UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

2005.1

GERENCIAMENTO DE OBJETOS DE REALIDADE VIRTUAL REUTILIZÁVEIS PARA AMBIENTES VIRTUAIS DE ENSINO

TRABALHO DE GRADUAÇÃO
EM
BANCO DE DADOS

Aluno: Leonardo Cabral de M. Sarmento, lcms@cin.ufpe.br.
Orientador: Fernando da Fonseca de Souza, fdfd@cin.ufpe.br.

Recife, 19 de Agosto de 2005.

A Deus

AGRADECIMENTOS

Muitas foram as pessoas que me ajudaram na confecção deste trabalho. Agradeço a todos que contribuíram de forma direta ou indireta, e em especial a:

- Meus pais, Leonardo e Eudócia, que me incentivaram por toda a vida, dando total apoio e confiança em todos os momentos, principalmente nos mais difíceis;
- Minhas irmãs, Renata e Larissa, que sempre me incentivaram nesta longa jornada e que agora dividem comigo esta felicidade;
- Minha avó, Marinã, que sempre confiou em mim e foi um incentivo para que eu sempre buscasse o fazer o meu melhor;
- Meus familiares que, mesmo distantes, sempre me proporcionaram momentos de alegria e me deram força para seguir nesta caminhada;
- A todos os amigos do Centro de Informática que compartilharam as dificuldades e alegrias ao decorrer da graduação;
- AVCIn, uma associação que só me deu alegria em todos os momentos do Centro de Informática;
- Meu grande amigo Mauro Quaresma que nos deixou no meio desta caminhada, mas que, mesmo com pouco tempo de convivência, conseguiu deixar marcas e ensinamentos que me acompanharão para resto da minha vida;
- Fernando Fonseca, meu orientador, por ter me apresentado este grande tema, e ter me guiado na construção deste trabalho, fazendo valiosas correções, sempre com muita paciência; e
- Marcos Aquino, que me deu um grande apoio para o desenvolvimento deste trabalho.

RESUMO

O crescente uso de técnicas de Realidade Virtual para a modelagem de ambientes Virtuais de Ensino, tem permitido a construção de ambientes mais interativos. A incorporação de técnicas de agentes inteligentes para identificar o perfil do usuário e modificar o mundo de acordo com suas necessidades, permite criar mundos mais próximos de sua realidade. Os ambientes atuais, porém, têm limitações relativas à adaptação através do nível de detalhe dos objetos, à impossibilidade de adaptação durante a interação do usuário e falta de mecanismos para reutilizar objetos de realidade virtual. Este trabalho propõe o desenvolvimento de um módulo de gerenciamento de objetos de realidade virtual que possa dar suporte à construção de ambientes virtuais que superem estas dificuldades. Este módulo, denominado MoGORViR, será construído utilizando o suporte a documentos XML oferecido pelo SGBD Oracle 10g. O MoGORViR será validado com uma aplicação que permite a construção de um mundo virtual através de objetos armazenados em um SGBD e que permite a visualização destes mundos em diferentes níveis de detalhe, simulando diferentes usuários.

ABSTRACT

The increasing use of techniques of Virtual Reality for the Education Virtual Environment modeling has allowed a more interactive environment construction. The incorporation of techniques of intelligent agents to identify the profile of the user and to modify the world in accordance with his or her necessities, allows to create virtual worlds next to the reality. The current environments, however, have relative limitations to the adaptation through the level of the objects details, to the adaptation impossibility during the interaction of the user and lack of mechanisms to reuse objects of virtual reality. This work considers the development of a module of object management of virtual reality that can give support to the virtual environment construction that surpasses these difficulties. This module, called MoGORViR, will be constructed using the support offered to XML documents by SGBD Oracle 10g. The MoGORViR will be validated with an application that allows the construction of a virtual world through objects stored in a SGBD and that it allows to the visualization of these worlds in different levels of detail, simulating different users.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

CAPÍTULO 1

INTRODUÇÃO

O advento da internet provocou uma revolução na nossa sociedade. As novas formas de comunicação propiciadas pela internet permitiram que barreiras de espaço e tempo fossem ultrapassadas, possibilitando uma maior integração entre as pessoas independentemente de sua localização geográfica.

Uma área que está se beneficiando largamente do uso da internet é a área de educação a distância. Segundo Tessarollo [Tessarollo, 2000], uma das vantagens dos cursos que são desenvolvidos na internet reside na facilidade de disponibilizar o conteúdo para alunos em qualquer parte do mundo, de maneira mais rápida e agilizada do que os métodos convencionais de educação a distância. Além disso, os recursos de comunicação da internet, quando utilizados em um curso a distância, podem tornar mais eficiente a comunicação entre o professor e o aluno e entre os alunos, se comparado com outros métodos convencionais, como, por exemplo, o correio comum.

Atualmente, o ensino através do computador em ambientes de Ensino a Distância – EAD tem estimulado o desenvolvimento de diversas ferramentas, enfocando tanto a transmissão das informações a longa distância, como a construção do conhecimento e maior interatividade do estudante com o sistema [Aquino, 2001].

Uma vez que computadores podem concentrar diferentes mídias em uma única aplicação e também podem incorporar interatividade, o uso da Realidade Virtual (RV) parece ser crucial no desenvolvimento de ambientes imersivos de transferência de conhecimento [Saldías, 2002].

O uso de técnicas de Realidade Virtual permite que o usuário possa realizar imersão, navegação e interação em um ambiente sintético tridimensional gerado por computador, utilizando canais multi-sensoriais em tempo real (Dizeró, 1999). Em ambientes de realidade virtual, o computador gera uma simulação de um mundo que pode ser o real ou imaginário, utilizando uma representação gráfica ou textual. A

interação, em particular, permite ao sistema detectar as entradas do usuário e modificar instantaneamente o mundo virtual e as ações sobre ele.

Tornar esta modificação do mundo virtual (adaptatividade) uma realidade vem sendo pesquisado largamente. Ambientes que possuem estas características são chamados neste trabalho de Ambientes Virtuais Adaptativos. Estes ambientes têm a capacidade de modelar o mundo de acordo com as características do usuário, tais como sua capacidade cognitiva, sua cultura digital, seu conhecimento das ferramentas de aprendizagem e sua visão de mundo.

A construção dos ambientes virtuais (AV) é realizada através da utilização de bibliotecas e padrões gráficos 3D [Osório et al., 2004], tais como Java3D [Java3D, 2004], VRML (Virtual Reality Modeling Language) [VRML, 1997] e X3D [X3D, 2004].

Como a adaptação em um AV envolve inserção, exclusão ou modificação de objetos no ambiente onde o usuário está inserido, a utilização de um SGBD para o armazenamento, recuperação e gerenciamento adequado desses objetos pode facilitar e agilizar a construção de AV Adaptativos.

Este trabalho tem como objetivo a especificação e implementação de um módulo de gerenciamento de objetos de RV para AV Adaptativos. Este módulo será utilizado no projeto “Ambientes Virtuais Adaptativos para Plataformas EAD“ que está sendo desenvolvido no CIn sob a coordenação do orientador deste trabalho.

1.1 JUSTIFICATIVA

A motivação para a realização deste trabalho se deve ao uso cada vez maior de ferramentas de ensino via Web, que agregam adaptatividade e ambiente de tutoria, e a utilização da Realidade Virtual (RV) para simulações, permitindo uma maior interatividade do usuário com o ambiente.

Entretanto, as plataformas de Ensino a Distância ainda não apresentam mundos virtuais personalizados que possam se adequar dinamicamente a qualquer tipo de usuário. Estas limitações estimulam os estudos de ambientes virtuais que se adaptem ao perfil do usuário e a sua capacidade de interagir e de aprender.

Os principais problemas identificados com os sistemas atuais são os seguintes:

- A inexistência de adaptação relativa à apresentação de objetos em vários níveis de detalhamento;
- A impossibilidade de modificar o ambiente em uma mesma sessão; e
- A inexistência de mecanismos de reutilização de objetos de um BD para a construção de mundos virtuais.

A utilização de um SGBD com suporte a XML para gerenciar os objetos virtuais será de extrema importância para a construção de um Ambiente Virtual Adaptativo que busque resolver os problemas acima.

1.2 OBJETIVOS

O objetivo deste trabalho de graduação é a especificação e implementação de um modelo de gerenciamento de objetos de RV reutilizáveis para ambientes virtuais de ensino, de maneira a facilitar o processo de criação de ambientes virtuais personalizados para cada usuário, considerando sua capacidade cognitiva, sua cultura digital, seu conhecimento com as ferramentas de aprendizagem e sua visão de mundo.

Assim, este trabalho pretende especificar todos os passos necessários para a criação, armazenamento e disponibilização de objetos componentes de mundos virtuais, de modo a permitir a reutilização desses objetos por diferentes sistemas que lancem mão de RV.

Serão discutidos testes realizados utilizando os recursos de manipulação de documentos XML do SGBDs Oracle 10g, de modo que se possa avaliar a quais as ferramentas disponíveis neste SGBD que poderão ser utilizadas para alcançar os objetivos deste trabalho. Será avaliada também a viabilidade de se utilizar o SGBD PostgreSQL 8.0 (software livre) para suportar esta tarefa.

Como resultado deste trabalho será construído um módulo de gerenciamento de objetos de realidade virtual reutilizáveis denominado MoGORViR. Como estudo para homologação do MoGORViR, será construído um sistema que permitirá a criação de mundos virtuais a partir de objetos armazenados em um SGBD. Estes objetos poderão ser visualizados em diferentes níveis de detalhamento simulando a existência de usuários com interesses distintos.

1.3 ESTRUTURA DO TRABALHO

Além deste capítulo introdutório contendo a contextualização, justificativa e objetivos, fazem parte deste trabalho ainda os capítulos descritos a seguir.

Capítulo 2 – Ambientes Virtuais de Ensino. Discute a importância destes ambientes e a contribuição da Realidade Virtual na melhorara de sua interatividade. Uma avaliação dos ambientes mostrados neste capítulo é realizada procurando discutir a sua importância para o desenvolvimento deste trabalho.

Capítulo 3 - Gerenciamento de Objetos de Realidade Virtual. Mostra as tecnologias mais usadas na construção de mundos virtuais, com destaque para as linguagens VRML e X3D (extensão de VRML com sintaxe XML). Também discute as vantagens da representação em XML e os recursos oferecidos por SGBD para manipular estes documentos.

Capítulo 4 – Módulo de Gerenciamento de Objetos de Realidade Virtual Reutilizáveis. Trata da especificação de requisitos, modelagem e desenvolvimento do módulo de gerenciamento de objetos de realidade virtual proposto neste trabalho. Mostra também a arquitetura geral de um sistema onde este componente poderá ser utilizado.

Capítulo 5 – Caso de Teste: Reutilização de Objetos Virtuais. Descreve a utilização dos recursos oferecidos pelo componente desenvolvido neste trabalho, através da construção de um sistema para a geração de mundos virtuais através da reutilização de mundos já existentes.

Capítulo 6 – Conclusão geral do trabalho, com sua contribuição e sugestões de trabalhos futuros.

CAPÍTULO 2

AMBIENTES VIRTUAIS DE ENSINO

Um Ambiente Virtual de Ensino (AVE) consiste no uso de recursos computacionais para permitir diversas formas de interação entre tutores e aprendizes, apoiado por conteúdos selecionados, com o objetivo de complementar ou substituir a aula presencial.

Os ambientes virtuais de ensino tiveram sua origem na segunda metade da década de noventa, quando os primeiros ambientes de educação baseada na Web foram desenvolvidos e utilizados em cursos a distância [Pequeno et al., 2004]. Segundo Lucena e Fuks (2000), o primeiro livro publicado sobre o assunto foi o Web-Based Instruction, de B. H. Khan, em 1997.

As ferramentas existentes nesses ambientes integram serviços de comunicações disponíveis na Internet - tais como o Correio Eletrônico e Salas de Discussão (Chat), com mecanismos de gerência de cursos e sistemas de envio de arquivos (upload). Todos estes recursos têm como elemento unificador as tecnologias utilizadas na Web, tais como HTML [HTML], CGI [CGI] e Java [Java, 2005].

O papel do professor neste ambiente passa a ser o de orientar e incentivar, não sendo mais um meio transmissor de conteúdo. A aprendizagem do aluno é realizada por meio da construção de conceitos e da interação com o professor, com colegas, com os recursos utilizados e com o conhecimento que está sendo estudado.

O uso de Realidade Virtual (RV) para a modelagem da interface de um ambiente virtual transforma a relação do usuário com o ambiente. O usuário deixa de ser apenas um expectador, passando a ter a possibilidade de imergir, navegar e interagir em um ambiente 3D, através de canais multi-sensoriais. A representação do conhecimento de uma forma mais próxima à realidade do usuário, somada ao aumento de interesse gerado pelo uso de uma nova tecnologia, permite que o conteúdo seja assimilado mais facilmente.

Desta maneira o usuário passa a “fazer parte” do mundo virtual, podendo visualizar, manipular e explorar o mundo em tempo real, através de seus sentidos,

aproveitando-se do conhecimento intuitivo a respeito do mundo físico para interagir no mundo virtual.

Com a integração das técnicas de Inteligência Artificial e Vida Artificial em ambientes virtuais tridimensionais, surgiram novas representações de ambientes que exploram o uso de entidades com certo grau de inteligência e diferentes formas de interações, provendo maior dinamicidade, realismo e usabilidade aos ambientes [Santos, 2004].

Segundo Aylett e Luck (2000) e Aylett e Cavazza (2001), os ambientes que utilizam tal interação são denominados Ambientes Virtuais Inteligentes (AVI). Um AVI é um ambiente virtual semelhante a um mundo real, habitado por entidades autônomas inteligentes (representações virtuais de formas de vida) exibindo uma variedade de comportamentos [Anastassakis et al., 2001].

Conforme Aylett e Luck [apud Osório et al., 2004], diversos fatores têm motivado a integração dessas técnicas. Primeiro, o aumento do poder computacional tem permitido não apenas a exploração de um alto grau de realismo visual, mas a adição de uma camada de inteligência aos ambientes. Segundo, a disponibilidade de bibliotecas e padrões gráficos 3D, tais como OpenGL [OpenGL, 2004], Java3D [Java3D, 2004], VRML [VRML, 1997] e, mais recentemente, X3D [X3D, 2004], tem promovido o desenvolvimento de ambientes 3D. Terceiro, as técnicas de Inteligência Artificial (IA), tais como as de agentes inteligentes e de processamento de linguagem natural, têm amadurecido em paralelo, podendo ser exploradas nas interações entre os usuários e o ambiente.

De acordo com Rickel et al. (2002), Gratch et al. (2002) e Anastassakis et al.(2001), as aplicações potenciais destes ambientes são consideráveis, podendo ser empregados em uma variedade de áreas, especialmente relacionadas com a simulação, o entretenimento e a educação [Osório et al., 2004].

Na seção 2.1 são descritos alguns exemplos de AVE, com o objetivo de conceituar esse tipo de ambiente e sua aplicação.

Na seção 2.2 são descritos alguns exemplos de AVI, com o objetivo de mostrar suas principais aplicações e funcionalidades.

2.1 ANÁLISE DE AVE EXISTENTES

Nos últimos anos foram desenvolvidos inúmeros AVE, principalmente para gerenciamento de cursos a distância. Nesta seção serão analisados alguns deles para que se possa exemplificar suas funcionalidades básicas e suas aplicações. Os sistemas que serão avaliados são o AulaNet [AulaNet, 2004], o WebCT [WebCT, 2004] e o Learning Space [LearningSpace, 2004]. Estas ferramentas de ensino foram escolhidas por apresentarem os recursos básicos existentes em um AVE e por terem sido produzidas por diferentes tipos de instituição: universidade (AulaNet – PUC Rio), universidade e depois empresa privada (WebCT – University of British Columbia e Universal Learning Tecnology) e empresa privada (Learning Space – Lotus Development Corporation).

