FUNDAÇÃO DE ENSINO EURÍPIDES DA ROCHA SOARES
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA – UNIVEM
PROGRAMA DE MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

AVALIAÇÃO DAS BIBLIOTECAS DE RECONHECIMENTO E SÍNTESE DE FALA EM AMBIENTES VIRTUAIS

Eduardo Filgueiras Damasceno

Marília
2005

EDUARDO FILGUEIRAS DAMASCENO

AVALIAÇÃO DAS BIBLIOTECAS DE RECONHECIMENTO E SÍNTESE DE FALA EM AMBIENTES VIRTUAIS

Dissertação apresentada ao Programa de Mestrado do Centro Universitário Eurípides de Marília, mantido pela Fundação de Ensino Eurípides Soares da Rocha, para obtenção do Título de Mestre em Ciência da Computação.

Orientador:
Prof. Dr. José Remo Ferreira Brega

Marília
2005

DAMASCENO, Eduardo Filgueiras

Avaliação Das Bibliotecas De Reconhecimento E Síntese de Fala em Ambientes Virtuais / Eduardo Filgueiras Damasceno; orientador: José Remo Ferreira Brega.

Marília, SP: [s.n.], 2005.

121 f.

Dissertação (Mestrado em Ciência da Computação) - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha.

1.Reconhecimento de Fala 2. Realidade Virtual 3.Ambiente Virtual 4 Java 3D 5. WorldToolKit 6 Java Speech 7 Microsoft Speech

CDD: 006

EDUARDO FILGUEIRAS DAMASCENO

AVALIAÇÃO DAS BIBLIOTECAS DE RECONHECIMENTO E SÍNTESE DE FALA EM AMBIENTES VIRTUAIS

Banca examinadora da dissertação apresentada ao Programa de Mestrado da UNIVEM / F.E.E.S.R., para obtenção do Título de Mestre em Ciência da Computação.

Resultado: _____

ORIENTADOR: Prof. Dr. José Remo Ferreira Brega

1º EXAMINADOR:_____

2º EXAMINADOR:_____

Marília, ___ de_____ de 2005.

Dedico este trabalho, com todo amor do meu coração, a minha família, à minha compreensiva esposa e amada filha.

AGRADECIMENTOS

Agradeço,

Primeiramente a Deus, que sempre esteve ao meu lado fortalecendo-me para que eu jamais desanimasse na concretização deste trabalho;

à minha esposa Tatiane, pelo carinho, amor e apoio incondicional;

à minha filha que desde da barriga viu o papai dela a estudar;

a todos os meus amigos do curso de mestrado principalmente aos colegas de viagem Júnior, (José Barbosa) , Montanha (Fábio Ramos) e Leila;

ao Prof. Dr. José Remo Ferreira Brega, pelos preciosos ensinamentos, por sua compreensão frente às minhas dificuldades, por sua amizade e entusiasmo;

à todos professores do Programa de Mestrado, por todo incentivo e colaboração;

à secretária Elizabete (Dona Beth), pela alegria e simpatia com que nos recebe;

à vida, pela oportunidade diária de me tornar uma pessoa melhor e ser feliz.

DAMASCENO, Eduardo Filgueiras. **Avaliação das Bibliotecas de Reconhecimento e Síntese de Fala em Ambientes Virtuais**. 2005. 121 f. Dissertação (Mestrado em Ciência da Computação) - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2005.

RESUMO

Com a crescente inovação da tecnologia multimodal para interface de Ambientes Virtuais a necessidade de ter um ambiente leve e de fácil operação do usuário faz com que os recursos de fala sejam incorporados a aplicação, mas a perda de desempenho quando esta tecnologia é utilizada faz com que o usuário perca a interação e o envolvimento. Com esta premissa o objetivo deste trabalho é elucidar algumas práticas e técnicas de programação para Ambientes Virtuais fazendo a comparação entre duas Bibliotecas de geração de ambiente virtuais: Java 3D e a WorldToolKit, e como estas bibliotecas se comportam em conjunto com as bibliotecas Java Speech API e Microsoft Speech API, ambas para reconhecimento e síntese de fala.

Palavras-chave: Reconhecimento de Fala, Realidade Virtual, Ambiente Virtual, Java 3D, WorldToolKit, Java Speech, Microsoft Speech

DAMASCENO, Eduardo Filgueiras. **Avaliação das Bibliotecas de Reconhecimento e Síntese de Fala em Ambientes Virtuais**. 2005. 121 f. Dissertação (Mestrado em Ciência da Computação) - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2005.

ABSTRACT

With the increasing innovation of the multimodal technology for interface of virtual environments there is a need to have a lightweight environment, which is for the user to lead the speech resources in the application. We have observed here is some less in performance when using this technology, which leads to loss of interaction a loose involvement by the user. The Objective of this work is to issues some practice programming techniques in the light of above environment through the comparison two 3D Libraries for virtual environment developed in Java 3D and WorldToolKit, as well as speech recognition and synthesis Java Speech API and Microsoft Speech API.

Keywords: Speech Recognition, Virtual Reality, Virtual Environment., Java 3D, WorldToolKit , Java Speech , Microsoft Speech.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ANN	-	Artificial Neural Networks
API	-	Application Programming Interface
ARPA	-	Advanced Research Projects Agency
ASR	-	Automatic Speech Recognition
AV	-	Ambiente Virtual
AWT	-	Abstract Window ToolKit
BPS	-	Bits Per Second
CPF	-	Cadastro De Pessoa Física
CSLU	-	Center For Spoken Language Understanding
DOF	-	Degree Of Freedom
DSP	-	Digital Signal Processing
DLL	-	Dynamic Library Linkage
FFT	-	Windowed Fast Fourier Transform
GUI	-	Graphical User Interface
HMM	-	Hidden Markov Models
JSAPI	-	Java Speech Application Programming Interface
JSGF	-	Java Speech Grammar Format
JSML	-	Java Speech Markup Language
MIPS	-	Milhões de Instruções por Segundo
MS	-	Milisegundos
MSAA	-	Microsoft Active Accessibility
MS-DOS		Microsoft Disk Operating System
MS-DCOM	-	Microsoft Distributed Component Object Model
MS-SAPI	-	Microsoft Speech Application Programming Interface
MS-SDK		Microsoft Speech Software Development Kit
MVC++	-	Microsoft Visual C++
NLP	-	Natural Language Processing
OCR	-	Optical Character Recognition
OLE	-	Object Linked and Emblebed
PCM	-	Pulse Code Modulation
RAF	-	Reconhecimento Automático da Fala
RG	-	Registro Geral
RNA	-	Redes Neurais Artificiais
RV	-	Realidade Virtual
SDK	-	Software Development Kit
SOM	-	Self-Organizing Map
SO	-	Sistema Operacional
TTS	-	Text-To-Speech
UML	-	Unified Modeling Language
URA	-	Unidade de Resposta Audível
VMM	-	Visible Markov Models
VRS	-	Voice Response Systems
XML	-	Extended Markup Language
WTK	-	WorldToolKit

SUMÁRIO

INTRODUÇÃO

A tecnologia de informação alcança patamares até então somente apenas sonhados pelo homem, principalmente a níveis de acessibilidade da informação. As interfaces homem-computador estão cada vez mais complexas e robustas, necessitando de uma investida em novas tecnologias de acesso. Uma das tecnologias de grande impacto por parte dos desenvolvedores de ambientes computacionais de alto desempenho é o uso de interfaces de voz.

Segundo Apaydin (2002), a nova geração de interfaces homem-computador será a de interfaces de fácil aprendizado e de alta acessibilidade destacando o uso das interfaces de voz, das quais seriam o equivalente a um terço de todas as interfaces criadas para os ambientes computacionais de alta complexidade. Isto por três motivos principais: primeiro a facilidade no aprendizado do ambiente através do uso de voz conforme Zelter e Johnson (1994), em segundo pela liberdade que o usuário pode ter, conforme destaca Tatham (2002), que descreve que a geração e a utilização de mensagens ou comandos falados e interpretados pelo computador, ou seja, síntese e reconhecimento da voz humana, permitindo que os olhos ou mãos estejam livres para a realização de outras tarefas e por final, é uma forma mais intuitiva de comunicação, pois o ser humano já utiliza a voz para se comunicar.

O alto grau de complexidade imposta por tarefas em diversas áreas da ciência exige uma maior atenção do usuário bem como sua atenção manipulando diversas interfaces de comunicação ao mesmo tempo.

Nestas interfaces, são apresentadas informações, dados complexos, de forma que cada vez mais o grau de interatividade seja diretamente proporcional ao volume de informações apresentadas ao usuário.

Uma interface tanto define as estratégias para a realização da tarefa, como conduz, orienta, recepciona, alerta, ajuda e responde ao usuário durante suas interações.

Entretanto mesmo com acesso facilitado ao conjunto de informações o usuário poderá sentir uma ansiedade ou frustração no uso das interfaces que utilizam recursos de imagens tridimensionais ou de informações complexas, como afirma Pressmam (2002).

Com a utilização da tecnologia de fala favorecendo a usabilidade dos sistemas, os recursos alocados para o processamento, consulta à informação, e visualização da informação, acabam se atenuando em função da alocação do mecanismo de reconhecimento e síntese de fala que consomem uma boa parte destes recursos computacionais.

Quando utilizam se bibliotecas de reconhecimento e síntese de fala em Ambientes Virtuais (AV), existe um consumo maior de recursos computacionais (memória, processamento e espaço em disco), isto porquê além da alocação dos processos citados há a necessidade de incorporara-los ao ambiente, que, por si só, exige um grande poder de processamento.

Sendo esta a premissa deste trabalho que faz um estudo das técnicas de desenvolvimento de ambientes computacionais que utilizem recursos de fala e que possuam a menor latência de processamento.

Neste trabalho foi desenvolvido dois AV controlados por fala (comando e controle), utilizando duas bibliotecas de programação de serviços de fala concorrentes, a Java Speech API e o Microsoft Speech API, onde os ambiente são criados com as bibliotecas gráficas 3D java3D e WorldToolKit.

Apresenta ainda informações que poderão ser consideradas como conceitos de relevância para a adoção de uma tecnologia de reconhecimento de fala em ambientes virtuais.

A mesma é composta por oito capítulos onde após esta breve introdução o Capítulo 1 faz uma elucidação das definições sobre a tecnologia de fala.

O Capítulo 2 traz a tona os métodos de reconhecimento de fala como também um conjunto de abordagens e técnicas para o uso da tecnologia.

O Capítulo 3 descreve alguns mecanismos de reconhecimento de fala e faz uma comparação entre o Microsoft Speech API e o Java Speech API.

O Capítulo 4 faz menção a Realidade Virtual, Ambientes Virtuais e Avatares Humanóides com os quais é possível a aplicação a tecnologia de fala.

O Capítulo 5 é responsável por mostrar as bibliotecas de programação para reconhecimento de fala, sendo analisadas duas bibliotecas: Java Speech e Microsoft Speech.

O Capítulo 6 denota algumas considerações no uso das bibliotecas de programação descritas no Capítulo 5.

O Capítulo 7 elucida como foram criados os ambientes com os recursos da tecnologia de fala e também aborda questões das linguagens de programação escolhidas bem como a

técnica de programação orientada a objetos Java e a programação para Windows através de mensagens e suas diferenças quanto a processamento dos dados e recursos do sistema.

Ao término desta, são apresentadas conclusões abrangendo e comentando os resultados obtidos quanto a processamento, latência, que são relevantes para atenuar a frustração do usuário ao operar um sistema com recursos computacionais diversificados. Posto juntamente ao encerramento aludindo aos trabalhos futuros.

E, por fim, estão as Referências Bibliográficas que fizeram parte desta dissertação.

1. DEFINIÇÕES

O uso de interfaces de fala em sistemas computacionais está cada vez mais em voga, sendo uma promessa para a acessibilidade do usuário. A utilização desta tecnologia segue uma tendência natural que visa a tornar a interação homem-máquina mais direta e efetiva.

Por meio da tecnologia de comunicação por fala, ou a condução do sistema por interface de fala, vem à tona uma questão antiga de certa relevância: a de que o computador possa compreender o que o ser humano deseja, sendo assim, segundo Lévy (1996), a superação da barreira de comunicação homem – computador depende diretamente da forma de abordagem desta comunicação, podendo esta estar superada ou não dependendo do contexto e da linguagem de controle utilizada.

As interfaces homem-computador estão cada vez mais complexas e robustas, necessitando cada vez mais de melhores dispositivos de hardware e uma integração de software que seja de igual teor de inovação. Para estas interfaces deverão ser criados modos de interação intuitivos e de fácil operação, já que por meio do recurso de fala o usuário poderá ter maior liberdade para execução de outras tarefas que exijam a manipulação de entradas de forma convencional e com o uso de sistemas de síntese de fala o usuário poderá receber informações de forma direta e objetiva.

Sabendo que os seres humanos utilizam-se principalmente da linguagem natural (linguagem falada) para comunicação, faz-se necessária uma interface condizente com a aplicação a ser desenvolvida e que esta aplicação entenda o que seu usuário (locutor) realiza.

Portanto o processo comunicação é baseado no compartilhamento de um modelo mental, ou seja, compartilhar uma idéia, e torná-lo comum entre o emissor e o receptor, conforme afirma Chomsky (1971). Isto é realizado por meio de uma linguagem, a qual é gerada por uma gramática de signos lingüísticos que por meio de convenção lingüística identificam os objetos do mundo real.

Segundo Simões (1999) um dos objetivos do reconhecimento de fala é construir formas (sistemas ou dispositivos) que transcrevam a fala em texto automaticamente e que o computador possa trabalhar com estas informações transcritas. Mas as aplicações de interfaces de fala vão além de transcrever um sinal analógico capturado (fala) para um sinal digital (texto).

Segundo Russel e Norvig (2004), a comunicação é a troca intencional de informações provocada pela produção e percepção de sinais extraídos de um sistema compartilhado de sinais convencionais, que no caso é a voz, e, portanto pode-se definir que um sistema de serviços de fala contém tanto o reconhecimento de fala como a síntese de fala.

A idéia principal nos sistemas de reconhecimento e síntese de fala atualmente sendo projetados é que estes possuam uma forma de compartilhar a idéia entre o receptor máquina e o emissor humano, caracterizando uma efetiva interface de comunicação.

Esta efetiva interface de comunicação está presente em todas as culturas, conforme afirma Chomsky (1971), e sua utilização permite a comunicação de forma mais natural e eficiente para o ser humano, que na sua maioria começa a esboçar o sentido da fala e da

audição, com certa compreensão, por volta dos 2 anos de idade. Entretanto neste limiar do aprendizado o ser humano é treinado a ouvir e expressar simploriamente alguns vocábulos, com os quais através da repetição e da comparação visual, táctil e auditiva, acaba fixando a correta pronuncia das palavras.

A mesma situação acontece com as máquinas, entretanto para que elas possam reconhecer e compreender uma linguagem falada devem se basear num conjunto de regras e vocábulos pré-definidos os quais denomina-se de gramática.

Como já mencionado, a interação com o computador por meio da fala fornece ao usuário uma maior liberdade, como por exemplo, através da geração de mensagens faladas, a qual permite que os olhos estejam livres para a realização de outras tarefas.

O reconhecimento da fala do usuário elimina a necessidade de digitação, tornando a execução de tarefas mais ágil e sem exigir, obrigatoriamente, a presença do mesmo junto ao terminal.

De acordo com Furui (1989), a compreensão da linguagem falada é a união de duas áreas correlatas: a) reconhecimento da fala; b) o processamento de linguagem natural.

O reconhecimento da fala é realizado através da conversão de uma entrada de dados via áudio convertidos em palavras que poderão ser utilizadas numa aplicação qualquer, ordinariamente utilizando técnicas de processamento digital de sinais (*Digital Signal Processing* – DSP). Isto é executado apenas para um vocabulário pré-definido para o reconhecimento correto das palavras pronunciadas.

Já o Processamento de Linguagem Natural (*Natural Language Processing* - NLP), segundo Furui (1989), utiliza-se de dados como, por exemplo: palavras pré-definidas e regras gramaticais, para dar um significado ao que foi falado.

As duas tecnologias em consonância auxiliam e flexibilizam o contexto da tecnologia de reconhecimento e síntese de fala.

Apesar dos investimentos na pesquisa de tecnologia de fala nos últimos anos, a síntese de fala e as tecnologias de reconhecimento de fala ainda possuem limitações significantes, tais como a dependência do hardware utilizado (microfone) e a técnica de processamento de sinais (mecanismo de reconhecimento).

Segundo Guilhoto (2002), freqüentemente a expressão "reconhecimento de voz" é utilizada com vários sentidos, que na verdade, refere-se a tecnologias distintas e conforme afirma Jurafsky e Martin (2000), o processamento de sinais de fala pode ser aplicado em quatro áreas principais: comando e controle por voz, reconhecimento de fala natural, síntese de fala e autenticação de voz, como mostra a ilustração da Figura 1.

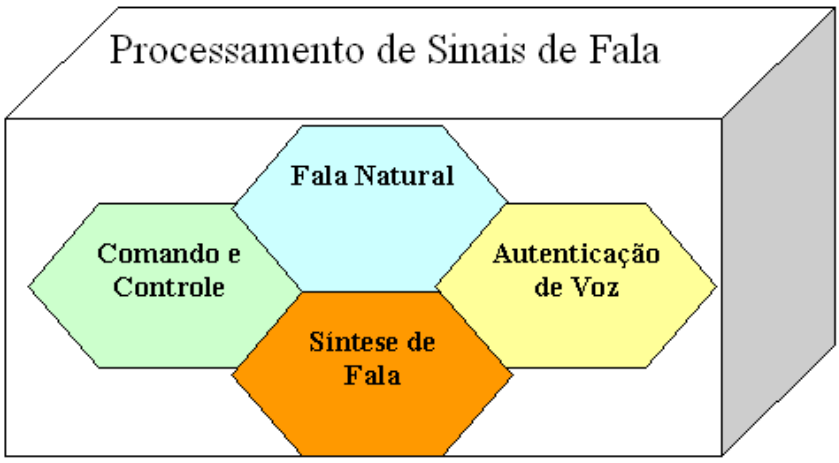


Figura 1 - Principais áreas de aplicação do processamento de voz

O reconhecimento de palavras utilizado nos sistemas de comando e controle de voz caracteriza-se por processar apenas um pequeno trecho de fala, de modo a identificar o tipo de ação que o sistema deve realizar.

Este processamento torna-se simplificado, uma vez que o sistema já sabe de antemão quais os comandos disponíveis para o usuário. Este é o caso de centrais de atendimento telefônico tais como a URA (Unidade de Resposta Audível), onde o usuário pode usar a voz ao invés de pressionar botões, de acordo com Franco (1997).

O reconhecimento de fala natural (ou fala contínua) envolve uma ou mais palavras combinadas em frases que tenham um sentido semântico, onde a fala reconhecida é convertida em texto.

O tipo de aplicação mais comum para essa tecnologia é o ditado de documentos, para uso em processadores de texto, escrita de e-mails, etc. Um mecanismo representante desta tecnologia dado por ZDNet (2001), que é o sistema ViaVoice da IBM que reconhece o ditado (fala contínua) e converte em um texto. Outro sistema de reconhecimento de fala é o Microsoft Speech descrito em Microsoft (2001), que utiliza processos estocásticos para analisar o que foi pronunciado.

Para o controle dos mecanismos de reconhecimento e síntese de fala são utilizadas bibliotecas de programação, que são denominadas de API – *Application Programming Interface,* que são responsáveis pela comunicação, ou seja, a interface de programação e acesso aos recursos do programa de fala.

A síntese de fala, por sua vez é o processo contrário ao do reconhecimento. O sintetizador recebe um texto na forma digital e transforma-o em ondas sonoras, ou em outras palavras, faz uma leitura em voz alta.

Segundo Ynoguti (1999), um programa de síntese de fala é útil nas situações em que o usuário não pode desviar a atenção para ler algo ou não tem acesso ao texto escrito, seja porque a informação está distante ou porque o usuário tem alguma deficiência visual.

A autenticação da voz baseia-se no fato de que a voz é única para cada pessoa e pode ser utilizada para identificar alguém. Os sistemas de autenticação podem ser aplicados para permitir o acesso de uma pessoa a uma determinada função no sistema.

A autenticação da voz é usada em diversas aplicações como sinônimo de entrada de dados via áudio, entretanto deve-se atentar ao fato de que a tecnologia de voz vai além de apenas uma entrada de dados, mas sim uma distinção entre duas ou mais vozes para a identificação do orador.

Para a compreensão dos conceitos relacionados ao reconhecimento e síntese de fala apresentada nesta dissertação é necessária uma breve elucidação dos termos que seguem abaixo.

1.1 Voz

Segundo Ferreira (2001), voz é um som ou conjunto de sons emitidos pelo aparelho fonador humano que pode expressar uma atividade ou uma sensação e até mesmo mostrar o estado psicológico de uma pessoa.

Se a informação deste som contiver um significado que possa ser expresso em uma palavra ou um conjunto de palavras, pode-se entender que este som é uma fala.

1.2 Fala

A fala é a representação da semântica dos conjuntos de sons da voz de um ser humano com o intuito de se expressar. Segundo Simões (1999), o estudo da fala se identifica com o estudo das técnicas de compreensão da informação com a função de: (1) armazenar o que foi pronunciado; (2) transmitir o sinal de voz por meio de canais cuja a banda é mais estreita do que a que seria necessária e; (3) extrair parâmetros ou informações da fala para que se permita a manipulação do sinal para os mais diversificados fins.

Pode-se entender que o reconhecimento de fala é um mecanismo que analisa o sinal de voz e procura determinar a seqüência de palavras correspondentes àquele sinal a fim de transcrever o que foi pronunciado de forma clara e concisa.

1.3 Palavra

A palavra é descrita por Ferreira (2001), como sendo a unidade mínima com som e significado que pode, sozinha, constituir enunciado de forma livre.

Segundo Chomsky (1971), palavra é a unidade pertencente a uma das grandes classes gramaticais, como por exemplo: substantivo, verbo, adjetivo, advérbio, abstraídas as diferentes realizações (marcas flexionais) que ela possa apresentar.

E a representação de uma palavra pode se dar por meio da união de um ou mais fonemas.

1.4 Fonema

Um fonema é a unidade mínima distintiva no sistema sonoro de uma língua das quais as palavras são compostas. O fonema é uma unidade abstrata, que restringe sua área de

atuação a um domínio psicológico e não a um físico, e que, portanto, não apresenta características acústicas.

Segundo Simões (1999), o que diferencia um fonema de outro é o seu papel distintivo dentro da língua. Por exemplo, na língua portuguesa, */p/* e */b/* representam fonemas diferentes, pois é a partir deles que se pode distinguir palavras como */pasta/* e */basta/*.

De acordo com Hugo (1995), as tecnologias em sua maioria nem sempre satisfazem as expectativas dos usuários principalmente no que diz respeito à língua portuguesa que possui uma variedade de palavras com grafias diferentes, como é o caso da letra "x", que pode ter o som de "ch", de "s", de "ks" ou de "z", dentre outras situações mais complexas como a representação de siglas, datas, ou mesmo números de telefone, CPF ou RG, que possuem formas diferentes de pronúncia que devem ser tratadas como aspectos difusos.

Os fonemas, dependendo da utilização do aparelho fonador, ainda podem ser subdivididos em categorias, como exemplo: orais, nasais e fricativos.

Segundo Pais (1989) a fonologia define a fonética como sendo o estudo das constituintes da fala, de acordo com suas propriedades e, por isto, se subdivide em: (1) fonética articulatória e; (2) fonética acústica.

O estudo de fonologia estabelece quais sons são constituintes da fala segundo seu aspecto funcional, bem como o papel de os relacionar às atividades da voz, principalmente por suas propriedades acústicas e articulatórias.

A fonética articulatória procura descrever os diversos sons da língua de acordo com a dinâmica (posição e movimentação) dos articuladores que constituem o trato vocal humano (lábios, língua, mandíbula, etc.).

Já a fonética acústica procura analisar os sons da fala como sinais acústicos; Por isso leva em conta suas características espectrais.

1.5 Pronúncia

Para Chomsky (1971), a pronúncia é a capacidade de se articular as palavras de uma língua, sendo que as palavras podem ter várias formas de pronúncia de acordo com a região do país.

Por ser uma manifestação do que pensa ou sente o indivíduo, esta depende do estado psicológico em que o orador (usuário) se encontra no ato de emitir uma sentença.

1.6 Prosódia

De acordo com Mosser (2000) a prosódia é a variação na altura, intensidade, tom, duração e ritmo da fala, carregando informações adicionais expressas (raiva, dor, alegria, sacarmos) pela seqüência de segmentos fonéticos.

Pais (1989), relata que a prosódia é uma característica regional da fala, composta por hábitos e sentimentos regionais do indivíduo e, portanto, um processamento específico é essencial para garantir a inteligibilidade do sinal de fala sintetizado e, principalmente, para assegurar a sua naturalidade.

1.7 Gramática

Para Ferreira (2001), a gramática é o conhecimento internalizado dos princípios e regras de uma língua particular para se expressar corretamente e dialogar com outro ser.

No entanto para Chonsky (1971), a gramática pode ser tratada como um conjunto de regras que definem um protocolo de comunicação, onde são registrados dados desde a pronuncia e a prosódia até a informação propriamente dita.

