UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DIEGO ROBERTO COLOMBO DIAS

SISTEMA AVANÇADO DE REALIDADE VIRTUAL PARA VISUALIZAÇÃO DE ESTRUTURAS ODONTOLÓGICAS

DISSERTAÇÃO DE MESTRADO

BAURU

2011

DIEGO ROBERTO COLOMBO DIAS

SISTEMA AVANÇADO DE REALIDADE VIRTUAL PARA VISUALIZAÇÃO DE ESTRUTURAS ODONTOLÓGICAS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho" como requisito parcial para obtenção do título de "Mestre em Ciências" – Área de Concentração: Sistemas de Computação.

Orientador: Prof. Dr. José Remo Ferreira Brega

BAURU

2011

Dedico este trabalho aos meus pais, que muito me incentivaram em meus estudos e a minha noiva Amanda, pelo amor e paciência durante mais uma fase da minha vida.

AGRADECIMENTOS

Quero agradecer primeiramente ao meu orientador, Prof. Dr. José Remo Ferreira Brega, por ter acreditado em mim e por todo seu apoio.

Aos amigos do LSTR, Anthony, Mário e Júnior, que me auxiliaram do desenvolvimento deste trabalho. Sem eles, não seria a mesma coisa.

Aos companheiros do SACI, Léo, André e Carlos, que acompanharam o desenvolvimento deste trabalho.

Ao Prof. Dr. José Roberto Pereira Lauris por ter acreditado e trabalhado comigo, apoiando em todas as dúvidas referentes a área odontológica.

Aos colaboradores diretos desta dissertação, Bruno Gnecco e Marcelo Guimarães. Agradeço pelos inúmeros conselhos e dicas preciosas.

A minha noiva, Amanda, pela compreensão no momentos difíceis e muito amor.

Aos meus pais, que mesmo longe sempre incentivaram meus estudos.

FINANCIAMENTO

Este trabalho foi possível graças a financiamentos disponibilizados pelo:

- XPTA.Lab;
- CNPq: Projeto vinculado à Faculdade de Odontologia de Bauru, Universidade de São Paulo; e
- CAPES.

"Determinação, coragem e autoconfiança são fatores decisivos para o sucesso. Não importa quais sejam os obstáculos e as dificuldades, se estamos possuídos por uma inabalável determinação, conseguiremos superá-los. Independentemente das circunstâncias, devemos ser sempre humildes, recatados e despidos de orgulho."

Dalai Lama.

RESUMO

DIAS, D. R. C.. Sistema Avançado de Realidade Virtual para Visualização de Estruturas Odontológicas. 119 f. Dissertação – Programa de Pós-Graduação em Ciência da Computação, Universidade Estadual Paulista "Júlio de Mesquita Filho". Bauru, 2011.

Formas de representação da informação são importantes para aquisição do conhecimento. O uso de tecnologias pode tornar esta aquisição eficiente. Neste contexto, esta dissertação apresenta uma estrutura de multiprojeção de baixo custo baseada em aglomerados gráficos, a qual é utilizada na visualização de modelos virtuais, com o intuito de auxiliar o ensino de estruturas odontológicas. Contudo, os modelos virtuais por si só não representam completamente as informações requeridas a uma ferramenta de ensino. Assim, é necessária a integração de conceitos pertinentes à área a qual os modelos virtuais representam, utilizando decrições semânticas em conjunto com tais modelos. Com a finalidade de definir uma ontologia de apoio a descrição semântica, são apresentados os conceitos que foram utilizados como atributos no desenvolvimento da ontologia proposta. Com o mesmo intuito das descrições, conteúdo multimídia também é agregado aos modelos virtuais. Ao final, tem-se um sistema que permite o total controle ao usuário, desde a criação e edição de modelos virtuais até as visualizações apresentadas pelo sistema de multiprojeção.

Palavras-chave: Sistema Avançado de Realidade Virtual, Java 3D, Aglomerados Gráficos, Descrições Semânticas

ABSTRACT

DIAS, D. R. C.. Advanced Virtual Reality System for Visualization of Dental Structure. 119 f. Dissertação – Programa de Pós-Graduação em Ciência da Computação, Universidade Estadual Paulista "Júlio de Mesquita Filho". Bauru, 2011.

Forms of information representation are important for knowledge acquisition. The use of technologies can make this acquisition efficient. In this context, this dissertation presents a multi-projection low-cost framework based on graphics clusters, which is used in the virtual model visualization in order to assist the teaching of dental structures. However, the virtual models alone do not fully represent the information required by a teaching tool. These require the integration of concepts relevant to domain, using semantic descriptions in conjunction with virtual models. In order to define an ontology to support the description, the used concepts are presented as attributes in the ontology created. With the same purpose of descriptions, multimedia content is also added to the virtual models. At the end, we have a system that allows the user full control, from creation and editing of virtual models to the views presented by the system multi-projection.

Keywords: Advanced Virtual Reality Systems, Java 3D, Graphics Clusters, Semantic Descriptions

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

RV	Realidade Virtual
AV	Ambiente Virtual
AG	Aglomerados Gráficos
VC	Visualização Científica
VI	Visualização de Informação
CAVE	*Cave Automatic Virtual Environment*
EVL	*Eletronic Visualization Laboratory*
CRT	*Cathodic Ray Tube*
SGI	*Silicon Graphics*
DLP	*Digital Light Processing*
LCD	*Liquid Crystal Display*
API	*Application Programming Interface*
XML	*Extensible Markup Language*
SAN	*Storage Area Network*
PDA	*Personal Digital Assistant*
TCP	*Transmission Control Protocol*
UDP	*User Data Protocol*
MPI	*Message Passing interface*
RDF	*Resource Description Framework*
OWL	*Web Ontology Language*
X3D	*Extensible 3D*
W3C	*World Wide Web Consortium*
URI	*Uniform Resource Identifier*
VRML	*Virtual Reality Modeling Language*

3D	*Three Dimensions*
NURBS	*Non Uniform Rational Basis Spline*
MPEG	*Moving Picture Experts Group*
3DAF	*3D Annotation Framework*
LSTR	Laboratório de Sistemas de Tempo Real
PC	Personal Computer
IP	Internet Protocol
JVM	*Java Virtual Machine*

LISTA DE CÓDIGOS

SUMÁRIO

1 INTRODUÇÃO

A Realidade Virtual (RV) é a forma mais avançada de interface do usuário com o computador em um ambiente sintético tridimensional gerado por computador (HANCOCK, 1995). Kirner e Siscoutto (2007) também definem a RV como a forma mais avançada de interação do usuário com aplicações executadas por computador, no entanto, eles definem características básicas que devem coexistir: imersão, interação e navegação em elementos de um Ambiente Virtual (AV) gerado por computador.

Os sistemas de multiprojeção podem proporcionar alto grau de imersão ao usuário, podendo ser implementados em sistemas tradicionais ou Aglomerados Gráficos (AG). Os AGs diferem dos tradicionais sistemas fortemente acoplados em alguns aspectos. Os sistemas tradicionais dividem sua tarefa principal em menores, as quais são distribuídas para os nós e logo após o processamento das mesmas é que a sincronização acontece. Já os AGs, têm como objetivo oferecer uma visão múltipla do mesmo conjunto de dados, ou seja, cada nó processa apenas os dados referentes ao seu interesse, gerando assim a imagem apenas daquela parte (GUIMARAES, 2004). O CAVE (CRUZ-NEIRA et al., 1992) proporciona alto grau de imersão, portanto, o sistema de multiprojeção utilizado nesta dissertação foi baseada em sua estrutura.

Outra característica provida por sistemas de RV é a interação com o ambiente tridimensional, que permite ao usuário visualizar o ambiente sobre qualquer ponto de vista, movimentar-se dentro dele e interagir com seus objetos virtuais. Ao interagir com um AV, o computador detecta e reage às ações do usuário respondendo a estas ações com modificações no ambiente, tão mais próxima ao esperado no universo real que conhecemos. Esta interação pode ser alcançada utilizando-se dispositivos de interface, sendo eles convencionais ou não convencionais.

Deste modo, a RV, por prover uma experiência de interface diferente do habitual, tem sido explorada por aplicações com fins educacionais e de treinamentos. A medicina tem utilizado sistemas de RV para vários tipos de aplicações, como em simulações de procedimentos cirurgícos.

Por outro lado, a ondontologia ainda não possui tal volume de pesquisas relacionadas. Por-

tanto, devido à evolução das tecnologias e a necessidade de ferramentas educacionais, as possibilidades de utilização de RV na odontologia são inúmeras, as quais vão desde o ensino da anatomia, até a simulação de tratamento clínico operatório. Assim, o desenvolvimento de um sistema de RV com base em estruturas dentárias é o início do que poderá evoluir, no futuro, para a representação completa do sistema estomatognático.

Com o avanço da tecnologia, dos equipamentos de imagens em geral e dos computadores, surge a possibilidade da construção de sistemas com alto nível de complexidade, que possam oferecer informações de maneira mais precisa do que dados puros e brutos. Para a representação desses dados, foram utilizados conceitos de Visualização Científica (VC), onde modelos 3D criados com o uso da computação gráfica, auxiliam no entendimento dos dados apresentados, confiando na habilidade poderosa dos humanos em visualizar. Desse modo, modelos dentários normalmente visualizados em livros por meio figuras planas, podem ser apresentados em modelos 3D, sendo possível a interação de modo intuitivo.

Outro conceito também utilizado é o de descrições semânticas e agregação de conteúdo multimídia, podendo auxiliar no processo de aquisição de conhecimento dos usuários. Neste trabalho, são efetuadas descrições semânticas em ambientes 3D, de maneira que, componentes do AV possam ser identificados de maneira isolada, facilitando assim, o reaproveitamento desses. Documentos multimídias também são utilizados em conjunto com os modelos virtuais, possibilitando ao usuário uma ferramenta de visualização que seja capaz de representar, por exemplo, modelos virtuais e vídeos relacionados a estes modelos.

A combinação dessas tecnologias apresenta a possibilidade de criação de um Sistema Avançado de RV para Estruturas Odontológicas, onde modelos de um AV são utilizados no auxílio ao entendimento dos dados contidos em estruturas odontológicas representadas por modelos virtuais e, ainda, possuindo as descrições semânticas, que podem ampliar o poder de obtenção de informações pertinentes ao AV.

Os benefícios ocasionados pelo uso de técnicas de RV no uso de sistemas virtuais no auxílio ao ensino são inúmeros. Assim sendo, este trabalho procura provêr um ambiente interativo e imersivo ao usuário, de modo que, o uso da RV possa auxiliar na formação de profissionais com melhor capacitação, estimulando o aprendizado por meio de aulas interativas.

Para a geração de conteúdo ao sistema desenvolvido, foi criado um módulo de edição de modelos virtuais que permite ao usuário criar sub-modelos com partes de modelos complexos. As descrições semânticas e agregações de documentos multimídia também são realizadas neste módulo.

1.1 OBJETIVOS

Os objetivos específicos deste trabalho são:

- Implantar uma estrutura de multiprojeção de baixo custo;
- Desenvolver uma aplicação Java 3D distribuída que seja capaz de carregar modelos virtuais e possa ser executada em uma estrutura baseada em AG;
- Realizar interações com o AV por meio de controle não convencional;
- Implementar uma aplicação desktop ao usuário que permita a editoração de modelos virtuais, descrição e agregação de conteúdo multimídia; e
- Implementar uma aplicação de controle do AG, onde as execuções de tarefas possam ser realizadas de modo intuitivo, tornando a estrutura do sistema de multiprojeção transparente ao usuário do sistema.

1.2 ESTRUTURA DA DISSERTAÇÃO

Esta dissertação está estruturada da seguinte maneira:

- O Capítulo 2 apresenta conceitos relacionados a Visualização de Informação (VI) e a sua sub-área, a VC. Também são apresentados alguns trabalhos correlatos que fazem uso de técnicas de RV voltadas à área médica e odontológica.
- O Capítulo 3 apresenta um histórico sobre a estrutura de multiprojeção baseada em CAVE. São apresentados conceitos relacionados ao uso de AGs na geração de imagens a sistemas de multiprojeção. São apresentadas ferramentas de desenvolvimento de aplicações baseadas em AG, destacando a biblioteca Glass, utilizada no desenvolvimento do sistema de multiprojeção desenvolvido neste trabalho.
- O Capítulo 4, com intuito de prover uma maneira de ser agregar descrições textuais e conteúdo multimídia aos modelos virtuais, aborda uma solução baseada em ontologias, a qual é utilizada na definição de termos relevantes aos modelos virtuais de estruturas odontológicas.
- O Capítulo 5 aborda o desenvolvimento da aplicação Java 3D para sistemas de multiprojeção baseados em AG. São apresentadas as principais características no processo de portabilidade de uma aplicação de projeção única para várias projeções.

- O Capítulo 6 apresenta a proposta e o desenvolvimento da ontologia de apoio utilizada nas descrições semânticas e agregações multimídias. A estrutura da ontologia criada é apresentada, como também, exemplos de instâncias criadas à partir desta ontologia.

- O Capítulo 7 apresenta a junção dos módulos apresentados nos Capítulos 5 e 6. Desse modo, é apresentado um sistema ao usuário que permite a criação e edição de modelos virtuais. As descrições semânticas e agregações multimídia também são realizadas por este sistema. Ao final, é apresentado um módulo de controle com o AG, que possibilita total controle sem a necessidade de se conhecer a estrutura desenvolvida. Para que o sistema fosse validado, este foi apresentado a profissionais da área odontológica, sendo avaliado por meio de um questionário.

- O Capítulo 8 apresenta as conclusões deste trabalho baseadas no desenvolvimento e avaliação do sistema. Uma seção apresenta os trabalhos futuros propostos. Os artigos publicados durante o desenvolvimento desta dissertação também são apresentados.

2 REPRESENTAÇÃO DA INFORMAÇÃO

De maneira a explorar o sistema cognitivo humano, a idéia de VI é apresentar ao usuário uma maneira de visualizar dados utilizando-se da sua principal percepção: a visão.

Do mesmo modo que a VI, a VC apresenta dados ao usuário, porém, sendo aplicada a objetos físicos , fenômenos da natureza ou posições de um domínio espacial.

Este capítulo apresenta conceitos e exemplos de aplicação de VC.

2.1 VISUALIZAÇÃO DE INFORMAÇÃO

Card e Mackinlay (1999) definem VI como sendo o uso de representações visuais de dados abstratos suportadas por computador e interativas para ampliar a cognição. O objetivo principal é ampliar a cognição, melhorando assim, o entendimento e o aproveitamento do conhecimento exposto.

Outra definição é a de Gershon, Eick e Card (1998) que definem a VI como sendo o processo de transformação de dados, informação ou conhecimento em uma forma visual natural as capacidades de visualização do ser humano.

Um modelo de estrutura de VI proposto por Card e Mackinlay (1999) é apresentado na Figura 1.

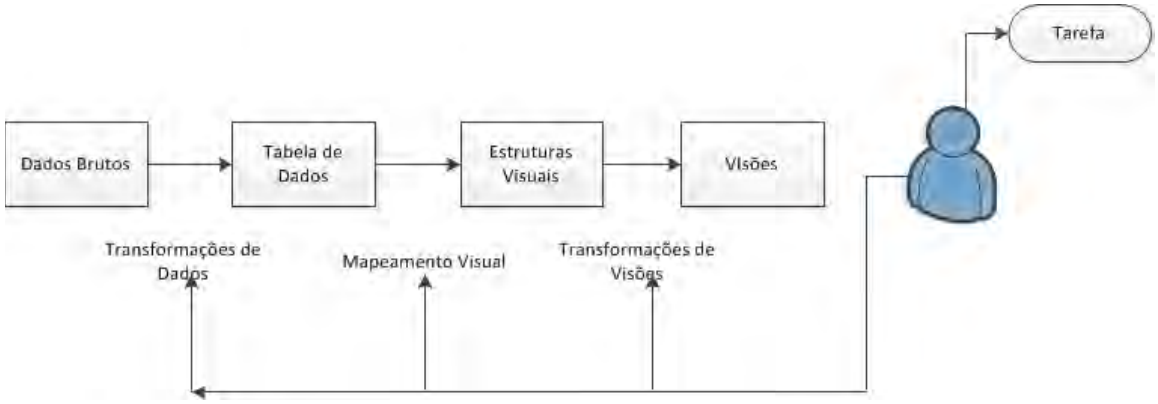


Figura 1: Modelo de geração de uma estrutura de VI
Fonte: (CARD; MACKINLAY, 1999)

A estrutura de VI é iniciada a partir dos dados brutos organizados em uma tabela de dados, chamada de entidade, da qual é gerada uma estrutura visual utilizada para representar as informações. Estas informações podem ser: gráficos de barra, setores, diagramas, esquemas e mapas.

Para que seja gerada essa representação, as entidades devem ser transformadas em formas gráficas, para que o usuário possa utilizar sua própria cognição no processo de extração das informações. A partir disto, criam-se visões que permitem ao usuário observar as estruturas visuais, para que decisões possam ser tomadas e tarefas sejam realizadas.

2.2 VISUALIZAÇÃO CIENTÍFICA

Uma área similar à VI é chamada de VC. O objetivo da VC, da mesma maneira que a VI, é proporcionar o entendimento dos dados apresentados, confiando na habilidade poderosa dos humanos em visualizar (ADAIME, 2005). A VC é geralmente aplicada à apresentação de objetos físicos, fenômenos da natureza ou posições de um domínio espacial, por meio de uma representação geométrica. Um exemplo de aplicação da VC é a visualização de partes do corpo humano, podendo ser realizada por meio de modelos virtuais.

Por ser considerada uma área multidisciplinar, tem sido utilizada como ferramenta em pesquisas e atividades educacionais.

Mccormick (1987) define a VC como sendo o uso de Computação Gráfica para criar modelos que ajudem na compreensão de conceitos ou resultados complexos, frequentemente associados a representações numéricas volumosas. Segundo Collins (1993), as primeiras técnicas de VC surgiram no século XII, sendo hoje aplicadas à representação de grandes volumes de dados

complexos. Porém, foi Mccormick (1987) que apresentou um trabalho intitulado *Visualization in Scientific Computing*, no evento SIGGRAPH do ano de 1987, sendo a primeira publicação na área, dando origem a vários outros trabalhos seguintes. Ultimamente, técnicas de VC tem sido utilizadas em análises de modelos 3D, permitindo ao usuário extrair informações de maneira fácil e rápida desses modelos.

Porém, deve-se estar atento à forma com que VC é utilizada. De acordo Globus e Raible (1994), a VC pode ser usada para gerar imagens bonitas, mas nem sempre essas transmitem informações científicas relevantes. De acordo com a definição de Mccormick (1987), é possível combinar técnicas de RV e VC, a fim de desenvolver um software de simulação de alta qualidade com propósito educacional (BARNES, 1996).

Algumas áreas tem empregado técnicas de VC em conjunto com RV. A área médica e odontológica tem investido nessas técnicas para a visualização e construção de sistemas de apoio a diagnóstico.

Fadel e Costa (2006), por meio de um histórico, apresentam os benefícios de se utilizar a RV em ferramentas de VC no auxílio ao ensino de Odontologia, citando algumas vantagens e desvantagens.

O trabalho de Santhanam et al. (2008), apresenta uma ferramenta para visualização médica de deformações em pulmão. A ferramenta é capaz de criar estas deformações a partir de imagens de um paciente real geradas por tomógrafos. A Figura 2 apresenta a imagem de um pulmão gerado por computador.

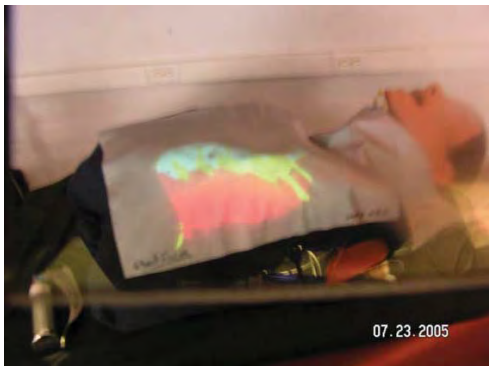
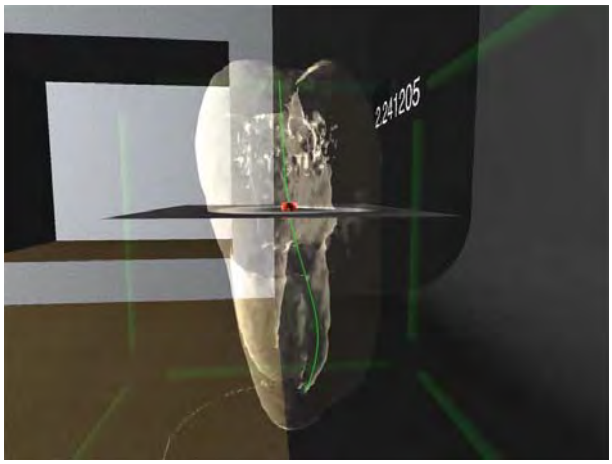


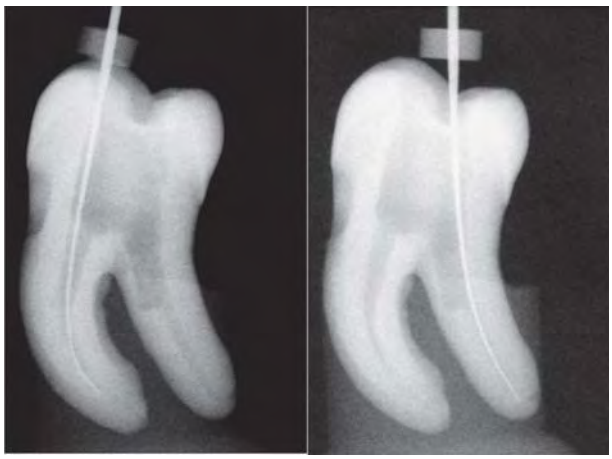
Figura 2: Visualização de um pulmão não deformado
Fonte: (SANTHANAM et al., 2008)

O trabalho de Germans et al. (2008), apresenta uma ferramenta de RV utilizada na mensuração do tamanho do canal radicular a partir de dados tomográficos. Também é apresentada uma comparação entre o método utilizando RV e o método convencional. A Figura 3 apresenta

os 2 métodos utilizados.



(a) Modelo virtual com o canal radicular sendo mensurado



(b) Modelo real com canal radicular sendo mensurado

Figura 3: Mensuração do canal radicular: virtual e convencional
Fonte: (GERMANS et al., 2008)

Silén et al. (2008) abordam as dificuldades de estudantes na área da saúde que necessitam aprender a anatomia. Desta forma, é apresentado um estudo em que alunos de medicina são questionados sobre o uso de 2 tipos de visualização, 3D ou convencional. A visualização 3D é apresentada por meio de modelos gerados no formato *quicktime* VR. A visualização convencional é apresentada por meio de vídeos. A Figura 4 apresenta os 2 tipos de mídias.

(a) Quadro de um filme de ressonância magnética

(b) Material em formato *quicktime* VR

Figura 4: Mídias utilizadas no ensino de anatomia
Fonte: (SILÉN et al., 2008)

Petersson et al. (2009) desenvolveram uma ferramenta semelhante a de Silén et al. (2008), no entanto, visando um ambiente *web*. A ferramenta é capaz de auxiliar no aprendizado de anatomia por meio de vídeos panorâmicos. Arquivos de vídeo no formato *quicktime* VR foram

utilizados. Um experimento foi realizado com alunos de medicina para avaliar a qualidade da ferramenta. A Figura 5 apresenta a tela inicial da ferramenta *web*.

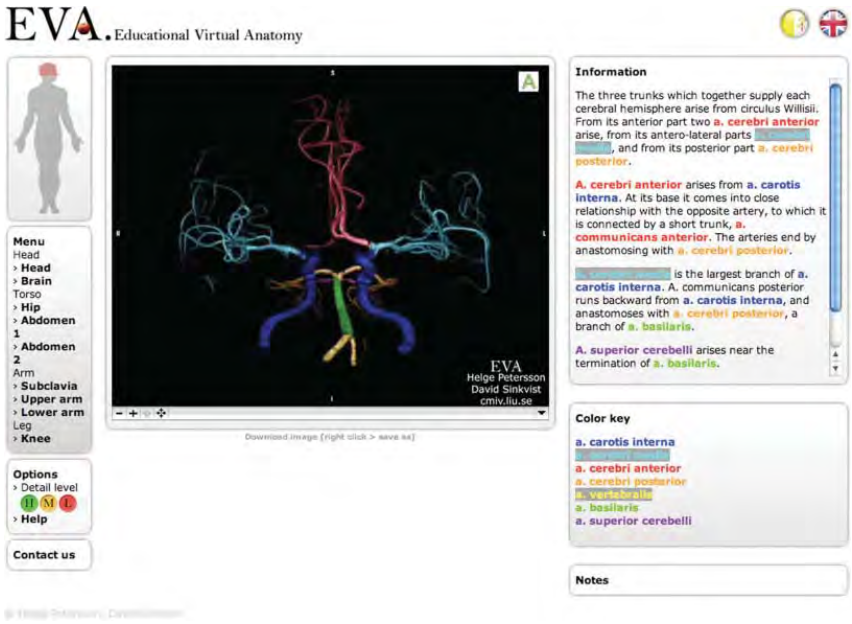


Figura 5: Ferramenta de visualização *web* EVA
Fonte: (PETERSSON et al., 2009)

2.3 CONSIDERAÇÕES FINAIS

A utilização de técnicas de VI e VC em sistemas de RV possibilitam a melhor aquisição de informação por parte do usuário.

Este capítulo destacou que aplicações voltadas à área de medicina tem-se beneficiado de tecnologias envolvendo RV e VC. Porém, aplicações relacionadas a odontologia não são exploradas da mesma maneira. Assim sendo, o desenvolvimento de um sistema de visualização para estruturas odontológicas pode ser de grande auxílio ao ensino.

Os conceitos que foram apresentados neste capítulo, são de grande importância para o desenvolvimento deste trabalho, pois modelos virtuais são utilizados como representação de informação ao usuário, com intuito de ampliar sua cognição.

3 SISTEMAS DE MULTIPROJEÇÃO DE BAIXO CUSTO E AGLOMERADOS GRÁFICOS

Com a crescente evolução e diminuição de preços dos microcomputadores, os sistemas fortemente acoplados começaram a dar espaço a sistemas baseados em aglomerados de computadores, que quando utilizados para o processamento de dados multimídia, são também chamados de AGs.

Com o intuito de minimizar custos e otimizar a escalabilidade das aplicações, os AGs tem sido utilizados também em aplicações de RV, que por necessitar de grande capacidade de processamento, tem-se beneficiado.

Os sistemas de multiprojeção são beneficiados diretamente por avanços tecnológicos na área de AG, pois cada vez mais estes são utilizados na implantação de tais sistemas.

Este capítulo incialmente apresenta uma visão geral sobre a conceitualização de AG e apresenta uma biblioteca, a Glass (GUIMARAES, 2004; GNECCO; GUIMARAES; ZUFFO, 2003), desenvolvida para facilitar o desenvolvimento de aplicações para AGs. Os componentes da Glass são descritos de maneira sucinta, demonstrando o funcionamento da biblioteca. Também é apresentado um exemplo de Sistema de Multiprojeção, o CAVE, e alguns dos *frameworks* disponíveis para o desenvolvimento de aplicações.

