Aspectos de Comunicação e de Funcionalidade de um Ambiente Virtual Colaborativo para Aplicações Educacionais

André L. S. Kawamoto(1) Aloísio Pinto(1) Jefferson Cantão(1)
Tereza G. Kirner(1,2) Claudio Kirner(1,2) Raul S. Wazslawick(3)
(1) Fundação Euripides de Marília - FEESR
(2) Centro de Pesquisas de São Carlos - CPSC/ NuPAC
(3) Universidade Federal de Santa Catarina - CTC/INE
{kawamoto, aloisiop, cantao, tkirner, ckirner}@fundanet.br; raul@inf.ufsc.br

Resumo
Este artigo apresenta o AVC-MV, um sistema de realidade virtual para Internet, que tem por objetivo dar suporte ao aprendizado colaborativo de acordo com a abordagem construtivista. São enfocados, principalmente, o protocolo de comunicação desenvolvido e os aspectos funcionais do sistema relacionados à sua atuação como sistema multi-usuário.

Abstract
This article presents the AVC-MV, a virtual reality system for the Internet, which aims to support collaborative learning, according to the constructionism approach. The article focuses mainly on the developed communication protocol, and the functionality issues related to its execution as a multi-user system.

Palavras-chaves: {protocolos, serviços e aplicações, ambiente virtual colaborativo, sistema distribuído, Internet}

Keywords: {protocols, services and applications, collaborative virtual environment, distributed system , Internet}

1. Introdução

Ambientes virtuais colaborativos com propósitos educacionais envolvem aplicações que possibilitam estudantes, dispersos fisicamente e conectados entre si através da Internet, aprender diferentes assuntos que sejam de interesse do grupo. Neste caso, os estudantes podem participar efetivamente da criação e modificação do mundo virtual (ou de parte dele), além atuar em outros experimentos [1].

Este artigo enfoca o ambiente virtual colaborativo AVC-MV (Ambiente Virtual Colaborativo - Museu Virtual), desenvolvido como parte do projeto Museu Virtual [2]. Um dos objetivos do projeto Museu Virtual é o desenvolvimento de ferramentas que possibilitem a criação de mundos virtuais por crianças e adolescentes de maneira colaborativa seguindo os conceitos do construtivismo [3], ou seja, enquanto o mundo virtual é criado, os estudantes se familiarizam e gradualmente adquirem e constroem conhecimento sobre o tema do mundo.

O AVC-MV foi implementado utilizando as linguagens Java e VRML 2.0, integradas por EAI (*External Authoring Interface*) [4] [5] [6], seguindo o modelo cliente-servidor. A comunicação entre o servidor e os clientes utiliza um protocolo, apresentado neste artigo, que dá suporte às funcionalidades que o sistema oferece.

O artigo pretende contribuir para o desenvolvimento de sistemas distribuídos similares ao AVC-MV e está organizado da seguinte forma: a seção 2 destaca trabalhos anteriores relacionados; a seção 3 apresenta os principais aspectos da implementação do sistema; a seção 4 descreve o protocolo de comunicação desenvolvido; a seção 5 exemplifica o funcionamento do sistema; e, por fim, a seção 6 resume as conclusões, indicando trabalhos futuros.

2. Trabalhos Relacionados

São poucos os ambientes virtuais colaborativos que dão suporte ao ensino, já desenvolvidos e em funcionamento, entre os quais se destacam o NICE [7], o HistoryCity [8] e o Gorilla World [9]. Além desses, alguns trabalhos mostram o desenvolvimento de ambientes virtuais distribuídos, como o projeto AGORA [10], e outros, descritos em [11].

O projeto NICE (*Narrative-based, Immersive, Constructivist/Collaborative Environment*), desenvolvido pelo *Interactive Computing Environments Laboratory* e pelo *Electronic Visualization Laboratory* da Universidade de Illinois, em Chicago, é um ambiente distribuído e imersivo, implementado utilizando tecnologia CAVE. O objetivo do NICE é fornecer um ambiente no qual crianças constroem e cultivam ecossistemas virtuais simples, colaboram, via rede, com outras crianças remotamente localizadas e criam histórias a partir de suas interações nos mundos real e virtual [7].

A arquitetura do NICE é baseada no modelo cliente-servidor e utiliza um protocolo desenvolvido para dar suporte às características de dados de realidade virtual e possibilitar aos usuários entrar e abandonar o ambiente facilmente a partir de qualquer lugar na Internet. O servidor NICE mantém a base de dados e assegura a consistência do sistema inteiro.

HistoryCity, desenvolvido em Cingapura, é um ambiente colaborativo distribuído, que visa dotar crianças de conhecimento a respeito da história da antiga Cingapura. Foi desenvolvido utilizando o software NetEffect [12] [13]. NetEffect segue uma arquitetura cliente-servidor que particiona um mundo virtual em comunidades, distribui essas comunidades entre um conjunto de servidores e migra os clientes de um servidor para outro na medida em que os clientes se movimentam através das comunidades. Tal arquitetura visa minimizar o tráfego de rede, particularmente o tráfego entre diferentes servidores.

