UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE CIÊNCIAS EXATAS E DA NATUREZA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

AMBIENTE VIRTUAL DISTRIBUÍDO WEB PARA MÁQUINAS DE MEDIR POR COORDENADAS (AVD-W)

RICARDO ZOTTINO

ORIENTADOR: PROF. DR. NIVALDI CALONEGO JUNIOR.

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, da Faculdade de Ciências Exatas e da Natureza, da Universidade Metodista de Piracicaba – UNIMEP, como requisito para obtenção do Título de Mestre em Ciência da Computação.

PIRACICABA
2006

Zottino, Ricardo

Ambiente virtual distribuído WEB para máquinas de medir por coordenadas AVD-W. Piracicaba, 2006.

152 p.

Orientador: Prof. Dr.Nivaldi Calonego Junior

Dissertação (mestrado) - Programa de Pós-Graduação em Ciência da Computação - Universidade Metodista de Piracicaba

1- Ambientes virtuais distribuídos. 2- Realidade virtual.

3- Sistemas distribuídos. 4- Shout3D. 5- RMI

À

Minha esposa Sonia.

Aos

Meus pais Antonio e Carmen.

AGRADECIMENTOS

Ao professor Dr. Nivaldi Calonego Junior, pela orientação, incentivo e paciência dispensados na formação deste seu orientando durante à elaboração desta dissertação.

À minha esposa Sonia, pela paciência, apoio e seu trabalho de leitura e revisão deste trabalho durante sua elaboração.

Aos meus pais Antonio e Carmen por minha educação a qual sem ela não teria alcançado mais este desafio.

Ao meu irmão Rodrigo e minha irmã Rafaela e a outras pessoas do meu convívio pessoal, não menos importantes, pelo apoio e incentivo.

Ao gerente de TI Rodrigo Meireles pela compreensão da importância deste trabalho na minha formação profissional e pessoal.

“Nada se torna real até ser experimentado,

mesmo um provérbio não significa nada para você

até sua vida poder ilustrá-lo.”

John Keats

(1795 - 1821)

AMBIENTE VIRTUAL DISTRIBUÍDO WEB PARA MÁQUINAS DE MEDIR POR COORDENADAS (AVD-W)

RESUMO

Este trabalho apresenta o desenvolvimento do Ambiente Virtual Distribuido Web (AVD-W) para a Máquina Virtual de Medir por Coordenadas (VCMM) utilizada no ensino da Metrologia. Esse ambiente demonstra através da modularização de suas classes, como ele pode ser reutilizado em outras aplicações que implementam Ambientes Virtuais Distribuídos, tornando-se um *middleware* de comunicação para aplicações que necessitam ser portável para múltiplas plataformas na Web. Desenvolvido na linguagem Java e *Virtual Reality Modeling Language* (VRML), utiliza a biblioteca Shout3D para o controle e renderização de cena VRML, e implementa o protocolo de comunicação VCMM encapsulando *Remote Method Invocation* (RMI) e Socket.

PALAVRAS-CHAVE: Ambientes Virtuais Distribuídos, Realidade Virtual, Sistemas Distribuídos, Java, RMI, Shout3D, VRML e X3D.

VIRTUAL ENVIRONMENT DISTRIBUTED WEB FOR VIRTUAL COORDINATE MACHINE MEASUREMENT (AVD-W)

ABSTRACT

This work presents the development of Virtual Environment Distributed Web (AVD-W) for Virtual Coordinate Measure Machine (VCMM) used in the education of the Metrology. This environment demonstrates through the modules and classes how it can be reused in other applications that implement Distributed Virtual Environments, becoming a middleware of communication for applications that need to be portable for multiple platforms in the Web. Developed in Java Language and Virtual Reality Modeling Language (VRML) and using Shout3D library to control and renderize VRMK scene, implements the protocol of communication VCMM encapsulating Remote Method Invocation (RMI) and Socket.

KEYWORDS: Virtual Distributed Environment, Virtual Reality, Distributed System, Java, RMI, Shout3D, VRML and X3D.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANS	*Advanced Network and Services*
API	*Application Programming Interface*
ARPA	*Advanced Research an Projects Agency*
AVD	Ambiente Virtual Distribuído
AVD-W	Ambiente Virtual Distribuido Web
CAVE	*Cave Automatic Virtual Environment*
CERN	*European Organization for Nuclear Research*
CMM	Máquina de Medir por Coordenadas (*Coordinates Measuring Machine*)
CORBA	*Common Object Request Broker Architecture*
CSCW	*Computer Supported Cooperative Work*
DIS	*Distributed Interactive Simulation*
DIVE	*Distributed Interactive Virtual Environment*
DRI	*Defense Research Internet*
EAI	*External Authoring Interface*
FAPESP	Fundação de Amparo à Pesquisa do Estado de São Paulo
FTP	*File Transfer Protocol*
HMD	*Head Mounted Display*
HTTP	*Hypertext Transfer Protocol*
HTML	*Hypertext Markup Language*
IDE	*Integrated Development Environment*
IDL	*Interface Definition Language*
IP	*Internet Protocol*
ISDN	*Integrated Services Digital Network*
ISO	*International Standards Organization*

JRE	*Java Runtime Environment*
JVM	*Java Virtual Machine*
JTC1	*Joint Technical Committee Information Technology*
LNCC	Laboratório Nacional de Computação Científica
MIOP	*Multicast Inter-ORB Protocol*
NIST	*National Institute of Standards and Technology*
NSF	*National Science Foundation*
OCI	*Open Communication Interface*
ORB	*Object Request Broker*
OSI	*Open Systems Interconnection*
PDF	*Portable Document Format*
PRVD	Protocolo de Realidade Virtual Distribuído
RMI	*Remote Method Invocation*
RNP	Rede Nacional de Pesquisa
RPC	*Remote Procedure Call*
RV	Realidade Virtual
RVD	Realidade Virtual Distribuída
SC24	*Subcommittee of Computer graphics and image* processing
SD	Sistemas Distribuídos
SICS	*Swedish Institute of Computer Science*
SE	*Standard Edition*
SDK	*Software Development Kit*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
UFRJ	Universidade Federal do Rio de Janeiro
URL	Universal Resource Location
UML	*Unified Modeling Language*

VCMM	Máquina Virtual de Medir por Coordenadas (*Virtual Coordinates Measuring Machine*)
VEOS	*Virtual Environment Operation System*
VRML	*Virtual Reality Modeling Language*
WIMP	*Windows, Icon, Menu and Pointer Device.*
WWW	*World Wide Web*
X3D	*Extensible 3D*
XML	*Extensible Markup Language*

1. INTRODUÇÃO

Atualmente os softwares, computadores e suas redes estão cada vez mais presentes em nosso cotidiano, bem como o acesso à Internet que a cada dia incorpora mais em nosso vocabulário o uso de uma palavra que até a poucos anos atrás, fazia parte somente dos domínios da ficção científica, como o Ciberespaço ou Ambiente Virtual (SANTOS, 2001).

Além disso, o desenvolvimento constante de novos equipamentos e programas destinados à simulação, torna possível hoje que usuários tenham a sensação de estar em uma outra realidade, ou seja, em uma realidade virtual (LEVY, 1996). Comumente a palavra virtual é empregada com a intenção de significar a pura e simples ausência da existência da realidade, supondo que realidade seja uma presença tangível, sendo assim o real é o que temos de concreto, enquanto o virtual seria o imaginário ou o que teremos. Para (CADOZ, 1997) este dualismo entre o real e o virtual, caracteriza que o real está entre o que percebemos e o que é, e o virtual entre o que é suscitado em nós e o que está fora de nós.

Mas o que é virtual? A palavra virtual vem do latim medieval *virtualis*, derivado por sua vez de *virtus*, que é força ou potência. Na filosofia Escolástica[1], virtual é o que existe em potência e não em ato. O virtual tende a atualizar-se, sem ter passado pela concretização efetiva ou formal. Por exemplo, uma árvore está virtualmente presente na semente. Em termos filosóficos, "o virtual não se opõe ao real, mas ao atual: virtualidade e atualidade são apenas duas maneiras de ser diferentes" (LEVY, 1996).

O termo Realidade Virtual surgiu em meados da década de 70 através das pesquisas de Myron Krueger com o Videoplace, onde esse sistema capturava

---

[1] Escolástica é um conjunto de doutrinas teológico-filosóficas da Idade Média, caracterizadas sobretudo pelo problema da relação entre fé e a razão (FERREIRA, 2000). Também representou o último período do pensamento cristão, que vai do começo do século IX até o fim do século XVI, designando o nome de escolástica, ou seja, filosofia ensinada nas escolas da época pelos mestres chamada por isso escolásticos.

imagens utilizando uma câmera de vídeo, para criar uma experiência de Realidade Virtual projetada em uma tela (NETTO, 2002; SHERMAN, 2003).

A arquitetura de instalação do Videoplace ilustra na Figura 1 que o participante fica posicionado na frente de um projetor de vídeo e de uma câmera, enquanto atrás dele está uma tela de *backlit*, material que produz um grande contraste na imagem capturada pela câmera, permite o computador distinguir o participante da tela de fundo.

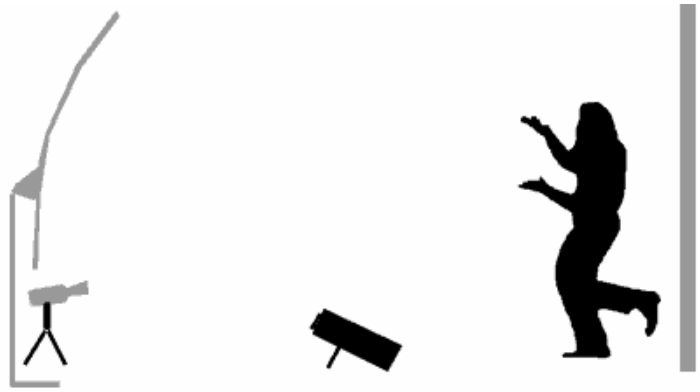


FIGURA 1 – ARQUITETURA DE INSTALAÇÃO DO VIDEOPLACE (PRAVEEN, 2003).

Essa técnica proporciona a integração das imagens captadas pela câmera de vídeo posicionada na frente do participante, com as imagens geradas através das simulações programadas no sistema, de forma a produzir imagens controladas através da movimentação da silhueta do corpo do participante, como a imagem reproduzida nas Figura 2 e 3.

FIGURA 2 – IMAGEM PRODUZIDA PELO VIDEOPLACE (ARTMUSEUM, 2005).

FIGURA 3 – IMAGEM PRODUZIDA PELO VIDEOPLACE (PRAVEEN, 2003).

A manipulação de imagem introduzida pelo Videoplace através da interação do usuário com a cena, permitiu sua caracterização como sendo uma aplicação de Realidade Virtual, como veremos no capítulo 2.

Desde então, segundo Claude Cadoz “a imagem hoje não é mais só esse objeto plano feito unicamente para os olhos; é um espaço no qual podemos incluir objetos que podemos tocar, manipular, ouvir e que resistem ou se animam sob nossas mãos” (CADOZ, 1997).

Segundo Antonio Valerio Netto et.al. a evolução da Realidade Virtual caminha em pesquisas envolvendo as mais diversas áreas, tais como: simulação militar, exploração espacial, médica, cinematográfica, engenharia, arquitetura, educação e entretenimento (NETTO, 2002).

Atualmente, vários autores relatam o uso de aplicações de Realidade Virtual no ensino e treinamento, comprovando a eficiência desta tecnologia para este tipo de finalidade (LOFTIN, 1995; JACOBSON, 2000).

No entanto, a evolução das pesquisas no campo da Realidade Virtual durante a década de 90 propiciaram o surgimento de vários Ambientes Virtuais Distribuídos (AVD) e Colaborativos, que na sua maior parte foi ou está atualmente implementada em C++. Essa opção era devido a limitação dos recursos de hardware e de software proprietários, fornecidos por empresas como: SUN, HP, IBM e Silicon Graphics existentes na época, que restringiam

seu uso devido ao seu alto custo, sendo utilizados somente por grandes empresas e centros de pesquisas.

Mas a evolução tecnológica dos meios de comunicação, processadores, computadores pessoais, dispositivos de interface gráfica e linguagens de programação como Java e VRML. Criaram as condições necessárias para o crescimento e o desenvolvimento da Realidade Virtual Distribuída (RVD) na Internet.

Desta maneira, podemos compartilhar a afirmação de John Vince de que “a Realidade Virtual está destinada a se tornar um importante modo de interação com sistemas de computador” (VINCE, 1992).

1.1. OBJETIVO DO TRABALHO

A decisão para a escolha deste trabalho foi baseada na evolução da Internet, das plataformas de hardware e de software e nas pesquisas utilizando técnicas de Realidade Virtual para a implementação de Máquinas Virtuais de Medir por Coordenadas (CALONEGO, 2004), desenvolvidas para ambiente estereoscópio, objetivando o uso no ensino da Metrologia. No entanto observou-se que o software produzido limitava o número de usuários, devido à necessidade de equipamentos, que apesar de baratos se comparados a outros dispositivos de visão estereoscópica, ainda são caros para muitos usuários.

Essa questão levou-nos a pensar em uma maneira de contribuir para que o uso da Realidade Virtual não imersiva, que utiliza dispositivos básicos dos computadores, possa ser utilizada em atividades cotidianas de estudo e trabalho em grupo, envolvendo pessoas que tenham necessidade de trocar informações baseadas em objetos 3D, utilizando a Web como meio de comunicação.

A solução para esta questão foi o desenvolvimento do protótipo do Ambiente Virtual Distribuido para Web (AVD-W), que implementa a Máquina Virtual de

Medir por Coordenadas (VCMM), que permitiu estudar os principais conceitos que envolvem o campo da Realidade Virtual Distribuída.

A concepção deste trabalho considerou a utilização máxima de recursos Open Source durante a pesquisa e desenvolvimento; a adoção de padrões definidos pela comunidade científica, para que o projeto possa ser reutilizado e aperfeiçoado em trabalhos futuros, possibilitando seu reuso em aplicações de Realidade Virtual Distribuída via Internet.

Para garantir a portabilidade deste programa ao maior número de plataformas atualmente existentes, foi adotado a linguagem de programação Java e como base de comunicação para a plataforma de desenvolvimento *Remote Method Invocation* (RMI) e Socket, comumente utilizada em trabalhos correlatos envolvendo Ambientes Virtuais Distribuídos e apresentado o uso da biblioteca Shout3D (POLEVOI, 2001) como renderizador de cena VRML na implementação deste trabalho.

1.2. METODOLOGIA

O desenvolvimento da pesquisa ocorreu em duas fases. A primeira caracterizada pela busca e leitura de trabalhos correlatos, fundamentando o projeto de software. A segunda fase utiliza a abordagem de desenvolvimento orientado a objetos, para propor uma solução da mimetização do comportamento de uma máquina de medir por coordenadas real.

A validação do modelo proposto foi efetuado com a implementação de um protótipo do software. Esse protótipo usa a técnica de modelagem de cenas para RV denominada “grafo de cena”.

O modelo de programação utilizado na aplicação que permite ao usuário interagir com a cena é a arquitetura cliente/servidor. O componente servidor obtém os comandos de controle de cena a partir da rede, decodifica esses comandos e atua no renderizador para modificar o estado da cena. A parte

cliente da aplicação define a interface de comandos que o usuário pode utilizar para enviar mensagens ao servidor.

A modelagem do protótipo usa a linguagem *Unified Modeling Language* (UML) (BOOCH, 1998) e a aplicação JUDE como ferramenta, para gerar a estrutura de classes para a linguagem de programação Java.

1.3. ESTRUTURA DO TRABALHO

Este trabalho apresenta no segundo capítulo uma revisão da literatura utilizada no desenvolvimento deste protótipo, para levantamento teórico, contextualização histórica e definição das principais tecnologias, como: Internet (CARVALHO, 1998), World Wide Web, Sistemas Distribuídos (COULOURIS, 2001; TANENBAUM, 2002), Realidade Virtual (KIRNER, 2004), VRML (VRML, 2003), X3D (X3D, 2005), Java (SUN, 2003), Shout3D (POLEVOI, 2001). Apresentamos também alguns trabalhos correlatos envolvendo Ambientes Virtuais Distribuídos e aplicações de Realidade Virtual, relatando quais suas principais características e tecnologias utilizadas, que serviram como referência para a implementação deste trabalho.

O terceiro capítulo descreve a elaboração do protótipo que implementa o AVD-W para VCMM, apresentando sua arquitetura, modelo de programação orientado a objetos, plataforma de hardware e software utilizados no decorrer do desenvolvimento do projeto.

O quarto capítulo apresenta os resultados obtidos com a utilização protótipo e considerações finais.

2. REALIDADE VIRTUAL DISTRIBUÍDA

Este capítulo apresenta os principais conceitos de Realidade Virtual Distribuida utilizados durante a especificação e desenvolvimento deste protótipo, resgatando suas origens através da história, compreendendo sua evolução, refletindo sobre seu funcionamento e aplicação no contexto deste trabalho.

Os conceitos apresentados a seguir estão organizados conforme as camadas de software utilizadas pela aplicação conforme ilustra a Figura 4. Os exemplos sobre os conceitos tiveram como base o próprio prótipo visando facilitar a compreenção do trabalho.

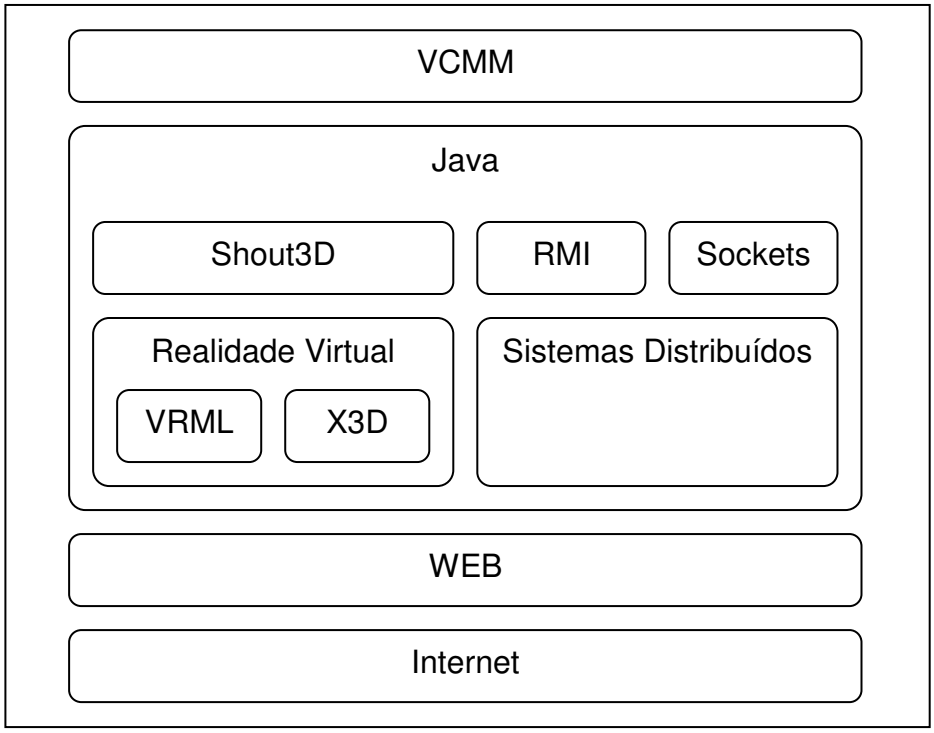


FIGURA 4 – CAMADAS DE SOFTWARE UTILIZADAS PELA APLICAÇÃO.

A camada da Internet descrita na seção 2.1, permite ser utilizada como meio de comunicação para interligar os computadores que integram o AVD-W, que com o uso da linguagem *Hypertext Markup Language* (HTTP) e seu protocolo Web descrito na seção 2.2, possibilitam executar na Web as aplicações desenvolvidas em A*pplets* Java descritas na seção 2.7, e também será

abordado como a linguagem Java pode auxiliar no desenvolvimento e na execução da aplicação VCMM em qualquer plataforma computacional.

O ambiente virtual descrito por cenas em VRML apresentado na seção 2.5, pode ser renderizado e controlado por intermédio da biblioteca Shout3D descrita na seção 2.8. Os fundamentos sobre Sistemas Distribuídos descritos na seção 2.3, possibilita o uso dos protocolos de comunicação RMI e Socket, para controlar e transmitir as mensagens de comandos VCMM entre as aplicações clientes e servidor.

Todas as camadas de softwares e componentes que formam a arquitetura do AVD-W ilustrados na Figura 4, serão discutidos em detalhes neste capítulo.

2.1. INTERNET

A Internet pode ser caracterizada como um conjunto formado por diversos tipos de redes de computadores interligados por todo o mundo, que tem em comum protocolos e serviços nos quais os usuários podem se conectar e utilizar tais serviços de informação e comunicação globalmente (CARVALHO, 1998).

O surgimento da Internet foi através do projeto da agência norte-americana *Advanced Research an Projects Agency* (ARPA), que tinha como objetivo conectar os computadores de seus departamentos. Esta conexão ocorreu em 1969, envolvendo: as Universidades da Califórnia em Los Angeles e Santa Bárbara, a Universidade de Utah e o Instituto de Pesquisa de Stanford. Essa rede passaria a ser conhecida como ARPANET.

Este projeto foi colocado à disposição de pesquisadores, que o estudaram durante a década de 70, resultando na concepção de um conjunto de protocolos denominado *Transmission Control Protocol/Internet Protocol* (TCP/IP), que até hoje é utilizado como a base para comunicação em rede de computadores. No início de 1980 a ARPA começou a integrar a rede ARPANET a redes de computadores de outros centros de pesquisa, e no mesmo período a Universidade da Califórnia de Berkeley implementou os

protocolos TCP/IP para o Sistema Operacional UNIX, possibilitando a sua integração com outras universidades ligadas à ARPANET.

Em 1985 a entidade americana *National Science Foundation* (NSF) interligou os supercomputadores de seus centros de pesquisas, resultando na rede conhecida como NSFNET, que em 1986 foi conectada à ARPANET, formando dois *backbones*[2] interligando todas as redes e computadores destas duas redes, que passou a ser conhecida oficialmente como INTERNET.

Em 1988 a NSFNET passou a ser mantida com o apoio da IBM, MCI Telecomunicações e o Instituto responsável pelas redes de Michigan (MERIT), formando uma associação conhecida como *Advanced Network and Services* (ANS). Nesse mesmo ano a Internet chegava ao Brasil através da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), pela Universidade Federal do Rio de Janeiro (UFRJ) e do Laboratório Nacional de Computação Científica (LNCC). Em 1989, o Ministério da Ciência e Tecnologia criou a Rede Nacional de Pesquisa (RNP) para coordenar e disponibilizar os serviços de acesso no Brasil.

Em 1990 o *backbone* da ARPANET foi desativado dando lugar ao *backbone Defense Research Internet* (DRI). Em 1991 e 1992 a ANS desenvolveu um novo *backbone*, conhecido como ANSNET, que passou a ser o principal *backbone* da Internet. Nessa mesma época, iniciou-se o desenvolvimento do *backbone* europeu chamado EBONE, interligando países da Europa.

A partir de 1993, a Internet deixou de ser uma instituição de natureza apenas acadêmica e passou a ser explorada comercialmente, tanto para a construção de novos *backbones* por empresas privadas quanto para o fornecimento de serviços diversos crescendo mundialmente.

Em abril de 1995 a RNP iniciou o processo de implementação comercial da Internet no Brasil e em 1996 outras empresas começaram a inaugurar seus

---

[2] Segundo (COULOURIS, 2001) backbone é o nome dado as ligações que interconectam os roteadores em uma rede, geralmente utilizando um meio de comunicação rápido e seguro para replicação dos dados.

próprios *backbones* comerciais, ampliando o acesso e a utilização da Internet para os mais diversos tipos de instituições.

A evolução da Internet como meio de comunicação e interligação dos mais diversos tipos de redes de computadores, vem permitindo através de sua utilização que as aplicações possam ser utilizadas além dos limites de uma rede local, desde que tais redes locais possuam uma conexão com a Internet.

Desta maneira, qualquer computador que atenda às configurações mínimas de hardware e software conforme descrito no capítulo 3, e que esteja conectado a uma rede local ou pela Internet, pode ser utilizado para executar a aplicação VCMM.

2.2. WWW (*WORD WIDE WEB*)

A Word Wide Web também conhecida mundialmente como WWW, W3 ou simplesmente Web foi desenvolvida a partir de 1989 pelo *European Organization for Nuclear Research* (CERN) em Genebra na Suíça, com o objetivo de interligar pesquisadores de vários institutos através da Internet.

A Web é um sistema de busca e obtenção de informações baseadas em títulos que descrevem os documentos Web e que podem conter textos, imagens e recursos de multimídia.

Originalmente os documentos web foram desenvolvidos a partir da linguagem HTML, com baseado em diretivas utilizandos o formato ASCII, que nos permitem definir o formato do documento e de suas ligações ou *hyperlink* com outros documentos, que por sua vez podem estar em outros locais da Web.

Um documento é localizado na Web através de um identificador conhecido como *Universal Resource Location* (URL), que identifica o tipo de servidor, o equipamento onde a informação reside e sua localização.

A utilização da Web ocorre através de um *browser* ou navegador Web, que utiliza o protocolo HTTP para transferir as informações contidas em um servidor Web para o *browser*.

Atualmente os *browsers* podem interagir com diversos tipos de documentos, servidores e serviços disponíveis, através da utilização de *plug-in. Plug-in* são aplicações específicas incorporadas ao *browser*, que o possibilitam reconhecer um determinado tipo de documento ou serviço fornecido por um servidor.

Desta maneira, observa-se que a evolução da Internet associada com a evolução da Web através de novos *plug-in*, permitem o desenvolvimento de novas aplicações virtuais distribuidas. Como o *plug-in* Java utilizado por este protótipo que permite a aplicação VCMM integrar ao *browser* novas funcionalidades como interação e manipular cenas através do uso de *applets* Java que serão apresentadas mais adiante neste capítulo.

2.3. SISTEMAS DISTRIBUÍDOS

A evolução da Internet e da Web possibilita que qualquer usuário da rede, acesse um dos serviços disponíveis em qualquer servidor localizado na rede mundial de computadores. Muitas organizações provêm serviços para suas redes locais, Intranet e Internet compartilhando seus recursos de software como arquivos, sistemas e bancos de dados e também recursos de hardware como servidores de arquivos, impressoras e outros dispositivos (COULOURIS, 2001).

Várias definições para Sistemas Distribuídos (SD) podem ser encontradas na literatura, destacando-se: SD são quaisquer componentes de hardware ou de software, localizados em uma rede de computador que se comunicam e coordenam suas atividades e ações, através do envio e recebimento de mensagens (COULOURIS, 2001); para (TANENBAUM, 2002) um sistema distribuído é uma coleção de computadores independentes que para seus usuários aparecem sob a forma de único sistema.

Essas definições possuem aspectos de hardware e de software que estão intrinsecamente relacionados, criando alguns desafios para o desenvolvimento de SD quanto a sua:

- **Heterogeneidade:** capacidade de suportar diferentes tipos de redes, sistemas operacionais, linguagens de computadores e outras diferenças;
- **Portabilidade:** característica que determina quando os componentes podem ser estendidos e reutilizados de várias maneiras permitindo várias interações;
- **Segurança:** capacidade de proteger de forma adequada os recursos compartilhados, mantendo as informações em segredo quando transmitidas através de mensagens pela rede;
- **Escalabilidade:** algoritmo utilizado para acessar os dados compartilhados, que devem evitar o gargalo no acesso quando novos usuários são adicionados ao sistema;
- **Gerenciamento de falha:** capacidade de gerenciar qualquer processo, computador ou rede que possa falhar independente do outro, contudo cada componente precisa ter um tratamento de erro específico caso ocorra alguma falha;
- **Concorrência:** capacidade de atender a vários usuários que podem fazem solicitações concorrentes em alguns recursos da rede. Cada recurso deve ser desenhado para garantir a concorrência do ambiente;
- **Transparência:** característica que permite ocultar os aspectos da distribuição dos componentes para o usuário do sistema.

O modelo de arquitetura de um SD define quais são seus componentes de sistema, como interagem entre si e como eles são alocados em uma rede de computadores. A arquitetura de um sistema e a definição de sua estrutura tem como principal objetivo assegurar que sua utilização seja confiável, disponível, gerenciável e adaptável.

Arquitetura Cliente/Servidor

O modelo básico da arquitetura cliente/servidor em SD é divido em 2 partes: (i) servidor, que é um processo que implementa um serviço específico, como por exemplo, um sistema de arquivo ou banco de dados, em nosso caso o servidor é um gerenciador de mensagens; (ii) cliente, que é um processo que solicita serviços a um servidor através do envio da solicitação e subseqüentemente aguarda a sua resposta. A arquitetura cliente/servidor é a mais comumente utilizada, baseada em um servidor que processa as solicitações de um ou mais clientes (COULOURIS, 2001; TANENBAUM, 2002).

A Figura 5 ilustra o envio das solicitações de serviços dos clientes para o servidor e a resposta das solicitações pelo servidor, ou seja, o envio e o recebimento de mensagens.

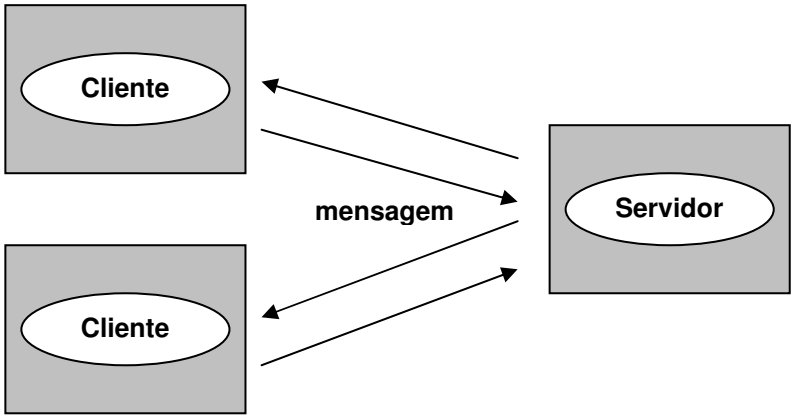


FIGURA 5 – ARQUITETURA CLIENTE SERVIDOR.

A comunicação entre um cliente e um servidor pode ser implementada por meio de um protocolo de comunicação simples quando a rede local é confiável. Nesse caso o cliente solicita um serviço, enviando um pacote de mensagem para o servidor, identificando o serviço requerido, junto com os dados necessários. A mensagem é então enviada para o servidor, que estará sempre aguardando pelo recebimento de novas solicitações, que serão processadas e seus resultados serão retornados para o cliente que as solicitou (TANENBAUM, 2002).

Variações do modelo cliente/servidor podem ser derivadas levando em consideração fatores como:

- Necessidade de computadores com recursos de hardware limitado;
- Dispositivos móveis, como Celulares, Palmtop, Notebooks;
- *Mobile code* ou *Applet*;

Neste protótipo vamos utilizar uma variação do modelo cliente/servidor chamada Web *Applet*, nesse modelo a aplicação cliente é executada a partir de um *browser* que solicita uma página HTML armazenada no servidor Web ilustrado na Figura 6. A página HTML por sua vez contém as diretivas necessárias para a execução da *Applet code*, ou seja, o código escrito na linguagem Java que será apresentado na seção 2.7.

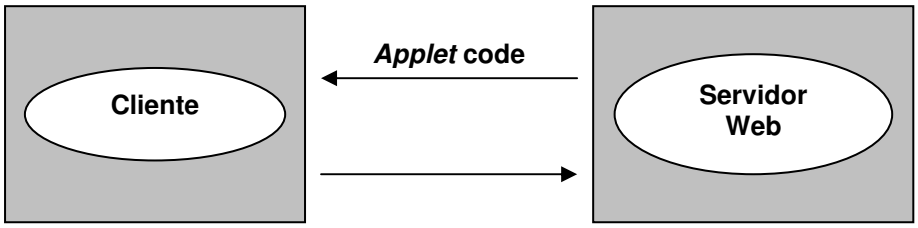


FIGURA 6 – CLIENTE SOLICITA O CÓDIGO DA *APPLET* PARA O SEVIDOR WEB.

Após o servidor localizar a página HTML e transferir todo o código da *Applet* para o cliente, inicia-se a execução local do código *Applet* no *browser* do cliente, sem a necessidade de manter a comunicação estabelecida com o servidor, conforme ilustra a Figura 7.

A vantagem neste tipo de arquitetura é o tempo de resposta, que não sofre atrasos ou variação de banda relacionada com a comunicação da rede, uma vez que a aplicação foi migrada para o cliente. A largura de banda em uma rede de computadores significa a quantidade máxima de informação que pode ser transmitida em um dado tempo, quando uma grande quantidade de recursos utiliza a mesma rede, a largura de banda disponível é compartilhada (COULOURIS, 2001).

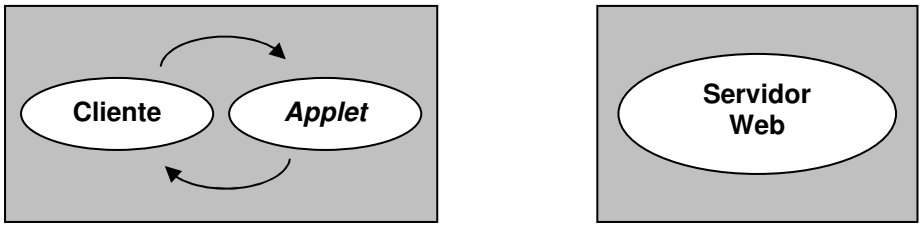


FIGURA 7 – CLIENTE INTERAGE SOMENTE COM O CÓDIGO EXISTENTE NA APPLET.

O termo protocolo é utilizado para referenciar um conjunto de regras e formatos que são utilizados para a comunicação entre processos no sentido de executar uma tarefa específica, ou seja, a especificação da seqüência de mensagens a serem trocadas e a especificação do formato de dados da mensagem (COULOURIS, 2001).

Para que um protocolo de comunicação possa ser implementado, é necessário utilizar um par de programas, localizados no computador que envia e no computador que recebe a mensagem, formando um conjunto de camadas lógicas. Cada camada é responsável por receber os dados da camada lógica acima, encapsular os dados e enviar para a camada abaixo para continuar o processamento da mensagem, conforme ilustra a Figura 8.

O processo inverso é realizado no computador que recebe a mensagem. As cinco camadas apresentadas na Figura 8 fazem parte do protocolo *Open Systems Interconnection* (OSI). O modelo de referência do protocolo OSI foi adotado pela *International Standards Organization* (ISO) para padronizar o desenvolvimento de novos protocolos, conforme Figura 9.

O protocolo *Internet Protocol* (IP) transmite datagramas de um *host* (equipamento ou computador) para outro *host*, se necessário utilizando roteadores, não existindo nenhuma garantia de entrega dos pacotes que podem se perder, serem duplicados ou serem entregues fora de ordem, devido o IP não possuir nenhum tipo de verificador de dados, apenas um verificador para o cabeçalho do pacote conforme Figura 10 (DEITEL, 2001).

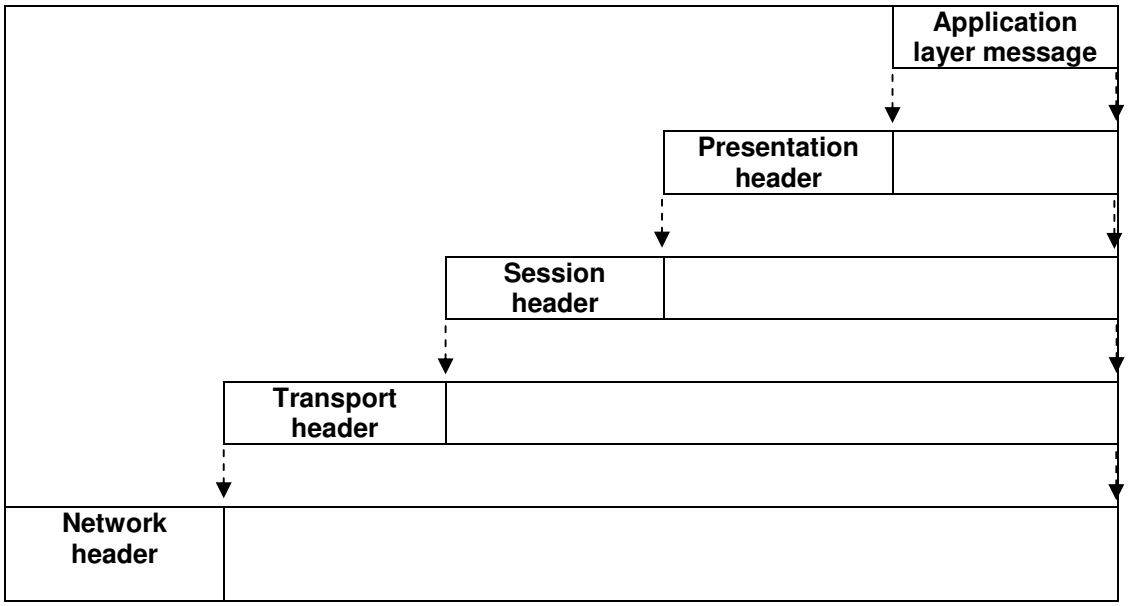


FIGURA 8 – ENCAPSULAMENTO APLICADO NAS CAMADAS DO PROTOCOLO.

O termo datagrama se refere à semelhança do modo de entrega, na qual uma carta ou telegrama são entregues. A característica essencial do datagrama de rede é que a entrega de cada pacote seja feita de uma só vez (COULOURIS, 2001).

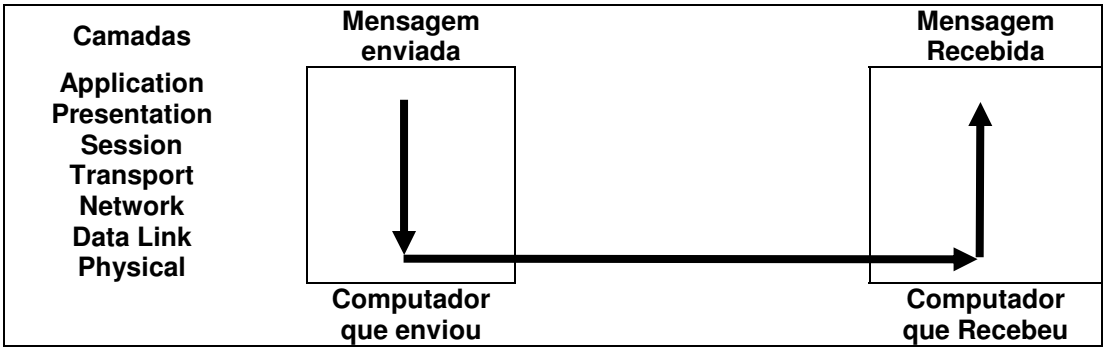


FIGURA 9 – CAMADAS DO MODELO OSI.

O protocolo *Transmission Control Protocol* (TCP) oferece a garantia da entrega confiável dos dados transferidos, enviando e recebendo confirmações de entrega, através de um canal de comunicação bidirecional estabelecido entre os dois pontos, ou seja, entre o computador que envia e o computador que recebe os dados. A camada TCP possui mecanismos que visam garantir a

integridade dos dados como: controle de seqüência, controle de fluxo, retransmissão de pacotes e verificação de erro ou c*hecksum*.

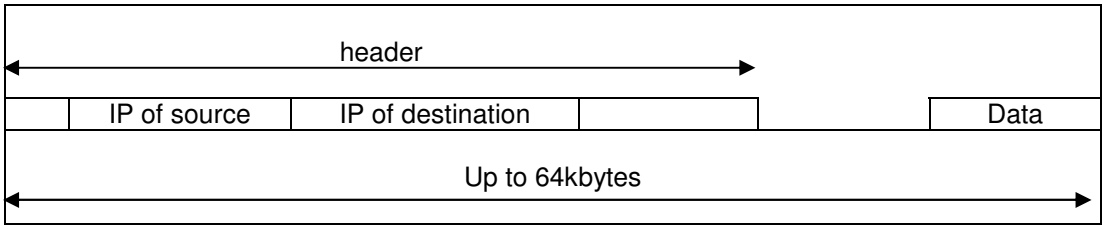


FIGURA 10 – LAYOUT DO PACOTE IP.

