UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS
PROGRAMA DE PÓS-GRADUAÇÃO EM FÍSICA

VRMoI - UM AMBIENTE VIRTUAL DISTRIBUÍDO PARA VISUALIZAÇÃO E
ANÁLISE DE MOLÉCULAS DE PROTEÍNAS

Ildeberto Aparecido Rodello

Tese apresentada ao Instituto de Física
de São Carlos, da Universidade de São
Paulo, para obtenção do título de
Doutor em Ciências: Física aplicada:
Física computacional.

Orientador: Prof. Dr. Claudio Kirner

São Carlos
2003

UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS
PROGRAMA DE PÓS-GRADUAÇÃO EM FÍSICA

**VRMoI - UM AMBIENTE VIRTUAL DISTRIBUÍDO PARA VISUALIZAÇÃO E
ANÁLISE DE MOLÉCULAS DE PROTEÍNAS**

Ildeberto Aparecido Rodello

São Carlos
2003

**MEMBROS DA COMISSÃO JULGADORA DA TESE DE DOUTORADO DE
ILDEBERTO APARECIDO RODELLO, APRESENTADA AO INSTITUTO
DE FÍSICA DE SÃO CARLOS, UNIVERSIDADE DE SÃO PAULO, EM 12-
02-2003.**

COMISSÃO JULGADORA:

Prof. Dr. Cláudio Kirner (Orientador e Presidente) – UNIMEP

Prof. Dr. Romero Tori – EP/USP

Prof. Dr. André C. Ponce de Leon Ferreira de Carvalho – ICMC/USP

Prof. Dr. Luciano da Fontoura Costa – IFSC/USP

Prof. Dr. Otavio Henrique Thiemann – IFSC-USP

Dedico esta tese a minha avó Margarida (in memoriam).

*De onde quer que a Senhora esteja,
tenho certeza que está orando por nós.*

Agradecimentos

Gostaria de agradecer a todos que me ajudaram na elaboração dessa tese. Em especial a:

Prof. Dr. Claudio Kirner, meu orientador, pela paciência e inteligência para me indicar o caminho a ser seguido.

Prof. Dr. José Remo Ferreira Brega e Prof. Dr. Antonio Carlos Sementille, pela amizade, pelo grande apoio e pelas valiosas sugestões.

Marcos Roberto Bonfadini, por me ensinar várias coisas a respeito de proteínas e também pelas sugestões.

Maristela Armelin Lima, pelo amor, carinho, respeito e sobretudo compreensão durante esses anos em que me dediquei a este trabalho.

Meus pais, Aparecido e Marina, meus irmãos, Leandro e Neliane, meu avô Roque e minha "Dindinha" Martha, pelo constante incentivo.

Fundação de Ensino Eurípides Soares da Rocha de Marília pela infra-estrutura oferecida.

Deus, por ter me abençoado com mais uma conquista em minha vida.

RESUMO

Este trabalho utiliza conceitos de Realidade Virtual e Sistemas Distribuídos para desenvolver um Ambiente Virtual Distribuído para visualização e análise de moléculas de proteínas, denominado VRMol. O sistema foi implementado com a linguagem Java, incluindo as APIs Java 3D e Java RMI, visando permitir que pesquisadores geograficamente dispersos troquem informações de uma maneira rápida e eficiente, acelerando a pesquisa e discussão remotas. Assim, foram desenvolvidos uma interface gráfica com Java 3D e um conjunto de métodos para troca de mensagens de acordo com o modelo de comunicação cliente/servidor, com Java RMI. Além disso, o sistema também permite a utilização de alguns dispositivos de entrada não convencionais como joystick e luvas.

ABSTRACT

This work use the Virtual Reality and the Distributed Systems concepts to develop a Distributed Virtual Environment to visualize and analyze molecules of proteins, called VRMol. The system was implemented with the Java programming language, including the Java 3D and Java RMI APIs, aiming to allow geographically disperse researches exchange information in a quick and efficient way, speeding up the remote research and discussion. Thus, was developed a graphical interface with Java 3D and a set of methods to exchange messages according to a client/server communication model with Java RMI. Furthermore, the system also allows the use of some non conventional input devices as joysticks and gloves.

Sumário

1. INTRODUÇÃO	15
1.1. OBJETIVOS E IMPORTÂNCIA	17
1.2. ORGANIZAÇÃO DO TRABALHO	18
2. REALIDADE VIRTUAL (RV)	19
2.1. CONCEITOS E DEFINIÇÕES	19
2.2. CLASSIFICAÇÕES.....	21
2.3. APLICAÇÕES DE REALIDADE VIRTUAL.....	23
2.3.1. <i>RV na educação</i>	23
2.3.2. <i>RV na área médica</i>	25
2.3.3. <i>RV na engenharia</i>	27
3. SISTEMAS DE REALIDADE VIRTUAL MULTIUSUÁRIO.....	28
3.1. CONCEITOS E DEFINIÇÕES	28
3.2. FATORES QUE INFLUENCIAM UM AVD.....	31
3.2.1. <i>Número de usuários</i>	31
3.2.2. <i>Número e tamanho dos mundos virtuais</i>	32
3.2.3. <i>Interconexão dos mundos</i>	32
3.2.4. <i>Número de objetos no mundo</i>	33
3.2.5. <i>Nível de detalhamento dos objetos</i>	33
3.2.6. <i>Comportamento dos objetos no mundo virtual</i>	33
3.2.7. <i>Interação entre os usuários no mundo virtual</i>	34
3.3. MODELOS DE COMUNICAÇÃO E ARMAZENAMENTO DE DADOS EM UM AVD	34
3.3.1. <i>Modelos de comunicação</i>	34
3.3.2. <i>Modelos de armazenamento de dados</i>	35
3.4. MEDIDAS DE AVALIAÇÃO DE DESEMPENHO	37
3.5. MODELOS DE SUPORTE PARA COMUNICAÇÃO E DESENVOLVIMENTO.....	39
4. ESTRUTURA E FUNÇÃO DAS PROTEÍNAS	41
4.1. CONCEITOS E DEFINIÇÕES	41
4.2. MODELAGEM MOLECULAR.....	43

4.3. MÉTODOS EMPÍRICOS PARA A DETERMINAÇÃO DA ESTRUTURA MOLECULAR DE UMA PROTEÍNA.....	44
4.3.1. <i>Cristalografia de proteínas</i>	44
4.3.2. <i>Ressonância magnética nuclear</i>	45
4.4. FORMAS DE REPRESENTAÇÃO	45
4.5. SOFTWARES DE SUPORTE	47
4.5.1. <i>Software O</i>	48
4.5.2. <i>Swisspdb Viewer</i>	49
4.5.3. <i>MDL Chime</i>	49
4.5.4. <i>Rasmol</i>	50
4.5.5. <i>WebMol</i>	50
5. RECURSOS UTILIZADOS	52
5.1. JAVA 3D	52
5.1.1. <i>Vantagens e desvantagens</i>	53
5.1.2. <i>Estrutura do Java 3D</i>	54
5.2. JAVA REMOTE METHOD INVOCATION (RMI)	56
5.3. PROTEIN DATA BANK - PDB	58
6. VRMOL - ANÁLISE E PROJETO	62
6.1. OBJETIVOS	62
6.2. ARQUITETURA	63
6.3. PRIMITIVAS DE COMUNICAÇÃO	67
6.3.1. <i>Métodos para conexão e desconexão</i>	68
6.3.1.1. Login.....	68
6.3.1.2. Logout.....	69
6.3.2. <i>Métodos para manipulação e controle de objetos</i>	70
6.3.2.1. Carregamento de um arquivo	70
6.3.2.2. Transferência de coordenadas	72
6.3.3. <i>Métodos para troca de mensagens textuais</i>	73
6.3.3.1. Chat.....	73
7. VRMOL - IMPLEMENTAÇÃO, FUNCIONAMENTO E AVALIAÇÃO.....	74
7.1. IMPLEMENTAÇÃO.....	74
7.2. SUPORTE A DISPOSITIVOS DE ENTRADA	75

7.2.1. Mouse	76
7.2.2. Joystick	77
7.2.2. Luvas.....	79
7.3. FUNCIONAMENTO	81
7.4. CONSIDERAÇÕES SOBRE DESEMPENHO.....	84
7.4.1. Atraso da rede de comunicação	84
7.4.2. Atraso de sintetização.....	85
8. CONCLUSÕES.....	88
8.1. CONTRIBUIÇÕES	89
8.2. RESULTADOS OBTIDOS	89
8.3. TRABALHOS FUTUROS	90
9. REFERÊNCIAS BIBLIOGRÁFICAS	92
APÊNDICE A - JAVA 3D	100
APÊNDICE B - PDB - PROTEIN DATA BANK.....	109

Lista de Figuras

Figura 1 – Principais formas de representação de moléculas (CHIMERA, 2003).	46
Figura 2 - Símbolos para representação de objetos no grafo de cena (BOUVIER, 1999).	54
Figura 3 - Exemplo de grafo de cena.	56
Figura 4 - Modelo de Comunicação: Camadas RMI (JAVARMI, 2002).	58
Figura 5 - Fases para o armazenamento de um registro PDB (BERMAN et al., 2000).	59
Figura 6 - Trechos de um registro PDB.	61
Figura 7 - Arquitetura do VRMol.	63
Figura 8 - Diagrama de Classes - VRMol.	64
Figura 9 - Grafo de Cena Genérico - VRMol.	65
Figura 10 - Ramo do Grafo de Cena Gerado pelo <i>Loader</i>	66
Figura 11 - Diagrama de seqüência de mensagens: login.	69
Figura 12 - Diagrama de seqüência de mensagens: logout.	70
Figura 13 - Diagrama de seqüência de mensagens: carregamento de um arquivo.	71
Figura 14 - Diagrama de seqüência de mensagens: transferência de coordenadas.	72
Figura 15 - Diagrama de seqüência de mensagens: chat.	73
Figura 16 - Interface Client.	75
Figura 17 - Interface Server.	75
Figura 18 - Métodos Abstratos da Classe <i>TornadoMouse</i>	76
Figura 19 - Implementação dos métodos <i>onEndDrag</i> e <i>onRotate</i>	77
Figura 20 - Trecho da classe joystick.	78
Figura 21 - Implementação do método <i>pollAndProcessInput</i>	79
Figura 22 - Movimentação com a luva.	80
Figura 23 - Telas de Login e Principal.	81
Figura 24 - Classe <i>UserInfo</i>	82
Figura 25 - Dois Participantes.	82
Figura 26 - Informações extraídas do PDB.	83
Figura 27 - Classe <i>FileInfo</i>	83
Figura 28 - Gráfico comparativo de atraso de sintetização.	86
Figura 29 - Atraso de sintetização da máquina 2.	87

Figura 30 - Hierarquia de classes Java 3D (BOUVIER, 1999).	101
Figura 31 - Hierarquia de classes: WakeupCondition (BOUVIER, 1999).	107
Figura 32 - Hierarquia de classes: Behavior.	108

Lista de Tabelas

Tabela 1 - Nomenclatura dos Aminoácidos classificados por carga.	42
Tabela 2 - Cores para representação dos átomos.	46
Tabela 3 - Comparação dos modelos de representação.....	47
Tabela 4 - Comparação dos softwares.....	51
Tabela 5 - Informações armazenadas em um registro PDB.....	60
Tabela 6 - Primitivas de comunicação.....	67
Tabela 7- Descrição dos métodos da classe <i>Glove</i>	81
Tabela 8 - Behaviors classificados por estímulos e objetos de mudança (BOUVIER, 1999).	106
Tabela 9 - Campos PDB separados por seção.	110

Lista de Abreviaturas e Siglas

2D - Bidimensional

3D - Tridimensional

ADIT - *AutoDep Input Tool*

API - *Application Program Interface*

AV - Ambiente Virtual

AVC - Ambiente Virtual Colaborativo

AVD - Ambiente Virtual Distribuído

BG - *Branch Group*

CAVE - *Cave Automatic Virtual Environment*

CORBA - *Common Object Request Broker Architecture*

COSY - *Correlated SpectroscopY*

CSCW – *Computer Supported Cooperative Work*

CPK - *Corey-Pauling-Kultin*

DOF - *Degree of Freedom*,

EAD - Educação à Distância

HMD - *Head Mounted Displays*

IHC - Interface Homem-Computador

IP - *Internet Protocol*

JDBC - *Java Database Connection*

JNI - *Java Native method Interface*

PC - *Personal Computer*

PDB - *Protein Data Bank*

NMR - *Nuclear Magnetic Resonance*

NOESY - *Nuclear Overhauser Enhancement SpectroscopY*

RCSB - *Research Collaboratory for Structural Bioinformatics*

RGB - *Red -Green-Blue*

RMI - *Remote Method Invocation*

RMN - Ressonância Magnética Nuclear

RV - Realidade Virtual

SDRV - Sistemas Distribuídos de Realidade Virtual

SGI - *Silicon Graphics*

TG - *Transform Group*

TCP – *Transfer Control Protocol*

UML - *Unified Modeling Language*

WAN - *Wide Area Network*

WoW - *Window on World System*

W2W - *World2World*

WTK - *WorldToolKit*

1. INTRODUÇÃO

Os recursos computacionais estão cada vez mais sendo aplicados em diversas áreas do conhecimento para diversos fins. Medicina, educação e entretenimento são alguns exemplos. As aplicações buscam, dependendo da área, otimizar operações, oferecer mais precisão, facilitar a comunicação, etc.

Oferecendo uma interface muito atraente e intuitiva, a Realidade Virtual (RV) em especial, aparece como uma tecnologia de destaque dentro desses recursos computacionais. Ela se sobressai pela forma como o usuário interage com a máquina.

Ao contrário das interfaces tradicionais que usam, dois eixos cartesianos, X e Y, a RV apresenta a possibilidade de elaboração de interfaces com recursos tridimensionais, usando três eixos: X, Y e Z. Assim, o usuário agora pode manipular os objetos com operações de translação, rotação e escala.

Essa característica é um grande atrativo para aqueles acostumados com as interfaces convencionais. Com a modelagem tridimensional, podem ser criados objetos reproduzindo tanto um objeto real, quanto um objeto fictício. Nessa interface o usuário tem a liberdade de atuar da forma que quiser. Este tipo de interface é denominado Ambiente Virtual (AV).

Para que essa atuação se torne ainda melhor, dispositivos não convencionais de entrada e saída como *joysticks*, capacetes, luvas e óculos estereoscópicos podem ser utilizados. Entretanto, na falta destes, os dispositivos convencionais, como mouse e teclado também podem ser empregados.

Sendo um recurso computacional, a RV também pode ser aplicada em diversas áreas de conhecimento, citando como exemplos aquelas já mencionadas. Neste trabalho, essas técnicas foram aplicadas para auxiliar a visualização e análise de moléculas de proteínas.

As proteínas são muito importantes para todos os seres vivos. O seu estudo, sua compreensão e a modelagem de suas partes constituintes (moléculas e componentes), bem como a determinação de sua relação estrutura-atividade possibilitam, por exemplo, a descoberta de curas para determinadas doenças, ou ainda elaboração de medicamentos auxiliares.

A meta principal das pesquisas nesse campo é desenvolver um ambiente para a análise de moléculas individuais, componentes, seqüência de aminoácidos e suas interações, objetivando a melhor compreensão de propriedades estruturais e bioquímicas (BOBBIE, 1997). Esse ambiente requer, principalmente, a utilização de recursos gráficos tridimensionais (3D) e ferramentas para a modelagem de dados.

Dentro deste contexto, a aplicação da RV visa auxiliar as pesquisas para descoberta de novos medicamentos e/ou estruturas moleculares, oferecendo um AV, no qual o pesquisador possa, não simplesmente visualizar cenários, mas também interagir com eles, baseando-se nas características principais de RV: imersão, interação e envolvimento (MORIE, 1994).

Além disso, o estágio atual das redes de comunicação, sobretudo com o suporte oferecido pela Internet, permite construir este ambiente agregando benefícios como a facilidade para o compartilhamento e troca de informações. Da junção de um suporte de rede de comunicação com um AV, origina-se um Ambiente Virtual Distribuído (AVD).

Um AVD permite que pessoas geograficamente dispersas atuem em um AV, usando uma rede de comunicação para melhorar o desempenho coletivo, por meio da troca de informações (ZYDA; SINGHAL, 1999).

Assim, este trabalho utiliza-se dos conceitos destacados anteriormente para desenvolver um AVD onde um grupo de cientistas e/ou pesquisadores possam discutir e analisar uma molécula de proteína. A partir dessa discussão e análise, conclusões podem ser rapidamente tomadas no sentido de agilizar, por exemplo, a determinação de uma proteína inibidora a partir da qual poderá ser fabricado um novo medicamento.

Os participantes deste AVD proposto, têm facilidade de comunicação por troca de mensagens textuais e pela visualização coletiva da molécula, além das informações sobre as características armazenadas no registro PDB - *Protein Data Bank*, da respectiva molécula.

1.1. Objetivos e importância

O objetivo principal desta Tese de Doutorado, é o desenvolvimento de um AVD para visualização e análise de moléculas de proteínas armazenadas no formato PDB.

É importante destacar que existem vários outros programas que satisfazem a maioria das necessidades dos pesquisadores no que diz respeito a análise molecular. Contudo, procura-se suprir uma deficiência destes que é o suporte à distribuição, ou seja, a facilidade para que pesquisadores geograficamente dispersos possam visualizar e trocar informações de forma rápida sobre uma mesma molécula. Acredita-se que isso possa, sobretudo, acelerar pesquisas.

Paralelamente ao objetivo principal deste trabalho, podem-se citar ainda outros objetivos como:

- Contribuições para desenvolvimento de AVDs e, por consequência, da RV como um todo;
- Utilização da linguagem Java e das suas API's Java RMI e Java 3D, ainda pouco utilizadas para fins de desenvolvimento de ambientes virtuais.
- Implementação dos dispositivos de entrada não convencionais (*joystick* e luvas) na API Java 3D.

A escolha de tais linguagens possibilita, sobretudo:

- Futuras comparações com outros modelos de plataformas utilizadas até o momento;
- Heterogeneidade e portabilidade para o AVD e;
- Utilização das facilidades oferecidas pela Internet como base para a distribuição de informações.

Acredita-se que a importância deste trabalho se enquadre na área de Física Aplicada, procurando oferecer aos pesquisadores mais uma ferramenta para auxiliar o desenvolvimento de seus trabalhos.

Além disso, essa mesma ferramenta pode ser utilizada como instrumento auxiliar em cursos presenciais ou não, sobre proteínas.

1.2. Organização do trabalho

Esta tese de doutorado é composta por 7 capítulos descritos resumidamente da seguinte forma:

O Capítulo 2 apresenta os conceitos de RV, suas características e discute algumas de suas áreas de aplicação.

No Capítulo 3, os conceitos e os principais tópicos relacionados a ambientes virtuais multiusuário são apresentados. São discutidos os fatores que os influenciam, os modelos de comunicação e de armazenamento de dados e as plataformas utilizadas para o desenvolvimento.

Os conceitos da área de aplicação deste trabalho são apresentados no Capítulo 4. São discutidos a estrutura e função das proteínas e os métodos utilizados para determinação de sua estrutura. São ainda discutidos e comparados, algumas formas de representação e alguns dos softwares mais usados por pesquisadores da área.

Os recursos de software utilizados na implementação do trabalho são apresentados no Capítulo 5. Uma visão geral das APIs Java 3D e Java RMI e também a base de dados PDB estão nesse capítulo.

O Capítulo 6 discute a análise e projeto do AVD para visualização e análise de moléculas de proteínas, denominado VRMol. São apresentados sua arquitetura, grafo de cena e o diagrama de classes. É descrito também, todo protocolo de comunicação por meio da explicação de funcionamento das primitivas.

No Capítulo 7 são abordados os aspectos de implementação e funcionamento do VRMol, bem como uma discussão sobre seu desempenho.

O Capítulo 8 relata as conclusões obtidas durante o desenvolvimento desta tese, apontando sua contribuição. Apresenta também, algumas sugestões para trabalhos futuros.

A tese é composta, ainda, por 2 apêndices:

O Apêndice A apresenta a API Java 3D. Ela foi utilizada para a implementação do suporte à renderização das moléculas de proteínas.

Por fim, a base de dados denominada PDB é explicada no Apêndice B.

2. REALIDADE VIRTUAL (RV)

Este capítulo tem como objetivo apresentar os principais conceitos relacionados à Realidade Virtual (RV). Inicialmente são apresentados os conceitos básicos e as definições juntamente com os dispositivos convencionais e não convencionais utilizados. Por fim são evidenciadas algumas áreas do conhecimento onde esses conceitos podem ser ou estão sendo utilizados.

2.1. Conceitos e definições

A interface entre homem (usuário) e máquina (computador) constitui-se em uma área de grande importância para a Ciência da Computação. Vários pesquisadores estudam combinação de cores, novos *lay-outs*, disposição de opções de menu e outros aspectos, em busca de uma maneira adequada para o usuário utilizar o computador. Nesse sentido, a Realidade Virtual (RV) aparece como sendo a forma mais avançada de interface entre usuário e computador até hoje disponível (KIRNER et al., 1996) (HANCOCK, 1995), oferecendo uma forma intuitiva e fácil das pessoas visualizarem, manipularem e interagirem com computadores (AUKSTAKALNIS; BLATNER, 1992).

Baseada na coexistência dos conceitos de *imersão*, *interação* e *envolvimento* (MORIE, 1994), a RV consiste em um ambiente tridimensional gerado por computador, dentro do qual o usuário pode agir de uma forma idêntica ao seu cotidiano, apto a atuar de acordo com 6 graus de liberdade: rotação e translação nos eixos X, Y e Z.