3.1 SISTEMAS DE MULTIPROJEÇÃO

Os sistemas de multiprojeção, geralmente compostos por múltiplas telas, proporcionam ao usuário diferentes pontos de vista de um mesmo ambiente, possibilitando assim, uma experiência com alto poder de imersão.

Entre os sistemas que podem ser implementados, tem-se o *Cave Automatic Virtual Environment* (CAVE), que se implementado utilizando AG, pode ser implantado com baixo custo, podendo também ser classificado como um Sistema de Realidade Virtual Avançado por utilizar alto poder de processamento gráfico e por suportar dispositivos não convencionais.

O primeiro CAVE foi desenvolvido pelo *Eletronic Visualization Laboratory* (EVL) na Universidade de Illinois, Chicago, sendo apresentado no SIGGRAPH'92 (CRUZ-NEIRA et al., 1992). Depois desta publicação, outros semelhantes foram desenvolvidas ao redor do mundo. No Brasil, o Laboratório de Sistemas integráveis da Universidade de São Paulo, desenvolveu a Caverna Digital (SOARES, 2005).

Cruz-Neira et al. (1992), implementaram o primeiro CAVE utilizando projetores *Cathodic Ray Tube* (CRT) com resolução 1280x1024 e paredes de 3m x 3m. As aplicações geravam projeções estéreo ativo. A estrutura do sistema foi proposta com 3 paredes *rear-project* e uma projeção no chão por meio de *down-project*. Tal estrutura foi proposta para gerar um novo senso de imersão ao usuário do sistema. As projeções eram geradas por estações *Silicon Graphics* (SGI) Crimson, porém, as aplicações atingiam apenas 8 quadros por segundo, não sendo suficiente para uma animação. Posteriormente, as estações SGI foram substituídas por sistemas SGI Onyx.

Na segunda geração de CAVE, proposta em 2001, o EVL focou suas pesquisas em melhorar a qualidade das projeções utilizando projetores Christie Mirage *Digital light Processing* (DLP), que possuem ganho em brilho, porém, sendo mais caros que os da geração anterior. O *frame rate* também foi aprimorado, passando a gerar 25 quadros por segundo (DEFANTI et al., 2009).

Foi proposta ainda uma terceira geração de CAVE, a qual utiliza projetores *Liquid Crystal Display* (LCD), provendo alta resolução de imagem e sendo visualizadas por meio de óculos polarizados. As paredes quadradas, anteriormente propostas nas duas outras gerações, também foram alteradas para uma sala em forma de pentágono. A Figura 6 apresenta a estrutura do StarCAVE (DEFANTI et al., 2009).

3.2 FERRAMENTAS DE DESENVOLVIMENTO

Para o desenvolvimento de sistemas para CAVE, foram propostos alguns *frameworks* para facilitar o trabalho dos desenvolvedores. Essas soluções podem ser divididas em: comerciais, de pesquisa e de código aberto. Esta seção descreve alguns *frameworks* desenvolvidos.

O primeiro *framework* foi desenvolvido pelo EVL e denominado CaveLib. Atualmente é comercializado pela Mechdyne (CAVELIB, 2011). O CaveLib possui uma *Application Programming Interface* (API) de baixo nível que abstrai o controle das tarefas ao desenvolvedor. Dentre as tarefas oferecidas pela API, tem-se a criação de pontos de vista, o sincronismo do aglomerado e compartilhamento de dados. A integração com bibliotecas de alto nível, como a OpenGL, é suportada.

Figura 6: StarCAVE
Fonte: (DEFANTI et al., 2009)

O VR Juggler permite ao desenvolvedor portar aplicações de RV. A configuração do ambiente VR Juggler é feita por meio de arquivos *Extensible Markup Language* (XML), que definem número de nós, disponibilizando endereços de *hardware* e rede; dispositivos de entrada, tais como luvas, rastreadores etc; configurações de telas perante *layout* físico e atribuição de nós. O VR Juggler é configurável em tempo de execução e possui uma interface gráfica de configuração, possibilitando que as configurações do aglomerado sejam alteradas durante a execução das aplicações. Suporta a maioria do sistemas operacionais e a renderização é feita pela OpenGL (VR JUGGLER, 2011).

O Chromium é um *framework* que permite ao desenvolvedor executar suas aplicações em um aglomerado de modo automático, apenas distribuindo os dados processados pelas funções OpenGL. Essa é uma vantagem em se utilizar esse *framework*, pois outros não possuem esse tipo de vantagem no desenvolvimento (CHROMIUM, 2011).

3.3 AGLOMERADOS GRÁFICOS

3.3.1 Definição

O desenvolvimento de arquiteturas de redes e dos computadores pessoais possibilitou grande avanço na área de processamento de alto desempenho. Sistemas fortemente acoplados, que são

utilizados em sistemas de multiprojeção, podem ser substituídos por AGs em muitas aplicações. O CAVE é um exemplo de aplicação utilizando multiprojeção.

Os aglomerados são caracterizados por um conjunto de nós interligados por uma rede local denominada *Storage Area Network* (SAN), dando ao usuário a impressão de que o processamento é efetuado por um único sistema (GNECCO; GUIMARAES; ZUFFO, 2003). O objetivo é oferecer uma visão múltipla de um mesmo conjunto de dados, sendo que, cada nó deve processar a parte referente à sua tarefa, e então gerar a imagem.

Os nós dos AGs podem ser diferentes uns dos outros, sendo eles: entrada, saída ou processamento. Essa heterogeneidade de nós, em sistemas de RV, é muito importante, pois além de processar e gerar imagens, o sistema deve prover maneiras para que o usuário possa interagir com o AV. Essa interação pode ser feita, por exemplo, utilizando-se celulares ou *Personal Digital Assistant* (PDA) como interface de entrada e saída de um AV, não limitando apenas o uso de computadores para representar nós de AGs (GUIMARAES, 2004).

Sistemas de RV devem realizar tarefas em tempo real. Dessa maneira, ao se utilizar AGs para realizar a mutiprojeção de um ambiente, o sistema deve no mínimo possibilitar a geração de uma taxa de 30 quadros por segundo. Em ambientes que utilizem estereoscopia, essa taxa passa a ser de 60 quadros por segundo, pois para cada olho do usuário deve ser gerada uma imagem de no mínimo 30 quadros por segundo (KIRNER; SISCOUTTO, 2007).

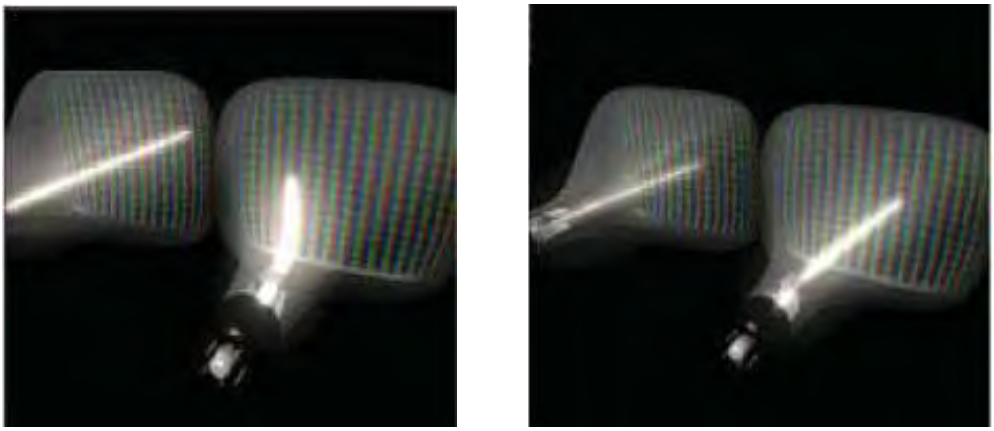
Para que os nós troquem informações, é necessária uma forma de sincronizar o controle de sequência e o controle de acesso. O controle de sequência é utilizado para determinar a ordem de execução dos processos, enquanto o controle de aSilicon Graphicscesso é utilizado quando há competição entre os processos para a manipulação de algum recurso compartilhado (GUIMARAES, 2004).

3.3.2 Sincronização em aglomerados gráficos

O desafio da sincronização em AGs, é que esta deve ser feita dentro de uma limitação de tempo. Existem 3 tipos de sincronização: *genlock*, *datalock* e *framelock* ou *swaplock*.

O tipo de sincronização *genlock* é responsável por controlar a apresentação das imagens. Esse controle pode ser obtido por *hardware* ou *software* (GUIMARAES, 2004; GNECCO; GUIMARAES; ZUFFO, 2003). Esta sincronização assegura que cada pixel em um sistema de visualização esteja sincronizado. A Figura 7a apresenta uma projeção onde não é aplicada a sincronização dos sinais de vídeo. Desse modo, os feixes dos monitores estão projetando os pixels em posições diferentes, portanto, eles não estão sincronizados. A Figura 7b apresenta

2 monitores que possuem sincronismo em suas projeções, onde os feixes projetam no mesmo local. Desse modo, eles estão sincronizados.



(a) Sem *genlock* (b) Com *genlock*

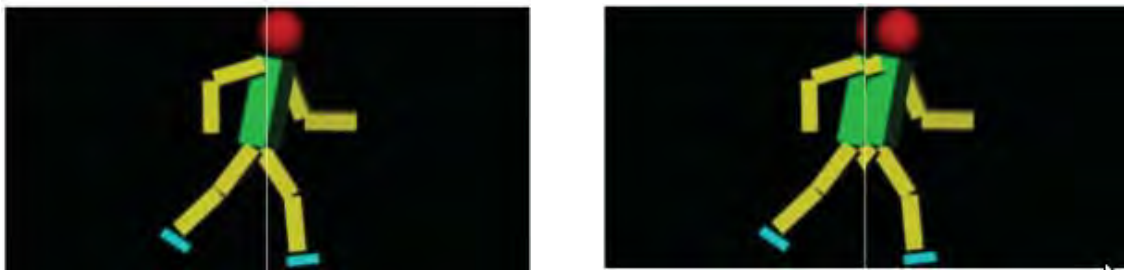
Figura 7: Sincronismo *genlock*
Fonte: (GUIMARAES, 2004)

Este tipo de sincronização está relacionada diretamente a sistemas de estereoscopia ativo, onde as imagens devem ser projetadas de maneira a serem sobrepostas, para que os óculos obturadores possam executar a tarefa de oclusão da lente (CARDOSO, 2003).

A sincronização de dados *datalock* é responsável pelo sincronismo quando há mudança nos dados. Essas mudanças acontecem quando surgem estímulos das aplicações em tempo real, como: eventos de *mouse*, teclado e outros dispositivos não convencionais. Essa sincronização de dados em AGs pode ser dividida em 3 abordagens:

- A distribuição de estímulos acontece quando o usuário, por meio de uma interface de entrada, gera um estímulo. Por exemplo, o pressionamento de uma tecla gera um estímulo, que é enviado para todos os nós Escravos;
- O cálculo centralizado dos resultados e distribuição é responsável pela centralização do tratamento dos estímulos no nó Mestre. Os estímulos gerados pelo usuário são calculados pelo nó Mestre e, só depois, as informações necessárias são transmitidas aos nós Escravos. Assim, o tráfego no meio de transmissão dos AGs é menor, pois é utilizado apenas para transmitir os resultados; e
- O cálculo centralizado dos dados gráficos e distribuição são responsáveis por permitir que o nó Mestre trate os estímulos, realize o particionamento dos dados e envie os resultados para os nós Escravos gerarem suas respectivas imagens.

O tipo de sincronismo *framelock* ou *swaplock* é responsável por tratar a diferença entre as imagens dos nós dos AGs. Segundo Guimaraes (2004), essa diferença pode acontecer quando uma imagem gerada por um nó possui centenas de polígonos, enquanto outro nó pode possuir outra imagem contendo milhares de polígonos. Dessa maneira, existe uma grande probabilidade de um nó terminar a tarefa antes do outro. Por isso, é necessário realizar a sincronização de conclusão dos quadros, garantindo assim, que todos os nós Escravos envolvidos terminem seus respectivos quadros ao mesmo tempo. A Figura 8a apresenta a geração de imagens sem o *framelock* e a Figura 8b apresenta a geração das imagens utilizando o *framelock*.



(a) Com *framelock* (b) Sem *framelock*

Figura 8: Sincronismo *framelock*
Fonte: (SOARES, 2005)

3.3.3 Propriedades das ferramentas baseadas em aglomerados gráficos

Para o desenvolvimento de aplicações de RV baseadas em AGs, deve-se ainda ter mais algumas preocupações com relação ao desempenho, extensibilidade, flexibilidade, simplicidade, robustez, portabilidade e heterogeneidade.

O desempenho tem influência na interatividade e imersão dos AVs. Quanto menor o desempenho, menor será a taxa de apresentação de quadros por segundo, causando assim, problemas conhecidos como *cybersickness* (dores de cabeça, náusea e instabilidade postural) aos usuários.

O ambiente deve também possibilitar a extensão das funcionalidades já existentes, devido ao surgimento de novos recursos que possam ser acrescentados ao sistema. Como exemplo, pode-se ter que adicionar, ao ambiente, os recursos de uma nova placa gráfica. Novos nós dos AGs devem ser acrescentados ao ambiente e devem ser utilizados pelas aplicações. A adição de nós deve ser feita em tempo de execução

Aplicações de RV podem ser complexas devido à necessidade de se adquirir conhecimento de diversas áreas para que se construir os AVs. Porém, o ambiente deve facilitar ao máximo a criação de aplicações, fazendo com que o desenvolvimento seja realizado sem o conhecimento

de todas as particularidades do sistema. O ambiente também deve tratar possíveis falhas de *hardware* e *software*, permitindo que as aplicações continuem a execução mesmo após alguma falha, como na falha de algum nó (GUIMARAES, 2004).

A portabilidade é uma propriedade recomendável a AVs. Ela consiste em facilitar a instalação do ambiente em diferentes arquiteturas de *hardware* e de *software*. Outras características podem ser consideradas, como suporte a dispositivos ou a habilidade de abrir e converter vários formatos de arquivos. O ambiente deve possibilitar a execução em nós heterogêneos. Isto é, nós com diferentes arquiteturas e sistemas operacionais devem ser capazes de executar a mesma aplicação.

3.4 GLASS

A Glass é uma biblioteca que foi desenvolvida pelo Laboratório de Sistemas Integráveis do Departamento de Engenharia da Universidade de São Paulo, denominada incialmente como DICElib (GNECCO et al., 2001). O objetivo principal dessa biblioteca é facilitar o desenvolvimento de aplicações que necessitem de sincronização entre nós de um AG.

A biblioteca consiste em um conjunto escalável de componentes que podem ser utilizados para desenvolver aplicações que utilizem AGs. Ela pode ser utilizada para portar aplicações de RV já utilizadas em sistemas fortemente acoplados.

O principal objetivo da Glass é prover um ambiente de fácil uso, tanto para desenvolver aplicações desde o início, como aplicações já existentes. Diferente de outras soluções disponíveis, a biblioteca não requer grande número de modificações no código fonte e na arquitetura da aplicação.

A biblioteca foi desenvolvida utilizando a linguagem C/C++, uma linguagem orientada a objetos, que possibilita a estruturação das classes na construção de sistemas complexos, e do uso de diferentes tipos de dados de forma simples.

A Figura 9 apresenta uma visão geral dos seus componentes. Inicialmente, tem-se o Arcabouço, que é composto por 3 componentes:

- O componente Instanciação tem por objetivo inicializar as aplicações conforme a arquitetura interna da Glass, que permite criar aplicações Servidor/Cliente ou Mestre/Escravo;
- O componente Protocolo encapsula bibliotecas, escondendo as diferenças entre os protocolos de comunicação *Transmission Control Protocol* (TCP), *User Data Protocol* (UDP), *Message Passing interface* (MPI), entre outros; e

- O componente Plugins fornece serviços de transmissão de eventos, de compartilhamento de dados, de barreiras de sincronização e de associação de funções. Porém, vale ressaltar que a Glass não está restrita apenas a esses *plugins*, podendo ser adicionados outros conforme a necessidade de cada aplicação, contudo, sem a modificação interna da biblioteca.

A biblioteca também possui um componente relacionado a criação ou portabilidade de novas aplicações. Este componente é denominado Aplicações, o qual possibilita o uso de diferentes protocolos, sem que a biblioteca seja modificada ou recompilada. Esse componente possui uma infra-estrutura de empacotamento e desempacotamento de mensagens, que suporta todos os tipos básicos (*integer*, *float*, *string*, entre outros). Este tipo de comportamento garante a interoperabilidade entre sistemas operacionais. Assim, pode-se, por exemplo, realizar execuções de alguns nós da mesma aplicação no Linux, Windows ou Mac OS (GUIMARAES, 2004).

A Glass possui 2 grupos de aplicações, sendo eles: i) as Aplicações Glass criadas pelos desenvolvedores e os programas exemplos; e ii) as Aplicações de Suporte que auxiliam o desenvolvimento aplicações.

As Aplicações Suporte, que possuem a funcionalidade de auxiliar o desenvolvimento de novas aplicações, são:

- O GTracer, que possibilita a visualização de maneira gráfica de todas as mensagens transmitidas entre os nós;
- O GEditor, que permite que aplicações para PDAs sejam geradas automaticamente; e
- O GVoicer, que auxilia o desenvolvimento automático de controle via voz para as aplicações Glass.

Os nós que manuseiam os dados de entrada dos dispositivos não precisam estar executando as aplicações, contudo, eles devem receber as entradas, processar e enviar os resultados para os nós que estão executando a aplicação. Os eventos devem ser recebidos por todos os nós interessados, caso isso não aconteça, incoerência no ambiente podem ocorrer. Ainda com relação a incoerências no ambiente, deve-se fazer o tratamento correto dos dados de entrada para aplicações de multiprojeção, pois o ponto de vista de cada imagem deve ser preciso (GUIMARAES, 2004).

A Glass oferece abstração dos dados e independência de outras bibliotecas. Deste modo, os tipos comuns de dados podem ser criados e manipulados. Entretanto, pode-se escolher outras

bibliotecas para trabalhar em conjunto com a Glass, por exemplo, a biblioteca OpenGL ou Java 3D (GUIMARAES, 2004; DIAS et al., 2010b).

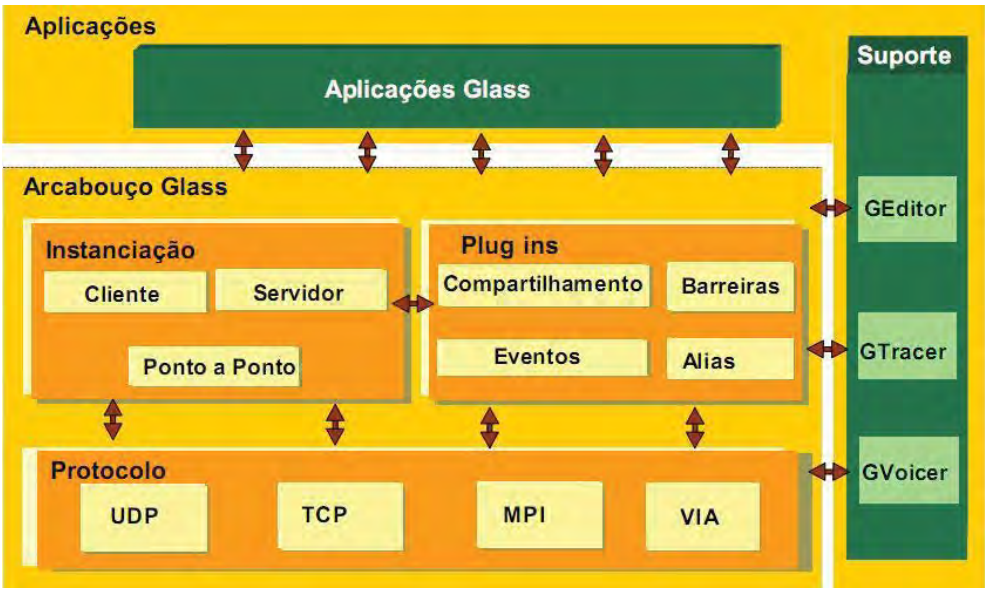


Figura 9: Arquitetura Glass
Fonte: (GUIMARAES, 2004)

O funcionamento interno da Glass pode ser realizado de duas maneiras: Sem replicação ou Com replicação.

Na arquitetura sem replicação, o nó Cliente deve receber as interações do usuário, construir e dividir as primitivas gráficas entre os nós do AG. Depois disso, o nó deve enviar todas as primitivas ao Mestre/Servidor. Quando o Mestre recebe todas as primitivas, os dados devem ser enviados para os nós Escravos, para que gerem e apresentem suas imagens. A Figura 10 apresenta a arquitetura sem replicação.

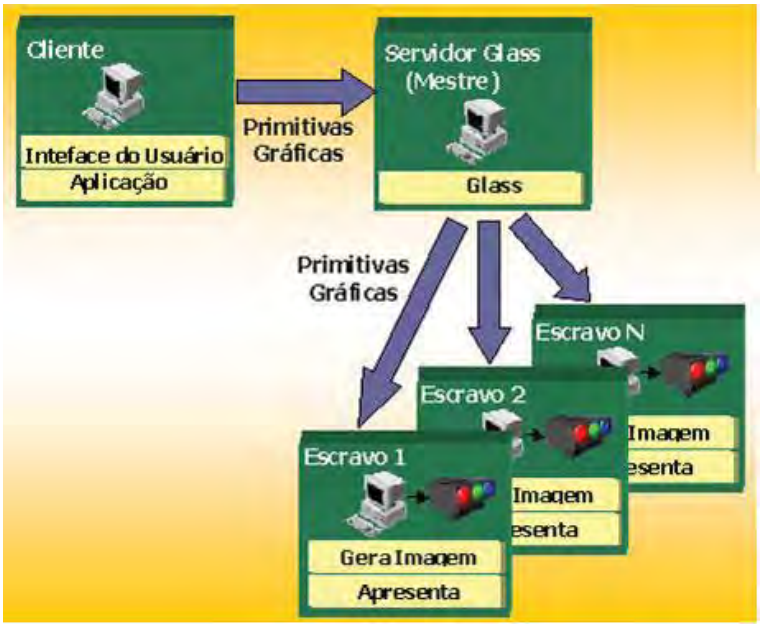


Figura 10: Aplicação Glass na arquitetura sem replicação

Fonte: (GUIMARAES, 2004)

Na arquitetura com replicação, o nó Cliente deve tratar as interações do usuário e gerar as primitivas de controle. Depois as primitivas são enviadas ao Mestre/Servidor, que deve enviar para todos os nós tratarem e gerarem suas imagens. A Figura 11 apresenta a arquitetura com replicação.

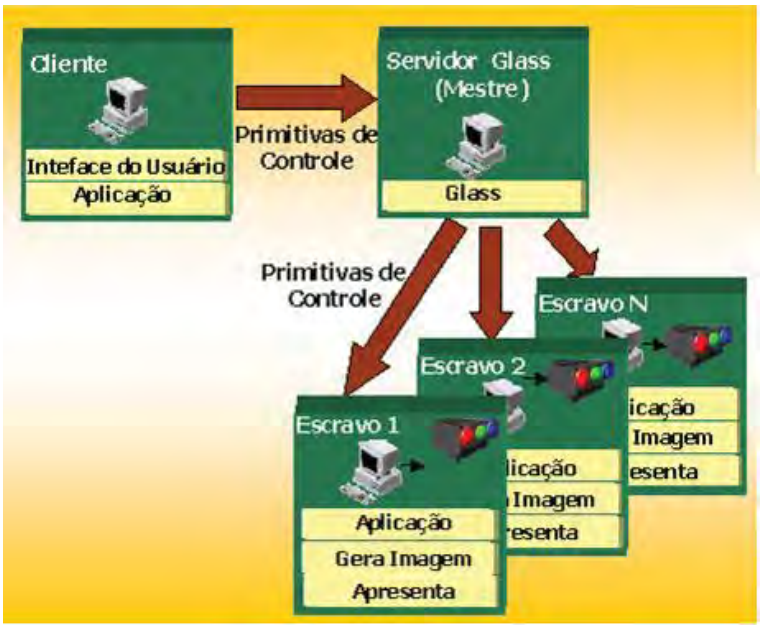


Figura 11: Aplicação Glass na arquitetura com replicação

Fonte: (GUIMARAES, 2004)

3.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou as definições de sistemas de multiprojeção e alguns dos *frameworks* possíveis para o desenvolvimento de tais sistemas.

Dentre as aplicações existentes de sistemas de multiprojeção, foi apresentada uma estrutura de CAVE, que permite ao usuário uma experiência interativa e imersiva com conteúdo 3D apresentado.

Como opção para o desenvolvimento de sistemas de multiprojeção, foram apresentados os AGs. Os AGs disponibilizam alto poder de processamento gráfico e formas de se controlar a geração de imagens, características necessárias ao desenvolvimento deste trabalho.

Portanto, a construção de sistemas de multiprojeção torna-se acessível à instituições de ensino em geral, pois a estrutura baseada em AG pode ser implantada com baixo custo.

4 DESCREVENDO O AMBIENTE VIRTUAL

Um AV pode ser descrito, de maneira que, objetos existentes no ambiente possam ser identificados automaticamente por computadores, ou até mesmo, para faciliar a procura por usuários do AV. Essa descrição pode ser utilizada em aplicações que necessitem obter informações do AV.

Ontologias podem ser utilizadas para descrever conhecimento pertinente a um determinado domínio. Deste modo, a sua utilização como ferramenta para a descrição de AVs pode ser de grande auxílio (DIAS et al., 2010a).

Neste capítulo são apresentados os conceitos e definições sobre a descrição de conhecimento em forma de ontologias. Também são apresentadas linguagens de desenvolvimento de ontologias, como o *Resource Description Framework* (RDF) (MANOLA; MILLER, 2004) e o *Web Ontology Language* (OWL) (MCGUINNESS; HARMELEN, 2004); conceitos do *Extensible 3D* (X3D)(X3D, 2011), linguagem usada para desenvolver AVs para internet. Também é apresentada uma seção sobre Descrições Semânticas em modelos virtuais.

4.1 ONTOLOGIA

Na filosofia, a ontologia lida com a natureza e a organização do ser, sendo o termo atribuído por Aristóteles. Filósofos utilizavam ontologias para tentar responder a perguntas como: “O que é um ser?” e “Quais são as características comuns de todos os seres?” (MAEDCHE, 2002). Existem várias definições distintas para o termo ontologia. Entre elas, podemos citar a definição de Guarino et al. (1993), onde eles afirmam que “uma ontologia é uma especificação formal explícita de uma conceitualização compartilhada”.

Vale ressaltar o significado das palavras utilizadas na definição de Guarino et al. (1993). A palavra “conceitualização” refere-se a um modelo abstrato de um fenômeno que o identifique; a palavra “explícita” refere-se aos tipos de conceitos utilizados, sendo necessário que estes sejam declarados de forma detalhada; a palavra “formal” representa a forma como a ontologia deve

ser processada, isto é, por máquina; a palavra "compartilhada" refere-se à noção de que uma ontologia deve representar um conhecimento consensual, isto é, esse conhecimento não deve estar restrito a grupo pequeno de pessoas, e sim, a grupos maiores (GUARINO et al., 1993).