O processo de verificação é realizado pela camada Data Link, que é responsável por calcular e inserir no início e final do frame, um bit especial para sua verificação. Quando um pacote é recebido o computador recalcula o bit *checksum* e o compara com o resultado do bit enviado no pacote que sendo igual é considerado correto e aceito, caso contrário o equipamento que recebeu solicita a retransmissão deste pacote.

Ao contrário do protocolo TCP, o protocolo *User Datagram Protocol* (UDP) não oferece nenhuma garantia de entrega confiável dos dados transferidos em seus pacotes, ou seja, ele não garante que os pacotes cheguem ao seu destino na mesma ordem em que foram enviados. Esta é principal desvantagem no uso deste tipo de protocolo, que requer controles adicionais por parte da aplicação para lidar com este problema.

Contudo uma aplicação distribuída que roda na camada de aplicação pode utilizar todos os serviços e protocolos de rede, bem como o sistema operacional do computador para coordenar e executar suas tarefas através da rede. Segundo (FARLEY, 1998) uma aplicação distribuída pode ser divida em quatro partes: processos, *threads*, objetos e agentes, neste protótipo utilizamos somente processos, *threads* e objetos conforme segue:

- **Processos:** são programas que estão sendo executados em um sistema operacional. Normalmente um programa é criado para descrever uma seqüência de passos em uma linguagem de programação, que depois são compilados em um programa executável. Um processo pode ser de uso exclusivo de uma aplicação ou de várias

aplicações e podem usar o mesmo processo para executar uma tarefa (FARLEY, 1998);

- ***Threads*:** são abstrações de atividades para o sistema operacional, este termo deriva frase 'thread of execution', ou seja, fluxo de execução (COULOURIS, 2001). Cada processo possui pelo menos uma *thread* de controle, e alguns sistemas operacionais permitem a criação de múltiplas *threads* controladas por um único processo. Cada *thread* pode ser executada e controlada independentemente de outra *thread*, contudo normalmente existe a necessidade de algum tipo de sincronização entre *threads*. Por exemplo, uma *thread* pode monitorar o recebimento de dados de uma conexão socket, enquanto ao mesmo tempo outra *thread* pode aguardar o envio de um determinado evento do usuário, como o movimento do mouse ou pressionar uma tecla;

- **Objetos:** são normalmente desenvolvidos com a abordagem de orientação a objetos, utilizando a cooperação de seus objetos através da passagem de mensagens. Um processo pode ser desenvolvido utilizando-se vários tipos de objetos, que podem ser acessados por uma ou várias *threads* dentro de um mesmo processo. Com o uso da tecnologia RMI ou *Common Object Request Broker Architecture* (CORBA), um objeto pode também ser distribuído logicamente por múltiplos processos em diversos computadores.

A comunicação entre objetos que estão no mesmo computador ou em computadores separados pode ser realizadas de diversas maneiras, através de chamadas de método remoto como:

O Remote Procedure Call (RPC) desenvolvido nos anos 80 permite a comunicação entre programas remotos, de maneira que um programa execute um processo que reside na memória de outro computador, permitindo desta maneira que partes de um aplicativo se comuniquem diretamente através da rede.

O RPC permite o controle de todas as funções de rede para ordenação dos dados (como *marshalling* e *unmarshalling*), bem como o empacotamento de argumentos e valores de retorno. Algumas de suas desvantagens são o suporte a um número limitado de tipos de dados simples, a falta de suporte para a passagem de objetos Java e a necessidade de descrever os métodos que serão invocados remotamente, utilizando a linguagem de definição de interface *Interface Definition Language* (IDL) (FARLEY, 1998; COULOURIS, 2001).

O Remote Method Invocation (RMI) implementado a partir do modelo RPC permite a comunicação, execução e controle de objetos distribuídos em Java. Suas vantagens são a transferência de objetos e tipos de dados complexos, através da serialização encapsulada no processo de comunicação.

Este encapsulamento é realizado pelo objeto transmissor que converte os dados e objetos de entrada em um fluxo de bytes, que são transmitidos pela rede. Ao chegar ao seu destino, o objeto receptor reconstrói automaticamente os dados e objetos de entrada para utilizá-los. Todo o controle de transmissão de dados sobre a rede, bem como a criação das *interfaces stubs*, *skeletons* são gerados diretamente a partir das classes existentes no programa.

Desde que, exista uma Interface que descreva quais são os métodos que devem ser implementados pelo servidor, e que ambas as classes: servidor e cliente estejam devidamente compilados, para que o processo de compilação, possa criar de maneira análoga ao IDL o *stub* e *skeletons*.

O processo de compilação será detalhado na seção 2.7, e no capítulo 3 descrito a utilização da Interface ParserPRVD_Interface que serão implementadas pelas classes servidoras ServerPRVD e ServerShout. Vale lembrar que o RMI suporta somente a linguagem Java e não requer nenhuma IDL para seu funcionamento em Java. Mas caso seja necessário a comunicação de uma aplicação Java com uma aplicação não Java, pode ser utilizada uma IDL Java introduzida na versão 1.2, que permite sua comunicação com qualquer linguagem que suporte o padrão CORBA (FARLEY, 1998; COULOURIS, 2001; DEITEL, 2001).

A troca de mensagens entre objetos e processos, também pode ser implementada manualmente através da manipulação de pacotes, utilizando sockets diretamente sobre o protocolo TCP/IP.

2.4. REALIDADE VIRTUAL

O termo Realidade Virtual é muito abrangente existindo várias definições na literatura, devido a sua constante evolução. Segundo (HANCOCK, 1994) a RV é a forma mais avançada de interface do usuário com o computador.

Para (LATTA, 1994) a RV é uma interface que permite a interação das pessoas através da simulação de um ambiente real, permitindo às pessoas visualizarem, manipularem e interagirem com representações extremamente complexas. Segundo (KIRNER, 2000) outros autores afirmam que RV é uma técnica avançada de interface relacionada a um ambiente sintético tridimensional gerado por computador, utilizando canais multi-sensoriais; ou como (SHERMAN, 2003) que descreve que a RV é um lugar que existe e que podemos experimentar.

Para (MORIE, 1994) a RV pode ser caracterizada pela coexistência integrada de três elementos básicos: interação, imersão e envolvimento. Para (SHERMAN, 2003) existem quatro elementos principais que permitem ao usuário interagir, sentir e observar o ambiente de RV, sendo o primeiro elemento o próprio ambiente virtual, a imersão, a sensibilidade das respostas do ambiente virtual e a interação. As integrações destes quatro elementos proporcionam o grau de realismo a ser experimentado pelo usuário dentro do ambiente virtual.

Segundo (KIRNER, 2004; WHYTE, 2002) o ambiente virtual pode ser definido e caracterizado através da interação, imersão, envolvimento e navegação no ambiente virtual.

As primeiras experiências com RV ocorreram na década de 50 com a invenção do Cinerama, Cinemascope e do Sensorama. O Sensorama foi criado em 1956

pelo cineasta Morton Heilig e permitia ao usuário ter as sensações da visão tridimensional, som estéreo e vibrações em um passeio simulado, conforme ilustra a Figura 11.

FIGURA 11 – SENSORAMA (ARTMUSEUM, 2005).

As primeiras experiências com controle remoto datam de 1961, quando Comeau e Bryan descrevem o primeiro sistema de circuito fechado de TV montado em um capacete, produzido pela Philco. Sendo que o primeiro capacete com imagens geradas por computador HMD *(Head Mounted Display),* foi projetado em 1968 por Ivan Sutherland na Universidade de Harvard, sendo considerado por muitos o marco da imersão e início da RV, conforme ilustra a Figura 12 (KIRNER, 2004).

As primeiras luvas de dados foram desenvolvidas na Universidade de Ilinois entre 1977 e 1982 por Thomas Zimmerman, que iniciou o projeto do DataGlove junto com San Wantman, que tinha um projeto para construir uma guitarra eletrônica virtual (ZIMMERMAN, 2005). Na década de 80, Jaron Lanier fundou a VPL Research e junto com Thomas Zimmerman e Young Harvill melhoraram o DataGlove conforme apresentado na Figura 14, que passou a ser comercializado junto a outros produtos como capacete de visualização chamado Eye Phones exibido na Figura 15. Mais tarde Jaron Lanier atuaria

como pesquisador chefe da empresa Eyematic Interfaces Inc, responsável por desenvolver a biblioteca Shout3D utilizada neste trabalho (LANIER, 2005).

FIGURA 12 – *HEAD-MOUNTED DISPLAY* (ARTMUSEUM, 2005).

FIGURA 13 – THOMAS ZIMMERMAN E O DATAGLOVE (ZIMMERMAN, 2005).

O uso dessas tecnologias permitem criar o ambiente virtual, também conhecido como mundo virtual. Caracterizado como o conteúdo de um espaço imaginário criado por meio de um ambiente baseado em computadores, onde através da simulação de objetos e conteúdos que podem ser manipulados, proporciona ao usuário experimentar a interação e a imersão neste ambiente através do uso da Realidade Virtual (SHERMAN, 2003).

As luvas permitem a interação do usuário com ambiente virtual, e tal interação constitui um dos aspectos mais importantes da interface da aplicação, estando relacionado diretamente com a capacidade do computador em detectar as ações do usuário no ambiente tridimensional e modificar a cena no mundo virtual. A interação pode se dar também através do uso de:

- **Dispositivos convencionais** são periféricos comumente encontrados na maioria dos computadores como: monitor, teclado e mouse, onde geralmente são utilizados em interações simples, como por exemplo, quando um usuário se movimenta no espaço tridimensional;

- **Dispositivos não convencionais** permitem uma maior interação, devido serem periféricos otimizados para uso específico como: capacetes de visualização e controle, óculos e telas estereoscópicas, sistemas baseados em projeções múltiplas ou *Cave Automatic Virtual Environment* (CAVE), que podem ser utilizados por usuários que requerem uma necessidade maior de manipulação e controle de objetos virtuais utilizando seus sentidos e movimentos naturais do corpo.

FIGURA 14 – DATAGLOVE COMERCIALIZADO PELA VPL (LANIER, 2005).

FIGURA 15 – EYE PHONE COMERCIALIZADO PELA VPL (LANIER, 2005).

A Imersão é o sentimento de se estar dentro do ambiente, este sentimento pode ser simplesmente mental, podendo ser caracterizado pelo estado e grau de engajamento e envolvimento do individuo, ou pode estar acompanhado de um sentimento físico. A imersão física é uma característica da RV, que através do uso de tecnologias que estimulam sinteticamente nossos sentidos, buscam alcançar a máxima imersão mental. Embora a percepção visual seja nosso sentido primário, outros sentidos também devem ser estimulados simultaneamente, a fim de proporcionar uma completa imersão, entre os quais o retorno auditivo, o tato e a força de reação ou *force-feedback*[3]. O uso de dispositivos de *force-feedback* são essenciais para auxiliar o posicionamento do participante de acordo com as condições físicas implementadas no ambiente virtual, onde a utilização de dispositivos não convencionais como luvas, capacetes e até mesmo roupas auxiliam e determinam a posição do participante e sua interação (SHERMAN, 2003).

A Realidade Virtual pode ser caracterizada com Imersiva ou Não-Imersiva:

- **Realidade Virtual Imersiva** se caracteriza-se pelo uso de dispositivos não convencionais, que proporcionam um alto realismo de imersão no ambiente virtual, isolando o usuário por completo do mundo real (WHYTE, 2002; KIRNER, 2004);

- **Realidade Virtual Não Imersiva** se caracteriza-se pela utilização de dispositivos convencionais sendo em alguns casos aceitável a sua utilização, como por exemplo, quando a interação com o ambiente virtual é pequena e restrita a tarefas de exibição e manipulação de objetos (PINHO, 2000). Uma outra característica da RV Não Imersiva é a utilização de interface do tipo *Window, Icon, Menu and Pointer Device* (WIMP) (DAM, 1997).

---

[3] "*Force-feedback*, são equipamentos capazes de gerar forças sobre algumas partes do corpo do usuário com o objetivo de simular a interação com objetos reais. Em geral estes dispositivos são constituídos por braços mecânicos articulados que possuem em suas juntas um ou mais motores que impedem a livre flexão destas juntas, enquanto o usuário movimenta." (KIRNER, 2004).

O envolvimento pode ser caracterizado pelo grau de motivação e engajamento que uma pessoa está determinada a realizar em uma atividade. Podendo este envolvimento ser passivo, como assistir um jogo de futebol pela televisão, ou ativo, como participar de um jogo qualquer com outra pessoa (KIRNER, 2000; NETTO, 2003). A Realidade Virtual tem potencial e recursos que permitem a exploração e interação do usuário com o mundo virtual nos dois tipos de envolvimento.

Os recursos atualmente disponíveis para a navegação no ambiente virtual, permitem ao usuário interagir, sentir e observar o ambiente tridimensional em tempo real. Permitindo a navegação no mundo virtual através do uso de dispositivos convencionais e não convencionais, utilizando a combinação de movimentos de translação, rotação e escala para reproduzir os mesmos movimentos que podem ser realizados no mundo real. De maneira que se pode deslocar e rotacionar sobre os três eixos cartesianos X, Y, Z resultando os movimentos em 6 graus de liberdade (três translação e três rotação) ou *Six Degree of Freedom* (6DOF) ilustrado na Figura 16.

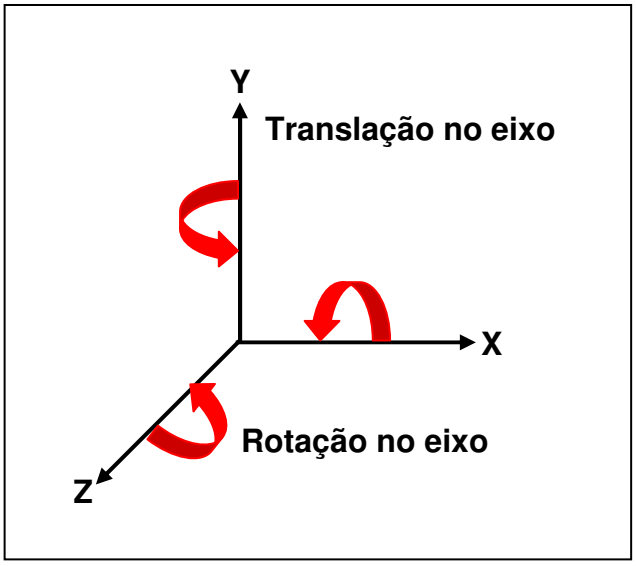


FIGURA 16 – EIXOS DE UMA CENA 3D.

A manipulação de todos os graus de liberdade disponíveis em cada objeto de um ambiente virtual exige grande capacidade computacional de hardware e software necessários para a renderização das imagens e para o

reconhecimento dos movimentos. O reconhecimento dos movimentos se dá pela captura de ações do usuário nos dispositivos de entrada, alterando as posições espaciais na cena. Causando a sensação de que a aplicação está funcionando em um ambiente tridimensional real, permitindo a exploração do ambiente e a manipulação natural dos objetos (KIRNER, 2000; NETTO, 2003).

A computação gráfica 3D envolve a criação de modelos matemáticos que são as esculturas ou objetos virtuais, organizados em cenas que formam e definem o espaço e o ambiente virtual. Os modelos podem ser iluminados de várias maneiras e visualizados por uma ou mais câmeras virtuais. A visão através de uma câmera é a conversão para uma imagem *rendering*, como um tipo de fotografia virtual (POLEVOI, 2001).

Existem basicamente dois tipos de animação que podem ser utilizadas em computação gráfica:

- ***Pre-rendered animation*** é uma animação realizada com modelo ou câmera que se move através de cada renderização, sendo o resultado uma seqüência de *frames* ou quadros, que através da exibição das imagens renderizadas em uma velocidade entre 24 a 30 frames por segundo, permite que o observador tenha a sensação de movimento não podendo interagir com a cena. A qualidade da renderização nesse tipo de animação pode ser extremamente alta, porque o criador pode refazer a renderização quantas vezes forem necessárias para gerar uma imagem boa. Cada um destes *frames* pode levar desde minutos até horas para serem renderizados, antes de serem utilizados para gerar uma seqüência de animação (POLEVOI, 2001).

- **Realtime 3D** é o processo de renderização que ocorre a cada frame o mais rápido possível, de acordo com o hardware e software existente no computador, onde o observador visualiza o processo de renderização em tempo real, podendo interagir com a cena. Neste processo são necessários no mínimo 10 *frames* por segundo, para que o observador tenha a sensação de movimento (POLEVOI, 2001).

2.5. *VIRTUAL REALITY MODELING LANGUAGE* (VRML)

Em 1989 um projeto chamado Scenario iniciado na Silicon Graphics, por Rikk Carey e Paul Strauss, que visava construir uma infra-estrutura para desenvolvimento de aplicações gráficas 3D, tinha como objetivo original criar uma grande variedade de interações, para aplicações distribuídas 3D e utilizar este ambiente para construir uma nova interface 3D para desktop. Em 1992 o *Iris Inventor 3D toolkit* foi apresentado como o primeiro produto com tais características, contendo muitas das semânticas encontradas atualmente no VRML (CAREY, 2000).

Uma segunda versão do *Inventor* foi lançada em 1994, denominada *Open Inventor*, devido ser portável para várias plataformas e por ser baseada em *OpenGL*, seu manual de referência descrevia todos os objetos e formato dos arquivos do *Open Inventor toolkit,* sendo utilizado para escrever a primeira versão da especificação VRML (CAREY, 2000).

Em Outubro de 1994 durante a 2ª Conferência Internacional da WWW em Chicago, a prmeira versão da especificação do VRML 1.0 foi publicada. Em 1995, algumas pequenas alterações foram realizadas na VRML 1.0, até ser constatada a necessidade de uma nova versão que oferece suporte a animação, interação e modelagem, sendo novamente modificado (CAREY, 2000).

Em Julho de 1996 durante a reunião internacional da ISO em Kyoto, o comitê *Join Technical Committee Information Technology* (JTC1) e o *Subcommittee Computer graphics and image processing* (SC24) aprovaram a versão VRML 2.0. desenvolvida em conjunto com a Silicon Graphics, Sony Corporation e Mitra.

Durante o encontro da SIGGRAPH'96 realizado em Agosto de 1996 em New Orleans a versão do VRML 2.0 foi apresentada. Em Abril 1997 a ISO aprova a versão conhecida como VRML97, que é sua última revisão deste padrão (VRML, 2003; CAREY, 2000).

A linguagem VRML é um formato de arquivo que descreve a interação de objetos 3D e ambientes virtuais. Desenvolvida para ser utilizada na Internet, Intranet e em rede local, como formato universal para troca de objetos 3D e multimídia, podendo ser usada em uma grande variedade de aplicações, como engenharia e visualizações científicas, apresentações multimídia, entretenimento, educação, Web pages e para compartilhar ambientes virtuais (CAREY, 2001; VRML, 2003).

Mas, para que *browsers* como Internet Explorer e Netscape Navigator possam exibir e manipular objetos e cenas 3D, além de exibir textos e imagens em HTML, é necessária a instalação de uma aplicação tipo *plug-in,* que permite ao *browser* interpretar novos tipos de arquivos, como interpretar arquivos com extensão .wrl no formato VRML. Desta maneira, o *plug-in* permite que o *browser* possa renderizar e interagir com o ambiente virtual criado na linguagem VRML (KIRNER, 2004).

Atualmente existem vários *plug-ins* para *browser* disponíveis para o padrão VRML e X3D, onde o padrão X3D será apresentado na seção 2.6. Dentre eles destacam-se:

- **Cortona VRML Client 4.2:** este *plug-in* VRML exibido na Figura 17 é o mais comumente utilizado, desenvolvido pela empresa ParallelGraphics o Cortona VRML Client 4.2 é um visualizador Web3D rápido e altamente interativo, que permite visualizações de modelos 3D pela Web. Possui um conjunto otimizado de renderizadores 3D que permite utilizar OpenGl ou DirectX, garantindo uma excelente qualidade visual em computadores com placas de vídeo básicas e avançadas.

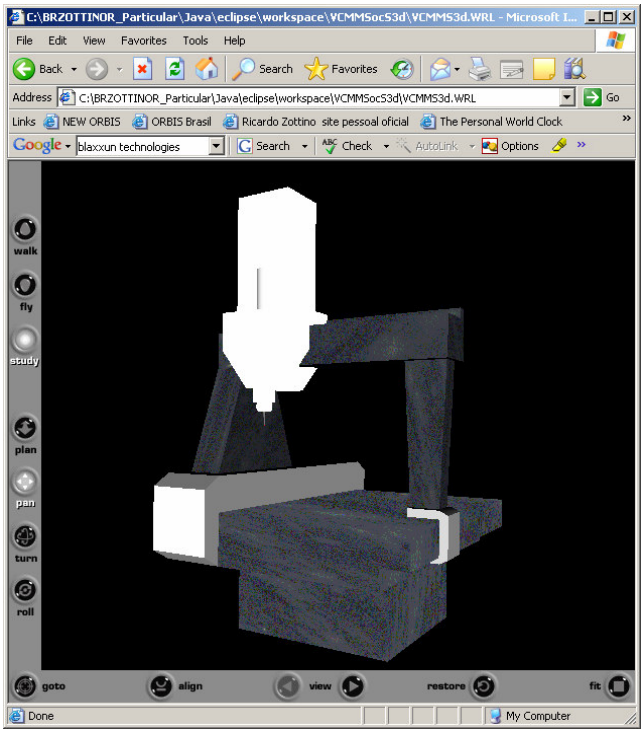


FIGURA 17 – CORTONA *PLUG-IN* EXIBINDO CENA DA VCMM.

Pode ser instalado em plataformas Windows e Macintosh é compatível com a maioria dos *browsers* existentes para a Internet, tais como: Internet Explorer, Nestcape Navigator, Mozilla, etc. Além disso, oferece ainda uma interface para automação que permite estender o controle da cena para uma aplicação externa, através de um kit para desenvolvimento de software SDK (*Software Development Kit*);

- **FluxPlayer 1.2:** desenvolvido pela empresa MediaMachines este *plug-in* para VRML oferece também o suporte a visualização de cenas em X3D, mas é restrito a plataforma Windows e para os *browsers* Internet Explorer e Nestcape Navigator, conforme ilustra a Figura 18.

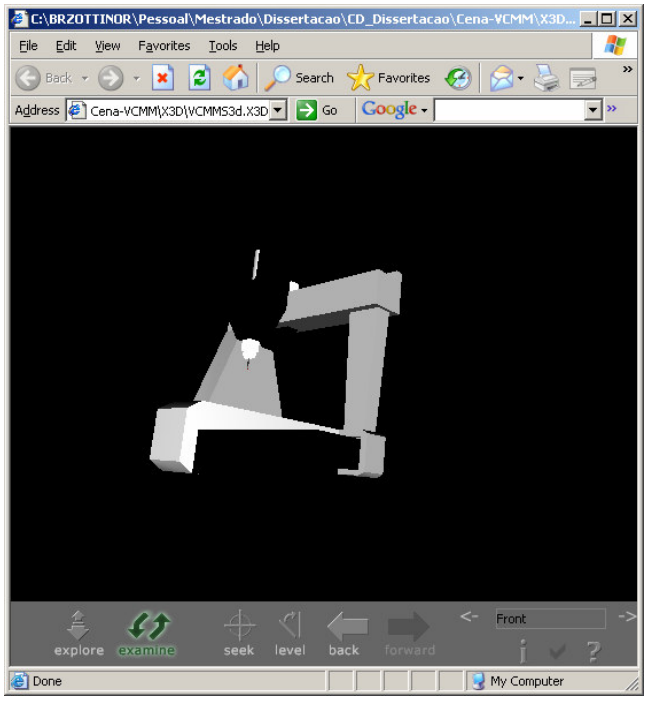


FIGURA 18 – FLUXPLAYER *PLUG-IN* EXIBINDO CENA VCMM.

- **Blaxxun Contact 5.3:** desenvolvido pela empresa Blaxxun este *plug-in* para VRML ilustrado na Figura 19. Suporta também a visualização e interação 3D de animações em Macromedia Flash e *streaming mídia*. Seu renderizador suporta DirectX ou OpenGL, que utilizando serviços do Blaxxun server, permitem aos seus usuários compartilhar mundos virtuais, salas de bate-papo, personalização de avatares[4] e gestos. Suporta somente a plataforma Windows e *browsers* como Internet Explorer e Nestcape Navigator. Também possui uma *Applet* em Java que permite visualizar cenas 3D sem a necessidade de instalação de um *plug-in* VRML no *browser*.

[4] Oriundo do sânscrito (avatãr) e na teologia indiana significa, cada uma das encarnações de um deus, especialmente de Vixnu. O termo avatar é usado para representar um humano virtual controlado pelo usuário (KIRNER, 2004).

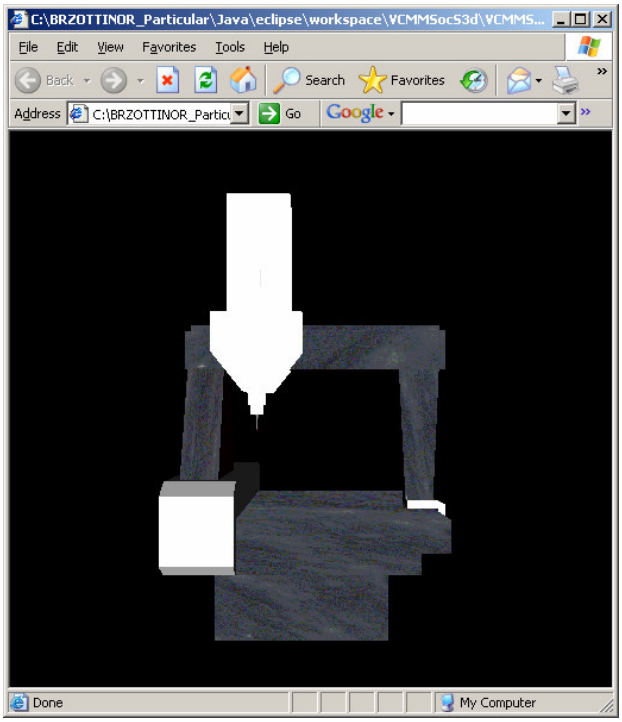


FIGURA 19 – BLAXXUN *PLUG-IN* EXIBINDO CENA VCMM.

- **CosmoPlayer 2.1.1:** é um *plug-in* para VRML desenvolvido pela empresa CosmoSoftware adquirida pela Computer Associates em outubro de 2004, que retirou o software e o site do mercado. Mas como cortesia permite ainda seu uso e distribuição para a comunidade científica através do *National Institute of Standards and Technology* (NIST). Atualmente este *plug-in* se encontra descontinuado, mas ainda oferece suporte para as plataformas Windows, Macintosh OS 9.x e Silicon Graphics e para os *browsers* Internet Explorer e Netscape Navigator conforme ilustra a Figura 20.

A utilização da linguagem de programação VRML possibilita descrever objetos 3D, permitindo definir seus detalhes como relacionamentos com outros objetos, localização, luzes e câmeras. Podendo ser chamada de linguagem de descrição de cenas 3D (POLEVOI, 2001).

A criação de objetos em um ambiente virtual utilizando a linguagem VRML é descrito por uma estrutura hierárquica conhecida como grafo de cena. O grafo de cena é um modelo que permite representar o ambiente virtual e todos os

seus elementos, relacionamentos e interações, por meio de um modelo matemático conhecido como grafo (KIRNER, 2004).

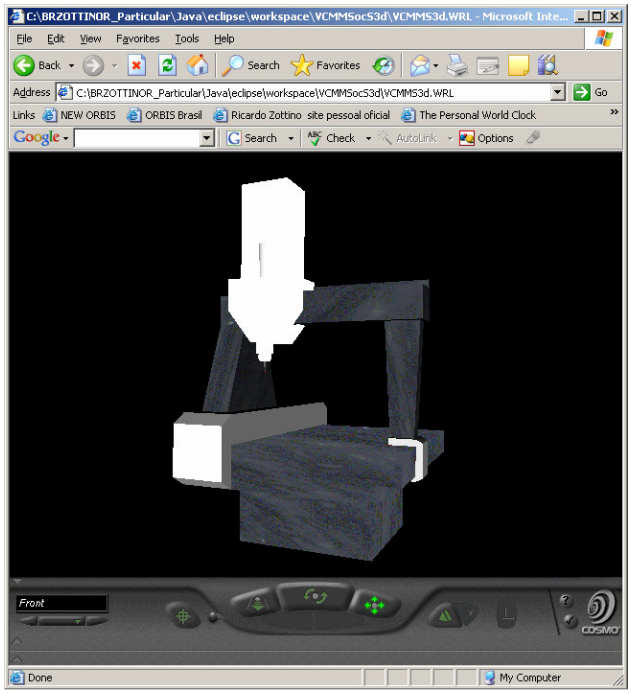


FIGURA 20 – COSMOPLAYER *PLUG-IN* EXIBINDO CENA VCMM.

A elaboração de um algoritmo eficiente para a resolução de um problema algorítmico constituiu o marco inicial da teoria de grafos, através do problema das pontes de Königsberg resolvido pelo matemático Suíço Leonhard Euler em 1736.

O campo de aplicação de grafos abrange os mais diferentes domínios da ciência, tais como: sistemas de informação geográfica, na determinação de rotas e conexões de transporte, circuitos elétricos e eletrônicos e em redes de computadores (GOODRICH, 2002).

A utilização de grafos em VRML é representada através da estrutura hierárquica em árvore, que descreve o grafo de cena da VCMM ilustrado na Figura 21. Os elementos de seu ambiente virtual são representados através de elipses denominadas de nó. Cada nó recebe um rótulo único de maneira que possa ser identificado no grafo de cena. Os nós superiores também conhecidos como pais transmitem suas características para os nós inferiores ou filhos.

Esta relação permite expressar um conjunto de características comuns a um nó ou vários nós, dependendo de sua estrura hierárquica. A linguagem VRML oferece suporte a 54 tipos diferentes de nós, estes nós permitem criar através de primitivas geométricas como cubos, esferas, cilindros e cones, os elementos do ambiente virtual. Propriedades relacionadas a aparência como texturas e cores, transformações lineares de translação e rotação também podem ser aplicadas em cada nó. A norma ISO/IEC 14772 descreve a sintaxe e semântica de todos os comandos da VRML (VRML, 2003).

A Figura 22 ilustra o grafo de cena da VCMM descrito de forma linear, ou seja, na linguagem VRML, editado com o auxilio do programa VrmlPad da Paralell Graphics, que permite a edição de cenas no padrão VRML.

A criação de um ambiente virtual, é portanto, a expressão dos elementos ou objetos do mundo real utilizando a VRML, que através de um modelo matemático denominado grafo de cena, permite descrever todos os objetos através de um grafo.

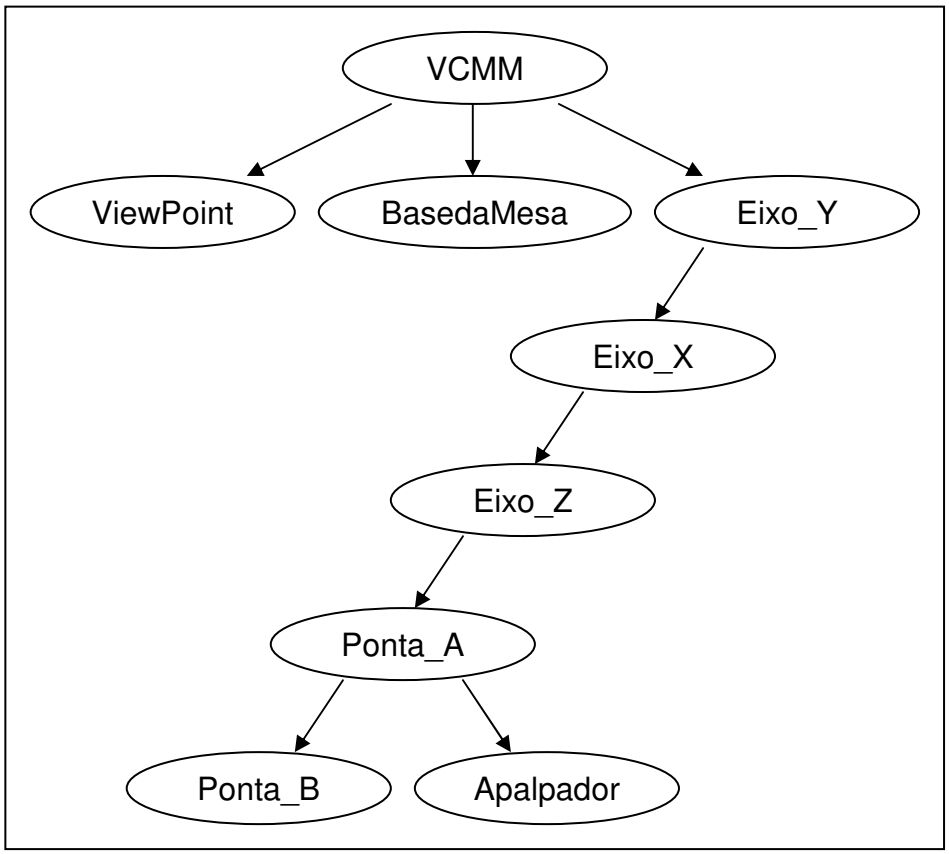


FIGURA 21 – GRAFO DE CENA DA VCMM.

A arquitetura do VRML permite que o *plug-in* possa interagir com o *browser*, exibindo as formas e sons do grafo de cena. Permitindo que o usuário possa interagir com o mundo virtual, através sensores de toque que podem ser inseridos no grafo de cena, que respondem a interação do usuário através dos objetos geométricos.

A Figura 24 apresenta o modelo conceitual de um *browser* VRML, onde o *browser* representa a aplicação que recebe a entrada de dado do usuário através de dispositivos e interfaces. Os principais componentes do *browser* são: *Parser*, *Scene*, *Route Graph* e *Audio/Visual Presentation*.

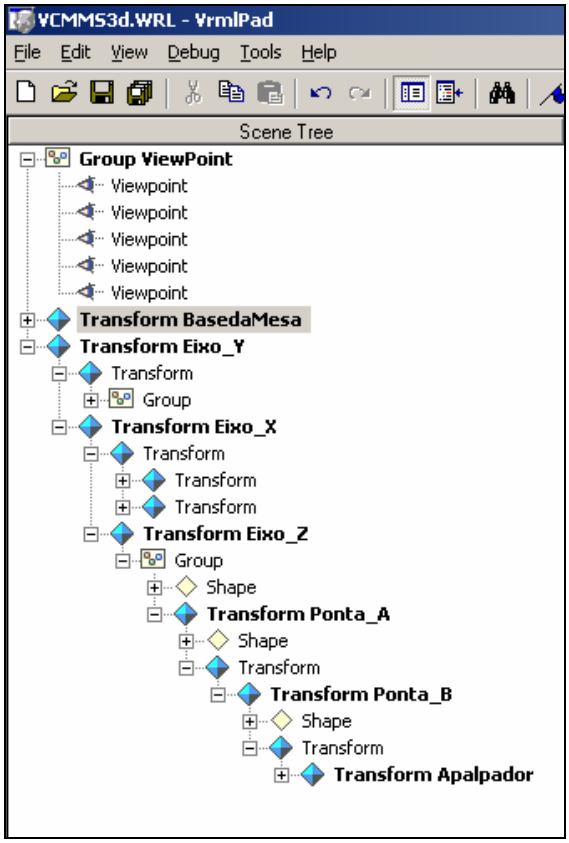


FIGURA 22 – GRAFO DE CENA DA VCMM.

O *Parser* tem a finalidade de receber e analisar o arquivo VRML e criar o grafo de cena. Os componentes do grafo de cena consistem em uma hierarquia de nós e *Route Graph*, que também é responsável pelo *Execution Engine* que

processa todos os eventos, leituras e edição do *Route Graph* e executa as alterações em cada nó. A entrada de dados do usuário geralmente afeta os sensores de toque e de navegação, onde o *Audio/Visual Presentation* executa a renderização e alteração dos nós envolvidos na interação do usuário.

```
#VRML V2.0 utf8

# VCMM - Virtual Coordenate Machine Measurement
# UNIMEP - Universidade Metodista de Piracicaba
# Developted by Nivaldi Calonego Junior
#

#view point
DEF ViewPoint Group {
}
#Base da Mesa
DEF BasedaMesa Transform {
    rotation 1.0 0.0 0.0 1.555
    children [
        #Inline { url "BaseDaMesa.wrl" }
        Group {
            children [
                Transform {
                    translation -10.0 8.0 -1.8
                    children [
                        Shape {
                            appearance Appearance {
                                texture ImageTexture {
                                    url "Marmore.gif"
                                }
                            }
                            geometry IndexedFaceSet {
                                creaseAngle 0.524
                                coord Coordinate {
                                    point [
                                        0.005 -6.847 4.007
                                        9.005 -6.847 4.007
                                        9.005 -6.847 -0.193
                                        0.005 -6.847 -0.193
                                        0.005 -6.847 4.007
                                        0.0 1.0 4.0
                                        9.0 1.0 4.0
                                        9.0 1.0 -0.2
                                        0.0 1.0 -0.2
                                        0.0 1.0 4.0
                                    ]
```

FIGURA 23 – TRECHO DO CÓDIGO VRML QUE IMPLEMENTA A VCMM.

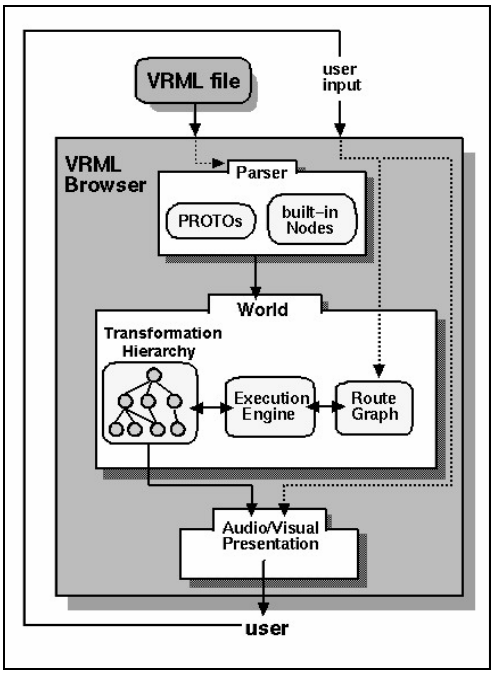


FIGURA 24 – MODELO CONCEITUAL DE UM BROWSER VRML.

2.6. *EXTENSIBLE X3D* (X3D)

Este novo padrão de arquivo 3D aberto foi desenvolvido em *Extensible Markup Language* (XML), para capacitar a comunicação de dados 3D em tempo real com todos os tipos de aplicações e redes. Suas características proporcionam seu uso em aplicações das áreas de engenharia, arquitetura, medicina, treinamento, simulação, multimídia, entretenimento, educacional e outras. (WEB3D, 2005).

