Através de outras definições existentes em [10,23,25] pode-se dizer que realidade virtual é uma técnica avançada de interface, onde o usuário pode navegar e interagir em um ambiente sintético tridimensional gerado por computador, estando completa ou parcialmente presente ou imerso pela sensação gerada por canais multi-sensoriais, sendo o principal a visão.

Um dos benefícios dos ambientes de RV é a capacidade de prover perspectivas vantajosas impossíveis de se obter no mundo real, como por exemplo, a navegação por dentro do corpo humano ou a análise de simulações físicas ou reações químicas em tempo real. A interação com o ambiente virtual é feita utilizando dispositivos próprios para RV, como capacetes de visualização, luvas, *mouses* tridimensionais, sensores de posicionamento, entre outros.

As aplicações de RV se diferenciam das aplicações convencionais, além dos tipos de dispositivos que utilizam, pela característica presencial ou imersiva do usuário no sistema. O usuário se sente imerso quando as sensações e eventos que acontecem no mundo virtual são sentidos por ele no mundo real. Atualmente se desenvolvem pesquisas buscando atuar principalmente sobre as percepções visuais, auditivas e tácteis.

Quanto à visão, o objetivo principal é fornecer ao usuário a sensação de profundidade, que não deve ser confundida com a noção de profundidade obtida pela distorção de perspectiva. A distorção de perspectiva permite distinguir objetos que estão próximos dos que estão mais distantes, enquanto a sensação de profundidade permite que o usuário seja inserido no ambiente virtual.

Para atingir esse objetivo, o sistema deve gerar, ao mesmo tempo, duas imagens diferentes, correspondendo às visões dos olhos esquerdo e direito. O cérebro humano processa as diferenças entre essas duas imagens gerando uma representação precisa da posição e da forma tridimensional do objeto dentro da cena. Esse é a verdadeira profundidade tridimensional que experimentamos no mundo real. Esse tipo de geração de imagem é chamado de visão estereoscópica ou estereoscopia.

A visão estereoscópica é obtida naturalmente com o emprego de um capacete de realidade virtual que possui dois visores, um para cada olho, ou através da utilização de óculos de abertura.

Sistemas de Realidade Virtual

Os sistemas de realidade virtual são compostos por dispositivos de entrada (rastreadores de posição, reconhecimento de voz, *joysticks* e mouses, por exemplo) e saída (visual, auditiva e táctil) capazes de prover ao usuário imersão, interação e envolvimento [28]. A idéia de imersão está ligada com o sentimento de se estar dentro do ambiente. A interação diz respeito à capacidade do computador detectar as entradas do usuário e modificar instantaneamente o mundo virtual e as ações sobre ele. O envolvimento, por sua vez, está ligado com o grau de engajamento de uma pessoa com determinada atividade, podendo ser passivo, como assistir televisão, ou ativo, ao participar de um jogo com algum parceiro. A realidade virtual tem potencial para os dois tipos de envolvimento ao permitir a exploração de um ambiente virtual e ao propiciar a interação do usuário com um mundo virtual dinâmico.

Além do hardware, os sistemas de realidade virtual necessitam de um software cuja composição é geralmente formada por três componentes: a componente de apresentação, de interação e de aquisição de dados.

A componente de apresentação é responsável pela representação, envolvendo aspectos de interface homem-computador e semiótica [37], carregamento do modelo e saída do sistema. Algumas aplicações definem um formato próprio de arquivo para o modelo, enquanto outras são capazes de ler vários modelos através de um esquema de cadeia de responsabilidade [16], delegando a leitura do arquivo para diferentes módulos (*plugins*). A componente de saída deve ser capaz de gerar dados para um ou uma combinação de dispositivos multi-sensoriais.

O componente de interação é responsável pela manipulação e navegação no ambiente. A dificuldade nesse caso é que, ao contrário dos ambientes convencionais, existem poucas metáforas de interação bem definidas para ambientes imersivos.

A aquisição dos dados é a componente mais complexa do sistema de RV. O sistema pode ser criado para funcionar especificamente com um determinado dispositivo de saída (um monitor, por exemplo) e um determinado dispositivo de entrada (um mouse 3-D) ou ser escalável e funcionar com qualquer composição de hardware. Nesse caso, a componente de aquisição de dados deve ser capaz de suportar a vasta gama de dispositivos, inclusive os que ainda não foram criados. Geralmente, os sistemas escaláveis utilizam bibliotecas ou *frameworks* que permitem que o desenvolvedor abstraia a aplicação dos dispositivos utilizados.

Os sistemas de realidade virtual são categorizados principalmente pela sua interface visual, pois a visão é o sentido que mais afeta as relações humanas. As principais categorias dos sistemas de RV são: baseado em monitor, parcialmente imersivos e imersivos.

Sistemas baseado em monitor

Alguns sistemas utilizam o monitor convencional do computador para visualizar o ambiente virtual em uma estação de trabalho tradicional. Um único usuário é rastreado através um sensor de posição preso à sua cabeça e vários dispositivos de entrada podem ser utilizados. A visão gerada pode ser monoscópica ou estereoscópica, seja via óculos 3-D, seja com um monitor auto-estéreo.

As vantagens desse sistema são:

- utilização de uma interface comum para o usuário: como o usuário não perde contato com o mundo real, ele pode utilizar o mouse, o teclado ou até mesmo um *joystick* para interagir;
- relativamente barato;
- mais de um usuário pode participar do ambiente, embora apenas um possa ser rastreado;
- o rastreamento do movimento do usuário limita-se à posição da sua cabeça e pode ser feito com uma simples câmera digital.

Entre as desvantagens estão:

- a falta de imersão: o campo de visão do usuário é limitado pelo monitor, que geralmente não passa de 21";
- violação da sensação de profundidade: os óculos 3-D são capazes de fornecer a sensação de profundidade até certo ponto. Quando os objetos presentes no ambiente virtual cruzam a borda do monitor, corre-se o risco de se perder a sensação de profundidade. Isto acontece porque um olho recebe a imagem atrás da borda, enquanto o outro pára de recebê-la.

Sistemas parcialmente imersivos

Os sistemas parcialmente imersivos são capazes de fornecer uma sensação de imersão maior do que os sistemas baseados em monitor, através do emprego de uma ou mais telas de projeção ou de vários monitores. As telas são maiores do que o monitor, permitindo maior mobilidade do usuário, e geralmente são projetadas por trás para evitar sombras geradas pelo próprio usuário.

Nesse tipo de sistema, o usuário começa a sair do mundo real e entrar no mundo virtual. Dessa forma, novos dispositivos de entrada, diferentes de mouse e teclado, devem ser empregados.

Como o campo de visão do usuário é maior do que nos sistemas baseados em monitor, mais informação visual será gerada, colocando em risco o desempenho da aplicação. Quanto mais imerso estiver o usuário, maior será o desconforto devido a atraso e latência do movimento.

As vantagens desse sistema são:

- maior campo de visão;
- não invasivo: diferente dos sistemas imersivos, vistos na seção seguinte, os semi-imersivos não agridem tanto o usuário, permitindo maior tempo de utilização;
- verdadeira grandeza dos objetos: algumas aplicações podem tirar proveito do tamanho das telas de projeção para exibir seus modelos em escala correta, permitindo melhor interpretação pelo usuário;
- permite um número maior de usuários simultâneos do que sistemas baseados em monitor.

As desvantagens são:

- violação da sensação de profundidade: existem dois tipos de violação da estereoscopia. O primeiro é o mesmo dos sistemas baseado em monitor. O segundo acontece quando o objeto está entre a mão do usuário e o seu olho. A mão deveria ser encoberta pelo objeto, mas de fato, isto não acontece e o objeto é oculto pela mão.

Sistemas imersivos

Os sistemas imersivos são aqueles que envolvem completamente o usuário. São exemplos de sistemas imersivos os capacetes de RV e a CAVE [11].

Os capacetes de RV são rastreados para fornecer ao sistema a rotação da cabeça do usuário enquanto o movimento de translação é controlado por um outro dispositivo, geralmente sem fio. Nos capacetes, a latência dos movimentos é crítica, pois pode causar desconforto maior do que os sistemas parcialmente imersivos.

Alguns benefícios dos capacetes de RV são:

- imersão completa;
- não há violação da sensação de profundidade: como não há superfície de projeção com bordas, as imagens estereoscópicas não são violadas;
- fácil configuração: a interface do capacete com o computador é igual a dos monitores convencionais, assim como a configuração no sistema operacional.

Algumas desvantagens:

- invasivo: o usuário é completamente removido do mundo real. Isso pode levar a cansaço físico e desconforto;
- mono-usuário: somente um usuário pode navegar no mundo virtual.

Nos sistemas imersivos do tipo CAVE, o usuário é completamente cercado por telas de projeção, som estéreo e o seu movimento de translação deve ser rastreado. Nesses ambientes, os dispositivos de entrada que o usuário utiliza devem ser sem fio para não atrapalhar a visualização do mundo virtual. Como nos sistemas parcialmente imersivos, a junção entre as telas não deve ser perceptível.

As imagens projetadas nas telas podem ser geradas por um único hardware ou por vários hardwares, um por tela, por exemplo. No último caso, um mecanismo de sincronização de quadros em hardware ou software [3] deve ser empregado.

São vantagens desse tipo de sistema:

- grande campo de visão: o usuário tem uma visão mais abrangente do mundo virtual;
- menos invasão: diferente dos capacetes, esse sistema é menos invasivo, pois o usuário ainda consegue ver seus movimentos;
- multi-usuários: vários usuários podem trabalhar juntos na mesma área.

Geralmente, somente um usuário é rastreado, sendo assim a perspectiva gerada não é apropriada para um outro usuário presente na mesma CAVE a não ser que os dois executem o mesmo movimento e estejam bem próximos um do outro.

Algumas desvantagens desse sistema são:

- espaço: esse tipo de sistema requer muito espaço para sua instalação;
- violação de oclusão: quando o objeto está entre a mão do usuário e o seu olho, a mão oculta o objeto, dando uma informação incorreta de profundidade;
- calibração: a fim de que todas as telas produzam um resultado visual com mesma quantidade de iluminação e as junções sejam imperceptíveis, os projetores devem ser sempre calibrados. A calibração dos projetores é um trabalho demorado e, dependendo da tecnologia empregada, pode ser muito caro e freqüente.

Dispositivos

Um das conseqüências do advento da realidade virtual foi a necessidade de se redefinir o paradigma de interface homem-computador. O sistema tradicional mouse-teclado-monitor foi substituído por dispositivos que permitissem maior imersão do usuário no ambiente virtual e o manuseio de todas as potencialidades dessa nova tecnologia.

Existem dois tipos de dispositivos: os de entrada e os de saída. De acordo com [37], os dispositivos de entrada estão divididos em:

- Rastreamento do corpo;
- Reconhecimento de voz e som;
- Controladores físicos.

Os dispositivos de saída, por sua vez, podem ser categorizados como:

- Visual;
- Auditivo;
- Táctil.

O presente trabalho não pretende analisar os dispositivos de reconhecimento de voz e som e os dispositivos de saídas auditivos e tácteis. Esses dispositivos fogem ao escopo desse trabalho.

Dispositivos de saída visual

A saída visual dos sistemas de realidade virtual, seja ele baseado em monitor, imersivo ou parcialmente imersivo, tem como principal preocupação a estereoscopia. Além disso, os sistemas que utilizam várias telas de projeção devem ser capazes de gerar as imagens na borda de maneira suave, ou seja, o usuário não deve perceber que naquele ponto existe uma junção.

Estereoscopia

A visualização em estéreo pode ser obtida utilizando-se uma solução ativa ou passiva. A solução passiva utiliza conceitos de óptica, como polarização, para separar as imagens dos olhos direito e esquerdo. Nesse modo, as duas imagens são geradas simultaneamente. Na solução ativa, as imagens são geradas alternadamente e uma combinação de óptica e eletrônica faz com que cada olho veja somente a sua imagem.

O efeito de estéreo passivo é obtido utilizando óculos de visualização que sejam capazes de separar duas imagens compostas em uma única imagem, podendo ser:

- Através da cor utilizando anaglifos: um anaglifo é uma figura obtida por dupla imagem, cada uma de um ponto diferente, impressa em duas cores contrastantes, que produzem, mediante o uso de óculos especiais, a ilusão de profundidade, de relevo;
- Através da cor por difração: a luz emitida pela imagem é refratada por micro-prismas existentes na lente dos óculos gerando a disparidade ocular;
- Polarização linear ou circular. A polarização linear é a mais comum pois provê uma boa separação se os polarizadores estiverem bem alinhados, porém é muito sensível à rotação da cabeça. A polarização circular é menos sensível à rotação da cabeça e produz uma boa separação, porém os óculos não são tão simples.