Esses conceitos são alcançados, principalmente, com a utilização de dispositivos visuais, auditivos e físicos especiais. Entretanto, nada impede que

também possam ser obtidos com os dispositivos convencionais, não tendo nesse caso, toda sua potencialidade utilizada.

A idéia de imersão é ligada ao sentimento de estar dentro do ambiente virtual. Normalmente, um sistema imersivo é obtido com o uso de dispositivos visuais especiais como capacetes de visualização (*HMD - Head Mounted Displays*) e óculos estereoscópicos (*Stereoglasses, Shutterglasses*) associados a dispositivos de rastreamento e captura de movimentos (*Body Tracking*). Tais dispositivos possibilitam a visão tridimensional do ambiente gerado, de maneira similar a um ambiente real.

O princípio de utilização dos capacetes consiste em projetar as imagens do ambiente virtual diretamente nos olhos do usuário por meio de dois pequenos monitores. Dessa forma os visores do capacete tornam-se os olhos do indivíduo dentro do mundo. Além disso, todos os movimentos da cabeça também podem ser transferidos ao ambiente, como se a pessoa estivesse realmente dentro daquele ambiente.

A imersão por meio dos óculos estereoscópicos, por sua vez, é alcançada em conjunto com monitores, mesas especiais ou salas com projeção das visões na paredes, piso e teto.

A utilização de óculos em conjunção com monitores é o mais comum, devido principalmente ao baixo custo. Nesse conjunto, as imagens são vistas de maneira estereoscópica na tela. As mesas especiais, também denominadas *Workbenches* (KRUEGER; FROEHLICH, 1994), consistem em uma plataforma com projeções na sua superfície e, com o auxílio dos óculos, as imagens são vistas com efeito de profundidade. Por fim, os sistemas imersivos baseados em salas com projeção das visões nas paredes, piso e teto, denominadas Cavernas Digitais (*CAVE - CAVE Automatic Virtual Environment*) (CRUZ-NEIRA, 1993) (ZUFFO et al. 2001), permitem ao usuário um alto grau de imersão, onde todos seus movimentos são capturados e transferidos para o ambiente. Este modelo, entretanto, é custoso.

Na ausência dos dispositivos especiais, a RV é classificada como *não imersiva*, onde o usuário somente visualiza o ambiente virtual de forma tridimensional diretamente na tela do monitor, interagindo com o ambiente por meio de dispositivos convencionais como *mouse* ou teclado. Estes, porém, limitam as ações dentro do ambiente. A RV não imersiva pode ser chamada de Sistemas de Janelas no Mundo (*WoW - Window on World System*), ou ainda RV de Mesa.

A idéia de interação é ligada com a capacidade do computador detectar as entradas do usuário e modificar instantaneamente o mundo virtual e as ações sobre ele. A detecção dessas entradas também pode estar associada ao uso de dispositivos especiais como por exemplo luvas, *spaceballs*, *joysticks* e *mouses3D*, além dos convencionais *mouse* e teclado.

Munido de luvas, o usuário ganha a capacidade de tocar, apontar, manipular e até sentir algum objeto no mundo virtual. As luvas fazem o papel de uma mão virtual dentro o ambiente, permitindo ao usuário realizar ações que serão respondidas pelo sistema. Além disso, os dispositivos físicos especiais podem trazer algum *feedback* para o usuário. Os *feedbacks* podem ser classificados em tátil, de força ou térmico.

No *feedback* tátil, são transmitidas para o usuário sensações táteis como a textura de uma determinada superfície. No *feedback* de força, o dispositivo utilizado se contrapõe a uma determinada força aplicada pelo usuário, oferecendo resistência e representando variáveis físicas como peso. Por fim, o *feedback* térmico busca representar as sensações de calor e frio.

A idéia de envolvimento, por sua vez, é ligada com o grau de motivação para o engajamento de uma pessoa com determinada atividade. O envolvimento pode ser *passivo*, como passear por um museu, ou *ativo*, como participar do desenvolvimento de um determinado projeto com algum parceiro. A RV tem potencial para os dois tipos de envolvimento, ao permitir a exploração de um ambiente virtual e ao propiciar a interação do usuário com o mundo virtual dinâmico.

2.2. Classificações

São várias as formas pelas quais um AV pode ser classificado. Vários autores propõem formas de classificação, não existindo uma aceita como sendo a única ou a principal. As classificações aqui apresentadas, são as encontradas com mais frequência na literatura.

Os AVs podem ser classificados de acordo com o modelo de interação entre usuário e ambiente, caracterizando assim os sistemas de telepresença, os sistemas de realidade aumentada e os sistemas de realidade melhorada.

Os sistemas de telepresença caracterizam aqueles ambientes onde o usuário atua em um ambiente real que, entretanto, é separado fisicamente do mesmo. Na telepresença, o usuário opera, utilizando-se de dispositivos e técnicas de RV, um telerobô remotamente localizado. Sistemas como este são utilizados para exploração de lugares perigosos (radioativos, profundezas do oceano, etc.) e para pequenos procedimentos cirúrgicos onde o médico está remotamente localizado.

Nos sistemas de realidade aumentada, imagens virtuais geradas pelo computador são combinadas com as imagens do ambiente real. Para tanto são utilizados óculos semitransparentes ou capacetes. A realidade melhorada, por sua vez, é uma melhoria em relação a realidade aumentada. O funcionamento é o mesmo que a realidade aumentada, porém a realidade melhorada enriquece a imagem combinada com dados e informações adicionais.

Os AVs podem ainda ser classificados de acordo com o modelo de participação do usuário, sendo classificada em passiva, exploratória e Interativa (ADAMS, 1994 apud NETTO et al., 2002, p.11).

Em uma participação passiva, o AV permite ao usuário somente a exploração pré-definida, sem a possibilidade de interferência. A rota e o ponto de vista é controlado pelo *software*.

Em um AV que permita a participação exploratória, o usuário tem a liberdade de escolher a rota e o ponto de vista, entretanto não tem liberdade para interação com o ambiente.

A participação interativa oferece todas as possibilidades que eram restritas nas participações passiva e exploratória. O usuário tem liberdade de escolher rota, ponto de vista e interagir com os objetos do ambiente.

Por fim, pode-se classificar os AVs de acordo com o grau de imersão proporcionado ao usuário. De acordo com essa classificação, tem-se: RV de simulação, RV de projeção, *Displays* Visualmente Casados e RV de mesa (PIMENTEL, 1995 apud NETTO et al., 2002, p.9).

A RV de simulação é aquela tida como a mais antiga. Ambientes desse tipo reproduzem um simulador (carro, avião, etc.), oferecendo ao usuário um ambiente similar à simulação em questão.

Na RV de projeção, o usuário não tem uma representação dentro do AV, mas pode se comunicar com objetos ou representações nele contidos. É também conhecida com Realidade Artificial.

Quando o usuário está equipado com dispositivos que lhe apresenta imagens exibidas diretamente nos olhos e que contém sensores que acompanham os movimentos de sua cabeça, tem-se o tipo *Displays Visualmente Casados*.

Na RV de mesa, por sua vez, são utilizados grandes monitores ou algum sistema de projeção para exibição do AV. É um tipo de RV que classifica os ambientes que não possuem dispositivos especiais.

2.3. Aplicações de realidade virtual

As áreas nas quais podem-se aplicar os conceitos de RV são muitas. Talvez seja possível afirmar que os conceitos são aplicados em praticamente todas as áreas do conhecimento. Entretanto, algumas áreas que merecem destaque são a Educação, a Medicina e a Engenharia.

2.3.1. RV na educação

A aplicação da RV na Educação, abre a possibilidade da construção de ambientes tridimensionais que permitam a interação e envolvimento do aluno com objetos nele disponíveis, o que possibilita ao aluno a manipulação da informação. O aluno poderá observar e interagir com situações difíceis ou até mesmo impossíveis de serem simuladas pelo professor no processo de ensino tradicional, onde tais situações são mostradas apenas nos livros, quadro-negro ou vídeo, o qual nem sempre é utilizado. (KUBO et al. 1999).

Além disso, a RV não limita o aluno a apenas alguns ângulos de visualização. Em fotos ou vídeos, a visualização é composta de ângulos predeterminados. Dentro do ambiente virtual, o aprendiz pode explorar os objetos disponíveis sob diferentes perspectivas, de acordo com sua vontade.

Algumas vantagens da utilização de RV na educação são (PANDELIDES, 1995) (EDWARDS, 1996) (SIMA, 1997):

- Estimula estudantes a serem aprendizes mais ativos, pois requer e promove maior interatividade;

- Promove a motivação do estudante, pois possibilita novas formas de visualização de informações, ilustrando também características, processos, etc;
- Possibilidade de múltiplas visões de objetos dentro do ambiente (sob diferentes ângulos), fornecendo oportunidade de melhor compreensão do assunto de estudo;
- Possibilita a visualização e exploração de lugares inexistentes ou de difícil acesso;
- Não restringe o aprendizado ao período regular de aula, possibilitando a realização de atividades educacionais através da internet;
- Possibilita a entrega imediata com *feedback*, sob várias formas (correio eletrônico, material impresso, salas de discussão, etc.);
- Permite que o estudante tenha seu próprio ritmo de aprendizado, o que também irá determinar o seu rendimento;
- Oferece muitas possibilidades para aprendizes não tradicionais, incluindo os desabilitados fisicamente e;
- Pode ser usada como uma poderosa ferramenta de treinamento e de simulação.

Atualmente, com o crescente desenvolvimento da Internet, a RV também pode auxiliar muito no processo de ensino sendo aplicada em Educação à Distância (EAD). Entre os principais motivos para a utilização de interfaces de RV em sistemas de EAD, destaca-se o alto grau de interação que tais interfaces proporcionam, ou seja, possibilitam que alunos e professores, separados fisicamente, se inter-relacionem uns com os outros dentro de uma aplicação educacional.

Tal aplicação permite que alunos e professores de diferentes localizações interajam em um mundo virtual compartilhado, via uma rede de comunicação (Internet), com o objetivo de cooperação e compartilhamento de recursos computacionais em tempo real, caracterizando um ambiente virtual multiusuário (DIZERÓ; KIRNER, 1998).

A aplicação de RV em EAD pode ser complexa no caso de sistemas de telepresença, onde é possível criar uma “reunião virtual” com professores e alunos.

Diferente dos sistemas de videoconferência, o aluno usa um capacete de visualização com rastreador, e “enxerga” o professor como se este estivesse na sua frente, e ao olhar para o lado, o aluno vê seus colegas que estão também “virtualmente” assistindo a aula. (PINHO, 1996).

As técnicas de RV aplicadas a sistemas de EAD permitem que o aluno construa seu próprio conhecimento sem se limitar a tempo/espaço e tendo o professor apenas como um orientador, não como alguém que retém o conhecimento. O aluno segue seu próprio ritmo, realizando um auto-aprendizado. E esta pode ser uma maneira excelente de permitir que portadores de deficiências realizem atividades que de outra forma lhe seriam difíceis ou até mesmo impossíveis.

2.3.2. RV na área médica

As principais áreas de aplicação da RV dentro das ciências médicas são (WATERWORTH, 1999): simulação e planejamento de cirurgias, cirurgia tanto local quanto remota, treinamento e aprimoramento médico e tratamento de fobias ou distúrbios psicológicos. Pode ser utilizada ainda para a visualização de grandes bases de dados médicas e áreas correlatas tais como a fisioterapia.

A simulação cirúrgica e o planejamento de cirurgias têm como objetivo planejar e testar antecipadamente um procedimento operatório delicado onde é necessária uma margem de erro mínima. Dentro do planejamento existe a possibilidade ainda, da cooperação remota de outro profissional, discutindo, sugerindo e visualizando informações adicionais como, por exemplo, exames. Para tanto, um ambiente virtual representando o ambiente real é modelado, e as ações são executadas sobre ele.

A técnicas de RV aplicadas à realização de cirurgias tanto locais quanto remotas podem auxiliar o cirurgião principalmente sob o aspecto de visualização. O cirurgião pode visualizar de forma estereoscópica, durante o procedimento, os exames do paciente ou ainda ter a área a ser operada aumentada ou indicada (MACHADO et al. 2001). A indicação de áreas é feita com a utilização da realidade aumentada e/ou melhorada em conjunto com exames como ultra-sonografia ou tomografia computadorizada.

Tratando-se de cirurgia remota, a telepresença é imprescindível. A presença de dispositivos *haptics* com *feedback* de tato e força e grande precisão também são necessários. Além disso, deve existir um suporte de rede robusto e eficaz, para garantir a contínua conexão entre o local da cirurgia e o cirurgião.

No treinamento e aprimoramento médico a RV também apresenta suas contribuições. O processo de treinamento médico convencional tem a prática testada primeiro em cobaias e depois com o acompanhamento de um médico experiente em uma situação real. Este processo tem várias desvantagens principalmente no caso de utilização de cobaias que podem não representar o cenário real. Além disso, disponibilidade de acompanhamento de um médico nem sempre é possível e programada.

Nesse sentido, a criação de simuladores possibilita que novos cirurgiões treinem o quanto quiserem, diminuindo sua ação em cobaias e apresentando conhecimento melhor quando da sua residência em acompanhamento a um médico experiente.

Os simuladores podem ainda serem utilizados para o aprimoramento médico, ensinando-o e treinando-o com uma nova técnica de intervenção cirúrgica, por exemplo. Assim, custos com cursos podem ser diminuídos e o médico não precisa ficar longe dos seus pacientes durante o período do curso. O mesmo poderá ser feito de acordo com sua disponibilidade, como já mencionado nas vantagens da RV aplicada na Educação.

No tratamento de fobias a RV se mostra eficaz, face aos trabalhos encontrados na literatura. A modelagem e simulação usando RV possibilitam, por exemplo, o tratamento da fobia de altura (JACOB et al., 2001). Sob o aspecto cognitivo, a RV pode auxiliar no tratamento de Autistas (DRUMOMND et al., 2001).

Vale destacar também, a aplicação da RV em áreas paralelas a medicina como a fisioterapia e biomecânica. Atualmente existem simuladores que auxiliam processos fisioterápicos em pacientes normais ou não.

Em pacientes que tiveram derrame cerebral ou aqueles com algum problema motor, existem sistemas que auxiliam a recuperação dos movimentos com o auxílio de dispositivos especiais. Em pessoas sem problemas patológicos, sistemas como o *Motion Analysis System* (2002) são usados para o controle motor funcional e análise de motilidade em atletas.

Por fim, ressalta-se que existem ainda alguns sistemas para auxílio deficientes, principalmente auditivos onde, por meio de sinais, vozes podem ser sintetizadas (SEMENTILLE et al., 2002)

2.3.3. RV na engenharia

No campo da engenharia, a RV pode ser aplicada em vários segmentos, tais como Automotiva, Civil, Naval ou Aeroespacial sendo utilizada principalmente para a redução do ciclo e dos custos de desenvolvimento de produtos (NETTO, 2002). Sendo assim, a principal utilidade da RV neste ramo é a prototipagem virtual.

Com a construção de um protótipo, decisões podem ser tomadas mais rapidamente e custos podem ser diminuídos. No segmento aeroespacial, por exemplo, as empresas podem construir o protótipo de um novo avião e realizar várias simulações que anteriormente só poderiam ser feitas com um protótipo real. Assim, erros podem ser detectados e corrigidos antecipadamente, diminuindo tempo e custos.

Desta forma, as principais vantagens da RV aplicada na área da Engenharia resumem-se em (NETTO, 2002): redução de tempo no desenvolvimento, diminuição de custos e melhoria da qualidade.

3. SISTEMAS DE REALIDADE VIRTUAL MULTIUSUÁRIO

Este capítulo destina-se a apresentar e discutir os aspectos relativos ao desenvolvimento de Sistemas de Realidade Virtual Multiusuário, que também podem ser denominados Sistemas Distribuídos de Realidade Virtual (SDRV) ou Ambientes Virtuais Distribuídos (AVD).

Primeiramente são apresentados alguns conceitos e definições. Em seguida, são discutidos os principais fatores que influenciam um AVD, os modelos que podem ser utilizados para comunicação, os modelos para o armazenamento de dados e as principais medidas utilizadas para se avaliar o desempenho.

Por fim, as formas de implementação de suporte à comunicação são apresentados.

3.1. Conceitos e definições

As aplicações de Realidade Virtual podem ser vistas sob um aspecto bastante amplo, variando de uma única pessoa, usando um único computador, até muitos usuários, usando um sistema distribuído.

Os AVDs, vêm crescendo e apresentam elevado potencial de aplicação. Esse tipo de sistema permite que os usuários geograficamente dispersos atuem em ambientes virtuais compartilhados, usando a rede para melhorar o desempenho coletivo, através da troca de informações (ZYDA; SINGHAL, 1999).

Dentro desse contexto, em AVDs as pessoas podem compartilhar um mesmo espaço virtual de trabalho (*workspace*), onde poderão ajudar-se na execução de uma determinada tarefa, usando os princípios de trabalho cooperativo baseado em computador (*CSCW – Computer Supported Cooperative Work*). Nesse sentido, classificar-se-á o sistema como um Ambiente Virtual Colaborativo (AVC).

Como pode ser observado, o principal diferencial de um AVC é a possibilidade de cooperação entre os usuários na execução de uma determinada tarefa. Em resumo, as propriedades desse ambiente relacionam-se com: espaço, presença e tempo compartilhados; comunicação entre os participantes e; interação com o ambiente.

De uma forma geral, os AVDs podem ser classificados em *centralizados* ou *distribuídos*. No modelo centralizado, todos os usuários compartilham o ambiente virtual, enquanto, no modelo distribuído, o ambiente virtual pode estar replicado (para mundos pequenos) ou particionado (para mundos virtuais de grande porte) (KIRNER et al. 1996). Esses dois modelos podem ser analisados inicialmente sob os aspectos de consistência e facilidade de manutenção.

Nos ambientes centralizados existe uma única base de dados onde todos os usuários atuam. Nesse modelo a consistência é garantida facilmente, entretanto pode apresentar problemas relacionados com escalabilidade e confiabilidade.

Em um ambiente distribuído replicado com n usuários, quando um usuário fizer qualquer alteração no ambiente virtual, isto deverá ser comunicado para todas as $(n-1)$ versões desse ambiente, pois cada usuário tem uma cópia completa do ambiente.

Em um ambiente distribuído particionado com n usuários, a situação é mais complexa, uma vez que o ambiente virtual é dividido em várias partes e cada máquina ficará encarregada de uma delas.

Durante a navegação pelo ambiente, o usuário pode penetrar em partições fora do seu domínio, de forma que sua máquina ou servidor deverá receber uma réplica dessa partição. Se existirem vários usuários em uma mesma região do ambiente virtual, esse grupo de usuários receberá uma cópia dessa região e, qualquer alteração no ambiente virtual, feita por um membro do grupo, será retransmitida somente para o restante do grupo.

Em ambos os casos, existe a preocupação principalmente com a consistência dos dados. No caso replicado, pode se utilizar o modelo *broadcast* para a difusão das mensagens e, no caso particionado, por sua vez, pode ser utilizado o modelo de *multicast* (TANENBAUM, 1995).

Os AVDs figuram entre os sistemas de software mais complexos já construídos. Esses ambientes devem satisfazer uma variedade de características como (STYTZ, 1996):

1. Resposta rápida a novos requisitos do sistema;
2. Capacidade de manutenção;
3. Suportar interação em tempo real
4. Fidelidade da inserção do usuário no mundo virtual em relação a uma referência;
5. Alta taxa de quadros por segundo, reusabilidade e portabilidade;
6. Ajustamento a novas interfaces e dispositivos de visualização e;
7. Requisitos para capacidades adicionais.

De uma forma geral, a criação de AVDs está bastante ligada com realismo visual e interação, sendo fundamental o suporte de áreas tais como computação gráfica 3D, modelagem gráfica, e interação homem-máquina. A elaboração de um AVD inclui atividades envolvendo:

1. Suporte de comunicação em rede;
2. Criação de ambientes virtuais;
3. Atuação no mundo real;
4. Criação de atores gerados por computador;
5. Inserção de fenômenos naturais e;
6. Uso de simulação tradicional.

Dentre as atividades, destaca-se o suporte de comunicação em rede. Por meio dele, fornece-se a base para que unidades computacionais heterogêneas separadas fisicamente sejam unificadas para implementar um único ambiente virtual.

Devido à necessidade de comunicação intensiva entre as máquinas do sistema toda vez que há uma atualização de posição é necessário utilizar-se de técnicas para minimizar o impacto dessa troca de mensagens. O *dead-reckoning* (GOSSWEILER, 1994) (FISHWICK, 1995) (STYTZ, 1996) é uma das técnicas utilizadas para, além de minimizar a troca de mensagens, suportar atrasos de comunicação.

Essa técnica trabalha com a previsão da posição de um elemento, levando em conta o seu trajeto, velocidade e posição anterior, decorrido um certo tempo. Todas as máquinas fazem o mesmo cálculo de previsão e reposicionam o elemento. Aquele que estiver gerenciando o elemento conseguirá verificar a diferença da

trajetória real com a trajetória calculada. Sempre que essa diferença atingir um valor máximo, o valor real da posição será então comunicado às outras máquinas para devida correção. Desta forma, não haverá necessidade de informar continuamente a posição de um elemento para as outras máquinas, o que diminuirá bastante a comunicação pela rede.