A arquitetura do HistoryCity consiste de um servidor mestre e *n* servidores 'pares' (*peer servers*). Cada servidor mantém um grafo completo da topologia, o que permite aos servidores se comunicarem sem atravessar um servidor intermediário. Um cliente está conectado a apenas um servidor em qualquer instante, mas pode migrar de um servidor para outro dinamicamente [13].

O Gorilla World foi desenvolvido por uma equipe do *Georgia Technology Institute*. Trata-se de um mundo virtual colaborativo para dar suporte ao aprendizado do comportamento de grupos de gorilas. O Gorilla World baseia-se no comportamento real dos gorilas que vivem no zoológico de Atlanta, e tem sido utilizado por crianças em algumas escolas americanas [9].

AGORA (que significa "mercado" ou "lugar público", em grego antigo), foi desenvolvido no Japão pelo *Fujitsu Laboratories*. Sua arquitetura foi projetada para apoiar o desenvolvimento de comunidades virtuais. Essa arquitetura emprega uma técnica para dividir e gerenciar o espaço virtual, e também se baseia no modelo cliente-servidor. O servidor gerencia as comunidades e mantém a consistência dos dados. Nesse projeto, foram desenvolvidos um modelo de mensagens para a comunicação e uma aplicação cliente, chamada *community browser*, para permitir a navegação no mundo virtual [10].

O sistema AVC-MV foi inspirado, em vários aspectos, nas idéias do projeto NICE, principalmente no que diz respeito à criação de um ambiente de ensino construtivista. Assim como os projetos AGORA, HistoryCity e NICE, o AVC-MV baseia-se numa arquitetura cliente-servidor, adaptada aos objetivos e recursos técnicos disponíveis ao projeto.

O Servidor AVC-MV é responsável pelo gerenciamento da comunicação entre os usuários e pela manutenção da consistência da base de dados do ambiente virtual, enquanto os clientes do sistema utilizam um navegador para explorar e manipular objetos no mundo virtual. Os principais benefícios obtidos por esse modelo são:

- Compatibilidade com as características modulares da programação em Java (usada na implementação), o que torna fácil a implementação e manutenção do software;
- Desenvolvimento flexível de interfaces no lado do cliente;
- Boa escalabilidade, uma vez que a adição e remoção de clientes é facilitada;
- Minimização do tráfego de rede, porque embora o servidor precise estar funcionando ininterruptamente, os clientes utilizam a conexão apenas para enviar ou receber mensagens necessárias;
- Heterogeneidade, que possibilita o uso de diferentes plataformas computacionais na Internet, acessando a aplicação. Para isso, basta utilizar o mesmo protocolo que o servidor.

Particularmente, as abordagens para organizar os conjuntos de dados distribuídos e traçar o comportamento dos usuários, utilizadas, respectivamente, pelo AGORA e NICE, foram muito úteis na especificação do AVC-MV.

3. Implementação do Sistema AVC-MV

Existem várias tecnologias para a construção de sistemas distribuídos de realidade virtual. Uma delas, que combina as linguagens VRML e Java, foi utilizada neste trabalho, principalmente por oferecer maior portabilidade e permitir o atendimento de grande quantidade de usuários.

VRML é uma linguagem para a descrição de ambientes tridimensionais virtuais que podem ser visualizados localmente ou transferidos através da Internet. VRML fornece recursos poderosos para a modelagem de modelos e cenas tridimensionais complexas; não fornece, porém, mecanismos para o controle de múltiplos usuários [5].

Java é uma linguagem orientada a objetos independente de plataforma, que visa o desenvolvimento de aplicações que poderão ser executadas em ambientes de rede e na Internet [4]. Atualmente, a maioria dos navegadores é capaz de executar código Java.

O AVC-MV, ilustrado na Figura 1, foi implementado utilizando a combinação VRML. Esta combinação possibilita que o código Java acesse e modifique o mundo virtual modelado em VRML/Java, de forma a fornecer aos usuários uma navegação suave e coerente. Para tanto, foi utilizada *External Authoring Interface* (EAI) [6], que permite uma comunicação bidirecional entre o applet Java e o *plug-in* VRML.