No processo de reconhecimento da fala, a gramática tem papel fundamental, pois é com ela que são determinados os vocábulos que podem ser reconhecidos e compreendidos pelo sistema reconhecedor.

Segundo Chomsky (1971), a gramática é utilizada para representar uma linguagem natural e deve apresentar um bom balanço entre sua expressividade e o processo de reconhecimento.

1.8 Definição de Modelo Acústico

O modelo acústico é um modelo de como o som deve se parecer com o que realmente foi pronunciado. Neste modelo são esperadas variações de um fonema para que se enquadre nos diferentes tipos de vozes e timbres (masculinos e femininos).

Segundo Brega et. al. (2002) nestes modelos encontram-se as diferenças entre a voz pronunciada e as características dos sons básicos da linguagem a ser reconhecida.

1.9 Acurácia

A acurácia pode ser definida de acordo com Ferreira (2001), como sendo uma propriedade da medida de uma grandeza física que foi obtida por instrumentos e processos isentos de erros sistemáticos.

A ausência de erros sistemáticos é dada de uma forma estatística, o que difere da precisão que é fornecida de acordo com a exatidão de cálculos necessários ao processo.

1.10 Mecanismo de Reconhecimento de Fala

Pode-se definir que mecanismo (do inglês *engine*) de reconhecimento ou de síntese como uma máquina, programa ou módulo responsável pela função principal do software.

O mecanismo pode ser encentrado como um conjunto de programas ou serviços ligados ao sistema operacional, que fornece atributos e funcionalidades tais como: timbre da voz a ser reconhecida, taxa de precisão, prosódia da fala, identificações de idade, e de sexo.

Cada fabricante possui um mecanismo que pode ser acessado por um conjunto de API que normalmente estão dispostas em um pacote de software SDK – *Software Development Kit.*

1.11 Definição de biblioteca de programação para serviços de fala

Application Programming Interface ou simplesmente API é um conjunto de rotinas e padrões estabelecidos por um fabricante de software para utilização de funcionalidades do software vendido. De modo geral, a API é composta por uma série de funções acessíveis somente por programação, e que permitem utilizar características do software menos evidentes ao usuário tradicional.

As API´s permitem criar uma interface com um sistema operacional, ou a máquina virtual, ou qualquer outra biblioteca ou DLL, que forneça funções que possam ser chamadas por programas de aplicação.

Nestas de rotinas e funções pré-compiladas e prontas que realizam uma tarefa comum para requisitar e obter os serviços de baixo nível oferecidos pelo computador.

As bibliotecas de programação para serviços de fala usam dois modelos de interface, que são descritas em Amundsen (2004*),*sendo definidas pelo modelo de níveis : a) High-Level (alto nível) e; b) Low-Level (baixo nível). Onde a definição de nível se refere à forma de acesso ao mecanismo de serviço de fala.

O modelo de alto nível fornece os serviços básicos de manipulação do reconhecimento e síntese de fala para uma simples entrada ou saída de dados via áudio, ideal para o uso de comando e controle de itens de menu, janelas, ícones e botões de controle.

Já o modelo de baixo nível fornece serviços detalhados sobre o controle serviços de fala bem como manipula diretamente seu comportamento, é utilizado para fazer o reconhecimento de ditado livre, ou de uma conversa.

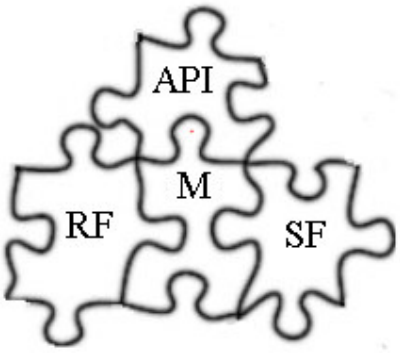


Figura 2 - Componentes de uma API

Na Figura 2 - Componentes de uma API, a parte destaca API, significa que existe um controle tanto de baixo como de alto nível para o Mecanismo M, e este controle é subdivido para as funções de Reconhecimento de Fala (RF) e para a Síntese de Fala (SF).

Para cada mecanismo M de reconhecimento e síntese de fala existe um conjunto de bibliotecas acessórias que realizam a interface de comando e defini-las não é o objetivo deste trabalho.

A principal vantagem do uso de API´s é que é possível acessar os serviços de fala, tanto em alto, quanto em baixo nível, independente do fabricante do mecanismo, aumentando a portabilidade destes sistemas.

Para os fabricantes que adotaram a tecnologia Java como principal padrão de desenvolvimento de API, os mecanismos disponíveis são: Dragon Natural Speech, IBM ViaVoice, Philips Speech, Texas Instruments, Chant Speech, Conversational, InRoad, Lernout & Houspie e Nuance Speech, onde todos estes mecanismos podem ser acessados pelo Java Speech API.

Entretanto para os fabricantes que adotaram o modelo dependente de mecanismo existe apenas uma forma de acesso e controle do mecanismo que é por meio do Microsoft Speech API que fornece suporte de maior parte das linguagens de programação para windows e suporta o padrão de desenvolvimento baseado em componentes do modelo COM/DCOM (Componnent Object Model – Distribuied Componnent Object Model).

Para esta pesquisa foi escolhidas as duas API´s para teste: a Java Speech API, por ter ser de criada na linguagem Java, o que a torna portável (*cross-plataform*) além de ser gratuita, e a Microsoft Speech API, que usa o mecanismo da Microsoft., mas não é portável (*cross-plataform*).

1.12 Definição de Processamento Digital dos Sinais de Fala

O objeto do estudo do processamento digital de sinais de fala é a representação digital, análise e extração de características e o desenvolvimento de modelos de síntese.

Todas estas ferramentas são cruciais na implementação de sistemas de comunicação falada, seja esta comunicação à distância ou comunicação homem-máquina.

Os Sistemas de Processamento de Sinais de Fala podem ser classificados em sistemas de codificação da fala (onde abrangem desde a extração dos sinais de entrada por voz até a codificação em pulsos magnéticos), os sistemas de síntese de fala (que são responsáveis pela retroalimentação ou *feedback* ao usuário), o reconhecimento de sinais de fala (que é o sistema responsável pela compreensão do que foi pronunciado) e o sistema de verificação e identificação do orador, o qual pode utilizar uma rede neural artificial para classificação e segmentação das características da voz do orador ou usuário.

1.12 A compreensão da mensagem falada

Ao se expressar verbalmente o orador utiliza um complexo sistema de símbolos e combinações destes símbolos, os quais o receptor da mensagem deve entender, isto é, consegui decifrar os símbolos e suas combinações em uma frase.

A ilustração na Figura 3 elucida que quando um orador está formulando a mensagem, este lança mão de signos lingüísticos e leis combinatórias, isto é, símbolos visuais ou lingüísticos que culminam em uma regra gramatical e lógica para a formulação de uma frase (utiliza-se da sintaxe e semântica da língua portuguesa).

Logo após, sua mente reflete inconscientemente a codificação fonética e psicolingüística, o que se denomina de locução neural da voz e por meio de um controle

neuromuscular e da atualização lingüística (sotaque, adquirido na região do orador), juntamente com o controle das cordas vocais é proferida a frase ou mensagem por meio de um meio comum entre o orador e o ouvinte.

Já o ouvinte, para compreender a mensagem do orador pelo meio de transmissão, movimenta a membrana basilar do ouvido, o qual faz a transdução neural ou neuronal, que é responsável por levar os sinais obtidos (escutados) até o cérebro, onde é feita uma decodificação fonética e uma interceptação psicolingüística, o qual reflete o reconhecimento do sotaque, a nova atualização lingüística e principalmente a reflexão dos fonemas que mais adiante serão identificados com os signos lingüísticos do ouvinte e logicamente organizados de acordo com as regras ou leis combinatórias (sintaxe e semântica) do ouvinte.

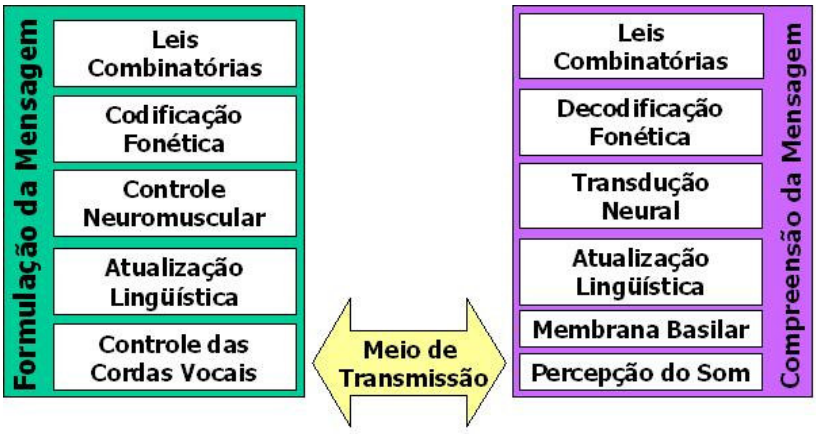


Figura 3 - Processo de formulação e compreensão da mensagem

As pesquisas científicas avançam cada dia mais para conseguir compreender a linguagem do ser humano, mas ainda, para que a máquina compreenda o que está sendo pronunciado existe uma dependência de contexto, e uma gramática de regras bem definidas.

1.13 A formulação da Fala

De acordo com Pizzolato (2001) o processo de produção ou geração da fala começa com a formulação da mensagem pelo orador e que expressa sua idéia ao ouvinte. O orador pode executar movimentos neuromusculares que movimentam as articulações do aparelho fonador ou órgão fonador.

O aparelho fonador visto na Figura 4 é composto pelos pulmões, brônquios e traquéia que são os órgãos respiratórios que permitem a corrente de ar, sem a qual não existiriam sons. A maioria dos sons que conhecemos são produzidos na expiração, servindo a inspiração como um momento de pausa; A laringe, onde ficam as cordas vocais, são elas que determinam a sonoridade (a vibração das cordas vocais) dos sons; A faringe, boca (e língua) e as fossas nasais são órgãos que formam a caixa de ressonância responsável por outra grande parte da variedade de sons.

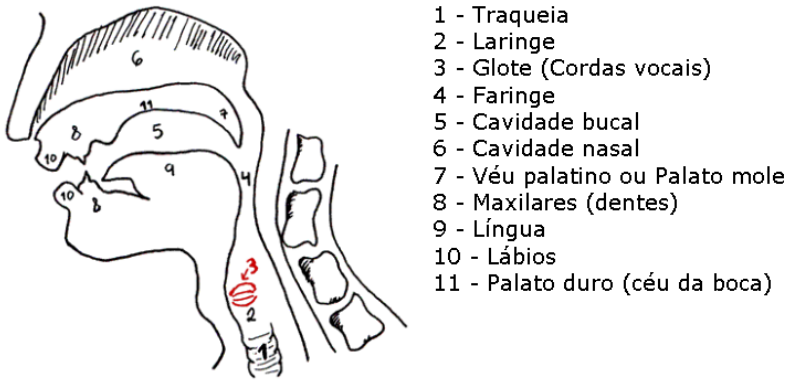


Figura 4 - Aparelho fonador humano

De acordo com cada movimento, expressão dos músculos, ou mesmo a força do ar nos pulmões, pode acarretar em um som diferente para o mesmo fonema.

Na laringe, situa-se a glote, órgão essencial na fonação, onde estão as cordas vocais. Seu funcionamento poderia ser resumido: quando a glote está aberta e as cordas vocais afastadas uma da outra, permitem a passagem livre da corrente aérea originária dos pulmões. Nessa posição, não produzem nenhum som – é a posição da respiração natural (pode ocorrer a pausa na fala, ou apenas a respiração) – quando completamente fechada a glote, interrompe a corrente aérea.

O fechamento brusco da glote provoca o fenômeno conhecido como “golpe de glote”, por exemplo ao se pronunciar a palavra “D’Água” ou “D’ Ângelo”, dentre outras encontradas em diversos idiomas, pode ser observado um ilustração na Figura 5.

Quando a glote está fechada, as cordas vocais muito próximas e relativamente tensas, opõem resistência à corrente aérea e conseqüentemente, vibram, produzindo o chamado som glotal.

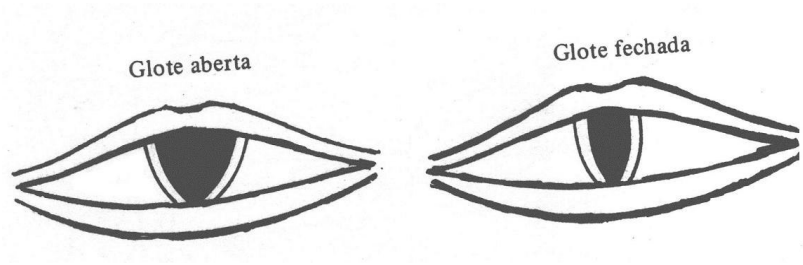


Figura 5 - Esquema da glote aberta e fechada

Essa última é a posição de fonação propriamente dita e o som glotal varia de acordo com a tensão das cordas vocais que são músculos, e de acordo com o estreitamento provocado. Quando abertas, as cordas vocais definem um espaço triangular, que se estreita para a fonação, Figura 5.

A epiglote é uma espécie de válvula que se situa no topo da laringe. Quando fechada, impede que os alimentos deglutidos entrem na laringe, em vez de serem dirigidos para o estômago. Quando aberta, a epiglote, permite a respiração e a fonação, embora não tenha, como se vê na Figura 5, um papel particular na fonação propriamente dita.

A faringe serve, juntamente com a cavidade bucal, de caixa de ressonância, que permite amplificar certos sons produzidos pela glote.

A úvula, situada ao fim do palato mole, na cavidade bucal tem duas posições: fechada, dirige toda a corrente de expiratória para a cavidade bucal, de modo a produzir os sons chamados orais; Aberta, dirige parte da corrente expiratória para as fossas nasais, que funcionam, então, como caixa de ressonância, reforçando outros harmônicos, de modo a auxiliar a produção dos sons chamados de nasais.

Segundo Spencer (1991), a cavidade bucal tem um papel de alta relevância na modulação do som produzido nas cordas vocais e na produção de ruídos. Com efeito, as cordas vocais produzem apenas movimentos periódicos, ou seja, sons no sentido de fonema, enquanto os movimentos aperiódicos, são considerados ruídos e estão são produzidos na cavidade bucal.

No que se refere ao som proveniente da laringe, a abertura da cavidade bucal que determina o seu tamanho, a posição da língua, que divide a cavidade bucal em duas caixas de ressonância variáveis, e a posição dos lábios que se projetando ou retraindo-se aumentam ou diminuem respectivamente a caixa de ressonância anterior delimitada pela língua, enfim, a combinação dessas posições define duas caixas de ressonância cujo tamanho, em cada caso determina a amplificação de dois sons distintos, o que chamamos de sons harmônicos glotais ou apenas vogais.

Existem as seguintes zonas de articulação, que são responsáveis pelos fonemas produzidos pelo nosso idioma [Português - Brasil], sendo elas: Bilabiais, Labiodentais, Linguodentais, Alveolares, Palatais, e Velares. Os Bilabiais são responsáveis pelo som de [p] e [b] e são emitidos com o fechamento dos lábios; Os Labiodentais são emitidos com a aproximação dos lábios inferiores à arcada dentária superior o que proporciona o som de [f] e[v]; Os Linguodentais são produzidos com o toque da língua na parte interna da arcada superior, refletindo o som de [t] e [d]; Os Alveolares são emitidos com o toque (ou apenas a aproximação) da língua nos alvéolos, sua aproximação produz o som equivalente a [s], [z], [n], [r],[l]; Os Palatais são produzidas com o dorso da língua aproximando-se do palato duro, o que reflete no sons de [nhe][che],[lhe],[ss]; Já os Velares tem o som de [k] e [g] que são emitidos pela aproximação do dorso da língua ao véu palatino.

Segundo PAIS (1981), o funcionamento das cordas vocais, o modo de articulação, a zona de articulação e a ressonância nasal combinam-se na produção dos diferentes sons da linguagem.

Segundo Pizzolato (2001), a taxa de informação contida no processo da formulação da mensagem, é algo aproximado de 50 bps (*Bits per Second*), o que significa de 6 a 8 sons por segundo. Após a conversão para fonemas e a inclusão da prosódia, a taxa de informação subiria para 200 bps. Até então a informação é discreta. No processo de conversão da mensagem para ondas sonoras, a informação passa a ser contínua e a taxa sobe para aproximadamente 2000 bps ao nível neuromuscular e algo entre 30.000 a 50.000 bps no nível do sinal acústico captado.

2. MÉTODOS DE RECONHECIMENTO DE FALA

Um sistema reconhecedor de fala é basicamente um comparador de padrões acústicos de voz associados a um conjunto de símbolos que codificam uma mensagem. Quando são feitas comparações de padrões, espera-se que o sinal de entrada a ser comparado não apresente muitas variações, a fim de facilitar o processo.

Para tal processo é necessário considerar as formas de captação do sinal de fala do ponto de vista do usuário, pois de acordo com a prosódia e a pronuncia das palavras e frases é possível estabelecer uma relação melhor de comparação entre os elementos.

Segundo Furui (1989), a escolha do método de reconhecimento depende essencialmente de algumas características genéricas do sinal de fala que se pretende reconhecer e que são habitualmente classificadas da seguinte forma:

- Espontânea: fala recolhida em situações reais cotidianas. Incluem além da mensagem composta na fala, um conjunto de fenômenos não lingüísticos tais como hesitações, ruídos diversos, tosse, riso e cacoetes de fala.

- Contínua: refere-se a um monólogo aplicado geralmente à leitura de um texto. Possui um vocabulário relativamente grande e esparso.

- Ligada ou ativada: é um conjunto de sinais de fala que é obtido por meio de uma entrada de áudio e comparada com um pequeno vocabulário, normalmente usado em sistema de comando e controle por fala.

- Palavras isoladas: é a fala de palavras com sentido de comando do sistema. O processo é dado com pausas antes e depois de cada palavra e depende de um vocabulário específico e pequeno.

De acordo com o hardware e software escolhidos existem restrições, mas é possível obter uma melhora considerável no desempenho do sistema de reconhecimento observando estes modelos. Estas restrições levam em consideração o grau de dependência do locutor, estilo de fala (forma de pronúncia da sentença falada) e tamanho do vocabulário de reconhecimento.

Segundo Ronald et. al (2000), é possível classificar os sistemas de reconhecimento de fala pela característica de dependência de locutor. E de acordo com Roe & Wilpon (1994), a dependência de locutor é, essencialmente, uma característica resultante do modo como o reconhecedor será implementado.

Pode-se notar que, à medida que locutores de diferentes aspectos lingüísticos (idade, sexo, nível sócio-cultural, entre outros fatores) são utilizados no treinamento do sistema, mais independente de locutor o sistema será, ou seja, quanto maior a variabilidade de pessoas participantes da base de treinamento, melhor poderá vir a ser o desempenho do sistema para um locutor genérico.

Os sistemas dependentes de locutor são capazes de reconhecer a fala de apenas um locutor, aquele para o qual foi treinado. Entretanto, caso um locutor que não tenha

previamente treinado o sistema, mas tiver o timbre de voz semelhante ao padrão acústico de um outro locutor o sistema poderá reconhecer algumas sentenças ou comandos.

Os sistemas independentes de locutor, ao contrário dos sistemas dependentes de locutor, são capazes de reconhecer a fala de qualquer pessoa, mesmo que esta não tenha participado do treinamento para obtenção dos padrões de referência no modelo acústico.

Segundo Borges (2001), um locutor pode apresentar variações na pronúncia, causada por fatores físicos, motrizes ou psicológicos. As variações de locução de um orador são denominadas de variações intralocutor, e são inerentes apenas a um locutor por vez.

Existem também as variações associadas às diferenças fisiológicas entre os locutores, como sexo e idade e às diferenças culturais e regionais como o sotaque. Estas são denominadas variações interlocutores.

Para o efetivo desenvolvimento de uma aplicação de reconhecimento de fala é necessária a definição do modelo de locutor, ou seja, se o sistema irá reconhecer as variações intralocutores ou interlocutor, pois para ambos os métodos existem abordagens distintas.

2.1 Abordagens para o reconhecimento de fala

O reconhecimento de fala funciona a partir da conversão da voz (sinal acústico) em um sinal digital de áudio. Por meio de um hardware (microfone e placa de som) é possível capturar o sinal acústico e transformá-lo em sinal digital.

Até há alguns anos atrás o reconhecimento de fala era feito de modo discreto, isto é, o usuário tinha que fazer uma pausa entre cada palavra ditada. Guilhoto (2002). Atualmente, o usuário já tem a possibilidade de efetuar ditados de forma contínua ao computador, entretanto

segundo a ZDNet (2001), mesmo com o avanço dos métodos e algoritmos de reconhecimento, as taxas de acurácia ainda não passaram dos 95 % de acerto.

Segundo Hugo (1995), existem duas grandes abordagens para o reconhecimento. Estas abordagens não representam uma classificação rígida, podendo até ser complementares entre si, a fim de solucionar o problema de reconhecimento: (1) o Método Global e; (2) o Método Analítico.

O Método Global, também conhecido por reconhecimento de palavras, utiliza basicamente as técnicas de reconhecimento com intuito de comparar, globalmente, a palavra e reconhecer as diversas formas de referência armazenadas.

O tratamento acústico preliminar é bem simples: a mensagem a identificar é considerada como uma forma atômica, sem problemas de segmentação.

Esta abordagem global se revela insuficiente ao se tratar com grandes vocabulários ou fala contínua. Faz-se, então, necessário adotar a abordagem analítica.

O Método Analítico consiste em segmentar a mensagem em constituintes elementares (fonemas, sílabas, etc). Depois de identificados os elementos, reconstituir a frase pronunciada por etapas sucessivas: fonética, léxica, sintática.

Outra abordagem é sugerida por Tatham (1995), onde relata que o reconhecimento de fala é realizado de quatro formas básicas, que possibilita o sistema a partir de uma entrada de voz (sinal digital) transcrever declarações reconhecidas em ortografia normal (de acordo com o idioma e a gramática utilizada):

- Abordagem baseada em Modelos: onde a voz de entrada é comparada com unidades modelo armazenadas, no esforço de encontrar a que melhor se parece com o modelo observado;

- Baseada em Conhecimento: tenta emular a habilidade do conhecimento humano para reconhecer voz, com o uso de elementos da inteligência artificial e raciocínio baseado em casos;

- Estocástica: explora as propriedades herdadas da estatística da ocorrência de sons de voz individuais;

- Conexionista: Por meio da teoria dos grafos é possível notar redes de um grande número de nós simples interconectados, representando fonemas e suas conexões, assim treinados para reconhecer a fala de um usuário definido.

Apesar de já terem sido desenvolvidos diversos sistemas de reconhecimento de fala contínua, eles estão longe da capacidade do ser humano de lidar com as diferenças de pronúncia, isto é, alofones contextuais (sotaque), além da presença de ruído, frases gramaticalmente incorretas, ou mesmo, palavras desconhecidas.

Além disso, quase sempre o objetivo do homem é compreender aquilo que foi dito e, neste âmbito concorrem diversas fontes de conhecimento tais como o conhecimento da sintaxe e semântica da língua, conhecimento do assunto falado, referência a frases anteriores, etc.

De acordo com Pizzolato (2001) diz que o reconhecimento de fala pode se dar em três abordagens distintas: (1) A abordagem acústica - fonética; (2) A abordagem de reconhecimento de padrões e; (3) A abordagem de inteligência artificial;

A primeira abordagem destacada por este pesquisador é baseada na teoria da acústica e fonética, que tem como premissa à existência de um conjunto finito e distinto de unidades fonéticas em uma linguagem, e que estas são caracterizadas por um conjunto de propriedades presentes no sinal de fala.

Na abordagem é baseada em padrões, há basicamente duas fases: (a) treinamento e (b) reconhecimento. No treinamento, padrões (consideravelmente suficientes para representar as variações de cada unidade sonora em uma população avaliada) são apresentados ao sistema e ele os armazena de uma forma concisa, procurando aprender as suas variações. No reconhecimento, cada padrão de entrada é comparado com os padrões armazenados e o que é mais semelhante empresta seu símbolo ou número característico para este segmento de fala.

A ultima abordagem citada utiliza os princípios de inteligência artificial para reunir os conhecimentos: (1) acústicos, (2) léxico, (3) sintático, (4) semântico e (5) pragmático.

O conhecimento de acústica utiliza evidências sonoras para detectar unidades fonéticas, o conhecimento léxico permite combinar evidências acústicas de tal forma a construir palavras de acordo com um dicionário, já o conhecimento sintático permite construir sentenças a partir de combinações corretas de palavras, e o conhecimento semântico permite entender o domínio da frase e verificar a consistência da mesma. E, por fim, o conhecimento pragmático permite resolver ambigüidades.

Devilliers (1996), expressou-se de uma maneira mais formalista ao tentar identificar a abordagem de seu trabalho. Na Figura 6 é possível estabelecer o nível de interação entre os sistemas compostos do reconhecimento de voz assim como a distinção de suas partes.