Outra solução de estéreo passivo são os monitores auto-estéreo [20]. Esses monitores possuem um padrão de iluminação e um sistema óptico especial que fica atrás da sua tela de cristal líquido. Esse sistema faz com que as colunas de *pixels* sejam alternadamente visíveis para os olhos esquerdo e direito quando o usuário está posicionado a uma determinada distância em frente ao monitor.

Outro tipo de estereoscopia, a ativa, apresenta as imagens do olho esquerdo e direito alternadamente no monitor ou na tela de projeção e uma solução de óptica e eletrônica é responsável por direcionar a imagem correta para cada olho. A solução mais comum são os óculos de abertura (*shutter glasses*) (Figura 10) juntamente com estéreo ativo progressivo (*page flipping*).

Figura 10: Óculos de abertura (*shutter glasses*) e emissores infravermelho.

O estéreo ativo progressivo envia um quadro de cada olho alternado para a placa de vídeo. As placas possuem uma saída, além da de vídeo, que permite a sincronização dos óculos de abertura (Figura 11). O sincronismo pode ser com fio ou sem fio, utilizando transmissão por infravermelho. Em cada quadro, o emissor de infravermelho comanda a abertura de uma das lentes dos óculos. Para que o cérebro humano consiga juntar as imagens e interpretá-las como sendo uma, a freqüência de apresentação de cada imagem para cada olho deve ser de 50 a 60 Hz. Alguns exemplos de placas gráficas que possuem o recurso de estéreo ativo progressivo são as Quadros da NVidia, as FireGLs da ATI e a WildCat da 3DLabs.

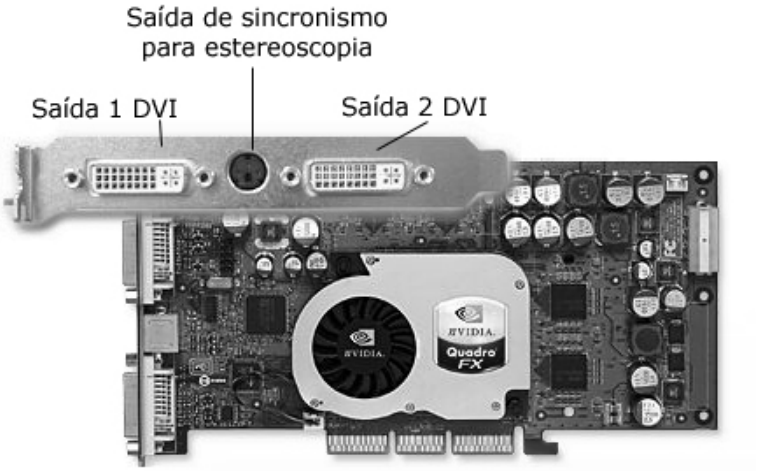

Figura 11: Placa Quadro FX: solução de estereoscopia ativa progressiva da NVidia.

Os óculos ativos são muito mais caros que os óculos passivos. A qualidade do estéreo passivo é boa, porém é muito dependente da qualidade da tela de projeção e do movimento da cabeça do usuário, enquanto o estéreo ativo não requer tela especial. Ambos os tipos de estéreo apresentam problemas de fantasmas. Ou seja, a lente que está fechada (ou com polarização trocada) não consegue bloquear completamente a imagem que deveria ser vista somente através da outra lente. Os óculos passivos são ideais para ambientes claros e de grande audiência, enquanto os ativos são ideais para pequenos grupos em salas menores, com pouca luminosidade.

Capacetes de RV

Os capacetes de RV (Figura 12i) são um exemplo de dispositivos de RV usados sobre a cabeça do usuário. Eles são formados por dois visores, um para cada olho do usuário. Outros exemplos de dispositivos desse gênero são capacetes com somente um visor (sem visão estereoscópica - Figura 12ii), binóculos (Figura 12iii) e monóculos (Figura 12iv). Um exemplo de emprego de um binóculo de realidade virtual é o projeto Visorama [27], cuja finalidade é a visualização de imagens panorâmicas.

Figura 12: Exemplos de dispositivos de RV: (i) capacete de RV estéreo, (ii) capacete de RV mono, (iii) binóculos e (iv) monóculos.

Os capacetes de RV são compostos por pequenos CRTs (tubos de raios catódicos) ou LCDs (telas de cristal líquido). Geralmente a entrada é feita por duas portas VGA. Para isso, a placa de vídeo do computador deve possuir duas saídas de vídeo sincronizadas e com a mesma resolução de imagem. É possível ainda duplicar a imagem de um visor no outro, porém a imagem final não irá produzir o efeito de estereoscopia. Se a placa de vídeo for capaz de gerar quadros estéreos intercalados, alguns capacetes são capazes de direcionar corretamente as imagens para cada olho, provendo visão estéreo com somente uma entrada de vídeo. Um exemplo desse tipo de capacete é o HMD 800 da empresa 5DT – Fifth Dimension Technologies.

Projetores

Os projetores desempenham um papel crucial nos sistemas de RV imersivos e parcialmente imersivos. O brilho e a resolução são fatores que melhoram a qualidade da imagem, logo aumentam a sensação de imersão. Alguns projetores possuem a capacidade de gerar imagens para estereoscopia ativa e existem ainda aparelhos que permitem a utilização de dois projetores para geração de estereoscopia passiva. Atualmente, existem três tipos de projetores disponíveis: CRT, LCD e DLP.

Os projetores **CRT**, tubos de raios catódicos, são os mais antigos e acompanham o advento da televisão. O projetor CRT é composto por três tubos responsáveis pela geração das três componentes primárias, vermelha, verde e azul. A imagem é formada nos três tubos e projetada, através de lentes, na tela de projeção.

A projeção através de telas de cristal líquido, **LCD**, tornou-se realidade no final da década de 1980, quando novas formas de painéis de matriz ativa, componente principal das telas de cristal líquido, foram desenvolvidas para os computadores portáteis. Os painéis de matriz ativa permitem que uma determinada linha e coluna da tela de cristal líquido sejam endereçadas e recebam uma carga capaz de ser mantida até o próximo ciclo de atualização da tela. Com essa carga é possível controlar a intensidade de luz que irá passar por aquele *pixel*. Ao fazer isso em pequenos e exatos incrementos, é possível criar uma escala de cinza. Os LCDs atuais possuem 256 níveis de escala de cinza. A resolução do LCD é determinada pela quantidade de linhas e colunas existentes na tela, sendo as mais comuns os de 800x600 e 1024x768 linhas por coluna.

Internamente, o projetor possui uma luz branca intensa que é dividida em três feixes que passam por 3 LCDs para gerarem as componentes, vermelha, verde e azul, da imagem. Os feixes resultantes são combinados e uma única imagem é projetada.

A tecnologia **DLP** é uma solução de projetores de vídeo que utiliza semicondutores ópticos para manipulação digital da luz. A geração da imagem pode ser explicada em três fases: o semicondutor, a imagem cinza e a adição de cor.

O integrado do semicondutor contém uma malha retangular com centenas de milhares de espelhos microscópicos. Cada um desses micro-espelhos mede menos que um quinto da largura de um fio de cabelo humano e correspondem a um *pixel* da imagem projetada. O semicondutor é usado conjuntamente com uma fonte de luz branca, lentes de projeção e o sinal gráfico digital.

Os micro-espelhos são montados em minúsculas dobradiças que permitem que eles girem em direção a fonte de luz ou em direção contrária a ela, correspondendo a um *pixel* claro e um escuro na tela de projeção.

O sinal digital correspondente à imagem ao chegar ao semicondutor, faz com que cada espelho seja ‘ligado’ e ‘desligado’ milhares de vezes por segundo. Quando um espelho é ‘ligado’ com mais freqüência, um pixel mais claro é produzido. Ao contrário, se o espelho for desligado com mais freqüência, o pixel gerado será mais escuro. Dessa forma, os projetores DLP podem gerar uma escala de 1024 níveis de cinza.

A luz branca, antes de alcançar o semicondutor, passa por um disco colorido. Esse disco filtra a luz gerando as componentes vermelha, verde e azul. O estado ‘ligado’ e ‘desligado’ de cada micro-espelho é sincronizado com as três componentes para produzir até 16,7 milhões de cores. Por exemplo, o espelho responsável por projetar um *pixel* roxo irá refletir as luzes vermelha e azul. O olho humano irá combinar as luzes intercaladas rapidamente e interpretará o resultado como sendo a cor matiz da imagem projetada.

A Figura 13 mostra os três tipos de projetores e a Tabela 1 apresenta as vantagens e desvantagens de cada tecnologia.

Figura 13: Projetores da empresa BARCO: (a) CRT, (b) LCD e (c) DLP.

Tecnologia	Vantagens	Desvantagens
CRT	• Excelente qualidade de imagem; • Excelente nível de cinza; • Os *pixels* não são visíveis; • Os tubos possuem vida longa; • Pouco ruidosos; • Melhor custo/benefício;	• Grandes e pesados; • Os tubos são caros; • Pouco brilho, exigindo uma sala com pouca iluminação; • Exigem ajustes periódicos; • Difícil configuração; • Exigem reconfiguração ao alterar o tamanho da tela;
LCD	• Pequenos e luminosos; • Rápida configuração; • Não há perda de alinhamento; • Lâmpadas de fácil substituição; • Muito mais brilho;	• Ruidosos; • As lâmpadas têm tempo de vida pequeno; • Lâmpadas caras; • É possível notar os *pixels*; • Tons de cores deficientes; • Escala de cinza deficiente;
DLP	• Pequenos e luminosos; • Rápida configuração; • Não há perda de alinhamento; • Lâmpadas de fácil substituição; • Muito mais brilho; • Tons de cores mais vivos; • Os *pixels* não são visíveis; • Não requerem uma sala escura;	• Ruidosos; • As lâmpadas têm tempo de vida pequeno; • Lâmpadas caras; • Escala de cinza deficiente;

Tabela 1: Vantagens e desvantagens dos projetores CRT, LCD e DLP

Workbenchs e a CAVE

Com o emprego de projetores e múltiplas telas de projeção é possível criar as mais variadas configurações de ambientes de visualização para RV. Nesta seção vamos analisar duas configurações que mais influenciaram no desenvolvimento de aplicações de RV: a *Responsive Workbench* e a *CAVE*.

A *Responsive Workbench* (Figura 14) é um espaço de trabalho interativo e tridimensional. Imagens estereoscópicas são projetadas em um topo de mesa horizontal que serve como superfície de visualização. A projeção é feita através de um sistema de projetor e espelhos e a visualização é feita através de óculos 3-D para estéreo ativo.

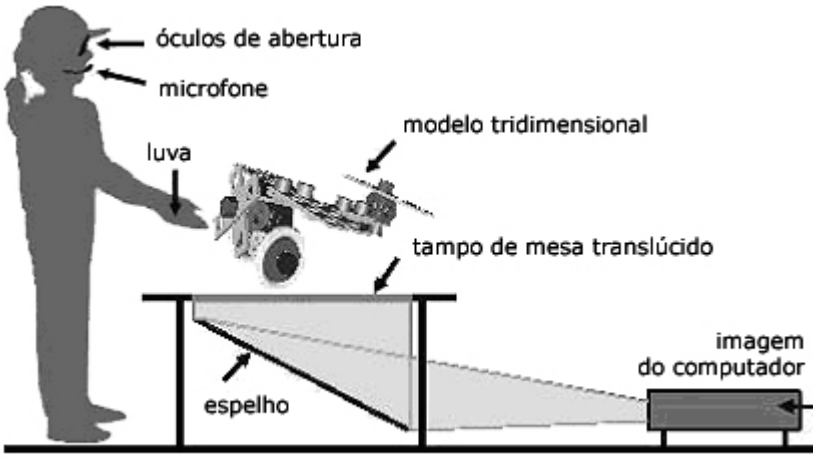

Figura 14: Configuração da *Responsive Workbench*.

O movimento da cabeça do usuário é acompanhado utilizando um sistema de rastreamento com 6 graus de liberdade. Dessa forma, o usuário pode ver o ambiente virtual através do ponto de vista correto, porém a utilização fica restrita a somente um usuário. Em [1] é apresentada uma solução para que dois usuários possam utilizar a *workbench*. A Figura 15, extraída de [1], apresenta a problemática de dois usuários visualizando o mesmo objeto, de pontos de vistas diferentes, sendo que somente um deles está sendo rastreado.

