

UNIVERSIDADE ESTADUAL PAULISTA
“Júlio de Mesquita Filho”
Programa de Pós - Graduação em Ciência da Computação

ANTHONY FERREIRA LA MARCA

**SISTEMA GERADOR DE APOIO A UM DICIONÁRIO
TEMÁTICO VISUAL-GESTUAL BASEADO EM
REALIDADE VIRTUAL**

BAURU
2011

ANTHONY FERREIRA LA MARCA

**SISTEMA GERADOR DE APOIO A UM DICIONÁRIO
TEMÁTICO VISUAL-GESTUAL BASEADO EM
REALIDADE VIRTUAL**

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração em Processamento de Imagens e Visão Computacional, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. José Remo Ferreira Brega

BAURU
2011

RESUMO

Devido à popularização e a necessidade de se aprender Língua de Sinais, surgem ferramentas que auxiliam neste tipo de comunicação. No Brasil, por exemplo, o governo disponibiliza profissionais de Língua de Sinais para auxiliarem as comunidades, no entanto, se pensar na hipótese de que pessoas vivem em lugares de difícil acesso, há uma dificuldade na locomoção dos profissionais a tais lugares, além de trazer gastos excessivos ao governo. Assim, uma ferramenta que simule tais profissionais, explorando amplamente os recursos tecnológicos (custo-benefício) para enriquecer e disponibilizar a ferramenta para as comunidades, se torna viável a fim de suprir esta lacuna.

Com o intuito de resolver esta necessidade, o trabalho desenvolve um sistema gerador de apoio a um dicionário temático com características de Sistemas Avançados de Realidade Virtual, estando direcionado à Língua de Sinais. O sistema propõe o desenvolvimento de uma aplicação capaz de proporcionar a visualização estereoscópica em Língua de Sinais, a partir de representações de modelos 3D, além de permitir a descrição dos modelos em três idiomas, garantindo a usabilidade do sistema.

A fim de o sistema suportar melhorias, propiciando a imersão e a interação do usuário, o sistema gerador de apoio a um dicionário temático implementa uma estrutura de multiprojeção baseada em aglomerados gráficos, ficando a cargo de uma biblioteca, a sincronização e a distribuição dos dados entre os nós de renderização.

O sistema também oferece recursos de edição de movimentos, sendo estes, posteriormente recuperados pelo sistema gerador de apoio a um dicionário temático para a visualização e representação dos sinais em um Ambiente Virtual. Os recursos utilizados disponibilizam dois dispositivos não convencionais para o rastreamento corporal, os controladores Wiimotes e a Luva P5-Glove.

O trabalho apresenta a incorporação de todas estas funcionalidades de interpretação e edição de movimentos em Língua de Sinais, em um único ambiente de desenvolvimento, a partir da biblioteca de sincronização e de chamadas remotas, passando ao usuário a existência de um único sistema.

Palavras – chaves: Sistema Avançado de Realidade Virtual, Realidade Virtual, Língua de Sinais, Dicionário Temático.

ABSTRACT

Due to the popularity and need to learn sign language, there are tools that help in this type of communication. In Brazil, for example, the government offers professional Sign Language to assist communities, however, if think the hypothesis that people live in places difficult access, there is a difficult in movement of professionals to such places, besides bringing government overspending. Thus, a tool that simulates a professional, fully exploiting the technological resources (cost-benefit) to enrich and provide a tool for communities, it becomes possible order to fill this gap.

In order to solve this need, the dissertation develops a generating system a thematic dictionary with features of an advanced Virtual Reality system, being directed to sign language. The system proposes to develop an application capable of providing stereoscopic visualization of sign language, from representation de 3D objects, beyond allows the description of objects in three languages, ensuring the usability of the system.

In order to system support improvements, enabling user interaction and immersion, the generating system to support a thematic dictionary implements a multprojection structure based on graphics clusters, getting the responsibility a library, data synchronization and distribution between rendering dictionary nodes.

The system also provides resources for editing movements, and these, later, are retrieved by generating system to support a thematic dictionary for the visualization and representation of signals in a Virtual Environment. The resources used, provide two unconventional devices for tracking body, the Wiimote controllers and P5 glove.

The work show the incorporation of all these interpretation and editing functionally of movements in Sign Language, in a single development environment from synchronization library and the remote calls, passing the user to existence of a single system.

Keywords: Advanced Virtual Reality System, Virtual Reality, Sign Language, Thematic Dictionary.

LISTA DE FIGURAS

Figura 1 – Tela do MS-DOS. (ORLANDO, 2006)	22
Figura 2 – Exemplo de manipulação indireta com o uso de <i>scroll bars</i> (BAPTISTA, 2007)..	24
Figura 3 – Configuração de mãos (SCHNEIDER, 2008).....	31
Figura 4 - Configuração final do sinal laranja (a) e do sinal aprender (b) (SCHNEIDER, 2008)	31
Figura 5 – Movimento nos Gestos TRABALHAR (a) e EM-PÉ (b) (SCHNEIDER, 2008) ...	32
Figura 6 - Formação do sinal do substantivo TELEFONE (a) e do verbo TELEFONAR (b) (SCHNEIDER, 2008)	33
Figura 7 – Alfabeto Manual e Números em LIBRAS (NASCIMENTO, 2008)	34
Figura 8 – Determinação de pontos com o referente presente (SCHNEIDER, 2008)	36
Figura 9 – Dicionário de LIBRAS <i>On-Line</i> (FUSCO, 2004).....	38
Figura 10 – Participação de Humanos Virtuais em um Ambiente Virtual (PLETSCH, 2006)	41
Figura 11 – Modelagem de Humano Virtual (NEDEL; SCHNEIDER, 2006)	44
Figura 12 - Comunicação por Gestos (MODESTO, 2006 apud DOUVILLE et. al., 1996)	46
Figura 13 - Graus de Liberdade de um Humanóide (WELTER JR; HOUNSELL, 2004).....	47
Figura 14 - Caverna Digital Gerenciada por Aglomerados Gráficos (GUIMARÃES, 2004)..	49
Figura 15 – Arquitetura Glass (GUIMARÃES, 2004)	50
Figura 16 – Diagrama de Classe do arcabouço Glass (GUIMARÃES, 2004)	51
Figura 17 – Aplicação Glass na arquitetura Sem Replicação (GUIMARÃES, 2004)	52
Figura 18 – Aplicação Glass com arquitetura Com Replicação (GUIMARÃES, 2004)	53
Figura 19 - Diagrama de Classe Protocolo (GUIMARÃES, 2004)	54
Figura 20 – Processo de Sincronização (GUIMARÃES, 2004).....	55
Figura 21 – Utilização do padrão XML para integração de aglomerados remotos. (GUIMARÃES, 2004).....	57
Figura 22 - Arquitetura do Sistema Gerador de Apoio de um Dicionário Temático	61
Figura 23 - Modularização do Sistema.....	63
Figura 24 - Módulo Interpretador do Sistema de Projeção	64
Figura 25 - Diagrama de Caso de Uso Interpretador.....	65
Figura 26 - Diagrama de Caso de Uso Editor.....	66
Figura 27 - Diagrama de Classe Editor	67
Figura 28 – Diagrama de Classe Interpretador.....	68
Figura 29 - Grafo de Cena do AV	70

Figura 30 - Grafo de Cena do HV	71
Figura 31 - Grafo de Cena do AV do objeto X3D.....	71
Figura 32 - Planta Baixa do Sistema de Projeção (DIAS et. al., 2010).....	73
Figura 33 - Estereoscopia	74
Figura 34 - Classe “NewMouseRotate”	75
Figura 35 - Classe “NewMouseTranslate”	76
Figura 36 - Método <i>prerender</i> ()	77
Figura 37 - Carregar X3D.....	78
Figura 38 - Método <i>mouseClicked</i> ().....	79
Figura 39 - Método <i>prerender</i> () X3D.....	80
Figura 40 - Comandos SSH.....	81
Figura 41 - Base de Dados.....	83
Figura 42 - Interface inicial do sistema	84
Figura 43 - Interface dos Módulos Editores	86
Figura 44 - Interface gráfica do Editor Wiimote	87
Figura 45 - Interface gráfica do Editor da Luva	88
Figura 46 - Interface Descrição	89
Figura 47 - Objeto X3D.....	90
Figura 48 – Humano Virtual.....	91
Figura 49 - Estrutura de conexão do dispositivo Wiimote	92
Figura 50 - Movimentos do Wiimote na Coordenada Z.....	94
Figura 51 - Movimento dos Wiimotes na Coordenada X.....	94
Figura 52 - Movimento dos Wiimotes nos dois Membros	95
Figura 53 - Movimentos da Luva P5-Glove.....	95
Figura 54 - Módulo Interpretador no Sistema de Projeção	96
Figura 55 - Representação da Palavra Soldado	97
Figura 56 - Descrição de Palavras	97
Figura 57 - Representação de Casa no Sistema de Projeção	98
Figura 58 - Representação da Palavra Árvore	99
Figura 59 - Esquema para um sistema de estéreo passivo por polarizador (ALBERTO et. al., 2011).....	113
Figura 60 - Tipos de Paralaxe (RAPOSO et. al., 2004).....	114
Figura 61 - Classe Movimento	115

Figura 62 - Método Construtor da Classe "ExecutaMovimento"	116
Figura 63 - Método rotBracoDireito.....	117
Figura 64 - Métodos set() do Braço Direito	118
Figura 65 - Método GravaMovimentoBD()	118
Figura 66 - Infravermelho e Wii Motion Plus (CONTROLLERS AT NINTENDO: WII CONSOLE, 2010).	120
Figura 67 - Coordenadas do Wii Remote (CONTROLLERS AT NINTENDO: WII CONSOLE, 2010)	121
Figura 68 - Método AngloReferencialDegreesInvertido()	125
Figura 69 - Método ireAxToDegrees().....	125
Figura 70 - Método onIrEvent()	126
Figura 71 - Método onMotionSensingEvent().....	126
Figura 72 - Método onIrEvent()	127
Figura 73 - Método <i>onStatusEvent()</i>	128
Figura 74 - Método <i>onButtonsEvent</i>	128
Figura 75 - Luva P5-Glove e a Torre Infravermelha (DAVISON, 2007)	130
Figura 76 - A classe CP5DLL (DAVISON, 2007).....	131
Figura 77 - Dados da Luva P5-Glove (DAVISON, 2007)	132
Figura 78 - Eixos de coordenadas da Luva P5-Glove (DAVISON, 2007).....	133
Figura 79 - A orientação da Luva P5-Glove – Pitch, yaw e roll (DAVISON, 2007).....	133
Figura 80 - LEDs da Luva P5-Glove (DAVISON, 2007)	134
Figura 81 - Construtor da classe "Glove"	135
Figura 82 - Detalhes da Luva	135
Figura 83 - Métodos update() e gets()-	136
Figura 84 - Método close()	136
Figura 85 - Método InicializaLuva() e parte do Método startPolling()	137
Figura 86 - Teorema de Tales - Dedos Indicador e Mindinho	137
Figura 87 - Método <i>RegulaValorIndicador_Mindinho()</i>	138
Figura 88 - Operação de Subtração e Atualização de Valores	138
Figura 89 - Método de Execução e de Gravação de Movimentos.....	139
Figura 90 - Movimento do Pulso.....	139

LISTA DE TABELAS

Tabela 1 - Comparações de Sistema de Língua de Sinais.....	41
Tabela 2 - Argumentos da Modularização do Sistema.....	64
Tabela 3 - Argumentos da Representação de LIBRAS no Sistema de Projeção.....	66
Tabela 4 - Descrição da Interface Inicial do Sistema.....	86
Tabela 5 - Descrição da Interface de Editoração.....	87
Tabela 6 - Descrição dos aspectos da Interface de Descrição.....	91

LISTA DE SIGLAS

2D – *Two Dimensional*

3D – *Three Dimensional*

API – *Application Programming Interface*

AV – *Ambiente Virtual*

BSL – *British Sign Language*

DICELib – *Distributed Cave Engine Library*

DOF – *Degree of Freedom*

FENEIS - *Federação Nacional para Educação e Integração de Surdos*

GNU - *General Public License*

GUI – *Graphical User Interface*

H-ANIM – *Humanoid Animation Working Group*

HST – *Hand Sign Translator*

HV – *Humano Virtual*

IHC – *Interação Humano – Computador*

LIBRAS – *Língua Brasileira de Sinais*

LN – *Linguagem Natural*

MS-DOS – *Microsoft Disk Operational System*

OSV – *Objeto-Sujeito-Verbo*

PDA – *Personal digital assistants*

PS2 – *Porta de Conexão*

RA – *Realidade Aumentada*

RV – *Realidade Virtual*

SCP – *Segury Copy*

SQL - *Structured Query Language*

SSH – *Segure Shell*

SOV – *Sujeito-Objeto-Verbo*

SVO – *Sujeito-Verbo-Objeto*

TCP/IP - *Transmission Control Protocol/Internet Protocol*

UDP – *User Datagram Protocol*

USB - *Universal Serial Bus*

VRML – *Virtual Reality Modeling Language*

X3D – *Extensible 3D Graphics*

XML – Extensible Markup Language

WIMP – Windows, icons, menus and a pointing device

Sumário

Capítulo 1 – Introdução	13
Capítulo 2 – Aspectos Gerais de comunicação e interface.....	15
2.1 – História da Comunicação	15
2.2 – Conceitos sobre Comunicação.....	17
2.3 – Métodos de Comunicação	18
2.4 – Interface – Uma Visão Geral	19
2.5 – A Interface Gráfica - GUI (<i>Graphical User Interface</i>)	21
2.6 – Comunicação Humano-Computador	23
2.7 – Interação Natural	25
2.8 – Tecnologias de Rastreamento	25
2.9 – Dispositivos de Rastreamento	27
2.10 – Considerações Finais	28
Capítulo 3 – Língua de Sinais	29
3.1 - LIBRAS.....	30
3.2 - Fonologia da Língua de Sinais	30
3.3 – Morfologia da Língua de Sinais	32
3.4 – Sintaxe da Língua de Sinais	35
3.5 – Convenções de LIBRAS.....	35
3.6 – Tecnologia e Deficientes Auditivos	36
3.7 - Ferramentas de Língua de Sinais	38
3.8 – Considerações Finais	40
Capítulo 4 – Humanos Virtuais	41
4.1 – Humanos Virtuais	41
4.2 – Avatares e Agentes Virtuais	42
4.3 – Aspectos dos Humanos Virtuais.....	43
4.3.1 – Aparência.....	43
4.3.2 – Movimento	44
4.3.3 – Interatividade	44
4.3.4 – Comunicação	45
4.3.5 – Gestos	45
4.4 – Articulações em Humanos Virtuais.....	46
4.5 – Considerações Finais	47
Capítulo 5 – Tecnologias Envolvidas.....	48
5.1 – Aglomerados Gráficos.....	48
5.2 – A Biblioteca Glass	49
5.2.1 – O arcabouço Glass	51
5.2.2 – Componente Instanciação.....	52
5.2.3 – Componente Protocolo	53
5.2.4 – Componente Plugins.....	54
5.2.5 – Aplicações Glass	56

5.3 – Java 3D	58
5.4 – X3D (<i>Extensible 3D Graphics</i>)	58
5.5 – Protocolos de Comunicação – SSH e SCP	59
5.6 – Considerações Finais	60
Capítulo 6 - Descrição do Projeto	61
6.1 – Modularização do Ambiente	62
6.2 – Modelagem do Sistema Gerador do Dicionário Temático	65
6.3 – Ambientes Virtuais	69
6.4 – Classe Principal	72
6.5 – Interpretador de Língua de Sinais.....	73
6.6 – Editor da Língua de Sinais	82
6.7 – Base de Dados	83
6.8 – Entendendo a Interface do Sistema	84
6.9 – Considerações Finais	92
Capítulo 7 - Resultados	93
7.1 – Testes do Editor de Movimentos	93
7.2 – Testes na Representação de Sinais	96
7.3 – Considerações Finais	99
Capítulo 8 - Conclusões.....	100
Referência Bibliográfica.....	102
ANEXO A - LEI Nº 10.436, DE 24 DE ABRIL DE 2002.....	109
APÊNDICE A - DIAGRAMA DE CLASSE DO MÓDULO EDITOR.....	110
APÊNDICE B - DIAGRAMA DE CLASSE DO MÓDULO INTERPRETADOR.....	111
APÊNDICE C - ESTEREOSCOPIA.....	112
APÊNDICE D - GERAÇÃO E GRAVAÇÃO DE MOVIMENTOS	115
APÊNDICE E - CONTROLADORES WIIMOTE.....	120
E.1 – Bibliotecas de conectividade.....	122
E.2 – API WiiuseJ	122
E.3 – Projetos Wiimote.....	123
E.4 – Módulo Editor	124
APÊNDICE F - A LUVA P5-GLOVE.....	130
F.1 - A API JAVA	131
F.2 – Examinando os Dados da Luva.....	132
F.3 – Módulo Editor	134

Capítulo 1 – Introdução

Com a crescente evolução da Realidade Virtual (RV) e com as novas bibliotecas disponíveis para a implementação e visualização de ambientes tridimensionais, é possível desenvolver aplicações de baixo custo com ferramentas livres e mecanismos de sincronização para alcançar maior interoperabilidade. A combinação desta com computadores mais rápidos e ferramentas de softwares poderosos, leva ao desenvolvimento de sistemas altamente imersivos e interativos, simulando as leis que regem o mundo real.

Assim, a RV sendo a técnica mais avançada de interface entre o usuário e o computador, proporcionando imersão, navegação e interação do usuário com o ambiente sintético tridimensional (KIRNER; SISCOOTTO, 2007), faz a possibilidade do uso de Língua de Sinais em Sistemas Avançados de Realidade Virtual, viabilizando uma alternativa computacional para a comunicação entre as comunidades que usufruem da Língua de Sinais.

Com a possibilidade da utilização de recursos oferecidos pela RV, um sistema que aborde estes conceitos para melhorar e estimular o aprendizado em Língua de Sinais se torna visível e interessante, visto que há uma dificuldade na comunicação entre as comunidades que utilizam Língua de Sinais.

Dentre os recursos oferecidos pela RV, a multiprojeção baseada em aglomerados gráficos, possibilitam o desenvolvimento de uma aplicação que propicia o aumento da imersão e da interação, sendo possível a implementação de estereoscopia e de representações de sinais em tamanho real, garantindo o envolvimento do usuário com o sistema.

Com a finalidade de aumentar a usabilidade do sistema por permitir que comunidades de línguas diferentes possam o utilizar, modelos virtuais 3D são vinculados com movimentos de sinais e descritos em três idiomas.

Além do sistema utilizar estes recursos para a representação em Língua de Sinais, o sistema possibilita a edição dos movimento através de dispositivos não convencionais, permitindo o rastreamento corporal. Dentre os dispositivos estão: os controladores Wiimotes e a Luva P5-Glove.

Enfim, tem-se o intuito de desenvolver um sistema gerador de apoio de um dicionário temático visual-gestual que possa auxiliar e estimular na comunicação entre as comunidades envolvidas, sendo estas, pessoas principiantes ou especialistas em Língua de Sinais que vivem ou não em lugares de difícil acesso. Desta forma um sistema que explore amplamente os recursos de RV, disponibilizando dispositivos não convencionais e um sistema de projeção

para a representação e edição de movimentos em Língua de Sinais se mostra viável. A fim de tornar o sistema disponível a diferentes Língua de Sinais, este permite a descrição de modelos virtuais em três idiomas, além de permitir a integração dos módulos interpretador e editor através de protocolos de comunicação, dando ao usuário a visão de um único ambiente.

O trabalho está organizado em mais sete capítulos.

O Capítulo dois relata sobre comunicação e interface, apresentando a história da comunicação até os tempos atuais, mostrando conceitos gerais e específicos agregados a métodos interativos, considerações gerais e específicas sobre interface, interface gráfica e interação Humano-Computador. Apresentando também fundamentos teóricos sobre tecnologias e dispositivos de rastreamento.

O Capítulo três trata Língua de Sinais, mostrando conceitos e aspectos envolvidos na construção e recepção dos sinais, além de focar-se principalmente no caso da Língua Brasileira de Sinais (LIBRAS), apresentando a sua sintaxe (fonológico, morfológico, sintático e semântico), suas convenções e as tecnologias disponíveis aos deficientes auditivos, ilustrando sistemas já disponíveis.

O Capítulo quatro aborda Humanos Virtuais (HVs), mostrando conceitos gerais, diferenças entre avatares e agentes virtuais, definições e exemplos de seus aspectos e articulações.

O Capítulo cinco apresenta as tecnologias utilizadas na implementação do sistema, relatando uma breve introdução sobre cada. Apenas a biblioteca Glass é apresentada mais detalhadamente, pois é uma biblioteca recente e pouco difundida, além de ser um dos principais componentes utilizado no trabalho.

O Capítulo seis descreve o projeto todo, detalhando as fases do projeto e de desenvolvimento.

O Capítulo sete descreve os testes e resultado do trabalho, tanto do módulo editor quanto do módulo interpretador, o sistema gerador do dicionário temático.

Por fim o Capítulo oito, apresentando a conclusão e possíveis trabalhos futuros.

Capítulo 2 – Aspectos Gerais de comunicação e interface

O processo de comunicação é um dos fenômenos mais importantes da história da espécie humana que vem se modificando ao longo dos tempos. Compreendê-lo, implica voltar no tempo e buscar suas origens, desde o surgimento da fala, do desenvolvimento das linguagens até as tecnologias atuais utilizadas para o seu aprimoramento (interfaces) (PERLES, 2007).

As sessões seguintes descrevem a evolução da comunicação, desde as primeiras representações gráficas até as tecnologias atuais utilizadas para a representação da comunicação. Uma das tendências para a representação da comunicação é a interação natural, dando origem a tecnologias e a dispositivos de rastreamento corporal.

2.1 – História da Comunicação

A linguagem, a cultura e a tecnologia são elementos indispensáveis no processo de comunicação, sendo que a linguagem é a cognição simbólica mais apropriada e óbvia para tal comunicação. A cultura e a tecnologia são aspectos que ao longo do tempo foram sofrendo grandes mudanças, afetando diretamente na maneira de comunicação das comunidades da época. (PERLES, 2007).

As primeiras representações gráficas eram uma tentativa de copiar a imagem real registrando-a como observada. Assim surgem os primeiros símbolos, que representam conceitos não completamente explícitos, mas exigindo certo conhecimento prévio para serem significativos. Desta forma, relacionando símbolos e fonemas, evolui-se de hieróglifos para a linguagem escrita, sendo a primeira Língua oficial o sânscrito (ROCHA et. al., 2009).

Com padrões bem definidos e comuns da comunicação, o desafio passava-se no alcance da mensagem, fazendo com que o homem da época se preocupasse em desenvolver meios que possibilitassem um maior alcance e maior rapidez na transmissão.

As tribos indígenas utilizavam tambores para transmitir códigos sonoros e fogo como códigos visuais para comunicação a distância. As civilizações utilizavam mensageiros encarregados de levar pessoalmente as mensagens escritas por seus imperadores (ROCHA et. al., 2009).

Com o conhecimento e relativo domínio da eletricidade, no século XIX, foi construído o telégrafo utilizando linhas de energia (Samuel Morse), na qual mais tarde foi utilizado para

a transmissão da voz e a invenção do telefone, por Alexander Graham Bell (1876) (ROCHA et. al., 2009).

Com a descoberta das ondas eletromagnéticas por Heinrich Hertz em 1866, possibilitou a comunicação a grandes distância sem a necessidade de cabos e fios. Tanto a eletricidade quanto as ondas eletromagnéticas são hoje utilizadas na comunicação (telefone, rádio, televisão e Internet), entretanto estão gradativamente sendo substituídas pelo novo canal de comunicação, a luz (ROCHA et. al., 2009).

O uso de recursos Multimídia nasce para tornar a comunicação mais interessante e completa, pois o interesse do receptor é um importante fator no sucesso da comunicação, assim como a coesão e a coerência da mensagem. Desta maneira, pode-se dizer que o uso de recursos Multimídia deve ser aplicado com técnicas e cuidados, visando bons resultados ao receptor. A soma destes recursos, simultaneamente, torna a mensagem mais atrativa preenchendo e prendendo mais a atenção do receptor (ROCHA et. al., 2009).

Com os avanços tecnológicos permitindo o desenvolvimento de computadores, nasce um novo meio de comunicação, a comunicação homem-máquina. Com o processo evolutivo das tecnologias, surge a Internet para revolucionar os meios de comunicação na qual milhões de pessoas podiam comunicar-se de maneira rápida, barata e a longa distância.

Assim, a comunicação veio alterando profundamente a dinâmica das relações entre as pessoas, pois tem permitido, graças à tecnologia, a multiplicação quase ilimitada de contato entre os indivíduos, ultrapassando barreiras tanto culturais quanto de distância.

Um exemplo desta comunicação em massa é um *weblog*, ou *blog* como é popularmente conhecido. Este é uma espécie de diário virtual hospedado na Internet, e por isso acessível a milhões de pessoas. Neste “diário” que não necessita de uma atualização regular, o autor pode escrever sobre o assunto que desejar, quando desejar, sem intermediários. Com a sua chegada, basta apenas um computador conectado à Internet para que qualquer mortal possa fazer comunicação em massa (GARAVELLO, 2009).

Há *blogs* individuais e coletivos, com mais de um autor. Os textos, ou *posts*, são dispostos de forma cronológica, de modo que o mais recente apareça no topo da página. A maioria desses sites dispõe também de um sistema de comentários, que gera uma lista de discussão para cada *post* (GARAVELLO, 2009).

A simplicidade do sistema, sua gratuidade e as inúmeras possibilidades que oferecem, atraem milhões de pessoas pelo mundo, transformando o novo meio em uma febre, quase um fenômeno sociológico (ARAUJO, 2006).

Logo, com o desenvolvimento da informática e da eletrônica houve uma grande revolução nos meios convencionais da comunicação, sendo os formatos analógicos substituídos gradativamente por formatos digitais, na qual armazenam a informação em códigos numéricos, seja texto, áudio ou vídeo (ROCHA et. al., 2009).

Contudo, ainda existe dificuldade de certos grupos de pessoas em se comunicarem, devido a níveis de deficiência presentes nos mesmos (auditiva, por exemplo), surgindo métodos e ferramentas que auxiliam tais pessoas neste tipo de comunicação, como por exemplo, a Interface Gráfica.

2.2 – Conceitos sobre Comunicação

De imediato, pode-se classificar comunicação, segundo o dicionário Aurélio (FERREIRA, 2010), como o “ato ou efeito de transmitir mensagens por meio de métodos e/ou processos convencionados”, a exemplo de um, a linguagem verbal ou não-verbal. Mas tal classificação é insuficiente para descrever o fenômeno, além de servir ao longo processo de desenvolvimento da linguagem para simplificar um dos fenômenos mais importantes da socialização, cujos limites sempre estão por vir, conforme ressalta Baitello (1998).

A comunicação é algo que se aprende a fazer. Não somente aprende a se comunicar, mas também se usa a comunicação para aprender como nos comunicar. Nossas experiências incluem a lembrança de pessoas falando e gesticulando para nós. Aprende-se como fazer as coisas pela prática, empiricamente, por meio da tentativa e do erro. A maior parte de nossas experiências em comunicação é fruto de aprendizado.

Falar e escrever não são habilidades naturais, são aprendidas com os pais, amigos, familiares e na escola. À medida que se cresce, aprende-se algo das experiências de comunicação, pois compreende-se que isso é útil.

A comunicação humana é um processo que envolve a troca de informação utilizando os sistemas simbólicos como suporte para este fim. Para a troca de entendimento é necessário mais do que palavras, como também emoções e a necessidade de tornar comum conhecimentos, idéias, instruções ou qualquer outra mensagem, sendo verbal, escrita ou corporal.

A tarefa de se comunicar só está concluída quando há a compreensão, aceitação e ação resultante do indivíduo receptor, tendo como finalidade afetar o seu comportamento.

Para a realização da comunicação existem alguns elementos indispensáveis:

- Emissor: o que emite a mensagem;

- Receptor: o que recebe a mensagem;
- Mensagem: o conjunto de informações transmitidas;
- Código: a combinação de signos utilizados na transmissão de uma mensagem, sendo que a comunicação só se concretizará se o receptor souber decodificar a mensagem;
- Canal de Comunicação: meio pela qual a mensagem é transmitida (propagada);
- Contexto: a situação a que a mensagem se refere;
- Ruído: qualquer perturbação na comunicação; e
- Feedback: a retroalimentação da informação, permitindo que o receptor reaja em cima da informação (FERREIRA, 2007).

2.3 – Métodos de Comunicação

Com a introdução da internet na natureza social, muitas discussões referentes a formatos e conteúdos efervesceram o campo da comunicação e, especialmente a partir da difusão do conceito de Web 3.0, ficou evidente que as interações por meio da interface são o principal meio de sociabilidade. Como confirma Johnson (2001), “a interface já alterou o modo como usamos computadores (...). Mas está fadada a mudar outros domínios da experiência contemporânea” (JOHNSON, 2001).

Pelo fato da comunicação estar crescendo em larga escala, passou-se a utilizar métodos que passaram a auxiliar e a potencializar o processo de produção, envio e recepção das mensagens. A tecnologia passou a fazer parte da comunicação humana, assim como, passou a participar da maioria das atividades desenvolvidas pela humanidade ao longo do seu desenvolvimento.

Devido ao avanço das tecnologias, houve a possibilidade de aumentar o potencial da informação e conectar seus conteúdos utilizando inúmeros métodos de interatividade que fazem dos usuários um agente de comunicação em rede por meio de interfaces (FRANCO, 2006).

Para o entendimento desta comunicação, é essencial impor a idéia de interface pelo fato da mesma ser responsável por realizar a comunicação entre meios heterogêneos, sendo um conjunto de dispositivos a partir dos quais os usuários interagem a fim de estabelecer uma comunicação entre as partes envolvidas (FRANCO, 2006).

A Visualização da Informação é outro método que auxilia na comunicação, explorando o sistema cognitivo do ser humano, tendo como ideia apresentar ao usuário uma forma de visualizar informações utilizando a sua percepção principal: a visão.

Segundo Card (1999), Visualização da Informação é “o uso de representações visuais de dados abstratos suportadas por computador e interativas para ampliar a cognição”, melhorando assim, o entendimento e o aproveitamento do conhecimento exposto.

Outro método muito utilizado para a comunicação é a RV, sendo uma técnica de interface avançada na qual proporciona visualização, imersão e interação do usuário com elementos do AV tridimensional gerado por computador e em tempo real (KIRNER, SISCOOTTO, 2007). Tal técnica envolve estudos e recursos ligados com a percepção, hardware, software, interface com o usuário, fatores humanos e aplicações. A elaboração desses sistemas exige domínio em dispositivos não convencionais em computadores de alta tecnologia e computação gráfica 3D, além de tecnologias de multimídia e de abordagens de trabalho cooperativo (BRAGA, 2001).

2.4 – Interface – Uma Visão Geral

De acordo com o dicionário Aurélio (FERREIRA, 2010), a palavra interface provê do meio físico ou lógico através do qual um ou mais dispositivos ou sistemas incompatíveis conseguem comunicar entre si. Além do uso em termos tecnológicos, foi difundido em contextos para indicar ligação, produção em relações e conexão, como por exemplo, a utilização da expressão comercial: “interface entre clientes e fornecedores”.

Segundo Lévy (1993) a noção de interface aborda algo que estabelece contato, comunicação, tradução entre meios diferentes ou algo que produz ou permite relacionamentos.

O termo interface vai de termos mais técnicos até a uma conotação mais genérica, como tradução, transformação e passagem. Deste modo, existe uma dificuldade em descrever o que seriam exatamente as interfaces, devido à intensidade e flexibilidade do conceito. O domínio mais propício em relação ao uso técnico inclui elementos de hardware e software, responsáveis pela interação entre sistemas e destes com usuários, sendo chamada de interfaces de usuário (*user interface*).

O instituto de Interação Humano – Computador (IHC) estuda a interação entre humanos e computadores tendo a Ergonomia e outras áreas correlatas adotando a abordagem de interface do usuário, imergindo assim, a figura da GUI. Desta forma o centro do

pensamento que se analisa é em especial a ideia da tradução da lógica da máquina para a do ser humano e a de controle de equipamentos. Por este caminho, a interface com o usuário seria disponibilizar um dispositivo que mantivesse um canal de comunicação do mesmo com a máquina ou lhe oferecesse um aparato para controlar algum equipamento (ORLANDO, 2006).

Para Lévy (1993) num sentido mais técnico, a palavra interface provê de um artefato que permite a comunicação entre sistemas informáticos ou a conexão destes a redes de comunicação, designando a interface como uma operadora de transcodificação e de administração de fluxos de dados. Em sua perspectiva mais geral propõem interface como: “... todos os aparatos materiais que permitem a interação entre o universo da informação digital e o mundo ordinário”.

Interfaces como *hardware* são artefatos de entrada e saída de informações no sistema, sendo alinhados entre equipamentos como teclados, *mouses*, no nível de entrada, e monitores, impressoras, no nível de saída. Já interface no nível de *software*, pode-se associá-lo à ideia de ambiente operacional, tendo desde os navegadores e sistemas de correio eletrônico para o acesso a Web, permitindo a visualização de informações presentes nas redes até simples processadores de textos em que a interface refere-se ao acesso a recursos, ao ambiente de trabalho, comandos e operações (ORLANDO, 2006).

Outra abordagem técnica para interface é enxergá-la como sendo camadas que vão recobrando os dígitos binários do computador, e deste modo, traduzindo e interpretando a comunicação entre as partes envolvidas (JOHNSON, 2001). Domingues relata vários tipos de interface dentre esta abordagem de camadas, porém trata especificamente de tipos baseados em menus e da GUI. Assim, partindo da natureza de computadores que trabalham com dígitos binários (0 e 1), progressivamente vão se sobrepondo camadas lógicas que permitem uma interação com o equipamento, tornando-o cada vez mais acessível (DOMINGUES, 2001)

De modo geral, a interface em camadas seria como uma progressiva tradução de um ambiente exterior (uso de seres humanos), para o ambiente interior de instruções e circuitos que trabalham com dígitos binários, ou inversamente. Segundo Lévy (1993), o computador pode ser visto como “folheado, uma rede de interfaces sucessivas”.

Para uma interface ganhar espaço entre os usuários, esta deve oferecer conceitos de conforto, eficácia e intuitividade. Entende-se por intuitividade o nível de facilidade e simplicidade com que a interface pode ser utilizada (BAERENTSEN, 2001).

Portanto, percebe-se que ao se pensar em dispositivos de interface, o ideal seria a integração e compatibilidade total das partes, existindo padronizações que facilitariam na troca de informações entre os envolventes provendo formas confortáveis, eficazes e intuitivas de interação entre os usuários finais e o sistema.

2.5 – A Interface Gráfica - GUI (*Graphical User Interface*)

Segundo JOHNSON (2001), a interface gráfica é a grande revolução simbólica dos tempos de hoje sendo a “mágica da revolução digital”. Esta utiliza uma combinação de tecnologias e dispositivos para fornecer uma plataforma com a qual o usuário pode interagir, isto é, esquematizada por um conjunto de elementos reunidos pela sigla WIMP (*windows, icons, menus and a pointing device*) – janelas, ícones, menu e um apontador de tela. Neste sistema utiliza-se um apontador de tela (*mouse*) para controlar a posição do cursor e apresentar informações organizadas em janelas e representadas por meio de ícones. Os comandos disponíveis são compilados por meio de menus e acionado através do dispositivo de ponteiro.

Os recursos surgiram para substituir o modelo que tinha um teclado e um monitor de baixa resolução trabalhando por meio de linhas de comando. Com a noção de espaço-informação introduzido por Engelbart (1962), Johnson assegura que este trabalho foi fundamental para converter o computador de simples equipamentos de cálculos para máquina simbólica, ou seja, uma “interação amigável” entre o computador e usuários.

Engelbart (1962) também é o inventor de outros componentes fundamentais na lógica espaço-informação, como a manipulação direta e o mouse. Segundo ele, para a ilusão de espaço-informação realmente funcionar, é fundamental que o usuário possa simplesmente apontar para algo da tela e poder expandir seus conteúdos ou arrastá-lo para onde fosse conveniente, sem a necessidade de fazê-lo através de linhas de comandos.

A Figura 1 ilustra uma tela que simula a interface de linha de comando do MS-DOS, mostrando o conteúdo de um diretório, chamado de pasta GUI. Na parte inferior da tela, um exemplo de comando para cópia de arquivo, seguindo sua sintaxe rígida.

```

      <DIR>          12/01/04   18:25 .
      <DIR>          12/01/04   18:25 ..
INSTALL  LOG           2,623  13/01/04   2:22 INSTALL.LOG
UNWISE   EXE          148,480  07/10/98  17:16 UNWISE.EXE
DOC       <DIR>          13/01/04   2:21 doc
CTL3D    DLL           19,387  15/06/95   0:00 Ctl3d.dll
CTL3DV2  DLL           19,725  14/10/93  17:57 Ctl3dv2.dll
EXAMPLE1 DAT           54,901  01/02/01   1:40 Example1.dat
MENUS    ZIP           65,763  01/02/01   1:40 Menus.zip
SETUP    EXE          230,912  01/02/01   1:40 Setup.exe
MESGS    ZIP           87,835  01/02/01   1:40 Mesgs.zip
PATTERNS DAT          559,721  01/02/01   1:40 Patterns.dat
PROGS    DAT           198     01/02/01   1:40 Progs.dat
README~1 HTM           9,650  21/02/01  12:27 ReadMe.html
README   TXT           385     05/02/01  15:59 readme.txt
README   WRT          28,416  05/02/01  11:23 Readme.wri
WISISX   DAT          677,120  01/02/01   1:40 Wisisx.dat
WISISY1$ DAT          395,264  01/02/01   1:40 Wisisy1$.dat
WISISY2$ DAT          19,739  01/02/01   1:40 Wisisy2$.dat
WISISZ   DAT          12,853  01/02/01   1:40 Wisisz.dat
      17 arquivo(s)      2,320,972 bytes
      3 pasta(s)         4,759,00 MB livre(s)

C:\isiswtmp>copy mesgs.zip c:\tmp_

```

Figura 1 – Tela do MS-DOS. (ORLANDO, 2006)

Kay (1990) introduziu a ideia de metáfora do *desktop* para solucionar o problema do espaço restrito do monitor. A proposta dava profundidade ao espaço da tela, sendo a tela como uma escrivaninha e cada janela como um “papel”, na qual eram empilhados em uma pilha sendo o atual o topo da mesma.

Segundo Johnson (2001), a ideia de metáfora de *desktop* foi afinada ao longo da história até se tornar popular com o Macintosh e posteriormente com o *Windows*, reconhecendo-se a importância do poder de aproximação e simplificação da metáfora, “o idioma essencial da interface gráfica contemporânea”. Desta forma, já era possível visualizar diversos documentos simultaneamente.

Com o aperfeiçoamento das janelas, um gerenciador de janela foi conduzido para facilitar a interação entre o sistema de janelas e aplicações, este, responsável por lidar com os dispositivos de hardware como o dispositivo de ponteiro e o hardware gráfico. A simulação proporcionada pelos gerenciadores de janelas, incluindo a interação entre janelas e outros elementos gráficos, produzia um ambiente de desktop.

As interfaces representam a informação que trafega na essência da máquina em uma linguagem visual que permite ao usuário manipulá-la. Para Johnson (2001), devido a crescente importância do ciberespaço, “o único acesso a este universo paralelo de zeros e uns” se fará através da interface (JOHNSON, 2001).

Portanto, a interface precisa ser invisível, sua capacidade de produzir uma relação eficiente entre o humano e a tecnologia deve ser tão densa e consistente que ela mesma tende a desaparecer. Precisa ser uma conexão tão fina com elementos cognitivos, motores, sensoriais e agora emocionais do ser humano que as trocas entre os dois reinos estariam propensas a ser diretas, como se não fossem mediadas (ORLANDO, 2006).

2.6 – Comunicação Humano-Computador

Discussões relevantes em torno das relações entre humanos e computadores se deram já na década de 1950 e sendo marcante nos anos de 1960. Neste período, é grande o número de pesquisas que discutem a “interação” entre humanos e computadores. A partir de então, a discussão sobre interação cresce e alastra nas décadas seguintes consolidando em grupos de pesquisas e no mercado com produtos para públicos de maior escala, como foi o caso do Apple Macintosh, em 1984. Desde então, estas idéias cresceram e excederam os limites deste universo criando-se as primeiras críticas e debates. Hoje, refletem em área como Comunicação, Sociologia, Filosofia, Arte e Educação (ORLANDO, 2006).

O ponto de partida para alcançar um conjunto de referências básicas de estudos foram os trabalhos ligados ao amparo da IHC. A IHC tem relativamente um grande número de estudos extensos, incluindo temas advindos de diversas áreas, sendo denominada interdisciplinar (ORLANDO, 2006).

Segundo Rodello (2003), “a ideia de interação é ligada com a capacidade de o computador detectar as entradas do usuário e modificar instantaneamente o AV e as ações sobre ele...”.

A interação Humano-Computador está ligada ao fato de como o usuário comunica-se com a máquina a fim de impor suas exigências, fazendo desta uma ferramenta para simplificar, apoiar e executar uma determinada tarefa. A comunicação ocorre dentro de um contexto social e organizacional influenciada pelo estilo da interface (CUSIN, 2005).

Existem diversas maneiras nas quais o usuário interage com o sistema. De um lado, o usuário provê a informação imediatamente ao computador para a execução de um número limitado de tarefas. De outro, são utilizados dispositivos altamente interativos e paradigmas, como manipulação direta e aplicações de RV, na qual o usuário provê sucessivas instruções que serão avaliadas constantemente pelo sistema (CUSIN, 2005).