3.2. Fatores que influenciam um AVD

Como mencionado, os AVDs figuram entre os sistemas de software mais complexos já construídos e, para que apresente as características que lhe são inerentes, uma série de fatores devem ser levados em consideração.

Os fatores adiante apresentados e comentados afetam, de alguma forma, o desempenho e/ou a complexidade de um AVD. A complexidade, por sua vez, também afeta o desempenho, prejudicando o realismo, na forma de atrasos ou lentidão.

Procura-se estabelecer um nível de complexidade que torne o AVD aceitável, tanto em realismo quanto em desempenho. Os fatores que devem ser levados em consideração para que isso ocorra são (GREENHALGH 1996(2)): número de usuários, número e tamanho dos mundos virtuais, interconexão dos mundos, número de objetos no mundo, nível de detalhamentos dos objetos, comportamento dos objetos no mundo virtual e interação entre os usuários no mundo virtual.

3.2.1. Número de usuários

De acordo com a quantidade de usuários que o ambiente pode suportar de forma simultânea, os sistemas de RV podem ser classificados em monousuário (*single user*) ou multiusuário (*multi-user*).

Em um sistema monousuário é possível a existência de outros usuários que são, entretanto, somente meros espectadores, uma vez que a interação com e a navegação pelo ambiente somente são disponíveis a um único usuário.

Os sistemas multiusuários, por sua vez, podem ser subdivididos em outras duas categorias: *pequena escala*, correspondendo a atuação simultânea de ordem

de 1 dezena de usuários e *grande escala*, acima de 1 dezena de usuários. Nesses sistemas todos os usuários têm liberdade de interação e navegação em qualquer parte do ambiente.

Nota-se que os sistemas multiusuário são mais complexos e exigem um poder maior de computação, relacionado a aspectos como gerenciamento desses usuários (*login*, segurança, etc.). Além disso, o número de tarefas que podem ser executadas pelos usuários é outro fator de interferência.

3.2.2. Número e tamanho dos mundos virtuais

Essa característica corresponde à quantidade dos mundos que podem estar ativos no sistema. Em sistemas monousuário, somente um mundo estará ativo por vez, facilitando o gerenciamento. Em sistemas multiusuário, por outro lado, vários mundos podem estar ativos, com usuários diferentes em diferentes mundos e ainda, diferentes partes de um mesmo mundo, de forma que a complexidade de gerenciamento aumenta no sentido principalmente se estabelecer limites de onde termina ou começa um mundo (particionamento).

Além da quantidade, o tamanho dos mundos é um fator a ser considerado. Mundos menores podem apresentar desempenho melhor para os processo de carregar ou liberar mundos em relação aos mundos maiores.

3.2.3. Interconexão dos mundos

A interconexão entre os mundos corresponde à forma como os diferentes mundos estão limitados e como os usuários ultrapassam esse limite. Como mencionado, um AVD pode estar replicado ou particionado.

No modelo replicado, o sistema preocupa-se tão somente em definir os limites entre os mundos e o portal de entrada. No modelo particionado, por sua vez, além dos limites, o sistema preocupa-se, também, em localizar e carregar o mundo antecipadamente, de forma que o usuário não perceba a mudança.

3.2.4. Número de objetos no mundo

Os objetos que compõem cada mundo também são fatores a serem discutidos em termos de quantidade, nível de detalhes e comportamento. Eles estão relacionados de forma que um equilíbrio contribui para o desempenho do sistema. A quantidade de objetos corresponde a quantos objetos estarão presentes na cena, de acordo com um certo nível de detalhamento e um comportamento permitido.

3.2.5. Nível de detalhamento dos objetos

Objetos mais próximos necessitam de um nível de maior de detalhes, implicando em um número maior de polígonos. Tal nível, entretanto, não deve ser mantido para o mesmo objeto, quando a uma distância maior, uma vez que o realismo tende a ser mantido, reduzindo-se os detalhes dos objetos à medida que o ponto de vista deles se afasta.

3.2.6. Comportamento dos objetos no mundo virtual

O comportamento de um objeto depende do nível de interação que o usuário pode ter. Objetos que o usuário pode somente observar têm um nível de complexidade menor que aquele com o qual o usuário pode interagir, modificar, ajustar etc.

O padrão de comportamento, entretanto, depende da finalidade do mundo virtual. AVCs necessitam de um padrão comportamental mais apurado, principalmente se estiver sendo utilizado para treinamento (OLIVEIRA et al., 2000).

Em um AVC, o usuário pode ainda ter a possibilidade de exploração do objeto segundo dois aspectos: *objetivo* ou *subjetivo* (GREENHALGH, 1996(1)). No aspecto objetivo, os usuários estão em consenso a respeito de um determinado objeto, todos interessados nas mesmas informações. No aspecto subjetivo, o usuário poder ter acesso a informações exclusivas e adicionais a respeito do objeto, isto é, além daquelas presentes no aspecto objetivo.

O modelo subjetivo leva o usuário a conseguir uma maior interação com o ambiente, de forma que este passa a oferecer um conjunto maior de operações disponíveis. Dentre as operações pode-se citar o acesso à base de dados exclusivas de onde o usuário possa filtrar informações.

3.2.7. Interação entre os usuários no mundo virtual

Esse aspecto aborda fatores como representação dos usuários no ambiente (*avatares*) e troca de mensagens entre eles.

A representação acrescenta ao gerenciamento do sistema a preocupação com a localização e visualização dos *avatares*, pontos de vista, riqueza de movimentos e expressões faciais. Tais usuários podem ainda se comunicar, tendo como modelos, a comunicação por texto ou por voz.

3.3. Modelos de comunicação e armazenamento de dados em um AVD

Existem vários modelos que podem ser seguidos para o desenvolvimento de um AVD no que diz respeito a como os dados podem ser armazenados e como ocorre a troca de mensagens. Cada um tem suas vantagens e desvantagens, que são discutidas a seguir.

3.3.1. Modelos de comunicação

Os modelos de comunicação em um AVD são basicamente divididos em: Modelo Cliente/Servidor, Ponto a Ponto, Computação Distribuída Orientada a Objetos e Híbrido.

O paradigma Cliente/Servidor em um AVD implica na comunicação direta somente do cliente com o servidor ou vice-versa, não havendo a troca de mensagens entre os clientes. As vantagens e desvantagens deste modelo estão associadas aos fatores básicos de qualquer modelo cliente/servidor.

Uma desvantagem é a possibilidade de falha no servidor, o que tornaria o sistema inoperante. Para contornar esse problema, a técnica da replicação de servidores pode ser utilizada. Além disso, o servidor é um gargalo em potencial e, em se tratando de um AVD onde coexistem vários usuários, danos podem ser causados em termos de latência. Um eficiente mecanismo de balanceamento de cargas pode solucionar o problema (CARNEIRO; ÁRABE, 1998).

O paradigma Ponto a Ponto implica na comunicação direta entre os *hosts*, sem intermédio de um ou mais servidores. O ponto de estudos neste modelo refere-se à complexidade de gerenciamento do ambiente, uma vez que além do gerenciamento local, cada *host* deve também tratar os acessos remotos.

O paradigma da Computação Distribuída Orientada a Objetos consiste na comunicação do cliente com o serviço de objetos do solicitador e do receptor de serviços que são intermediados por um agente (*broker*). Exemplos deste modelo podem ser encontrados naqueles baseados na arquitetura CORBA (*Common Object Request Broker Architecture*) (CORBA, 2001)

O paradigma híbrido consiste na combinação do modelo Cliente/Servidor e do Ponto a Ponto.

Os modelos de comunicação seguem as estruturas básicas de comunicação que são *unicast*, *broadcast* e *multicast*. Na estrutura *Unicast*, a comunicação dos dados é conhecida como “um-para-um”, sendo realizada somente entre dois *hosts*. *Broadcast* (difusão) é o mecanismo de comunicação dos dados conhecido como “um-para-vários”, em que um *host* envia um dado para todos os outros *hosts* da rede. *Multicast* é o mecanismo de comunicação dos dados conhecido como “um-para-vários” ou “vários-para-vários” em uma única operação. A importância do *multicast* é decorrente da restrição do número de usuários em um mesmo escopo de recepção de mensagens, fazendo um mapeamento entre os grupos de usuários para usar de forma eficiente a largura de banda, reduzir a latência e aumentar a vazão sobre as redes (HAGSAND, 1996).

3.3.2. Modelos de armazenamento de dados

O armazenamento dos dados é um outro fator importante em um AVD, porque implica transferência e consistência das informações dos estados de um

ambiente. Decidir onde as informações serão armazenadas afeta principalmente as escalas de dados do ambiente virtual, os requisitos de comunicação e a confiabilidade no sistema em geral (MACEDONIA; ZYDA, 1997). Pelos mecanismos atuais, a escolha da forma de distribuição e armazenamento dos dados ainda está restrita ao objetivo da aplicação.

Segundo Macedonia e Zyda (1997), o armazenamento dos dados em um ambiente virtual pode ser restrito a: Banco de Dados Centralizado e Compartilhado, Banco de Dados Replicados de Mundos Homogêneos, Banco de Dados Distribuídos e Compartilhado com Alterações Ponto-a-Ponto, e Banco de Dados Distribuídos e Compartilhado Cliente/Servidor.

No modelo Banco de Dados Centralizado e Compartilhado, os dados estão centralizados em uma única máquina, que atua como servidor, permitindo uma melhor consistência do AVD, uma vez que os clientes mantêm somente os dados pertinentes a cena em memória e apenas um usuário pode modificar o banco de dados em um determinado tempo. As desvantagens desse modelo começam com a limitação no número de participantes no ambiente. Quanto maior a quantidade de usuário, maior será a dificuldade em manter a consistência do banco de dados dos objetos em tempo real, tornando o servidor o gargalo da aplicação.

No Banco de Dados Replicados de Mundos Homogêneos, cada *host* conectado na aplicação possui uma réplica completa dos dados do ambiente (o terreno, modelo geométrico, texturas, e comportamento de todos representantes no ambiente virtual) sendo que a consistência de todos os mundos é alcançada pela utilização de mecanismos como “*heartbeat*” (o *host* envia periodicamente mensagens informando o seu estado).

As vantagens dessa abordagem é que as mensagens são relativamente pequenas por serem somente mensagens de atualização. As desvantagens estão diretamente relacionadas com a inflexibilidade e o aumento das informações do ambiente virtual. Deste modo, todas as informações do banco de dados devem estar em todos os *hosts*, o que aumenta a possibilidade do sistema tornar-se inconsistente à medida que o número de participantes aumenta no mundo.

No modelo Banco de Dados Distribuído e Compartilhado com Alterações Ponto-a-Ponto, cada *host* possui apenas uma parte do ambiente em sua base de dados. Logo de início percebe-se que existe um maior número de mensagens, mas usa-se a técnica de *multicast* para diminuir o tráfego.

Este modelo é apropriado para se construir ambientes distribuídos de larga escala e tem como desvantagem a dificuldade de escalabilidade por causa do custo da comunicação associada com a confiabilidade e a consistência de dados através de WANs (*Wide Area Network*).

Por fim, o Banco de Dados Distribuídos e Compartilhado Cliente/Servidor baseia-se na técnica Cliente/Servidor, tendo o banco de dados particionado em seus clientes. Um agente chamado *broker* é responsável pelo conhecimento do servidor para manter o mundo atualizado. Um cliente pode requisitar um dado de forma transparente.

3.4. Medidas de avaliação de desempenho

Todos os aspectos discutidos até o momento, afetam de alguma forma o desempenho do AVD, podendo então ser analisados segundo o prisma de confiabilidade e atraso.

A confiabilidade da rede é uma medida que reflete a quantidade de dados que são perdidos pela rede durante o trajeto do *host* origem para o *host* destino. O dado perdido pode ser classificado como: dado interrompido e dado corrompido.

O dado interrompido é quando ele não chega por completo ao *host* destino. O dado corrompido é quando o conteúdo dos pacotes de dados é alterado durante a transmissão e o que chega ao *host* destino é basicamente inútil. Muitos pesquisadores consideram a interrupção e a corrupção de dados essencialmente equivalentes, isto porque, em ambos os casos, o dado não atinge efetivamente a aplicação destino. A confiabilidade da rede freqüentemente força um compromisso entre a largura de banda e a latência (ZYDA;SINGHAL, 1999).

Atraso é o tempo decorrido entre a atuação do usuário no sistema e a visualização do resultado na aplicação, ou seja, é o tempo entre o usuário realizar uma interação no dispositivo de entrada (por exemplo, luva) do sistema e o resultado dessa ação ser observado no dispositivo de saída (por exemplo, capacete) da aplicação (WLOKA, 1995).

As maiores frustrações nos usuários, em geral, são causadas principalmente pelos atrasos nas respostas de interação do usuário com o sistema. Em um Sistema

de RV, esse problema fica ainda mais acentuado, principalmente em sistemas imersivos.

Em geral, o atraso é um dos principais fatores limitantes para aplicações de RV. Segundo Wloka (1995) as principais fontes de atraso encontradas em AVs são: atraso de dispositivos de entrada do usuário, atraso do processamento, atraso de sintetização, atraso de sincronização e atraso induzido pela taxa de quadros por segundo.

O atraso de dispositivos de entrada do usuário consiste no tempo de comunicação de dados desses dispositivos, envolvendo o atraso decorrente da geração de sinal e da comunicação. Dependendo do tipo de dispositivo e do modo de operação, o atraso varia de 10 a 120 ms, com vazão entre 30 e 50 quadros por segundo.

O atraso de processamento é altamente dependente da aplicação, sendo assim, altamente variável. Quanto mais complexo for o sistema, maior será o atraso de processamento. O atraso, nesse caso, varia entre 1 e 500 ms (é claro que algumas aplicações podem ter um atraso superior a 500 ms, mas nesse caso não seriam caracterizadas como imersivas).

O atraso de sintetização é caracterizado pelo tempo decorrido desde o envio de dados para o hardware de sintetização até que esse mesmo dado seja mostrado no monitor. A taxa de atualização do monitor (tipicamente entre 60 - 72 Hz) causa um atraso de sintetização que pode variar de 0 a 17 ms, tipicamente. O atraso de sintetização é altamente dependente da complexidade da cena a ser mostrada, e assim esse atraso é variável durante a execução de uma aplicação.

O atraso de sincronização envolve as aplicações paralelas de Realidade Virtual que processam vários estágios de entrada do usuário: processamento de dispositivos de entrada do usuário, processamento dependente da aplicação e sintetização. Atraso de sincronização é definido como o tempo total em que uma amostra (dados) espera entre os estágios de entrada sem a existência de processamento. Por exemplo, um dispositivo de entrada envia uma nova amostra para a porta serial logo depois da aplicação ler esta porta. Assim, a aplicação ficará ocupada, processando as informações antigas. Essa aplicação levará um certo tempo até ler novamente a porta serial. Esse tempo, em que os novos dados de entrada ficam sem serem processados, é chamado de atraso de sincronização. No melhor caso, o atraso de sincronização é zero: quando cada estágio escreve suas

saídas exatamente antes da leitura do próximo estágio. No pior caso, a saída em cada estágio ocorre justamente depois da leitura do próximo estágio.

O atraso induzido pela taxa de quadros ocorre quando os dados da tela não são atualizados e os dados encontram-se em processo de atualização. A soma de todos esses atrasos é conhecida como atraso de fim-a-fim.

Além desses fatores, a escalabilidade (TANENBAUM, 1995) é um outro ponto a ser analisado. Ela consiste em se analisar a capacidade de crescimento de um sistema em termos de usuários conectados ao mundo. Quanto maior a quantidade de usuários, maior será a dificuldade de gerenciamento do mundo e por consequência, o sistema fica sujeito a atrasos empobrecendo o desempenho.

3.5. Modelos de suporte para comunicação e desenvolvimento

São vários os modelos de suporte de comunicação que podem ser utilizados. Geralmente o desenvolvedor procura um ambiente que lhe abstraia detalhes de comunicação. São vários os pacotes de software que oferecem tais serviços. Pode-se dividi-los em três categorias: baseados em *sockets*, *toolkits* e *middlewares*.

Os *sockets* são primitivas que estabelecem um canal de comunicação inter-processos, por meio de chamadas de sistemas. No *socket* estabelece-se uma porta de comunicação onde mensagens são enviadas e recebidas, baseando-se em um domínio de endereço (*Unix Domain* ou *Internet Domain*) e em um tipo (*Stream Sockets* ou *Datagram Sockets*). Como exemplo pode-se citar aplicações desenvolvidas na linguagem Java e VRML (KUBO, 2000) (KIRNER et al., 2001).

Na categoria *toolkits* se enquadram os modelos desenvolvidos especificamente para um conjunto de desenvolvimento. Como exemplo pode-se citar o *World2World* (W2W) e *WorldToolKit* (WTK) (SENSE8, 1998), DIVE (HAGSAND, 1996), MASSIVE (GREENHALGH; BENFORD, 1995), entre outros.

O *World2World* é uma solução em rede baseada no modelo cliente/servidor para simulações 3D interativas em tempo real. Os componentes clientes do *World2World* integram-se tanto com *WorldToolKit* quanto *WorldUp* (SENSE8, 1998), facilitando o desenvolvimento de simulações multiusuários em ambos os produtos.

Por fim tem-se os *middlewares*, que são independentes de plataformas. Como exemplo pode-se citar o modelo CORBA (*Common Object Request Broker*

Architecture) (VINOSKI, 1994) (CORBA, 2001), Java RMI (*Remote Method Invocation*) (JAVARMI, 2002) além de outros.

O modelo CORBA é um padrão estabelecido pela OMG (*Object Management Group*) para desenvolvimento de aplicações distribuídas segundo o modelo cliente/servidor, usando o paradigma de orientação a objetos. O modelo CORBA baseia-se em um barramento de interconexão de objetos (*Broker*), habilitando a comunicação transparente entre clientes e objetos remotos.

Segundo Deriggi Jr. et. al (1998) e (1999), os sistemas desenvolvidos com base em *toolkits* tendem a ter um desempenho melhor quando comparados à outras plataformas, entretanto apresentam problemas relacionados a portabilidade e dependência de plataforma específica. Esse problema pode ser solucionado utilizando-se a plataforma CORBA. A desvantagem relacionada ao CORBA é a ausência de um padrão para estabelecimento de comunicação em ambientes virtuais. Propostas têm sido apresentadas como em (DERIGGI JR. et al., 1999), (SEMENTILLE, 1999) e (KUBO, 2000). A desvantagem associada ao desenvolvimento com *sockets*, refere-se principalmente a escalabilidade.

4. ESTRUTURA E FUNÇÃO DAS PROTEÍNAS

Este capítulo trata de proteínas, área de aplicação desta tese e onde foram empregadas as técnicas e recursos de RV apresentados nos capítulos anteriores. São apresentados os conceitos e as definições relacionadas à estrutura e função das mesmas. São discutidos também: os métodos para se obter uma proteína, o modelo de armazenamento e as formas de representação. Por fim, é feita uma comparação dos principais softwares que auxiliam a análise de moléculas de proteínas.

4.1. Conceitos e definições

As proteínas são macromoléculas compostas por peptídeos, cuja fórmula geral é $R-CH(NH_2)-COOH$, onde R representa a cadeia lateral do aminoácido (aa) e é o que distingue cada um dos 20 aa codificados pelo código genético. Este código genético é interpretado pelo ribossomo dando origem às proteínas (BRANDEN; TOOZE, 1991). A Tabela 1 apresenta uma lista destes 20 aa agrupados de acordo com suas propriedades de carga. As ligações entre aminoácidos numa proteína são denominadas ligações peptídicas.

As proteínas são de importância fundamental para quase todas as funções da célula: são responsáveis pelas funções biocatalíticas; transporte celular; mediação na conversão energética; suporte celular na forma do citoesqueleto, matriz extracelular, e membrana celular (DOSE, 1982). As proteínas ainda são responsáveis pelo sistema imunológico. Na condição de anticorpos elas agem "desativando" partículas estranhas, como bactérias e vírus, aos organismos.

O estudo da estrutura de proteínas também é fundamental para cientistas de diversas áreas tais como: medicina – para a elaboração de novos e melhores

fármacos contra as mais diversas doenças; agricultura - para melhorar o crescimento, produtividade e aumentar a resistência das plantações contra pragas, etc.

Tabela 1 - Nomenclatura dos Aminoácidos classificados por carga.

Nome	Código (3 letras)	Código (1 letra)	Tipo
Glicina	GLY	G	Glicina
Alanina	ALA	A	Hidrofóbicos
Valina	VAL	V	
Fenilalanina	PHE	F	
Prolina	PRO	P	
Metionina	MET	M	
Isoleucina	ILE	I	
Leucina	LEU	L	
Aspartato	ASP	D	Carregados
Glutamato	GLU	E	
Lisina	LYS	K	
Arginina	ARG	R	
Serina	SER	S	Polares
Treonina	THR	T	
Tirosina	TYR	Y	
Histidina	HIS	H	
Cisteína	CYS	C	
Asparagina	ASN	N	
Glutamina	GLN	Q	
Triptofano	TRP	W	

Para que se possa entender o funcionamento e as características de uma proteína, são necessários estudos modificando suas propriedades fazendo-se mutações sítio dirigidas, avaliações do seu sítio ativo ou elaborações de inibidores. E para tanto, é intrinsecamente necessária a construção de modelos atômicos confiáveis de sua estrutura tridimensional.

A determinação da estrutura de uma proteína pode ser feita por meio de métodos teóricos, envolvendo modelagem molecular e/ou por métodos experimentais como cristalografia de *raios-X* e espectroscopia por ressonância magnética nuclear (RMN).