2.1.1. AulaNet

O AulaNet é um ambiente para a administração, desenvolvimento, manutenção e assistência de cursos na Web, desenvolvido no Laboratório de Engenharia de Software (LES) do Departamento de Informática da PUC-Rio.

O AulaNet auxilia o docente na tarefa de disponibilizar o conteúdo de seu curso na internet. Segundo Fuks (2001), o conteúdo é separado da navegação, fazendo com que os docentes só se preocupem com a produção dos conteúdos didáticos usando suas ferramentas habituais, como o editor de textos, e deixem por conta do ambiente a gerência e a navegação dos aprendizes. Além disso, o AulaNet oferece ao docente uma variedade de serviços, que podem ser usados no curso de forma a complementá-lo.

O AulaNet está baseado nas seguintes premissas [Lucena et al., 1998]:

- Os cursos criados devem possuir grande capacidade de interatividade, de forma a atrair a participação intensa do aluno no processo de aprendizado;
- O autor do curso não precisa ser necessariamente um especialista em Internet;

- Os recursos oferecidos para a criação de cursos devem corresponder aos utilizados em uma sala de aula convencional, acrescidos de outros normalmente disponíveis no ambiente Web; e
- Deve ser possível a reutilização de conteúdos existentes em mídia digital, através, por exemplo, da importação de arquivos.

Os serviços do AulaNet são divididos baseados no princípio que para aprender em grupo um indivíduo tem que compartilhar idéias (se comunicar), estar em sintonia com os outros participantes do grupo (se coordenar) e realizar as tarefas satisfatoriamente (cooperar).

Os serviços de comunicação são os seguintes: correio eletrônico, lista de discussão, videoconferência, bate-papo (chat) e debates on-line pela rede. Todo contato com o professor é feito exclusivamente através de um endereço de correio eletrônico. As aulas ao vivo ocorrem em um horário fixo. A interação nas aulas ao vivo acontece através da troca de mensagens pelo correio eletrônico. Os debates ao vivo podem acontecer através de um programa de bate-papo ou de videoconferência. As listas de discussões permitem aos alunos colocarem mensagens que podem ser lidas em rede pelos demais alunos.

Os serviços de coordenação fornecem os meios para minimizar os problemas decorrentes do trabalho em grupo e maximizar a cooperação entre seus membros. O AulaNet fornece suporte para avisos, plano de aulas, tarefas (inclusive com suporte para upload dos arquivos com a resolução das tarefas), avaliações on-line e relatório de participação dos aprendizes.

Os serviços de cooperação permitem que todos os participantes contribuam para a construção do curso. O AulaNet permite a indicação de referências externas ao site do curso, a disponibilização de documentos, o download de documentos e a co-autoria do curso (mediante indicação do docente).

Durante a criação do curso o professor escolhe quais os serviços que ficarão disponíveis para os alunos. Nesta etapa ele pode estruturar o curso de acordo com o conteúdo a ser apresentado, das suas necessidades e do perfil dos alunos. A Figura 1 apresenta os recursos didáticos disponíveis durante a construção de um curso.

Os serviços são colocados à disposição do docente durante a criação e atualização de um curso, permitindo a ele selecionar quais vão se tornar serviços disponíveis aos aprendizes, configurando a área de trabalho do curso.

Figura 1: Recursos disponíveis para criação de curso no AulaNet

A Figura 2 mostra a área de trabalho do curso, ela é composta de uma janela principal e de um menu em formato de controle remoto. A janela principal é onde o aprendiz interage com os conteúdos didáticos, com o instrutor e com os demais aprendizes. O controle remoto é um menu de serviços que fornece uma facilidade de navegação construída através da seleção prévia feita pelo docente dos mecanismos de comunicação, coordenação e cooperação [Fuks, 2001].

Figura 2: Área de Trabalho do AulaNet

2.1.2. WebCT

O WebCT (World Wide Web Course Tool), que foi desenvolvido originalmente pela University of British Columbia, Vancouver no Canadá, em 1997, e adquirido, em 1999, pela empresa Universal Learning Technology (ULT), é um gerenciador de cursos a distância baseado em uma arquitetura cliente/servidor que utiliza a Internet como tecnologia de apoio à comunicação. Pode, também, ser utilizado simplesmente como repositório de material instrucional para cursos presenciais.

Os principais objetivos do WebCT são [Goldberg, 1996]:

- Ser fácil de usar e não requerer nenhum conhecimento técnico por parte do autor do curso ou por parte do aluno;
- Ser totalmente baseado na Web para o curso disponibilizado aos alunos e para a interface apresentada ao autor do curso; e
- Fornecer um conjunto de ferramentas para melhorar a experiência de aprendizado on-line.

O WebCT é um sistema baseado na Web, independente de plataforma e que não necessita de instalação de nenhum ferramenta na máquina do usuário. O sistema pode ser acessado de qualquer computador que tenha acesso à internet.

Ao criar um curso, o instrutor poderá utilizar as ferramentas disponíveis no servidor. Estas ferramentas permitem a comunicação e colaboração entre os participantes do curso, bem como a administração de diversos aspectos por parte do professor. Além disto, o professor também tem a possibilidade de configurar o ambiente através da escolha de cores e ícones que serão apresentados e da disposição dos elementos na interface. A Figura 3 mostra as ferramentas disponíveis para a criação de cursos do WebCT.

Figura 3: Ferramentas disponíveis para a criação de cusos do WebCT

O WebCT possui ferramentas para comunicação (fórum, correio eletrônico, chat), organização (calendário, indexação, pesquisa, quadro de avisos), cooperação entre estudantes (páginas pessoais), disponibilização de conteúdo (glossário, dicas, figuras, apresentações, cd-rom) e avaliação de desempenho (quiz, auto-teste, visualização de progresso).

O WebCT possui dois modos de visualização do curso. A primeira visão é linear, onde o usuário navega por um caminho usual através do conteúdo do curso. A segunda visão é hierárquica, onde o usuário visualiza as páginas do curso através de uma árvore. A Figura 4 mostra a tela principal de um Curso do WebCT.

Figura 4: Tela principal de um Curso do WebCT

Segundo Bac...(200), o WebCT não possibilita que o professor configure os outros recursos (mail, chat, entre outros) de acordo com as características de seu grupo de alunos. Sendo assim, o WebCT exige que a metodologia do curso se adapte aos recursos tecnológicos disponibilizados. Além disto, como cada usuário criado no WebCT está vinculado a um único curso, para que um aluno participe de mais de um curso seu cadastro precisa ser feito tantas vezes quantos forem os cursos do qual ele participará.

2.1.3. Learning Space

LearningSpace é um ambiente desenvolvido pela Lotus Development Corporation, uma subsidiária da IBM, para a criação e gerenciamento de cursos a distância, baseado no ambiente *groupware* Lotus Notes/Domino. Este ambiente possibilita que educadores, sem nenhuma habilidade em programação, desenvolvam e integrem nos seus cursos conteúdos multimídia [Lotus, 1998].

Segundo Bac... (2000), o Lotus Notes é baseado numa arquitetura cliente/servidor, onde usuários do servidor, conhecido como Domino, podem fazer acesso às bases de dados lá depositadas através de estações cliente ou ainda através de um Web browser. Este sistema de gerência de informações dá suporte ao

LearningSpace na geração de cursos pelos usuários, bem como na distribuição e manutenção de informações.

O Learning Space contém uma ferramenta de gerenciamento central e cinco módulos de banco de dados Lotus Notes altamente integrados. Os cinco módulos do Learning Space são os seguintes:

- Agenda – apresenta a estrutura do curso criada pelo gerente mostrando o que deve ser feito e quando. Na agenda são encontrados os objetivos do curso, exercícios, testes de reforço e avaliações. A Figura 5 mostra a tela principal do módulo de Agenda;
- Centro de Mídia – contém todos os conteúdos relacionados a um curso, bem como links para outras bases de dados do Lótus Notes ou para outras páginas da Web. Todo o material disponível pode ser consultado através de palavras-chave, título ou autor e pode ser acessado mediante download;
- Sala de Curso – é um ambiente interativo onde estudantes discutem com colegas ou instrutores sobre os temas relacionados ao curso. Estas discussões podem ser feitas através de mensagens de e-mail, videoconferência, chat e quadro-branco cooperativo. Um importante recurso é a possibilidade de gravar sessões de chat ou videoconferência para posterior consulta;
- Perfil dos participantes – é uma coleção de informações a respeito dos estudantes e instrutor(es) , como fotografia, dados pessoais, endereço, formação acadêmica, experiências e interesses. O estudante pode tornar algumas dessas informações disponíveis apenas para o instrutor; e
- Gerenciador de avaliação – permite que os instrutores verifiquem o rendimento de cada um dos participantes do curso. Os resultados de testes e tarefas são enviados automaticamente para o Gerenciador de avaliação.

O Learning Space oferece diversas ferramentas para a administração dos cursos. As ferramentas podem ser divididas nos seguintes grupos:

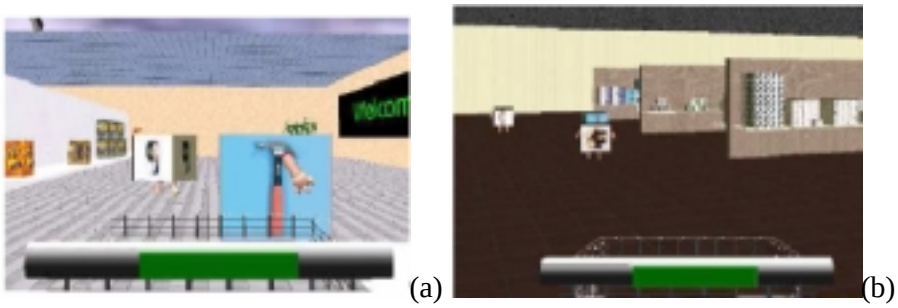
- Ferramentas de Gerenciamento do curso – permitem a criação e gerenciamento de recursos através de um grande número de Wizards. Permite a movimentação, cópia, compartilhamento e distribuição dos cursos existentes.;
- Ferramentas de Administração – permitem o controle de cursos e matrícula. Possibilita reiniciar, arquivar ou encerrar um curso já completado. Também existe a opção de criar cursos que não necessitam de matrícula; e
- Ferramentas de customização – permitem alterar a aparência e criar módulos personalizados para um determinado curso.

Figura 5: Tela principal do módulo de Agenda do Learning Space

2.2 ANÁLISE DE AVI EXISTENTES

Chittaro e Ranon (2002) utilizam um modelo do usuário na adaptação da estrutura e layout de uma loja virtual. Os usuários podem se deslocar e obter

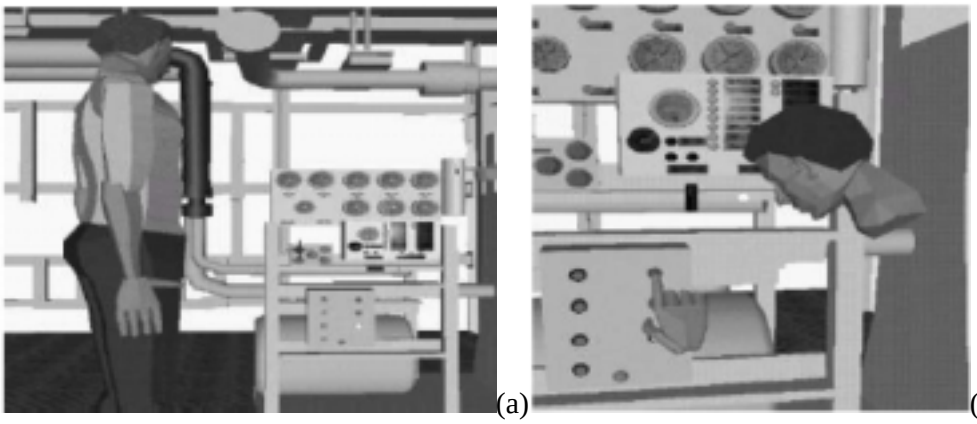
informações através de objetos que se deslocam pela loja. Inicialmente é criado um modelo de usuário com base em informações requisitadas através de um formulário, posteriormente este modelo é atualizado com base nas ações do usuário. As Figuras 6 (a) e (b) mostram a adaptação na loja virtual.



(a) (b)

Figura 6: Adaptação da loja 3D [Chittaro e Ranon, 2002]

Rickel and Johnson (1997) propõe a utilização de um agente virtual inteligente, STEVE (Soar Training Expert for VirtualEnvironment), para atuar como tutor em cursos de treinamento. STEVE auxilia o usuário no uso de um equipamento através de um ambiente tridimensional interativo. A Figura 7 (a) e (b) mostra o ambiente proposto.

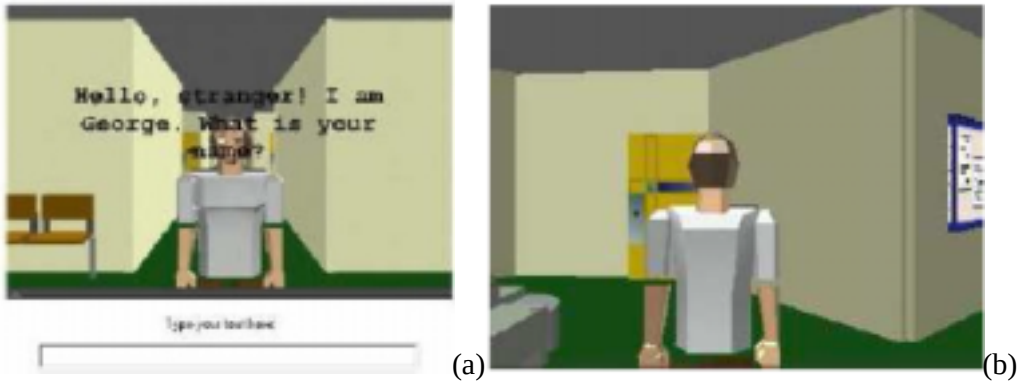


(a) (b)

Figura 7: STEVE ao lado de um equipamento e demonstrando seu uso [Rickel and Johnson, 1997]

Panayiotopoulos et al (1999) apresentam um guia virtual que conduz um usuário para locais relevantes em uma universidade virtual, bem como apresenta documentos multímidia, de acordo com os interesses do mesmo. O agente virtual se comunica com o usuário que indica suas necessidades através de linhas de comando.

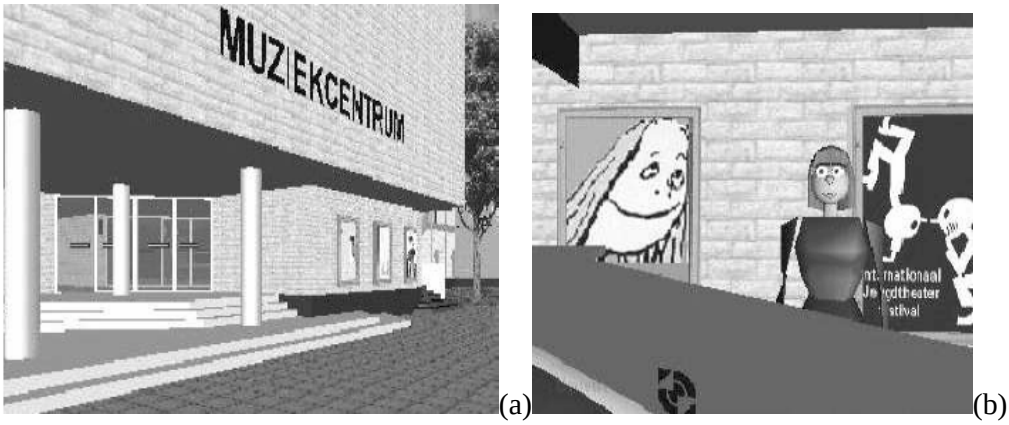
A Figura 8 (a) e (b) mostra a interface de entrada ao ambiente e a movimentação do agente.