Considerado a evolução do VRML o X3D é considerado um padrão mais refinado e robusto, sendo suas principais características:

- **Integração com XML:** possibilita a integração com Web Services, redes distribuídas e transferência de dados entre plataformas e aplicações;
- **Componentes:** permite a criação de pequenos códigos 3D especializados que podem ser reutilizados;

- **Extensível:** seus componentes podem ser estendidos permitindo adicionar novas funcionalidades;
- **Perfil:** oferece um conjunto de parâmetros para as necessidades de cada aplicação;
- **Evolucionário:** permite que cenas desenvolvidas em VRML possam ser facilmente atualizadas para X3D, preservando o conteúdo desenvolvido em VRML;
- ***Broadcast/Embedded Application Ready:*** permite a comunicação desde dispositivos móveis até supercomputadores;
- ***Real-Time:*** propicia a execução de gráficos em alta qualidade em tempo real;
- **Interativo:** permite a inclusão de áudio e vídeo bem como dados em 3D;
- **Gráficos 3D:** possibilita o uso de geometrias poligonais, paramétricas, transformações hierárquicas, iluminação, materiais e texturas;
- **Gráficos 2D:** possibilita o uso de texto, vetores 2D e superfícies planas.
- **Animação:** permite o uso de temporizadores e interpoladores para conduzir animações contínuas, animação de *humanóide* (tipo de comportamento realizado por um objeto representando um humano) e *morphing* (processo de translação da forma de um objeto para outra, por exemplo, um objeto quadrado se transformando em triângulo) (POLEVOI, 2001);
- **Áudio e Vídeo Espacial:** permite que recursos audiovisuais sejam mapeados dentro da geometria da cena;
- **Interação usuário:** capacidade de estender as funcionalidades do *browser* para criar tipos de dados definidos pelos usuários;

- **Scripting:** permite alterar dinamicamente a cena via programação ou linguagem script;

- **Rede:** possibilita a criação de cenas X3D em outros equipamentos localizados na rede, possibilitando ligar os objetos com outras cenas ou equipamentos localizados na Web;

- **Simulação física:** animação humana, geoespacial, integração com protocolos como *Distributed Interactive Simulation* (DIS).

A característica que mais se destaca no X3D em comparação com VRML, é o uso de um conjunto de componentes, que permitem aos desenvolvedores utilizarem e criarem diferentes tipos de perfis para suas aplicações. De maneira que estes componentes possam ser individualmente estendidos ou modificados, criando novos níveis ou novos componentes que podem introduzir novas caracteristicas, como por exemplo: *streaming* de vídeo.

Através deste mecanismo, especificações mais avançadas podem evoluir rapidamente, devido aos desenvolvimentos de uma área não aguardar a especificação de uma outra. Sendo importante lembrar que a conformidade dos requisitos para um determinado componente não pode ser ambígua, devendo ser definida através do seu perfil, nível e componente da aplicação.

A seguir é apresentado os quatro perfis básicos existentes no X3D conforme ilustra a Figura 25.

- ***Interchange:*** permite a comunicação e intercâmbio entre aplicações, suportando geometrias, texturas, iluminação básica e animação;

- ***Interactive:*** possibilita a interação com ambientes 3D adicionando vários sensores para a navegação e interação do usuário, por exemplo, sensor de toque;

- ***Immersive:*** proporciona a imersão completa em gráficos 3D, incluindo suporte a áudio, colisão e script;

- ***Full:*** permite a definição de nó como uma superfície NonUniform Rational B-Splimes (NURBS), Humanoide Animation (H-Anim) e componentes geoespaciais.

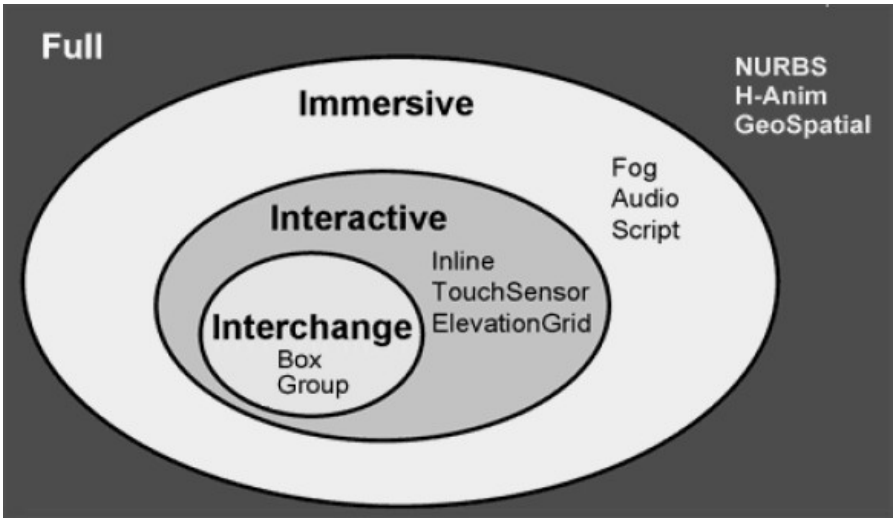


Figura 25 – Perfis disponíveis no padrão X3D.

Outros dois perfis se encontram em desenvolvimento e estudo:

- **MPEG-4:** possibilita a transmissão de *broadcast*, para dispositivos móveis e aparelhos celulares;

- ***CAD Distillation Format* (CDF):** permite a conversão de dados no formato CAD, para um formato aberto de publicação e interação.

Conceitualmente a arquitetura X3D é um ambiente virtual que pode ser carregado através da rede e dinamicamente modificado através de vários mecanismos. Sua semântica permite descrever o funcionamento abstrato e o comportamento de cada objeto. Mas não permite definir dispositivos físicos como, por exemplo, a resolução da tela ou qual o dispositivo de entrada e saída que está sendo utilizado.

Cada aplicação X3D implicitamente estabelece um espaço definido para seu ambiente e para todos os objetos contidos nele podendo: definir e criar vários objetos 2D, 3D e multimídia; especificar *hyperlinks* com outros arquivos e

aplicações; definir o comportamento dos objetos; se conectar com módulos externos ou outras aplicações através de programação ou linguagens de script;

2.7. JAVA

Em 1991, um grupo de engenheiros de projetos da SUN liderados por Patrick Naughton e James Gosling, queriam desenvolver uma pequena linguagem de computador que poderia ser utilizada em dispositivos de TV a cabo, e que oferece-se suporte a equipamentos com pouca memória e com diferentes tipos de processadores, sendo denominado este projeto de Green (HORSTMANN, 1999).

Os requisitos para escrever pequenos códigos levaram a equipe a utilizar uma linguagem chamada UCSD Pascal, que anteriormente havia tentando ser implementada em PC por Niklaus Wirth. Então, os engenheiros do projeto Green desenvolveram uma linguagem portável, que gerava códigos intermediários para uma máquina hipotética, atualmente chamada de *Java Virtual Machine* (JVM), onde este código intermediário poderia ser utilizado em qualquer processador que tivesse um interpretador apropriado para Máquina Virtual Java.

Com toda experiência em UNIX adquirida pelo pessoal da Sun, eles substituíram o Pascal por C++, modificando a estrutura procedural do Pascal pela orientação a objetos do C++. Então, James Gosling a chamou de Oak em homenagem a árvore que ficava em frente a janela de seu escritório na Sun, descobrindo mais tarde que este nome já era atribuído a uma outra linguagem. Mais tarde o nome Java foi sugerido em homenagem a cidade de origem do café, que era importado e utilizado em uma cafeteria que o grupo de desenvolvimento da Sun frequentava (DEITEL, 2001).

Em 1992, o projeto Green entregou seu primeiro produto chamado *7, leia-se *Star Seven*, que era um controle remoto inteligente, baseado em um computador SPARCstation fechado em uma caixa, conforme ilustra a Figura 26. Infelizmente nenhuma empresa se interessou pelo projeto da Sun e a

equipe tentava encontrar um outro destino para aplicação da sua tecnologia durante os anos de 1993 e 1994.

FIGURA 26 – PROTÓTIPO DO PROJETO *7 DA SUN.

Em 1994, com o avanço da Web e a introdução das páginas HTML, muitas pessoas já utilizavam um *browser* não comercial chamado Mosaic, que estava sendo desenvolvido na Universidade de Illinois desde 1993. Nesse mesmo ano Gosling disse em uma entrevista que poderiam desenvolver um *browser* com características cliente/servidor, arquitetura neutra, real-time, reutilizável e seguro.

Em 1996, após várias revisões a Sun lançou a primeira versão do Java 1.01, desde então várias melhorias foram incluídas na linguagem até chegar a versão atual Java.

A Sun define a tecnologia Java como sendo uma linguagem de programação e uma plataforma. A linguagem de programação Java pode beneficiar os usuários da Internet e da Web fornecendo acesso seguro, independência de plataforma, e permitindo que suas aplicações possam ser executadas de qualquer lugar da Internet.

Segundo (KEN, 2002) os desenvolvedores podem criar aplicações em linguagem Java beneficiando-se do desenvolvimento de um código único, não sendo necessário direcionar suas aplicações para cada tipo de plataforma de hardware e software.

Para (DEITEL, 2001) Java é uma linguagem de alto nível que pode ser utilizada para qualquer propósito, suas principais características são portabilidade entre plataformas, orientação a objetos, reutilizável e segura.

Muitas vezes Java é relacionada com C e C++, mas sua estrutura é um pouco diferente. Na maioria das linguagens de programação, os programas são compilados ou interpretados especificamente para rodar em um tipo de plataforma. Enquanto Java, por ser uma linguagem diferente, seus programas são compilados e interpretados, podendo o mesmo código rodar sob qualquer plataforma suportada por uma máquina virtual Java.

Atualmente a Sun disponibiliza Java em várias plataformas como Windows, Linux, Solaris, Macintoch entre outras conforme ilustra a Figura 27. O protótipo do AVD-W foi implementado em Java e executado o mesmo código nas plataformas, Windows 2000 e XP, Linux Mandrake 9 e Macintosh OS9 e Macintosh X.

Basicamente a plataforma Java é distribuída através do *Software Development Kit* (SDK) ou do *Standard Edition* (SE) conforme apresentado na Figura 27. Ambas utilizam a mesma Máquina Virtual Java e o mesmo conjunto de interfaces de programação Java *Application Programming Interface* (API) para executar as aplicações desenvolvidas.

A distribuição SDK é voltada para o uso de desenvolvedores, que precisam ter ferramentas disponíveis e API necessárias para programação. Enquanto a distribuição SE é voltada para os usuários que somente precisam ter o ambiente Java instalado em seu computador para executar programas na linguagem Java.

O JVM possui um interpretador específico para cada tipo de plataforma conforme ilustra a Figura 27, responsável por fazer a interpretação dos arquivos *bytecodes* e o gerenciamento do programa durante sua execução, gerando comandos em linguagem de máquina para o computador. “Todos os programas Java precisam ter uma máquina virtual para rodar” (THOMAS, 1997), ou seja, é necessário que cada computador possua previamente

instalado o JVM ou tenha um *browser* que suporte o Java *plug-in*, para que aplicações Java possam ser executadas.

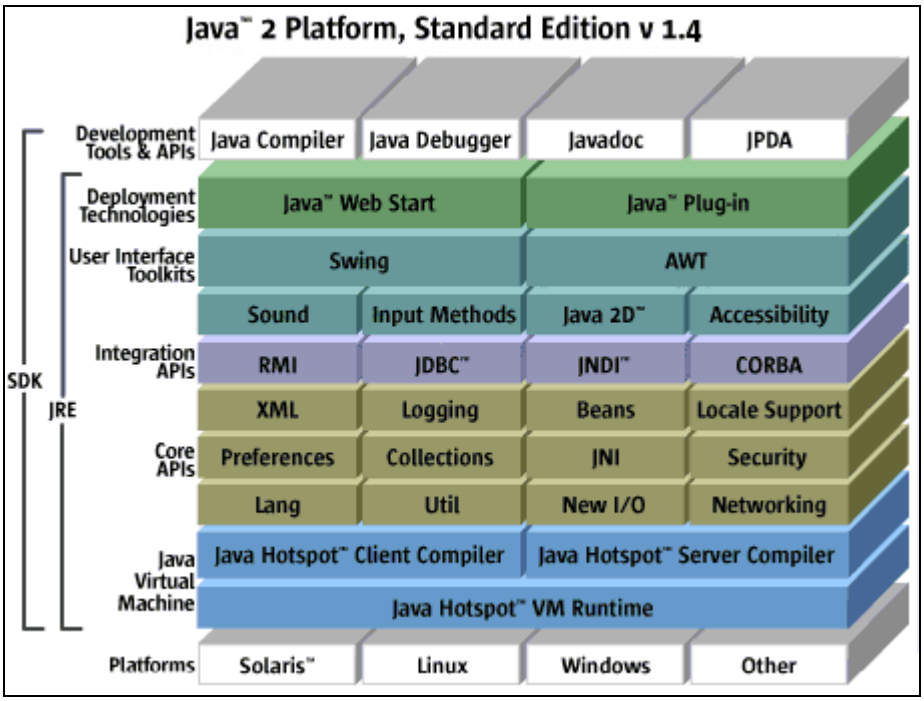


FIGURA 27 – PLATAFORMA DA LINGUAGEM JAVA.

Outra finalidade do JVM é manter a integridade do sistema e do programa para que nenhum comportamento hostil do programa interfira no funcionamento do computador ou que outros programas que estão alocados na memória do computador interfiram no funcionamento da JVM.

Normalmente um programa escrito na linguagem Java possui a extensão .java, como por exemplo, o programa myProgram.java que ao ser compilado utilizando o compilador Java **javac.exe**, traduz o programa para uma linguagem intermediária chamada Java *bytecode*. Este conjunto de códigos em *bytecode* cria um arquivo no formato binário com a extensão .class, neste exemplo o arquivo myProgram.class que será interpretado e executado no computador por uma máquina virtual, ou seja, pela JVM conforme ilustra a Figura 28.

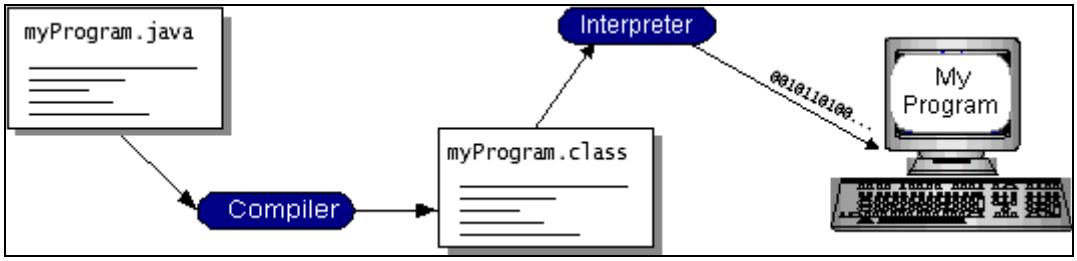


Figura 28 – Seqüência utilizada para compilar e executar uma aplicação Java.

Quando utilizado o protocolo RMI, é necessário que ao concluir a compilação uma segunda etapa seja realizada para que o *stub* e *skeleton* possam ser gerados e permitir a comunicação entre o cliente e o servidor RMI. O utilitário **rmic.exe** é responsável por criar a classe cliente ServerShout_Stub.class e a classe servidor ServerShout_Skel.class. Estas classes devem estar disponíveis junto com o restante da aplicação, a fim de permitir a comunicação remota com o objeto servidor.

Os programas compilados em *bytecodes* podem ser executados em qualquer tipo de plataforma que tenha um interpretador Java, possibilitando que o mesmo *bytecode* rode em várias plataformas, como ilustrado na Figura 29. Porém, ao executar *applet*s em *browsers* como, por exemplo: Internet Explorer ou Netscape Navigator ou outros *browsers* é necessário que se tenha instalado o Java *plug-in* para que ele tenha suporte a JVM (THOMAS, 1997).

O Java *plug-in* é um componente que permite a execução de *applets* Java em um *browser* estendendo suas funcionalidades. Sua instalação acompanha a *Java Runtime Environment* (JRE) presente nas distribuições Java SDK e SE. Existe também um utilitário chamado appletviewer.exe que acompanha a plataforma Java SDK, que tem a capacidade de executar uma *applet* sem a necessidade da existência de um *browser* previamente instalado.

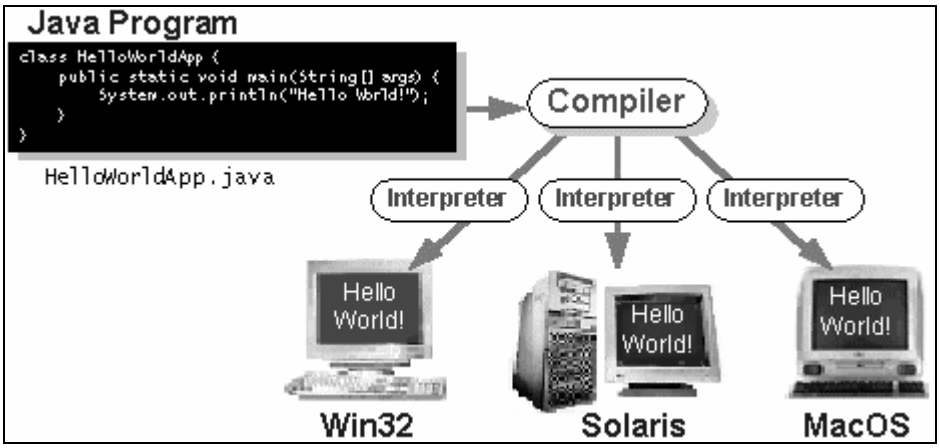


Figura 29 – Programa pode ser executado em diferentes plataformas.

A interface de programação de aplicação Java ou Java API é formada por um conjunto de bibliotecas de classes e componentes de software pré-definidos, que oferecem inúmeros recursos, baseados na seguinte estrutura:

- Bibliotecas de Classes ou pacotes são coleções de classes;
- Classes são programas Java, que podem estar distribuídos pela rede local ou pela Web, contudo ficam geralmente armazenados no servidor local, para poderem prover uma resposta mais rápida as solicitações do cliente;
- Métodos são partes da classe que realizam tarefas que retornam informações ao completar suas tarefas;
- *Applet* segundo (KEN, 2002) é uma pequena aplicação que roda dentro de uma página Web. Uma *applet* pode executar várias tarefas, interagir com os usuários através do *browser*, sem necessitar recursos de um servidor Web após ser carregada. Algumas *applets* podem conversar com o servidor para realizar uma tarefa específica.

Para que uma *applet* possa ser executada em um *browser* é necessário que seja criado um arquivo HTML, que contenha a tag **<applet></applet>** especificando qual a classe a ser executada pelo browser.

Todas as tags HTML iniciam com sinal de menor **<**, e terminam com um sinal de maior **>**. A tag **<applet>** possui vários parâmetros que podem ser obrigatórios ou não dependendo de seu uso, conforme descreve a documentação da API do Java plug-in. O primeiro parâmetro **code=** indica qual o arquivo que contém a classe *applet* a ser carregada, o segundo parâmetro **archive=** especifica o nome da biblioteca geralmente com a extensão **.jar** ou **.zip**, o terceiro e quarto parâmetro respectivamente indicam a largura **width=** e a altura **height=**, medidos em *pixels* do frame que será criado para execução da *applet* no *browser*. A tag **<param name="" value=>** especifica um atributo e seu valor que pode ser passado para a classe que será executada.

A Figura 30 exibe o arquivo HTML que executa a aplicação AVD-W para Máquinas de Medir por Coordenadas (CMM).

```
<HTML>
<HEAD>
<TITLE> Ambiente Virtual Distribuído Web para Máquinas de Medir por
Coordenadas </TITLE>
</HEAD>
<BODY>
<APPLET CODE="ExamineApplet.class" ARCHIVE="shout3d.zip" WIDTH=600
HEIGHT=400>
<param name="src" value="VCMM.WRL">
  <param name="backgroundColorR" value="0.9">
  <param name="backgroundColorG" value="0.9">
  <param name="backgroundColorB" value="0.9">
  <param name="headlightOn" value="true">
  <param name="Interface" value="Console">
</APPLET>
Para alterar a interface edite o arquivo VCMM.html e mude o
parâmetro Interface
    name="Interface" value="Scrollbar"
    name="Interface" value="Console"
</BODY>
</HTML>
```

FIGURA 30 – PROGRAMA HTML QUE INICIALIZA A *APPLET* DA APLICAÇÃO VCMM.

Durante o desenvolvimento desse projeto fez-se necessário o uso de pacotes e bibliotecas para a linguagem Java, que em sua maioria estão disponíveis na distribuição Java da Sun e também o uso de outros pacotes desenvolvidos por terceiros como Shout3D e Java3D, conforme descrito na Tabela 1.

2.8. SHOUT3D

A biblioteca Shout3D permite exibir gráficos 3D e animações interativas na Web diretamente em um *browser* através de uma *Applet* Java, não sendo necessário nenhum *plug-in* VRML ou aplicações adicionais. Compatível com a maioria dos *browsers* existentes no mercado que suporte *plug-in* Java, Shout3d permite interagir e manipular objetos e cenas previamente criadas em VRML ou em pacotes gráficos e de animações 3D como Studio Max 3D, Spazz3D, podendo seus objetos e animações ser exportados para o formato VRML (POLEVOI, 2001).

A biblioteca Shout3D foi desenvolvida pela empresa Eyematic Interfaces, Inc. (Califórnia, US) durante 3 anos liderou as pesquisas de tecnologia na área de reconhecimento facial. Seu centro de pesquisa incluía universidades da Alemanha e Estados Unidos. Em Fevereiro de 2001 foi fundada a Eyematic Japan e firmado uma *join venture* com a Eyematic Interfaces, Inc. (Califórnia, US). Em maio de 2004 a Eyematic Japan mudou de nome para N-Vision devido a equipe de pesquisa da antiga Eyematic criarem uma nova companhia chamada Neven Vision, Inc. (Califórnia, US) e retiraram do mercado todas ações da Eyematic Japan.

Atualmente a Shout3D não está mais disponível no mercado, alguns e-mail foram enviados para pessoas que trabalharam na Eyematic, na busca por informações sobre a biblioteca, mas não foi obtida nenhuma informação sobre sua continuidade.

A API Shout3D para Java contém todos os métodos e classes necessárias para executar todas as atividades de criação, manipulação e exibição de cenas 3D. Essas classes podem ser representadas por câmeras, luzes, modelos e outras classes que provem interação com os objetos da cena. Funcionalidades adicionais podem ser implementadas a partir da especialização de suas classes originais.

TABELA 1 – PACOTES E BIBLIOTECA DE CLASSES EM JAVA.

Pacote	**Descrição**
Java.applet	*Java Applet Package* Esse pacote contém a classe *Applet* e várias interfaces que permitem a criação de *applets* e sua integração com o browser.
Java.awt	*Java Abstract Windowing Toolkit Package* Pacote contendo as classes e interfaces exigidas para manipular interfaces gráficas.
Java.io	*Java Input/Output Package* Este pacote contém classes que suportam a entrada e saída de dados.
Java.lang	*Java Language Package* Contém classes e interfaces requeridas por muitos programas Java e são automaticamente importados pelo compilador em todos os programas.
Java.net	*Java Networking Package* Contém classes que permitem os programas se comunicarem via redes.
Java.util	*Java Utilities Package* Contém várias classes de utilitários como: interfaces de manipulação de data, hora, processamento de números, armazenamento e processamento de dados.
j3dcore j3dutils	*Java 3D* Este pacote permite a manipulação de geometrias 3D utilizadas para a construção rápida de gráficos complexos em DirectX, OpenGL, VRML e também permite a manipulação de sons.
Vecmath	*3D vector math package* Este pacote implementa vetores e cálculos matemáticos.
Shout3d	*Shout3D* Contém 3 sub pacotes shout3d.core, shout3d.math and shout3d.sound.
Shout3d.core	*Shout3D* Pacote principal contendo todas as classes e interfaces que implementa todo a arquitetura gráfica de cenas do Shout3D.
Shout3d.hanim	*Shout3D* Contém 5 classes na qual implementa a especificação H-Anim para figuras humanóides.
Shout3d.math	*Shout3D* Contém 2 classes MatUtil e Quaternion, para trabalhar com matrizes, vetores e rotação.
Shout3d.sound	*Shout3D* Pacote contendo a classe JavaSound, utilizada para tocar sons em uma applet Shout3D.

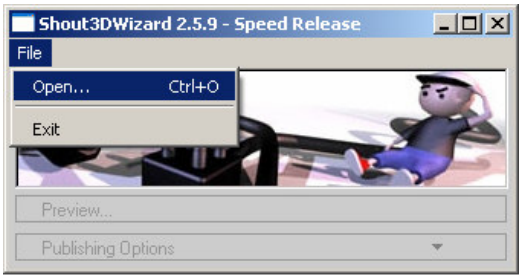
O Shout3D possibilita exibir cenas 3D sem nenhuma programação simplesmente utilizando utilitário Shout3Dwizard.exe para converter cenas VRML para um formato proprietário do Shout3d, a fim de serem executadas em um *browser* em conjunto com sua biblioteca Shout3D por meio de uma *applet* Java.

Devido ao Shout3D ser baseado em VRML seus arquivos são uma extensão do formato VRML. Um arquivo com extensão .s3d é uma cena VRML convertida para o padrão Shout3D, através do utilitário Shout3Dwizard.exe que

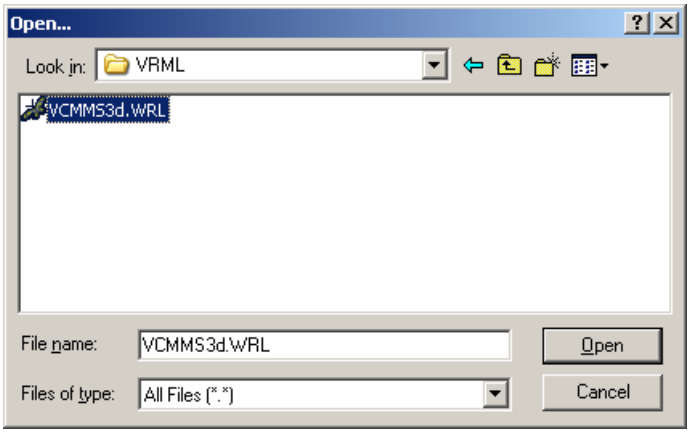
mantém a estrutura original do VRML e agrega novas funcionalidades que não são suportadas pela VRML, como interações dinâmicas sobre o modelo, não sendo necessário que o modelo VRML seja recriado.

Para exemplificar o uso da biblioteca Shout3D sem a necessidade de programação em Java, descrevemos abaixo as etapas necessárias para realizar a conversão de uma cena no formato VRML para o formato Shout3D. Partindo da premissa que a linguagem JAVA e o Shout3D estejam previamente instalados e configurados, conforme descrito na seção B.1.7.

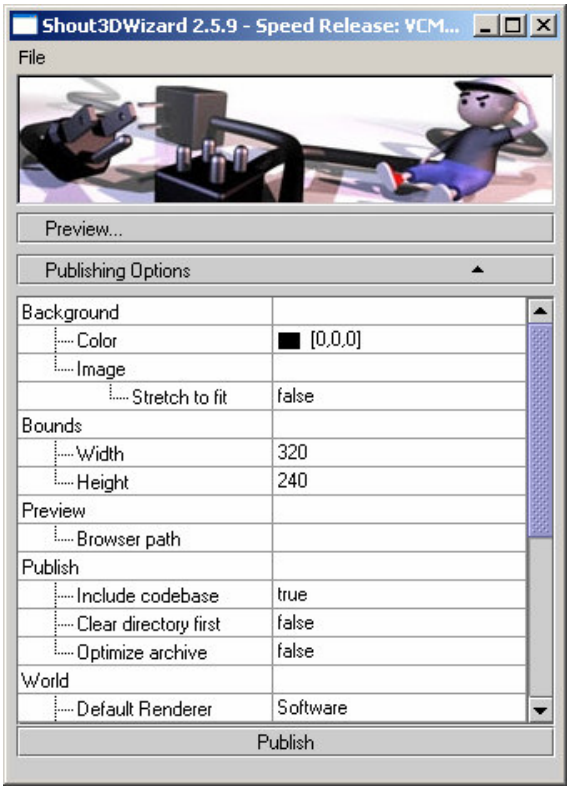
1. Executar o utilitário **Shout3Dwizard.exe**.



2. Selecionar o menu **File** e clicar na opção **Open**.

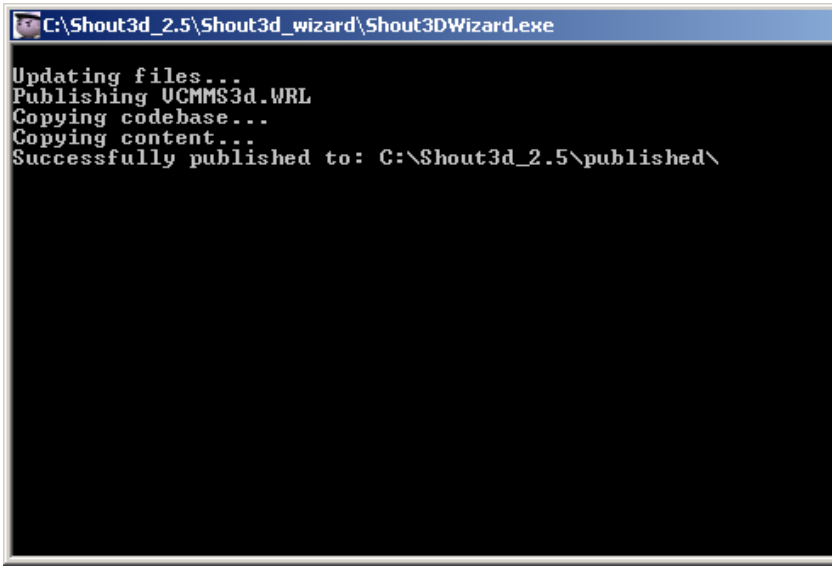


3. Neste caso vamos utilizar a cena VRML Vcmms3d.wrl que descreve a Máquina de Medir por Coordenadas (CMM).



4. Clicar no **Publish** para iniciar a conversão.

5. O resultado da conversão é apresentado na janela do prompt de comando, conforme apresentado abaixo.



O resultado da conversão é armazenado na pasta C:\Shout3d_2.5\published, onde as estruturas de pastas apresentadas na Figura 31 são criadas para que a cena possa ser executada.

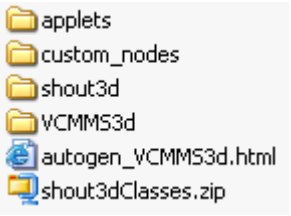


FIGURA 31 – ESTRUTURA DE PASTAS GERADO PELO UTILITÁRIO SHOUT3DWIZARD

Para cada nova cena convertida é criada uma nova pasta com o nome do arquivo da cena, em nosso caso foi criada a pasta VCMMS3d. Dentro desta pasta é criada a nova cena VCMMS3d.s3z no formato Shout3D e os arquivos de imagens associados a cena VRML, em nosso caso o arquivo Marmore.gif.

O arquivo HTML autogen_VCMMS3d.html também é criado automaticamente, ele é responsável por descrever os parâmetros da *applet* que será executada para carregar a cena VCMMS3d.s3z no browser conforme Figura 31. Mas devido a um bug deste utilitário, é necessário alterar o parâmetro **code="shout3d/Shout3DApplet.class"** da applet gerada na Figura 32, para **code="applets/ExamineApplet.class"**, para que o código HTML possa ser carregado pelo *browser* e funcione corretamente conforme Figura 33. Desta forma poderá ser exibido pelo Shout3D conforme Figura 34.

```
<HTML>
<HEAD>
<TITLE>Autogenerated HTML : VCMMS3d</TITLE>
</HEAD>
<BODY>

<APPLET CODEBASE="." CODE="shout3d/Shout3DApplet.class"
ARCHIVE="shout3dClasses.zip" WIDTH=320 HEIGHT=240>
<param name="src" value="VCMMS3d/VCMMS3d.s3z">
<param name="headlightOn" value="true">
<param name="regcode" value="">
<param name="regname" value="">
<param name="antiAliasingEnabled" value="false">
<param name="bilinearFiltering" value="false">
<param name="loadResourcesInSeparateThread" value="true">
</APPLET>

</BODY>
</HTML>
```

FIGURA 32 – ARQUIVO AUTOGEN_VCMMS3D.HTML GERADO AUTOMATICAMENTE.

```
<HTML>
<HEAD>
<TITLE>Autogenerated HTML : VCMMS3d</TITLE>
</HEAD>
<BODY>

<APPLET CODEBASE="." CODE="applets/ExamineApplet.class"
ARCHIVE="shout3dClasses.zip" WIDTH=320 HEIGHT=240>
<param name="src" value="VCMMS3d/VCMMS3d.s3z">
<param name="headlightOn" value="true">
<param name="regcode" value="">
<param name="regname" value="">
<param name="antiAliasingEnabled" value="false">
<param name="bilinearFiltering" value="false">
<param name="loadResourcesInSeparateThread" value="true">
</APPLET>

</BODY>
</HTML>
```

FIGURA 33 – ARQUIVO AUTOGEN_VCMMS3D.HTML ALTERADO.

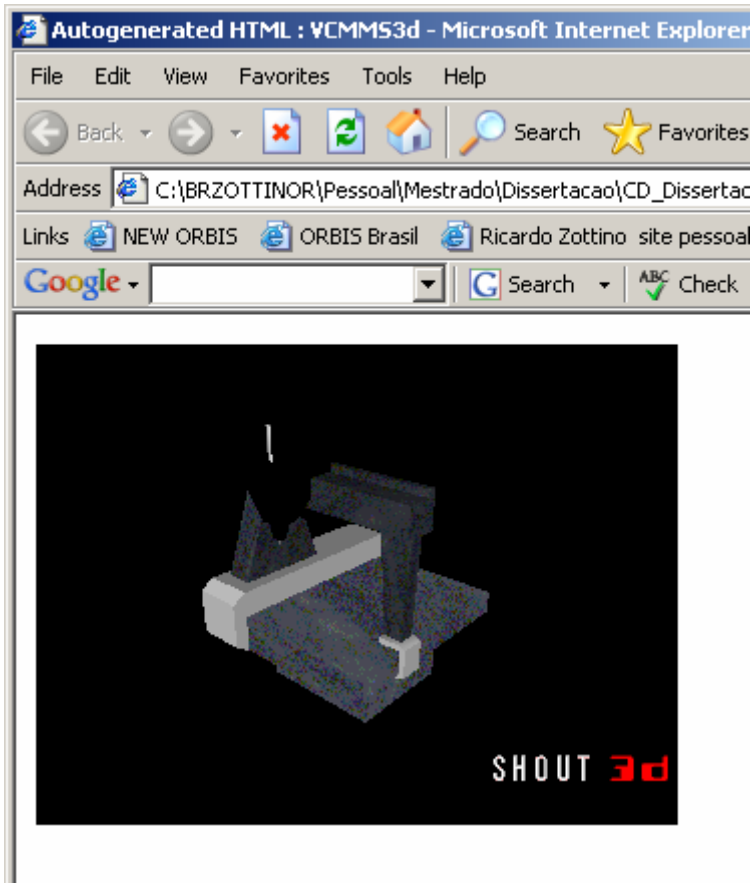


FIGURA 34 – SHOUT3D EXIBINDO CENA VCMM.

De maneira semelhante ao Shout3D outras plataformas foram desenvolvidas baseadas em gráficos de cenas, em que foram adotadas várias abordagens conforme as necessidades de mercado. Alguns exemplos são Cult3D, Viewpoint e Virtue3D (POLEVOI, 2001).

Um comparativo realizado por (CHEN, 2001) apresenta algumas plataformas Web3D que utilizam Java para exibir seus objetos e cenas 3D:

- Blaxxun3D: produto comercializado pela Blaxxun Interactive Inc, que utiliza Java e pode exibir modelos 3D na Internet, lê arquivos no formato VRML, possui sua própria API para a programação de animação de modelos 3D em Java;

- Cortona: comercializado pela Parallel Graphics Inc, desenvolvido em Java também pode exibir modelos 3D na Internet, lê arquivos no formato VRML, mas não possui uma API para programação;

- Xj3D: é um projeto open-source do Consórcio Web3D, também desenvolvido em Java, sendo sua renderização gráfica baseada em Java3D, que permite exibir arquivos no formato VRML e X3D;

- Shoult3D: foi desenvolvido pela Eyematic Interfaces Inc, utiliza Java e pode exibir modelos 3D na Internet, convertendo de VRML para um arquivo em formato proprietário, que possui sua própria API para a programação de animação de modelos 3D.

2.9. TRABALHOS CORRELATOS

Esse tópico apresenta outros trabalhos de pesquisa, que estejam relacionados com o desenvolvimento de ambientes virtuais distribuídos ou colaborativos, contribuindo para identificar os principais conceitos, aspectos e dificuldades encontrados no desenvolvimento deste tipo de aplicação.

Durante esta busca foram encontrados vários trabalhos, alguns demonstrando quais as principais considerações sobre o desenvolvimento de ambientes

virtuais distribuídos, outros apresentando técnicas específicas, como comunicação, criação e manipulação de avatares, modelo de interface para navegação em ambiente virtual, entre outros.

O *Distributed Interactive Virtual Environment* (DIVE) é um sistema de Realidade Virtual Distribuído Interativo, multi-usuário baseado em Internet, que proporciona navegação e visualização espacial 3D, possibilitando a interação de usuários e aplicações. Sua primeira versão surgiu em 1991, desenvolvido na Suécia pelo *Swedish Institute of Computer Science* (SICS). Permite a interação de um usuário que provoca a difusão do evento na rede, através da comunicação *multi-cast*, não possuindo servidor centralizado. O compartilhamento de estado é análogo à memória compartilhada, onde um conjunto de processos interage através de acessos concorrentes a essa "memória compartilhada" (GREENHALGH, 1996; SAAR, 1999; DIVE, 2004). Ele é o precursor de outros ambientes, como o Avango, que oferece um *framework* para desenvolvimento de ambientes virtuais distribuídos, e utiliza um modelo de programação análogo ao DIVE. O Avango é desenvolvido em C que distribui as informações através da replicação de uma cena para todos os processos que participam da aplicação distribuída (TRAMBEREND, 2000).

Noutra linha, aparece o SPIN-3D em que se explora a plataforma CORBA/ORB de sistemas distribuídos para implementar o Ambiente Virtual Colaborativo (AVC). Este disponibiliza cenas 3D destinadas à colaboração entre pequenos grupos utilizando VRML. Sua implementação é na linguagem C++, com CORBA e ORBacus, utilizando o *framework Open Communication Interface* (OCI) para implementar o protocolo *Multicast Inter-ORB Protocol* (MIOP). O trabalho de colaboração entre os usuários é realizado através da replicação do mundo virtual VRML por um servidor, que faz cópia ou "clone" da cena para cada cliente conectado (PICARD, 2001).

O COVEN é uma concepção mais ampla de Realidade Virtual Distribuída. Ele se apresenta como derivação do DIVE e do dVS (GREENHALGH, 1996; COVEN, 2004) e foi desenvolvido por um consórcio Europeu, no período de 1994 a 1998, objetivando a análise de requisitos e viabilidade de aplicações

Computer Supported Cooperative Work (CSCW). Utiliza múltiplos servidores para o gerenciamento das comunicações, estes definem espaços de interação diferentes, permitindo comunicação em longas distâncias para a colaboração em ambientes virtuais, e uso de multimídia para reuniões (GREENHALGH, 1996; COVEN, 2004).

Outro experimento é o *Virtual Environment Operation System* (VEOS), implementado em 1993, é um ambiente de programação C e AutoLISP, desenhado para a prototipação rápida de um Ambiente Virtual Distribuído, em que seu estado é armazenado em um banco de dados contendo tuplas privadas e públicas. Neste caso, o banco de dados reflete o estado atual do mundo virtual, cujo modelo enfatiza a comunicação assíncrona e a distribuição baseada em entidades (GREENHALGH 1996; COCO, 1993; SAAR, 1999).

A questão da persistência de um mundo virtual é o foco do NPSNET. Em desenvolvimento desde 1993, sua versão atual é o NPSNET-V, que implementa um componente Java para o desenvolvimento de aplicações cliente-servidor, ponto-a-ponto ou *standalone*. Seu objetivo é disponibilizar uma plataforma capaz de implementar no "*cyberspace*" a persistência de um mundo virtual que não necessita ser desativado para manutenção ou upgrade do sistema, buscando qualidades como extensibilidade de conteúdo e aplicações, escalabilidade para mundos complexos e com grande número de participantes e composição heterogênea de conteúdo e aplicações (GREENHALGH, 1996; SAAR, 1999; NPSNET, 2004).