Uma vez determinadas, experimentalmente ou teoricamente, as coordenadas da estrutura molecular podem ser armazenadas na base de dados denominada PDB, tornando-se disponível a todos os interessados em tal estrutura.

Estas coordenadas permitirão a visualização das moléculas de proteína nos estudos citados anteriormente e em diversas outras áreas da ciência.

4.2. Modelagem molecular

O projeto de uma nova estrutura é um processo interativo que se inicia quando um químico identifica um componente que parece ter um perfil biológico interessante e termina quando, tanto o perfil da atividade, quanto a síntese química da nova entidade química, estiverem otimizados (RICHON, 1994).

As abordagens tradicionais para a descoberta de uma estrutura contam com uma síntese passo a passo e um programa de visualização para otimizar, além dos perfis da atividade de um grande número de componentes, sua geometria e reatividade.

Um dos princípios básicos é que a atividade biológica é dependente da colocação tridimensional de um grupo funcional específico. Avanços no desenvolvimento de novos modelos matemáticos que descrevem fenômenos químicos e o desenvolvimento de interfaces mais intuitivas (como RV), juntamente com a disponibilidade de computadores mais rápidos fornecem aos cientistas, um novo conjunto de ferramentas computacionais.

Tais ferramentas em conjunto com os métodos de pesquisa tradicionais têm alcançado êxito ao examinar as propriedades estruturais dos componentes existentes e também, desenvolver e quantificar uma hipótese que se relaciona com tais propriedades.

A química computacional/modelagem molecular é uma ciência que representa numericamente as estruturas moleculares para a simulação de seu comportamento com equações de quantidade e física clássica. A computação permite aos químicos, gerar e apresentar os dados moleculares, incluindo as geometrias (comprimento das ligações, ângulos das ligações e ângulos de torção), energias (calor de formação, energia de ativação, etc.), propriedades eletrônicas (momentos, cargas, potencial de ionização, etc.), propriedades espectroscópicas (modos vibracionais, mudanças químicas) e propriedades de massa (volume, superfície, difusão, viscosidade, etc.).

Tais métodos de modelagem podem ser auxiliados por recursos da cristalografia (Raio X ou RMN) na determinação das coordenadas de uma estrutura

molecular que será armazenada na base de dados PDB. Tais coordenadas permitirão a simulação e a visualização das moléculas.

A simulação de uma molécula mostra o comportamento de sua estrutura sob diferentes aspectos tais como número de ions ou átomos modelados, hibridação e esquema de ligações (BULATOV; GRIMES, 1996).

Com as coordenadas contidas no PDB, recursos gráficos e técnicas de *design*, as moléculas podem ser visualizadas em estações de trabalho. A visualização mostra detalhes pertinentes a cada molécula, facilitando a atuação de um especialista. As pesquisas referentes a visualização são direcionadas para a otimização de algoritmos de renderização e para a possibilidade da visualização distribuída.

4.3. Métodos empíricos para a determinação da estrutura molecular de uma proteína

Existem duas técnicas para se determinar a estrutura atômica de uma proteína: a *difração por raios-X* (cristalografia de proteínas) e a *espectroscopia de ressonância magnética nuclear* (RMN ou NMR - *Nuclear Magnetic Resonance*). A limitação básica de cada uma consiste na limitação do tamanho da proteína cuja estrutura tridimensional quer obter-se (PANEPUCCI, 1994).

4.3.1. Cristalografia de proteínas

A difração por raios-X é o método mais utilizado para a determinação da estrutura tridimensional de uma proteína. A técnica se baseia na absorção e emissão de raios-X pelos elétrons da proteína. O resultado final é um arquivo contendo uma lista de átomos da proteína com 5 valores experimentais: as coordenadas X, Y e Z e um fator de agitação térmica B e opcionalmente um fator de ocupação Occ.

É importante ressaltar que a técnica de difração por raios-X necessita de grandes quantidades de proteína num estado de altíssima pureza e não tem restrições quanto ao tamanho da proteína.

4.3.2. Ressonância magnética nuclear

A Ressonância Magnética Nuclear utiliza as diferenças nos picos de espectro resultante dos campos magnéticos do núcleo e dos átomos vizinhos ao núcleo para gerar a estrutura molecular. Para tanto, existem dois tipos de experimentos de RMN.

A espectroscopia correlacionada (*correlated spectroscopy* – COSY) resulta em um mapa bidimensional, onde os picos definem conectividades entre os prótons separados por não mais que três ligações covalentes. A espectroscopia bidimensional de melhoramento nuclear *Overhauser* (2D nuclear Overhauser enhancement spectroscopy – NOESY), por sua vez, fornece um mapa bidimensional, onde os picos representam as distâncias entre pares de prótons separados por menos de 5 *Angstroms*. A intensidade de um pico fornece a distância entre os prótons que deram origem ao pico.

A RMN, embora não necessite de cristais de proteína, tem limites no tamanho da proteína cuja estrutura tridimensional quer-se obter.

4.4. Formas de representação

A visualização tridimensional da estrutura é importante para ajudar na compreensão da relação entre as características estruturais e funcionais da molécula estudada. As principais formas de representação visual de uma estrutura molecular são: *wireframe*, *sticks*, *balls and sticks*, *ribbons* e CPK (*Corey-Pauling-Kultin*). A Figura 1 ilustra essas formas de representação.

O modelo de linhas (*wireframe*): mostra os átomos da maneira mais tradicional, no formato de linhas representando as ligações entre eles.

O modelo *sticks*, por sua vez, representa as ligações com cilindros sólidos.

A representação no modelo *balls and sticks* consiste em utilizar esferas (bolas) para os átomos e cilindros sólidos para as ligações entre eles.

O CPK é um modelo que apresenta somente os átomos. Estes são representados por esferas cujo raio é baseado na medida de van der Waals.

Por fim, o modelo *ribbon*, representa a cadeia principal (*backbone*) da molécula por meio de uma seta achatada, denominada fita. Estas fitas caracterizam as hélices alfa.

Figura 1 – Principais formas de representação de moléculas (CHIMERA, 2003).

Além disso, para a identificação dos átomos na molécula pode ser utilizado um esquema de cores, adotado inicialmente no modelo CPK. A Tabela 2 apresenta os átomos e suas respectivas cores para representação.

Tabela 2 - Cores para representação dos átomos.

Elemento	Cor	RGB	Hexadecimal
Carbono	Cinza Claro	[200,200,200]	C8C8C8
Oxigênio	Vermelho	[240,0,0]	F00000
Hidrogênio	Branco	[255,255,255]	FFFFFF
Nitrogênio	Azul Claro	[143,143,255]	8F8FFF
Enxofre	Amarelo	[255,200,50]	FFC832
Fósforo/Ferro	Laranja	[255,165,0]	FFA500
Cloro	Verde	[0,255,0]	00FF00
Bromo/Zinco/Cobre/Niquel	Marrom	[165,42,42]	A52A2A
Sódio	Azul	[0,0,255]	0000FF
Magnésio	Verde Escuro	[42,128,42]	2A802A
Cálcio/Manganês/Alumínio/Prata	Cinza Escuro	[128,128,128]	808080
Outros	Rosa	[255,20,147]	FF1493

Dependendo da informação que se deseja extrair, uma forma diferente de representação pode se adotar. A Tabela 3 compara alguns dos modelos apresentados comparando o tipo da representação, a descrição da representação e também as informações representadas e não representadas. (CASIDAY; FREY, 2000).

Tabela 3 - Comparação dos modelos de representação.

Tipo de Representação	Descrição da Representação	Informações Representadas (Prós)	Informações não Representadas (Contras)
Sticks	Mostra as ligações entre os átomos na forma de cilindros (<i>sticks</i>).	Informa as conexões entre os átomos e fornece uma pequena idéia da conformação tridimensional.	Não ilustra o volume da molécula ou seus átomos constituintes. Fornece visão limitada da conformação tridimensional.
CPK	Mostra os átomos como esferas com tamanho de raio proporcionais aos raios dos átomos de van der Waals	Mostra os volumes relativos dos componentes. É um bom indicador da forma e tamanho da molécula.	Difícil visualização de todos os átomos na molécula e como os átomos estão conectados.
Ribbon	Mostra as moléculas como um <i>backbone</i> , com uma fita (<i>ribbon</i>) caracterizando as hélices alfa.	Mostra a estrutura secundária da proteína.	Não mostra átomos individuais e outras importantes características estruturais da molécula.

4.5. Softwares de suporte

Devido a crescente descoberta e interpretação de genes e novas proteínas, aumentou-se muito a quantidade de informações que devem ser armazenadas e analisadas. O correto armazenamento e principalmente a correta e eficiente recuperação e compreensão dessas informações torna-se um fator imprescindível para o avanço das pesquisas. Nesse sentido, o auxílio computacional é de fundamental importância.

Segundo Lander et al. (LANDER et al. 1991), a compreensão e interpretação de grandes quantidades de dados que são coletados requerem avanços em áreas da computação como análise de seqüências, armazenamento e recuperação de informações e prognóstico de estruturas de proteínas.

Tais avanços são motivados principalmente pela genética e bioquímica. Com a decifragem do código genético humano (projeto Genoma), as seqüências de DNA

(ácido dextroribonucleico), devem ser corretamente armazenadas para depois serem recuperadas e analisadas. No mesmo sentido, descoberta de novas proteínas, também tem requisitos de armazenamento, além da visualização tridimensional e análise de sua estrutura.

O simples armazenamento pode não parecer difícil, entretanto o modelo de recuperação que permitirá ao usuário extrair informações compreensíveis é complicado (LANDER et al. 1991). Tratando-se de proteínas, por exemplo, a recuperação ou escolha incorreta de uma sequência pode não alcançar resultados esperados.

O fator de complexidade pode ainda aumentar considerando-se a utilização de bancos de dados colaborativos, capazes de permitir o trabalho conjunto de diversos centros de pesquisa ou laboratórios. Nesse modelo, deve-se prever um mecanismo eficiente de propagação de atualizações.

O trabalho cooperativo foi impulsionado e facilitado com o surgimento da Internet nos meados de 1989. A internet impulsionou, sobretudo, a velocidade de disseminação das informações que outrora eram restritas a grupo de pesquisas. Atualmente as descobertas são vinculadas quase que em tempo real e ficam disponibilizadas na grande quantidade de base de dados disponíveis.

Nesse sentido, diversos *softwares* auxiliam os pesquisadores de várias maneiras. Eles se enquadram principalmente nas seguintes categorias (PDB, 2002): Cristalografia, RMN, verificação e análise de estruturas, modelagem e simulação e visualização gráfica. Alguns que merecem destaque são:

4.5.1. Software O

Desenvolvido por T.A. Jones e M. Kjeldgaard, Professores do Departamento de Biologia Molecular da Universidade *Uppsala*, Suécia e do Departamento de Química da Universidade *Aarhus*, Dinamarca, respectivamente, o software O é um ambiente para modelagem macromolecular usado para construção e manipulação de modelos baseados em mapas de densidade de elétrons (JONES; KJELDGAARD, 2002).

Depois de carregadas, as informações moleculares são mantidas em uma base de dados com estrutura pré-definida com a finalidade de adicionar versatilidade

ao sistema. A base de dados é equivalente a um simples sistema de arquivos, onde cada arquivo é denominado *datablocks* ou *parameter blocks*. A cada *datablock* é associado um cabeçalho contendo campos tais como: nome, tamanho, status de leitura/escrita e tipo de dados , entre outros.

O software O permite também, a criação de macros para otimizar sua utilização. Um outro detalhe que merece destaque é o suporte para utilização de óculos estereoscópicos.

4.5.2. Swisspdb Viewer

O Swisspdb Viewer é um software para visualização gráfica e análise de macromoléculas desenvolvido na Suíça por meio da colaboração do *Swiss Institute of Bioinformatics*, da *GlaxoSmithKline R&D* de Genebra e do *Structural Bioinformatics Group* de Basel (GUEx et al. 2002).

Ele oferece uma interface que permite a análise de várias proteínas ao mesmo tempo, possibilitando a superposição das mesmas. Com a possibilidade de superposição, a dedução de alinhamentos estruturais e a comparação de sítios ativos são facilitadas. Além disso, pelo Swisspdb Viewer também podem ser obtidos as mutações dos aminoácidos, ligações de hidrogênio, ângulos e distância entre átomos.

É possível ainda ler mapas de densidade de elétrons e visualizar a molécula de forma estereoscópica.

4.5.3. MDL Chime

O MDL Chime é um *plugin* que possibilita a exibição de moléculas tanto no formato 2D (bidimensional) quanto 3D (Tridimensional). Ele foi desenvolvido pela *MDL Information Systems, Inc.*, sediada em San Leandro, CA, Estados Unidos (MDL 2002).

Sendo um *plugin*, ele permite que cientistas visualizem moléculas usando os mesmos navegadores utilizados na internet. Permite, ainda, que esta visualização possa ser feita com Java, em forma de *applets*.

4.5.4. Rasmol

O Rasmol é uma ferramenta para visualização gráfica de proteínas, ácidos nucleicos e pequenas moléculas. Ela foi originalmente desenvolvida por Roger Sayle do *Biomolecular Structures Group, Glaxo Wellcome Research & Development Stevenage, Hertfordshire*, Reino Unido (SAYLE, 2001).

Apresenta várias opções de cores e esquemas de representação moleculares, dentre os quais pode-se citar: *wireframes*, *sticks*, CPK, *ball and stick* e *ribbons*, entre outros. Além disso, suporta uma série de formatos de arquivos de entrada, não tão somente o PDB.

Vale ressaltar, ainda, que os átomos podem ser rotulados e diferentes partes da molécula podem ser representadas ou colorizadas independentemente do resto da molécula ou ainda exibida em outras representações simultaneamente.

4.5.5. WebMol

O WebMol foi desenvolvido por Dirk Walther, do *Department of Cellular and Molecular Pharmacology* da *University of California, San Francisco* (WALTHER, 2002).

Ele foi implementado na linguagem Java para exibição e análise de informações estruturais de um registro PDB. Pode ser utilizado tanto na forma de *applet* quanto na forma de *standalone application*.

Suas principais características são: (WALTHER, 2002):

- Representação das estruturas no formato *wireframe*;
- Diferentes esquemas de cores para tipos de átomos, estrutura secundária e *B-factor*, entre outros;
- Estéreoscopia;
- Translação, *zoom* e manipulações por meio do mouse;
- Seleção de foco e;
- Seleção de resíduos específicos, entre outros.

Há de se destacar novamente que estes softwares são somente alguns dentre os vários que existem. Buscou-se como critério de escolha, selecionar os mais significativos ou aqueles mais utilizados no Instituto de Física de São Carlos. A Tabela 4 procura resumi-los e compará-los tomando como parâmetros o tipo de hardware e Sistema Operacional que opera, o tipo de distribuição, se apresenta suporte para vários usuários, o formato da base de dados e algumas características interessantes.

Tabela 4 - Comparação dos softwares.

Software	Arquitetura	Distribuição	Multiusuário	Formato da Base de Dados	Características Interessantes
O	<ul style="list-style-type: none"> • PC (Linux, Windows NT) • SGI IRIS 4D • Evans & Sutherland ESV workstations 	Licenciado	Não	<ul style="list-style-type: none"> • PDB aliado a uma estrutura própria 	<ul style="list-style-type: none"> • Suporte para estereoscopia
Swisspdb Viewer	<ul style="list-style-type: none"> • SGI • PC (Linux) • Macintosh 	Freeware	Não	<ul style="list-style-type: none"> • PDB 	<ul style="list-style-type: none"> • Análise de várias proteínas ao mesmo tempo.
Rasmol	<ul style="list-style-type: none"> • PC (Windows) • Macintosh • VMS • SGI 	Freeware	Não	<ul style="list-style-type: none"> • PDB e várias outras. 	<ul style="list-style-type: none"> • Grande variedade de formas de exibição.
WebMol	<ul style="list-style-type: none"> • PC (Windows, Linux) • Macintosh <p>Obs. (Acesso via internet)</p>	Licença Acadêmica	Web	<ul style="list-style-type: none"> • PDB 	<ul style="list-style-type: none"> • Implementado em Java. • Roda tanto como <i>applet</i> ou como <i>standalone application</i>.

5. RECURSOS UTILIZADOS

Para o desenvolvimento do trabalho, é necessário o conhecimento dos recursos que são apresentados nesse capítulo: a base de dados PDB e a linguagem de programação Java por meio das suas APIs Java 3D e Java RMI.

A API Java 3D foi escolhida para o desenvolvimento do suporte para a renderização das moléculas bem como para a implementação dos dispositivos de interação não convencionais: *joystick* e luva. Mais detalhes do Java 3D podem ser encontrados no Apêndice A.

A API Java RMI, por sua vez, foi utilizada para o desenvolvimento do protocolo de comunicação. A razão de utilização dessas duas APIs foi a tentativa de manter a uniformidade do paradigma e da linguagem de desenvolvimento. Além de se tentar observar sua viabilidade para a implementação de AVDs.

A base de dados PDB foi utilizada como formato de entrada dos dados no ambiente. É uma base muito utilizada tanto por pesquisadores quanto por outros softwares.

5.1. Java 3D

Java 3D é um conjunto de classes desenvolvido pela *Sun Microsystems* (SUN, 2002) com o objetivo de ser uma interface para o desenvolvimento e apresentação de programas em Java com conteúdo tridimensional, otimizando renderizações.

Java 3D segue o paradigma de escreva uma vez e rode em qualquer lugar (*Write once, run anywhere*), fornecendo suporte para várias plataformas, vários dispositivos de visualização e vários dispositivos de entrada.

5.1.1. Vantagens e desvantagens

Como principais vantagens de Java 3D pode-se citar (SELMAN, 2002):

- **Implementação somente em Java:** é interessante desenvolver o código da aplicação, a persistência e a interface do usuário em somente uma linguagem, que ainda oferece facilidades para a portabilidade.
- **Suporte para aplicações distribuídas:** A linguagem Java é sinônimo de desenvolvimento para Internet. *Applets* e *servlets* podem ser desenvolvidos com facilidade em Java. Além disso, um suporte de comunicação pode ser alcançado com Java RMI, fornecendo independência de plataforma e níveis de segurança.
- **Descrição da cena:** O modelo de grafo de cena facilita o entendimento, o reuso, o desenvolvimento e a manutenção da aplicação, onde objetos são representados e controlados por nós.
- **Suporte para dispositivos não convencionais:** Java 3D oferece suporte para dispositivos não convencionais como Capacetes e projetores para Cavernas Digitais.
- **Abstração dos mecanismos de renderização:** Possui mecanismos para otimização de renderização dos grafos de cena.

Java 3D também oferece algumas desvantagens, dentre as quais pode-se citar (SELMAN, 2002):

- Algumas características oferecidas por linguagens gráficas mais apuradas como *OpenGL* podem não ser alcançadas em Java 3D.
- A abstração dos mecanismos de renderização também pode não ser desejável para desenvolvedores que necessitam de maior controle do grafo de cena.
- O coletor automático de lixo de Java pode depreciar o desempenho da cena.
- A distribuição da aplicação para os usuários pode tornar-se difícil uma vez que é necessário o *download* e a instalação da biblioteca Java 3D. Para usuários inexperientes, este será um grande obstáculo.

5.1.2. Estrutura do Java 3D

Um programa Java 3D cria instâncias de objetos Java 3D que são colocados em uma estrutura hierárquica chamada de grafo de cena (*scene graph*). Esta estrutura segue um modelo que especifica o conteúdo de um universo virtual e como este conteúdo será renderizado.

Em um grafo de cena são definidos a geometria, sons, luzes, localização, orientação e aparência dos objetos java 3D, os quais são organizados de acordo com o modelo pai-filhos de relacionamento.

Como toda estrutura baseada em grafos, o grafo de cena é composto por nós (*nodes*) e arcos (*arcs*). Um nó irá representar um elemento e, um arco irá representar o relacionamento entre os elementos. A Figura 2 apresenta os nós e os arcos presentes em um grafo de cena java 3D.

Os primeiros dois símbolos, *Virtual Universe* e *Locale*, representam objetos de classes específicas. Os outros três símbolos, *Group*, *Leaf* e *Node Component*, indicam subclasses de objetos específicos. Por fim, o último símbolo é usado para representar um objeto de qualquer outra classe.

Figura 2 - Símbolos para representação de objetos no grafo de cena (BOUVIER, 1999).

As linhas em formato de setas indicam os relacionamentos entre os objetos. As linhas contínuas representam um relacionamento pai-filho entre dois objetos e as linhas pontilhadas, por sua vez, indicam uma referência para um objeto.

Em cada grafo de cena existe um objeto *VirtualUniverse*, que tem uma lista dos objetos *Locale*. Normalmente existe um universo por aplicação. Ele é a base sobre a qual o grafo de cena é montado.

Os objetos *locale*, por sua vez, fornecem um ponto de referência em termos de localização dos objetos no universo virtual, ou seja, uma posição no universo na qual irá ser colocado o grafo de cena. Normalmente existe um *locale* por universo.

Um grafo de cena é construído de nós em relacionamentos pai-filho, formando uma árvore, que possui um nó raiz. Outros nós são acessíveis seguindo os arcos a partir da raiz. Um grafo de cena é formado de árvores originadas em um objeto *Locale* e só existe um caminho da raiz de uma árvore a cada uma de suas folhas.

Cada caminho especifica o estado e informações sobre determinada folha, como local, orientação, e tamanho de um objeto visual. Dessa forma, os atributos visuais de cada objeto dependem apenas de seu caminho no grafo. Java 3D tira proveito desse fato e renderiza os nós na ordem que determinar ser mais eficiente. O programador geralmente não tem controle sobre a ordem de renderização dos objetos. Além disso, as representações gráficas de um grafo de cena podem servir como documentação para programas escritos em Java 3D.