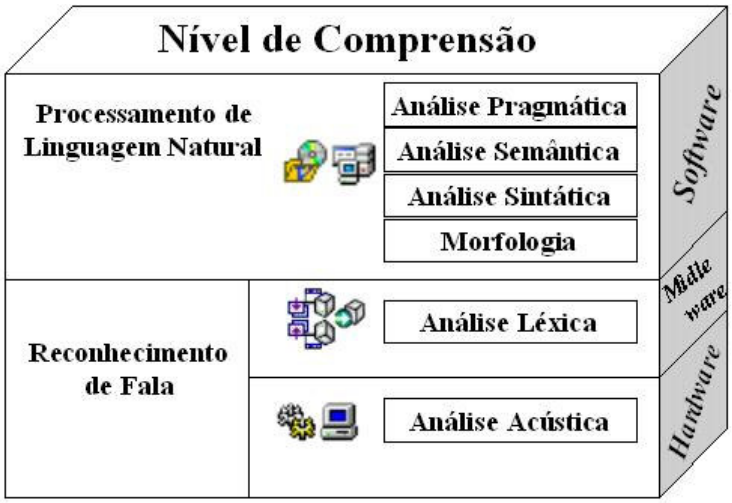


Figura 6 - Níveis do sistema de compreensão da linguagem falada Devillers (1996)

Na Figura 6 observa-se que para exercer uma efetiva compreensão de uma frase dita por um orador é necessário que o software reconhecedor consiga resolver e interpretar as seguintes análises: Pragmática, Semântica, Sintática e Morfológica; E depender interinamente de módulo intermediários que fazem a análise léxica e outros serviços disponíveis pelo sistema de hardware para obter a análise acústica. Estes dois últimos processos pertencem a atividade de reconhecimento de fala.

Dentre todas a abordagens citadas a que melhor obteve resultados comprovados foi a de reconhecimento de padrões.

O reconhecimento da fala é o processo de extração automática da informação lingüística do sinal de voz. A informação lingüística contida no sinal de voz está codificada de modo que o elevado grau de variabilidade do sinal, causada pelo ambiente e pelo locutor, praticamente não interfere na percepção da informação pelo homem.

De acordo com Hugo (1995), a tarefa de reconhecer palavras faladas pode ser dividida em três fases: Aquisição do sinal de voz, Pré-processamento (extração de parâmetros) e

Processamento (Reconhecimento do Padrão). E cada um deste processos realiza transformações nos dados recebidos, como pode ser visto na Figura 7.

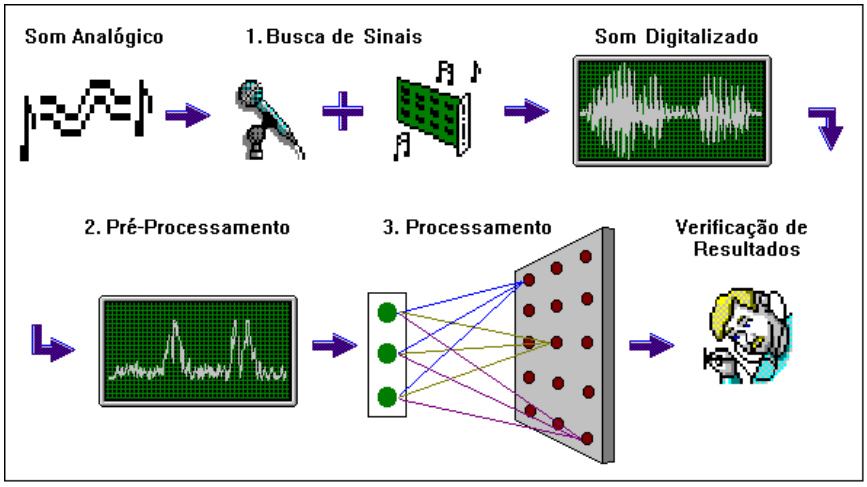


Figura 7 - As fases empregadas para reconhecimento de fala, Hugo (1995)

A primeira fase realizada é a captação do som da fala e sua transformação em sinal digital para manipulação por um computador. Obviamente, o computador não pode armazenar o som como ele é.

Para armazenar o som, o formato da informação precisa ser alterado para que o computador possa armazenar ou manipular os sons na memória. O som é captado por meio de um microfone, o qual está acoplado a uma placa digitalizadora.

Em seguida é realizado o pré-processamento de sinais (extração de parâmetros) o qual é uma forma de organizar ou ajustar os sinais captados procurando torná-los passíveis de processamento na fase posterior do processo de reconhecimento de fala. Ele é responsável por gerar o vetor característico do padrão, também conhecido como *feature number,* a ser analisado posteriormente.

De acordo com Trevino (2003), esta fase pode ser dividida em duas outras subfases:

1. Pré-processamento: utilização de técnicas de "limpeza" do sinal, como normalização, escala, suavização, etc. e;

2. Extração de características: eliminação de sinais redundantes ou insignificantes por meio de seleção ou transformação.

Por último a fase de processamento ou reconhecimento de padrões que de acordo com Castro e Prado (2002) são as propriedades que possibilitam o agrupamento de objetos semelhantes dentro de uma determinada classe ou categoria, mediante a interpretação de dados de entrada, que permitam a extração das características relevantes desses objetos.

Segundo Fagundes (1993), um dos pontos-chave no reconhecimento de fala é a comparação dos padrões. O objetivo é determinar o grau de similaridade entre dois padrões ou a distância entre eles.

Por exemplo, um padrão de teste **T** pode ser definido como sendo a concatenação de trechos de alguma fala durante o período de duração desta fala sendo $\mathbf{T = \{t_i, t_2, t_3, t_4 \ldots t_F\}}$, onde cada $\mathbf{t_i}$ é um vetor corresponde à informação espectral do tempo **I** (inicial) e **F** (final), é o número total de seqüências onde foram captados os sinais de fala, chamados também de quadros da fala. De forma similar pode definir um conjunto de padrões de referência: $\mathbf{R = \{r_i, r_2, r_3, r_4 \ldots r_F\}}$.

Ao se assumir que um banco de padrões contém **V** padrões, o objetivo é encontrar um uma referência em $\mathbf{r_n}$ que seja similar ao padrão $\mathbf{v_n}$ e que tenha o tempo de duração $\mathbf{t_1}$ a $\mathbf{t_n}$ estipulado de acordo com a prosódia do orador. Entretanto, este fonema pode ou não fazer

parte de uma palavra ou sentença, o processo agora é estabelecer qual será a junção mais correta do fonema para se criar uma palavra.

Normalmente para o processo identificar a palavra é realizado um cálculo probabilístico que indica a maior semelhança entre o modelo proposto **V** e o que foi encontrado na captura, o que significa que para se encontrar a maior semelhança deve-se obter a probabilidade de se encontrar o fonema correto para a palavra correta dentro de um espaço de sinais espectrais difusos.

Rabiner (1989), procura solucionar este problema por meio de técnicas que estabeleçam a maior probabilidade de se encontrar o fonema certo na ordem certa em um espaço difuso. Neste caso devem ser empregados modelos estocásticos e probabilísticos de Markov.

2.2 Modelos de Reconhecimento de Padrões

Segundo Ynoguti (1999), no início dos anos 70 é que se inicia a uso dos modelos de Markov para área de processamento de sinais com ênfase no reconhecimento da fala contínua.

Em seu contexto, os modelos de Markov tinham por tarefa modelar de forma estocástica (probabilística) e muito segura. A cadeia de caracteres formada por meio de um conjunto de sinais dispersos e difusos (sinais de fala). Isto porque, segundo Pizzolato (2001), seus parâmetros podem ser estimados com precisão e de uma forma bem definida que podem ser aplicados no reconhecimento de fala contínua (onde não existem silêncios longo entre palavras) e permitem uma fácil integração com os modelos sintáticos.

De acordo com Vieira e Lima (2001), os modelos de Markov servem para modelagem de uma seqüência de eventos, onde esses modelos trabalham com a ordem das palavras na

sentença, podendo utilizar a ordem visível das palavras (*Visible Markov Models,*VMM) ou a ordem "oculta" dessas palavras (*Hidden Markov Models,* HMM), ou seja, um nível de abstração mais alto com relação à possível seqüência das palavras na sentença.

No caso dos HMM, que é o modelo mais utilizado, esse nível adicional de abstração permite inserir estruturas adicionais, para visualizar a ordem das categorias das palavras.

Para Ynoguti (1999), um HMM é uma composição de dois processos estocásticos, uma cadeia de Markov oculta, relacionada à variação temporal, e um processo observável, relacionado à variabilidade espectral.

Esta combinação provou ser poderosa para lidar com as fontes mais importantes de ambigüidade, e flexível o suficiente para permitir a realização de sistemas de reconhecimento com dicionários grandes (na ordem de dezenas de milhares de palavras).

De acordo com Fagundes (1993), o sistema de reconhecimento de fala também necessita saber quando é que um fonema acaba e outro começa.

Para entender os modelos de Markov de estados ocultos (*Hidden Markov Models*), primeiramente deve-se compreender os modelos de Markov nos estados observáveis (*Visible Markov Models*).

Primeiramente deve-se compreender que os modelos são máquinas de estados finito onde existe a mudança de um estado atual **i** para um estado futuro **j** a cada unidade de tempo **t** e associado ao estado **j** um novo instante $\mathbf{t_1}$, ou seja, os modelos de Markov (observáveis ou ocultos) proporcionam modelos matemáticos para uma variedade de sistemas que ao longo do tempo, movimentam-se por meio de um conjunto enumerável de estados.

A evolução desses sistemas é regida por um mecanismo probabilístico cuja característica principal está na forma de dependência entre o comportamento passado e futuro, do sistema, relativamente a um instante fixado. Especificamente, o comportamento futuro do sistema a partir de um instante fixado depende de seu comportamento passado apenas por meio do estado que o sistema ocupa naquele instante.

Existe também a probabilidade da máquina não entrar em nenhum estado (o que significa que não ouve registro de entrada) e de não haver alteração de estado. Portanto deve-se contar com a probabilidade da transição do estado I para o estado J sendo de forma discreta e dada na seguinte expressão: $\mathbf{a_{ij} - P\ (q_t = j \mid q_{t-1} - i)}$

Um exemplo destacado por Pizzolato (2001), descreve didaticamente a atuação de um modelo de Markov observável:

Suponha três situações: (1) dia ensolarado; (2) dia nublado; (3) dia chuvoso. Cada situação pode ser representada por um estado e as transições de um estado para outro (incluindo aqui a possibilidade de se permanecer em um mesmo estado) são indicadas por setas com valores probabilísticos associados conforme ilustrado na Figura 8.

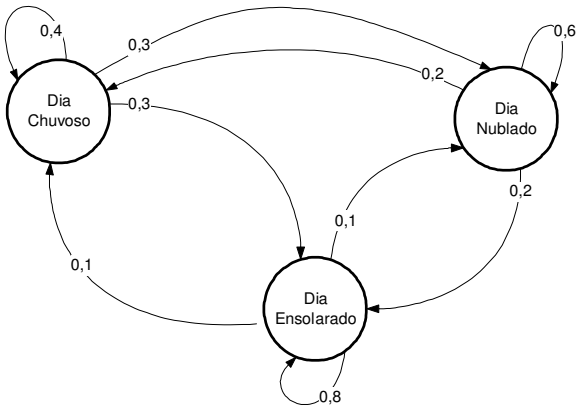


Figura 8 - Modelo de Markov de estados observáveis, Pizzolato (2001)

Para uma descrição probabilística completa do sistema visto na Figura 8 é necessária uma especificação do estado atual (tempo t) bem como dos estados predecessores. Para o caso dos modelos de primeira ordem, a dependência probabilística é truncada para somente o estado anterior.

Dado o processo acima podemos ter a probabilidade de se ter 8 dias consecutivos da seguinte forma: ensolarado – ensolarado – ensolarado – chuvoso – chuvoso – ensolarado – nublado – ensolarado.

Para resolver este problema deve-se definir a seqüência de observações sendo O = {3,3,3,1,1,3,2,3} , onde [3] corresponde a ensolarado, [2] nublado e [1] chuvoso.

$$
\begin{aligned}
\text{Assim, } P[0 \mid \text{modelo}] &= P[3,3,3,1,1,3,2,3] \\
&= P[3] \bullet P[3|3] \bullet P[3|3] \bullet P[1|3] \ldots \bullet P[3|3] \\
&= 1 \bullet (a_{33})^2 \bullet (a_{31}) \bullet (a_{11}) \bullet (a_{13}) \ldots . \bullet (a_{33}) \\
&= 1 \bullet (0{,}8)^2 \bullet (0{,}1) \bullet (0{,}4) \bullet \ldots . \\
&= 1.536 \bullet 10^{-4}
\end{aligned}
$$

Este é um processo de Markov de estados observáveis, visto que cada estado corresponde a um evento observável de forma determinística.

Agora que se pode compreender como trabalha o modelo observável será mostrado o trabalho do modelo oculto, que é uma extensão do modelo observável.

O modelo oculto é o modelo de observação onde não é possível determinar com certeza qual estado está sendo observado, e para tal problema usa-se a probabilidade para se prever qual é o estado seguinte para se determinar o estado atual.

O símbolo atual por ser de definição probabilística pode ser prognosticado, mas somente por meio de outro processo que produz a seqüência de observações compatíveis.

Segundo Vieira e Lima (2001), um modelo de Markov de estados ocultos é um modelo probabilístico definido à custa de dois processos agrupados.

Um desses processos probabilísticos é um processo de Markov discreto (cadeia de Markov com *N* estados), cujas transições entre estados sinalizam uma mudança das propriedades estatísticas do sinal que modelam (ou que as geram). E, associado a cada estado da cadeia existe outro processo (contínuo ou discreto) que descreve a estatística do sinal associado a esse estado.

Pode-se aplicar estes modelos ao reconhecimento de fala uma vez que a fala se caracteriza por ser uma seqüência de segmentos quase estacionários, mas com propriedades diferentes, correspondentes a diferentes tipos de timbres de voz, entonações, sexo e idade do locutor.

Segundo Ynoguti (1999), pode-se definir uma estrutura de um HMM sendo definido como um par de processos estocásticos (**X, Y**). O processo **X** é uma cadeia de Markov de primeira ordem, e não é diretamente observável, enquanto que o processo **Y** é uma seqüência de variáveis aleatórias que assumem valores no espaço de parâmetros acústicos (*observações*).

Por meio das seqüências geradas pelo HMM pulando de um estado para outro, emitindo uma observação a cada salto. Em geral, para o reconhecimento de fala, é utilizado um modelo simplificado de HMM conhecido como modelo *left-right*, no qual a seqüência de estados associada ao modelo tem a propriedade de, à medida que o tempo aumenta, o índice

do estado aumenta (ou permanece o mesmo), isto é, o sistema caminha da esquerda para a direita no modelo de acordo com a Figura 9

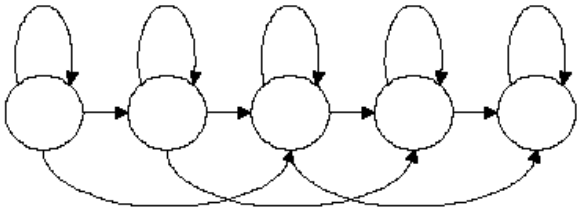


Figura 9 - Modelo para um HMM left-rigth de 5 estados, Ynoguti (1999)

Segundo Pizzolato (2001), na tarefa de reconhecimento de fala, geralmente são adotadas duas simplificações da teoria de modelos de Markov, que podem ser formalizadas da seguinte ordem:

- Hipótese de Markov de primeira ordem: a história não tem influência na evolução futura da cadeia se o presente é especificado.

- Hipótese de independência das saídas: nem a evolução da cadeia nem as observações passadas influenciam a observação atual se a última transição da cadeia é especificada.

Um sistema de reconhecimento de fala gera hipóteses de palavras ou seqüências de palavras. Destas hipóteses pode resultar uma única seqüência de palavras, uma coleção de n melhores seqüências de palavras, ou um conjunto de hipóteses de palavras parcialmente superpostas sem nexo. Isto é feito num processo de busca no qual se compara uma seqüência de vetores de características acústicas com os modelos das palavras que estão no vocabulário do sistema.

Este processo visa encontrar a seqüência mais provável de marcas, rótulos ou símbolos que correspondam a uma dada seqüência de palavras.

Para computar a seqüência de estados mais provável, normalmente é utilizado o algoritmo de Viterbi, segundo Ynogutti (1999), isto porque a probabilidade das seqüências pode ser de ordem logarítmica, ou seja, muito pequena, e pode atrapalhar na precisão do reconhecimento.

Segundo Hoson (2003), esta consiste no fato de que para cada palavra existe uma lista de possíveis palavras que se lhe seguem; uma generalização deste raciocínio é associar a probabilidade em cada palavra com o conjunto de palavras seguinte.

Como exemplo para uma probabilidade bi-grama, tem-se:

$$P(\text{Eu comi pão}) = P(\text{eu} \mid \text{início}) * P(\text{comi} \mid \text{eu}) * P(\text{pão} \mid \text{comi})$$

Para pequeno conjunto analisado, as probabilidades são calculadas a partir da contagem de co-ocorrências das palavras:

$$P(\text{comi} \mid \text{eu}) = \frac{P(\text{eu comi})}{P(\text{eu})} = \frac{\text{Nº de ocorrências (eu comi)}}{\text{Nº de ocorrências (eu)}}$$

Isto se faz com o pressuposto de que há uma transcrição de palavras no conjunto de treino.

Este é um verdadeiro problema, e a única solução é possibilitar ao locutor “treinar” os modelos acústicos.

No caso de um sistema dependente de locutor, basta fornecer um texto predeterminado ao usuário para este ditar. Analisando sistemas independentes de locutor, por exemplo, no caso de uma central telefônica, não é viável pedir a cada usuário falar durante 15 minutos para treinar o mecanismo reconhecedor de fala.

A solução para este caso é efetuar um conjunto de treino do sistema por várias pessoas, sendo os pesos das transições dos HMM's ajustado de acordo com a média dos modelos, de forma a tornar o sistema capaz de reconhecer o maior número de usuários possível.

A solução para este caso é efetuar um treino conjunto do sistema por várias pessoas, sendo os pesos das transições dos HMM's ajustado de acordo com a média dos modelos, de forma a tornar o sistema capaz de reconhecer o maior número de usuários possível.

De acordo com Fagundes (1993), o treino do software é baseado em um conjunto de redes neurais artificiais, onde é aplicado um modelo denominado de SOM (*Self-Organizing Map*) que é utilizado para aprender fonemas, que posteriormente serão transformados em palavras, por meio de regras gramaticais aprendidas automaticamente. O treinamento é realizado com amostras dos espectrais de curta duração, ou seja, de pequenos segmentos de fala, que são apresentados à rede na ordem natural da fala, o que contribui para a auto-organização na rede SOM.

No treino basta que o locutor dite um conjunto de palavras pré-definidas (em torno de 200 a 300 palavras) ou um pequeno texto, para que o sistema de reconhecimento por meio de uma rede neural artificial, possa gravar as definições padrões de pronúncia de cada fonema do locutor.

2.3 Reconhecimento Automático da Fala

O Reconhecimento Automático de Fala (*Automatic Speech Recognition*) baseia-se na identificação de padrões de fala. Sendo este padrão estabelecido, por treino do sistema pelo orador, pode-se obter resultados diferentes, para ambientes diferentes, isto é, além da relevância da utilização de um orador conhecido, de um vocabulário finito, deve-se tomar certos cuidados ao se preparar um sistema para o reconhecimento, a saber:

O Reconhecimento Automático da Fala é o processo de extração automática da informação lingüística do sinal de voz. A informação lingüística contida no sinal de voz está codificada de modo que o elevado grau de variabilidade do sinal, causada pelo ambiente e pelo locutor, praticamente não interfere na percepção da informação pelo homem.

Segundo testes aplicados por Hoson (1995), verificou-se que a acurácia no reconhecimento de fala chega até aos 90 % de certeza na maioria dos casos em condições triviais de uso.

As condições triviais de uso são determinadas por Sun (1999) onde para que o sistema de reconhecimento de fala possa ter uma acurácia superior a 95% o orador deve ter realizado o treino do mecanismo, pronunciar as palavras sem interrupções, e a captação do som deverá ser realizada em um lugar onde não exista interferência de outros sons.

Entretanto ao se aplicar situações onde o vocabulário é pequeno e o microfone não reconhecendo ruídos, o reconhecimento chega a ultrapassar a faixa dos 98% de acurácia, segundo IBM (1999).

A dificuldade é maior quando se deseja reconhecer a fala contínua, essencialmente por que quanto maior o vocabulário maior a semelhança entre as palavras e seus fonemas, ficando

mais complexa a análise da palavra realmente ditada e quando existe o efeito da co-articulação das palavras, isto é, quando existirem sotaques entre os falantes da língua, e a velocidade em que pronunciam suas palavras.

Portanto uma boa definição da gramática utilizada faz se necessário juntamente com a elucidação dos fatores que comprometem o uso da tecnologia.

As restrições citadas irão influenciar características como acurácia, tipo de aplicação, custo, entre outras.

2.4 Aplicações de Linguagens Naturais

As pessoas se comunicam para conversar, comandar, interrogar, responder, prometer, convencer as outras pessoas. Mas para que haja a comunicação é necessário um veículo, uma forma de as duas entidades comunicantes compartilharem o mesmo modelo.

Segundo Furui (1989) a linguagem falada é a forma natural de o homem interagir com o mundo e a tendência é ter cada vez mais aparelhos eletrônicos controlados por voz, eliminando assim o mouse e o teclado.

Os seres humanos utilizam-se principalmente da Linguagem Natural para se comunicarem. A Linguagem Natural é uma das formas mais humanas de manifestação externa da atividade mental e um meio muito rico para comunicação entre as pessoas.

A Linguagem Natural é a comunicação estruturada e inteligente entre pessoas. Ela consiste de sons organizados, vocabulário, estruturas tais como alfabetos ou outras representações simbólicas, gramática ou sintaxe, significado dependente da estrutura ou semântica e métodos de interpretação daquilo que é ouvido ou lido.

Além de um veículo para a comunicação, as duas entidades devem possuir meios de se comunicar. Em um sentido amplo, uma interface é um dispositivo que serve de limite comum a várias entidades comunicantes, as quais se exprimem em uma linguagem específica a cada uma. Para que a comunicação seja possível, o dispositivo deve assegurar a conexão física entre as entidades e efetuar as operações de tradução entre os formalismos existentes em cada linguagem. Uma vez que a comunicação esteja estabelecida, a interação (ação recíproca) pode ocorrer entre as partes.

No caso da interface homem-computador, a conexão entre estas duas partes integrantes da comunicação se realiza entre a manifestação externa e os órgãos sensoriais e motores do usuário.

Nesta interface há uma linguagem de interação permitindo ao usuário, por meio de um vocabulário e de uma sintaxe, expressar as operações que se deseja efetuar. Estas operações manipulam comandos ou dados.

Segundo Hugo (1995), o vocabulário desta linguagem de interação pode ser formado de diferentes modos:

- Códigos numéricos;
- Códigos mnemônicos;
- Palavras da língua portuguesa;
- Palavras desconhecidas;
- Pictogramas (Sistema de escrita de natureza icônica, baseada em representações bastante simplificadas dos objetos do mundo real).

Os trabalhos de Furness e Barfield (1995), apresentam algumas vantagens de se empregar a tecnologia da fala nos sistemas de computação, dentre elas a possibilidade de não requerer treinamento do usuário, pois o uso da fala não exige que a pessoa seja treinada ou tenha determinadas habilidades.

Com a fala é possível uma maior agilidade na requisição de comandos onde a informação é fornecida mais rapidamente que quando digitada e o usuário pode utilizar o sistema enquanto está se movendo ou fazendo uma outra atividade que requer o uso das mãos.

Com ênfase em linguagem falada, destacam-se duas vantagens na sua utilização em um sistema de comunicação homem-máquina, a primeira diz que a fala é indispensável em algumas situações, para substituir outros canais de comunicação, como a relação piloto-avião ou para deficientes físicos.

Já a segunda vantagem postula que a fala pode ser mais eficaz que outros modos de comunicação. Em comparação à linguagem escrita, a fala permite melhor desempenho em questões de tempo de resolução de problemas e emissão de mensagens.

Os sistemas tendem a tornar-se mais complexos à medida que a interação e o reconhecimento da fala e o diálogo entre a máquina e o homem se torna mais evoluído.

O ser humano ao se comunicar com o computador sem utilizar dispositivos convencionais (teclado, mouse, etc.) torna-se mais livre para desempenhar funções onde seus membros são necessários.

A interação por meio do comando de voz sendo ela por Reconhecimento Automático da Fala, ou por comando e controle faz com que o usuário sinta-se mais imerso e entretido com Ambiente Virtual.

Mas alguns usuários podem perder a imersabilidade e a interatividade quando o sistema não possui uma resposta em tempo satisfatório ou tem um tempo latência grande entre o reconhecimento e a execução do comando.

Algumas aplicações de reconhecimento de voz foram usadas em Ambientes Virtuais utilizando sua forma básica sem se preocupar com a interatividade e imersão dentro do ambiente, como é o caso das aplicações descritas por Hunt e Walker (2000) e por Pizzolato e Rezende, (2003).

Notavelmente a pesquisa de Oliveira et. al. (2000), decifra alguns problemas dentro do Ambiente Virtual associado a um sistema de reconhecimento de fala. Outros trabalhos descrevem apenas os erros de rejeição ou de incompreensão dos comandos de voz como é o caso dos trabalhos de Meiguins et ali (2003), e de Rodrigues (2001).