(a) (b)

Figura 8: Interface de entrada da “Virtual University” e a movimentação do agente[Panayiotopoulos et al., 1999]

Nijholt e Hulstijn (2000) apresentam um teatro virtual, construído em VRML, onde o usuário pode navegar, entrando em salas de concerto e admirar quadros. O agente virtual fornece informações sobre shows e ingressos e se comunica através de linguagem natural. A Figura 9 (a) e (b) mostra o lado externo do teatro e a interação com o agente virtual.



(a) (b)

Figura 9: Lado externo do teatro e a interação com o agente virtual[Nijholt e Hulstijn, 2000]

Rizzo et al (2002) apresentam a utilização de um AVI para o tratamento de crianças hiper-ativas. Um avatar representa uma professora e objetos se movimentam pelo ambiente, a interação da criança com o ambiente é monitorada para se fazer uma análise de seu comportamento. A Figura 10 ilustra a sala virtual.

Figura 10: Sala de aula virtual [Rizzo et al., 2002]

Rizzo et al (2002) também mostram um escritório virtual onde objetos se movem e o ambiente é modificado para testar a capacidade cognitiva dos usuários. Avatares também pedem para os usuários realizarem determinadas tarefas no ambiente. A interação dos usuários é monitorada constantemente. A Figura 11 mostra várias configurações do ambiente.

Figura 11: Várias configurações do escritório virtual [Rizzo et al, 2002].

Uma biblioteca virtual, habitada por agentes, construída em VRML é apresentada por [Anastassakis et al, 2001]. O agente cliente requisita informações sobre livros ao agente bibliotecário, responsável por localizar e apresentar o item desejado. A Figura 12 apresenta a interface da biblioteca virtual.

Figura 12: Interação entre os agentes cliente e bibliotecário [Anastassakis et al, 2001]

2.3 CONSIDERAÇÕES FINAIS

Os Ambientes Virtuais de Ensino abordados na seção 2.1 possuem diversas ferramentas para auxiliar na autoria de cursos, gestão e comunicação. Apesar de terem o mesmo objetivo, cada ambiente cria cursos com sua própria estrutura e metodologia.

Estes ambientes foram desenvolvidos para facilitar o processo de aprendizagem do aluno. Buscando este objetivo, os ambientes permitem a criação de cursos que utilizam diversas ferramentas para promover a comunicação, a colaboração e a avaliação dos alunos, permitindo ainda o acesso a conteúdo em diversos formatos.

Apesar disto, a interação dos alunos com o material instrucional é muito pequena, limitando-se na maioria das vezes a leitura ou visualização do conteúdo. Além do mais, a maioria dos AVE atuais não permite que o ambiente se adapte às características de cada usuário e nem conseguem gerar ambientes interativos e dinâmicos, já que não possuem recursos avançados de interface.

Sendo assim, o uso de Ambientes Virtuais Inteligentes passa a ser uma alternativa muito interessante para a construção de cursos a distância, uma vez que permitem um alto grau de interatividade e permite modelar o mundo tridimensionalmente de uma forma mais próxima à realidade do usuário.

Os exemplos de AVI mostrados na seção 2.2 demonstraram a necessidade de se utilizar um mecanismo de interação com o usuário para auxiliá-lo durante a sessão. Para isso foram usados agentes inteligentes. Outra característica importante que se pôde verificar nos exemplos estudados, refere-se ao fato de que a maioria dos ambientes utiliza VRML e Java, demonstrando a importância destas linguagens na construção de mundos virtuais. Verifica-se também a preocupação de se manter um Perfil do Usuário para adaptar o ambiente em função das características do usuário. Estas modificações, porém, não são feitas instantaneamente, só sendo percebidas nas sessões subseqüentes.

Outra questão em aberto é a reutilização de mundos, ou parte deles, para compor novos ambientes. Isto seria importante, pois a reutilização de objetos e conteúdos proporcionaria a economia de tempo e recursos.

O módulo de gerenciamento de objetos de realidade virtual desenvolvido neste trabalho será de grande importância para a construção de um Ambiente Virtual Adaptativo que busque resolver os problemas acima.

CAPÍTULO 3

REPRESENTAÇÃO E ARMAZENAMENTO DE OBJETOS DE REALIDADE VIRTUAL

Este trabalho tem o objetivo de armazenar, recuperar e gerenciar objetos de Realidade Virtual. Para isto é necessário conhecer a forma como esses objetos são representados e os recursos oferecidos pelos SGBD para facilitar o seu gerenciamento.

Nos últimos anos muitos avanços foram realizados no desenvolvimento de ambientes virtuais 3D. As linguagens VRML [VRML, 1997] e Java3D [Java3D, 2004] têm sido as mais utilizadas para a construção destes ambientes por propiciarem o desenvolvimento de mundos virtuais mais detalhados e interativos via Web.

Recentemente, o desenvolvimento da linguagem X3D [X3D, 2004] abriu oportunidade para que diversos problemas relacionados com as linguagens tradicionais de representação de objetos 3D fossem resolvidos. Esta linguagem é uma extensão de VRML no formato XML. Isto possibilita a utilização das diversas tecnologias relacionadas a XML para manipular os mundos descritos em X3D.

Existem vários SGBD no mercado que trazem algum tipo de suporte para a manipulação de documentos XML. A extensão do suporte oferecido varia bastante entre os SGBD.

A seção 3.1 discute a estrutura e os recursos oferecidos pela linguagem VRML, que é considerada a principal ferramenta para a construção de ambientes virtuais.

A seção 3.2 mostra as diferenças existentes entre a linguagem X3D, que será utilizada para modelar os objetos virtuais neste trabalho e sua antecessora, a linguagem VRML.

A seção 3.3 discute os recursos oferecidos pela linguagem Java3D, que permite um maior controle sobre o ambiente virtual, e que pode ser utilizada para exibir mundos VRML previamente construídos.

A seção 3.4 discorre sobre as características da linguagem XML e as diversas tecnologias existentes para manipulá-la, com o objetivo de determinar a melhor abordagem para manipulação de objetos X3D.

Na seção 3.5 são apresentadas as funcionalidades mais comuns existentes em SGBD com suporte a XML e são detalhados os recursos oferecidos pelos SGBD Oracle 10g (um dos lídres de mercado) e PostgreSQL 8.0 (software livre).

3.1 A LINGUAGEM VRML

A linguagem VRML (Virtual Reality Modeling Language) [Ames et al, 1997] é utilizada para construir mundos virtuais tridimensionais na Web. Ela é usada para descrever objetos geométricos 3D e combiná-los em cenas ou mundos virtuais.

Da mesma forma que páginas Web são descritas em HTML (Hyper Text Mark-up Language) e interpretadas por um *browser* para Internet, os arquivos VRML são interpretados por *plug-ins* específicos acoplados a esses *browsers*. Esta flexibilização resulta na capacidade de produzir ambientes de interação usuário e máquina mais realistas [Hittner, 2003].

Algumas funcionalidades que a linguagem VRML oferece e que por isto a torna apreciada pelos desenvolvedores são: integração de modelos de dados 3D, 2D, textos e componentes multimídia; criação de ambientes virtuais 3D interativos e distribuídos; utilização em diversas plataformas de hardware dos ambientes de desenvolvimento e de visualização; integração com outros formatos disponíveis, tal como HTML, por exemplo; e simplicidade na edição do código [Aquino 2005].

Os componentes funcionais mais importantes para a formação de um arquivo VRML são (BOOK VRML, 2004):

- Estrutura de grafo de cena;
- Arquitetura de eventos;
- Sensores;
- Scripts e Interpoladores;
- DEF e USE;
- Prototipagem;
- Cenas Distribuídas;

3.1.1 Estrutura de grafo de cena

Arquivos VRML descrevem objetos e mundos 3D utilizando um grafo hierárquico da cena. As entidades no grafo da cena são conhecidas como nós. O VRML define 54 tipos diferentes de nós, incluindo, por exemplo, geometria primitiva, propriedades da aparência, som e vários tipos de nós de agrupamento. Os nós armazenam seus dados nos *fields* (campos), existindo 20 tipos diferentes que podem ser usados para armazenar os *values* (valores) necessários para descrever as propriedades da cena (BOOK VRML, 2004). O Quadro 1 mostra os tipos de nós existentes em VRML.

Quadro 1: Tipos de nós em VRML [Aquino 2005]

TIPO DO NÓ	NÓS
Nós de agrupamento	Anchor, Billboard, Collision, Group, Transform
Grupos especiais	Inline, LOD, Switch
Nós comuns	AudioClip, DirectionalLight, PointLight, Script, Shape, Sound, SpotLight, WorldInfo
Sensores	CylinderSensor, PlaneSensor, ProximitySensor, SphereSensor, TimeSensor, TouchSensor, VisibilitySensor
Geometria	Box, Cone, Cylinder, ElevationGrid, Extrusion, IndexedFaceSet, IndexedLineSet, PointSet, Sphere, Text
Propriedades geométricas	Color, Coordinate, Normal, TextureCoordinate, Appearance
Aparência	FontStyle, ImageTexture, Material, MovieTexture, PixelTexture, TextureTransform
Interpoladores	ColorInterpolator, CoordinateInterpolator, NormalInterpolator, OrientationInterpolator, PositionInterpolator, ScalarInterpolator
Nós "mutuamente inibidores" (somente uma instância destes nós pode estar ativa)	Background, Fog, NavigationInfo, Viewpoint

Uma característica importante de um grafo de cena é que os nós que aparecem antes afetam os posteriores. Na Figura 13, representando conteúdo de um arquivo VRML, por exemplo, o nó “Appearance” afeta a aparência do cubo definido no nó “Box”.

```
#VRML V2.0 utf8
# Exemplo de uma caixa
Shape {
    appearance Appearance {
      material Material {
         diffuseColor 1 0 0
      }
   }
   geometry Box {
      size 1 1 1
   }
}
```

Figura 13: Exemplo de Arquivo VRML

A referida Figura 13 apresenta um código VRML que representa um cubo(Box). O *header* (cabeçalho), apresentado na primeira linha, é obrigatório em qualquer arquivo VRML; ele identifica o arquivo como sendo VRML, sua versão, e o conjunto de caracteres internacional que será utilizado. O caracter # representa o início de um comentário, e pode ser aplicado em qualquer parte do arquivo. O nó *Shape* descreve a geometria da estrutura 3D do objeto e a sua aparência. A aparência (nó *appearance*) descreve propriedades relacionadas ao material e à textura. O nó material especifica as propriedades materiais da superfície; o campo *diffuseColor* determina o modo com que a luz reflete sobre a superfície dos objetos para criar a sua cor. Por último, na geometria do nó *Shape*, um cubo é descrito, onde são definidos o seu tipo (*Box*) e suas dimensões (*size*).

3.1.2 Arquitetura de Eventos

Alguns nós VRML geram eventos em resposta às mudanças ambientais ou à interação com o usuário. A distribuição de eventos fornece um mecanismo através do

qual estes eventos são propagados para efetuar mudanças em outros nós (VRML, 2004).

Cada tipo de nó define os nomes e os tipos de eventos que este pode gerar ou receber, além das indicações da rota, definindo os trajetos do evento entre geradores do evento e os seus receptores (BOOK VRML, 2004).

O processamento destes eventos pode mudar o estado do nó, gerar eventos adicionais, ou mudar a estrutura do grafo de cena. Todos os recursos de animação e interação são baseados na adição de *comportamentos* aos objetos da cena, possuindo eventos que o nó pode receber (*eventIn*) e enviar (*eventOut*).

3.1.3 Sensores

Sensores são nós que possuem a capacidade de gerar eventos respondendo a ações do usuário ou com o passar do tempo (VRML, 2004).

Os sensores permitem que o usuário interaja com objetos através de um clique, arrasto ou apenas se aproximando deles. É possível, por exemplo, utilizar um sensor para que acender uma luz ou abrir uma porta com a aproximação do usuário.

3.1.4 Scripts e Interpoladores

O nó Script permite utilizar rotinas escritas em JavaScript ou Java para realizar desde simples decisões lógicas até análises complexas de todos os eventos do ambiente. Ele é ativado quando recebe um evento.

O nó Script proporciona uma poderosa extensão a VRML com a qual é possível programar os sensores e interpoladores do ambiente [Ames et al, 1997].

Os Interpoladores permitem a construção de animações através da descrição do objeto em posições intermediárias da animação. O interpolador gera automaticamente a movimentação de uma posição a outra, possibilitando uma maneira eficiente de criar as animações.

3.1.5 DEF e USE

O uso dos atributos DEF e USE permite a reutilização de objetos dentro de um mesmo arquivo. Na primeira vez que o objeto é descrito é utilizado o atributo DEF para designar um nome para o objeto. Quando for necessário reutilizar o objeto

em outra parte do mundo basta utilizar o comando USE seguido do nome do arquivo. Por exemplo, para construir uma sala com dez computadores, basta declarar o primeiro computador e utilizar o comando USE para construir os outros.

3.1.6 Prototipagem

Em um arquivo VRML, o uso do mecanismo conhecido como prototipagem permite que o usuário possa criar os seus próprios tipos de nós, definindo dentro dele uma combinação dos tipos de nós existentes, que podem tornar o VRML mais fácil de ser usado e reduzir o tamanho dos arquivos (BOOK VRML, 2004).

3.1.7 Cenas Distribuídas

Uma grande vantagem do VRML é a possibilidade de construir o mundo virtual através de objetos pré-definidos em arquivos distintos. Isto favorece o reuso de objetos, facilitando a construção de novos mundos.

Existem dois modos de tornar isto possível, a utilização dos nós Inline e EXTERNPROTO. O nó Inline executa o arquivo VRML, indicado através da url, inserindo o seu conteúdo no mundo virtual. O nó EXTERNPROTO é usado para chamar um PROTO definido em outro arquivo.

3.2 A LINGUAGEM X3D

X3D [X3D, 2004] é um padrão XML para formatos de arquivo 3D, que permite comunicação em tempo real de dados tridimensionais através de qualquer aplicação, inclusive aplicações de rede. Ele tem um poderoso conjunto de características para uso em engenharia e visualização científica, CAD e arquitetura, visualização médica, treinamento e simulação, multimídia, entretenimento e educação, entre outros.

O desenvolvimento do X3D está a cargo da Web3D Consortium [Web3Da, 2004]. Sua especificação já está sendo avaliada pelo comitê da ISO/IEC e se espera que ela seja aprovada em ainda este ano.

Na criação do X3D, foi aproveitado o trabalho realizado pelo VRML97, utilizando as premissas básicas e estendendo-as para prover uma maior flexibilidade. As mudanças começaram pela especificação, que sofreu um completo ajuste, sendo

divididas em três partes: conceitos abstratos; formato de codificação para arquivos; e acesso à linguagem de programação (X3D, 2004). Além dessas, muitas outras modificações ocorreram, onde as principais foram: uma maior precisão com a iluminação e modelo de eventos; expansão das capacidades do grafo de cena; revisão e unificação do modelo da API; múltiplos formatos de codificação são aceitos: VRML, XML e o binário comprimido; e arquitetura modular, permitindo uma faixa de níveis para serem adotados e suportados por diversos tipos de aplicações ou uma plataforma em particular.

Os recursos de X3D são agrupados em componentes, definindo uma coleção específica de nós que possui um conjunto de funcionalidades. Um perfil é uma coleção de componentes para um nível de suporte específico. Todos os arquivos X3D requerem a definição do perfil que está em uso, na qual pode ser suprida com a requisição de componentes adicionais pelo usuário - ou por níveis maiores que aqueles providos pelo perfil [Web3Dc, 2004].

A Figura 14 apresenta a estrutura dos perfis em X3D conforme foi especificada pela Web3D. Cinco perfis foram criados - Core, Interchange, Interactive, Immersive e Full - com níveis de funcionalidade crescentes e incrementais.