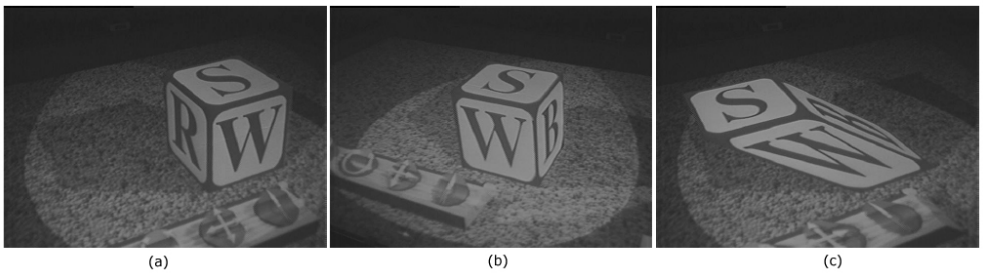

Figura 15: (a) A imagem vista por um usuário do lado esquerdo da mesa. (b) a mesma imagem vista do lado direito. (c) A imagem vista por um usuário à esquerda da mesa se a mesma fosse gerada do ponto de vista da direita.

A solução empregada consiste na geração seqüencial de 4 imagens: L1 L2 R1 R2, onde L1 e R1 são as vistas do olho esquerdo e direito do primeiro usuário e L2 e R2 as mesmas vistas para o segundo usuário. Para que o primeiro usuário não veja a imagem do segundo, os óculos 3-D devem ser capazes de fechar as duas lentes simultaneamente.

A CAVE é um sistema de RV baseado em projeção desenvolvido inicialmente no Electronic Visualization Lab da Universidade de Illinois [11]. Ela é uma sala cujas paredes, o teto e o chão são superfícies de projeção (Figura 16).

A implementação da CAVE requer que as projeções estejam sincronizadas e as bordas das imagens sejam combinadas, para que a junção seja imperceptível. A geração da perspectiva do usuário em uma CAVE não é um problema simples e deve ser tratado pela aplicação. A perspectiva é calculada com base na posição do usuário, a rotação de sua cabeça e a tela de projeção.

A partir do lançamento da CAVE em 1992, várias configurações de telas de projeção foram sendo criadas e batizadas com outros nomes, como ImmersaDesk, IWall, PowerWall, Holobench e Holospace.

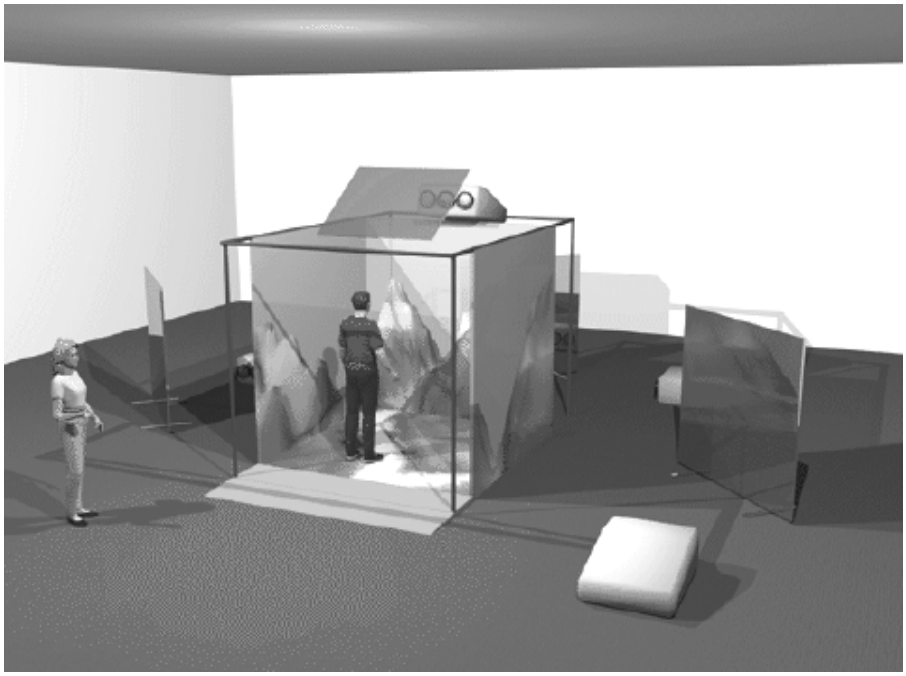

Figura 16: Ilustração de uma CAVE.

Dispositivos de rastreamento

O modo como os participantes interagem com o sistema de realidade virtual influencia enormemente suas experiências no ambiente virtual. Os modos de interação afetam a facilidade de uso do sistema, a sensação de imersão do usuário e a variedade de ações que ele pode tomar dentro do ambiente virtual.

Em realidade virtual, um importante dispositivo de entrada é o rastreador de posição que pode ser utilizado para acompanhar a posição do corpo e os movimentos do usuário, assim como a posição de outros objetos que estão sendo por ele utilizados.

Existe uma variedade de dispositivos de rastreamento, cada um utilizando uma tecnologia diferente, entre eles, os eletromagnéticos, mecânicos, acústicos e inerciais e ópticos. Ao analisar as tecnologias utilizadas pelos rastreadores, três fatores devem ser levados em consideração: precisão e velocidade de resposta do sensor; interferência do meio; restrições (fios, conexões mecânicas, etc.):

- **Eletromagnéticos**

- Princípio de Funcionamento: os rastreadores eletromagnéticos utilizam campos magnéticos para medir posição e orientação. O sistema é composto por transmissor e receptor em forma de bobina. Um sensor unidimensional para estimar a posição no eixo Z, por exemplo, é composto por uma única bobina transmissora orientada na direção Z. Quando uma corrente é aplicada à bobina, um campo magnético é gerado. No receptor, o campo induz uma voltagem máxima proporcional à intensidade do campo magnético medido em uma bobina orientada na mesma direção do campo.

A voltagem induzida fornece a distância do transmissor ao receptor, assim como a diferença de alinhamento entre os eixos.

- Precisão/Velocidade: esses sistemas são bastante precisos, cerca de 1 a 2 mm para posição e 0.1º para orientação. A velocidade de captura de dados é de 100 a 200 medidas/segundo.
- Interferência do meio: a presença de metal pode causar interferência eletromagnética.
- Restrições: pequeno espaço de utilização devido ao alcance do campo magnético gerado. O receptor deve estar cerca de 1-3 metros do transmissor, não havendo necessidade de linha de visada desobstruída.
- Exemplos: FasTrack da Polhemus (Figura 17i) e o Flock of Birds da Ascension (Figura 17ii).

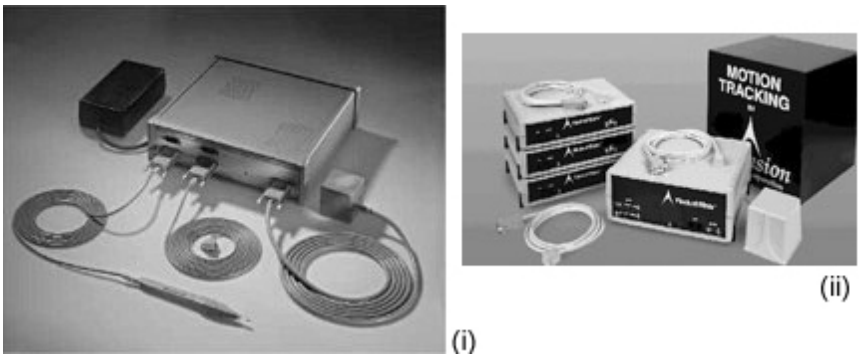

Figura 17: (i) FasTrack e (ii) Flock of Birds.

➢ **Mecânicos**

- Princípio de Funcionamento: os rastreadores mecânicos medem ângulos e distância entre juntas. Dada uma posição conhecida, todas as outras podem ser determinadas pela relação entre as juntas. Os rastreadores podem estar presos ao chão ou anexos ao corpo do usuário, usualmente na forma de um exoesqueleto. As rotações e as distâncias podem ser medidas por engrenagens, potenciômetros ou sensores de dobra (Figura 18).

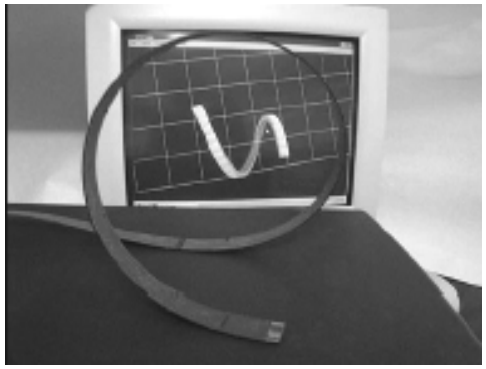

Figura 18: Sensor de dobra.

A vantagem dos sistemas mecânicos é a facilidade da adição da funcionalidade de *force feedback*. Esse tipo de dispositivo restringe o movimento do usuário aplicando uma força contrária ao seu movimento.

- Precisão/Velocidade: por serem mecânicos, possuem alta precisão (0.1º de rotação). A latência média é de 200 ms.
- Interferência do meio: não sofrem interferência do meio.
- Restrições: a própria arquitetura do rastreador pode restringir o movimento do usuário, caso o mesmo seja preso ao chão ou possua muitas juntas.
- Exemplos: o Phantom (Figura 19i) da Sensable Technologies e a mão mecânica da EXOS (Figura 19ii).

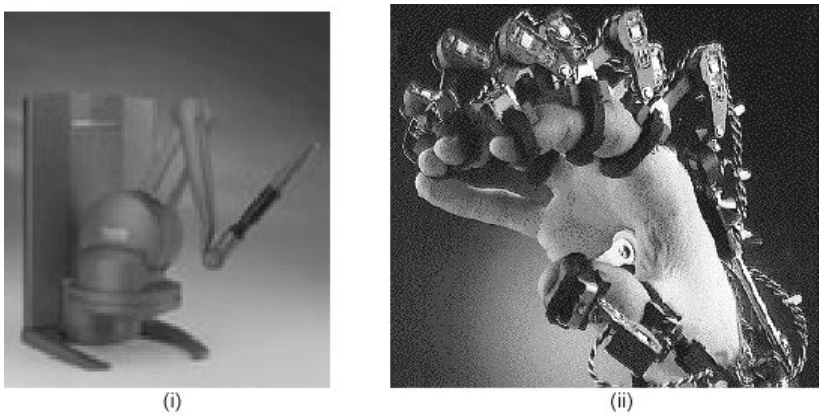

Figura 19: (i) Phantom da Sensable e a mão mecânica da EXOS.

➢ **Acústicos**

- Princípio de Funcionamento: rastreadores acústicos utilizam, tipicamente, ondas sonoras ultra-sônicas para medir distância. Os métodos mais usados são o cálculo do tempo de vôo e coerência de fase. Em ambos, o objetivo é converter tempo em distância. Um único par transmissor/receptor fornece a distância do objeto em relação a um ponto fixo.

  O resultado é uma esfera em cuja superfície o objeto está localizado. Como visto na Figura 20, a adição de um segundo receptor restringe a região a um círculo e um terceiro receptor restringe a dois pontos, sendo que um deles é facilmente descartado. Portanto, para estimar a posição são necessários um transmissor e três receptores ou um receptor e três transmissores. Para estimar posição e orientação, são necessários três transmissores e três receptores.

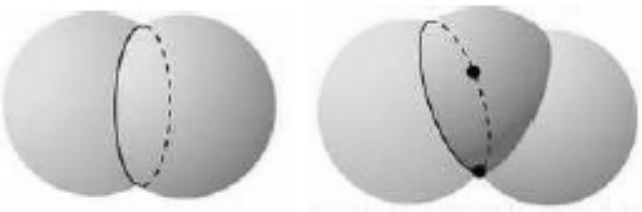

Figura 20: Interseção entre duas esferas (um círculo) e entres três (dois pontos).

- Precisão/Velocidade: existe um atraso inerente a espera do sinal. Esse atraso é intensificado devido à baixa velocidade de propagação do som.
- Interferência do meio: as propriedades do som limitam esse método. O desempenho é degradado na presença de um ambiente ruidoso. O som

deve percorrer um caminho sem obstrução entre os autofalantes e os microfones.

- Restrições: a configuração do sistema não é cara, pois o equipamento necessário é composto de microfones, alto-falantes e um computador.

