Implementação de Serviços de Voz em Ambientes Virtuais

Eduardo Filgueiras Damasceno[1], Tatiane Valau Pereira[2], José Remo Ferreira Brega[3]

[1]UFMS - Universidade Federal de Mato Grosso do Sul
Campu Universitário de Dourados - MS

[2]Programa de Pós-Graduação em Ciência da Computação
PPG-CC- Centro Universitário Eurípides de Marília - UNIVEM
[1]eduarododamasceno@uol.com.br, [2]tatitvp@bol.com.br, [3]remo@fundanet.br

Resumo: Esse artigo apresenta o uso de uma técnica de implementação de um ambiente virtual usando serviços de voz em Java com as bibliotecas "Java Programming 3D" e "Java Speech".
Palavras chaves: Ambientes virtuais, Reconhecimento de Voz, JAVA.

Implementing Voice Services in Virtual Environments

Abstract: This paper presents the use of one implementation technique of a virtual environment using voice services in Java with the libraries "Java Programming 3D" and "Java Speech".
Keywords: Virtual Environment, Java Programming, Speech Recognition.

(Received October 3, 2004 / Accepted February 11, 2005)

1. Introdução

Uma das maiores vantagens da tecnologia de Realidade Virtual (RV) é a visualização de informações complexas de forma simples para facilitar a compreensão das mesmas pelo usuário. Entretanto, mesmo com acesso facilitado ao conjunto de informações o usuário poderá sentir uma ansiedade ou frustração no uso das interfaces que se utilizam de recursos de imagens tridimensionais como destaca [12].

Segundo [1] a nova geração de interfaces homem-computador será 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 dois motivos principais: primeiro a facilidade no aprendizado do ambiente através do uso de voz conforme [18] e segundo pela liberdade que o usuário pode ter, conforme [17] 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, permite que os olhos ou mãos estejam livres para a realização de outras tarefas.

Com a utilização da tecnologia de serviços de voz favorecendo a usabilidade do sistema, os recursos alocados para o processamento e renderização do ambiente virtual acabam se atenuando em função da alocação do mecanismo de reconhecimento e síntese de voz. Daí parte a premissa deste trabalho que propõe o estudo das técnicas de programação e de desenvolvimento de ambientes virtuais que utilizem o serviços de voz sem perder performance e sem deixar que o usuário saia da ilusão criada pelo ambiente devido a delays de processamento.

2. Definição do Problema

O ser humano percebe o mundo através de seu sistema sensorial o qual une as informações audiovisuais, táteis e olfativas, onde os sentidos predominantes são os três primeiros que a tecnologia de RV já empregou em diversos equipamentos, mas a combinação de dois sentidos como o visual e o auditivo criam uma imersabilidade ao ambiente.

Olhos e cérebro trabalham juntos para perceber e interpretar as informações do mundo virtual fazendo com que o usuário sinta-se entretido com o ambiente, ou seja, sinta-se imerso no mesmo. Mas quando uma delas tem um atraso no processamento da imagem ou se adianta uma imagem do som, ou vice-versa, o cérebro percebe a disfunção e acorda o usuário tirando-o da imersabilidade e o faz atentar para o problema de sincronismo entre o visual e o audível.

Demonstrar a técnica de implementação do serviços de voz em Java dentro de um ambiente virtual é a proposta deste trabalho. Neste exposto destaca-se a criação do ambiente Java com as bibliotecas Java 3D e Java Speech, ambas em decorrentes implementações e atualizações de seus criadores, e a utilização de um mecanismo de reconhecimento da IBM, denominado de ViaVoice.

3. Trabalhos anteriores

Trabalhos anteriores destacam-se pela aplicação do reconhecimento de voz e síntese utilizando sua forma básica sem se preocupar com a interatividade e imersão dentro do ambiente, como é o caso das aplicações descritas por [7] e por [11]. Notavelmente a pesquisa de [10] decifra alguns problemas dentro do ambiente virtual associado a um sistema de reconhecimento e síntese de voz.

Outros trabalhos descrevem apenas os erros de rejeição ou de incompreensão dos comandos de voz como é o caso dos trabalhos de [9] e de [13].