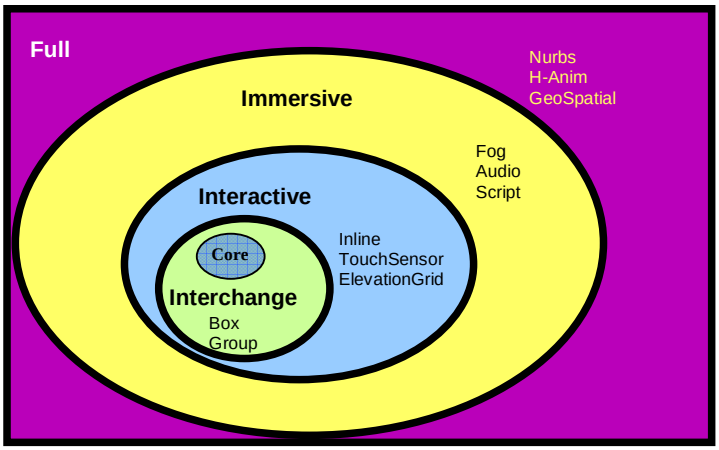


Figura 14: Estrutura dos perfis em X3D (adaptado de [Web3Dc, 2004])

A arquitetura modular foi empregada no X3D porque a maioria das aplicações não necessita de todos os seus recursos, tão pouco todas as plataformas dão suporte a todas as funcionalidades definidas na especificação.

Um arquivo X3D pode ser composto pelos mesmos componentes funcionais que existem em VRML, além de alguns componentes adicionais, que serão apresentados posteriormente. A Figura 15 mostra o mesmo cubo que havia sido apresentada na Figura 13, desta vez codificado como X3D.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN"
"http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D profile='Immersive'
xmlns:xsd='http://www.w3.org/2001/XMLSchemainstance'
xsd:noNamespaceSchemaLocation='http://www.web3d.org/specifications/x3d-
3.0.xsd'>
   <head>
      <meta name='filename' content='Exemplo.x3d'/>
      <meta name='description' content='Exemplo de arquivo X3D.'/>
      <meta name='created' content='Agosto 2005'/>
   </head>
   <Scene>
      <Shape>
         <Appearance>
            <Material diffuseColor='1 0 0'/>
         </Appearance>
         <Box size='1 1 1'/>
      </Shape>
   </Scene>
</X3D>
```

Figura 15: Exemplo de Arquivo X3D

O *header* de um arquivo X3D é composto por duas *tags*: a primeira identifica o arquivo como sendo XML, sua versão e o conjunto de caracteres internacional UTF-8; já na segunda, é declarado o DOCTYPE X3D que especifica o caminho da DTD de validação XML. Em seguida o nó raiz X3D é definido. Eele identifica o perfil X3D como *Immersive* e os Schemas XML de validação. Depois, um *header* opcional é criado. Ele pode conter componentes e informações. Neste caso, ele possui um nome, uma descrição e uma data de criação. Posteriormente, é definido um elemento *Scene*. Ele é um nó raiz que contém um ou mais elementos do grafo de cena. Nos elementos seguintes é bastante visível a semelhança entre os nomes dos elementos X3D em relação aos nós de VRML. O Quadro 2 mostra uma comparação entre as sintaxes das duas linguagens.

Quadro 2: Comparativo da sintaxe do VRML e X3D

VRML	X3D
Shape{ }	<Shape></Shape>
Appearance{ }	<Appearance></Appearance>
Material{ }	<Material />
Geometry tipo_geometria{ }	<tipo_geometria />
ROUTE <nome_nó>.<nome_evento> TO <nome_nó>.<nome_evento>	<ROUTE fromNode=' ' fromField=' ' toNode=' ' toField=' '/>
Script { url [“”] }	<Script url= “” />
DEF <nome_do_nó> <tipo_do_nó>{ <corpo> }	<tipo_de_nó DEF= “”> </tipo_de_nó>
USE <nome_do_nó>	<tipo_de_nó USE= “”/>
Transform { }	<Transform></Transform>
PROTO <nome_do_protótipo>[<declaração_da_interface>] { <definição> }	<ProtoDeclare name=' '> <ProtoInterface></ProtoInterface> <ProtoBody></ProtoBody> </ProtoDeclare>
EXTERNPROTO <nome_protótipo> [<declaração externa>] URL ou [URLs]	<ExternProtoDeclare name=' ' url='" " "http:// "'> <field name=' '/> </ExternProtoDeclare>

Além de X3D conter os mesmos componentes funcionais existentes em VRML, ele possui alguns adicionais, como é o caso do EXPORT e IMPORT. Esses comandos servem respectivamente para definir quais os elementos de um arquivo externo que poderão ser importados e para fazer com que o arquivo externo fique disponível para uso e possíveis distribuições de eventos.

3.3 A LINGUAGEM JAVA3D

A linguagem Java3D [Java3D, 2004] consiste em uma hierarquia de classes Java, para o desenvolvimento de sistemas gráficos tridimensionais, que permite a criação e manipulação de objetos geométricos através de construtores de alto nível. Com o uso de Java3D é possível construir mundos virtuais com muita flexibilidade, sendo as cenas representadas por grafos e a visualização gerenciada

automaticamente. A Figura 16 apresenta um exemplo de representação de um cubo em Java3D.

Por se tratar de uma linguagem de programação completa e não apenas uma linguagem de descrição, Java3D possui todas as funcionalidades inerentes à linguagem de programação de alto nível, particularmente das características de orientação a objetos da linguagem Java. Isto implica em definição de classes, herança, utilização de estruturas de controle e declaração, e muitos outros [Teichrieb, 1999].

```
Color3f blue;
blue = new Color3f(0.0f, 0.0f, 1.0f);
ColoringAttributes ca;
ca = new ColoringAttributes(blue,
ColoringAttributes.FASTEST);
Appearance a;
a = new Appearance();
a.setColoringAttributes(ca);
Box cube = new Box(2.0f, 2.0f, 2.0f, a);
```

Figura 16: Representação do objeto Cubo de cor azul em Java3D [Aquino 2005]

Java3D foi construída com o objetivo de criar uma API que fosse independente de plataforma e que tivesse um nível de desenvolvimento intermediário entre VRML e OpenGL[OpenGL 2004]/DirectX[Direct3D, 2004], não sendo tão simples quanto VRML e possuindo a flexibilidade inerente a uma linguagem de programação. Java3D permite tanto a construção de programas *standalon*e, quanto a construção de *Applets*, que podem ser executados em janelas de *Browsers* para Web.

Uma grande vantagem de Java3D é que ela permite a importação de mundos VRML, tornando possível, que investimentos feitos na construção de conteúdo sejam preservados.

Para funcionar, Java3D requer que se tenha instalado as API Java3D, Direct3D/OpenGL e JRE. Estes componentes somados têm um tamanho aproximado de 25 Mbytes. Isto pode tornar algumas aplicações Java3D inviáveis.

Espera-se que, no futuro, *browsers* convencionais (por exemplo, Netscape Navigator [Netscape, 2004] e Internet Explorer [Microsoft, 2004]), que já dão suporte à Java, possam incluir suporte à Java3D e permitir que os mundos

VRML e Java3D sejam carregados diretamente no *browser*, sem a necessidade do usuário se preocupar com tarefas como a instalação prévia de um *plug-in* [Aquino 2005].

3.4 TECNOLOGIAS PARA MANIPULAÇÃO DE DOCUMENTOS XML

XML é uma meta-linguagem de marcação desenvolvida pelo consórcio W3C (World Wide Web Consortium) [W3C, 2004], sendo considerada um padrão para a publicação e transferência de dados de diferentes domínios de aplicação na Web [XML,2003; Bradley, 2002].

XML utiliza marcadores (*tags*) para delimitar os dados, de forma similar a HTML. A diferença entre elas é que HTML foi projetada para descrever como controlar a exibição de dados, enquanto que XML foi projetada para descrever os dados, sendo as *tags* responsáveis por definir a semântica dos dados por ela circunscritos. Além disto, em XML as *tags* não são predefinidas, o autor deve construir suas próprias *tags* e montar a estrutura do documento desejada.

A Figura 17 mostra um exemplo simples de documento XML representando uma biblioteca.

```
<?xml version="1.0" encoding ="UTF-8"?>
<biblioteca>
  <livro ISBN=”16465”>
    <autor>
      Leonardo Sarmento
    </autor>
    <autor>
      Larissa Cabral
    </autor>
    <ano>
      2005
    </ano>
    <titulo>
        Contos de família
    </titulo>
    <editora>
        Livros para a juventude
    </editora>
  </livro>
  <livro ISBN=”54664”>
    <author>Chaudhri</author>
    <author>Rashid</author>
    <author>Zicari</author>
    <ano>
      1999
    </ano>
    <titulo>
```

```
        XML Data Management
    </titulo>
  </livro>
</biblioteca>
```

Figura 17: Exemplo de documento XML representando uma biblioteca

A linguagem XML está sendo utilizada para resolver diversos problemas tais como: integração com Web Services, redes distribuídas e aplicações para transferência de dados e arquivos; arquivamento, reuso e filtragem de uma grande quantidade de dados; integração baseada em páginas XML, facilitando o desenvolvimento de páginas Web e maximizando a interoperabilidade com outras linguagens da Internet; e suporte a um grande número de ferramentas, como stylesheets que permitem o trabalho em qualquer formato nativo XML [Aquino 2005] .

A especificação de XML não se restringe apenas à especificação de uma linguagem para descrição de dados. Existe uma série de tecnologias relacionadas que permitem a manipulação de documentos XML para diversos propósitos. As principais especificações serão abordadas a seguir.

3.4.1 DTD e XML Schema

O esquema representando a organização dos elementos que podem ser definidos em determinado arquivo XML pode ser descrito de duas formas: através do uso de um DTD (Document Type Definition) [XML, 2004] ou de um XML Schema [XSD, 2003]. Ambos descrevem os tipos de elementos que um documento deve conter, bem como a ordem que devem aparecer e restrições sobre o conteúdo. Um documento XML é dito válido se ele segue as normas definidas em um arquivo DTD ou XML Schema determinado.

A utilização de esquemas garante que os dados estão corretos antes de serem utilizados por outras aplicações e que o formato especificado foi seguido.

DTD foi a primeira recomendação da W3C para a especificação de esquemas de documentos XML. Uma DTD descreve o esquema em esquema basicamente com o uso de cláusulas ELEMENT para declarar elementos e ATTLIST para declarar a lista de atributos de um elemento.

A Figura 18 mostra a DTD usada para validar o documento XML mostrado na Figura 17. Nesta figura pode-se ver que o elemento básico do documento é uma biblioteca que é formada por um ou mais livros (símbolo +). Os livros, por sua vez, têm um ou mais autores, um ano, um título e podem ter ou não uma editora (símbolo ?). Os elementos autor, ano, titulo e editora têm uma estrutura simples formada apenas por texto.

```
<!ELEMENT biblioteca (livro+)>
<!ELEMENT livro (autor+, ano, titulo, editora?)>
<!ATTLIST livro ISBN CDATA #REQUIRED>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT ano (#PCDATA)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT editora (#PCDATA)>
```

Figura 18: Exemplo de esquema para documento XML definido através de DTD

A recomendação da W3C para descrever a estrutura de documentos XML é a XML SCHEMA. XML Schema é um padrão mais abrangente que o DTD, pois dá suporte a um conjunto maior de tipos de dados primitivos; permite definir novos tipos de dados; dá suporte a herança; oferece suporte a *namespaces* e utiliza sintaxe XML. A Figura 19 descreve o esquema do documento mostrado na Figura 17 através do uso de XML SCHEMA.

```
<?xml version =“1.0” encoding =“UTF-8” ?>
<Schema xmlns="http://www.w3.org/2001/XMLSchema.xsd">
 <element name=”biblioteca”>
  <complexType>
    <element ref=”livro” minOccurs=”1”/>
  </complexType>
 </element>
 <element name=”livro”>
   <complexType>
     <sequence>
        <element name=”autor” minOccurs=”1”/>
        <element name=”ano” type=”integer”/>
        <element name=titulo” type=”string”/>
        <element name=”editora” type=”string” minOccurs=”0”/>
     </sequence>
   </complexType>
   <attribute name=”ISBN” type=”integer”/>
 </element>
</Schema>
```

Figura 19: Exemplo de esquema para documento XML definido através de um XML SCHEMA

Apesar XML SCHEMA trazer mais recursos para a especificação de esquemas de documentos XML, DTD continua sendo amplamente utilizado. A maioria das ferramentas existentes ainda utiliza DTD para fazer a validação de documentos XML. Espera-se que no futuro a utilização de XML SCHEMA vá aos poucos substituindo a de DTD.

3.4.2 XPath e XQuery

XPath [XPA, 2003] foi a primeira linguagem de consulta a dados XML recomendada pela W3C. Ela utiliza expressões de caminho para selecionar fragmentos de um documento XML. Estas expressões se assemelham às expressões utilizadas para se navegar em sistemas de arquivo tradicionais como Unix ou DOS.

A Figura 20 exemplifica algumas consultas que podem ser feitas utilizando XPath. A consulta (a) retorna todos os elementos livro contidos no elemento biblioteca. A consulta (b) retorna todos os elementos titulo que são filhos do elemento biblioteca. A consulta (c) retorna o valor de todos os atributos ISBN em qualquer parte do documento. A consulta (d) retorna os elementos autor do livro cujo atributo ISBN é igual a “16465”.

```
(a) /biblioteca/livro
(b) /biblioteca//titulo
(c) //@ISBN
(d)/biblioteca/livro[@ISBN=”16465”]/autor
```

Figura 20: Exemplos de consultas XPath

Apesar de permitir diversos tipos de consulta, a linguagem XPath tem algumas limitações. Ela apenas retorna fragmentos do documento XML, não sendo capaz de gerar estruturas diferentes das existentes no documento. Ela também não permite a realização de junção entre dados XML.

A linguagem XQuery [XQU, 2003]. é a mais recente recomendação da W3C para realizar consultas em dados XML. Ela corrige as deficiências encontradas na linguagem XPath.

XQuery utiliza uma expressão FLWR (pronunciada “flower”), que é construída a partir de cláusulas FOR, LET,WHERE e RETURN, para realizar suas

consultas. Assim como em uma consulta SQL, estas cláusulas devem aparecer em uma ordem específica. Uma expressão FLWR associa valores a uma ou mais variáveis e utiliza esses valores para construir um resultado. Suas expressões de busca utilizam a sintaxe de XPath. Atualmente os mais importantes fabricantes de SGBD comercias, tais como IBM, Oracle e Microsoft, já dão suporte ao uso de XQuery.

A Figura 21 mostra o exemplo de uma consulta XQuery e o valor retornado, considerando que o documento XML da Figura 17 é usado na consulta.

```
Consulta:
<leonardo>
    <livros>
FOR $b in doc(“biblioteca.xml”)//livro
WHERE some $a in $b/autor satisfies $a=“Leonardo Sarmento”
RETURN $b/titulo
    </livros>
</leonardo>

Resultado:
<leonardo>
    <livros>
    <titulo>
      Contos de família
    </titulo>
  </livros>
</leonardo>
```

Figura 21: Exemplo de consulta XQuery e seu resultado

3.4.3 DOM

DOM (Document Object Model) [DOM, 2003] é uma tecnologia que permite o processamento de dados XML através de aplicações. DOM é um padrão desenvolvido pela W3C. DOM define um modo padrão para possibilitar o acesso e a manipulação de documentos XML. DOM apresenta um documento XML como uma estrutura de árvore e permite o acesso a esta estrutura através de um conjunto de objetos. Alguns *browsers* Web são capazes de processar instruções DOM definidas em scripts HTML.

A API JDOM [JDOM, 2004], que é similar a DOM, foi desenvolvida com o intuito de tornar a manipulação de documentos XML por Java uma tarefa o mais simples possível.

JDOM representa um arquivo XML como uma árvore composta de nodos de elementos, atributos, comentários, instruções de métodos, nodos textos, seções CDATA, e assim por diante. A árvore toda é analisada ao mesmo tempo. JDOM pode acessar qualquer parte da árvore a qualquer momento. Todos os tipos diferentes de nodos em uma árvore são representados por classes concretas ao invés de interfaces. Além disso, JDOM pode usar as convenções de código e as bibliotecas de classes existentes na linguagem Java [Aquino 2005] .

A Figura 22 mostra o exemplo de um trecho de código em JDOM para modificar o nome da editora do livro cujo atributo ISBN é igual a “16465”.