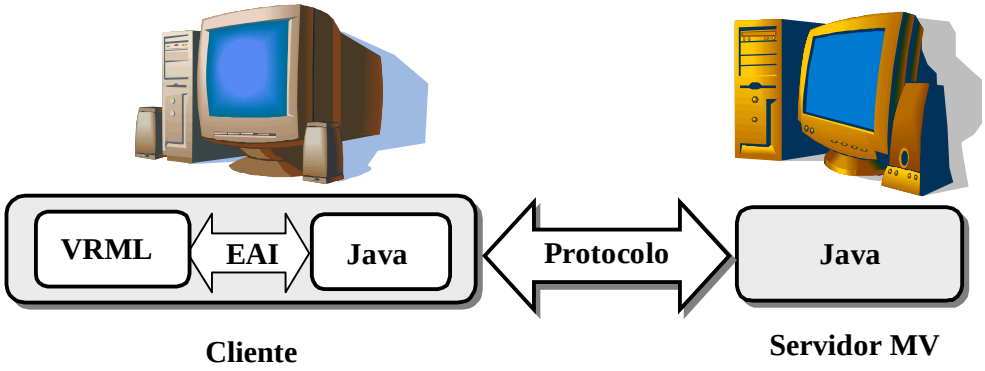


Figura 1 - Visão Geral do Sistema AVC-MV

Em uma aplicação que utiliza EAI, o VRML é responsável por mostrar a cena 3D ao usuário, enquanto Java manipula todos os eventos e mensagens no mundo virtual. Por exemplo, quando um avatar se move no ambiente virtual, um sensor VRML (Proximity Sensor – PS) detecta sua nova posição e informa ao código Java, que transmite esta informação para todos os usuários no sistema. A comunicação implementada no AVC-MV emprega a tecnologia de *sockets* e necessitou de várias classes de implementação. Essas classes utilizam um protocolo de comunicação desenvolvido no projeto, descrito na próxima seção.

4. Protocolo de Comunicação

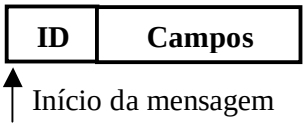
4.1 Estrutura do Protocolo

O protocolo de comunicação desenvolvido visa atender os requisitos do sistema no que diz respeito à transmissão de dados do ambiente de realidade virtual (como posicionamento de avatares, movimentação de objetos, etc.), de forma a viabilizar todas as funções necessárias.

Os elementos que participam deste protocolo são:

- **S**, que representa o Servidor AVC-MV;
- $\mathbf{C_n}$, que representa o conjunto de clientes conectados ao sistema;
- $\mathbf{C_0}$, que representa um único cliente do sistema, emissor da mensagem.

O formato das mensagens utilizadas pelo protocolo é esquematizado a seguir.



Nesse formato, o primeiro campo (ID) é um valor inteiro que determina qual o tipo da mensagem e os campos seguintes formam uma estrutura variável de acordo com o número e tipo de parâmetros que cada mensagem requer. Essas mensagens foram definidas com base nos dados exigidos pelos métodos de EAI para manipulação dos objetos VRML, e visam manter a comunicação entre os usuários e a coerência do mundo virtual. Os diagramas de eventos no tempo a seguir ilustram o protocolo desenvolvido.

a) Entrada no Sistema

A Figura 2(a) mostra o protocolo para entrada no sistema. Inicialmente, o cliente que deseja entrar no sistema envia uma mensagem do tipo 0 (*requerConexao*) ao servidor. Ao receber esta mensagem, o servidor responde ao cliente com uma mensagem do tipo 1 (*conexaoOK*), e logo em seguida manda um *broadcast* a todos os outros clientes avisando a entrada do novo usuário através de mensagens do tipo 2 (*avisaEntradaUsuario*). Ao receber o *conexaoOK*, o cliente envia uma mensagem do tipo 3 (*requerListaUsuarios)* para receber a lista de usuários conectados ao mundo e uma mensagem do tipo 5 (*requerListaObjetos*) para receber a lista de objetos que foram inseridos no mundo. Essas mensagens são respondidas pelo servidor, com mensagens do tipo 4 (*listaUsuarios*) e do tipo 6 (*listaObjetos*), respectivamente.

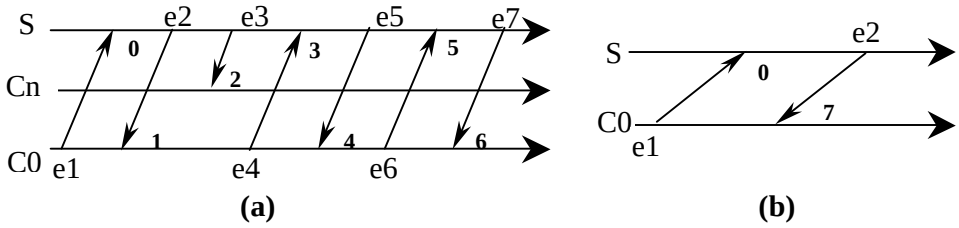


Figura 2 - Protocolo para Entrada no Sistema

Caso não seja possível ao cliente entrar no mundo escolhido (por exemplo, caso o limite de usuários tenha sido alcançado), o servidor responde a mensagem de *requerConexao* com uma mensagem do tipo 7 (*conexaoNaoOK*), como mostra a Figura 2(b). Ao cliente, existe a possibilidade de aguardar para tentar a conexão mais tarde, ou escolher um outro mundo e tentar conectar-se.

b) Mudança de Mundo

A Figura 3(a) ilustra o protocolo para mudança de mundo virtual. O usuário que deseja mudar de mundo envia uma mensagem do tipo 8 (*mudaMundo*) ao servidor. O servidor envia uma mensagem de *conexaoOK* ao usuário, e avisa a todos os clientes conectados no mundo em que o usuário se encontra que o usuário abandonou o mundo virtual, através de uma mensagem do tipo 9 (*avisaSaidaUsuario*). Em seguida, avisa a todos os usuários do novo mundo a entrada do usuário, através de uma mensagem *avisaEntradaUsuario*. Isso feito, o usuário requer a lista de usuários e a lista de objetos inseridos no mundo de forma análoga ao protocolo de entrada no sistema. Caso não seja possível a mudança de mundo, o servidor responde ao pedido do usuário com uma mensagem *conexaoNaoOK*, como mostra a Figura 3(b).