4. Noções gerais sobre os Serviços de Voz

[2] elucida alguns termos que são necessários para a compreensão de como o serviço de voz funciona, tais como:

a) Fonema: sendo a menor unidade de som que as palavras são compostas;

b) Modelo acústico: que é um modelo de como os sons das palavras deveriam se representar;

c) Expressão: qualquer seqüência de voz entre dois períodos de silêncio;

d) Pronúncia: que é a forma de se falar as palavras (incluindo o sotaque e o regionalismo das mesmas);

e) Gramática: são as regras de reconhecimento, ou seja, o conjunto de palavras e conjunções válidas para a ativação de uma ação no sistema;

f) Treinamento: é o processo pelo qual o mecanismo de reconhecimento passa para identificar o sotaque e a pronuncia dos vocábulos do usuário;

g) Precisão: É uma variável que estabelece se o que foi pronunciado pode ser representado em sua equivalência pela gramática, ou seja, é a variável que representa a acurácia do sistema.

Segundo [6], freqüentemente a expressão “reconhecimento de voz” é utilizada com vários sentidos, que na verdade, referem-se a tecnologias distintas o que também é afirmado por [8]. O processamento da voz pode ser aplicado em quatro áreas principais: a) comando e controle por voz; b) reconhecimento de fala natural; c) síntese de fala; e d) autenticação de voz;

Segundo [17], um aspecto importante dentro do reconhecimento de voz é a escolha do procedimento de reconhecimento de fala por meio de modelos e para isso há uma especificação dos tipos de fala como os modelos de fala contínua e os de fala de palavras isoladas.

Neste trabalho optou-se em utilizar um mecanismo de reconhecimento de voz que facilitasse a implementação do serviço no ambiente virtual, e de acordo com os critérios descritos por [11], a opção de escolha do IBM ViaVoice foi a que mais se provou aplicável, por ser um sistema estável e possuir um conjunto de bibliotecas de programação para Java e C++.

Existem limitações em todas as bibliotecas de mecanismos de reconhecimentos e síntese de voz, na descrição de [16] tais limitações podem ser classificas em dois grandes grupos: os de limitações causadas por erros de pronúncia; e das limitações causadas por erros de áudio.

A limitação da biblioteca foi observada pelas duas linguagens, sendo que prevalece o treinamento do usuário como fator preponderante para uma utilização correta do ambiente atenuando os erros.

O sistema proposto é base para a avaliação nas duas linguagens (Java e C++), portanto deve representar a possibilidade da congruência das duas linguagens, ou seja, a implementação do serviço de voz deve ser com os mesmos comandos e a mesma gramática em língua portuguesa como está descrita na Figura 1.

```
grammar Bibmov;
public <ordem> = avatar <numero> <acao> | avatar
                 <numero> <direcao>;
public <numero> = um | dois | tres | quatro | cinco | seis |
                  sete | oito | nove | dez {numero};
public <acao> = sentar | levantar | (posição | onde) | (andar
                <numero> passos a <direcao> | caminhar)
                | parar | acenar | sim | nao | finalizar | pular
                | voltar  <ordem>;
public  <direcao>  =  direita  |  esquerda  |  frente  |  tras
                 <ordem>;
```

Figura 1: Gramática proposta para o sistema uniVoice

A aplicação apresentada baseia-se nos trabalhos de [2], onde foram propostas bibliotecas de movimentação de agentes e avatares humanóides, para aplicações de Realidade Virtual. A incorporação da biblioteca de movimentos veio favorecer o desenvolvimento da aplicação para a comparação dos recursos de voz ao processamento da plataforma gráfica desenvolvida em Java 3D conforme pode ser observado na Figura 2.

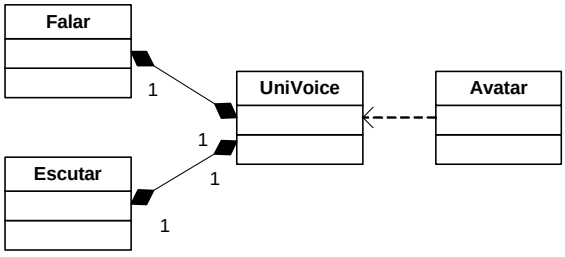

Figura 2 - Modelagem do sistema uniVoice

A Figura 2 tenta elucidar através de UML como é projetado a incorporação do serviço e voz dentro da aplicação e como o avatar compreende o que é pronunciado.

É notório observar que a classe <UniVoice> tem uma incorporação da classe <avatar>, ou seja, um uso da biblioteca de movimentos e possui uma extensão para duas outras classe <Falar> e <Escutar> onde podem ser instanciadas independentemente favorecendo assim o multiprocessamento, conforme [5].

5. A utilização da biblioteca de movimentos

De acordo com os trabalhos de [2], onde foram propostas bibliotecas de movimentação de agentes e avatares humanóides, foram definidas as seguintes movimentações: andar, parar, pular, sentar, levantar, as quais são voltadas para as noções básicas de direcionamento como direita, esquerda, acima e abaixo.