```
SAXBuilder builder = new SAXBuilder();
Document doc = builder.build(“biblioteca.xml”);

x = XPath.newInstance(“/biblioteca/livro[@ISBN=’16465’/editora]”);
Element editora = (Element)x.selectSingleNode(doc);

editora.setText(“Nova Virtual”);
```

Figura 22: Exemplo de código da API JDOM

3.5 GERENCIAMENTOS DE DADOS XML POR SGBD

Os protocolos XML são amplamente utilizados em aplicações desenvolvidas para a Web ou que usam a Web para a transferência de dados. Dados XML necessitam ser mantidos em um Banco de Dados (BD) e, consequentemente, existe a necessidade de se gerenciar adequadamente estes dados. Novos desafios surgem em decorrência desta necessidade, uma vez que a tecnologia de BD precisa ser adaptada para garantir o armazenamento e manipulação de dados XML de forma eficiente [Mello, 2002].

Um documento XML pode ser considerado uma coleção de dados, da mesma forma que um documento em um BD. Porém, um dado convencional de BD é totalmente estruturado enquanto um dado XML é um dado semi-estruturado [Abiteboul et al, 2000].

Um SGBD XML define um modelo lógico específico para documentos XML, armazenando e recuperando dados de acordo com este modelo [Bourret, 2003; XML, 2003b; XDB, 2003]. A definição de um SGBD XML não impõe nenhuma restrição sobre o modelo de armazenamento físico a ser adotado.

Apesar de alguns produtos de SGBD XML disponíveis no mercado não serem totalmente homogêneos em termos de gerenciamento de dados, algumas características já são consideradas de consenso, sendo encontradas atualmente nestes produtos, ou sendo características que se desejam alcançar no futuro [XML, 2003b; Bourret, 2003].

Nas seções seguintes serão mostrados os recursos disponíveis para gerenciamento de documentos XML presentes nos SGBD Oracle 10g [Oracle, 2005] e PostgreSQL 8.0 [PostgreSQL, 2005]. Foram escolhidos estes SGBD porque o Oracle 10g é um produto comercial com vários recursos disponíveis e o PostgreSQL é um dos poucos SGBD open source com suporte a gerenciamento de documentos XML.

3.5.1 SUPORTE a XML no ORACLE 10g

Oracle XML DB é o termo utilizado para descrever a tecnologia presente no Oracle Database 10g que permite uma alta performance no armazenamento e recuperação de XML. Esta tecnologia estende a popular Oracle relational database, oferecendo todas as funcionalidades associadas com um Banco de Dados XML nativo [Oracle, 2005].

A Figura 23 mostra os esquemas de armazenamento de documentos XML do Oracle. Ele permite que seja associado, opcionalmente, um XML SCHEMA para validação dos documentos que serão adicionados à uma determinada tabela do SGBD. A utilização do XML SCHEMA apesar de trazer um pequeno Overhead no armazenamento, permite que o Oracle realize diversas otimizações no acesso através de seus recursos de armazenamento objeto – relacional.

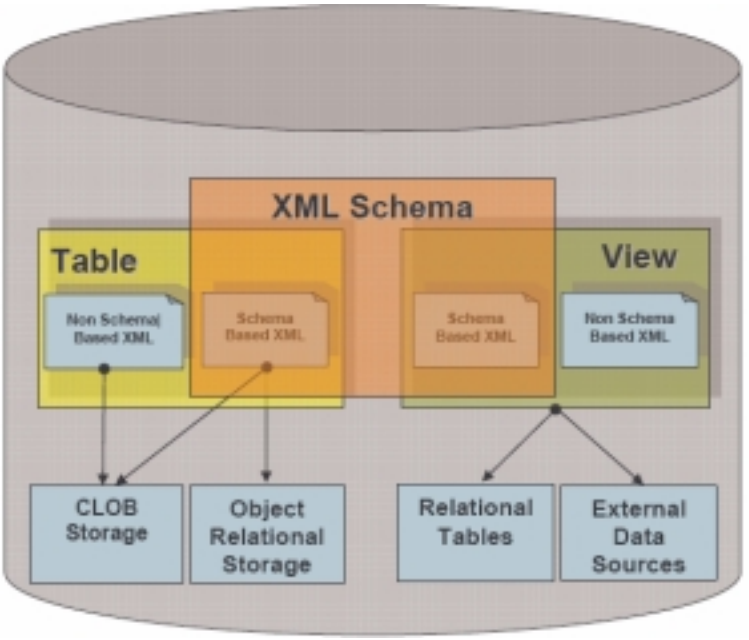


Figura 23: Esquema de armazenamento de documentos XML do Oracle 10g [Oracle, 2005]

O Oracle permite a utilização tanto da linguagem de consulta XPath, quando da linguagem XQuery. Ele permite ainda a utilização de diversas interfaces de acesso ao SGBD como mostra a Figura 24.

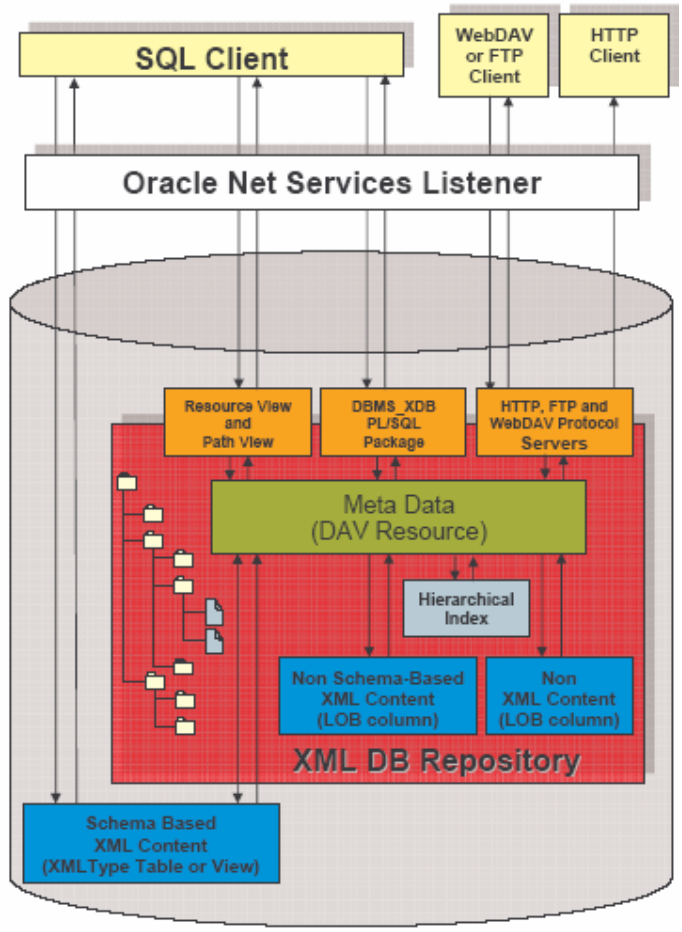


Figura 24: Mostra as diversas interfaces disponíveis para acessar o repositório de documentos XML. [Oracle, 2005]

3.5.2 SUPORTE a XML no POSTGRESQL 8.0

O suporte fornecido pelo PostgreSQL para XML é através de funções que manipulam dados XML. Diferentemente do Oracle, ele não fornece nenhum tipo especial de dado que possa ser usado para armazenar documentos XML. Os documentos são armazenados em campos texto, podendo ser acessados diretamente por qualquer comando SQL.

As funções definidas podem ser utilizadas como qualquer outra função definida pelo PostgreSQL, podendo ser utilizadas em cláusulas SELECT, FROM e WHERE.

O PostgreSQL tem uma série de funções que utilizam a linguagem XPath para acessar determinados fragmentos de documentos armazenados no banco de dados. Estas funções são explicadas a seguir:

- xpath_string(identificador_coluna, expressaoXPath): usada principalmente na cláusula SELECT para delimitar os dados que serão retornados;
- xpath_boolean(identificador_coluna, expressaoXPath): usada principalmente na cláusula WHERE para definir que condições os dados pesquisados devem obedecer;
- xpath_number(identificador_coluna, expressaoXPath): retorna um número, o que pode ser útil para fazer comparações numéricas;
- xpath_nodeset(identificador_coluna, expressaoXPath): usada para retorno o resultado da consulta em apenas uma linha da tabela; e
- xpath_table(key, documento, relação, xpaths, critérios): usada para construir uma tabela virtual.

O PostgreSQL contém também uma função para validar um documento XML com um DTD. Este recurso pode ser utilizado para garantir que todos os documentos armazenados em uma determinada tabela seguem o padrão requerido.

3.5.3 Comparativo entre o Oracle 10g e o PostgreSQL 8.0

Nas seções anteriores foram mostradas as funcionalidades presentes nos SQGB Oracle 10g e PostgreSQL 8.0 para a manipulação de documentos XML. O Quadro 3 mostra um comparativo destas funcionalidade com base nas características que um SGBD com suporte a XML deve fornecer segundo Bourret (2003).

Com base neste quadro comparativo fica claro que o SGBD Oracle 10g fornece muito mais recursos para a manipulação de documentos XML. Este SGBD, porém, por ser um produto comercial de alto custo, impossibilita que pequenas aplicações possam utilizá-lo. Para estas aplicações o PostgreSQL 8.0 torna-se uma opção interessante uma vez que suporta os recursos básicos necessários para a manipulação de documentos XML e não apresenta qualquer acréscimo nos custos de desenvolvimento e distribuição do software.

Quadro 3: Comparativo de Funcionalidades entre o Oracle 10g e o PostgreSQL 8.0

SGBD / **Funcionalidade**	**Oracle 10g**	**PostgreSQL 8.0**
Coleções	Armazenamento como uma coluna do tipo XMLType de uma tabela relacional ou como objeto em uma tabela XMLType. Permite definir se os dados serão validados por XDS ou não. A associação de XDS permite que sejam feitas otimizações de acesso.	Armazenamento como uma coluna de uma tabela relacional. Permite a criação de novos tipos, o que pode ser usado para forçar a validação por DTD.
Consultas	Permite o uso de XPath, XQuery e SQL.	Permite o uso de XPath e SQL.
Atualizações	Permite que sejam feitas atualizações parciais caso o documento esteja relacionado a um XML Schema	Permite atualizar apenas o documento por inteiro.
Gerenciamento de Transações	Bloqueio ao nível de documento completo.	Bloqueio ao nível de documento completo.
API	HTTP, WebDAV, FTP, PL/SQL e JDBC.	PL/SQL, JDBC.

CAPÍTULO 4

MÓDULO DE GERENCIAMENTO DE OBJETOS DE REALIDADE VIRTUAL REUTILIZÁVEIS

Neste capítulo serão detalhados os requisitos, a modelagem e o desenvolvimento do Módulo de Gerenciamento de Objetos de Realidade Virtual Reutilizáveis (MoGORViR), bem como as decisões de projeto e as limitações do protótipo.

O MoGORViR foi projetado para servir como a base para a construção de um Ambiente Virtual Adaptativo. Ele será utilizado na construção do sistema AVPersonal que foi apresentado em (Aquino 2005) e que se encontra em fase de desenvolvimento.

A seção 4.1 descreve a arquitetura do sistema AVPersonal, enfatizando as responsabilidades do MoGORViR e os requisitos necessários para atender a suas necessidades.

A seção 4.2 descreve as decisões de projeto que nortearam a modelagem do MoGORViR.

A seção 4.3 descreve o diagrama de casos de uso e a arquitetura escolhida para o desenvolvimento do projeto.

A seção 4.4 discute as dificuldades encontradas na implementação e os resultados alcançados.

4.1 ARQUITETURA DO AVPERSONAL

Segundo Aquino (2005), o objetivo geral do AVPersonal é desenvolver uma infra-estrutura para gerenciamento de mundos virtuais, cuja adaptatividade seja realizada em tempo real. Estes mundos devem ser construídos com componentes reutilizáveis, armazenados em um SGBD XML, onde cada componente possa conter vários níveis de informação.

À medida que o usuário interage com o ambiente, o *user level* pode ser modificado caso seja identificada uma evolução do seu nível de conhecimento. De acordo com o *user level*, o sistema determina se novos objetos devem ser inseridos no ambiente virtual e/ou se eles devem ser atualizados. Desta forma, o sistema pode se adaptar ao usuário durante a sua interação com o ambiente [Aquino, 2005].

A Figura 25 mostra a arquitetura do AVPersonal. O modelo do usuário (MU) guarda a representação do conhecimento do usuário em relação ao ambiente. O modelo do ambiente (MA) mantém informações relativas às mudanças do ambiente. Estas informações são obtidas através de sensores.

Uma sociedade de agentes é responsável pela análise dessas informações (contidas no MU e MA) e pela atualização dos objetos do mundo. Para tanto, quatro agentes são necessários: o Agente Pessoal que acompanha as ações do usuário e atualiza o MU sempre que for detectada alguma alteração no seu perfil; o Agente Ambiente que verifica o estado atual do ambiente virtual e as modificações ocorridas nele, armazenando-as no MA; o Agente de Comunicação que esclarece as dúvidas do usuário e lhe fornece informações sobre o ambiente; e, por fim, o Agente Atualizador, que gerencia o processo de atualização do mundo em função das informações fornecidas pelos outros agentes [Aquino, 2005].

Além disso, durante o processo de tomada de decisão, o Agente Atualizador consulta a Ontologia do Domínio, que possui a representação do modelo do AV, e de acordo com essa representação, gera consultas ao SGBD para recuperar os objetos que deverão ser utilizados na atualização do mundo [Aquino, 2005].

Um SGBD com suporte a XML (ver seção 3.5.2) armazena todos os objetos definidos pela ontologia. A recuperação de tais objetos é realizada através de consultas XML, tendo em vista esses objetos serem especificados em X3D (ver seção 3.2)..

O Gerenciador de Objetos é responsável pela conversão dos arquivos em X3D para VRML, permitindo a visualização do mundo virtual por um número maior de aplicações.

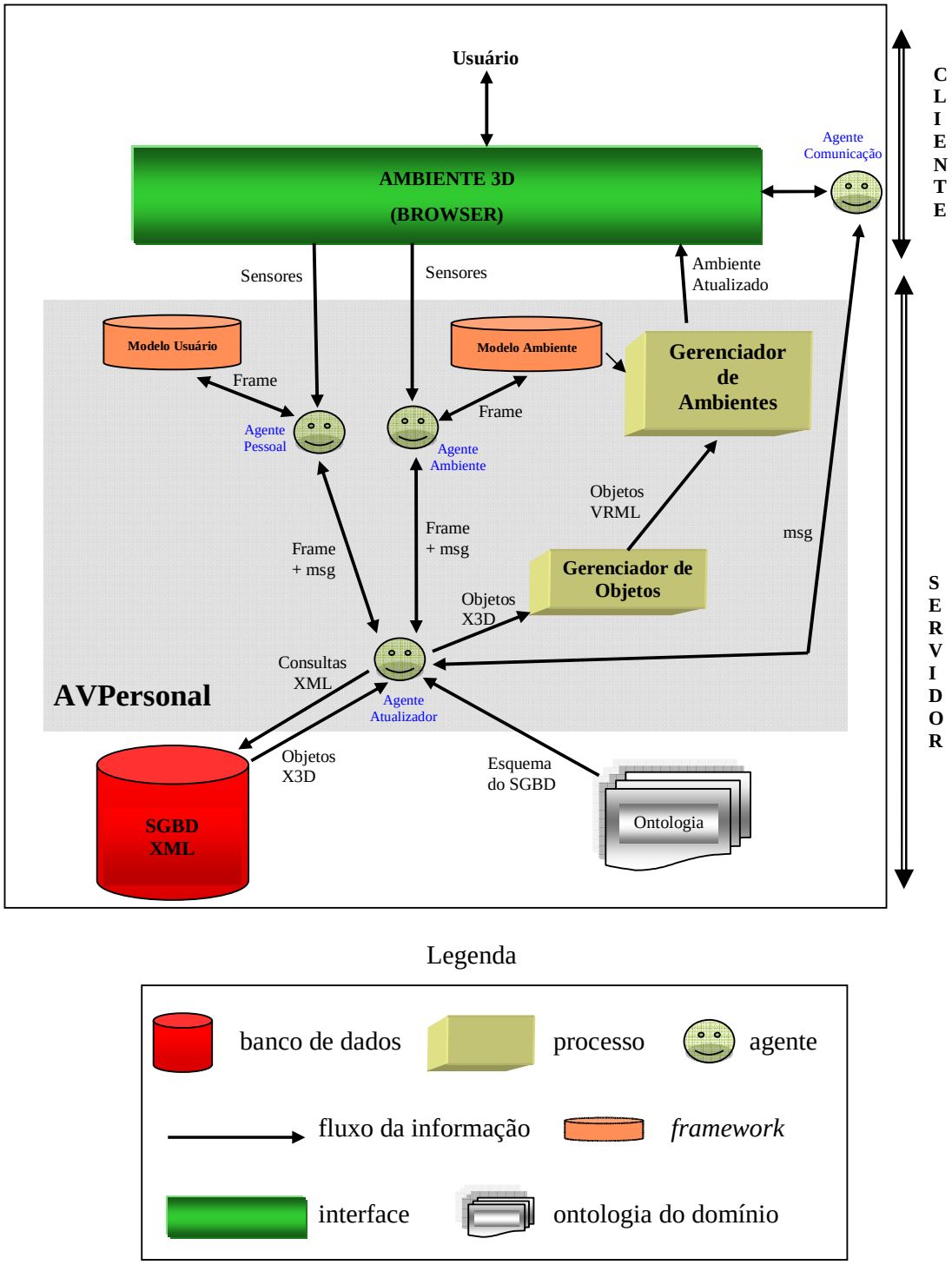


Figura 25: Arquitetura do AVPersonal [Aquino, 2005]

O papel do MoGORViR nesta arquitetura é prover uma camada de abstração entre o Agente Atualizador e o SGBD XML, através da disponibilização de serviços

de armazenamento e recuperação de objetos de realidade virtual. O MoRGORViR deve ser capaz de filtrar os elementos que são pertinentes ao usuário considerando seu atributo”userLevel”.

A seção 4.2 discutirá as decisões de projeto que foram tomadas no desenvolvimento deste módulo.

A seção 4.3 discute a arquitetura do sistema, dando ênfase aos mecanismos utilizados para armazenar e recuperar os dados.

4.2 DECISÕES DE PROJETO

A primeira decisão que teve que ser feita se referiu ao SGBD utilizado. A seção 3.5 deste trabalho descreveu os recursos de manipulação de documentos XML oferecidos por dois SGBD, o Oracle 10g e o PostgreSQL.

Devido à pouca disponibilidade de tempo e falta de experiência do aluno em relação a este assunto, decidiu-se que o MoGORViR seria desenvolvido primeiramente em apenas um SGBD. O SGBD escolhido foi o Oracle 10g que, como foi mostrado no comparativo da seção 3.5.3, oferece mais recursos para a manipulação de documentos XML.

Apesar disto, durante modelagem do módulo foi dispensado um cuidado especial no sentido de que a solução escolhida pudesse ser adaptada para o PostgreSQL sem maiores dificuldades.

Para realizar as consultas ao SGBD foi escolhida a linguagem XPath. Apesar do Oracle também oferecer suporte à linguagem de consulta XQuery, que é mais expressiva, este recurso só veio a ser disponibilizado na versão Oracle 10g R2, lançada há pouco menos de um mês, o que impossibilitou a sua adoção para a realização deste trabalho.

O armazenamento dos documentos XML está sendo feito em uma tabela do tipo XMLType. Foi associada a esta tabela uma versão modificada do XML SCHEMA da linguagem X3D acrescido do atributo “userLevel”. Isto garante que o SGBD somente irá armazenar objetos que sigam o padrão X3D, além de permitir que o Oracle realize diversas otimizações de acesso e manipulação dos dados XML.

4.3 MODELAGEM DO PROTÓTIPO

A Figura 26 mostra o Diagrama de Casos de Uso do MoGORViR. Observe-se que o único usuário dos serviços do MoGORViR é o Agente Atualizador. Os serviços disponibilizados são os seguintes:

- Recuperar Objeto – Utilizado para recuperar objetos do SGBD.
- Listar Objetos – Permite ter uma visão de todos os objetos armazenados no SGBD. Pode ser usado como um passo intermediário para a recuperação de um objeto; e
- Inserir Objeto – Permite armazenar um objeto X3D no SGBD.
- Recuperar Fragmento – Permite recuperar um fragmento de um objeto X3D definido pelo atributo "DEF".
- Listar Fragmentos – Lista todos os fragmentos de objeto existentes no SGBD ou em um determinado objeto.
- Adaptar Objeto – Permite adaptar um objeto ou fragmento de objeto ao perfil do usuário, através do atributo "userLevel".

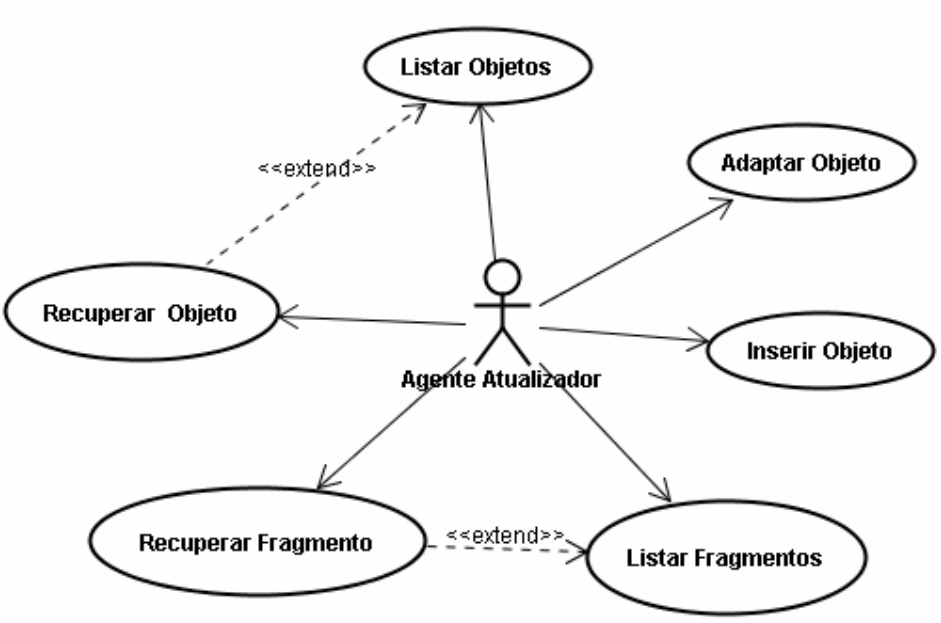


Figura 26: Diagrama de Casos de Uso do MoGORViR

A Figura 27 mostra o Diagrama de Classes do MoGORViR. A classe “RepositórioX3dFactory” é a responsável pela criação dos repositórios específicos para cada SGBD. Atualmente só existe o repositório para utilização com o Oracle, mas é possível com pequenas mudanças estender a aplicação para diferentes SGBD.

A classe “AdaptadorObjetoUsuario” é responsável por adaptar os mundos virtuais ao perfil do usuário, podendo acrescentar ou retirar partes do mundo.

A classe “RepositorioObjetosX3D” é uma interface através da qual os outros módulos do sistema entraram em contato com o MoGORViR. Ela determina métodos para inserir objetos X3D, buscar objetos X3D, listar todos os objetos X3D, listar fragmentos destes objetos e recuperar fragmentos de objetos.

A classe “RepositorioObjetosX3DOracle” é a implementação da classe “RepositorioObjetosX3D” para o SGBD Oracle.

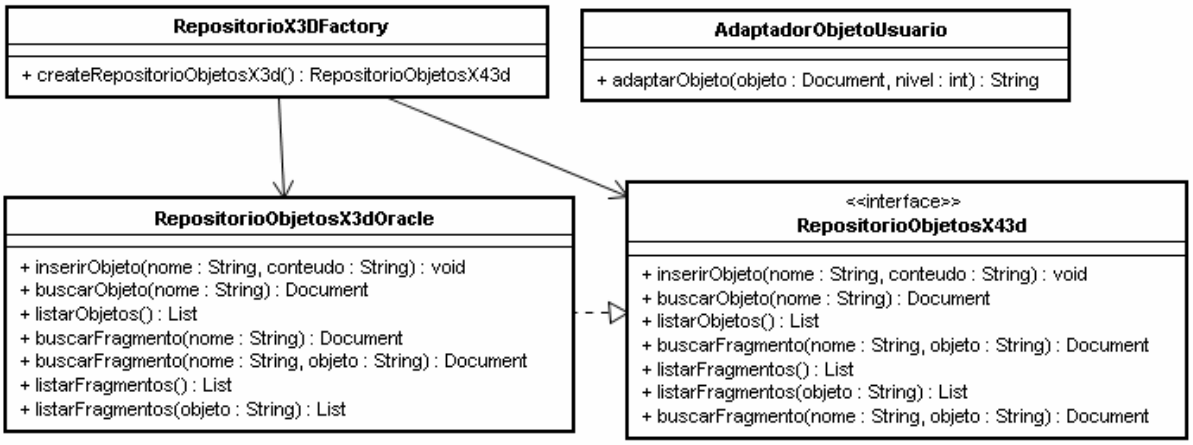


Figura 27: Diagrama de classes do MoGORViR

Maiores informações sobre a construção deste módulo podem ser encontrados no Anexo A, que contém todo o código fonte do MoGORViR.

CAPÍTULO 5

CASO DE TESTE: REUTILIZAÇÃO DE OBJETOS VIRTUAIS