Os trabalhos citados anteriormente representam concepções diferentes para a implementação de ambientes para o desenvolvimento de aplicações em Realidade Virtual. Projetos como o MASSIVE (MASSIVE, 2004) e World ToolKit (SENSE8, 2004) continuam sendo aprimorados a cada nova versão. Mas, muitos dos ambientes virtuais colaborativos, distribuídos e *frameworks* não tiveram continuidade, devido à forte correlação do software com plataformas específicas ou à falta de um planejamento para continuidade do projeto em futuros projetos de pesquisa, ou mesmo abandono do projeto inicial. A pesquisa bibliográfica revela outras implementações, ADVICE (RODRIGUES,

2003), AVIARY (GREENHALGH, 1996; SAAR, 1999; SNOWDON, 2004), Ambiente Virtual Interativo Tridimensional (AVIT) (IPOLITO, 1999), VIRTUS (SAAR, 1999), VUE (GREENHALGH, 1996).

Dos ambientes distribuídos apresentados, destacamos o VIRTUS, que tem como objetivo utilizar padrões abertos de software no desenvolvimento de plataforma colaborativa multi-usuário.

VIRTUS

O sistema *"VIRTUS: A Collaborative Multi-User Platform"*, é uma plataforma multi-usuário distribuída, que permite a seus usuários a manipulação e o compartilhamento de cenas VRML. O interesse em desenvolver este sistema originou-se, devido a maioria dos Ambientes Virtuais Distribuídos desenvolvidos até 1999, terem restrições quanto ao uso de aplicações em plataformas proprietárias. Isto estimulou o desenvolvimento do sistema VIRTUS, que teve sua arquitetura desenvolvida com base nos seguintes requisitos:

- **Portabilidade:** o sistema deveria ser aberto para várias plataformas de software e hardware, não sendo específico para nenhuma área de aplicação;

- **Consistência**: todos os usuários distribuídos pelo mesmo mundo virtual supostamente teriam a mesma visão do mundo, sua manipulação deveria ser imediatamente refletida no mundo dos demais usuários;

- **Dinâmico:** os usuários podem entrar e sair do ambiente virtual a qualquer momento, sendo este evento replicado para todas as outras instâncias;

- **Particionamento:** mundos complexos deveriam ser particionados para minimizar a comunicação e reduzir a complexibilidade de seu controle;

- **Extensibilidade:** o sistema deveria ser implementado de maneira a facilitar a implementação de novos protocolos de comunicação, novos objetos e funcionalidades como gerenciamento de colisão entre objetos;

- **Banco de dados:** o sistema deveria ter uma interface com banco de dados para os objetos, avatares e comportamentos;

- **Suporte a comportamento complexo:** o sistema não deveria se limitar a simples movimentos e comportamento dos objetos, sendo necessário um banco de dados para armazenar o comportamento de todos os objetos.

O sistema VIRTUS foi implementado utilizando VRML97, Java 1.0, Java 1.1 e *External Authoring Interface* (EAI), sendo seu cliente executado em máquinas Silicon Graphics O2 e Indigo2 com IRIX6.x, PCs com Windows NT 4.0 e Linux utilizando Netscape Navigator 3.1S, 4.05, Microsoft Internet Explorer, CosmoPlayer 1.0.2 e 1.1 em SGI e CosmoPlayer 2.1 em NT. O servidor foi instalado em um SPARC10 Sun com Solaris 2.6 como servidor HTTP e FTP, em um segmento de rede Ethernet 10Mbps e um PC conectado via *Integrated Services Digital Network* (ISDN), ou seja, uma linha telefônica digital utilizada para prover uma alta transferência de dados.

Com base nos trabalhos correlatos apresentados, os problemas detectados para a elaboração de ambientes virtuais distribuídos podem ser significativamente reduzidos através da delimitação de problemas, tais como: gerenciamento de usuários, colisão, comunicação de dados, consistência, portabilidade e particionamento de mundos virtuais.

3. DESCRIÇÃO DO AMBIENTE VIRTUAL DISTRIBUÍDO

Neste capítulo discute-se a elaboração do protótipo do Ambiente Virtual Distribuído Web (AVD-W) para Máquinas de Medir por Coordenadas VCMM apresentado o projeto e a implementação de suas classes e interfaces.

Esta aplicação é destinada ao ensino da Metrologia e oferece suporte para interação local ou remota na utilização da VCMM. Devido ao seu grau de desacoplamento de controles, esta aplicação pode ser reutilizada em ambientes virtuais análogos ao discutidos neste trabalho.

A finalidade de uma Máquina de Medir por Coordenadas (CMM) é determinar medidas geométrico-dimensionais, na qual são modelados vinte e um graus de liberdade. A Figura 35(a) ilustra a máquina real e a Figura 35(b) ilustra a máquina modelada em VRML, com seus principais componentes.

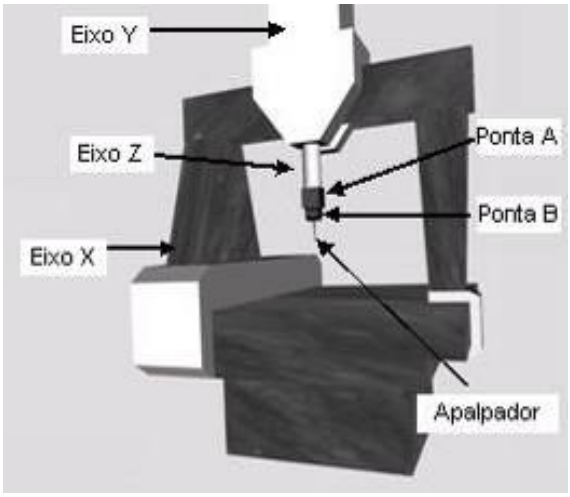
Essa máquina permite medir a geometria de superfícies com tolerância mínima, oferecendo incertezas da ordem de milésimo de milímetro (μm), onde exemplos de geometrias verificadas em tal máquina incluem formas das superfícies, posição de furos, distâncias de extremidades, etc. A CMM atende as normas internacionais que definem sua linguagem de programação.

A programação de máquinas CMM requer do operador conhecimento específico, habilidade de programação e também acesso ao ambiente de programação no qual a codificação será testada. Além disso, o processo de medir usado na máquina real depende do comportamento de alguns componentes individuais, tais como: a escala de medida, a estratégia de amostragem, as peças ou partes a serem medidas, entre outros. Posto que um mesmo programa usado para medir uma mesma peça pode apresentar resultados distintos em realizações sucessivas de medição, em função dos erros e incertezas inerentes ao processo.

A aplicação desenvolvida neste trabalho disponibiliza uma interface gráfica, cujo objetivo é reduzir a complexidade da tarefa de programação e criar uma

interface amigável para reduzir os erros léxicos e sintáticos intrínsecos à tarefa de programação.

a) Máquina Real (CMM)



b) Máquina Virtual implementada em VRML (VCMM)

FIGURA 35 – ILUSTRAÇÃO DA MÁQUINA DE MEDIR POR COORDENADAS.

O projeto privilegiou a flexibilidade de uso da aplicação, definindo uma interface única para o transporte de comandos da máquina através de uma rede de computadores, permitindo o uso da VCMM em ambiente local ou Web. Esta aplicação pode ser utilizada em treinamento à distância ou em aulas práticas nos cursos de Engenharia que têm a Metrologia como disciplina.

3.1. O MODELO

Com o objetivo de fazer com que o AVD-W tenha uma abrangência ampla em relação a sua utilização, foi desenvolvida uma camada de protocolos apresentada na Figura 36, permitindo que possa ser implementada de maneira a oferecer suporte a diferentes tipos de serviços como: interação, navegação e comandos necessários para cada tipo de ambiente virtual. A aplicação VCMM utiliza a arquitetura cliente-servidor, implementando somente a transmissão de comandos, ou seja, somente os movimentos permitidos nos Eixos X, Y, Z e nas

Pontas A e B do Apalpador, que são manipulados por comandos a partir da interface tipo “Console” ou “Scrollbar’ que serão replicados nas cenas dos usuários.

Este desenvolvimento permite que o projeto da aplicação se torne modular, possibilitando a implementação e o teste de cada uma das partes e serviços, facilitando a geração dos casos de teste e depuração. De outro lado pode introduzir problemas de sincronização e latência identificadas em outras aplicações, que requerem especialmente o uso de som e imagem conjugados, mas não no caso da VCMM onde esta questão não é significativa.

Neste sentido, visando simplificar o desenvolvimento do protótipo, a renderização da interação e navegação na cena por um usuário, não será replicada para os demais usuários conectados ao protótipo. A ausência desta funcionalidade não implicará na utilização e no resultado do protótipo, em função de que todos os usuários estarão visualizando cenas distintas, mas com o mesmo posicionamento dos Eixos e Pontas da máquina.

Desta maneira, observamos que desacoplamento entre as camadas proporciona uma melhor implementação de controles necessários para cada tipo de aplicação em um ambiente virtual.

Outra característica do Protocolo de Realidade Virtual Distribuído (PRVD) é que ele encapsula todo o controle de envio e recebimento de mensagens entre a aplicação e seus respectivos serviços, abstraindo a utilização de diferentes tipos de protocolos de comunicação. Isso significa que o desenvolvedor não necessita conhecer nem se preocupar com detalhes de implementação, para utilizar uma comunicação via Socket ou RMI em sua nova aplicação.

Esta flexibilidade parte do princípio que uma mesma aplicação possa ter comportamentos diferentes, a partir de determinadas situações ou necessidades de uso, de maneira que possam ser facilmente reconfiguradas. Tais mudanças de comportamento podem requerer modificação da arquitetura da aplicação ou mesmo em sua comunicação.

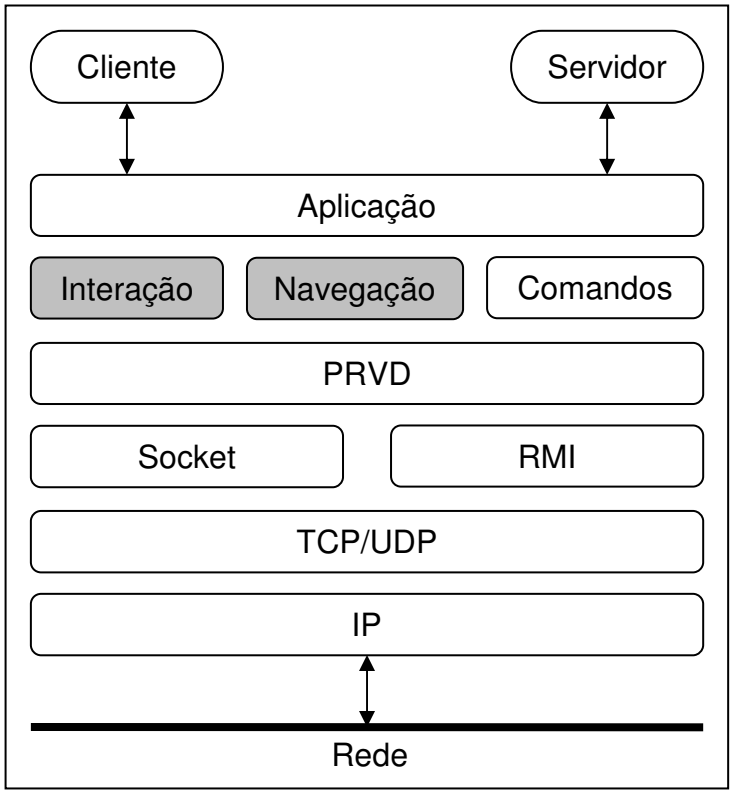


Figura 36 – Camada de protocolos.

Para ilustrar este tipo de necessidade temos como exemplo, a aplicação VCMM para apresentar resultados de execução diferentes, conforme os diagramas de caso de uso apresentado nas Figuras 37, 38 e 39. Esses casos de uso podem ser configurados a partir da *Applet*, através de parâmetros de configuração que determinam qual o tipo de comunicação, interação e visualização serão utilizados.

O Caso de Uso A apresentado na Figura 37, ilustra a situação em que o ator Professor interage com a cena através de comandos da VCMM, que serão transmitidos através da rede e visualizados pelo ator Aluno e Professor após sua renderização. Este modelo apresenta somente a interação da cena pelo ator Professor. Exemplo: Professor apresentando para seus Alunos como programar comandos CMM.

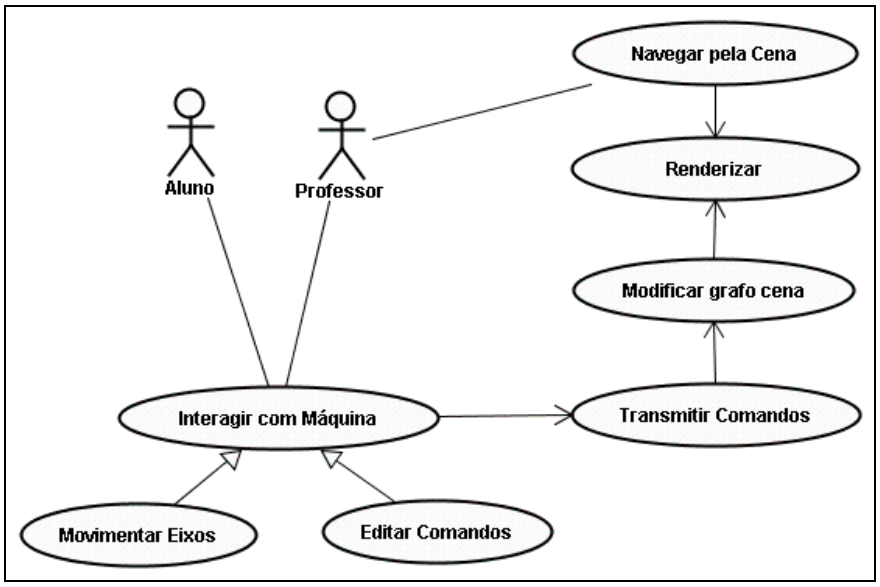


FIGURA 37 – CASO DE USO A.

O Caso de Uso B apresentado na Figura 38, ilustra a situação em que o ator Aluno interage com a cena através de comandos VCMM, que serão transmitidos através da rede e visualizados pelo ator Professor e Aluno após sua renderização. Este modelo apresenta somente a interação da cena pelo ator Aluno. Exemplo: Alunos programando comandos CMM, sendo monitorados pelo Professor.

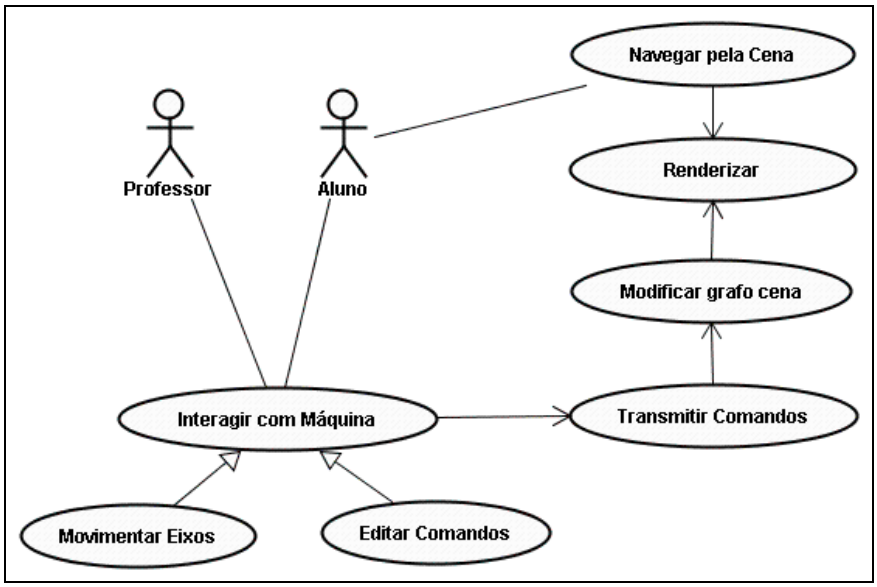


FIGURA 38 – CASO DE USO B.

O Caso de Uso C apresentado na Figura 39, ilustra a situação em que o ator Aluno e Professor interagem e visualizam a cena através de comandos da VCMM simultaneamente, apresentando um modelo de colaboração entre ambos. Exemplo: Estudo em grupo envolvendo Alunos e Professor em uma mesma cena.

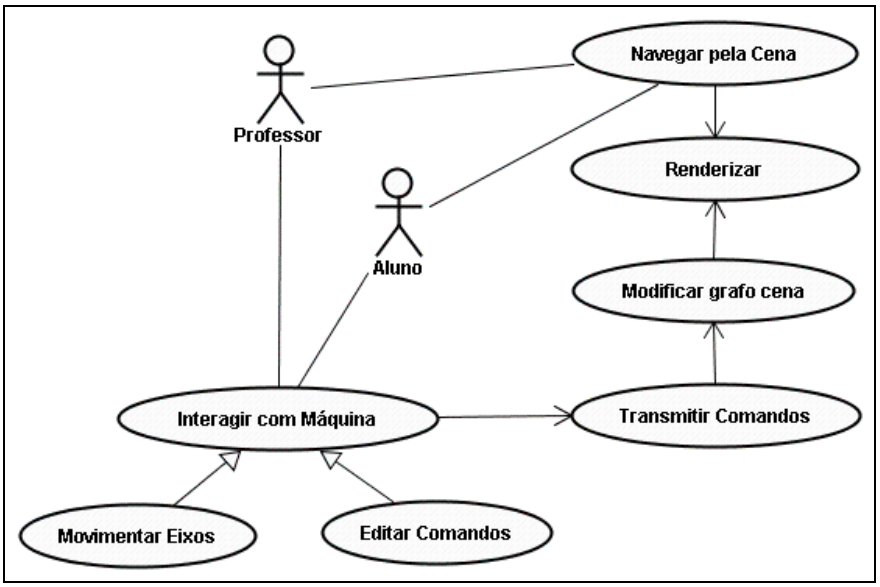


FIGURA 39 – CASO DE USO C.

Os casos de uso apresentados temos os usuários identificados como Aluno e Professor, onde cada um deles podem assumir papéis distintos em cada um dos casos de uso, podendo interagir e ou visualizar a cena da aplicação.

A Figura 40 ilustra o ambiente utilizado para o desenvolvimento e implementação do protótipo AVD-W com base em uma rede local, utilizando uma conexão por cabo de par trançado categoria 5 do tipo *loopback* de 100Mbs, sendo suas principais aplicações:

- **Servidor de comandos:** aplicação responsável por gerenciar a troca de mensagens entre os clientes, que encapsula todo o controle do protocolo de comunicação Socket e RMI, e os controles de movimentação inerentes aos comandos VCMM descritos na Tabela 2 e analisando-os;

- **Renderizador:** responsável por carregar e gerenciar as atualizações da cena, que está sendo compartilhada com outros clientes pelo servidor de comandos, esta aplicação também permite que o usuário possa ter acesso a navegação na cena VCMM.

- **Cliente:** permite que o usuário possa interagir com os comandos da VCMM, enviando-os para o servidor de comandos.

As aplicações descritas acima serão detalhadas na seção 3.2, onde serão apresentadas as configurações permitidas e suas respectivas utilizações no ambiente virtual.

FIGURA 40 – AMBIENTE DE DESENVOLVIMENTO DO PROTÓTIPO.

O computador a esquerda da Figura 40 está executando a aplicação SceneApplet permitindo que o usuário visualize e navegue na cena VCMM, enquanto o notebook localizado a direita envia os comandos VCMM através da aplicação ConsoleApplet.

3.2. O PROTÓTIPO DO AMBIENTE VIRTUAL DISTRIBUIDO

A organização das classes permitiu a elaboração do projeto para comunicação utilizando RMI ou Socket, apresentados a seguir através de diagramas de classes em *Unified Modeling Language* (UML) (BOOCH, 1998).

A Figura 41 apresenta uma visão macro do diagrama de classe da aplicação VCMM, apresentando inicialmente as principais classes da aplicação e sua finalidade, posteriormente apresentando a implementação do projeto utilizando a transmissão de comandos com o modelo RMI e Socket.

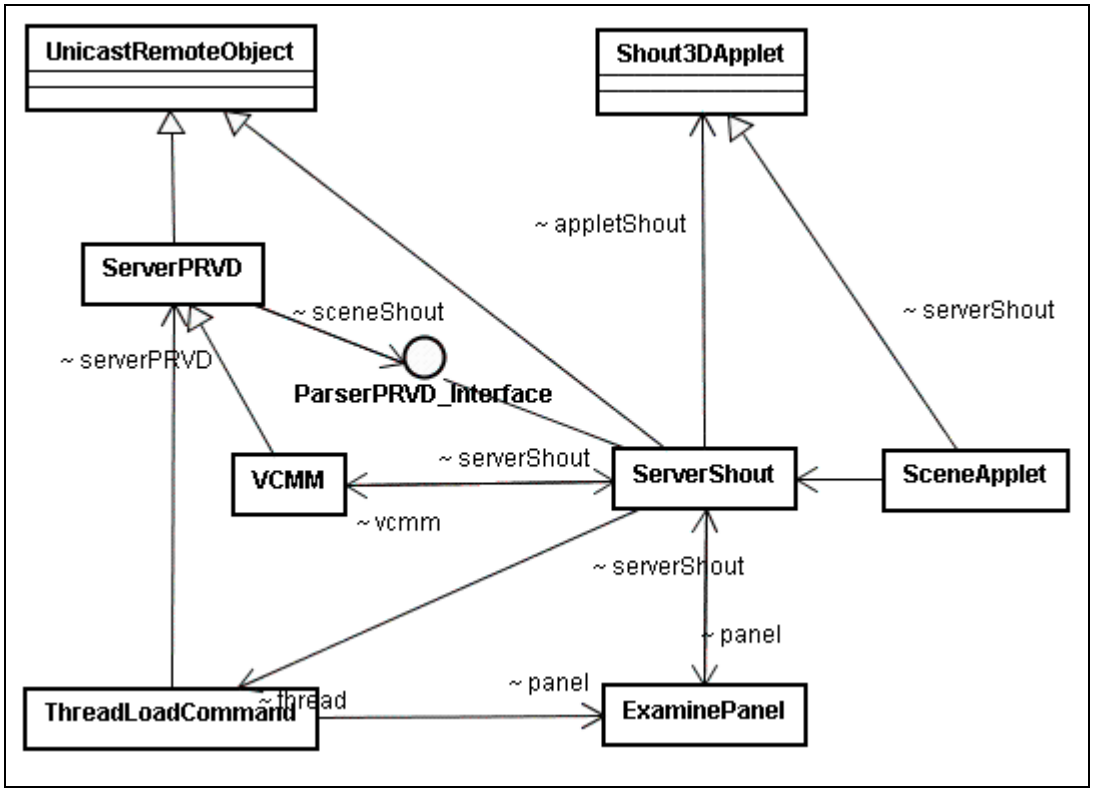


FIGURA 41 – DIAGRAMA DE CLASSES DA APLICAÇÃO VCMM.

A partir de *applets* distintas pode-se iniciar a aplicação SceneApplet que permite a visualização da cena VCMM, e também a aplicação ConsoleApplet que permite a interação dos comandos da VCMM, conforme ilustra a Figura 41.

A classe SceneApplet tem o objetivo de carregar e renderizar uma cena VRML previamente construída, permitindo que o usuário possa interagir, navegar, rotacionar, mover e manipular a cena utilizando dispositivos convencionais.

Por ser uma aplicação tipo *applet* sua chamada é realizada a partir de um arquivo HTML SceneApplet.html. Este arquivo necessita que alguns parâmetros sejam configurados tais como: nome do arquivo de cena VCMM.WRL, qual tipo de comunicação a ser utilizada RMI ou Socket, conforme ilustrado na Figura 30 da seção 2.7.

O pacote Shout3D através da classe Shout3DApplet renderiza o grafo de cena criado a partir do código VRML armazenado no arquivo VCMM.WRL, sendo responsável por todo o controle de renderização dos comandos de interação, rotação, translação, navegação e manipulação de cena e de câmera na *applet*.

A especialização da classe SceneApplet a partir da classe Shout3DApplet permitiu criar uma aplicação do tipo *applet*, que implementa o controle de transmissão de comandos por RMI ou Socket, que pode ser configurado na inicialização da *applet* na aplicação.

A classe ExaminePanel tem a finalidade de criar um *painel*, onde será exibida e manipulada a cena pela *applet*, de maneira similar ao modo *Examine* do VRML. Para permitir a transmissão e atualização dos comandos VCMM pela rede e a renderização na cena, foi necessário alterar o código da classe ExaminePanel originalmente implementado pela biblioteca Shout3D, criando um novo construtor ExaminePanel(Shout3Dapplet,int,int,int,int,String) que permiti-se instanciar o objeto ServerShout, quando o tipo de comunicação utilizado for Socket. Novos métodos também foram criados como loadPRVD_Command e renderPRVD_Command e alterando o método onPostRender.

O método loadPRVD_Command declarado *synchronized*, cria uma região de exclusão mútua de acesso aos dados do objeto, para garantir o recebimento de novos comandos enviados pela classe ThreadLoadCommand, que adiciona os comandos recebidos em um buffer de comandos do tipo vetor serializando os as informações recebidas.

O método renderPRVD_Command é responsável por retirar o primeiro comando do buffer de comandos e os enviar para o objeto ServerShout, para que este possa ser renderizado.

A alteração do método onPostRender permite que após cada renderização da cena controlada pela biblioteca Shout3D o método loadPRVD_Command possa ser executado controlando a serialização dos comandos.

A classe ServerShout é responsável por implementar os métodos set_translation(String, float) e set_rotation(String, float) que executam os comandos apropriados da biblioteca Shout3D, que permitem a alteração da cena e consequentemente sua renderização. Desta maneira, esta classe funciona como um tradutor dos métodos implementados na aplicação e os comandos utilizados pela biblioteca Shout3D, permitindo que novas funcionalidades para manipulação de objetos e cenas 3D possam ser implementadas nesta classe, utilizando recursos da biblioteca gráfica.

A classe VCMM é responsável por executar a análise léxica dos comandos VCMM implementados, utilizando o método runShoutCommand(String) que recebe uma *string* contendo o comando a ser analisado. Essa *string* é separada em Tokens, que são padrões de caracteres com um significado específico do código fonte. Neste caso, os comandos da linguagem de comando numérico do VCMM, descritos na Tabela 2 controlam o deslocamento dos eixos conforme descrito na semântica e nas demais restrições existentes na linguagem CMM.

Cada Token gerado a partir do método StringTokenizer é comparado com os comandos válidos da linguagem VCMM, tais como: mx, my, mz, rx, ry e reset. Ao encontrar um token válido, o respectivo método do comando implementado é chamado. O método que implementa o comando VCMM é responsável por estabelecer os limites físicos de funcionamento da máquina virtual, de acordo com as características da máquina real.

TABELA 2 – CONTROLE DE COMANDO DOS EIXOS DA VCMM.

Sintaxe	Semântica
Reset	Colocar os eixos na posição inicial
mx param	move o eixo x até a posição "param"
my param	move o eixo y até a posição "param"
mz param	move o eixo z até a posição "param"
ra param	rotaciona o eixo a até a posição "param"
rb param	rotaciona o eixo b até a posição "param"

A classe ServerPRVD implementa o serviço de transferência de comandos entre o cliente e o servidor, oferencedo suporte de comunicação via RMI ou Socket.

A classe ConsoleApplet é responsável por iniciar a interface da aplicação que permite a edição de comandos VCMM ou movimentação dos eixos e pontas da máquina virtual. Por também se tratar de uma aplicação tipo *applet* sua chamada é realizada a partir de um arquivo HTML ConsoleApplet.html, em que necessita ser configurado o parâmetro de comunicação a ser utilizado RMI ou Socket para executar a aplicação. A função da classe ConsoleApplet é enviar os comandos do usuário para um servidor de comandos VCMM.

Para permitir a comunicação por RMI, foi criada a interface ParserPRVD_Interface que estende a interface Remote do pacote java.rmi, identificando que este objeto pode ser acessado remotamente. Os métodos remotos da interface ParserPRVD_Interface descrevem quais métodos que a classe cliente pode utilizar para interagir com o objeto remoto no servidor, sendo os métodos descritos abaixo:

- put_command: tem a finalidade de executar um comando VCMM;
- set_translation: cuja função é executar a translação de um nó da cena, em uma determinada coordenada X, Y, Z;
- set_rotation: tem a função de executar a rotação de um nó da cena, em uma determinada coordenada X, Y, Z.

Apresentamos a seguir os diagramas de seqüência utilizados nos modelos de comunicação Socket e RMI. Seu detalhamento permite a descrição do funcionamento de suas classes, a ordem em que os objetos são criados e a seqüência de mensagens trocadas entre eles. Todas as classes apresentadas neste protótipo tratam qual é o tipo de comunicação adotado no início da aplicação permitindo seu uso em ambos os casos.

Modelo utilizando Socket

O diagrama de seqüência utilizado no modelo por Socket ilustrado na Figura 42, exibe usuário Professor iniciando a aplicação SceneApplet para visualizar a cena VCMM, podendo também ser iniciado pelo usuário Aluno, conforme os casos de uso exibidos na Figuras 37, 38 e 39.

Para que a aplicação funcione utilizando a comunicação por Socket, é necessário que a linha de parâmetro “Communication” na *applet*, esteja configurada com o valor “Socket”, conforme segue o exemplo: **<param name="Communication" value="Socket">**.

Ao ser carregada a classe SceneApplet e constatado que o tipo de comunicação Socket será utilizado, é criado o objeto ExaminePanel passando os seguintes parâmetros ao seu construtor: objeto Shout3Dapplet que originou a applet; sua posição inicial em x e y medidos em pixels; sua largura e altura em pixels e o tipo de comunicação.

O construtor da classe ExaminePanel instancia o vetor commandBuffer, responsável pela serialização dos comandos que serão recebidos pela classe ServerShout, que também é instânciada pelo construtor da classe ExaminePanel para que os comandos possam ser renderizados na cena.

Ao ser instânciado o objeto ServerShout utilizando Socket seu construtor cria os objetos VCMM e ThreadLoadCommand, para que possa ocorrer a análise e transmissão dos comandos pela rede. O construtor da classe VCMM ao ser executado cria um servidor Socket devido a classe VCMM ser derivada da classe ServerPRVD, que implementa o servidor ServerSocket que vai aguardar o recebimento dos comandos enviados pela rede.

O objeto instânciado ThreadLoadCommand declarado como thread terá a função de ler os comandos enviados pela rede ao servidor de comandos ServerPRVD e enfileirá-los no vetor de comando commandBuffer. Para que isso ocorra, o construtor da classe ExaminePanel inicia a execução da *thread* ThreadLoadCommand, utilizando o método loadPRVD_Command para adicionar cada comando recebido pelo servidor de comandos na fila commandBuffer.

Durante a execução do objeto ExaminePanel o método onPostRender executa o método loadPRVD_Command que retira cada comando armazenado na fila do vetor commandBuffer para que possa ser renderizado, e utilizando o método renderPRVD_Command para enviar cada comando VCMM através do método runShoutCommand para a classe ServerShout. Quando a mensagem de comando chega ao método runShoutCommand na classe VCMM, sua sintaxe é analisada de acordo com os parâmetros da Tabela 2, e o método correspondente ao comando é executado, podendo utilizar um dos métodos set_translation ou set_rotation disponíveis na classe ServerShout para executar a renderização da cena utilizando os comandos da biblioteca Shout3D.

Desta forma, quando utilizamos o modelo de comunicação por Socket a aplicação que visualiza a cena no caso a classe SceneApplet executa a função de renderizador e servidor de comandos VCMM.

Por outro lado, temos também o Professor iniciando a aplicação ConsoleApplet para interagir com comandos da VCMM, que também pode ser iniciado pelo usuário Aluno conforme os casos de uso exibidos na Figuras 37, 38 e 39.

Para que a aplicação ConsoleApplet funcione utilizando a comunicação por Socket é necessário que a linha de parâmetro “Communication” na *applet*, esteja configurada com o valor “Socket”, conforme segue o exemplo: **<param name="Communication" value="Socket">**, e também seja especificado qual tipo de interface será utilizada para interagir com os comandos VCMM.

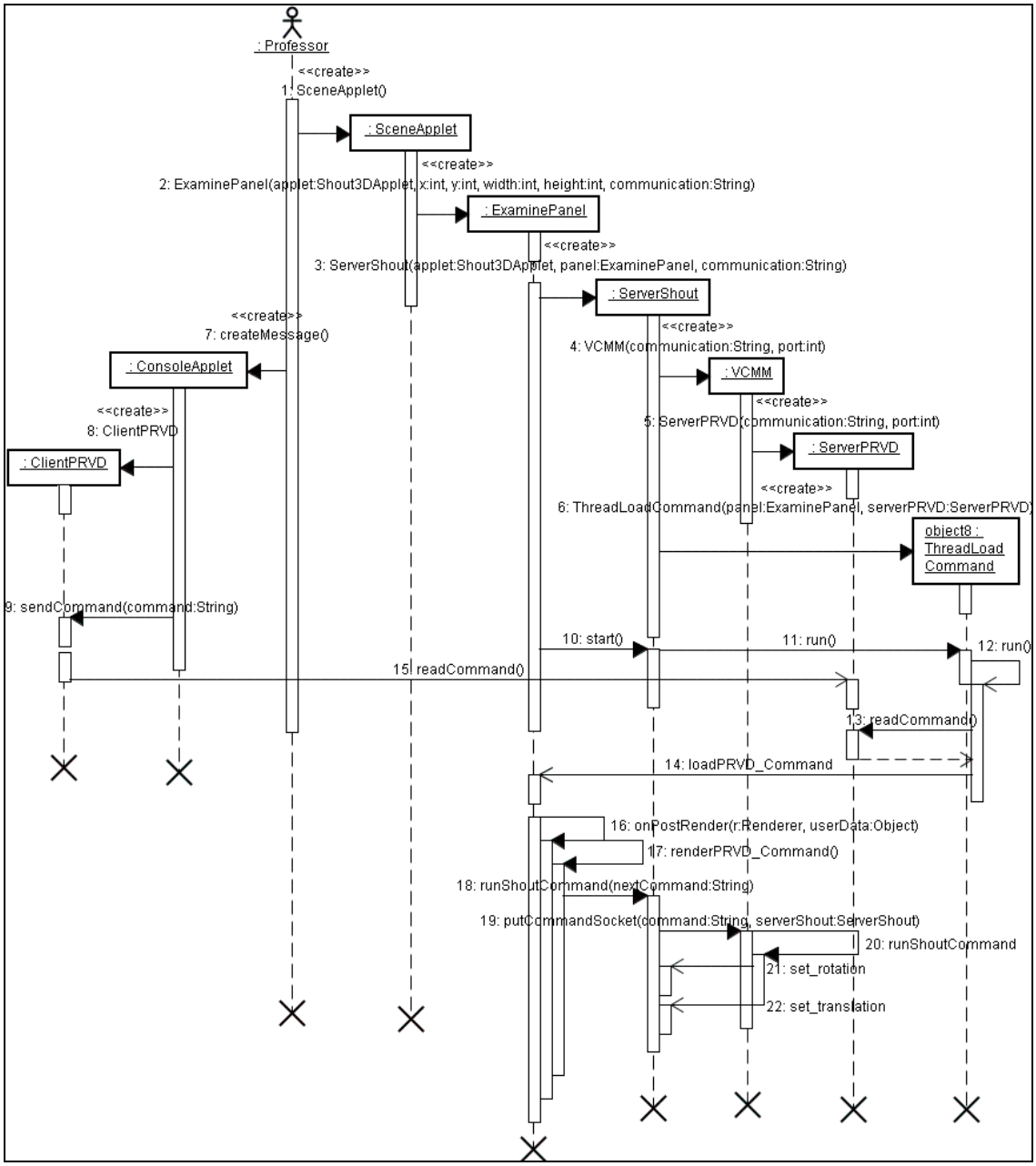


FIGURA 42 – DIAGRAMA DE SEQÜENCIA SOCKET.

As interfaces permitidas são “Console” e “Scrollbar”, onde a interface “Console” permite que os comandos VCMM sejam digitados pelo usuário conforme ilustra a Figura 45. Na interface “Scrollbar” o usuário pode interagir com os comandos utilizando as barras de rolagem de cada eixo e ponta respectivamente, conforme ilustra a Figura 46. A configuração da linha de parâmetro “Interface” na *applet*, pode ser configurada conforme o exemplo: **<param name="Interface" value="Console">**.

Ao ser carregada a classe ConsoleApplet primeiro é verificado qual o tipo de interface será utilizada, depois é constatado qual o tipo de comunicação utilizado, neste caso por Socket. Então é criado o objeto ClientPRVD responsável por estabelecer a conexão com servidor ServerPRVD, para que os comandos do usuário possam ser encaminhados para o servidor de comandos através do método sendComand e consequentemente renderizado. Desta maneira a aplicação ConsoleApplet atua como cliente da classe SceneApplet.

Modelo utilizando RMI

O diagrama de seqüência utilizado no modelo RMI ilustrado na Figura 43, exibe usuário Professor iniciando a aplicação SceneApplet para visualizar a cena VCMM, podendo também ser iniciado pelo usuário Aluno conforme os casos de uso exibidos na Figuras 37, 38 e 39.

Para que a aplicação funcione utilizando a comunicação por RMI, antes de iniciar é necessário que o utilitário **rmiregistry.exe** esteja sendo executado em pelo menos um computador que esteja conectado a rede da aplicação, caso a aplicação esteja sendo executada em um único computador este também necessita estar executando o **rmiregistry.exe**. A linha de parâmetro “Communication” na *applet*, deve estar configurada com o valor “RMI”, conforme segue o exemplo: **<param name="Communication" value="RMI">**.

Ao ser carregada a classe SceneApplet e constatado que o tipo de comunicação RMI está sendo utilizado é instânciado o objeto ExaminePainel para realizar o controle e a renderização da cena.

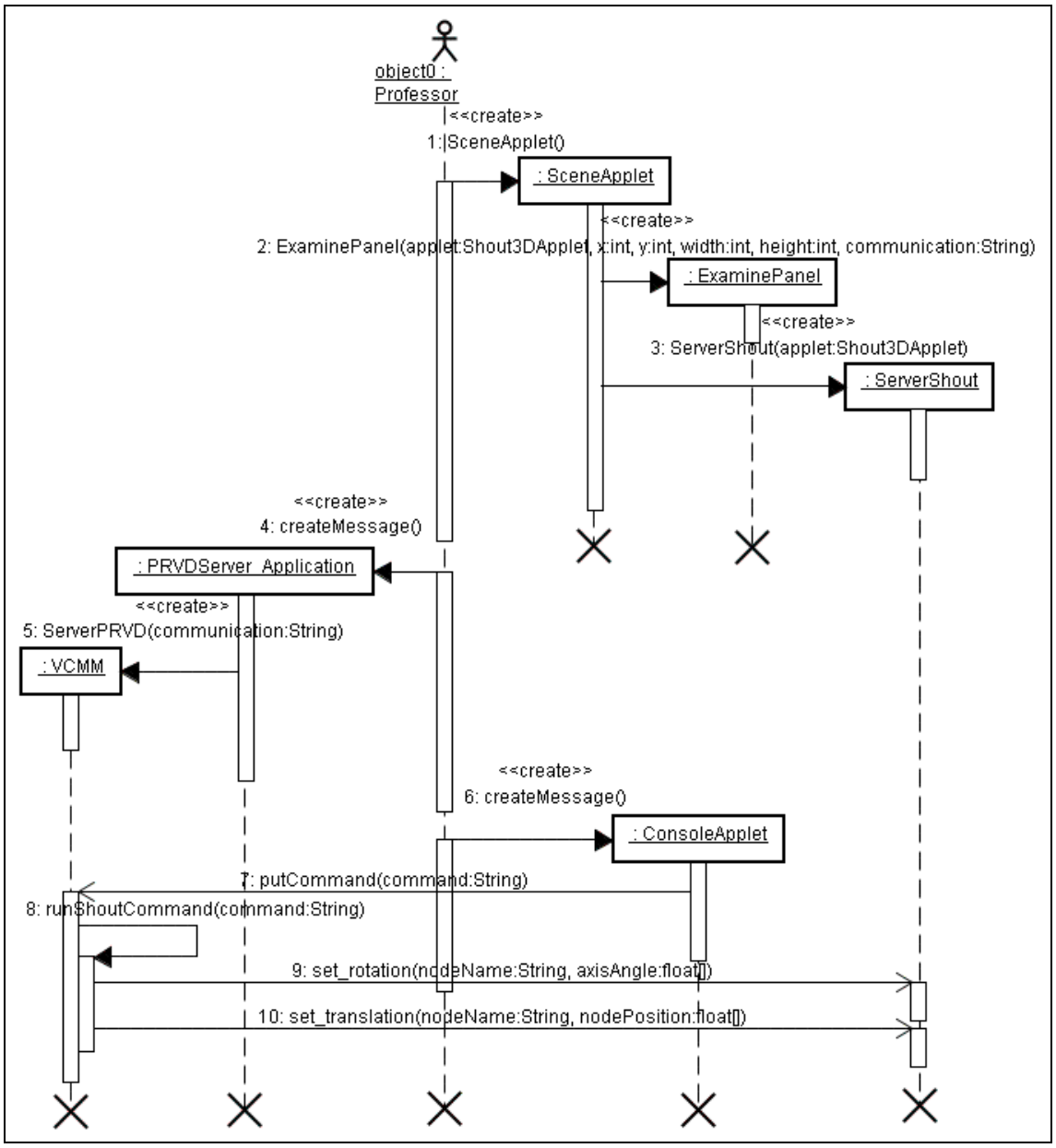


FIGURA 43 – DIAGRAMA DE SEQÜÊNCIA RMI.