Nesta fase atual do trabalho está sendo utilizado o Java 3D que é uma API (*Application Programming Interface*), desenvolvida pela Sun Microsystems para renderizar gráficos 3D interativos usando Java. Pretende-se até o final deste trabalho uma comparação com a mesma biblioteca de movimentos utilizando WorldToolKit acessado pela linguagem C++.

Por obter-se maior realismo para o movimento do avatar foi utilizado o conceito de graus de liberdade (*Degree Of Freedom* -DOF), e representada cada parte do corpo, sendo integrada uma a uma, para se compor os movimentos humanóides.

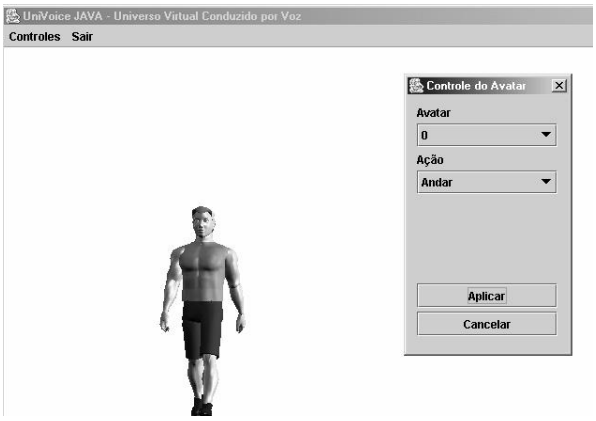

Figura 3 - Ambiente uniVoice

6. O mecanismo de reconhecimento

Neste trabalho optou-se em primeiro momento a utilização da linguagem Java com suas extensões para Java 3D™ e a Java Speech.

Para o procedimento de reconhecimento de voz foi escolhido o processo de fala de palavras isoladas aplicadas a uma pequena gramática de teste como mostra a Figura 1.

Segundo [13], a API Java Speech é uma interface de software capaz de manipular diversos mecanismos de reconhecimento e também de síntese de voz de fabricantes diferentes, isto por sua característica independente de plataforma que é herdada da linguagem Java.

O processo de reconhecimento de voz é diferente da compreensão da fala, segundo [11], a compreensão da fala está além do reconhecimento de voz no qual existe a tradução do sinal de fala para um texto, enquanto na compreensão da fala é gerada uma ação para o que foi reconhecido.

7. Detalhes da Implementação

Neste trabalho pretende-se observar o comportamento da biblioteca de reconhecimento e síntese de voz Java Speech junto com a biblioteca de geração de ambientes tridimensionais Java 3D para detectar e sanar o fator de gargalo de processamento, ou seja, observar seu comportamento em comparação com outras bibliotecas de desenvolvimento de ambientes tridimensionais e de reconhecimento de voz.

De acordo com [15], uma interface de voz 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.

Ao observar as duas últimas técnicas de implementação de serviços de voz obteve-se as análises da Figura 4.

A Figura 4 mostra que é possível ter cerca de 98% de acurácia quando se utiliza a técnica de reconhecimento de fala através do diálogo direto, um exemplo deste diálogo direto é quando o sistema realiza perguntas para o usuário e este responde, e deste modo as ações são executadas passo a passo.

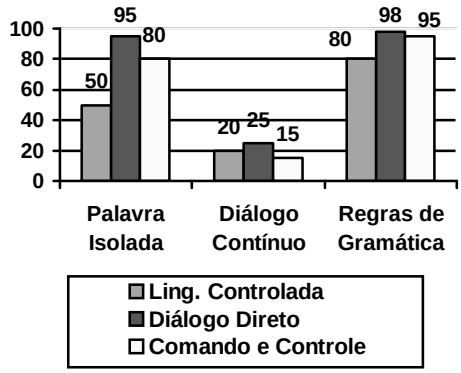

Figura 4 - Gráfico de Acurácia em (%)
Obs: Tempo médio das respostas calculado em função do clock da máquina virtual.

Porém, a figura também denota que quando existe uma gramática concisa é possível se chegar a 95% de acurácia com a tecnologia de comando e controle.

Mas quando se deseja uma aplicação que tenha um tempo de resposta que não deixe o usuário sair de sua imersão virtual é necessário observar a figura 5, onde é possível identifica que quando é utilizada a tecnologia de comando e controle para uma abordagem de diálogo contínuo é possível termos uma resposta do sistema melhor do que quando utilizamos as regras de gramática.

Neste item, é necessário observar que quanto mais a acurácia (em percentagem) é de acordo com o comando o usuário deseja uma ação e de que o tempo de resposta (em milissegundos) refere-se ao ambiente virtual implementado com a linguagem Java.