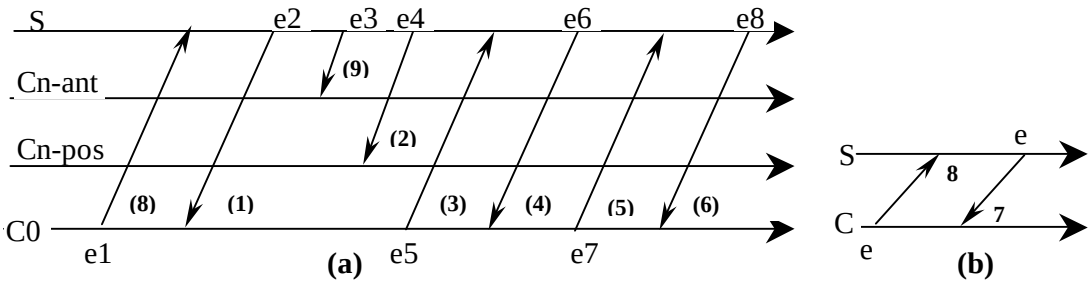


Figura 3 - Protocolo para Mudança do Mundo

c) Mudança na Posição e Orientação do Avatar

Quando um avatar se movimenta no mundo virtual, essa informação é transmitida ao servidor através de uma mensagem do tipo 11 (*mudaPosicaoAvatar*) e mandada para todos os outros clientes do mundo através de uma mensagem do tipo 12 (*atualizaPosicaoAvatar*), como mostra a Figura 4(a).

Quando o avatar de um usuário gira no mundo virtual, essa informação é transmitida ao servidor através de uma mensagem do tipo 13 (*mudaOrientacaoAvatar*) e mandada para todos os outros clientes do mundo, através de uma mensagem do tipo 14 (*atualizaOrientacaoAvatar*), conforme ilustra a Figura 4(b).

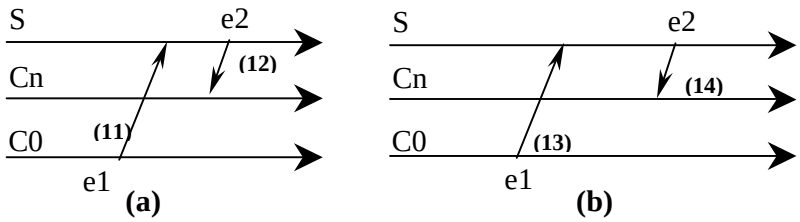


Figura 4 - Protocolos para Mudança de Posição e Orientação de Avatares

d) Inserção de Objeto

Para inserir um novo objeto no mundo virtual, um cliente envia uma mensagem do tipo 15 (*insercaoObjeto*). Ao receber essa mensagem, o servidor responde com uma mensagem do tipo 16 (*insercaoObjetoOK*) e notifica a todos os outros clientes do mundo virtual com uma mensagem do tipo 17 (*novoObjeto*). Isso é mostrado pela Figura 5a. No caso de não ser possível a inserção do novo objeto, é enviada uma mensagem do tipo 18 (*erroInsercaoObjeto*) ao usuário que requisitou a inserção -Figura 5(b).

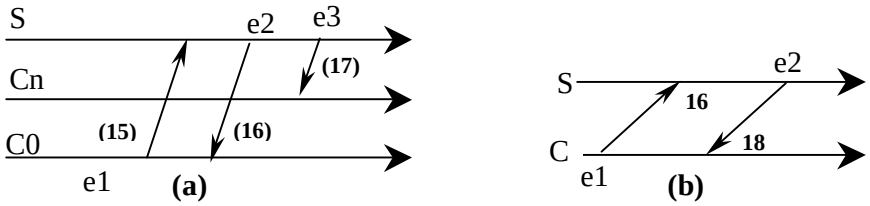


Figura 5 - Protocolo para inserção de objetos

e) Mudança na Posição e Orientação de Objeto

Quando um cliente controla um objeto inserido no mundo, ele pode alterar a posição e a orientação desse objeto. Isso é feito, respectivamente, através do envio de mensagens do tipo 19 (*mudaPosicaoObjeto*) e do tipo 21 (*mudaOrientacaoObjeto*), recebidas pelo servidor, que informa a todos os usuários conectados ao mundo com mensagens do tipo 20(*atualizaPosicaoObjeto*) e 22 (*atualizaOrientacaoObjeto*) - Ver Figura 6.