  Devido às restrições de interferência, a distância média entre receptor e transmissor são alguns metros, contudo, sistemas mais precisos podem cobrir áreas de até 40x30m.
- Exemplos: Logitech Tracker (Figura 21i) com alcance médio de 15 metros e o FarReach da Infusion Systems (Figura 21ii) com alcance de 12 metros.

Figura 21: (i) Logitech Tracker e (ii) FarReach.

➢ **Inerciais**

- Princípio de Funcionamento: utilizam magnetômetros passivos, acelerômetros e girômetros. Os magnetômetros passivos medem o campo magnético do ambiente (geralmente da Terra) e fornecem medidas angulares. Os girômetros fornecem medida angular mais precisa e os acelerômetros fornecem medidas lineares. Todos são baseados na segunda lei de movimento de Newton, $\boldsymbol{F = ma}$ e $\boldsymbol{M = I\alpha}$, sendo assim, o sistema deve integrar a leitura para obter a velocidade e a posição.
- Precisão/Velocidade: Devido à etapa de integração, o erro obtido a cada integração tende a aumentar. A utilização de filtros e outros sensores ajuda a diminuir esse erro.

  A vantagem desses dispositivos é a sua precisão, sendo capazes de alcançar resolução angular de até 0.2º em determinadas condições de utilização.
- Interferência do meio: Não existe interferência, pois o sistema é autocontido, não havendo necessidade de um ponto externo para obtenção de dados;
- Restrições: Não existe limitação física para o espaço de trabalho, sendo o mesmo limitado somente pela conexão entre o dispositivo e o computador.
- Exemplos: 3D-Bird da Ascension Technology (Figura 22i) e o Intertrax2 da InterSense (Figura 22ii).

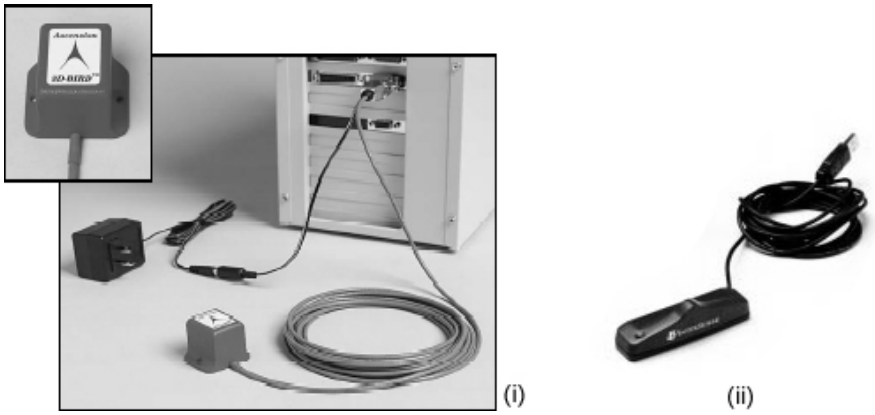

Figura 22: (i) 3D-Bird e (ii) Intertrax2.

- **Ópticos**

- Princípio de Funcionamento: sistemas ópticos utilizam informação visual para rastrear o usuário. O método mais comum é obter a imagem através de uma câmera de vídeo e empregar técnicas de visão computacional para determinar a posição do objeto.

  Quando somente um sensor (câmera) é utilizado, a posição do ponto observado é representada por uma reta que passa pelo ponto e pelo centro de projeção. Observando mais de um ponto ou usando mais de um sensor, é possível determinar corretamente a posição e orientação do objeto sendo rastreado.

  Outro equipamento óptico bastante utilizado é o composto por emissores de luz, como os diodos (LEDs) ou laser.
- Precisão/Velocidade: a velocidade de captura depende muito do sensor empregado. Uma câmera padrão NTSC consegue capturar imagens a taxas de 30 quadros por segundo, limitando a amostragem. A precisão dos dados depende das técnicas de visão computacional empregadas: calibração de câmera, extração de informação da imagem e utilização de filtro para evitar tremido.
- Interferência do meio: Não sofre interferência do meio.
- Restrições: A câmera deve estar sempre enxergando o objeto sendo rastreado ou o emissor de luz não pode ser obstruído.
- Exemplos: Existem muitos equipamentos ópticos utilizados em captura de movimento e em realidade virtual. Alguns exemplos de dispositivos usados em RV são o DragonFly [39] (Figura 23) e o LaserBird da Ascension Technology (Figura 24).

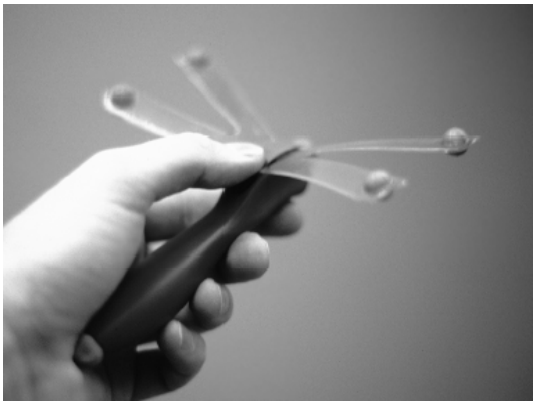

Figura 23: Dispositivo óptico DragonFly desenvolvido pelo Instituto Fraunhofer de Engenharia Industrial da Alemanha.

Figura 24: LaserBird da Ascescion Technology: sensor e feixe de laser.

Controladores físicos

Além dos dispositivos de rastreamento, os controladores físicos complementam as formas de interação do usuário com o sistema de RV. Esses controladores variam desde simples dispositivos, como o *joystick*, a dispositivos mais complexos, como um simulador de cabine de avião. Os controladores podem servir para aplicações específicas ou serem genéricos e, portanto, utilizáveis em qualquer aplicação.

Os controladores físicos podem fornecer três tipos de informação para o sistema de RV: analógica, digital e posicional.

A informação analógica é um valor contínuo fornecido pelo dispositivo. Por exemplo, um *joystick* possui uma leitura analógica quando sua haste é movimentada. A informação digital é do tipo 0 ou 1. Um exemplo de controlador digital são os botões do mouse.

A informação posicional é obtida através dos dados vindos do dispositivo ou da conversão da informação analógica. O resultado é composto por 3 coordenadas de translação e pelos 3 ângulos de Euler, ou por um quatérnio. Um dispositivo óptico, por exemplo, é capaz de fornecer a posição e orientação de um objeto sendo rastreado. Um exemplo de dispositivo óptico capaz de fornecer informação com 6 graus de liberdade é o Dragonfly [39].

Vários dispositivos podem ser agrupados para formar um único controlador físico. Um *joystick* é um exemplo de dispositivo que fornece informação analógica e digital.

Realidade Aumentada

A Realidade Aumentada (RA) é uma variação da Realidade Virtual (RV). Em RV, o usuário é imerso em um ambiente sintético e não participa do mundo real a sua volta. A RA permite que o usuário veja o mundo real com objetos virtuais superpostos ou combinados com ele. Portanto, a RA suplementa a realidade, ao invés de substituí-la completamente. Para o usuário, os objetos reais e os virtuais coexistem no mesmo espaço. A Figura 25 mostra um exemplo onde objetos reais e virtuais compõem um ambiente.

Figura 25: Informação virtual sobre o mundo real como auxílio a navegação

Fonte: How Stuff Works (http://www.howstuffworks.com).

Além da imagem virtual sobreposta à imagem real, os sistemas de RA podem aumentar o mundo real através da inclusão de som, resposta ao tato e cheiro. Essas tecnologias ainda estão em fase de pesquisa embrionária.

Os componentes mais utilizados nesse tipo de ambiente são os capacetes de RA e os equipamentos de rastreamento. Os ambientes de RA também podem ser executados em sistemas baseados em monitor e de projeção, através da combinação da imagem virtual e da imagem real capturada por uma câmera de vídeo.

Os capacetes de RA podem combinar a imagem virtual e real de duas maneiras: óptica e vídeo. Em ambos os casos, o sistema deve alinhar os objetos virtuais com os reais. Esse processo chama-se registro e é um dos problemas mais difíceis de se resolver em RA [7].

Tecnologia Óptica e de Vídeo

A combinação da imagem real com a imagem virtual pode ser feita de duas maneiras: tecnologia óptica e de vídeo. Cada uma possui suas vantagens e desvantagens. Esta seção apresenta as características de cada tecnologia e aponta suas diferenças. Em [36], encontra-se mais informações sobre o assunto.

Os dispositivos mais empregados em sistemas de RA são os visores montados sobre a cabeça do usuário, sejam capacetes, monóculos ou outros. Chamaremos, de forma genérica, esses dispositivos de capacete de RA.

Os capacetes de RA com tecnologia óptica funcionam através da colocação de combinadores ópticos na frente do olho do usuário. Esses combinadores são translúcidos, de tal maneira que o usuário consiga enxergar o mundo real através dele. Eles também são parcialmente reflexivos para que o usuário possa ver imagens virtuais geradas por saídas de vídeo acopladas no

capacete e refletidas nos combinadores. A Figura 26 apresenta um diagrama conceitual da tecnologia óptica.

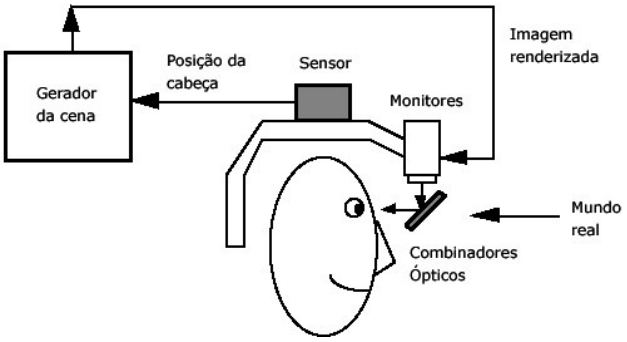

Figura 26: Capacete de RA com tecnologia óptica.

Os capacetes de RA com vídeo funcionam através da combinação de um capacete fechado (como um capacete de RV) e uma ou duas câmeras de vídeo montadas na cabeça do usuário. As câmeras de vídeo irão prover ao usuário a visão do mundo real. A imagem das câmeras é combinada com o gráfico virtual e o resultado é enviado para o visor. A Figura 27 ilustra a tecnologia de vídeo.

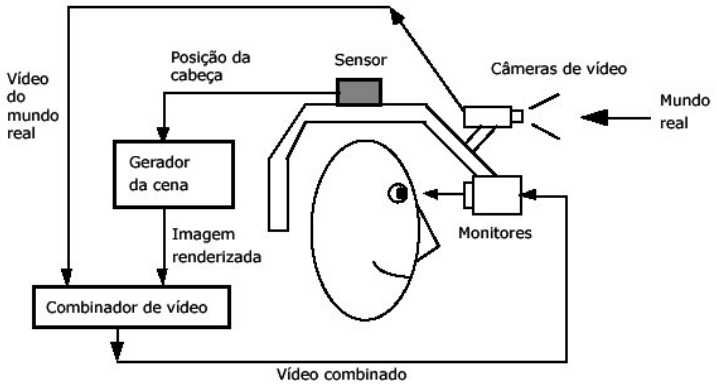

Figura 27: Capacete de RA com tecnologia de vídeo.

A combinação da imagem real com a imagem gráfica pode ser feita através de *chroma-keying* ou baseada na informação de profundidade. *Chroma-keying* consiste na substituição do pano de fundo de uma imagem por outra imagem. O pano de fundo é formado por uma única cor, digamos verde, que nenhum objeto virtual utiliza. A combinação substitui a cor verde da imagem virtual pela imagem do mundo real.

Uma composição mais sofisticada consiste na utilização da informação de profundidade da cena real. Se o sistema possuir a informação de profundidade de cada *pixel* da imagem real, é possível comparar cada *pixel* da imagem real com os da imagem virtual e superpor os objetos virtuais com os objetos reais. Outra maneira de conseguir esse resultado é através da posse do modelo tridimensional da cena. Outras técnicas de oclusão em ambientes de realidade aumentada são apresentadas em [15].

Os ambientes de RA também podem ser baseados em monitor ou utilizar sistemas de projeção, ao invés do capacete de RA. A Figura 28 mostra a configuração desse tipo de sistema. Uma ou mais câmeras, estáticas ou móveis, filmam o ambiente real. No caso móvel, a câmera pode ser presa a um robô cuja

posição está sendo rastreada. A imagem de vídeo do mundo real e a imagem virtual são combinadas da mesma forma dos capacetes de RA com tecnologia de vídeo, porém a imagem resultante é apresentada no monitor ou nas telas de projeção.