Outra definição que pode ser utilizada como complemento a definição de Guarino et al. (1993), é a de Swartout et al. (2002), onde ele afirma que: "uma ontologia é um conjunto de termos ordenados hierarquicamente para descrever um domínio que pode ser usado como um esqueleto para uma base de conhecimentos."

Segundo Guarino et al. (1993), uma ontologia pode ser também considerada como um modo de definição de conteúdo específico sobre o conhecimento de um domínio a ser compartilhado e utilizado por agentes. A descrição do conhecimento pode ser dividida em 3 níveis:

- formato de representação da linguagem;
- protocolo de comunicação entre os agentes, que define padrões que devem ser seguidos para que os agentes possam trocar informações entre si; e
- especificação do conteúdo do conhecimento, que é a maneira de como o conhecimento será conceituado.

Uma ontologia pode ser classificada em diferentes categorias, de maneira que diferentes tipos de ontologias possam ser desenvolvidas de acordo com o nível de generalidade necessária. A Figura 12 apresenta 4 tipos de ontologias propostas por Maedche (2002).

Os 4 tipos de classificação de ontologia são:

- Ontologias de alto-nível são usadas para definir conceitos gerais; como espaço, tempo, evento etc. Conceitos gerais são, geralmente, independentes de um domínio. Dessa maneira, uma ontologia de alto-nível pode ser aplicada a grandes comunidades de usuários. Um exemplo de ontologia de alto-nível é uma que descreva meios de transporte;
- Ontologias de Domínio são usadas para descrever vocabulários relacionados a um domínio genérico. Esse tipo de ontologia é uma especialização das ontologias de alto-nível. Um exemplo de ontologia de domínio, especializada a partir de uma ontologia de meio de transportes, é uma que descreva automóveis;
- Ontologias de Tarefa são usadas para descrever vocabulários que estejam diretamente relacionados a uma tarefa ou atividade genérica. Esse tipo de ontologia é uma especialização das Ontologias de alto-nível. Um exemplo de ontologia de tarefa, especializada a

partir de uma ontologia de meio de transportes, é uma que descreva automóveis que sejam terrestres, isto é, devem ser capazes de executar a tarefa de trafegar em vias terrestres; e

- Ontologias de Aplicação são usadas para descrever domínios específicos. Esse tipo de Ontologia pode especializar conceitos tanto das ontologias de domínio, como também das ontologias de tarefas. Um exemplo de uma ontologia de Aplicação, especializada a partir de uma que descreva automóveis, é outra que descreva carros de luxo.

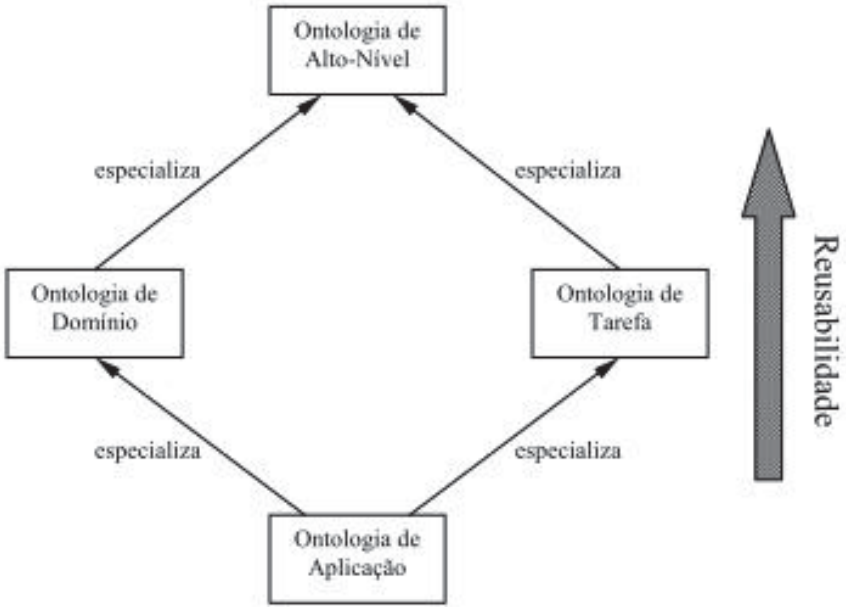


Figura 12: Classificação de ontologias
Fonte: (MAEDCHE, 2002)

Existem algumas vantagens destacadas no uso de ontologias. Uma lista é proposta por Hinz (2006), onde são apresentadas algumas vantagens na sua aplicação na Ciência da Computação. Entre elas, pode-se citar:

- Ontologias devem fornecer um vocabulário para a representação de bases de conhecimentos. Esse vocabulário deve possuir uma conceitualização que evite interpretações ambíguas;

- Devem prover forma de compartilhar o conhecimento. Ontologias que modelem de maneira correta certo domínio podem ser reutilizadas por grupos que necessitem da descrição do mesmo domínio;

- Devem fornecer uma descrição exata do conhecimento, isto é, por ser escrita em linguagem formal, não deve haver espaços para diferentes interpretações;

- A partir de uma Ontologia já proposta, esta deve ser expressa em linguagens diferentes, sem que sua conceituação seja alterada; e
- Uma nova ontologia pode ser estendida a partir de uma Ontologia genérica, de maneira que, esta se adapte a um domínio específico.

Ainda sobre as vantagens de se utilizar ontologias em Ciência da Computação, algumas características relevantes devem ser consideradas quanto ao seu desenvolvimento. Schiessl (2007) as define como:

- Clareza: as definições devem ser completas, viáveis e objetivas, sendo que, todas as definições devem ser documentadas em linguagem natural;
- Coerência: inferências consistentes com as definições devem ser permitidas no desenvolvimento de ontologias;
- Extensível: uma ontologia deve permitir que novos termos sejam adicionados baseando-se no vocabulário já existente, sendo que, não seja necessária a alteração de definições e termos já existentes;
- Mínimo compromisso com implementação: a conceituação deve ser especificada a nível de conhecimento, sem depender de codificação específica; e
- Mínimo compromisso ontológico: uma ontologia deve, no mínimo, prover suporte as atividades de compartilhamento necessárias.

4.2 RESOURCE DESCRIPTION FRAMEWORK

O RDF é um padrão do *World Wide Web Consortium* (W3C) utilizado para codificar o conhecimento, inicialmente proposto para a codificação de metadados (dados que descrevem dados), sendo, atualmente, utilizado para decompor o conhecimento em pequenas partes. Por meio de regras, é possível representar conhecimento sobre: pessoas, lugares entre outros; e também seus relacionamentos (HAYES, 2010).

O projeto do RDF destina-se a satisfazer os seguintes objetivos:

- Um modelo simples de dados;
- Semântica formal;

- Vocabulário baseado em *Uniform Resource Identifier* (URI);
- Sintaxe baseada em XML;
- Tipo de Dados XML Schema; e
- Permite que declarações sejam feitas a partir de qualquer recurso.

O modelo de tipos de dados do RDF é simples para que aplicações possam processar e manipular. O modelo de dados é independente de qualquer serialização específica de sintaxe. O RDF tem uma semântica formal, que prove uma base confiável para raciocinar sobre o significado de expressões. O vocabulário é baseado em URIs, tendo fragmentos identificadores (referências URI). As referências URI são utilizadas para nomear todas os recursos em RDF. A sintaxe é baseada no padrão do XML, podendo ser usado para codificar modelos de dados para a troca de informação entre as aplicações. Os tipo de dados XML Schema pode ser usado para representar dados em RDF, auxiliando na troca de informação entre aplicações RDF e XML.

O *framework* permite que qualquer pessoa faça declarações sobre qualquer recurso, entretanto, as declarações, apesar de serem livres a qualquer pessoa, não são livres de inconsistências. O RDF não previne essa prática. Desta maneira, a declaração feita por alguém pode ser equivocada.

A estrutura das expressões em RDF deve ser formalizada em forma de triplas, cada uma composta de um sujeito, um predicado e um objeto. A Figura 13 ilustra um exemplo de tripla RDF.

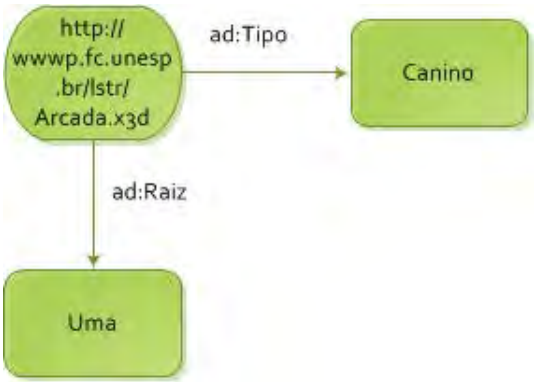


Figura 13: Tripla RDF

Cada tripla representa um relacionamento entre um conhecimento especificado, sendo que, a direção do arco é um ponto significante. O ponto deve sempre apontar para o objeto. Um nó deve possuir uma URI que possa identificá-lo, sendo um literal ou um nó em branco. Uma URI

de referência ou um literal é usada para identificar o que o nó representa. A URI utilizada como predicado identifica o relacionamento entre o que esta sendo representado por nós conectados. Quando uma URI assume o papel de predicado pode ser usado como um nó no grafo (KLYNE; CARROL, 2004).

Os tipos de dados são usados em RDF para representar os valores possíveis como: inteiro, ponto flutuante e datas. Um tipo de dado consiste em um espaço léxico, sendo esse dividido em: valor de espaço e mapeamento léxico para o valor. Por exemplo, o mapeamento léxico para o valor para tipos de dados XML Schema, onde cada elemento do espaço valor é representado por T e F, tem duas representações léxicas. A Figura 14 ilustra o mapeamento.

Value Space	{T, F}
Lexical Space	{"0", "1", "true", "false"}
Lexical-to-Value Mapping	{<"true", T>, <"1", T>, <"0", F>, <"false", F>}

Figura 14: Mapeamento do léxico para valor
Fonte: (KLYNE; CARROL, 2004)

O RDF não permite que novos tipos de dados sejam definidos. Para que sejam definidos outros tipos de dados, devem ser utilizados tipos de dados XML Schema, que prove uma extensibilidade que pode ser aplicada para o uso em RDF (BRICLKEY; GUHA, 2004).

Os literais são usados para identificar valores, tais como, números e datas que representam significados léxicos. Qualquer coisa representada por um literal pode também ser representado por uma URI, mas o uso de literais é mais intuitivo. Um literal deve ser utilizado como um objeto de uma declaração RDF, mas nunca como sujeito ou predicado.

As triplas RDF devem representar fatos que indiquem um relacionamento entre duas entidades. Esse relacionamento pode ser representado da mesma maneira que um banco de dados relacional, onde as colunas são definidas como: sujeito e objeto; o nome da tabela corresponde ao predicado da tripla RDF.

4.3 X3D

O X3D (X3D, 2011) é um padrão para a definição e comunicação em tempo real, interagindo com conteúdo 3D para modelagem de ambientes virtuais; o X3D prove codificação utilizando a sintaxe XML, sendo utilizado tanto em ambiente *web*, quanto aplicações *desktop*.

O padrão X3D foi criado para adicionar novas características ao seu antecessor, o *Virtual*

Reality Modeling Language (VRML) (VRML, 2011). Além de que, foram também adicionadas novas APIs, novo formato de codificação, conformidades mais rigorosas e uma arquitetura de componentes baseada em perfis, que prove compatibilidade com o VRML.

A utilização da codificação X3D para a realização de descrições semânticas advém da facilidade que a linguagem XML prove aos desenvolvedores; vários *frameworks* existentes na literatura, permitem que descrições semânticas sejam efetuadas em documentos que possuam sintaxe XML.

4.4 DESCRIÇÕES SEMÂNTICAS DE AMBIENTES 3D

Os ambientes 3D são caracterizados pela modelagem e composição de elementos geométricos poligonais ou, na maioria dos ambientes, por objetos relacionados às superfícies *Non Uniform Rational Basis Spline* (NURBS). Porém, esses ambientes são limitados quanto à sua descrição semântica. Isto é, não se tem nenhuma informação referente aos elementos desse ambiente que possa ser compartilhada com o visitante.

A descrição semântica de ambientes 3D fornece um meio de facilitar a extração de objetos semânticos que façam parte de ambientes complexos, possibilitando assim, que esses objetos possam ser examinados de maneira eficiente. Os objetos anotados semanticamente podem auxiliar na geração de bibliotecas de alto nível, que são utilizadas para criar ambientes 3D diferentes utilizando esses objetos. Esta prática também pode ser utilizada para auxiliar motores de busca, processando consultas formuladas em linguagem natural, com o intuito de obter características relevantes sobre o ambiente 3D, como: encontrar os dentes incisivos em um ambiente 3D que represente uma arcada dentária.

Uma maneira de realizar anotações semânticas em ambientes 3D é utilizando-se de conceitos de *web* Semântica (BERNERS-LEE; HENDLER, 2001). Esses conceitos podem ser aplicados utilizando-se:

- RDF: responsável por criar conjuntos de regras semânticas que geram descrições, sendo que a partir dessas é gerado o RDF Schema, responsável por criar vocabulários; e
- OWL: é responsável por implementar as ontologias. Possui maior nível de semântica quanto ao RDF, porém, ambos utilizam sintaxe XML.

Alguns trabalhos correlatos são citados na literatura, sendo que, a maioria utiliza tecnologias X3D e conceitos de anotações semânticas relacionadas à *web* semântica. Vale ressaltar que,

nenhum trabalho correlato apresentado agrega conhecimentos sobre as áreas: médica/odontológica, RV e anotações semânticas. Os trabalhos apresentados nessa seção visam apresentar exemplos que possibilitem a realização de descrições semânticas em documentos multimídias em geral, podendo ser ambientes virtuais ou não. Entre eles, pode-se citar:

- O grupo MPEG (2011), que definiu padrões para codificar e descrever os dados. Alguns dos padrões criados pelo grupo são: o *Moving Picture Experts Group* (MPEG) 4 e 7. O MPEG-4 considera um documento multimídia como um conjunto de diferentes objetos que o desenvolvedor/usuário pode interagir; existe uma extensão do MPEG-4, onde ele passou a suportar descrições de objetos utilizando sintaxe XML, que foi denominada XMT-A. O MPEG-7, da mesma maneira que o MPEG-4, é utilizado para descrever documentos multimídia, mas este utiliza diferentes graus de interpretação para o significado da informação;

- Bilasco e Gensel (2006) propuseram uma ferramenta que permite o reuso de modelos 3D. Um modelo genérico de anotação semântica é utilizado, denominado 3DSEAM, sendo que, esse modelo permite que modelos 3D sejam indexados considerando seu conteúdo visual, geométrico e aspectos semânticos. A ferramenta foi denominada 3DAF, e utiliza o MPEG-7 como padrão nas anotações; e

- Outro trabalho é o de Pittarello e Faveri (2006), que prove uma maneira para que desenvolvedores de ambientes 3D possam selecionar e extrair objetos semânticos, ou gerar buscas que se refiram a um alto nível de propriedade do ambiente. Este trabalho propôs uma abordagem alternativa para a realização de anotações semânticas em ambientes 3D, integrando 2 padrões da *web*: a linguagem X3D e a *web* semântica. Pittarello utiliza uma ontologia de apoio para realizar as anotações; e define zonas semânticas para complementar a descrição dos objetos em um AV.

4.5 CONSIDERAÇÕES FINAIS

A descrição de AV utilizando ontologias permite que estes sejam utilizados na construção de novos AV. Além disso, neste trabalho documentos multimídia podem ser integrados aos AV com o intuíto de gerar um nível maior de informação ao usuário.

Este capítulo apresentou algumas definições de ontologia e suas aplicações. Algumas vantagens foram apresentadas sobre o uso de ontologias em Ciência da Computação.

Foram apresentados alguns padrões utilizados na descrição de documentos multímidias,

dentre eles, o padrão proposto por Pittarello e Faveri (2006), no qual foi baseada a forma de descrição utilizada nesta dissertação.

5 SISTEMA AVANÇADO DE REALIDADE VIRTUAL

Este capítulo apresenta o desenvolvimento da estrutura de multiprojeção implantada durante o desenvolvimento deste trabalho. Dentre as etapas de pesquisa, tem-se: a proposta e implantação da estrutura de multiprojeção, contendo medidas, especificações e configurações dos dispositivos utilizados, e o desenvolvimento da aplicação Java, contendo o Browser X3D distribuído responsável pela renderização de cena e interação com o AG (barreiras de sincronismo, distribuição de dados etc)(DIAS et al., 2010b).

5.1 MINI CAVE

Para o desenvolvimento deste projeto foi necessária a implantação de uma estrutura de multiprojeção não convencional de baixo custo. A estrutura foi implantada no Laboratório de Sistemas de Tempo Real (LSTR), da Universidade Estadual Paulista. A estrutura é composta por dispositivos, como:

- 6 *Personal Computer* (PC): Intel Core i7. 8 gb de RAM. 500 gb de HD. Placas Gráficas NVIDIA FX Quadro 1800. Placa de Rede 1 gigabit;
- 1 Switch Gigabit 3Comm;
- 6 projetores de alta definição BenQ W1000;
- 3 pares de lentes polarizadoras;
- 3 telas de alto brilho, com medidas de 2,5m x 1,5m;
- 3 suportes (de teto) para projetores, contendo bandejas para 2 projetores;
- 6 cabos de vídeo D-Sub blindados com 10 m cada; e
- 6 cabos de controle Serial com 10m cada.

Para a implantação da estrutura proposta, foi necessária uma sala de tamanho amplo, devido ao tamanho das telas e a distância mínima dos projetores. A planta baixa do projeto é apresentada na Figura 15.

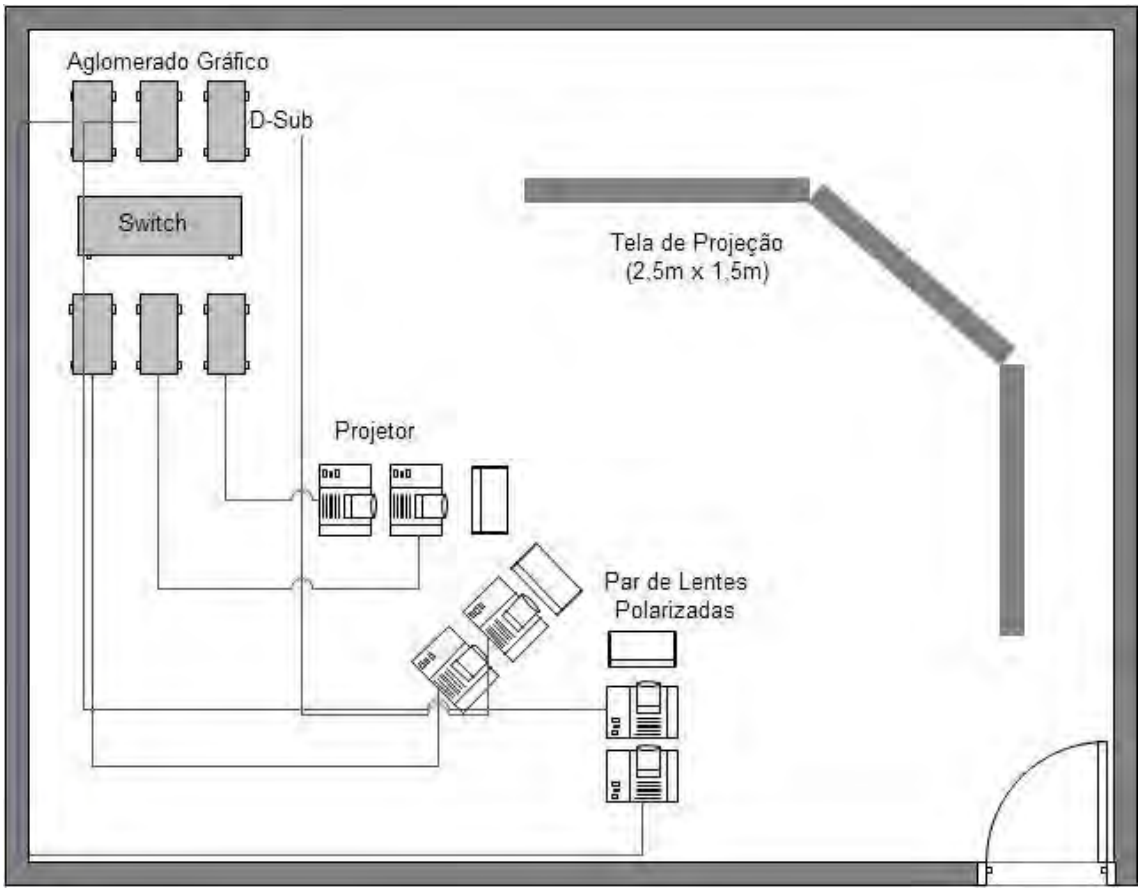


Figura 15: Planta baixa da estrutura de multiprojeção não convencional desenvolvida no projeto

Um aglomerado contendo 6 PC foi utilizado, visando fornecer um alto poder de processamento para a realização das renderizações necessárias as aplicações desenvolvidas. A Figura 16 apresenta o AG proposto neste projeto. Os detalhes de configuração do AG são descritos no Apêndice A.

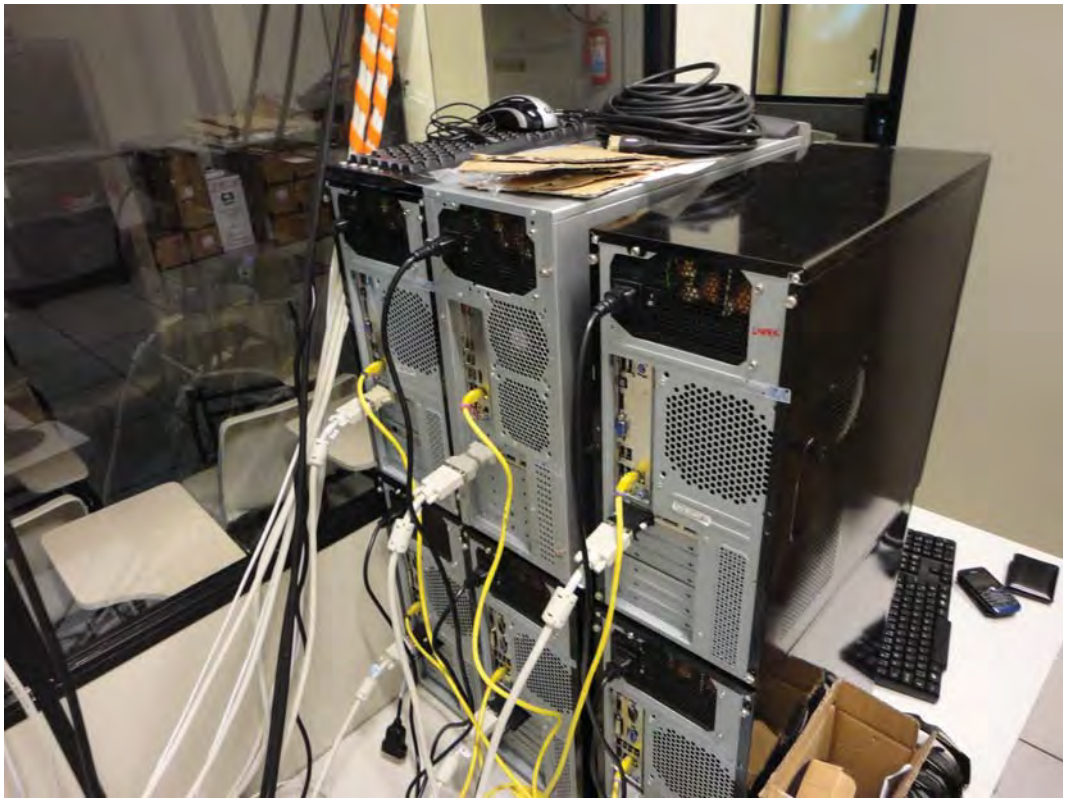


Figura 16: Aglomerado gráfico utilizado

Para a geração de imagens do aglomerado, foram utilizados 6 projetores de alta definição. O uso de 6 projetores é devido o sistema possibilitar o uso de estereoscopia passiva. Dessa maneira, torna-se necessário o uso de 2 projetores para cada tela do sistema, um gerando imagem para o olho direito e, o outro, para o olho esquerdo. As duas imagens são sobrepostas. Com as duas imagens sendo geradas sobrepostas, surge a necessidade de se "filtrar" os sinais gerados pelos projetores. Portanto, lentes polarizadoras são posicionadas nas saídas dos projetores, havendo assim, uma diferenciação nos sinais emitidos. A Figura 17 apresenta as lentes polarizadoras utilizadas.

Os projetores e lentes são posicionados sobre um suporte capaz de posicionar 2 projetores, um sobre o outro, possibilitando assim, a geração das imagens sobrepostas. A Figura 18 apresenta o suporte utilizado.

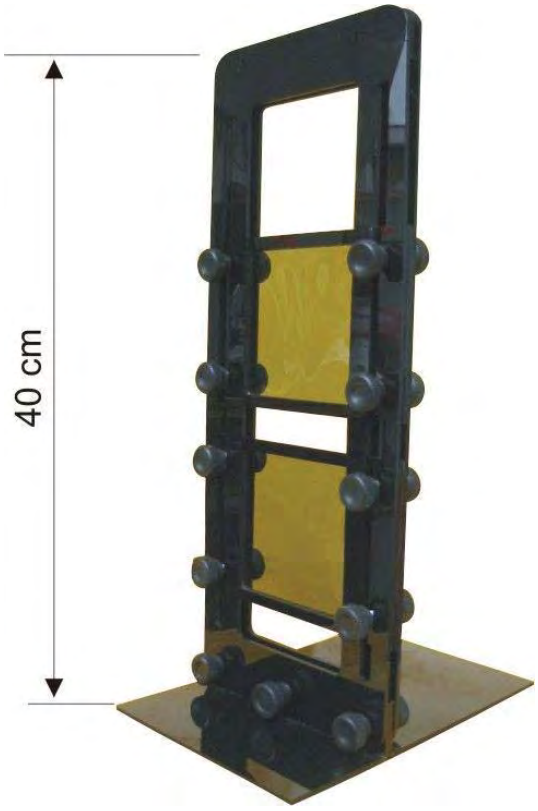


Figura 17: Lente polarizadora
Fonte: (TECNOGLASSES, 2011)

Figura 18: Suporte utilizado para o posicionamento correto dos projetores e lentes polarizadoras

As lentes polarizadoras geram o sinal polarizado. Porém, as telas são tão importantes quanto às lentes, pois elas possuem um revestimento de alto brilho que reflete as projeções

aos óculos polarizados. A Figura 19 apresenta a estrutura compostas por 3 telas.