Em seguida é instânciado o objeto ServerShout passando ao seu construtor a *applet* que originou a aplicação. Devido a classe ServerShout ser derivada da classe UnicastRemoteObject e implementar os métodos da interface ParserPRVD_Interface, podemos manipular a cena através dos comandos remotos enviando-os para a *applet* da biblioteca Shout3D que originou a aplicação, ou seja, fazer com que a classe ExamineApplet faça a renderização da cena.

Após executar a aplicação SceneApplet, o usuário inicia a aplicação PRVDServer_application sua finalidade é receber os comandos enviados pelos clientes que estarão interagindo com interface de comandos VCMM, verificar sua sintaxe através da classe VCMM e encaminhar os comandos para o servidor ServerShout a fim de que a cena seja renderizada.

A comunicação utilizando RMI ocorre através da interface ParserPRVD_Interface, que é iniciada a partir da classe cliente ConsoleApplet, que envia o comando do usuário através do método remoto putCommand que é cliente RMI da classe servidora VCMM. A classe VCMM por sua vez realiza as mesmas verificações e utiliza os mesmos métodos para executar os comandos VCMM apresentado anteriormente no modelo Socket. A diferença é que utilizando RMI, ao invés dos comandos serem enviados através de pacotes pela rede, eles serão executados remotamente na classe ServerShout, a partir dos métodos set_rotation e set_translation renderizando a cena diretamente através do objeto ExaminePanel.

A classe ConsoleApplet é responsável por implementar o método ActionListner para capturar os comandos e eventos gerados pelo usuário a partir da interface da aplicação, ela instancia a interface ParserPRVD_Interface como cliente RMI do ServerPRVD, ou seja, da classe VCMM que a implementa. A interface ParserPRVD_Interface define quais os comandos do protocolo PRVD que podem ser executados, proporcionando a comunicação utilizando RMI e a renderização da cena em conjunto com a classe ExaminePanel.

Desta forma, para executar o protótipo utilizando o modelo RMI é necessário que as três aplicações SceneApplet, PRVDServer_Application e ConsoleApplet sejam iniciadas para que a solução funcione.

3.3. PLATAFORMA DE DESENVOLVIMENTO

O desenvolvimento deste ambiente ocorreu de maneira gradativa, começando com a análise dos modelos de comunicação com Sockets e RMI para dar suporte a aplicações para a rede e Internet. Esses modelos impuseram

restrições quanto a comunicação utilizando *applets,* devido às políticas de segurança implementadas pelos *browsers* e pelo ambiente de execução da máquina Java.

As possibilidades de uso do pacote Shout3D como renderizador e interface da cena, apresentaram taxas de renderização entre 38 e 40 frames por segundo, durante o período em que foram executados testes com sucesso em ambientes Windows, Linux e Macintosh, utilizando os seguintes equipamentos:

- Desktop AMD Duron 850Mz, 128Mb, Vídeo S3 Trio3D/2X 8Mb, Windows 2000 SP3, Internet Explorer 6.0.2800.1106 SP1, Java Plug-in 1.4.0_01 e Linux Mandrake 9.1;

- Macintosh PowerPC G4 800Mhz, 128Mb, Mac OS X versão 10.3, browser Safari 1.1, Java Plug-in 1.3.1 e Java 1.4.1;

- Notebook Dell Latitude C400, Pentium III 1.2Ghz, 256Mb, Video Intel82830 48Mb, Windows 2000 SP3, Internet Explorer 6.0.2800.1106 SP1, Java Plug-in 1.4.0_01;

- Notebook Toshiba Satellite A45-S1202, Celeron 2.80GHz, 512Mb, Video Intel82852 64Mb, Windows XP Home Edition SP1, Internet Explorer 6.0.2800.1106.xpsp2.030320-1720, Java Plug-in 1.4.2;

- Notebook Dell Latitude D610, Intel Pentium Mobile 1.60Ghz, 512Mb, Video Intel 915GM 128Mb, Windows XP Professional Version 2002 SP2, Internet Explorer 6.0.2900.2180.xpsp2_gdr.050301-1519.

Vários softwares Open Source foram utilizados durante a elaboração do protótipo.

A aplicação foi desenvolvida utilizando a linguagem de programação Java versão 1.4.2_05. Tendo como ambiente Integrado de desenvolvimento para a linguagem Java, o Eclipse Platform como *Integrated Development Environment* (IDE) versão 2.1.2 Build id: 200311030802.

O software JUDE UML Modeling Tool, nos auxiliou a criar os diagramas de caso de uso e de classes, possibilitando exportá-los no formato .bmp. No início do projeto utilizamos um outro software chamado ArgoUML e Poseidon, outros softwares como *plug-in* VRML e editores foram descritos no decorrer deste trabalho.

4. CONSIDERAÇÕES FINAIS

A modularidade adotada na construção do projeto, mostra a portabilidade oferecida pelo AVD-W, que permite implementar diferentes tipos de controle de cena, tornando a construção e adequação da interface do usuário, portável para diferentes tipos de aplicações.

Utilizando este modelo que associa o uso de camadas de protocolos, modularização de classes, obteve-se um alto grau de flexibilidade, conseguindo manter independente a camada de comunicação da camada de renderização da aplicação, permitindo também sua utilização em diferentes plataformas de computadores.

A Figura 44 ilustra a interface do usuário que proporciona a visualização da cena e navegação. Os controles de interação da máquina são apresentados na Figura 45 e 46. A opção pela interface “Scrollbar”, ou seja, por barra de rolagem permitem movimentar cada um dos Eixos e Pontas descritas na Figura 35(b). Os movimentos de rolagem das barras são codificados em comandos que são transmitidos para a máquina. A Figura 45 mostra a opção por console de edição de comandos CMM, opcional às barras de rolagem.

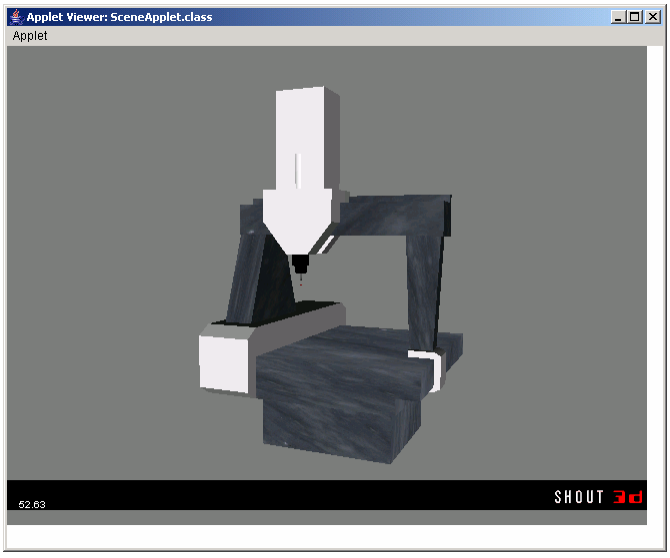


FIGURA 44 – CLIENTE VISUALIZANDO A CENA A PARTIR DO SCENEAPPLET.

A aplicação pode apresentar resultados de execução diferentes, conforme os casos de uso das Figuras 37, 38 e 39, que podem ser configurados a partir de suas *applets*. Os parâmetros de configuração da *applet* determinam qual o tipo de comunicação por Sockets ou RMI e o tipo controle de interação que poderão ser utilizados.

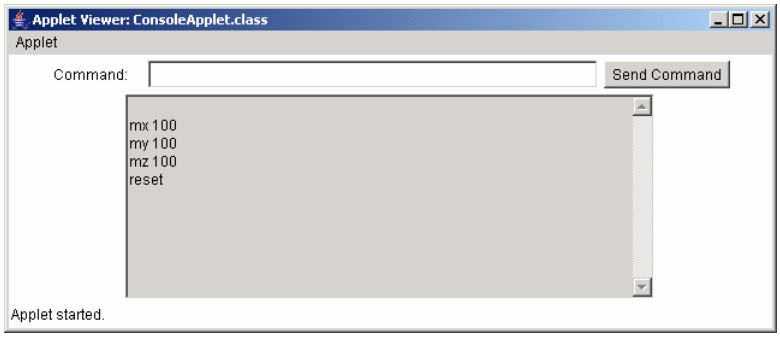


FIGURA 45 – CLIENTE INTERAGINDO POR COMANDOS NO CONSOLEAPPLET.

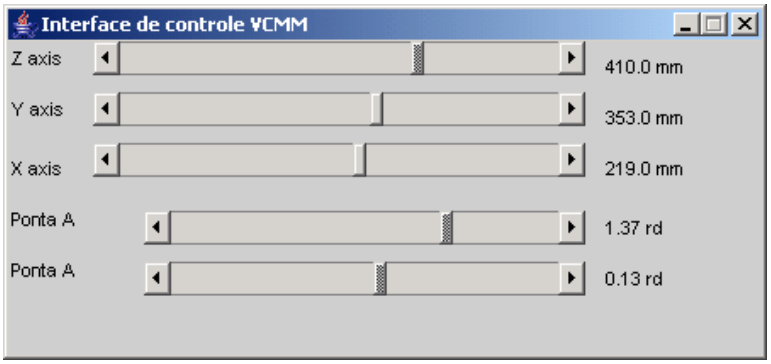


FIGURA 46 – CLIENTE INTERAGINDO POR BARRAS DE ROLAGEM NO CONSOLEAPPLET.

O desenvolvimento utilizando o modelo RMI e Socket, permitiu constatar que o encapsulamento no controle de fluxo, serialização são alguns dos fatores que tornaram o uso do RMI mais abstrato e simples, quando comparando com o modelo Socket. Pois, quando iniciadas as modificações no código para implementar o controle de comunicação por Socket, a necessidade de implementar controles de fluxo, serialização, sincronização de objetos e uso de threads demonstraram o quanto são importantes tais conhecimentos, para o desenvolvimento e construção de uma aplicação distribuida.

Cabe lembrar que o escopo deste trabalho não inclui questões referentes à performance de comunicação considerando a utilização de RMI ou Socket, visto que em muitos dos ambientes virtuais distribuídos analisados, alguns se utilizam um modelo híbrido de controle de comunicação com RMI e Socket.

A questão sobre o controle de interpolação dos comandos VCMM nos Eixos também não foi abordada neste trabalho, devido existir diferentes possibilidades para a implementação do controle de interpolação dos pontos nos Eixos, fazendo com que as características de tolerâncias da máquina possam ser simuladas e oferecendo uma fidelidade mais precisa da máquina real, através do uso de tolerâncias.

O controle de interpolação poderia ser realizado no servidor VCMM, fazendo com que todas as máquinas virtuais conectadas a ele, realizassem a interpolação ao mesmo tempo. A princípio ocasionaria dois problemas distintos: o primeiro seria a quantidade de mensagens colocadas na rede para renderizar um comando, podendo causar uma lentidão em sua transmissão, dependendo da velocidade da rede e dos computadores conectados; o segundo problema seria que computadores mais rápidos vão renderizar a cena mais rapidamente que computadores mais lentos, fazendo com que a cena não seja exibida simultaneamente em todos os computadores.

Quanto às características de controle de interpolação, simultaneidade no controle da cena, quantidade de usuários, diversidade de equipamento e outros controles de ambiente virtual são requeridos, os problemas de desenvolvimento ficam mais complexos, conforme relatado na seção 2.9 sobre trabalhos correlatos.

Freqüentemente, as pesquisas envolvendo o desenvolviemnto de ambientes virtuais distribuídos não permitem sua continuidade devido muitas vezes sua arquitetura de hardware e software ser proprietário como HP-UX, SUN-Solaris ou IBM-AIX, ou por serem desenvolvidos em linguagens que não são portáveis para outros tipos de plataformas, podendo desta maneira serem inviabilizados por questões de custos para a alocação de recursos.

Desta maneira, este projeto torna possível a utilização desta aplicação para múltiplas plataformas, proporcionando sua reutilização para as situações descritas em seus casos de usos, possibilitando também seu uso em aplicações correlatas. Observando que os recursos necessários para o seu desenvolvimento e utilização são facilmente encontrados na maioria das configurações de computadores atualmente disponíveis no mercado, e também devido a sua portabilidade, flexibilidade e independência de seus componentes, permitindo um alto desacoplamento entre as camadas de software, comunicação, renderização e controle da aplicação.

Concluímos que devido à natureza evolutiva do software ocorrer através da implementação de novas funcionalidades e melhorias, em que resultam em novas versões do sistema, as pesquisas envolvendo o desenvolvimento do AVD-W para VCMM não ficarão restritas somente nesta versão. Ao contrário, este trabalho permitiu originar novos trabalhos, como o Aplicativo de Realidade Virtual Distribuída utilizando renderização em X3D, desenvolvido por Antonio Mathias Rüdiger Verona que implementou novos comandos de deslocamento e interpolação para a VCMM utilizando a biblioteca X3D e JAVA3D permitindo o uso desta nova tecnologia.

Desta forma, outros trabalhos podem ser originados a partir deste ambiente onde através da alteração ou especialização da classe ServerShout permitiu que o padrão X3D fosse utilizado como renderizador e a especialização da classe VCMM, inseridos novos controles de interpolação.

REFERÊNCIAS BIBLIOGRÁFICAS

ARTMUSEUM. Disponível na Internet em 12 Novembro 2005 http://www.artmuseum.net/w2vr/timeline/Krueger.html.

SNOWDON, D. N. (1994). "AVIARY: Design Issues for Future Large-Scale Virtual Environments." MIT Presence 3(4): 288-308, 1994.

BOOCH, Grady; Rumbaugh, James; Jacobson, Ivar. **The Unified Modeling Language User Guide.** Indianapolis: Pearson Education, 1998.

CADOZ, Claude. **Realidade Virtual.** São Paulo: Editora Ática, 1997.

CALONEGO, Nivaldi Junior; Kirner, Claudio; Kirner, Tereza G.; Abackerli, Álvaro José. **Implementation of a Virtual Environment for Interacting with Coordinate Measuring Machine.** VECIMS 2004 – IEEE International Conference on Virtual Environment Human-Computer Interface and Measurement Systems. Boston, 2004.

CAREY, Rikk; Bell, Gravin. **The Annotated VRML 2.0 Reference Manual.** Addison Wesley, 2000.

CARVALHO, José Eduardo Maluf de. **Introdução às Redes de Micros.** Makron Books, 1998.

CHEN, Bing-Yu, Nishita, Tomoyuki. **JGL and its Applications as a Web3D Platform.** ACM WEB3D'2001, Paderbon Germany, 2001.

COCO, Geoffrey P. **The Virtual Environment Operating System: Derivation, Function and Form.** University of Washington, 1993. Disponível na Internet em 31 Março 2004 http://www.hitl.washington.edu/publications/th-93-1/th-93-1.pdf.

COULOURIS, George; Dollimore, Jean; Kindberg, Tim. **Distributed Systems Concepts and Design.** Third Edition. Addison Wesley, 2001.

COVEN. Disponível na Internet em 31 Março 2004 http://coven.lancs.ac.uk.

DAM, Andries van. **Post-WIMP User Interfaces**. Communication of the ACM, Vol. 40, No. 2, February 1997.

DEITEL, H. M.; DEITEL, P. J. **Java Como Programar.** Abrange Java2 Apresentando Swing. 3ª Edição. Porto Alegre: Bookman, 2001.

DIVE. Swedish Institute of Computer Science (SICS). Disponível na Internet em 27 Março 2004 http://www.sics.se/dce/dive/dive.html.

FARLEY, Jim. **JAVA Distributed Computing**. Sebastopol: O'Reilly, 1998.

FERREIRA, Aurélio Buarque de Holanda. **Miniaurélio Século XXI: O minidicionário da língua portuguesa**. 4 Edição. Rio de Janeiro: Nova Fronteira, 2000.

GREENHALGH, Chris. **Supporting Complexity in Distributed Virtual Reality Systems.** Technical Report NOTTCS-TR-96-6, Department of Computer Science, The University of Nottingham, Nottingham, UK, 1996.

GOODRICH, Michael T.; TAMASSIA, Robert. **Estruturas de dados e algoritmos em Java.** Porto Alegre: Bookman, 2002.

HANCOCK, D. **Viewpoint: Virtual reality in search of middle ground.** IEEE Spectrum. September, 1994.

HORSTMANN, Cay S.; Cornell, Garry. **Core Java Volume I – Fundamentals.** Palo Alto: Sun Microsystems Press, 1999.

IPOLITO, Juliano Ribeiro. **AVIT-Ambiente Virtual Interativo Tridimensional, Uma Plataforma Configurável para Desenvolvimento de Ambientes Virtuais Interativos Multi-Usuários.** Universidade Federal de São Carlos, 1999.

JACOBSON, Robert. **After the virtual reality gold rush: the virtual world paradigm.** Computer Graphics, Vol. 17, No. 6, 1993. Pág. 695-698.

JACOBSON, Robert. **Information Design.** Cambridge: MIT Press, 2000.

KEN, Arnold; Gosling, James; Holmes, David. **The Java Programming Language.** Third Edition. Boston: Addilson Wesley, 2002.

KIRNER, Claudio; Ipólito, Juliano R. **Projeto de Ambientes Virtuais Multi-Usuários Usando Java e VRML.** 3rd WRV 2000 - Workshop on Virtual Reality, Gramado Brasil, Outubro 2000.

KIRNER, Claudio; Mendes, Sueli B. T. **Sistemas Operacionais Distribuídos; Aspectos Gerais e Análise de sua Estrutura.** Rio de Janeiro: Editora Campus, 1988.

KIRNER, Claudio; Pinho, M. S. **Uma introdução à Realidade Virtual,** 2000. Disponível na Internet em 17 Maio 2003 http://grv.inf.pucrs.br/Pagina/TutRV/tutrv.htm.

KIRNER, Claudio. ***VESIV-Virtual Environment for Shared Interactive Visualization***, CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico ASCIN - Assessoria de Cooperação Internacional, Convênios Bilaterais, Convênio/País: Programa de Cooperação Brasil-Alemanha em Tecnologia da Informação, Número de Processo: 690012/02-0 ASCIN/CNPq, Título do Projeto: *VESIV-Virtual Environment for Shared Interactive Visualization*, Coordenador do Lado Brasileiro: Prof. Dr. Claudio Kirner. 2003.

KIRNER, Claudio; Tori, Romero. **Realidade Virtual: conceito e tendências**. São Paulo: Editora Mania de Livro, 2004.

KIRNER, Claudio. **Sistemas de Realidade Virtual.** Disponível na Internet em 13 Maio 2003 http://www.dc.ufscar.br/~grv/tutrv/tutrv.htm.

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

LANIER, Jaron. Bibliografia. Disponível na Internet em 12 Outubro 2005 http://www.advanced.org/jaron.

LATTA, J. N.; Oberg, D. J. **A conceptual virtual reality mode,** IEEE Computer Graphics & Applications, Pág. 23-29, January 1994.

LEVY, Pierre. **O que é Virtual?** São Paulo: Editora 34, 1996.

LOFTIN, R. B.; Kenney, P. J. **Training the Hubble space telescope flight team,** IEEE Computer Graphics and Application, Pág. 31-37, September, 1995.

MASSIVE. Disponível na Internet em 31 Março 2004 http://www.crg.cs.nott.ac.uk/research/systems/MASSIVE-3.

MORIE, J. F. **Inspiring the future: merging mass communication, art, entertainment and virtual environment.** Computer Graphics, Pág. 135-139, May 1994.

NETTO, Antonio Valerio; Machado, Liliane dos Santos; Oliveira, Maria Cristina F. **Realidade Virtual: Fundamentos e Aplicações.** Florianópolis: Editora Vistual Books, 2002.

NETTO, Antonio Valerio; Machado, Liliane dos Santos; Oliveira, Maria Cristina Ferreira de. **Realidade Virtual - Definições, Dispositivos e Aplicações.** Disponível na internet em 19 Março 2003 http://www.sbc.org.br/reic/edicoes/2002e1/tutoriais/RV-DefinicoesDispositivosEAplicacoes.pdf.

NPSNET. Disponível na Internet em 31 Março 2004 http://www.npsnet.org/~npsnet/v/.

PICARD, Stéphane Louis Dit; Degrande, Samuel; Gransart, Christophe; Chaillou, Christophe; Saugis, Grégory. **Comunication Platform for Synchronous Collaborative Virtual Environmnet**. 2001. Disponível na Internet em 01 Abril 2004 http://www.lifl.fr/~louisdit.

PINHO, Márcio Serolli. **Um Modelo de Interface para Navegação em Mundos Virtuais.** Pontifícia Universidade Católica do Rio Grande do Sul. Disponível

na Internet em 17 Maio 2003 http://grv.inf.pucrs.br/Pagina/Publicacoes/Bike/Portugues/Bike.htm.

POLEVOI, Rob. **Interactive Web Graphics with Shout3D.** Sybex, 2001.

PRAVEEN. **Myron Krueger**. Disponível na Internet em 21 Junho 2003 http://bubblegum.parsons.edu/~praveen/thesis/html/wk05_1.html.

RODRIGUES, Silviane G.; Oliveira, Jauvane C., Peixoto, Marcos V. **ADVICE: Um Ambiente VIrtual Colaborativo para o Ensino a Distância**, 2003. Disponível na Internet. Em 31 Março 2004 http://www.lncc.br/~jauvane/papers/WTD_Webmidia_VersaoFinal.pdf.

SAAR, Kurt. **VIRTUS: A Collaborative Multi-User Platform.** ACM VRML99, Pág. 141-152, Paderborn Germany, 1999.

SANTOS, Gildasio Mendes dos. **A realidade do virtual.** Campo Grande: Editora UCDB, 2001.

Sense8. **World ToolKit.** Disponível na Internet em 27 Março 2004 http://www.sense8.com.

SHERMAN, R. William; Craig, B. Alan. **Understanding Virtual Reality; Interface, Application and Design.** San Francisco: Morgan Kaufmann, 2003.

SUN. **Java Tutorial.** Disponível na Internet. Em 05 Maio 2003 http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html.

TANENBAUM, Andrew S.; Steen, Maarten Van. **Distributed Systems; Principles and Paradigms.** New Jersey: Prentice Hall, 2002.

TENENBAUM, Aaron M.; LANGSAM, Yedidyah, AUGENSTEIN, Moshe J. **Estruturas de dados usando C.** São Paulo: Makron Boolks, 1995.

THOMAS, Michael D.; Patel, Pratik R.; Hudson, Alan D.; Ball, Donald A. Jr. **Programando em Java para Internet.** São Paulo: Makron Books, 1997.

TRAMBEREND, Henrik. **Avango: A Distribuited Virtual Reality Framework.** GMD – German National Research Center for Information Technology, 2000. Disponível na Internet em 29 Março 2004 http://www.avango.org.

VINCE, John; EARNSHAW, Rae. **Virtual Worlds on the Internet**. IEEE, 1998.

VRML. **VRML97 Functional specification and VRML97 External Authoring Interface (EAI) International Standard ISO/IEC 14772-1:1997, ISO/IEC 14772-2:2002.** Disponível na Internet em 17 Maio 2003 http://www.web3d.org/fs_specifications.htm.

ZIMMERMAN, Thomas. Home Page. Disponível na Internet em 12 Outubro de 2005. http://www.almaden.ibm.com/cs/people/zimmerman/tzim.html.

ZOTTINO, Ricardo; Calonego, Nivaldi Junior, Abackerli, Álvaro Jóse, Kirner, Tereza Gonçalves; Kirner, Claudio. **Ambiente Virtual Distribuído Web para Máquinas de Medir por Coordenada utilizando Shout3D e RMI.** SVR 2004 VII Symposium on Virtual Reality, São Paulo, 2004. Pág. 255 – 264.

ZOTTINO, Ricardo; Calonego, Nivaldi Junior; Abackerli, Álvaro José; Coletta, Carlos Eduardo Della. **Acesso via Internet para Máquinas de Medir por Coordenadas.** WRA'2004 I Workshop sobre Realidade Aumentada. Piracicaba, 2004.

WEB3D. **Web3D.** Web3D Consortium, 2003. Disponível na Internet em 11 Maio 2005 http://www.web3d.org.

WHYTE, Jennifer. **Virtual Reality And The Built Enviroment.** Oxford: Architectural Press, 2002.

X3D. **Extensible 3D (X3D).** International Draft Standards. Disponível na Internet em 11 Outubro 2005 http://www.web3d.org.

REFERÊNCIAS CONSULTADAS

AHO, Alfred V., **Compiladores Princípios, Técnicas e Ferramentas.** Rio de Janeiro: LTC, 1995.

NEWMAN, Alexandre. **Usando Java o Guia de Referência Mais Completo.** Rio de Janeiro: Editora Campus, 1997.

BENEDIKT, Michael L. **Cyberspace: First Steps.** Cambridge: MIT Press,

BLAXXUN. **Blaxxun.** Blaxxun Interactive, Inc. Disponível na Internet em 17 Maio 2003 http://www.blaxxun.com/products/blaxxun3d.

BROWN, Kirk; Petersen, Daniel. **Ready-to-Run Java 3D.** New York: John Wiley & Sons, 1999.

BURDEA, Grigore; Coiffet, Philippe. **Virtual Reality Technology.** New York: John Wiley & Sons, 1994.

COLLIN, Perter. **Dicionário de Informática, Multimídia e Realidade Virtual.** Companhia Melhoramentos, 2001.

CORTONA, **Cortona Jet.** Parallel Graphics, Inc., 2003. Disponível na Internet em 17 Maio 2003 http://www.parallelgraphics.com/products/jet.

CRAIG, Larman. **Utilizando UML e Padrões Uma Introdução a Analise e ao Projeto Orientado a Objetos.** Porto Alegre: Bookman, 2000.

DEITEL, H. M.; DEITEL, P. J. **Java Como Programar.** Apresentando Projeto Orientado a Objetos com UML e Padrões de Projetos. 4ª Edição. Porto Alegre: Bookman, 2004.

DELAMARO, Marcio Eduardo. **Como construir um compilador: utilizando ferramentas Java.** São Paulo: Novatec, 2004.

DROZDEK, Adam. **Estrutura de dados e algoritmos em C++.** São Paulo: Thomson, 2002.

EARNSHAW, R. A.; Vince, J. A.; Jones, H. **Virtual Reality Applications.** Loundon: Academic Press, 1995.

FERNANDES, Tatiana Cavalcanti, Sztajnberg, Alexandre. **Aspectos de Comunicação em uma Aplicação Distribuída de Realidade Virtual Utilizando Java e VRML.** 3rd WRV 2000 - Workshop on Virtual Reality, Gramado Brasil, Outubro 2000.

FOLEY, James D. **Computer Graphics: Principles and Practice.** Second Edition in C. São Paulo: Addison-Wesley, 1996.

FUNKHOMER, Thomas A. **RING: A Client-Server System for Multi-User Virtual Environments.** AT&T Bell Laboratories. ACM 1995 Symposium on Interactive 3D Graphics, Monterey California, USA, 1995.

FURGERI, Sergio. **Java2 Ensino Didático.** 2ª Edição. São Paulo: Editora Erica, 2001.

FURLAN, José Davi. **Modelagem de objetos através da UML: the UNIFIELD Modeling Language; análise e desenho orientados a objeto.** São Paulo: Makron, 1998.

FURTADO, Antonio Luz. **Teoria dos grafos: algoritmos.** Rio de Janeiro: LTC, 1973.

GORDON, Lescinsky; Costa, Touma; Goldin, Alex; Fudim, Max; Cohen, Amit. **Interactive Scene Manipulation in The Virtue3D System.** ACM Web3D'2002, Tempe Arizona, USA. February 2002.

HAMIT, Francis. **Realidade Virtual e a exploração do espaço Cibernético.** São Paulo: Berkeley, 1993.

HILL JUNIOR, Francis S. **Computer Graphics** / **Using OPENGL**. 2ª Edição. New Jersey: Prentice-Hall, 2001;

LEVINE, Roberti I. **Inteligencia Artificial e Sistemas Especialistas.** São Paulo: McGraw-Hill, 1988.

LOFTIN, Bowen; Chen, Jin X.; Rizzo, Skip; Goebel, Martin; Hirose Michitaka. **Proceedings IEEE Virtual Reality 2002.** California: IEEE, 2002.

MATOS, Alexandre Veloso de. **UML Prático e Descomplicado.** 3ª Edição. São Paulo: Editora Erica, 2003.

MELLO, Rodrigo; Chiara, Ramon; Villela, Renato. **Aprendendo Java2.** São Paulo: Novatec, 2002.

PARENTE, André. **Imagem Máquina: A Era das Tecnologias do Virtual.** Rio de Janeiro: Editora 34, 1999.

COMPUTAÇÃO, Sociedade Brasileira de. **Proceedings of 5 Symposium on Virtual Reality – V SVR.** Edited by VIDAL, Augusto, Creto; Kirner, Cláudio. Fortaleza: SBC, 2002.

PAGE-JONES, Meillir. **Fundamentos do Desenho Orientado a Objetos com UML.** São Paulo: Makron, 2001.

PRESSMAN, Roger S. **Engenharia de Software.** 5ª Edição. Rio de Janeiro: McGraw-Hill, 2002.

RUMBAUGH, James. **Modelagem e Projetos Baseados em Objetos.** Rio de Janeiro: Campus, 1994.

ROBERTSON, George; Czeminski, Mary; Dantzich, Maarten van. **Immersion in Desktop Virtual Reality.** ACM UIST 97, BanJJ Alberta, Canada, 1997.

SEBESTA, Robert W. **Conceitos de Linguagens de Programação.** 5ª Edição. Porto Alegre: Bookman, 2000.

SOMMERVILLE, Ian. **Engenharia de Software.** 6ª Edição. São Paulo: Addilson Wesley. 2003.

SCRIMGER, Rob. **TCP/IP, A Bíblia.** Rio de Janeiro: Campus, 2002.

SZWARCFITER, Jayme Luiz. **Grafos e algoritmos computacionais.** Rio de Janeiro: Editora Campus, 1988.

STANNEY, Kay M. **Handbook of virtual environments: design, implementation, and applications.** New Jersey: Lawrence, 2002.

WALNUM, Clayton. **Java em Exemplos.** São Paulo: Makron Books, 1997.

WHITE, Arthur T. **Graphs, Groups and Surfaces.** Amsterdam: North-Holland, 1973.

VIEIRA, Marcelo Domingos Barreto. **Ensaios em Realidade Virtual.** Faculdade de Ciência e Tecnologia. Universidade de Cuiabá, 2001.

Apêndice A. ARTIGOS PUBLICADOS

Este apêndice apresenta os artigos publicados durante o desenvolvimento deste trabalho no período de 2003 a 2004 nos seguintes eventos:

A.1. Acesso via Internet para Máquina Virtual de Medir por Coordenadas. In: I WORKSHOP SOBRE REALIDADE AUMENTADA, 2004, Piracicaba. I Workshop sobre Realidade Aumentada. Piracicaba: UNIMEP - Universidade Metodista de Piracicaba, 2004. v. 1, p. 41-44.

A.2. Ambiente Virtual Distribuido Web para Máquinas de Medir por Coordenadas Utilizando Shout3D e RMI. In: SVR2004 VII SYMPOSIUM ON VIRTUAL REALITY, 2004, São Paulo. SVR2004 VII Symposium on Virtual Reality. SBC - Brazilian Computer Society, 2004. v. 1, p. 255-264.

A.3. Ambiente Virtual Distribuido Web - AVDW. In: 2ª MOSTRA ACADÊMICA UNIMEP, 2004, Piracicaba. 2ª Mostra Acadêmica UNIMEP. Gráfica Unimep, 2004. p. 9-9.

A.1. ACESSO VIA INTERNET PARA MÁQUINA VIRTUAL DE MEDIR POR COORDENADA

Acesso via Internet para Máquina Virtual de Medir por Coordenadas

Ricardo Zottino, Nivaldi Calonego Junior, Álvaro José Abackerli, Carlos Eduardo Della Coletta
Faculdade de Ciências Matemáticas, da Natureza e da Terra
UNIMEP - Universidade Metodista de Piracicaba
Programa de Pós Graduação em Ciência da Computação
Caixa Postal 68 – 13.400-901 – Piracicaba – SP – Brasil
zottino@zipmail.com.br, ncalonego@unimep.br, abackerli@unimep.br, ccoletta@widesoft.com.br

Abstract

This article discusses the Virtual Coordinates Measurement Machine support for Internet application programs. Firstly, a synthesis on related works is present, reviewing different Virtual Distributed Environments models, and its influences on the system architecture. This discussing comprehend in a dedicate communication protocol and portable for multiple platforms, that will be responsible to manage the data exchange in a Collaborative Virtual Environment Distributed.

Introdução

A evolução das pesquisas no campo da realidade virtual durante a década de 90 propiciaram o surgimento de vários ambientes virtuais, distribuídos e colaborativos, que na sua maior parte está atualmente implementada em C++. Essa opção é devido à limitação dos recursos de hardware e de software existentes que induzem ao uso de plataformas proprietárias, tais como: SUN, HP-UX, IBM RISC, Silicon Graphics. A evolução tecnológica dos computadores pessoais, o barateamento de dispositivos de interface gráfica e o crescimento da rede mundial de computadores criaram as condições necessárias à Realidade Virtual Distribuída (RVD) via Internet. Atualmente, os processadores embutidos nos computadores pessoais oferecem a performance necessária para o processamento gráfico, e os meios de comunicação oferecem largura de banda necessária para garantir taxas de comunicação suficiente para que essas aplicações a Internet. Neste contexto, emerge a criação de uma camada de protocolo portável para múltiplas plataformas que faça uso das tecnologias Java, Shout3D, VRML ou X3D para a implementação de Realidade Virtual Distribuída para a Internet.

Trabalhos Correlatos

O DIVE (Distributed Interactive Virtual Environment) é um sistema de Realidade Virtual Distribuído Interativo, multi-usuário baseado em Internet, que proporciona navegação e visualização espacial 3D, possibilitando a interação de usuários e aplicações. Desenvolvido na Suécia pela Swedish Institute of Computer Science (SICS), sua primeira versão surgiu em 1991. A interação de um usuário provoca a difusão do evento na rede, com comunicação multi-cast, não possuindo servidor centralizado. O compartilhamento de estado é análogo à memória compartilhada, onde um conjunto de processos interage através de acessos concorrentes a essa "memória compartilhada" [6, 13, 5]. Ele é o precursor de outros ambientes, como o Avango, que oferece é um framework para desenvolvimento de ambientes virtuais distribuídos, que utiliza um modelo de programação análogo ao DIVE. O Avango é desenvolvido em C que distribui as informações através da replicação de uma cena para todos os processos que participam da aplicação distribuída [15].

Noutra linha, aparece o SPIN-3D em que se explora a plataforma CORBA/ORB de sistemas distribuídos para implementar o Ambiente Virtual Colaborativo (AVC). Disponibiliza cenas 3D destinadas à colaboração entre pequenos grupos utilizando VRML. A implementação é linguagem C++, com CORBA e ORBacus, utilizando o framework Open Communication Interface (OCI) para implementar o protocolo Multicast Inter-ORB Protocol (MIOP). O trabalho de colaboração entre os

usuários é realizado através da replicação do mundo virtual VRML por um servidor, que faz cópia ou "clone" da cena para cada cliente conectado [10].

O COVEN é uma concepção mais ampla de RVD. Ele se apresenta como derivação do DIVE e do dVS e foi desenvolvido por um consórcio Europeu, no período de 1994 a 1998, objetivando a análise de requisitos e viabilidade de aplicações Computer Supported Cooperative Work (CSCW). Ele utiliza múltiplos servidores para o gerenciamento das comunicações, estes definem espaços de interação diferentes, permitindo comunicação em longas distâncias para a colaboração em ambientes virtuais, e uso de multimídia para reuniões [6, 3].

Outro experimento é o VEOS (Virtual Environment Operation System), implementado em 1993, é uma ambiente de programação C e AutoLISP, desenhado para a prototipação rápida de um Ambiente Virtual Distribuído, em que o estado é armazenado um banco de dados contendo tuplas privadas e publicas. Neste caso, o banco de dados reflete o estado atual do mundo virtual, cujo modelo enfatiza a comunicação assíncrona e a distribuição baseada em entidades [6, 2, 13].

A questão da persistência de um mundo virtual é o foco do NPSNET. Em desenvolvimento desde 1993, sua versão atual é o NPSNET-V, que implementa um componente Java para o desenvolvimento de aplicações cliente-servidor, ponto-a-ponto ou produtos standalone. Seu objetivo é disponibilizar uma plataforma capaz de implementar no "cyberspace" a persistência de um mundo virtual que não necessita ser desativado para manutenção ou upgrade do sistema, buscando qualidades como extensibilidade de conteúdo e aplicações, escalabilidade em mundos complexos e com grande número de participantes e composição heterogênea de conteúdo e aplicações [6, 13, 9].

Os trabalhos citados anteriormente representam concepções diferentes para a implementação de ambientes para o desenvolvimento de aplicações em Realidade Virtual. Projetos como o MASSIVE [8] e WorldToolKit [14] continuam em aprimoramento de novas versões. Mas, muitos dos ambientes virtuais colaborativos, distribuídos e frameworks não tiveram continuidade, devido à forte correlação do software com plataformas específicas ou à falta de um planejamento para continuidade do projeto em futuros projetos de pesquisa, ou mesmo abandono do projeto inicial. A pesquisa bibliográfica revela outras implementações, tais como: ADVICE [12], AVIARY [6, 13, 1], AVIT [7], VIRTUS 13], VUE [6].

Os problemas detectados para a elaboração de ambientes virtuais distribuídos podem ser significativamente reduzidos pela delimitação do problema. Neste caso, o desenvolvimento do suporte de uma "Máquina Virtual de Medir por Coordenadas" para acesso via Internet. Dos modelos anteriormente apresentados, será experimentado o modelo de programação cliente-servidor, implementando os mecanismos de comunicação via RMI e Socket.

Arquitetura do Ambiente

Para a concepção da proposta considerou-se: a utilização máxima de recursos Open Source; a adoção de padrões da comunidade científica, para que o projeto possa ser reutilizado e aperfeiçoado em trabalhos futuros; possibilitando seu reuso em aplicações de Realidade Virtual Distribuída via Internet. A visão macroscópica apresentada na Figura 1, ilustra a adoção do modelo cliente-servidor em que as modificações do ambiente virtual estão associadas a diferentes serviços (som, imagem, interação, navegação e comandos), conforme ilustra a Figura 2.