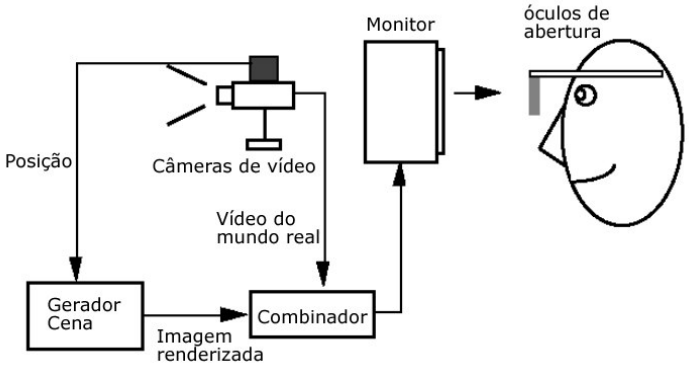

Figura 28: Sistema de RA baseado em monitor.

Ao comparar os três sistemas de RA, as seguintes características devem ser analisadas [7]:

- **Simplicidade**: O sistema óptico possui apenas um fluxo de dados, o virtual, enquanto o sistema de vídeo deve tratar da captura da imagem de vídeo e da geração da imagem virtual. O sistema de vídeo adiciona um atraso na visualização da imagem real na ordem de dezenas de milisegundos. A imagem de vídeo deve estar bem sincronizada com a imagem virtual para que não ocorra distorção temporal. O sistema de vídeo apresenta ainda uma distorção proveniente da câmera que deve ser corrigida. Os sistemas ópticos também podem apresentar distorção dependendo das lentes utilizadas, porém são bem menos perceptíveis.
- **Resolução**: A resolução do sistema de vídeo é limitada pela resolução da câmera e do visor, ou seja, nos sistemas de vídeo, o mundo real é visto com a resolução da câmera, enquanto que nos sistemas ópticos o mundo é visto como ele é. No sistema óptico existe uma perda de iluminação devido ao fator de reflexão existente no combinador. Dessa forma, quando desligado, o capacete de RA óptico funciona como óculos escuros.
- **Segurança**: O sistema de vídeo substitui a visão do usuário pela imagem capturada. Se em algum momento o sistema falhar, por falta de energia, por exemplo, o usuário perde completamente a visão. Esse problema não ocorre no sistema óptico.
- **Distância ocular**: No sistema óptico a distância ocular é a verdadeira distância entre os olhos do observador. No sistema de vídeo, a distância ocular é determinada pela distância entre as duas câmeras de vídeo utilizadas. Se elas não estiverem na mesma posição dos olhos do observador, uma disparidade de posicionamento irá surgir.
- **Técnicas de composição**: Um problema no sistema óptico é o dos objetos virtuais estarem sempre à frente dos objetos reais. Uma solução para isso seria um dispositivo capaz de bloquear a luz vinda do mundo real, porém é uma solução muito complexa. Ao contrário, os sistemas de vídeo

combinam a imagem real e a virtual em um processo controlado, permitindo que diversas técnicas sejam utilizadas, como o emprego da informação de profundidade da cena para ocultar os objetos virtuais. Outro problema é o foco, pois a imagem virtual gerada no sistema óptico sempre está no foco do olho do observador, o que nem sempre é verdade, pois o mesmo pode estar olhando para outra parte da cena real. Para resolver esse tipo de problema, um sistema de rastreamento do movimento da pupila do observador deverá ser empregado.

- **Atraso**: A tecnologia de vídeo permite reduzir discordâncias temporais entre o mundo real e a imagem virtual. O sistema óptico apresenta a imagem real instantaneamente, porém a imagem virtual possui um atraso. Esse atraso pode ser adicionado à imagem real no sistema de vídeo, reduzindo o erro visual.
- **Registro**: Nos sistemas ópticos, a única informação de posição é o rastreador que acompanha o movimento da cabeça do usuário. No sistema de vídeo, técnicas de visão computacional permitem recuperar a posição e orientação da câmera.
- **Brilho e contraste**: O sistema de vídeo pode ajustar o brilho e o contraste da imagem virtual para que os mesmos estejam de acordo com a imagem virtual. No sistema óptico o controle só é feito na imagem virtual.

Registro

O registro de imagens consiste basicamente na determinação dos parâmetros de uma função de mapeamento que relaciona duas imagens da mesma cena. O objetivo do registro é ressaltar ou eliminar diferenças de interesse entre essas imagens, geralmente para fins de comparação ou alinhamento das imagens. As imagens podem ser tomadas em diferentes tempos, de diferentes sensores, ou de diferentes pontos focais.

O registro de imagens é necessário como etapa preliminar em um grande número situações relacionadas ao processamento de imagens e visão computacional, tais como: reconhecimento de padrões, sensoriamento remoto, visão robótica, diagnóstico médico através de imagens, realidade aumentada, reconstrução de volumes, etc.

Os erros de registro são difíceis de se controlar adequadamente devido aos requisitos de precisão e da numerosa quantidade de fontes erros. Os erros podem ser estáticos e dinâmicos. Erros estáticos são aqueles que causam erros de registro mesmo quando o ponto de vista do usuário e os objetos do ambiente estão completamente parados. Os erros dinâmicos são aqueles que só aparecem quando o usuário ou os objetos se movem.

As principais causas de erro estático são: distorção óptica, erros no sistema de rastreamento, desalinhamento mecânico (dos combinadores ópticos, por exemplo) e erros nos parâmetros de visualização (centro de projeção, dimensões da janela, diferença entre a posição e orientação do rastreador e da câmera, campo de visão).

Os erros dinâmicos ocorrem devido a atrasos no sistema. O atraso é medido desde o instante no qual foi registrada uma medida do rastreador até o instante em que a imagem final chegou aos olhos do usuário. O erro dinâmico é percebido quando o usuário mexe a cabeça e a cena virtual fica parada,

começando a se mover alguns instantes depois. Esse erro pode causar bastante desconforto para o usuário dos sistemas de RA e deve ser corrigido. As maneiras de amenizar o erro são: reduzir o atraso diretamente, através de técnicas de otimização; prever o movimento do usuário, renderizando a cena através da posição prevista ao invés da posição medida [8]; ou sincronizar os quadros da imagem real e virtual através de sistemas de vídeo.

Aplicação

A Realidade Aumentada e a Realidade Virtual são tecnologias que vêm sendo empregadas nas mais diversas áreas.

Na medicina, através da simulação cirúrgica, visualização de imagens de ultra-som sobre o corpo do paciente, estudo de anatomia, entre outros. A Figura 29 ilustra a visualização em tempo real de uma biopsia em uma mama.

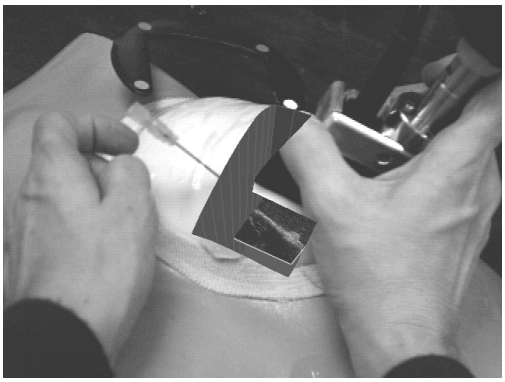

Figura 29: Visualização em tempo real de uma biopsia em uma mama.
Fonte: Comp. Science Dept., Univ. of North Carolina.

Na área de manutenção e construção, marcadores podem ser anexados a objetos onde as pessoas estão trabalhando e o sistema de realidade aumentada pode mostrar as instruções de manutenção.

Os militares têm empregado a realidade virtual e a realidade aumentada no intuito de prover à tropa informações vitais sobre seus arredores. Vários exemplos podem ser citados: construções inimigas podem ser marcadas virtualmente através de dados obtidos pela inteligência; regiões minadas podem ser rastreadas e identificadores virtuais colocados sobre o solo; a simulação de um tiro de artilharia pode ser executada e a região atingida pode apresentar uma explosão virtual, juntamente com os dados de correção do tiro; movimentos de tropas inimigas que estão camufladas podem ser previstos e apresentados na forma virtual; apresentação do mapa de batalha virtual (Figura 30), simuladores de vôo, entre outros.

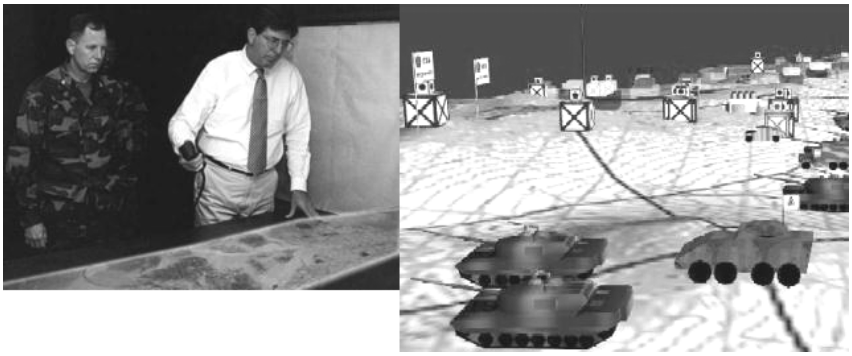

Figura 30: Visualização do campo de batalha em uma *Workbench*.
Fonte: US Naval Research Laboratory.

A RA pode ainda ser empregada na área de educação e informação. Eventos históricos podem ser recriados em ambientes de sala de aula e museu, permitindo que crianças e adultos possam participar da História.

Há também, obviamente, aplicações de RV e RA na área de entretenimento, onde personagens de jogos de aventura podem aparecer sobre uma mesa [42] ou, com o auxílio de equipamentos portáteis, o jogador pode sair pela rua perseguindo inimigos, como no jogo ARQuake [33].

Os sistemas de realidade virtual vieram para revolucionar a forma como nós interagimos com sistemas complexos em computador. As aplicações são muitas e é difícil predizer onde os ganhos e os benefícios da Realidade Virtual serão mais significativos. O certo é que não haverá um único padrão nas interfaces de RV. A tecnologia e as limitações de custos farão com que o tipo de aplicação defina o nível de sofisticação da tecnologia a ser aplicada.

Neste cenário, muitas aplicações, na busca de soluções para problemas específicos, acabarão por gerar novos usos e soluções para problemas de outras áreas.

Ferramentas

Os sistemas de RV precisam de um software para apresentar aplicação para o usuário final. Os softwares de RV são complexos, pois os dispositivos de RV são complexos. A fim de facilitar a criação de aplicações de RV varias soluções foram sendo criadas com o intuito de prover um ambiente de desenvolvimento para RV. Um ambiente de desenvolvimento permite que o usuário se concentre no desenvolvimento da aplicação de RV ao invés de se preocupar na gerência do ambiente de RV.

O ambiente de desenvolvimento de RV define uma arquitetura que inclui componentes para gerência de dispositivos de entrada, apresentação das saídas visual, auditiva e outras, além de um mecanismo de configuração. Alguns ambientes ainda estendem a arquitetura a fim de gerenciar também recursos como, alocação de memória, multi-tarefas e comunicação.

Existem ainda ambientes de desenvolvimento de RA cuja finalidade é abstrair a etapa de calibração da câmera e reconhecimento de padrões através do fornecimento de uma matriz de transformação que irá posicionar corretamente os objetos do mundo virtual sobre o ambiente real.

Várias ferramentas para desenvolvimento de aplicações em RV e RA existem disponíveis comercialmente ou gratuitamente, entre elas, o Avango [41], Vess [12], Diverse [5], VRJuggler [9], ViRAL (*Virtual Reality Abstraction Layer*), ARToolkit [24] e OpenCV [22]. Neste trabalho foram analisados o VRJuggler e o ViRAL. As outras bibliotecas serviram como referência para pesquisa, mas não foram profundamente analisadas.

VRJuggler

O VRJuggler é uma biblioteca de realidade virtual, com código aberto, multi-plataforma e independente de dispositivo [9]. Ela fornece uma abstração dos dispositivos de entrada e saída.

Junto a ela, qualquer biblioteca de grafo de cena (OpenGL Performer, OpenSceneGraph, OpenSG) pode ser usada para renderizar a cena, porém a integração do VRJuggler com essas bibliotecas exige que toda a biblioteca seja recompilada.

O VRJuggler é composto de vários módulos, sendo *vrj*, *jccl*, *gadgeteer* e *vpr* os principais. O módulo *vrj* é responsável pela integração dos outros módulos. Ele fornece uma camada acima do sistema operacional, permitindo que uma aplicação desenvolvida com o VRJuggler possa executar em diferentes sistemas, não só sistemas operacionais, mas em diversos sistemas de RV, como em um monitor, em uma CAVE ou em um HMD.