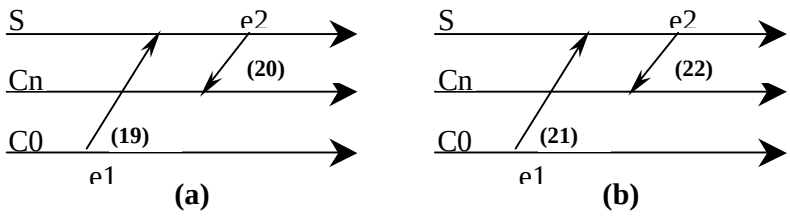


Figura 6 - Protocolos para Alteração de Posição e Orientação de Objetos

f) Mensagens de Chat

As mensagens de chat podem ser do tipo pública ou privativa. Quando um cliente envia uma mensagem pública, destinada a todos os outros clientes conectados, é enviada uma mensagem do tipo 23 (*enviaMensagemPublica*) ao servidor, que faz o *broadcast* dessa mensagem a todos os usuários do mundo através de uma mensagem do tipo 24 (*mensagemPublica*), como mostra a Figura 7(a).

No caso do envio de uma mensagem destinada a apenas um outro cliente do mundo, é enviada uma mensagem do tipo 25 (*enviaMensagemPrivativa*) ao servidor, que repassa a mensagem ao usuário escolhido através de uma mensagem do tipo 26 (*mensagemPrivativa*), como mostra a Figura 7(b).

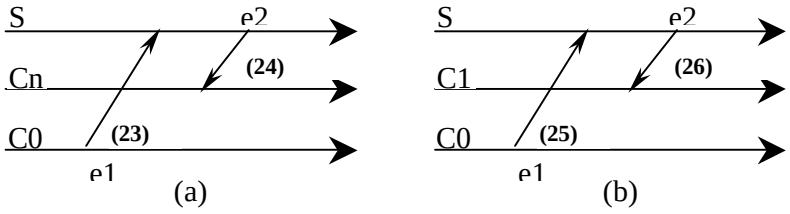


Figura 7 - Protocolos para Envio de Mensagens de *Chat*

g) Desconexão

Quando um usuário deseja desconectar-se do sistema, ele envia uma mensagem do tipo 10 (*requerDesconexao*). Ao receber uma mensagem desse tipo, o servidor envia uma mensagem do tipo 9 (*avisaSaidaUsuario*) a todos os outros clientes do mundo. A Figura 8 ilustra o protocolo para a desconexão

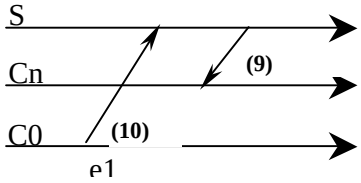


Figura 8 - Protocolo para Desconexão

4.2 Detalhamento das Mensagens

As mensagens utilizadas pelo protocolo são esquematizadas a seguir.