A raiz de cada subgrafo do grafo de cena (ou *branch graph*) é representado por um *BranchGroup*. Normalmente existem diversos *branch graphs* por *locale*. Existem duas categorias de subgrafo: *view branch graph* e *content branch graph*. Normalmente existem vários *branches* por *locale*.

O *content branch graph* especifica o conteúdo do universo virtual (geometria, aparência, comportamento, localização, som e iluminação), enquanto o *view branch graph* especifica os parâmetros de visualização tais como localização e direção.

A Figura 3 mostra um exemplo de um grafo de cena. Nele observa-se a presença de um *virtual universe* e um *locale*. Abaixo do *locale* pode-se observar a presença de dois subgrafos.

Figura 3 - Exemplo de grafo de cena.

As principais vantagens na estruturação de ambiente em grafo de cena são as facilidades de identificar e corrigir erros antecipadamente e a possibilidade de buscar otimizações na renderização da cena.

5.2. Java remote method invocation (RMI)

Java Remote Method Invocation (RMI) é uma solução baseada na linguagem Java para possibilitar invocação de objetos remotos por meio da máquina virtual Java. Assim, torna-se possível a distribuição de aplicações escritas na linguagem Java sob o paradigma de orientação a objetos.

Java RMI foi utilizada para fornecer a base para troca de informações entre os participantes. Como principais vantagens, pode-se citar (JAVARMI, 2002):

- **Orientação a objetos:** RMI pode passar tanto objetos completos como argumentos e retornar valores, não apenas tipos de dados predefinidos .

- **Comportamento móvel:** RMI pode mover o comportamento (implementações da classe) tanto do cliente para o servidor quanto do servidor para o cliente.
- **Padrões de projeto:** A passagem de objetos permite o uso de todo o poder da tecnologia de orientação a objetos na computação distribuída. Quando se consegue passar o comportamento, pode-se utilizar a orientação a objetos nas soluções distribuídas.
- **Protegida e segura:** RMI usa mecanismos de segurança Java (*Security Manager*) para proteger os sistemas no *download* de implementações.
- **Fácil de escrever e usar:** RMI torna simples a escrita de servidores remotos e, conseqüentemente, de clientes para acessar estes servidores. Uma vez a escrita sendo fácil, a manutenção também é.
- **Conexão com sistemas legados:** RMI interage com os outros sistemas existentes por meio da *Java Native method Interface* (JNI). Assim, usando RMI/JNI, conexões a servidores já existentes podem ser feitas. Além disso, RMI se conecta as bases de dados relacionais por meio do JDBC (*Java Database Connection*).
- **Escreva uma vez, rode em qualquer lugar:** RMI segue a abordagem Java de "*Write Once, Run Anywhere*", onde qualquer sistema RMI é 100% portátil para qualquer máquina virtual Java.
- **Coleta de lixo distribuída:** RMI usa a característica de coleta de lixo distribuída para coletar objetos servidores remotos que não são mais referenciados por clientes.
- **Computação paralela:** RMI é *multithreaded*, permitindo servidores explorarem as facilidades Java para gerenciamento de *threads* em implementações concorrentes.

O modelo de comunicação utilizado pelo RMI é mostrado na Figura 4. Logicamente, existe uma transparência no envio das mensagens, de forma que o usuário observa um caminho horizontal entre cliente e servidor. Na realidade, a comunicação passa por 4 camadas.

Para solicitar um serviço ao programa servidor, o programa cliente executa uma chamada ao *stub*, que busca a referência remota do objeto procurado na

camada *Remote Reference*. Uma vez encontrada a referência, estabelece-se uma comunicação pela camada de transporte.

Figura 4 - Modelo de Comunicação: Camadas RMI (JAVARMI, 2002).

As chamadas serializadas pelo *stub* cliente são recebidas e desserializadas pelo *skeleton* do servidor. Uma vez recebidas, o programa servidor executa a requisição e envia resposta ao cliente, seguindo o caminho inverso daquilo que foi explicado anteriormente.

Os *stubs* e *skeletons* são criados pelo compilador RMI, chamado *rmic*. As referências remotas, por sua vez, são armazenadas em um serviço de nomes, denominado *rmiregistry*.

5.3. Protein data bank - PDB

Como mencionado, a grande maioria das estruturas molecular é armazenada em uma base de dados denominada PDB.

A primeira versão do PDB foi criada em 1971 no *Brookhaven National Laboratory* (BNL) com a finalidade de armazenar estruturas macromoleculares.

Em Outubro de 1998, o gerenciamento do PDB tornou-se responsabilidade do *Research Collaboratory for Structural Bioinformatics* (RCSB) e foi modernizado tecnologicamente. No seu início contava somente com 7 estruturas armazenadas e com a disseminação do PDB, este número foi aumentando. Em 06 de Maio de 2002 possuía aproximadamente 18000 estruturas armazenadas. (PDB, 2002).

De acordo ainda com dados fornecidos pelo RCBS (PDB, 2002), aproximadamente 82,3% das estruturas foram determinadas por métodos de cristalografia de Raios-X, 15,3% por Ressonância Magnética Nuclear (NMR) e 2,4% por modelagem molecular.

Tabela 5 - Informações armazenadas em um registro PDB.

Conteúdo Geral (Para todos os depósitos)
<i>Source</i> - Especificações tais como espécie, linhagem, etc. ou descrição do método de síntese química.
<i>Sequence</i> - Sequência completa de todos os componentes macromoleculares.
Estrutura química dos cofatores e grupos prosthetic.
Nomes de todos os componentes na estrutura.
Descrição qualitativa das características da estrutura.
Citações da literatura para a estrutura submetida.
Coordenadas Tridimensionais.
Itens Adicionais para estruturas determinadas por raio-X
Fatores de temperatura e ocupância (occupancies) relacionada a cada átomo.
Condições de cristalização
Dados do cristal, incluindo dimensões de unidade celular.
Presença de simetria não-cristalográfica
Informações descrevendo os tipos de métodos usados para coletar os dados de difração incluindo instrumento, tamanho da onda (<i>wavelength</i>), temperatura e programas de processamento.
Dados estatísticos
Informações de refinamento como fator R, limites de resolução e outros.
Fatores de estrutura - h, k, l, Fobs, σ Fobs.
Itens Adicionais para estruturas determinadas por NMR
Número do modelo para cada conjunto de coordenadas depositadas e uma indicação se um deve ser designado como representativo, ou um modelo de energia minimizada média.
Informações descrevendo os tipos de métodos utilizados, instrumentação, intensidade do campo magnético e outros.
Condições da amostra, incluindo taxa de concentração da macromolécula, sais, agentes antibacteriais e outros.
Condições experimentais tais como temperatura, pH, pressão e outros.
Heterogeneidade não covalente da amostra, incluindo auto agregação.
Heterogeneidade química da amostra.
Lista de experimentos RMN utilizados para determinar a estrutura, incluindo aqueles utilizados para determinar transmissão de ressonância, dados NOE/ROE, dados dinâmicos e outros.
Arquivos relacionados utilizados para derivar a estrutura descrita.

A Figura 6 mostra 4 trechos de um registro de uma molécula no formato PDB. O registro é um arquivo do tipo texto onde cada linha indica um campo cujo nome é a primeira palavra. Cada uma das linhas armazena uma informação específica da molécula de proteína depositada.

Pode-se observar as linhas HEADER e TITLE, que indicam o nome da proteína depositada e data de depósito e tipo e procedência da proteína, respectivamente. As linhas REMARK trazem informações como a resolução e restrições ao erro admitido na coleta de dados, entre outros. SEQRES traz a sequência de resíduos que compõem a molécula e ATOM contém as coordenadas

relativas de cada átomo que compõe cada resíduo da proteína. A estrutura completa de um registro PDB é apresentada no apêndice B.

```

HEADER      KINASE                      09-DEC-97   1A06
TITLE       CALMODULIN-DEPENDENT PROTEIN KINASE FROM RAT
...
REMARK      3  DATA USED IN REFINEMENT.
REMARK      3  RESOLUTION RANGE HIGH (ANGSTROMS) : 2.5
REMARK      3  RESOLUTION RANGE LOW  (ANGSTROMS) : 6.0
REMARK      3  DATA CUTOFF          (SIGMA(F)) : 2.0
REMARK      3  DATA CUTOFF HIGH     (ABS(F)) : 1000000000.00
...
SEQRES      1  332  MET PRO GLY ALA VAL GLU GLY PRO ARG TRP LYS GLN ALA
SEQRES      2  332  GLU ASP ILE ARG ASP ILE TYR ASP PHE ARG ASP VAL LEU
SEQRES      3  332  GLY THR GLY ALA PHE SER GLU VAL ILE LEU ALA GLU ASP
SEQRES      4  332  LYS ARG THR GLN LYS LEU VAL ALA ILE LYS CYS ILE ALA
...
ATOM        1  N   TRP    10      109.622  32.123  34.299  1.00 55.67    N
ATOM        2  CA  TRP    10      110.230  30.823  34.018  1.00 59.89    C
ATOM        3  C   TRP    10      111.129  30.378  35.173  1.00 59.38    C
ATOM        4  O   TRP    10      112.244  30.889  35.326  1.00 59.75    O
ATOM        5  CB  TRP    10      111.103  30.865  32.755  1.00 58.85    C
...

```

Figura 6 - Trechos de um registro PDB.

6. VRMol - ANÁLISE E PROJETO

Este capítulo apresenta e discute as principais características relacionadas aos aspectos de análise e projeto de um AVD para visualização e análise de moléculas de proteínas, denominado VRMol.

São apresentados os principais objetivos do projeto VRMol, a arquitetura proposta, a análise orientada a objetos realizada e a estruturação do ambiente em um grafo de cena. É apresentado, também, o protocolo de comunicação desenvolvido e utilizado.

6.1. Objetivos

Após se conhecer algumas das áreas de aplicação da RV, pode-se observar seu grande potencial. É importante destacar que as aplicações que foram discutidas são somente alguns exemplos justamente para se observar tal potencial. A RV é aplicada em outras áreas, não somente àquelas delineadas na Seção 2.3 do Capítulo 2. Nesse trabalho em especial, procura-se aplicar este potencial para visualização e análise de moléculas de proteínas.

A utilização da RV na área de visualização e análise de proteínas tem sido investigada desde o início da década de 70. Vários experimentos têm sido feitos desde essa época, porém, por fatores como falta de equipamentos adequados, tais experimentos tiveram dificuldades em alcançar o êxito. Atualmente com o crescimento do poder computacional e diminuição nos custos de equipamentos, espera-se mais sucesso para a área.

Posto isso, destaca-se, novamente, que o principal objetivo do trabalho é construir um Ambiente Virtual para permitir que pesquisadores geograficamente

dispersos possam, em tempo real, cooperar no estudo de uma determinada molécula de proteína visualizada de forma tridimensional. Para tanto as máquina deverão ter suporte para comunicação (rede de comunicação ou modem) e o(s) arquivo(s) que contém dados sobre as moléculas deverão estar no formato PDB.

É objetivo ainda, pelo fato de existirem usuários remotos com a possibilidade de estarem usando arquitetura diferentes (tanto de hardware quanto de software), que o sistema seja portátil e heterogêneo.

6.2. Arquitetura

No desenvolvimento do VRMol optou-se pelo modelo de comunicação cliente/servidor e pelo modelo de armazenamento banco de dados replicados de mundos homogêneos. A decisão de utilizar-se desses modelos, refere-se à impossibilidade da serialização de objetos Java 3D por Java RMI. As vantagens e desvantagens desses modelos foram discutidas na Seção 3.3 do Capítulo 3.

A Figura 7 mostra a arquitetura proposta para o sistema. Nela observa-se a presença de três módulos: dois do lado cliente e um do lado servidor.

O lado cliente se compõe com os módulos *VRMol* e *VRMolClient*, que se relacionam. O módulo *VRMol* representa uma classe que fornece a interface para a visualização da molécula com Java3D, enquanto o módulo *VRMolClient* realiza e gerencia a troca de mensagens com o módulo servidor, com suporte em Java RMI.

Figura 7 - Arquitetura do VRMol.

Os eventos gerados pelo usuário na interface *VRMol* são capturados pelo módulo *VRMolClient* e transferidos para o módulo servidor representado pela classe *VRMolServer*, que transfere o evento gerado para todos os participantes.

A classe *VRMolServer* gerencia, além da troca de mensagens, a entrada e saída dos usuários do sistema. Toda e qualquer ação realizada pelo usuário passa pelo servidor que reflete a mesma para todos os participantes.

A implementação dessa arquitetura engloba outros componentes e segue o paradigma de programação orientado a objetos. Dessa forma, faz-se necessário para o correto desenvolvimento, a utilização de algum método de análise para sistemas orientados a objetos. A metodologia adotada é a UML (*Unified Modeling Language*) (BOOCH et al. 2000) por ser amplamente usada para análise e projeto de sistemas orientados a objetos.

Com base nessa metodologia foi elaborado um diagrama de classes mostrado na Figura 8. Por questões de facilidade de leitura e compreensão, as classes do diagrama encontram-se sem métodos e atributos especificados.

Pelo diagrama de classes da Figura 8, pode-se observar, além dos componentes básicos da arquitetura, seus inter-relacionamentos com elementos marginais como o suporte para dispositivos não convencionais.

Figura 8 - Diagrama de Classes - VRMol.

Em concordância com a arquitetura mostrada na Figura 7, as principais classes do projeto são: *VRMol*, *VRMolClient* e *VRMolServer*. As classes *VRMolClient* e *VRMolServer* implementam as interfaces *Client* e *Server*,

respectivamente, enquanto a classe *VRMol* busca oferecer suporte aos meios de interação como, mouse, joystick, luvas (*gloves*), óculos estereoscópios (*glasses*) e capacetes (HMD).

Vale ressaltar ainda, que a molécula de proteína é construída com base nas coordenadas fornecidas pelo respectivo registro PDB da molécula. Esse carregamento é feito com base em um *loader* especial (classe *Loader*) para a conversão das coordenadas em um tipo *scene* da biblioteca Java 3D.

Pode-se observar também, que o controle dos usuários e arquivos é feito pela classe *VRMolServer* com o auxílio das classes *FileInfo* e *UserInfo*. Estas classes contêm atributos para tanto.

Figura 9 - Grafo de Cena Genérico - VRMol.

Ainda como parte do projeto, fez-se necessária a elaboração e a análise do grafo de cena, uma vez que o Java 3D renderiza seus ambientes de acordo com essa característica. O grafo de cena do VRMol é mostrado na Figura 9. Por ele

pode-se antecipadamente identificar erros e providenciar correções e otimizações. Nele pode-se observar três ramos: visualização (BG3 - *BranchGroup3*), iluminação (BG2 - *BranchGroup2*) e a cena em si (BG1 - *BranchGroup1*).

O ramo de visualização constrói o suporte para a renderização do ambiente. Os nós de destaque são a plataforma de visualização (*ViewPlatform*), que orienta a visualização e o *Canvas3D* que oferece o suporte gráfico para a renderização.

O ramo de iluminação inclui os nós de iluminação e fundo (*background*) do ambiente. Por fim, o ramo que representa a cena, é criado com base na classe *loader*. A cada arquivo aberto um novo ramo é criado e adicionado ao nó *locale*. Isso é feito para criar controles independentes para cada molécula.

Figura 10 - Ramo do Grafo de Cena Gerado pelo *Loader*.

Nesse ramo pode-se observar que cada átomo da molécula é estruturado dentro de um *TransformGroup* (TG) (em destaque com um retângulo pontilhado). O grafo completo de cada átomo é mostrado na Figura 10.

6.3. Primitivas de comunicação

O suporte de rede do VRMol está implementado em Java RMI. Ele fornece base para todas as trocas de mensagens que, na realidade, ocorrem por ativação de métodos das classes que representam o cliente e o servidor. Esses métodos dividem-se em três tipos: métodos para conexão/desconexão, métodos para troca de mensagens textuais e métodos para manipulação e controle das moléculas. A Tabela 6 apresenta as primitivas.

Os métodos para conexão/desconexão abrangem aqueles responsáveis por estabelecer e terminar as conexões entre os clientes e o servidor. Correspondem as primitivas *login* e *logout* no lado servidor e *ReceiveEnter* e *ReceiveExit* no lado Cliente.

Tabela 6 - Primitivas de comunicação.

Métodos para conexão/desconexão	
Servidor	Cliente
login(String name, Chatter chatter)	receiveEnter(String name, Chatter chatter)
logout(String name)	receiveExit(String name)
Métodos para troca de mensagens textuais	
Servidor	Cliente
chat(String name, String message)	receiveChat(String name, String message)
Métodos para manipulação e controle das moléculas	
Servidor	Cliente
StoreOpenedFile(String file, String textString)	receiveFile(String file, String textString)
requestFile(String name, String filename)	requestFile(String file)
broadTranslation(double Trans[], double Rotat[])	receiveTranslation(double Trans[], double Rotat[])

Os métodos para manipulação e controle das moléculas compreendem as primitivas que tratam do carregamento de arquivos e difusão de interações. As primitivas desse tipo são, do lado servidor: *StoreOpenedFile*, *requestFile* e *broadTranslation*; do lado cliente: *receiveFile*, *requestFile* e *receiveTranslation*.

Por fim os métodos para troca de mensagens textuais são os que transmitem as mensagens textuais entre os integrantes do sistema. São as primitivas *chat* (servidor) e *receiveChat* (cliente).

6.3.1. Métodos para conexão e desconexão

Como mencionado, estes métodos abrangem aqueles responsáveis por estabelecer e terminar as conexões entre os clientes e o servidor. Os métodos reúnem-se em duas operações: Login e Logout.

6.3.1.1. Login

Operação que cuida da inclusão de um novo participante no Ambiente. Quando o módulo servidor recebe uma invocação do método *login*, terá o nome do participante como parâmetro. Tal informação é armazenada como um objeto da classe *UserInfo*. O Servidor então avisa a todos os outros participantes da entrada do novo integrante do grupo.

Individualmente, o novo integrante recebe uma notificação pela qual solicita todos os arquivos que já estavam abertos antes da sua entrada. Tais mensagens são abordadas adiante. A Figura 11 mostra a seqüência de ativações para *login*. Os métodos utilizados são:

- *Init()* : Estimula o módulo VRMolClient a conectar-se com o módulo VRMolServer.
- *login(name)*: Solicita ao módulo servidor, VRMolServer, a inclusão de um novo participante (name). Para cada participante, o servidor cria um objeto *UserInfo* com a finalidade de armazenar todos os participantes.
- *receiveEnter(name)*: Propaga a entrada de um novo participante.
- *add()*: Recebimento da adesão de um novo participante.

Figura 11 - Diagrama de seqüência de mensagens: login.

Caso existam arquivos abertos, o novo cliente é estimulado a receber os mesmos para que não haja inconsistências no conteúdo do ambiente em questão. Os métodos com essa finalidade são:

- *requestFile(filename)*: Quando se é um novo participante e já existe algum arquivo sendo tratado pelos outros participantes, este é requisitado.
- *receiveFile(filename, file)*: Recebimento do arquivo solicitado.
- *loadFile(filename)*: Arquivo é carregado em Java 3D pelo módulo VRMol.

6.3.1.2. Logout

Operação que trata da saída de um participante, eliminando o relativo objeto criado com base na classe *UserInfo* para que mensagens não sejam enviadas para um destino que não existe mais. A Figura 12 mostra a seqüência de mensagens da operação *Logout*, cujos métodos são:

- *disconnect()*: Estimula o módulo VRMolClient a desconectar-se do módulo servidor, VRMolServer

- *logout(name)*: Solicita ao módulo VRMolServer a exclusão do participante. A referência ao objeto *UserInfo* que representava o participante é excluída.

Figura 12 - Diagrama de seqüência de mensagens: logout.

- *receiveExit(name)*: Propaga a saída de um participante.
- *remove()*: Realiza a remoção do participante.

6.3.2. Métodos para manipulação e controle de objetos

Englobam todas as mensagens utilizadas para gerenciar os objetos. Tratam das operações de carregamento de um arquivo e de transferência de coordenadas (translação, rotação e escala). Os métodos são: *StoreOpenedFile*, *requestFile* e *broadTranslation*; do lado servidor e *receiveFile*, *requestFile* e *receiveTranslation*, do lado cliente.

6.3.2.1. Carregamento de um arquivo

Quando algum arquivo é carregado por algum participante, ele é enviado para o servidor que cria um objeto *FileInfo* para salvar sua referência e comunica a todos os outros integrantes da existência de um novo objeto no mundo. Todo integrante recebe uma cópia do arquivo.

Apesar de se saber das desvantagens que este modelo oferece principalmente em termos de consistência, ele foi adotado pelo fato do grafo de cena

do Java 3D não ser serializável e, conseqüentemente, não poder ser enviado usando Java RMI.

Figura 13 - Diagrama de sequência de mensagens: carregamento de um arquivo.

A Figura 13 mostra a sequência de métodos usada para carregar um arquivo. Os métodos usados são:

- *loadFileToServer(nome do arquivo, arquivo)*: Método que ativa o módulo VRMolClient, comunicando o carregamento de um novo arquivo.
- *StoreOpenedFile(nome do arquivo, arquivo)*: Solicita ao módulo VRMolServer o armazenamento do arquivo que foi aberto.
- *requestFile(nome do arquivo)*: Avisa os módulos VRMolClient do carregamento de um novo objeto.
- *requestFile(nome do arquivo, arquivo)*: Os módulos VRMolClient solicitam o carregamento do arquivo.
- *ReceiveFile (nome do arquivo, arquivo)*: Módulo VRMolServer envia o arquivo para os participantes.