O módulo *jccl* é um sistema de configuração baseado em XML. O usuário utiliza arquivos de configuração para definir quais os dispositivos de entrada serão utilizados e a disposição dos dispositivos de saída. O desenvolvedor não precisa se preocupar com qual dispositivo o usuário irá utilizar para interagir com a aplicação, ou como ele irá visualizá-la. Baseado nos arquivos de configuração, a biblioteca carrega os *drivers* adequados.

O módulo *gadgeteer* é o sistema de gerência de dispositivos. Ele trata da configuração, controle, aquisição e representação do dado dos dispositivos de RV.

Finalmente, o módulo *vpr* é responsável por prover uma abstração independente de plataforma para *threads*, *sockets* (TCP/UDP) e comunicação serial.

Uma aplicação no VRJuggler é um objeto e sua classe deve derivar da classe de aplicação da biblioteca. A Figura 31 ilustra as principais classes de aplicação disponíveis nativamente no VRJuggler.

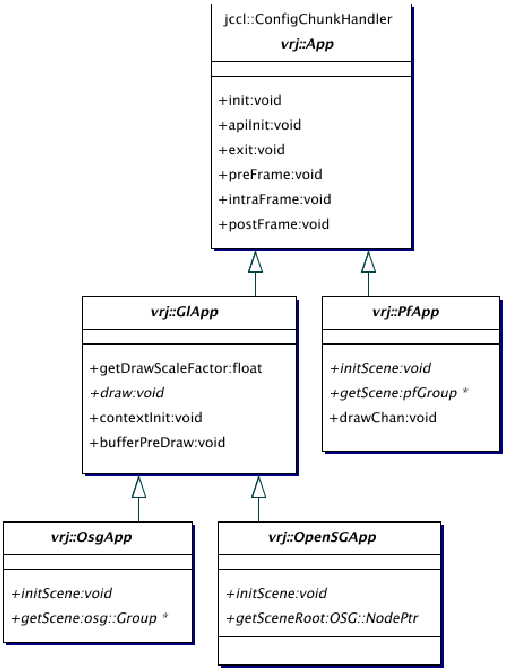

Figura 31: Hierarquia de classes de aplicação do VRJuggler.

A classe *vrj::App* é uma interface que define os principais métodos que uma aplicação deve implementar. A classe *vrj::GlApp* é, também, uma interface para uma aplicação que irá utilizar OpenGL, e a classe *vrj::PfApp* é uma interface para uma aplicação que irá utilizar Perfomer. Derivadas de *vrj::GlApp* estão as aplicações que utilizam grafos de cena para renderização, como a *vrj::OsgApp* que utiliza o OpenSceneGraph e a *vrj::OpenSGApp*, utilizando o OpenSG.

Ao definir a interface da aplicação, o VRJuggler tem controle do laço principal aplicação. A função **main** de uma aplicação desenvolvida com o VRJuggler deve ser algo do tipo:

```
#include <vrj/Kernel/Kernel.h>
#include <simpleApp.h>
int main( int argc, char *argv[] )
{
1 vrj::Kernel *kernel=vrj::Kernel::instance();   // Get the kernel
2 simpleApp *app = new SimpleApp();  // Create the app object
3 kernel->loadConfigFile(...);       // Configure the kernel
4 kernel->start();                   // Start the kernel thread
5 kernel->setApplication( app );     // Give application to kernel
  kernel->waitForKernelStop();       // Block until kernel stops
  return 0;
}
```

Tabela 2: Função main de uma aplicação utilizando o VRJuggler.

A linha identificada como 1 na Tabela 2 retorna o objeto núcleo (*kernel*) do VRJuggler. Na linha 2, um objeto da aplicação (S*impleApp*) é instanciado. Na linha 3, os arquivos de configuração são passados para o objeto *kernel* para que ele se configure. Na linha 4, o VRJuggler começa a ser executado. Nesse momento, o *kernel* cria uma nova *thread* de execução e começa o seu processamento interno. Qualquer mudança de estado do *kernel* a partir de agora

exige uma reconfiguração do VRJuggler, seja através da inclusão de novos arquivos de configuração ou da mudança da aplicação ativa. A aplicação que seria executada pelo *kernel* foi configurada na linha 5.

A Tabela 3 apresenta algumas vantagens e desvantagens no uso da biblioteca VRJuggler:

Vantagens	Desvantagens
• Código aberto; • Multi-plataforma, multi-dispositivos de RV; • Abstração dos dispositivos através de arquivos de configuração; • Desenvolvimento da aplicação é simples, bastando criar uma classe que derivará de alguma das inter-faces ilustradas na Figura 31; • Compilação de uma nova aplicação é bastante rápida (tendo já os arquivos binários pré-compilados do núcleo do VRJuggler).	• O arquivo de configuração não é fácil de ser editado manualmente; • A interface de edição de arquivos de configuração, desenvolvida em Java, é complicada e não permite que novos dispositivos sejam configura-dos através dela; • A compilação dos módulos do VRJuggler é bastante complicada, exigindo que sejam sempre utilizados os arquivos binários pré-compilados; • O VRJuggler depende de muitas bibliotecas externas, tornando ainda mais difícil a sua compilação; • O usuário não possui controle do laço principal da aplicação; • Uma aplicação desenvolvida com o VRJuggler é completamente imersiva, ou seja, não é possível utilizar ao mesmo tempo o VRJuggler e um sistema de diálogos (menus, listas, botões, etc.); • Os eventos gerados pelos dispositivos de entrada são pré-definidos e não podem ser estendidos.

Tabela 3: Vantagens e desvantagens do VRJuggler.

ViRAL (*Virtual Reality Abstraction Layer*)

O ViRAL é uma ferramenta utilizada para facilitar o desenvolvimento de aplicações de RV. As aplicações criadas com o ViRAL são independentes de dispositivo porque o ViRAL abstrai o contexto onde elas serão executadas. As aplicações que utilizam o ViRAL não precisam saber, por exemplo, em quantas janelas, com quantos usuários ou com quais dispositivos ela irá executar. Todas essas variáveis são configurações definidas pelo operador.

O ViRAL pode ser usado de duas maneiras: como a aplicação principal ou embutido (*embedded*). Ao ser executado como a aplicação principal, ele carrega

as suas janelas e menus e os diversos *plugins* criados pelos desenvolvedores, sendo um deles a aplicação propriamente dita. Um *plug-in* é um arquivo utilizado para alterar, melhorar ou estender as operações de uma aplicação principal (no caso, o ViRAL).

A segunda maneira de se utilizar o ViRAL é embuti-lo em uma aplicação. O ViRAL funciona como escravo e a aplicação principal é o mestre. O ViRAL possui seis sistemas que podem ser utilizados embutidos em uma aplicação, sendo eles, o sistema de ambiente, usuário, janela, dispositivo, cena e *plug-in*.

Sistema de Cenas

Uma cena é uma aplicação que é carregada com um *plug-in* pelo ViRAL. O sistema de cenas é o mais simples de todos. Ele é unicamente responsável por armazenar todas as cenas que foram carregadas pelo sistema de *plugins*.

Sistema de Usuário

O termo "usuário" no ViRAL se refere à "entidade" que está visualizando a cena, ou seja, ele é uma câmera no mundo virtual. Um "usuário" pode ser a visão de uma pessoa real posicionada em uma CAVE, uma visão externa dessa pessoa, ou uma câmera móvel, por exemplo.

O "usuário" possui como atributos a distância entre os olhos (para visualização estereoscópica), a orientação da cabeça e a posição e orientação do corpo. Outro atributo importante que o "usuário" possui é a cena da qual ele irá participar. Além disso, o "usuário" pode estar acompanhado de várias superfícies de projeção. A Figura 32 ilustra a interface de configuração de usuário.

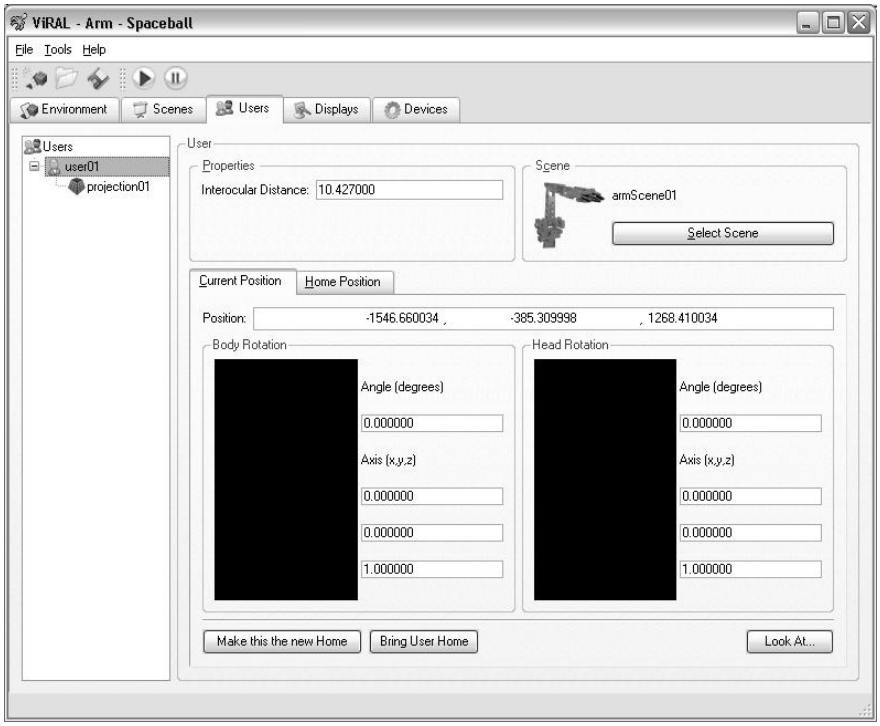

Figura 32: Interface de configuração de usuário no ViRAL.

Sistema de Janela

Uma aplicação mestre (ou um *plug-in* de cena), pode ser executada em diversas configurações de janelas. Para cada janela está associada uma projeção. A projeção é a conexão da janela com a cena, pois uma projeção pertence ao "usuário" e o "usuário" contém uma cena.
Uma janela possui um Canal que define como será exibida a projeção: visão mono, do olho esquerdo, do olho direito ou em estéreo. A Figura 33 ilustra a interface de configurações de janelas no ViRAL.

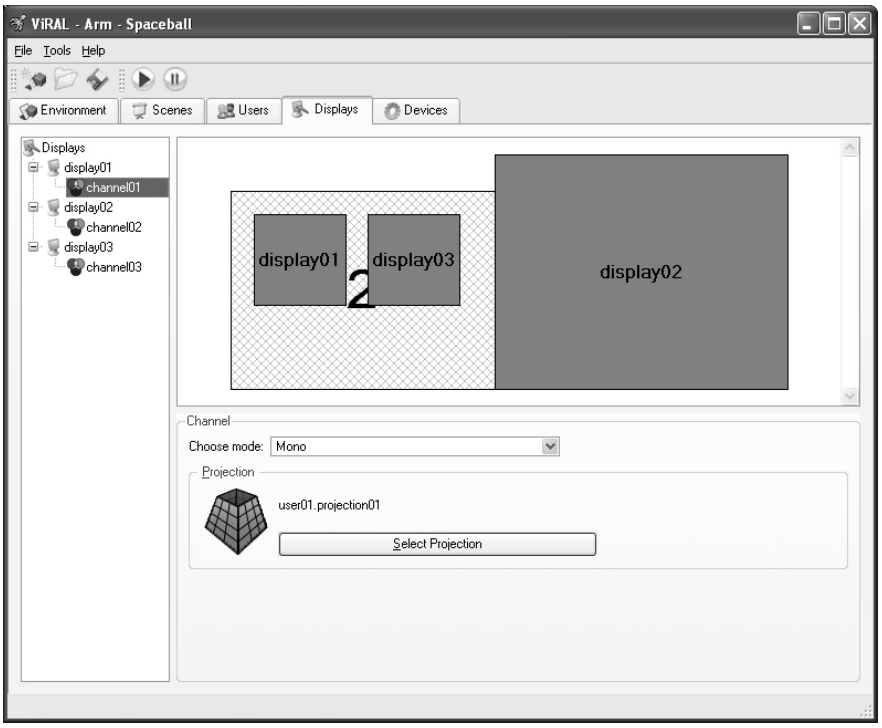

Figura 33: Interface de criação de janelas no ViRAL.