- ***requerConexao***

0	ModeloWRL	URL	apelido	nome	email	J	adicional	nomeMundo

- ***conexaoOK***

1

- ***avisaEntradaUsuario***

2	Modelo WRL	URL	apelido	nome	email	j	adicional

- ***requerListaUsuarios***

3

- ***listaUsuarios***

4	Modelo WRL	URL	apelido	nome	email	j	adicional	-1

loop

- ***requerListaObjetos***

5

- ***listaObjetos***

6	IdObjeto	Modelo WRL	coordX	coordY	coordZ	-1

loop

- ***conexaoNaoOK***

7	mensagem

- ***mudaMundo***

8	nomeMundo

- ***avisaSaidaUsuario***

9	apelido

- ***requerDesconexao***

10	nomeMundo

- ***mudaPosicaoAvatar***

11	apelido	coordX	coordY	coordZ	IdObjeto

- ***atualizaPosicaoAvatar***

12	apelido	coordX	coordY	coordZ	IdObjeto

- ***mudaOrientacaoAvatar***

13	apelido	eixoX	eixoY	eixoZ	Variacao

- ***atualizaOrientacaoAvatar***

14	apelido	eixoX	eixoY	eixoZ	variacao

- ***insercaoObjeto***

15	URLObjeto	coordX	coordY	coordZ

- ***insercaoObjetoOK***

16

- ***novoObjeto***

17	IdObjeto	URLObjeto	coordX	coordY	coordZ

- ***erroInsercaoObjeto***

18	Mensagem

- ***mudaPosicaoObjeto***

19	IdObjeto	coordX	coordY	coordZ

- ***atualizaPosicaoObjeto***

20	IdObjeto	coordX	coordY	coordZ

- ***mudaOrientacaoObjeto***

21	IdObjeto	eixoX	eixoY	eixoZ	variacao

- ***atualizaOrientacaoObjeto***

22	IdObjeto	eixoX	eixoY	eixoZ	variacao

- ***enviaMensagemPublica***

23	mensagem

- ***mensagemPublica***

24	mensagem

- ***enviaMensagemPrivativa***

25	IdPartner	mensagem

- ***mensagemPrivativa***

26	IdPartner	mensagem

Descrição dos Campos das Mensagens

- **Modelo WRL** - é uma *string* contendo o modelo virtual que será utilizado pelo usuário como avatar no mundo virtual.
- **URL -** é uma *string* que indica o endereço da Internet onde o *Modelo WRL* pode ser encontrado.
- **Apelido** - é uma *string* informando qual apelido será utilizado pelo usuário no mundo virtual.
- **Nome –** *string* com o nome do usuário.
- **E-mail –** *string* contendo o endereço de correio eletrônico do usuário.
- **J –** tamanho em *bytes* do campo adicional preenchido pelo usuário.
- **Adicional –** *string* contendo o texto do campo adicional do usuário.
- **IdObjeto –** valor *inteiro* identificador do objeto.
- **URLObjeto** - é uma *string* contendo o modelo virtual que será utilizado pelo sistema para representar o objeto no mundo virtual.
- **coordX,coordY,coordZ –** valores reais das coordenadas X, Y e Z respectivamente com relação ao sensor de proximidade VRML (ProximitySensor) colocado no centro do mundo virtual.
- **nomeMundo –** *string* representando o nome do mundo ao qual o usuário vai se conectar ou onde está navegando.
- **eixoX,eixoY,eixoZ –** valores *reais* que indicam em torno de qual eixo o objeto ou avatar vai se movimentar. O campo **variacao** é um valor real que identifica a variação, em radianos, na rotação do avatar ou objeto.
- **Mensagem -** *string* contendo mensagem de erro ou texto.
- **IdPartner** - é uma *string* para identificar o usuário ao qual se deseja enviar uma mensagem privativa.

O protocolo dá suporte ao funcionamento do sistema, descrito na próxima seção.

5. Funcionalidade do Sistema

O funcionamento do sistema é bastante simples. Para conectar-se ao sistema, os clientes acessam uma página HTML hospedada no mesmo endereço em que o servidor está executando. Essa página contém um *applet* que oferece a interface do usuário.

Uma vez conectado ao sistema, é possível ao usuário navegar e interagir em um mundo virtual, inserindo e manipulando objetos, e, além disso, utilizar um *chat* de texto para comunicar-se com outros usuários remotos. As interfaces do servidor e do cliente são descritas a seguir. As ilustrações são de uma aplicação sobre Mundos Virtuais de Capoeira, desenvolvida pelos autores do presente artigo para testar o potencial e funcionalidade do sistema distribuído [14]

5.1 Interface do Servidor

A interface do servidor é mostrada na Figura 9. Vários componentes podem ser identificados na figura, incluindo status, lista de usuários, lista de mundos, dados do usuário (endereço IP, nome, email), botão de desconexão, lista de objetos.

Quando um mundo é selecionado na lista de mundos, os usuários conectados a este mundo são mostrados na lista de usuários e todos os objetos que foram inseridos no mundo são mostrados na lista de objetos. Se um usuário é selecionado, sua informação é mostrada nas caixas de texto. É possível desconectar um usuário do sistema, através do botão de desconexão.

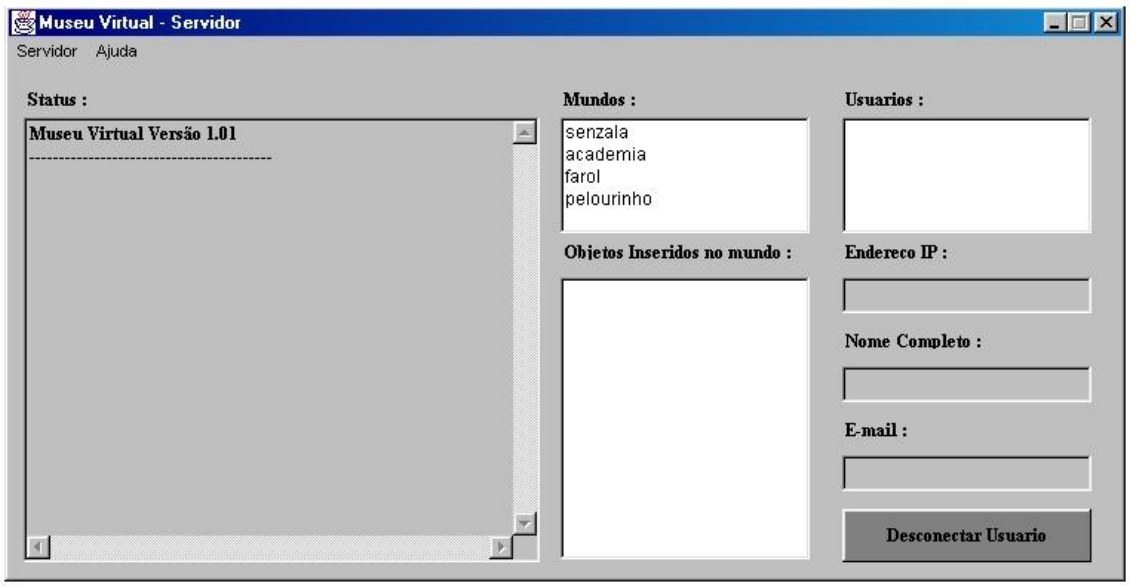


Figura 9 - Interface do Servidor

5.2 Interface do Cliente

O cliente foi implementado para ser executado em qualquer navegador de internet capaz de visualizar arquivos VRML e executar *applets* Java. Ao acessar o endereço em que o servidor se encontra funcionando, é carregada uma página HTML contendo o applet mostrado na Figura 10.

A interface do usuário fornece uma área de *chat*, contendo o local em que as mensagens são exibidas (*ChatArea*) e o local em que as mensagens são redigidas (ChatText), status, lista de usuários, o navegador VRML mostrando o mundo virtual, e vários botões para a interface do usuário.