- *loadFile(nome do arquivo)*: O arquivo recebido é carregado pela interface Java 3D.

6.3.2.2. Transferência de coordenadas

Esta operação trata do envio das coordenadas do objeto assim que algum evento de modificação, tanto de posicionamento quanto de rotação é executado. A Figura 14 mostra a sequência. Os métodos utilizados são:

- *sendTranslation(Trans[], Rotat[])*: Método que envia os vetores de translação e rotação para o módulo VRMolClient.
- *sendTranslation(Trans[], Rotat[])*: Envia os vetores para o servidor difundir as modificações para os participantes.

Figura 14 - Diagrama de sequência de mensagens: transferência de coordenadas.

- *receiveTranslation(Trans[], Rotat[])*: Estimula os clientes a receberem as modificações provenientes do módulo VRMolServer.
- *TranslateObj(Trans[], Rotat[])*: Transfere os valores de translação e rotação para serem aplicados no objeto.

6.3.3. Métodos para troca de mensagens textuais

Corresponde às mensagens para troca de mensagens via chat textual. Refere-se aos métodos *Chat*, (lado servidor) e *receiveChat* (lado cliente), originando uma operação denominada *Chat*.

6.3.3.1. Chat

É um tipo de operação que tem por objetivo difundir uma mensagem de texto enviada por algum dos participantes. Por se tratar de um único grupo, a mensagem é enviada a todos os integrantes.

A Figura 15 apresenta a sequência dos métodos utilizados. Eles são:

- *sendmessage(mensagem)*: Invoca o módulo cliente, VRMolClient, a enviar uma mensagem textual.
- *chat(mensagem)*: A mensagem é passada ao módulo servidor para que seja difundida a todos os participantes.

Figura 15 - Diagrama de sequência de mensagens: chat.

- *receiveChat(mensagem)*: Clientes recebem mensagem vinda do servidor.
- *display(mensagem)*: Método que exibe a mensagem recebida.

7. VRMol - IMPLEMENTAÇÃO, FUNCIONAMENTO E AVALIAÇÃO

Este capítulo apresenta e discute as principais características relacionadas aos aspectos de implementação, funcionamento e avaliação do VRMol.

Inicialmente é apresentada a implementação, mesclada com detalhes do funcionamento. É apresentado, também, como o VRMol suporta dispositivos especiais (joystick e luvas). Por fim, é feita a avaliação do ambiente sob dois aspectos: impacto da rede e renderização.

7.1. Implementação

Como mencionado no capítulo 6, que trata da análise e projeto do VRMol, as classes *VRMolClient* e *VRMolServer*, implementam as interfaces *Client* e *Server*. Essas interfaces são mostradas nas Figuras 16 e 17, respectivamente.

As interfaces fornecem somente os protótipos dos métodos a serem implementados e utilizados pela aplicação. Cada método pode retornar uma exceção `java.rmi.RemoteException`, que foi suprimida das figuras por questões de apresentação.

As interfaces oferecem flexibilidade de implementação e portabilidade em acessos remotos. Nelas, os métodos são padronizados de forma que sua implementação possa ser feita de acordo com a plataforma cliente a ser utilizada.

Assim, todos os métodos que trocam informações remotas estão definidos nas interfaces. Após sua implementação, as interfaces são compiladas gerando os *stubs* e os *skels* que tem a finalidade de localizar e instanciar um objeto remoto por meio do Java RMI.

```

public interface Client extends java.rmi.Remote
{
    public void receiveEnter(String name, Chatter chatter) throws ...;
    public void receiveExit(String name) throws ...;
    public void receiveChat(String name, String message) throws ...;
    public void receiveWhisper(String name, String message) throws ...;
    public void receiveTranslation(double Trans[], double Rotat[]) throws...;
    public void receiveFile(String file, String textString) throws ...;
    public void requestFile(String file) throws...;
}

```

Figura 16 - Interface Client.

```

public interface Server extends java.rmi.Remote
{
    public void login(String name, Chatter chatter) throws ...;
    public void logout(String name) throws ...;
    public void chat(String name, String message) throws ...;
    public void broadTranslation(double Trans[], double Rotat[]) throws ...;
    public void StoreOpenedFile(String file, String textString) throws ...;
    public void requestFile(String name, String filename) throws ..;
    public Chatter lookup(String name) throws ...;
}

```

Figura 17 - Interface Server.

É interessante observar que os métodos definidos nas interfaces são aqueles que compõem as primitivas de comunicação apresentadas na Tabela 6.

Além das classes VRMolClient e VRMolServer, é interessante destacar a implementação da classe VRMol. Nela é criado o grafo de cena da Figura 9.

7.2. Suporte a dispositivos de entrada

As principais formas de interação com o AV são o mouse e a barra de movimentação. Por meio delas o participante pode atuar na molécula em seis graus de liberdade (6DOF - *Degrees Of Freedom*), efetuando rotações e translações em qualquer sentido dos eixos X, Y e Z.

A finalidade da barra de movimentação é mudar o ponto de vista do participante em relação ao AV como um todo, enquanto o mouse muda o ponto de vista do participante em relação ao objeto.

Além do mouse e da barra de movimentação, existe também a possibilidade do usuário utilizar *joystick* ou luvas como dispositivos de entrada. Para tanto, é necessário a implementação das classes *joystick* e *gloves* com o respectivo suporte para o dispositivo utilizado.

7.2.1. Mouse

Existe a dificuldade de se implementar o mouse como dispositivo de entrada em AVs por que ele retorna somente as coordenadas de maneira bidimensional, ou seja, somente as coordenadas para os eixos X e Y. Desta forma, é necessário algum tipo de conversão para que se possa utilizá-las em ambientes tridimensionais.

Uma solução para este problema foi implementada por Selman (2002) na classe *TornadoMouse*¹, conjugando a movimentação do mouse com o acionamento dos botões. Tal classe é pai para as classes *TornadoMouseRotate*, *TornadoMouseTranslation* e *TornadoMouseScale*, que implementam rotação, translação e escala, respectivamente.

Para se explorar toda potencialidade dessas classes, é necessário sobrescrever os métodos abstratos mostrados na Figura 18. Eles possibilitam o controle sobre os eventos gerados pelas ações do mouse no ambiente.

```
public void onStartDrag( Object target ) { }  
public void onEndDrag( Object target ){ }  
public void onApplyTransform( Object target ) { }  
public void onAdjustTransform( Object target, int xpos, int ypos ) { }  
public void onRotate( Object target, Point3d point3d ) { }  
public void onScale( Object target, Vector3d scale ) { }  
public void onTranslate( Object target, Vector3d vTranslation ) { }
```

Figura 18 - Métodos Abstratos da Classe *TornadoMouse*.

¹ A utilização de mouse 2D em ambientes 3D é amplamente discutida por pesquisadores da Interface entre Homem e Computador (IHC). Várias técnicas foram propostas, entretanto sua discussão foge do escopo dessa Tese.

Destacam-se os métodos *onRotate*, *onTranslate*, *onScale* e *onEndDrag*. A implementação destes é fundamental no VRMol. Nos métodos *onRotate*, *onTranslate*, *onScale* são capturados os valores que serão transmitidos para os participantes assim que o botão do mouse, tanto direito quanto esquerdo, seja liberado gerando o evento *onEndDrag*. A Figura 19 mostra a implementação dos métodos *onEndDrag*, onde os dados são enviados e *onRotate*, onde são capturados e preparados.

```
public void onEndDrag( Object target ) {
    client.sendTranslation(this.Trans, this.Rotat);
}

public void onRotate( Object target, Point3d point3d ) {
    this.Trans[0] = this.Tx;
    this.Trans[1] = this.Ty;
    this.Trans[2] = this.Tz;
    this.Rotat[0] = point3d.x;
    this.Rotat[1] = point3d.y;
    this.Rotat[2] = point3d.z;
}
```

Figura 19 - Implementação dos métodos *onEndDrag* e *onRotate*.

7.2.2. Joystick

No *joystick* existe também o problema da dimensão das coordenadas. Além disso, a quantidade de botões pode ser maior, dependendo do tipo de dispositivo utilizado. Konno (1999) apresenta uma solução para a utilização do *joystick* em AVs implementados em Java 3D.

A Figura 20 apresenta um trecho da implementação da classe joystick. Nele, pode-se observar o controle de interrupção dos botões definidos pelos identificadores BUTTON1, BUTTON2, BUTTON3 e BUTTON4, e os protótipos dos métodos para capturar o posicionamento (*getXPos*, *getYPos* e *getZPos*) e o acionamento de um botão (*getButtons*).

```
public class Joystick {  
    public static final int BUTTON1 = 0x0001;  
    public static final int BUTTON2 = 0x0002;  
    public static final int BUTTON3 = 0x0004;  
    public static final int BUTTON4 = 0x0008;  
  
    static {  
        System.loadLibrary("joystick");  
    }  
  
    private int joyID = 0;  
    public Joystick(int id) {  
        joyID = id;  
    }  
    public native int getNumDevs();  
    public native float getXPos(int id);  
    public native float getYPos(int id);  
    public native float getZPos(int id);  
    public native int getButtons(int id);  
    ...  
}
```

Figura 20 - Trecho da classe joystick.

É interessante notar ainda, que para o perfeito funcionamento desta classe é necessário o auxílio de uma DLL (*Dynamic Link Library*). Isto é feito na linha `System.loadLibrary("joystick")`.

A captura das coordenadas e sua transmissão é implementada no método *pollAndProcessInput* (Figura 21). A transmissão ocorre quando um evento, no caso o acionamento de *BUTTON1*, acontece.

```

public void pollAndProcessInput(Joystick joy, Sensor joySensor, float joyPos[],
float joyRot[]) {
    int buttons = joy.getButtons();

    if ( (buttons & Joystick.BUTTON1) != 0) {
        Trans[0] = joyPos[0];
        Trans[1] = vrmolMain.Ty;
        Trans[2] = vrmolMain.Tz + joyPos[2];
        Rotat[0] = vrmolMain.Rx;
        Rotat[1] = vrmolMain.Ry;
        Rotat[2] = vrmolMain.Rz;

        vrmolMain.client.sendTranslation(this.Trans, this.Rotat);
    }
}

```

Figura 21 -Implementação do método *pollAndProcessInput*.

7.2.2. Luvas

Além do mouse e do *joystick*, foi testada a utilização de luvas como dispositivo de entrada para a movimentação da molécula. O modelo de luva usado foi o *Data Glove 5*, distribuído pela *Fifth Dimension Technologies* (5DT, 2002).

A utilização da luva é diferente dos outros dispositivos, uma vez que não tem manche ou botões. Para efetuar a movimentação no eixo X, foi implementado que o usuário deve primeiramente flexionar todos os dedos e esticar somente o dedo indicador. Após fazer este gesto, a rotação da luva para a direita fará com que a molécula mova-se no sentido positivo do eixo. Rotacionando para a esquerda, a molécula fará o movimento contrário, movendo-se negativamente.

Continuando com o gesto descrito acima, caso o usuário incline a mão para baixo, a molécula se movimentará negativamente no eixo Y e, caso incline a mão para cima, a molécula se movimentará positivamente no eixo Y do ambiente tridimensional.

Para mover a molécula no eixo Z, é preciso flexionar todos os dedos e esticar somente o dedo indicador e o dedo médio. Inclinando a mão para a cima,

movimenta-se negativamente no eixo e, inclinando-se a mão para baixo, move-se positivamente no eixo Z.

O trecho de código onde é feita a implementação desses movimentos é mostrada na Figura 22.

```
//Se estiver apontando o indicador
if ((readBuffer[1] < 0) && (readBuffer[2] > 0) && (readBuffer[3] < 0) &&
(readBuffer[4] < 0) && (readBuffer[5] < 0)) {
    //Move no X
    if((readBuffer[7] < 0) && (readBuffer[7] > -100)) coordenadas[0]++;
    else if ((readBuffer[7] > 0) && (readBuffer[7] < 100)) coordenadas[0]--;
    //Move no Y
    if((readBuffer[6] < 0) && (readBuffer[6] > -100)) coordenadas[1]++;
    else if ((readBuffer[6] > 0) && (readBuffer[6] < 100)) coordenadas[1]--;
}

//Se estiver Apontando o indicador e o médio
if ((readBuffer[1] < 0) && (readBuffer[2] > 0) && (readBuffer[3] > 0) &&
(readBuffer[4] < 0) && (readBuffer[5] < 0)) {
    //Move no Z
    if((readBuffer[6] < 0) && (readBuffer[6] > -100)) coordenadas[2]++;
    else if ((readBuffer[6] > 0) && (readBuffer[6] < 100)) coordenadas[2]--;
}
```

Figura 22 - Movimentação com a luva.

Os movimentos são baseados nos dados colhidos na porta serial onde a luva se encontra conectada. Eles são armazenados em uma variável do tipo vetor, *readBuffer[]*, com 7 posições. O dado armazenado na posição 0 corresponde ao cabeçalho de dados. Da posição 1 até a posição 5, estão os estados do dedo polegar ao mínimo, sucessivamente. Por fim, as posições 6 e 7, tratam da inclinação e rotação da luva, respectivamente.

A relação e descrição dos métodos que implementam as principais funcionalidades da luva estão na Tabela 7.

Tabela 7- Descrição dos métodos da classe *Glove*.

Método	Descrição
<code>public void Disconnect()</code>	Desconecta a Luva da porta serial, liberando-a para outra aplicação e/ou dispositivo
<code>public void setCoordinates(int X, int Y, int Z)</code>	Inicia as coordenadas iniciais da luva.
<code>public int[] getCoordinates()</code>	Retorna um vetor de inteiros de 3 posições, com as coordenadas X, Y, Z, na sequência.
<code>public int[] getGloveData()</code>	Retorna um vetor de inteiros com 7 posições, passando os dados com a posição do dedo, indicador, médio, anelar, mínimo, inclinação e rotação.

7.3. Funcionamento

A Figura 23(a) apresenta a tela inicial do VRMol. Destaca-se a tela de login, onde se fornecendo um nome (apelido) e o IP ou nome da máquina servidora, o usuário consegue conexão com o servidor.

Para cada usuário que se conecta, o módulo servidor cria uma instância da classe *User/Info* (Figura 24). Esta classe auxilia o servidor a difundir as mensagens do sistema para os participantes.

Uma vez conectado, o usuário já tem a sua disposição um chat textual (Figura 23(b)). Sua entrada é comunicada a todos os outros participantes e suas mensagens são difundidas a todos.

Figura 23 - Telas de Login e Principal.

Logo acima da tela, observa-se uma barra que oferece ações de movimentação dos objetos. Todo evento gerado por essa barra é difundido a todos os participantes, de modo que as alterações de posicionamento são refletidas a todos.

```
public class UserInfo {  
    public String name;  
    public Chatter chatter;  
    public UserInfo(String name, Chatter chatter)  
    {  
        this.name = name;  
        this.chatter = chatter;  
    }  
}
```

Figura 24 - Classe *UserInfo*.

As Figura 25 (a) e (b), mostram a interação de dois participantes. Ambos têm a mesma visão da molécula e trocam informações via chat textual. Além disso, tem-se um menu onde se pode acessar informações pertinentes ao arquivo PDB cuja molécula está sendo representada. É interessante notar que os dois participantes estão utilizando sistemas operacionais diferentes. A Figura 25 (a) refere-se ao Windows 2000 e a Figura 25 (b), ao Windows XP.

Figura 25 - Dois Participantes.

Tais informações são acessadas por meio do menu *info*, oferecendo-se duas opções: informações gerais sobre a molécula, correspondendo ao nome da molécula, autor, informações sobre publicações, etc. e os comentários contidos nos campos *remark* de um registro PDB. A Figura 26 mostra uma dessas informações sendo acessadas.

Figura 26 - Informações extraídas do PDB.

Para que não haja conflitos entre as ações dos participantes, um mecanismo de travamento (*lock*) é ativado na seleção da molécula, garantindo que somente um dos participantes vai interagir até que a mesma seja liberada. Todas as ações que este fizer na molécula serão difundidas a todos os participantes. Todas essas ações são auxiliados por uma instância da classe *FileInfo*, criada para cada arquivo aberto (Figura 27).

```
public class FileInfo {
    public String filename;
    public double Rotat[] = new double[3];
    public double Transl[] = new double[3];
    boolean locked = 0;

    public FileInfo(String file, double Rot[], double Trans[]){
        this.filename = file;
        this.Transl[1] = Trans[1];
        this.Transl[2] = Trans[2];
        this.Transl[3] = Trans[3];
        this.Rotat[1] = Rot[1];
        this.Rotat[2] = Rot[2];
        this.Rotat[3] = Rot[3];
    }
}
```

Figura 27 - Classe *FileInfo*.

Na classe *FileInfo* é interessante observar a existência dos campos *Rotat[]*, *Transl[]* e *locked*. Os campos *Rotat[]* e *Transl[]* armazenam as coordenadas de rotação e translação da molécula, possibilitando que, quando um novo participante inicia sua sessão, a molécula seja visualizada sob o mesmo ângulo que os outros participantes já o faziam. A função do campo *locked* é garantir o sincronismo de interação com a molécula. Assim que algum participante inicia alguma interação com a molécula, este campo tem seu valor mudado para 1, proibindo que outro participante interaja até que aquele que outrora estava interagindo termine, tornando o valor desse campo 0.

7.4. Considerações sobre desempenho

Como mencionado na Seção 3.4., o atraso é um dos fatores que mais prejudica o desempenho de um AVD. Existem dois aspectos interessantes para serem discutidos: o atraso da rede de comunicação e o atraso de sintetização.

7.4.1. Atraso da rede de comunicação

É interessante destacar que o impacto das primitivas de comunicação é mínimo perante o protocolo utilizado. Levando-se em consideração os métodos utilizados para troca de mensagens, observa-se que os métodos `ReceiveTranslation(double Trans[], double Rotat[])` e `BroadTranslation(double Trans[], double Rotat[])` são aqueles que podem causar mais impacto em termos de quantidade de informação sendo transmitida em razão do tamanho das informações que transporta.

Na linguagem Java, o tipo de dado *double* ocupa 64 bits² (8 bytes). Assim, o tamanho da mensagem é, no máximo, 128 bits (16 bytes). Segundo Tanenbaum (1996, p.522), o tamanho máximo de um datagrama do protocolo TCP/IP é 64 Kbytes, com a área de dados tendo que ocupar, no mínimo, 46 bytes. Desta forma,

² Em conformidade com a norma *IEEE Standard for Binary Floating Point Arithmetic*, ANSI/IEEE Std. 754-1985 (IEEE, 1985).

os datagramas que trafegam têm sempre o tamanho mínimo, sendo necessário a utilização do campo de preenchimento do cabeçalho TCP/IP para alcançar tal tamanho. Portanto, analisando quanto ao atraso, observa-se que este é dependente do roteamento do protocolo TCP/IP, não sendo possível seu controle pela aplicação.

Em testes realizados em uma rede local, foi simulado a participação de até 5 participantes. Foi incentivada a troca de mensagens entre as máquinas com a movimentação da molécula (o que ativa os métodos acima citados). Utilizando-se o método *System.currentTimeMillis()*, nativo da linguagem Java, foram obtidos os tempos de envio e recebimento. Para todas as máquina o atraso manteve-se em, no máximo, 15 ms.

7.4.2. Atraso de sintetização

Como mencionado, as moléculas são compostas por átomos. Sendo assim, a quantidade de átomos constituintes determina a complexidade da cena. É interessante notar ainda que, segundo o grafo de cena projetado no VRMol, para cada átomo é criado um ramo do grafo mostrado na Figura 10. Desta forma, supõe-se que quanto maior a molécula, maior poderá ser o atraso de sintetização.

Para se verificar essa hipótese e o impacto desse atraso, foram utilizados arquivos planejados, aumentando-se gradativamente a complexidade das moléculas carregadas. Foram gerados arquivos no formato pdb contendo 50, 100, 150, 200, 300, 400, 500, 600 e 700 átomos. Para os testes foram utilizadas duas máquinas separadamente.

A primeira máquina, denominada máquina 1, tem processador Intel celeron com 433 MHZ, placa de vídeo *onboard 64 bits 3D AGP Graphics accelerator* com memória gráfica de 8 MB. A máquina 1 tem, ainda, 196 MB de memória RAM e tem o Windows 98 como sistema operacional.

A segunda máquina, denominada máquina 2, tem processador Pentium III com 733 MHz, placa de vídeo *3dlabs Oxygen GVX1* com memória gráfica de 32 MB, 512 MB de memória RAM e sistema operacional Windows 2000.

A Figura 28 apresenta o gráfico comparativo do desempenho das duas máquinas quanto ao tempo, em milissegundos(ms), gasto para renderizar uma cena. Para obtenção dos resultados, utilizou-se o método *getLastFrameDuration()*, nativo

da API Java 3D, que retorna o tempo gasto para renderização do último quadro. A cada movimentação que se executava com a molécula (translação, por exemplo), media-se o tempo.

Figura 28 - Gráfico comparativo de atraso de sintetização.

Pelos resultados observa-se que, para máquina 1, que é uma máquina inferior, o tempo cresce linearmente tornando impraticável a utilização do AV, uma vez que, para moléculas com 700 átomos tem-se a geração de 2,63 quadros por segundo. Para contornar tal problema, pode-se contar com o evidente desenvolvimento e aumento da capacidade de processamento dos computadores e das placas gráficas.