Sistema de Plug-in

No ViRAL existem 3 tipos de *plugins*: de dispositivo, de cena e de objeto de cena. Cada um desses *plugins* herdam de uma classe específica do ViRAL e exportam seus métodos. O sistema de *plugins* é responsável pelo carregamento dinâmico e conexão de eventos. A Figura 34 mostra alguns *plugins* que foram carregados pelo sistema.

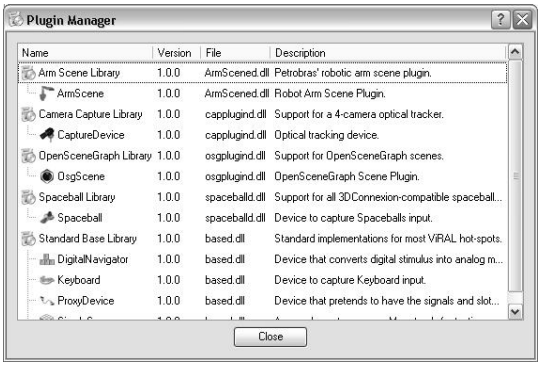

Figura 34: Gerenciador de *plugins* do ViRAL.

Um dispositivo novo é criado como uma biblioteca de carregamento dinâmico. Nessa biblioteca, o desenvolvedor deve criar uma classe que herda sua interface da classe *vral::Device*. A Tabela 4 mostra a interface dessa classes e os principais métodos que devem ser implementados.

```
namespace vral : public Object
{
  class Device
  {
      virtual const Object * getSender() const;
      virtual const Object * getReceiver() const;
      virtual void dispatchChanges() = 0;
      virtual QWidget * loadPropertyFrame();
  };
}
```

Tabela 4: Interface da classe *vral::Device*.

De baixo para cima, o método *loadPropertyFrame* é responsável por retornar o diálogo (*QWidget*) do Qt de configuração daquele dispositivo. A Figura 35 ilustra o diálogo de configuração de uma luva de realidade virtual. O método *dispatchChanges* transmite os eventos de um dispositivo, geralmente uma vez a cada passo de execução. O sistema de eventos no ViRAL utiliza o mecanismo de *signal/slot* do Qt. Esse mecanismo é utilizado para comunicações entre objetos. Os sinais (*signals*) são emitidos por objetos quando os mesmo mudam de estado. As aberturas (*slots*) são utilizadas para receber sinais.

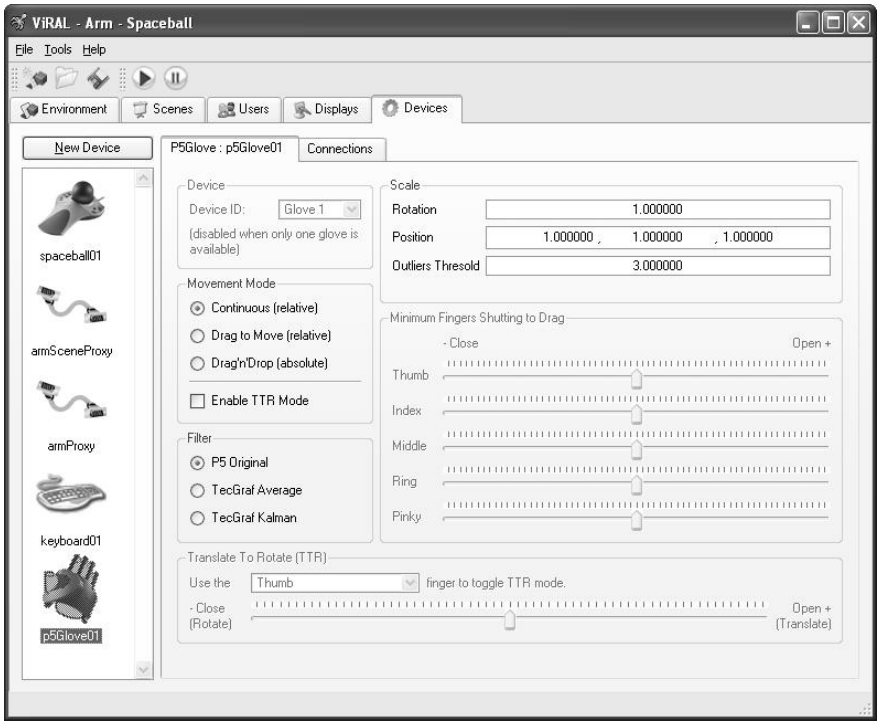

Figura 35: Configuração de uma luva de RV no ViRAL.

Os métodos *getSender* e *getReceiver* são utilizados para implementação de uma mecanismo de redirecionamento (*proxy*). Um exemplo de uso desse mecanismo é quando se desejar transformar um usuário do ViRAL em um dispositivo. O usuário não é um dispositivo, mas a partir de um dispositivo de redirecionamento, os eventos podem ser enviados a ele. Dessa forma é possível modificar a orientação e a posição do usuário através de um dispositivo de RV.

Os *plugins* de cena e de objetos de cena possuem o mesmo princípio de funcionamento dos *plugins* de dispositivo, porém as interfaces desses *plugins* são definidas pelas classes *vral::Scene* e *vral::SceneObject*, como visto na tabela Tabela 5.

```
namespace vral
{
  class SceneObject : public Object
  {
    virtual QWidget * loadPropertyFrame() = 0;
  };

  class Scene : public SceneObject
  {
    virtual bool isReady() const = 0;
    virtual bool checkTranslation() = 0;

    virtual void initializeContext() = 0;
    virtual void preDrawImplementation);
    virtual void drawImplementation();
  };
}
```

Tabela 5: Interface das classes *vral::Scene* e *vral::SceneObject*.

No ViRAL existe uma hierarquia que define quem é cena e quem é objeto de cena. Uma cena é a raiz dessa hierarquia e todos os seus filhos são objetos dessa cena. O método *loadPropertyFrame* carrega o diálogo de configuração do objeto de cena. *isReady* informa ao ViRAL que a cena está pronta para ser executada. O método *checkTranslation* valida uma movimentação dentro do ViRAL.

As cenas são responsáveis por configurar o estado do OpenGL para que o ViRAL possa utilizá-lo para renderização. Para isso existe o método *initializeContext*. Os métodos *preDrawImplementation* e *drawImplementation* têm por finalidade atualizar a cena e renderizá-la, respectivamente. O ViRAL calcula automaticamente as matrizes para serem carregadas pelo OpenGL baseado nas configurações de projeção e do modo de exibição de um canal.

A Figura 36 ilustra a interface de configuração de uma cena com um objeto de cena.

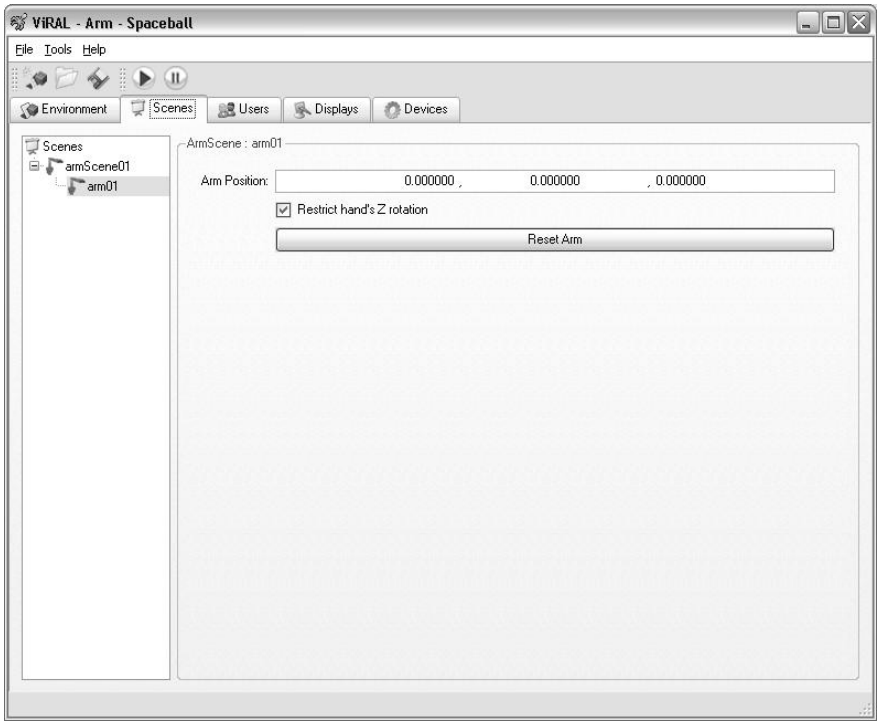

Figura 36: Interface de configuração de uma cena: *armScene01* é uma cena e *arm01* é um objeto de cena.

Sistema de Dispositivo

O sistema de dispositivo é responsável pela conexão e transmissão de eventos. Os eventos são conectados através da interface ilustrada na Figura 37. A cada quadro o ViRAL chama o método *dispatchChanges* dos seus *plugins*. Apesar dos *plugins* poderem funcionar em *threads* independentes, os dados só devem ser transmitidos mediante essa chamada de método.

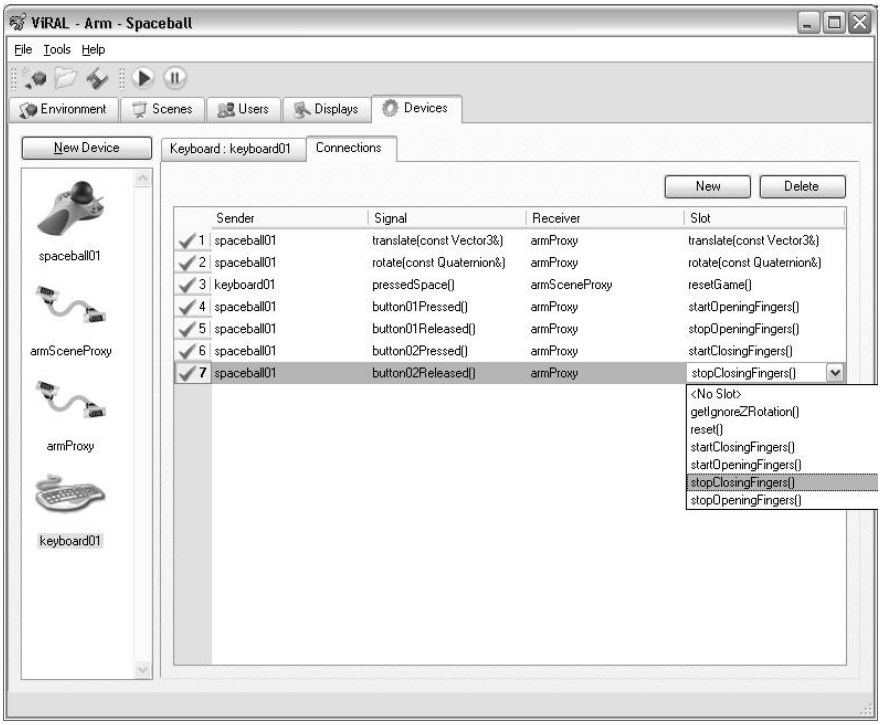

Figura 37: Interface de conexão de eventos no ViRAL.

Sistema de Ambiente

O Sistema de Ambiente é responsável pela gravação e leitura de dados de uma configuração do ViRAL. Ele armazena os arquivos utilizando um formato XML comprimido.

Esse sistema visita todos os objetos instanciados dentro do ViRAL (dispositivos, janelas, cenas, etc.) e grava as suas informações utilizando um padrão de projeto chamado Memento [21]. Todo *plug-in* criado para o ViRAL deve implementar dois métodos herdados de sua classe base. A Tabela 6 ilustra como um *plug-in* de cena grava e recupera as informações de um arquivo. A classe *vral::DataNode* é a classe que implementa o padrão Memento.

```
class CaptureDevice : public vral::Device
{
   // Métodos para gravação e leitura de dados
   virtual void getMemento( vral::DataNode &node )
   {
      // include the parent's memento
      Object::getMemento( node );

      node.setInt( "gridSize", mGridSize );
      node.setInt( "cameraNumber", mCameraNumber );
   }
   virtual void setMemento( const vral::DataNode &node )
   {
      mGridSize = node.getInt( "gridSize" );
      mCameraNumber = node.getInt( "cameraNumber" );
   }

   // Atributos que devem ser gravados
   int mCameraNumber;
   int mGridSize;
}
```

Tabela 6: *Plug-in* de cena com métodos de gravação e leitura de arquivo.