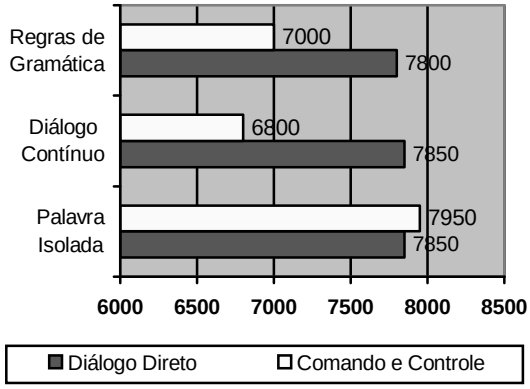

Figura 5 - Gráfico do tempo médio de resposta em milissegundos (ms)

Na
Figura 6 destaca-se o desenvolvimento do AV por meio da união do acionamento do processo de criação do objeto virtual e da alocação do recurso de voz.

Para este processo é estabelecido o seguinte critério: 1) Verificação do estado do mecanismo de reconhecimento de voz; 2) Verificação da gramática após o acionamento do reconhecedor; 3) Acionar o comando do Java 3D e modificar o estado do reconhecedor para liberar mais memória; 4) Verificação do nó através do *Locale* e do *BranchGroup*, usando a parametrização da gramática; 5) A inserção do objeto virtual e o acionamento do sintetizador de voz para dar a resposta audível ao usuário e 6) Desligamento do sintetizador.

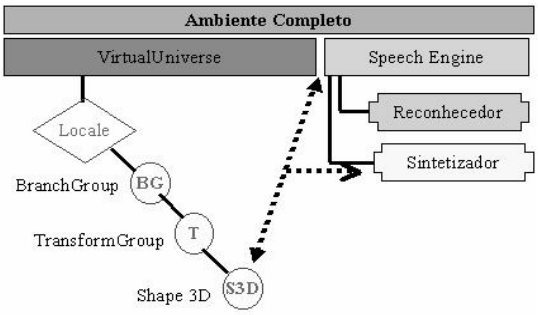

Figura 6 - Visão da programação do ambiente

Através destes passos é possível criar o AV mais adequado sem considerável perda de performance causada pelo processamento das instruções e da geração do ambiente virtual.

Outra técnica para atenuar a perda de performance do Java 3D é a utilização das bibliotecas (pacotes ou packages) não mais na sua forma convencional, visto na Figura 7, mas na forma identificada completa, visto na Figura 8. Esta forma de programação além de atenuar a perda de performance deixa o programa mais legível segundo [14], assegurando o entendimento para futuras manutenções.

Nas figuras 7 e 8 está sendo identificado o mesmo código de programa da classe <Avatar.Java> usado no trabalho de [2].

O ganho de performance observado foi de em média 20 % como mostra a Figura 9.

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.media.j3d.*;
```

Figura 7: Forma convencional de programação Java 3D