Esse modelo flexibiliza tanto o desenvolvimento quanto o uso. O desenvolvimento permite o projeto modular, possibilitando a implementação e o teste de cada uma das partes e serviços, facilitando a geração dos casos de teste e a depuração. Mas, introduz problemas com sincronismo necessário a determinadas aplicações, especialmente no que tange ao som e imagem conjugados. Mas, no caso da "Máquina Virtual de Medir por Coordenadas - VCMM" essa questão não é significativa.

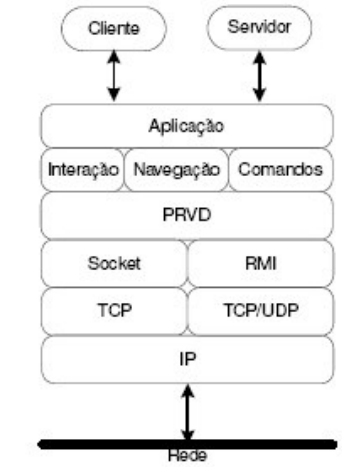


Figura 1 – Camada dos protocolos.

O módulo "Protocolo de Realidade Virtual" (PRVD), usa o arquivo de configurações, Figura 2, para determinar protocolos de comunicação ponto-a-ponto, multi-cast ou broadcast para cada um dos serviços. Assim, diferentes clientes têm a possibilidade de estarem associados a diferentes servidores, permitindo que a interação e os comandos sejam compartilhados, de acordo as configurações definidas.

Os mecanismos de comunicação podem ser configurados por serviço para usar o *Transfer Control Protocol* ou *Datagram Control Protocol* sobre o *Internet Protocol*. Os usuários devem usar o arquivo de configuração para determinar como ocorrerá a execução das atividades tanto no lado dos servidores quanto no lado dos clientes. Essa decisão poder ser útil, por exemplo: (i) para os dispositivos de navegação, dado que a perda de alguns pontos de vista não prejudicará a visualização; (ii) iniciar clientes em máquinas diferentes; (iii) isolar ambientes virtuais diferentes e em execução numa mesma rede.

A definição para a utilização de *Remote Method Invocation* (RMI/JAVA) e *Sockets* como mecanismos de controle e comunicação, proporcionará ao final do projeto, a realização a análise de desempenho comparativa entre dois modelos citados [4].

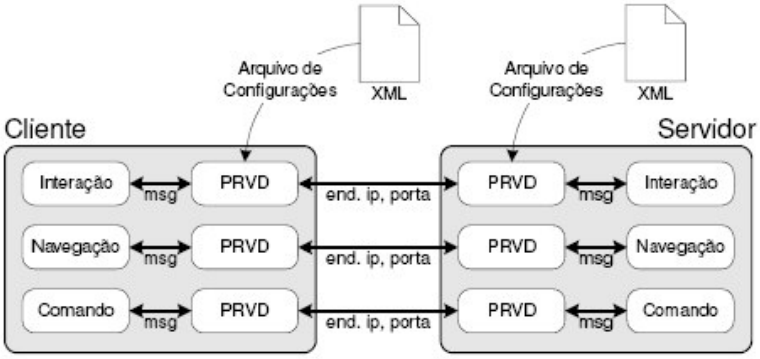


Figura 2 – Diagrama da lógica de serviços

Os fatores considerados críticos para a implementação da VCMM é o número de usuários, número de cenas, objetos e de aplicações ou serviços, e a complexidade dos conhecimentos da Engenharia, necessário à base de transformações que representam a dinâmica do sistema.

Com o intuito de delimitar o escopo deste trabalho e consequentemente diminuir sua complexidade, estaremos abordando a integração de uma quantidade limitada de usuários em um único mundo virtual, correspondendo à quantidade de alunos de uma sala de aula. Mas, a aplicação oferece a possibilidade de uso via Internet, dados que serão implementadas as cenas no formato Shout3D [11], havendo a possibilidade de uso dos formatos X3D [16].

Considerações finais

Essa abordagem apresenta vantagens, por exemplo: (i) o uso de UDP para os dispositivos de navegação, dado que a perda de alguns pontos não necessariamente prejudica a visualização; (ii) iniciar clientes em máquinas diferentes, para explorar recursos específicos, tais como, rastreadores óticos; (iii) isolar ambientes virtuais diferentes em execução na mesma rede. Apesar das possíveis dificuldades de

sincronização, decorrentes do modelo, podem ser discutidas da ótica dos requisitos, que não é o foco deste trabalho.

Referências

[1] AVIARY. http://www.crg.cs.nott.ac.uk/people/Dave.Snowdon/vr/aviary/, 2004.

[2] COCO, Geoffrey P. The *Virtual Environment Operating System: Derivation, Function and Form.* University of Washington, 1993. http://www.hitl.washington.edu/publications/th-93-1/th-93-1.pdf, 2004.

[3] COVEN. http://coven.lancs.ac.uk, 2004.

[4] Deitel, II. M. **Java Como Programar**. Bookman, 2001.

[5] DIVE. Swedish Institute of Computer Science (SICS). http://www.sics.se/dce/dive/dive.html, 2004.

[6] Greenhalgh, Chris. *Supporting Complexity in Distributed Virtual Reality Systems*. Technical Report NOTTCS-TR-96-6, Department of Computer Science, The University of Nottingham, Nottingham, NG7 2RD, UK.

[7] Ipolito, Juliano Ribeiro. *AVIT – Ambiente Virtual Interativo Tridimensional, Uma Plataforma Configurável para Desenvolvimento de Ambientes Virtuais Interativos Multi-Usuários.* Universidade Federal de São Carlos, 1999.

[8] MASSIVE.http://www.crg.cs.nott.ac.uk/research/systems/MASSIVE-3, 2004.

[9] NPSNET. http://www.npsnet.org/~npsnet/v/, 2004.

[10] Picard, Stéphane Louis Dit, Degrande, Samuel, Gransart, Christophe, Chaillou, Christophe, Saugis, Grégory. *Comunication Platform for Synchronous Collaborative Virtual Environmnet*, 2001.
http://www.lifl.fr/~louisdit/, 2004.

[11] Polevoi, Rob. **Interactive Web Graphics with Shout3D**. Sybex, 2001.

[12] Rodrigues, Silviane G., Oliveira, Jauvane C., Peixoto, Marcos V. *ADVICE: Um Ambiente VIrtual Colaborativo para o Ensino a Distância*, 2003.
http://www.lncc.br/~jauvane/papers/WTD_Webmidia_VersaoFinal.pdf, 2004.

[13] SAAR, Kurt. *VIRTUS: A Collaborative Multi-User Platform.* ACM VRML99, Pag. 141-152, Paderborn Germany, 1999.

[14] Sense8.http://www.sense8.com, 2004.

[15] Tramberend, Henrik. Avango: *A Distribuited Virtual Reality Framework*. GMD – German National Research Center for Information Technology, 2000. http://www.avango.org, 2004.

[16] X3D. Extensible 3D (X3D). International Draft Standards. Disponível na Internet.
http://www.web3d.org/fs_specifications.htm, 2003.

A.2. AMBIENTE VIRTUAL DISTRIBUIDO WEB PARA MÁQUINAS DE MEDIR POR COORDENADAS UTILIZANDO SHOUT3D E RMI

Ambiente Virtual Distribuído Web para Máquinas de Medir por Coordenadas utilizando Shout3D e RMI

Ricardo Zottino[1], Nivaldi Calonego Junior[1], Álvaro José Abackerli[2], Tereza Gonçalves Kirner[1], Cláudio Kirner[1]

[1]Faculdade de Ciências Matemáticas, da Natureza e da Terra
[2]Faculdade de Engenharia, Arquitetura e Urbanismo
UNIMEP - Universidade Metodista de Piracicaba
Caixa Postal 68 – 13.400-901 – Piracicaba – SP – Brasil

zottino@zipmail.com.br, {ncalonego, abakerli, tgkirner, ckirner}@unimep.br

Abstract. *This paper presents a Virtual Measurement Machine Coordinate (VCMM) for Web, using Shout3D library as the render. The Virtual Reality Distributed Protocol (PRVD) for Web is implemented using RMI and Socket technology. The purpose of this prototype is to show the application modularization and its components, showing also how this protocol can be reused in applications that implements virtual distributed environment, becoming a communication middleware for applications that need to be portable to different platforms.*

Resumo. *Este artigo apresenta a Máquina de Medir Virtual por Coordenadas (VCMM) para Web, utilizando a biblioteca Shout3D como renderizador, e a implementação do Protocolo de Realidade Virtual Distribuído (PRVD) para Web, desenvolvido com tecnologias RMI e Socket. O objetivo deste protótipo é demonstrar através da modularização da aplicação e de seus componentes, como este protocolo pode ser reutilizado em aplicações que implementam ambientes virtuais distribuídos, tornando-se um middleware de comunicação para aplicações que necessitam ser portáveis para múltiplas plataformas.*

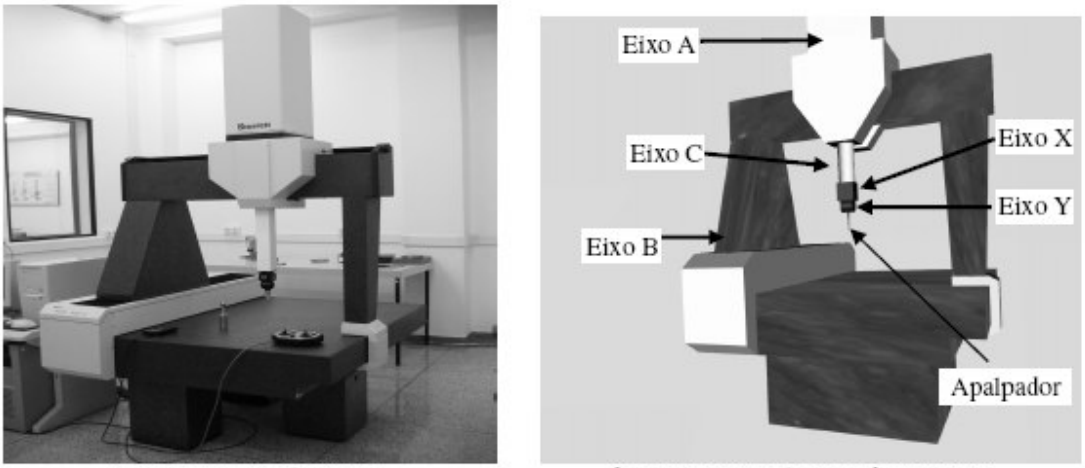
1. Introdução

A Realidade Virtual (RV) inclui tecnologias de interface que exploram canais multi-sensoriais para causar nos usuários a sensação de imersão ao navegar e interagir em um espaço tridimensional gerado por processamento computacional. Ambientes de RV são projetados para atender a essas características, havendo uma organização lógica em termos de subsistemas de simulação, de renderização, de interação e de comunicação, que apresentam um diferencial vantajoso a esse tipo de aplicação quando comparada a outros tipos de programas de interação entre homem e computador. As principais vantagens são, a interação com os ambientes a partir de quaisquer pontos de vista, alterações contínuas e as características de aplicação em tempo real.

Experimentos de implementação de Máquinas de Medir por Coordenadas, usando técnicas de RV, [Calonego, 2003; Kirner, 2003] foram desenvolvidos para ambiente estereoscópio, objetivando o uso no ensino da Metrologia. No entanto observou-se que o software produzido limita o número de usuários, havendo a

necessidade de equipamentos que, apesar de baratos se comparados a outros dispositivos de visão estereoscópica, ainda são caros para muitos usuários. Assim, o objetivo deste artigo é descrever a implementação de um de ambiente de RV aplicado ao ensino da Metrologia, discutindo o suporte para interação local ou remota na utilização da máquina de medir comando numérico em ambiente distribuído Web. O foco é uma Máquina de Medir por Coordenadas (CMM) usada para medidas geométrico-dimensionais, da qual são modelados vinte e um graus de liberdade correspondentes aos erros paramétricos. A Figura 1(a) ilustra a máquina real e a Figura 1(b) ilustra a máquina modelada e os principais componentes.

Essa máquina permite medir a geometria de superfícies com pequena tolerância, oferecendo incertezas da ordem de milésimo de milímetro (μm). Exemplos de geometrias verificadas em tal máquina incluem formas das superfícies, posição de furos ~~buracos~~, distâncias de extremidades, etc. A CMM atende a normas internacionais que definem uma linguagem de programação, denominada linguagem de Comando Numérico, para a definição e execução de tarefas automatizadas [Forbes, 1994].



(a) Máquina Real (CMM) (b) Máquina Implementada (VCMM)

Figura 1 – Ilustração da Máquina de Medir.

A programação de máquinas de comando numérico, como a CMM, requer do operador, conhecimento específico na aplicação, habilidade de programação na linguagem de comando numérico, e também acesso ao ambiente de programação no qual a codificação será testada. Além disso, o processo de medir usado na máquina real depende do comportamento de alguns componentes individuais, tais como a escala de medida, a estratégia de amostragem, as peças (partes a serem medidas), entre outros – posto que um mesmo programa usado para medir uma mesma peça pode apresentar resultados distintos em realizações sucessivas da medição, em função dos erros e incertezas inerentes ao processo.

O ambiente discutido neste artigo cria uma interface de realidade virtual para um simulador de CMM. O simulador gera um conjunto de números que correspondem aos erros encontrados na máquina real. Esses dados permitem a aplicação da simulação aos comandos da "Máquina de Medir Virtual por Coordenadas" (VCMM), oferecendo ao programador resultados visuais que permitem a análise e a validação dos dados

resultantes da aplicação real. A VCMM pode ser programada usando-se uma interface gráfica, cujo objetivo é reduzir a complexidade da tarefa de programação, criar uma interface mais amigável para reduzir os erros léxicos e sintáticos intrínsecos à tarefa de programação.

O projeto privilegiou a flexibilidade de uso da aplicação, definindo uma interface única para o transporte de comandos da máquina através de uma rede de computadores, permitindo o uso da VCMM em ambiente Web ou local. Isto permite que a aplicação seja utilizada em treinamento a distância ou em aulas práticas nos cursos de Engenharia que têm a Metrologia como disciplina.

Para mostrar esse ambiente, optou-se por desmembrar o artigo em sessões, tais que: (i) a sessão 2 discute o modelo orientado a objetos e os fundamentos do ambiente utilizado para a implementação; (ii) apresenta as diferentes plataformas utilizadas na implementação e nos experimentos; (iii) sessão 4 tece as considerações finais.

2. O Modelo

O objetivo de fazer com que o software tenha abrangência ampla em relação ao número de usuários determinou o modelo arquitetural, dado que o programa simulador deve executar remotamente. Esta condição é imposta pela necessidade de haver um banco de dados que armazene parâmetros de simulação e os respectivos resultados dos experimentos. No caso de se ter usuários executando localmente a aplicação completa, há a necessidade de transporte do banco de dados e do simulador para a máquina do usuário, o que dificulta atualizações e a manutenção da consistência do banco. Assim, adotou-se uma concepção em que os usuários são categorizados como "Cliente" e "Professor".

Usuário "Cliente" é um usuário que tem acesso ao programa, aos dados para a simulação e à interface de programação, mas que não tem prerrogativas para efetuar alterações parâmetros pré-definidos e armazenados no banco de dados ou nos modelos de simulação. O usuário "Professor" tem essas prerrogativas, determinando modelos de simulação de máquinas de medir por coordenadas e parâmetros dessas respectivas máquinas, que sejam válidos e que possam ser utilizados por usuários clientes. Assim, a aplicação é projetada para atender às características determinadas pelas interfaces com os usuários, conforme ilustra a Figura 2.

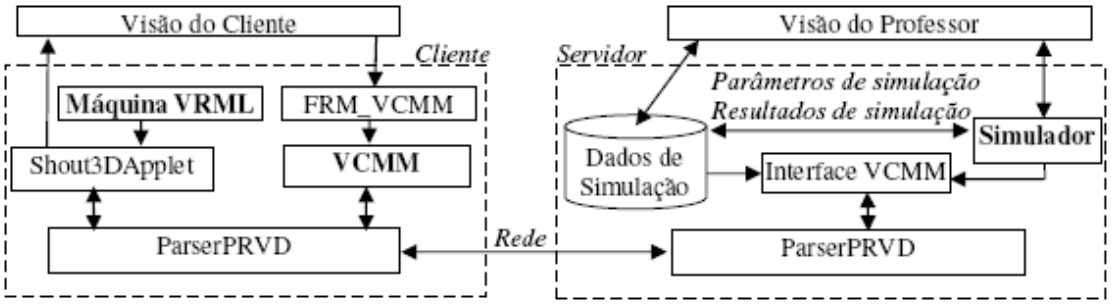


Figura 2 – Modelo lógico da Máquina de Medir Virtual.

Na visão do Cliente, o módulo "FRM_VCMM" é a interface que permite enviar comandos de controle que movimentam os eixos da máquina. Essa interface possui

formas diferentes, havendo a necessidade de invocar o método de execução do respectivo comando implementado no módulo "VCMM", que criar o controle da máquina. A comunicação entre esses módulos é definida por um protocolo que determina o formato do comando a ser executado e os argumentos do respectivo comando. Esses dados são enviados ao módulo "ParserPRVD" no cliente, que implementa os mecanismos de comunicação entre o lado cliente e o lado servidor da aplicação, implementado no módulo "ParserPRVD" no lado servidor.

Na visão do Professor, há dados que chegam por intermédio da rede que contêm os comandos a serem simulados. Assim, o módulo "Interface VCMM" atua como um *Proxy*, recebendo comandos a serem decodificados e enviados ao "Simulador" e recebendo os novos valores dos argumentos dos comandos para enviá-los ao visualizador (módulo "Shout3DApplet"). Shout3D é uma forma de exibir gráficos 3D e animações interativas na Web, sem a necessidade de utilizar aplicações plug-in, exibindo seus gráficos diretamente em um browser através de uma *applet* Java [Deitel, 2001], podendo ser visualizado na maioria dos browsers existentes no mercado que suporte *applet* Java da Sun. Por ser baseado em VRML possibilita exibir cenas 3D sem nenhuma programação e ao mesmo tempo, fornecer um conjunto de ferramentas, ou seja, um pacote em Java que possibilita a interação com o usuário, sendo atualmente comercializado pela Eyematic [Polevoi, 2001], [VRML, 2003], [SUN, 2003].

O módulo Shout3DApplet atua sobre o grafo de cena da VCMM causando no usuário a sensação de estar utilizando a máquina real. Como os resultados do simulador mostram variações da ordem de milésimo de milímetro (μm), invisíveis a olho nu, são necessárias transformações em escala nos respectivos eixos.

A organização destes módulos permitiu a elaboração de projetos para comunicação usando sockets e RMI, modelos que são apresentados nas Figuras 3 e 4, respectivamente, usando diagramas de classes [Booch, 2000]. Os diagramas ilustram a organização lógica das classes oferecendo flexibilidade proporcionando sua reutilização em diferentes ambientes de execução.

2.1 – O uso de Sockets

A classe FRM_VCMM da Figura 3 ilustra as diferentes possibilidades de interface com o Cliente. Um objeto instanciado de VCMM cria a interface com o usuário, e implementa a interface "CommandPRVD" (Comando do Protocolo de Realidade Virtual Distribuída) e que também implementa a interface "ParserPRVD" (Parser do Protocolo de Realidade Virtual Distribuída para a VCMM). A interface "CommandPRVD" encapsula a comunicação de rede, permitindo o uso de diferentes tecnologias de comunicação, sendo testadas duas implementações para este projeto, uma utilizando Sockets e *Remote Method Invocation* (RMI/Java).

A classe SceneApplet tem o objetivo de renderizar a cena VRML, carregada a partir de um servidor de arquivo/web e proporcionar a interação do usuário na cena que está carregada na memória de seu o computador. O pacote Shout3D é responsável pela renderização dos comandos recebidos através da interface PRVD e também pela renderização da interação do usuário na cena.

O ConsoleApplet tem a função de enviar os comandos do usuário para um servidor de comandos PRVD (PRVDServer_applicaton), que fará a validação dos

comandos da máquina e a conversão destes comandos para uma linguagem intermediaria PRVD.

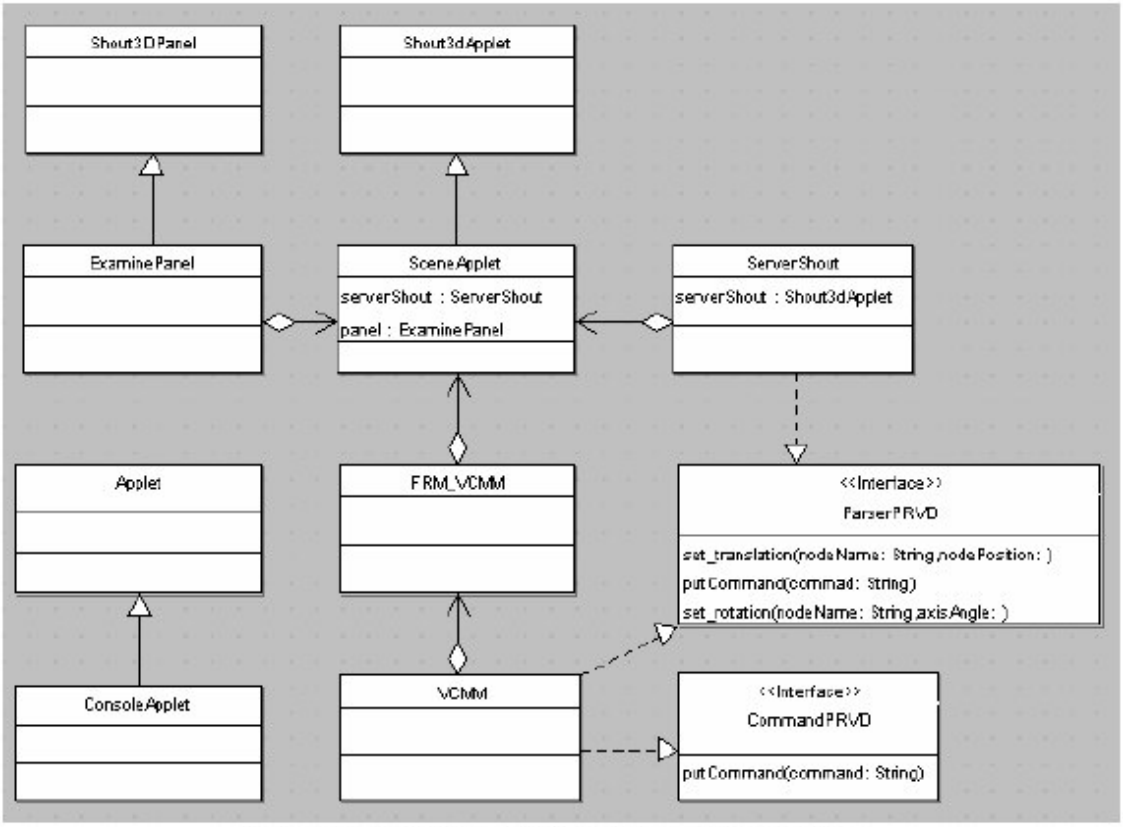


Figura 3 – Diagrama de classes utilizando Socket

A classe PRVDServer_Application funciona como um *Proxy* entre a aplicação de edição de comandos do usuário e a aplicação de renderização de cena "visualização usuário". Seu objetivo é receber os comandos de máquina do usuário, verificar sua sintaxe e convertê-lo para o protocolo PRVD, de maneira que possam ser solicitados e transmitidos para a aplicação de renderização (SceneApplet).

2.2 – O uso de RMI

O diagrama de classes da Figura 4 apresenta o projeto orientado a objetos implementado para Shout3D e RMI no transporte do PRVD. O uso de RMI foi implementado com o objetivo de se manter um servidor centralizado, que sirva a demanda de aplicações em execução via Internet. Mas, em ambientes de ensino em laboratório, há preocupação da apresentação do instrutor/professor. Neste caso, pode ser desejável que as máquinas dos usuários repitam os procedimentos que são demonstrados, fazendo com que haja um único console (máquina do instrutor) e diversos visualizadores (um para cada dado cliente).

A classe SceneApplet, derivada do pacote Shout3Dapplet, implementa o controle da cena através de um objeto tipo painel Shout3Dpanel. Este painel é responsável por todo o controle e renderização da cena, através dos comandos do pacote Shout3D. Em nosso exemplo derivamos o Shout3Dpanel da classe implementada ExaminePanel.java fornecida junto com o pacote Shout3D, que proporciona toda a

interação e controle de rotação, translação e de câmeras da cena, também é instanciando na classe SceneApplet um objeto RMI ServerShout.

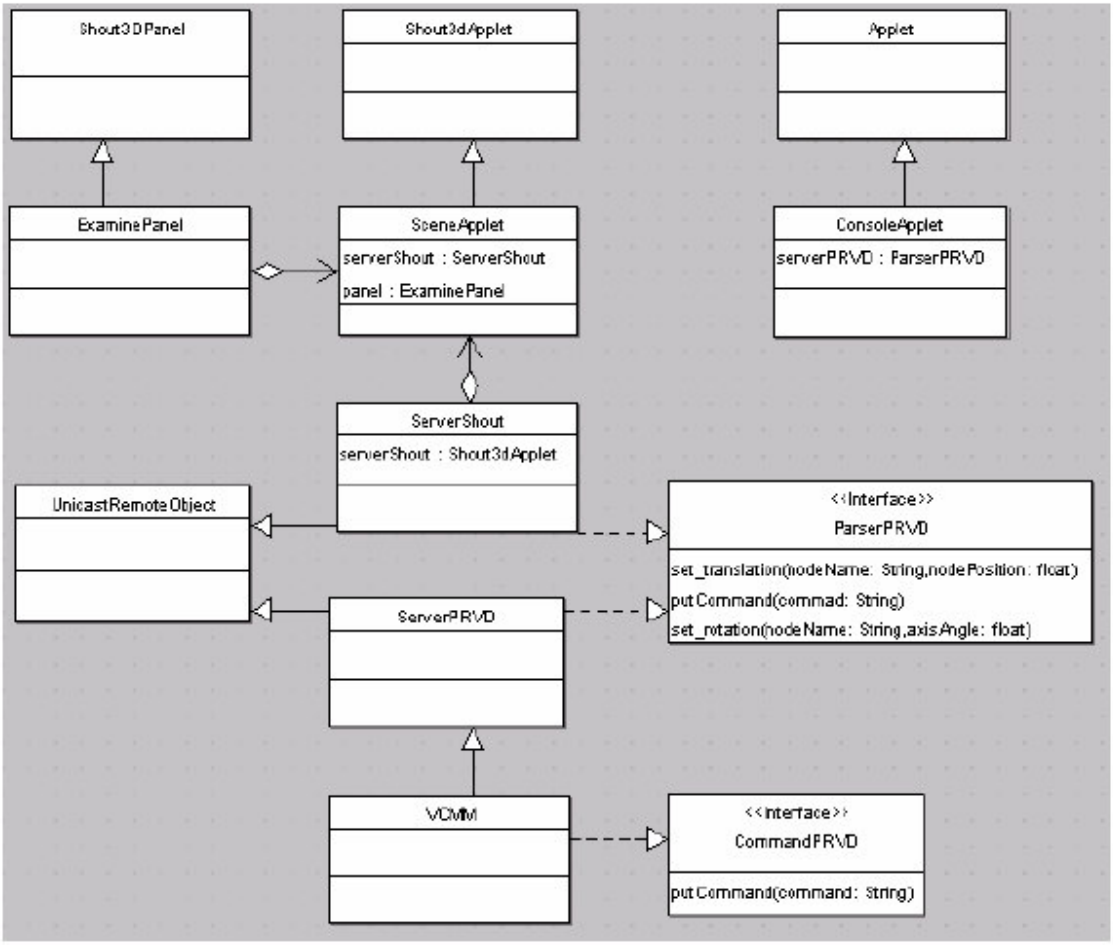


Figura 4 – Diagrama de classes utilizando RMI

A classe ServerShout é derivada de RMI que implementa a interface ParsePRVD, responsável pela conversão das sintaxes de comando PRVD para os comandos Shout3D recebidos de um cliente RMI.

O PRVDServer_Application é uma aplicação que instancia a classe ServerPRVD como um servidor, funcionando como um *Proxy* entre a aplicação ConsoleApplet e a SceneApplet.

ServerPRVD é uma classe derivada de RMI que implementa a interface ParsePRVD, tendo instanciado uma interface ParserPRVD como cliente RMI de ServerShout e instanciando uma interface CommandPRVD como servidor RMI, a classe ServerShout fica sendo responsável pelo recebimento dos comandos de um cliente console através da interface CommandPRVD, converter para PRVD e transmitir para um servidor ServerShout.

A classe VCMM por sua vez é derivada de ServerPRVD e implementa a interface CommandPRVD, responsável por validar e implementar todos os comandos, parâmetros e limites da VCMM recebidos do ConsoleApplet e converter para PRVD.

O ConsoleApplet implementa ActionListener para capturar os comandos e eventos gerados pelo usuário, instanciando uma interface ParserPRVD como cliente RMI do PRVDServer. A interface ParserPRVD define os comandos do protocolo PRVD que devem ser implementados para proporcionar a comunicação entre ServerShout e ServerPRVD.

Nos modelos apresentados nas sessões 2.1 e 2.2 observa-se que a modularidade do projeto, possibilita implementar diferentes controles de cena, tornando a construção e adequação da interface do usuário portável para diversos tipos de aplicações. A Figura 5 ilustra a interface do usuário que proporciona a visão da cena, interação e navegação. Os controles de edição dessa máquina são apresentados na Figura 6(a). As barras deslizantes (A, B, C, X, Y) movimentam cada um dos eixos conforme a denominação utilizada na Figura 1(b). Os movimentos de rolagem das barras são codificados em comandos que são transmitidos para a máquina. A Figura 6(b) mostra o console de edição de comandos CMM, opcional às barras de rolagem.

A aplicação pode apresentar resultados de execução diferentes, conforme os casos de uso das Tabelas 1, 2 e 3, que podem ser configurados a partir da *applet*. Os parâmetros de configuração da *applet* determinam o tipo de comunicação a ser utilizada sockets ou RMI e o tipo controle de edição que poderá ser utilizado.

A Figura 5 mostra o resultado dos movimentos simulados provocados pelos comandos emitidos, utilizando-se as interfaces apresentadas na Figura 6 (a) ou Figura 6 (b). As Tabelas A, B e C ilustram os possíveis casos de uso para esta aplicação para diferentes condições de ensino ou treinamento.

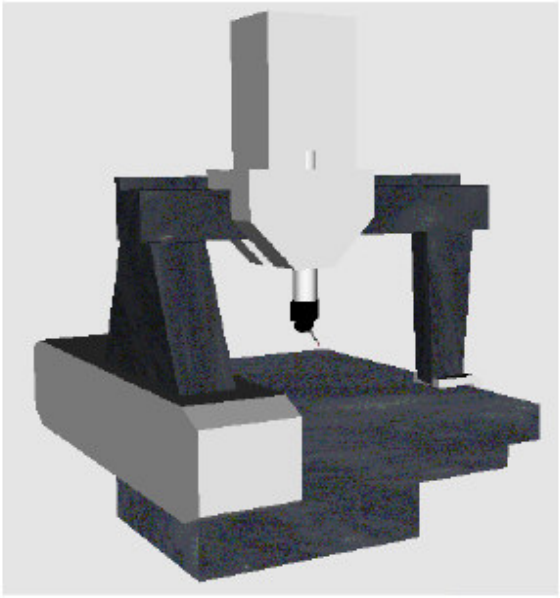
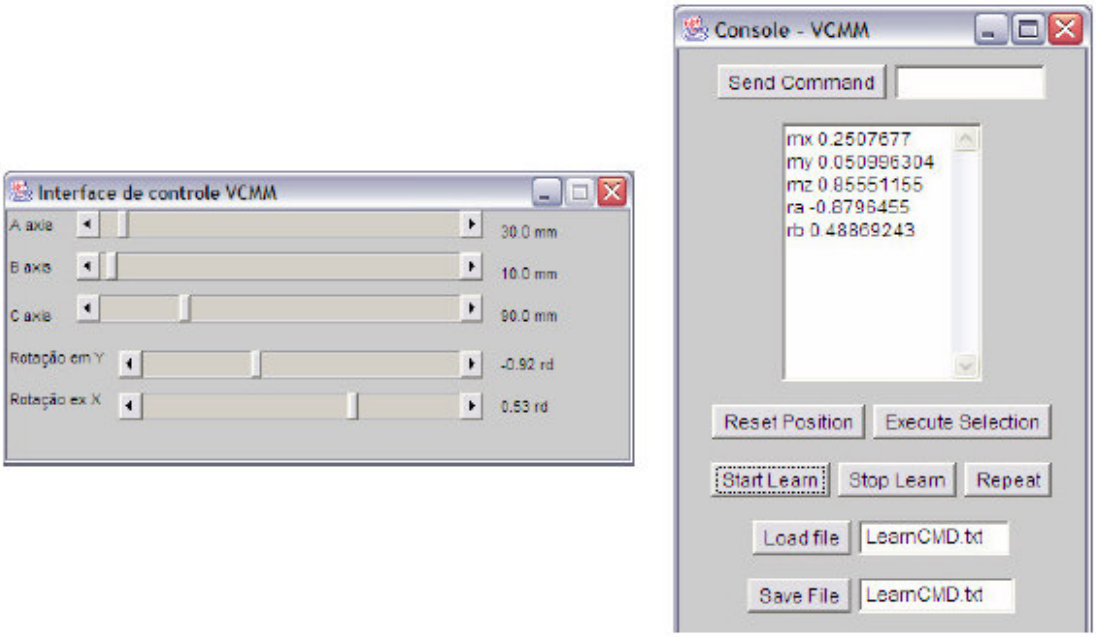


Figura 5 – Interface usuário para interação e navegação na cena

Os casos de uso apresentados podem ser atendidos independente do modelo arquitetural do programa a ser implementado, isto é, há casos em que a implementação do modelo usando RMI é mais adequada que o modelo utilizando Sockets.



(a) Controle de edição comandos por barra de rolagem (b) Controle de edição de comandos

Figura 6 – Controle de edição

3. Plataforma de Desenvolvimento

O desenvolvimento deste ambiente ocorreu de maneira gradativa, começando a análise dos modelos de comunicação com Sockets e RMI para suporte de aplicações para a Internet. Esses modelos impõem restrições à comunicação utilizado *applets* devido às políticas de segurança implementadas pelos browsers e pelo ambiente de execução da máquina Java. Refletindo as possibilidades de uso do pacote Shout3D como renderizador e interface da cena. Durante este período foram executados testes com sucesso em ambientes Windows, Linux e Macintosh, apresentando taxas de renderização entre 38 e 40 frames por segundo, nos seguintes equipamentos:

- Desktop AMD Duron 850Mz, 128Mb, Video S3 Trio3D/2X 8Mb, Windows 2000 SP3, Internet Explorer 6.0.2800.1106 SP1, Java Plug-in 1.4.0_01 e Linux Mandrake 9.1
- Macintosh PowerPC G4 800Mhz, 128Mb, Mac OS X versão 10.3, browser Safari 1.1, Java Plug-in 1.3.1 e Java 1.4.1
- Notebook Dell Latitude C400, Pentium III 1.2Ghz, 256Mb, Video Intel82830 48Mb, Windows 2000 SP3, Internet Explorer 6.0.2800.1106 SP1, Java Plug-in 1.4.0_01
- Notebook Toshiba Satellite A45-S1202, Celeron 2.80Hz, 512Mb, Video Intel82852 64Mb, Windows XP Home Edition SP1, Internet Explorer 6.0.2800.1106.xpsp2.030320-1720, Java Plug-in 1.4.2

Tabela 1 – Caso de Uso A

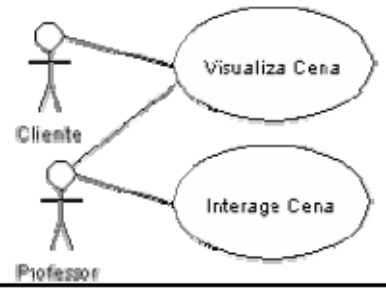
	O ator Professor interage com a cena através de comandos de máquina, que serão transmitidos através da rede e visualizados pelo ator Cliente e Professor após sua renderização. Este modelo apresenta somente a interação da cena pelo ator Professor. Exemplo: Professor apresentado para seus Clientes como programar comandos CMM.

Tabela 2 – Caso de Uso B

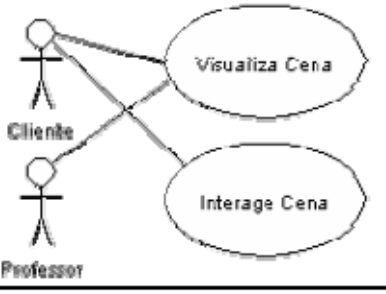
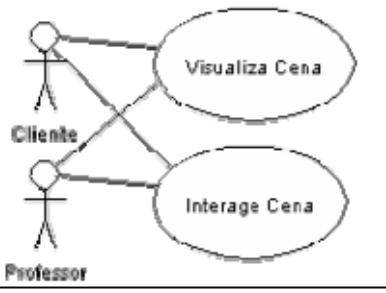
	O ator Cliente interage com a cena através de comandos de máquina, que serão transmitidos através da rede e visualizados pelo ator Professor e Cliente após sua renderização. Este modelo apresenta somente a interação da cena pelo ator Cliente. Exemplo: Clientes programando comandos CMM, sendo monitorados pelo Professor.

Tabela 3 – Caso de Uso C

	O ator Cliente e Professor interagem e visualizam a cena através de comandos de máquina simultaneamente. Este modelo apresenta uma colaboração entre ambos. Exemplo: Estudo em grupo envolvendo Clientes e Professor em uma mesma cena.

4. Conclusão

A abordagem modular adotada na construção da aplicação de VCMM, mostra a portabilidade oferecida pelo protocolo de realidade virtual distribuída para web. Utilizando este modelo de camadas de comunicação e componentes, obteve-se um alto grau de flexibilidade, consegui-se manter independente a camada de comunicação da camada de renderização da aplicação.

Neste estágio do trabalho as questões referentes à performance de comunicação considerando a utilização de RMI ou Socket ainda não foram abordadas, visto que em muitos dos ambientes virtuais distribuídos analisados, utilizam um modelo híbrido de controle de comunicação com RMI e Socket. Cabe observar que, freqüentemente, estes ambientes não geram continuidade nas pesquisas, devido sua arquitetura ser proprietária (HP-UX, SUN, AIX), muitas vezes desenvolvidas em linguagens que não são portáveis para as plataformas atuais [Zottino, 2004].

O modelo proposto tornou a aplicação portável para múltiplas plataformas, o que proporciona sua reutilização para as situações descritas nos casos de uso, mas também

para aplicações correlatas, visto também que os recursos necessários para o seu desenvolvimento e utilização são facilmente encontrados na maioria das configurações de computadores atualmente disponíveis [Zottino, 2004].

As pesquisas, envolvendo VCMM e o protocolo de comunicação para ambientes virtuais distribuídos, mostram que os mesmos podem ser facilmente reutilizados e adaptados para uso em novos padrões como X3D [X3D, 2003], com pequenas adequações na classe ServerShout.

5. Referências Bibliográficas

Booch, Grady; Rumbaugh, James; Jacobson, Ivar. **UML: Guia do Usuário**. 2ª. Ed, Rio de Janeiro: Campus, 2000. 472 p. ISBN 85-352-0562-4

Calonego Junior, Nivaldi, **Relatório de visita científica**, CNPq, Processo: 490182/02-0, Local da Missão: ZGDV - Darmstadt – Alemanha, Período da Missão: 15/01/2003 a 15/02/2003.

Deitel, H. M.; Deitel, P.J. **Java Como Programar**. (Trad. Edson Furnankiewicz) 3 ed., Porto Alegre: Bookman, 2001. 1021p. ISBN 85-7307-727-1

Forbes, A.B. *Validation of software for dimensional metrology*. NPL Report DICT 225/94. Teddington, UK. Feb. 1994.