Nos modelos de interação há pelo menos dois participantes envolvidos: o usuário e o sistema. Ambos são complexos pelo fato de se comunicarem e “verem” as tarefas de maneira

diferente. O papel da interface é suprir esta lacuna de comunicação e funcionar efetivamente como um tradutor entre as partes envolvidas (usuário e sistema) de modo a permitir a interação. Entretanto, existe a possibilidade de falhas em diversos pontos e por vários motivos.

Segundo Dix (1998), o uso de modelos de interação auxilia a entender o que está “entrando” na interação e identificar as dificuldades em traduzir os anseios do usuário em ações para o sistema, proporcionando fundamentos para comparar interações diferentes e analisar problemas de interação.

A manipulação direta é o modelo de interação mais intuitivo da comunicação Humano-Computador, pelo fato de poder apontar para os objetos tridimensionais diretamente. Já a manipulação indireta, o usuário interage utilizando interface bidimensional na qual os valores são passados para a manipulação dos objetos tridimensionais. A Figura 2 ilustra a utilização da manipulação indireta, utilizando-se *scroll bars* para a passagem de valores ao Humano Virtual (HV) (CUSIN, 2005).

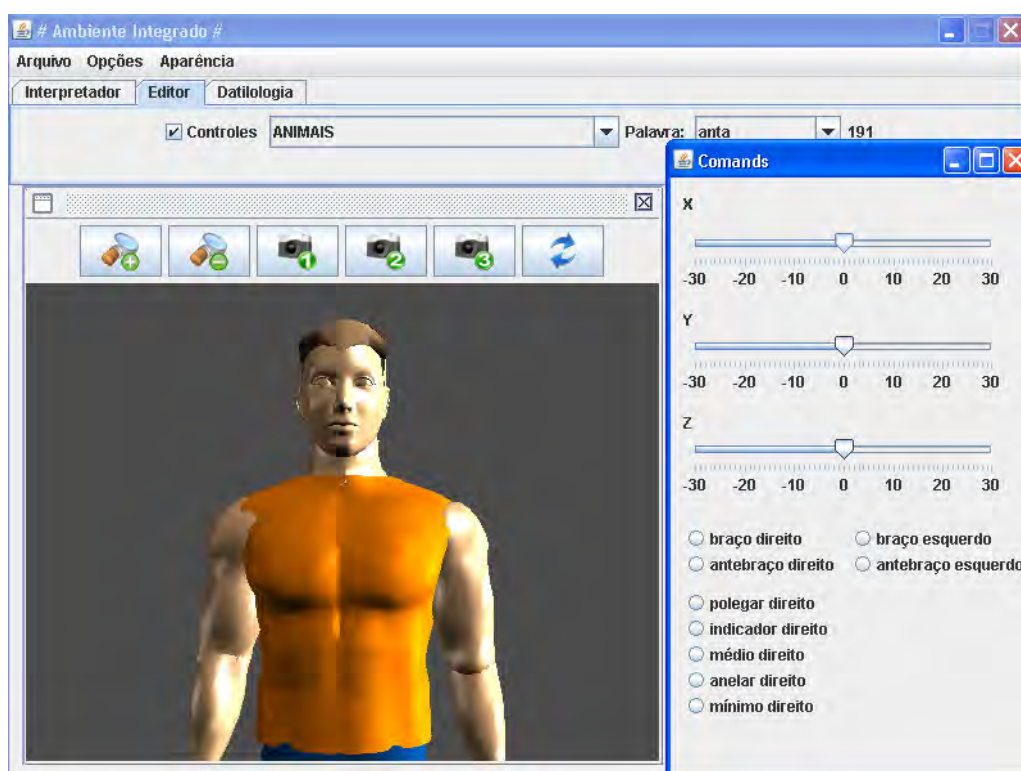


Figura 2 – Exemplo de manipulação indireta com o uso de *scroll bars* (BAPTISTA, 2007)

A interação é vista como um diálogo entre o usuário e o computador. Para sua efetivação existem diversos estilos de interfaces, como por exemplo, a Interface de Linha de Comando e Menus, Linguagem Natural (LN), Apontar-e-Clicar e Interfaces Tridimensionais.

Tal escolha tem consequência intensa na natureza da comunicação entre as partes envolvidas (CUSIN, 2005).

2.7 – Interação Natural

Com a evolução da interação humano-computador nas últimas décadas há uma tendência nesta área na qual vem sendo chamada de interação natural (VALLI; 2005). As maneiras de interação vêm evoluindo no sentido de tornar a relação humano-computador mais eficiente e confortável.

Dentro deste fluxo de mudança, tecnologias capazes de rastrear o corpo do usuário (parte dele ou completo) ganham visibilidade. Dentre várias técnicas existentes e possíveis, há técnicas de rastreamento que utilizam apenas tecnologias acessíveis e genéricas, fazendo uso de dispositivos genéricos capazes de realizarem o rastreamento corporal, como por exemplo, câmeras comuns consideradas populares na atualidade. Tal técnica pode ser chamada de rastreamento corporal monocular (TEICHRIEB; FIGUEIREDO, 2010).

Diante aos progressos realizados nesta área, o rastreamento de objetos passa também a ser aplicado ao rastreamento corporal, considerando o corpo humano como um modelo 3D a ser rastreado. Contudo, ainda existe a possibilidade de utilizar técnicas de rastreamento corporal com modelos 2D, possuindo métodos de rastreamento mais simples para alcançar resultados mais velozes e precisos, porém perdendo as informações de profundidade (TEICHRIEB; FIGUEIREDO, 2010).

Perante este contexto, surgem ferramentas que possibilitam a implementação de técnicas de rastreamento corporal monocular. Na sua maioria estas técnicas se agrupam em três maneiras de rastreamento: rastreamento do corpo de modo simplificado, da face e/ou cabeça e das mãos. No entanto, grande parte destas técnicas dificilmente conseguem combinar satisfatoriamente robustez, precisão e execução em tempo real (TEICHRIEB; FIGUEIREDO, 2010).

2.8 – Tecnologias de Rastreamento

Os rastreadores são dispositivos que permitem à aplicação rastrear a posição e/ou orientação tridimensional de objetos ou partes do corpo do usuário, assim cada sistema de rastreamento tem um dispositivo que gera um sinal, um sensor que detecta o sinal e uma unidade de controle que processa o sinal e envia as informações para a CPU. Alguns sistemas solicitam que você conecte o componente do sensor ao usuário (ou ao equipamento do

usuário). Nesse tipo de sistema, você posiciona os emissores de sinal em pontos fixos no ambiente. Alguns sistemas são configurados de forma contrária: o usuário veste os emissores enquanto é rodeado por sensores anexados ao ambiente (STRICKLAND, 2007).

Os sinais enviados dos emissores para os sensores podem assumir várias formas, incluindo sinais mecânicos, ultrassônicos, magnéticos, por extração de imagem, óticos e sem referencial. Cada tecnologia tem seu próprio conjunto de vantagens e desvantagens (BURDEA, 2005).

O rastreamento mecânico é utilizado quando se prioriza a alta velocidade e a precisão do mesmo, contendo uma alta taxa de atualização. Tal mecanismo contém uma conexão física entre o usuário e a aplicação, limitando-o na execução de movimentos e oferecendo pouca mobilidade.

O rastreamento ultrassônico ou acústico determina uma posição enviando ondas de som ultrassônicas até um conjunto de receptores que os capta. Desta forma o controlador comanda a emissão do som e o percebe quando captado, assim, pode-se calcular a posição atual através do tempo decorrido desde a emissão até a recepção do mesmo.

O rastreamento magnético mede os campos magnéticos gerados passando uma corrente elétrica sequencialmente através de três bobinas de fios dispostas em uma orientação perpendicular uma à outra. Cada pequena bobina se torna um ímã, e os sensores do sistema medem como seu campo magnético afeta as outras bobinas. Essa medição informa ao sistema a direção e a orientação do emissor. Um bom sistema de rastreamento eletromagnético é bastante sensível, com baixos níveis de latência. Uma desvantagem desse sistema é que qualquer coisa que gere um campo magnético pode interferir nos sinais enviados aos sensores.

O rastreamento por extração de imagem calcula a posição do usuário em função da posição das luzes (*led's*) colocadas nas partes do corpo que são filmadas por uma câmera e posteriormente processadas.

O rastreamento ótico é uma inversão dos rastreadores por extração de imagem, ou seja, em vez da câmera capta os *led's* sobre o corpo, a câmera esta sobre o corpo e capta os *led's* do ambiente. Assim pode-se calcular a posição e orientação do usuário de acordo com as informações captadas pela câmera dos *led's* que acendem em pulsos sequenciais. Tais informações são enviadas pela câmera a uma unidade de processamento na qual interpreta os dados e calcula a atual posição e orientação. A desvantagem do sistema é que a linha do sinal entre a câmera e um *led* pode ser obscurecida, interferindo no processo de rastreamento. A luz do ambiente ou radiação infravermelha também pode tornar o sistema menos efetivo. Este

tipo de rastreamento esta diretamente relacionado à qualidade das imagens capturadas pela câmera e de algoritmo de reconhecimento de padrões utilizados sobre os mesmos.

Por fim tem-se o rastreamento sem referencial na qual mede inclinações e giros a partir de uma posição inicial, proporcionando “liberdade” ao usuário por não ter um conjunto de fios ligados a ele, por não ficar restrito a uma área onde sensores captam os movimentos e por não ficar “amarrado” a um computador. Tal rastreamento é dividido em três categorias: Inclinômetros, Chave de Inclinação e os Sensores Piezoelétricos de pressão e torção.

Os Inclinômetros são pêndulos que calculam a inclinação de um objeto a partir de uma posição anterior onde o pêndulo estava parado. A Chave de Inclinação calcula o mesmo, porém de maneira digital. E os Sensores Piezoelétricos geram uma corrente elétrica ou mudam sua resistência quando sofrem mudanças em suas formas, assim podem ser utilizados para calcular a flexão dos dedos das mãos ou de juntas como cotovelos e joelhos, por exemplo (BURDEA, 2005).

2.9 – Dispositivos de Rastreamento

Os dispositivos devem permitir que as ações do usuário possam ser transmitidas em tempo real ao sistema de RV/RA. Assim os dispositivos de rastreamento (ou *tracking*) detectam e monitoram os movimentos do usuário tornando a interação com os ambientes RV/RA mais intuitivo. Tal tarefa conecta dispositivos nas partes do corpo humano coletando informações dos movimentos realizados e os envia para o sistema, sempre tendo um ponto ou estado como referência para calcula à orientação ou deslocamento (MACHADO, 2010).

Nas técnicas de rastreamento passivo, são utilizados câmeras ou sensores óticos para capturar os movimentos. Já nas técnicas de rastreamento ativo, os sensores são posicionados diretamente sobre o objeto de rastreamento, como luvas dotadas de um ou mais sensores, por exemplo.

Após as informações serem capturadas e processadas pelo sistema, deve existir um *feedback* para o usuário. Este pode ser visto através de dispositivos de saída que irão fornecer as sensações físicas relacionadas ao ambiente de RV/RA. Com as tecnologias atuais já é possível oferecer estímulos aos sentidos da visão, audição e físico.

Os dispositivos visuais exploram a visão, muito importante na obtenção e na determinação do nível de imersão no sistema. Nesta área de visualização de informação podem se destacar os monitores, projetores, óculos e filtros que garantem a profundidade (visão estereoscópica), óculos com filtros coloridos para separar as imagens em vermelho e

azul, vermelho e verde e vermelho e ciano (estereoscopia por anaglifo), óculos polarizadores que utilizam filtros para polarizar a luz em duas direções ortogonais, dentre outros. Todos estes métodos que utilizam filtros fazem com que cada olho do usuário perceba apenas uma das imagens de um par estereoscópico produzido para cada cena do AV (NETTO et.al., 2002).

Os dispositivos auditivos têm como característica básica simular o modelo humano de audição. Em sistemas de RV/RA este tipo de som é conhecido como som 3D, pois deve variar de acordo com a movimentação do usuário (MACHADO, 2010).

Por fim tem os dispositivos físicos, na qual estimulam as sensações relacionadas ao tato, tensão muscular e temperatura. Estes permitem ao usuário identificar texturas, elasticidade, forma, temperatura e resistência em ambientes de RV/RA. O objetivo destes equipamentos é aumentar o nível de imersão do usuário ou permitir que o mesmo interaja com o AV. Há também dispositivos de saída que visam em alterar ou gerar perturbações ao equilíbrio corporal do usuário, como em simuladores de videogames e voo, por exemplo, (MACHADO, 2010).

2.10 – Considerações Finais

Este capítulo apresentou a evolução das formas de comunicação, desde a sua utilização em tempos passados até os meios de comunicação atuais, através de interfaces gráficas bem definidas e intuitivas. Assim, há o enriquecimento na literatura para a descoberta de novas soluções de determinados problemas, estes podendo ser considerados complicados sem a utilização de novas tecnologias, possibilitando alternativas flexíveis para a solução de problemas relacionados à comunicação.

O próximo capítulo apresenta a Língua de Sinais, focando-se principalmente em LIBRAS, pois esta é a utilizada no desenvolvimento do Sistema Avançado de Realidade Virtual proposto.

Capítulo 3 – Língua de Sinais

A comunicação é uma necessidade humana, sendo a linguagem oral e a escrita às maneiras mais comuns de comunicação. Segundo Góes (1996), a língua utilizada pelo indivíduo depende do grupo em que está inserido, sendo ouvinte, a comunicação se estabelece em termos oral-auditivos. No entanto, para deficientes auditivos a comunicação se estabelece em termos gestual-visuais, em que gestual significa o conjunto de elementos linguísticos manuais, corporais e faciais necessários para a articulação de movimentos.

Conceituando formalmente a Língua de Sinais, pode-se dizer que este tipo de linguagem se refere a estruturas linguísticas utilizadas pela comunidade de deficientes oral-auditivos, na expressão, na elaboração de pensamentos e na comunicação, sendo formadas pela associação de movimentos das partes do corpo acima do quadril e produzidas através do canal gestual-visual (PONTES; ORTH, 2009).

A Língua de Sinais ou a Língua Gestual refere-se basicamente ao uso de gestos e sinais em vez de sons na comunicação.

Segundo Karnopp (1994) enquanto o emissor constrói uma sentença a partir de vários elementos (mão, ombro, expressão facial, etc.) o receptor utiliza a visão para entender o que foi transmitido. Desta forma, os sinais construídos baseavam-se nos limites articulares do corpo e nos limites do sistema visual humano.

A Língua de Sinais não é uma língua universal e, da mesma forma que a língua oral, é diferente em vários países devido à influência cultural, podendo até mesmo apresentar sinais que variam entre regiões e entre comunidades de deficientes auditivos. Assim como cada país tem sua língua oficial, cada país também possui sua Língua de Sinais (MARCATO, 2000).

No Brasil, a Língua de Sinais LIBRAS foi reconhecida em 24 de abril de 2002 como meio legal de comunicação e expressão das comunidades surdas, de acordo com a lei Nº. 10.436 (Anexo A), decretada pelo Congresso Nacional e sancionada pelo então presidente da república, Fernando Henrique Cardoso (FUSCO, 2004).

As sessões seguintes apresentam detalhadamente LIBRAS, sendo a Língua de Sinais oficial do Brasil e a utilizada no processo de desenvolvimento do trabalho.

3.1 - LIBRAS

Santarosa (2000) afirma que “língua” indica um específico sistema de signos que é utilizado por uma comunidade para comunicação. Portanto, LIBRAS é uma língua natural surgida entre os deficientes auditivos brasileiros com o intuito de acatar as necessidades comunicativas de sua comunidade.

Brito (1995) afirma que são línguas naturais como as línguas orais, pois efervesceram espontaneamente da interação entre os deficientes auditivos, além de, por meio de sua estrutura, poder expressar qualquer conceito desde o descritivo/concreto ao emocional/abstrato.

LIBRAS é dotada de toda a complexidade e utilidade encontradas nas línguas orais e, assim como elas, possuem gramáticas próprias, com regras específicas em seus níveis linguísticos, fonológico (configurações e movimentos dos elementos que estão envolvidos nos sinais), morfológico (formação de sinais), sintático (sequência dos sinais) e semântico (significado da sequência de sinais).

3.2 - Fonologia da Língua de Sinais

A fonologia analisa os fonemas de uma língua. Entende-se por fonemas unidades mínimas que estabelecem distinção entre os vocábulos de uma língua. Os objetivos desta área são a determinação das unidades que formam os sinais e dos possíveis padrões de combinações entre essas unidades tanto quanto as variações permitidas e possíveis.

Na Língua de Sinais são utilizadas as partes superiores do quadril para a geração dos sinais, estes sendo produzidos por meio de expressões manuais (movimento das mãos, braços e antebraços) e não-manuais (movimento da face, dos olhos, da cabeça e do tronco).

Brito (1995) estudou alguns aspectos muito importantes para a representação computacional dos sinais, referentes à constituição de LIBRAS. Esta apresentou três parâmetros primários que se combinam: a configuração das mãos, o movimento e o ponto de articulação, além de apresentar parâmetros secundários: a disposição das mãos, a orientação das mãos, a região de contato e seus componentes não manuais como a expressão facial.

A configuração da mão consiste de como as mãos e os dedos estão posicionados, sendo possíveis 46 configurações diferentes. A Figura 3 ilustra algumas dessas possíveis configurações (SCHNEIDER, 2008).



Figura 3 – Configuração de mãos (SCHNEIDER, 2008)

O ponto de articulação é o local dentro do raio do alcance das mãos aonde são feitos os sinais, podendo estes tocar em alguma parte do corpo ou estar em um espaço neutro. (QUADROS; KARNOPP; 2004). Por exemplo, os sinais TRABALHAR, BRINCAR e CONSERTAR são feitos no espaço neutro e os sinais ESQUECER, APRENDER e PENSAR são feitos na testa. Existem palavras que se diferem entre si apenas pela diferença do ponto de articulação, como laranja (Figura 4 (a)) e aprender (Figura 4 (b)), por exemplo.

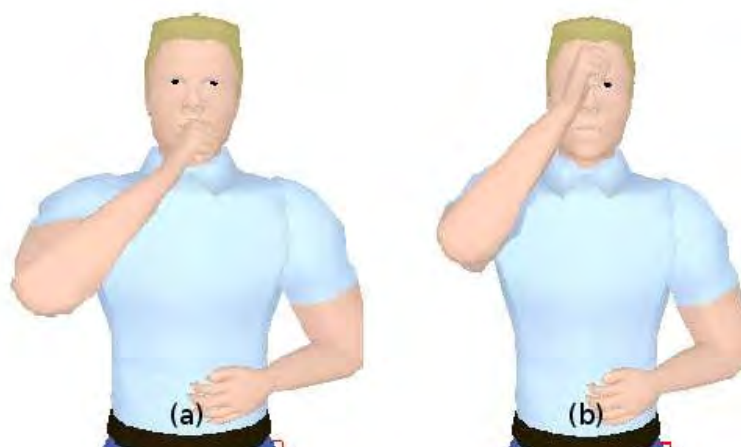


Figura 4 - Configuração final do sinal laranja (a) e do sinal aprender (b) (SCHNEIDER, 2008)

O movimento é a variação da posição espacial de um objeto (sinal) no decorrer de um tempo, isto é, é a execução de um sinal. Segundo Brito (1974), o movimento em LIBRAS se localiza nas mãos, pulsos e antebraço, podendo haver repetições no mesmo e características

como velocidade e tensão. Os sinais podem ter ou não movimentos. Por exemplo, no sinal TRABALHAR, tem-se o movimento para frente e para trás de ambos os braços (Figura 5 (a)), já no sinal EM-PÉ, não há movimento, apenas repousa-se uma mão na configuração adequada sobre a outra (Figura 5 (b)).

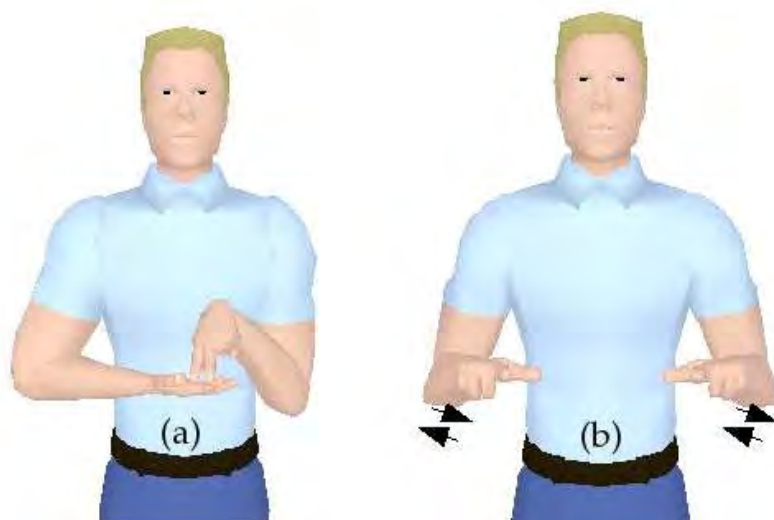


Figura 5 – Movimento nos Gestos TRABALHAR (a) e EM-PÉ (b) (SCHNEIDER, 2008)

A Orientação indica a direção para a qual a palma da mão aponta na produção do sinal. Brito (1974) enumera seis tipos de orientação da palma da mão: para cima, para baixo, para o corpo, para frente, para a esquerda e para a direita.

As Expressões não manuais (Facial e/ou Corporal) são de extrema importância na Língua de Sinais, pelo fato de dar entonação (negação, interrogação, por exemplo) para o entendimento dos sinais.

Na formação de sinais há também restrições para reduzir a complexidade dos sinais, para que os mesmos sejam facilmente executados e entendidos. Segundo Battison (2000) há duas restrições na produção do sinal envolvendo as duas mãos: a condição de simetria (configuração igual das mesmas) e a condição de dominância (uma mão produzindo o sinal e a outra servindo como apoio).

3.3 – Morfologia da Língua de Sinais

Morfologia é o estudo da estrutura dos sinais (léxico), de como os mesmos são formados. Um exemplo é a formação nominal, que é realizada por meio da alteração de movimento do sinal do verbo. Por exemplo, a representação do sinal TELEFONE (Figura 6

(a)) é igual à representação do sinal TELEFONAR (Figura 6 (b)), entretanto no substantivo tem-se o movimento representado por flechas, indicando “ir e voltar” do cotovelo e no verbo tem-se o movimento de uma curva direcional, indicando que o movimento será curvilíneo e unidirecional (SCHNEIDER, 2008).

Segundo Cusin (2004), o sistema morfológico diz respeito à formação das palavras, considerando características relativas a gênero (masculino/feminino), grau (intensidade, movimento, velocidade), tempo (passado, presente, futuro) e negação.

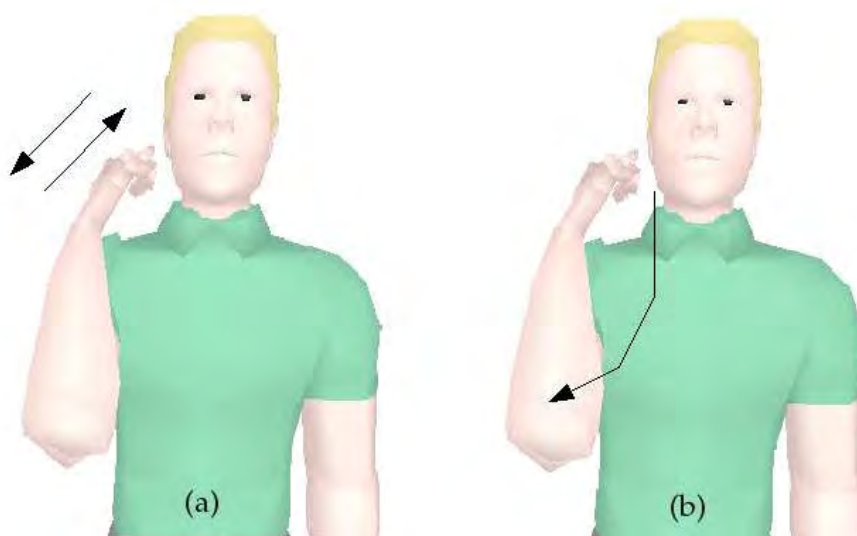


Figura 6 - Formação do sinal do substantivo TELEFONE (a) e do verbo TELEFONAR (b) (SCHNEIDER, 2008)

A estrutura dos sinais de LIBRAS é bastante complexa por apresentar muitos aspectos relevantes, tais como a forma de como é adicionado no sinal à negação e a flexão, por exemplo.

Em relação à negação, há três maneiras desta ser adicionada em LIBRAS: por meio do movimento de negação com a cabeça juntamente com o sinal a ser negado, por meio do movimento de negação do dedo indicador seguido do sinal a ser negado ou por meio de expressão facial. Já a flexão é feita por meio de (SCHNEIDER, 2008):

- Pessoa: mudam as referências pessoais no verbo. Feito por meio de apontamento propriamente dito (apontar com o dedo). Os referentes são inseridos no espaço à frente do sinalizador, por meio do apontamento em diferentes locais (QUADROS; KARNOPP, 2004);

- Grau: proporciona diferenças para menor, maior, mais próximo, muito, entre outros. Alguns desses possuem sinal próprio, tal como grande e pequeno;
- Gênero: em LIBRAS há uma dificuldade na marcação de gênero. O sinal relativo a homem ou mulher é executado depois do sinal a ser flexionado. Há determinadas palavras que tem sinais diferentes quanto à flexão, como é o caso das palavras pai e mãe; e
- Foco temporal: adverte temporalidade. O tempo em LIBRAS é representado por aspectos espaciais no qual passado é apontado com um movimento para trás da mão e futuro com movimento de mão para frente. Palavras como ontem, amanhã e hoje tem sinais próprios em LIBRAS;

Na Língua de Sinais LIBRAS, ainda existe a possibilidade de fazer combinações de palavras para a geração de novas, porém diferentemente das línguas orais que geralmente adicionam prefixos ou sufixos nas raízes para novas combinações, esta utiliza movimentos e contornos no espaço de sinalização para enriquecer as palavras. (SCHNEIDER, 2008).

Em LIBRAS ainda existe a possibilidade da soletração manual, representando manualmente a ortografia portuguesa através das mãos, ou seja, o alfabeto manual (Datilologia). A seguir é ilustrada a Figura 7 que representa em Datilologia o alfabeto português e os números de 0 a 9 (BAPTISTA, 2007).

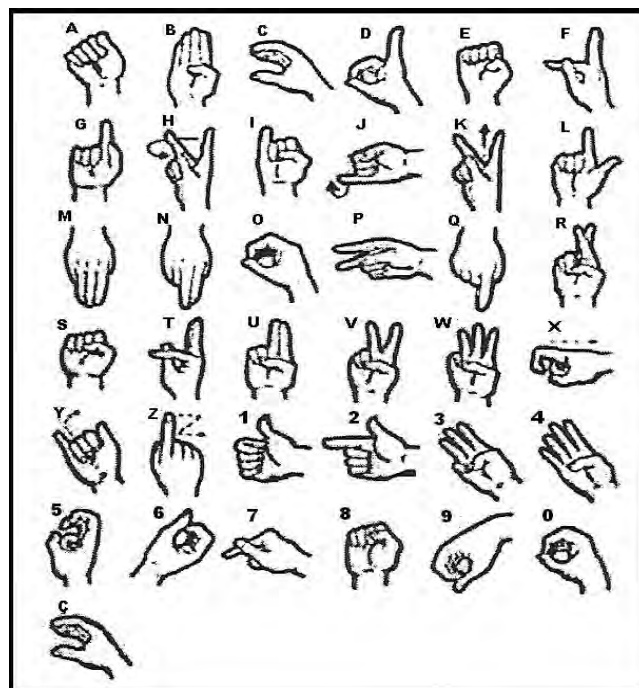


Figura 7 – Alfabeto Manual e Números em LIBRAS (NASCIMENTO, 2008)

3.4 – Sintaxe da Língua de Sinais

Segundo Quadros (1997) LIBRAS é uma língua organizada espacialmente de maneira altamente restringida. Avaliar determinados elementos da sintaxe de uma língua requer visualizar esse sistema de modo visual-espacial e não oral-auditivo.

A comunicação em qualquer língua é dotada de regras. Assim nasce a necessidade de se conhecer a estrutura de concepção das sentenças, com a finalidade de analisar o sistema sintático da Língua de Sinais. Em LIBRAS existe uma flexibilidade quanto à ordenação das palavras em uma sentença. Quanto às regras sintáticas, Quadros (1999) assume que há três possíveis maneiras de ordenar as palavras em LIBRAS: Sujeito-Verbo-Objeto(SVO), Objeto-Sujeito-Verbo(OSV) e Sujeito-Objeto-Verbo (SOV), as demais combinações são consideradas inadmissíveis.

Em relação às maneiras de ordenação de palavras, existem algumas características relevantes, tais como (BRITO, 1974):

- Todas as frases com a ordem SVO são gramaticais;
- Os advérbios temporais e de frequência não devem obstruir uma relação entre o verbo e o objeto. Por exemplo, não existe “REMO GANHAR ANTEONTEM UMA MOTO”, mas sim, “ANTEONTEM REMO GANHOU UMA MOTO”;
- As ordens OSV e SOV acontecem apenas quando há determinada coisa a mais na sentença, como marcas não-manuais; e
- Apesar de acontecerem construções SOV e OSV agregadas à marca não-manuais, se houver uma estrutura complexa na posição de objeto, não será possível mudá-la de ordem. Por exemplo, a ordem SOV não pode haver quando o objeto é uma oração subordinada, tal como “Eu quero que Remo trabalhe mais”.

3.5 – Convenções de LIBRAS

Como qualquer outro meio de comunicação, LIBRAS também possui suas convenções (NASCIMENTO, 2008), tais como:

- Grafia: os sinais em LIBRAS são representados na língua portuguesa em letras maiúsculas, como CARRO, MOTO, por exemplo;

- Datilologia: utilizada para expressar nomes de pessoas, lugares e outras palavras que não tem sinal, sendo representado por palavras separadas por hífen, exemplo: J-O-S-É; R-E-M-O;
- Verbos: sempre apresentados no infinitivo e suas concordâncias e conjugações sempre realizadas no espaço;
- Frases: obedecendo as estruturas de LIBRAS (fonológica, morfológica e sintática); e
- Pronomes pessoais: sempre representados pelo sistema de apontação. O emissor utiliza o indicador apontado para si para se referir a ele mesmo (Figura 8 (a)) e o indicador apontado para o receptor para referenciá-lo (Figura 8 (b)).

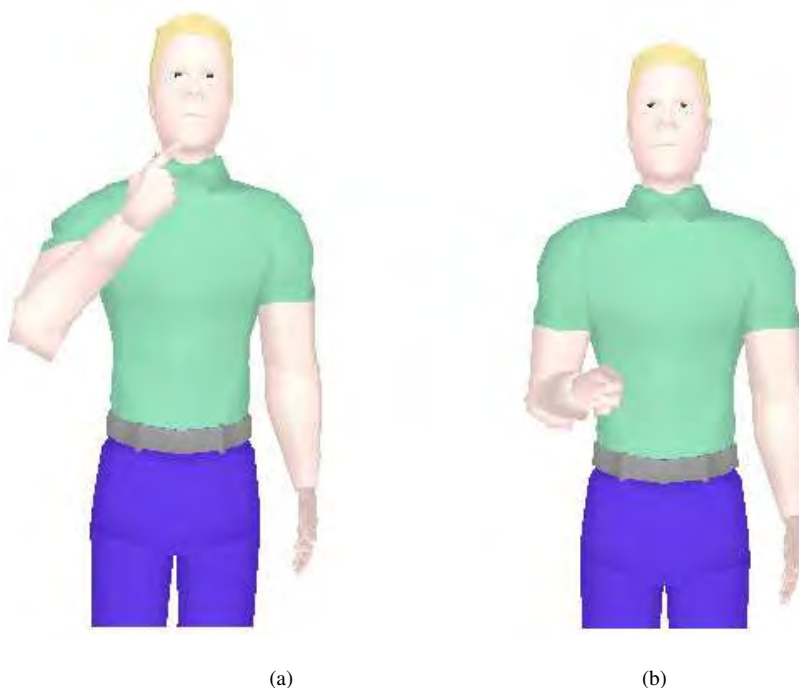


Figura 8 – Determinação de pontos com o referente presente (SCHNEIDER, 2008)

3.6 – Tecnologia e Deficientes Auditivos

Com a crescente evolução das tecnologias, surgiram métodos e ferramentas computacionais para auxiliarem e potencializarem os seres humanos em suas tarefas cotidianas. Entretanto tais ferramentas seriam de extrema importância se utilizadas para colaborarem com a inclusão social de deficientes auditivos, que por meio destas, abririam um

novo horizonte para o canal de comunicação entre eles e com indivíduos ouvintes (BAPTISTA, 2007).

Segundo Rocha (2000), no Brasil, é raro ambientes computacionais que trabalhem com a Língua de Sinais, porém este fato vem mudando cada vez mais. Além disso, Dizeu e Caporali (2005) ressaltam que deve haver a conscientização de que toda esta tecnologia disponível não é de acesso a toda a população com deficiência auditiva, especialmente considerando a realidade social, cultural e econômica do país.

Pellegrino (1996) expõe que a tecnologia via computador pode proporcionar:

- Incremento da competência de comunicação do deficiente auditivo, estimulando a atenção necessária à expressão orofacial do interlocutor;
- Gera a integração da Língua de Sinais com a língua verbal;
- Promove o acesso de deficientes auditivos e familiares a um material que possa oferecer andamento do treinamento no lar.
- Melhora o atendimento aos deficientes auditivos; e
- Valoriza e atualiza a pedagogia especial, comprometida com o processo de transformação social, de acordo com a necessidade real da educação dos deficientes auditivos;

No Brasil foi desenvolvido um projeto pela equipe FENEIS – Federação Nacional para Educação e Integração de Surdos, chamado Dicionário de LIBRAS – *on-line*. O objetivo do trabalho era divulgar em larga escala a língua brasileira de sinais, tornando o material didático recreativo e disponível *on-line*, e desta maneira, potencializar instrutores e agentes para atingir todos os pontos do Brasil por meio da Internet (FUSCO, 2004).

Na primeira fase do trabalho, foi dada ênfase ao desenvolvimento de material educativo para crianças na faixa escolar, tendo LIBRAS sempre presente e com temas relacionados ao currículo do ensino fundamental, de acordo com as normas do ministério de educação. O Dicionário disponibiliza um pequeno vocabulário, que aos poucos foi sendo ampliado, dividido em quinze categorias: alimentos, animais, repartições públicas e afins, cores, datas, diversos, lar, natureza, números, países, estados e cidades; pessoas e parentesco; profissões, roupas, transportes e verbos. A Figura 9 ilustra a interface inicial do projeto (FUSCO, 2004).



Figura 9 – Dicionário de LIBRAS *On-Line* (FUSCO, 2004)

Souza e Pinto (2003) complementam que em uma sociedade informatizada, todos precisam de um computador presente no processo de aprendizagem, até mesmo os deficientes auditivos. Para a educação dos indivíduos que se comunicam de modo visual, a utilização de ferramentas computacionais passa a ser indispensável.

3.7 - Ferramentas de Língua de Sinais

Quando se pensa na elaboração de softwares para pessoas com necessidades especiais, devem-se levar em consideração tais necessidades, o que faz com que ferramentas de auxílio a essas pessoas sejam escassas, visto que os softwares mais comuns são feitos com foco em pessoas que não possuem tais necessidades. Atualmente já existem alguns softwares para os usuários deficientes auditivos. Dentre eles alguns já usam animação de HVs.

O *Hand Sign Translator* (HST) (HOLDEN; ROY, 1992b) (HOLDEN; ROY, 1992a) é um tradutor da língua inglesa oral para a Língua de Sinais chamada *signed english*, criada por educadores australianos. É uma representação manual do inglês através do uso de sinais e soletramento de palavras por meio dos dedos. Os sinais usados em *signed english* sempre representam uma letra ou uma palavra e a tradução do inglês é feita por mapeamento direto, pois a sintática e a semântica não se alteram de uma para outra língua. Ele foi criado para ser uma ferramenta educacional e atualmente funciona da seguinte forma: o usuário fornece à aplicação uma entrada textual em inglês e recebe como saída uma animação das mãos gerando a entrada em *signed english*. Por utilizar apenas as mãos é uma língua de mais fácil assimilação, porém não auxilia no aprendizado de qualquer outra língua oficial.

Vsigns (PAPADOGIORGAKI et. al., 2004) é um gerador de animações a partir da notação de Língua de Sinais chamada *SignWriting* (SUTTON, 1995), muito utilizada pela comunidade surda. Este gera uma sequência de animações em *Virtual Reality Modeling Language* (VRML). Um ponto positivo é a utilização do *SignWriting*, visto que muitos deficientes auditivos não são alfabetizados por encontrarem dificuldades em assimilar a escrita, porém a animação não é natural e também não trata de questões importantes, tais como o espaço de enunciação.

O *Virtual Signing - ViSiCast* (BANGHAM et. al., 2000) é um projeto que busca desenvolver meios de acesso às informações e serviços, em Língua de Sinais, para deficientes auditivos. Este consiste no desenvolvimento de sistemas computacionais que usam HVs para a animação de gestos da *British Sign Language* (BSL), tendo como base dois trabalhos, o *Sign-Anim* (MARCHAL; et. al.; 2001) e *Tessa* (LINCOLN et. al., 1998), sendo sistemas tradutores e animadores de texto para Língua de Sinais. Ambos utilizam captura de imagens através do uso de sensores individuais para as mãos, corpo e face. Entretanto essa captura possui alguns problemas, tais como a dificuldade de se calibrar os instrumentos e a gravação de uma grande quantidade de sinais. Um ponto positivo no *ViSiCast* é a naturalidade dos gestos, contudo o mesmo não trata de detalhes importantes, tais como o espaço de enunciação e as diferentes formas de se executar um gesto (mais tenso, indicando raiva, por exemplo).

O X-LIBRAS (FUSCO, 2004) é um ambiente virtual estruturado de informações utilizando a tecnologia XML, permitindo a criação de uma arquitetura padronizada de armazenamento e intercâmbio de informações de sinais de LIBRAS. O vocabulário possui um mecanismo unificado, padronizado e flexível, permitindo a facilidade do desenvolvimento de ferramentas computacionais utilizando a Língua de Sinais LIBRAS. A partir deste vocabulário, implementar um sistema de RV que permita a visualização dos sinais em LIBRAS num ambiente tridimensional se torna trivial.

A tabela 1 apresenta comparações entre sistemas de Língua de Sinais, apresentando característica e atributos comuns e diferentes entre o sistema proposto e outros já disponíveis.

Tabela 1- Comparações de Sistema de Língua de Sinais

Nome do Sistema	Características e Atributos					
	Interpretador	Editor	Língua de Sinal	Entrada	Saída	Interação
<i>Hand Sign Translator</i>	SIM	NÃO	ASL	Textual	Animação das mãos	Clique do Mouse
<i>ViSiCast</i>	SIM	SIM	BSL	Textual	Animação do HV	Clique do mouse e Dispositivo não convencional
X-LIBRAS	SIM	NÃO	LIBRAS	Textual	Animação do HV	Clique do Mouse
Protótipo Proposto	SIM	SIM	TODAS	Modelo Virtual	Animação do HV em Tamanho Real junto a Descrições	Dispositivo não convencional, Estereoscopia e Transformações

3.8 – Considerações Finais

Este capítulo apresentou Línguas de Sinais, principalmente focando-se em LIBRAS. Foram apresentados os seus níveis linguísticos, suas convenções e tecnologias acessíveis à comunidade de deficientes auditivos. Por fim, foram apresentados quatro trabalhos relacionados à Língua de Sinais, destacando suas funcionalidades, vantagens e desvantagens, além de apresentar uma tabela comparativa entre as ferramentas e a proposta do trabalho.

O próximo Capítulo apresenta HV, destacando seus conceitos, aspectos de aparência e suas possíveis articulações com seus respectivos graus de liberdade (DOFs).

Capítulo 4 – Humanos Virtuais

Há décadas, foi proposta por pesquisadores a ideia de que todas as tarefas humanas poderiam ser realizadas por máquinas, desde envolvimento em jogos até robôs autômatos. Entretanto, para a situação tecnológica da época essa hipótese era considerada impossível, sendo mais uma ficção científica do que a própria realidade.

Com o passar do tempo e avanço tecnológico, a ideia de HVs tornou-se viável, sendo possível a representação de máquinas em uma maneira mais racional. Desta maneira era possível substituir seres humanos na realização de atividades em um mundo físico de tarefas onde havia riscos, evitando problemas de presença, segurança e limitações físicas (MODESTO, 2006).

As sessões seguintes apresentam conceitos sobre HVs, seus aspectos físicos e suas possíveis articulações com seus respectivos DOFs, atribuindo ao HV a flexibilidade do movimento em determinadas juntas.

4.1 – Humanos Virtuais

Segundo Badler (1997), HVs são modelos que são usados como substitutos de “humanos reais” em testes ergonômicos baseados em computadores para projetos de veículos, trabalho de campo, ferramentas, linhas de produção, para simulação previa antes da construção, ensino de tutoriais e para representação humana em AVs de tempo real. Se necessário, HVs podem apresentar uma aparência bastante realista, baseada em princípios fisiológicos e biomecânicos. A Figura 10 ilustra a ideia de HVs na participação em AVs.