A Figura 38 ilustra a interface de carregamento de um ambiente dentro do ViRAL.

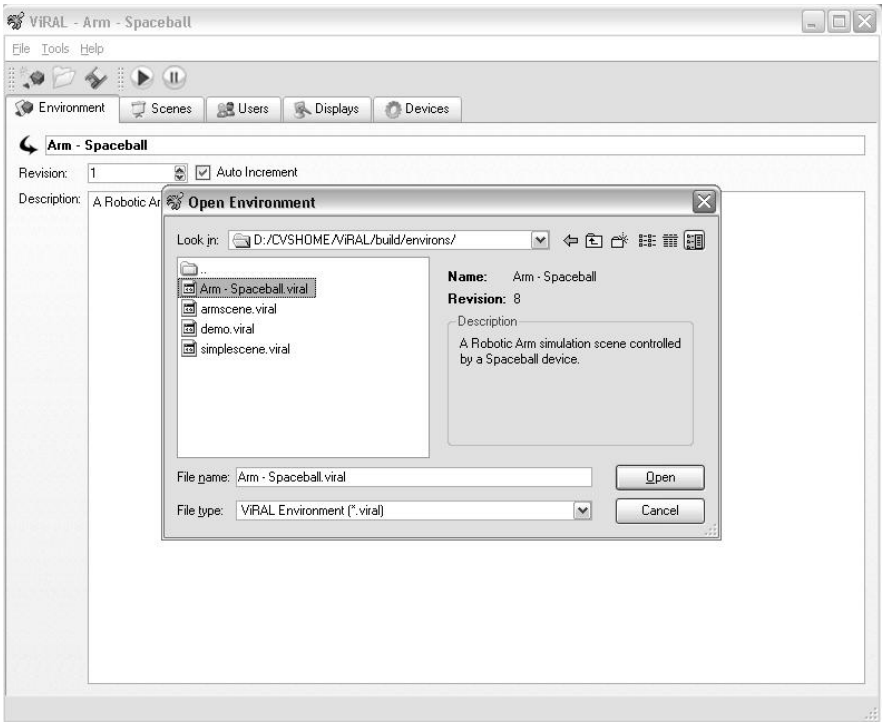

Figura 38: Carregando um ambiente no ViRAL.

Vantagens e Desvantagens

A Tabela 7 apresenta algumas vantagens e desvantagens da utilização do ViRAL. A utilização do Qt [35] pelo ViRAL traz uma grande vantagem que é a rápida construção de interfaces gráficas, multi-plataforma.

Vantagens	Desvantagens
• Interface gráfica através do Qt. • Uso de sistema de *plugins*; • Uso embutido permite integrar aplicações *desktop* e aplicações imersivas; • Rápido desenvolvimento de novos *plugins*; • Criação de eventos arbitrários permitindo programação de componentes; • Reconfiguração em tempo de execução.	• Exige conhecimento de orientação a objeto, tornando o uso para leigos mais demorado; • Novas aplicações ou *plugins* devem ser desenvolvidas com Qt, que é uma ferramenta comercial;

Tabela 7: Vantagens e desvantagens do ViRAL.

Conclusão

Este documento teve como um dos objetivos apresentar o conceito de grafo de cena e mostrar sua importância para a visualização em tempo real. A visualização em tempo real é, por sua vez, requisito básico de aplicações de realidade virtual.

A introdução a conceitos básicos de RV foi outro dos objetivos do texto, que fez um apanhado geral dos tipos de RV, dispositivos utilizados e tecnologias correlatas, como a realidade aumentada.

Finalmente, foram apresentadas duas ferramentas de software para a geração de aplicações de RV utilizando a noção de abstração de dispositivos, característica fundamental nas ferramentas de RV atuais.

Foram apresentados o VRJuggler, que é uma das ferramentas mais conhecidas internacionalmente e o ViRAL, desenvolvido no TecGraf/PUC-Rio para suprir algumas necessidades encontradas com o uso do VRJuggler.

A utilização do ViRAL mostrou-se bastante fácil e rápida, apesar de exigir conhecimentos mais avançados de orientação a objetos, padrões de projeto e programação de componentes.

A combinação de grafos de cena e ferramentas de realidade virtual está se consolidando como uma das maneiras de se desenvolver aplicações em ambientes complexos de navegação e interação. Esse fato é muito importante, pois ambas as tecnologias procuram fornecer uma interface de mais alto nível, restando ao desenvolvedor apenas a imaginação para criar novas aplicações.

Referências Bibliográficas

1 AGRAWALA, M., BEERS, A., FROEHLICH, B., HAN- RAHAN, P., MCDOWALL, I., AND BOLAS, M. **The Two-User Responsive Workbench: Support for Collaboration through Individual Views of a Shared Space**. In Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, p. 327-332, Los Angeles, 1997.

2 AKENINE-MÖLLER, T., HAINES, E. **Real-time Rendering**. 2ª edição, A K Peters, Massachusetts, 2002.

3 ALLARD, J., GOURANTON, V., LAMARQUE, G., MELIN, E., RAFFIN, B. **Softgenlock: Active Stereo and GenLock for PC Cluster**. VII International Immersive Projection Technologies Workshop, IX Eurographics Workshop on Virtual Environments, 2003.

4 ARAÚJO, R. B. **Especificação e análise de um sistema distribuído de realidade virtual**. Tese de Doutorado, Departamento de Engenharia de Computação e Sistemas Digitais, Escola Politécnica da Universidade de São Paulo, São Paulo, Junho, 1996.

5 ARSENAULT, L., KELSO, J., KRIZ, R., DAS-NEVES, F. **DIVERSE: a Software Toolkit to Integrate Distributed Simulations with Heterogeneous Virtual Environments**. Technical Report TR-01-10, Computer Science, Virginia Tech, 2001.

6 AUKSTAKALNIS, S., BLATNER, D. **Silicon Mirage: The Art and Science of Virtual Reality**. Editora Peatchpit Press, Berkeley, CA, 1992.

7 AZUMA, R. T. **A Survey of Augmented Reality**. In Presence: Teleoperators and Virtual Environments, pp. 355-385, Agosto, 1997.

8 AZUMA, R., BISHOP. G. **Improving Static and Dynamic Registration in a See-Through HMD**. Proceedings of SIGGRAPH, In Computer Graphics, Annual Conference Series, pp. 197-204, 1994.

9 BIERBAUM, A. **VR Juggler: A Virtual Platform for Virtual Reality Application Development**. Master Thesis, Iowa State University, 2000

10 BURDEA, G., COIFFET, P. **Virtual Reality Technology**. Editora John Wiley & Sons, Nova Iorque, 1994.

11 CRUZ-NEIRA, C., SANDIN, D. J., DEFANTI, T. A. **Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE**. ACM Computer Graphics, vol. 27, número 2 , pp 135-142, Julho, 1993.

12 DALY, J., KLINE, B., MARTIN, G. **A. VESS: Coordinating Graphics, Audio, and User Interaction in Virtual Reality Applications**. IEEE Virtual Reality, Março, 2002.

13 **Demeter Terrain Engine.** `http://demeter.sf.net` (Janeiro 2004).

14 FERREIRA, A. G. **Uma Arquitetura para a Visualização Distribuída de Ambientes Virtuais**. Dissertação de Mestrado, PUC/RJ, 1999.

15 FUHRMANN, A. HESINA, G., FAURE, F., GERVAUTZ, M. **Occlusion in Collaborative Augmented Environments**. Proceedings 5th Eurographics Workshop on Virtual Environments, pp. 179-190, 1999.

16 GAMMA, E., HELM, R., JOHNSON, R., VLISSIDES, J. **Design Patterns - elements of reusable object-oriented software**. Addison-Wesley Longman, Inc., 1995.

17 GIBSON, W. **Neuromancer**. Editora Ace Books, 1984.

18 GOURAUD, H. **Continuous Shading of Curved Surfaces**. IEEE Transactions on Computers, C-20(6):623-629, Junho, 1971.

19 GREENE, N., KASS, M., MILLER, G. **Hierarchical Z-Buffer Visibility**. Computer Graphics, volume 27, Annual Conference Series, pp. 231-238, 1993.

20 **How does the DTI display work?** Dimension Technologies Inc. `http://www.dti3d.com/technology.asp` (Dezembro 2003)

21 HUDSON, T., et al. **Accelerated Occlusion Culling using Shadow Frusta**. Symposium on Computational Geometry, pp. 1-10, 1997.

22 **Intel Open Source Computer Vision Library**. Intel Research. `http://www.intel.com/research/mrl/research/opencv` (Janeiro 2004).

23 JACOBSON, L. **Virtual Reality: A Status Report**. AI Expert, pp. 26-33, Agosto, 1993.

24 KATO, H., BILLINGHURST, M., POUPYREV, I., TACHIBANA, K. **Virtual Object Manipulation on a Table-Top AR Environment**. In Proceedings of ISAR, Outubro, 2000.

25 KRUEGER, M. W. **Artificial Reality II**. Editora Addison-Wesley, Massachussets, 1991.

26 LUEBKE, D., et al. **Level of Detail for 3D Graphics**. Morgan Kaufmann; 1ª edição, Julho, 2002.

27 MATOS, A., GOMES, J., PARENTE, A., SIFFERT, H., VELHO, L., **O sistema Visorama: Um novo sistema de Multimídia e Realidade Virtual**. II Workshop Multimedia and Hypermedia System, São Carlos, Brasil, Maio 1997.

28 MORIE, J. F. **Inspiring the Future: Merging Mass Communication, Art, Entertainment and Virtual environments**. ACM SIGGRAPH Computer Graphics, 28(2), pp. 135-138, Maio 1994.

29 **OpenGL® - The Industry Standard for High Performance Graphics** `http://www.opengl.org` (Janeiro 2004).

30 **OpenGL Performer Getting Started Guide**. SGI Document Number 007-3560-004, Janeiro, 2004.

31 **OsgVortex – Simulation Framework** `http://www.vrlab.umu.se/research/osgvortex/` (Janeiro 2004).

32 PHONG, B. **Illumination for Computer Generated Pictures**. CACM, 18(6), Junho, 1975.

33 PIEKARSKI, W., THOMAS, B. **ARQuake: the Outdoor Augmented Reality Gaming System**. Comm. ACM, vol. 45, no. 1, pp. 36-38, Janeiro 2002.

34 POULIN, P., FOURNIER, A. **A Model for Anisotropic Reflection**. Proc. Siggraph, Agosto, 1990.

35 **Trolltech - Creators of Qt - The multi-platform C++ GUI/API**. `http://www.trolltech.org/` (Fevereiro 2004).

36 ROLLAND, J., HOLLOWAY, R., FUCHS, H. **A Comparison of Optical and Video See-Through Head-Mounted Displays**. SPIE Proceedings volume 2351: Telemanipulator and Telepresence Technologies, pp 293-307, 1994.

37 SHERMAN, W. R., CRAIG, A. B. **Understanding Virtual Reality: Interface, Application, and Desing**. Editora Morgan Kaufmann, 2003.

38 SILVA, M. H. **Tratamento Eficiente de Visibilidade Através de Árvores de Volumes Envolventes**. Dissertação de Mestrado, PUC/RJ, 2002.

39 STEFANI, O., HOFFMANN, H., RAUSCHENBACH, J. **Design of Interaction Devices for Optical Tracking in Immersive Environments**. In: Human-Centred Computing: Cognitive, Social and Ergonomic Aspects, Volume 3 of the Proceedings of HCI International, Junho 2003.

40 SUTHERLAND, I. **Sketchpad: The First Interactive Computer Graphics**. Tese de Ph.D., Mass. Institute of Technology, 1963.

41 TRAMBEREND, H. **Avango: A Distributed Virtual Reality Framework**. Proceedings of Afrigraph '01, ACM, 2001.

42 ULBRICHT, C., SCHMALSTIEG, D. **Tangible Augmented Reality for Computer Games**. Relatório Técnico TR-186-2-03-02, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Fevereiro, 2003.

43 WALSH, A. E. **Understanding Scene Graphs**. Dr Dobb's Journal, 27:7, 17-26, 2002.

44 WOO, M., NEIDER, J. DAVIS, T., SHREINER, D. **OpenGL(R) Programming Guide: The Official Guide to Learning OpenGL, Version 1.2**. Addison-Wesley Pub Co, 3ª edição, Agosto, 1999.

45 ZHANG, H., MANOCHA, D., HUDSON, T., HOFF, K. **Visibility Culling using Hierarchical Occlusion Maps**, Computer Graphics SIGGRAPH '97 Proceedings, pp. 77-88, Agosto 1997.