Kirner, Claudio, VESIV - ***Virtual Environment for Shared Interactive Visualization***, CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico ASCIN - Assessoria de Cooperação Internacional, Convênios Bilaterais, Convênio/País: Programa de Cooperação Brasil-Alemanha em Tecnologia da Informação, Número de Processo: 690012/02-0 ASCIN/CNPq, Título do Projeto: VESIV - *Virtual Environment for Shared Interactive Visualization*, Coordenador do Lado Brasileiro: Prof. Dr. Claudio Kirner. 2003.

Polevoi, Rob. **Interactive Web Graphics with Shout3D**. San Franciso: SYBEX, 2001. 397p. ISBN 0-7821-2860-2.

SUN. Java Tutorial. http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html, 2003.

VRML. VRML97 Functional specification and VRML97 External Authoring Interface (EAI) International Standard ISO/IEC 14772-1:1997, ISO/IEC 14772-2:2002. http://www.web3d.org/fs_specifications.htm, 2003.

Weckenmann, A.; Eitzet, H.; Garmer, M.; Weber, H. **Functionality-oriented evaluation and sampling strategy in co-ordinate metrology**. Precision Engeneering, 17(4):244-252, 1995.

X3D. Extensible 3D (X3D). International Draft Standards. http://www.web3d.org/fs_specifications.htm, 2003.

Zottino, Ricardo, Calonego, Nivaldi Junior, Abackerli, Álvaro José, Coletta, Carlos Eduardo Della. Acesso via Internet para Máquina Virtual de Medir por Coordenadas. Anais do WRA'2004, I Workshop sobre Realidade Aumentada. Universidade Metodista de Piracicaba, 2004.

A.3. AMBIENTE VIRTUAL DISTRIBUIDO WEB – AVDW

CO.06

AMBIENTE VIRTUAL DISTRIBUIDO WEB – AVDW

Ricardo Zottino (Mestrado em Ciência da Computação);
Nivaldi Calonego Junior (Orientador - Faculdade de Ciências Matemáticas, da Natureza e Tecnologia da Informação)

1. Introdução

As pesquisas no campo da Realidade Virtual durante a década de 90 propiciaram o surgimento de vários ambientes virtuais distribuídos e colaborativos (KIRNER, 2003), que em sua maior parte estão implementados na linguagem C++ e geralmente para plataformas proprietárias, tais como: SUN, HP-UX, IBM-AIX e Silicon Graphics.

Devido a restrições tecnológicas ou financeiras, muitos desses projetos foram descontinuados, por não serem portáveis para os computadores pessoais atualmente disponíveis no mercado. Dentro deste contexto, surge a necessidade da criação de uma camada em maior grau de abstração para a implementação de um protocolo portável para múltiplas plataformas, criando a necessidade de uso de novas tecnologias, tais como Java (DEITEL, 2001) (SUN 2003), Shout3D (POLEVOI, 2001) e VRML (VRML, 2003).

Este artigo apresenta um modelo para o desenvolvimento para Ambiente Virtual Distribuído e sua implementação, usando uma metodologia Orientada a Objeto (OO) aplicada no desenvolvimento do software de Realidade Virtual Distribuída para a Internet, que aplica conceitos de Realidade Virtual (RV) para criar um protótipo da "Máquina de Medir por Coordenadas Virtual" (VCMM) (ZOTTINO, 2004).

Parte do software deste trabalho foi implementado e testado para uso em computadores locais e remotos, mostrando que o modelo proposto para a implementação de Ambientes Virtuais Distribuídos para a Internet é viável.

2. Objetivo do Trabalho

Apresentar o protótipo da aplicação para Ambientes Virtuais Distribuídos (AVD), utilizado como base para o desenvolvimento de aplicações em ensino de metrologia. Trabalhos anteriores mostram o modelo geométrico da máquina e aspectos de sua dinâmica (CANONEGO, 2003), mas deixam abertas questões relativas ao Protocolo de Comunicação para Realidade Virtual Distribuída (PRVD) para Internet, responsável pela transmissão de informações entre as camadas e os componentes de uma aplicação, neste caso a Máquina Virtual de Medir por Coordenadas (VCMM) (ZOTTINO, 2004).

3. Desenvolvimento

A concepção deste trabalho considerou a utilização máxima de recursos "Open Source" atualmente disponíveis, o uso de uma arquitetura de software orientada a serviços e a adoção de padrões estabelecidos pela comunidade científica, possibilitando sua reutilização em aplicações de Realidade Virtual Distribuída para a Internet. Neste caso, o protocolo de comunicação utiliza as tecnologias Socket e Remote Method Invocation (RMI) como mecanismos de transmissão das informações (DEITEL, 2001) (SUN, 2003).

Apesar das dificuldades e problemas detectados na elaboração de Ambientes Virtuais Distribuídos, estes podem ser significativamente reduzidos através da delimitação do problema, que em nosso caso abordará a utilização dentro de um ambiente controlado, através do número de usuários.

Uma visão macro de um modelo cliente-servidor ilustrado na Figura 1, permite que as modificações de um ambiente virtual podem estar associadas a diferentes tipos de serviços como interação, navegação e comandos, em nosso caso estamos abordando somente a utilização do serviço de transmissão de comandos na implementação do PRVD.

A modelagem do PRVD é apresentada no diagrama de classe da Figura 2, que usa a linguagem semi-formal denominada *Unified Modeling Language* (UML) (BOOCH, 2000), para a representação do modelo orientado a objetos.

Neste modelo observamos a classe ServerShout, que é derivada de Unicast Remote Object (RMI) e implementa uma interface Socket, que por sua vez implementa a interface ParserPRVD, responsável pela conversão das sintaxes de comandos do PRVD, como por exemplo o comando de translação de cena "*set_translation(node Name:String, nodePosition:float)*", para os comandos nativos da biblioteca Shout3D recebidos de um cliente.

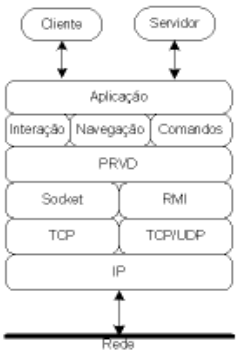


Figura 1 – Camada de protocolos

Do outro lado tem-se a classe ServerPRVD, que é derivada de Unicast Remote Object (RMI) e também implementa a interface Socket, que por sua vez implementa a interface ParserPRVD. ServerPRVD é responsável pelo transmissão das informações entre a camada de rede e a camada da aplicação, enviando e recebendo informações, em nosso caso comandos da VCMM.

Note-se que a aplicação pode ser executada com as tecnologias RMI ou Socket, alterando o comportamento das classes ServerShout e ServerPRVD, o uso de uma ou da outra é determinado pelo usuário, em tempo de execução, ou seja, não há necessidade de compilação através de parâmetros inseridos no arquivo do tipo html que inicializa uma Applet Java em um Browser.

A classe VCMM implementa a interface CommandPRVD, responsável por validar todos os comandos, parâmetros e limites da VCMM, que serão recebidos do ConsoleApplet e convertendo para comandos do protocolo PRVD. Como VCMM é derivada de ServerPRVD, herda todos as suas características como métodos e controles por ela implementados. Por exemplo, o comando VCMM "mx" implementado dentro da classe VCMM realiza um deslocamento do "Eixo X da Máquina Virtual de Medir por Coordenadas". De maneira análoga é possível criar novos controles para outros tipos de máquinas ou equipamentos semelhantes a VCMM, utilizando a mesma estrutura de classes.

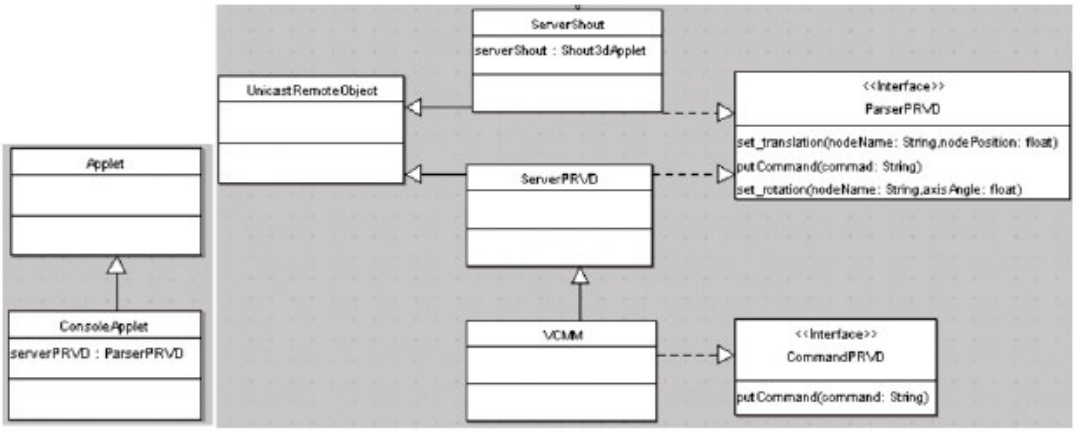


Figura 2 – Diagrama de Classe do Modelo

4. Resultado e Discussão

A abordagem modular adotada na construção da aplicação de VCMM, mostra a portabilidade oferecida pelo protocolo de realidade virtual distribuída para web. Utilizando este modelo de camadas de comunicação e componentes, obteve-se um alto grau de flexibilidade, mantendo independente a camada de comunicação da camada de renderização e também camada da aplicação, e também sua utilização em diferentes plataformas de computadores (X3D, 2003).

5. Considerações Finais

As pesquisas, envolvendo VCMM e o protocolo de comunicação para Ambientes Virtuais Distribuídos, mostram que os mesmos podem ser facilmente reutilizados e adaptados para uso em novos padrões como X3D (X3D 2003), com pequenas adequações na classe ServerShout em futuros trabalhos.

Referências Bibliográficas

BOOCH, G.; RAUMBAUGH, J.; JACOBSON, I.. **UML: Guia do Usuário**. 2ª. Ed, Rio de Janeiro: Campus, 2000. 472 p. ISBN 85-352-0562-4

CALONEGO JUNIOR, N., **Relatório de visita científica**, CNPq, Processo: 490182/02-0, Local da Missão: ZGDV - Darmstadt – Alemanha, Período da Missão: 15/01/2003 a 15/02/2003.

DEITEL, H. M.; DEITEL, P.J. **Java Como Programar**. (Trad. Edson Furnankiewicz) 3 ed., Porto Alegre: Bookman, 2001. 1021p. ISBN 85-7307-727-1

KIRNER, C., VESIV - ***Virtual Environment for Shared Interactive Visualization***, CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico ASCIN - Assessoria de Cooperação Internacional, Convênios Bilaterais, Convênio/País: Programa de Cooperação Brasil-Alemanha em Tecnologia da Informação, Número de Processo: 690012/02-0 ASCIN/CNPq, Título do Projeto: VESIV - *Virtual Environment for Shared Interactive Visualization*, Coordenador do Lado Brasileiro: Prof. Dr. Claudio Kirner. 2003.

POLEVOI, R.. **Interactive Web Graphics with Shout3D**. San Franciso: SYBEX, 2001. 397p. ISBN 0-7821-2860-2.

SUN. Java Tutorial. http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html, 2003.

VRML. VRML97 Functional specification and VRML97 External Authoring Interface (EAI) International Standard ISO/IEC 14772-1:1997, ISO/IEC 14772-2:2002. http://www.web3d.org/fs_specifications.htm, 2003.

X3D. Extensible 3D (X3D). International Draft Standards.http://www.web3d.org/fs_specifications.htm, 2003.

ZOTTINO, R. et al.. **Acesso via Internet para Máquina Virtual de Medir por Coordenadas**. Anais do WRA'2004, I Workshop sobre Realidade Aumentada. Universidade Metodista de Piracicaba, 2004.

ZOTTINO, R. et al**. Ambiente Virtual Distribuído Web para Máquinas de Medir por Coordenadas utilizando Shout3D e RMI**. Anais do SRV'2004, VII Simpósio de Realidade Virtual. São Paulo, 2004.

Apêndice B. Instalação e Configuração do Ambiente Virtual Distribuído

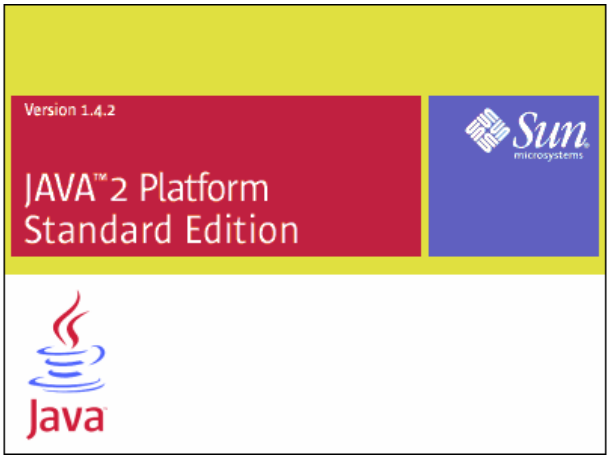
Este apêndice tem o propósito de apresentar o procedimento de instalação do Ambiente Virtual Distribuído para os sistemas operacionais Windows 2000, Windows XP. Lembramos que este procedimento não abordará o processo de instalação dos respectivos sistemas operacionais, partindo do pressuposto que o mesmo já se encontra previamente instalado e em funcionamento.

B.1. Windows 2000 e XP

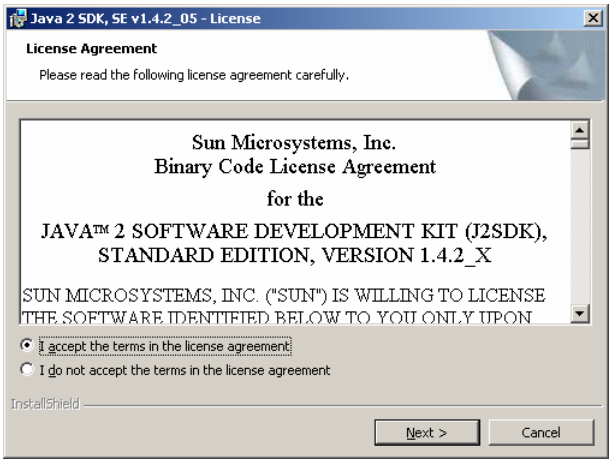
B.1.1. Instalação da Plataforma Java 2 SDK, SE v1.4.2_05

O pasta padrão da instalação da plataforma Java 2 no Windows é **c:\j2sdk1.4.2_05** para o pacote SDK e **c:\Program Files\Java\j2re1.4.2_05** para o run time.

1. Executar o programa **j2sdk-1_4_2_05-windows-i586-p.exe** para iniciar a instalação da Plataforma JAVA 2 Standard Edition.

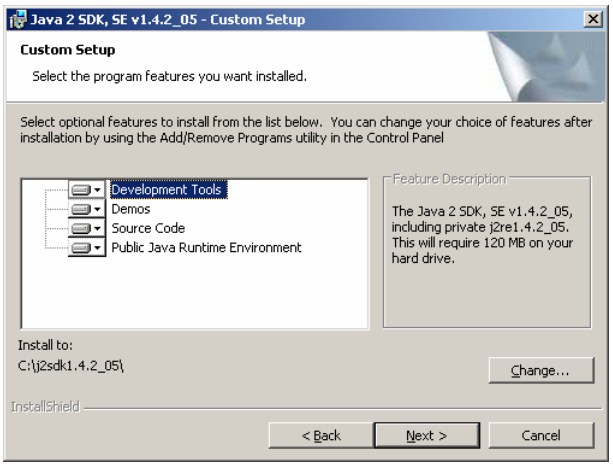


2. Selecionar a opção **“I accept the terns in the license agreement”**.



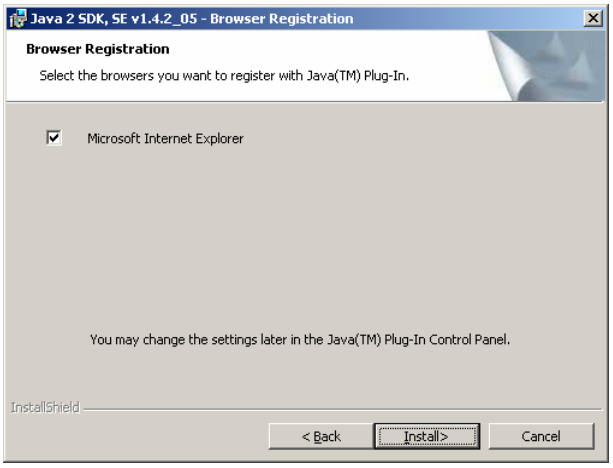
3. Clicar no botão **Next**.

4. Caso seja necessário é possível escolher quais características deverão ser instaladas. Em um ambiente de desenvolvimento normalmente se instala toda a opção padrão, como apresentado abaixo.



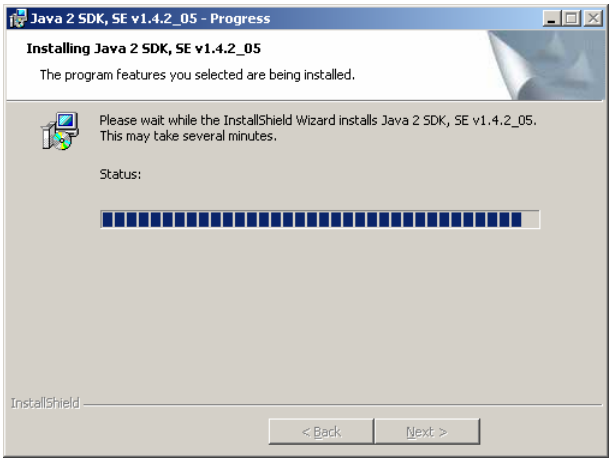
5. Clicar no botão **Next**.

6. Esta opção permite configurar o plug-in Java no *browser* que já se encontra previamente instalado no sistema operacional do computador, simplesmente selecionando o *browser* que se deseja instalar o plug-in.

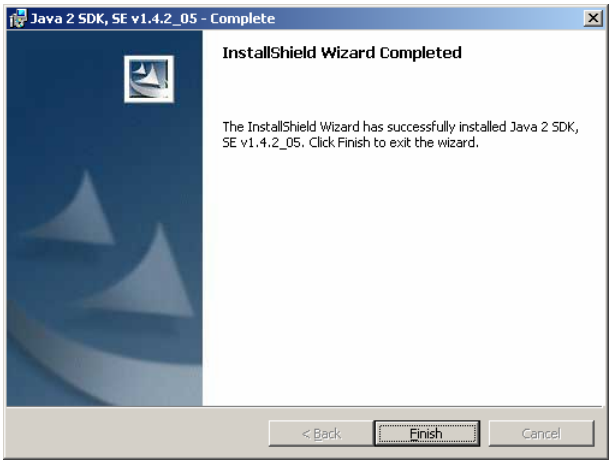


7. Clicar no botão **Install**.

8. Aguardar a conclusão do processo de instalação.



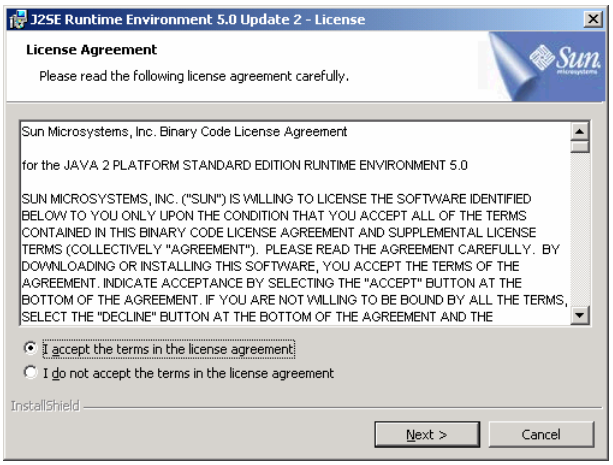
9. Depois de concluída a instalação.



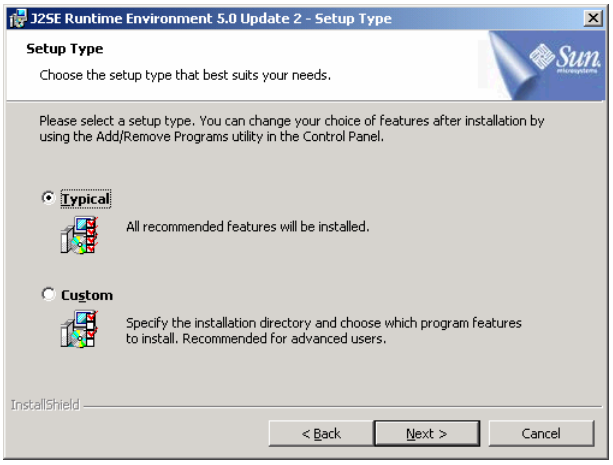
10. Clicar no botão **Finish**, para finalizar a instalação.

11. Se o computador em estiver conectado a Internet, o programa de instalação pode verificar se existe alguma atualização que possa ser realizada.

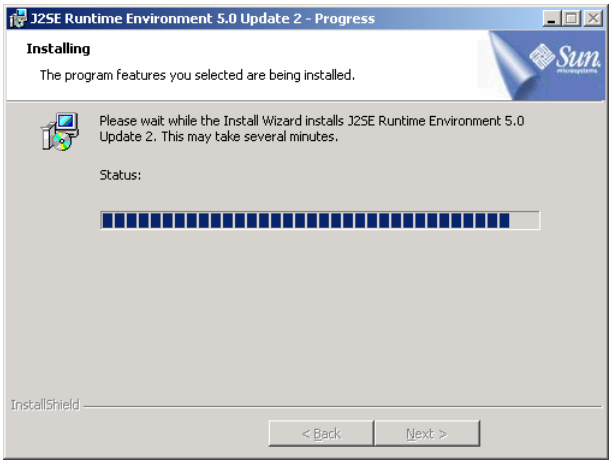
12. Aguardar a atualização.



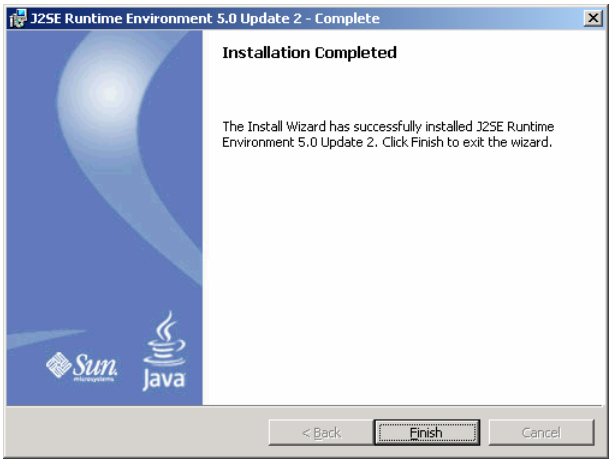
13. Selecionar a opção **"I accept the terms in the license agreement"** e clicar no botão **Next**.



14. Selecionar a opção **Typical** e clicar no botão **Next**.



15. Aguardar a conclusão do processo de instalação.

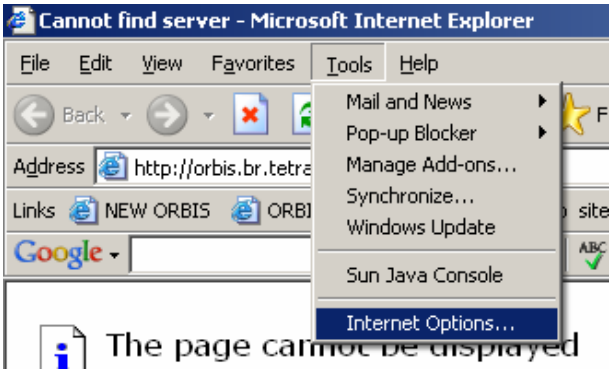


16. Clicar no botão **Finish**, para finalizar a instalação.

B.1.2. HABILITANDO O PLUG-IN JAVA (SUN) NO INTERNET EXPLORER

1. Executar o programa **Internet Explorer** para iniciar a configuração do plug-in Java (Sun).

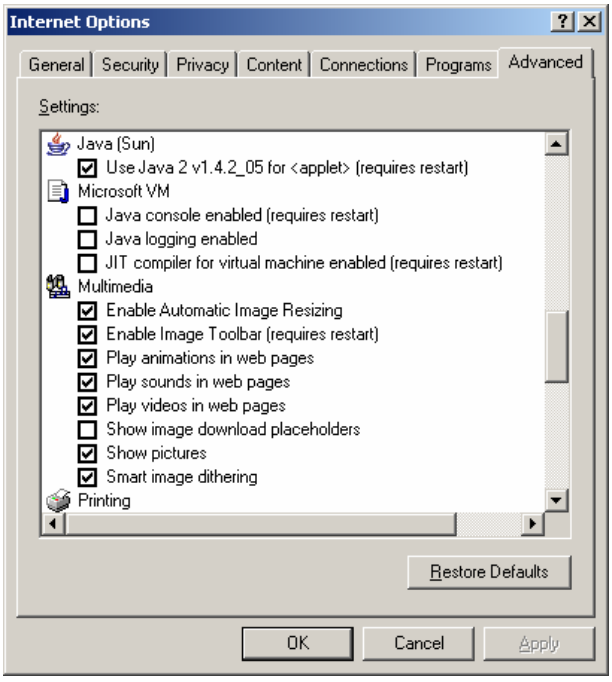
2. Clicar na opção **Tools** localizada barra de ferramentas.



3. Selecionar a opção **Internet Options**.

4. Selecionar a guia **Advanced**.

5. Selecionar o plug-in Java (Sun) como exibido na imagem abaixo, caso exista o plug-in Java (Microsoft VM) é necessário desativa-lo.



B.1.3. CONFIGURAÇÃO DO AMBIENTE PARA EXECUÇÃO DA APLICAÇÃO

Iniciamos o desenvolvimento do projeto em 2003 com o Windows 2000, e durante seu desenvolvimento realizamos a atualização para Windows XP, onde identificamos algumas incompatibilidades entre os sistemas operacionais, plataforma Java e alguns utilitários, sendo necessário ajustes para permitir seu funcionamento, conforme descrito a seguir.

B.1.3.1. CONFIGURAÇÃO DO ARQUIVO JAVA.POLICY

Para que as *applets* possam ser executadas é necessário que o arquivo java.policy seja alterado, este arquivo fica localizado na pasta *default* de instalação do Java SDK C:\j2sdk1.4.2_05\jre\lib\security.

Sua alteração pode ser realizada utilizando o utilitário Java **policytools.exe**, ou simplesmente alterando o arquivo com editor de texto, e inserindo a linha **permission java.security.AllPermission** no final do arquivo java.policy.

Esta alteração permite que *applets* originadas de outros computadores, possam utilizar os métodos remotos RMI e também enviar e receber mensagens de outros computadores utilizando Socket. A cópia do arquivo java.policy utilizado no protótipo pode ser encontrado na pasta \AVDW do CD anexo.

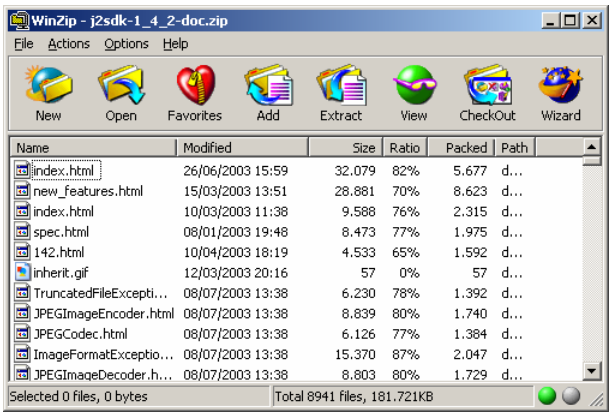
B.1.3.2. CONFIGURAÇÕES PARA USO DO RMI COM WINDOWS XP

Para que os utilitários **rmic.exe** e **rmiregistry.exe** possam funcionar corretamente no Windows XP, é necessário que estes utilitários sejam

executados a partir da pasta onde se encontra localizado a aplicação, e que a variável de ambiente CLASSPATH também esteja apontada para a pasta onde se encontra localizada a aplicação, caso contrário a aplicação não consegue registrar seu endereço no rmiregistry. Este problema não ocorre com o Windows 2000.

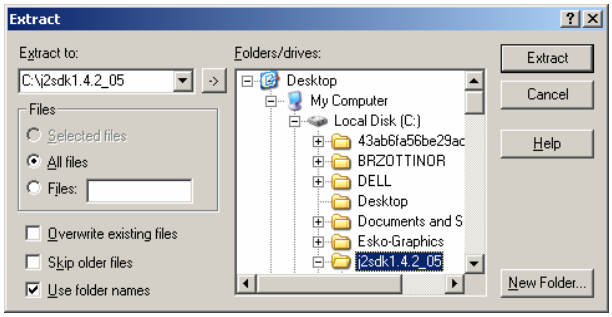
B.1.4. INSTALAÇÃO DA DOCUMENTAÇÃO JAVA 2 SKD

1. Abrir o arquivo compactado **j2sdk-1_4_2-doc.zip** para possa ser instalada a documentação da Plataforma Java 2.



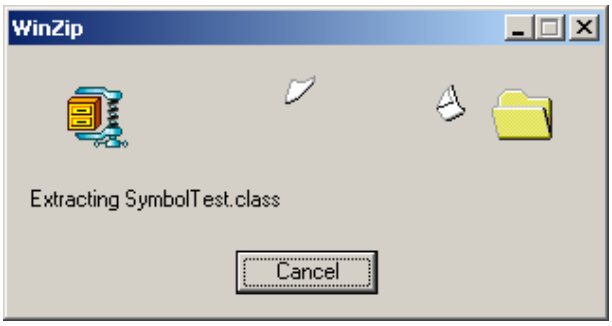
2. Clicar no botão **Extract**.

3. Selecionar a pasta **C:\j2sdk1.4.2_05** onde normalmente se encontra a Plataforma Java 2 instalada.



4. Selecionar as opções **All Files** e **Use folder names** e depois clicar no botão **Extract**, para que todas a documentação seja extraída.

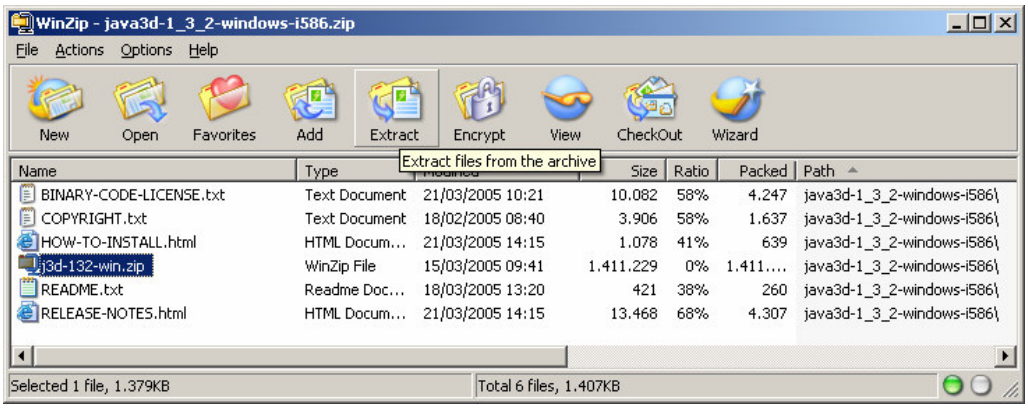
5. Aguardar o término da extração dos arquivos.



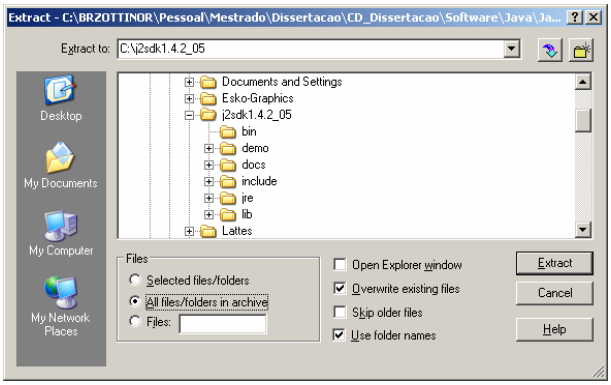
6. Após concluir a extração dos arquivos fechar o utilitário Winzip, para finalizar a instalação.

B.1.5. INSTALAÇÃO DO JAVA3D

1. Abrir o arquivo **java3d-1_3_2-windows-i586.zip** para que possa ser instalado o pacote Java3D.



2. Selecionar o arquivo **j3d-132-win.**zip e clicar no botão **Extract**.



3. Selecionar a pasta **C:\j2sdk1.4.2_05** onde normalmente se encontra a Plataforma Java 2 instalada, selecionar também as opções **All files/folders in archive** e clicar no botão **Extract**.

4. Após concluir a extração dos arquivos fechar o utilitário Winzip, para finalizar a instalação.

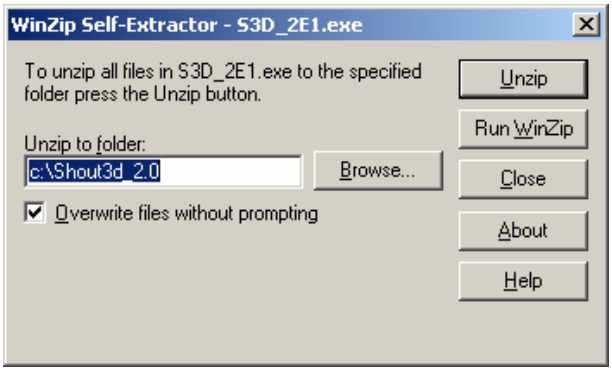
B.1.6. INSTALAÇÃO DO SHOUT3D V2.0

1. Executar o programa **clickme.exe**.

2. Clicar a opção **ACCEPT**.

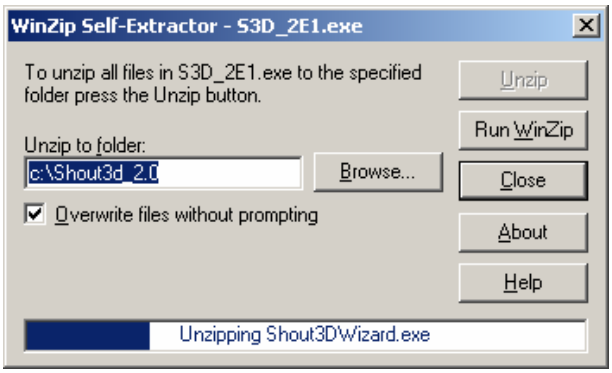
3. Clicar na opção **Shout3D demo**.

4. Descompactar os programas na pasta padrão **c:\Shout3d_2.0**.

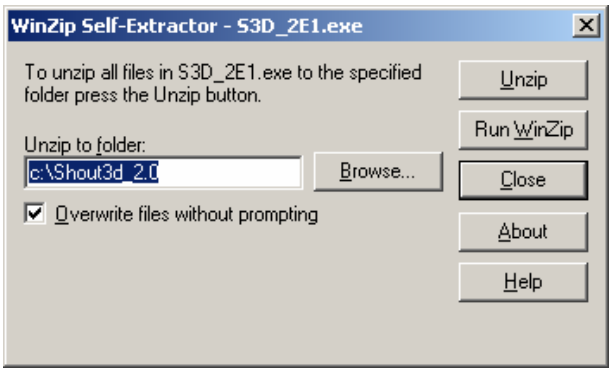


5. Clicar no botão **Unzip**.

6. Aguardar o término da descompactação.



7. Depois de concluída a extração dos arquivos, clicar no botão **OK**.



8. Clicar no botão **Close** para finalizar a instalação.

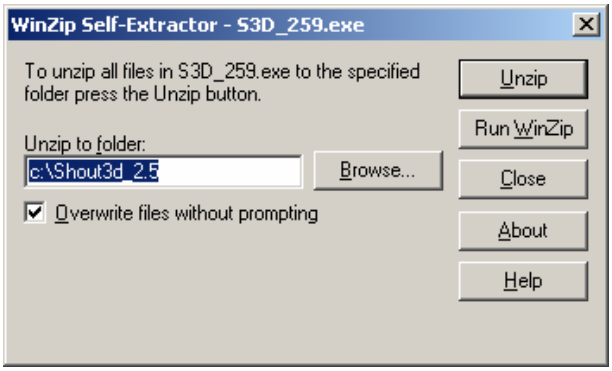
9. Clicar na opção **Exit** para finalizar a instalação.

B.1.7. INSTALAÇÃO DO SHOUT3D V2.5

O procedimento a seguir descreve a instalação do upgrade da biblioteca gráfica Shout3D para a versão 2.5.

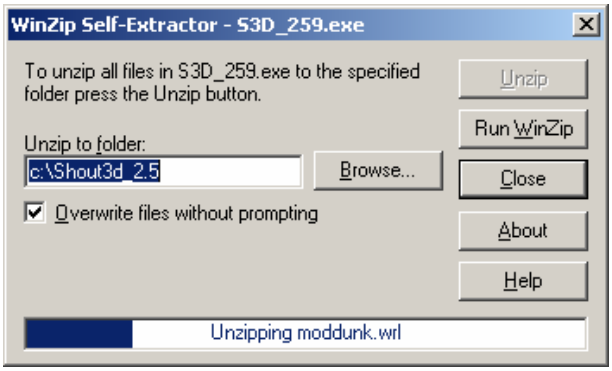
1. Executar o programa **S3D_259.exe**.

2. Descompactar os programas na pasta padrão **c:\Shout3d_2.5**.

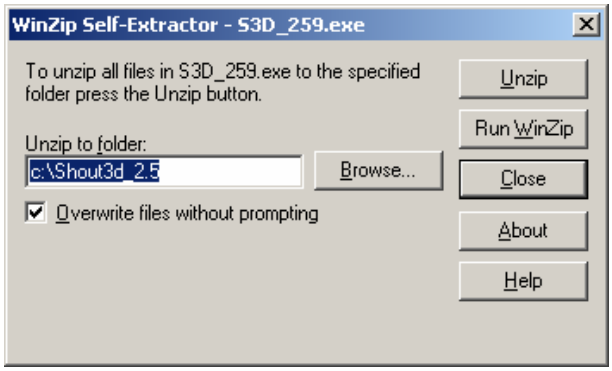


3. Clicar no botão **Unzip**.

4. Aguardar o término da descompactação.



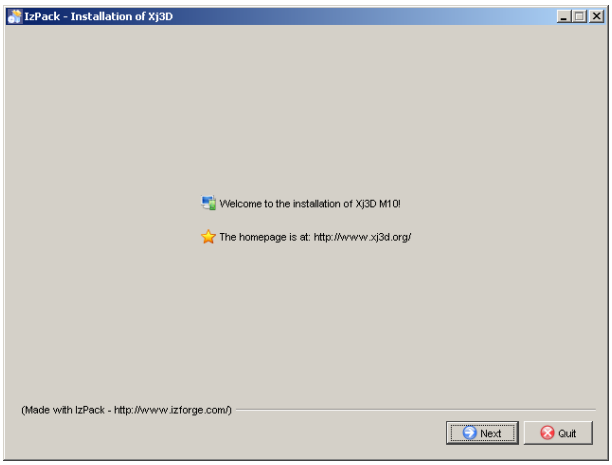
5. Depois de concluída a extração dos arquivos, clicar no botão **OK**.