Figura 10 – Participação de Humanos Virtuais em um Ambiente Virtual (PLETSCH, 2006)

Segundo Vilela (2002), HVs são entidades interativas de *software* com grau variável de autonomia, inseridos em AVs. Suas características gráficas e seus movimentos simulam basicamente o comportamento humano. Além do mais, em vez de ficarem restritos a movimentos previamente definidos, HVs são capazes de comportar-se de forma autônoma e inteligente em ambientes dinâmicos, podendo apresentar até mesmo individualidade e personalidade.

Os HVs não estão relacionados somente a ideia de movimentos e expressões (CHAI; HODGINS; 2003), mas também, a ideia de representar emoção, reação, interação e tomada de decisão bem próxima a “natural” (BATES; LOYALL, 1992).

Na utilização de HVs deve levar em consideração os diversos domínios de utilização e os relacionamentos deste com os demais seres humanos pertencentes ao mundo real. Dentre estes domínios estão: Análise de Fatores Humanos (altura, comportamento), Diferenças entre Humanos (características próprias), Geração e Compreensão de Instruções (comunicação de sensações e experiências transportada aos HVs), Simulação Bio-Médica (estrutura físicas e funcionais) e Análise de Forma e Movimento (entendimento do que se vê e sente de maneira geométrica) (BATES; LOYALL, 1992).

Em relação à modelagem, o projeto de HVs deve levar em consideração alguns critérios importantes, tais como: Aparência (caricatura até modelo físico), Funcionalidade (ações), Tempo, Autonomia (ações inteligentes), Individualidade (personalidade), Atributos Físicos e Biomecânicos (limitações funcionais), Movimentação (captura de movimentos) e Inteligência (tomada de decisão) (BADLER, 1997).

Deste modo, pode-se perceber que HVs são bem coerentes a “humanos reais”, sendo utilizados para simularem e avaliarem as relações entre os seres humanos em AVs, onde são de amplo valor na concepção de documentação e pesquisas que permitem avaliar o impacto da presença humana nestes domínios.

4.2 – Avatares e Agentes Virtuais

Os HVs podem ser classificados em duas maneiras diferentes: Avatar e Agente Virtual. Segundo Badler (1997), agente virtual é um HV desenvolvido e controlado por software e avatar um HV controlado por um usuário do mundo real.

Segundo Thalmann et.al (1999), avatar é uma representação gráfica de um indivíduo real no AV, junto ao seu comportamento. Agente virtual é uma aplicação que tem

comportamento autônomo, não necessariamente precisa ser representado por uma entidade gráfica.

Segundo Torres et. al. (2002), agentes virtuais têm sido utilizado em vários projetos na implementação de personagens virtuais, transmitindo a ideia da ilusão de vida. Tais agentes possuem diversas arquiteturas para sua implementação, tais como os agentes reativos, que simplesmente ficam implícitos as suas regras de comportamento, e os agentes cognitivos, que raciocinam para decidir os objetivos a alcançar, os planos a seguir e as ações a executar, por exemplo.

4.3 – Aspectos dos Humanos Virtuais

Os HVs possuem cinco aspectos fundamentais:

- Aparência;
- Movimento;
- Interatividade;
- Comunicação; e
- Gestos descritos.

As subseções seguintes apresentam detalhadamente os conceitos de cada aspecto essencial para a renderização de um HV em um AV, ficando restrito apenas ao tipo de aplicação que este HV estará designado.

4.3.1 – Aparência

A aparência de um avatar pode vir desde uma caricatura até modelos físicos bem definidos. Para o processo de definição dos modelos de HVs, deve ser analisada a escala do corpo, o cálculo da segmentação, os centros das juntas e a aspiração de se conseguir uma aparência realista.

Os HVs podem ser representados por um conjunto de segmentos geométricos conectando as juntas. Contudo nem todas as juntas do corpo humano podem ser representadas por uma junta no modelo. Aquelas que podem ser representadas nos modelos são juntas sinoviais, que possuem de um a três ângulos de liberdade (*DoF*) e limites especificados. Também podem ser concebidos utilizando modelos poliedrais, que são compostos por uma

geometria simples totalizando por volta de dois mil poliedros de fácil manipulação e escalabilidade (AZUOLA et. al., 1994). A Figura 11 ilustra uma modelagem de um HV.

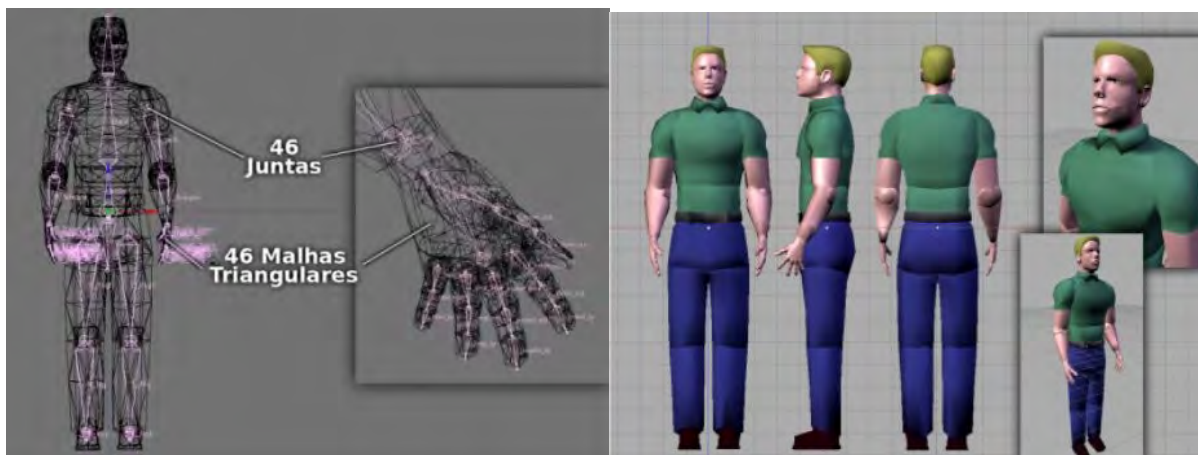


Figura 11 – Modelagem de Humano Virtual (NEDEL; SCHNEIDER, 2006)

4.3.2 – Movimento

O movimento é um aspecto importante para a representação de HVs, visto que este deve ser tão real e coerente quanto possível. Para isso, na representação dos movimentos, há a necessidade de um grande número de juntas e detalhamento na modelagem das mesmas para ficar de acordo com as exigências da aplicação (THALMANN et. al., 1999).

Existem várias técnicas para a geração de movimentos de um agente. Segundo Badler (1997), os movimentos sintetizados em relação aos outros tipos, permitem a geração de movimentos para mais de uma junta simultaneamente. As transformações pré-armazenadas (mudança dos ângulos das juntas) em uma base de dados ou biblioteca podem ser recuperadas e reutilizadas em tempo real para determinado movimento, alterando a orientação e velocidade do modelo (GRANIERI et. al., 1995). Desta forma ganha-se em velocidade, armazenamento e segurança algorítmica (redução da computação).

4.3.3 – Interatividade

A interatividade é utilizada para manipular o HV, sendo ele avatar ou um agente virtual. A interação deve especificar movimentos de maneira simples por meio de rastreamento de sensores, rastreamento de resultado final (por exemplo, cinemática inversa), invocação externa (como menus), sintetizadores de voz ou botões de ação (que sintetizam ou interpretam dados pré-armazenados). Sendo admissível também a realização do movimento

por meio de dispositivos de entrada de dados dirigíveis como mouses, teclados, luvas, entre outros (AZUOLA et. al., 1994).

Outra maneira de interação é a utilização da fala. Esta pode ser acionada criando-se um banco linguístico de ações argumentadas e elaboradas por meio de síntese de movimento de baixo nível. Esta técnica pode ser utilizada com vantagem em aplicações de AVs imersivos (MODESTO, 2006).

4.3.4 – Comunicação

Com o surgimento de jogos interativos e efeitos especiais realistas em filmes, o desenvolvimento de agentes que tenham essas características autônomas e em tempo real seria o primórdio da computação gráfica.

Na comunicação dos agentes são utilizadas técnicas de animação computadorizadas. Imagine que agentes tenham comportamentos autônomos e respondam de acordo com os estímulos humanos em tempo real, a aplicação teria que ser capaz de restringir os movimentos do usuário para que os comportamentos dos agentes sejam criados com antecedência, e assim, recuperados e modificados em tempo real. Entretanto com os jogos altamente imersivos e interativos, os usuários ficam infinitamente com inúmeras possibilidades de movimentos em um AV (MODESTO, 2006).

O HV se comunica com o usuário através da fala, de movimentos labiais, movimentos dos olhos, movimentos de mãos, postura corpórea, movimento de um membro e até sinais de mãos.

Os “Humanos Reais” apresentam certo tipo de veracidade em seus comportamentos. Em HVs essa veracidade se passa por meio da emoção e do estado mental da observação de seu comportamento, visto que cada comportamento utiliza meios para representar como está o estado interno do HV, de maneira que estes devem estar sem conflito e consistentes (MODESTO, 2006).

4.3.5 – Gestos

Os braços dos HVs têm a capacidade de buscar ou segurar um determinado objeto do AV, além de ser capaz de enfatizar e complementar a comunicação por meio de sinais e gestos (CASSELL et. al., 1994). A Figura 12 ilustra a comunicação de um HV através de gestos.

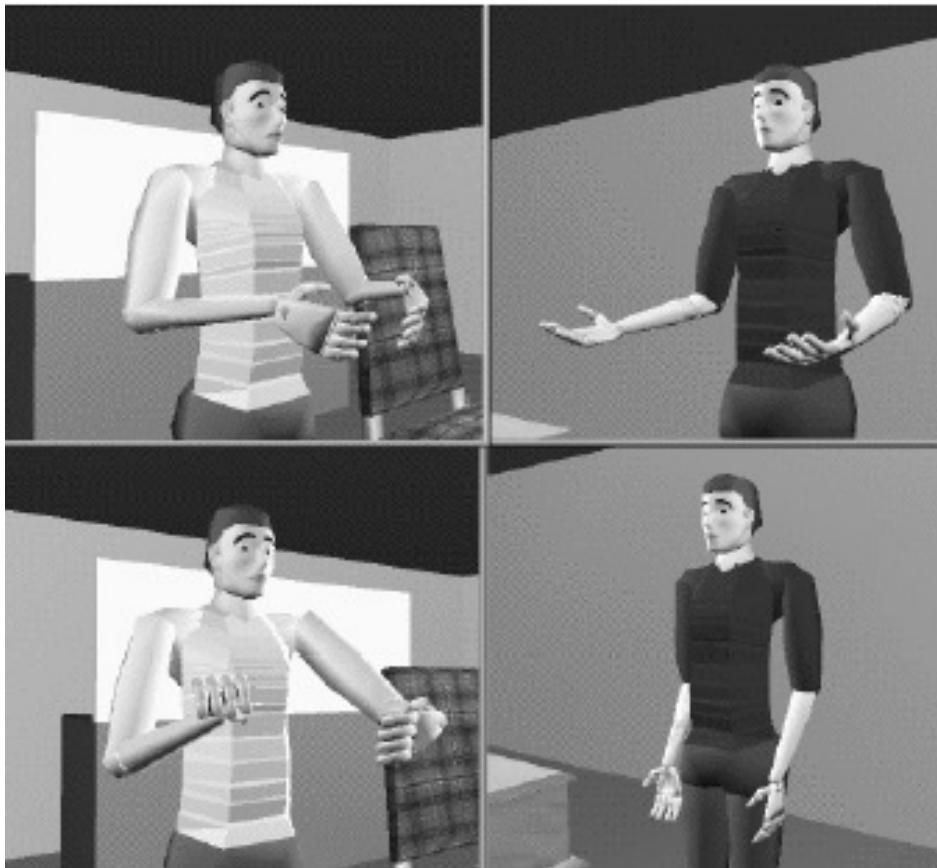


Figura 12 - Comunicação por Gestos (MODESTO, 2006 apud DOUVILLE et. al., 1996)

Os gestos além de ser de extrema importância na comunicação verbal por enfatizar e complementar a fala, na comunicação não-verbal se faz o mesmo, pelo fato de indicar ao usuário localizações de objetos e participantes do AV. Segundo Cassel et.al. (1994), gestos também podem ser utilizados para incrementar a comunicação entre interlocutores.

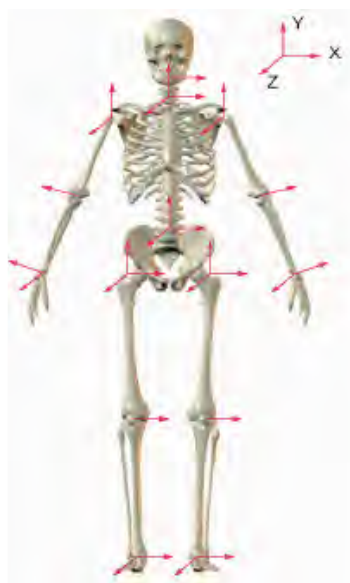
4.4 – Articulações em Humanos Virtuais

Articulação tem haver com a mobilidade de HVs, sendo que alguns trabalhos se enfocam mais na construção do esqueleto, outros se concentram na modelagem de músculos e outros ainda buscam modelar a pele e o cabelo.

Outra característica importante que deve ser levado em consideração para analisar o tipo de mobilidade que o HV precisa, esta relacionada à aplicação, pois esta pode ser em entretenimento de animações de jogos, simulação de expressões faciais, simulação de movimentos em atividades esportivas, dentre outras.

Nas animações de HVs as articulações são geralmente criadas por meio de esqueletos, sendo uma figura articulada com ligações rígidas entre as juntas. Para a geração de movimentos é necessário primeiro computar o movimento para a estrutura (esqueleto), e se satisfeito com a sequência, criar-se representações de alto nível de movimento cobrindo o esqueleto com superfícies deformáveis para a modelagem da pele (MODESTO, 2006).

A Figura 13 ilustra um esqueleto articulado com seus respectivos DOFs, determinando os graus de liberdade de cada articulação.



Articulação	DOFs
Cabeça	1
Pescoço	3
Ombro	2
Cotovelo	2
Punho	2
Cintura	3
Anca	3
Joelho	1
Tornozelo	2

Figura 13 - Graus de Liberdade de um Humanóide (WELTER JR; HOUNSELL, 2004)

4.5 – Considerações Finais

Este capítulo apresentou as características e utilidades de HVs, junto a conceitos e maneiras de implementação, dependendo apenas do cenário que o mesmo será inserido. Desta forma aspectos de modelagem e funcionalidade são implementados de acordo a necessidade do usuário para/com o AV.

O próximo capítulo apresenta as tecnologias utilizadas para o desenvolvimento do trabalho, apresentando as justificativas e realçando a importância e a utilidade de cada para a utilização no sistema.

Capítulo 5 – Tecnologias Envolvidas

Nos últimos anos tem-se vivenciado uma rápida evolução dos sistemas computacionais. Os computadores levaram a civilização contemporânea a uma terceira revolução, a chamada revolução da informação. Antes a civilização presenciou a revolução agrícola e a industrial. A revolução da informação intensificou um aumento expressivo na disseminação de informações. Este fato acrescentou a capacidade intelectual da humanidade com impacto direto nas ciências.

Com o surgimento de tecnologias dispostas a enriquecer e aprimorar o desenvolvimento de ferramentas computacionais para auxiliar em problemas, o processo de solução tem se tornado eficiente e viável. Desta forma, tecnologias gratuitas e flexíveis ganham espaço no desenvolvimento de ferramentas para fins dispersos.

Diante este contexto, a seguir são apresentadas as tecnologias estudadas ao longo do processo de desenvolvimento do trabalho, sendo divididas e brevemente expostas em sessões.

5.1 – Aglomerados Gráficos

Os aglomerados surgiram devido a avanços significativos na área de processamento de alto desempenho, substituindo os sistemas fortemente acoplados. São caracterizados por um conjunto de nós interconectados por intermédio de uma rede local, dando ao usuário a impressão de um único sistema. Esta arquitetura teve grandes avanços por oferecer especialmente baixos custos, escalabilidade e alta flexibilidade (BUYA, 1999).

O aglomerado gráfico é utilizado nas projeções de aplicações de RV e tem por objetivo oferecer uma visão múltipla do mesmo conjunto de dados, fazendo com que cada nó processe apenas os dados referentes à sua relevância, logo, gerando apenas a imagem correspondente a sua parte (GUIMARÃES, 2004).

Para que os aglomerados gráficos garantam a interatividade e a geração de imagens em tempo real das aplicações, requer-se que as tarefas sejam realizadas em tempo real, ou seja, obedecendo a um limite de tempo.

Por exemplo, uma aplicação de multiprojeção estereoscópica precisa garantir que a taxa de atualização dos quadros seja no mínimo 60 vezes por segundo, assim, não há atrasos na projeção.

A comunicação e a sincronização das imagens são algo de grande importância. A comunicação é necessária para a troca de informações entre os nós e a sincronização, fornece o controle de sequência e o controle de acesso, mantendo a ordem dos processos e a exclusão mútua de recursos, garantindo a consistência do sistema (GUIMARÃES, 2004).

A Figura 14 ilustra a utilização de aglomerados gráficos em uma Cave.

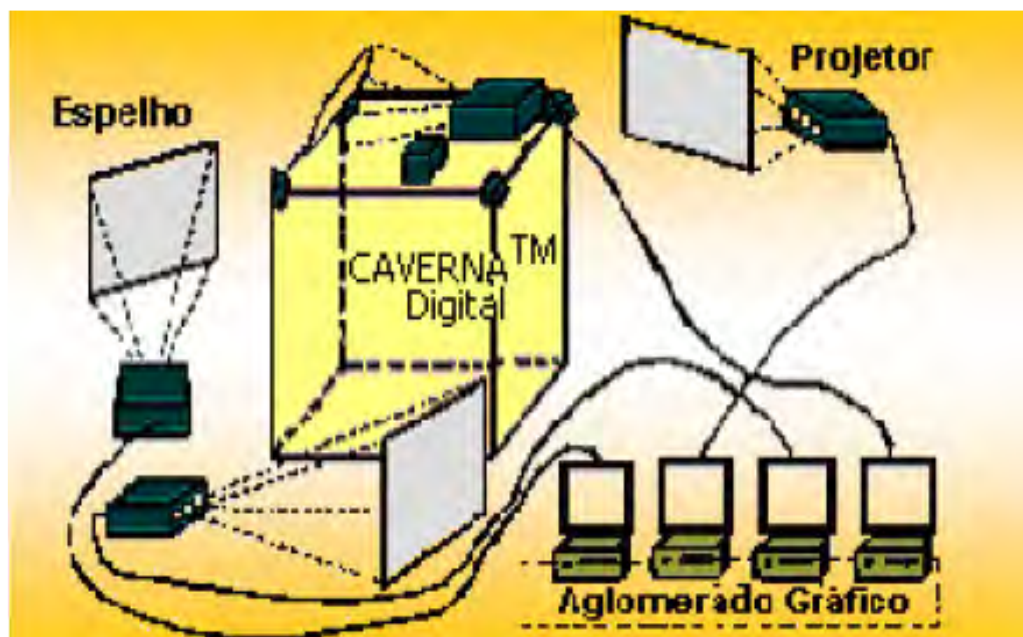


Figura 14 - Caverna Digital Gerenciada por Aglomerados Gráficos (GUIMARÃES, 2004)

5.2 – A Biblioteca Glass

A biblioteca Glass foi desenvolvida pelo Laboratório de Sistemas Integráveis do departamento de Engenharia de Sistemas Eletrônicos da Universidade de São Paulo, sendo denominada como DICElib. Inicialmente, esta biblioteca oferecia às aplicações apenas comunicação e sincronização entre os nós de um aglomerado gráfico, ficando carente na parte responsável pela facilidade de uso de dispositivos, facilidade de configuração e tolerância a falhas. Assim surgiu a Glass que adicionava tais recursos e aprimorava os já existentes, tornando-a uma biblioteca avançada para o desenvolvimento de aplicações de RV baseadas em aglomerados gráficos (GUIMARÃES, 2004).

Pelo fato da biblioteca Glass ter sido desenvolvida por Marcelo Guimarães (2004), a descrição desta sessão foi baseada inteiramente em sua tese, assim, as referências dos respectivos parágrafos não estão citadas.

As aplicações utilizam os serviços oferecidos pela Glass conforme suas necessidades, podendo ser utilizada também em qualquer aplicação baseada em aglomerado e até mesmo para portar aplicações de sistemas fortemente acoplados para serem executados em aglomerado gráfico sem a necessidade de refazer o projeto.

A Figura 15 apresenta uma visão geral dos componentes da Glass, isto é, sua arquitetura.

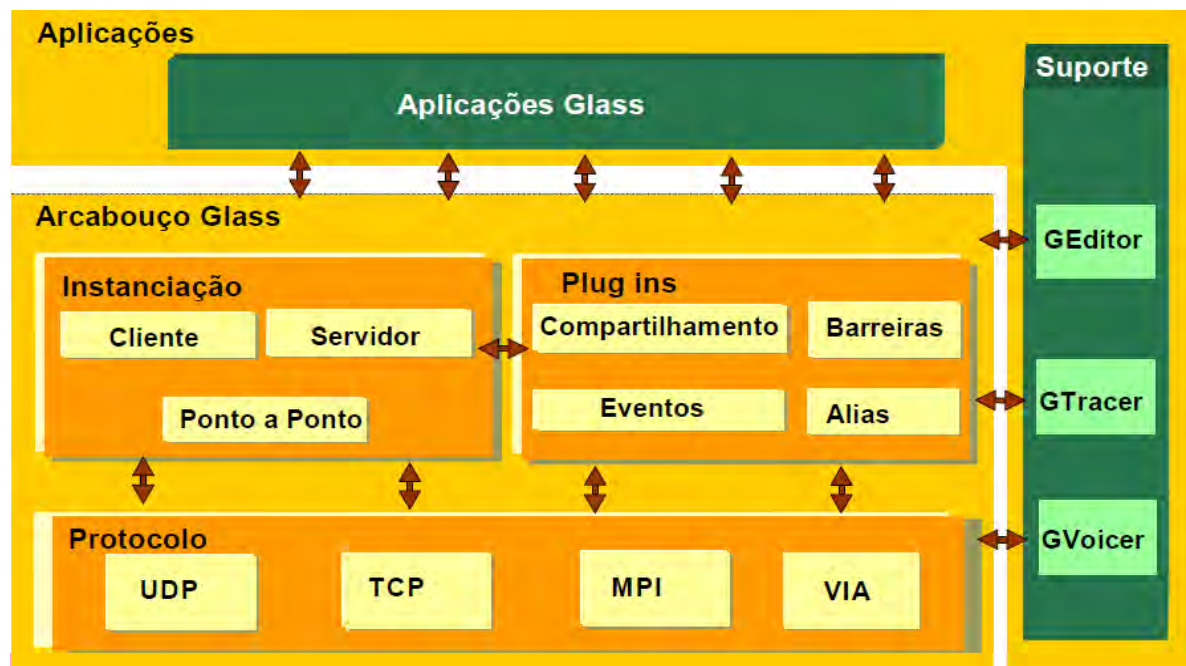


Figura 15 – Arquitetura Glass (GUIMARÃES, 2004)

Primeiramente tem-se o arcabouço da Glass, que é constituído pelos componentes: Instanciação, Protocolo e os *Plugins*.

O Instanciação tem como objetivo iniciar as aplicações conforme a arquitetura da Glass, permitindo a aplicação ser Cliente ou Servidor. O componente Protocolo encapsula bibliotecas, ocultando as diferenças entre os protocolos de comunicação, já que este componente possui uma infra-estrutura de empacotamento/desempacotamento de mensagens suportando todos os tipos básicos de dados, garantindo assim, a interoperabilidade entre sistemas operacionais.

O componente *Plugins* é responsável pelo gerenciamento dos *plugins*. Estes fornecem serviços de transmissão de eventos, de compartilhamento de dados, de barreiras de sincronização, de associações de funções, além de outros que podem ser adicionados conforme a necessidade, sem a modificação de sua estrutura interna.

Por fim tem-se o componente Aplicação, este é composto pelas Aplicações Glass e pelas Aplicações de Suporte. As aplicações Glass são as aplicações de RV baseado em aglomerados gráficos e as de suporte, são as ferramentas que dão auxílio ao desenvolvimento das Aplicações Glass.

5.2.1 – O arcabouço Glass

O arcabouço fornece a reutilização do código, sendo oferecidos por classes, objetos e até mesmo a reutilização do projeto, liberando o desenvolvedor dos aspectos comuns daquele domínio de aplicação, podendo até ser comparado com o conceito de classes abstratas.

O arcabouço Glass consiste em um modelo de classes que representa uma infraestrutura reutilizável e adaptável para as aplicações de RV baseadas em aglomerados gráficos, visando em atender diversas propriedades dessa técnica de desenvolvimento de *software*, como extensibilidade, flexibilidade e compreensibilidade.

A Figura 16 apresenta o Diagrama de Classe do arcabouço Glass, onde a chave é a classe abstrata *Glass*.

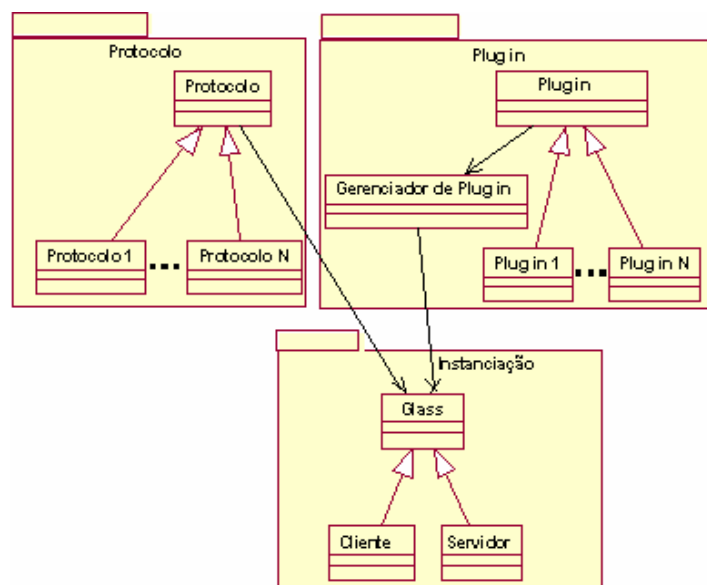


Figura 16 – Diagrama de Classe do arcabouço Glass (GUIMARÃES, 2004)

Nota-se a flexibilidade do projeto, visto que novos *plugins*, protocolos e arquitetura da aplicação podem ser adicionados a biblioteca, estendendo-a a novos serviços.

5.2.2 – Componente Instanciação

Este componente cria as instâncias das aplicações Glass utilizando-se a arquitetura Cliente /Servidor. Quando uma aplicação Glass é executada, inicialmente é executado um Servidor Glass, que permanece na espera para troca de comunicação com Clientes posteriormente iniciados. Ele gerencia todas as aplicações Clientes (conexão e desconexão) e as mensagens de comunicação (envio e recebimento). Todas as mensagens transmitidas são filtradas pelo Servidor a fim de reduzir as atualizações redundantes.

A Figura 17 apresenta o funcionamento da Glass com a arquitetura Sem Replicação.

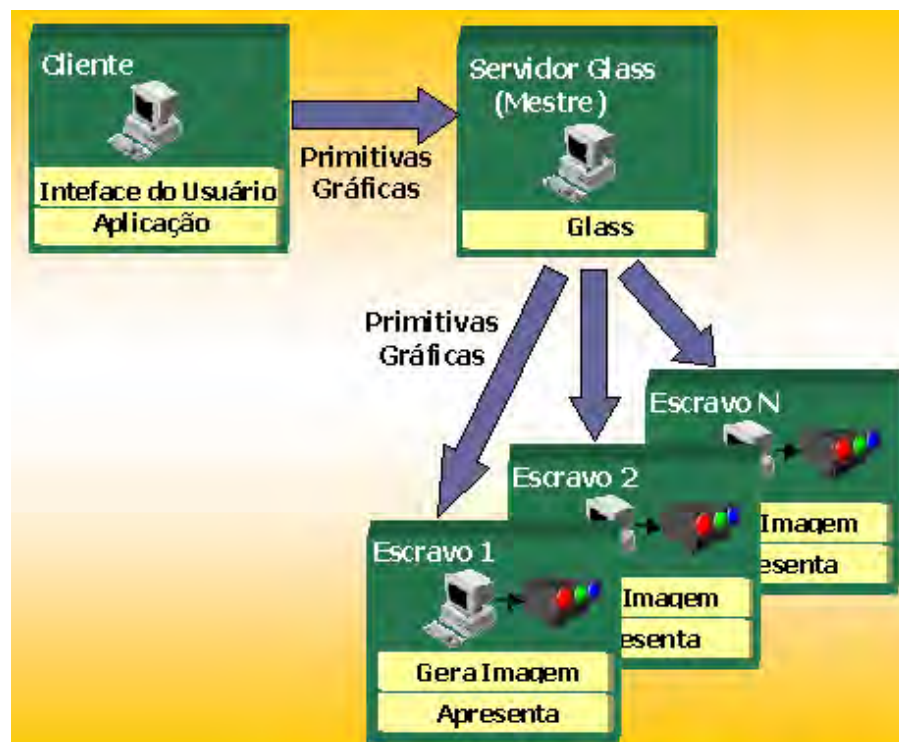


Figura 17 – Aplicação Glass na arquitetura Sem Replicação (GUIMARÃES, 2004)

Neste tipo de aplicação o nó Cliente recebe as interações do usuário, constrói e divide as primitivas gráficas entre os nós do aglomerado gráfico e posteriormente as envia para o Servidor Glass (Mestre). Quando o Mestre as recebe, as envia para os nós Escravos para gerarem e apresentarem as imagens.

A Figura 18 apresenta o funcionamento da Glass com a arquitetura Com Replicação.

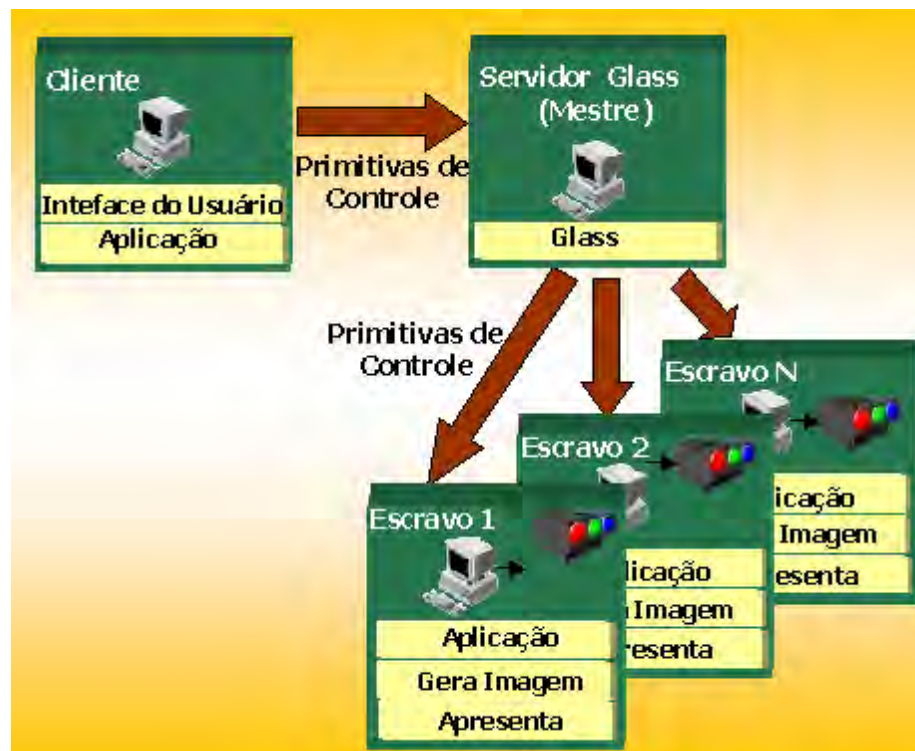


Figura 18 – Aplicação Glass com arquitetura Com Replicação (GUIMARÃES, 2004)

Neste tipo de aplicação o nó Cliente trata as interações do usuário gerando primitivas de controle que são enviadas para o Servidor Glass. Posteriormente são enviadas pelo Servidor para todos os nós que iram tratar e gerar as respectivas imagens.

Portanto, as aplicações podem ser desenvolvidas na arquitetura Com ou Sem Replicação, visando à comunicação e a sincronização entre os nós sendo totalmente transparente aos usuários.

5.2.3 – Componente Protocolo

O componente Protocolo foi projetado para possibilitar a utilidade de qualquer protocolo de comunicação pelas aplicações Glass, pelo fato da escolha do mesmo representar um ganho efetivo em uma determinada aplicação.

A Figura 19 a seguir apresenta o Diagrama de Classe Protocolo, onde a chave é a classe Protocolo.

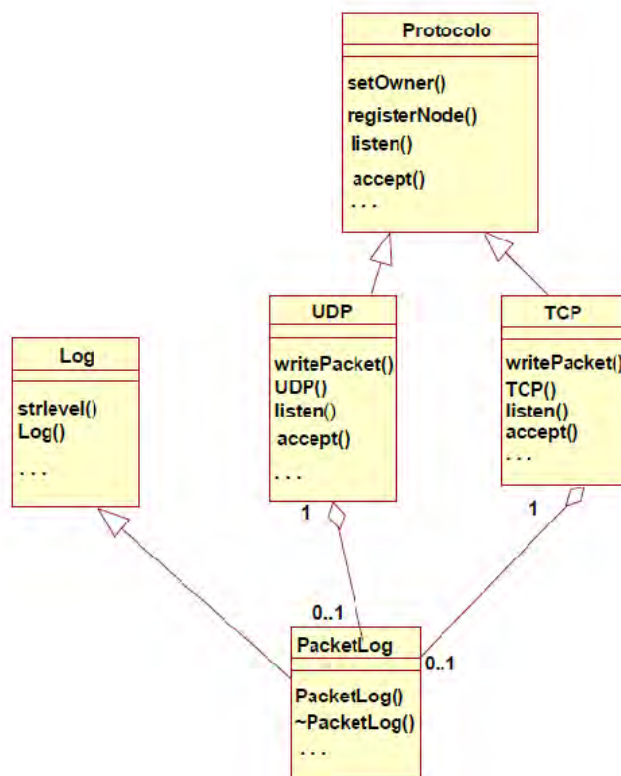


Figura 19 - Diagrama de Classe Protocolo (GUIMARÃES, 2004)

A classe Protocolo fornece as primitivas de envio e recebimento de dados e possuem as classes filhas UDP e TCP. O UDP é utilizado em ambientes onde a taxa de erros de transmissão é baixa e o TCP utilizado em ambientes onde ocorrem grandes retardos na transmissão de dados, devido a altas taxas de erros.

O Diagrama apresenta também a classe *PacketLog*, responsável por oferecer serviços de depuração de pacotes, ou seja, oferece serviços de armazenamento de todas as mensagens transmitidas entre os nós, independente do protocolo utilizado na transmissão.

O componente Protocolo demonstra como a Glass é flexível na parte de comunicação, podendo incorporar novos protocolos conforme a necessidade, porém os utilizando de maneira estática.

5.2.4 – Componente Plugins

O arcabouço da Glass é composto por quatro *plugins*, porem tal componente oferece flexibilidade podendo aumenta-los de acordo com a necessidade da aplicação. Os plugins fornecem recursos para o desenvolvimento de aplicações, como a sincronização (*plugin*

Barreiras), o compartilhamento de dados (*plugin* Compartilhamento), a transmissão de Eventos (*plugin* Eventos) e a associação de funções (*plugin* Alias).

O *plugin* Barreira garante a sincronização de valores e/ou tarefas. A Figura 20 apresenta o processo de sincronização.

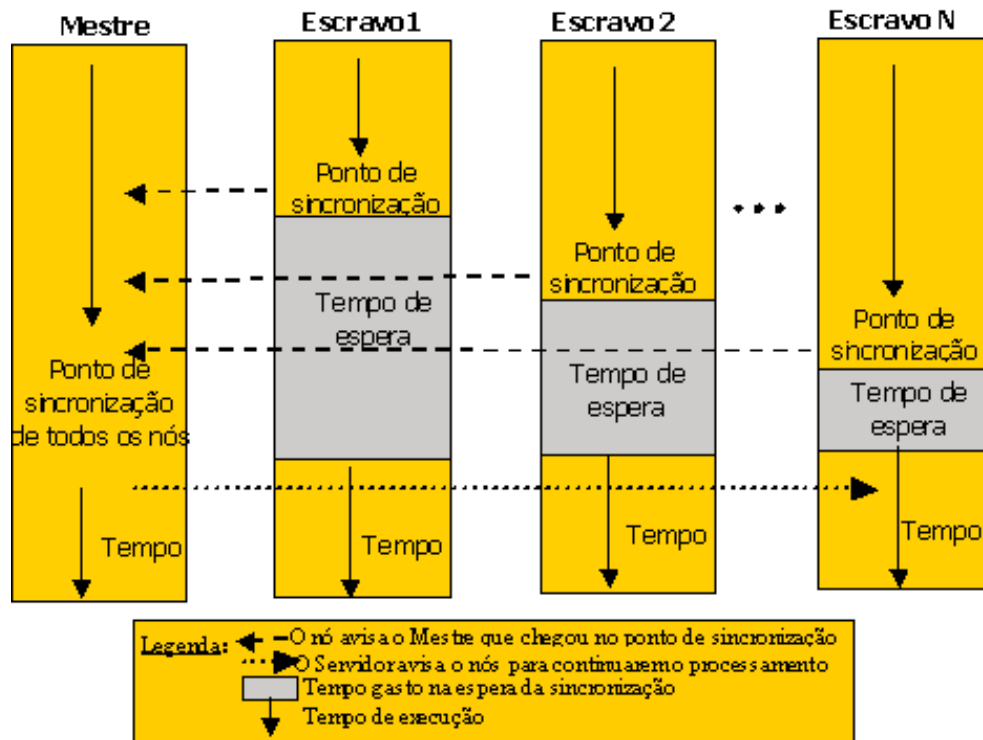


Figura 20 – Processo de Sincronização (GUIMARÃES, 2004)

Os nós Escravos avisam o Mestre que chegaram ao ponto de sincronização. Assim, todos ficam aguardando que todos os outros cheguem ao mesmo ponto, permanecendo no estado de espera. Por fim, quando o Mestre recebe a mensagem de sincronização de todos, este distribui uma mensagem broadcast avisando para que todos os nós Escravos continuem o processamento.

Além do mais, a Glass é capaz de perceber e recuperar *deadlocks* do sistema (travamentos), devido a finalização de modo anormal de uma aplicação. Quando um determinado nó termina sua execução de forma anormal, o Servidor da Glass detecta que este nó não está mais ativo e o desconsidera em tempo de execução, não aguardando mais a sua mensagem de sincronização. Entretanto quando é o Servidor que termina de modo anormal, o próximo nó (ordem de criação) é denominado como Servidor, em tempo de execução e o processo continua.

O *Plugin* Compartilhamento oferece recursos para a troca e compartilhamento de informações entre os nós, possibilitando a criação de variáveis assíncronas.

As variáveis podem ser variáveis já existentes (*int*, *float*) como também estruturas de dados. Entretanto deve haver um método que transforme esse tipo em string (empacota) para o envio e outro que a transforme para o novo tipo criado (desempacota).

Quando se pensa em vários processos compartilhando as mesmas variáveis, há o problema de condição de corrida (exclusão mútua). Neste caso a *Glass* adota um sistema de hierarquia, ou seja, um sistema com prioridade. Cada aplicação *Glass* é associada a um número durante a sua conexão com o Servidor, assim, os números são atribuídos de maneira crescente aos processos. Aquele que possuir o menor valor, maior será sua prioridade, além de que, apenas os nós que declararam à variável receberam as mensagens de atualização.

O *Plugin Alias* permite a associação de métodos a nós do aglomerado em tempo de execução. Desta forma, a responsabilidade de um nó de gerar a imagem da tela direita, em um sistema de multiprojeção, pode ser mudada para a tela da esquerda em tempo de execução. Assim, não há a necessidade de realizar o mapeamento de cada nó para uma determinada tela do computador de vídeo, antes da inicialização da aplicação.

Esta funcionalidade também pode ser utilizada para implementar um sistema de tolerância a falhas, na qual um determinado nó do aglomerado pode assumir uma tarefa de outro nó que parou de desempenhar.

Por fim tem-se o *Plugin Evento*, o qual fornece recursos que garantem a distribuição correta dos dados gerados por dispositivos de entradas (teclado, mouse, PDAs) para os nós responsáveis pelo tratamento.

Esse tratamento é de extrema importância, visto que em aplicações de multiprojeção, os pontos de vista de cada imagem devem ser precisos para não ocorrer incoerência na mesma.

Apesar do objetivo do *plugin* ser o tratamento de dados de dispositivos, este pode também ser usado para o envio de qualquer tipo de dado, como por exemplo, avisar que todos os nós devem finalizar a execução da aplicação corrente.

5.2.5 – Aplicações Glass

A *Glass* possui dois grupos de aplicações, as Aplicações Glass e as Aplicações de Suporte. As Aplicações Glass são as aplicações desenvolvidas pelos programadores, na qual

podem utilizar de outras bibliotecas para a implementação das mesmas e ainda podendo utilizar diversos dispositivos de multiprojeção e de interação em multiplataforma.

Esta biblioteca foi implementada para aglomerados gráficos, podendo ter nós heterogêneos ligados por uma rede local. Contudo, o projeto está sendo estendido para que possa haver aplicações heterogêneas remotas, utilizando-se o padrão XML para a transferência de dados. O XML além de facilitar a interação de dados de diferentes fontes e incompatíveis, permite a compressão, é suportado pela maioria dos produtos disponíveis e possibilita que diferentes nós visualizem os mesmos dados de maneira diferente.