O Trabalho de Nguyen(2003), postula alguns aspectos da comunicabilidade entre os usuários de sistemas de RV e os avatares humanóide, entretanto não destaca a queda de desempenho atribuída a renderização e ao uso de processos para os serviços de fala.

Trabalhos relacionados à computação ubíqua utilizando dispositivos de realidade virtual e aumentada com serviços de fala são abordados nos experimentos de Goose et. Al (2003), mas o enfoque do artigo não abrange a usabilidade do usuário nem os problemas de erros de rejeição de comandos.

3. BIBLIOTECAS DE PROGRAMAÇÃO PARA RECONHECIMENTO E SÍNTESE DE VOZ

O avanço dos conhecimentos em reconhecimento e síntese de voz faz com que as empresas desenvolvedoras desta tecnologia adotem novos métodos de processamento dos sinais de fala que possam levar a atenuação dos erros de reconhecimento ou incompreensão da mensagem falada.

Existem diversos sistemas de reconhecimento e síntese de fala, conforme pode ser visto na Tabela 1, nesta pesquisa foram relatados os fabricantes e os idiomas padrão de reconhecimento e síntese de fala.

O Objetivo desta parte do trabalho foi identificar os fabricantes juntamente com seus produtos, os datados mostrados na Tabela 1 foram estabelecidos de acordo com cada documentação fornecida diretamente pelo fabricante.

Os produtos da Tabela 1, mostram como ferramentas de aumento de produtividade mas no entanto, sua taxa de acurácia não passa dos 95 % em alguns casos como afirma Hoson (2003), onde a pesquisa destaca que estes produtos ainda estão em fase de aprimoramento e que de acordo com a tecnologia e a abordagem utilizada para o reconhecimento da fala é possível chegar a uma acurácia superior a 97 %.

Tabela 1 - Sistemas Reconhecedores de Fala

Produto	Fabricante	Idioma Reconhecedor
IBM-ViaVoice	IBM	Port / Ing / Esp
Dragon Naturally Speaking	Scan Soft	Port / Ing
Microsoft Speech	Microsoft	Ing / Jap / Chi
Free Speech	Philips	Ing / Jap / Chi
Voice Xpress	L & H	Ing / Esp
20/20 Speech	Arix	Ing / Jap / Rus
Listem for Windows	Verbex Voice Systems Inc.	Port / Ing / Ale
In Cube	Command Corp	Port / Ing
Phonetic Engine	Speech Systems Inc.	Ing / Fra / Ale
PureSpeech Recognition Engine	Pure Speech	Ing / Ale / Ita
PowerSecretary	Articulate Systems	Ing / Ale / Ita /Jap
Vpro-XD	Voice Processing Corporation	Ing / Fra / Esp
PlainTalk	Apple	Ing / Fra / Ale / Jap
SpeechManager	Silicon Graphics Inc.	Ing

Além do sistema reconhecedor a ser escolhido para a aplicação tridimensional é importante notar também qual é a biblioteca de programação ou a interface de programação que o sistema permite utilizar para criar-se produtos com base nos serviços de fala disponível juntamente com as bibliotecas de processamento do ambiente virtual.

A Tabela 2 destaca que a maior parte das ferramentas de produtividade possuem bibliotecas que podem ser acessada tanto com a linguagem Java quanto C/C++ .

Tabela 2 - Biblioteca de Programação de acordo com o produto

Produto	Biblioteca de Programação	Linugagem
IBM-ViaVoice	Java Speech / MS-Speech API	Java ou C++
Dragon Naturally Speaking	MSSpeech 4.0	Java ou C++
Microsoft Speech	MS-Speech API / Java Speech	Java/C++/Delphi/VB
Free Speech	FSAPI	Java ou C++
Voice Xpress	Java Speech / MSSpeech 4.0	Java ou C++
20/20 Speech	ARIX Speech API	C++
Listem for Windows	MS-Speech 4.0	C++
In Cube	ICSpeech API / MS-Speech 4.0	C++
Phonetic Engine	Java Speech / MS-Speech 4.0	Java ou C++
PureSpeech Recognition Engine	Java Speech / MS-Speech 4.0	Java ou C++
PowerSecretary	Java Speech / MS-Speech 4.0	Java ou C++
Vpro-XD	Java Speech / MS-Speech 4.0	Java ou C++
PlainTalk	Java Speech	Java
SpeechManager	SGI Speech	C++

O desenvolvimento de um ambiente virtual conduzido por voz por meio da técnica de comando e controle de voz, deve ser implementado de forma que se obtenha o máximo desempenho da linguagem de programação escolhida e o máximo de rendimento do hardware, e para tal, é necessário que a equipe de desenvolvimento escolha a linguagem e a biblioteca de programação adequada e que seja compatível com a biblioteca gráfica desejada.

Por isso a escolha de qual API será empregada no desenvolvimento de um sistema deve ser feita a partir de uma avaliação minuciosa dos critérios a seguir:

- **Redirecionamento de amostras de áudio**: é primordial que a API permita o redirecionamento das amostras de áudio produzidas pelo mecanismo de síntese de voz para arquivos e/ou outros dispositivos além da saída de áudio padrão.

- **Mecanismos disponíveis**: deve-se avaliar a qualidade dos resultados produzidos pelos mecanismos de síntese de voz que podem ser manipulados com a API bem como seus custos financeiros.

- **Facilidade de uso:** a Biblioteca de programação, ou a API deve ser sucinta de forma que seu aprendizado e conseguinte emprego no desenvolvimento de aplicações seja simples e rápido.

- **Configuração de parâmetros:** é desejável que a API permita ao desenvolvedor configurar os parâmetros de síntese de voz como velocidade, afinação, timbre, língua, etc. tanto por invocação de funções, procedimentos e/ou métodos como por meio da inserção de marcações no texto a ser sintetizado.

- **Linguagem de Programação:** é interessante que a API tenha sido escrita na mesma linguagem empregada no desenvolvimento do sistema onde o engenho de síntese de voz será embutido, o que certamente facilitará sua adoção. Caso a API tenha sido escrita em uma linguagem diferente é desejável que exista um *binding* (ponte ou ligação) para aquela na qual a aplicação foi desenvolvida, caso contrário será necessário empregar algum mecanismo que permita a interação entre diferentes linguagens o que tende a dificultar o desenvolvimento e a manutenção da aplicação.

- **Portabilidade:** é desejável que a API possa ser utilizada no maior número possível de ambientes de desenvolvimento e sistemas operacionais.

Dentre as bibliotecas disponíveis no mercado foi feita uma avaliação das APIs Java Speech API, SUN (1998) , MS-SAPI - Microsoft *Speech API,* MICROSFT (2000), descritas no trabalho de Damasceno (2004) em que são considerados os aspectos supracitados. Alguns critérios tais como a qualidade dos mecanismos de síntese de voz que podem ser manipulados com a API e facilidade de uso foram avaliados de forma subjetiva. A Tabela 3 a seguir sumariza os resultados obtidos.

Na análise descrita em ZDNet(2001) destacam-se dois sistemas de reconhecimento e síntese de voz com os quais este trabalho se manteve em foco, o IBM ViaVoice 8.0 e o Microsoft Speech.

Tabela 3 - Avaliação das API (Java Speech e MS API)

Critério	Java Speech	Microsoft Speech
Versão da Biblioteca utilizada	1.1	5.0
As amostras de áudio podem ser redirecionadas para arquivos e/ou outros dispositivos	Não	Sim
A API está acoplada apenas a um único engenho de síntese de voz	Não	Não
Idioma Padrão	Inglês	Inglês
Aceita o idioma Português – Brasil	Sim	Não
Classificação da qualidade dos resultados	Boa	Boa
Como é classificado o aprendizado e o emprego da API	Fácil	Difícil
A API permite configuração de parâmetros	Sim	Sim
Linguagem na qual a API foi desenvolvida	Java	C++
A Biblioteca API possui *bindings* com outras linguagens	Não	Sim
Em quais ambientes de desenvolvimento a API pode ser utilizada	JDK	MVC++
Aceita Programação com Bibliotecas 3D	Sim	Sim
Biblioteca Padrão 3D	Java3D	Direct-X

3.1 Limitações das Bibliotecas de Programação

As limitações para o Reconhecimento de Voz são descritas como erros de acurácia e de entrada de áudio e segundo SUN (1998) podem ser classificadas como: Erros de Rejeição de Comandos, Erros de Pronúncia e Erros de Áudio;

Os erros de rejeição de comandos são mostrados quando o usuário fala, mas o reconhecedor não entende o que foi dito, ou que foi pronunciado não existe referência na gramática.

Os erros de pronúncia são estabelecidos quando o mecanismo reconhecedor reconhece uma palavra diferente para o que foi pronunciado.

Os erros de áudio são ocasionados por causa de alguma interferência, o reconhecedor admite como pronunciado algum ruído, entretanto o usuário não pronunciou alguma palavra,

mas o reconhecedor a reconheceu. Os mecanismos reconhecedores de fala cometem erros, e estudar estes erros para perceber que falha ocorreu, pode ajudar muito a minimizar o novo surgimento dos mesmos erros em outras ocasiões. Compreender porque ocorreu o erro, os fatores que geraram este erro e como treinar o usuário para minimizar a quantidade de erros ocorridos, são importantes conceitos para se desenvolver uma aplicação que utiliza o serviço de fala.

Tabela 4 - Erros de Reconhecimento e possíveis causas. SUN (1998)

Problema	Possível Causa
Rejeição ou Erro de Pronúncia	O usuário pronuncia uma palavra fora do vocabulário
	A sentença do usuário não está descrita na gramática
	O usuário fala antes do sistema estar pronto para ouvir
	Palavras no vocabulário tem a fonética semelhante, por exemplo, pata, bata.
	O usuário tem uma pausa muito demorada no meio de uma sentença
	O usuário gagueja ou tem uma pronúncia infalsa
	O usuário não termina a pronúncia no fim da sentença
	O usuário está ferido ou com frio
	A voz do usuário é muito diferente da voz padrão armazenada nos modelos padrões de voz
	A entrada áudio do computador está mal configurada
	O microfone está mal configurado.
Erros de Áudio	Sons vocálicos não falados (Tossir, rir, bocejar)
	Fala / Som de fundo ativa o reconhecimento de voz
	O usuário fala com outra pessoa e o computador reconhece as palavras do diálogo como comandos

A acurácia é usualmente fornecida em percentagens de palavras que foram reconhecidas e confirmadas pelo usuário numa gramática livre e o desenvolvedor deve considerar os seguintes fatores: (1) Percentual de erros deve ser baixo para ambientes silenciosos; (2) Microfones de alta qualidade e equipamentos de áudio podem melhorar a acurácia; e; (3) usuário deve falar claramente, mas naturalmente, sem inflexões de voz ou ênfase em algum fonema;

A Tabela 4, demonstra as possíveis causas dos erros de reconhecimento. Os mecanismos reconhecedores de fala cometem erros, e estudar estes erros para perceber que

falha ocorreu, pode ajudar muito a minimizar o novo surgimento dos mesmos erros em outras ocasiões. Compreender porque ocorreu o erro, os fatores que geraram este erro e como treinar o usuário para minimizar a quantidade de erros ocorridos, são importantes conceitos para se desenvolver uma aplicação de reconhecimento de voz.

Utilizar uma gramática simples pode ajudar a melhorar o percentual de palavras reconhecidas.

Sistemas de ditado possuem uma gramática complexa, mas utilizar-se de bons microfones, ambientes silenciosos ou adaptados para receber processamento de áudio, e se o usuário tiver uma pronúncia clara e com um computador com recursos suficientes a percentagem de acertos é muito alta entretanto deve-se levar em consideração que o ambiente onde está sendo realizado os testes de reconhecimento deve ser silencioso além dos dispositivos estarem devidamente regulados e com grau de impedância baixa.

4. REALIDADE VIRTUAL

Segundo Pessoa e Araújo (1999), a associação dos termos realidade e virtual é um contra-senso, pois o primeiro refere-se àquilo que existe de fato, enquanto que o segundo refere-se àquilo que se pode vir a existir, mas ainda não se concretizou.

Realidade Virtual (RV), é o estado da mente no qual se modificam aspectos da cultura e a maneira de vivê-la quotidianamente. Ela é um ambiente no qual o usuário participa de um mundo "virtualmente" real, que o faz pensar que está realizando ações, quando na verdade não está, porque é tudo imaginário.

Pode-se pensar que real é tudo aquilo o que se pode ver, sentir, tocar e ouvir. Porém todos esses sentidos não passam de estímulos elétricos interpretados pelo nosso cérebro. Dessa forma, pode-se reproduzir todos estes estímulos artificialmente e fazer com que uma pessoa sinta-se em um ambiente que ela imagine ser o real. É disto que trata a RV, de ambientes criados artificialmente, mas que provêem as sensações de um ambiente real.

Pode-se relacionar várias interpretações e definições para o termo RV. De acordo com Burdea e Coiffet (1994), a RV é uma simulação na qual a computação gráfica é usada para criar uma visão de mundo real. No entanto, o mundo sintetizado não é estático e responde às

ações do usuário (gestos, comandos verbais, etc.) proporcionando a interatividade em tempo real.

Sendo a RV a forma de interação mais avançada de interface homem-computador permitir a interação em um Ambiente Virtual (AV) onde o homem possa se comunicar com o computador por meio de uma linguagem, deixando para trás o teclado e o mouse, e apenas usando a voz para suas requisições, provoca uma mudança na concepção dos sistemas.

Esta mudança de concepção é no sentido de usar canais multisensoriais, tais como visão, audição e fala para que o usuário possa imergir no ambiente.

Cabe salientar que o AV exige interfaces multimídia interativas requerendo tecnologias e dispositivos apropriados, capazes de envolver o cérebro humano no vasto mundo do virtual.

4.1 Ambiente de Realidade Virtual

Conforme Burdea e Coiffet (1994), um Ambiente de Realidade Virtual, ou apenas Ambiente Virtual (AV), é um ambiente computacional de alto desempenho, que tem a intenção de iludir o usuário de melhor forma possível para que este seja convencido de que está realmente dentro do ambiente mostrado.

O AV pode ter várias formas representando prédios, objetos, pessoas, animais, mas, todas com certa precisão geométrica, cores, texturas e iluminação. E também o ambiente virtual pode não ter nenhuma referência física, constituindo um modelo abstrato. Mesmo assim, os atributos de cores, texturas, etc., continuam sendo importantes para uma boa visualização e imersão.

Furness e Barfield (1995), definem que um AV é uma representação de um modelo computacional ou de banco de dados o qual pode ser utilizado interativamente ou manipulado por um ou mais participantes.

Uma definição mais objetiva é dada por Singhal e Zyda (1999), que um AV é um mundo simulado por computador consistindo de representações computacionais de agentes reais ou imaginários, objetos e processos; além de equipamentos para interface homem-computador.

Segundo Syngal e Zyda (1999), um AV é o local de interação entre usuário - computador, que pode reproduzir a realidade de forma quase perfeita, ou criar uma fantasia onde atores virtuais representam um mundo real ou imaginário.

Atores ou Agentes Virtuais representam um modelo computacional de uma figura humana que pode mover-se e interagir em um AV. Se o comportamento de um ator virtual está ligado aos movimentos de um participante humano, este ator é denominado de um avatar.

Em AVs desenvolvidos mais recentemente é notável a presença de atores virtuais, personagens criados pela informática para auxiliar, informar, e interagir com usuários com características comuns numa forma amigável, seja ela representada por um humano, ou por uma forma gráfica humanóide de imagem agradável, tal característica visa conquistar a atenção dos usuários, fazendo com que estes por sua vez se sintam à vontade para interagir com o personagem apresentado.

4.2 Representação de Usuários em Ambientes de Realidade Virtual

No dia-a-dia as pessoas possuem representações, isto é, um artifício qualquer capaz de caracterizar um indivíduo em particular. A carteira de identidade é um bom exemplo, pois, possui um número único e uma representação visual (foto) capazes de distinguir uma determinada pessoa na sociedade. Isso também é possível em ambientes de RV com o uso de avatares.

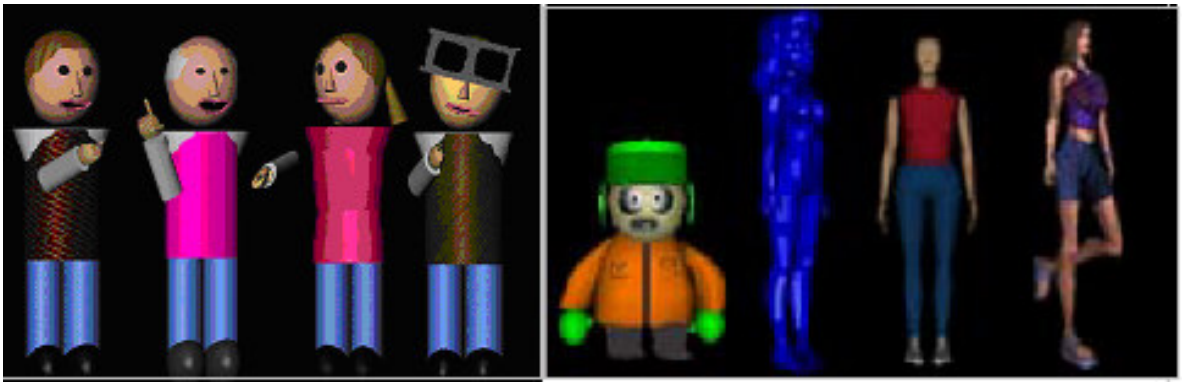
De acordo com Ferreira (1999), um avatar é uma representação gráfica de um usuário num AV. A palavra avatar é originária da mitologia hindu, onde "*avatãr"* significa a encarnação de um deus em um corpo mortal.

Uma representação virtual do usuário (avatar) representa um bom avanço no sentido de comunicação visual, pois assim pode ser possível observar as reações do usuário às mensagens enviadas e recebidas.

Quando se deseja que o avatar possua uma representação mais realística é necessário ter-se em mente alguns fatores físico e psicológicos que trazem para o usuário o sentido de imersão no sistema.

Segundo Çapin (1999), pode-se obter uma imersão do usuário ligado ao seu avatar por meio dos seguintes fatores: (1) percepção do mundo virtual; (2) localização no ambiente virtual; (3) representação e visualização de outros avatares e objetos virtuais; e a representação social com vestes ou formatos decorativos.

Um avatar pode ser construído, ou modelado, por composição de outras formas geométricas como na Figura 10 (a) ou por cálculos matemáticos com a utilização de softwares de modelagem tridimensionais como os humanóides da Figura 7 (b)



(a) composição por formas geométricas (b) criação por modelagem tridimensional

Figura 10 - Formas diferentes de avatar

Os avatares podem possuir diversas formas como mostra a Figura 10, como também podem ser considerados apenas símbolos, não necessita ser caracterizado como um ser humano, entretanto para ter maior sentido de imersão no ambiente faz-se necessário a utilização de representações mais elaboradas e que o usuário se identifique com a representação gráfica do avatar. Neste sentido é aplicado a utilização de humanóide ou humanos virtuais como é possível visualizar na Figura 11.

Figura 11 - Avatar cansado de tanto estudar (jogo "*The Sims*")

A escolha do layout do avatar é uma decisão que irá refletir diretamente no desempenho da aplicação. Representações mais complexas demandam maior capacidade de processamento, enquanto que soluções mais simples ou híbridas podem produzir resultados satisfatórios com custos reduzidos.

Na utilização de humanóides existe um maior envolvimento do usuário, pois o mesmo sente-se transportado para dentro da aplicação, onde pode se movimentar e interagir com o sistema por meio da sua própria representação.

Segundo Balder (1997), um agente é a representação da figura humana virtual, criada e comandada por computador e um avatar é um humano virtual controlado pelo usuário.

Segundo Thalmann (1995) eles podem ser representados por ícones 2D, filmes, formas 3D, e até por corpos completos. Ambos são muito utilizados no desenvolvimento de aplicações onde seja necessária a representação humana, podendo ser representados em AV.

De acordo com Badler (1997), um humanóide deve ser o mais realístico possível, respeitando as limitações do equipamento. Além dos objetos que compõem o avatar humanóide, representando cada parte do corpo humano, esse objeto deve ser interligado por juntas e articulações.

A Figura 12 mostra o modelo implementado no trabalho de Brega et al. (2003), existem juntas entre: cabeça e o pescoço, pescoço e o ombro, ombro e o cotovelo, cotovelo e pulso, ombro e cintura, cintura e o quadril, quadril e o joelho, e joelho e o tornozelo.

Para se alcançar maior realismo, o problema básico é a definição correta de juntas e articulações, as quais também devem possuir graus de liberdade (*Degree Of Freedom* -DOF).

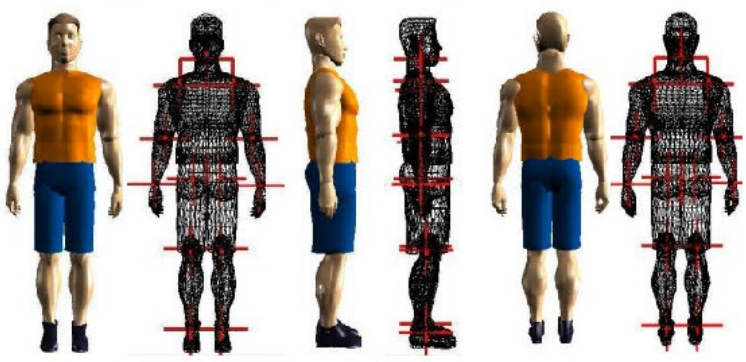


Figura 12 - Modelo de Avatar Humanóide de Brega et al (2003)

No estudo dedicado a implementação do Avatar Humanóide em WorldToolKit, a modelagem do mesmo é um pouco menos elaborada, como pode-se observar na Figura 13, onde nota-se que a implementação da biblioteca é de demonstração dos movimentos.

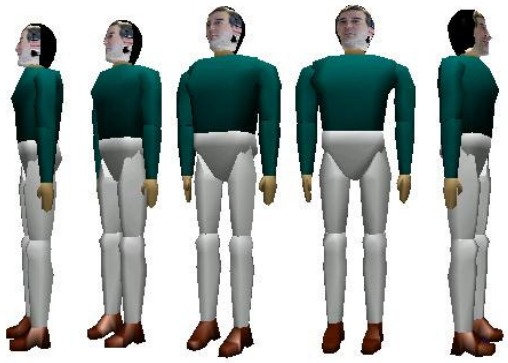


Figura 13 - Modelo de Avatar Humanóide de Brega et al (2001)

De acordo com Brega et al (2001), a partir do modelo mostrado na Figura 13, é possível propor alterações nos objetos para simular uma movimentação humana.

Segundo os estudos realizados no mesmo trabalho, para que as interações obtivessem maior realismo possível é possível à representação por equações de cinemática, deixando o modelo humanóide virtual com movimentos mais realísticos.

5. CONSIDERAÇÕES NO USO DAS BIBLIOTECAS

Nesta seção são apresentadas algumas considerações sobre o uso das bibliotecas de desenvolvimento de AVs controlados por voz, ou seja, a comparação entre as API´s de desenvolvimento Java Speech e Microsoft – Speech para construção de AVs.

As premissas desta dissertação são de que para cada tipo de ambiente onde se deseja ter o controle por voz existe uma limitação de recursos computacionais. Estas poderão ser atenuadas se atendidas as considerações da discussão deste trabalho.

5.1 A Biblioteca Java Speech

Esta biblioteca é uma interface de software capaz de manipular diversos mecanismos de reconhecimento e também de síntese de fala de fabricantes diferentes. Isto pela sua característica *coss-plataform* (portável entre plataforma) que é intrínseca da linguagem Java.

É uma tecnologia no estado da arte, segundo a IBM (2002), e SUN (1999), esta interface ou bibliotecas de software fornece e habilita um programa escrito em Java à sintetizar vozes (de timbres masculinos e femininos, sendo estas vozes atribuídas a pessoas de idades variando entre a criança e idoso) e reconhecer comandos de fala e ditado (de um locutor pré-definido ou por um padrão de voz sem locutor).

A Java Speech API (JSAPI), desenvolvida pela SUN (1999) em cooperação com diversas empresas de tecnologia de fala, define uma interface que fornece aos desenvolvedores uma forma avançada e fácil de integrar esta tecnologia aos sistemas em desenvolvimento.

A JSAPI define um padrão de fácil utilização, sendo este um conjunto de classes abstratas e de interfaceamento com o serviço de voz, e escondem do programador os detalhes da programação do mecanismo de reconhecimento e síntese de voz.

O que é mostrado em forma de interface para o programador são dois núcleos os responsáveis pelo serviço de voz (síntese e reconhecimento).

O reconhecimento de fala fornece ao sistema a habilidade de escutar e entender a linguagem falada determinando o que foi pronunciado, ou seja, processa a entrada de áudio contendo um sinal de fala e converte em texto, podendo este ser um conjunto de estados ou de funções.

A síntese de fala realiza o processo inverso, ou seja, por meio de um texto escrito sintetiza voz. Com o padrão JSAPI, usuários podem escolher qualquer software ou fabricante de síntese ou reconhecimento de fala.

As vantagens da utilização do JSAPI são: (1) Fornecer suporte para síntese de fala, comando de voz e reconhecimento de ditado; (2) Fornecer uma plataforma robusta, independente de fornecedor; e (3) Integração com a tecnologia JAVA incluindo a Java Media API (para acesso a áudio, vídeo, gráficos em 2D e 3D).