Isso pode ser confirmado pelo comportamento da máquina 2. Seu tempo também cresce, porém não tanto quanto o tempo da máquina 1. Para o experimento com a molécula de 700 átomos, tem-se a geração de 33,4 quadros por segundo.

Com a máquina 2 ainda foram testados arquivos preparados com 800, 900 e 1000 átomos e um arquivo pdb real contendo 2344 átomos³. Os resultados estão plotados na Figura 29.

³ Referente a molécula 1A06 - CALMODULIN-DEPENDENT PROTEIN KINASE FROM RAT, obtida por *download* em <http://www.rcsb.org/pdb/>.

Figura 29 - Atraso de sintetização da máquina 2.

Observa-se que o tempo também cresce linearmente, porém de forma bem menos acentuada que a máquina 1. Com 2344 átomos tem-se a relação de aproximadamente 12,3 quadros por segundo.

Com o acréscimo de usuários no AV o tempo não teve grandes variações, uma vez que toda sintetização é executada localmente. Desta forma, não se achou relevante mostrar tais dados.

8. CONCLUSÕES

Aliando as necessidades das áreas correlatas à análise molecular para a descoberta e estudo de novas drogas e também para o estudo de moléculas de proteínas já existentes com as características de RV, observa-se a integração dessas duas áreas.

A RV fornece a possibilidade de uma maior integração entre o especialista e os estudos que estão sendo realizados por meio de uma interface com as características de interação, imersão e envolvimento. Assim, os processos relacionados à análise molecular poderão ser facilitados, possibilitando avanços cada vez mais rápidos e mais precisos.

Ressalta-se, ainda, a utilização das facilidades de distribuição de informações para possibilitar a cooperação de especialistas geograficamente dispersos, em tempo real.

Desta forma, os especialistas que analisam e pesquisam estruturas moleculares, têm à sua disposição um ambiente que permite a visualização molecular 3D com liberdade de movimento. Além disso, a cooperação de especialistas remotos terá importância fundamental no estudo de novas estruturas moleculares, acelerando a troca de informações.

No tocante à educação, vale ressaltar que o AV desenvolvido pode ser utilizado como uma ferramenta de apoio ao ensino. O professor pode preparar as moléculas a serem discutidas e estudadas e utilizar-se do ambiente, inclusive em cursos não presenciais, caracterizando a Educação a Distância.

É importante ainda destacar, que este trabalho mostra mais uma área onde os princípios da RV podem ser aplicados, apresentando assim, o grande potencial dessa área.

8.1. Contribuições

Este trabalho procurou trazer contribuições para duas áreas: ciência da computação, especificamente para RV e para visualização e análise de moléculas de proteínas.

Em visualização e análise de proteínas, contribuiu-se principalmente com o oferecimento de um ambiente com suporte para visualização coletiva com troca de mensagens. O VRMol pode trabalhar em conjunto com os softwares já utilizados pelos pesquisadores uma vez que se utiliza de uma base de dados no mesmo formato, o PDB.

Em RV, contribuiu-se com a proposta e implementação de um AVD, utilizando o paradigma de orientação a objetos e a linguagem Java, tanto para implementação da parte gráfica, com Java 3D, quanto do suporte à troca de informações via rede de comunicação, com Java RMI.

A combinação de Java 3D e Java RMI, apesar de pouco utilizada até o momento, mostrou-se promissora, oferecendo principalmente facilidade de implementação e heterogeneidade. Espera-se que mais AVDs possam ser idealizados e implementados, aumentando a gama de comparações, principalmente em relação a outras combinações como aquelas destacadas na Seção 3.5.

Além disso, o trabalho contribuiu com a possibilidade de utilização de dispositivos de interação não convencionais (*joystick* e luvas), executando em conjunto com Java 3D.

Por fim, vale ressaltar que a combinação Java RMI e Java 3D mostrou-se bastante interessante, uma vez que o desenvolvimento segue o mesmo paradigma dentro de uma mesma linguagem base para programação, apresentando portabilidade, heterogeneidade e baixo custo.

8.2. Resultados Obtidos

Os resultados obtidos refletem-se em termos dos seguintes trabalhos publicados:

RODELLO, I. A.; BONFADINI, M. R.; SEMENTILLE, A. C.; BREGA, J. R. F.; KIRNER, C. **VRMol - Um Ambiente Virtual Colaborativo para Visualização e Análise de Moléculas de Proteínas**. In: V SVR - Simpósio Brasileiro de Realidade Virtual. Resumos... Fortaleza. 2002.

RODELLO, I. A.; SEMENTILLE, A. C.; BREGA, J. R. F.; KIRNER, C. **Um Suporte de Rede em Java Rmi para Ambientes Virtuais Colaborativos Desenvolvidos em Java 3D**. In: V SVR - Simpósio Brasileiro de Realidade Virtual. Resumos... Fortaleza. 2002.

RODELLO, I. A., KUBO, M. M., KIRNER, C., TORI, R. **Análise dos Principais Fatores para o Desenvolvimento de Sistemas Distribuídos de Realidade Virtual**. In: IV SVR - Simpósio Brasileiro de Realidade Virtual. Anais... Florianópolis. pp. 314-325. Out. 2001.

RODELLO, I. A.; KIRNER, C.; BEHR, J.; HILDEBRAND, A. **Integrating VRML Worlds Using Avalon and CORBA**. In: III WRV – Workshop de Realidade Virtual. Anais... pp. 181-190. Gramado. Out. 2000.

RODELLO, I. A.; KIRNER, C. **Um Ambiente Virtual Distribuído para Análise Molecular**. In: II WRV – Workshop de Realidade Virtual. Anais... pp. 12- 23. Marília. 1999.

8.3. Trabalhos Futuros

Este trabalho deverá ser continuado, através de desdobramentos e novas idéias a serem incorporadas. Consta em seguida algumas sugestões para os trabalhos futuros:

- Implementação de outras formas de visualização em adição ao modelo CPK utilizado, como por exemplo, *ribbons* e *sticks*.
- Suporte a colaboração. Pode-se utilizar a base de comunicação já implementada para estender a capacidade do VRMol de modo a suportar

o trabalho cooperativo, com funções que permitam a manipulação da molécula.

- Implementação de simulações com a introdução de comandos que permitam um comportamento dinâmico da molécula.
- Otimização do grafo de cena na tentativa de diminuição do atraso de sintetização.
- Comparações com outros modelos de comunicação como CORBA e *toolkits* proprietários.

9. REFERÊNCIAS BIBLIOGRÁFICAS

_____. **Biomech Motion Analysis System**. 2002. Disponível em: <<http://www.health.uottawa.ca/biomech/csb/software/biomech.htm>>. Acesso em: 08 jan 2003.

5DT. **Fifth Dimension Technologies**. Disponível em <<http://www.5dt.com/>>. Acesso em: 20 nov. 2002

ADAMS, L. **Visualização e Realidade Virtual**. Makron Books. p. 255-259. São Paulo, 1994.

AUKSTAKALNIS, S.; BLATNER, D. **Silicon Mirage: The art and Science of Virtual Reality**. Peatchpit Press. Berkeley. 1992.

BERMAN, H. M.; WESTBROOK, J.; FENG, Z.; GILLILAND, G.; BHAT, T. N.; WEISSIG, H.; SHINDYALOV, I. N.; BOURNE, P. E. **The Protein Data Bank**. Nucleic Acids Research. v. 28, n. 1, p. 235-242, 2000. Disponível em: <<http://nar.oupjournals.org/cgi/content/full/28/1/235>>. Acesso em: 04 dez. 2002.

BOUVIER, D. J. **Getting Started with the Java 3D API**. Sun Microsystems, Inc. Disponível em: <<http://java.sun.com/products/java-media/3d/collateral/>>. Acesso em: 1999.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I.; **UML – Guia do usuário**. Editora Campus. 2000.

BRANDEN, C.; TOOZE, J. **Introduction to Protein Structure**. Garland Publishing Inc. 1991.

BULATOV, V. L.; GRIMES, R. W. **Visualization of Molecular Dynamics Simulations**. Eurographics UK Chapter. 1996.

CARNEIRO, T. ; ÁRABE, J.N. **Load Balancing for Distributed Virtual Reality Systems**. In: SIBGRAPI'98 - International Symposium on Computer Graphics, Image Processing and Vision. Proceedings... Rio de Janeiro. 1998.

CASIDAY, R.; FREY, R. **Iron Use and Storage in the Body: Ferritin and Molecular Representations**. Washington University. 2000. Disponível em: <<http://wunmr.wustl.edu/EduDev/LabTutorials/Ferritin/Ferritin.html>>. Acesso em: 11 nov. 2002.

CORBA. **Common Object Broker Architecture**. Disponível em: <<http://www.corba.org>>. Acesso em: 2001.

CHIMERA. **Molecular Representations**. Disponível em: <<http://www.cgl.ucsf.edu/chimera/docs/UsersGuide/representation.html>>. Acesso em: 13 jan. 2003.

CRUZ-NEIRA, C.; SANDIN, D. J.; DEFANTI, T. A. **Surround-screen projection-based Virtual Reality: The Design and Implementation of the CAVE**. In: ACM SIGGRAPH'93. Proceedings... Anaheim. Jul. 1993.

DERIGGI JR. F. V. **Suporte de Comunicação para Sistemas Distribuídos de Realidade Virtual**. Dissertação (Mestrado em Ciência da Computação). PPG-CC - Programa de Pós Graduação em Ciência da Computação. UFSCar - Universidade Federal de São Carlos. São Carlos. SP. Brasil. Ago. 1998.

DERIGGI JR., F., KUBO, M.M., SEMENTILLE, A.C., SANTOS, S.G., BREGA, J.R.F., KIRNER, C. **CORBA Platform as Support for Distributed Virtual Environment**. In: IEEE Virtual Reality. Proceedings... Texas. Mar. 1999.

DIZERÓ, W. J., KIRNER, C. **Professor Virtual: A Realidade Virtual como Suporte ao Ensino de Informática a Distância**. In: XVIII Simpósio Brasileiro de Computação. Anais... Belo Horizonte. MG. Brasil. 1998.

DOSE, K. **Biochemistry**. EPU—Publisher of the University of São Paulo. Springer. 1982.

DRUMMOND, R.; COSTA, R. M. E. M.; CARVALHO, L. A. **Ambientes Virtuais para a estimulação cognitiva de autistas**. In: IV SVR - SBC Symposium on Virtual Reality. Anais... p. 219-225. Florianópolis. SC. Brasil. Out. 2001.

EDWARDS, T. **Virtual Reality and Education**. Disponível em <<http://www.mindspring.com/~rigole/vr.htm>>. Acesso em 1996.

FISHWICK, P.A. **Simulation Model Design and Execution: Building Digital Worlds**. Prentice-Hall. Englewood Cliffs. 1995.

GOSSWEILER, R. et al. **An Introductory Tutorial for Developing Multiuser Virtual Environments**. Presence. Vol. 3. Nro. 4. pp. 255-264, 1994.

GREENHALGH, C. **Approaches to Distributed Virtual Reality Systems**. Relatório Técnico (NOTTCS-TR-96-5); Universidade de Nottingham, 1996(1).

GREENHALGH, C. **Supporting Complexity in Distributed Virtual Reality Systems**. Relatório Técnico (NOTTCS-TR-96-6); Universidade de Nottingham, 1996(2).

GREENHALGH, C., BENFORD, S. **MASSIVE: A Collaborate Virtual Environment for Teleconferencing**. ACM Transactions on Computer-Human Interaction. V. 2. Nro. 3. pp. 239-261. Set. 1995.

HANCOCK, D. **Viewpoint: Virtual Reality in Search of Middle Ground**. IEEE Spectrum. v. 32. n. 1. 1995.

HAGSAND, O. **Interactive Multiuser Vessels in the DIVE System**. IEEE Multimedia. Spring. v. 3. n. 1. p. 30-39. 1996.

IEEE. **Bus Architecture & Microprocessors / Microcomputers Standards Subscription**. ANSI/IEEE Std. 754-1985 1985. Disponível em <<http://standards.ieee.org/catalog/olis/busarch.html>>. Acesso em 16 Dez. 2002.

JACOB, L. J.; NEDEL, L. P.; FREITAS, C. M. Da S. **Avaliação das técnicas de interação em ambientes imersivos: uma proposta de aplicação no tratamento de fobia de altura**. In: IV SVR - SBC Symposium on Virtual Reality. Anais... pp. 12-22. Florianópolis. SC. Brasil. Out. 2001.

JAVARMI. **Getting Started using RMI**. Disponível em <<http://java.sun.com/j2se/1.4/docs/guide/rmi/getstart.doc.html>>. Acesso em 17 Mai. 2002.

JONES, T. A.; KJELDGAARD, M. **The Entire O Version 5.10 manual**. Disponível em <http://www.imsb.au.dk/~mok/o/o_man/manual.html>. Acesso em 28 Nov. 2002.

KIRNER T.G., KIRNER C., KAWAMOTO A.L.S., CANTÃO J., PINTO A., WAZLAWICK R.S., **Development of a Collaborative Virtual Environment for Educational Applications**. In: ACM Web3D 2001 Symposium. Proceedings... pp. 61-68. Fev. 2001.

KIRNER et al.. **Sistemas de Realidade Virtual**. Apostila do I Ciclo de Palestras de Realidade Virtual; UFSCar - Universidade Federal de São Carlos; 54 p. 1996. Disponível em <<http://www.realidadevirtual.com.br/publicacoes/publicacoes.htm>>. Acesso em 08 Jan. 2003.

KRUEGER, W.; FROEHLICH, B. **The Responsive Workbench**. IEEE Computer Graphics and Applications. Mai. 1994.

KUBO, M.M. **Suporte de Comunicação para Sistemas Colaborativos Interoperáveis de Realidade Virtual**. Dissertação (Mestrado em Ciência da Computação). PPG-CC - Programa de Pós Graduação em Ciência da Computação. UFSCar - Universidade Federal de São Carlos. São Carlos. SP. Brasil. 2000.

KUBO, M. M., VICENTIN, V. J., DERIGGI JR., F. V., KIRNER, C. **Educação e Treinamento a Distância Baseada na Tecnologia de Realidade Virtual**. In: V Workshop de Informática na Educação - WIE'99/SBC'99. Anais... Rio de Janeiro. RJ. Brasil. pp. 669-680. 1999.

LANDER, E. S.; LANGRIDGE, R.; SACCOCIO, D. M. **Computing in Molecular Biology: Mapping and Interpreting Biological Information**. IEEE Computer. Nov. 1991.

MACEDONIA, M. R., ZYDA, M. J. **A Taxonomy for Networked Virtual Environments**. IEEE Multimedia. Vol. 4. Nro. 1. pp. 48-56. 1997.

MACHADO, L. dos S.; ZUFFO, M. K.; MORAES, R. M.; LOPES, R. D. **Modelagem tátil, visualização estereoscópica e aspectos de avaliação em um simulador de coleta de medula óssea**. In: IV SVR - SBC Symposium on Virtual Reality. Anais... pp. 23-31. Florianópolis. SC. Brasil. Out. 2001.

MDL. **MDL Information Systems, Inc.** Disponível em <<http://www.mdli.com/>>. Acesso em 04 Dez. 2002.

MORIE, J. F. **Inspiring the Future: Merging Mass Communication, Art, Entertainment and Virtual Environments**. Computer Graphics. Vol. 28. Nro. 2. pp. 135-138. 1994.

NETTO, A. V. **Realidade Virtual economiza tempo e dinheiro na cadeia automotiva**. Revista Engenharia Automotiva e Aeroespacial. Ano 2. Nro. 9. pp. 32-37. Editora Market Press. São Paulo. SP. Brasil. Abr. 2002. Disponível em <http://www.icmc.usp.br/~aneto/rev_engauto.htm>. Acesso em 24/10/02.

NETTO, A. V.; MACHADO, L. dos S.; OLIVEIRA, M. C. F. **Realidade Virtual - Definições, Dispositivos e Aplicações**. Revista Eletrônica de Iniciação Científica - REIC. Ano II. Vol. II. Número I. Mar 2002. Disponível em <<http://www.sbc.org.br/reic/edicoes/2002e1/>>. Acesso em: 12 nov. 2002.

OLIVEIRA, J. C.; SHEN X, GEORGANAS N.D. **Collaborative Virtual Environment for Industrial Training and e-Commerce**. In: Workshop on Application of Virtual Reality Technologies for Future Telecommunication Systems. IEEE Globecom'2000 Conference. Proceedings... San Francisco. 2000.

PANEPUCCI, E. **Modelagem Molecular da Enzima Gliceraldeído-3-Fosfato Desidrogenase de T. Cruzi e Análise de Potenciais Inibidores Específicos**. Dissertação (Mestrado em Física). IFSC - Instituto de Física de São Carlos. USP - Universidade de São Paulo. São Carlos. 1994.

PANTELIDES, V. **Reasons to Use Virtual Reality in Education**. VR in schools. Vol. 1. 1995.

PDB. **Protein Data Bank Current Holdings**. Disponível em <<http://www.rcsb.org/pdb/holdings.html#growth>>. Acesso em 06 mai 2002.

PIMENTEL, K.; TEIXEIRA, K. **Virtual Reality - Through the New Looking Glass**. 2^a Ed. McGraw-Hill. 1995.

PINHO, M. S. **Realidade Virtual como Ferramenta de Informática na Educação**. In: VII SBIE Simpósio Brasileiro de Informática na Educação. Anais... Belo Horizonte. MG. Brasil. 1996.

RICHON, A. B. **An Introduction to Molecular Modeling**. Mathematech. 1. 83. 1994. Disponível em <<http://www.netsci.org/Science/Compchem/feature01.html>>. Acesso em ago 2000.

SAYLE, R. **Rasmol: Molecular Graphics Visualisation Tool**. Disponível em <<http://www.bernstein-plus-sons.com/software/rasmol/>>. 2001. Acesso em 03 Dez. 2002.

SELMAN, D. **Java 3D Programming**. Manning Publications Co. 2002.

SEMENTILLE, A. C.; BREGA, J. R. F.; MELO, W.; ABEL Jr., Z.; RODELLO, I. A. **Um interpretador para LIBRAS baseado em um ambiente integrado para o reconhecimento de gestos e posturas**. In: V SVR - Simpósio Brasileiro de Realidade Virtual. Anais... Fortaleza. CE. Brasil. Out. 2002.

SEMENTILLE, A. C. **A Utilização da Arquitetura CORBA na Construção de Ambientes Virtuais Distribuídos**. Tese (Doutorado em Ciências). IFSC - Instituto de Física de São Carlos. USP - Universidade de São Paulo. São Carlos. 186p. 1999.

SENSE8 CORPORATION. **WorldToolKit Reference Manual – Release 7**. Mill Valley. CA. 1998.

SIMA - Support Initiative for Multimedia Applications. **Exploiting Virtual Reality Techniques in Education and Training: Technological Issues** Disponível em <<http://www.man.ac.uk/MVC/SIMA/vrtech/toc.html>>. Acesso em ...1997.

STYTZ, M.R. **Distributed Virtual Environments**. IEEE Computer Graphics & Applications, 16(3); pp. 19-31, Mai. 1996.

SUN. **Sun Microsystems Inc**. Disponível em <<http://www.sun.com>>. Acesso em 13/Mai/2002.

TANENBAUM, A. S. **Computer Networks**. Prentice-Hall. 1996.

TANENBAUM, A. S. **Distributed Operating Systems**. Prentice-Hall. 1995.

VINOSKI, S. **CORBA: Integrating Diverse Applications within Distributed Heterogeneous Environments**. IONA Technologies. 1994.

WATERWORTH, J. A. **Virtual Reality in Medicine: A Survey of the State of the Art**. 1999. Disponível em <<http://www.informatik.umu.se/~jwworth/medpage.html>>. Acesso em 24/10/02.

WALTHER, D. **WebMol - Java PDB Viewer**. Disponível em <<http://www.cmpchem.ucsf.edu/~walther/webmol.html>>. Acesso em 03 Dez. 2002.

WLOKA, M. M. **Lag in Multiprocessor Virtual Reality**. Presence. Vol. 4, Nro.1. pp. 50-63. 1995.

ZUFFO, J. A.; SOARES, L. P.; ZUFFO, M. K.; LOPES, R. D. **Caverna Digital - Sistema de Multiprojeção Estereoscópico baseado em Aglomerados de PCs para Aplicações Imersivas em Realidade Virtual**. In: IV SVR - SBC Symposium on Virtual Reality. Anais... pp. 139-147. Florianópolis. SC. Out. 2001.

ZYDA, M.; SINGHAL, S. **Networked Virtual Environments: Design and Implementation**. Addison Wesley Pub. 1999.

APÊNDICE A - JAVA 3D

Este apêndice apresenta mais informações a respeito da API utilizada para o desenvolvimento do suporte à parte gráfica do trabalho: Java 3D. Esta API foi inicialmente apresentada na seção 5.1.

É apresentada a hierarquia de classes, bem como é gerado o conteúdo de uma cena Java 3D e como interações podem ser adicionadas. Vale ressaltar que a API Java 3D oferece muito mais recursos do que os que estão sendo apresentados. Aqueles recursos que não são relevantes para o trabalho não estão aqui contidos.