A Figura 21 apresenta a integração de dois aglomerados remotos, onde os nós são heterogêneos. Nesta arquitetura, os nós Clientes recebem as interações dos usuários; os nós Escravos executam as tarefas vindas do Mestre; e os nós Mestres além de receber os dados dos nós Clientes e enviar os comandos para os nós Escravos, comunicam-se entre si, utilizando o padrão XML, para manter a sincronia entre os aglomerados gráficos.

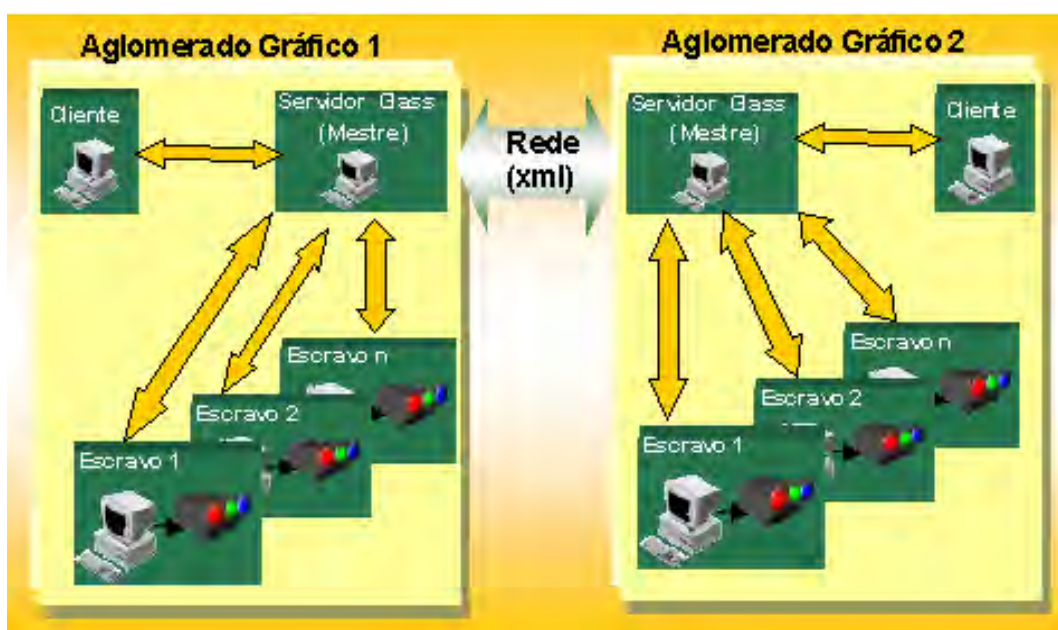


Figura 21 – Utilização do padrão XML para integração de aglomerados remotos. (GUIMARÃES, 2004)

Por fim têm-se as Aplicações de Suporte que auxiliam no desenvolvimento de aplicações da Glass. Neste grupo têm-se três aplicações:

- O *GTracer*: possibilita a visualização gráfica de todas as mensagens transmitidas entre os nós;
- O *GEditor*: permite a geração automática de aplicações para PDAs; e

- O *GVoicer*: auxilia o desenvolvimento automático do controle via voz das aplicações.

5.3 – Java 3D

A API Java 3D consiste em uma hierarquia de classes Java que serve como uma interface para o desenvolvimento de sistemas gráficos tridimensionais. O programador necessita trabalhar apenas com construtores de alto nível para criar, manipular e visualizar objetos geométricos, tendo os detalhes de visualização gerenciados automaticamente. Um programa Java 3D cria instâncias de objetos gráficos que são colocados em um grafo de cena, que também pode conter luz, som e outros elementos que possibilitam a criação de universos virtuais. O grafo de cena consiste em uma combinação de objetos 3D em uma estrutura de árvore que especifica o conteúdo do universo virtual e como este deve ser visualizado (DAVISON, 2007).

Java 3D foi construída com o objetivo de criar uma API que fosse independente de plataforma, semelhante ao VRML (AMES et. al., 1997). Então, a *Intel*, a *Silicon Graphics*, a *Apple* e a *Sun Microsystems*, em colaboração, definiram a especificação Java 3D. Em 1998, a Sun colocou a sua primeira implementação para download.

Java 3D consiste em uma API baseada nas bibliotecas gráficas *OpenGL* (WRIGHT Jr; SWEET, 2000) e *DirectX* (DUNLOP, 2000), e os programas podem ser escritos como aplicação, *applet*, ou ambas.

Atualmente, Java 3D pertence a Oracle e sua última versão disponível para download se encontra na 1.5.3 . O download pode ser realizado pelo site da Oracle (ORACLE, 2011).

5.4 – X3D (*Extensible 3D Graphics*)

O X3D é um padrão aberto para distribuir conteúdo 3D. Ele não é uma API de programação, tão pouco um formato de arquivo para troca de geometrias. Ele é uma linguagem em forma de texto para descrever formas em 3D e interagir com o meio ambiente, combinando geometrias e descrições de comportamentos em um simples arquivo, contendo vários formatos de arquivos, incluindo o XML. É a próxima revisão da especificação ISO VRML97, incorporando os avanços dos recursos disponíveis nos últimos dispositivos gráficos comerciais tanto quanto melhoras em sua arquitetura (WEB3D CONSORTIUM, 2011).

Um arquivo X3D tem a extensão .x3d e é estruturado da seguinte maneira:

- O cabeçalho de arquivo em XML (Configurações);
- O cabeçalho em X3D (Definição da versão, do perfil e configurações);
- Informações META (Comentários);
- Perfis (Componentes – Funcionalidades);
- Nós;
- Campos; e
- Valores.

O X3D é estruturado em grafo de cena, possui uma única interface de programação de aplicações (API), suporta múltiplas codificações de arquivos, emprega uma arquitetura modular para prover uma maior extensibilidade e flexibilidade e vem sendo reestruturada ao longo dos anos para acomodar as evoluções de padrões, permitindo um ciclo de vida mais útil (WEB3D CONSORTIUM, 2011).

O padrão X3D é utilizado no desenvolvimento do sistema, por ser estruturado em forma de grafo de cena, garantindo uma fácil integração com o Java 3D, por permitir a extensibilidade e flexibilidade, possibilitando a utilização para outros fins, além de oferecer padronizações que garantem a utilização em diversos AVs e ferramentas.

5.5 – Protocolos de Comunicação – SSH e SCP

O protocolo *Secure Shell* (SSH) é um protocolo de rede que permite a conexão com outros computadores na rede, de forma a executar comandos de uma unidade remota. Este utiliza uma arquitetura cliente/servidor para proporcionar a autenticação, a criptografia e a integridade dos dados trafegados na rede (DESIRE, 2010).

Uma de suas mais utilizadas aplicações é o chamado *Tunnelling*, que oferece a capacidade de redirecionar pacotes de dados. O processo se caracteriza por duas máquinas ligadas ao mesmo servidor SSH, que faz apenas o redirecionamento das requisições do computador que está sob firewall. Tal prática não é ilegal caso o fluxo de conteúdo esteja de acordo com as normas da instituição (DESIRE, 2010).

O SSH faz parte da suíte de protocolos TCP/IP que torna segura a administração remota de um servidor Unix.

O Protocolo *Secury Copy* (SCP) apenas implementa a transferência de arquivos. No entanto o protocolo em si não provê nenhuma autenticação ou segurança; ele depende do protocolo abaixo, o SSH, para fazê-lo (DESIRE, 2010).

Para a execução de uploads de arquivos, o cliente fornece ao servidor os arquivos a serem enviados, opcionalmente incluindo-se seus atributos básicos, como por exemplo, suas permissões de acesso. Para fazer o download dos arquivos, o cliente envia um pedido ao servidor requisitando os arquivos ou os diretórios a serem baixados e este fornece ao cliente seus subdiretórios e arquivos correspondentes. Portanto o download é "orientado ao servidor", o que impõe um risco de segurança, quando se estiver conectado a um servidor malicioso.

5.6 – Considerações Finais

O capítulo apresentou as tecnologias utilizadas para o desenvolvimento do trabalho, apresentando conceitos e destacando o papel e a importância de cada.

O próximo capítulo apresenta a descrição do projeto, destacando a arquitetura do sistema e sua modularização, além de ilustrações demonstrando o modo da organização de cada grafo de cena respectivo ao seu AV e/ou HV. É também apresentado a maneira com que o usuário interage e manipula o sistema, explicando suas interfaces e realçando os resultados obtidos com o desenvolvimento do trabalho.

Capítulo 6 - Descrição do Projeto

Com a finalidade de suprir os objetivos apresentados na introdução, a ferramenta propõe o desenvolvimento de uma aplicação capaz de proporcionar e estimular o aprendizado e compreensão de Língua de Sinais, visto que os sinais utilizam gestos humanos e são realizados no espaço. Desta forma, disponibilizar um complemento tecnológico para a educação especial, estimulando os envolvidos através de imagens computadorizadas, valoriza o processo de comunicação e aprendizagem.

O sistema é destinado ao domínio de LIBRAS, possibilitando a interpretação e a edição de movimentos através de dispositivos não convencionais. As principais tarefas do sistema são: interpretação, edição e gravação de movimentos em LIBRAS.

Na tarefa de interpretação é utilizado uma estrutura de multiprojeção baseada em aglomerados gráficos, garantindo a estereoscopia de imagens e o processamento distribuído dos dados. A fim de manter a sincronização dos dados para garantir o realismo da multiprojeção em um Sistema de Projeção (Wall), a biblioteca Glass é utilizada. Vale ressaltar, que o Sistema de Projeção foi baseado na estrutura da Cave definida por Carolina Cruz-Neira (CRUZ-NEIRA et al., 1993).

Na tarefa de edição de movimentos são utilizados dispositivos não convencionais para o rastreamento corporal. Dentre os dispositivos estão os controladores Wiimotes e a luva P5-Glove. Os controladores são anexados nos braços e antebraços direito e esquerdo para simularem os movimentos reais. A luva é anexada na mão direita e simula os movimentos da mão e dos seus referidos dedos, repassando-os ao HV.

Após a edição dos movimentos em tempo real, a tarefa de gravação permite que os movimentos gerados sejam armazenados em um banco de dados para posterior representação em LIBRAS no Sistema de Projeção. A Figura 22 ilustra a arquitetura do sistema.

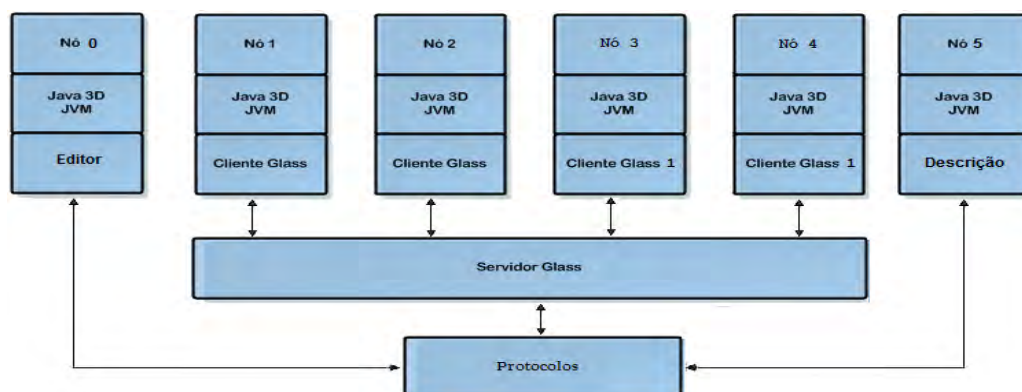


Figura 22 - Arquitetura do Sistema Gerador de Apoio de um Dicionário Temático

Neste modelo de arquitetura são utilizadas seis aplicações, sendo as nomeadas de um a quatro, responsáveis pela visualização estereoscópica de modelos virtuais tridimensionais e do HV. Estas são sincronizadas pelo Servidor Glass a partir de clientes que estão associados às mesmas. Cada par (1 e 2, 3 e 4) são vinculados com um tipo de cliente Glass, permitindo a sincronização das imagens e a troca de informação entre os modelos virtuais e o HV.

A aplicação nomeada cinco também é projetada no Sistema de Projeção, ficando a responsabilidade de aumentar a usabilidade do sistema, permitindo a descrição dos modelos virtuais em três idiomas.

A aplicação inicial é a nomeada zero, está habilita os protocolos para iniciar todas as demais aplicações de multiprojeção no Sistema de Projeção, além de permitir a edição e o armazenamento dos movimentos na máquina local.

Enfim, o sistema propõe uma aplicação de RV capaz de aumentar a imersão e a interação dos envolvidos do sistema com Língua de Sinais. Desta forma, um HV interpreta e edita movimentos de LIBRAS em um ambiente sintético tridimensional, a partir de uma interface gráfica que provê mecanismos auxiliares. Tais mecanismos além de proverem a multiprojeção estereoscópica e sincronismo, proveem dispositivos não convencionais para a edição e gravação de movimentos. Ambos os módulos (interpretação e edição) utilizam um banco de dados para o armazenamento e recuperação de dados.

6.1 – Modularização do Ambiente

Um detalhe a ser destacado é que o sistema foi adaptado e aprimorado por Fabricio Baptista (BAPTISTA, 2007), tendo como principal objetivo agrupar as funções de edição e interpretação, incorporando estas facilidades em um único ambiente e na mesma interface.

Com o intuito de aderir à mesma metodologia, o usuário ainda continua com a visão de um único sistema, porém como as tarefas associadas a cada módulo são totalmente independentes, este foi dividido no módulo interpretador e no módulo editor.

A Figura 23 ilustra a modularização do atual sistema e a Tabela 2 apresenta a sua legenda.



Figura 23 - Modularização do Sistema

Tabela 2 – Argumentos da Modularização do Sistema

Item	Descrição
Argumento 1	Métodos para inicializar o interpretador local com a palavra para a representação em LIBRAS e receber o retorno visual
Argumento 2	Comandos remotos para inicializar a multiprojeção e encerrá-la
Argumento 3	Instruções de movimentos
Argumento 4	Comandos SQL para retornar o caminho do modelo virtual e carrega-lo
Argumento 5	Métodos para inicializar os dispositivos

O módulo interpretador pode ser visualizado de duas maneiras. A primeira, permite a representação de palavras brasileiras para sinais de LIBRAS localmente, sendo a palavra analisada e seus dados correspondentes recuperados do banco de dados e visualizados através do HV. A segunda, possibilita a visualização estereoscópica de sinais no Sistema de Projeção, por meio de modelos virtuais tridimensionais com suas descrições em três idiomas: português, inglês e espanhol.

A estrutura de tradução dentro do módulo interpretador foi dividida em duas partes menores: o módulo de análise e o módulo de RV. O módulo de análise é responsável por receber a sentença em LIBRAS, analisa-las e gerar as instruções dos movimentos correspondentes, tais instruções são enviadas para o módulo de RV que posteriormente executa os movimentos.

Quando acionado o módulo interpretador, além da visualização de movimentos localmente, permite a associação de palavras com modelos virtuais e os copia para a máquina servidora do *cluster* (aglomerado gráfico), por meio de comandos SCP, deixando os modelos virtuais disponíveis a todas as máquinas.

Se acionado o interpretador do Sistema de Projeção, este dispara o protocolo SSH para acionar os mecanismos de sincronização na ativação da multiprojeção. Assim, o Servidor, o modelo virtual e o HV são instanciados e sincronizados em suas respectivas telas de projeção, provendo a visualização estereoscópica, junto a sua descrição e representação em LIBRAS. A Figura 24 ilustra detalhadamente o módulo interpretador do Sistema de Projeção.

O módulo editor permite ao usuário do sistema, gravar os movimentos previamente gerados pelos controladores Wiimote e pela Luva, sendo possível descartar os movimentos caso estes não sejam satisfatórios ao usuário. O módulo Editor também exclui todas as instruções de uma determinada palavra antes de grava-las no banco de dados, garantindo a consistência e a coerência dos movimentos.

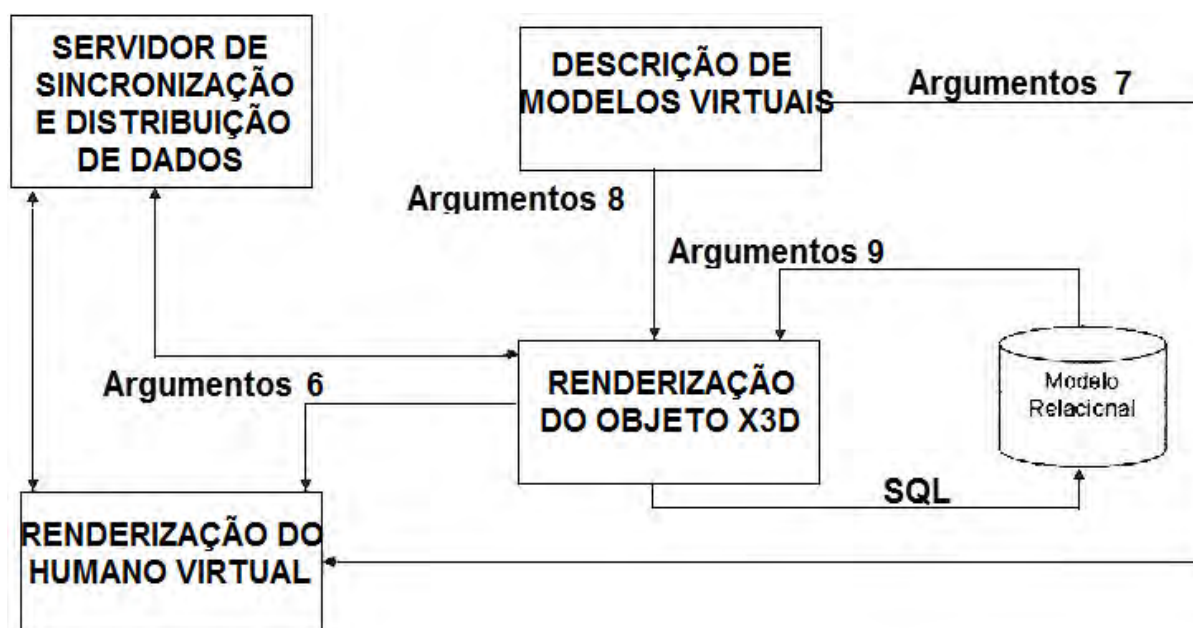


Figura 24 - Módulo Interpretador do Sistema de Projeção

A Tabela 3 apresenta a legenda da Figura 24, ressaltando os seus argumentos.

Tabela 3 – Argumentos da Representação de LIBRAS no Sistema de Projeção

Item	Descrição
Argumento 6	Variáveis compartilhadas, eventos compartilhados e barreiras de sincronização
Argumento 7	Comandos remotos para iniciar o HV
Argumento 8	Comandos remotos para iniciar o modelo virtual
Argumento 9	Comandos SQL para indicar o caminho do modelo virtual

6.2 – Modelagem do Sistema Gerador do Dicionário Temático

A Figura 25 ilustra o Diagrama de Caso de Uso do sistema, descrevendo suas funcionalidades, ou seja, apresentando a sequência de eventos do ator que utiliza o sistema para completar o processo interpretador ou editor.

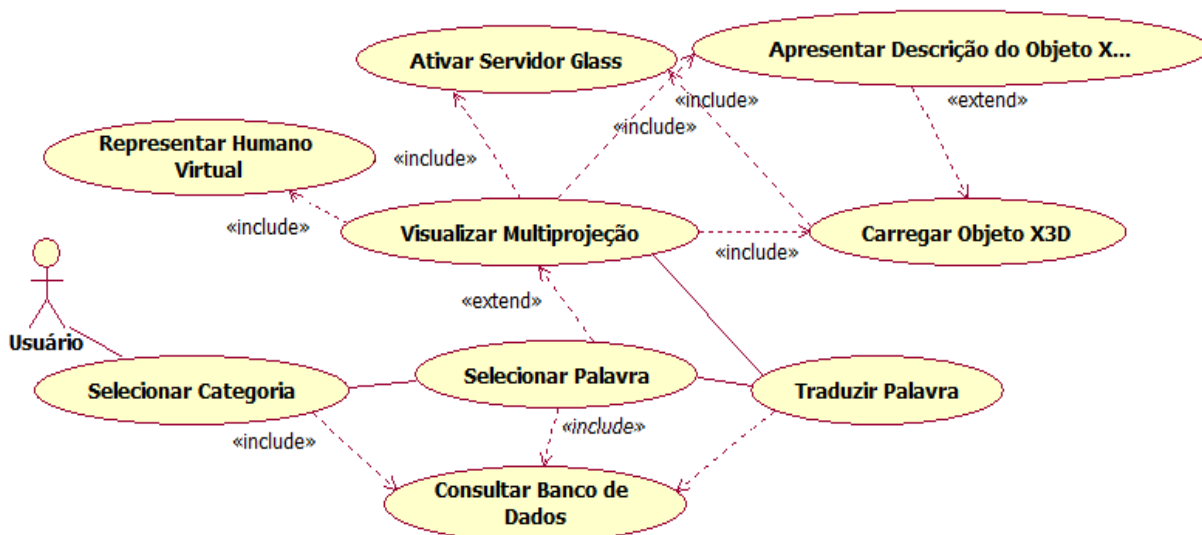
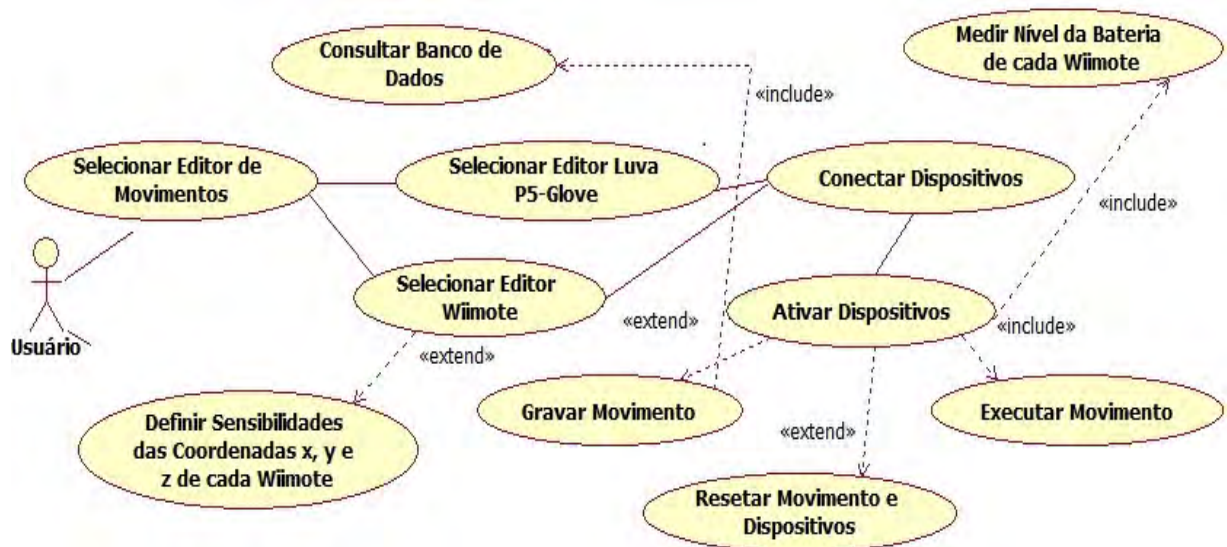


Figura 25 - Diagrama de Caso de Uso Interpretador



De acordo com os diagramas de caso de uso, em um primeiro momento, o usuário pode habilitar o módulo interpretador ou o módulo editor. Se optar pelo interpretador (Figura 25), este possibilita a representação de palavra para gestos em LIBRAS na máquina local, além de permitir a multiprojeção no Sistema de Projeção, ativando automaticamente o Servidor Glass, o HV e o objeto x3d (o modelo virtual) junto a sua descrição, sendo ambos estereoscópicos.

Caso opte pelo módulo editor (Figura 26), este permite utilizar os controladores Wiimotes, ou a Luva P5-Glove ou ambos. Com a escolha dos dispositivos, é possível conectá-los e ativá-los, além de que, se for utilizado os controladores Wiimotes, este ainda permite definir a sensibilidade das coordenadas x, y e z. Após estas operações o sistema habilita e permite: executar os movimentos, gravá-los e se necessário, reiniciar o HV juntamente com os dispositivos.

Para a visualização da estrutura e do relacionamento das classes mais relevantes implementadas no sistema, a Figura 27 e a 28 ilustram o diagrama de classe simplificado do respectivo, sendo o completo apresentado no Apêndice A e no B. O diagrama de classes foi dividido em dois diagramas de classes, o do módulo interpretador e do módulo editor. Isto se fez necessário, devido ao grande número de classes presentes no diagrama, tornando o entendimento melhor se apresentando separadamente. A Figura 27 apresenta o diagrama de classe do módulo editor e a Figura 28 o diagrama de classe do módulo interpretador com seus respectivos relacionamentos.

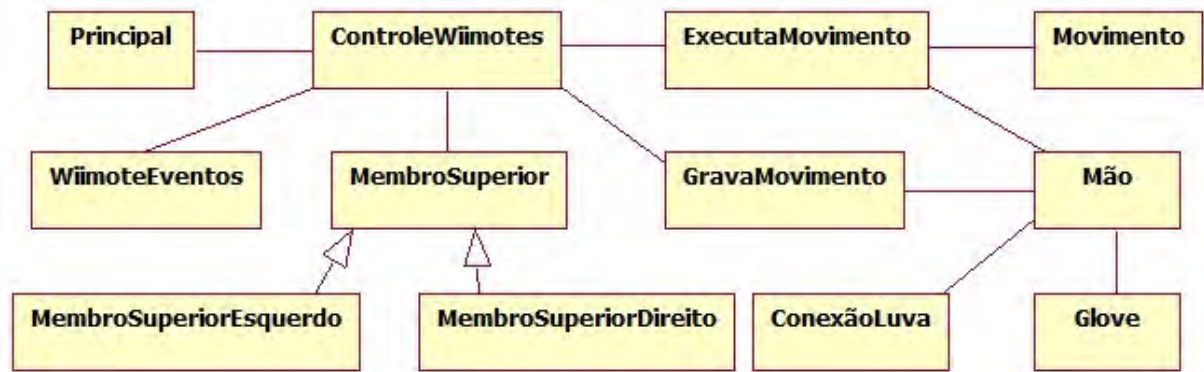


Figura 27 - Diagrama de Classe Editor

O módulo editor possui as seguintes classes principais:

- WiimoteEventos: reconhece controladores Wiimote para cada membro (braço e antebraço, esquerdo e direito);
- ControleWiimote: cria a interface para conectar, ativar, gravar e resetar os controladores, sendo a classe que mantém a comunicação com a classe Principal;
- ExecutaMovimento: executa os movimentos das partes superiores do corpo humano, juntamente com a mão direita e seus dedos;
- GravarMovimento: grava o movimento executado em um banco de dados;
- Movimento: armazena a granularidade mínima dos movimentos gerados;
- MembroSuperior: normaliza os valores gerados pelo Wiimote, calculando o seu respectivo ângulo e recebendo os eventos dos movimentos gerados por cada Wiimote;
- MembroSuperiorDireito e MembroSuperiorEsquerdo: utilizam a classe MembroSuperior para atualizarem as coordenadas do braço e antebraço direito e esquerdo;
- Glove: inicializa e encerra a conexão com a luva, além de atualizar os valores gerados por ela para cada dedo;
- Mao: normaliza os valores gerados pela luva, calculando o respectivo ângulo e recebendo os eventos de movimento da mesma; e
- Conexao_Luva: criando a interface para ativar, resetar e gravar os movimentos gerados pela luva, além de manter a conexão com a classe Principal.

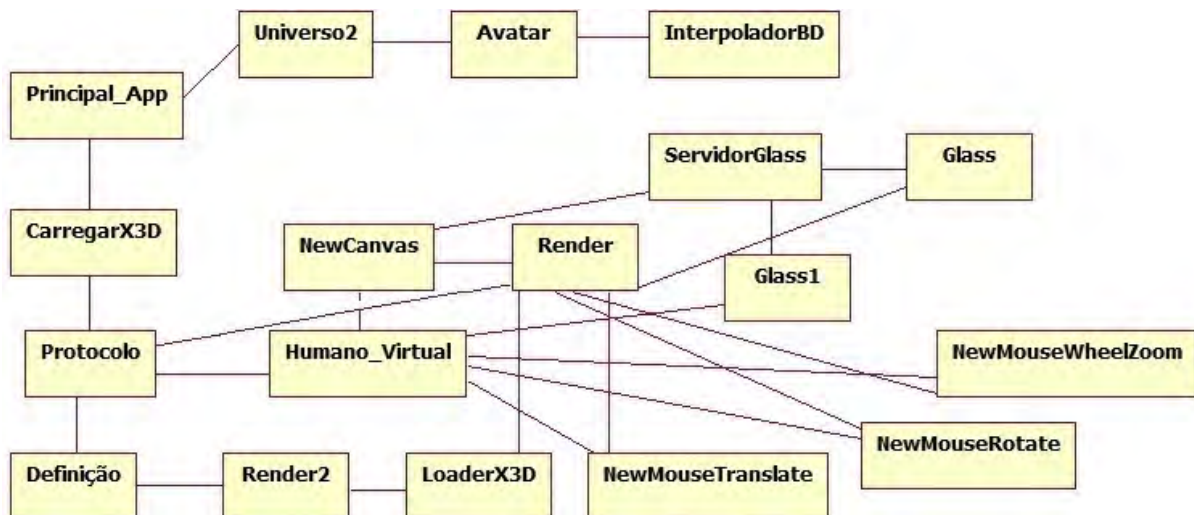


Figura 28 – Diagrama de Classe Interpretador

O módulo interpretador possui as seguintes classes principais:

- **Universo2**: responsável por renderizar o HV na aplicação local, mantendo a comunicação com a classe principal que controla o acesso às funcionalidades do sistema;
- **Avatar**: carrega as partes do corpo do avatar com suas formas, texturas e transformações, criando o grafo de cena;
- **InterpoladorBD**: executa os movimentos e fica sendo acionado sempre que ocorra um evento, pois estende a classe Behavior;
- **CarregarX3D** e **LoaderX3D**: carregam o modelo virtual tridimensional;
- **Protocolo**: envia os parâmetros necessários para as classes responsáveis por renderizar o objeto X3D, o HV e a descrição do modelo virtual;
- **Humano_Virtual**: carrega o humanóide e o sincroniza com a classe Glass1, para a utilização de estereoscopia;
- **Descrição** e **Render**: carregam uma prévia do modelo virtual junto a suas descrições;
- **NewCanvas**: sobrescreve a classe Canvas3D para a utilização do polimorfismo do método prerender, responsável por renderizar o objeto do AV de acordo com as novas coordenadas disponíveis;
- **Render**: renderiza o modelo virtual e o sincroniza com a classe Glass, garantindo a estereoscopia;
- **ServidorGlass**: sincroniza os dados dos clientes Glass;

- NewMouseTranslate, NewMouseRotate e NewMouseWheelZoom: oferecem o comportamento do mouse para as classes Humano_Virtual e Render, permitindo que as novas coordenadas do mouse sejam distribuídas e sincronizadas entre as aplicações através do ServidorGlass.

6.3 – Ambientes Virtuais

Os módulos de edição e interpretação agem diretamente sobre os AVs. Este um pré-requisito fundamental para a exibição do Avatar na representação e edição de LIBRAS e da visualização do modelo virtual.

Para a implementação dos AVs inicialmente é necessário definir um modelo de grafo de cena para cada componente, este sendo utilizado para gerar e demonstrar as relações entre os nós do grafo, ou seja, apresentar os componentes responsáveis por posição, escala, cores, formas e pontos de vista do objeto. A Figura 29 ilustra o grafo de cena do AV do Avatar.

O grafo de cena apresenta resumidamente os componentes deste ambiente, sendo possível observar que a partir do nó Raiz, são ligados os componentes de iluminação (Luz e LuzD) e o nó Geral, tendo como dependente um nó de transformação denominado Grupo e um nó denominado Mouse, necessários para possibilitar a movimentação dos objetos.

O objeto Interpolador e o InterpoladorBD são responsáveis por realizarem todos os movimentos de interpretação e de edição no HV, respectivamente, sendo representado pelo objeto Avatar.

O nó de transformação denominado Carregar é responsável por manipular a visualização da câmera no AV, sendo possível a visualização estereoscópica do Avatar. Este possui os nós ViewPlatform e AvatarEscala, responsáveis respectivamente, por captar a visão do AV e de aplicar as transformações sobre o mesmo.

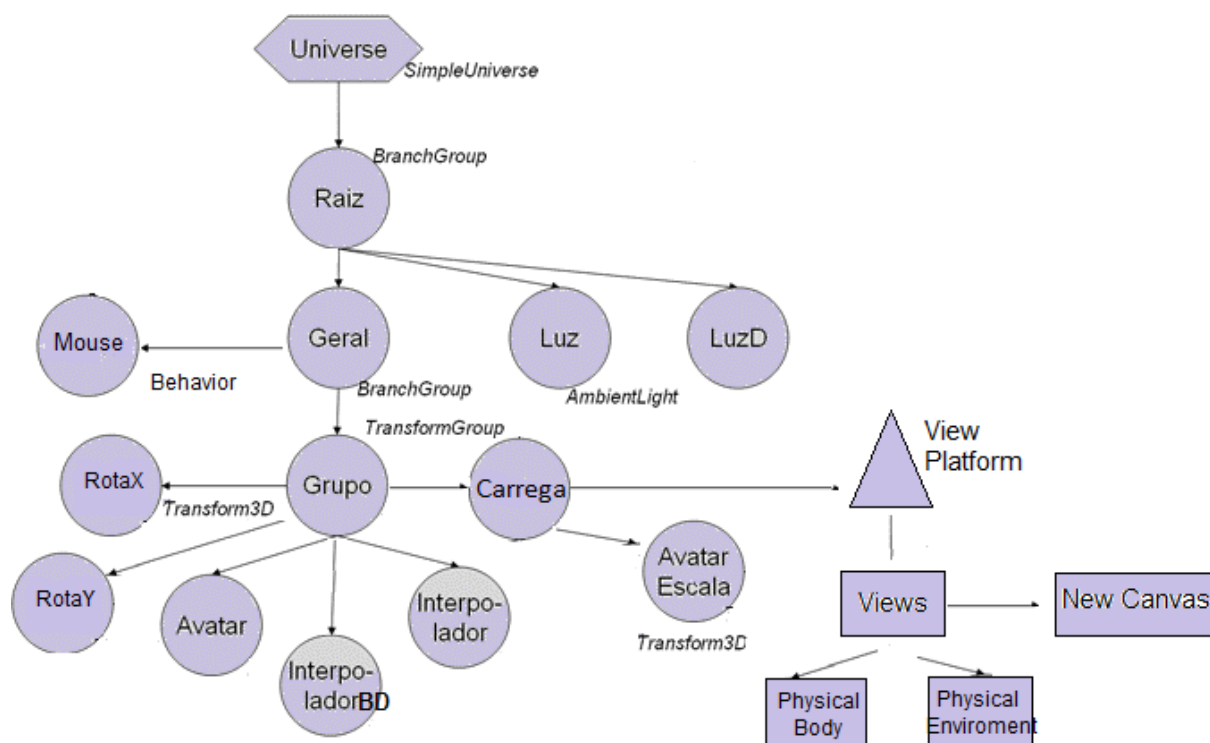


Figura 29 - Grafo de Cena do AV

Para compreender o funcionamento do HV, ou seja, a maneira como ele executa e edita os movimentos, é importante explicar a estrutura hierárquica do objeto Avatar, sendo as partes do corpo divididas em nós, conforme ilustrado na Figura 30. A hierarquia determina qual componente do corpo pode se movimentar sem que outras partes se movam juntas ou qual parte se movimentada, movem os demais nós descendentes.

De acordo com a Figura 30, o nó principal é o nó corpo, na qual a partir dele, pode-se obter toda a hierarquia. A partir do elemento tronco são conectados todos os objetos que representam cada membro do corpo humano, possuindo uma referência para cada uma de suas filhas. O grafo de cena ilustra apenas os membros da parte direita do avatar.

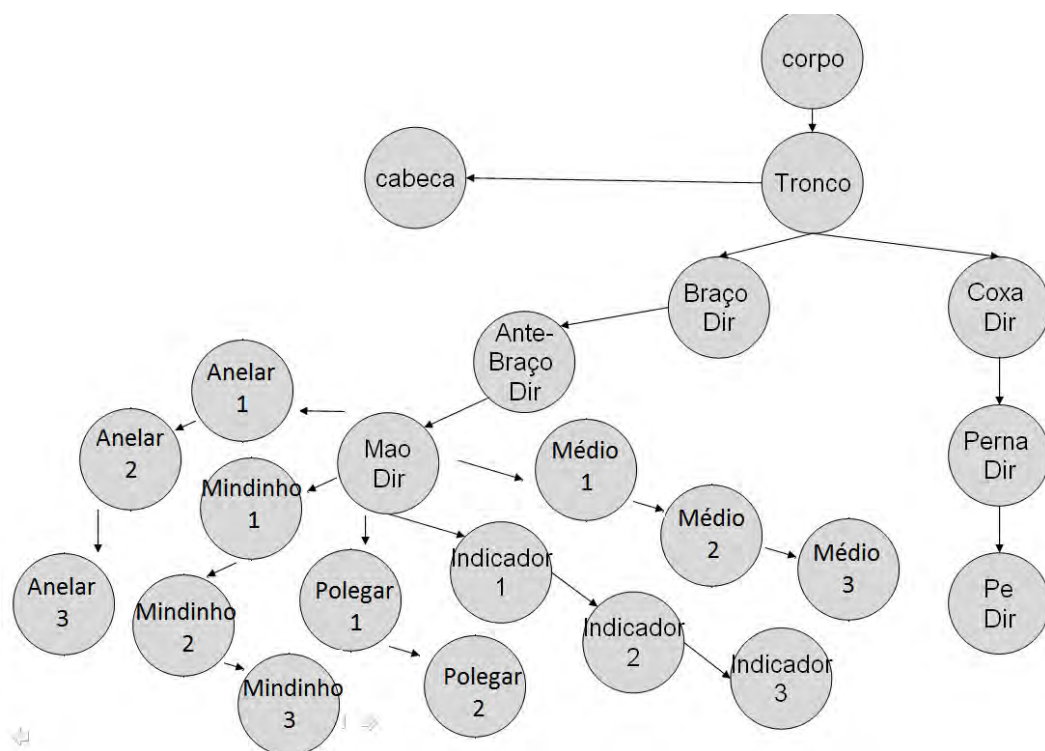


Figura 30 - Grafo de Cena do HV

O grafo de cena que representa resumidamente o AV onde são renderizados os modelos virtuais é apresentado na Figura 31. Neste grafo há dois nós iniciais, o sceneRoot e o sceneGroup, ambos responsáveis por anexar a cena ao universo.

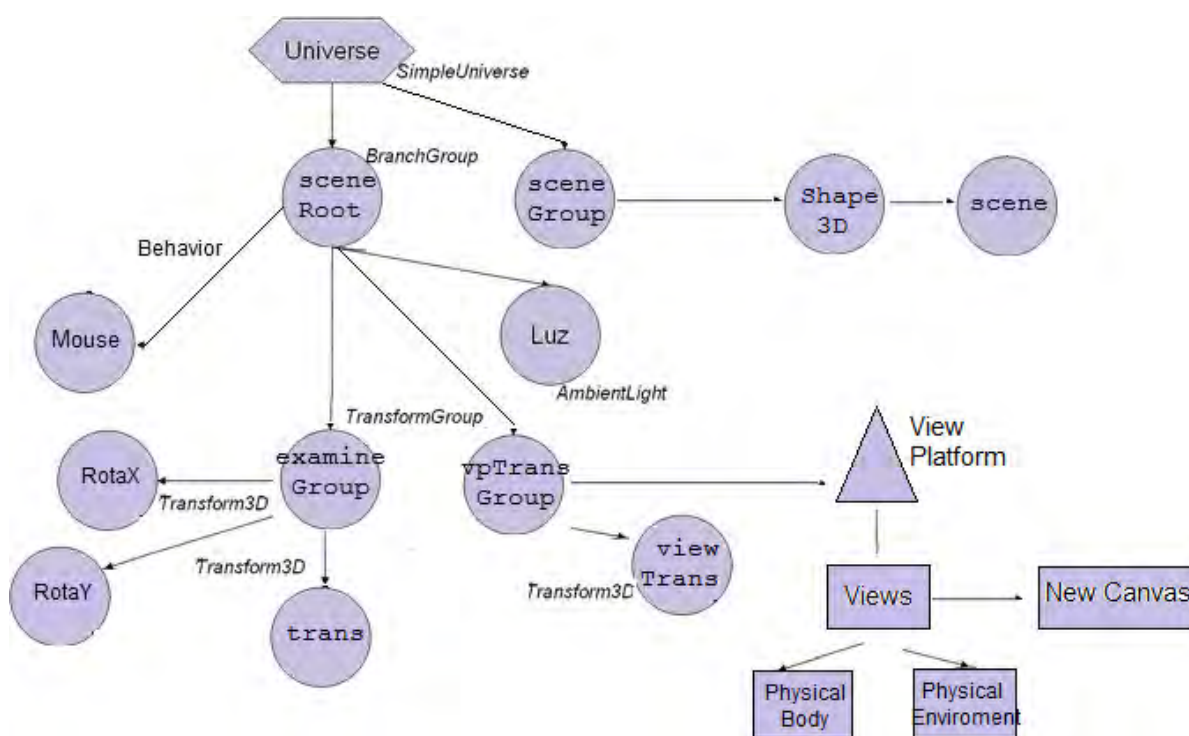


Figura 31 - Grafo de Cena do AV do objeto X3D

O `sceneGroup` renderiza um nó `Shape` do tipo `Scene`, responsável por carregar o modelo virtual.