O IBM ViaVoice foi escolhido como mecanismo a ser associada ao JSAPI neste trabalho pela sua facilidade de configuração e programação dos eventos de serviços de fala . É

possível identificar de forma clara funcionamento em camadas do mecanismo, como que pode ser observado na Figura 14.

Figura 14 - Arquitetura do IBM Speech for Java, IBM (2001)

Por ser escrita numa linguagem de livre domínio, JAVA, e por ser totalmente orientada a objeto, o projeto do mecanismo de reconhecimento respeita o processo de desenvolvimento proposto pelo JCP – *Java Community Process,* o qual especifica que para cada lançamento de uma nova API deve-se ter o acordo da comunidade de desenvolvedores Java que estabelece as classes interfaces e os processos de sincronização e troca de mensagens entre os objetos para que os fabricantes de produtos que poderiam utilizar a tecnologia possam ter acesso sem terem de criar *bindings* (complementos de código em outra linguagem).

5.2 A Biblioteca Microsoft Speech

A Biblioteca de programação para Windows, a Microsoft Speech API é uma biblioteca que provê recursos de síntese e reconhecimento de voz para aplicativos baseados na plataforma windows, utilizando o mecanismo Microsoft Speech, (MICROSOFT 2000). Este conjunto de rotinas de programação consiste de um conjunto de arquivos DLL (bibliotecas dinâmicas) que contém os códigos de programação necessários para dar funcionalidade a

biblioteca como um todo, o quadro da Tabela 5 mostra alguns destes arquivos e suas atribuições, todos encontrados na pasta C:\Windows\Speech\

Tabela 5 - Arquivos DLL para reconhecimento de voz no SAPI

Arquivo	Funcionalidade
speech.dll	Responsável por ligar o mecanismo de reconhecimento pelo microfone
spchtel.dll	Responsável por associar o mecanismo de reconhecimento ao telefone
vcmshl.dll	Responsável por associar o mecanismo de reconhecimento ao sistema operacional
Xlisten.dll	Responsável por receber as mensagens passadas pelo mecanismo de reconhecimento
Xvoice.dll	Responsável por controlar o tipo de voz associado e o usuário genérico
Xcommand.dll	Responsável por associar as regras de comando e controle para o mecanismo e reconhecimento

A representação da utilização dos arquivos DLL pode dar da forma expressa na Figura 15 que destaca é que para reconhecimento de fala é necessário a atribuição das tarefas associadas ao processo de reconhecimento do que foi pronunciado, (realizado pelo arquivo Xlisten.dll) e a conferência com a gramática realizado pelo arquivo Xcommand.dll e ambos os arquivos estão no laço de repetição das mensagens geradas pelo sistema operacional.

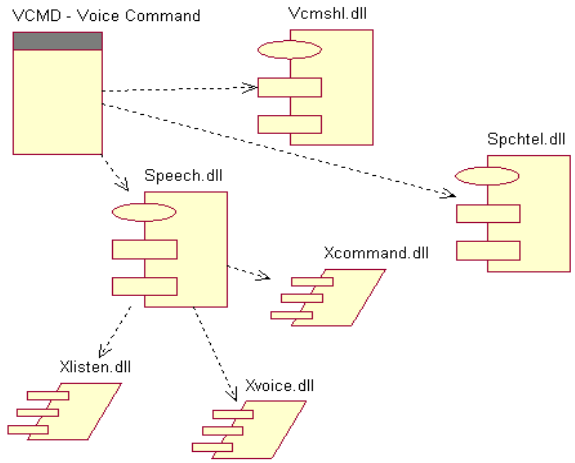


Figura 15 - Dependência entre os arquivos DLL de comando de voz

Além deste conjunto de bibliotecas associadas à tecnologia de voz disponibilizada em Microsoft (2000), possui um outro conjunto de bibliotecas que auxiliam a programação de novos sistemas de síntese e reconhecimento de fala, O suporte de programação é forneceido por um conjunto de bibliotecas é denominada de Microsoft Speech SDK (*Software Development Kit*).

O Microsoft Speech API 5.1 é um conjunto de bibliotecas de desenvolvimento para o sistema operacional Windows que fornece ao desenvolvedor a possibilidade de utilizar uma interface de voz para fazer a interação com a aplicação desenvolvida. Neste conjunto de bibliotecas é possível criar funcionalidade de síntese e de reconhecimento de voz.

Com o Speech SDK pode-se criar componentes (*ActiveX ou DLL*) próprios baseados nesta interface que podem ser utilizados em uma plataforma Win32.

É possível utilizar as linguagens C/C++, Java, Delphi, Visual Basic, para acessar as bibliotecas de serviços de fala deste conjunto de desenvolvimento, entretanto deve-se levar em conta que quanto mais próximo a linguagem estiver do formato de desenvolvimento do Speech SDK, mais rápida ficará a aplicação final. O Speech SDK foi desenvolvido em linguagem C++.

O MS-Speech SDK não é uma aplicação para um usuário final, uma biblioteca de interface de tela ou de voz com facilidades do tipo botões de comando ou menus, ele é um conjunto de rotinas escritas em C++ que provê o desenvolvedor facilidades para criar aplicações incorporando o serviço de fala como fator de acessibilidade ao sistema.

6. AMBIENTES DESENVOLVIDOS

Para a efetiva comparação das bibliotecas de programação de serviços de fala JSAPI e MS-SAPI, foram desenvolvidas aplicações que unindo a tecnologia de serviços de fala e Realidade Virtual, criando um Ambiente Virtual (AV) contendo um avatar humanóide que respondesse a comandos falados pelo usuário.

As linguagens e seus paradigmas de programação escolhidos foram Java por retratar bem o paradigma de orientação a objetos e a linguagem C/C++ para Windows com ambiente de desenvolvimento Microsoft Visual C++ (MVC++), onde se pode encontrar características de uma linguagem orientada a objeto.

Quanto à gramática de controle utilizada, esta representa os comandos definidos por Brega et. Al. (2001), tanto para a aplicação em Java quanto em MVC++.

6.2 Aplicação em Java – Protótipo 1

Para o desenvolvimento da aplicação em linguagem Java, foram observados os conceitos de programação descritos em Damasceno (2004), para incorporar o sistema de reconhecimento de fala ao AV.

Este protótipo foi desenvolvido com Java, utilizando a API Java3D bem como a API JavaSpeech 1.1 por meio do IBM ViaVoice, que está demonstrado na Figura 16 .

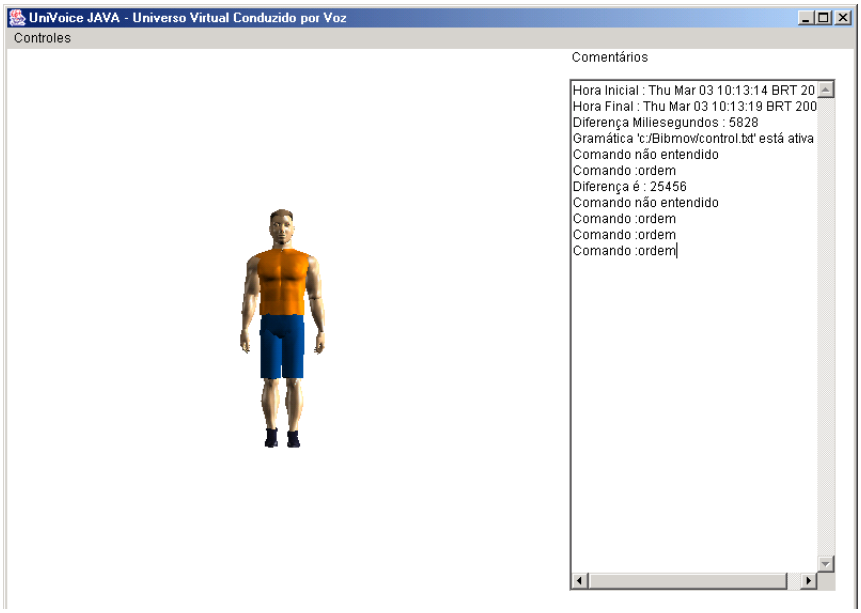


Figura 16 - Protótipo Java para o estudo comparativo

Neste protótipo chamado Protótipo 1, foi utilizada uma gramática de regras que suporta o diálogo discreto entre uma aplicação e o usuário. Segundo Damasceno (2004), o que pode ser dito é definido explicitamente por meio de um conjunto de regras descritas utilizando uma linguagem denominada "*Java Speech Grammar Format*" ou JSGF.

Com o JSGF pode-se determinar os comandos esperados: frases inteiras, palavras isoladas ou palavras/frases seguindo um determinado padrão. As regras de uma gramática podem ser carregadas em tempo de execução através de um arquivo texto, uma URL ou uma regra por vez.

Segundo SUN (1998) o JSGF define uma representação textual para ser utilizada no processo de reconhecimento de voz. Gramáticas são usadas para determinar quais os

comandos que o reconhecedor deverá processar, e também para descrever os modos de pronúncia que o usuário poderá utilizar.

As regras são identificadas por nomes e possuem uma cadeia de caracteres associada representando o que foi dito. Elas podem ser alteradas, apagadas ou inseridas dinamicamente.

A Figura 17 mostra a gramática usada no protótipo 1 criado em Java, a qual destaca-se pela facilidade de uso e pela recursão dos comandos.

```
grammar  Bibmov;
public <ordem> = avatar <numero> <ação> | avatar <numero> <direção>;
public <numero> = um | dois | três | quatro | cinco | seis | sete | oito
| nove | dez (numero);
public <ação> = sentar | levantar | (posição | onde) | (andar <numero>
passos a <direção> | caminhar) | parar | acenar | sim | não | finalizar
| pular | voltar  <ordem>;
public <direção> = direita | esquerda | frente | trás <ordem>;
```

Figura 17 - Gramática usada no Protótipo 1

A aplicação apresentada baseia-se nos trabalhos de Brega et al. (2001), onde foram propostas bibliotecas de movimentação de agentes e avatares humanóides, para aplicações de RV. A incorporação da biblioteca de movimentos veio favorecer o desenvolvimento da aplicação para a comparação dos recursos de voz e processamento da plataforma gráfica desenvolvida em Java 3D.

Os movimentos do avatar são definidos pela biblioteca de movimentos definida por Brega et al. (2003), e a frase composta em português é por exemplo: "***avatar um andar dois passos a direita***", conforme pode ser visto na Figura 17.

Na Figura 18 é possível notar que o ambiente também possui controle do tipo menu para que o usuário possa posicionar o avatar em uma outra posição. Nesta também é

observado no canto superior direito uma seção destinada aos comentários sobre o sistema, onde o usuário poderá observar as características de processamento dos comandos e do AV.

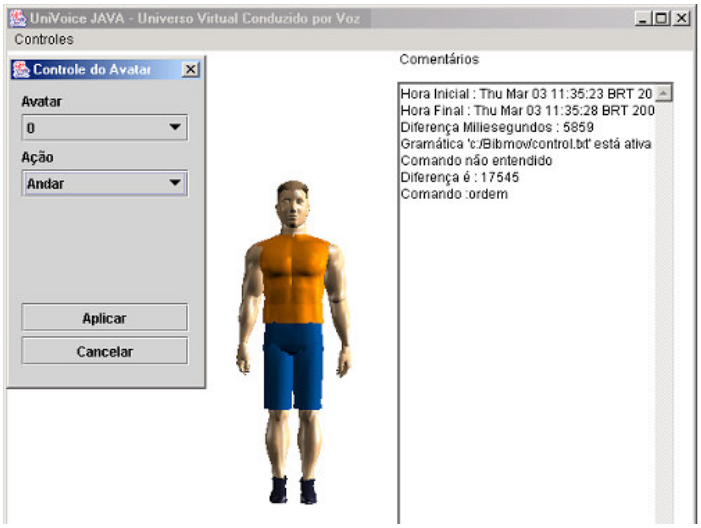


Figura 18 - Controles de manipulação do ambiente do protótipo 1

O ambiente também possui um *feedback* audível por síntese de fala por meio da tecnologia de Text-to-Speech (TTS), definida por DUTOIT(1999), que dá as boas vindas ao usuário ao entrar no sistema, bem como mostra um auxílio para a localização do avatar no ambiente tridimensional.

Os controles de manipulação do ambiente virtual são demonstrados no canto superior esquerdo, estes controles possuem os movimentos especificados por Brega (2001), na parte direita da aplicação estão os comentários do sistema, onde é mostrado ao usuário do ambiente o *benchmark* para futuras avaliações e os comandos de voz foram reconhecidos e executados pelo avatar no ambiente.

6.3 Aplicação em C – Protótipo 2

A aplicação objeto de estudo doravante definida como protótipo 2 foi criado em linguagem C/C++ para Windows com o uso do ambiente de desenvolvimento Microsoft Visual C++ 6.0 (MVC++) da Microsoft.e utiliza a API WorldToolKit da Sense8 com acesso a API Microsoft Speech 5.1 através das bibliotecas SAPI – Speech API da Microsoft que é demonstrado na Figura 19.

O protótipo 2 é construído com base na API do WorldToolKit da Sense8, o qual pode ser executado como uma janela console do Windows.

Uma das características da implementação em MVC++ , é o fato de que existe uma janela de controle Windows com a qual a API é instanciada, e referida, pois para o reconhecimento de voz é necessária a comunicação via mensagens (*Message Passing*).

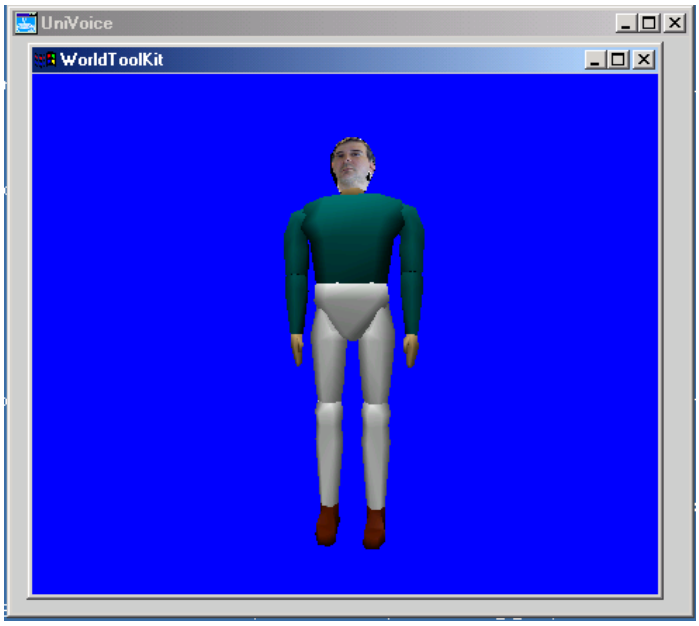


Figura 19 - Protótipo MVC++ para o estudo comparativo

É com o recurso de troca de mensagens entre o sistema de serviços de fala MS-SAPI e o WorldToolKit que é possível simular a abordagem multitarefa e iludir o usuário.

No protótipo 2, é observado que o avatar está sob uma janela windows com a qual as rotinas de programação do World ToolKit troca mensagens.

Os comandos de interação por voz são definidos em inglês devido a da limitação da biblioteca de serviços de fala da Microsoft, pois os idiomas que ela habilita o desenvolvedor são: o inglês, o japonês e o chinês, conforme Microsoft (2003).

Diferente das regras de gramática criada em JSGF, as regras de gramática necessárias para o MS-SAPI são livres contexto ou CFG – *context-free grammar,* que também é um conjunto de estruturas que definem quais palavras poderão ser pronunciadas, e este conjunto pode ser identificado por *tags* (marcadores) em linguagem XML, conforme é mostrado na Figura 20.

```
<GRAMMAR LANGID="409">
    <DEFINE>
        <ID NAME="VID_Avatar"   VAL="1"/>
        <ID NAME="VID_Acao"     VAL="30"/>
        <ID NAME="VID_sentar"   VAL="31"/>
        <ID NAME="VID_levantar" VAL="32"/>
        <ID NAME="VID_posicao"  VAL="33"/>
        <ID NAME="VID_andar"    VAL="34"/>
        <ID NAME="VID_passo"    VAL="35"/>
        <ID NAME="VID_parar"    VAL="36"/>
        <ID NAME="VID_acenar"   VAL="37"/>
        <ID NAME="VID_sim"      VAL="38"/>
        <ID NAME="VID_nao"      VAL="39"/>
    </DEFINE>
    <RULE ID="VID_Avatar" TOPLEVEL="ACTIVE">
        <O>Please</O>
        <P>boy</P>
        <RULEREF REFID="VID_Acao" />
    </RULE>
    <RULE ID="VID_Acao" >
        <L PROPID="VID_Acao">
            <P VAL="VID_sentar">sit</P>
            <P VAL="VID_levantar">stand</P>
            <P VAL="VID_posicao">jump</P>
            <P VAL="VID_andar">walk</P>
            <P VAL="VID_passo">step</P>
            <P VAL="VID_parar">stop</P>
            <P VAL="VID_acenar">sign</P>
            <P VAL="VID_sim">yes</P>
            <P VAL="VID_nao">no</P>
        </L>
    </RULE>
</GRAMMAR>
```

Figura 20 - Gramática CFG utilizada no protótipo 2 em MVC++

Existe uma distinção na concepção de uma aplicação baseada em Java e outra construída na plataforma Windows MSVC++. Visualmente as interfaces de comunicação são semelhantes, mas as tecnologias implementadas em ambas linguagens são de paradigmas diferentes.

Quanto a este paradigma o que destoa é que a na programação Windows com o MSVC++ é na verdade um programa orientado a procedimento que simula uma orientação a objeto por meio do uso de mensagens.

6.4 Considerações na implementação do AV pelo MVC++

A plataforma de desenvolvimento Microsoft Visual C++ (MSVC++) é um ambiente integrado de programação de sistema que une as facilidades de um compilador com a interface amigável e intuitiva do windows aliada a imensidade de recursos disponíveis da linguagem C.

Com este ambiente de desenvolvimento é possível criar de forma mais acessível programas que se beneficiem da interface gráfica advinda do sistema operacional Windows.

Segundo Kruglinski (1997), a programação para Windows é diferente de velho estilo de programação procedural, pois mesmo utilizando recursos conhecidos deste paradigma a programação Windows recorre a tecnologia de uso de mensagens passadas a interrupções de programa.

De acordo com Sebesta (2002), para simular a orientação a objetos através de uma linguagem tipicamente procedural é necessário a criação de eventos de sistema ou *Handle*

Events, que nada mais são do que o envio de mensagens ao núcleo do SO causando uma interrupção e desviando o fluxo do programa temporariamente.

Segundo Patterson e Hennessy (2000), o enfoque de que o Windows ser um sistema operacional multitarefa preemptiva é destacado porque cada programa recebe uma fatia de tempo do processador, e, durante a sua fatia de tempo é que cada aplicação é de fato executada ou colocada em espera. Quando a fatia de tempo se esgota a próxima aplicação começa a se executada e a fatia anterior entra em estado de suspensão, aguardando a sua próxima fatia de tempo.

Portanto para um programa Windows possa reagir a eventos o desenvolvedor deve manipular o laço de mensagens.

Segundo Patterson e Hennessy (2000), a repetição de mensagens é uma parte de todas as aplicações Windows, seu objetivo é o de receber e processar mensagens enviadas pelo núcleo do SO.

```
int APIENTRY WinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR lpCmdLine,
                     int nCmdShow)
{
   MSG msg;
   MyRegisterClass(hInstance, WndProc);
    g_fpCurrentPane = EntryPaneProc;
    if ( SUCCEEDED( CoInitialize( NULL ) ) )
    {
        if (!InitInstance( hInstance, nCmdShow ))
        {
          return FALSE;
        }

        while (GetMessage(&msg, NULL, 0, 0))
        {
            TranslateMessage(&msg);
            DispatchMessage(&msg);
            SetupAvatar();
        }

       CoUninitialize();
    }
    return msg.wParam;
}
```

Figura 21 - Trecho de código do loop de mensagens

Quando uma aplicação é excetuada, são enviadas mensagens para ela quase que de forma contínua que são armazenadas até que possam ser lidas e processadas.

O trecho de código destacado na Figura 21 denota o caso do uso de mensagens para enviar a mensagens de configuração do ambiente virtual escrito em MVC++ chamando uma função para criação do mundo em WorldToolKit.

No trecho destacado a mensagem é recebida pela estrutura MSG, segundo Schildt (1997), todas as mensagens do windows respeitam o formato descrito na Figura 22.

```
Ttypedef struct tagMSG
{
   HWND hwnd; /* janela à qual se destina a mensagem */
   UINT message;    /* mensagem                        */
   WPARAM wParam;  /* parâmetros da mensagem          */
   LPARAM lParam;  /* parâmetros da mensagem          */
   DWORD  time; /* momento em que a mensagem foi  enfileirada */
   POINT pt;    /* coordenadas X,Y do mouse */
} MSG;
```

Figura 22 - Estrutura da mensagem do windows

6.5 Considerações na implementação do AV em Java

A linguagem de programação Java é uma linguagem orientada a objetos que foi projetada para ser leve, simples e portável a diversas plataformas e sistemas operacionais. Entretanto é uma linguagem segura e robusta no que tange os conceitos de escalabilidade, independência de plataforma, gerenciamento de memória, reuso e principalmente segurança.

A forma de se desenvolver um sistema na linguagem Java é divergente na forma de se desenvolver um sistema na linguagem C/C++, isto, pois sua a principal característica é a orientação a objetos e o reuso de componentes.

A linguagem Java utiliza-se de um recurso de programação de janelas e formulários denominados AWT que nada mais é do que um superconjunto de classes de funções de tela.

Segundo de Deitel e Deitel (2003), o pacote AWT (*abstract window toolkit*) estão associados com os recursos da interface gráfica com o usuário na plataforma local, e quando um programa que usa deste recurso é executado em outra plataforma sua aparência é diferente da original.

Além da utilização do recurso de programação de inclusão de pacotes identificados foi detectado que o uso do pacote de desenvolvimento swing é redundante para o Java, utilizando formulário na plataforma Windows, mesmo contrariando os ensinamentos de Deitel e Deitel (2003), pois ambas (a classe canvas3D e o *CANVAS* do AWT) são especificações da classe Graphics, que acompanha o pacote AWT, conforme pode ser vista na Figura 23

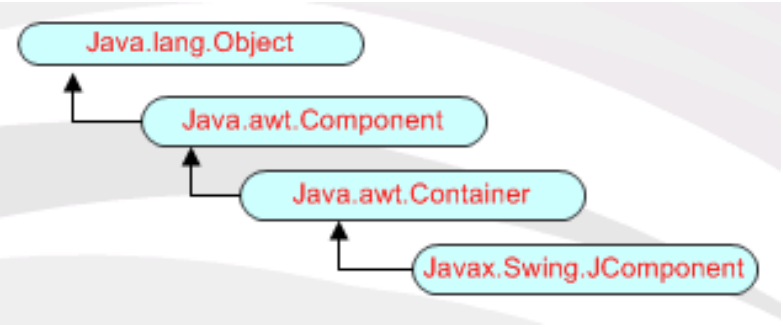


Figura 23 - Derivações de classes AWT, Deitel e Deitel (2003)

Os componentes do pacote Swing não são sobrecarregados pelos recursos GUI (*Graphical User Interface*) complexos da plataforma em que são utilizados, mas dependem da vinculação da AWT, do qual o Java3D se baseia para desenhar objetos na tela.

Mesmo que cada componente AWT possua uma interação separada para o acesso a plataforma, o fato de que quando é utilizado algum componente Swing junto ao AWT existe o que se pode denominar de perda por sobrecarga, ou seja, o pacote Swing por ser uma classe

derivada do AWT, este é processado primeiro e depois reprocessado para dar vazão ao pacote Swing.

É possível identificar visualmente quando o componente AWT é usado, Figura 25, pois a janela do usuário no ambiente windows é um pouco menos elaborada e dependente do sistema operacional Windows.

Figura 24 - Interface com o pacote Swing – Metal

Entretanto quando é utilizado o pacote Swing, Figura 24, a interface é mais elaborada, porém para o carregamento (*loading*) do conjunto de bibliotecas do Java 3D o processamento acaba se atenuando a medida que o Ambiente Virtual é carregado.

Figura 25 - Interface com o pacote AWT – Windows

Em C++ para utilizar-se dos recursos de interface gráfica baseada em janelas e formulários é necessário optar por dois caminhos distintos e de técnicas de programação diferentes.

6.6 Considerações sobre Java 3D (Sun) e World ToolKit (Sense 8)

Java 3D é uma API que se presta ao desenvolvimento de sistemas gráficos tridimensionais, servindo de ferramenta de desenvolvimento de interfaces para sistemas de Realidade Virtual.

Por ser uma adição da linguagem Java, esta por sua vez herda todos os atributos da linguagem o que facilita o desenvolvimento por ser orientado a objeto.

A classe da biblioteca Java 3D segundo Sowizral, Rushforth, e Deering, (1998), provê uma interface mais simples que a maioria das outras bibliotecas gráficas como o OpenGL, Direct3D, OpenInventor, World ToolKit, dentre outros disponíveis no mercado e por ser gratuita de *cross-plataform,* (independente de plataforma) facilita a portabilidade para vários ambientes operacionais o que conduz ao desenvolvedor uma maior rapidez no desenvolvimento de sistemas complexos e um nível mais elevado que outras APIs gráficas

Segundo os estudo divulgado em ShowCases (2000) a maioria das bibliotecas de programação tridimensionais são escritas em C/C++ e boa parte destas utilizam-se de módulos ou fazem referências as bibliotecas do OpenGL para realizar a renderização.