1. Hierarquia de Classes

A Figura 30 apresenta a hierarquia de classes da API Java 3D. Além dos itens já discutidos na Seção 3.5 (Virtual Universe e Locale), destacam-se:

- **View:** Descreve os parâmetros e as políticas de visualização. Está ligado a uma classe *ViewPlatform*, que define um ponto de visão dentro de um *locale*. Em conjunto com um nó *Transfom3D*, o *ViewPlatform* define um ponto de referência para o posicionamento e orientação do usuário dentro do mundo virtual.
- **PhysicalBody:** Recebe parâmetros para ajuste do ponto de visão do usuário como por exemplo, localização dos olhos.
- **PhysicalUniverse:** Descreve o ambiente do usuário. Esta descrição serve de suporte para a classe *View*. Descreve dispositivos de visualização (monitores, óculos), sensores de entrada, e dispositivos de áudio, entre outros.

- **Screen3D:** descreve o suporte para o dispositivo de visualização. É sempre ligado a um *Canvas 3D*.
- **Canvas3D:** É derivada da classe *Canvas* do *Abstract Windowing Toolkit* (AWT), nativo do Java. Cria base para exibição do grafo de cena. Pode ser visto como o plano de imagem.

Figura 30 - Hierarquia de classes Java 3D (BOUVIER, 1999).

- **ObjectGraphObject:** Tem duas subclasses: *Node* e *NodeComponent* que compõem o grafo de cena.

- **Node:** É uma superclasse abstrata das classes *Group* e *Leaf*, representando um item do grafo de cena
- **Group:** É uma superclasse usada na especificação da localização e orientação dos objetos visuais no universo virtual. Duas subclasses da classe *group* são *BranchGroup* e *Transform Group*, representados no grafo de cena por BG e TG, respectivamente.
- **Leaf:** É uma superclasse usada na especificação da forma, som e comportamento (*behavior*) dos objetos visuais no universo virtual. Esses objetos não têm filhos.
- **NodeComponent:** É uma superclasse usada na especificação de geometria, aparência, textura e propriedades do material de um nó *shape3D*, ou seja, um conjunto de atributos para os *nodes*. Os *NodeComponents* não são parte do grafo de cena, sendo somente referenciados.
- **Transform3D:** Os objetos dessa classe representam transformações de geometria 3D, como translação, rotação e escala. Geralmente estes objetos são usados em conjunto com um objeto *TransformGroup*.
- **Alpha:** Objetos desta classe são usados para criar uma função variante de tempo. Uma classe *Alpha* produz um valor entre zero e um dependente do tempo e parâmetros do objeto. São comumente usados com objetos interpoladores para fornecer animações de objetos visuais.

Uma vez definidos e apresentados os principais componentes, pode considerar que o desenvolvimento básico de um programa em Java 3D consiste de sete passos (BOUVIER, 1999):

1. Criar um objeto Canvas 3D;
2. Criar um objeto Virtual Universe
3. Criar um objeto locale e relacioná-lo ao objeto Virtual Universe

4. Construir um view branch graph
 - a. Criar um objeto View
 - b. Criar um objeto ViewPlatform
 - c. Criar um objeto PhysicalBody
 - d. Criar um objeto PhysicalEnvironment
 - e. Relacionar os objetos ViewPlatform, PhysicalBody, PhysicalEnvironment ao objeto View.
5. Construir o(s) Content Branch Graph(s)
6. Compilar o(s) Branch Graph(s);
7. Inserir os subgrafos em um objeto locale.

2. Geração do conteúdo

Existe em Java 3D um rico conjunto de características que habilita a construção de conteúdos 3D complexos. Vale ressaltar que não existe um formato próprio para conteúdo em Java 3D. Existem, basicamente, duas formas para a geração desse conteúdo: construção do conteúdo dentro da própria aplicação Java, usando as classes Java 3D para criação de geometrias e usando *loaders* para carregar um conteúdo previamente construído. Classes *File loader* habilitam a leitura de conteúdos a partir de arquivos em outros formatos, tais como:

VRML (*Virtual Reality Modeling Language*)

OBJ (*Alias/Wavefront object*)

LW3D (*Lightwave 3D scene*)

Pode-se ainda, construir um *File loader* próprio para a aplicação que se está desenvolvendo.

2.1. Criação de Geometria

Um nó *leaf Shape3D* constrói um objeto 3D baseado na geometria (*Geometry*) que se refere à forma ou estrutura da forma e, na aparência (*Appearance*), que se refere a coloração, transparência e sombreamento da forma.

Existem 14 diferentes tipos de geometria, agrupados em:

- **Simple geometry:** `PointArray` , `LineArray` , `TriangleArray` e `QuadArray`;
- **Strip geometry:** `LineStripArray` , `TriangleStripArray` e `TriangleFanArray`;
- **Indexed simple geometry:** `IndexedPointArray` , `IndexedLineArray` , `IndexedTriangleArray` e `IndexedQuadArray`;
- **Indexed stripped geometry:** `IndexedLineStripArray` , `IndexedTriangleStripArray` e `IndexedTriangleFanArray`.

3. Capabilities (Permissões)

O grafo de cena construído por um programa Java 3D poderia ser usado diretamente para renderização. Entretanto, a representação não é muito eficiente. A flexibilidade construída em cada objeto de um grafo de cena torna a representação do universo virtual sub-otimizada. Portanto, uma representação mais eficiente é usada para melhorar a performance de renderização.

Java 3D tem uma representação interna para um universo virtual e métodos para fazer a conversão. Há dois jeitos para que o sistema faça a conversão para a representação interna. Um deles é compilar cada grafo. O outro é inserir o grafo em um universo virtual para torná-lo vivo. A compilação de um *branch graph* otimiza-o para uma renderização mais rápida.

Essa conversão acarreta alguns efeitos. Um deles é fixar o valor das transformações e outros objetos no grafo. A menos que especificado, o programa não tem a permissão para alterar os valores dos objetos do grafo depois que estes forem tornados vivos.

Há casos quando um programa precisa alterar valores de um objeto, depois que este se torna vivo, como por exemplo, para alterar o valor de um objeto *TransformGroup*, quando tem-se animações. Para isso acontecer, a transformação deve ser capaz de mudar depois que o objeto esteja vivo.

Para tanto, cada *SceneGraphObject* tem um conjunto de bits de permissão (*capabilities*). Os valores desses bits determinam quais permissões existem para o objeto depois que este é compilado ou se torna vivo. O conjunto de permissões varia por classe.

Tais permissões controlam o acesso aos atributos de um nó após já ter se tornado vivo ou sido compilado. É interessante observar que, quanto menos permissões forem concedidas, mais otimizações poderão ser feitas.

Vale ressaltar ainda, que cada nó tem suas próprias permissões para leitura e escrita, que podem ser herdadas de sua classe pai.

4. Interações

Em Java 3D, interações e animações são determinadas com objetos *Behavior*. A classe *Behavior* é uma classe abstrata que fornece mecanismos para mudar o grafo de cena.

O propósito de um objeto *Behavior* em um grafo de cena é alterá-lo, ou alterar seus objetos, em resposta a alguns estímulos. Um estímulo pode ser o pressionar de uma tecla, um movimento de mouse, a colisão de objetos, a passagem do tempo, algum outro evento, ou uma combinação de todos esses, entre outros. As alterações produzidas podem ser a adição ou a remoção de objetos ao grafo de cena, mudança dos atributos desses objetos, ou rearranjo deles. As possibilidades são limitadas apenas pelas permissões dos objetos do grafo.

Por meio de *behaviors*, Java 3D fornece suporte aos dispositivos convencionais (*mouse*, teclado) e não convencionais (*joysticks*, luvas, entre outros). Java 3D provê acesso a teclado e *mouse* usando a API padrão para suporte a esses dispositivos por meio dos *behaviors* *KeyNavigatorBehavior* e *MouseBehavior*, respectivamente.

Adicionalmente, Java 3D provê acesso a uma variedade de dispositivos de entrada, como *trackers* e *joysticks*. Estes dispositivos possuem entradas contínuas bem definidas. *Joysticks*, por exemplo, produzem dois valores contínuos entre -1.0 e 1.0 que Java 3D armazena internamente como uma matriz de transformação, com um dos valores como sendo a translação do eixo X e outro como sendo a translação do eixo Y.

A Tabela 8 abrange algumas das possibilidades de *Behavior*, as mais simples (um estímulo resulta em uma mudança).

Uma classe *behavior* pode ser ainda personalizada. Para tanto é necessário, no mínimo, a implementação dos métodos *initialization* e *processStimulus*, da classe

abstrata *Behavior*. Essa classe personalizada tem um construtor e pode ter outros métodos.

Tabela 8 - Behaviors classificados por estímulos e objetos de mudança (BOUVIER, 1999).

	Objeto de mudança			
Estímulo (razão de mudança)	TransformGroup (objetos visuais mudam de orientação ou local)	Geometria (objetos visuais mudam de forma ou cor)	Grafo de Cena (adicionar ou remover objetos)	Ponto de Vista (mudança de lugar ou direção do ponto de vista)
Usuário	Interação	Específico da aplicação	Específico da aplicação	Navegação
Colisões	Objetos visuais mudam de orientação ou lugar	Objetos visuais mudam aparência quando colidem	Objetos visuais desaparecem com a colisão	Vista muda com a colisão
Tempo	Animação	Animação	Animação	Animação
Lugar de visão	“Billboard”	Nível de detalhe (LOD)	Específico da aplicação	Específico da aplicação

O *behavior* precisa de uma referência a seu objeto de mudança para poder executar as mudanças. O construtor pode ser usado para isso. Caso contrário, outro método deverá guardar essa informação.

O método *initialization* é executado quando o grafo de cena contendo a classe *behavior* se torna vivo. Esse método é responsável por ajustar o evento inicial e a condição inicial das variáveis que fazem parte da classe.

O método *processStimulus*, por sua vez, é invocado quando o evento de disparo especificado para o *behavior* ocorre, ou seja, é um método responsável por responder ao disparo de evento. Este disparo é implementado pelo método *WakeupOn*. Ele é usado nos métodos *initialize* e *processStimulus* para ativar o disparo para o *behavior*.

São vários os critérios que podem ser utilizados para o disparo. Eles são apresentados na hierarquia de classes da Figura 31.

Vale ressaltar que os mecanismos de implementação de uma classe *Behavior* são simples, entretanto, deve-se saber que uma classe implementada de forma

errada pode degradar a performance do sistema. Duas coisas devem ser evitadas: consumo demasiado de memória e condições de disparo desnecessárias.

Figura 31 - Hierarquia de classes: WakeupCondition (BOUVIER, 1999).

4.1. Picking

É comum em ambientes virtuais, existir mais de um objeto visual presente. Para podermos manipular esses objetos individualmente, utiliza-se de uma técnica chamada *Picking*.

A classe *Picking* é implementada por um *behavior* tipicamente disparado por eventos causados pelos botões do mouse. O usuário posiciona o ponteiro do mouse em cima do objeto visual de sua escolha e pressiona um botão, disparando a "operação" *picking*.

Essa operação consiste no seguinte: um raio é projetado no mundo virtual com origem na posição do ponteiro do mouse. A interseção desse raio com os objetos do mundo virtual é computada. O objeto visual mais perto do plano de imagem que fizer interseção com o raio será o escolhido para interação.

O teste de interseção requer uma grande demanda de processamento. A técnica de *picking* consome grande quantidade de recursos, mais ainda conforme a complexidade da cena.

Como mencionado, Java 3D fornece suporte aos dispositivos convencionais (*mouse*, teclado) e não convencionais (*joysticks*, luvas, entre outros). A Figura 32 apresenta a hierarquia das classes disponibilizada pelo Java 3D para suporte aos dispositivos convencionais.

Figura 32 - Hierarquia de classes: Behavior.

APÊNDICE B - PDB - PROTEIN DATA BANK

Este apêndice apresenta a base de dados utilizada para o armazenamento de registros de macromoléculas denominada PDB. Ela foi inicialmente apresentada na seção 5.3. e aqui são explicadas as principais seções que constituem uma referência e os campos que as compõem.

1. Seções de uma referência

Uma referência é um registro de uma macromolécula e é formada por uma coleção de campos que, por facilidade de apresentação, estão divididos em 12 seções. No total são 49 campos. A Tabela 9 apresenta a divisão em seções com uma breve descrição da seção e a relação dos campos.

2. Ordem dos Registros

Todos os campos contidos em um registro PDB devem aparecer de acordo com uma ordem definida. Existem alguns campos que são obrigatórios que, quando não fornecidos, aparecem como NULL. Os campos classificados como opcionais poderão se tornar obrigatórios quando certas condições específicas vierem a existir. A seguir serão apresentados, em ordem, os campos que compõem um registro PDB.

2.1. HEADER

É formado por: uma classificação da molécula, data de depósito e uma identificação única denominada *IdCode*. Sua existência é obrigatória.

Tabela 9 - Campos PDB separados por seção.

Seção	Descrição	Conjunto de Campos
Title	Descreve o experimento e as macromoléculas presentes na referência.	HEADER, OBSLTE, TITLE, CAVEAT, COMPND, SOURCE, KEYWDS, EXPDTA, AUTHOR, REVDAT, SPRSDE, JRNL
Remark	Apresenta detalhes experimentais, anotações, comentários e outras informações não presentes nas outras seções.	Todos os REMARKs (1, 2, 3, etc)
Primary structure	Contém a sequência de resíduos em cada cadeia da macromolécula.	MODRES, DBREF, SEQADV, SEQRES
Heterogen	Contém a descrição completa dos resíduos não padrão presentes na referência.	HET, HETNAM, HETSYN, FORMUL
Secondary structure	Descreve as hélices, planos e voltas encontradas na proteína e estruturas polipeptídicas.	HELIX, SHEET, TURN
Connectivity	Permite a especificação da existência e localização de ligações disulfide ou outras conectividades químicas.	SSBOND, LINK, HYDBND, SLTBGR, CISPEP
Miscellaneous Features	Descreve características na molécula tais como o site ativo.	SITE
Crystallographic	Descreve a geometria do experimento cristalográfico	CRYST1
Coordinate transformation	Descreve transformações no sistema de coordenadas.	ORIGXn, SCALEn, MTRIXn, TVECT
Coordinate	Especifica a coleção de coordenadas atômicas	MODEL, ATOM, SIGATM, ANISOU, SIGUIJ, TER, HETATM, ENDMDL
Connectivity	Fornece informações sobre conectividade química.	CONNECT
Bookkeeping	Contador e indicação de final de arquivo.	MASTER, END

2.2. OBSLTE

Sua existência é obrigatória em referências removidas. Indica, também, quais novas referências substituíram a referência removida.

2.3. TITLE

Contém um título para o experimento ou análise que está sendo representada na referência. Sua existência é obrigatória.

2.4. CAVEAT

Alerta para a existência de erros na referência, sendo obrigatória sua existência nesse caso.

2.5. COMPND

É um campo obrigatório que descreve o conteúdo macromolecular de uma referência. Para cada componente macromolecular, detalhes relevantes como nome das macromoléculas e sinônimos são especificados.

2.6. SOURCE

Especifica a fonte química ou biológica de cada molécula na referência, descrevendo tanto o nome comum quanto o nome científico. É um campo obrigatório.

2.7. KEYWDS

Contém um conjunto de termos relevantes para a referência, que fornecem um significado simples para a classificação, podendo também ser usados para a geração de arquivos de índices. É um campo obrigatório

2.8. EXPDTA

É um campo obrigatório que apresenta informações sobre a técnica experimental utilizada.

2.9. AUTHOR

Contém o nome das pessoas responsáveis pelo conteúdo da referência. É um campo obrigatório.

2.10. REVDAT

Contém um histórico das modificações feitas na referência desde sua divulgação. Apresenta a data da modificação e uma lista dos registros alterados. É um campo obrigatório.

2.11. SPRSDE

Contém uma lista dos *IdCodes* das referências que se tornaram obsoletas devido a uma determinada referência. É um campo opcional, tornando-se obrigatório quando existir uma substituição da referência.

2.12. JRNL

Contém a primeira citação da literatura que descreve o experimento que resultou no depósito do conjunto de coordenadas. É um campo opcional.

2.13. REMARK 1

Relaciona, da mais recente para mais antiga, importantes publicações relacionadas à estrutura apresentada na referência. Vale ressaltar que não são citações repetidas do campo JRNL. É um campo adicional.

2.14. REMARK 2

É um campo obrigatório que estabelece a mais alta resolução, em *Angstroms*, que foi utilizada na construção do modelo. Contém dois formatos. O primeiro é utilizado para estudos de difrações e o segundo, para outros tipos de experimentos nos quais a resolução não é relevante (NMR e modelos teóricos).

2.15. REMARK 3

Outro campo obrigatório que, por sua vez, apresenta informações de programas de refinamento utilizando estatísticas relacionadas.

2.16. REMARK N

Do 4 ao 999, este campo opcional possibilita anotações livres.

2.17. DBREF

Fornece uma referência cruzada entre as seqüências PDB e as referências correspondentes em outra base de dados. É um campo opcional, que se torna obrigatório para cadeias peptídicas maiores que 10 resíduos e para ácidos nucléicos com referências no NDB (*Nucleic Acid Database*)

2.18. SEQADV

É um campo opcional que identifica conflitos existentes entre a seqüência no registro ATOM do PDB e a seqüência na referência na base de dados indicada no registro DBREF.

2.19. SEQRES

Contém a seqüência de aminoácidos ou ácidos nucléicos dos resíduos em cada cadeia da macromolécula. É um campo opcional que se torna obrigatório caso o registro ATOM exista.

2.20. MODRES

Fornece descrições de modificações feitas nos resíduos de ácidos nucléicos ou proteínas. É opcional.

2.21. HET

Usado para descrever resíduos não padrão tais como inibidores, moléculas solventes e ions, que não sejam nenhum dos aminoácidos padrão ou nenhum dos ácidos nucleicos. Para cada resíduo existe um identificador *HetID*. É opcional caso não exista grupos não padrão.

2.22. HETNAM

Dá o nome químico de um determinado componente representado por um *HetID*. É opcional caso não exista grupos não padrão.

2.23. HETSYN

Fornece sinônimos para o componente correspondente ao nome químico identificado no registro HETNAM. É opcional.

2.24. FORMUL

Apresenta a fórmula química e a carga do resíduo não padrão. É um campo opcional.

2.25. HELIX

Usado para identificar a posição de hélices nas moléculas. As mesmas são nomeadas e numeradas. São marcados, também, os resíduos de início e fim e o tamanho. É opcional.

2.26. SHEET

Usado para identificar a posição de planos nas moléculas. Os mesmos são nomeados e numerados. São marcados, também, os resíduos de início e fim e o tamanho. É opcional.

2.27. TURN

Identifica voltas e outros pequenos *loops* que normalmente conectam outros segmentos de estrutura secundária. É um campo opcional.

2.28. SSBOND

É um campo opcional que identifica cada ligação disulfide na proteína e estrutura peptídica, especificando os dois resíduos envolvidos.

2.29. LINK

Especifica a conectividade entre resíduos não pertencente a estrutura primária. É um campo opcional.

2.30. HYDBND

Campo opcional que especifica ligações de hidrogênio na referência.

2.31. SLTBRG

Especifica as pontes de sal contidas na referência.

2.32. CISPEP

Substitui o uso de registro de notas de rodapé para identificar os peptídeos cis. É opcional.

2.33. SITE

Fornece a identificação de grupos incluindo importantes *sites* na macromolécula. É opcional.

2.34. CRYST1

É um campo obrigatório que descreve os parâmetros da unidade celular, grupo *space* e valor Z. Se a estrutura não foi determinada por meios cristalográficos, uma unidade *cube* é definida.

2.35. ORIGX1 ORIGX2 ORIGX3

Apresenta a transformação das coordenadas ortogonais contidas na referência para as coordenadas submetidas. É obrigatório.

2.36. SCALE1 SCALE2 SCALE3

Outro campo obrigatório, que apresenta a transformação das coordenadas ortogonais contidas na referência para coordenadas fracionais cristalográficas.

2.37. MTRIX1 MTRIX2 MTRIX3

Apresenta transformações representando simetrias não cristalográficas. É um campo opcional.

2.38. TVECT

Apresenta vetor de translação para estruturas covalentemente conectadas. É opcional.

2.39. MODEL

Especifica o número serial do modelo quando múltiplas estruturas são representadas em uma simples referência. É utilizada freqüentemente em casos de estruturas determinadas por NMR. É opcional.

2.40. ATOM

É um campo obrigatório que apresenta as coordenadas atômicas dos resíduos padrão. Apresentam também, fatores de ocupação e temperatura para cada átomo.

2.41. SIGATM

Apresenta o desvio padrão dos parâmetros atômicos representados nos registros ATOM e HETATM. É opcional.

2.42. ANISOU

É um campo opcional que apresenta fatores de temperatura anisotrópica.

2.43. SIGUIJ

Apresenta o desvio padrão dos fatores de temperatura anisotrópica proporcionais a um fator de 10^4 (Angstroms²). É um campo opcional.

2.44. TER

É um campo obrigatório caso o registro ATOM exista. Indica o final de uma lista de registros ATOM / HETATM para uma cadeia.

2.45. HETATM

Apresenta as coordenadas atômicas para os heteroátomos (não padrão). É usado por moléculas de água e átomos presentes no grupo HET.

2.46. ENDMDL

São emparelhados com o registro MODEL para agrupar estruturas individuais encontradas nas coordenadas. É opcional, sendo obrigatório caso o registro MODEL exista.

2.47. CONECT

Especifica a conectividade entre átomos os quais as coordenadas são fornecidas. A conectividade é descrita usando o número serial do átomo. É um registro obrigatório na presença de grupos HET.

2.48. MASTER

É um contador que lista o número de linhas na referência de coordenadas ou arquivo para os tipos de registro selecionados. É obrigatório.

2.49. END

É um campo obrigatório que delimita o final da referência PDB.