O nó `sceneRoot` possui dois nós de transformação, o nó `examineGroup` e o nó `vpTransGroup`. O nó `examineGroup` é responsável por realizar a movimentação do objeto, possuindo nós de `Transform3D` para a aplicação de transformações. O nó de transformação denominado `vpTransGroup` é responsável por manipular a visualização da câmera no AV, sendo possível a visualização estereoscópica dos modelos virtuais. Este possui os nós `ViewPlatform` e `viewtrans`, responsáveis respectivamente por, captar a visão do AV e de aplicar as transformações sobre o mesmo.

6.4 – Classe Principal

Como visto nos diagramas de classes apresentados e especificados nas sessões anteriores, o sistema possui diversas classes, tanto para o módulo interpretador quanto para o módulo de edição. Assim, nesta sessão são apresentadas as classes que compartilham os dois módulos, e nas subsequentes, são apresentados detalhadamente cada módulo.

A classe `Principal` é a responsável pela interface inicial do aplicativo, fornecendo a interface gráfica para acessar todas as funcionalidades do sistema. A partir dela são instanciados os objetos necessários para o tratamento dos eventos gerados pelo usuário. Assim, há eventos implementados que permitem a tradução de palavras para movimentos de LIBRAS, além de eventos que instanciam objetos para a edição de movimentos, que vinculam palavras com modelos virtuais e que acionam e encerram a multiprojeção.

A classe `Universo2` é uma instância de um `SimpleUniverse`, contendo o ambiente responsável por representar os movimentos de LIBRAS, tanto para o módulo interpretador quanto para o módulo editor da aplicação local. Na formação da cena são utilizados objetos gráficos, um conjunto de luzes, uma cor predominante para o fundo, além de comportamentos de mouse associados à cena, possibilitando rotações, translações e escalas.

A classe `Avatar` é instanciada pelo `Universo2`, responsável por implementar a representação do corpo humano no AV. Esta carrega os objetos geométricos do corpo do HV(arquivos obj), além de classes que representam cada membro possuindo referências para os objetos da cena. Cada uma dessas classes contém as propriedades e os métodos necessários para movimenta-las.

6.5 – Interpretador de Língua de Sinais

O módulo interpretador age diretamente sobre o HV e sobre os objetos X3D, possibilitando a visualização estereoscópica de movimento em LIBRAS, a partir de eventos gerados pelo clique do mouse sobre os modelos estereoscópicos, além de permitir a visualização da descrição dos modelos virtuais em três idiomas.

O módulo interpretador é acionado via interface Principal, sendo necessário previamente, vincular um modelo X3D a uma palavra. Com a associação deve-se copiar o modelo X3D para o servidor via protocolo SCP, disponibilizando-o para todas as máquinas do *cluster*.

O *cluster* é composto por seis computadores, sendo um o servidor/cliente e cinco clientes. Este faz parte de um Sistema de Projeção, sendo composta pelo *cluster*, por um switch, por seis projetores, por três pares de lentes polarizadas, além de óculos polarizadores. A projeção é realizada em três telas refletoras posicionadas matematicamente à frente dos projetores. A Figura 32 ilustra a planta baixa do Sistema de Projeção.

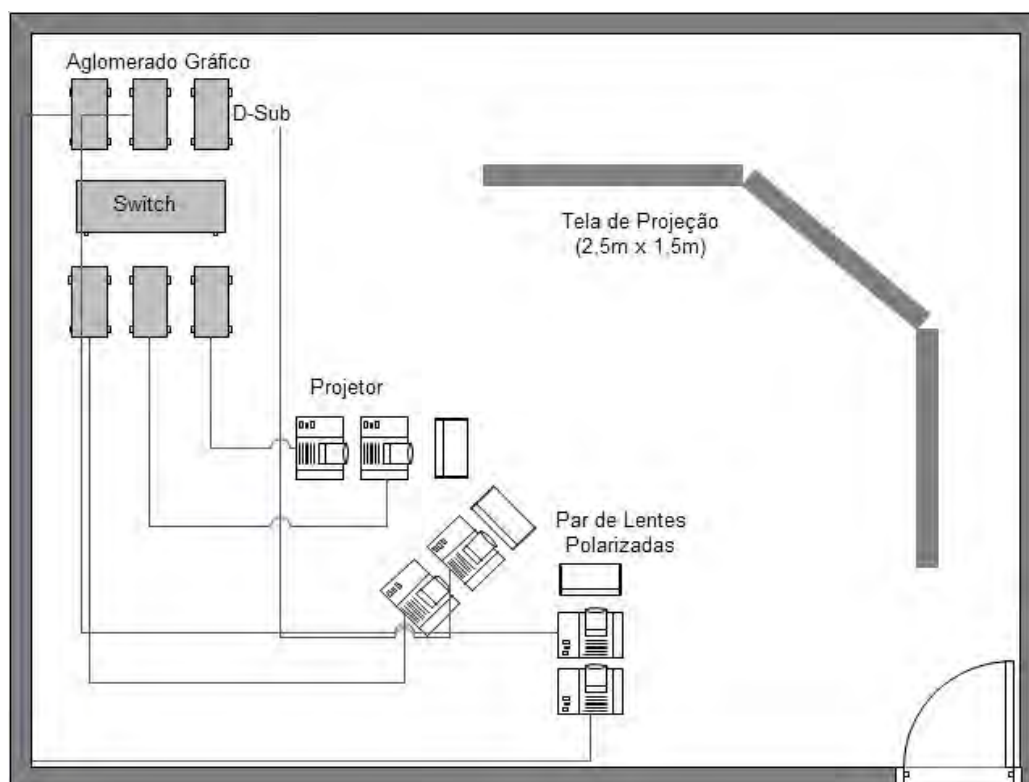


Figura 32 - Planta Baixa do Sistema de Projeção (DIAS et. al., 2010)

Após a associação da palavra ao modelo X3D, se torna possível à execução da aplicação no Sistema de Projeção. A aplicação é acionada via protocolo SSH, na qual é

disparado pela aplicação local. Assim que disparado, ele envia à aplicação das máquinas remotas, o caminho do modelo a ser renderizado juntamente com o ip da máquina servidora, fazendo com que todas as máquinas sincronizem com o Servidor Glass.

A primeira aplicação a ser executada na tela intermediária do Sistema de Projeção é a do HV. Este é projetado por dois projetores, um responsável pela imagem do olho direito e outro pela imagem do olho esquerdo.

O HV estende um JFrame e implementa os comportamentos do mouse, além de ter uma instancia de um cliente Glass responsável por garantir a sincronização dos dados.

Na elaboração do AV para o humanóide são definidos os pontos de vista para cada olho, ficando apenas a cargo do cliente Glass, definir qual instancia é para o olho direito e qual é para o olho esquerdo, conforme ilustra a Figura 33.

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

View view = new View();
PhysicalBody myBod = viewO.getPhysicalBody();
myBod.setLeftEyePosition(new Point3d(+.0045, 0.0, 0.0));
myBod.setRightEyePosition(new Point3d(-.0045, 0.0, 0.0));
view.setPhysicalBody(myBod);
view.setPhysicalEnvironment(viewO.getPhysicalEnvironment());
view.attachViewPlatform(universe.getViewingPlatform().getViewPlatform());
:

public void Estereoscopia() {
    if (glass.getG().getNodeId() == 65537) {
        canvas3D.setMonoscopicViewPolicy(View.LEFT_EYE_VIEW);
    } else if (glass.getG().getNodeId() == 65538) {
        canvas3D.setMonoscopicViewPolicy(View.RIGHT_EYE_VIEW);
    }
}
```

Figura 33 - Estereoscopia

Como visto na Figura 33, os métodos *setLeftEyePosition* e *setRightEyePosition* do próprio Java3D, garantem a divisão das imagens para os olhos correspondentes, ficando apenas a responsabilidade da sincronização. Assim, a biblioteca Glass vem para suprir esta lacuna, sendo utilizada para a sincronização das imagens. Na identificação das instancias foram utilizados os id gerados pela Glass, porém podem-se utilizar nomes disponibilizados pelo *plugin* Alias da biblioteca, conforme visto no capítulo cinco.

Para a realização das transformações, rotação, translação e escala, foi necessário modificações nas classes *MouseRotate*, *MouseTranslate* e *MouseWheelZoom*. As

modificações se tornaram essenciais, para que os novos valores das coordenadas x, y e z sejam captados e enviados a Glass, assim os dados são distribuídos e sincronizado entre as imagens pelo Servidor Glass em tempo de execução.

A Figura 34 ilustra a nova implementação da classe MouseRotate, sendo o método *getTransform()* sobrescrito para garantir a distribuição e a sincronização dos dados.

Após a captação dos novos valores, este o aplica sobre o HV e os envia como evento para o Servido Glass, garantindo a sincronização entre as imagens.

```
public class NewMouseRotate extends MouseRotate {

    public void getTransform() {
        grupo.getTransform(rotaX);
        Matrix4d mat = new Matrix4d();

        rotaX.get(mat);
        rotaX.setTranslation(new Vector3d(0.0, 0.0, 0.0));
        if (invert) {
            rotaX.mul(rotaX, transformX);
            rotaX.mul(rotaX, transformY);
        } else {
            rotaX.mul(transformX, rotaX);
            rotaX.mul(transformY, rotaX);
        }

        Vector3d translation = new Vector3d(mat.m03, mat.m13, mat.m23);
        rotaX.setTranslation(translation);
        rotaX.get(mat);
        grupo.setTransform(rotaX);
        try {
            glass.getEvento().enqueueEvent(mat);
        } catch (IOException ex) {
            Logger.getLogger(Render.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

Figura 34 - Classe “NewMouseRotate”

A Figura 35 ilustra a nova implementação da classe MouseTranslate, sendo o método *getTransform()* sobrescrito para garantir a distribuição e a sincronização dos dados. A classe MouseWheelZoom é idêntica a classe MouseTranslate, se diferenciando apenas pelo fato de tratar a coordenada z, em vez das coordenadas x e y.

Após a captação dos novos valores, este o aplica sobre o HV e os envia para o Servidor Glass como variáveis compartilhadas, ficando a cargo do Servidor a distribuição e a sincronização dos mesmos.

```

public class NewMouseTranslate extends MouseTranslate {

    @SuppressWarnings("static-access")
    public void getTransform() {
        tAvatarEscala.mul(tAvatarEscala, transformX);
        tAvatarEscala.get(vetor);
        glass.getX().setData(vetor.x);
        glass.getY().setData(vetor.y);
        try {
            glass.getX().sendUpdate();
            glass.getY().sendUpdate();
        } catch (IOException ex) {
            Logger.getLogger(NewMouseTranslate.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

```

Figura 35 - Classe “NewMouseTranslate”

A renderização do objeto é executada por uma *thread* associada a um loop de renderização, sendo acionada toda vez que uma nova alteração é percebida no AV. Desta forma foi necessário descobrir qual método Java implementa esta *thread* responsável pela renderização, para que fossem adicionados a ela, objetos Glass que garantissem a sincronização e a distribuição de dados. O método é o `prerender()`, este presente na classe `Canvas3D`. A Figura 36 ilustra a sobrescrita do método pertencente à nova classe `Canvas3D`, a `NewCanvas`.

O método implementa todas as ações necessárias para garantir a estereoscopia e a troca de informação com o modelo X3D (descrito posteriormente), além de herdar todas as suas funcionalidades padrões.

Inicialmente, é verificado se há alguma alteração na informação trocada com o modelo X3D, caso ocorra, esta é repassada ao método `Executa()` para a execução do movimento.

Posteriormente, é atualizado o evento e aplicado ao HV para a nova realocação do mesmo no AV. As variáveis que armazenam as coordenadas x, y e z, são atualizadas pelas variáveis compartilhadas da Glass para o novo posicionamento do Humanóide no AV, mantendo a sincronização das imagens, pois o que é alterado em uma aplicação é repassado à outra em tempo de execução pela biblioteca Glass.

Após o novo posicionamento o método que implementa a barreira de sincronismo, denominado `mysync()`, é invocado. Este é responsável por garantir que todas as máquinas estejam no mesmo ponto de execução, caso não, ele aguarda até que todas estejam. Logo, ele atualiza novamente todos os valores e repete todo o processo.

```

class NewCanvas extends Canvas3D {

    NewCanvas(java.awt.GraphicsConfiguration graphicsConfiguration) {
        super(graphicsConfiguration);
    }

    @Override
    public void preRender() {
        super.preRender();
        try {
            glass.getPalavra().getUpdate();
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(Humano_Virtual.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
            Logger.getLogger(Humano_Virtual.class.getName()).log(Level.SEVERE, null, ex);
        }
        Palavra = glass.getPalavra().getData();
        if (Palavra != null) {
            Executa(Palavra);
            glass.getPalavra().setData(igual);
            try {
                glass.getPalavra().sendUpdate();
            } catch (IOException ex) {
                Logger.getLogger(Render.class.getName()).log(Level.SEVERE, null, ex);
            }
        } else {
        }
        if (glass.getEvento().isEmpty() == false) {
            try {
                rotaY.set((Matrix4d) glass.getEvento().getEvent());
                grupo.setTransform(rotaY);
                glass.getEvento().clear();
            } catch (IOException ex) {
                Logger.getLogger(Render.class.getName()).log(Level.SEVERE, null, ex);
            } catch (EmptyException ex) {
                Logger.getLogger(Render.class.getName()).log(Level.SEVERE, null, ex);
            } catch (ClassNotFoundException ex) {
                Logger.getLogger(Render.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
        vetor.z = glass.getZ().getData();
        vetor.x = glass.getX().getData();
        vetor.y = glass.getY().getData();
        try {
            glass.my_sync();
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(Humano_Virtual.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
            Logger.getLogger(Humano_Virtual.class.getName()).log(Level.SEVERE, null, ex);
        }
        tAvatarEscala.setTranslation(vetor);
        carrega.setTransform(tAvatarEscala);
    }
}

```

Figura 36 - Método *prerender()*

Para a execução dos movimentos há duas classes responsáveis, a classe ExecutorBD e a classe InterpoladorBD. A classe ExecutorBD faz buscas no banco de dados por movimentos

correspondentes a palavra vinculada ao modelo X3D. Os movimentos são executados a partir do método *Executa()* da classe InterpoladorBD.

A classe InterpoladorBD permite associar um conjunto de movimentos ao Avatar e controlá-los, além de aplicar transformações em um conjunto de nós, visto que o mesmo possui métodos que agem sobre partes (nós) individuais do HV (*TransformGroup*), sofrendo as alterações apenas os membros descendentes (nós filhos) do nó atualizado. As transformações geométricas que podem ser aplicadas ao HV (*Transform3D*) é a translação, a rotação, o redimensionamento ou a composições destas.

A classe responsável por renderizar os modelos X3D é a classe Render. Está implementa os comportamentos do mouse, além de ter uma instancia de um cliente Glass responsável por garantir a sincronização dos dados.

Da mesma forma que foi implementado os pontos de vista para cada olho e os comportamentos de Mouse no AV do Humanóide para garantir a sincronização das imagens, neste também é implementado.

Há uma classe responsável por carregar o modelo X3D, a classe LoaderX3D. Esta é invocada dentro do método *setupNavigation()*, sendo atribuída a uma Scene (cena) para o posicionamento dentro do grafo de cena, conforme ilustra a Figura 37.

```
load = new loaderX3D();
Scene scene = null;
scene = load.loadFile(this.path);

if (scene != null) {

    sceneGroup = scene.getSceneGroup();

    Map<String, Shape3D> nameMap = scene.getNamedObjects();

    for (String name : nameMap.keySet()) {
        System.out.printf("Name: %s\n", name);
    }
}
```

Figura 37 - Carregar X3D

Após carregar o modelo X3D e posicioná-lo no grafo de cena, é implementado um método capaz de captar o clique no objeto, sabendo distinguir quando clicado no objeto e quando não. Este método é chamado de *mouseClicked()* e é ilustrado na Figura 38.

```

@Override
public void mouseClicked(MouseEvent e) {

    pickCanvas.setShapeLocation(e);
    PickResult result = pickCanvas.pickClosest();
    if (result == null) {
        System.out.println("Nada Clicado");
    } else {
        Shape3D s = (Shape3D) result.getNode(PickResult.SHAPE3D);
        if (s != null) {
            try {
                String pesq = "SELECT palavra FROM vincular WHERE vincular.caminho = '"
                    + pal + "'";
                Statement statement = conexao.createStatement();
                ResultSet resultado = statement.executeQuery(pesq);
                while (resultado.next()) {
                    palavra = resultado.getString("palavra");
                }
            } catch (Exception ee) {
                System.out.println(ee);
            }
            if (palavra != null) {
                glass.getPalavra().setData(palavra);
                try {
                    glass.getPalavra().sendUpdate();
                } catch (IOException ex) {
                    Logger.getLogger(Render.class.getName()).log(Level.SEVERE, null, ex);
                }
                System.out.println(palavra);
            } else {
                JOptionPane.showMessageDialog(null, "Modelo X3D nao vinculado
                    + "com a palavra correspondente");
            }
        } else {
        }
    }
}
}

```

Figura 38 - Método *mouseClicked()*

O método implementa o *pickCanvas*, este responsável por armazenar o contorno do modelo X3D em um objeto do tipo *Shape3D*, assim se o objeto *Shape3D* não estiver *null* (vazio), conterá o objeto respectivo.

Após a confirmação da existência do objeto é realizado uma busca no banco de dados, com o nome do respectivo, para encontrar a palavra correspondente a ele. Com o retorno da palavra do banco, esta é enviada ao Servidor Glass para que seja encaminhada ao HV, ocorrendo à realização do movimento respectivo.

Há a necessidade da sobrescrita do método responsável por gerenciar a renderização do AV. O método *prerender()* é o responsável pelo ato e foi mudado da mesma forma que o do HV, porém com algumas alterações, para garantir o sincronismo dos dados relevantes a ele, conforme ilustra a Figura 39.

```

@Override
public void preRender() {
    super.preRender();
    if (glass.getEvent().isEmpty() == false) {
        try {
            rotaY.set((Matrix4d) glass.getEvent().getEvent());
            examineGroup.setTransform(rotaY);
            glass.getEvent().clear();

        } catch (IOException ex) {
            Logger.getLogger(Render.class.getName()).log(Level.SEVERE, null, ex);
        } catch (EmptyException ex) {
            Logger.getLogger(Render.class.getName()).log(Level.SEVERE, null, ex);
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(Render.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    trans.z = glass.getTransz().getData();
    trans.x = glass.getTransx().getData();
    trans.y = glass.getTransy().getData();
    viewTrans.setTranslation(trans);
    vpTransGroup.setTransform(viewTrans);
    try {
        glass.getDataLock().sync();
        glass.getTransx().getUpdate();
        glass.getTransy().getUpdate();
        glass.getTransz().getUpdate();
    } catch (IOException ex) {
        Logger.getLogger(Render.class.getName()).log(Level.SEVERE, null, ex);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(Render.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Figura 39 - Método *prerender()* X3D

O método implementa todas as ações necessárias para garantir a estereoscopia e a troca de informação com o HV (descrito anteriormente), além de herdar todas as suas funcionalidades padrões.

Inicialmente é atualizado o evento e aplicado ao modelo X3D para o novo posicionamento no AV. As variáveis que armazenam as coordenadas x, y e z, também são atualizadas por meio das variáveis compartilhadas da Glass e redimensionam o modelo X3D para sua nova posição no AV.

Após o novo posicionamento é invocado a barreira responsável pelo sincronismo dos dados. Esta garante que todas as máquinas estão no mesmo ponto de execução, caso não

estejam, esta aguarda até que todas estejam. Logo, atualizam-se todos os valores e um novo processo é repetido.

Uma consideração importante a ser ressaltada, é fato de existirem dois clientes Glass estruturados diferentemente, um para o HV e outro para os modelos X3D. Isto se fez necessário, pois se houver apenas um cliente Glass de um único tipo, as aplicações se comportariam da mesma forma, ou seja, onde o HV fosse posicionado, o modelo X3D também seria. Por este fato, foi necessário estruturar dois clientes diferentes para garantir a dependência do posicionamento de ambos em seus AVs.

A classe responsável por descrever os modelos X3D associados previamente à palavra cadastradas no banco de dados e renderizar uma prévia do objeto em um “JInternalFrame” na mesma interface, é denominada “Definicao”.

Esta classe permite escolher uma palavra em um menu disponível na mesma interface, oferecendo a visualização do objeto associado a mesma, além de oferecer possíveis descrições do objeto. Tais descrições estão disponíveis em três línguas, a portuguesa, a inglesa e a espanhola, além de permitir as alterações de seus significados e grava-los no banco de dados.

A classe responsável pela renderização prévia do objeto X3D é chamada Render2. Está estendendo a classe “JInternalFrame”, permite carregar o objeto na mesma interface principal. A classe é simples, contendo apenas componentes comuns para a construção de um AV, como por exemplo, iluminação, comportamentos de mouse e componentes de transformação, além de possuir um loader X3D para renderizar o modelo X3D vinculando-o ao nó *Scene* para inseri-lo ao grafo de cena.

Após a pré-visualização do modelo X3D é possível visualizá-lo estereoscopicamente no Sistema de Projeção, junto ao seu movimento em LIBRAS através do HV. Desta forma, comandos SSH são enviados ao Servidor para o encerramento das aplicações correntes e acionado a execução da nova aplicação, conforme ilustra a Figura 40. Vale ressaltar que este procedimento pode ser repetido quantas vezes necessário.