Este estudo tem como objetivo avaliar se o MoGORViR poderá atender os requisitos necessários para servir como base para um Ambiente Virtual Adaptativo. Os principais problemas existentes com os sistemas atuais se referem à inexistência de adaptação relativa à apresentação de objetos em vários níveis de detalhamento, a impossibilidade de modificar o ambiente em uma mesma sessão e a inexistência de mecanismos de reutilização de objetos de um BD para a construção de mundos virtuais.

Como o sistema onde este módulo deverá ser integrado ainda não foi desenvolvido, será construída, neste caso de teste, uma aplicação que testará a capacidade do MoGORViR de suprir as carências citadas acima. Esta aplicação simulará as necessidades do Ambiente AVPERSONAL, descritas no capítulo anterior.

A aplicação consiste em um sistema para a criação de mundos virtuais a partir de objetos armazenados em SGBD. Estes objetos poderão ser visualizados em diferentes níveis de detalhamento simulando a existência de usuários com interesses distintos.

Através de uma interface Web, o usuário terá acesso à seção de cadastro, onde será possível alimentar o sistema com objetos X3D. Esta carga poderá ser feita de acordo com a conveniência do usuário, já sendo possível o acesso à seção de criação de mundos virtuais personalizados após o armazenamento dos primeiros objetos.

A Figura 28 mostra o fluxo de navegação da seção de criação de mundos virtuais personalizados. Na primeira página o usuário visualizará os ambientes cadastrados no SGBD e poderá escolher um deles para servir como base para a construção de seu ambiente personalizado. Para auxiliar esta decisão, são disponibilizadas as opções de visualizar um determinado mundo e seu código fonte.

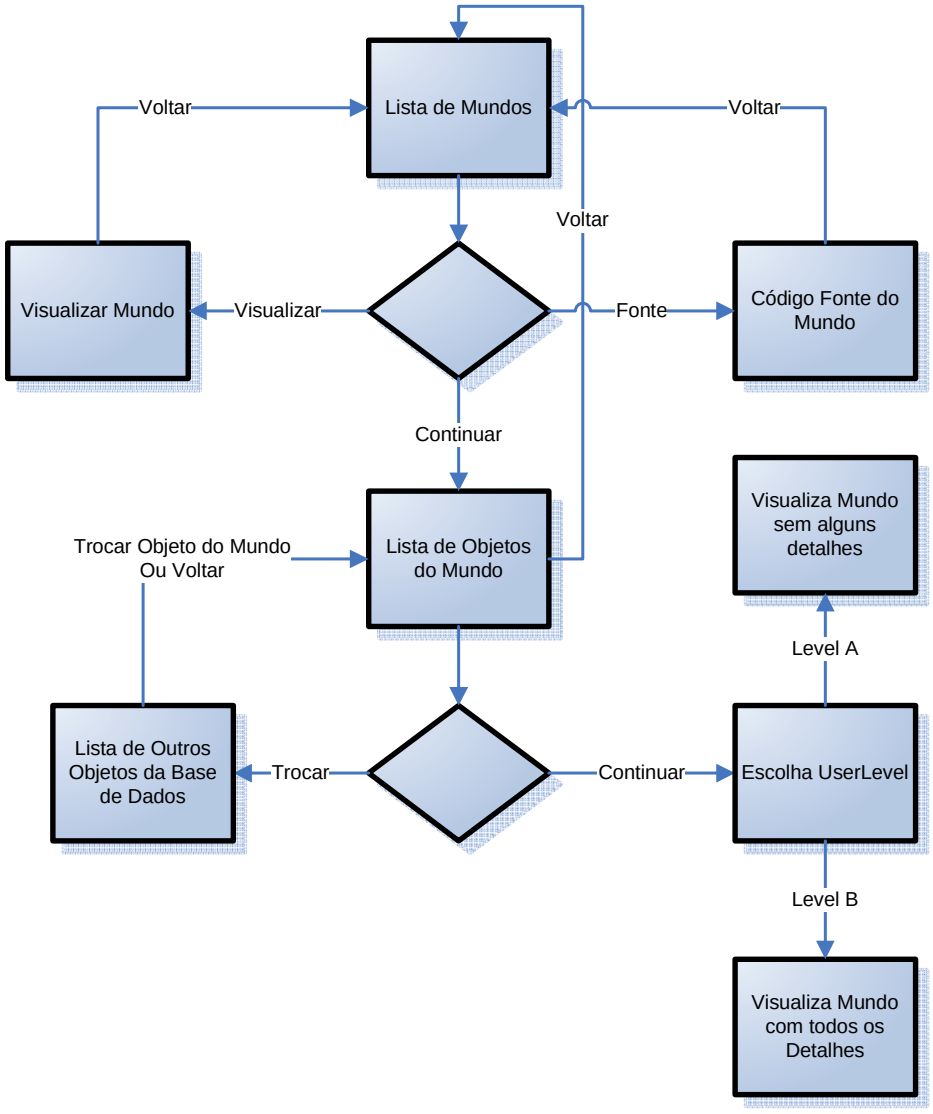


Figura 28: Fluxo de navegação do sistema de caso de teste do MoGORViR

Após a escolha do mundo o sistema exibe uma lista dos objetos contidos neste mundo. Estes objetos podem ser outros documentos X3D que foram armazenados no SGBD ou fragmentos de código reutilizáveis descritos através do atributo “DEF”.

O usuário pode escolher qualquer destes objetos para ser substituído por outro objeto armazenado no SGBD. Para isto ele deve selecionar o objeto indesejado e escolher a opção “Trocar”, será exibida então uma tela mostrando todos os outros objetos ou fragmentos de objetos armazenados no SGBD. Após escolher o objeto

substituto o usuário deverá escolher a opção confirmar. Após isto ele será levado à tela contendo os objetos do mundo base, onde a mudança do objeto já estará registrada. O usuário poderá repetir este procedimento quantas vezes for necessário.

Quando o usuário terminar de montar o seu mundo, ele deverá então escolher a opção continuar. Será mostrada uma tela contendo duas opções, visualizar o mundo construído como um usuário com userLevel “A” ou “B”. Ao clicar em uma das opções o sistema filtra o mundo construído de acordo com o userLevel do usuário, cria um arquivo X3D contendo todas as informações escolhidas e redireciona o browser para a exibição deste mundo.

Este mundo construído não é armazenado na base de dados, servindo apenas para demonstrar a capacidade do MoGORViR, de inserir objetos X3D em uma base de dados e recuperá-los completamente ou apenas algum fragmento, para auxiliar na construção de um mundo virtual através de objetos reutilizáveis. A formatação do mundo de acordo com o userLevel do usuário permite que um Ambiente Virtual Adaptativo apresente objetos detalhados de acordo com o perfil do usuário. Outro fator importante é que o MoGORViR permite a construção do ambiente virtual em tempo real, permitindo assim, que uma mudança do perfil possa provocar uma mudança instantânea no mundo virtual.

O Anexo C mostra uma série de telas ilustrando o funcionamento da aplicação descrita acima.

CAPÍTULO 6

CONCLUSÃO E TRABALHOS FUTUROS

A disseminação da internet tem incentivado cada vez mais a construção de ferramentas de ensino. Os ambientes que agregam adaptatividade, ambiente de tutoria e utilização da Realidade Virtual (RV) para simulações, permitem uma maior interatividade do usuário com o ambiente, propiciando um aprendizado mais efetivo.

Entretanto, os ambientes atuais ainda apresentam algumas limitações, sendo as mais importantes:

- A inexistência de adaptação relativa à apresentação de objetos em vários níveis de detalhamento;
- A impossibilidade de modificar o ambiente em uma mesma sessão; e
- A inexistência de mecanismos de reutilização de objetos de um BD para a construção de mundos virtuais.

Este trabalho teve como objetivo a construção de um módulo de gerenciamento de objetos de realidade virtual reutilizáveis que desse apoio à construção de Ambientes Virtuais Adaptativos que pudessem superar os problemas acima.

Neste contexto, foi construído o MoGORViR que agregou diversos serviços inteligentes de armazenamento e gerenciamento de objetos de realidade virtual. A criação do caso de teste do capítulo 5 permitiu demonstrar que o MoGORViR pode ser um elemento importante para a construção de ambientes virtuais que superem as barreiras tradicionais.

Durante a construção deste módulo ocorreram alguns problemas que impediram que mais recursos fossem adicionados e que o sistema fosse portado para outros SGBD como o PostgreSQL 8.0, por exemplo. A primeira delas, diz respeito à inexperiência do aluno em lidar com a manipulação de documentos XML, especialmente em conjunto com um SGBD. A segunda se refere a dificuldades na realização de testes, ocasionadas pela pouca disponibilidade de mundos virtuais

descritos em X3D e o fato de que o ambiente ao qual este módulo deve ser integrado ainda está em fase de construção.

Como trabalho futuro, tem-se que a principal necessidade é a realização de testes mais completos com uma base maior de documentos, para que se possa avaliar a robustez da solução construída e aperfeiçoá-la conforme as necessidades. Outra tarefa muito interessante seria portar o MoGORViR para trabalhar com outros SGBD, em especial para o PostgreSQL 8.0, que por ser um software gratuito permitirá a disponibilização deste trabalho para um número maior de pessoas.

Apesar dos testes realizados, somente quando o sistema AVPersonal estiver pronto é que se poderá avaliar com precisão se o MoGORViR atende a todas as necessidades do sistema, possibilitando fazer os ajustes pertinentes.

REFERÊNCIAS BIBLIOGRÁFICAS

[Abiteboul et al, 2000] ABITEBOUL, S., BUNEMAN, P., SUCIU, D. Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann, 2000.

[AulaNet, 2004] Sistema AulaNet de Ensino a Distância. http://www.aulanet.com.br. Último acesso em: agosto/2005.

[Ames et al, 1997] AMES, L. A.; NADEAU, D. R.; MORELAND, J. L. The VRML 2.0 Sourcebook, 2nd. Edition, John Wiley & Sons, Inc., 1997

[Anastassakis et al., 2001] ANASTASSAKIS, G.; RITCHING, T.; PANAYIOTOPOULOS, T. Multi-agent Systems as Intelligent Virtual Environments. LNAI 2174, 2001

[Aquino, 2001] AQUINO, M. S. Educação a Distância sob as Óticas Groupware e Sistema Produtivo. Monografia. 2001. 97 f. Monografia (Especialização em Engenharia de Produção) – Curso de Especialização em Engenharia de Produção, CCT/UFPB, Campina Grande – PB, 2001.

[Aquino, 2005] AQUINO, M. S. Uso de técnicas de Gerenciamento de Objetos 3D e de Agentes Inteligentes para Melhora da Usabilidade de Ambientes Virtuais Adaptativos. Exame de Qualificação e Proposta de Tese. CIN/UFPE, Recife – PE, 2005.

[Bourret, 2003] BOURRET, R. XML and Databases. Disponível em: http://www.rpbourret.com/xml/XMLDatabaseProds.htm

[Chittaro e Ranon, 2002a] CHITTARO, L.; RANON, R. *New Directions for the Design of Virtual Reality Interfaces to E-Commerce Sites.* Proceedings of AVI 2002: 5th International Conference on Advanced Visual Interfaces, ACM Press, New York, 2002

[Chittaro e Ranon, 2002b] CHITTARO, L.; RANON, R. Dynamic generation of personalized VRML content: a General Approach and its Application to 3D E-Commerce. Proceedings of Web3D 2002: 7th International Conference on 3D Web Technology, ACM Press, New York, 2002

[CGI] http://hoohoo.ncsa.uiuc.edu/cgi/overview.html. Último acesso em: agosto/2005.

[Direct3D, 2004] Direct3D Microsoft Corporation
http://www.microsoft.com/windows/directx/default.aspx Último acesso em: agosto/2005.

[DOM, 2003] W3C Document Object Model. Disponível em: http://www.w3c.org/DOM. Último acesso em: agosto/2005.

[Goldberg, 1996] GOLDBERG, M. W. Using a Web-Based Course Authoring Tool to Develop Sophisticated Web-Based Course. Book Title: "Web Based Instruction (WBI)", Editor Badrul H Khan, University of Texas at Brownsville, 1996.

[HTML] http://www.w3.org/MarkUp. Último acesso em: agosto/2005.

[Java, 2005] Java Technology. Disponível em: http://www.sun.com/java/about/. Último acesso em: agosto/2005.

[Java3D, 2004] Sun Microsystems Java 3D Engineering Team. Java 3D API Tutorial. Java3D API 1.3.1. http://java.sun.com/developer/onlineTraining/java3d . Último acesso em: agosto/2005.

[JDOM, 2004] http://www.jdom.org/ . Último acesso em: agosto/2005.

[LearningSpace, 2004] LearningSpace. http://www.lotus.com/lotus/offering3.nsf/wdocs/learningspacehome. Último acesso em: agosto/2005.

[Lotus, 1998] Lotus White Paper: Learning Space – Solutions for Anytime Learning. November, 1998. http://www.lotus.com/home.nsf/tabs/learnspace .

[Lucena et al., 1998] LUCENA, C.J.P., FUKS, H., MILIDIU, R., MACEDO, L., SANTOS, N., LAUFER, C., RIBEIRO, M.B., FONTOURA, M.F., NOYA, R.C., CRESPO, S., TORRES, V., DAFLON, L.; LUKOWIECKI, L. AulaNet - An Environment for the Development and Maintenance of Courses on the Web, Proceedings of ICEE'98 - International Conference On Engineering Education, Rio de Janeiro, Agosto, 1998.

[Lucena, e Fuks, 2000] LUCENA, C.J.; FUKS, H. Professores e aprendizes na Web: a educação na era da Internet. Edição e organização: Santos, N.; Rio de Janeiro: Clube do Futuro, 2000.

[Mello, 2002] Mello, R. S. Uma Abordagem Bottom-Up para a Integracao Semantica de Esquemas XML, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brasil, Tese de doutorado, 2002.

[Microsoft, 2004] Microsoft.com Home. Disponível: Microsoft Corporation site. http://www.microsoft.com. Último acesso em: agosto/2005.

[Netscape, 2004] Netscape Netcenter. Disponível: Netscape site. http://www.netscape.com. Último acesso em: agosto/2005.

[Nijholt e Hulstijn, 2000] NIJHOLT, A.; HULSTIJN, J. Multimodal Interactions with Agents in Virtual Worlds. In: Kasabov, N. (ed.): Chapter 8 - Future Directions for Intelligent Information Systems and Information Science, Physica-Verlag: Studies in Fuzziness and Soft Computing, 2000.

[Osório et al., 2004] OSORIO, F.S.; MUSSE, S. R.; SANTOS, C. T.; HEINEN, F.; BRAUN, A.; SILVA, A. T. Ambientes Virtuais Interativos e Inteligentes: Fundamentos, Implementação e Aplicações Práticas. Tutorial in JAI – Jornada de Atualização em Informática/SBC. Salvador, Bahia, 2004.

[OpenGL, 2004] OpenGL API. Silicon Graphics. http://www.sgi.com/products/software/opengl. Último acesso em: agosto/2005.

[Oracle, 2005] http://www.oracle.com. Último acesso em: agosto/2005.

[Panayiotopoulos et al., 1999] PANAYIOTOPOULOS, T.; ZACHARIS, N.; VOSINAKIS, S. Intelligent Guidance in a Virtual University. Advances in Intelligent Systems – Concepts, Tools and Applications, pp. 33-42, Kluwer Academic Press, 1999.

[Pequeno et al., 2004] PEQUENO, M.; LOUREIRO, R. C.; SILVA, C. Modelo para Gestão e Implementação de Ambientes Virtuais de Aprendizagem numa Perspectiva de Interface Adaptativa. VIII Congreso de Educación a Distancia CREAD MERCOSUR/SUL 2004, 7 al 10 de septiembre 2004 - Córdoba – Argentina. http://www.iua.edu.ar/cread2004/trabajos/contenido/ponencias/8-9/C/cuarto.pdf

[PostgreSQL, 2005] http://www.postgresql.org/. Último acesso em: agosto/2005.

[Rickel e Johnson, 1997] RICKEL, J.; JOHNSON, W. Integrating Pedagogical Capabilities in a Virtual Environment Agent. Proceedings of the 1st International Conference on Autonomous Agents, February, ACM Press. 1997.

[Santos, 2004] SANTOS, C. T. Um ambiente Virtual Inteligente e Adaptativo Baseado em Modelos de Usuário e Conteúdo. São Leopoldo: PIPCA da UNISINOS, 2004. Dissertação de Mestrado.

[Saldias, 2002] SALDÍAS, G. M. J. C. AZEVEDO, F. M. LUZ, R. P. Virtual Reality in Intelligent Tutoring Systems.

[Teichrieb, 1999] Teichrieb, V. *Avatares como Guias Interativos para Auxílio na Navegação em Ambientes Virtuais Tridimensionais*. Recife: Pós-Graduação em Ciência da Computação do CIn/UFPE, 1999. Dissertação de Mestrado.

[Tessarollo, 2000] TESSAROLLO, M. R. M. Ambiente de Autoria de Cursos a Distância (AutorWeb). Campinas: Instituto de Computação da UNICAMP, 2000. Dissertação de Mestrado.

[VRML, 1997] VRML97 Functional specification - ISO/IEC 14772-1:1997. http://www.web3d.org/x3d/vrml

[WebCT, 2004] Web Course Tool – WebCT. http://www.webct.com. Último acesso em: agosto/2005.

[Web3Da, 2004] Web3D Consortium. Open Standards for Real-Time 3D Communication. Disponível em: http://www.web3d.org/x3d. Último acesso em: agosto/2005.

[Web3Db, 2004] Web3D Consortium. X3D Viewers, Browsers & Plug-ins http://www.web3d.org/applications/tools/viewers_and_browsers/. Último acesso em: agosto/2005.

[Web3Dc, 2004] Web3D Consortium. X3D Overview. Disponível em http://www.web3d.org/x3d/overview.html. Último acesso em: agosto/2005.

[X3D, 2004] X3D Overview. http://www.web3d.org/x3d/overview.html. Último acesso em: agosto/2005.

[XDB, 2003] XML:DB Initiative: Enterprise Technologies for XML Databases. Disponível em: http://xmldb-org.sourceforge.net. Último acesso em: agosto/2005.

[XML, 2003] eXtensible Markup Language. Disponível em: http://www.w3c.org/xml. Último acesso em: agosto/2005.

[XML, 2003b] XML.COM: Introduction to Native Xml Databases. Disponível em: http://www.xml.com/lpt/a/2001/10/31/nativexmldb.html. Último acesso em: setembro/2004.

[XPA, 2003] XML PATH Language (XPATH). Disponível em: http://www.w3c.org/TR/xpath. . Último acesso em: agosto/2005.

[XQL, 2004] XML Query Language (XQL). Disponível em: http://www.ibiblio.org/xql/xql-proposal.html. Último acesso em: agosto/2005.

[XSD, 2003] W3C XML SCHEMA. Disponível em: http://www.w3c.org/xml/Schema. Último acesso em: agosto/2005.

ANEXO A

Este anexo contém o código fonte do MoGORViR. Ele é composto das seguintes classes: RepositorioObjetosX3dOracle, RepositorioObjetosX3d, AdaptadorObjeto e RepositorioX3dFactory.