Quando o botão *User* é clicado, habilita a janela mostrada na Figura 11, onde o usuário entra com seus dados pessoais (nome, apelido, email). Existe também um campo onde o usuário pode fornecer qualquer informação adicional, se desejado.

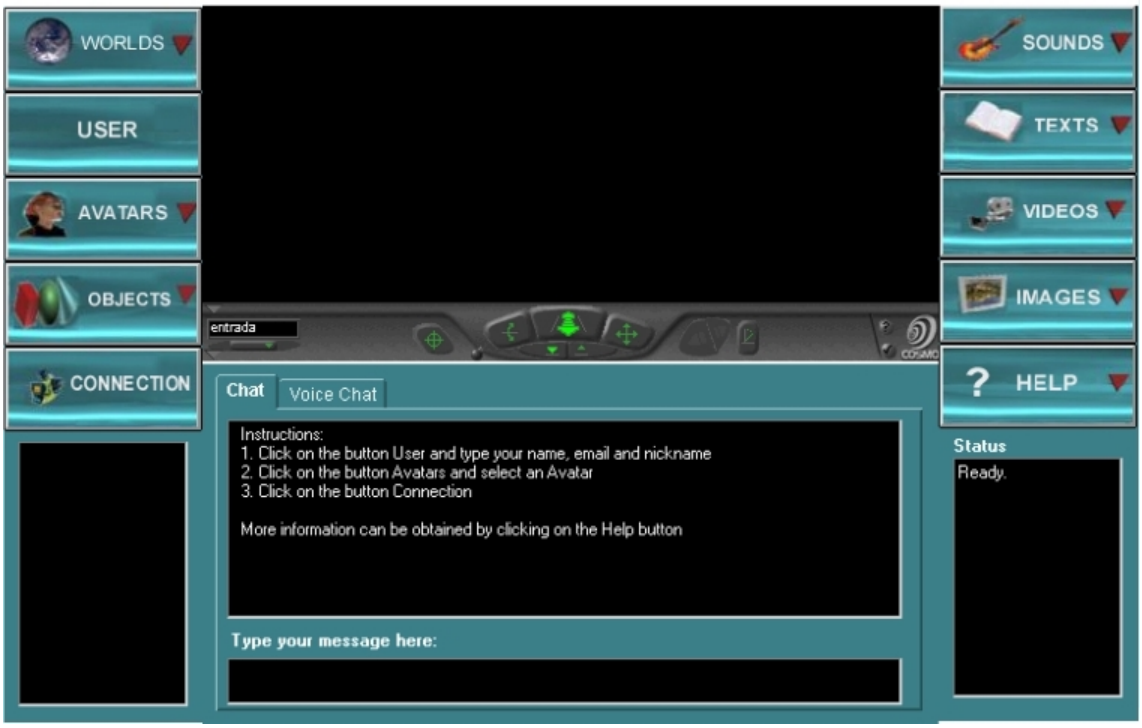


Figura 10 - Interface do Cliente

O botão *Avatar* habilita uma janela contendo todos os avatares possíveis de serem utilizados para representar o usuário no mundo virtual. A interface para a seleção de um avatar é mostrada na Figura 12.

Figura 11 - Entrada de Dados do Usuário

Figura 12 - Escolha de Avatar

O botão de Mundos habilita uma janela que exibe ao usuário todos os possíveis mundos virtuais em que o usuário pode navegar. A interface para a escolha de um mundo é mostrada na Figura 13.

Figura 13 - Escolha de Mundo

Uma vez que o usuário forneceu todas as informações necessárias, basta clicar no botão de conexão para entrar em um ambiente virtual compartilhado, onde ele é capaz de visualizar o que os outros usuários estão fazendo e interagir com eles.

Se um usuário deseja inserir um novo objeto no mundo, ele clica no botão Objetos, e uma janela, apresentada na Figura 14, é apresentada. Nessa janela, vários objetos podem ser escolhidos para inserção.

É possível também modificar a posição e orientação de objetos inseridos no mundo. Para tanto, é necessário clicar no objeto desejado no mundo virtual. Quando um objeto é selecionado, o usuário passa a controlá-lo e pode rotacionar ou transladar o objeto através da janela que é apresentada na Figura 15.

Figura 14 - Inserção de objetos

Figura 15 - Alteração no Posicionamento do Objeto

Para possibilitar a comunicação entre usuários, foi implementado um *chat* de texto. A interface do *chat* (constante da Figura 10) consiste da *ChatArea* e *ChatText*. A ChatArea ecoa as mensagens vindas de todos os usuários conectados ao sistema. *ChatText* é o local em que os usuários digitam as mensagens a ser enviadas a todos os outros usuários. A lista de usuários mostra os usuários conectados ao sistema. Um usuário inicia um *chat* privativo com outro clicando no apelido desejado. A interface do *chat* privativo utiliza a janela apresentada na Figura 16.