```
@Override
public void run() {
    Basic ssh = new Basic();
    ssh.Ssh("lstr3", "123456", "192.168.1.3",
           "sh /projetos/Messenger/dist/exec_REMOTE_finish_estereoscopia.sh ");
    ssh.Ssh("lstr3", "123456", "192.168.1.3",
           "sh /projetos/Messenger/dist/exec_REMOTE_estereoscopia.sh " + caminho_remoto
           + " " + ip + " " + nome);
}
```

Figura 40 - Comandos SSH

O método `run()` instancia o objeto `Basic`, esta uma classe que possui os métodos responsáveis por acionar comando SSH e SCP, sendo disponibilizados através de uma API para Java, denominada “`ganymed-ssh2.jar`”.

6.6 – Editor da Língua de Sinais

O módulo editor é um recurso oferecido pelo sistema gerador de um dicionário temático, agindo diretamente sobre o HV, para possibilitar a edição de movimentos através de dispositivos não convencionais. Este é acionado via interface principal, sendo necessário previamente, escolher a categoria juntamente com a sua palavra correspondente. Ressaltando que a módulo editor só funciona na aplicação local.

A interface Principal oferece em um momento inicial, a escolha de se utilizar o dispositivo Wiimote ou a Luva ou ambos. Desta forma fica a critério do usuário utilizar o que lhe for mais conveniente.

Ambos os dispositivos utilizam classes em comuns, oferecendo funcionalidades de execução de movimentos, gravação de movimentos e armazenamento da granulidade mínima de cada movimento gerado. Estas classes estão disponíveis no Apêndice D, sendo detalhadas para o entendimento dos processos.

Com a especificação dos processos de execução e gravação de movimentos, é possível desenvolver classes que recebam (*Listener*) os dados dos dispositivos, transformando-os em valores aceitáveis através de cálculos matemáticos.

Há dois dispositivos não convencionais responsáveis pelo rastreamento corporal, um denominado controlador Wiimote e outro denominado Luva P5-Glove. O controlador Wiimote, através de seu acelerômetro (rastreamento mecânico) e de seu infravermelho (rastreamento ótico), é capaz de capturar os movimentos da coordenada x, y e z dos membros superiores humanos, sendo x e y captado por meio do acelerômetro e o z por meio do infravermelho. O Apêndice E apresenta mais sucintamente o controlador Wiimote, especificando fundamentos teóricos, possíveis projetos e a maneira da captura dos movimentos.

A Luva de dados P5-Glove, através de sensores (rastreamento mecânico) e de *LEDs* presentes no dispositivo (rastreamento ótico), permite a captura dos movimentos produzidos pela mão e por seus dedos, respectivamente. O Apêndice F apresenta detalhadamente os conceitos da luva e a maneira de utilização para a captura dos dados do membro referido.

Enfim, o módulo editor oferece recursos de edição de movimentos para o sistema gerador de um dicionário temático, por meio de dispositivos não convencionais que capturam os movimentos dos membros superiores do corpo humano, juntamente com os movimentos da mão direita e seus respectivos dedos, repassando-os ao HV em tempo de execução, para a visualização estereoscópica através da multiprojeção.

Para maiores esclarecimentos da utilização dos dispositivos e geração de movimentos, os Apêndices D, E e F estão disponíveis.

6.7 – Base de Dados

Na maioria das ferramentas há um banco de dados responsável por recuperar e/ou armazenar informações relevantes do sistema, este sendo um modelo relacional, orientado a objeto, hierárquico ou em rede. Nesta ferramenta não é diferente, existe um modelo de banco de dados relacional servindo como base para o interpretador de sentença e para o editor de movimentos de palavras.

Atualmente o banco de dados MY-SQL está sendo utilizado, devido ser independente de plataforma, totalmente gratuito e atender aos requisitos necessários para a execução da aplicação. Este por utilizar a linguagem *Structured Query Language* (SQL) para consultas e gravações, torna-se possível a migração sem dificuldade para qualquer outro banco relacional que utilize tal estrutura. A Figura 41 ilustra o diagrama entidade relacionamento do MY-SQL.

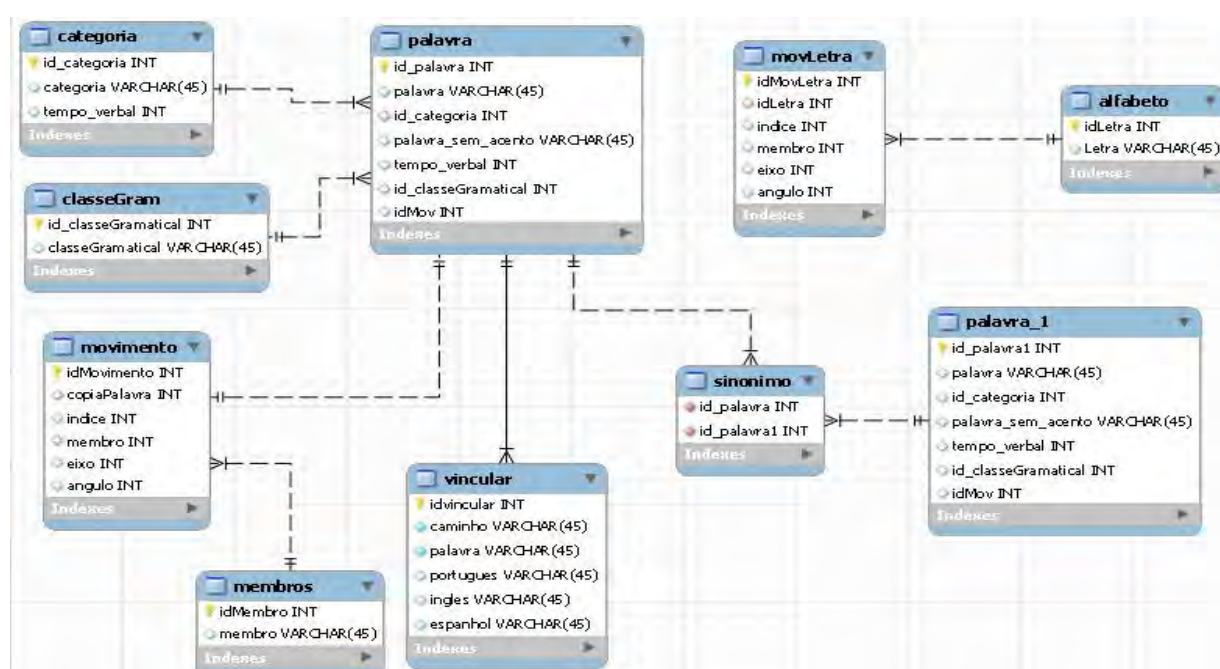


Figura 41 - Base de Dados

6.8 – Entendendo a Interface do Sistema

Com a definição das funcionalidades do sistema, de sua estruturação e da descrição dos módulos, é necessário entender a interface gráfica para a utilização. A Figura 42 ilustra a interface inicial do sistema.



Figura 42 - Interface inicial do sistema

Na interface Principal é possível visualizar a interpretação de palavras para sinais em LIBRAS por meio do HV presente na mesma. Na escolha da palavra, há controles deslizantes que permitem ao usuário selecionar a categoria da palavra juntamente com a sua correspondente. Esta já se encontra armazenada em um banco de dados, tirando a possibilidade de selecionar palavras que ainda não estejam presentes no mesmo.

Em um primeiro momento, há uma palavra padrão a ser interpretada ou editada, porém através do painel “Tradução” há a possibilidade de trocá-la. A Tabela 1 descreve cada função, mapeando os botões que possibilitam a utilização da ferramenta.

Partindo das opções descritas acima, o sistema deve ser inicializado com a escolha da categoria e palavra no painel “Tradução”. Desta forma, o interpretador pode realizar buscas no banco de dados e recuperar instruções para que sejam interpretadas pelo HV, ou seja, um sistema gerador de um dicionário temático de palavras com representação em LIBRAS.

Os movimentos gerados podem ser visualizados de diversas maneiras, pois o AV possibilita a rotação, a translação e o zoom do HV.

Com a busca do interpretador de palavras por instruções no banco de dados, há a necessidade de se criar e armazenar novas instruções de palavras (movimentos). Assim, a ferramenta disponibiliza os módulos editores que se tornam visíveis quando selecionado as opções 5 e 6, conforme mostrado na Tabela 4. A Figura 43 ilustra uma visão geral da interface dos módulos editores.

Tabela 4 – Descrição da Interface Inicial do Sistema

Item	Descrição
1 – Tradução	Possibilita a escolha de uma categoria de palavra com sua correspondente para posterior execução, edição ou visualização pelo Sistema de Projeção.
2 – Universo para Representação de LIBRAS	AV contendo um HV responsável pela execução de movimentos em LIBRAS
3 – Módulo Interpretador Glass	Aciona comandos SSH para a visualização estereoscópica do HV, do objeto X3D junto a sua descrição no Sistema de Projeção.
4 – Fecha as aplicações Glass	Encerra todas as aplicações correntes do Sistema de Projeção
5 - Editor de movimento com controladores Wiimote	Habilita a edição dos movimentos da palavra selecionada no painel Tradução pelos Wiimotes.
6 - Editor de movimentos com a luva P5 - Glove	Habilita a edição dos movimentos da palavra selecionada no painel Tradução pela luva.
7 – Vincular	Vincula a palavra corrente com um Objeto X3D.
8 – Habilitar Navegação Wiimote	Permite navegar com o controlador no Sistema de Projeção, emulando os comportamentos de mouse e teclado.

O módulo editor sofreu mudanças bruscas. No modelo do Fabrício Baptista (BAPTISTA, 2007), o módulo de edição possuía controles não imersivos, como por exemplo, controles deslizantes (*scroll bar*) e grupo de botões de opção, responsáveis por mover os membros selecionados. Neste atual modelo, com o intuito de aumentar a imersão e a interação do usuário com o sistema, os controles deslizantes foram substituídos por controladores Wiimote. Neste módulo também foi adicionado uma luva responsável por representar a mão

do usuário virtualmente no sistema, sendo capaz de captar e reproduzir virtualmente os movimentos produzidos pelo mesmo no AV.

Os itens descritos na Tabela 5 possibilitam que o usuário tenha o controle da sequência de instruções necessárias para a geração de movimentos, levando em consideração que a palavra já foi previamente escolhida na interface inicial.

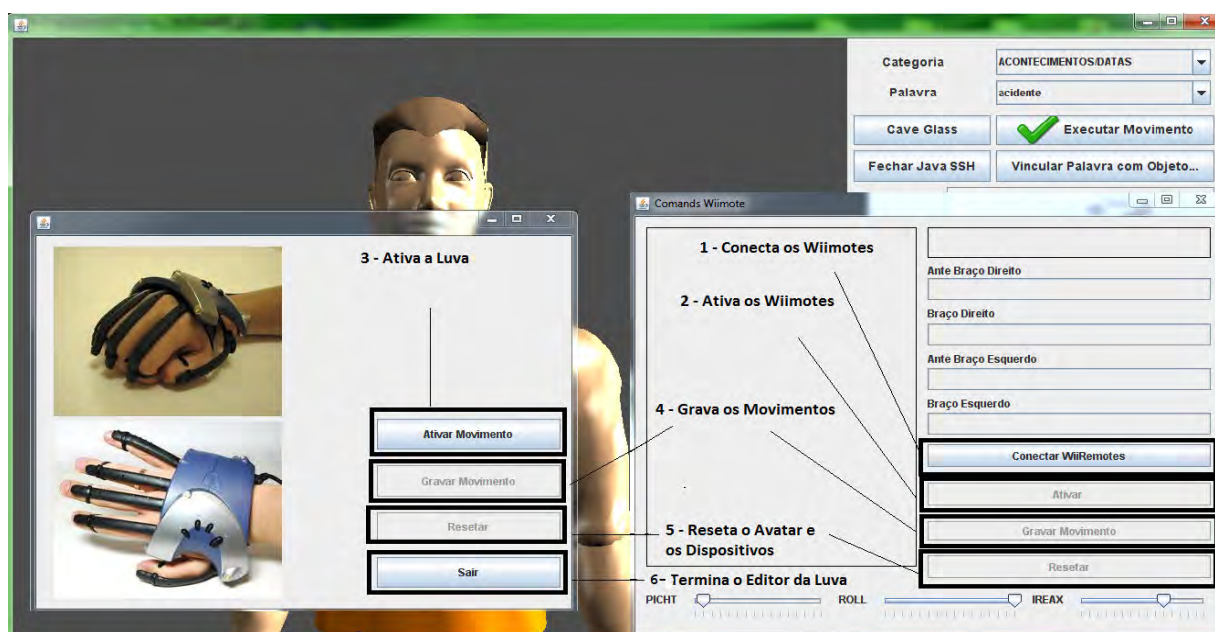


Figura 43 - Interface dos Módulos Editores

Tabela 5 – Descrição da Interface de Editoração

Item	Descrição
1 – Conectar os Wiimotes	Conectam até 4 Wiimotes
2 – Ativar os Wiimotes	Ativam os Wiimotes para a representação de movimentos
3 – Ativa a Luva	Ativa os movimentos da luva juntamente com os dedos
4 – Gravar os Movimentos	Ambos os módulos gravam os movimentos gerados pelos Wiimotes e pela luva
5 – Reseta o Avatar e os Dispositivos	Ambos os módulos resetam o Avatar e seu respectivo dispositivo
6 – Termina o Editor da Luva	Fecha a parte de edição da Luva

Primeiramente é necessário conectar os Wiimotes, sendo possível conectar até quatro controles concorrentemente. Após a conexão de cada, é mostrada a carga da bateria do respectivo junto a sua imagem à esquerda, conforme ilustrado na Figura 44. Lembrando que os controladores Wiimotes conectam no computador via *bluetooth*. Com a conexão estabelecida é necessário ativa-lo, e para isso, basta pressionar o botão “A” do controlador e logo o botão “Ativar” da interface gráfica correspondente. Com o controlador já ativado o HV responde aos eventos gerados pelo mesmo, tornando possível a gravação dos dados no banco.

Quando acionado o botão “Gravar” no painel de interface do Wiimote, todos os estímulos recebidos pelo HV do controlador são gravados no banco de dados até o momento em que o botão foi acionado. Caso o movimento gerado não seja o desejado, existe a possibilidade de acionar o botão “Resetar”, cuja função é apagar todos os movimentos gerados e posicionar o HV em sua posição inicial. A Figura 44 ilustra a interface gráfica do Editor dos controladores Wiimotes.

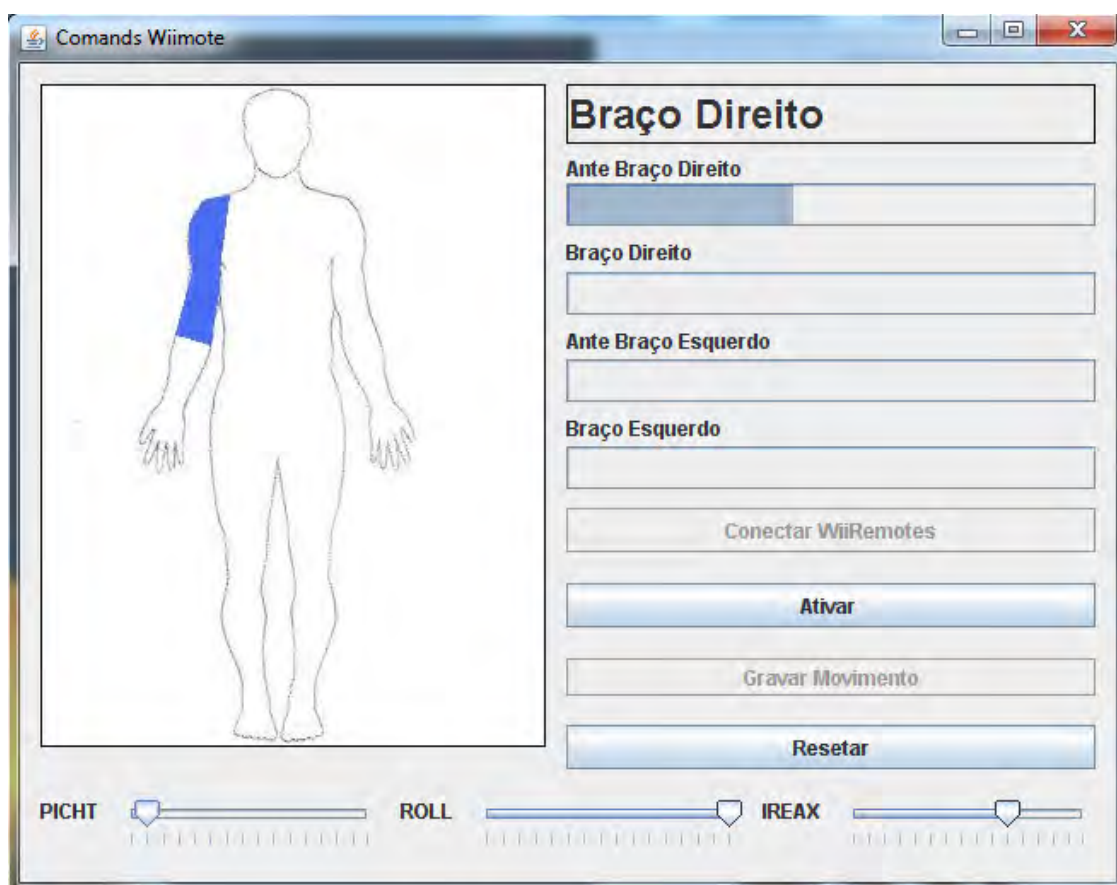


Figura 44 - Interface gráfica do Editor Wiimote

A interface permite também, em tempo de execução, mudar a sensibilidade das coordenadas x, y e z dos controladores Wiimotes, sendo chamadas respectivamente de

“PITCH”, “ROLL” e “IREAX”. Cada coordenada é responsável por definir um limite mínimo para cada evento gerado pelos Wiimotes, ou seja, se o valor do evento gerado for maior que o limite definido pela coordenada correspondente, o mesmo é processado e enviado ao HV para execução, caso contrário, este é desconsiderado.

Na interface da luva não há a possibilidade de conectá-la, visto que existe um driver a ser instalado previamente, conforme descrito no Apêndice F. Após a luva ser calibrada por meio do aplicativo instalado no sistema operacional Windows, já é possível utilizá-la na aplicação.

Inicialmente é necessário ativá-la através do botão “Ativar” presente no painel de interface da mesma. Após este evento, o HV já responde aos estímulos enviados da luva para a aplicação, sendo possível a gravação dos mesmos. Quando acionado o botão “Gravar”, todos os movimentos gerados desde a ativação da luva são gravados em um banco de dados, havendo a possibilidade de descartá-los caso o botão “Resetar” for pressionado. Tal botão apaga todos os movimentos gerados pela luva e posiciona o HV em sua posição inicial. A Figura 45 ilustra a interface de edição da luva.

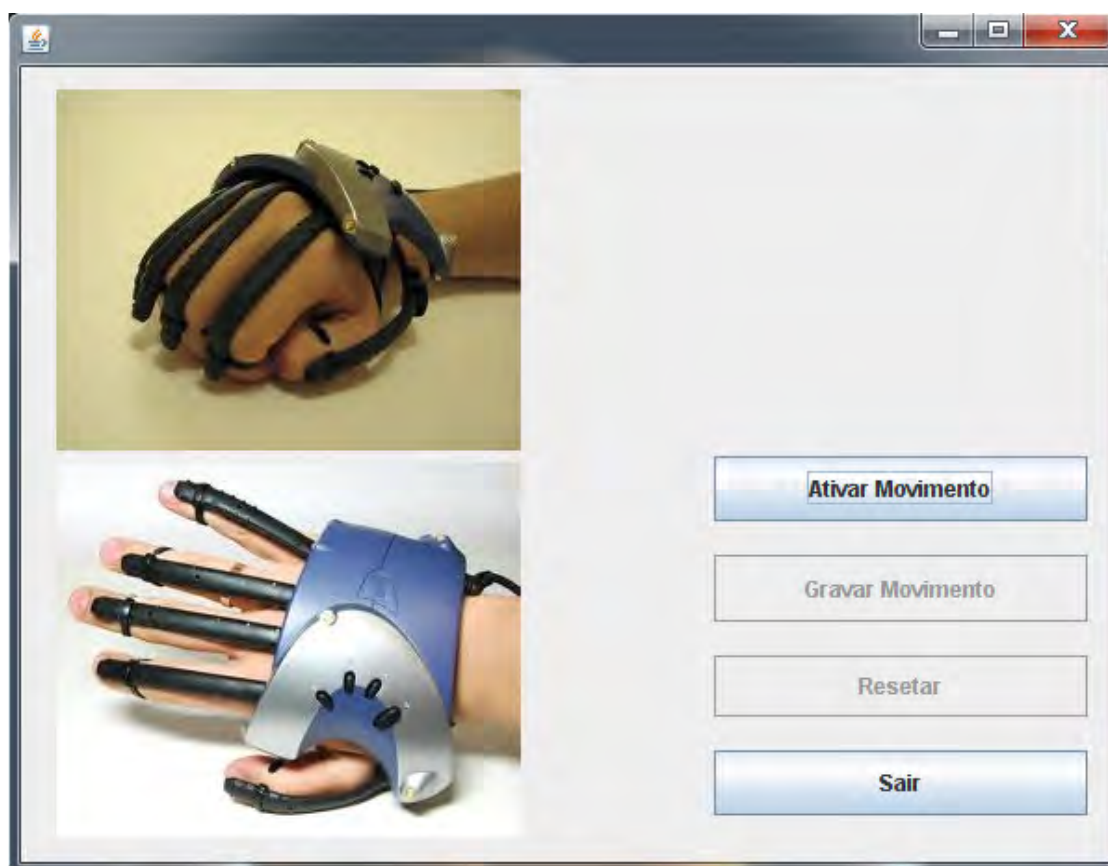


Figura 45 - Interface gráfica do Editor da Luva

Para a visualização estereoscópica dos movimentos através do HV renderizado em tamanho real, é necessário acionar o botão três da interface principal, conforme apresentado na Tabela 1. Assim que pressionado, comandos SSH dispararam a execução da aplicação no Sistema de Projeção. Dentre as aplicações estão:

- Um Servidor Glass: para distribuir os dados e sincronizar as imagens;
- Um HV estereoscópico: para interpretar modelos em LIBRAS;
- Um Modelo X3D estereoscópico: para a representação de palavras; e
- Uma Interface de descrição: para apresentar as palavras vinculadas a modelos virtuais em um menu, junto a suas descrição nos idiomas: português, inglês e espanhol.

A Figura 46 ilustra a interface da descrição de palavras.

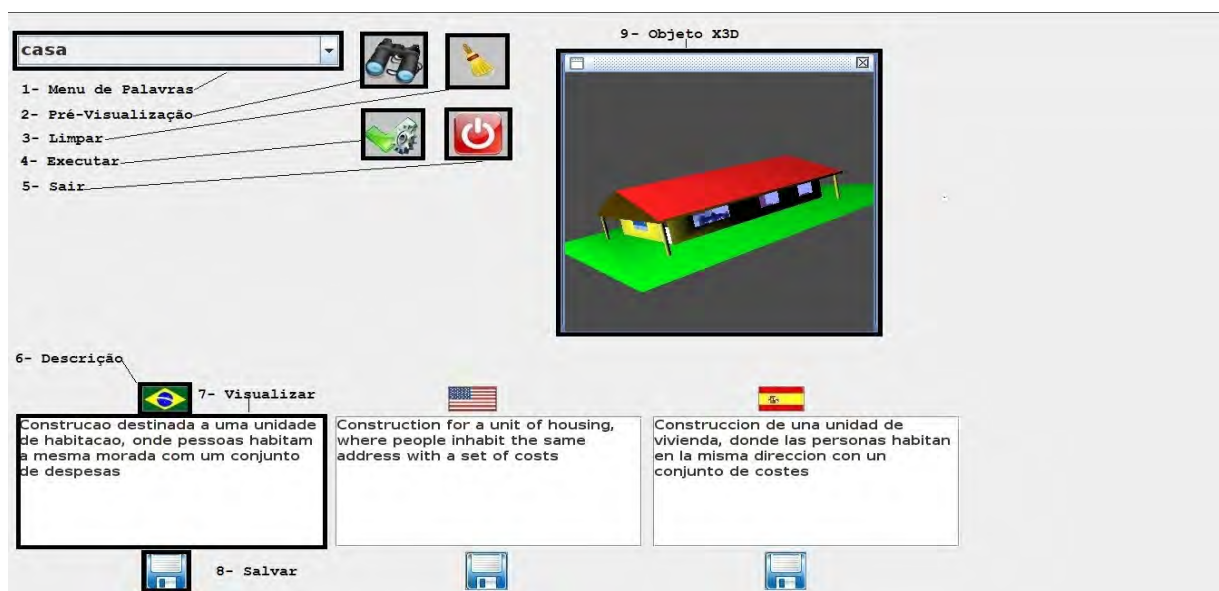


Figura 46 - Interface Descrição

Como ilustrado na Figura 46, a interface oferece oito aspectos importantes a serem realçados. Para um melhor entendimento, a Tabela 6 mostra o significado de cada aspecto.

Tabela 6 – Descrição dos aspectos da Interface de Descrição

1 – Menu de Palavras	Permite escolher uma palavra pré-vinculada a um objeto X3D
2 – Pré-visualização	Permite acionar a pré-visualização do objeto X3D vinculado á palavra
3 – Limpar	Apaga a pré-visualização junto às descrições
4 – Executar	Dispara comandos SSH para o servidor para o encerramento das aplicações correntes do Sistema de Projeção (HV e objeto X3D), iniciando o Servidor Glass, o HV e o novo objeto X3D
5 – Sair	Fecha a interface descrição
6 – Descrição	Permite acionar e encerrar a descrição da palavra corrente do menu de palavras, sendo na língua exposta pela bandeira.
7 – Visualizar	Visualiza e desvisualiza a descrição das palavras na língua referida
8 – Salvar	Grava a nova informação na língua correspondente da respectiva palavra
9 – Objeto X3D	AV responsável pela pré-visualização do modelo X3D

A interface do modelo X3D é ilustrada na Figura 47, sendo ilustrada sem a aplicação de estereoscopia.

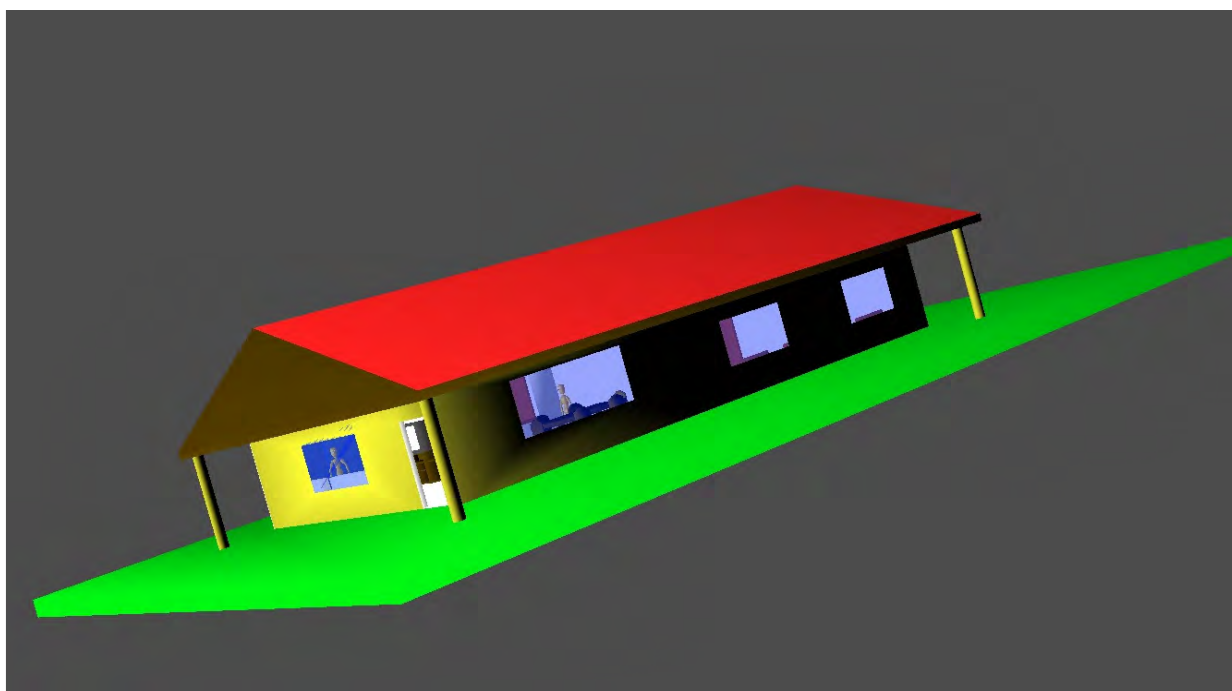


Figura 47 - Objeto X3D

Neste AV onde é renderizado o modelo X3D, é permitido aplicar as transformações sobre o objeto, sendo estas: a translação, a rotação e a escala. Além de ser possível aplicar todas estas transformações, o AV reconhece o clique do mouse sobre objeto. Assim que o clique ocorrer, a Glass envia um evento ao HV para a execução do movimento em LIBRAS do referido. A Figura 48 ilustra o HV.



Figura 48 – Humano Virtual

Este ambiente permite as mesmas transformações aplicadas sobre o modelo X3D, além de movimentos em LIBRAS realizados pelos membros superiores do HV.

No decorrer do trabalho é relatado muita vezes sobre a aplicação de transformações nos objetos dos AVs, estas são oferecidas através do dispositivo Wiimote. O controlador além de possibilitar a edição de movimentos, permite ao usuário aplicar estas transformações de maneira intuitiva, pois possui botões e acelerômetros que emulam as funcionalidades do teclado e mouse.

A interface do controle utiliza tecnologia sem fio, permitindo ao usuário mobilidade na navegação do sistema. Este realiza conexão com os computadores via interface *bluetooth*.

A classe responsável por fazer o mapeamento do controle para comandos de mouse e teclado é denominada Robot. A Figura 49 ilustra a maneira da estrutura da conexão do controlador.

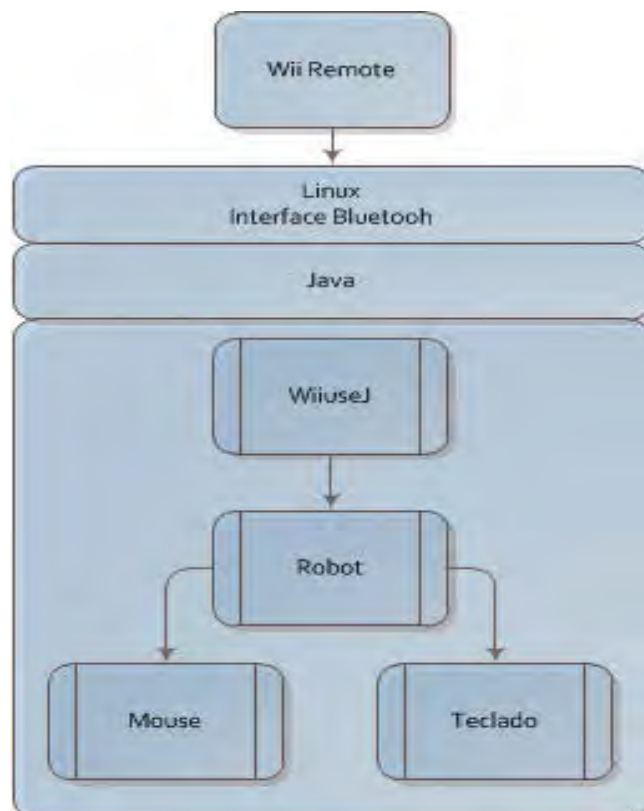


Figura 49 - Estrutura de conexão do dispositivo Wiimote

Para o encerramento de todas as aplicações do Sistema de Projeção, basta pressionar o botão quatro da interface principal, conforme mostrado na Tabela 1. Assim, todas as aplicações encerrarão e a execução retornará à interface principal da máquina local.

6.9 – Considerações Finais

O capítulo seis descreveu todo o processo de desenvolvimento do sistema, dando detalhes e explicações do sistema gerador de um dicionário temático e dos recursos oferecido, o editor de movimento, além de descrever toda a maneira de modularização do sistema com suas respectivas interfaces. Foram apresentados e estruturados os grafos de cena dos AVs juntamente com o HV e os modelos X3D, além da representação das funcionalidades e relacionamentos do sistema através de diagramas UML.

O próximo Capítulo apresenta os resultados dos testes realizados no sistema, apresentando Figuras para melhor ilustra-los.

Capítulo 7 - Resultados

Este capítulo demonstra simplificadaamente a execução de todos os módulos do sistema. Inicialmente a definição da configuração de hardware e de software utilizados nessas simulações torna-se necessária, visto que são utilizados dois sistemas operacionais e um *cluster* para a interpretação em LIBRAS.

No processo de desenvolvimento do módulo editor e na simulação dos movimentos gerados pelo mesmo, foi utilizado um notebook com processador i5 2.4GHZ, 4GB de memória RAM DDR3, um HD de 640GB SATA II e uma placa de vídeo integrada *Intel HD Graphics*. O sistema operacional utilizado neste segmento foi o Windows 7 de 64 bits, devido a luva P5-Glove ter drivers que podem ser melhores manuseados no referido.

No processo de desenvolvimento do módulo interpretador e na simulação dos movimentos realizados pelo mesmo, foi utilizado um *cluster* contendo seis computadores, um servidor/cliente e cinco clientes. Nos computadores foi utilizada a seguinte configuração: um processador i7 2.8GHZ, 8GB de memória RAM DDR3, um HD de 500GB SATA II e uma placa de vídeo NVIDIA FX QUADRO 1800 de 768MB DDR3. O sistema operacional utilizado foi o Kubuntu 10.10, devido este ser o instalado no *cluster* utilizado.

Nas sessões subsequentes são apresentado simplificadaamente a execução de ambos os módulos.

7.1 – Testes do Editor de Movimentos

O módulo editor, sendo responsável pela geração de movimentos por meio de dispositivos não convencionais, foi utilizado em algumas palavras para a simulação e validação da ferramenta.

Para iniciar a geração de movimentos, inicialmente é necessário escolher a palavra a ser traduzida para LIBRAS na interface principal, logo, conecta-se e ativa-se os dispositivos não convencionais através de suas interfaces respectivas, conforme descrito na sessão 6.7. Com a prontidão dos dispositivos, basta realizar os movimentos em Língua de Sinais da referida palavra e gravá-los no banco de dados. A Figura 50, 51 e 52 ilustram exemplos de movimentos com a utilização dos dispositivos.

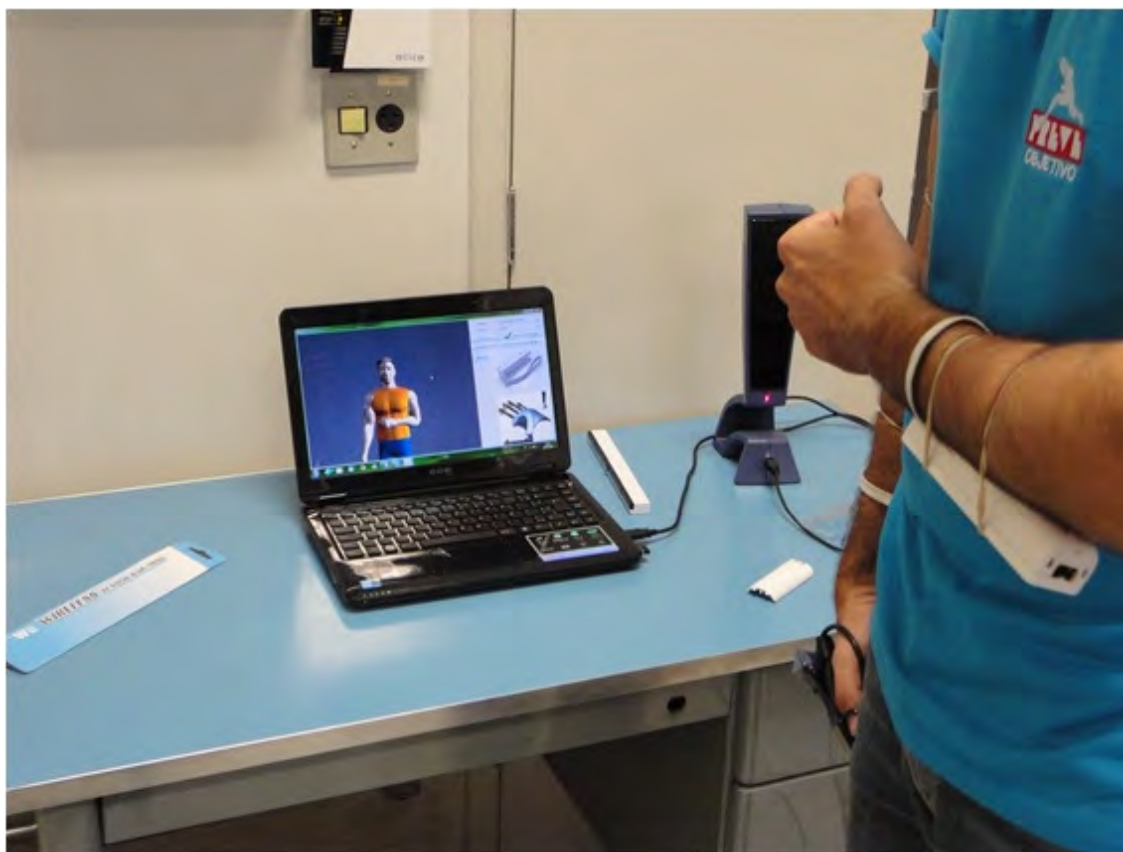


Figura 50 - Movimentos do Wiimote na Coordenada Z

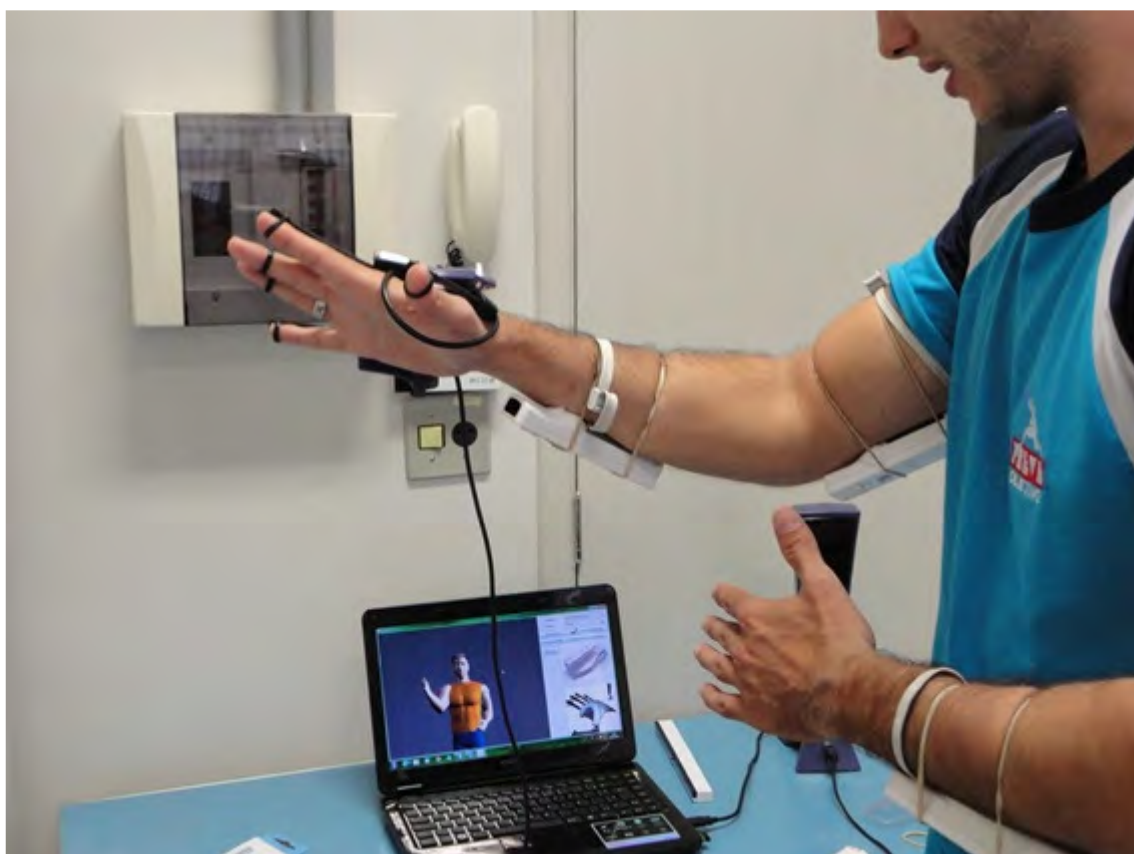


Figura 51 - Movimento dos Wiimotes na Coordenada X

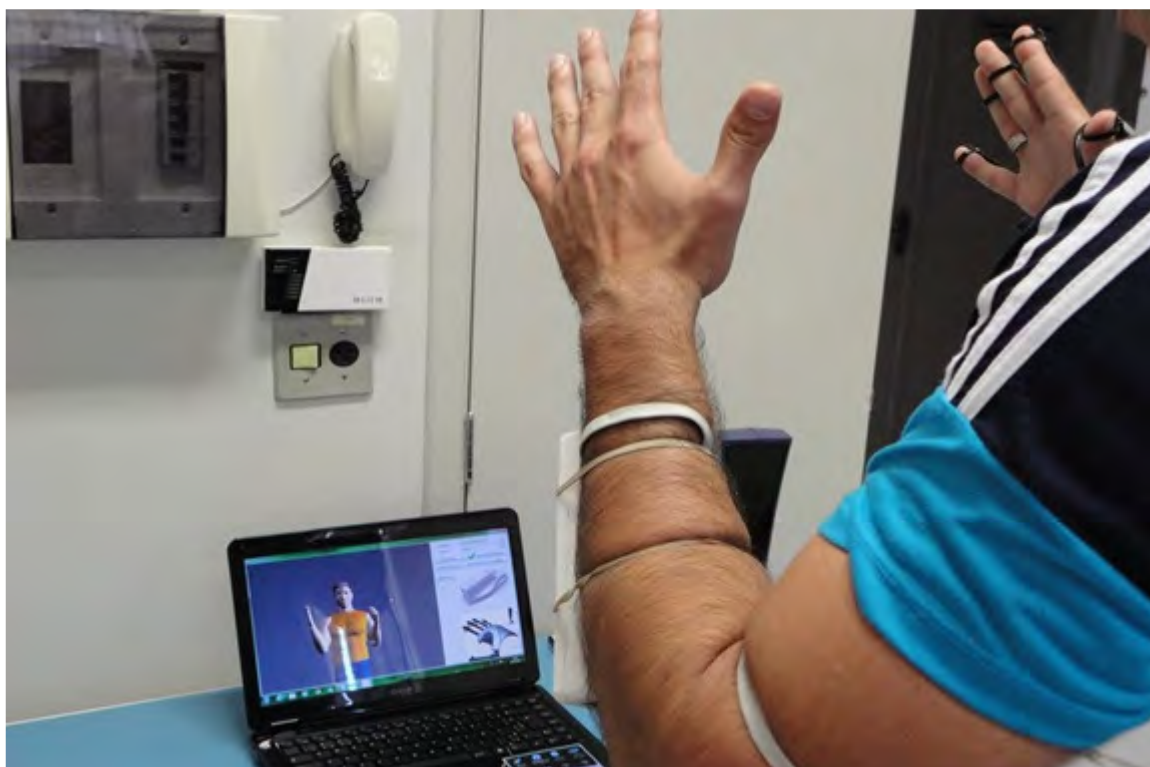


Figura 52 - Movimento dos Wiimotes nos dois Membros

Conforme apresentado nas Figuras 50, 51 e 52, com a utilização de quatro controladores é possível garantir os movimentos dos membros superiores do corpo humano, sendo os braços e antebraços direito e esquerdo.

A seguir é apresentada a Figura 53, ilustrando alguns exemplos de movimentos executados através da Luva P5-Glove.

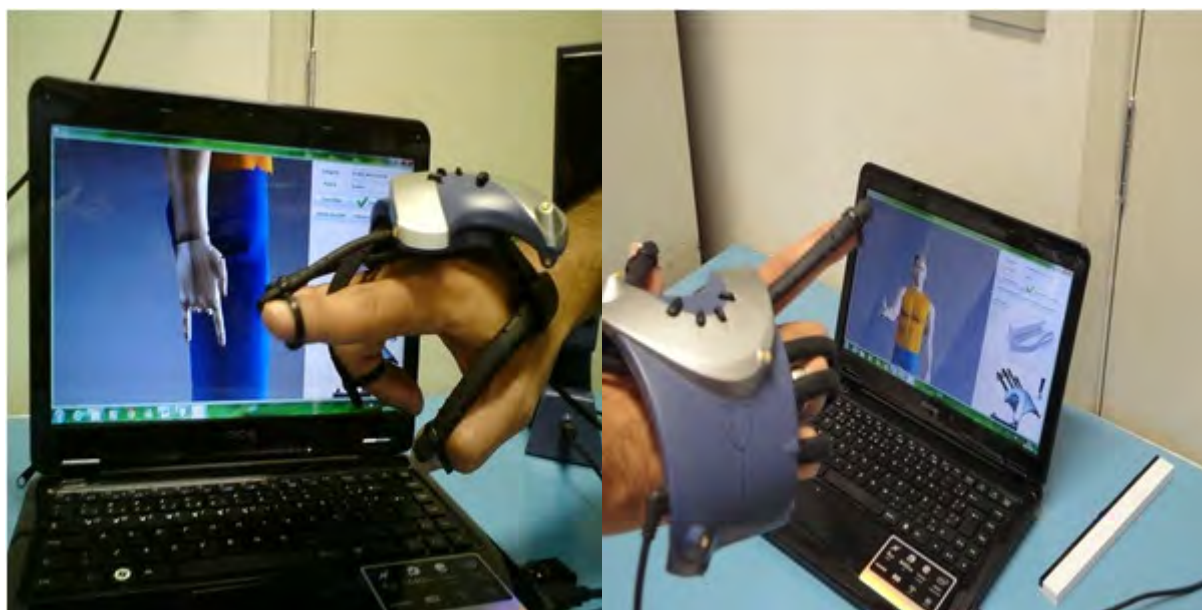


Figura 53 - Movimentos da Luva P5-Glove

Como observado na Figura 53, com a utilização da Luva P5-Glove é possível garantir os movimentos no eixo x de todos os dedos da mão direita, sendo esta a mais utilizada na representação de movimentos em LIBRAS, oferecendo um maior refinamento na representação de movimentos.

Com a possibilidade da edição de novas palavras para a representação em Língua de Sinais, através de dispositivos não convencionais, o sistema gerador de um dicionário temático se torna bastante viável na representação de sinais. A sessão 7.2 apresenta os testes obtidos pelo módulo interpretador.

7.2 – Testes na Representação de Sinais

O módulo interpretador, sendo responsável pela representação de movimentos em Língua de Sinais e por disponibiliza-los visualmente ao usuário, foi utilizado em algumas palavras para a simulação e validação da ferramenta. A Língua de Sinais utilizada nos testes foi LIBRAS.

Com a escolha do primeiro modelo virtual a ser visualizado estereoscopicamente pela ferramenta, o Sistema de Projeção apresenta-se conforme ilustra a Figura 54.



Figura 54 - Módulo Interpretador no Sistema de Projeção

De acordo com a Figura 54, o modelo a ser visualizado representa a palavra soldado. Para uma melhor visualização do modelo 3D e dos movimentos em LIBRAS, é permitida a rotação, a translação e a escala de ambos. A Figura 55 ilustra a representação em LIBRAS do modelo soldado.

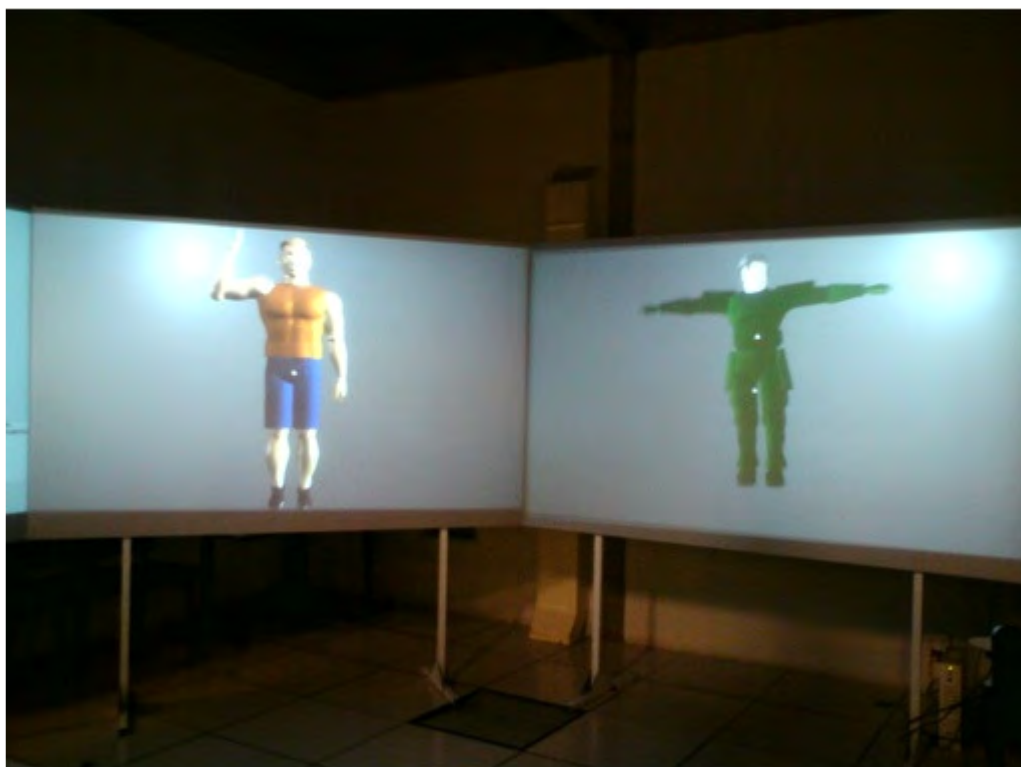


Figura 55 - Representação da Palavra Soldado

Para a escolha e visualização de uma nova palavra, a Figura 56 ilustra a interface responsável.



Figura 56 - Descrição de Palavras

Nesta interface é possível a escolha de uma nova palavra, com sua pré-visualização e seus significados. No exemplo da Figura 56, é escolhida a palavra casa, com sua pré-visualização disponível no canto superior direito e com suas descrições em seus respectivos idiomas na parte inferior da interface.

Para a visualização estereoscópica e representação em LIBRAS, é necessário enviá-los a Glass através do botão quatro, conforme ilustrado na Figura 46. A Figura 57 ilustra Sistema de Projeção com o novo modelo 3D e com sua representação em LIBRAS. Lembrando que as imagens não estão nítidas por causa do efeito de estereoscopia, faltando à filtragem dos dados através dos óculos polarizadores.



Figura 57 - Representação de Casa no Sistema de Projeção

Para a visualização e descrição de outro modelo 3D, basta realizar os mesmos passos que foram descritos anteriormente para o modelo casa. A Figura 58 ilustra a representação da palavra árvore com seu respectivo modelo 3D.



Figura 58 - Representação da Palavra Árvore

7.3 – Considerações Finais

Este capítulo apresentou alguns testes obtidos com o desenvolvimento do trabalho, tanto para a representação visual de movimentos por meio do sistema gerador de um dicionário temático, quanto para os recursos de edição de movimentos oferecidos.

A seguir são apresentadas as conclusões do trabalho juntamente com sugestões de melhorias e aperfeiçoamento do sistema, como propostas de trabalhos futuros.

Capítulo 8 - Conclusões

O levantamento bibliográfico realizado nos primeiros tópicos deste trabalho foi essencial para o desenvolvimento desta aplicação, visto que foram apresentadas as justificativas, estruturas físicas, linguísticas e tecnológicas que proporcionaram o desenvolvimento de um sistema gerador de apoio de um dicionário temático visual-gestual voltado a Língua de Sinais.

O escopo do trabalho foi utilizar técnicas que garantissem fácil acesso à ferramenta pelas comunidades interessadas, disponibilizando e integrando maneiras de interpretação e edição de movimentos em Língua de Sinais em um único ambiente.

Para garantir estes objetivos, houve a necessidade da integração de bibliotecas de desenvolvimento com recursos de RV, mostrando a possibilidade de uma combinação robusta por trazer interoperabilidade e benefícios, a fim de auxiliarem as comunidades envolvidas.

Para a interpretação dos sinais foi utilizado três telas, onde em uma é multiprojetado estereoscopicamente um HV responsável em representar em Língua de Sinais significados de modelos virtuais, onde são multiprojetado estereoscopicamente por outra tela. A fim de tornar o sistema disponível a diferente Língua de Sinais, a terceira tela apresenta descrições sobre objetos em três idiomas.

Devido à necessidade de ter um grande volume de modelos virtuais vinculados a palavras para a representação em Língua de Sinais, optou-se pela utilização de modelos baseados na linguagem X3D, sendo esta amplamente difundida na Internet, visto que, o sistema não permite a modelagem, mas sim, a geração do grafo de cena do respectivo modelo virtual.

Na edição dos sinais são utilizados dois dispositivos não convencionais, o controlador Wiimote e a Luva P5-Glove. O controlador é responsável por captar os movimentos dos braços e antebraços esquerdo e direito, já a luva capta os movimentos da mão direita e de seus dedos. As tarefas de edição, realizadas através destes dispositivos, podem aumentar o nível de imersão e interação do usuário com o sistema, sendo estes, responsáveis por simular os movimentos da Língua de Sinais, ficando apenas restrito a algumas limitações dos dispositivos (descrito no Capítulo 6).

Com o intuito de unificar as tarefas de interpretação e edição de movimentos em um único ambiente, apresentou-se desafios para esta implementação. Entre eles destacam-se a

integração das classes, o compartilhamento de algumas funções essenciais entre as tarefas e a integração entre as aplicações, sendo supridas ao longo do processo de desenvolvimento.

Com a utilização de protocolos de comunicação (SSH e SCP) e uma biblioteca de sincronização e distribuição de dados (Glass), foi possível a unificação das funcionalidades do sistema em um único ambiente, fazendo com que o usuário tenha a percepção de uma única ferramenta de envolvimento.

Na descrição do desenvolvimento do projeto diversas imagens, diagramas e códigos foram apresentados, com a finalidade de ilustrar da melhor forma possível todo o processo, facilitando a visualização dos processos necessários para a criação efetiva do sistema. Os resultados obtidos da aplicação respondem de maneira rápida e precisa às instruções enviadas pelo usuário à ferramenta, limitando-se apenas, ao tempo levado para que os comandos SSH disparem os serviços necessários e que a biblioteca identifique os nós do aglomerado gráfico.

Como trabalhos futuros, novas maneiras de interação poderiam ser adicionadas ao trabalho, acrescentando possíveis comparações entre as técnicas utilizadas, realçando vantagens e desvantagens, além de apresentar facilidades e dificuldades. Dentre estas técnicas, estão os rastreadores de movimentos, estes proveem interfaces intuitivas e adaptativas, como por exemplo, o novo dispositivo *Kinect*, da Microsoft. Tal dispositivo, foi atualizado por sua distribuidora em 2011, tornando-o um dispositivo mais preciso e eficiente, nomeado de *kinect Sports Season 2* (MICROSOFT, 2011).

Um trabalho futuro poderia ser incorporar ao sistema a interpretação e edição de gírias da Língua de Sinais, visto que, dependendo da influência cultural e regional, um novo modo de visualização de sinal pode ser introduzido.

Outro aspecto interessante seria o módulo editor salvar os movimentos em formato XML, desta forma, seria possível difundir a ferramenta via Internet, disponibilizando-a para comunidades que vivem em lugares de difícil acesso.

Outra necessidade notada é que o trabalho seria aprimorado se a ferramenta fosse utilizada por um especialista em LIBRAS junto ao processo de desenvolvimento do sistema, agregando valores à ferramenta para maior reconhecimento e validação pela literatura.

Referência Bibliográfica

- Ames, A.L.; Nadeau, D.R.; Moreland, J.L. **VRML 2.0 Sourcebook**. 2º ed. New York: John Wiley, 1997. 654 p.
- Araujo, A. V. **Weblog e jornalismo: os casos de no mínimo Weblog e observatório da imprensa (blog)**. Dissertação de Mestrado da Universidade de São Paulo, USP. 2006.
- Azuola, F. et al. **Building anthropometry-based virtual human models**. In Proc. IMAGE VII Conf., 1994.
- Badler N. **Real-time humans**. In International Conference on Digital Media Futures. Pacific Graphics, 1997.
- Baerentsen, K.B. **Intuitive user interfaces**. Scandinavian Journal of Information Systems. Janeiro, 2001.
- Baitello, J. N. **Comunicação, mídia e cultura**. Revista da Fundação Saede. v.12, n. 4. Out./Dez., 1998. São Paulo. P. 11-16.
- Bangham, J. A. et al. **Virtual signing: capture, animation, storage and transmission - an overview of the visicast project**. IEE Seminar on “Speech and language processing for disabled and elderly people”, 2000.
- Baptista, F. **F-LIBRAS - Ambiente Integrado de Ensino-Aprendizagem para Língua Brasileira de Sinais**. Dissertação do Programa de Mestrado em Ciência da Computação, Fundação de Ensino “Eurípides Soares da Rocha”, Marília, 2007.
- Bates, J.; Loyall, A.; Reilly, W. **Integrating Reactivity, Goals and Emotions in a Broad Agent**. Proceedings of the 14th Annual Conference of the Cognitive Science Society, Indiana, Julho, 1992.
- Battison, R. **Phonological deletion in american sign language**. In: Congresso da Sociedade Brasileira de Computação, Curitiba – PUCPR, 2000.
- Bernardes, J. et al. **Integrating the Wii controller with enJine: 3D interfaces extending the frontiers of a didactic game engine**. Comput.Entertain. 2009.
- Bradshaw, D.; Ng, K. **Analyzing a Conductors Gestures with the Wiimote**. London Conference. University of Leeds. ICSRiM, 2008.
- Braga. M. **Realidade Virtual e Educação**. Revista de Biologia e Ciências da Terra, v. 1, n. 1. 2001.
- Brito, L. F. **Por uma gramática de Língua de Sinais**. Rio de Janeiro, Tempo Brasileiro: UFRJ, Departamento de linguística e filosofia, 1995.

Brito, L. F. **Uma abordagem fonológica dos sinais da LSCB**. Espaço: Informativo técnico científico do INES, Rio de Janeiro, v.1, n.1, p.20–43, 1974.

Burdea, G. **Force and Touch Feedback for Virtual Reality**. Editora John Wiley & Sons – ISBN: 047102141-5. 2005.

Buss, S. R. **Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods**. Abril. 2004.

Buyya, R. **High performance cluster computing**. Programming and Applications. v. 2. Prentice Hall - PTR. 1999.

Card, S. K. et al. **“Readings in Information Visualization: Using Vision to Think”**. Morgan Kaufmann Series in Interactive Technologies, Academic Press, 1999.

Cassell, J. et al. **Animated conversation: Rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents**. In Computer Graphics, Annual Conf. Series, p. 413–420. ACM. 1994.

Chai J.; Xiao, J.; Hodgins, J. **Vision-based Control of 3D Facial Animation**. In Eurographics SIGGRAPH Symposium on Computer Animation, p. 193-206. ACM, San Diego CA. 2003.

Cochard, D.; Pham, K. **Navigation in a 3D environment: Using the Wiimote to map natural gestures**. Disponível em:
video.davidcochard.com/projects/CS6456/Cochard_Pham_report.pdf . Acessado em 24 de setembro de 2010.

Controllers at Nitendo: **Wii Console**. Disponível em:
<http://www.nintendo.com/wii/console/controllers> . Acessado em 24 de novembro de 2010.

Cruz-Neira, C.; Sandin, D. J.; Defanti, T. A. **Surround-screen projection-based virtual reality: The design and implementation of the cave**. ACM SIGGRAPH '93 Proceedings, Anaheim, p. 135 – 142. 1993

Cusin, C. A. **Um Sistema Multimodal com Movimentos Interpretados em Linguagem Natural Controlada**. Dissertação de Mestrado em Ciência da Computação. UNIVEM. Marília. São Paulo. 2005.

Cusin, G. M. **Ambiente Virtual interpretador de sentenças em português para Língua Brasileira de Sinais**. Dissertação de Mestrado. UNIVEM. Marília. 2004.

Davison, A. **Pro Java 6 3D Game Development: Java 3D, JOGL, JInput and JOAL APIs**. Editora Apress. ISBN: 13: 978-1-59059-817-7. New York – Estados Unidos da América, 2007.

Desire, N. **Um Modelo de Gerência de Segurança para Middleware Baseado em Tuple para Ambientes Difusos e Nômades**. Tese de Doutorado. Escola Politécnica da Universidade de São Paulo. 2010.

Dias, D. R. C. et al. **Java 3D para Sistemas de Multiprojeção utilizando Aglomerados Gráficos**. In: WRVA 2010, 2010, São Paulo. Anais do WRVA 2010, 2010. v. 1.

Dix, A. J. et al. **Human-Computer Interaction**. 2º ed. Prentice Hall. 1998.

Dizeu, L. C. T. B.; CAPORALI, S. A. **A Língua de Sinais constituindo o surdo como sujeito**. Educ. Soc. v. 26 n. 91. Campinas. Mai./Ago. 2005.

Domingues, D. G. **O uso de metáforas na computação**. São Paulo. Dissertação de Mestrado, Escola de Comunicações e Artes da Universidade de São Paulo. p. 157. 2001.

Douville, B.; Levison, L.; Badler, N. **Task level object grasping for simulated agents**. Presence. 1996.

Dunlop, R. **Sams teach yourself directx 7 in 24 hours**. Indianapolis, IN: Sams, p. 583. 2000.

Engelbart, D. C. **Augmenting human intellect: a conceptual framework. Summary report**. Menlo Park (CA): Stanford Research Institute. SRI Project n. 3578, contract AF49(638) – 1024. Prepared for Director of Information Sciences, Air Force Office of Scientific Research, Washington 25, DC. p. 133. Outubro. 1962.

Fernandes, J. J. C. **Interação com Ecrã de Larga Escala Usando Dispositivo Multi-Sensores**. Dissertação de Mestrado em Engenharia Informática e de Computadores. Instituto Superior Técnico. Lisboa. 2010.

Ferreira, A. B. H. **Novo Dicionário Aurélio da Língua Portuguesa**. Editora Positivo. ISBN 9788538528241. 2010.

Ferreira, V. A. **O processo de comunicação**. II Jornada Científica FIB – Faculdades Integradas de Bauru, v. 2007, p. 195 – 196. 2007.

Franco, J. M. **Comunicação e Interface: Da necessidade da estética**. Artigo da Universidade Federal de Juiz de Fora – UFJF. 2006. Disponível em URL: <http://www.cencib.org/simposioabciber/PDFs/CC/Joana%20Franco.pdf>. Acessado em 25 de fev. 2010.

Fusco, E. **X-LIBRAS: Um Ambiente Virtual para a Língua Brasileira de Sinais**. Dissertação de Mestrado. UNIVEM. Marília, 2004.

Garavello, M. P. E. **Os blogs e o jornalismo de texto: as eleições presidenciais de 2006 no Brasil 2009**. Dissertação de mestrado da Universidade de São Paulo, USP. 2009.

Gl.Tter. **WIIYOURSELF! – A native C++Wiimote library by gl.tter**. 2008. Disponível em: <http://wiiyourself.gl.tter.org/>. Acessado em 26 de setembro de 2010.

Granieri J.; Crabtree J.; Badler N. **Off-line Production and Real-time Playback of Human Figure Motion for 3D Virtual Environments**. In VRAIS Conf. IEEE Press, 1995.

Góes, M. C. R. **Linguagem, Surdez e Educação**. Editora Autores Associados. Campinas – São Paulo. 1996.

Guimarães, M. P. **Um ambiente para o desenvolvimento de aplicações de Realidade Virtual baseadas em aglomerados gráficos**. Tese de Doutorado. Escola Politécnica da Universidade de São Paulo. São Paulo. 2004.

Haruna, N. C. A. **Teorema de Thales: Uma abordagem do processo ensino-aprendizagem**. Dissertação de Mestrado em Educação Matemática. Pontifícia Universidade Católica de São Paulo – PUC –SP. 2000.

Hay, S.; Newman, J.; Harle, R. 2008. **Optical tracking using commodity hardware**. In : 7th IEEE International Symposium on Mixed and Augmented Reality 2008 , p. 15-18. Setembro, 2008. Piscataway, NJ, USA: IEEE, p. 159-60.

Holden, E. J.; Roy, G. G. **The graphical translation of english text into signed english in the hand sign translator system**. In Kilgour, A. and Kjeldahl, L., Editora, Computer Graphics Forum (EUROGRAPHICS '92 Proceedings), v. 11, p. 357–366. 1992a.

Holden, E.-J.; Roy, G. G. **Learning tool for signed english**. In SAC '92: Proceedings of the 1992 ACM/SIGAPP Symposium on Applied computing, p 444– 449, New York, NY, USA. ACM Press. 1992b.

Johnson, S. **Cultura da Interface: como o computador transforma nossa maneira de criar e comunicar**. Rio de Janeiro: Jorge Zahar Ed. Tradução Maria Carmelita Pádua Dias. Revisão técnica: Paulo Vaz. P. 189. 2001.

Karnopp, L. B. **Aquisição do parâmetro de configuração de mão na língua brasileira de sinais (LIBRAS): estudo sobre quatro crianças surdas, filhas de pais surdos**. Dissertação de mestrado em letras – Instituto de Letras e Artes, Pontifícia Universidade Católica do Rio Grande do Sul – PUCRS. Porto Alegre, 1994.

Kay, A.. **User Interface: a personal view**. In: LAUREL, Editora Brenda The art of human-computer interface design. Reading (MA): Addison – Wesley Publishing. Support: S. Joy Mountford. 4º printing, 523 p. 191 – 207. 1990.

Kirner, C.; Siscoutto, R. **“Fundamentos de Realidade Virtual e aumentada”**. In: Kirner, C.; Siscoutto, R. Realidade Virtual e aumentada – Conceito, Projeto e Aplicações. Porto Alegre: Editora SBC, 2007.

Laforest M. **Wiiuse**. Disponível em: <http://www.wiiuse.net/>. Acessado em 30 setembro de 2010.

Legislação de LIBRAS. Disponível em: <http://www.libras.org.br/leilibras.php>. Acessado em 13 de julho de 2011.

Lee, JC. **Hacking the Nintendo Wii remote**. IEEE Pervasive Computing, p. 39-45. 2008.

Lévy, P. **As tecnologias da inteligência: o futuro do pensamento na era da informática**. Rio de Janeiro: Editora 34. Tradução: Carlos Irineu Costa. 1º ed, 4º reimpressão. p. 208. (Coleção Trans). 1993

Lincoln, M.; Cox, S. J.; Nakisa, M. **The development and evaluation of a speech to sign translation system to assist transactions**. Journal of Human-computer Studies. 1998.

Machado, Liliane S. **Dispositivos não convencionais para interação e imersão em Realidade Virtual e Aumentada**. In: José R. F. Brega, Judith Kelner. (Org.). Interação em realidade virtual e realidade aumentada. Bauru: Canal 6, 2010, v 1, p. 23-33.

Marcato, S. A. et al. **Um Ambiente para a Aprendizagem da Língua de Sinais**. In: SBC 2000 – XX Congresso da Sociedade Brasileira de Computação, PUCPR - Curitiba, agosto de 2000.

Microsoft, 2011. *Microsoft hiring points to next generation kinect*. Disponível em: <http://thenextweb.com/microsoft/2011/03/21/microsoft-hiring-points-to-next-generation-kinect/>. Acessado em 01 de julho de 2011.

Modesto, F. et al. **Fundamentos e Tecnologia de Realidade Virtual e Aumentada**, Cap. Humanos Virtuais e Avatares, Livro do Pré Simpósio – Belém PA, 02 de Maio de 2006.

Nascimento, C., A. **Sistema de Comunicação em LIBRAS em um Ambiente Distribuído de Realidade Virtual**. Dissertação de Mestrado. UNIVEM. Marília, 2008.

Nedel, L. P.; Schneider, A. R. A. **Animação de humanos virtuais voltada para Língua de Sinais**. In: XVII Simpósio Brasileiro de Informática na Educação, 2006, Brasília. Anais do XVII SBIE - Informática e Educação na Construção da Cidadania. Brasília : Positiva, v. 1. 2006.

Netto, A. V.; Machado, L. S.; Oliveira, M. C. F. **Realidade Virtual: Fundamentos e Aplicações**. Editora Visual Books, Florianópolis – SC. 2002.

Nintendo, 2008. **Introducing Wii Motion Plus, Nintendo's upcoming accessory for the revolutionary Wii Remote**. Disponível em: http://www.nintendo.com/whatsnew/detail/eMMuRj_N6vntHPDycCJAKWhEO9zBvyPH. Acessado em 24 de setembro de 2010.

Oracle. **Java SE Desktop Technologies – Java 3D API**. Disponível em: <http://www.oracle.com/technetwork/java/javase/tech/index-jsp-138252.html>. Acessado em 26 de maio de 2011.

Orlando, R. **Dispositivos da interface: um estudo sobre tecnologias de comunicação**. Tese de Doutorado em Ciências da Comunicação. USP – São Paulo, 2006. Disponível em URL: <http://www.teses.usp.br/>. Retirado em 22 de fev. 2010.

Papadogiorgaki, M. et al. **Synthesis of virtual reality animations from swml using mpeg-4 body animation parameters**. In Workshop on the Representation and Processing of Sign

Languages, 4th International Conference on Language Resources and Evaluation LREC. 2004.

Peek, B. 2007. **Managed Library for Nintendo's Wiimote**. Disponível em: <http://blogs.msdn.com/coding4fun/archive/2007/03/14/1879033.aspx>. Acessado em 24 de setembro de 2010.

Pellegrino, C. N. **Estudos em Imagens Falantes: Estimulação do ensino e treinamento de Leitura Labial e Língua de Sinais - LIBRAS VIA CD-ROM em crianças surdas**. Dissertação de Mestrado. UNICAMP. Campinas. 1996.

Perles, J. B. **Comunicação: conceitos, fundamentos e história**, 2007. Biblioteca on-line de Ciência da Comunicação. Disponível em www.bocc.ubi.pt. Acessado em 24 fev. 2010.

Pletsch, E. L. **Modelagem de Percepção de Humanos Virtuais Baseada em Dados Geométricos e Ray-Casting**. Dissertação de Mestrado. Universidade do Vale do Rio dos Sinos, UNISINOS. São Leopoldo. 2006.

Pontes, A. M.; Orth, A. I. **Uma Proposta de Interface de Software Orientada à Linguagem de Sinais**. Disponível em: <http://www.unicamp.br/~ihc99/Ihc99/AtasIHC99/art31.pdf>. Acessado em 15 dez. 2009.

Quadros, R. M.; Karnopp, L. **Língua de Sinais Brasileira: Estudos Linguísticos**. Artmed, 2004.

Quadros, R. M. **Educação de Surdos – Aquisição da Linguagem**. Artes Médicas. 1º ed. Porto Alegre, 1997.

Quadros, R. M. **Diversidade e unidade nas línguas de sinais: LIBRAS e ASL**. IN: SKLIAR, Carlos. (Org.) Atualidade da Educação Bilíngue para Surdos. Porto Alegre, v. 2, p. 195-207. 1999.

Raposo, A. B. et al. **Visão Estereoscópica, Realidade Virtual, Realidade Aumentada e Colaboração**. In: SBC. (Org.). Anais do XXIV Congresso da Sociedade Brasileira de Computação, v. 2, p. 289-331. 2004.

Rocha, H. V. et al. **Um Ambiente para a Aprendizagem da Língua de Sinais**. In: SBC 2000 – XX Congresso da Sociedade Brasileira de Computação, PUCPR - Curitiba, agosto de 2000.

Rocha, H. V. et al. **Conceitos Básicos sobre Videoconferência**. Mini Curso Virtual da Equipe de Educação a Distância do Centro de Computação da UNICAMP. Campinas – SP. 2009. Disponível em: http://www.ead.unicamp.br/minicurso/video/texto/tabela_conteudo.html. Acessado em 19 maio 2010.

Rodello, I. A.; Brega, J. R. F.; Sementille, A. C. **Interação com dispositivos de entrada não convencionais em ambientes virtuais desenvolvidos com Java 3D**. Proceedings SVR 2003 – VI Symposium on Virtual Reality. 15-18 Outubro. Ribeirão Preto, SP – Brasil.

Santarosa, L. M. **Telemática y la inclusión virtual y social de personas con necesidades especiales: un espacio posible en la Internet**. in: V Congresso Ibero-Americano de Informática na Educação- RIBIE: Viñadelmar, 2000.

Schneider, A. R. A. **Animação de Humanos Virtuais Aplicada para Língua Brasileira de Sinais**. Dissertação de Mestrado. UFRGS. Porto Alegre, 2008.

Souza, V. C.; PINTO, S. C. C. S. **Sign WebMessage: uma ferramenta para comunicação via web através da Língua Brasileira de Sinais – Libras**. XIV Simpósio Brasileiro de Informática na Educação, UFRJ - Rio de Janeiro, nov. de 2003.

Strickland, S. **“HowStuffWorks – Como funciona o equipamento de Realidade Virtual”**. Publicado em 10 de agosto de 2007. <http://eletronicos.hsw.uol.com.br/equipamentos-realidade-virtual6.htm>. Acessado em 01 de fev. de 2011.

Sutton, V. **Signwriter manual**. In Deaf Action Committee for SignWriting. 1995.

Teichrieb, V. ; Figueiredo, L. S. **Interação Natural**. In: José Remo Ferreira Brega; Judith Kelner. (Org.). Interação em Realidade Virtual e Aumentada. Bauru: Canal 6 Editora, 2010, v 1, p. 9-22.

Thalmann, D. et al. **Avatar in Network Virtual Environments**, England: Willey, 1999.

Torres, J. A. R.; Maciel, A.; Nedel, L. P. Uma **Arquitetura para Animação de Humanos Virtuais com Raciocínio Cognitivo**. SVR 2002 - SYMPOSIUM ON VIRTUAL REALITY, 2002, Fortaleza - CE Proceedings. Fortaleza: SBC. 2002.

Tsapanidou, A. A. **Virtual Room Scene with the Wii Remote/Motion Plus**. Dissertação de Mestrado. The Nottingham Trent University – School of Science and Technology. 2009.

Valli, A. **Notes on Natural Interaction**. 2005.

Vilela, I. M. O. **Modelos e Aplicações de Humanos Virtuais**. I Workshop de realidade virtual e visualização científica do LAMCE(Laboratório de Métodos Computacionais em Engenharia). UFRJ - Rio de Janeiro, 2002.

Web3D Consortium. **X3D Resources**. Disponível em <http://www.web3d.org/x3d/content/help.html>. Acessado em 26 de maio de 2011.

Welter JR, A.; Hounsell, M. S. **Representação de Humanos Virtuais**. REIC. Revista Eletrônica de Iniciação Científica JCR, <http://www.sbc.org.br/reic/>, v. 4, n. 1, p. 1-20. 2004.

WiiuseJ, 2008. **Java API for Wiimotes**. Disponível em: <http://code.google.com/p/wiiusej/>. Acessado em 01 de fevereiro de 2011.

Wrigth Jr., R.S.; Sweet, M. **OpenGL SuperBible**. 2ª ed. Indianapolis, Indiana: Waite Group Press, p. 696. 2000.

ANEXO A - LEI Nº 10.436, DE 24 DE ABRIL DE 2002

Dispõe sobre a Língua Brasileira de Sinais – LIBRAS e dá outras providências.

O PRESIDENTE DA REPÚBLICA

Faço saber que o Congresso Nacional decreta e eu sanciono a seguinte Lei:

Art. 1º É reconhecida como meio legal de comunicação e expressão a Língua Brasileira de Sinais – LIBRAS e outros recursos de expressão a ela associados.

Parágrafo único. Entende-se como Língua Brasileira de Sinais – LIBRAS a forma de comunicação e expressão, em que o sistema linguístico de natureza visual-motora, com estrutura gramatical própria, constitui um sistema linguístico de transmissão de ideias e fatos, oriundos de comunidades de pessoas surdas do Brasil.

Art. 2º Deve ser garantido, por parte do poder público em geral e empresas concessionárias de serviços públicos, formas institucionalizadas de apoiar o uso e difusão da Língua Brasileira de Sinais – LIBRAS como meio de comunicação objetiva e de utilização corrente das comunidades surdas do Brasil.

Art. 3º As instituições públicas e empresas concessionárias de serviços públicos de assistência à saúde devem garantir atendimento e tratamento adequado aos portadores de deficiência auditiva, de acordo com as normas legais em vigor.

Art. 4º O sistema educacional federal e os sistemas educacionais estaduais, municipais e do Distrito Federal devem garantir a inclusão nos cursos de formação de Educação Especial, de Fonoaudiologia e de Magistério, em seus níveis médio e superior, do ensino da Língua Brasileira de Sinais – LIBRAS, como parte integrante dos Parâmetros Curriculares Nacionais – PCNs, conforme legislação vigente.

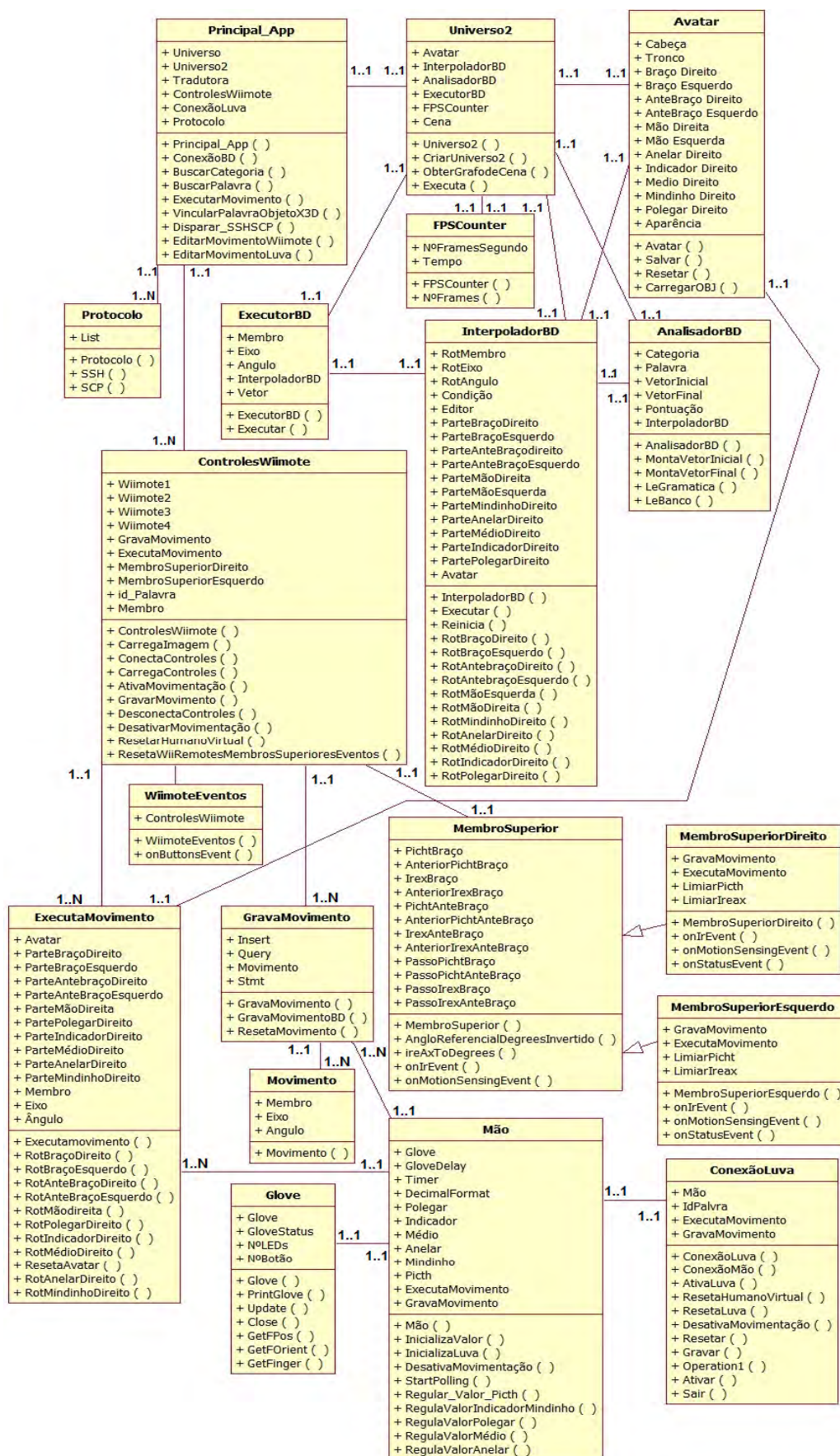
Parágrafo único. A Língua Brasileira de Sinais – LIBRAS não poderá substituir a modalidade escrita da língua portuguesa.

Art. 5º Esta Lei entra em vigor na data de sua publicação.

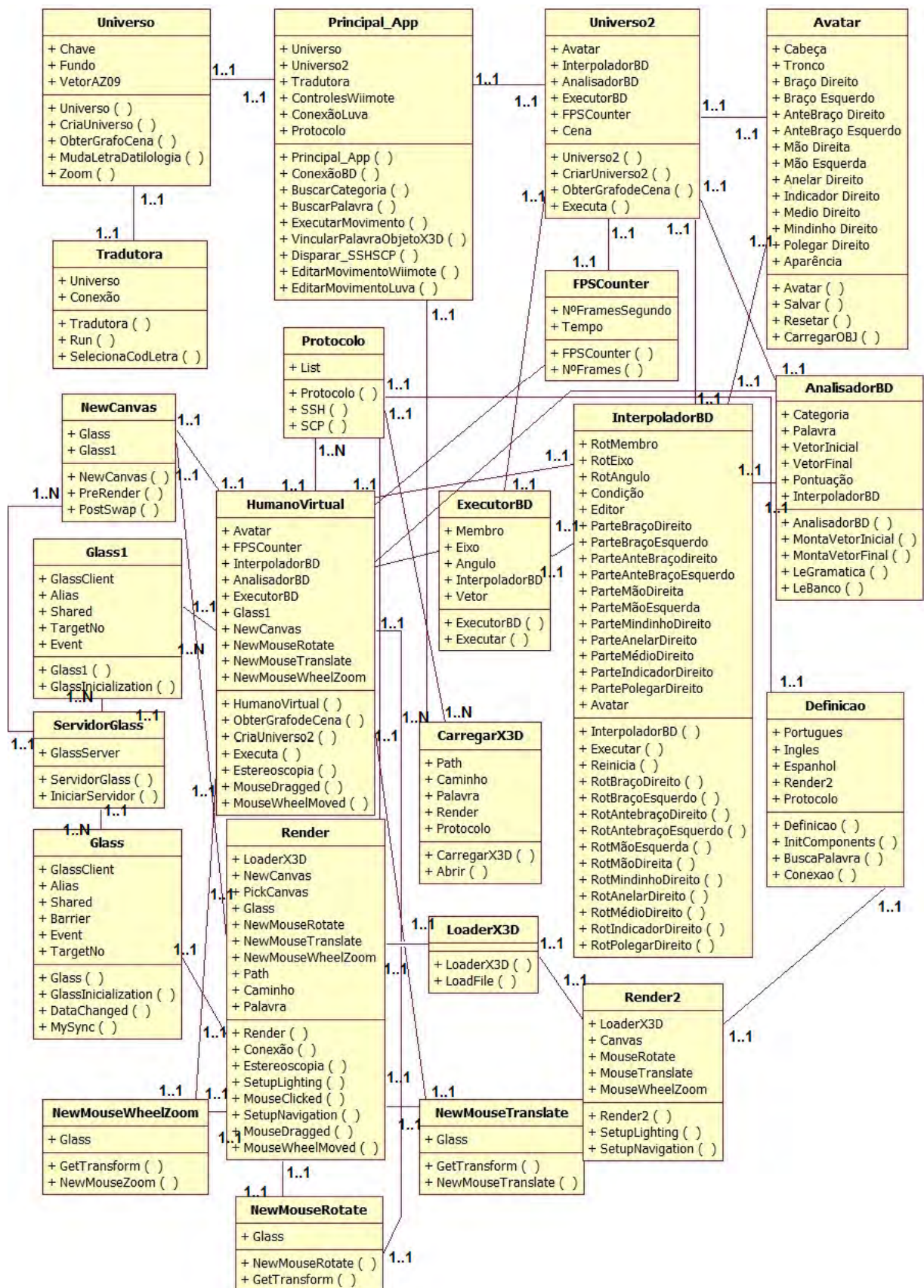
Brasília, 24 de abril de 2002; 181º da Independência e 114º da República.

FERNANDO HENRIQUE CARDOSO (LEGISLAÇÃO DE LIBRAS, 2011)

APÊNDICE A - DIAGRAMA DE CLASSE DO MÓDULO EDITOR



APÊNDICE B - DIAGRAMA DE CLASSE DO MÓDULO INTERPRETADOR



APÊNDICE C - ESTEREOSCOPIA

Um dos objetivos de sistemas que utilizam RV ou RA é estimular a parte visual do usuário dando-lhe a sensação tridimensional. Desta forma, o sistema deve gerar simultaneamente duas imagens diferentes, correspondendo às visões de cada olho. Este tipo de geração de imagem é conhecido como visão estereoscópica, ou simplesmente estereoscopia (RAPOSO et. al., 2004).

A estereoscopia é a simulação de duas imagens da cena que são projetadas nos olhos em pontos de observação ligeiramente diferentes, sendo que o cérebro as funde de maneira a obter informações quanto à profundidade, distância, posição e tamanho dos objetos, gerando uma sensação de visão em 3D.

A estereoscopia oferece efeitos passivos e efeitos ativos. Nos efeitos passivos, pode-se observar a perspectiva, a iluminação, a oclusão, a sombra e o gradiente da textura. Nos efeitos ativos tem-se a estereoscopia visual (RAPOSO et. al., 2004).

A perspectiva é relativa a tamanhos, ou seja, as figuras mais ao fundo da imagem são menores do que as mais à frente. Ressaltando que ao fundo e mais a frente são características impróprias, pois a imagem é planar. A iluminação adiciona à imagem uma descrição melhor da forma dos objetos presentes, contribuindo com o aumento do grau de realismo. A oclusão esconde partes de objetos, ou por inteiro, que estejam por de trás de outros. A sombra impõe a ideia de que o objeto pode não estar sob o plano, atribuindo a necessidade da sombra para aumentar o realismo da imagem. E por fim a gradiente de textura, que usado de diversos tamanhos, ou seja, com um decrescimento nos tamanho, pode indicar pontos onde se projetam algumas direções, dando a noção de profundidade.

A estereoscopia visual é um sistema óptico cujo componente final é o cérebro humano, pois ele faz a fusão das duas imagens captadas por cada olho resultando em noções de profundidade (RAPOSO et. al., 2004).

Estereoscopia classifica-se em estereoscopia passiva e ativa. Na estereoscopia passiva, as imagens do par estéreo são reproduzidas sobrepondo-se. Assim são utilizados óculos com algum filtro passivo (como polarização da luz, ou separação das cores do espectro, também chamada de anáglifo) para que cada olho veja apenas uma das imagens (RAPOSO et. al., 2004).

No estéreo anaglífico, a filtragem é feita por cores, sendo utilizadas as cores vermelha, azul e atualmente, o verde. Cada cor é direcionada apenas para um olho, sendo a cor verde atribuída a um dos olhos para melhorar a qualidade da imagem.

Na utilização de polarizadores lineares, dois projetores são utilizados e cada qual contendo uma lente polarizadora na frente, responsável por polarizar, verticalmente ou horizontalmente, a luz emitida pelo projetor. A Figura 59 ilustra este tipo de estéreo.

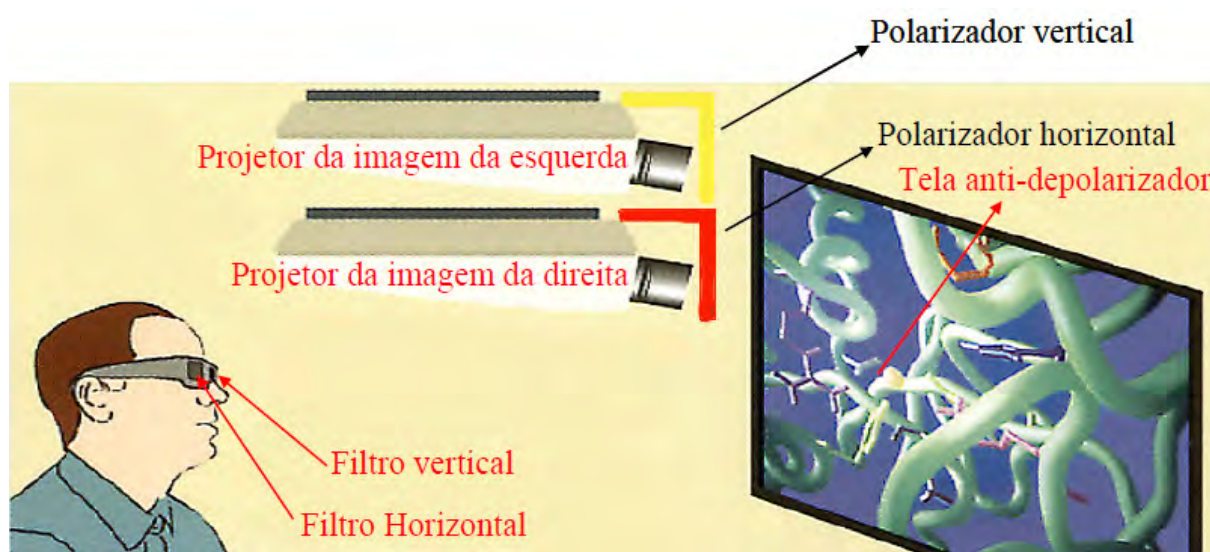


Figura 59 - Esquema para um sistema de estéreo passivo por polarizador (ALBERTO et. al., 2011)

Na estereoscopia ativa os usuários utilizam óculos obturadores (*shutter glasses*), controlados em geral por um sinal infravermelho, trabalhando em sincronia com os projetores ou monitores. As lentes alternam entre dois estados, transparentes ou opacas. Quando o projetor exibe a imagem esquerda os óculos fecham a passagem de luz para o olho direito, permitindo que a imagem referente ao olho esquerdo seja somente captada por ele, e vice versa (RAPOSO et. al., 2004).

Já em fundamentos matemáticos, para a criação e visualização de imagens há a paralaxe, está sendo a distância entre os pontos correspondentes das imagens de cada olho para a imagem projetada na tela. A disparidade e a paralaxe são similares, porém uma é medida na retina e outra medida na tela do computador (RAPOSO et. al., 2004).

Segundo Raposo et. al. (2004), há três tipos de paralaxe: a paralaxe zero, a negativo e a positivo. A paralaxe zero se encontra no plano de projeção, tendo a mesma projeção para os dois olhos. A paralaxe negativo faz com que o cruzamento dos raios de projeção para cada olho esteja entre os olhos e a tela de projeção, dando a percepção de o objeto estar saindo da tela. A paralaxe positivo indica que o ponto está atrás do plano de projeção, dando a

impressão de que o objeto está atrás da tela de projeção (RAPOSO et. al., 2004). A Figura 60 ilustra os três tipos de paralaxe.

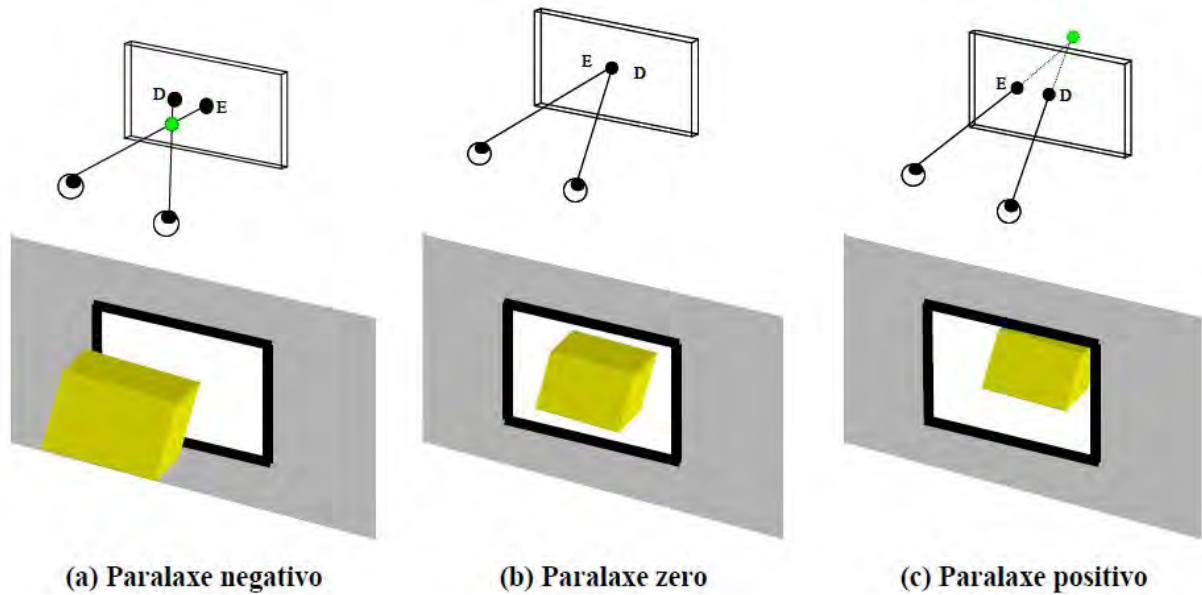


Figura 60 - Tipos de Paralaxe (RAPOSO et. al., 2004)

Vale ressaltar que a distância do observador à tela afeta diretamente no efeito estereoscópico, sendo que, quanto maior à distância, maior será o efeito estereoscópico percebido pelo usuário.

APÊNDICE D - GERAÇÃO E GRAVAÇÃO DE MOVIMENTOS

A classe “Movimento” é a responsável por armazenar a granularidade mínima dos movimentos gerados pelos dispositivos não convencionais, tanto para os controladores Wiimote, quanto para a Luva P5-Glove. A Figura 61 ilustra a classe.