```
package br.ufpe.cin.tg.mogorvir;


import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.StringReader;
import java.io.Writer;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Iterator;
import java.util.List;
import java.util.Vector;

import oracle.jdbc.OracleDriver;
import oracle.jdbc.OraclePreparedStatement;
import oracle.jdbc.OracleResultSet;
import oracle.sql.CLOB;

import org.jdom.Document;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.jdom.output.XMLOutputter;
import org.jdom.output.Format;

public class RepositorioObjetosX3dOracle implements RepositorioObjetosX3d {
	private Connection conexao;

	protected RepositorioObjetosX3dOracle() {

	}

	private void conectar() throws SQLException{
		DriverManager.registerDriver(new OracleDriver());
		conexao = DriverManager.getConnection(
				"jdbc:oracle:thin:@casa02:1521:orcl",
				"SYSTEM", "fonseca");
	}

	private void desconectar() throws SQLException {
```

```
		if(this.conexao != null) {
			this.conexao.close();
		}
	}

	public void inserirObjeto(String nome, String conteudo) throws SQLException {
		CLOB clob = null;
		String query;
		PreparedStatement pstmt = null;
		try{
			this.conectar();

			query = "INSERT INTO x3d_object_table" +
					" VALUES (?, XMLType(?)) ";

			pstmt = conexao.prepareStatement(query);

			clob = getCLOB(conteudo, conexao);
			pstmt.setObject(1, nome);
			pstmt.setObject(2, clob);

		} catch(SQLException sqlexc){
			sqlexc.printStackTrace();
			throw sqlexc;
		} finally {
			this.desconectar();
		}
	}

	public Document buscarObjeto(String nomeObjeto) throws SQLException {
		String arquivo="";
		Document doc = new Document();
		try{
			this.conectar();

			OraclePreparedStatement stmt =
				(OraclePreparedStatement) conexao.prepareStatement(
						"SELECT
extract(objeto_x3d,'/X3D').getClobVal()" +
						" conteudo FROM x3d_object_table"+
						" WHERE nome_arquivo_x3d = ?");
			stmt.setString(1,nomeObjeto);

		   OracleResultSet res = (OracleResultSet)stmt.executeQuery();
		   if(res.next()) {
			arquivo = res.getString("conteudo");
			doc = this.constroiDocumento(arquivo);
		   }

		} catch(SQLException sqlexc){
			sqlexc.printStackTrace();
			throw sqlexc;
		} finally {
			this.desconectar();
		}

		return doc;
	}
```

```
	public Document buscarFragmento(String nomeFragmento) throws SQLException {
		String conteudo = "";
		Document doc = new Document();
		try{
			this.conectar();