Figura 19: Telas de alto brilho

5.2 JAVA 3D PARA SISTEMAS DE MULTIPROJEÇÃO UTILIZANDO A GLASS

Visando utilizar a estrutura do Mini CAVE implantado, esta seção apresenta os passos de desenvolvimento de uma aplicação Java 3D capaz de renderizar AVs de modo distribuído, fazendo assim, necessário um AG para a realização das tarefas de renderização.

Nesta seção, são apresentados os passos de desenvolvimento da aplicação Java 3D utilizando a biblioteca Glass para o compartilhamento de dados de posicionamento do AV. Os AVs utilizados pelo sistema foram desenvolvidos no formato X3D. O *loader* Xj3D (XJ3D, 2011) foi escolhido para gerar o grafo de cena do conteúdo X3D à aplicação Java 3D.

A Figura 20 apresenta o relacionamento entre as classes Java e os objetos Glass. O servidor Glass é responsável por sicronizar e distribuír dados compartilhados para os nós clientes. A camada Java 3D é executada sobre uma máquina virtual, o que a torna independente de plataforma. Uma camada cliente Glass é reponsável pela comunicação entre os nós clientes e o servidor.

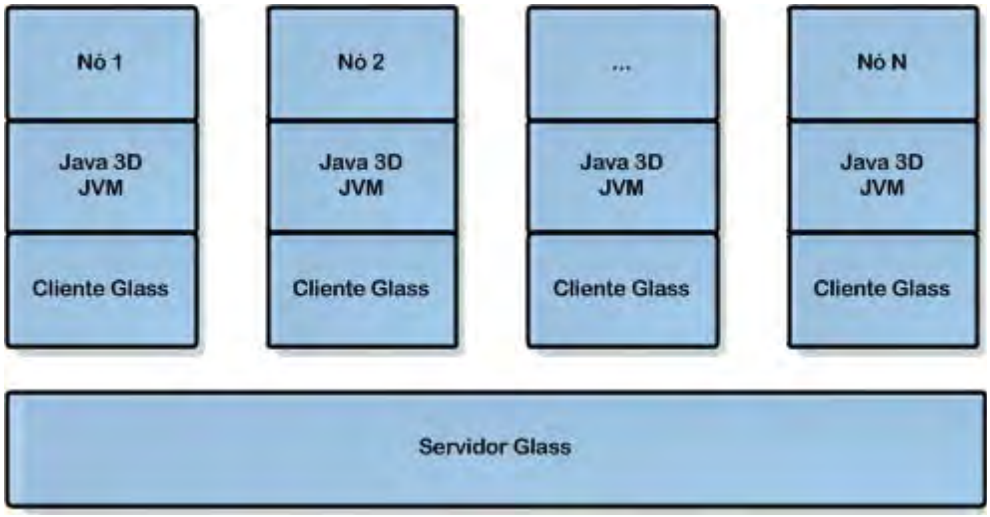


Figura 20: Relacionamento Java 3D e objetos Glass

5.2.1 Objetos Glass

Primeiramente, são criados os objetos Glass, dos tipos: GlassServer, GlassClient, Barrier, Shared, Event e Alias. Porém, para que esses objetos fossem utilizados na linguagem Java, foi necessário a criação de *bindings* JNI, responsáveis por gerar as bibliotecas nativas para a utilização no desenvolvimento de aplicações Java. As funcionalidades dos objetos Glass são:

- O objeto do tipo GlassServer, responsável por sincronizar e distribuir os dados do AG. É um nó servidor;
- O objeto do tipo GlassClient, responsável por instanciar a aplicação Java 3D como nó cliente do AG;
- O objeto do tipo Barrier,que possibilita o sincronismo entre as aplicações Java 3D executadas pelos nós do AG;
- O objeto do tipo Shared, responsável pelo compartilhamento de dados das coordenadas do AV;
- O objeto do tipo Event, capaz de compartilhar eventos; e
- O objeto do tipo Alias, utilizado para a configuração e nomeação dos nós.

Uma classe foi implementada para a declaração e instanciação dos objetos, a classe **Glass**. Para que este módulo possa ser apresentado, trechos de código foram utilizados no intuito de

facilitar o entendimento. A Figura 21 apresenta o Diagrama de Classes Glass, contendo todos seus objetos e métodos. Esses componentes são descritos no decorrer deste capítulo.

Figura 21: Diagrama de classe Glass

Os pacotes utilizados no desenvolvimento do classe **Glass** são apresentados no trecho Código 5.1

5.1: Pacotes utilizados no desenvolvimento da classe Glass

```
package xj3d;

import Glass.Alias;
import Glass.Barrier;
import Glass.Event;
import Glass.GlassClient;
import Glass.GlassException;
import Glass.GlassServer;
```

```
import Glass.Shared;
import Glass.TCP;
import java.io.IOException;
import javax.vecmath.Matrix4d;
import javax.vecmath.Vector3d;
```

Os objetos declarados são provenientes da biblioteca Glass. Os objetos são do tipo GlassClient, GlassServer, Barrier, Shared, Event, Alias. O objeto targetnode é responsável por definir qual será a quantidade de nós cliente. O trecho de Código 5.2 apresenta a declaração dos objetos.

5.2: Declaração dos objetos Glass

```
public class Glass {

    private GlassClient g;
    private GlassServer gs;
    private Barrier datalock;
    private Barrier framelock;
    private Shared<Double> transx;
    private Shared<Double> transy;
    private Shared<Double> transz;
    private Event<Matrix4d> event;
    private Event<Vector3d> eventvec;
    private Alias<String> view_set;
    private int target_node = 6;

}
```

A classe **Glass** possui 2 construtores: Glass( ) e Glass(String Server). O construtor Glass( ) é executado sem nenhum parâmetro, criando assim, um nó do tipo servidor. O construtor Glass(String Server) recebe como parâmetro o endereço *Internet Protocol* (IP) do nó servidor, sendo executado pelos nós clientes.

Quando um objeto servidor é criado, esse fica a esperar conexões dos nós clientes. Dessa maneira, é aconselhável que o nó servidor seja executado em um computador que não seja utilizado na renderização do AV. Porém, um computador pode executar mais de uma instância da aplicação, sendo assim, cliente e servidor ao mesmo tempo. Os trechos de Código 5.3 e Código 5.4 apresentam, respectivamente, os construtores para objeto servidor e cliente.

5.3: Construtor servidor Glass

```
public Glass() {
```

```
        System.out.println("Starting server");
        try {
            gs = new GlassServer(new TCP());
            System.out.println("Servidor aguardando conexoes...");
        } catch (GlassException e) {
            System.out.println("Error: Server initialization: " + e);
        }

    }
```

5.4: Construtor cliente Glass

```
public Glass(String ip) {
        glassInitialization(ip);
    }
```

As aplicações Java que utilizam a Glass devem implementar um método de inicialização dos componentes distribuídos. O método glassInitialization(String server) é responsável por esse processo. Existe um parâmetro de entrada para o método, o *String server*. Por meio desse, é possível atribuir o endereço do servidor Glass, isto é, todos os nós cliente iniciam o método recebendo como parâmetro o endereço ip do servidor e instanciando um objeto do tipo cliente.

Como mencionado, se o construtor da classe **Glass** for executado sem parâmetro de entrada, o nó será executado como servidor, isto é, ele será responsável por gerenciar os outros nós do AG. Dessa maneira, o nó servidor deve ser o primeiro a ser instanciado. Os nós cliente são instanciados logo após a definição do nó servidor.

Os objetos do tipo Barrier, *datalock* e *framelock*, são instanciados com id (número de identificação) diferentes. O objeto *datalock* recebe como parâmetro o id 1 e o objeto *framelock* recebe como parâmetro o id 2. Os objetos anglex, angley, transx, transy, transz são responsáveis pelas primitivas de controle da aplicação, sendo objetos de translação e rotação. Os nomes entre aspas, por exemplo, **"anglex"**, representam os nomes que os objetos compartilhados tem perante os outros nós do AG. Os objetos do tipo Event também são responsáveis pelas primitivas de controle. Estes recebem a matriz ou vetor de transformação dos objetos Java 3D. O trecho de Código 5.5 apresenta o método glassInitialization.

5.5: Método de inicialização da Glass

```
    private void glassInitialization(String server) {

        try {
            if (server.equals("")) {
```

```
            System.out.println("Starting server");
            GlassServer gs = new GlassServer(new TCPT());
        } else {
            System.out.println("Connecting to: " + server);
            setG(new GlassClient(new TCP(), server));

        } /* create a barrier for swaplock */
        this.setDatalock(new Barrier(1));
        this.setFramelock(new Barrier(2));

        /* create our shared data, and set its value */

        this.setTransx(new Shared<Double>("transx", null));
        this.setTransy(new Shared<Double>("transy", null));
        this.setTransz(new Shared<Double>("transz", null));
        this.setEvent(new Event<Matrix4d>("event"));
        this.setEventvec(new Event<Vector3d>("eventvec"));
```

A aplicação desenvolvida utiliza uma arquitetura com replicação. Dessa maneira, o que diferencia um nó cliente de outro são seus posicionamentos de câmeras perante o AV. Em uma estrutura como a que foi proposta, deve-se utilizar 3 câmeras, isto é, uma para cada tela (esquerda, centro e direita).

A instanciação dos objetos do tipo Alias também é executada no método glassInitialization(). Os Alias são definidos pelo objeto viewset. Este possui o método addAlias(), onde são passados os apelidos que serão atribuídos aos nós do AG.

São geradas duas câmeras para cada tela do sistema de multiprojeção, pois como o sistema possibilita o uso de estereoscopia, deve-se ter uma câmera para cada olho. O trecho de Código 5.6 apresenta a criação dos Alias.

5.6: Associação de Alias

```
/* create and set our view set function alias */
        setView_set(new Alias<String>("view_set", " "));
        getView_set().addAlias("rightC", "rightC");
        getView_set().addAlias("rightB", "rightB");

        getView_set().addAlias("leftC", "leftC");
        getView_set().addAlias("leftB", "leftB");

        getView_set().addAlias("frontC", "frontC");
        getView_set().addAlias("frontB", "frontB");
```

Depois dos objetos Alias criados, estes devem ser associados a cada nó do AG. Dessa maneira, é possível atribuir diferentes apelidos para os nós cliente. O método associate() recebe como parâmetro o id e o alias ao qual o nó deve ser associado. Essa associação de apelidos e id dos nós clientes é feita pelo primeiro nó cliente instanciado, nesse exemplo, o nó com id 65537. O primeiro nó cliente instanciado é identificado pelo método isMaster(). O trecho de Código 5.7 apresenta a associação dos nós e seus apelidos.

5.7: Associação de Alias com ID

```
if (getG().isMaster()) {
                getView_set().associate(65537, "leftB");
                getView_set().associate(65538, "leftC");
                getView_set().associate(65539, "frontB");
                getView_set().associate(65540, "frontC");
                getView_set().associate(65541, "rightB");
                getView_set().associate(65542, "rightC");

            }
```

5.2.2 Sincronização dos nós

A sincronização dos nós do AG é importante para a integridade da multiprojeção, pois os AVs devem ser renderizados ao mesmo tempo, ou seja, com a mesma taxa de quadros por segundo.

A sicronização neste trabalho é feita por *software*, isto é, objetos do tipo Barrier são utilizados. Assim, sempre que um nó cliente estiver no ponto de sincronismo, este ficará em espera até que todos os outros nós cliente estejam no mesmo ponto, para nesse momento possam realizar a renderização. Dessa forma, a taxa de quadros é mantida, independente da quantidade de polígonos que cada nó esteja renderizando. A Figura 22 apresenta o processo de sincronização. O processo de sincronismo é apresentado em detalhes no decorrer desta subseção.

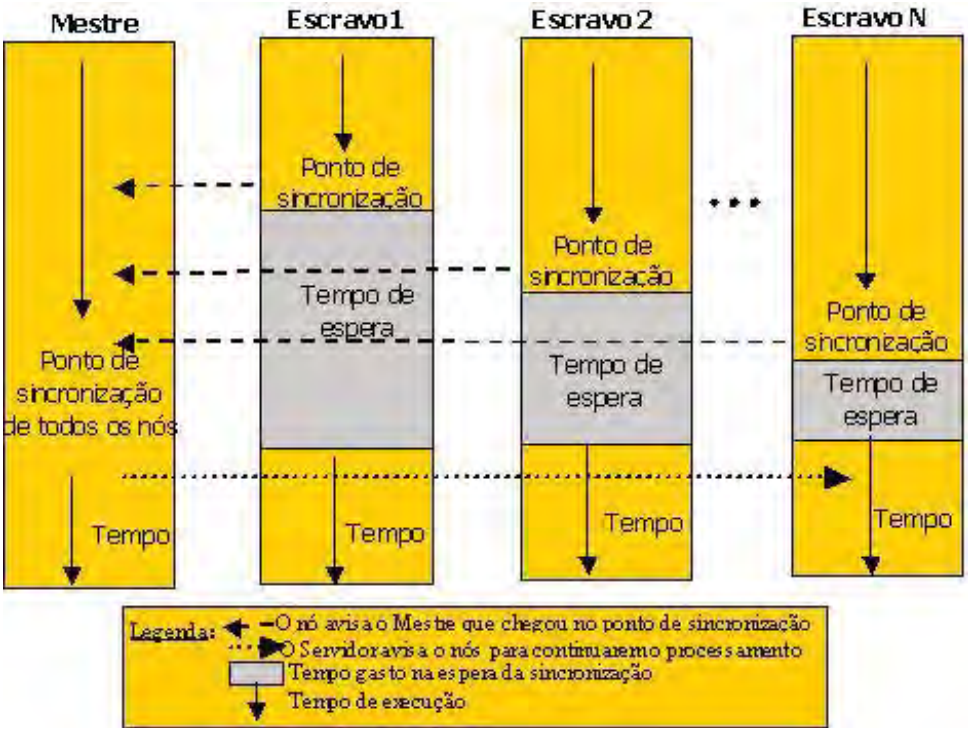


Figura 22: Processo de sincronização

Fonte: (GUIMARAES, 2004)

O AV, depois de ter o grafo de cena gerado, passa por constantes transformações, como translações e rotações. Sempre que o ambiente receber alguma interação do usuário, os objetos compartilhados são atualizados. O método setdata() é utilizado para alterar o valor dos objetos distribuídos responsáveis pelas transformações no AV. Assim, o AV tem, inicialmente, as posições X, Y e Z definidas pela aplicação. O trecho de Código 5.8 apresenta a chamada do método setdata() na inicialização da classe **Glass**. O método datachanged() será sempre executado após a alteração dos valores dos objetos distribuídos. No momento que os objetos compartilhados executam o método sendUpdate(), os valores atualizados são enviados ao nó servidor, para que esses possam ser sincronizados e enviados a todos os nós cliente.

5.8: Método para atualizar valor dos objetos distribuídos

```
getTransx().setData(0.0);
getTransy().setData(0.0);
getTransz().setData(2.0);

try {
    data_changed();
} catch (IOException e) {
    // TODO
}
```

Quando os objetos são atualizados, todos os nós são sincronizados, para que no momento da renderização todos os nós do AG realizem suas tarefas ao mesmo tempo. Por isso, o método my_sync() possui uma chamada ao objeto Barrier *datalock*, responsável pelo sincronismo dos dados. O objeto Barrier *datalock*, por meio de seu método sync(), fica em modo de espera até que todos os nós tenham executado o mesmo método. Depois que todos os nós são sincronizados, os objetos de transformação do AV são atualizados localmente por meio do método getupdate() e, só depois, que ocorre a renderização em todos os nós cliente.

O trecho de Código 5.9 apresenta o método my_sync(), responsável pelo sincronismo dos nós cliente do AG na aplicação desenvolvida.

5.9: Método de sincronismo dos objetos Glass

```
public void my_sync() throws IOException, ClassNotFoundException {
        getDatalock().sync();

        this.getTransx().getUpdate();
        this.getTransy().getUpdate();
        this.getTransz().getUpdate();

    }
```

5.2.3 Java 3D para sistemas de multiprojeção

Esta seção apresenta os objetos e métodos provenientes da classe **Render**. Essa classe é responsável por carregar e renderizar o ambiente. Seus objetos utilizam o sincronismo provido pela classe **Glass**. A Figura 23 apresenta o Diagrama de Classe **Render**.

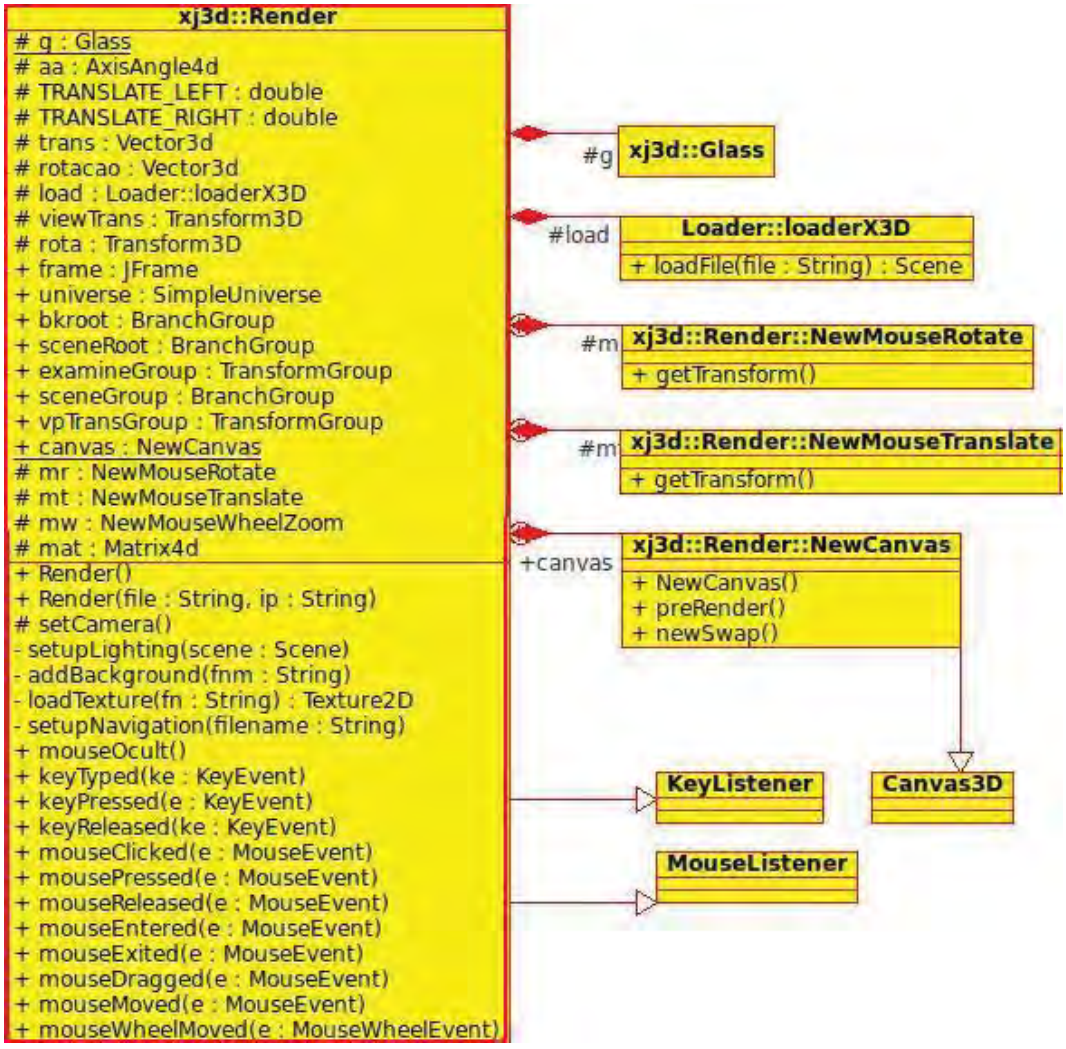


Figura 23: Diagrama de classe Render

5.2.4 Configuração das câmeras

Os AV utilizados pelo sistema são arquivos do tipo X3D. Portanto, para que esses arquivos sejam renderizados, deve ser gerado um grafo de cena com seu conteúdo. O objeto load, instanciado a partir da classe **Loader**, é responsável por carregar o modelo virtual.

A configuração das câmeras para cada olho é feita na inicialização da classe. O objeto PhysycalBody configura a posição dos olhos por meio de duas translações no eixo X, sendo uma para cada olho. O trecho de Código 5.10 apresenta a forma como é feita a associação.

5.10: Configuração de paralaxe

```
View view0 = universe.getViewer().getView();

View view = new View();
PhysicalBody myBod = view0.getPhysicalBody();
```

```
myBod.setLeftEyePosition(
        new Point3d(+.0045, 0.0, 0.0));
myBod.setRightEyePosition(
        new Point3d(-.0045, 0.0, 0.0));
view.setPhysicalBody(myBod);
view.setPhysicalEnvironment(
        view0.getPhysicalEnvironment());
view.attachViewPlatform(
        universe.getViewingPlatform().getViewPlatform());
```

As câmeras devem ser associadas aos nós cliente. Dessa maneira, os alias definidos anteriormente na classe **Glass** são associados aos nós de projeção, configurando a posição dos olhos para que a visualização possa ser apresentada utilizando estereoscopia. O objeto yMatrix é responsável por realizar a rotação da câmera no eixo Y, fazendo assim, com que seja gerada 3 posições de câmeras diferentes. O trecho de Código 5.11 apresenta a configuração das câmeras da tela da esquerda, para os olhos direito e esquerdo.

5.11: Configuração de câmera da tela esquerda

```
protected void setCamera() {

if ((g.getView_set().getData() == "leftC")) {
    canvas.setMonoscopicViewPolicy(View.RIGHT_EYE_VIEW);

    Vector3d vec = new Vector3d(2.0, 0.0, 0.0);
    Matrix3d getMatrix = new Matrix3d();
    Matrix3d yMatrix = new Matrix3d();
    viewTrans.getRotationScale(getMatrix);
    viewTrans.set(vec);
    yMatrix.rotY(Math.PI / 4.0);
    yMatrix.mul(getMatrix);
    viewTrans.setRotation(yMatrix);
    vpTransGroup.setTransform(viewTrans);

}
if ((g.getView_set().getData() == "leftB")) {
    canvas.setMonoscopicViewPolicy(View.LEFT_EYE_VIEW);

    Matrix3d getMatrix = new Matrix3d();
    Matrix3d yMatrix = new Matrix3d();
    viewTrans.getRotationScale(getMatrix);
    yMatrix.rotY(Math.PI / 4.0);
    yMatrix.mul(getMatrix);
```

```
        viewTrans.setRotation(yMatrix);
        vpTransGroup.setTransform(viewTrans);

    }
```

5.2.5 Controle de interação com o ambiente virtual

O método responsável por alterar as coordenadas X, Y e Z do AV é o getTransform(), sendo este sobrecarregado da classe **MouseTranslate**. Toda vez que o usuário necessitar modificar o estado do ambiente, as novas coordenadas geradas são compartilhadas aos nós objetos Glass, para que os outros nós também possam ser alterados. O método getTransform() é capaz de gerar valores para a translação do AV em 2 eixos, X e Y. Os objetos, com seus valores atualizados, são enviados no decorrer do método, porém, o sincronismo se dá somente no momento da renderização.

5.12: Método de translação da aplicação via *mouse*

```
public class NewMouseTranslate extends MouseTranslate {

    public void getTransform() {
        Vector3d vec = new Vector3d();
        viewTrans.mul(viewTrans, transformX);
        viewTrans.get(vec);


        java.text.NumberFormat nfx = java.text.NumberFormat.
            getNumberInstance();
        nfx.setMinimumFractionDigits(2);
        nfx.setMaximumFractionDigits(2);
        String strx = nfx.format(vec.x);
        strx = strx.replaceAll(",", ".");
        vec.x = Double.parseDouble(strx);

        java.text.NumberFormat nfy = java.text.NumberFormat.
            getNumberInstance();
        nfy.setMinimumFractionDigits(2);
        nfy.setMaximumFractionDigits(2);
        String stry = nfy.format(vec.y);
        stry = stry.replaceAll(",", ".");
        vec.y = Double.parseDouble(stry);
```

```
            g.getTransx().setData(vec.x);
            g.getTransy().setData(vec.y);


            try {

                g.getTransx().sendUpdate();
                g.getTransy().sendUpdate();
            } catch (IOException ex) {
                Logger.getLogger(Render.class.getName()).log(Level.SEVERE,
                    null, ex);
            }

        }
}
```

O método que realiza rotações no AV é o getTransform(), porém, esse é proveniente de outra classe de controle para *mouse*, a classe **MouseRotate**. Esse método utiliza um objeto do tipo Matrix4d para gerar as transformações ao AV. Dessa maneira, sempre quando houver alteração nos eixos de rotação do AV, um objeto do tipo Event é atualizado.

5.13: Método de rotação da aplicação via *mouse*

```
public class NewMouseRotate extends MouseRotate {

        public void getTransform() {

            examineGroup.getTransform(rota);

            Matrix4d mat = new Matrix4d();
            // Remember old matrix
            rota.get(mat);
            rota.setTranslation(new Vector3d(0.0, 0.0, 0.0));
            if (invert) {
                rota.mul(rota, transformX);
                rota.mul(rota, transformY);
            } else {
                rota.mul(transformX, rota);
                rota.mul(transformY, rota);
            }

            // Set old translation back
            Vector3d translation = new Vector3d(mat.m03, mat.m13, mat.m23);
```

```
            rota.setTranslation(translation);

            rota.get(mat);
            examineGroup.setTransform(rota);
            try {

                g.getEvent().enqueueEvent(mat);
            } catch (IOException ex) {
                Logger.getLogger(Render.class.getName()).log(Level.SEVERE,
                    null, ex);
            }


        }
    }
```

A atualização dos valores dos objetos distribuídos também pode ser feita por meio de interação com o teclado. A diferença entre os 2 métodos, é que o método keyPressed(), utilizado pelo teclado, utiliza objetos Glass do tipo Event. Portanto, ele envia matrizes e vetores, e não apenas valores brutos como os objetos do tipo compartilhamento, porém, o funcionamento perante o usuário é semelhante.

5.14: Método de translação da aplicação via teclado

```
public void keyPressed(KeyEvent e) {
        // TODO

        //    if (g.isMaster()) {
        int key = e.getKeyCode();


        switch (key) {

            case '1':

                trans.z = (float) (trans.z + TRANSLATE_LEFT);

                try {
                    g.getEvent().enqueueEvent(trans);
                    System.out.println("Mandei");
                } catch (IOException ex) {
                    Logger.getLogger(Glass.class.getName()).log(Level.
                        SEVERE, null, ex);
```

```
            }

            break;

        case '2':

            trans.z = (float) (trans.z - TRANSLATE_LEFT);

            try {
                g.getEvent().enqueueEvent(trans);

            } catch (IOException ex) {
                Logger.getLogger(Glass.class.getName()).log(Level.
                    SEVERE, null, ex);
            }

            break;
```

Para que a interação do usuário com o AV seja feita de maneira intuitiva, uma alternativa ao teclado e *mouse* foi implementada utilizando o Wii Remote (NINTENDO, 2011). O controle permite ao usuário realizar translações e rotações no AV de maneira intuitiva, pois, por meio de botões e acelerômetros, foi possível emular as funcionalidade do teclado e *mouse*.