De acordo com Burdea e Boian (2001), o WorldToolKit é um ambientes de desenvolvimento de aplicações para realidade virtual para diversas plataformas e que contém mais de 1000 funções abstratas escritas em linguagem C/C++.

Sendo o Java 3D é uma interface criada para o desenvolvimento de aplicações gráficas tridimensionais executada no topo de bibliotecas gráficas de mais baixo nível, tais como OpenGL e Direct3D como destaca a Figura 26, e portanto dependente de duas outras camadas de acesso, a Native Graphic Call que é uma extensão do Java Media Framework da linguagem Java.

Segundo Barrileaux (2001), as imagens no Java não podem ser carregadas como uma operação de processamento linear, ou único. As imagens são carregadas com processos assíncronos e o status é monitorado até o final do carregamento onde os processos multiescalares são novamente sincronizados e devolvidos ao processamento linear.

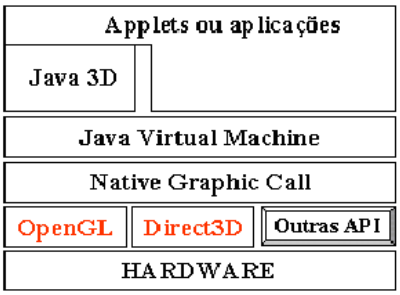


Figura 26 - Arquitetura do Java 3D

Segundo Deitel e Deitel (2003), a camada ou framework responsável pelo carregamento de arquivos de imagens é a Native Graphic Call que está diretamente ligada ao processamento das aplicações onde é fundamental o acesso eficiente de imagens gráficas.

Esse acesso é garantido pelo mecanismo de carregamento (*loading*) da Máquina Virtual Java, a sua principal tarefa é carregar os arquivos "*.class*" e interoretar os *bytecodes* neles presentes.

Como se pode ver na Figura 27, a máquina virtual contém um ***class loader***, responsável por carregar as classes, sendo elas desenvolvidas pelo programador, como as da *Java API*, e um **mecanismo de execução**, no qual este é o responsável por interpretar os *bytecodes*.

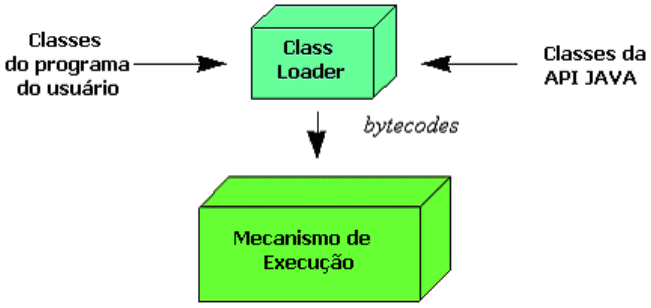


Figura 27 - Mecanismo de Execução da Máquina Virtual Java

Em *Java* há dois tipos de métodos: **Java e nativo**. Os **métodos Java** são escritos nesta linguagem, compilados para bytecodes e armazenado em arquivos ".class". Os **métodos**

nativos são escritos em outra linguagem diferente de *Java*, e compilados para a linguagem específica de um compilador particular e armazenados em uma *Dinamically Linked Library* cuja forma exata é totalmente dependente da plataforma, para a plataforma Java 3D são os seguintes arquivos listados na Tabela 6

Tabela 6 - Arquivos do Native Java

Nome do Arquivo	Uso
J3d.dll	Arquitetura J3D
J3daudio.dll	Arquitetura J3D – audio
J3DUtils	Arquitetura J3D – interação
awt.dll	Abstract Windows Tool Kit
Msvcrt.dll	Multiprocessamento – Threads
Java.dll	Máquina Virtual Java
Jawt.dll	Carregamento de arquivos com interface AWT
Fontmanager.dll	Gerenciador de Fontes

Segundo Deitel e Deitel (2003), enquanto que os métodos Java são independentes da plataforma, os nativos não o são. Quando um programa chama um método nativo, a Máquina Virtual Java carrega a biblioteca de vínculo dinâmico que o contém e o invoca. O uso desses métodos torna as aplicações específicas para um sistema, porém permite acesso direto aos recursos do *Java* não seriam possíveis, o que se pode concluir que os métodos nativos são a ligação entre um programa *Java* e o sistema operacional.

A arquitetura de carregamento dos arquivos de vínculo dinâmico é mostrada na Figura 28.

Segundo Venners (2001), a forma de carregamento de arquivos de vínculo dinâmico depende do sistema operacional utilizado. Pode-se notar que no ambiente operacional Microsoft Windows é realizado um carregamento de arquivos de modo simultâneo através do uso de threads, por isso a necessidade da biblioteca msvcrt.dll ser carregada no início do

processo. Uma aplicação *Java* pode usar dois tipos de *class loader*: Um *"primordial"* e o objeto *class loader*.

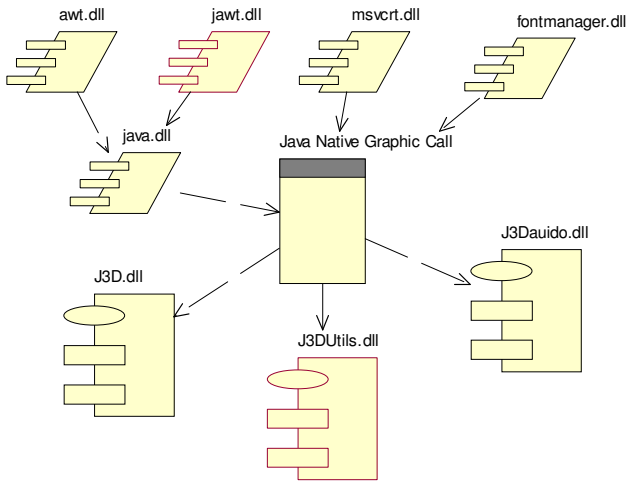


Figura 28- - Carregamento de arquivos de vínculo dinâmico

O *primordial* faz parte da implementação da máquina virtual, é único, e geralmente carrega classes do disco rígido. Em tempo de execução, uma aplicação pode instalar objetos *class loader*, que faz o carregamento das classes de forma personalizada, por exemplo, fazendo o descarrego*(download)* de classes através da rede. Estes "novos" *class loaders* são instanciados como outra classe *Java* qualquer.

A Máquina Virtual Java mantém um registro de qual *class loader* carregou uma determinada classe, e quando esta referencia outra, esta última será carregada pelo mesmo *class loader* da primeira, tendo este *class loader* que atualizar o seu próprio espaço de nomes, o qual é independente dos outros, como pode ser demonstrado na Figura 29

De acordo com Furgeri (2002) e Deitel e Deitel (2003) existe uma diferença de processamento na plataforma Java para implementações em sistemas operacionais tais como o Solaris e Windows.

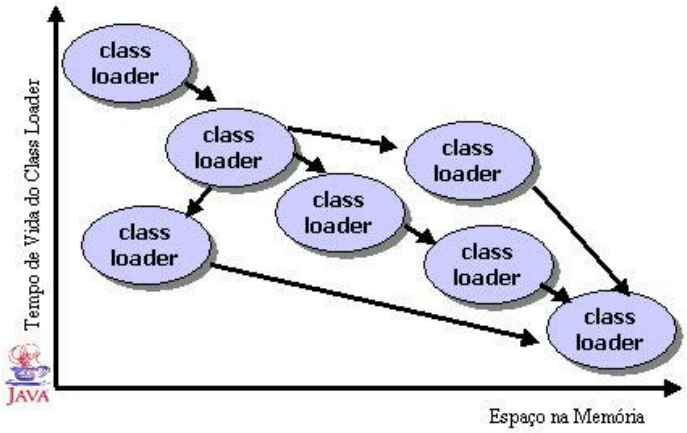


Figura 29 - Esquema geral do class loader

A plataforma Java Solaris executa uma thread de uma determinada prioridade até a sua conclusão ou até que uma thread de prioridade mais alta fique pronta e entre na fila de prioridades. Nesse ponto, segundo Patterson e Hennessy (2000), ocorre a preempção, que é a alocação no processador à thread de mais alta prioridade enquanto que a thread que estava sendo executada fica a esperar o término desta nova *thread*.

Nas implementações de 32 bits de Java para plataforma Windows, as *thread* recebem uma fração de tempo. Isso significa que cada *thread* recebe uma quantidade limitada de tempo para ser executada em um processador, e quando esse tempo expira, a *thread* é colocada em estado de espera enquanto todas as outras de mesma prioridade obtêm suas oportunidades de utilizar essa fração de tempo.

Segundo Sebesta (1996), o carregamento de arquivos na linguagem C/C++ é realizado em um único fluxo, visto que a linguagem pura é monoescalar, ou seja, possui apenas um único fluxo de execução.

Como o WorldToolKit é baseado em bibliotecas C/C++ é notório que o processamento do carregamento das imagens siga as mesmas convenções.

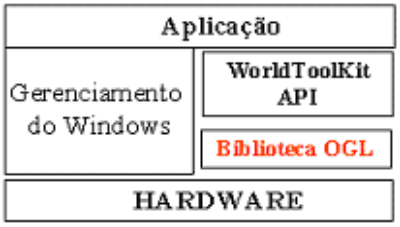


Figura 30 - Arquitetura do WorldToolKit da Sense 8

Na Figura 30 é possível notar que a biblioteca gráfica do WorldToolKit possui menos pontes de ligação, mas mesmo com menos pontes de ligação o processamento é linear, não permitindo a divisão do fluxo, causando uma pequena latência no carregamento de arquivos.

Segundo Kruglinski (1997) , a função WinMain é por padrão *Multihread*, pois as funções em destaque no código da Figura 31 são de características de programas que necessitam o multiprocessamento, pois por meio delas é possível o programa captar se o usuário pressionou o teclado ou clicou com o mouse, ou no caso deste trabalho pronunciou alguma palavra.

```
01  Int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE hPrevInstance,LPSTR
02      lpCmdLine,int nCmdShow){
03      MSG msg;
04      MyRegisterClass(hInstance, WndProc);
05      g_fpCurrentPane = EntryPaneProc;
06      if ( SUCCEEDED( CoInitialize( NULL ) ) ) {
07             if (!InitInstance( hInstance, nCmdShow )) {
08                return FALSE;
09             }
10             while (GetMessage(&msg, NULL, 0, 0))  {
11                TranslateMessage(&msg);
12                DispatchMessage(&msg);
13                SetupAvatar();
14             }
15          CoUninitialize();
16      }
17      return msg.wParam;
18  }
```

Figura 31 - Código da Função WinMain().

```
01  Int    APIENTRY    WinMain(HINSTANCE    hInstance,    HINSTANCE    hPrevInstance,
02             LPSTR lpCmdLine,int nCmdShow){
03      MSG msg;
04      HANDLE     hIOMutex= CreateMutex (NULL, FALSE, NULL);
05      MyRegisterClass(hInstance, WndProc);
06      g_fpCurrentPane = EntryPaneProc;
07      if ( SUCCEEDED( CoInitialize( NULL ) ) ) {
08             if (!InitInstance( hInstance, nCmdShow )) {
09               return FALSE;
10             }
11           while (GetMessage(&msg, NULL, 0, 0))  {
12             TranslateMessage(&msg);
13                DispatchMessage(&msg);
14              WaitForSingleObject( hIOMutex, INFINITE );
15                    SetupAvatar();
16              ReleaseMutex( hIOMutex);
17             }
18          CoUninitialize();
19      }
20      return msg.wParam;
21  }
```

Figura 32 - Função WinMain() Multithread

7. RESULTADOS

Os resultados relevantes deste trabalho estão relacionados com a efetiva comparação entre as bibliotecas de reconhecimento de fala incorporadas a Ambientes Virtuais.

Os resultados foram obtidos utilizando uma máquina Pentium III - Intel, com 256 Mb RAM, sem placa aceleradora de vídeo e com placas de som on-board, com as configurações mais próximas das utilizadas na pesquisa de Burdea e Coiffet (2001), para tentar encontrar detalhes diatônicos à incorporação do serviço de voz ao ambiente virtual. As ferramentas de software, bem como suas versões estão destacadas na Tabela 7, onde também se destaca o uso de ferramentas de uso gratuito e de outras proprietárias.

Tabela 7 - Ferramentas utilizadas

Requisito	Linguagem Java	Linguagem C
Versão	Java 1.4.2	ANSI
Biblioteca 3D	Java 3D 1.2	WorldToolKit 7.0
Biblioteca de Voz	Java Speech 1.1	Microsoft Speeh 5.1
Ambiente de Desenvolvimento	JED Plus 2.0	Microsoft Visual C++ 6.0
Mecanismo de Fala	IBM ViaVoice	Microsoft Speech

O sistema operacional escolhido foi o Microsoft Windows 2000 profissional, com os seguintes serviços iniciado escritos na Tabela 8.

Tabela 8 - Serviços inicializados no Windows 2000

	Nome do Serviço	*Arquivo em Execução*
1	Chama de procedimento remoto (RPC)	C:\WINNT\system32\svchost -k rpcss
2	Sistema de eventos do COM+	C:\WINNT\System32\svchost.exe -k netsvcs
3	Gerenciador de contas de segurança	C:\WINNT\system32\lsass.exe
4	Mensageiro	C:\WINNT\System32\services.exe
5	Extensões de driver de instrumentação e gerenciamento do Windows	C:\WINNT\System32\services.exe
6	Serviço de registro remoto	C:\WINNT\system32\regsvc.exe

Para o desenvolvimento na linguagem Java é necessário conhecer os serviços do componente do ViaVoice que são os seguintes:

- vvuiam.exe – ViaVoice User Interface Manager, que é necessário para que o sistema reconhecedor possa identificar a voz o usuário

- engine.exe – Mecanismo de tratamento do ViaVoice que reconhece o que foi pronunciado e compara com a gramática descrita.

Estes serviços ocupam boa parte da memória do computador e como são programas independentes podem ser invocados tanto incorporados ao AV, quanto por meio de *bidings* (pontes) de acesso usando o envio de mensagens (message passing).

Para o desenvolvimento do Protótipo 2 em ambiente Windows é necessário conhecer os seguintes programas:

- spchapi.exe – Biblioteca de reconhecimento de voz para Microsoft Speech 5.0

- svchost.exe – programa responsável pela agendamento de tarefas e trocas de mensagens na plataforma Windows.

Para a análise foi utilizado um computador Pentium III, com processador de 600 MHz, placa mãe modelo ASUS-PW3, contendo este um pré-processador matemático ASUS-HITECH 8118, placa de vídeo Intel 740c, cache de memória de 512 Kb, com disco rígido de 40GB de marca Quantum, sendo que este foi particionado em duas unidades de 15Gb e uma de 9 Gb, 256 Mb de memória RAM, microfone e caixas de som da marca MTk.

No Windows 2000 foram instalados os seguintes softwares descritos na Tabela 9, pra a plataforma P1 para ser usado com Java 3D e plataforma P2, para ser usado com WorldToolKit (WTK).

Os mecanismos para as plataformas foram o IBM ViaVoice 9 para P1 e Microsoft Speech Comand & Control, para P2.

Tabela 9 - Descrição das plataformas de software usadas

Plataforma	Software
Plataforma P1: *Java , Java 3D e* *Java Speech*	Java 2 SDK vs. 1.4.02 Java 3D vs 1.2 para OpenGL Editor JedPlus 2.0 Java Speech API IBM ViaVoice 8.0 Acelerador 3D ASUS OpenGL
Plataforma P2: *MS-Visual C++* *MS-Speech API, WorldToolKit*	WorldToolKit vs. 7.0 Microsoft Visual C++ 6.0 Microsoft Speech API SDK 5.0 Microsoft Speech Microsoft Command & Control 2.0 Acelerador 3D ASUS DirectX
Comum as Plataformas	Microsoft Office 2000 Medidor de Recursos ASUS PW3

Os métodos de avaliação de desempenho foram o tempo de resposta em vídeo e tempo de resposta do sistema em relação à geração do ambiente, que são destacados como motivos de perda de da sensação de imersão.

7.1 Avaliação dos dados obtidos

Os dados foram obtidos nas condições de silêncio no ambiente e com um microcomputador configurado conforme Tabela 9, onde as funções de atribuição e contagem de tempo decorrido foram estabelecidas por meio da média de dois métodos: : a) tempo registrado entre o início e o final do processo auferido por uma função da linguagem; b) registro as operações de tempo registrado pelo contador de ciclos registrado por uma função registrada no programa.

Ao final foi auferida a média de cada análise, e descartando-se os valores de limite inferior e limite superior, visto que estes dados poderiam refletir dados viciados, ou seja, poderiam conter lixo da memória ou dados já processados.

7.2 Taxa de quadros por segundo (Framerate per Second - FPS)

Nos estudos de Burdea e Coiffet (2001) constam que o WorldToolKit, na versão 7.0, aplicada para a plataforma MS-DOS é realmente mais rápido que o Java3D, visto que deve-se considerar que uma plataforma de console de comando é muito mais eficiente para aplicativos de realidade virtual, pois a mesma pode alocar todos os recursos para o mecanismo de renderização do ambiente tridimensional.

Quando a mesma aplicação é executada em uma plataforma de janelas como o MS-Windows, é notável que o tempo de renderização diminua, pois o controle passa a ser do sistema operacional e não mais da aplicação.

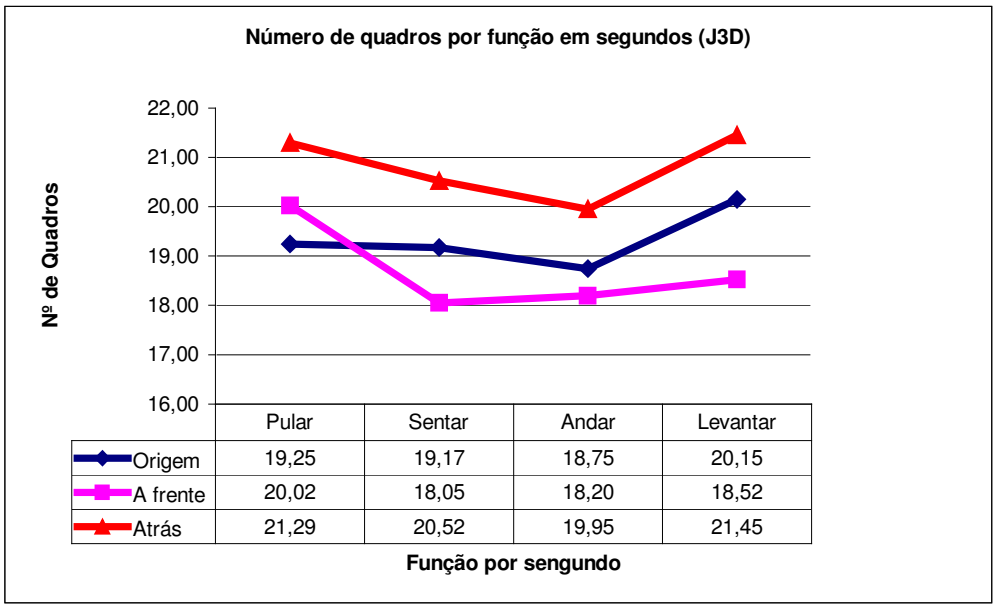
Já a aplicação de Java3D, utiliza os recursos de controle de janela do MS-Windows e, portanto tem um desempenho mais eficiente, quanto a este quesito.

Agora considerando o número de polígonos renderizados como sendo o fator limitante de desempenho, os testes realizados utilizando as mesmas estruturas poligonais refletiram que o processamento destes polígonos em WorldToolKit tem maior desempenho do que em Java3D, entretanto, ao aplicar os recursos de fala, tem-se as seguintes análises a partir da Figura 33 e Figura 34. Ambas demonstram que de acordo com o nível de detalhamento do objeto virtual é possível ter maior ou menor taxa de quadro por segundo.

O nível de detalhamento dos avatares humanóides foi percebido quando ouve uma diferença em relação às taxas encontradas na Origem (0,0,0), a Frente (0, 0, -10) e Atrás (0,0,10). As localizações descritas são por Barrileuax (2001) como forma de ilustração de uma coordenada virtual simulando um ambiente real.

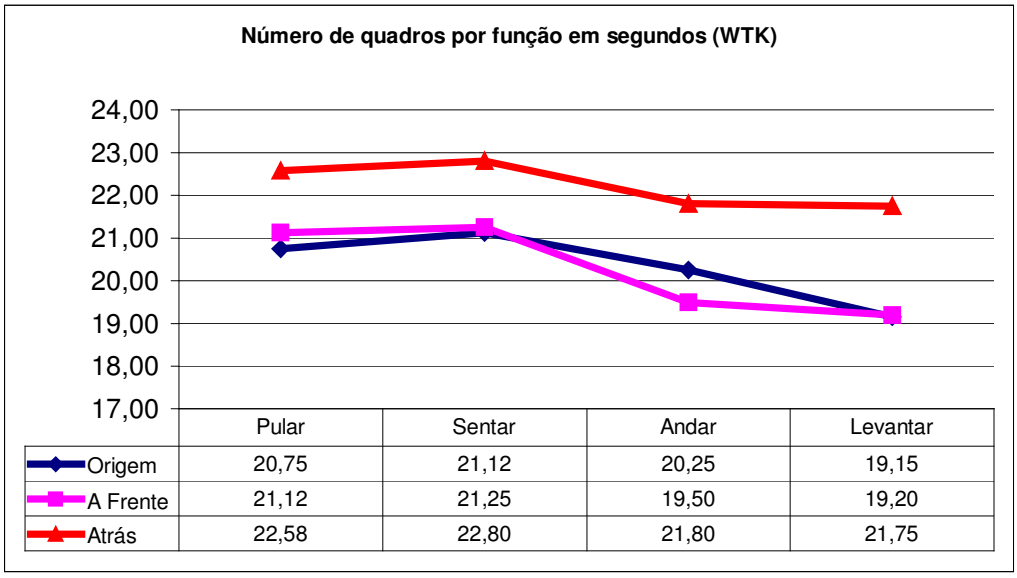
O uso de características dos polígonos e estruturas passíveis de modificação, tais como forma, cor iluminação, transparência, podem resultar em diferença de desempenho em Java3D e o que não ocorre com atenuação do processamento em WorldToolKit.

O WorldToolKit trata de forma diferente as primitivas gráficas 3D do Java3D, ele as trata no mesmo nível que o OpenGL, e portanto age diretamente na placa gráfica. O Java3D, mesmo acessando as bibliotecas do OpenGL ou do DirectX, ainda tem a interface de comunicação com a Máquina Virtual (Java Virtual Machine) e a chamada as bibliotecas de Native Graphic Call, como podem ser vista no arcabouço destacado na Figura 26.



	Pular	Sentar	Andar	Levantar
Origem	19,25	19,17	18,75	20,15
A frente	20,02	18,05	18,20	18,52
Atrás	21,29	20,52	19,95	21,45

Figura 33 – Taxa de Quadros por segundo por função em Java3D



	Pular	Sentar	Andar	Levantar
Origem	20,75	21,12	20,25	19,15
A Frente	21,12	21,25	19,50	19,20
Atrás	22,58	22,80	21,80	21,75

Figura 34 – Taxa de Quadros por segundo por função em WorldToolKit

É possível observar que o desempenho com a biblioteca gráfica WorldToolKit é ligeiramente maior do que o desempenho do Java3D, em torno de 6 % como também destaca Burdea e Coiffet (2001). O que é destoante é o fato de que quando o usuário utiliza o comando levantar o Java3D ganha processamento em aproximadamente de 5%, ficando assim mais rápido do que o WorldToolKit.

Isto é possível, pois os movimentos de sentar são simplórios, articulando poucos objetos, entretanto ao comando levantar, como articula mais objetos é possível especular que de acordo com o nível de detalhamento e o volume de objetos sendo movidos ou renderizados e a utilização da biblioteca de recursos de fala é possível ter uma queda mensurável em WorldToolKit , mas o desempenho é maior do que Java3D, trazendo uma melhor definição pela taxa de quadros por segundo.

A Figura 36 mostra o gráfico comparativo de processamento a medida que a interface de voz é utilizada.

A preocupação com a interface de fala é importante, devido ao fato do usuário não perder a imersão com o AV. Na maioria das implementações encontradas tais como a de Apaydin (2002) e de Meiguins (2003) utilizam o recurso de fala a todo o momento, criando uma sobrecarga no sistema.

Segundo Sun (1998), o mecanismo de reconhecimento e síntese de fala obedecem a uma máquina de estados e estes estados ou momentos segundo Apaydin (2001) realizam a troca de mensagens ocasionando a execução de um comando ou o acionamento de um evento.

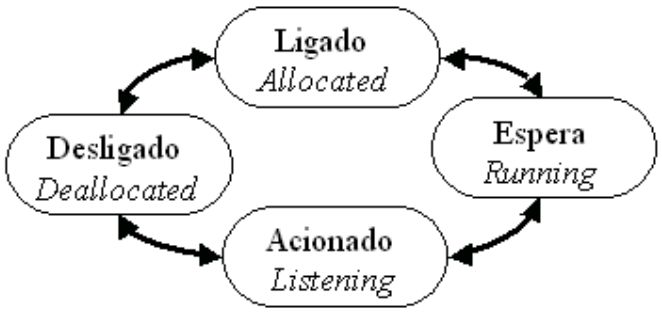


Figura 35 - Máquina de Estados da biblioteca de fala

Para que haja a compatibilidade entre as bibliotecas de reconhecimento de fala, observou-se os mecanismos para que fosse possível encontrar alguma semelhança na máquina estado e o que foi encontrado reflete a seguinte operação demonstrada na Figura 35. A observação pertinente é que esta figura representa um conjunto dos estados observados nas duas APIs.