Figura 16 - *Chat* Privativo

Os botões de Sons, Textos, Vídeos e Imagens, quando clicados, exibem janelas similares à mostrada na Figura 17. Estas janelas apresentam mundos virtuais onde é possível navegar e acessar vários sons, textos, vídeos e imagens relacionadas ao tema do mundo em que se navega.

Figura 17 - Acesso a Vídeos

A funcionalidade do Sistema AVC-MV foi testada extensivamente através de exemplos completos de aplicação, como o Mundo da Capoeira, tendo demonstrado resultados satisfatórios.

6. Conclusões e Trabalhos Futuros

Este artigo apresentou o AVC-MV (Ambiente Virtual Colaborativo – Museu Virtual), desenvolvido para suportar aplicações educacionais envolvendo aprendizado colaborativo. O AVC-MV é um sistema multi-usuário distribuído que opera na Internet e pretende melhorar e facilitar o aprendizado, de acordo com os conceitos do construtivismo.

O reduzido número de sistemas similares existentes atualmente motivou o desenvolvimento do AVC-MV. Além disso, aplicações educacionais podem ser melhoradas extensivamente com o uso de ambientes virtuais colaborativos, principalmente se estes são acessáveis pela Internet.

O artigo buscou contribuir para o desenvolvimento de sistemas semelhantes, de forma que o foco principal direcionou-se para aspectos relacionados à implementação do projeto.

O AVC-MV representa uma experiência proveitosa de desenvolvimento de ambientes virtuais colaborativos, principalmente no Brasil. Como refinamentos e melhorias no sistema, em curto prazo, estão planejados:

- Incorporação de recursos gráficos e de som ao *chat*;
- Otimização da troca de mensagens, objetivando uma melhor performance do sistema;
- Inclusão de um módulo para monitoramento e análise do comportamento dos usuários, enquanto eles estiverem usando o sistema;
- Extensão da biblioteca de objetos e de mundos virtuais, de modo a permitir a construção e navegação de novos mundos virtuais relacionados a temas diversos;
- Teste do sistema em escolas, visando identificar a usabilidade do sistema e sua contribuição ao processo de aprendizado de crianças e adolescentes.

Outras inovações, também consideradas, são:

- Uso de avatares mais realistas como, por exemplo, com faces reais obtidas de fotos dos usuários;
- Adição de uma ferramenta de autoria ao sistema, que permita a construção colaborativa de novos objetos;
- Geração de uma nova versão do sistema, utilizando software alternativo, como Java3D e X3D;
- Busca de novas aplicações educacionais, como jogos *on line*.

Agradecimentos

Os autores agradecem ao programa ProTem-CC - Fase IV do CNPq, pelo apoio financeiro concedido ao desenvolvimento do Projeto Museu Virtual.

Referências Bibliográficas

[1] C. Youngblut. Educational Uses of Virtual Reality Technology. Technical Report D2128, Institute for Defense Analysis, USA, 1999.

[2] R.S. Wazlawick, L.C. Fagundes, and T.G. Kirner. Museu Virtual: Ferramenta de Autoria para Criação de Museus em Realidade Virtual para Apoiar a Aprendizagem Colaborativa via Internet. Projeto ProTem - Fase IV - CPNq, Brazil, 1999.

[3] J. Piajet. To Understand is to Invent: The Future of Education. Grossman, New York, 1973.

[4] Java 2 Platform – API Specification. http://java.sun.com/j2se/1.3/docs/api/index.html/

[5] VRML - The Virtual Reality Modeling Language. International Standard ISO/IED 14772-1:1997. http://www.web3d.org/Specifications/VRML97/

[6] VRML - Proposal for a VRML 2.0 Informative Annex – External Authoring Interface Reference. http://www.vrml.org/WorkingGroups/vrml-eai/ ExternalInterface.html/

[7] A. Johnson, M. Roussos, J. Leigh, C. Barnes, C. Vasilakis, and T. Moher. The NICE Project: Learning Together in a Virtual World. Proceedings of the IEEE Virtual Reality Annual International Symposium, Atlanta, Georgia, 1998, pp. 176-183

[8] HistoryCity. http://www.historycity.org.sg/

[9] D. Alison, B. Wills, F. Hodges, and J. Wineman. Gorillas in the Bits. Proceedings of IEEE Virtual Reality Annual International Symposium, March 1997, pp. 69-76.

[10] H. Hiroaki et al. Space Sharing Architecture for a Virtual Three-Dimensional Virtual Community. IOP/IEE Distributed Systems Engineering, 5:101-106, 1998.

[11] S. Singhal, and M. Zyda. Networked Virtual Environments: Design and Implementation. Prentice-Hall, 1999.

[12] T. K. Das et al. Developing Social Virtual Worlds using NetEffect. Proceedings of the WET-ICE 97, 1997.

[13] T. K. Das et al. NetEffect: A Network Architecture for Large-Scale Multi-User Virtual Worlds. International Conference on Virtual Reality Software Technology, 1997.

[14] T.G. Kirner, C. Kirner, A.L.S. Kawamoto, J. Cantão, A. Pinto, and R.S. Wazlawick. Development of a Collaborative Virtual Environment for Educational Applications. Proceedings of the ACM Web3D 2001 Symposium, February 2001, p. 61-68.