A interface do controle utiliza tecnologia sem fio, portanto, sendo um controle que permite um alto grau de liberdade ao usuário do sistema. A biblioteca utilizada no desenvolvimento do sistema foi a WiiuseJ (WIIUSEJ, 2011). Ela permite que controles Wii Remote realizem conexão com computadores por meio de interface *Bluetooth*.

A classe **Robot** é utilizada para fazer o mapeamento da interação do controle para comandos de *mouse* e teclado. A Figura 24 apresenta a estrutura de conexão do Wii Remote.

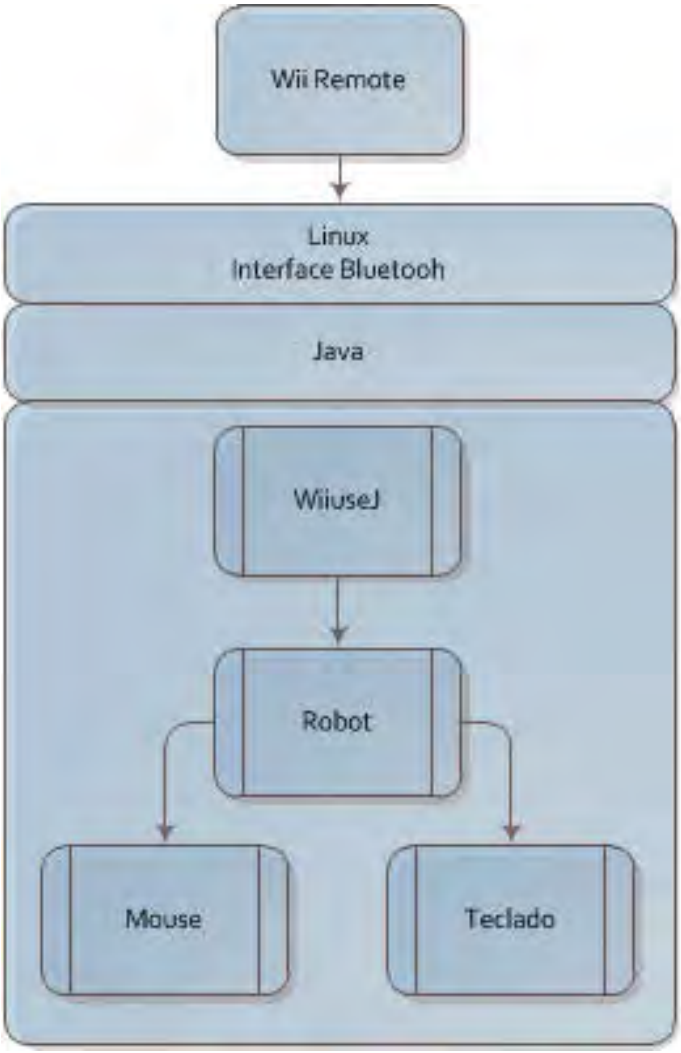


Figura 24: Estrutura de conexão do Wii Remote

5.2.6 Renderização sincronizada

Além da interação, os dados devem ser sincronizados no momento em que o AV é renderizado. A classe responsável por renderizar o AV é a **Canvas3D**. O método preRender() é chamado durante todo o período em que a aplicação é executada. Esse método foi sobrescrito para que todos os nós do AG pudessem desenhar seu objeto Canvas ao mesmo tempo. Os objetos de transformação compartilhados, depois de atualizados, são atribuídos ao objeto examineGroup, por meio do método setTransform(), realizando assim a transformação para todos os nós cliente do AG.

O método preRender() é capaz de receber interações por meio de objetos do tipo Event e Shared. Dessa maneira, todas as interações, sendo elas provenientes de *mouse*,teclado ou Wii Remote, podem ser recebidas pelo método. O método my_sync() é responsável pelo sincronismo dos nós. O trecho de Código 5.15 apresenta a sobrescrita do método preRender().

5.15: Método de renderização Java 3D

```
@Override
        public void preRender() {
            super.preRender();

            if (g.getEvent().isEmpty() == false) {
                try {
                    rota.set((Matrix4d) g.getEvent().getEvent());
                    examineGroup.setTransform(rota);
                    g.getEvent().clear();

                } catch (IOException ex) {
                    Logger.getLogger(Render.class.getName()).log(Level.
                        SEVERE, null, ex);
                } catch (EmptyException ex) {
                    Logger.getLogger(Render.class.getName()).log(Level.
                        SEVERE, null, ex);
                } catch (ClassNotFoundException ex) {
                    Logger.getLogger(Render.class.getName()).log(Level.
                        SEVERE, null, ex);
                }
            }

            try {
                g.my_sync();
            } catch (IOException ex) {

                Logger.getLogger(Render.class.getName()).log(Level.SEVERE,
                    null, ex);
            } catch (ClassNotFoundException ex) {
                Logger.getLogger(Render.class.getName()).log(Level.SEVERE,
                    null, ex);
            }



            trans.x = g.getTransx().getData();
            trans.y = g.getTransy().getData();
            trans.z = g.getTransz().getData();

            paint(getGraphics());
            viewTrans.setTranslation(trans);
            vpTransGroup.setTransform(viewTrans);
```

```
    }
```

Após a fase de renderização, é utilizada um objeto Barrier para a execução da troca de quadros. O método newSwap() foi criado para essa tarefa. Como cada nó do AG executa sua tarefa de maneira independente, diferentes quantidades de polígonos do AV podem influenciar na taxa de quadros dos nós. Dessa maneira, tem-se à necessidade de se sincronizar essa troca de quadros. O trecho de Código 5.16 apresenta o método responsável pela troca de quadros da aplicação Java 3D.

5.16: Método de *swap* Java 3D

```
public void newSwap() {
        g.getFramelock().sync();
    }
```

Por último, resumidamente, é apresentada a sequência de execução da aplicação Java 3D distribuída. O usuário executa a aplicação Java 3D, gerando assim, o grafo de cena da aplicação por meio da classe **Loader**. As configurações de luzes e opções de transformação são habilitadas (Anexo A). Em seguida, os objetos Glass são instanciados e o método de renderização, preRender(), passa a ser executado durante o decorrer da aplicação. Os métodos de interação *mouse*, teclado e Wii Remote, são executados pelo usuário quando algum estímulo ao ambiente é solicitado. Por último, o usuário finaliza a aplicação. A Figura 25 apresenta o diagrama de sequência da execução da aplicação Java 3D distribuída.

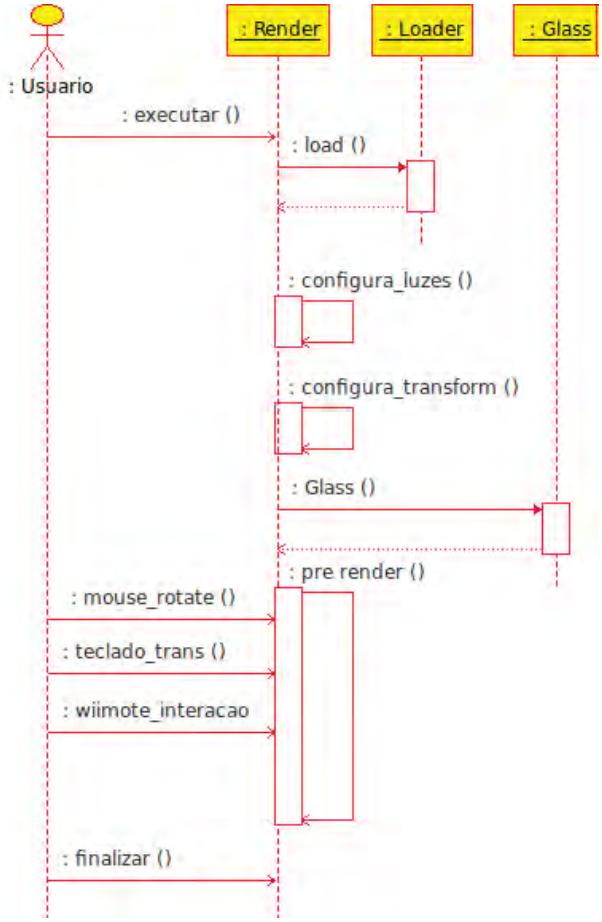


Figura 25: Diagrama de sequência - Java 3D distribuída

5.3 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada a implantação de uma estrutura de multiprojeção, o Mini CAVE. A estrutura possibilita ao usuário a sensação de imersão, interação e navegação, características de um sistema de RV. O Mini CAVE provê a capacidade de realizar projeções estereoscópicas, utilizando-se de óculos polarizados e lentes polarizadoras.

Para o desenvolvimento da aplicação Java 3D distribuída, foi necessária a descrição dos passos de desenvolvimento, por se tratar de uma aplicação que utiliza conceitos não difundidos na criação de aplicações corriqueiras. Dessa forma, foram apresentadas as definições dos objetos distribuídos necessários, como: barreiras, objetos compartilhados, eventos compartilhados e alias.

Para provêr um nível maior de interação ao usuário, Wii Remotes foram utilizados, possibilitando ao usuário a interação por meio de um dispositivo intuitivo e utilizando conexão sem fio.

6 DESCRIÇÃO SEMÂNTICA DE AMBIENTES VIRTUAIS

As informações geradas por um AV são apresentadas por meios geométricos, o que é mais intuitivo do que textos e figuras planas. Porém, somente a utilização de formas geométricas não é suficiente, de modo que, a integração e compartilhamento dos dados representados por estas possam ser apresentados de forma textual. Por isso, é utilizada a informação semântica, ou seja, o significado conceitual de cada parte que integra o AV.

Os AVs propiciam grande quantidade de informação ao usuário. Portanto, é importante que essas informações possam ser classificadas da melhor maneira. Utilizando-se informação textual em adição a informação gráfica, é possível aprimorar a visualização de AVs, melhorando a aquisição de conhecimento dos usuários.

Para o desenvolvimento deste trabalho, uma ontologia foi definida, portanto, os conceitos utilizados são apresentados.

Para que os modelos utilizados no sistema possam ter informações adicionais, e não apenas a informação visual, um módulo de descrição semântica foi desenvolvido. Este módulo é capaz de:

- Separar modelos 3D em X3D/VRML por nós de transformação;
- Gerar instâncias RDF/XML dos modelos 3D com as partes descritas semanticamente; e
- Agregar outros tipos de mídia aos modelos 3D, tais como: vídeo, áudio, texto, apresentações multimídia etc.

6.1 EDIÇÃO DE MODELOS VIRTUAIS

A edição de modelos 3D em X3D/VRML é feita utilizando a API JDom (JDOM, 2011), que permite acesso, manipulação e exportação de arquivos X3D/XML em aplicações Java. Assim, modelos X3D/VRML são separados por nós de transformação, tornando possível a geração de modelos menores com partes desse modelo. Um módulo de editoração foi desenvolvido para

que o usuário possa gerar conteúdo para o sistema de multiprojeção. O Código 6.1 apresenta um objeto X3D complexo, isto é, com todos os dentes da arcada dentária.

6.1: Documento X3D com todos os dentes da arcada

```
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/
    specifications/x3d-3.0.dtd">
<X3D profile='Immersive' version='3.0'>
<head>
</head>
   <Scene>

        <Transform DEF='Lower_Incisor'>          </Transform>
        <Transform DEF='Lower_Incisor1'>         </Transform>
        <Transform DEF='Lower_Incisor2'>         </Transform>
        <Transform DEF='Lower_Incisor3'>         </Transform>
        <Transform DEF='Lower_Molar'>            </Transform>
        <Transform DEF='Lower_Molar1'>           </Transform>
        <Transform DEF='Lower_Molar2'>           </Transform>
        <Transform DEF='Lower_Molar3'>           </Transform>
        <Transform DEF='Lower_Premolar' >        </Transform>
        <Transform DEF='Lower_Premolar1' >       </Transform>
        <Transform DEF='Lower_Premolar2' >       </Transform>
        <Transform DEF='Lower_Premolar3' >       </Transform>
        <Transform DEF='Lower_Canine' >          </Transform>
        <Transform DEF='Lower_Canine1' >         </Transform>
        <Transform DEF='Upper_Canine' >          </Transform>
        <Transform DEF='Upper_Canine1' >         </Transform>
        <Transform DEF='Upper_Incisor' >         </Transform>
        <Transform DEF='Upper_Incisor1' >        </Transform>
        <Transform DEF='Upper_Incisor2' >        </Transform>
        <Transform DEF='Upper_Incisor3' >        </Transform>
        <Transform DEF='Upper_Molar' >           </Transform>
        <Transform DEF='Upper_Molar1' >          </Transform>
        <Transform DEF='Upper_Molar2' >          </Transform>
        <Transform DEF='Upper_Molar3' >          </Transform>
        <Transform DEF='Upper_Premolar' >        </Transform>
        <Transform DEF='Upper_Premolar1' >       </Transform>
        <Transform DEF='Upper_Premolar2' >       </Transform>
        <Transform DEF='Upper_Premolar3' >       </Transform>

   </Scene>
</X3D>
```

O Código 6.2 apresenta um modelo criado utilizando o módulo de separação de modelos, onde apenas os dentes molares são gerados no novo modelo.

6.2: Documento X3D com apenas os dentes molares

```
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN" "http://www.web3d.org/
    specifications/x3d-3.0.dtd">
<X3D profile='Immersive' version='3.0'>
<head>
</head>
   <Scene>

        <Transform DEF='Lower_Molar'>          </Transform>
        <Transform DEF='Lower_Molar1'>         </Transform>
        <Transform DEF='Lower_Molar2'>         </Transform>
        <Transform DEF='Lower_Molar3'>         </Transform>

   </Scene>

</X3D>
```

A ideia de se criar modelos com partes específicas do ambiente, é devida à necessidade em se visualizar e descrever apenas partes do AV. Dessa maneira, um modelo odontológico 3D que contenha todos os dentes, pode ser representado por algumas partes separadas, refinando assim a forma de representação.

6.2 ANATOMIA DENTAL

Para a criação da ontologia utilizada neste trabalho, alguns conceitos básicos sobre odontologia devem ser apresentados. Portanto, esta seção apresenta a estrutura externa dos dentes permanentes, destacando os atributos que foram utilizados no desenvolvimento do sistema.

As funções dos dentes devem estar bem definidas. Assim sendo, é apresentada a razão da existência dos dentes, as modificações que sofrem, a utilidade de cada dente, suas principais características, dentições e grupos, nomenclaturas que designam seus elementos e porções. Suas funções serão apresentadas separadamente e em conjunto. A estrutura dos dentes é estudada pela Morfologia Dental, porém, neste trabalho, são apresentados apenas as caracterísitcas externas dos dentes.

Esta seção é baseada na obra de Figún e Garino (1989).

6.2.1 Definição

A Anatomia Dental é um ramo da biologia que estuda a organização dos dentes com parte integrante do sistema dental e aparelho mastigador. Ainda que a Anatomia pareça ser uma ciência descritiva e estática, não é dessa maneira que a estrutura do dente deve ser idealizada.

Os grupos dentários são formados por dentes que são semelhantes quanto ao material, porém, variando suas formas, volume e posição. A definição dos grupos existentes são de grande importância para este trabalho.

Figún e Garino (1989), afirmam que o conhecimento anatômico dos dentes é de extrema importância ao odontólogo, pois ao formular um diagnóstico ou estabelecer um tratamento, a estrutura dental deve ser conhecida a priori.

A combinação dos dentes no mesmo maxilar constitue o arco dental. A maxila (arcada superior) e mandíbula (arcada infeior) são portadoras dos dentes e como esqueleto das partes moles; os músculos formam as paredes bucais e são responsáveis por produzir os movimentos mandibulares; as articulações temporomandibulares produzem as excursões da mandíbula; os tecidos moles formam: lábios, bochechas, palato, palato mole, fauce, soalho da boca, língua e glândulas salivares.

Nos humanos as dentições são divididas em duas: decídua e permanente; a primeira dentição, a decídua, começa a ser formada aos 6 meses de idade, estando completa por volta dos 3 anos de idade. Nesta dentição existem 20 dentes, sendo 10 na arcada superior e 10 na arcada inferior; a segunda dentição, a permanente, começa a ser formada por volta dos 6 anos, estando completa por volta dos trezes anos de idade. Nesta dentição existem 32 dentes, sendo 16 na arcada superior e 16 na arcada inferior.

6.2.2 Funções dos dentes e do sistema dental

Cada tipo de dente, ou um grupo, tem sua função definida. As funções podem ser descritas em: mastigatória, fonética, estética e preservação.

- Mastigatória: está destinada a produzir a fragmentação dos alimentos utilizando 2 fatores fundamentais: a força apresentada pelos músculos da mastigação e os dentes;
- Fonética: tem com elemento a boca compondo o aparelho da fonação, controlando a mecânica da respiração, movimentos laríngeos, vibração das cordas vocais e acomodação das cavidades ressonadoras e dos pontos de articulação;

- Estética: os dentes também contribuem com a estética, porém, ainda tem um papel importante quanto a posição da musculatura facial; e
- Preservação: é a função exercida pelo dente para manter seu próprio posição no arco dental, evitando possíveis deslocamentos.

Os dentes, mesmo os pertencentes a um mesmo grupo, possuem diferentes formas. Dessa maneira, deve-se considerar a morfologia geral do dente para se definir sua nomenclatura. A forma de um prisma é utilizada na comparação de um dente, sendo divida em duas partes: uma de menor altura, que corresponde à coroa dental, e outro maior, à raiz. A Figura 26 apresenta as duas partes do dente.

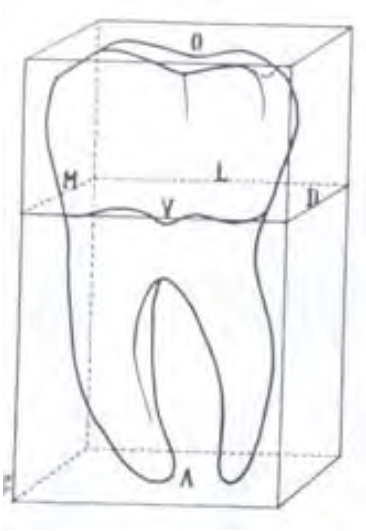


Figura 26: Partes do dente
Fonte: (FIGÚN; GARINO, 1989)

Os dentes são subdivididos em grupos. Os grupos são:

- Incisivos: é formado por um dente central e outro lateral em cada hemi-arco. Possui coroa uniforme, tem uma porção radicular única, é bordo oclusal e aresta da união das faces livres. Existem 4 incisivos superiores e 4 incisivos inferiores. São os primeiros dentes a entrar em contato com os alimentos.
- Caninos: é representado por um elemento em cada hemi-arco de cada dentição. Os dentes deste grupo são bordo oclusal, formando duas vertentes dando-lhe a forma de um "V". São unirradiculares.

- Pré-molares: só existem na dentição permanente, representados por 2 em cada hemi-arco da dentição. Possuuem face oclusal e são unirradicularres, com exceção do primeiro pré-molar superior, quando apresenta duas raízes.

- Molares: são maiores que os dentes que os precedem, sendo constituídos por 3 dentes em cada hemi-arco da dentição na dentadura permanente. Possui face oclusal e porção radicular múltipla. Quanto ao número de raízes, os molares inferiores possuem 2 raízes, enquanto os superiores posuem uma terceira raíz.

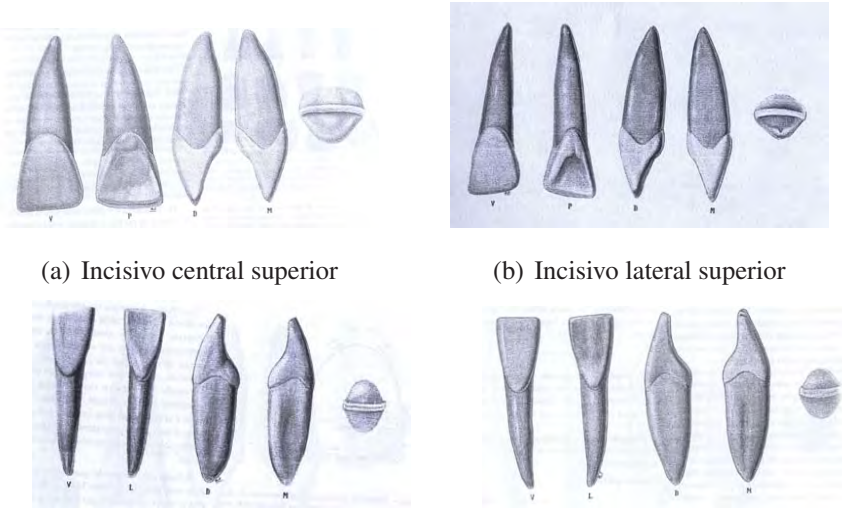
Os dentes também são representados por números. É utilizado um intervalo entre 11 e 48. A Figura 27 apresenta a estrutura da arcada permanente, a qual é utilizada neste trabalho.

Figura 27: Dentição permanente
Fonte: (PRIMAL, 2011)

6.2.3 Morfologia dos dentes permanentes

Esta seção apresenta a morfologia dos dentes permanentes de acordo com seus grupos. Podem existir diferentes morfologias para apresentação dos dentes, porém, é apresentada sua forma mais frequente.

Os dentes incisivos ocupam a parte anterior do arco, sendo os primeiros a entrarem em contato com os alimentos. Estes tem a tarefa de cortar os alimentos, podendo também, realizar a tarefa de roer. Devem ser apresentados 4 tipos de dentes: os central superiores, lateral superiores, central inferiores e lateral superiores. A Figura 28 apresenta os dentes incisivos.

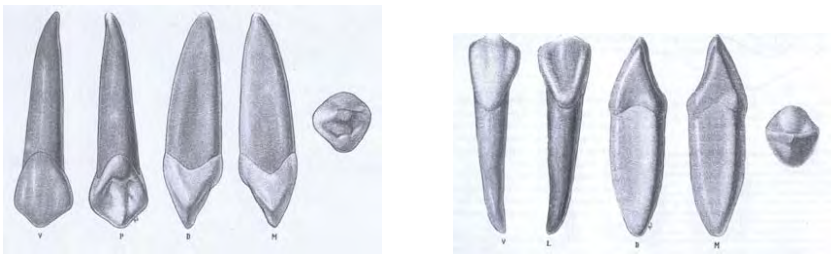


(a) Incisivo central superior (b) Incisivo lateral superior

(c) Incisivo central inferior (d) Incisivo lateral inferior

Figura 28: Dentes incisivos
Fonte: (FIGÚN; GARINO, 1989)

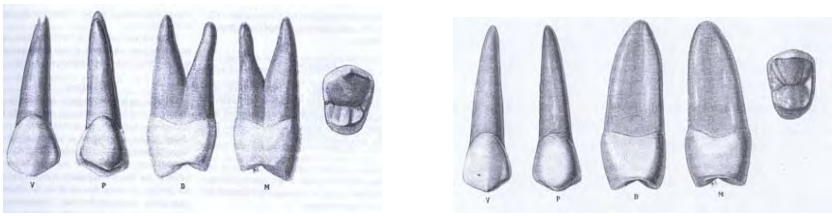
Os dentes caninos ocupam a parte do arco por detrás dos incisivos laterais, que apresentam borda oclusal com duas vertentes, determinando um vértice. Estes tem a tarefa de cortar os alimentos que necessitem de grande força para serem cortados. As pontas dos dentes caninos são afiadas e dilacerantes. A Figura 29 apresenta os dentes caninos.



(a) Canino superior (b) Canino inferior

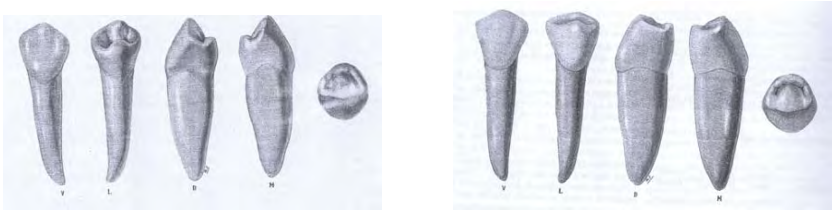
Figura 29: Dentes caninos
Fonte: (FIGÚN; GARINO, 1989)

Os dentes pré-molares são localizados atrás dos caninos. São responsáveis pela trituração dos alimentos, por meio do encontro das superfícies superior e inferior, realizado pelo movimento da mandíbula. A Figura 30 apresenta os dentes pré-molares superiores. A Figura 31 apresenta os dentes pre-molares inferiores.



(a) Primeiro pré-molar superior (b) Segundo pré-molar superior

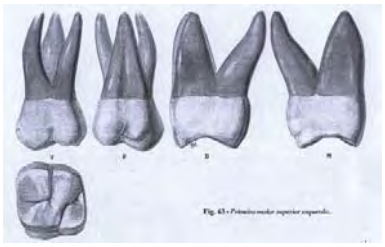
Figura 30: Dentes pré-molares superiores
Fonte: (FIGÚN; GARINO, 1989)



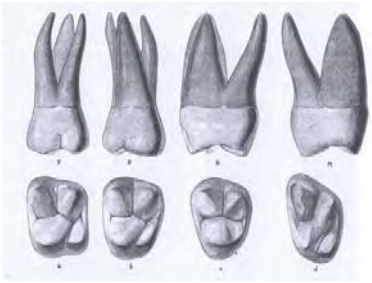
(a) Primeiro pré-molar inferior (b) Segundo pré-molar inferior

Figura 31: Dentes pré-molares inferiores
Fonte: (FIGÚN; GARINO, 1989)

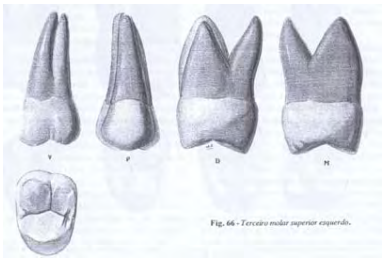
Os dentes molares são os maiores do arco dentário. Os inferiores possuem 2 raízes, enquanto os superiores possuem 3, sendo 1 língual e 2 vestibulares. A função dos molares é semelhante a dos pré-molares, porém, a eficiência mastigatória é maior, devido a superfície da coroa possuir uma complexidade maior. A Figura 32 apresenta os dentes molares superiores. A Figura 33 apresenta os dentes molares inferiores.



(a) Primeiro molar superior



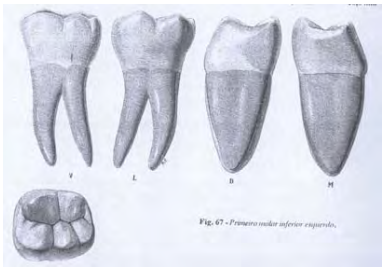
(b) Segundo molar superior



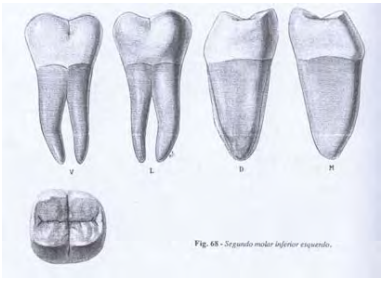
(c) Terceiro molar superior

Figura 32: Dentes molares superiores

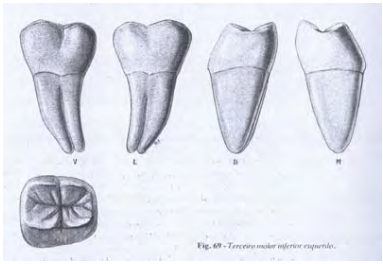
Fonte: (FIGÚN; GARINO, 1989)



(a) Primeiro molar inferior



(b) Segundo molar inferior



(c) Terceiro molar inferior

Figura 33: Dentes molares inferiores

Fonte: (FIGÚN; GARINO, 1989)

6.3 DESCRIÇÃO DOS MODELOS VIRTUAIS

A fim de se combinar informação visual e textual, foi desenvolvido um módulo que possibilita a descrição dos modelos virtuais criados pelo sistema. Para realização das descrições semânticas, uma ontologia de apoio foi criada para auxiliar este processo.