A observação do comportamento do AV ao acionar o mecanismo de reconhecimento de fala é primordial para que o usuário não se sinta frustrado na inicialização do sistema e nem do decorrer do uso do mesmo.

A Figura 36, imprime o comportamento do AV de acordo com os estados identificados na Figura 35.

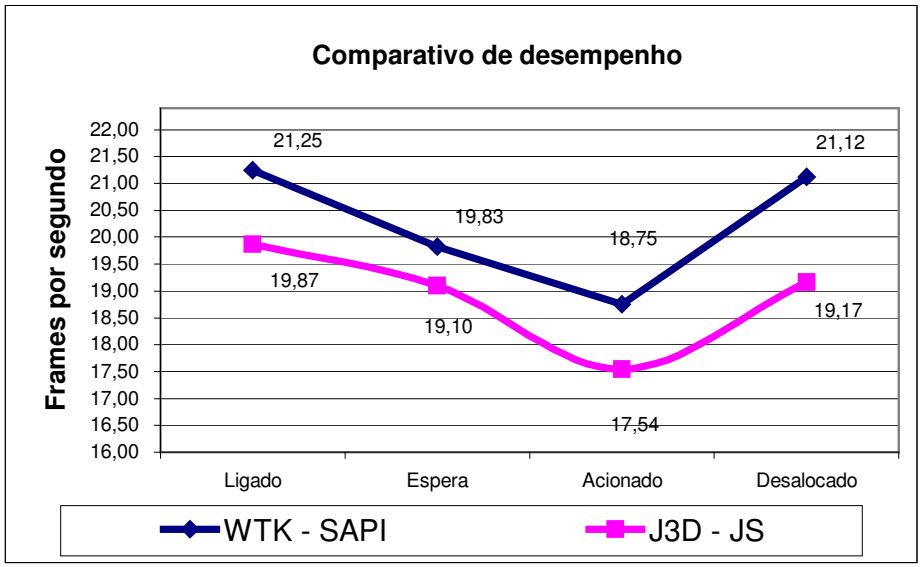


Figura 36 - Gráfico comparativo da perda de desempenho

É possível identificar que quando o sistema é acionado, ou seja, quando o usuário pronuncia alguma palavra que pertence às regras gramaticais definidas é possível garantir que utilizando a tecnologia WorldToolKit -SAPI (Sense8 e Microsoft) existe uma perda maior de

processamento do que se utilizar a tecnologia Java (Java3D – Java Speech), onde encontra-se a diminuição de 6 %, quando o mecanismo é ligado e de 12 % quando o mecanismo é acionado.

É possível obter um melhor desempenho do Java menor do que 1% quando o mecanismo de reconhecimento de fala é usado, mas quando ele é desalocado, tem-se uma queda no desempenho devido ao tempo de recuperação da memória por meio do *Garbage Collection*.

Segundo Sun (1998) e Microsoft (2000), o processo de alocação não é executado logo quando o sistema se inicia, visto que é necessária a alocação de recursos significativos como processador, espaço em disco e memória, para acesso exclusivo do sistema de áudio e seus componentes, tais como microfone, placa de som.

Para se melhorar o desempenho em Java é necessário à inclusão de um processo que execute em separado para dar vazão ao processamento dos recursos de vídeo.

O trecho de código mostrado na Figura 37 é usado para ganhar maior processamento dentro do AV no Protótipo 1, criado em Java.

```
Engine engine;
{
  engine = Central.createRecognizer();
  new Thread(new Runnable() {
  public void run() {
      try {
        engine.allocate();
      }
      catch (Exception e) {
       e.printStackTrace();
     }
   }
  }
 ).start();
 engine.waitEngineState(Engine.ALLOCATED);
 }
}
```

Figura 37 - Código fonte da divisão de processamento

De acordo com os trabalhos de Damasceno e Brega (2004) é necessário para que os eventos da interface de voz sejam acionados juntamente com os eventos do ambiente virtual é que estes estejam diretamente ligados a plataforma gráfica ou a biblioteca de programação gráfica utilizada.

No Protótipo 1, escrito em Java verificou-se que o ganho de processamento utilizando o pacote AWT ao invés da Swing é possível conseguir aproximadamente 18 % mais de processamento conforme pode ser vislumbrado na Figura 38, onde é possível encontrar singelas diferenças entre o processo de síntese de fala e de reconhecimento de fala.

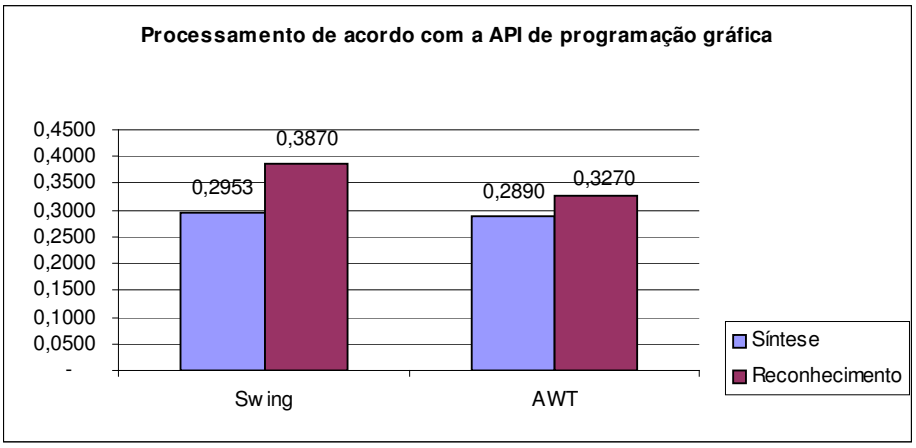


Figura 38 - Gráfico de processamento (milissegundos) de acordo com o pacote gráfico

Segundo SUN (1998) quando um evento é acionado ele fica alocado a uma fila associada a um pacote. No caso da síntese de fala a fila está associada apenas ao pacote Java Media Framework o qual aciona os eventos de som, entretanto no caso do reconhecimento de fala a fila está associada à plataforma gráfica AWT e por isso é necessário um sincronismo com a fila de eventos AWT. Para a implementação do sincronismo da fila AWT de eventos foi utilizado o seguinte trecho de programa visto na Figura 39.

```
EventQueue q = Toolkit.getDefaultToolkit().getSystemEventQueue();
```

Figura 39 - Trecho de programa

Os componentes do pacote Swing não são sobrecarregados pelos recursos GUI complexos da plataforma em que são utilizados, mas dependem da vinculação da AWT, do qual o Java3D se baseia para desenhar objetos na tela. Mesmo que cada componente AWT possua uma interação separada para o acesso a plataforma, o fato de que quando é utilizado componentes swing junto ao Java 3D existe o que pode-se denominar de perda por sobrecarga, ou seja, o pacote swing por ser uma classe derivada do pacote AWT, é processado primeiro e depois reprocessado para dar vazão ao pacote Swing.

No Protótipo 2, escrito em MVC++, não é possível ter as mesmas conclusões com a versão do WorldToolKit 7, que está disponível para estudo. Isto devido ao fato de que o processamento requerido para o reconhecimento de voz é usado apenas no ambiente Windows por meio de repetições de mensagens como visto anteriormente na Figura 21. O processo de abertura em modo console impede o processo de reconhecimento de fala, deixando apenas o processo de síntese de voz ser utilizado, pois este não necessita do sistema de repetições de mensagens.

7.3 Técnica de Programação

Tanto a linguagem Java quanto a linguagem C usam alguns artifícios que facilitam a codificação de um sistema, entretanto de acordo com a facilidade do artifício usado, maior é o uso de recursos de linguagem para poder identificar o que o usuário realmente está codificando.

Segundo Pressman (2002) as linguagens de programação são veículos de comunicação entre seres humanos e os computadores. De acordo com Kernighan e Pike (2000) é possível abusar deste processo de comunicação simplificando alguns comandos e procedimentos de acordo com a característica da linguagem.

Um exemplo claro é possível obter de Kruglinski (1997), que retata como o ambiente da linguagem C encontra as bibliotecas (LIB) e os arquivos de cabeçalho (*hiddens files)* que são necessários para a codificação de maneira mais rápida..

Na linguagem Java, segundo Furgeri (2002), o acesso às bibliotecas é dado de acordo com o pacote que ela pertence, assim como na linguagem C/C++ , existem diversas bibliotecas prontas, mas no caso da linguagem Java essas bibliotecas são denominadas de classes e são dispostas a partir do diretório de gerenciamento do Java SDK, de acordo com os ensinamentos de Deitel e Deitel (2003).

Pode-se concluir que na mesma proporção as linguagens tem suas características similares ao que tange a forma de acesso dos arquivos de cabeçalho e pacotes de classes.

Mas para atenuar a perda de desempenho do Java 3D é necessário a utilização das bibliotecas (pacotes ou *packages*) não mais na sua forma convencional, visto na Figura 40, mas na forma identificada completa, visto na Figura 41.

Esta forma de programação além de atenuar a perda de desempenho deixa o programa mais legível, segundo Sebesta (1996), assegurando o entendimento para futuras manutenções.

```
001  import java.sql.*;
002  import javax.swing.*;
003  import java.awt.*;
004  import java.awt.event.*;
005  import □til.□til.*;
```

Figura 40 – Forma convencional de programação Java 3d

```
001    import java.awt.GraphicsConfiguration;
002    import java.awt.event.ActionListener;
003    import java.awt.event.ActionEvent;
004    import java.awt.event.WindowAdapter;
005    import java.awt.MenuBar;
006    import javax.media.j3d.VirtualUniverse;
007    import javax.media.j3d.Local
```

Figura 41 - forma de programação identificada completa

A melhora de desempenho que pode ser observada e quantificada foi de em média 20% como denota a ilustração da Figura 42.

Além da utilização do recurso de programação de inclusão de pacotes identificados foi detectado que o uso do pacote de desenvolvimento swing é redundante para o Java 3D, mesmo contrariando os ensinamentos de Deitel e Deitel (2003), pois ambas (a classe canvas3D e o canvas do AWT) são especificações da classe Graphics, que acompanha o pacote AWT.

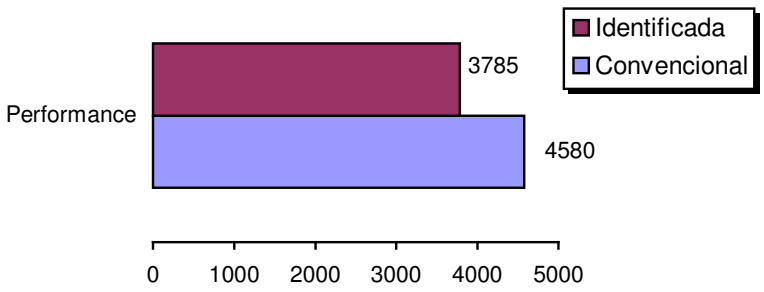


Figura 42 - gráfico de tempo gasto para processar o ambiente em milissegundos

7.3 Abordagem Adotada para o Reconhecimento de fala

De acordo com Shriver e Rosenfeld, (2003), uma interface de fala pode ser implementada utilizando três técnicas diferentes: a) utilizando uma Linguagem Natural Controlada; b) diálogo direto, ou seja, por perguntas e respostas e; c) por comando e controle.

A abordagem utilizada foi de implementar o registro das palavras no idioma em inglês e depois o idioma em português. Devido às diferenças de pronúncia e a flexão das palavras no idioma inglês, esta abordagem não obteve o sucesso esperado, entretanto o desempenho do ambiente virtual não se provou alterado.

A Tabela 10 demonstra a acurácia dos acertos das palavras nos mecanismos IBM ViaVoice e Microsoft, sendo estes controlados pelas bibliotecas API, Java Speech e Microsoft Speech.

Tabela 10 - Comparação entre os idiomas e as bibliotecas de programação em porcentagem de acertos

	Português		Inglês	
Mecanismo	**JSAPI**	**MS-SAPI**	**JSAPI**	**MS-SAPI**
IBM ViaVoice	83,0%	82,0%	81,0%	79,0%
Microsoft	Não	Não	79,0%	80,0%

Como não foi encontrada referência para o idioma em português para o mecanismo Microsoft sua análise passou diretamente para o idioma inglês.

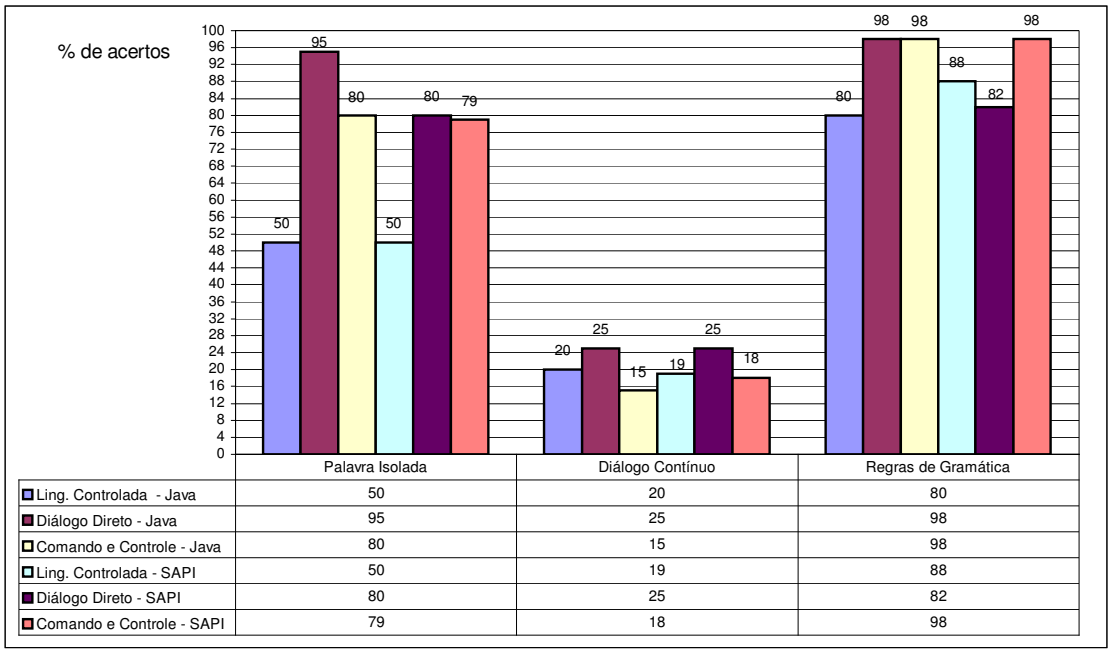
Um dado importante é que quando é utilizada a biblioteca Java Speech, não há eventos de erros ou o mecanismo trata apenas os erros sendo pronúncia, ou seja, ele verifica se o que o usuário pronunciou está na gramática definida.

Já a biblioteca MS-Speech API, consegue redirecionar o que foi pronunciado para um arquivo temporário que pode ser usado como verificador ou corretor na segunda vez em que o usuário pronuncia a frase.

Ao observar as técnicas de implementação de serviços de fala obteve-se as seguintes análises quanto a percentagem de acertos do reconhecimento nas duas plataformas como segue a Figura 43.

É possível identificar que para a abordagem de palavras isoladas a diferença se encontra na abordagem de comando e controle, onde se percebe que o MS-Speech é 1% menos eficiente que o Java Speech e em na abordagem para diálogos contínuos o MS-Speech é 3% mais eficiente do que o Java Speech.

E na abordagem de comando e controle para regras de gramática provou-se pelo estudo que não existe diferença entre a taxas de acerto, chegando a ponto de se encontrar até 98% de acurácia nas duas plataformas.



	Palavra Isolada	Diálogo Contínuo	Regras de Gramática
Ling. Controlada - Java	50	20	80
Diálogo Direto - Java	95	25	98
Comando e Controle - Java	80	15	98
Ling. Controlada - SAPI	50	19	88
Diálogo Direto - SAPI	80	25	82
Comando e Controle - SAPI	79	18	98

Figura 43 - Acurácia dos sistemas de reconhecimento de fala

Em termos do desempenho é possível identificar a partir da

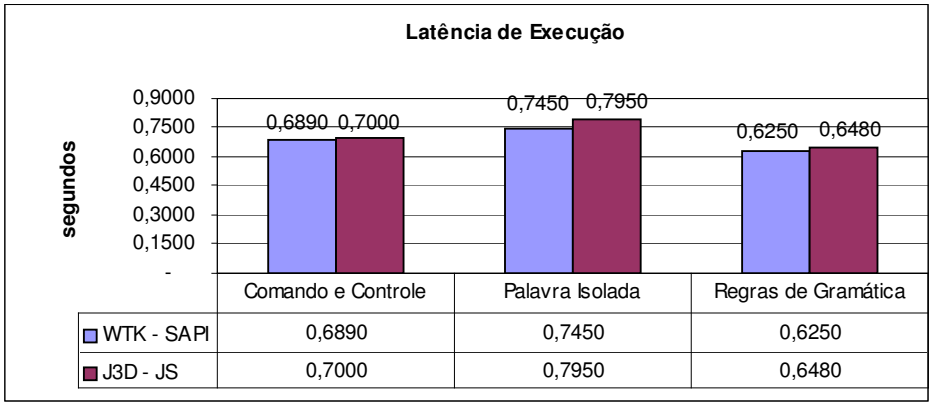


Figura 44 algumas outras reflexões úteis para análise entre os tipos de abordagens de reconhecimento de fala dentro de Ambientes Virtuais, e prova que o MS-Speech é 2 % tem menor desempenho e do que o Java Speech na abordagem de comando e controle e o Java Speech tem um desempenho maior quando utilizada uma abordagem de regras de gramática.

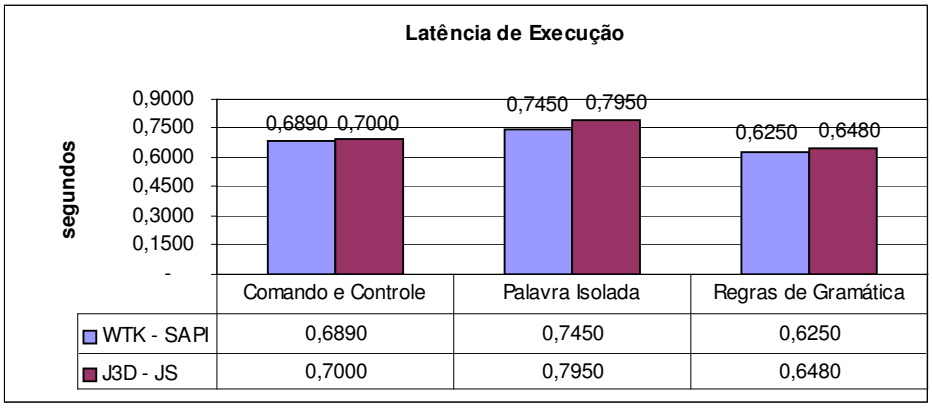


Figura 44 - Latência de Execução

7.4 Carregamento de Arquivos (*Loading*)

Conforme foi exposto no Capítulo 6 é possível notar que existe diferença de processamento de acordo com a plataforma e sistema operacional que a biblioteca gráfica é utilizada.

Os testes foram executados na plataforma de 32 bits Windows 2000, e os resultados são mostrados na Figura 46.

A metodologia adotada para este teste foi a de se desfragmentar as partições do HD, antes e depois da execução de cada iteração com o programa.

É possível observar que quando é utilizado formas geométricas nas duas bibliotecas Java3D e WorldToolKit para carregar via comandos executados por meio de regras de gramática, o resultado obtido é que ao realizar o carregamento de uma forma geométrica o WorldToolKit é 2 % mais rápido do que o Java 3D, mas, quando utilizando um arquivo no formato *waveform* é possível notar que o Java 3D é acima de 7 % mais eficiente do que o WorldToolKit, visto o que foi exposto no Capítulo 6.

Os arquivos de teste foram estabelecidos de acordo com seu formato (polígonos) e de acordo com seu tamanho em bytes.

Quanto à textura foi utilizado um arquivo em formato (jpg) de 12 Kbytes, que pode ser vista na Figura 45. As cores e luminosidades dos objetos forma mantidos iguais, ou seja, uma luz de cor branca direcional aos objetos e outra luz de mesma cor para o iluminar o ambiente.

Figura 45 - Arquivo de Textura

A estrutura que foi carregado para teste foi a estrutura da cabeça com comando de fala “Load Head” para o MS-Speech e “Carregar Cabeça”

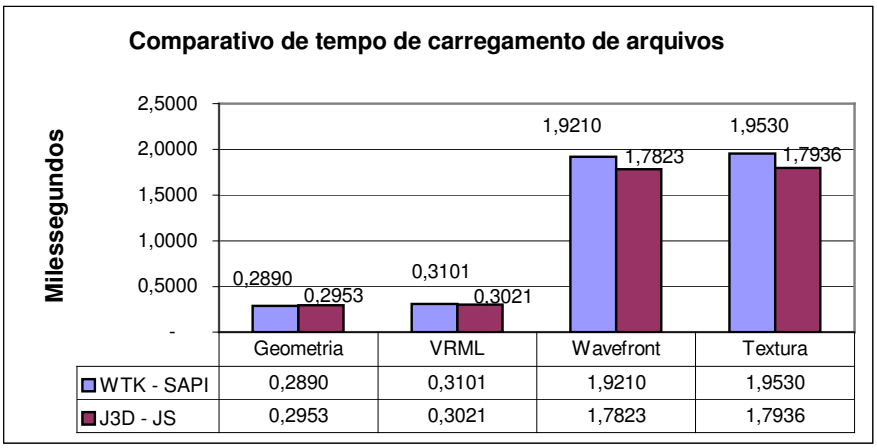


Figura 46 – Latência no carregamento de arquivos

Os arquivos de imagens utilizados nos testes são de diferentes formato, para o Java 3D, utilizou-se os arquivos wavefront (.obj) e para o WorldToolKit, foi utilizado os arquivos de definição neutra (.nff).

O formato de arquivo 3D wavefront (.obj) é um padrão de formato de arquivo para imagens tridimensionais, sua estrutura é textual (ASCII), ou seja, é possível a edição do arquivo em editores de texto comuns e definições de campos e de estruturas de imagens são hierárquicas, podendo ser visualizado uma ou mais imagens tridimensionais em cada arquivo. Basicamente o arquivo possui a seguinte estrutura: estrutura de definição de objetos, cor, tamanho e número de polígonos, para cada estrutura é possível vincular um arquivo de material ou textura, ou seja, para cada arquivo (.obj) pode existir um arquivo de material (.mtl). ou apenas um arquivo (.obj) incorporando o arquivo de material (.mtl).

Já o WorldToolKit, trabalha com um modelo de arquivo 3D diferente, mesmo sendo um formato de arquivo do tipo (ASCII), a estrutura que o compõe é uma representação de formas poligonais genéricas, ou seja, a união de vário polígonos para representar uma imagem.

Para cada característica destes polígonos existe uma representação em um arquivo binário (.bff) que define a cor, iluminação, distorções, material (textura) utilizado.

7.5 Divisão do processamento (*threads*)

Quando um sistema necessita de grandes recursos processamento e que se deseje que o usuário não perca a interação e o sentido de imersão com o sistema, é possível fazer a divisão do processamento, isto é possível por meio de *Threads.*

Segundo Deitel e Deitel (2003), *thread* é um fluxo de execução do programa que tem acesso ao processador e ser iniciado ou colocado em espera (*sleep*) de acordo com sua prioridade.

Um *thread* consiste em uma pilha que contempla os estados dos registradores do processador e entra na lista de execução do *scheduler* (escalonador de tarefas) do sistema operacional, o qual pode compartilhar todos os recursos do processador.

Cada *thread* em um processo opera independente de outra, ao menos que se deseje que um *thread* visualize outro e trabalhem juntos. Para tal procedimento é usado o conceito de sincronização, segundo Patterson e Hennessy (2000).

A Linguagem Java é a uma linguagem de programação de uso geral que disponibiliza o controle de fluxo de execução para o programador de aplicativos. Em C/C++ estas funcionalidades não estão disponíveis e para ser realizado o mesmo trabalho é necessário o uso de repetição de mensagens quando a programação é para Windows, portanto dependendo diretamente do sistema operacional e não da primitiva da linguagem como em Java.

Segundo Kruglinski (1997), em muitas plataformas de computadores, os programas C e C++ podem executar *multithreading* usando bibliotecas de código API específicas para o sistema.

Essas bibliotecas de programação são dependente da plataforma C, que está sendo realizado projeto, se o desenvolvedor optar pela plataforma MFC (*Microsoft Foundation Class*) ele irá utilizar as seguintes bibliotecas**: msmfc40.dll , msmfc51.dll ou msmfc60.dll** que dão suporte a programação com MFC.

E quando se deseja utilizar as bibliotecas C/C++ em tempo de execução, é necessária a seguinte biblioteca: LIBCMT.Lib e MSVCRT.Lib, com as quais poderá ser garantida a execução de um *thread*.