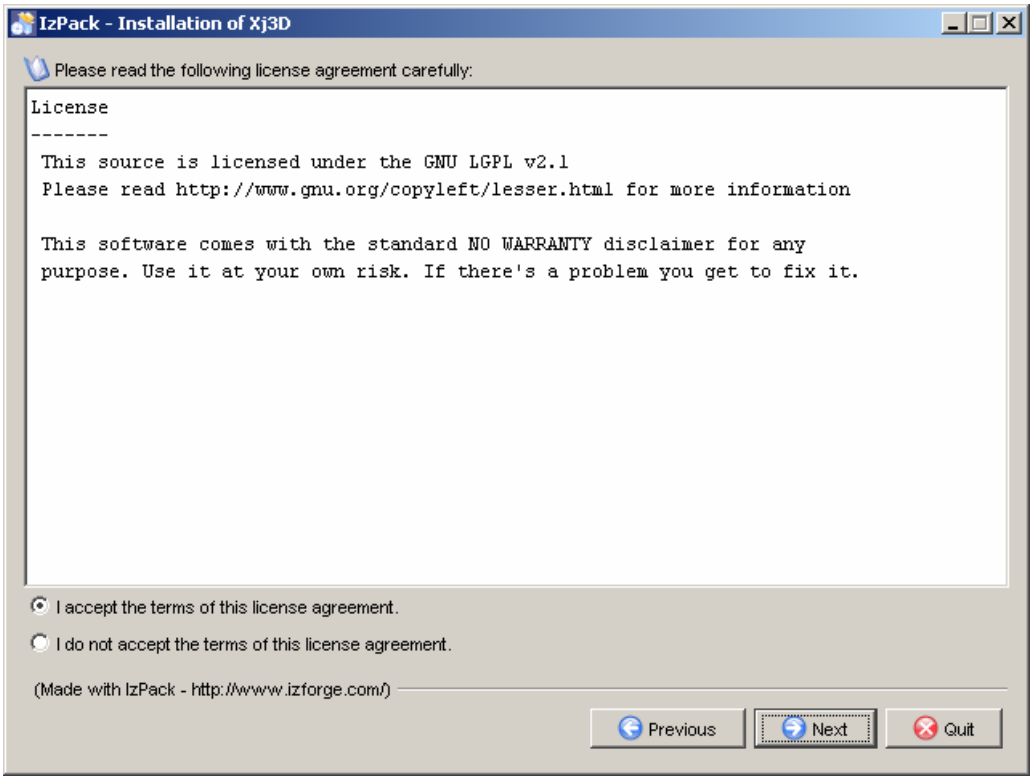
6. Clicar no botão **Close** para finalizar a instalação.

B.1.8. INSTALAÇÃO DO BROWSER XJ3D

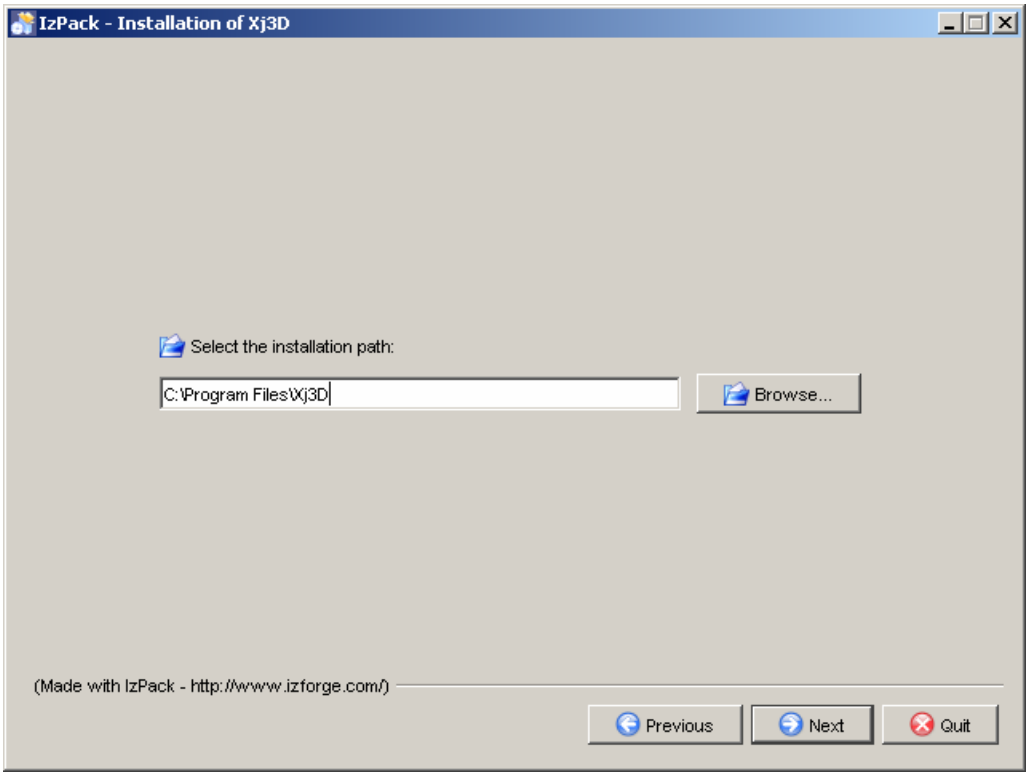
1. Executar o programa **Xj3D-M10-windows.jar** para instalar o browser X3D.



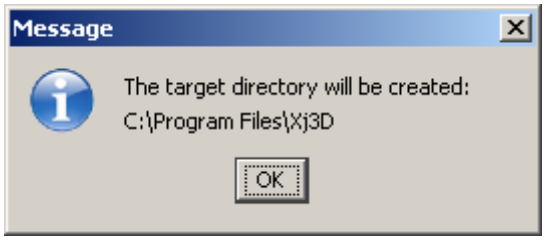
2. Clicar no botão **Next**.



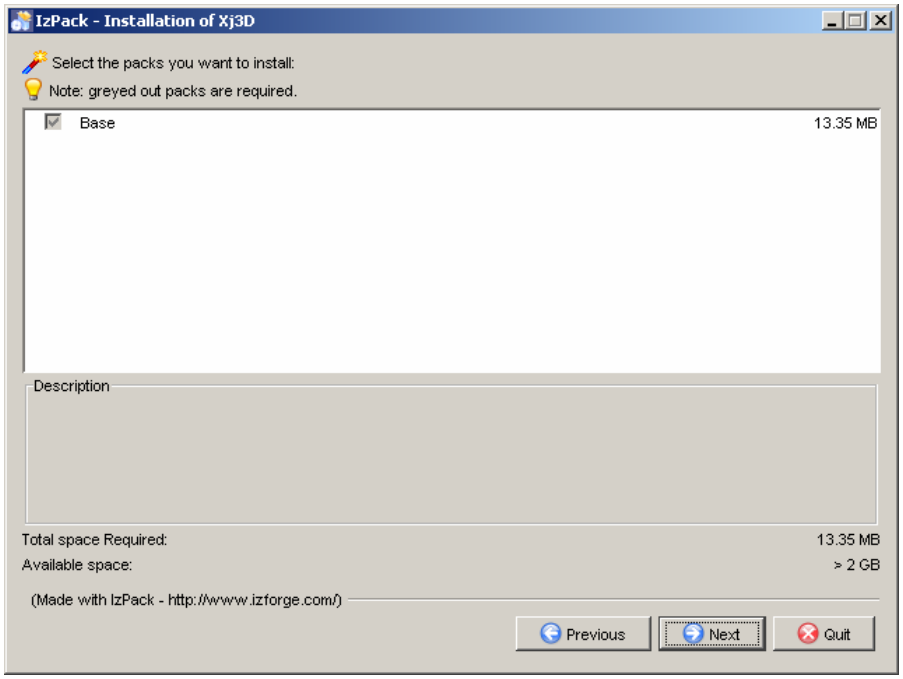
3. Selecionar a opção **"I accept the terms of this license agreement"** e clicar no botão **Next**.



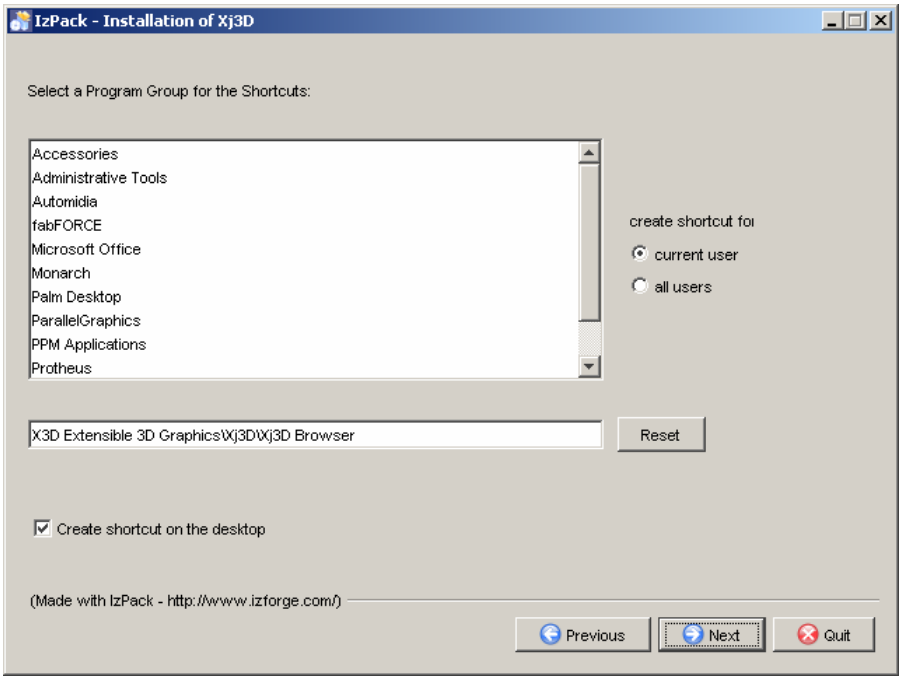
4. Clicar no botão **Next**.



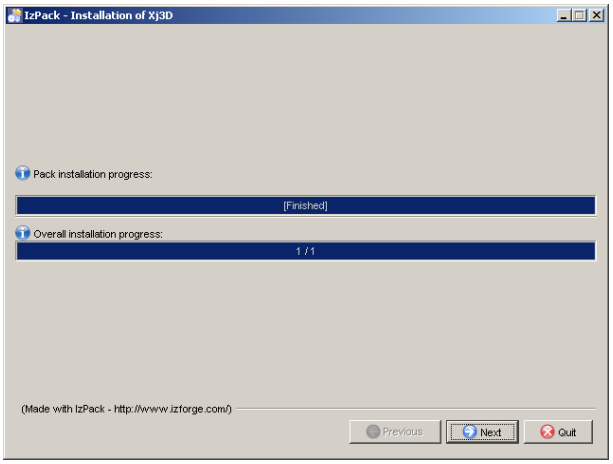
5. Clicar no botão **OK**.



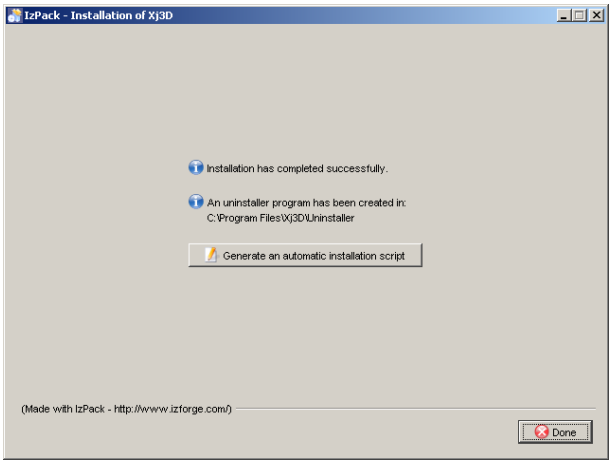
6. Clicar no botão **Next**.



7. Clicar no botão **Next**.



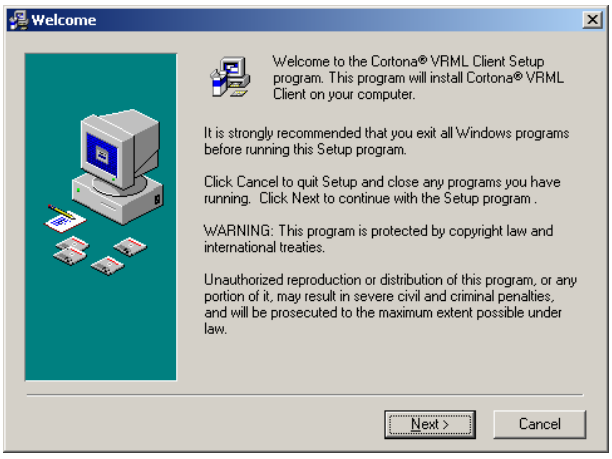
8. Clicar no botão **Next**.



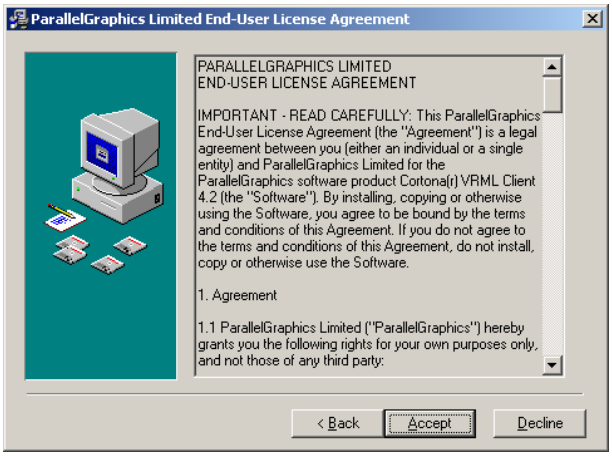
9. Clicar no botão **Done** para finalizar a instalação.

B.1.9. INSTALAÇÃO DO PLUG-IN VRML CORTONA

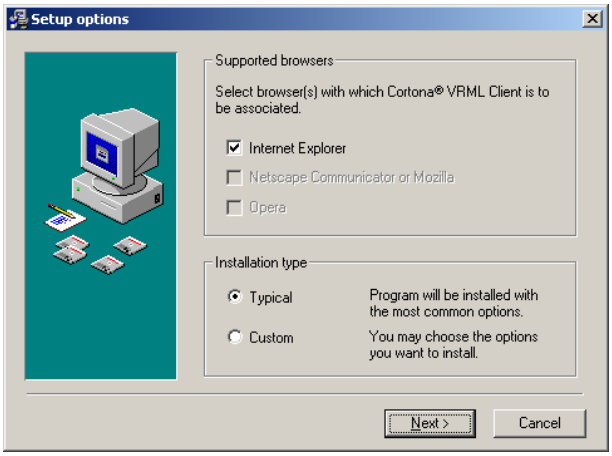
1. Executar o programa **Cortvrml.exe** para instalar o plug-in VRML Cortona Client 4.2.



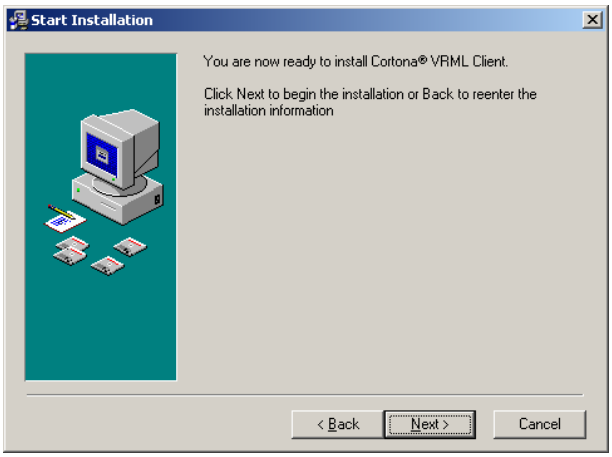
2. Clicar no botão **Next**.



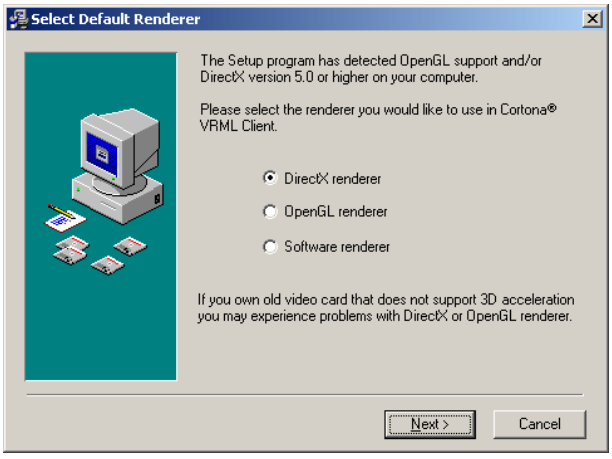
3. Clicar no botão **Accept**.



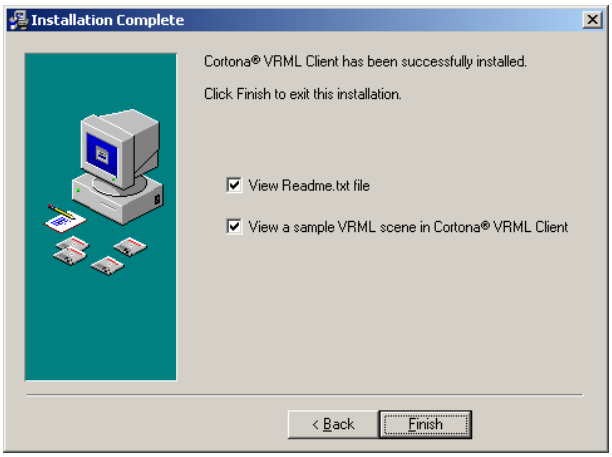
4. Clicar no botão **Next**.



5. Clicar no botão **Next**.



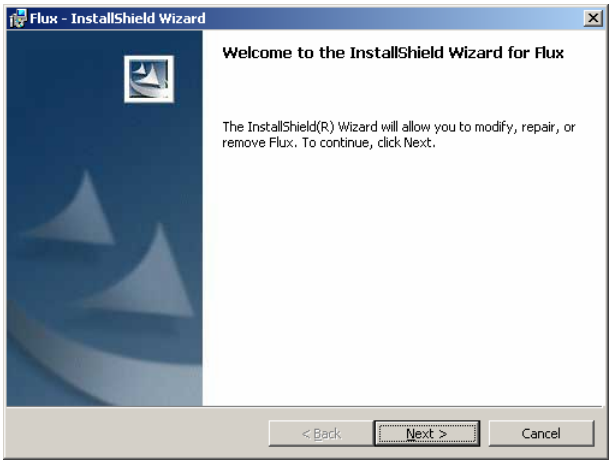
6. Clicar no botão **Next**.



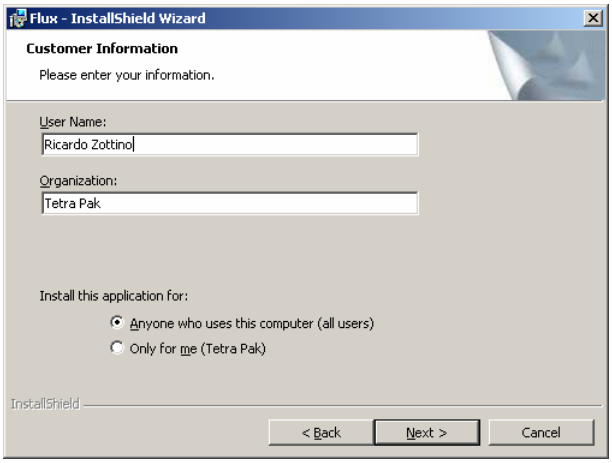
7. Clicar no botão **Finish** para finalizar a instalação.

B.1.10. INSTALAÇÃO DO PLUG-IN VRML E X3D FLUXPLAYER

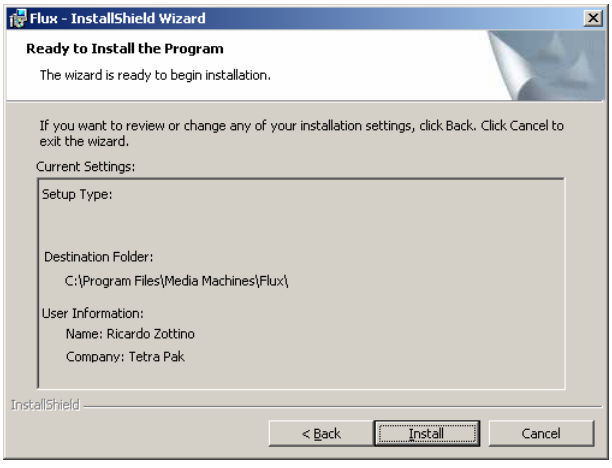
1. Executar o programa setup.exe para instalar o plug-in VRML e X3D FluxPlayer 1.2.



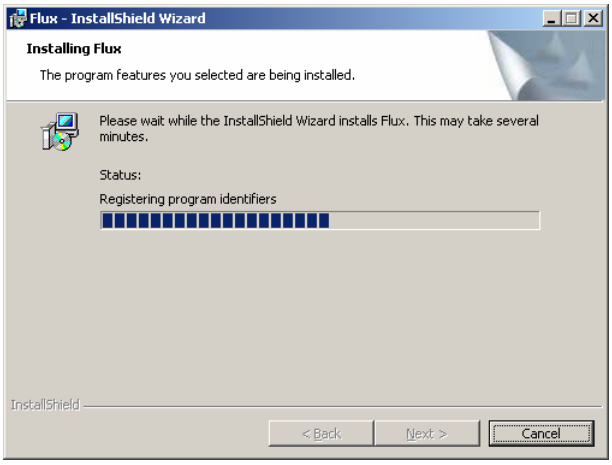
2. Clicar no botão **Next**.



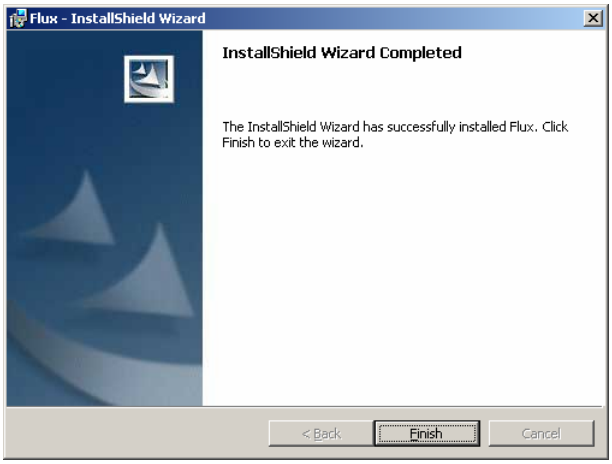
3. Clicar no botão **Next**.



4. Clicar no botão **Install**.



5. Aguardar a conclusão do processo de instalação.

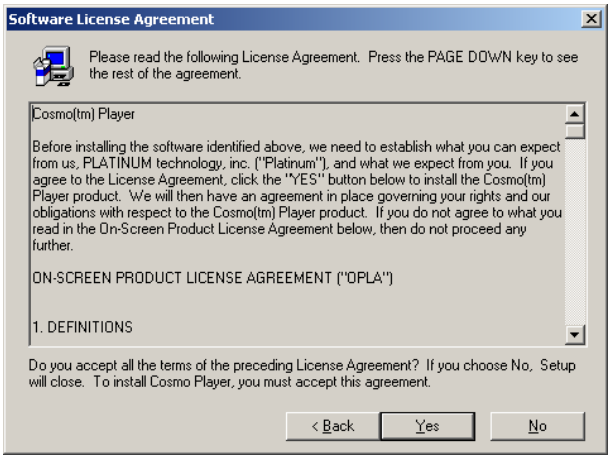


6. Clicar no botão **Finish** para finalizar a instalação.

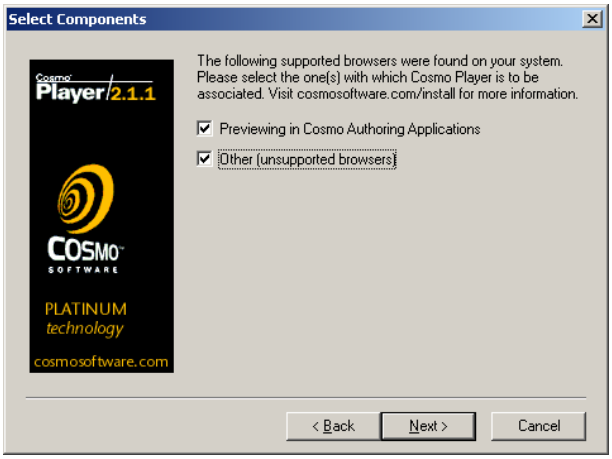
B.1.11. INSTALAÇÃO DO PLUG-IN VRML COSMOPLAYER

1. Executar o programa **cosmo_win95nt_eng.exe** para instalar o plug-in VRML CosmoPlayer 2.1.1.

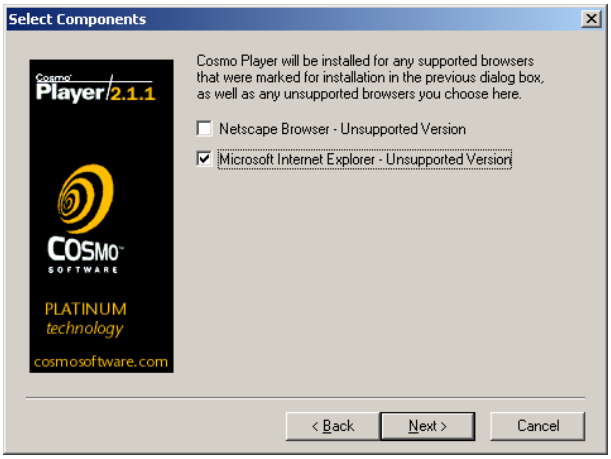
2. Clicar no botão **Next**.



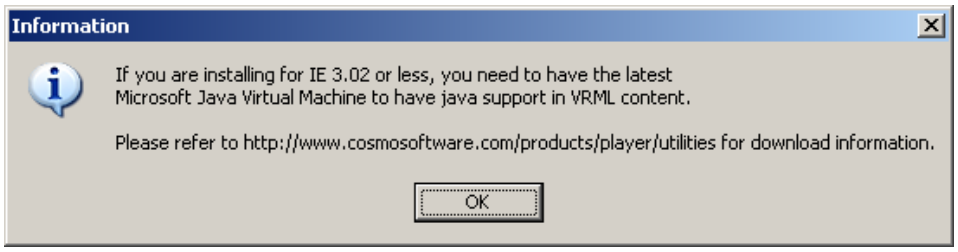
3. Clicar no botão **Yes**.



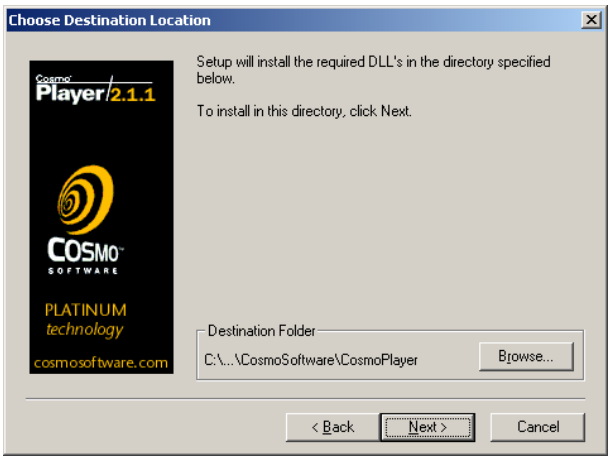
4. Selecionar as opções **“Previewing in Cosmo Authoring Applications”** e **“Other (unsupported browsers)”**, e clicar no botão **Next**.



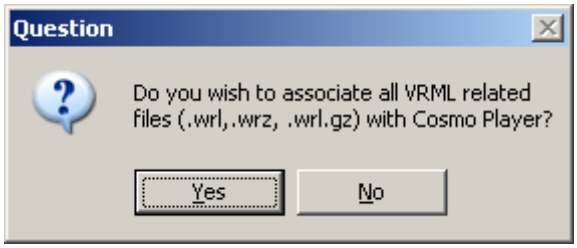
5. Selecionar o browser **"Microsoft Internet Explorer – Unsupported Version"** e clicar no botão **Next**.



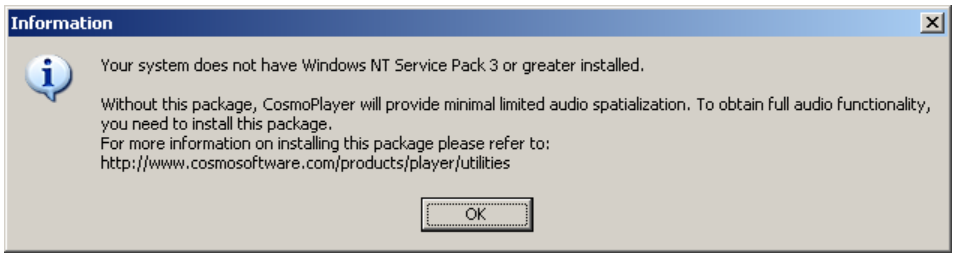
6. Clicar no botão **OK**.



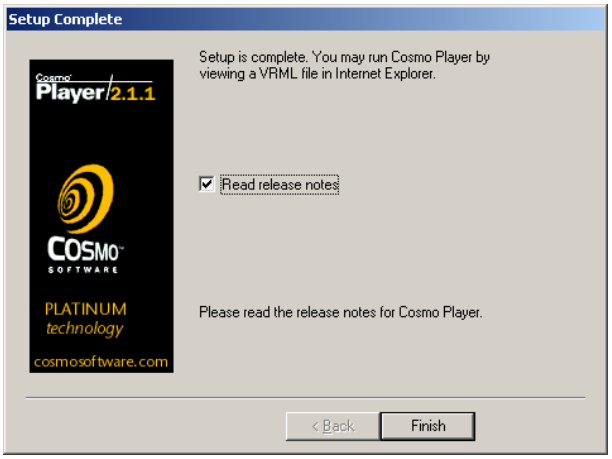
7. Clicar no botão **Next**.



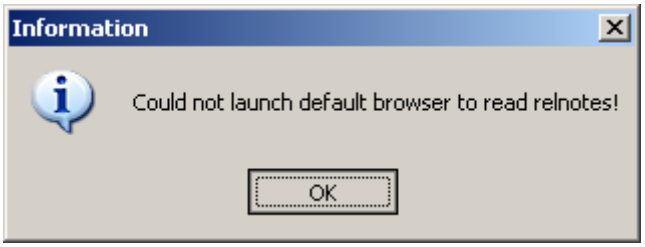
8. Clicar no botão **Yes**.



9. Clicar no botão **OK**.



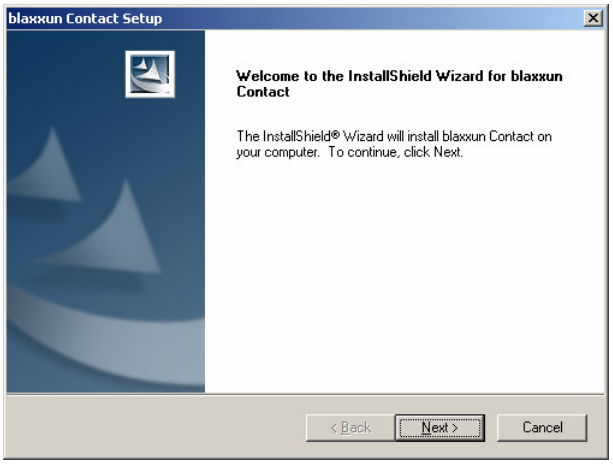
10. Clicar no botão **Finish**.



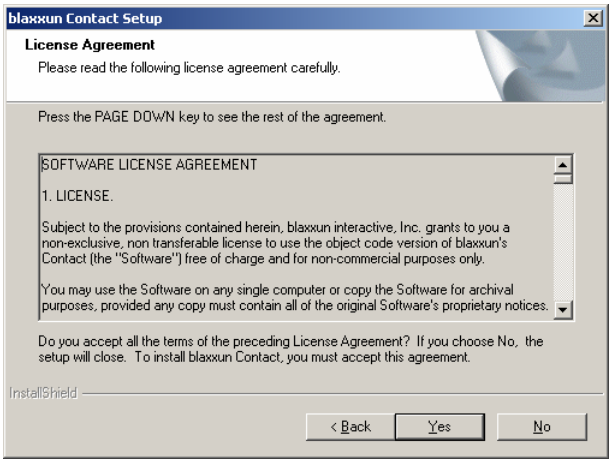
11. Clicar no botão **OK** para finalizar a instalação.

B.1.12. INSTALAÇÃO DO PLUG-IN VRML BLAXXUN

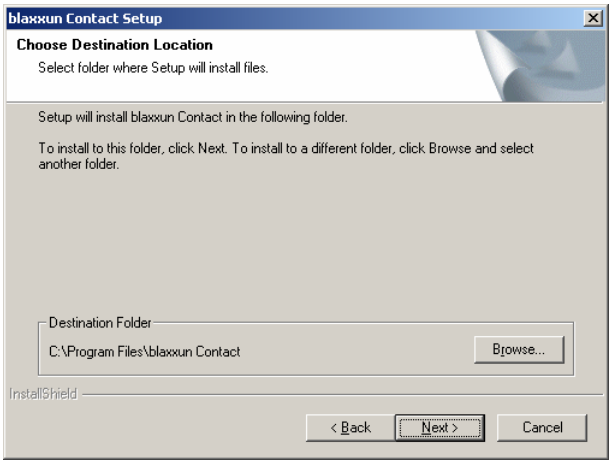
1. Executar o programa **blaxxunContact53.exe** para instalar o plug-in VRML Blaxxun Contact 5.3.



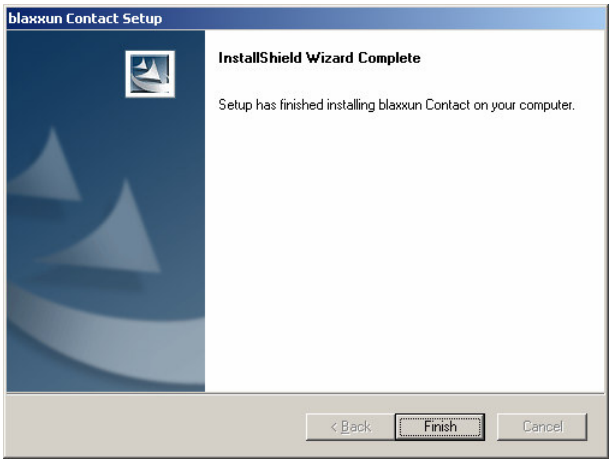
2. Clicar no botão **Next**.



3. Clicar no botão **Yes**.



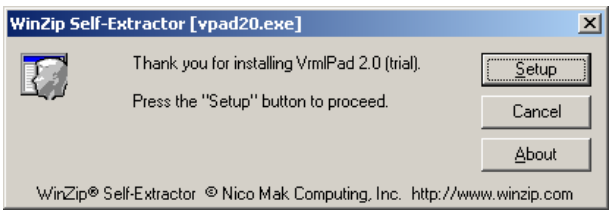
4. Clicar no botão **Next**.



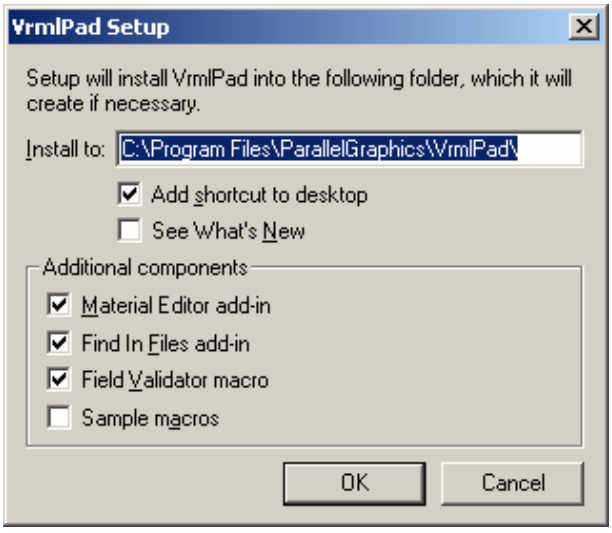
5. Clicar no botão **Finish** para finalizar a instalação.

B.1.13. INSTALAÇÃO DO EDITOR PARA VRML VRMLPAD

1. Executar o programa **vpad21.exe** para instalar editor VRML.



2. Clicar no botão **Setup**.

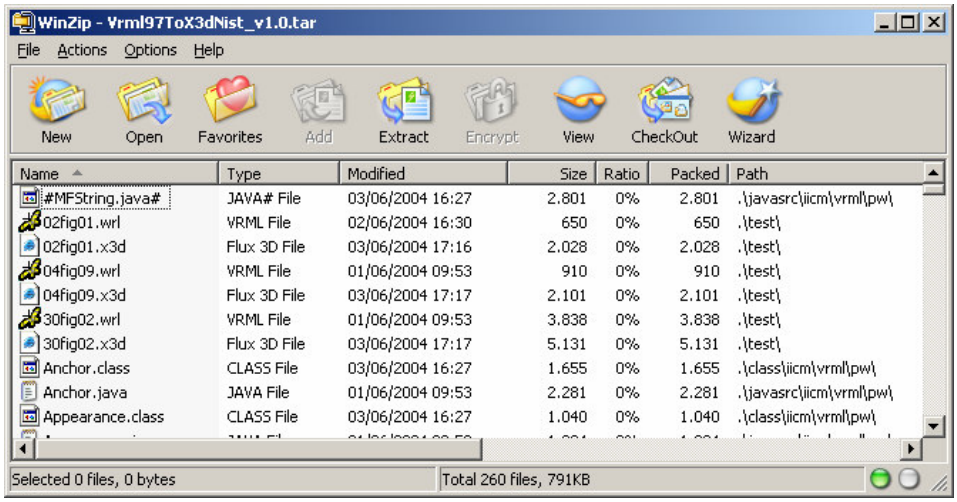


3. Clicar no botão **OK** para finalizar a instalação.

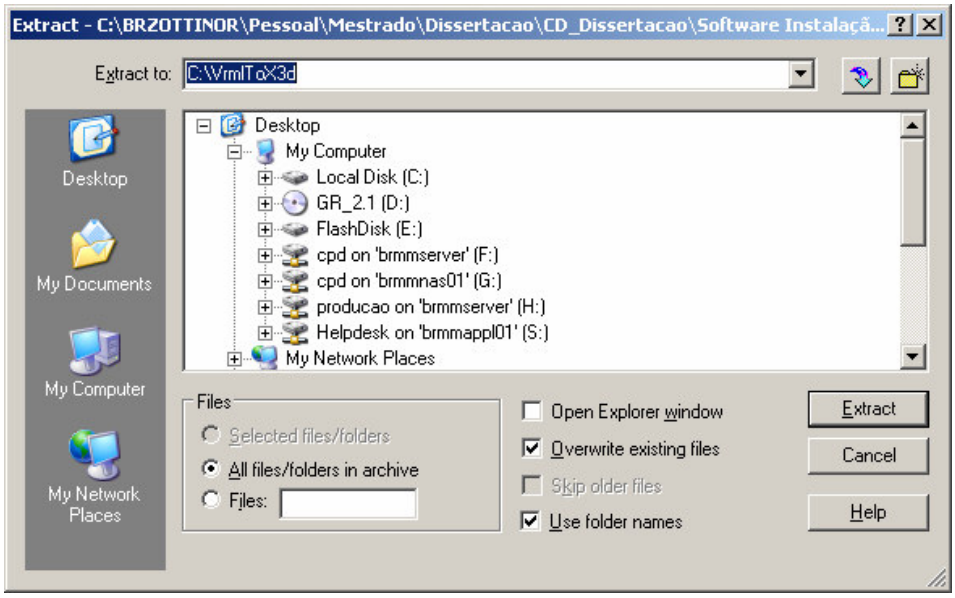
B.1.14. INSTALAÇÃO DO VRML97TOX3DNIST

Utilitário para realizar conversão do formato VRML97 para X3D.

1. Abrir o arquivo compactado **Vrml97ToX3dNist_v1.0.tar** para que possa ser instalado o conversor de formato VRML para X3D.



2. Clicar no botão **Extract.**



3. No campo Extract to, digitar o nome **C:\VrmlToX3d** da nova pasta que será criada no disco C.

4. Clicar no botão **Extract.**

Para utilizar o utilitário basta executar o comando localizado na pasta **C:\VrmlToX3d\test\v2x3dtest.bat** informando o nome da cena VRML e nome da cena a ser convertida em X3D, conforme segue o exemplo: v2x3dtest.bat VCMMS3d.wrl VCMM.x3d.

Apêndice C. CONTEÚDO CD

O CD em anexo contém todo os arquivos e informações utilizadas durante a elaboração deste trabalho, bem como todos os códigos fontes, softwares e utilitários que podem ser copiados e instalados de acordo com seus respectivos direitos de uso.

Para utilizá-lo basta abrir o arquivo **index.html** e mais informações poderão ser encontradas.