A Figura 34 apresenta as classes e atributos da ontologia.

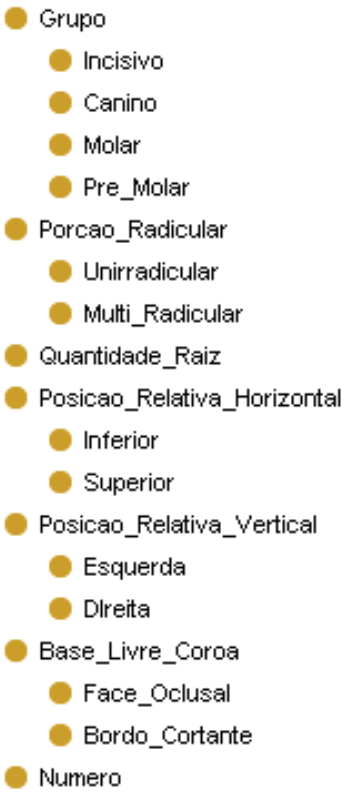


Figura 34: Classes da ontologia de apoio à descrição de modelos virtuais

A definição das classes e atributos foi orientada por profissionais da área, para que a descrição fosse feita de maneira eficiente. Para a implementação do arquivo RDF Schema foi utilizada a aplicação Protege. O RDF Schema é responsável por prover uma estrutura de classe para a instanciação de objetos RDF. O arquivo RDF Schema é apresentado no trecho de Código 6.3.

6.3: RDF Schema

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE rdf:RDF [
        <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
        <!ENTITY a 'http://protege.stanford.edu/system#'>
        <!ENTITY kb 'http://protege.stanford.edu/kb#'>
        <!ENTITY rdfs 'http://www.w3.org/2000/01/rdf-schema#'>
]>
```

```
<rdf:RDF xmlns:rdf="&rdf;"
         xmlns:a="&a;"
         xmlns:kb="&kb;"
         xmlns:rdfs="&rdfs;">
<rdfs:Class rdf:about="&kb;Arcada"
         rdfs:label="Arcada">
        <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdf:Property rdf:about="&kb;Base_Livre_Coroa"
         a:maxCardinality="1"
         rdfs:label="Base_Livre_Coroa">
        <rdfs:domain rdf:resource="&kb;Arcada"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&kb;Grupo"
         a:maxCardinality="1"
         rdfs:label="Grupo">
        <rdfs:domain rdf:resource="&kb;Arcada"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&kb;Material"
         rdfs:label="Material">
        <rdfs:domain rdf:resource="&kb;Arcada"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&kb;Numero"
         a:maxCardinality="1"
         rdfs:label="Numero">
        <rdfs:domain rdf:resource="&kb;Arcada"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&kb;Porcao_Radicular"
         a:maxCardinality="1"
         rdfs:label="Porcao_Radicular">
        <rdfs:domain rdf:resource="&kb;Arcada"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&kb;Posicao_Relativa_Horizontal"
         a:maxCardinality="1"
         rdfs:label="Posicao_Relativa_Horizontal">
        <rdfs:domain rdf:resource="&kb;Arcada"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
```

```
<rdf:Property rdf:about="&kb;Posicao_Relativa_Vertical"
          a:maxCardinality="1"
          rdfs:label="Posicao_Relativa_Vertical">
         <rdfs:domain rdf:resource="&kb;Arcada"/>
         <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&kb;Quantidade_Raiz"
          a:maxCardinality="1"
          rdfs:label="Quantidade_Raiz">
         <rdfs:domain rdf:resource="&kb;Arcada"/>
         <rdfs:range rdf:resource="&rdfs;Literal"/>
</rdf:Property>
</rdf:RDF>
```

A criação de instâncias dos modelos 3D utilizados permite que informações sejam atribuídas aos modelos. Por isso, sempre que necessário, o usuário pode agregar alguma informação pertinente ao modelo visualizado.

A descrição é feita utilizando-se os atributos especificados pela ontologia de apoio. A estrutura do documento RDF utiliza o *namespace* **ad**, que representa a ontologia criada para descrição de estruturas odontológicas.

O trecho de Código 6.4 apresenta um exemplo de descrição realizada em um documento RDF, onde o modelo e o nó de transformação tem atributos preenchidos pelo usuário, para que esses possam ser reutilizados posteriormente por aplicações *desktop* ou *web*.

6.4: Descrição RDF

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ad="http://wwwp.fc.unesp.br/lstr/#">
  <rdf:Description rdf:about="http://wwwp.fc.unesp.br/lstr/Canino_Inferior.
      x3d/Lower_Canine">
    <ad:Numero>43</ad:Numero>
    <ad:Grupo>Caninos</ad:Grupo>
    <ad:Porcao_Radicular>Unirradicular</ad:Porcao_Radicular>
    <ad:Quantidade_Raiz>1</ad:Quantidade_Raiz>
    <ad:Base_Livre_Coroa>Bordo Cortante</ad:Base_Livre_Coroa>
    <ad:Posicao_Relativa_Horizontal>Inferior</ad:
        Posicao_Relativa_Horizontal>
    <ad:Posicao_Relativa_Vertical>Direito</ad:Posicao_Relativa_Vertical>
  </rdf:Description>
  <rdf:Description rdf:about="http://wwwp.fc.unesp.br/lstr/Lower_Premolar.
```

```
      x3d/Lower_Premolar1">
    <ad:Posicao_Relativa_Vertical>Direito</ad:Posicao_Relativa_Vertical>
    <ad:Posicao_Relativa_Horizontal>Inferior</ad:
        Posicao_Relativa_Horizontal>
    <ad:Base_Livre_Coroa>Face  Oclusal</ad:Base_Livre_Coroa>
    <ad:Quantidade_Raiz>2</ad:Quantidade_Raiz>
    <ad:Porcao_Radicular>Multirradicular</ad:Porcao_Radicular>
    <ad:Grupo>Pre-molares</ad:Grupo>
    <ad:Numero>32</ad:Numero>
  </rdf:Description>
</rdf:RDF>
```

O documento RDF representado pelo trecho de Código 6.4 possui um atributo principal, o qual indica a localização do AV que está sendo descrito, o *<rdf:Descrition>*. Primeiro, é criada uma *tag <rdf:Descrition>*, a qual representa o início de uma nova descrição.

O atributo *rdf:About* recebe o valor do local em que o AV está localizado, ou seja, a URI. Desse modo, todas as *tags* que estejam preenchidas até a *tag </rdf:Description>*, fazem parte da descrição do AV referente a URI.

A estrutura do documento RDF é alterada em decorrência da realização de novas descrições, sendo alocadas sempre no final do documento. O trecho de Código 6.4 apresenta duas descrições.

6.4 AGREGAÇÃO DE CONTEÚDO MULTIMÍDIA

Este módulo também permite que o usuário agregue outros tipos de mídia que possam auxiliar no entendimento da visualização. Juntamente com os atributos propostos para a ontologia de apoio, atributos providos pelo *Dublin Core* (DUBLIN CORE, 2011) também são utilizados. Deste modo, arquivos com vários formatos como: áudio, vídeo, apresentações multimídias e textos, podem ser combinados com os modelos virtuais.

O *namespace* **dc** representa o conjunto de atributos *Dublin Core*, utilizados para descrever documentos multimídia. Os atríbutos disponíveis ao usuário no módulo de agregação multimídia são 15, porém, somente alguns atributos são recomendados:

- o **title**, que fornece o título do documento agregado;
- o **description**, que permite que o usuário forneça breves descrições do documento agregado;

- o **source**, que define a fonte desde documento; e
- o **type**, que indica o tipo de mídia agregada.

O trecho de Código 6.5 apresenta um exemplo de um documento RDF com a agregação de conteúdo multimídia juntamente com a descrição semântica do modelo virtual.