O gráfico da Figura 47 mostra as diferenças entre o processamento na linguagem Java em comparação com MVC++, o que neste exemplo provou-se que Java utilizando *multithreading* é 23% mais rápido do que um programa de reconhecimento de fala programado em MVC++, sem utilizar qualquer biblioteca tridimensional como Java 3D e WorldToolKit, e quando são utilizados estas biblioteca a diferença cai para 15%.

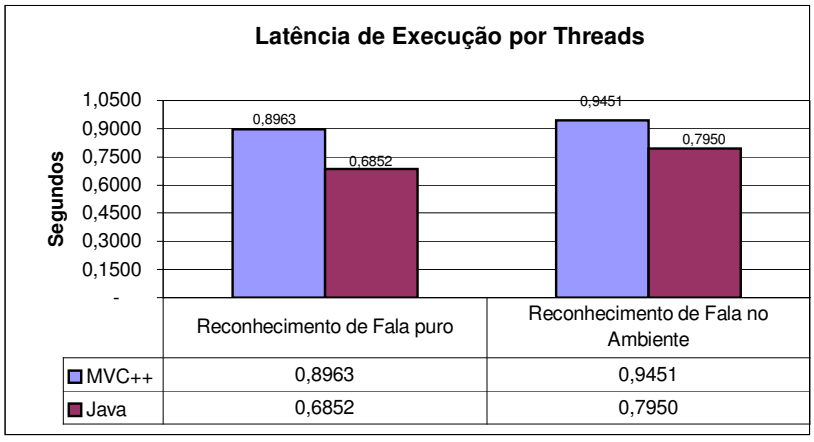


Figura 47 - Latência de Execução por *Threads*

Quando é usado no sistema um fluxo de execução com Síntese e Reconhecimento de Fala a latência ainda é maior em MVC++ do que em Java como comprova o gráfico da Figura 48.

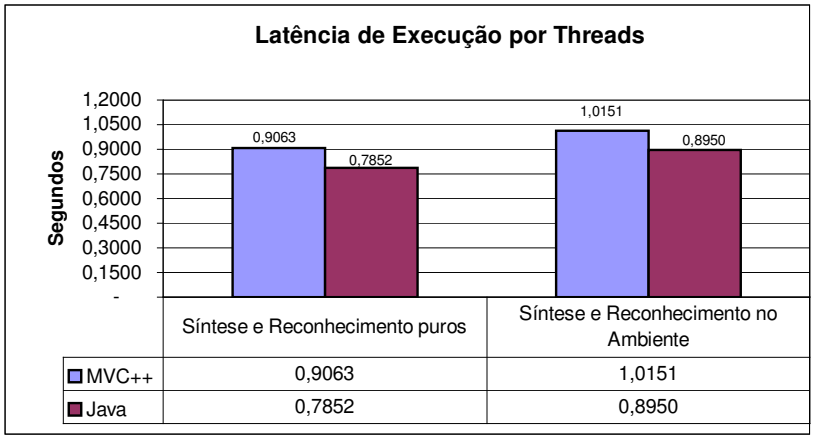


Figura 48 - Latência utilizando *threads* para reconhecimento e síntese de fala

Embora Java seja talvez a linguagem de programação portável, certas partes da linguagem são dependentes da plataforma. Em particular, há diferenças entre as plataformas Java implementadas em sistemas operacionais Windows e Solaris.

A plataforma Solaris, por ser não preemptiva, executa um *thread* de uma determinada prioridade até a sua conclusão ou até que um *thread* de prioridade mais alta fique pronto. Neste ponto, ocorre a preempção (isto é, o processador é alocado a *thread* de prioridade mais alta enquanto o *thread* que está sendo executado antes deve esperar).

Nas implementações de 32 bits de Java para Windows, por se tratar de um sistema operacional preemptivo, os *threads* recebem uma fração de tempo. Isso significa que cada *thread* recebe uma quantidade limitada de tempo para ser executado em um processador e, quando esse tempo expira, o *thread* é colocado em estado de espera enquanto os outros

threads de igual prioridade obtêm suas oportunidades de utilizar a sua fatia de tempo num esquema de rodízio. Então o *thread* original retorna a execução.

Portanto para a plataforma Windows pode-se definir a prioridade mais alta para um *thread* e esperar a execução, favorecendo assim as técnicas de programação de ambiente tridimensionais.

O *thread* compartilha um espaço de endereçamento de um único processo, otimizando o sistema e consumindo menos recursos, mas aí aparece também a grande desvantagem deste recurso, pois, por causa desta característica (serem executadas no mesmo espaço de endereçamento), os *threads* compartilham este espaço sem nenhuma proteção ou restrição o que permite que um *thread* possa alterar dados de um outro ou vice versa.

Para as aplicações produzidas em Java 3D, isso pode ocorrer uma atribuição de lixo entre os *threads* do mecanismo de reconhecimento e a geração do ambiente virtual. Já para o worldTooKit, como o *thread* de chamada ao mecanismo de geração do ambiente é encerrada e iniciada para cada chamada do evento de envio de mensagem do Windows.

7.6 O Compilador Java JIT – *Java Just In Time*

Termo JIT do compilador Java é usado para indicar que um processo é capaz de responder instantaneamente à demanda de recursos computacionais. A linguagem Java possui a partir da versão 1.3.1 este processo de compilação embutido no compilador JavaC.

Para cada instrução lida interpretada, ao invés de se criar um ByteCode, é criado um código nativo, ou seja, um código de máquina específico para o sistema operacional executor da Máquina Virtual, aumentando assim o desempenho das aplicações Java.

Entretanto o código nativo é somente do pacote de primitivas Java, as API não geram código nativo, ou ficam dependentes de outras bibliotecas de vinculo dinâmico (.dll) que estão ligadas a máquina virtual Java, como é o caso da Java 3D e do Java Speech.

O uso deste método de compilação incorporado ao compilador JavaC, provê em torno de 27% de desempenho a mais do que na compilação comum.

7.7 O Otimizador de Código MVC++

Durante o desenvolvimento do projeto, a ferramenta da Microsoft, MVC++, provou-se estável na geração de um código leve, segundo Kruglinski (1997), que se beneficia das bibliotecas de interface gráfica com o usuário (*graphical user interface* – GUI).

O compilador MVC++ possui um verificador de código que otimiza a execução do fluxo de mensagens entre processo por meio de um conjunto de bibliotecas associadas diretamente com o sistema operacional, a MFC – *Microsoft Foundation Classes*, que são estes conjuntos de bibliotecas para geração de interfaces GUI.

Como o desenvolvimento do ambiente virtual não foi possível utilizar uma biblioteca MFC, a alternativa foi fazer a chamada as bibliotecas (.h) do WorldToolKit que acionam o ambiente virtual e fazer com que elas sejam executadas sempre.

Por meio da ferramenta Spy++, que acompanha o pacote de desenvolvimento MVC++, foi possível detectar e determinar as áreas do código onde existe uma maior perde de desempenho, observar as bibliotecas que estão sendo realmente utilizadas bem como seu desempenho.

CONCLUSÕES

Para a criação de um AV não imersivo, em que o usuário se sinta entredito com o sistema é necessário à incorporação de recursos multímodos de interação e uma das formas mais acessíveis de interação não convencional é o uso da tecnologia de reconhecimento e síntese de fala.

Para o desenvolvimento de aplicações onde é necessário este recurso deve-se tomar algumas precauções na escolha da plataforma de desenvolvimento, pois mesmo escolhendo uma plataforma em detrimento de outra, pode-se observar que sempre existirão restrições de uso que poderão ser notadas pelo usuário, assim diminuindo a imersão do mesmo.

No âmbito de se escolher entre plataforma este trabalho tenta demonstrar algumas características das duas linguagens de desenvolvimento. A linguagem Java e a linguagem C/C++.

Pelo mesmo prisma, observa-se que a linguagem Java interage melhor com sua plataforma de desenvolvimento de ambiente tridimensionais (Java 3D) como pode ser visto nas análises do Capítulo 6.

Daí tem-se um paradoxo de uma linguagem pré-compilada pode ser mais rápida do que uma linguagem compilada. Isto devido ao fato do estilo de programação usado pelo desenvolvedor e pelos critérios e organização da linguagem na criação do AV.

A observação dada no item 6 do Capítulo 7 traz a reflexão de que o código Java por meio de seu compilador JIT, possui uma compilação e não um interpretação como é de conceitualização da linguagem, e isto auxiliado à definição de prioridade dos *threads*, faz com que o código gerado seja de maior desempenho para as funções de renderização e de reconhecimento de fala.

A definição das prioridades foi atribuída de modo que o usuário final não perceba uma mudança de comportamento do sistema, seja por um atraso, ou, seja por um adiantamento da renderização. O fluxo de processo escolhido foi o de reconhecimento de fala. A ordem das prioridades foi a seguinte: Em primeiro o reconhecimento de fala, em segundo a renderização do ambiente e em mesma prioridade a síntese de fala.

Quando se desenvolve em ambiente Java, que é orientado a objeto, não há a necessidade de preocupação com referências ou mensagens, nem mesmo com o lixo, pois, as propriedades do *Garbage Collection* fazem com que a máquina virtual faça todo o trabalho árduo.

Já quando está lidando com a programação diretamente com Windows por meio do MVC++ é necessário compreender como o sistema operacional trabalha simulando tecnologia de orientação a objetos por meio de trocas de mensagens entre os procedimento, pode-se notar o estilo de programação na Figura 21. Além do estilo de programação é necessário também retratar que Java possui uma forma diferente no carregamento de arquivos (*loading*), como visto na Figura 26 e Figura 30 e comprovado pelas análises do gráfico da Figura 46

Pode-se observar que o processamento do Ambiente Virtual em Java 3D é mais rápido do que o processamento em WorldToolKit como comprova a Figura 33 e a Figura 34, onde é analisado o tempo de renderização ou quadros por segundo (FPS), onde também é notado que Java perde menos recursos ao alocar um objeto virtual, no caso teste um avatar humanóide, em diferentes posições do espaço virtual.

Quanto a questão da divisão do fluxo de execução mais uma vez a linguagem Java imprime uma vantagem sobre a linguagem C/C++, pois obtém cerca de 23 % mais de agilidade no processamento, como é provado no gráfico da Figura 47, visto que os atributos do mundo virtual estão na mesma plataforma (Java 3D) juntamente com a biblioteca de controle do mecanismo (Java Speech).

É possível obter maior processamento quando o fluxo é divido entre a geração do ambiente virtual e o mecanismo de reconhecimento, isto tanto em Java quanto em MVC++, mas o ganho não é tão compensador, esta vantagem pode ser vista na Figura 48.

O fato que mais denota a relevância da pesquisa é de que não é necessário adquirir um mecanismo de geração de AVs, como o WorldToolKit da Sense 8 para gerar um Ambiente Tridimensional pois os recursos do Java 3D podem ser tão compensadores quanto. E quando for necessária a inclusão de recursos de reconhecimento e síntese de fala a perda de desempenho de Java é mais constante, o que pode significar que um Ambiente Virtual com reconhecimento de fala desenvolvido em Java com a técnica de programação correta tende a ter o baixa latência, isto comparado ao mesmo ambiente desenvolvido em MVC++ com WorldToolKit.

TRABALHOS FUTUROS

Para promover um avanço na área de Realidade Virtual com o uso de interfaces de fala, os trabalhos futuros serão o de comparação dos métodos descritos nesta dissertação com outras bibliotecas gráficas como o OpenGL e o VRJuggler que estão ligados as plataformas C/C++ e Java.

Além destas espera-se encontrar por meio da biblioteca 3D, GlScene para a linguagem Delphi, uma maior facilidade no desenvolvimento do ambiente virtual e bem como o desenvolvimento dos recursos de fala.

O uso de outros mecanismos para compor os serviços de fala poderá ser outra alternativa, desde que possam reconhecer o idioma Português do Brasil.

A partir destas conclusões sobre as bibliotecas de reconhecimento e síntese de fala aplicada a ambientes virtuais será possível montar um delineamento para os próximos trabalhos que utilizarão desta forma de interação com o usuário.

REFERÊNCIAS BIBLIOGRÁFICAS

AMUNDSEN, M. C. **MAPI, SAPI, and TAPI Developer's Guide**, disponível em: http://www.ssuet.edu.pk/taimoor/books/0-672-30928-9/index.htm, acessado em jul 04

APAYDIN, O. **Networked Humanoid Animation Driven By Human Voice Using Extensible 3D (X3D), H-ANIN and Java Speech Open Standards**. Naval Postgraduate School, Monterey, California; 2002.

BADLER, N. I., ***Virtual Humans for Animation, Ergonomics, and Simulation***, IEEE Workshop on Non-Rigid and Articulated Motion, Porto Rico, 1997.

BARRILLEAUX, J. ***3D User Interfaces with Java 3D***, Manning Publications Co. Greenwitch: 2001.

BORGES, L.A., ***Sistema de adaptação ao locutor utilizando autovozes***, Dissertação de mestrado em Engenharia pela USP, São Paulo, 2001.

BREGA, J. R. F.; SEMENTILLE, A. RODELLO, I. A., MELO, W. C. C. de. ***Uma Biblioteca em Java 3D para Movimentação de Agentes e Avatares Humanóides em Ambientes Virtuais, 5***º.SBC Symposium on Virtual Reality, Fortaleza – CE, Outubro, 2002.

BURDEA, G.; COIFFET, P. ***Virtual Reality Technology***. New York: John Wiley & Sons, 1994

_____, G.; BOIAN, R.F. ***WorldToolKit Vs. Java 3D : A Performance Comparison.*** CENTER FOR ADVANCED INFORMATION PROCESSING, Technical Report CAIP TR-259, The State University of New Jersey, New Jersey: 2001.

CASTRO, A. A. M. de. e PRADO P. P.L. do ***Algoritmos Para Reconhecimento de Padrões***, Revista Ciências. Exatas, Taubaté, v. 5-8, p. 129-145, São Paulo: 2002

CHOMSKY, N. ***Linguagem e pensamento***. 2a ed. Petrópolis: Vozes, 1971.

CHIOVATO, A. G. et al. ***Sistema de Reconhecimento de Fala com Ruído para Automóveis***, INATEL – Instituto Nacional de Telecomunicações, 2001. Acesso de http://www.inatel.br em 21/06/2003.

ÇAPIN, K. T. et al. ***Avatars in Networked Virtual environments***, John Wiley & Sons, Ltd. New York. 1999.

DAMASCENO, E.F. ***Implementação de Serviços de Voz em Ambientes Virtuais.*** Conferencia Nacional do CEFET, Teresina-PI, 2004

_____., BREGA, J. R. F. ***Implementação de Serviços em Java.*** Simpósio Brasileiro de Jogos de Computadores, Curitiba-PR, 2004

DEITEL H.M., DEITEL P.J. ***Java Como Programar.*** Bookman, 4ª Ed. Rio de Janeiro:2003.

DEVILLIERS, E.M. ***Implementing Voice Recognition And Natural Language Processing*** In The NPSNET Networked Virtual Environment. Naval Postgraduate school, California, 1996.

DUTOIT, T. ***A Short Introduction to Text-to-Speech Synthesis*** <http://tcts.fpms.ac.be/synthesis/introtts.html > 1999., acesso em 10 Out. 2003.

FAGUNDES, R. D. R. ***Reconhecimento de Voz, Linguagem Contínua, usando Modelos de Markov***. Tese de Mestrado, Escola Politécnica da USP, 1993.

FERREIRA, A.B.H. ***Novo Dicionário Aurélio - Século XXI***, Ed. Nova Fronteira, 2001.

FERREIRA, L. ***O Espaço Digital Imersivo*** . Anais da IX a COMPÓS (IX encontro da Associação Nacional de Programas de Pós-Graduação em Comunicação) 1999. disponível em: http://www.eco.ufrj.br/lucianaferreira acesso em 11 Nov. 2003

FRANCO H., NEUMEYER L., KIM Y. ,RONEN O. ***Automatic pronunciation scoring for language instruction***. in Proceeding Conference on Acoustic Speech and Signal Processing. Munique, 1997 .

FURUI S. **Digital Speech Processing, Synthesis and Recognition**. Marcel Dekker, Inc., 1989.

FURGERI S. ***Java 2 – Ensino Didático***. Ed. Érica, São Paulo: 2002

FURNESS, T. A. e BARFIELD, W, "**Introduction to Virtual Enviornment and Advanced Interface Design**, Prentice-Hall 1995.

_____ **Speech Recognition - Past, Present and Future**. NTT Review: 1995.

GOOSE S., SUDARSKY S., ZHANG X., NAVAB **N., *Speech-Enabled Augmented Reality Supporting Mobile Industrial Maintenance,* in PERVASIVE** computing, New York 2003.

GUILHOTO,P. J. S. ***Reconhecimento de Voz***. Universidade de Coimbra, Portugal 2002.

HART, W. **Win32 System Programming**, New Jersey, Prentice-Hall, 2001.

HAZEN T. J. and GLASS, J. R. – "***A Comparison of Novel Techniques for Instantaneous Speaker Adaptation***". Spoken Language Systems Group. Laboratory for Computer Science MIT, Cambridge. Disponível em www.sls.lcs.mit.edu/sls/publications/1997/eurospeech97-hazen.pdf. Acesso em 10 Jul. 2003.

HOSON, John-Paul, ***Automatic Speech Recognition at CSLU***, < www.cslu.ogi.edu/asr/> acesso em 11 Nov. 2003

HUGO, M. **Uma Interface de Reconhecimento de Voz para o Sistema de Gerenciamento de Central de Informação de Fretes**. Dissertação de Mestrado UFSC, 1995.

IBM Speech Recognition Group. **A Real-time Isolated-Word Speech Recognition System for Dictation Transcription**. IEEE International Conference on Acoustics, Speech and Signal Processing, 1985.

____, The IBM **Via Voice Home Page** http://alphaworks.ibm.com/speech/ , 1999. acesso em 15 Jul. 2003

____, **IBM Speech for JAVA** em http://alphaworks.ibm.com/tech/speech/ ,2001. acesso em 15 Jul. 2003

JURAFSKY, D., MARTIN, J. ***Speech and Language Processing***. New Jersey, Prentice-Hall, 2000.

KERNIGHAN B. e PIKE R., ***The Pratice of Programming***, Addison-Wesley, New York, 2000

KRUGLINSKI, D. J.: ***Inside Visual C++***, 4a Ed, Microsoft Press , 1997.

LÉVY, Pierre. "***O que é o Virtual*?"** São Paulo: Editora 34, 1996.

MARTINS, J. A. Avaliação de Diferentes Técnicas Para Reconhecimento De Fala, Tese de Doutorado,UNICAMP,1997

MEIGUINS, B.S Et. Ali. **Interação em Ambiente Tridimensionais Utilizando Comandos de Voz**, in Proceedings of Symposium on Virtual Reality, Ribeirão Preto, SP 2003.

MICROSOFT, ***Microsoft Speech API Developer´s Guide***, disponível em www.microsoft.com/speech/, 2000, acesso em 15.set. 2003.

_____, ***Microsoft Speech API 4.0***. Online documentation, disponível em: http://www.microsoft.com/speech/old/, 1998, acesso em 15 set. 2003

MOSSER, V. ***Introduction to Speech Synthesis***. Disponível em <www.clark.net/pub/mhsmedia/physics/intro.html > 2000. Acesso em 15 Set. 2003

NOMA, T.; BADLER, N. I., ***A Virtual Human Presente***r, Proc. IJCAI '97 Workshop on Animated Interface Agents: Making then Intelligent, 45-51, 1997.

NGUYEN,T.H. **Building an Interactive 3D Online Storefront.** Centre for Animation and Interactive Media School of Creative Media Faculty of Art, Design and Communication. RMIT University, Melbourne Australia (2003)

OLIVEIRA J.C. et al **VIRTUAL THEATER for Industrial Training: A Collaborative Virtual Environment,** Canadá**,** 2000, disponível em www.mcrlab.uottawa.ca/papers/cscc2000-Joliveira.pdf

PAIS, C. T., **Introdução À Fonologia**, Ed. Global, São Paulo,1981.

PATTERSON, D.A. e HENNESSY, J.L. ***Organização e projeto de computadores***, 2ª ed. Editora LTC, Rio de Janeiro, 2000.

PESSOA, B. D. S., ARAÚJO, M. S. C. F. ***Criação de mundos tridimensionais na Internet: Uma introdução operacional a VRM***L. Escola de Tecnologia da Informação ETI'99. 1999. acesso em 16 SET. 2003. Disponível: http://www.di.ufpe.br/~pet/CursosdoETI/VRML/apostila.zip,

PIZZOLATO,E. B. Reconhecimento de Fala, Conceitos, Fundamentos e Técnicas, UFSCAR, 2001

_____ ; Rezende M.N. ***Issues to Consider when Adopting Commercial Speech Interface in Virtual Worlds***, in Proceedings of Symposium on Virtual Reality, Ribeirão Preto, SP 2003.

PETRY, A. ***Revox Voice Recognition System Applied To Industry Control***, Universidade de Caxias do Sul – Centro de Tecnologia e Ciências Exatas – Departamento de Ciência da Computação– Caxias do Sul, 2000.

PRESSMAN, R.S. **Software Engineering: A Practitioner´s Approach,** 5ª ed. McGraw-Hill, New York, 2002

RABINER,L.R. ***A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition***. Proceeding of the IEEE, Vol. 77, nº 2, February 1989, pp 257-286.

RODRIGUES, J.F. ***Estudo e Desenvolvimento de Aplicações Java com Reconhecimento e Síntese de Voz***. Relatórios Técnicos do ICMC. São Carlos, 2001.

ROE, D.B. e WILPON J. G., "***Voice communication between humans and machines***". National Academy of Sciences, 1994.

ROSENFELD R., ***Universal Human-Machine Speech Interface: A White paper***, Carnegie Mellon Universit, 2001

RONALD A. C., et al. ***Survey of the State of the art in Human Language Technology***. Disponível em http://cslu.cse.ogi.edu/HLTsurvey/HLTsurvey.html 2000, Acesso em 16 Set. 2003

RUSSELL,S. **Artificial intelligence**, 3th. Edition, Elsevier, New York.2004.

SEBESTA, R. ***Conceitos de Linguagens de Programação***, 3ª ed., Rio de Janeiro, BookMan 2002

SCHILDT, H. ***C Completo e Total***. 3ª ed. Makron Books, São Paulo, 1997.

SIMÕES, F.O. ***Implementação de um sistema de conversão Texto-Fala para o português do Brasil***. Dissertação de mestrado em Engenharia Elétrica, Campinas, 1999.

SINGHAL, Sandeep e ZYDA, Michael. ***Networked Virtual Environments : Design and Implementation***, (Siggraph Series), ACM Press, New York, 1999.

SOWIZRAL, H.; RUSHFORTH, K.; DEERING, M. "***The Java 3D API Specification***". Addison Wesley 1998

SHOWCASES, ***Virtual ShowCases: IST – Information Socity Technologies*** (IST-2000-28610) disponível em http://www.ist.edu/showcases.htm, acessado em 10 fev 2005, :2000.

SPENCER, A. P., ***Anatomia Humana Básica***, 2º ed. Editora Manole, São Paulo, 1991.

SUN. Java™ Speech API Programmer's GuideVersion 1.0 . 1998, acessado em 12 Jul. 2003

_____, ***The Java Speech Markup Language Specification*** http://java.sun.com\products\java-media\speech, 1997. acessado em 13 Jul. 2003

THALMANN, N. M., ***Digital Actors for Interactive Television***, Proc. IEEE, August 1995.

THALMANN, D., ***Introduction to Virtual Environments***, Computer Graphics Lab Swiss Federal Institute of Technology, disponível em : http://vrlab.epfl.ch/~thalmann/VRcourse/intro.pdf., 2001, acessado em 01/06/2003

TATHAM, M. ***Speech Recognition. Disponível*** em: <http://www.essex.ac.uk/ speech/teaching/erasmus/recognit.html > Acesso em 15 Jul. 2003.

TREVINO M., ***Speech recognition technology finds a voice*** ,CMP Media LLC, a United Business Media company , 2001. disponível em <http://www.cmp.com/ > Acesso em 01 Nov. 2003.

VENNERS, Bill. ***Inside the Java Virtual Machine*** disponível em: http://www.artima.com/insidejvm/ed2/ch05JavaVirtualMachine1.html 2001, acesso em 01/02/2005

VIEIRA, J. et al. ***Localização de Fontes Sonoras com Tolerância a Ambientes Reverberantes***, Anais do III Festival Nacional de Robótica – Lisboa-Pt.,2003.

VIEIRA, R. e LIMA V. L. S. de. ***Lingüística computacional: princípios e aplicações***, Centro de Ciências Exatas e Tecnológicas -UNISINOS. São Leopoldo - RS, 2001

YNOGUTI, C. A. ***Reconhecimento de fala contínua usando modelos ocultos de Markov***. Campinas, 1999.

ZELTER D. & JOHNSON M. B. ***Interacting with Virtual Environments***, Ed. John Wiley & Sons, New York, 1994

ZOTOVICI, A., "***Java 3D Aplicado a Multimídia e Hipermídia***" Online: disponível na Internet via http://www.pcs.usp.br/~zotovici/hmmono.htm em 30/08/2002

ZDNET ***Avaliação dos sistemas de reconhecimento de voz,*** disponível em http://www.zdnet.pt/pcmagazine/analises/software/0002/voz.shtml, 2001, acesso em 01 Nov. 2003.