```
public class Movimento {  
  
    private int membro;  
    private int eixo;  
    private float angulo;  
  
    public Movimento(int membro, int eixo, float angulo) {  
        this.membro = membro;  
        this.eixo = eixo;  
        this.angulo = angulo;  
    }  
  
    public float getAngulo() {  
        return angulo;  
    }  
  
    public int getEixo() {  
        return eixo;  
    }  
  
    public int getMembro() {  
        return membro;  
    }  
}
```

Figura 61 - Classe Movimento

A classe é bem simples contém apenas um construtor que recebe as informações respectivas aos movimentos descritos pela sua instância e métodos *get()*, que retornam o ângulo, o membro e o eixo do respectivo movimento instanciado.

Esta classe é utilizada, posteriormente, como um vetor de movimento, sendo armazenadas previamente, instruções de um respectivo movimento gerado pelos dispositivos.

A classe responsável por executar os movimentos em tempo de edição é a classe chamada “ExecutaMovimento”. Esta oferece a execução dos movimentos para ambos os dispositivos, sendo os movimentos executados, assim que gerados. Desta forma, as instruções são passadas ao HV em tempo de execução para a pré-visualização dos movimentos.

O construtor da classe recebe o objeto Avatar, criando-se nós *TransformGroup* para cada membro do objeto, na qual são rotacionados de acordo com seus métodos respectivos. A Figura 62 ilustra o método construtor da classe.

```

public ExecutaMovimento(Avatar avatar) {
    parteBracoD = avatar.bracodireito;

    parteAntebracoD = avatar.bracodireito.antebraodireito;
    parteMaoD = avatar.bracodireito.antebraodireito.maodireita;

    parteBracoE = avatar.braco esquerdo;
    parteAntebracoE = avatar.braco esquerdo.antebrao esquerdo;
    parteMaoE = avatar.braco esquerdo.antebrao esquerdo.mao esquerda;

    partePolegarD1 = avatar.bracodireito.antebraodireito.maodireita.polegardireito;
    parteIndicadorD1 = avatar.bracodireito.antebraodireito.maodireita.indicadordireito;
    parteMedioD1 = avatar.bracodireito.antebraodireito.maodireita.mediodireito;
    parteAnelarD1 = avatar.bracodireito.antebraodireito.maodireita.anelardireito;
    parteMindinhoD1 = avatar.bracodireito.antebraodireito.maodireita.mindinhodireito;

    partePolegarD2 = avatar.bracodireito.antebraodireito.maodireita.polegardireito.polegardireito2;
    parteIndicadorD2 = avatar.bracodireito.antebraodireito.maodireita.indicadordireito.indicadordireito2;
    parteMedioD2 = avatar.bracodireito.antebraodireito.maodireita.mediodireito.mediodireito2;
    parteAnelarD2 = avatar.bracodireito.antebraodireito.maodireita.anelardireito.anelardireito2;
    parteMindinhoD2 = avatar.bracodireito.antebraodireito.maodireita.mindinhodireito.mindinhodireito2;

    parteIndicadorD3 = avatar.bracodireito.antebraodireito.maodireita.indicadordireito.indicadordireito2.indicadordireito3;
    parteMedioD3 = avatar.bracodireito.antebraodireito.maodireita.mediodireito.mediodireito2.mediodireito3;
    parteAnelarD3 = avatar.bracodireito.antebraodireito.maodireita.anelardireito.anelardireito2.anelardireito3;
    parteMindinhoD3 = avatar.bracodireito.antebraodireito.maodireita.mindinhodireito.mindinhodireito2.mindinhodireito3;

    a = avatar;
}

```

Figura 62 - Método Construtor da Classe "ExecutaMovimento"

A seguir a Figura 63 ilustra o método responsável por rotacionar o braço direito, totalizando-se em 19 métodos responsáveis por aplicar transformações em seu membro referido.

```

/**
 * Rotaciona o Braço Direito de acordo com os parametros de entrada.
 * @param eixo 0 -> eixo X, 1 -> eixo Y, 2 -> eixo Z.
 * @param angulo Angulo em Radianos que o Braço Direito deve ser rotacionado.
 */
public void rotBracoDireito(int eixo, float angulo) {
    switch (eixo) {
        case 0:
            parteBracoD.getTransform(t1);
            t2.setIdentity();
            t2.rotX(angulo);
            t1.mul(t2);
            parteBracoD.setTransform(t1);
            break;
        case 1:
            parteBracoD.getTransform(t1);
            t2.setIdentity();
            t2.rotY(angulo);
            t1.mul(t2);
            parteBracoD.setTransform(t1);
            break;
        case 2:
            parteBracoD.getTransform(t1);
            t2.setIdentity();
            t2.rotZ(angulo);
            t1.mul(t2);
            parteBracoD.setTransform(t1);
            break;
    }
}

```

Figura 63 - Método rotBracoDireito

Com a elaboração dos movimentos pré-visualizados, já se torna possível a gravação dos mesmos no banco de dados. A classe responsável por este ato é denominada “GravaMovimento”. Esta grava todos os movimentos gerados por ambos os dispositivos em um vetor provisório do tipo *Movimento*, que se confirmados, estes são gravados permanentemente no banco de dados remoto.

Possui métodos *set()* que gravam no vetor provisório as instruções dos movimentos, tendo como parâmetro o membro que foi movimentado, além de adicionar no vetor, o eixo, o ângulo e o membro. A Figura 64 ilustra o método *set()* do braço direito, responsável por armazenar as coordenadas x, y e z do respectivo.

```

/**
 * Grava a rotação do Braço Direito no eixo X.
 * @param BracoDireitoX Angulo Rotacionado.
 */
public void setBracoDireitoX(float BracoDireitoX) {
    Movimento.add(new Movimento(1, 0, BracoDireitoX));
}

/**
 * Grava a rotação do Braço Direito no eixo Y.
 * @param BracoDireitoY Angulo Rotacionado.
 */
public void setBracoDireitoY(float BracoDireitoY) {
    Movimento.add(new Movimento(1, 1, BracoDireitoY));
}

/**
 * Grava a rotação do Braço Direito no eixo Z.
 * @param BracoDireitoZ Angulo Rotacionado.
 */
public void setBracoDireitoZ(float BracoDireitoZ) {
    Movimento.add(new Movimento(1, 2, BracoDireitoZ));
}

```

Figura 64 - Métodos set() do Braço Direito

A classe também permite apagar todos os movimentos armazenados no vetor provisório (*ResetaMovimento()*), além da inserção no banco de dados dos movimentos gerados (*GravaMovimentoBD*). A Figura 65 ilustra o método de armazenamento no banco de dados.