6.5: Descrição RDF com agregação de conteúdo multimídia

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ad="http://wwwp.fc.unesp.br/lstr/#"
    xmlns:dc="http://purl.oclc.org/dc/#">
  <rdf:Description rdf:about="http://wwwp.fc.unesp.br/lstr/Canino_Inferior.
      x3d/Lower_Canine">
    <ad:Numero>43</ad:Numero>
    <ad:Grupo>Caninos</ad:Grupo>
    <ad:Porcao_Radicular>Unirradicular</ad:Porcao_Radicular>
    <ad:Quantidade_Raiz>1</ad:Quantidade_Raiz>
    <ad:Base_Livre_Coroa>Bordo Cortante</ad:Base_Livre_Coroa>
    <ad:Posicao_Relativa_Horizontal>Inferior</ad:
        Posicao_Relativa_Horizontal>
    <ad:Posicao_Relativa_Vertical>Direito</ad:Posicao_Relativa_Vertical>
    <dc:title>Arcada</dc:title>
    <dc:description>Video sobre Arcadas Dentarias</dc:description>
    <dc:source>http://wwwp.fc.unesp.br/lstr/Arcada.avi</dc:source>
    <dc:type>Video</dc:type>
  </rdf:Description>
</rdf:RDF>
```

Os arquivos multimídia também podem ser visualizados no Mini CAVE, intercalando com a visualização dos modelos virtuais. Este tipo de visualização é apresentado em exemplos no Capítulo 7.

6.5 CONSIDERAÇÕES FINAIS

As descrições realizadas em modelos virtuais possibilita a classificação e identificação de partes do modelo, característica que auxilia na criação de novos modelos que necessitem apenas de partes de vários modelos.

Com o intuito de apresentar os conceitos utilizados na ontologia desenvolvida, este capítulo apresentou um levantamento sobre as características externas da dentição permanente.

A agregação de conteúdo multimídia permite que documentos que possuam informação pertinente ao modelo virtual sejam vinculados a estes. Dessa forma, procura-se melhorar a qualidade de informação adquirida na visualização dos modelos virtuais, pois apenas a utilização das formas geométricas pode não satisfazer a necessidade.

Foi apresentado neste capítulo o módulo de descrição semântica desenvolvido neste trabalho, apresentando exemplos de instâncias RDF contendo descrições e agregações de conteúdo que são utilizados no Capítulo 7, onde a aplicação é descrita.

7 APLICAÇÃO DO USUÁRIO

A aplicação ao usuário permite que sejam geradas as visualizações no sistema de multiprojeção de maneira facilitada. Assim, o usuário terá uma interface simples que possibilita total controle das projeções desejadas.

A aplicação foi desenvolvida para ser executada em computador que não esteja, necessariamente, conectado diretamente ao AG. A aplicação foi desenvolvida utilizando a linguagem Java, podendo assim, ser executada em qualquer computador que possua uma máquina virtual Java, tornando-se independente da arquitetura utilizada no AG. É composta de botões, menus e guias de visualização.

O intuito foi disponibilizar uma forma fácil de se realizar as descrições semânticas e agregações, e a interação com as funcionalidades do AG.

Todo o conteúdo gerado por esta aplicação, deve ser sincronizado com um servidor *web*. Para isso, deve haver uma sincronização dos dados.

Os servidores devem possuir conexão entre si. A Figura 35 apresenta a estrutura de comunicação entre a Aplicação do Usuário com o servidores do AG e *web*.

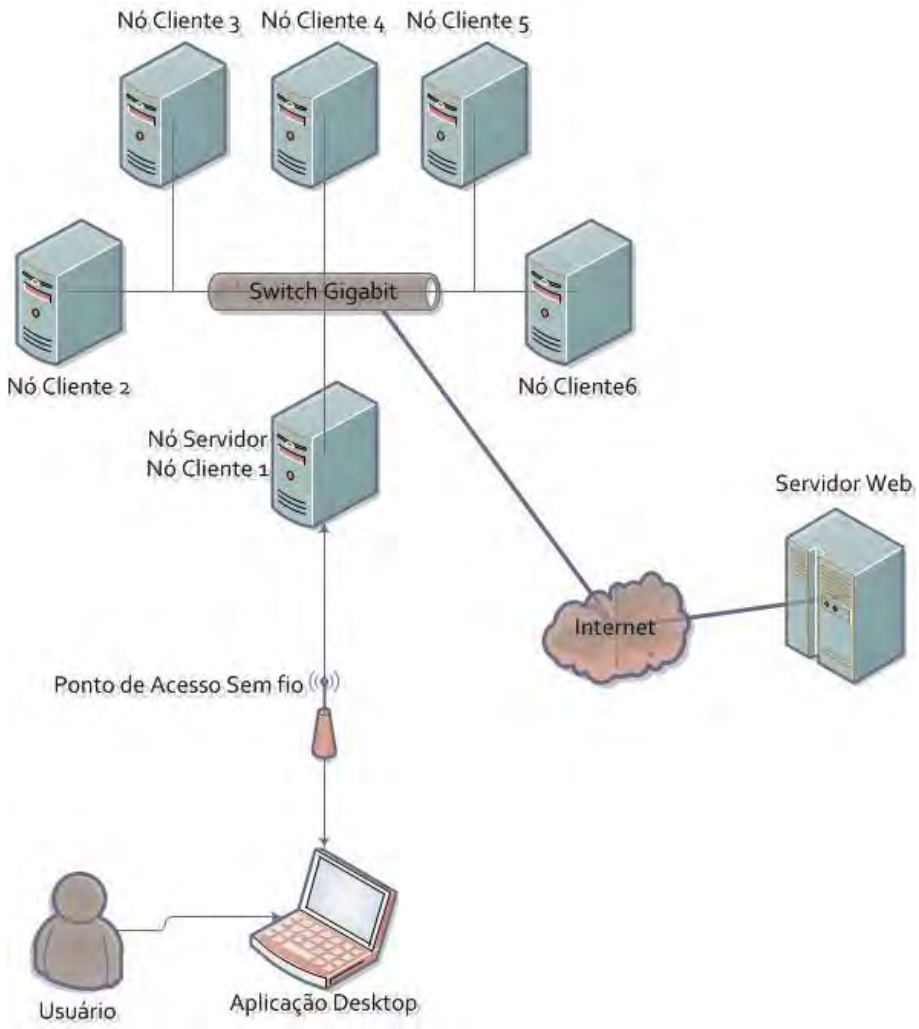


Figura 35: Estrutura de comunicação entre o aglomerado e o usuário

Na estrutura proposta, o conteúdo agregado deve estar disponibilizado localmente no servidor do AG, sendo posteriormente, juntamente com os modelos e arquivo RDF, adicionados a um servidor *web* para que o conteúdo possa ser acessado também por outras aplicações, por exemplo, um *browser* executando diretamente em um navegador *web* que permita a visualização dos modelos virtuais e suas descrições semânticas.

7.1 FUNCIONALIDADES DA APLICAÇÃO

A Figura 36 apresenta o fluxo de execução da aplicação utilizando as funcionalidades descritas. Dentre as funcionalidades da aplicação, pode-se citar:

- Editoração de Modelos Virtuais: esta funcionalidade possibilita ao usuário a utilização do módulo de editoração de modelos virtuais apresentado na seção anterior, permitindo ao usuário carregar, criar e editar os modelos virtuais 3D;

- Criação de Instâncias RDF: esta funcionalidade permite ao usuário a realização de descrições semântica. Por meio de menus de seleção é possível preencher os campos relacionados aos atributos listados na ontologia proposta na seção anterior, e agregar conteúdo multimídia; e

- Controle do Sistema do Mini CAVE: esta funcionalidade permite ao usuário fazer chamadas ssh diretamente ao servidor do AG. Desta maneira, a execução dos *scripts* é invisível ao usuário, tornando-se uma simples tarefa realizada por uma interface gráfica provida de botões e caixas de seleção. Os *scripts* utilizados para o controle do sistema de multiprojeção são apresentados no Apêndice C.

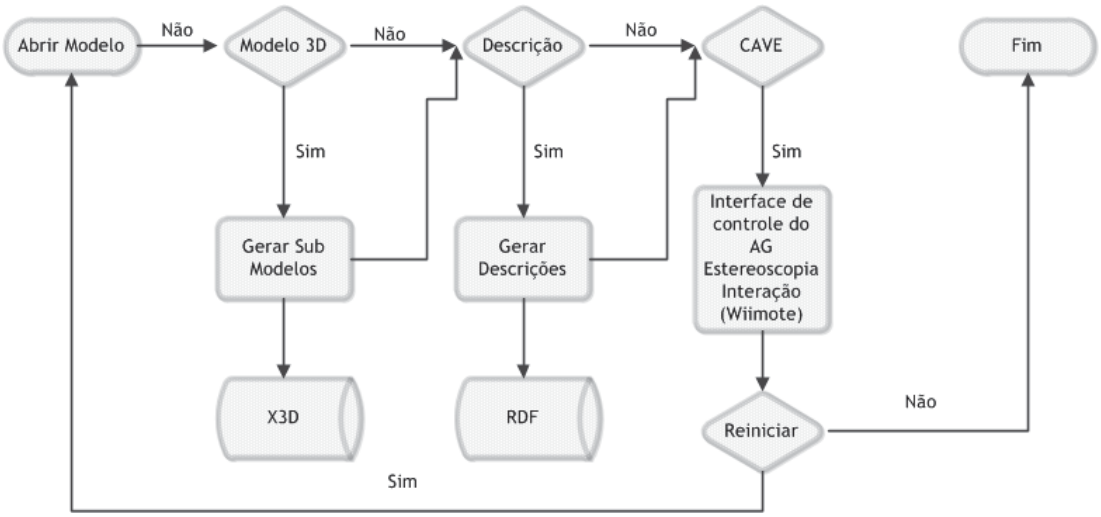


Figura 36: Fluxo de execução do sistema

7.2 INTERFACE GRÁFICA

A interface inicial é apresentada na Figura 37, separada em:

- Modelos: responsável por apresentar os componentes do modelo virtual. Os componentes do modelo virtual podem ser selecionados, de modo que, novos modelos virtuais são criados selecionando esses componentes;

- Visualização: apresenta os modelos virtuais carregados na aba Modelos, de modo que o usuário consiga visualizar o modelo que será criado ou editado;

- Descrição do Modelo: permite que atributos relacionados a modelos odontológicos possam ser relacionados aos modelos virtuais utilizados no sistema. Documentos multimídias também são agregados aos modelos; e

- Controle do Mini CAVE: possui controles que executam o módulo do Mini CAVE. Também possui um campo de nomeação dos modelos virtuais criados.

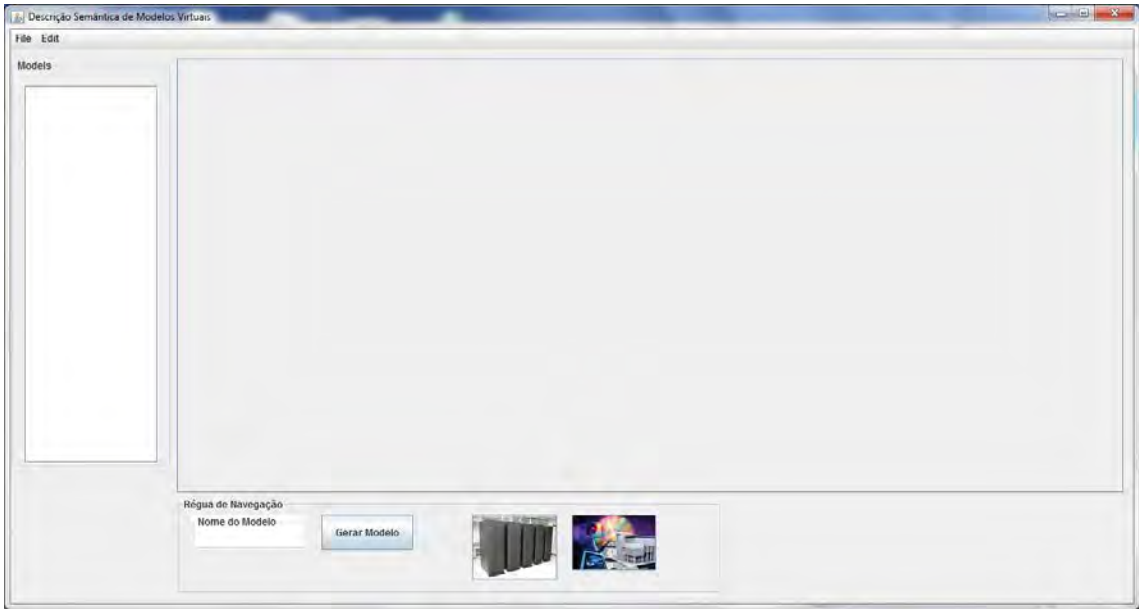


Figura 37: Interface inicial da aplicação do usuário

7.3 A UTILIZAÇÃO DO SISTEMA

Utilizando a interface inicial do sistema, depois de selecionado o modelo virtual, este pode ser utilizado para gerar novos modelos que podem ser descritos semanticamente. A Figura 38 apresenta a interface inicial do sistema com um modelo virtual sendo visualizado.

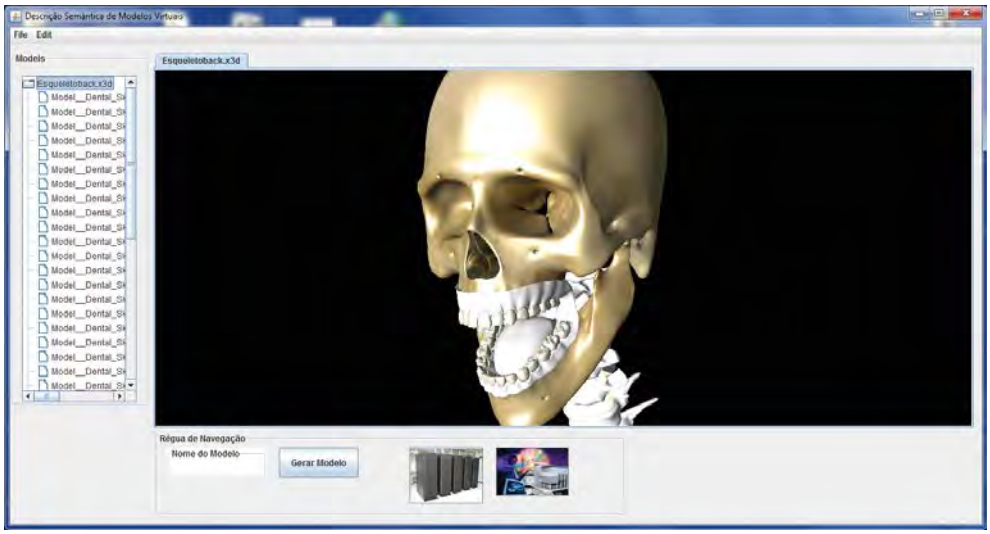


Figura 38: Modelo virtual visualizado

A Figura 39 apresenta a seleção dos componentes de um AV requeridos para criação de um

novo modelo utilizando apenas os dentes incisivos inferiores.

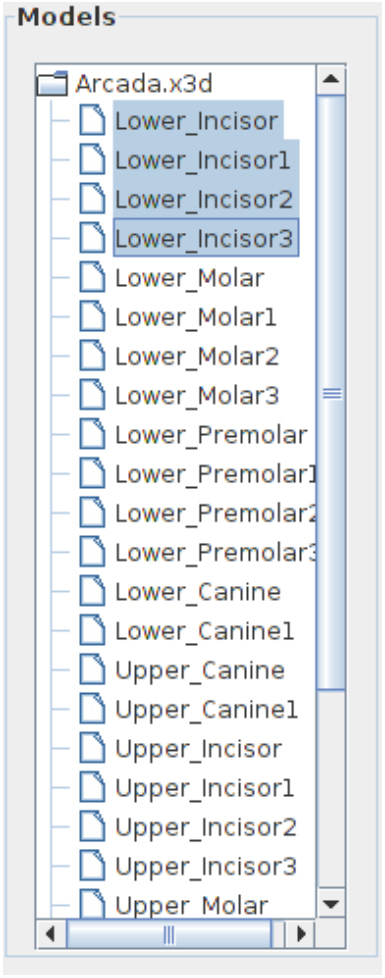


Figura 39: Seleção de dentes para a criação de um novo modelo virtual

Depois de criar um novo modelo virtual, este pode ser visualizado pela aplicação. A Figura 40 apresenta a visualização e geração da descrição de um modelo virtual contendo apenas os incisivos inferiores.

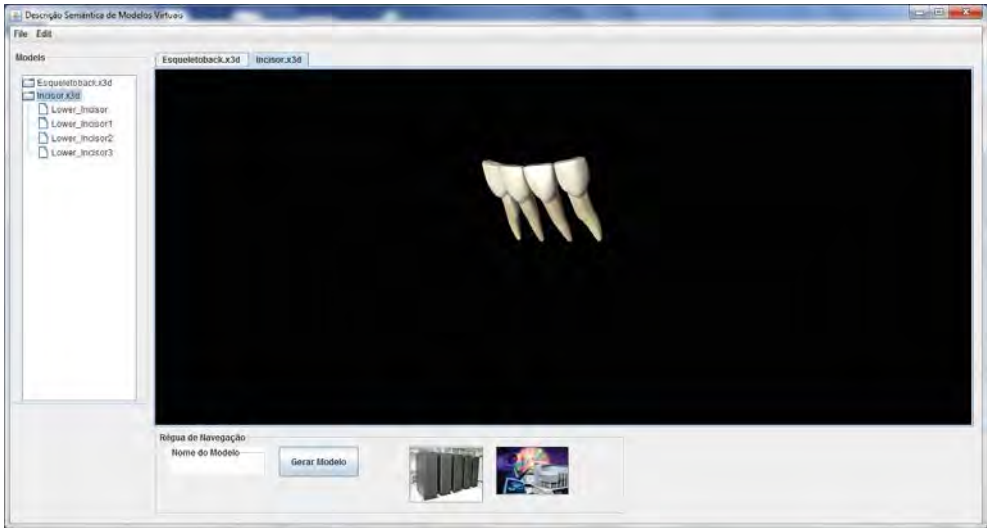


Figura 40: Modelo virtual contendo somente os dentes incisivos inferiores

A Figura 41 apresenta o módulo de descrição e agregação de conteúdo multimídia. Após a seleção dos atributos, o botão **Salvar** é pressionado, gerando assim uma nova descrição no documento RDF, vinculando o modelo virtual aos atributos. A Figura 41 apresenta a descrição de um modelo virtual do dente número 42.

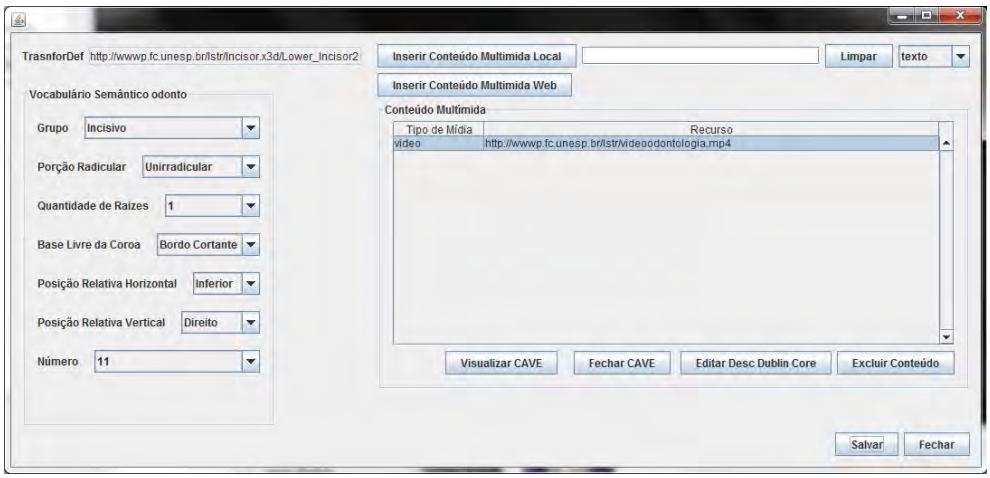


Figura 41: Descrição do modelo virtual e agregação de conteúdo multimídia

Juntamente com a descrição, pode-se também, agregar conteúdo multimídia, o qual pode ser: vídeo, texto, som, *web* e outros. A Figura 42 apresenta os atributos utilizados na descrição do conteúdo multimídia agregado.

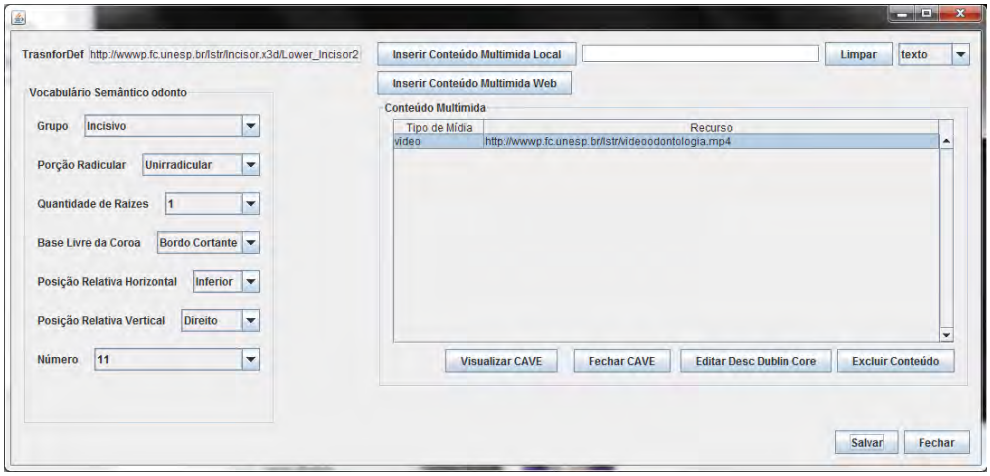


Figura 42: Atributos *Dublin Core* utilizados na descrição do conteúdo multimídia agregado ao modelo virtual

7.4 ACESSO AO MINI CAVE

O controle do sistema de multiprojeção é todo realizado por meio de interfaces visuais, ocultando ao usuário todos os comandos que deveriam ser executadas em terminais. A interface de controle é ativada por meio do botão Módulo CAVE. Esta interface permite ao usuário realizar controles como:

- Escolher a quantidade de telas que será utilizada na projeção, sendo possível disponibilizar duas telas para multiprojeção do ambiente virtual e uma para documentos multimídias;
- Informar se a visualização será, ou não, estereoscópica;
- Visualizar, em modo texto, o que acontece no AG durante a execução da visualização;
- Desligar e ligar os projetores; e
- Desligar o AG.

A Figura 43 apresenta a interface de controle do Mini CAVE.

Figura 43: Interface de controle do Mini CAVE.

Quando um modelo virtual é selecionado para ser visualizado no Mini CAVE, deve-se escolher se a projeção será estereoscópica ou não, e a quantidade de telas que serão utilizadas. Quando selecionada a quantindade de telas a serem utilizadas, os projetores que não forem utilizados devem ser desligados. A Figura 44 apresenta a interface de controle dos projetores.

Figura 44: Interface de controle dos projetores

A Figura 45 apresenta uma projeção onde são utilizadas duas telas para a visualização do modelo virtual e uma para a visualização de um vídeo agregado ao modelo visualizado. O modelo virtual apresentado é composto por dentição permanente e componentes do crânio.

Figura 45: Visualização no Mini CAVE - modelo virtual do crânio e documento multimídia

A Figura 46 apresenta a projeção de um modelo virtual composto pela dentinção permanente, o qual utiliza as 3 telas de projeção para a visualização.

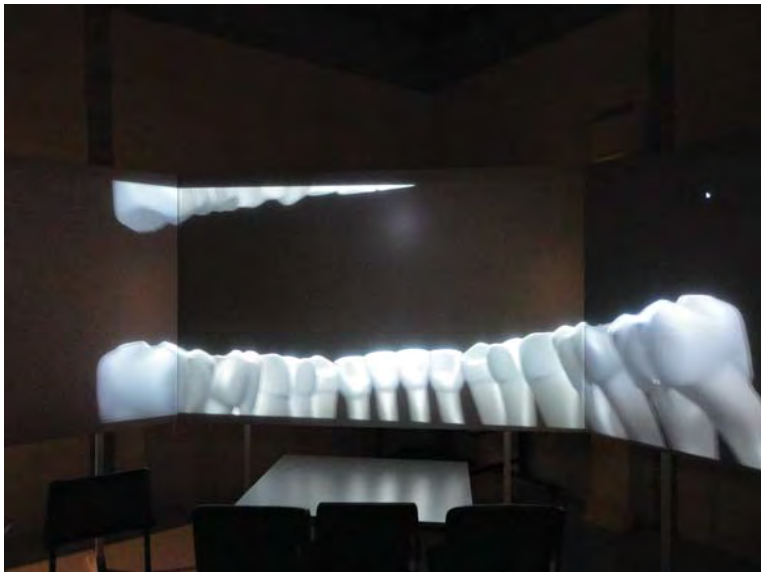


Figura 46: Visualização no Mini CAVE - modelo virtual da dentição permanente

7.5 PRÉ AVALIAÇÃO DO SISTEMA

Com intuito de avaliar o sistema desenvolvido, este foi apresentado a 8 profissionais da área odontológica, todos pertencentes a Faculdade de Odontologia de Bauru, Universidade de São Paulo. A formação dos profissionais foi dividida da seguinte maneira: 3 professores da anatomia, 1 professor da odontopediatria, 2 professores de saúde coletiva e 2 doutorandos de odontologia.

O sistema foi apresentado aos profissionais, com todas as funcionalidades descritas neste capítulo.

Com relação a avaliação da interação do AV utilizando o Wii Remote, foi solicitado que cada profissional utilizasse o controle durante 10 minutos.

A cada profissional foi entregue um questionário visando avaliar:

- Experiência prévia com aplicações de RV aplicadas a educação;
- A utilização algum tipo de documento multimídia em suas aulas;
- Opinião sobre o nível de informação gerado pelo módulo de descrição semântica;
- Opinião sobre o nível de informação gerado pelo módulo de agregação de documentos multimídia;
- Habitualidade com o controle Wii Remote e o grau de dificuldade encontrado na sua utilização;
- Utilização da ferramenta no auxílio ao ensino de estruturas odontológicas; e
- Viabilidade da implantação de um Mini CAVE em uma Instituição de ensino;

7.5.1 Resultados

Dos 8 profissionais participantes na avaliação do sistema, 3 relataram experiência prévia com aplicações de RV aplicadas ao ensino. Com relação ao uso de recursos multimídia em suas aulas, todos já utilizavam.

A utilização do Wii Remote como dispositivo de interface foi bem assimilada pelos profissionais. A questão 5, relacionada ao Wii Remote, permitia que fossem utilizados pesos diferentes para demonstrar o grau de dificuldade com a interação efetuada. Foram utilizados os valores 1, 2 e 3. Sendo 1 relacionado ao maior e 3 ao menor nível de dificuldade.

As possíveis transformações do AV são: translação, escala e rotação. Desta forma, cada profissional pode definir diferentes níveis de dificuldade para cada transformação. Em geral, todos os profissionais tiveram dificuldade com a transformação de rotação, no entanto, todos conseguiram interagir com o AV.

Os resultados obtidos por meio dos questionários foram compilados e apresentados na Tabela 1.

Tabela 1: Resultado do questionário de avaliação do sistema

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Profissional 1	S	S	CP	CP	N	AD - T2, E1, R2	CP	TV
Profissional 2	S	S	CUP	CP	N	PD - R3	CP	TV
Profissional 3	N	S	CP	CP	N	AD - T3, E2, R1	CP	TV
Profissional 4	N	S	CP	CP	N	AD - T2, E3, R2	CP	PV
Profissional 5	N	S	CP	CP	N	AD - T1, E3, R1	CP	TV
Profissional 6	N	S	CP	CP	N	PD - T2, R2	CP	VD
Profissional 7	N	S	CP	CP	N	AD - T1, E3, R1	CP	TV
Profissional 8	S	S	CP	CP	N	PD - T1, E3, R2	CP	VD

A legenda dos dados apresentados na Tabela 1 :

- **S** - Sim;
- **N** - Não;
- **CP** - Concordo plenamente;
- **CUP** - Concordo um pouco;
- **AD** - Alguma dificuldade - T (Translação) / E (Escala) / R (Rotação);
- **PD** - Pouca dificuldade;
- **TV** - Totalmente viável;
- **PV** - Parcialmente viável; e
- **VD** - Viável com algumas dificuldades.

Q1 Você já teve alguma experiência prévia com aplicações de Realidade Virtual para fins educacionais?

Q2 Você utiliza documentos multimídias em suas aulas?

Q3 Você concorda que o módulo do sistema responsável pelas descrições dos modelos virtuais pode transmitir um nível maior de informação aos alunos?

Q4 Você concorda que o módulo do sistema responsável pelas agregações multimídias com os modelos virtuais pode gerar um nível maior de informação aos alunos?

Q5 Você está habituado a utilizar o controle WiiMote?

Q6 Você encontrou dificuldades de assimilação com o uso do controle Wiimote no sistema de visualização?

Q7 Você concorda que essa aplicação possa ser utilizada em auxílio ao ensino de estruturas odontológicas?

Q8 Você acredita que a implantação da Mini Cave seria viável a uma Instituição de Ensino? Qual a maior dificuldade que você acha que encontraria para implantar a Mini Cave em uma instituição de ensino?

A questão 9 foi elaborada de maneira dissertativa, para que o profissional pudesse expressar sua opinião geral sobre o sistema.

Q9 Qual a sua opinião geral sobre o sistema apresentado?

7.5.2 Opiniões sobre o sistema

Dentre algumas situações que poderiam dificultar a implantação de um Mini CAVE, as citadas mais vezes foram: custo, pessoal treinado e espaço físico. Entretanto, todos os profissionais acreditam que o sistema será de grande auxílio ao ensino nas Instituições que possuam espaço e recursos disponíveis a implantação de um Mini CAVE, ressaltando que, a estrutura implantada neste trabalho é de baixo custo.

Ao final do questionário foi solicitada a opinião dos profissionais quanto a opinião geral sobre o sistema apresentado.

Todos os profissionais questionados acreditam que o sistema realmente possa ser utilizado como ferramenta no auxílio do ensino de estruturas odontológicas, destacando-se o uso do Mini

CAVE, que permitiu ao profissional apresentar o conteúdo de suas aulas de maneira diferenciada, combinando modelos virtuais com conteúdo multimídia.

Porém, todos acreditam que a evolução e implantação para fins educacionais, depende da modelagem de modelos virtuais mais próximo dos modelos reais visualizados em laboratórios de anatomia.

7.6 CONSIDERAÇÕES FINAIS

A interface com usuário é de extrema importância para que as funcionalidades do sistema possam ser utilizadas de maneira ideal. Assim sendo, este capítulo apresentou a estrutura e desenvolvimento da interface de aplicação ao usuário.

Foram apresentados exemplos de uso da aplicação, descrevendo a execução das tarefas de forma detalhada. Também foram apresentados exemplos de modelos virtuais visualizados no Mini CAVE.

Com intuito de validar o sistema, ao final deste capítulo foram apresentados os resultados obtidos por meio de um questionário. Este foi aplicado a profissionais da área odontológica, conseguindo assim, obter as opiniões de pessoas diretamente envolvidas no ensino de estruturas odontológicas.

8 CONCLUSÕES

As técnicas de VC e a biblioteca Glass apresentaram grande potencialidade no emprego em sistemas de RV para a odontologia. A multiprojeção feita por AGs, que auxiliou a proporcionar nível maior de imersão ao usuário, na opinião de profissionais da área odontológica, também pode auxiliar no aprendizado de conceitos relacionados a odontologia. Portanto, um sistema de RV utilizando técnicas de VC e conceitos de *Web* Semântica, mostra que a combinação destas áreas possibilitou a criação de uma aplicação robusta, com fins diversificados no meio odontológico, porém, sendo uma aplicação de baixo custo.

A biblioteca escolhida para o desenvolvimento do aglomerado utilizado neste trabalho foi a Glass. A solução adotada diferencia-se de outras soluções por utilizar a Java 3D para renderização de aplicações gráficas no AG. A renderização feita pelo Java 3D permite ao desenvolvedor criar aplicações utilizando uma linguagem de alto nível, facilitando assim, o desenvolvimento e a portabilidade de novas aplicações que visem utilizar AG. A utilização da Java 3D no desenvolvimento de aplicações de RV, possibilitou a utilização de uma linguagem orientada a objetos com alto poder de produtividade que, sendo utilizada em AG, torna-se um dos primeiros trabalhos presentes na literatura.

A utilização do controle não convencional mostrou-se intuitiva com relação à interação dos usuários com o Sistema de RV. Apresentando o trabalho a especialistas da área odontológica, a interface do controle foi facilmente assimilada.

O conhecimento da anatomia dental é de extrema importância a estudantes de cursos de odontologia, pois, todo seu aprendizado é baseado nessas estruturas. Assim sendo, a ferramenta desenvolvida neste projeto mostrou-se de grande valia para educadores, pois esta possibilita a criação de conteúdo rico em informação e, ainda, podendo ser visualizado em um Mini CAVE de maneira a estimular o aprendizado dos alunos.

A edição de modelos virtuais permitiu a educadores e alunos o desenvolvimento rápido e fácil de conteúdo para o sistema de multiprojeção desenvolvido. Assim, o usuário é capaz de gerar seu próprio conteúdo utilizando modelos virtuais existentes. Em conjunto com os

modelos virtuais, o conteúdo multimídia auxilia o entendimento das informações apresentadas nas visualizações.

O controle do AG por meio de uma aplicação gráfica possibilitou a fácil usabilidade do sistema, pois tornou-se desnecessário o conhecimento prévio da estrutura do AG, bastando ao usuário escolher os modelos virtuais e conteúdo multimídia que queira visualizar.

Deve ficar claro que o sistema de RV proposto não é limitado apenas a modelos odontológicos, podendo ser estendido a qualquer problema onde as informações possam ser representadas por modelos 3D.

8.1 TRABALHOS FUTUROS

Todos os documentos, modelos virtuais e RDF, são mantidos em um servidor *web*, portanto, como trabalho futuro é proposto um sistema *web* que permita a visualização dos modelos virtuais descritos semanticamente em um navegador *web*. Deste modo, os alunos também poderão utilizar o material gerado pelo educador, usufruindo das formas de interação e visualização, porém, não tendo a necessidade de um sistema de multiprojeção, bastando apenas um computador pessoal com acesso à internet.

A utilização da ferramenta para um nível maior de detalhes de estruturas odontológicas depende somente da criação de novos modelos. Desta maneira, com modelos mais detalhados, será possível criar novos atributos a ontologia proposta, de modo que, por exemplo, partes internas dos dentes possam ser descritas.

Há também o intuito de se desenvolver formas de os alunos visualizarem o contéudo semântico individualmente. A proposta consiste em utilizar dispositivos móveis, como celulares e tablets, para combinar o ambiente virtual, renderizado na Mini CAVE, com as informações semânticas.

8.2 TRABALHOS PUBLICADOS

8.2.1 Artigos publicados

- DIAS, D. R. C. ; OLIVEIRA, L. R. ; SCARTON, T. B. ; VIEIRA, A. M. ; BREGA, J. R. F. ; SEMENTILLE, A. C. ; GUIMARAES, M. P. ; LAURIS, J. R. P. . Sistema de Realidade Virtual para Estruturas Dentárias. In: 6º Workshop de Realidade Virtual e Aumentada - WRVA, Santos. Anais do WRVA 2009, 2009.

- GARCIA, L. M. L. S. ; DIAS, D. R. C. ; BREGA, J. R. F. . Descrição Semântica de Componentes em Ambiente Virtual 3D. In: 6º Workshop de Realidade Virtual e Aumentada - WRVA, Santos. Anais do WRVA 2009, 2009.

- DIAS, D. R. C. ; BREGA, J. R. F. ; RODELLO, I. A. ; LAURIS, J. R. P. . Sistema de Multiprojeção para o Auxílio ao Ensino de Odontologia. In: XXX Congresso da Sociedade Brasileira de Computação - CSBC 2010, 2010, Belo Horizonte. Anais do XXX Congresso da Sociedade Brasileira de Computação - CSBC 2010, 2010.

- DIAS, D. R. C. ; MARCA, A. F. L. ; NETO, M. P. ; BREGA, J. R. F. ; GUIMARAES, M. P. ; LAURIS, J. R. P. . Java 3D para Sistemas de Multiprojeção utilizando Aglomerados Gráficos. In: WRVA 2010, São Paulo. Anais do WRVA 2010, 2010.

- DIAS, D. R. C. ; MARCA, A. F. L. ; VIEIRA, A. M. ; NETO, M. P. ; BREGA, J. R. F. ; GUIMARAES, M. P. ; LAURIS, J. R. P. . Dental Arches Multi-projection System with Semantic Descriptions. In: VSMM 2010, Seoul. VSMM 2010, 2010. v. 1. p. 314-317.

8.2.2 Artigo aceito para publicação

- DIAS, D. R. C. ; BREGA, J. R. F. ; GUIMARAES, M. P. ; GNECCO, B. B. ; LAURIS, J. R. P. . 3D semantic models for odontology education. Communications in Computer and Information Science (Print), 2011.

REFERÊNCIAS

ADAIME, L. M. **Aplicação do Visualization Toolkit para pós-processamento de análises pelo método dos elementos**. Dissertação (Mestrado) — Departamento de Métodos Numéricos em Engenharia, UFPR, Curitiba, 2005.

BARNES, M. Virtual reality and simulation. In: **Proceedings of the 28th conference on Winter simulation**. Washington, DC, USA: IEEE Computer Society, 1996. (WSC '96), p. 101–110. ISBN 0-7803-3383-7. Disponível em: <http://dx.doi.org/10.1145/256562.256583>.

BERNERS-LEE, T. J.; HENDLER. The semantic web. **Scientific American**, 2001.

BILASCO, I. M.; GENSEL, J. 3d technologies for the world wide web. In: XI INTERNATIONAL CONFERENCE ON 3D WEB TECHNOLOGY. France, 2006. p. 65–74.

BRICLKEY, D.; GUHA, R. V. **RDF Vocabulary Description Language 1.0: RDF Schema: W3C Recommendation**. 2004. Disponível em: <http://www.w3c.org/TR/2004/REC-rdf-schema-20040210/>.

CARD, S. K.; MACKINLAY, J. D. **Readings in Information Visualization: Using Vision to Think**. USA: Academic Press, 1999.

CARDOSO, A. **Ambientes Virtuais - Projeto e Implementação**. Porto Alegre: SBC, 2003.

CAVELIB. 2011. Disponível em: <http://www.mechdyne.com/cavelib.aspx>.

CHROMIUM. **Chromium homepage**. 2011. Disponível em: <http://chromium.sourceforge.net/>.

COLLINS, B. M. **Data visualization: has it all been seen before ?** London: Academic Press, 1993. 3-28 p.

CRUZ-NEIRA, C. et al. The cave: audio visual experience automatic virtual environment. **Commun. ACM**, ACM, New York, NY, USA, v. 35, p. 64–72, June 1992. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/129888.129892>.

DEFANTI, T. A. et al. The starcave, a third-generation cave and virtual reality optiportal. **Future Gener. Comput. Syst.**, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 25, p. 169–178, February 2009. ISSN 0167-739X. Disponível em: <http://portal.acm.org/citation.cfm?id=1414095.1414281>.

DIAS, D. C. et al. Dental arches multi-projection system with semantic descriptions. In: **Virtual Systems and Multimedia (VSMM), 2010 16th International Conference on Virtual Systems and Multimedia**. Seoul: IEEE Xplore, 2010. (VSMM 2010, ISBN 978-1-4244-9027-1), p. 314 –317.

DIAS, D. R. C. et al. Java 3d para sistemas de multiprojeção utilizando aglomerados gráficos. In: **WRVA 2010**. São Paulo: SBC 2010, 2010.

DUBLIN CORE. **DCMI Home**. 2011. Disponível em: <http://dublincore.org/>.

FADEL, M. A. V.; COSTA, F. O. C. Utilização da realidade virtual em odontologia. In: **Anais do X Congresso Brasileiro de Informática em Saúde 2006**. Florianópolis: SBIS, 2006.

FIGÚN, M. E.; GARINO, R. R. **Anatomia Odontolóliga Funcional e Aplicada**. São Paulo: Panamericana, 1989. 658 p.

GERMANS, D. et al. Measuring in virtual reality: A case study in dentistry. **Instrumentation and Measurement, IEEE Transactions on**, v. 57, n. 6, p. 1177 –1184, june 2008. ISSN 0018-9456.

GERSHON, N.; EICK, S. G.; CARD, S. Information visualization. **ACM Interactions 5**, v. 2, p. 9–15, 1998.

GLOBUS, A.; RAIBLE, E. Fourteen ways to say nothing with scientific visualization. **Computer**, IEEE Computer Society, Los Alamitos, CA, USA, v. 27, p. 86–88, 1994. ISSN 0018-9162.

GNECCO, B. B. et al. Dicelib: a real time synchronization library for multi-projection virtual reality distributed environments. In: **Simpósio Brasileiro de Realidade Virtual**. Florianópolis: SBC, 2001.

GNECCO, B. B.; GUIMARAES, M. de P.; ZUFFO, M. K. Um framework para computação distribuída. In: **Simpósio Brasileiro de Realidade Virtual**. Ribeirão Preto: SBC, 2003.

GUARINO, N. et al. Toward principles for the design of ontologies used for knowledge sharing. In: **In Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer Academic Publishers, in press. Substantial revision of paper presented at the International Workshop on Formal Ontology**. [S.l.: s.n.], 1993.

GUIMARAES, M. P. **Um ambiente para o desenvolvimento de aplicações de realidade virtual baseadas em aglomerados gráficos**. Tese (Doutorado) — Universidade de São Paulo, 2004.

HANCOCK, D. Viewpoint: virtual reality in search of middle ground. **IEEE Spectrum**, 1995.

HAYES, P. **RDF Semantics: W3C Recommendation**. W3C, 2010. Disponível em: <http://www.w3.org/RDF/>.

HINZ, V. T. **Proposta de Criação de uma Ontologia de Ontologias**. Dissertação (Mestrado) — Universidade Católica de Pelotas, Pelotas, 2006.

JDOM. **JDOM**. 2011. Disponível em: <http://www.jdom.org/>.

KIRNER, C.; SISCOUTTO, R. **Realidade Virtual e Aumentada: Conceitos, Projeto e Aplicação**. Petrópolis: SBC, 2007. (1, ISBN 85-7669-108-6).

KLYNE, G.; CARROL, J. J. **Resource Description Framework (RDF): Concepts and Abstracts Syntax**. W3C, 2004. Disponível em: <http://www.w3.org/TR/rdf-concepts/>.

MAEDCHE, A. **Ontology learning for the semantic Web**. Boston: [s.n.], 2002.

MANOLA, F.; MILLER, E. **RDF Primer W3C Recommendation**. 2004. Disponível em: <http://www.w3.org/TR/rdf-syntax/>.

MCCORMICK, B. H. Visualization in scientific computing. **Computer Graphics: ACM**, p. 1–14, 1987.

MCGUINNESS, D. L.; HARMELEN, F. **OWL Web Ontology Language Overview**. 2004.

MPEG. **MPEG Brasil**. 2011. Disponível em: <http://www.mpeg.org.br/>.

NINTENDO. **Controllers at Nintendo**. 2011. Disponível em: <http://www.nintendo.com/wii/console/controllers>.

PETERSSON, H. et al. Web-based interactive 3d visualization as a tool for improved anatomy learning. **Anatomical Sciences Education**, Wiley Subscription Services, Inc., A Wiley Company, v. 2, n. 2, p. 61–68, 2009. ISSN 1935-9780. Disponível em: <http://dx.doi.org/10.1002/ase.76>.

PITTARELLO, F.; FAVERI, A. D. **Sematinc Description of 3D Environments: a Proposal Based on Web Standards**. 2006. Disponível em: <www.web3d.org/x3d/learn/web3d_2006/085-pittarello.pdf>.

PRIMAL. **Anatomy.TV**. 2011. Disponível em: <www.primalpictures.com>.

RAPOSO, A. et al. Visão estereoscópica, realidade virtual, realidade aumentada e colaboração. In: **Simpósio Brasileiro de Realidade Virtual**. [S.l.]: SBC, 2004. v. 2, n. ISBN 85-88442-95-7, p. 289–331.

SANTHANAM, A. et al. Modeling real-time 3-d lung deformations for medical visualization. **Information Technology in Biomedicine, IEEE Transactions on**, v. 12, n. 2, p. 257 –270, march 2008. ISSN 1089-7771.

SCHIESSL, J. M. Ontologia: O termo e a idéia. **Revista Eletrônica de Biblioteconomia e Ciência da Informação**, v. 24, 2007.

SILÉN, C. et al. Advanced 3d visualization in student-centred medical education. **Medical Teacher**, v. 30, n. 5, p. 115–124, 2008.

SOARES, L. P. **Um Ambiente de multiprojeção totalmente imersivo baseado em aglomerados de computadores**. Tese (Doutorado) — USP, São Paulo, 2005.

STEREOGRAPHICS (Ed.). **The StereoGraphics Developers' Handbook : Background on Creating Images for CrystalEyes and SimulEyes**. 1997. Disponível em: <www.cs.unc.edu/Research/stc/FAQs/Stereo/stereo-handbook.pdf>.

SUN (Ed.). **Sun Microsystem. The Java 3D API Specification**. [S.l.: s.n.], 2000.

SWARTOUT, B. et al. Toward distributed use of large-scale ontologies t. 2002.

TECNOGLASSES. **TECNOGLASSES**. 2011. Disponível em: <http://www.tecnoglasses.com.br/>.

VR JUGGLER. **Vr juggler homepage**. 2011. Disponível em: <http://vrjuggler.org/>.

VRML. **VRML Virtual Reality Modeling Language**. 2011. Disponível em: <http://www.w3.org/MarkUp/VRML>.

WHEATSTONE, C. On some remarkable, and hitherto unobserved, phenomena of binocular vision (part the first). **Philosophical Transactions of the Royal Society of London**, p. 94–371, 1838.

WIIUSEJ. **Java Api for Wiimotes : WiiUseJ**. 2011. Disponível em: <http://code.google.com/p/wiiusej/>.

X3D. **X3D Specifications**. 2011. Disponível em: <http://www.web3d.org/x3d/specifications/>.

XJ3D. **The Xj3D Project**. 2011. Disponível em: <http://www.xj3d.org/>.

APÊNDICE A – CONFIGURAÇÃO DO AGLOMERADO GRÁFICO

Devido à necessidade de bibliotecas de dependência e configurações referentes aos nós do aglomerado, é apresentado um apêndice que possibilita a reconstrução de um aglomerado igualmente configurado ao usado neste trabalho.

O Sistema Operacional utilizado foi o Kubuntu 10.10 devido a sua interface gráfica, o KDE. Testes foram realizados para verificar qual distribuição Linux seria mais indicada para a instalação nos nós do aglomerado gráfico. Durante os testes, o KDE foi mais estável que o Gnome em aplicações Java 3D executadas em tela cheia.

Os pacotes pré-compilados que devem ser instalados são:

```
openjdk-6-jdk.
libjava3d-java.
gcc e g++.
libboost-thread-dev 1.40.
nfs-kernel-server.
ssh.
autoconf.
```

Para que os nós possam trocar informação entre eles, deve-se configurar um servidor ssh em todos os nós. Os nós devem estar configurados de maneira que possam realizar conexões ssh sem a necessidade de utilização de senha. Os comandos necessários para a configuração são:

```
ssh-keygen -t RSA
scp /home/usuario/.ssh/id_rsa.pub usuario@ipno:/tmp
ssh usuario@ipno
cat /tmp/id_rsa.pub >>~/.ssh/authorized_keys2
```

Depois da execução dos comandos, os nós estarão aptos a realizar conexões ssh sem a necessidade de senha.

Também é necessário o compartilhamento de uma pasta criada em um nó que é designado como servidor do aglomerado, porém, este nó também será um nó cliente.

O diretório necessário para a execução deste trabalho foi criado como: /projetos. O primeiro passo é configurar o arquivo /etc/exports, para que seja definido o diretório a ser compartilhado e as permissões de acesso. A seguinte linha deve ser adicionada ao arquivo:

```
/projetos                192.168.1.0/6 (rw,no_root_squash,sync)
```

A segurança para o compartilhamento deve ser configurada. No arquivo /etc/host.allow é necessário adicionar as linhas:

```
Portmap: 192.168.1.0/6
Lockd: 192.168.1.0/6
Mountd: 192.168.1.0/6
Rquotad: 192.168.1.0/6
```

Para que o NFS funcione na inicialização do sistema, devem-se adicionar as seguintes linhas no arquivo /etc/rc.d:

```
chmod a+x rc.portmap
chmod a+x rc.nfsd
```

Por último, deve ser executado o comando:

```
Exportfs.
```

Nos nós cliente do aglomerado deve-se configurar o ponto de montagem NFS definido anteriormente no servidor. No arquivo /etc/fstab adicione a linha:

```
192.168.1.0:/projetos   /mnt/projetos   nfs  rw  0 0
```

APÊNDICE B – INSTALANDO A LIBGLASS

Abaixo é apresentada a compilação e instalação da biblioteca. Ao final da instalação devem ser gerados os arquivos de bibliotecas dinâmicas para ambientes Linux, ou seja, arquivos do tipo so.

A biblioteca pode ser obtida na página do desenvolvedor: http://libglass.sourceforge.net. A versão utilizada no aglomerado utilizado neste trabalho foi a 1.0.3.

Para que não haja problemas, uma ordem de compilação deve ser seguida. No diretório onde os arquivos da libGlass foram descompactados, deve-se gerar o arquivo configure. Este é responsável por gerar a configuração para a geração do arquivo make. Execute o seguinte comando:

```
sudo ./configure
```

Em seguida, agora na pasta src, deve-se executar o seguinte comando:

```
sudo make install
```

Depois da execução das etapas anteriores, os arquivos so já estarão instalados. Deve-se então, dentro da pasta bindings/Java/Glass, executar o arquivo make.

```
sudo make install
```

Ao final dos comandos a libGlass estará instalada. Deve-se realizar a instalação das bibliotecas em todos os nós do aglomerado.

APÊNDICE C – SCRIPTS DE EXECUÇÃO DO AGLOMERADO GRÁFICO

Este apêndice apresenta os scripts desenvolvidos para controlar as aplicações no AG. Deste modo, toda a execução feita pelo usuário por meio de botões, é na verdade realizada pela execução de scripts. Os scripts utilizados são:

- o desligamento do AG: para que o AG possa ser desligado sem a necessidade do usuário conhecer a estrutura, um script de desligamento foi implementado para ser executado pela interface gráfica do sistema do usuário;
- controle dos projetores: para que os projetores possam ser ligados e desligados de acordo com a necessidade da visualização a ser gerada pelo AG, um script de controle dos projetores foi implementado. Este script controla apenas as funções de ligar e desligar os projetores;
- cópia dos arquivos: todos os arquivos carregados pelo usuário na aplicação desktop devem ser copiados para o servidor do AG, pois os outro nós também devem ter acesso a estes arquivos; e
- escolha de visualizações estéreo ou monoscópica: um script de seleção do tipo de visualização também foi implementado. Ele é responsável por provêr ao usuário a opção de configuração das telas de projeção do modo desejado. Por exemplo, visualizações de AV podem ser executadas paralelamente com outros tipos de mídias.

O Script C.1 recebe como parâmetro o modelo virtual que o AG deverá renderizar. Também são atribuídos os diretórios dos arquivos no servidor do AG. Todos os outros nós são executados a partir deste script.

C.1: Executa da aplicação

```
#!/bin/bash
# Executa aplicacao no servidor remoto
```

```
# Testa parametros
#[ "$1" ] || { echo "FATAL: usage: $0 <modelo> , exiting." ; exit 1 ; }
MODEL=$1
#[ -f ${MODEL} ] || { echo "FATAL: model \"${MODEL}\" not found, exiting."
    ; exit 1 ; }

# Var
PATH="/mnt/projetos/dist"
BIN="/usr/bin"
REMOTE_MACHINES="6 2 5 1 4"
SYNERGY_MACHINES="2 1"
###COMMAND="export DISPLAY=:0.0 ; ${BIN}/java -jar ${PATH}/Xj3D.jar ${PATH
    }/Modelos/Esqueleto.WRL 192.168.1.3 &"
COMMAND="cd ${PATH}; export DISPLAY=:0.0 ; ${BIN}/java -jar ${PATH}/Xj3D.
    jar ${PATH}/Modelos/${MODEL} 192.168.1.3 &"


# Main
# Inicia Synergy
${BIN}/synergys &
/bin/sleep 5

for i in ${SYNERGY_MACHINES}
do
        echo " ... Iniciando Synergy ${i} ... "
        ${BIN}/ssh lstr${i}@192.168.1.${i} ${BIN}/synergyc 192.168.1.3 &
        /bin/sleep 5
        echo
done

# Maquina local
export DISPLAY=:0.0
echo " ... Inicialiazando aplicativo java na maquina local ..."
${BIN}/java -jar /projetos/dist/Xj3D.jar &
        /bin/sleep 5
${BIN}/java -jar /projetos/dist/Xj3D.jar Modelos/${MODEL} localhost &
        /bin/sleep 5

#Inicia Wiimote
${BIN}/java -jar /projetos/dist/AppWiiusej.jar &

# Executa nas maquinas remotas
for i in ${REMOTE_MACHINES}
```

```
do
        echo " ... Inicialiazando aplicativo java na maquina ${i} ..."
        /usr/bin/ssh lstr${i}@192.168.1.${i} ${COMMAND} &
        /bin/sleep 5
        echo
done
```

O Script C.2 finaliza todas as aplicações Java que estejam sendo executadas no AG.

C.2: Finaliza da aplicação

```
#!/bin/bash

COMMAND="killall java ; killall synergyc"

for i in 1 2 4 5 6
do
        echo " ... Finalizando java em maquina ${i} ..."
        /usr/bin/ssh lstr${i}@192.168.1.${i} ${COMMAND}
#       /bin/sleep 1
        echo
done

# Maquina local
echo " ... Finalizando java em maquina local..."
killall java
killall synergys
```

O Script C.3 apresenta uma lista com todos as aplicações que estejam sendo executadas pelo AG. Para que o usuário possa saber o que acontece no AG, as saídas do terminal são apresentadas na aplicação do usuário.

C.3: Lista da aplicações Java

```
#!/bin/bash
##set -x

PATH="/mnt/projetos/dist"
COMMAND="ps auxwww | grep java | grep -v grep"

for i in 1 2 4 5 6
do
        echo " ... Acessando maquina ${i} ..."
        /usr/bin/ssh lstr${i}@192.168.1.${i} ${COMMAND}
```

```
        /bin/sleep 1
        echo
done
```

O Script C.4 reinicia todos os nós do AG.

C.4: Reinicia o aglomerado gráfico

```
#!/bin/bash
# Copia remota de arquivos para maquinas

REMOTE_MACHINES="6 2 5 1 4"
COMMAND="sudo reboot"


# Executa nas maquinas remotas
for i in ${REMOTE_MACHINES}
do
        echo " ... Reboot machine ${i} ..."
        /usr/bin/ssh lstr${i}@192.168.1.${i} ${COMMAND}
        /bin/sleep 1
        echo
done
```

O Script C.5 desliga todos os nós do AG.

C.5: Desliga o aglomerado gráfico

```
#!/bin/bash
# Copia remota de arquivos para maquinas

REMOTE_MACHINES="6 2 5 1 4"
COMMAND="sudo halt"

# Executa nas maquinas remotas
for i in ${REMOTE_MACHINES}
do
        echo " ... Reboot machine ${i} ..."
        /usr/bin/ssh lstr${i}@192.168.1.${i} ${COMMAND}
        /bin/sleep 1
        echo
done

# Executa local
        sudo halt
```

O Script C.6 realiza a configuração necessária as portas seriais do servidor do AG e recebe como parâmetro o comando do usuário: ligar ou desligar.

C.6: Controle dos projetores

```
SET_PORT_CONF="stty 115200 cs8 time 5 -parenb -parodd cs8 hupcl -cstopb
    cread clocal -crtscts ignbrk -brkint -ignpar -parmrk -inpck -istrip -
    inlcr -igncr -icrnl -ixon -ixoff -iuclc -ixany -imaxbel -iutf8 -opost -
    olcuc -ocrnl -onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0
    ff0 -isig -icanon -iexten -echo -echoe -echok -echonl -noflsh -xcase -
    tostop -echoprt -echoctl -echoke -F /dev/ttyS"



# script "on:0 1 2"

COMMAND=`echo $1 | awk -F: '{print $1}'`
##EXEC_COMMAND="echo -e \"\r*pow=${COMMAND}#\r\" > /dev/ttyS"
STRING_PORTAS=`echo $1 | awk -F: '{print $2}'`

for i in ${STRING_PORTAS}
do
        ${SET_PORT_CONF}${i}
        echo -e "\r*pow=${COMMAND}#\r" > /dev/ttyS${i}
done
```

ANEXO A – JAVA 3D

A Java 3D é uma API desenvolvida pela Sun Microsystem para renderizar grafos de cena utilizando a linguagem Java. A renderização pode ser feita utilizando o OpenGL ou DirectX, ficando à cargo do código Java a descrição da cena e a lógica de programação, permitindo ao desenvolvedor criar e manipular formas geométricas em 3D (SUN, 2000).

Um ponto forte da Java 3D é a possibilidade de desenvolver aplicações puramente em Java independente do tamanho do AV que o desenvolvedor necessite renderizar. Outra característica de sistemas desenvolvidos em Java 3D é a independência de plataforma, ou seja, qualquer sistema operacional, que possua suporte a JVM, é capaz de executar aplicações Java 3D (SUN, 2000).

Um AV possui objetos geométricos com os quais o usuário pode interagir. Este ambiente pode ser denominado de duas maneiras: VirtualUniverse e SimpleUniverse, sendo essas as classes raiz que contém todos os outros elementos do AV; elementos não geométricos também podem ser anexados aos ambientes, sendo responsáveis por controlar ou influenciar o ambiente. A diferença entre os dois tipos de ambientes é que o SimpleUniverse permite que o grafo de cena seja criado de maneira rápida, utilizando como parâmetro um objeto **Canvas3D**, configurando de maneira automática todas as outras dependências. Também não é necessário o uso de um nó Locale. A Figura 47 apresenta a estrutura de um grafo de cena VirtualUniverse em Java 3D.

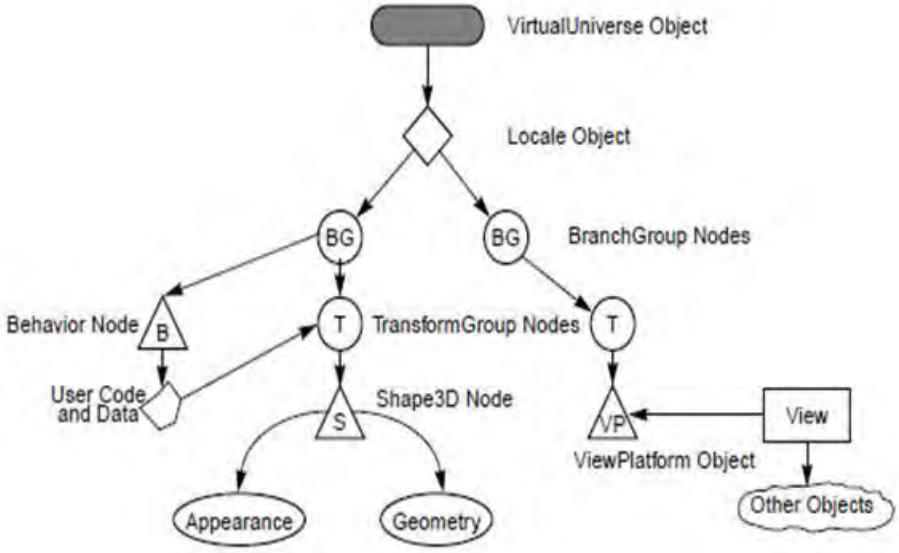


Figura 47: Grafo de cena do Java 3D
Fonte: (SUN, 2000)

O grafo de cena consiste em uma estrutura de árvore. Um ou vários subgrafos podem ser criados e adicionados a um nó VirtualUniverse. As conexões entre os nós são sempre representadas por relacionamentos diretos, isto é, pai para filho.

Um VirtualUniverse possui, no mínimo, um objeto Locale. O objeto Locale define uma região geográfica dentro do grafo de cena. Um BranchGroup serve como raiz para subgrafos, chamados de *branch graphs*, sendo o único objeto que pode ser anexado diretamente a um objeto Locale.

O nó Behavior contém o código para manipular a matriz de transformação associada à geometria dos objetos do AV. O nó TransformGroup está relacionado à posição, orientação e escala dos objetos do AV, sendo relativo ao objeto Locale. O nó Shape3D refere-se a dois objetos: Geometry e Appearance. O objeto Geometry descreve a geometria dos objetos do AV; o objeto Appearance descreve a aparência dos objetos, com relação à cor, textura etc.

Por último, o objeto ViewPlatform, que representa a visão do usuário dentro do AV, sendo referenciado por um objeto View, especificando os parâmetros necessários à renderização da cena do ponto de vista do usuário.

Um objeto importante para o desenvolvimento desse trabalho é provido pela classe **Canvas3D**, responsável por encapsular todos os parâmetros associados com a janela de renderização da cena. O objeto Canvas3D é anexado a um objeto View.

Além dos objeto Canvas3D e View, outros três objetos de visualização podem ser utilizados. Estes são: Screen3D, PhysycalBody e PhysicalEnvironment. O objeto Screen3D encapsula os atributos relacionados ao tamanho físico da janela de renderização, tais como altura e largura de tela. O objeto PhysycalBody encapsula os atributos relacionados ao corpo, ou seja, configuração de posição de cabeça e olhos. O objeto PhysicalEnvinronment é responsável por calibrar informações para rastreadores, tais como, cabeça e mãos.

Quando um grafo de cena é montado em Java 3D todos os objetos visíveis são renderizados, juntamente com o processamento de todos os nós de som e de comportamento. Também são processados dispositivos de entrada e eventos de colisão do ambiente.

A Java 3D permite, de maneira fácil, que o desenvolvedor controle informações referentes a cor, transparência e formato dos objetos. Permite o controle sobre efeitos de fundo, iluminação e efeitos de ambiente, como exemplo, neblina. Os objetos podem ser rotacionados e movidos em tempo de execução. O objeto Bound permite ao desenvolvedor definir um volume no espaço, sendo definido de três maneiras possíveis: como uma caixa, uma esfera ou um traçado de planos para definir um espaço. Com a definição desse objeto é possível aplicar operações de maneira isolada, ou seja, efeitos de ambiente podem ser aplicados a regiões de influência definidas por esses objetos.

Os objetos de um grafo de cena podem ser classificados como **Live** e **Compiled**. Um objeto é dito **Live** quando faz parte de um ambiente ativo; um objeto é dito **Compiled** quando, depois de aplicadas as capacidades dos objetos, o objeto é compilado para que a renderização seja otimizada. O trecho de Código A.1 atribui a capacidade de escrita a um objeto do tipo TransformGroup.

A.1: TransformGroup

```
TransformGroup myTrans = new trasnformGroup();
myTrans.setCapability(Transform.ALLOW_TRANSFORM_WRITE);
```

Depois de aplicada a capacidade de escrita pode-se aplicar transformações no ambiente Java 3D. O trecho de Código A.2 apresenta a transformação.

A.2: Aplicando transformações

```
myTrans.setTransform(myT3D);
```

Porém, o objeto myTrans não permite a leitura, pois a capacidade de leitura não foi atribuída. O trecho de Código A.3 retornará uma execeção.

A.3: Recuperando transformações

```
myTrans.getTransform(myT3D);
```

A atribuição de capacidades a objetos Java 3D deve ser feita de maneira correta, ou seja, apenas capacidades necessárias devem ser atribuídas. Dessa maneira, a renderização do grafo de cena será otimizada.

ANEXO B – ESTEREOSCOPIA

Uma imagem estereoscópica apresenta diferentes perspectivas para os olhos: esquerdo e direito. Dessa maneira, o cérebro sintetiza as imagens do mundo com profundidade estereoscópica (STEREOGRAPHICS, 1997). Segundo Wheatstone (1838), existe um único senso de profundidade. Este é produzido pela disparidade da retina, a estereopsia. Ele defendeu a idéia de que a visão humana funde duas imagens planas, gerando uma imagem estereoscópica.

B.1 DISPARIDADE DA RETINA

A disparidade da retina é causada pelo fato de que cada olho enxerga um ponto diferente do mundo, porém, a disparidade é processada pelo cérebro como uma só imagem do mundo visualizado. A habilidade da mente em combinar as imagens é denominada fusão, o que resulta no "sentimento" de profundidade.

Existe um exemplo simples para que se possa representar a disparidade da retina. Posicionando o dedo polegar e olhando fixamente para ele, os olhos convergem diretamente para o dedo. Dessa maneira, pode-se observar que o plano de fundo da imagem é visualizado em "dobro". Por outro lado, se o plano de fundo for observado com maior atenção acontecerá o contrário, dois dedos polegares são visualizados. Este fenômeno fisiológico é denominado disparidade da retina.

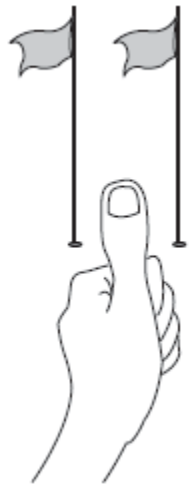


Figura 48: Único dedo, os olhos são convergidos para o dedo e o plano de fundo é visualizado duplicado

Fonte: (STEREOGRAPHICS, 1997)

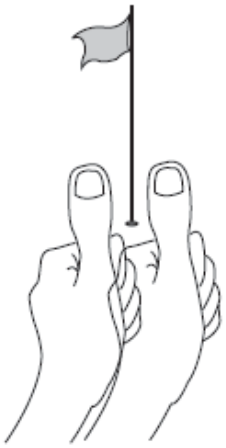


Figura 49: Dois dedos, os olhos são convergidos para o plano de fundo é dois dedos são visualizados

Fonte: (STEREOGRAPHICS, 1997)

B.2 PARALAXE

A paralaxe produz disparidade entre os olhos, gerando assim o efeito de estereoscopia. Diferente da disparidade de retina, que é a distância fisiológica entre os olhos, a paralaxe está relacionada diretamente à distância da projeção na tela de projeção. Dessa maneira, a paralaxe deve ser tratada para que possam ser geradas projeções estereoscópicas. Existem três tipos de paralaxe: Paralaxe Zero, Paralaxe Negativa e Paralaxe Positiva.

- A Paralaxe Zero, também chamada de Zero Paralax Settings (ZPS), é um ponto onde os dois olhos possuem a mesma projeção.
- A Paralaxe Negativa cruza os raios de projeção para cada olho entre os olhos e a tela de projeção. Dessa maneira, é gerada a sensação de que o objeto está "saindo" da tela.
- A Paralaxe Positiva cruza os raios de projeção para cada olho depois da tela de projeção, dando a sensação de que o objeto está atrás da tela.

No desenvolvimento deste trabalho são utilizados os três conceito de Paralaxe, para que os objetos visualizados permitam a sensação de estar dentro, plano ou fora da tela. As Figuras 50, 51 e 52, respectivamente, apresentam: Paralaxe Negativa, Paralaxe Zero e Paralaxe Positiva.

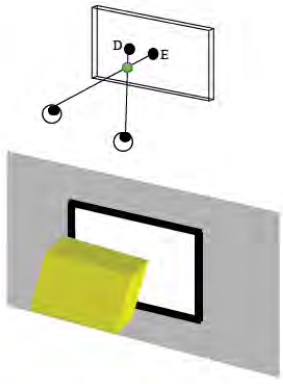


Figura 50: Paralaxe negativa
Fonte: (RAPOSO et al., 2004)

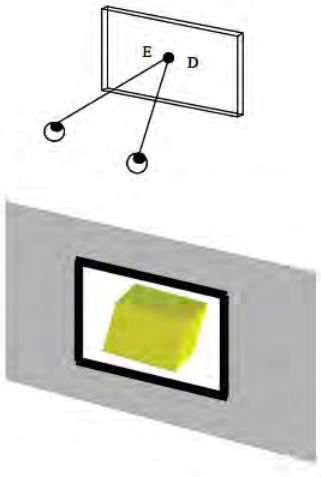


Figura 51: Paralaxe zero
Fonte: (RAPOSO et al., 2004)

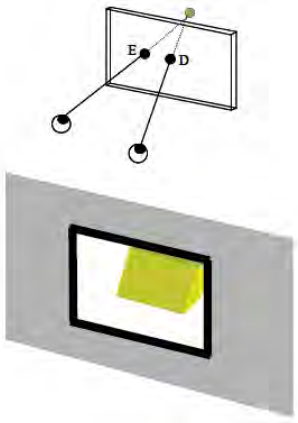


Figura 52: Paralaxe positiva
Fonte: (RAPOSO et al., 2004)

Com relação a Paralaxe Negativa, deve ser levado em consideração a distância do observador perante as telas de projeção. Quanto maior a distancia da tela, maior será o efeito estereoscópico. As Figuras 53 e 54 apresentam, respectivamente, a diferença de distancia para o observador 1 e 2, para Paralaxe Negativa e Positiva.

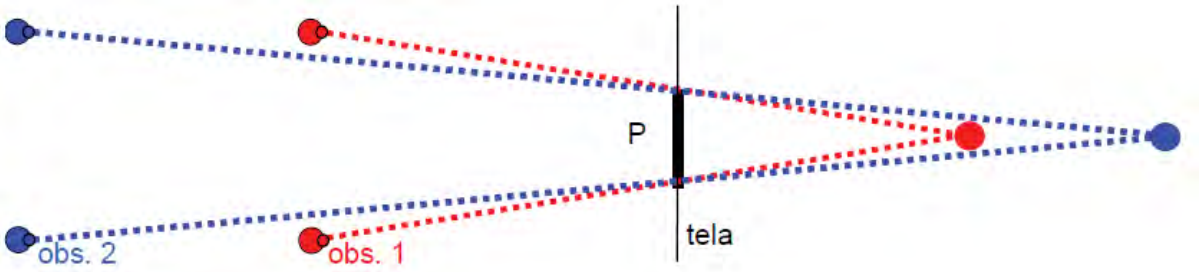


Figura 53: Distância do observador perante a tela de projeção com paralaxe positiva
Fonte: (RAPOSO et al., 2004)

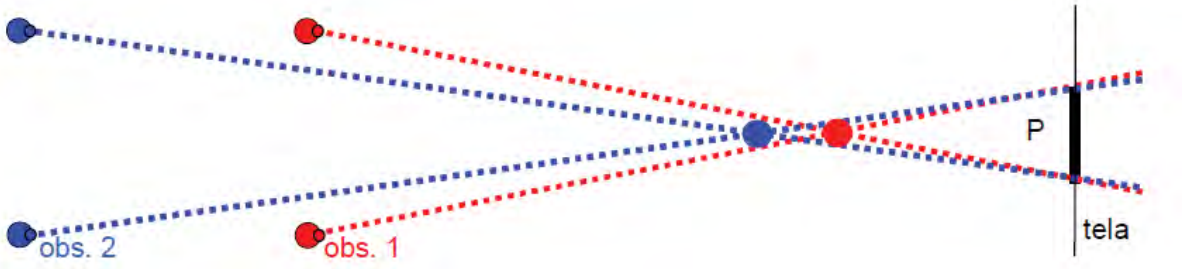


Figura 54: Distância do observador perante a tela de projeção com paralaxe negativa
Fonte: (RAPOSO et al., 2004)

Em geral o valor para a definição da paralaxe deve estar no intervalo [-1,5, 1,5], podendo causar desconforto na visualização se os valores estiverem fora deste intervalo (RAPOSO et al., 2004). Para que seja aplicada o uso de estereoscopia, alguns dispositivos devem ser utilizados. Os dispositivos estereoscópicos podem ser divididos em duas classes: estéreo passivo e estéreo ativo.

- O estéreo passivo filtra os sinais gerados por meio de óculos, separando assim, o sinal de cada olho. Exemplo: óculos polarizados filtrando sinais gerados por lentes polarizadoras.
- O estéreo ativo utiliza sistemas de exibição estéreo para conseguir combinar os sinais. Os óculos são capazes de gerar quadros separados para cada olho, não sendo assim necessário um filtro. Exemplo: óculos obturadores.