```
import java.awt.GraphicsConfiguration;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import javax.media.j3d.VirtualUniverse;
import javax.media.j3d.Locale;
import javax.media.j3d.ViewPlatform;
import javax.media.j3d.BranchGroup;
import javax.media.j3d.TransformGroup;
import javax.media.j3d.Transform3D;
import javax.media.j3d.Canvas3D;
import javax.media.j3d.AmbientLight;
```

Figura 8: forma de programação identificada completa

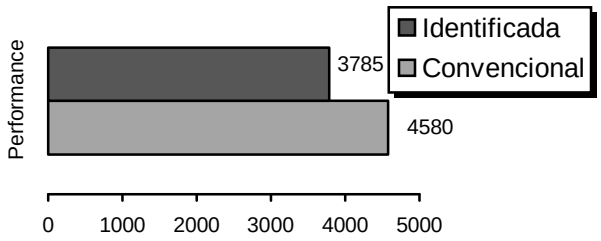

Figura 9: gráfico de performance em milessegundos

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 [3], pois ambas (a classe canvas3D e o canvas do AWT) são especificações da classe Graphics, que acompanha o pacote AWT.

Conforme [3], o pacote AWT (abstract window toolkit) está associado aos 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 fica diferente da original.

Os componentes do pacote Swing não são sobrecarregados pelos recursos GUI complexos da plataforma em que são utilizados, mas dependem da

vinculação da AWT, no qual o Java3D se baseia para desenhar objetos na tela (Figura 10).

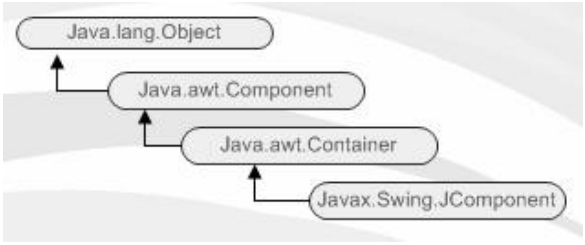

Figura 10: Superclasses comuns dos componentes GUI - java

Mesmo que cada componente AWT possua uma interação separada para o acesso a plataforma, o fato de que quando são utilizados 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 derivado do awt, este é processado primeiro e depois reprocessado para dar vazão ao pacote swing.

8. Conclusões e Trabalhos futuros

Na utilização de sistemas multi-modais, a interface de voz cria uma sensação de imersão e envolvimento, entretanto, quando são utilizados os recursos o sistema virtual tende a transparecer uma queda de performance gráfica.

A premissa do projeto é que um sistema de RV ao utilizar os recursos de voz torna-se mais atrativo (imersivo e envolvente) ao usuário, mas o tempo de resposta (feedback) diminui as expectativas do usuário afastando-o da ilusão criada.

A utilização de uma linguagem gratuita e independente de plataforma operacional favorece o desenvolvimento, pois os custos administrativos são reduzidos e o custo de implementação, pela linguagem responde a processos orientados a objetos, é possível o reuso do código.

Quanto a perda de imersão do usuário ao sistema, o Java possui bibliotecas incorporados como o Java Media Framework, que pode ser utilizada em conjunto para atenuar o atraso da síntese de fala, e quanto ao reconhecimento de fala, através das técnicas demonstradas em [4].

Na próxima fase do trabalho, será desenvolvido o mesmo ambiente virtual utilizando outra tecnologia, o C++ (com o Microsoft Visual C++) e o WorldToolKit (Sense8), com o acesso ao mesmo mecanismo de reconhecimento (IBM Via Voice), para poder determinar qual seria o mais indicado para uma aplicação multimodal em um ambiente virtual.

9. Referências

[1] 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.

[2] Brega, J. R. F.; Sementille, A. C.; Kirner, C.; Devidé, A. H.; Santos, F.; Beldi, L. H. P. (2001) "Uma Biblioteca para Movimentação de Agentes e Avatares Humanóides em Aplicações de Realidade Virtual", 4º.SBC Symposium on Virtual Reality, Florianópolis – SC, Outubro.

[3] Deitel, H. M. ; Deitel, P. Java Como Programar. 4ª.edição. Porto Alegre-RS: Bookman, 2003.

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

[5] DAMASCENO, E.F.., BREGA, J. R. F. Implementação de Jogos com Serviços de Voz em Java. Simpósio Brasileiro de Jogos de Computadores, Curitiba-PR, 2004

[6] Furness, T. A. & Barfield, W Speech Recognition - Past, Present and Future. NTT Review, 1995.

[7] Hunt A. & Walker W. A Fine Grained Component Architecture for Speech Application Development, SUN Research, Project: SMLI TR-2000-86, June 2000.

[8] Jurafsky, D., Martin, J. “Speech and Language Processing”. New Jersey, Prentice-Hall, 2000.

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

[10] Oliveira J.C. Et Ali VIRTUAL THEATER for Industrial Training: A Collaborative Virtual Environment, Canadá, 2000, disponível em www.mcrlab.uottawa.ca/papers/cscc2000-Joliveira.pdf

[11] Pizzolato,E. B. e 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.

[12] Pressman, R.S. Software Engineering: A Practitioner´s Approach, 5ª ed. McGraw-Hill, New York, 2002.

[13] Rodrigues, J.F. Estudo e Desenvolvimento de Aplicações Java com Reconhecimento e Síntese de Voz. Relatórios Técnicos do ICMC. São Carlos, 2001

[14] Sebesta R. W. “Conceitos de Linguagem de Programação”. 4ª ed. Editora Bookman, Rio de Janeiro. 1996.

[15] Shriver S. & Rosenfeld R. Keyword Selection, And The Universal Speech Interface Project, disponivel em <http://www.cs.cmu/~usi,> Acesso em 20 jul. 2003.

[16] Sun. Java™ Speech API Programmer's GuideVersion 1.0 . 1998, acessado em 12 Jul. 2003

[17] Tatham, Mark. Speech Recognition. 1995. Disponível em: <http://www.essex.ac.uk/ speech/teaching/erasmus/recognit.html > Acesso em 15 Jul. 2003.

[18] Zelter D. & Johnson M. B. Interacting with Virtual Environments, Ed. John Wiley & Sons, New York, 1994