			OraclePreparedStatement stmt =
				(OraclePreparedStatement) conexao.prepareStatement(
					"SELECT value(e).getClobVal() as conteudo FROM" +
					" x3d_object_table d, table(xmlsequence(extract(" +
					"d.objeto_x3d,'//*[@DEF=\"" + nomeFragmento +"\"]')))
e");

		   OracleResultSet res = (OracleResultSet)stmt.executeQuery();
		   if(res.next()) {
			conteudo = res.getString("conteudo");
			doc = this.constroiDocumento(conteudo);
		   }

		} catch(SQLException sqlexc){
			sqlexc.printStackTrace();
			throw sqlexc;
		} finally {
			this.desconectar();
		}
		return doc;

	}

	public Document buscarFragmento(String nomeFragmento, String nomeObjeto)
			throws SQLException {
		String conteudo = "";
		Document doc = new Document();
		try{
			this.conectar();

			OraclePreparedStatement stmt =
				(OraclePreparedStatement) conexao.prepareStatement(
				"SELECT value(e).getClobVal() as conteudo FROM" +
				" x3d_object_table d, table(xmlsequence(extract(" +
				"d.objeto_x3d,'//*[@DEF=\"" + nomeFragmento +"\"]'))) e"+
				  " WHERE nome_arquivo_x3d = ?");
			stmt.setString(1,nomeObjeto);

		   OracleResultSet res = (OracleResultSet)stmt.executeQuery();
		   if(res.next()) {
			conteudo = res.getString("conteudo");
			doc = this.constroiDocumento(conteudo);
		   }

		} catch(SQLException sqlexc){
			sqlexc.printStackTrace();
			throw sqlexc;
		} finally {
			this.desconectar();
		}
		return doc;
```

```
}

public List listarObjetos() throws SQLException {
        List lista = new Vector();
        String nome="";

        try{
                this.conectar();

                OraclePreparedStatement stmt =
                        (OraclePreparedStatement) conexao.prepareStatement(
                                        "SELECT nome_arquivo_x3d" +
                                        " nome FROM x3d_object_table");

           OracleResultSet res = (OracleResultSet)stmt.executeQuery();
           while(res.next()) {
                nome = res.getString("nome");
                lista.add(nome);
           }

        } catch(SQLException sqlexc){
                sqlexc.printStackTrace();
                throw sqlexc;
        } finally {
                this.desconectar();
        }

        return lista;
}

public List listarFragmentos() throws SQLException {
        List lista = new Vector();
        String nome="";
        String temp;
        int indice;

        try{
                this.conectar();

                OraclePreparedStatement stmt =
                        (OraclePreparedStatement) conexao.prepareStatement(
                           "select value(e).getClobVal() as nome from " +
                           "x3d_object_table d,table(xmlsequence(extract(" +
                           "d.objeto_x3d,'//*[@DEF]'))) e");

           OracleResultSet res = (OracleResultSet)stmt.executeQuery();
           while(res.next()) {
                temp = res.getString("nome");
                indice = temp.indexOf("DEF");
                nome = temp.substring(indice+5,temp.indexOf("\"",indice+5));
                lista.add(nome);
           }

        } catch(SQLException sqlexc){
                sqlexc.printStackTrace();
                throw sqlexc;
        } finally {
                this.desconectar();
```

```
		}

		return lista;

	}

	public List listarFragmentos(String nomeObjeto) throws SQLException {
		List lista = new Vector();
		String nome="";
		String temp;
		int indice;

		try{
			this.conectar();

			OraclePreparedStatement stmt =
				(OraclePreparedStatement) conexao.prepareStatement(
				   "select value(e).getClobVal() as nome from " +
				   "x3d_object_table d,table(xmlsequence(extract(" +
				   "d.objeto_x3d,'//*[@DEF]'))) e" +
				   " WHERE nome_arquivo_x3d = ?");
				stmt.setString(1,nomeObjeto);

		   OracleResultSet res = (OracleResultSet)stmt.executeQuery();
		   while(res.next()) {
			temp = res.getString("nome");
			indice = temp.indexOf("DEF");
			nome = temp.substring(indice+5,temp.indexOf("\"",indice+5));
			lista.add(nome);
		   }

		} catch(SQLException sqlexc){
			sqlexc.printStackTrace();
			throw sqlexc;
		} finally {
			this.desconectar();
		}

		return lista;

	}

	private static CLOB getCLOB(String xmlData, Connection conn) throws SQLException{
		CLOB tempClob = null;

		try{
		//	If the temporary CLOB has not yet been created, create one
			tempClob = CLOB.createTemporary(conn, true,
CLOB.DURATION_SESSION);

		//	Open the temporary CLOB in readwrite mode, to enable writing
			tempClob.open(CLOB.MODE_READWRITE);
		//	Get the output stream to write
			Writer tempClobWriter = tempClob.getCharacterOutputStream();
		//	Write the data into the temporary CLOB
			tempClobWriter.write(xmlData);
		//	Flush and close the stream
			tempClobWriter.flush();
```

```
                tempClobWriter.close();
        //      Close the temporary CLOB
                tempClob.close();

        } catch(SQLException sqlexp){
                tempClob.freeTemporary();
                sqlexp.printStackTrace();
        } catch(Exception exp){
                tempClob.freeTemporary();
                exp.printStackTrace();
        }
        return tempClob;
}

private Document constroiDocumento(String entrada) {
        SAXBuilder builder = new SAXBuilder();
        Document doc = null;
        try {
                doc = builder.build(new StringReader(entrada));
        } catch (JDOMException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
        return doc;
}

public static void main(String[] args) {
        RepositorioObjetosX3d repositorio = RepositorioX3dFactory.
                        createRepositorioObjetosX3d(RepositorioX3dFactory.ORACLE);
        Document resposta;
        String item;
        List lista;
        Iterator iterator;

        String nome = "room.x3d";
        XMLOutputter out = new XMLOutputter(Format.getPrettyFormat());

        try {
                resposta = repositorio.buscarObjeto(nome);
                out.output(resposta,System.out);
                resposta = AdaptadorObjeto.adaptarObjetoUsuario(resposta,"B");
                out.output(resposta,new FileOutputStream(new File("teste.x3d")));

                lista = repositorio.listarFragmentos(nome);
                iterator = lista.iterator();
                while(iterator.hasNext()) {
                        item = (String)iterator.next();
                        System.out.println(item);
                }

        } catch(Exception sqlexc) {
                sqlexc.printStackTrace();
        }
}

}
```

```
package br.ufpe.cin.tg.mogorvir;

import java.util.Iterator;
import java.util.List;

import org.jdom.*;
import org.jdom.xpath.XPath;

public class AdaptadorObjeto {
	public static Document adaptarObjetoUsuario(Document doc, String nivelUsuario) {
		XPath x;
		Element noAtual;
		Attribute atributo;
		List elementos;
		Iterator iterator;
		String nivelAtributo;

		try {
			x = XPath.newInstance("//*[@userLevel]");
			elementos = x.selectNodes(doc);

			iterator = elementos.iterator();
			while(iterator.hasNext()) {
				noAtual = (Element)iterator.next();
				atributo = noAtual.getAttribute("userLevel");
				nivelAtributo = atributo.getValue();

				if(nivelAtributo.hashCode() > nivelUsuario.hashCode()) {
					noAtual.detach();
				} else {
					noAtual.removeAttribute(atributo);
				}

			}

		} catch(Exception exc) {
			exc.printStackTrace();
		}

		return doc;
	}

	public void trocarElementos(Element in, Element out) {
		Parent parent = out.getParent();
		Element element;
		Document doc;

		int index = parent.indexOf(out);
		out.detach();
		if(parent instanceof Element) {
			element = (Element)parent;
			element.addContent(index,in);
		} else {
			doc = (Document)parent;
			doc.addContent(index,in);
		}
	}
```

```
}

package br.ufpe.cin.tg.mogorvir;

import java.sql.SQLException;
import java.util.List;
import org.jdom.Document;

public interface RepositorioObjetosX3d {

	public void inserirObjeto(String nome, String conteudo) throws SQLException;

	public Document buscarObjeto(String nomeObjeto) throws SQLException;

	public Document buscarFragmento(String nomeFragmento) throws SQLException;

	public Document buscarFragmento(String nomeFragmento, String nomeObjeto)
		throws SQLException;

	public List listarObjetos() throws SQLException;

	public List listarFragmentos() throws SQLException;

	public List listarFragmentos(String nomeObjeto) throws SQLException;
}

package br.ufpe.cin.tg.mogorvir;

public class RepositorioX3dFactory {
	public static final int ORACLE = 1;

	public static final RepositorioObjetosX3d createRepositorioObjetosX3d(int opcao) {
		RepositorioObjetosX3d retorno = null;

		if(opcao == RepositorioX3dFactory.ORACLE) {
			retorno = new RepositorioObjetosX3dOracle();
		} else {
			//Espaço para definir suporte a novos SGBD
		}
		return retorno;
	}
}
```

ANEXO B

Este anexo contém os scripts que foram utilizados para manipulações do SGBD pelo MoGORViR

```
CREATE TABLE x3d_object_table (
    nome_arquivo_x3d VARCHAR2(50) PRIMARY KEY,
    objeto_x3d XMLType
)

INSERT INTO x3d_object_table VALUES (?, XMLType(?))

SELECT VALUE(e).getClobVal() as nome FROM x3d_object_table d,table(xmlsequence(extract
("d.objeto_x3d,'//*[@DEF]'))) e  WHERE nome_arquivo_x3d = ?

SELECT VALUE(e).getClobVal() as nome FROM x3d_object_table d,table(xmlsequence(extract
("d.objeto_x3d,'//*[@DEF]'))) e

SELECT nome_arquivo_x3d nome FROM x3d_object_table

SELECT VALUE(e).getClobVal() as conteudo FROM x3d_object_table d, table(xmlsequence(extract
(d.objeto_x3d,'//*[@DEF="?"]'))) e WHERE nome_arquivo_x3d = ?
SELECT VALUE(e).getClobVal() as conteudo FROM x3d_object_table d, table(xmlsequence(extract
( d.objeto_x3d,'//*[@DEF="?"]')))

    SELECT extract(objeto_x3d,'/X3D').getClobVal() conteudo FROM x3d_object_table WHERE
                                        nome_arquivo_x3d = ?
```

ANEXO C

Neste anexo são demonstradas as telas do sistema que testou as funcionalidades do MoGORViR. Elas aparecem da Figuras 29 a 35

Figura 29: Tela inicial para criação de um mundo personalizado onde é escolhido o mundo base.

Figura 30 Tela que mostra os objetos existentes no mundo base. Estes objetos podem ser trocados por outros armazenados no SGBD.

Figura 31: Lista de objetos disponíveis para substituir os objetos do mundo base escolhido.

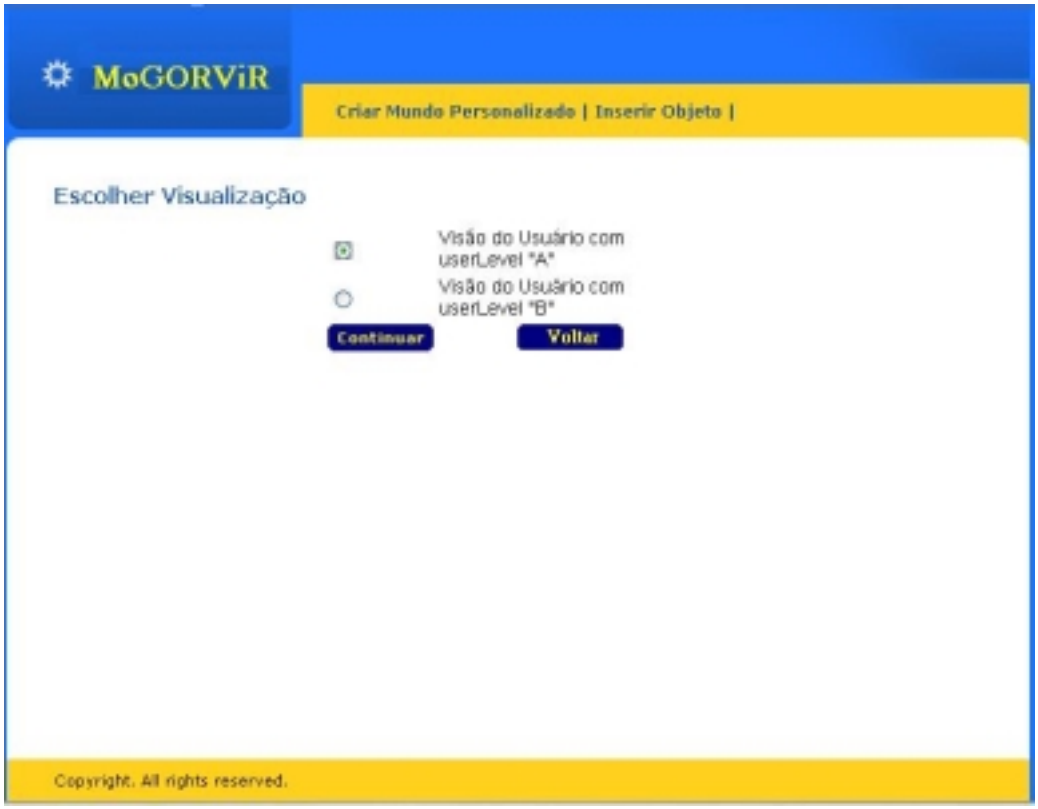


Figura 32: Tela onde o usuário escolhe se visualizará o mundo construído como um usuário de userLevel “A” ou “B”.

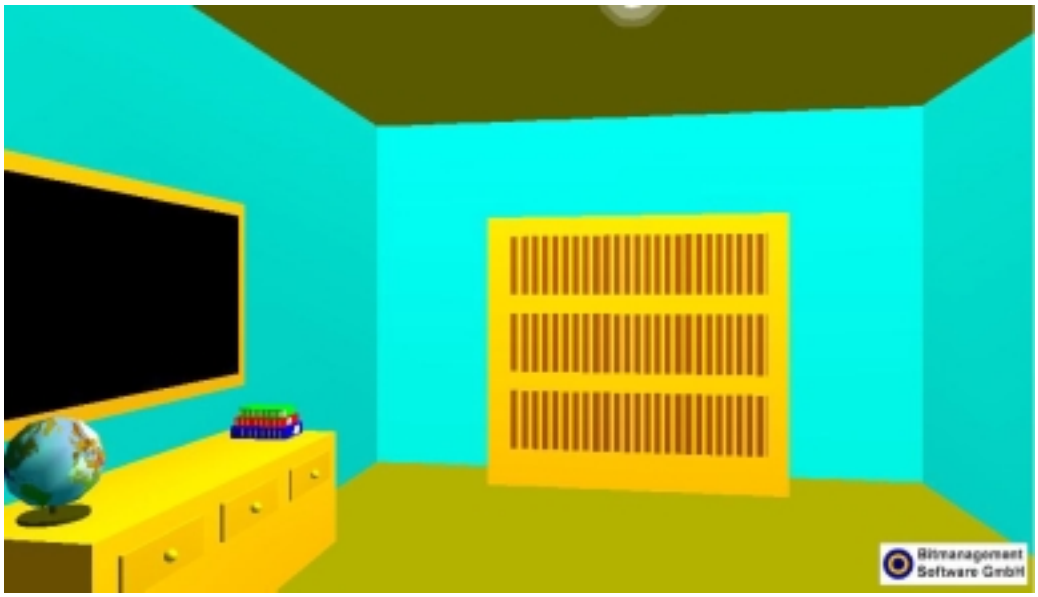


Figura 33: Mostra o mundo criado sob a ótica de um usuário com userLevel “A”.

Figura 34: Mostra o mesmo mundo da figura anterior, sendo visto desta vez sob a ótica de um usuário com userLevel “B”.

Figura 35: Mostra a tela onde é possível cadastrar novos objetos no SGBD.