```

public void GravaMovimentoBD(String id_palavra) {
    Movimento movimento = null;
    try {
        stmt.execute(" delete from movimento where copiaPalavra = " + id_palavra);

        for (int indice = 0; indice < Movimento.size(); indice++) {
            movimento = (Movimento) Movimento.get(indice);
            stmt.execute(Insert + "'" + id_palavra + "', " + indice + ", " + movimento.getMembro()
                + ", " + movimento.getEixo() + ", " + movimento.getAngulo() + "');");
        }
    } catch (Exception ex) {
        System.out.println(ex);
    }
}

```

Figura 65 - Método GravaMovimentoBD()

Antes da gravação dos movimentos, o método apaga todas as instruções que estão gravadas no banco de dados, através de comandos delete. Desta forma, há a garantia de que os movimentos executados serão os mesmos a serem gravados.

Foram descritas as classes em comum de ambos os dispositivos, tanto para o controlador Wiimote, quanto para a Luva P5-Glove. Nos próximos dois apêndices, serão apresentados detalhadamente cada dispositivo, descrevendo suas funcionalidades e particularidades.

APÊNDICE E - CONTROLADORES WIIMOTE

Com o lançamento do console Wii no final de 2006, a Nintendo trouxe um novo tipo de dispositivo de controle: o Wii Remote ou Wiimote. O Wiimote foi uma inovação de controladores de jogos existentes, no sentido de oferecer um novo meio de interação do usuário com mundos virtuais e uma nova interface para cada tipo de aplicação de jogo. Este controlador *wireless* é capaz de detectar os movimentos através de um acelerômetro e sensores ópticos na qual permite a interação e manipulação de objetos através de movimentos (TSAPANIDOU, 2009).



Figura 66 - Infravermelho e Wii Motion Plus (CONTROLLERS AT NINTENDO: WII CONSOLE, 2010).

A Nintendo nunca lançou oficialmente as especificações do Wiimote. Entretanto, pesquisadores têm examinado o seu funcionamento interno através da engenharia inversa e documentada suas características técnicas em detalhes para que este seja utilizado em outras aplicações específicas e plataformas diferentes, visto que oferece baixo custo, qualidade na produção e flexibilidade de integração (LEE, 2008).

O Wiimote é alimentado por meio de pilhas e seus principais componentes são um infravermelho, uma câmera, um acelerômetro, uma série de doze botões, um motor de vibração para *feedback* tátil, quatro *LEDs* azuis, um alto falante para *feedback* auditivo, um componente de memória flash interna e uma porta de expansão (LEE, 2008).

Sensores de movimento são alcançados através do acelerômetro que pode capturar dois eixos lineares (x e y) possuindo oito bits de resolução por eixo. O sistema de coordenadas é definido pela mão direita, sendo o eixo x positivo para a esquerda e o eixo y positivo para cima (BERNARDES, 2009).

A câmera infravermelha situada na extremidade frontal do Wiimote tem uma resolução de 1024 x 768 pixels, uma taxa de atualização de 100Hz e um campo de 45 graus de vista. A Nintendo oferece também uma barra de infravermelho que faz com que o Wiimote possa capturar o terceiro eixo do sistema de coordenadas, o eixo z (LEE, 2008).

O Wiimote é um dispositivo *bluetooth*, assim pode se comunicar com qualquer computador que possua o mesmo (LEE, 2008).

A Figura 67 ilustra o funcionamento dos eixos do dispositivo Wimote, sendo o x, y e z chamados respectivamente de *Roll*, *Pitch* e *Yaw*.

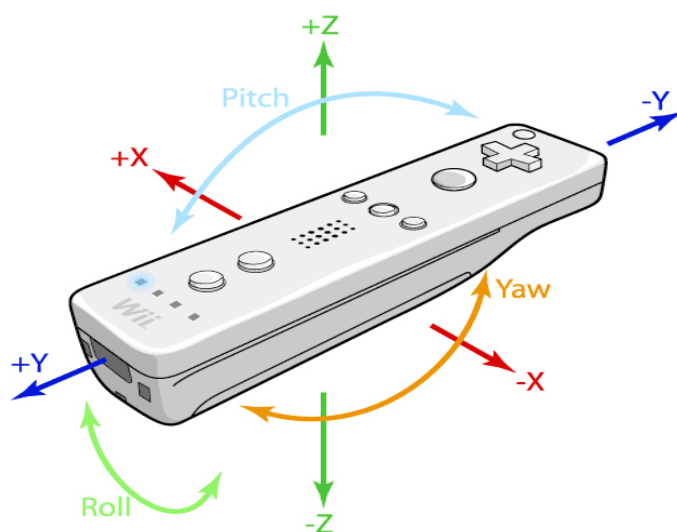


Figura 67 - Coordenadas do Wii Remote (CONTROLLERS AT NINTENDO: WII CONSOLE, 2010)

Apesar de o Wiimote apresentar uma grande capacidade de detecção de movimento, existem limitações que têm sido documentadas por pesquisadores e desenvolvedores. Empresas que utilizaram em seus projetos de desenvolvimento o uso deste controlador para o movimento de objeto 3D em uma cena 3D relataram que enfrentaram vários problemas, principalmente quanto à natureza das saídas do sensor. Alguns problemas foram os ruídos nos dados do acelerômetro, devido à forma da implementação do dispositivo físico, uma precisão limitada, a mudança constante do valor inicial do acelerômetro, a necessidade de derivação matemática na terceira dimensão, a câmera produzindo duas saídas dimensionais e valores de dados que precisam ser filtrados e manipulados matematicamente para posterior utilização (COCHARD; PHAM, s.d.).

Em junho de 2009, a Nintendo lançou uma extensão para o Wii Remote, chamada *Motion Plus*. A extensão *Motion Plus* prometia uma maior precisão em movimentos sensoriais remotos. Os dados obtidos por ele, em conjunto com os dados obtidos pelo

acelerômetro e pela câmera infravermelha (barra de sensor) podia permitir melhor acompanhamento da posição e orientação do controlador no espaço. Assim, com o *Motion Plus*, um mapeamento direto do movimento físico em uma cena virtual era possível (NINTENDO, 2008).

O dispositivo *Motion Plus* também é equipado com uma porta auxiliar, de modo que outras extensões possam ser conectadas ao controlador principal sem a necessidade da remoção do mesmo.

Enfim, o Wiimote é atualmente utilizado de diversas maneiras e em diferentes áreas, devido ao seu baixo custo, qualidade de produção e flexibilidade de integração nas mais diversas plataformas de desenvolvimento (FERNANDES, 2010).

E.1 – Bibliotecas de conectividade

Como mencionado anteriormente, o Wii Remote nunca foi destinado a ser utilizado em outras plataformas além do console Nintendo Wii. Assim, várias bibliotecas auxiliares livres foram desenvolvidas e apresentadas por pesquisadores e equipes para permitir a implementação de Wii Remote em softwares. Essas APIs podem ser usadas em uma variedade de linguagens e ambiente gráficos. Alguns exemplos incluem a biblioteca WiimoteLib desenvolvida por Brian Peek's Managed para a plataforma Windows, com suporte do Visual Basic e C # (PEEK, 2007), a biblioteca WiiYourself em C++ (GL.TTER, 2008), a biblioteca Wiiuse desenvolvida por Michael Laforest, escrito em C (LAFOREST, s.d.) e a biblioteca WiiuseJ, construída em cima da Wiiuse utilizando JNI para JAVA (WIIUSEJ, 2008). Estas APIs fornecem semelhantes funcionalidade quando se trata de conectividade com o Wii Remote. No entanto, o nível de processamento dos dados brutos extraídos do dispositivo varia.

Para os fins deste projeto, a biblioteca WiiuseJ foi escolhida, pois oferece suporte a multiplataforma e à JAVA, visto que a aplicação proposta está desenvolvida nesta linguagem.

E.2 – API WiiuseJ

A API WiiuseJ está licenciada sob a GNU GPLv3 e LGPLv3 GNU para não uso comercial. Suporta vários controles Wii Remote, extensões como o *Nunchuk* e o controle clássico, mas não o *Motion Plus*. Ele é executado em um único segmento e é compatível com

uma série de dispositivos de *bluetooth*, tanto para Linux quanto Windows (TSAPANIDOU, 2009).

No início, esta biblioteca parecia ser menos flexível do que as demais, porém permitia a estimativa mais precisa da orientação e posição do controlador no espaço. Os dados relevantes do infravermelho são tratados internamente de uma maneira avançada, sendo o número de pontos visíveis levado em consideração e os resultados calculados baseados nesse número. O Yaw (eixo z) do controlador é então calculado com relação a estes dados fornecidos pelo infravermelho. A proximidade do Wiimote com a barra de infravermelho é fornecido pela própria biblioteca, de modo que ainda seja necessário cálculos pelo desenvolvedor para obter precisamente a terceira dimensão. Além disso, a opção de especificar a proporção do monitor usado (por exemplo, 4:3, 16:9) está disponível, de modo que a tela virtual está definida e os valores do par de coordenadas x e y são obtidos nesta resolução de tela virtual (LAFOREST, s.d.).

Algumas características importantes da API WiiuseJ (WIIUSEJ, 2008) são:

- Obtêm eventos do acelerômetro em diferentes formas (orientação, força e aceleração);
- Obtêm eventos da câmera infravermelha;
- Obtêm eventos dos botões;
- Obtêm eventos de expansão;
- Faz o Wiimote vibrar;
- Retorna o estado da bateria; e
- Seta as luzes de cada Wiimote.

Portanto foi utilizado a API WiiuseJ, pois apresenta grandes avanços no tratamento de valores brutos extraídos pelo controlador e possui métodos que auxiliam no desenvolvimento do aplicativo em JAVA.

E.3 – Projetos Wiimote

O Wii Remote tem sido utilizado em vários projetos que diferem em sua natureza e escopo. De acordo com o seu modelo de interação, estes projetos poderiam cair em grupos separados. Naturalmente, este agrupamento não é rigoroso, os critérios são baseados no grau de semelhança entre os modelos (TSAPANIDOU, 2009).

Uma abordagem é utilizar o Wiimote como um dispositivo de rastreamento de movimento. Neste caso, o controlador é fixado em um ou mais membro monitorado pela barra

de sensor infravermelho. Lee (2008) escreveu um artigo chamado “*Optical Tracking Using Commodity Hardware*”, na qual Hay, Newman e Harle (2008), discutem um método para utilizar controladores Wiimote como um sistema de visão estéreo para rastreamento 3D ou captura de movimento em tempo real (HAY et. al., 2008). Da mesma forma, Bradsaw e Ng (2008), desenvolveram um sistema para controlar os movimentos da mão condutora utilizando vários controladores Wiimote.

Outro paradigma de interação envolve o uso do controlador como uma real interface do usuário para operar e controlar robôs e braços robóticos. Neste caso, a orientação do Wiimote é a parte vital, o movimento do robô é mapeado no sistema de coordenada (x, y e z) para o dispositivo de entrada (TSAPANIDOU, 2009).

O tema do uso do Wii Remote como um dispositivo de entrada para os mundos virtuais e 3D é bastante comum em aplicações da realidade. Nesta área o controlador é utilizado como um dispositivo apontador. Cochard e Pham desenvolveram uma cena 3D repleta de objetos com formas geométricas, onde o usuário pode mover e colocar esses objetos em qualquer parte da cena. O projeto foca o reconhecimento de gestos naturais, com a finalidade de avaliar a usabilidade de uma interface deste tipo (COCHARD; PHAM, s.d.).

Na maioria dos casos acima, os projetos incorporam o reconhecimento de gestos. Deste modo, padrões de movimento, ao invés de posições absolutas do Wii Remote, são levados em consideração (TSAPANIDOU, 2009).

E.4 – Módulo Editor

O controlador Wiimote garante que todo o movimento gerado pelo braço e antebraço direito e esquerdo, são repassados ao HV. Desta forma, há uma sequencia lógica responsável pelo processo, sendo todas as classes referências da API wiiusej.jar.

Inicialmente há a necessidade de desenvolver uma classe responsável por captar todos os eventos gerados pelos controladores Wiimotes, denominada “MembroSuperior”. Esta inicializa as variáveis que armazenam as posições do controlador, converte os valores decimais para ângulos, além de implementar a classe “WiimoteListener”. A Figura 68 ilustra o método *AngloReferencialDegreesInvertido()* responsável pela conversão dos valores.

```

public float AngloReferencialDegreesInvertido(float PichtRoll) {
    if ( (int)PichtRoll == 180 ) {
        return (float) 90;
    } else {
        if( PichtRoll > 0 )
            return (float) Math.abs(PichtRoll - 90);
        else
            return (float) Math.abs(PichtRoll + 270);
    }
}

```

Figura 68 - Método AngloReferencialDegreesInvertido()

Pelo fato dos controladores serem posicionados inversamente na parte inferior do braço e antebraço direito e esquerdo, devido aumentar significativamente a movimentação e de tornar os botões mais acessíveis, o método calcula os ângulos de maneira invertida.

Os ângulos calculados variam entre 0 e 180 graus, sendo apenas atribuídos a coordenada x. O movimento da coordenada y foi retirado dos membros superiores, devido este ser impossível de ser realizado através do dispositivo Wiimote, pois a anatomia do corpo humano não permite tal movimento. Como o dispositivo se move de maneira uniforme em todos os eixos e o movimento é realizado pelo pulso e propagado ao longo do antebraço e do braço, ou seja, de maneira não uniforme, o movimento se torna impossível de ser repassado ao HV.

A coordenada z é calculada de outra maneira, pois está é gerada pelo Wiimote através da barra de infravermelho. A Figura 69 ilustra o método responsável por calcular os ângulos da coordenada z, variando entre 0 e 180 graus.

```

public float ireAxToDegrees(int ireAx) {
    return (float) ((ireAx / 20) * 1.8);
}

```

Figura 69 - Método ireAxToDegrees()

Com os valores já normalizados, os métodos responsáveis por receberem os eventos gerados pelos controladores já podem ser descritos.

O método *onIrEvent()* verifica qual membro está enviando a atualização da posição da coordenada z (braço ou antebraço) e faz com que a variável correspondente receba este valor normalizado, conforme ilustra a Figura 70.

```

public void onIrEvent(IrEvent ire) {

    if (ire.getWiimoteId() == idBraco) { //Braco
        ireAx_Braco = ireAxToDegrees(ire.getAx());
    }

    if (ire.getWiimoteId() == idAnteBraco) { //AnteBraco
        ireAx_AnteBraco = ireAxToDegrees(ire.getAx());
    }

}

```

Figura 70 - Método onIrEvent()

O método *onMotionSensingEvent()* é semelhante ao *onIrEvent()*, porém este trabalha com o acelerômetro do dispositivo recebendo e utilizando a coordenada x, conforme ilustra a Figura 71. Vale ressaltar que o método também pode utilizar a coordenada y através do acelerômetro disponível no controlador.

```

public void onMotionSensingEvent(MotionSensingEvent mse) {

    if (mse.getWiimoteId() == idBraco) { //Braco
        PICT_Braco = AngloReferencialDegreesInvertido(mse.getOrientation().getAPitch());
    }

    if( mse.getWiimoteId() == idAnteBraco ) { //AnteBraco
        PICT_AnteBraco = AngloReferencialDegreesInvertido(mse.getOrientation().getAPitch());
    }

}

```

Figura 71 - Método onMotionSensingEvent()

Com a necessidade de se desenvolver classes específicas para os membros superiores direito e esquerdo, as classes “MembroSuperiorDireito” e “MembroSuperiorEsquerdo” foram criadas para suprir esta necessidade. Estas estendem a classe “MembroSuperior”, tornando possível a captação de eventos gerados pelos controladores Wiimotes para cada membro. Com a utilização da hierarquia de ambas as classes, estas sobrescrevem os métodos de acordo com as suas peculiaridades.

As classes “MembroSuperiorDireito” e “MembroSuperiorEsquerdo” são idênticas, alterando apenas em qual membro superior está destinada, uma sendo responsável pelo direito e a outra pelo esquerdo. Por esse motivo é descrito apenas a classe “MembroSuperiorDireito”.

Em seu método construtor há a inicialização de variáveis que armazenam as posições das coordenadas, além de variáveis que armazenam o estado da bateria de cada dispositivo e instâncias para execução e gravação dos movimentos.

O método *onIrEvent()* responsável pelos eventos gerados pelos controladores na coordenada z, verifica inicialmente qual membro foi atualizado, encaminhando os novos dados para o respectivo membro. Este utiliza uma variável auxiliar para armazenar o resultado da subtração da posição atual menos a posição antiga, garantindo que o novo valor será a diferente entre eles.

Após a operação, o valor é transformado em radiano e atribuído aos métodos para a execução e armazenamento dos movimentos, conforme ilustra a Figura 72. Os valores antigos são atualizados pelos novos valores para um novo procedimento.

```
public void onIrEvent(IrEvent ire) {  
  
    super.onIrEvent(ire);  
  
    if (ire.getWiimoteId() == idBraco) { //Braco  
        if (Math.abs(a_ireAx_Braco - ireAx_Braco) < LimIREAX) { // a_ireAx != ireAx  
            Universo2.interpolador.editor = 1;  
            passoireAx_Braco = a_ireAx_Braco - ireAx_Braco;  
            executaMovimento.rotBracoDireito(2, (float) Math.toRadians(passoireAx_Braco));  
            executaMovimento.rotAnteBracoDireito(2, (float) Math.toRadians(-passoireAx_Braco));  
            gravaMovimento.setBracoDireitoZ(passoireAx_Braco);  
            gravaMovimento.setAnteBracoDireitoZ(-passoireAx_Braco);  
            a_ireAx_Braco = ireAx_Braco;  
        }  
    }  
  
    if (ire.getWiimoteId() == idAnteBraco) { //AnteBraco  
        if (Math.abs(a_ireAx_AnteBraco - ireAx_AnteBraco) < LimIREAX) { // a_ireAx != ireAx  
            Universo2.interpolador.editor = 1;  
            passoireAx_AnteBraco = a_ireAx_AnteBraco - ireAx_AnteBraco;  
            executaMovimento.rotAnteBracoDireito(2, (float) Math.toRadians(passoireAx_AnteBraco));  
            gravaMovimento.setAnteBracoDireitoZ(passoireAx_AnteBraco);  
            a_ireAx_AnteBraco = ireAx_AnteBraco;  
        }  
    }  
}
```

Figura 72 - Método onIrEvent()

O método *onMotionSensingEvent()* executa semelhantemente ao método *onIrEvent()*, porém este trabalha com a coordenada x. Desta forma o valor calculado é atribuído aos métodos juntamente com o número correspondente a sua coordenada.

O método *onStatusEvent()* calcula o nível da bateria de cada controlador Wiimote normalizando seus valores, conforme ilustra a Figura 73.

```

public void onStatusEvent(StatusEvent se) {

    if ((se.getWiimoteId() == idBraco) && (BatteryLevelBraco != null)) { //Braco
        BatteryLevelBraco.setValue((int) (100 * se.getBatteryLevel()));
    }

    if ((se.getWiimoteId() == idAnteBraco) && (BatteryLevelAnteBraco != null)) { //AnteBraco
        BatteryLevelAnteBraco.setValue((int) (100 * se.getBatteryLevel()));
    }

}

```

Figura 73 - Método *onStatusEvent()*

Para o reconhecimento do dispositivo pelo sistema, além de ser conectado via *bluetooth*, há a necessidade de se pressionar o botão A do controlador.

A classe “WiimoteEventos” é utilizada para fazer o reconhecimento de cada controlador Wiimote, tendo o método *onButtonsEvent*, responsável por verificar se o botão A do dispositivo foi acionado. Caso este for acionado, o método envia o id do dispositivo para a classe “ControlesWiimote”, associando o controlador com uma das partes superiores do corpo humano. A Figura 74 ilustra o método *onButtonsEvent*.

```

public void onButtonsEvent(WiimoteButtonsEvent wbe) {
    if (wbe.getButtonsJustReleased() == 8) {
        ControlesWiimote.idWiimote = wbe.getWiimoteId();
        this.controlesWiimote.CarregaControle( wbe.getWiimoteId() );
    }
}

```

Figura 74 - Método *onButtonsEvent*

Por fim, tem-se a classe responsável por disponibilizar a interface para a utilização do dispositivo, denominada “ControlesWiimotes”. Esta estende um *JFrame* e oferece a interface para acessar todas as funcionalidades dos controladores Wiimotes.

Há quatro botões principais nesta interface, um chamado *Ativar*, um chamado *Gravar Movimento*, um chamado *Conectar WiiRemotes* e outro chamado *Resetar*, além de *scrollbars* que definem as sensibilidade das coordenadas x, y e z.

O botão *Conectar WiiRemotes* é o primeiro a ser acionado, este conecta os controladores com o sistema via *bluetooth*. Com a conexão estabelecida, o sistema associa cada controlador com uma das partes dos membros superiores, sendo associado com o pressionamento do botão A. O sistema possui uma ordem de vinculação entre os Wiimotes e as partes superiores do corpo humano, sendo: antebraço direito, braço direito, antebraço esquerdo e braço esquerdo.

O botão *Ativar* permite ao sistema iniciar a captação de eventos gerados pelos controladores, além de gravar todos os movimentos gerados pelos eventos em um vetor provisório.

O botão *Gravar Movimento* permite desativar e resetar a Luva, além de resetar o HV e de gravar todos os dados que estão no vetor provisório em um banco de dados remoto.

Por último, tem-se o botão *Resetar*, este responsável por resetar todos os dados dos controladores, do vetor provisório e do HV, além de tornar possível novamente todo o processo de geração de movimento.

APÊNDICE F - A LUVA P5-GLOVE

A luva de RV P5-Glove é uma luva de dados de baixo custo podendo ser utilizada para jogos 3D e AVs. O fabricante inicial da luva P5 era a empresa *Essential Reality*, que saiu do negócio meados de 2004. Como consequência é possível encontrar marcas novas da Luva P5 para venda por cerca de \$50 dólares. Outros tipos de luvas de dados profissionais custam muito mais (DAVISON, 2007).

O usuário movimenta a mão na frente de uma torre receptora contendo dois sensores infravermelhos. Eles detectam os *LEDs* visíveis da luva (existem no total oito) e os converte em coordenadas (x, y, z) para a posição da luva e uma orientação em termos de pitch, yaw e roll.

A luva é conectada à torre via porta de conexão (PS2), que é conectada à porta *Universal Serial Bus* (USB) do computador. A luva também tem sensores que captam a flexibilidade (dobra) dos dedos e quatro botões na parte superior, conforme ilustrado na Figura 75.



Figura 75 - Luva P5-Glove e a Torre Infravermelha (DAVISON, 2007)

Existe um grupo no *Yahoo* que possui um *link* para o *download* dos drives da luva P5 para Windows, Linux e Mac, além de trocarem dúvidas, sugestões e soluções (<http://tech.dir.groups.yahoo.com/group/p5glove/>). Estes drivers tratam de modo dual, ou seja, fornecem dados de maneira relativa e absoluta. A posição relativa é o último

deslocamento da posição da luva, enquanto que a posição absoluta é relativa a uma origem fixa no espaço.

O software original da P5 foi melhorado por *Carl Kenner*, na qual oferece melhor filtragem de dados, melhor precisão, melhor acesso à informação digital e aos *LEDs*. Com tais avanços, foi desenvolvida e disponibilizada uma API para Java, provendo as funções da Luva P5 (outras linguagens de programação também são suportadas, incluindo Delphi, Visual Basic, C e Visual C ++).

F.1 - A API JAVA

A P5DLL é um arquivo .dll responsável por disponibilizar as funções da P5 para o Windows e o arquivo CP5DLL é um arquivo .JAR responsável por disponibilizar estas funções para o Java.

A classe CP5DLL consiste de numerosas constantes públicas, variáveis e métodos e três classes internas: *P5Data*, *P5Info* e *P5State*, conforme ilustrado na Figura 76. *P5Data* pode ser ignorada, pois só é incluída para compatibilidade com versões anteriores, a sua funcionalidade foi substituída pela *P5Info* e *P5State*.

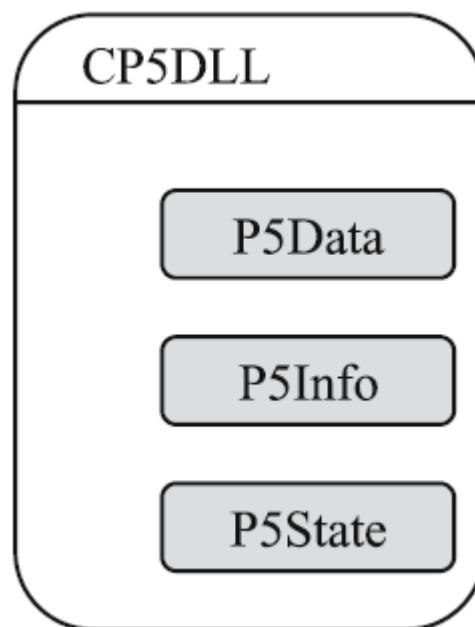


Figura 76 - A classe CP5DLL (DAVISON, 2007)

Os mais de quarenta métodos da classe pública CP5DLL de nível superior são usados para inicializar e configurar a luva conectada ao PC. Por exemplo, é possível ajustar a quantidade de dados de filtragem. As variáveis públicas em *P5State* expõe a atual posição

da luva, a rotação, os botões, valores e os *LEDs*. Há um método chamado *update ()* que deve ser chamado sempre que os valores precisam ser atualizados. A maioria das vinte e cinco variáveis públicas em *P5Info* asseguram detalhes do dispositivo, como o fabricante nome e os números de versão.

F.2 – Examinando os Dados da Luva

Existe uma aplicação que permite examinar os dados da luva P5, posições, orientações, botões acionados, *LEDs* ligados e informações de frames gerados pelo dispositivo, conforme ilustrado na Figura 77.

Section	Parameter	Value
Filtered Hand Info	X,Y,Z:	-46, 85, 787
	Pitch,Yaw,Roll:	-187, -77, -18
Finger Bends	Thumb:	455
	Index:	574
	Middle:	550
	Ring:	613
	Small:	720
Frames Info.	Frame No.:	570
	Frame Rate:	31
Buttons	A	☐
	B	☐
	C	☐
	D	☐
LEDs	0	☐ []
	1	☒ 25, 124, 701
	2	☐ []
	3	☒ 517, 1947, -5533
	4	☒ -379, -496, 1550
	5	☐ []
	6	☐ []
	7	☒ -234, 39, 476

Figura 77 - Dados da Luva P5-Glove (DAVISON, 2007)

Os filtros da P5 suavizam os dados causados por falhas da torre de infravermelho detectados por alguns *LEDs*. Os números reportados é a média da corrente posição dividida pela corrente rotação.

Os números (x, y e z) são valores absolutos relativos a eixos de coordenadas mostrados na Figura 78. A origem é centralizada perto da base da torre.

A orientação da luva é dada em termos de *pitch*, *yaw* e *roll*, que são ilustrados na Figura 79.

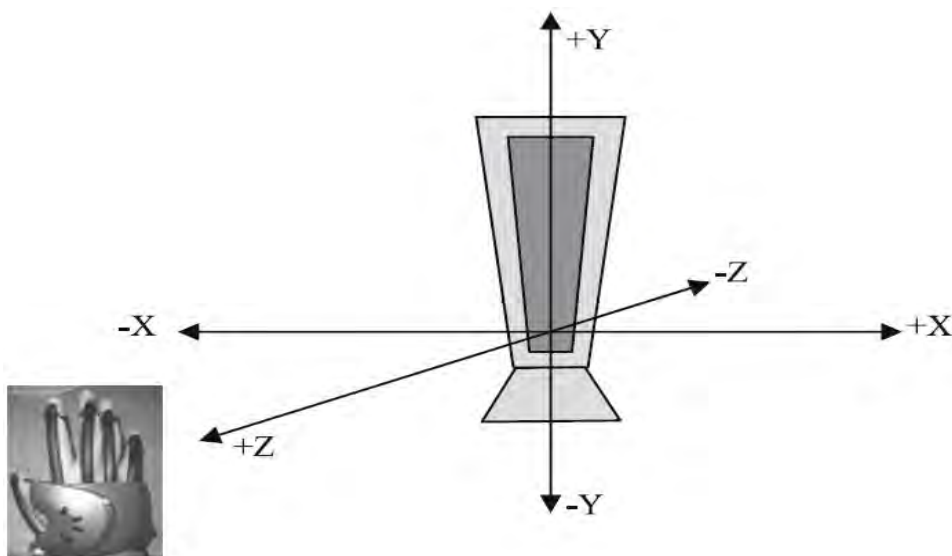


Figura 78 - Eixos de coordenadas da Luva P5-Glove (DAVISON, 2007)

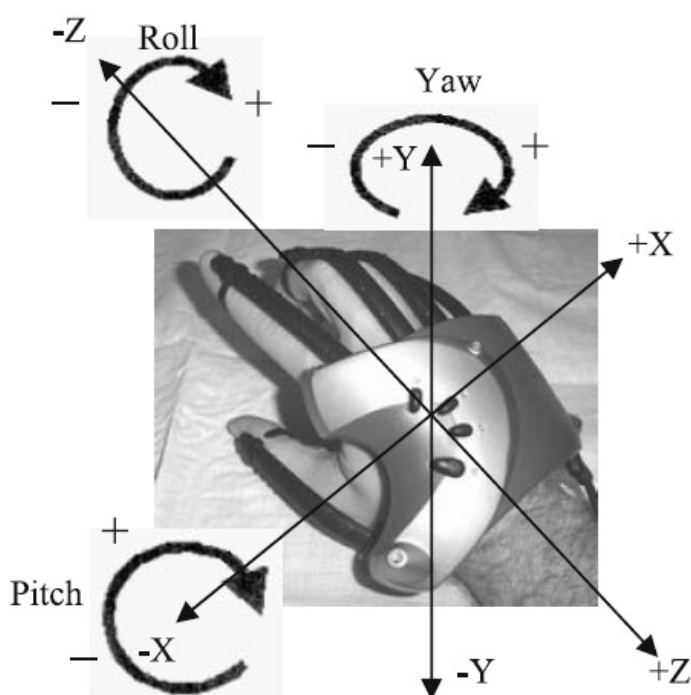


Figura 79 - A orientação da Luva P5-Glove – Pitch, yaw e roll (DAVISON, 2007)

Os dados da dobra dos dedos dependem da configuração da calibração da luva, na qual é calibrada via painel de controle do Windows, sendo acoplado na instalação do dispositivo. Na luva, os dedos estendidos tipicamente mostram um valor aproximado de mil, enquanto os dedos dobrados registram em torno de quatrocentos.

Há quatro botões no topo da luva, A, B, C e D. Quando o botão D é pressionado a luva automaticamente desliga, assim a mudança dos botões não é detectada pelo JAR CP5DLL e não mostrada na aplicação *ShowGlove*.

Existem oito *LEDs* na luva numerados, conforme mostrado na Figura 80.

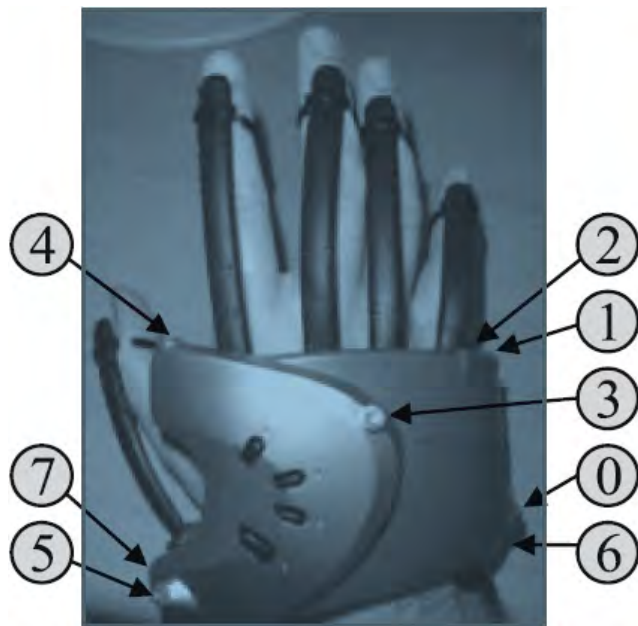


Figura 80 - LEDs da Luva P5-Glove (DAVISON, 2007)

A reabilitação da posição e orientação dos dados depende da visibilidade da torre em relação aos *LEDs*. Um dos problemas é que os dedos dos usuários podem bloquear os *LEDs* do ponto de vista. Uma simples solução é o usuário manter seus dedos tortos e dobrar seu punho bem como as costas da mão direcionada levemente para a torre. O treinamento da luva é despercebido tão quanto o treinamento do uso do mouse.

F.3 – Módulo Editor

A luva P5-Glove é o primeiro dispositivo a ser descrito. Esta garante o movimento da mão direita juntamente com seus cinco dedos apenas na coordenada x, permitindo ao HV a possibilidade do dobrando dos dedos.

Vale ressaltar que todas as classes pertencentes ao monitoramento da Luva (inicialização, captação de dados e encerramento), importam uma API disponível para Java, denominada “CP5DLL.jar”.

A classe “Glove” é a responsável por captar todos os eventos gerados pela Luva. Seu construtor cria uma instância do objeto CP5DLL e o inicializa, conforme ilustra na Figura 81.

```

public Glove()
{
    gloves = new CP5DLL();

    // inicializa a Luva
    if (!gloves.P5_Init()) {
        System.out.println("P5 Initialization failed");
        System.exit(1);
    }

    gloves.P5_SetForwardZ(-1); // a posição z em direção ao usuário
    gloves.P5_SetMouseState(-1, false); // desativa o mouse

    // Verifica a quantidade de Luva detectada
    int numGloves = gloves.P5_GetCount();
    if (numGloves > 1) {
        System.out.println("Too many gloves detected: " + numGloves);
        System.exit(1);
    }

    printGloveInfo();
    gloveState = gloves.state[0]; // armazena uma referência para o estado da luva
} // fecha a Glove()

```

Figura 81 - Construtor da classe "Glove"

O objeto da classe pode acessar luvas múltiplas, porém é utilizado apenas uma. Assim, uma referência para o estado único da luva é armazenado em *gloveState*, e isso é utilizado posteriormente para atualizar e acessar os dados da luva. O método *printGloveInfo()* emprega a classe *CP5DLL.Info* para imprimir os detalhes sobre o dispositivos. A Figura 82 ilustra o fragmento do código de uma abordagem geral.

```

CP5DLL.P5Info gloveInfo = gloves.info[0];
System.out.println(
    "Vendor ID:" + gloveInfo.vendorID +
    "; Product ID:" + gloveInfo.productID +
    "; Version: " + gloveInfo.version);

```

Figura 82 - Detalhes da Luva

Para acessar e atualizar os dados do objeto Luva, há o método *update()* e os métodos *gets()*. O método *update()* chama o método respectivo da classe "CP5DLL.P5State". Os métodos *gets()* leem as variáveis públicas expostas pela classe "CP5DLL". Por instância, os métodos *getFPos()* e *getFOrient()* retornam as posições filtradas da luva, enquanto o método *getFingers()*, retorna um vetor com os valores de dobra dos cinco dedos. A Figura 83 ilustra os métodos.

```

public void update()
{ gloveState.update(); }

public float[] getFPos()
// Filtra as coordenadas x, y e z
{ return new float[]{gloveState.filterPos[0],
                    gloveState.filterPos[1],
                    gloveState.filterPos[2]}; }

public float[] getFOrient()
// filtra as orientações Pitch, Yaw e Roll
{ return new float[] {gloveState.filterPitch,
                    gloveState.filterYaw,
                    gloveState.filterRoll}; }

public short[] getFingers()
// Pega os valores de dobra dos cinco dedos
{ return gloveState.fingerAbsolute; }

```

Figura 83 - Métodos update() e gets()-

Por fim, tem-se o fechamento da Luva, na qual o método *close()* restaura a configuração do modo mouse e fecha a conexão com a Luva P5. A Figura 84 ilustra o método.

```

public void close()
{
    gloves.P5_RestoreMouse(-1); // restaura o modo mouse
    gloves.P5_Close();
}

```

Figura 84 - Método close()

Há uma classe responsável por ficar atualizando as variáveis globais que recebem todos os valores gerados pela Luva em um intervalo de tempo, transformando-os em valores decimais e logo em ângulos. Assim, estes dados são enviados as classes responsáveis pela execução e gravação.

Inicialmente é instanciado um objeto da Luva e um temporizador que fica atualizando os dados da luva a cada 200ms. O temporizador é inicializado através do método *startPolling()*, na qual possui um *ActionListener* que dispara uma atualização de dados. Os valores devem ser atualizados na *thread* deste segmento de evento, por utilizar um temporizador *javax.swing*, responsável por executar o manipulador de ações (*pollPerformer*), conforme ilustrado na Figura 85.

```

public void InicializaLuva() {
    glove = new Glove();
    df = new DecimalFormat("0");
    startPolling();
}
private void startPolling() {

    ActionListener pollPerformer = new ActionListener() {

        public void actionPerformed(ActionEvent e) {
            glove.update();
            val = glove.getFOrient();
            pitch = Float.parseFloat(df.format(val[0]));
            Pitch = RegularValorPitch(pitch);
            dedos = glove.getFingers();
            for (int i = 0; i < 5; i++) {
                Polegar = RegulaValorPolegar(dedos[0]);
                Indicador = RegulaValorIndicador_Mindinho(dedos[1]);
                Medio = RegulaValorMedio(dedos[2]);
                Anelar = RegulaValorAnelar(dedos[3]);
                Mindinho = RegulaValorIndicador_Mindinho(dedos[4]);
            }

            :

            pollTimer = new Timer(GLOVE_DELAY, pollPerformer);
            pollTimer.start();
        }
    };
}

```

Figura 85 - Método InicializaLuva() e parte do Método startPolling()

Inicialmente é atualizado todos os dados através do método *update()* da classe “Glove”, para que posteriormente os dados sejam tratados, transformados em decimais e atribuídos às variáveis globais.

Após a conversão dos dados em decimais, estes são convertidos em ângulos e são enviados para execução e possível gravação. Para esta conversão é utilizado o teorema de tales (HARUNA, 2000). A Figura 86 ilustra os cálculos necessários para encontrar os ângulos, segundo o teorema de tales, dos dedos indicador e mindinho, por exemplo.

$$\begin{array}{c}
 90 \quad | \quad 500 \\
 Y \quad | \quad X \\
 0 \quad | \quad 1100
 \end{array}$$

$$\frac{Y-0}{90-0} = \frac{X-1100}{500-1100}$$

$$\frac{Y-0}{90} = \frac{X-1100}{600}$$

$$-600y = 90x - 99000$$

$$600y = 99000 - 90x$$

$$y = \frac{99000 - 90x}{600}$$

Figura 86 - Teorema de Tales - Dedos Indicador e Mindinho

A Figura 87 ilustra como este cálculo foi aplicado ao HV, existindo mais cálculos para o restante dos dedos da mão.

```
public float RegulaValorIndicador_Mindinho(float valor) {  
    if (valor <= 500) {  
        return 90;  
    }  
    if (valor >= 1100) {  
        return 0;  
    } else {  
        return (99000 - (90 * valor)) / 600;  
    }  
}
```

Figura 87 - Método *RegulaValorIndicador_Mindinho()*

Para calcular exatamente a nova posição de um determinado membro do HV, há a necessidade de se subtrair o valor atual pelo valor antigo, para que não se somam os valores da nova posição. Desta maneira, cria-se uma variável para armazenar o novo valor, antes que esta seja atribuída aos métodos responsáveis pela execução e armazenamento dos movimentos. Os atuais valores são atribuídos aos antigos para uma nova execução, conforme ilustra a Figura 88.

```
Universo2.interpolador.editor = 1;  
passo_Polegar = A_Polegar - Polegar;  
passo_Indicador = A_Indicador - Indicador;  
passo_Medio = A_Medio - Medio;  
passo_Anelar = A_Anelar - Anelar;  
passo_Mindinho = A_Mindinho - Mindinho;  
  
A_Polegar = Polegar;  
A_Indicador = Indicador;  
A_Medio = Medio;  
A_Anelar = Anelar;  
A_Mindinho = Mindinho;
```

Figura 88 - Operação de Subtração e Atualização de Valores

Após todas estas operações já é possível invocar os métodos de execução e gravação dos movimentos, passando como parâmetro o eixo e o valor do ângulo em radianos. A Figura 89 ilustra estas operações.

```

executaMovimento.rotPolegar(0, (float) Math.toRadians(-passo_Polegar));
executaMovimento.rotPolegar2(0, (float) Math.toRadians(-passo_Polegar));
executaMovimento.rotIndicador(2, (float) Math.toRadians(-passo_Indicador));
executaMovimento.rotIndicador2(2, (float) Math.toRadians(-passo_Indicador));
executaMovimento.rotIndicador3(2, (float) Math.toRadians(-passo_Indicador));
executaMovimento.rotMedio(2, (float) Math.toRadians(-passo_Medio));
executaMovimento.rotMedio2(2, (float) Math.toRadians(-passo_Medio));
executaMovimento.rotMedio3(2, (float) Math.toRadians(-passo_Medio));
executaMovimento.rotAnelar(2, (float) Math.toRadians(-passo_Anelar));
executaMovimento.rotAnelar2(2, (float) Math.toRadians(-passo_Anelar));
executaMovimento.rotAnelar3(2, (float) Math.toRadians(-passo_Anelar));
executaMovimento.rotMindinho(2, (float) Math.toRadians(-passo_Mindinho));
executaMovimento.rotMindinho2(2, (float) Math.toRadians(-passo_Mindinho));
executaMovimento.rotMindinho3(2, (float) Math.toRadians(-passo_Mindinho));

gravaMovimento.setPolegarX(-passo_Polegar);
gravaMovimento.setPolegar2X(-passo_Polegar);
gravaMovimento.setIndicadorZ(-passo_Indicador);
gravaMovimento.setIndicador2Z(-passo_Indicador);
gravaMovimento.setIndicador3Z(-passo_Indicador);
gravaMovimento.setMedioZ(-passo_Medio);
gravaMovimento.setMedio2Z(-passo_Medio);
gravaMovimento.setMedio3Z(-passo_Medio);
gravaMovimento.setAnelarZ(-passo_Anelar);
gravaMovimento.setAnelar2Z(-passo_Anelar);
gravaMovimento.setAnelar3Z(-passo_Anelar);
gravaMovimento.setMindinhoZ(-passo_Mindinho);
gravaMovimento.setMindinho2Z(-passo_Mindinho);
gravaMovimento.setMindinho3Z(-passo_Mindinho);

```

Figura 89 - Método de Execução e de Gravação de Movimentos

Vale ressaltar que na execução do movimento do pulso, foi definido uma condição (*if()*) antes da realização dos mesmos procedimentos dos dedos da luva, pois o movimento é muito sensível, havendo a necessidade da limitação do movimento. A Figura 90 ilustra o trecho de código do movimento do pulso.

```

if (Math.abs(A_Pitch - Pitch) > Lim_Mao) {
    passo_Pitch = A_Pitch - Pitch;
    executaMovimento.rotMaoDireita(2, (float) Math.toRadians(passo_Pitch));
    gravaMovimento.setMaodireitaZ(passo_Pitch);
    A_Pitch = Pitch;
}

```

Figura 90 - Movimento do Pulso

Por fim, tem-se a classe responsável por disponibilizar a interface para a utilização do dispositivo, denominada “Conexao_Luva”. Esta estende um *JFrame* e oferece a interface para acessar todas as funcionalidades da Luva P5

Há três botões principais nesta interface, um chamado *Ativar*, um chamado *Gravar* e outro chamado *Resetar*.

O botão *Ativar* é o primeiro a ser acionado, este permite inicializar os valores da Luva e disparar a atualização dos dados, além de gravar todos os movimentos gerados em um vetor provisório e habilitar os demais botões da interface.

O botão *Gravar* permite desativar e resetar a Luva, além de resetar o HV e de gravar todos os dados que estão no vetor provisório em um banco de dados remoto.

Por último, tem-se o botão *Resetar*, este responsável por resetar todos os dados da Luva, do vetor provisório e do HV, além de tornar possível novamente todo o processo de geração de movimento.