UNIVERSIDADE FEDERAL DE SÃO CARLOS
DEPARTAMENTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

EXTENSIBILIDADE DE AMBIENTES VIRTUAIS COLABORATIVOS ATRAVÉS DE ESTÓRIAS INTERATIVAS NÃO LINEARES

Diego Daniel Duarte

São Carlos – SP
Maio12005

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

D812ea

Duarte, Diego Daniel.
Extensibilidade de ambientes virtuais colaborativos através de estórias interativas não lineares / Diego Daniel Duarte. -- São Carlos : UFSCar, 2009.
113 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2005.

1. Realidade virtual. 2. Extensibilidade. 3. VRML (Linguagem de programação). 4. VEML. I. Título.

CDD: 006 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Extensibilidade de Ambientes Virtuais Colaborativos através de Estórias Interativas não lineares"

DIEGO DANIEL DUARTE

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Profa. Dra. Regina Borges de Araujo
(Orientadora – DC/UFSCar)

Prof. Dr. José Hiroki Saito
(DC/UFSCar)

Profa. Dra. Wu Shin Ting
(FEEC/UNICAMP)

São Carlos
Maio12005

E ainda que tivesse o dom da profecia e conhecesse todos os mistérios e toda a ciência, e ainda que tivesse toda a fé, de maneira tal que transportasse os montes e não tivesse amor, nada seria.

I Corinthios 13:2

Dedico este trabalho a:

Meus tios Nivaldo e Marlene, pois, sem eles, a caminhada que fiz até aqui nem se quer teria começado;
À Silvio Tadao Fujisaki, grande professor e amigo.

Agradecimentos

À minha família que sempre me apoiou e incentivou, minha mãe, meu pai, minhas irmãs e minha namorada.

Aos meus amigos que, de uma forma de outra, colaboraram para a conclusão deste trabalho, em especial à Alexandre, Altieres, Fernando, João Paulo, Murilo e Yuji.

Aos meus professores da faculdade que me educaram não só com a formação acadêmica, mas também com a formação para a vida, principalmente André, Kleber e Rodrigo. E à minha orientadora, Regina, por ter acreditado e confiado em mim.

E, por fim e mais importante, a Deus.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Ambientes Virtuais Colaborativos (AVCs) Tridimensionais têm sido desenvolvidos principalmente pelas indústrias de jogos de computador, treinamento militar e industrial e desenvolvimento de protótipos. Entretanto, muitos dos sistemas existentes são focados em aplicações específicas e tem uma arquitetura de suporte intimamente ligada à aplicação. Como conseqüência disto, alterações na aplicação geralmente implicam em alterações nesta arquitetura de suporte também, tornando difícil e custoso desenvolver-se ou estender AVCs existentes.

Este trabalho apresenta uma nova abordagem para construção e extensão de AVCs através da integração de estórias interativas não lineares e Realidade Virtual. Com esta abordagem, as aplicações de AVCs são especificadas como estórias não lineares que podem ser alteradas parcial ou completamente, tornando fácil aos desenvolvedores construir ou estender aplicações de AVCs, tais como campanhas dinâmicas de shoppings virtuais, cursos padronizados de treinamento, jogos educacionais e de entretenimento, etc.

Para permitir a descrição de AVCs como estórias não lineares foi desenvolvido uma linguagem de descrição baseada em XML, chamada VEML. O uso do VEML para construção de AVCs pode facilitar a comunicação entre os diferentes profissionais envolvidos na construção de aplicações complexas de AVC.

Abstract

3D Collaborative Virtual Environments (CVE) applications have been impelled mainly by the computer games industry, military and industrial training and product design development. However, most of the existing systems are focused on specific tasks and have a supporting structure, which is closely tight to the application. As a consequence, a change in the application usually means a change in the supporting structure too, making it difficult and expensive to develop new or to extend existing CVEs.

This work presents a novel approach to building and extending CVEs through the integration of interactive non-linear stories and Virtual Reality. With this approach, CVEs applications are composed as non-linear stories which can be changed either completely or partly, making it easier for developers to build and/or extend CVEs applications, such as highly dynamic marketing campaigns to 3D virtual shops, customized training courses, educational and entertainment games, etc.

To allow the description of the CVE through interactive non-linear stories was developed a description language based on XML, named VEML. The use of a VEML for building a CVE application can facilitate the communication among the different professionals involved in the building of complex VE applications.

1. Introdução

Com o alcance em larga escala e o impacto sócio-econômico provocado pela WWW, novas tecnologias estão sendo consideradas para tornar natural a interação homem-máquina no ambiente da Web. A realidade virtual é uma das tecnologias cujas limitações estão sendo superadas para que múltiplos usuários possam compartilhar um mesmo ambiente virtual tridimensional sintetizado pelo computador, em aplicações que vão desde treinamento ao entretenimento.

Os Ambientes Virtuais Colaborativos (AVCs), ambientes 3D sintéticos compartilhados entre múltiplos usuários que colaboram entre si para atingir um objetivo comum, têm sido aplicados com sucesso na área de educação, treinamento e entretenimento. Na área de treinamento, os AVCs possibilitam a realização de simulações nas quais usuários remotamente localizados podem se deparar com situações semelhantes às que ocorrem na realidade, sem, contudo, haver a necessidade de despender gastos com os equipamentos utilizados no treinamento real, nem com o deslocamento de pessoas.

Outra utilização promissora dos AVCs é no campo do entretenimento. A indústria dos jogos de computador, em especial os que utilizam simulações 3D em jogos de aventura e estratégia em tempo real, tem crescido vertiginosamente nos últimos anos **[Oliveira et al., 2003]**. Neste tipo de aplicação os usuários são representados por

entidades, denominadas avatares, que visam representar, de forma realista, o usuário e seu comportamento (movimento, ações, expressões, etc.). Este avatar é imerso em um ambiente, real ou imaginário, integrado a múltiplas mídias, e com resposta em tempo-real, possibilitando a sensação de "presença" do usuário no ambiente do jogo. Os jogos de computador, principalmente os que envolvem educação, são exemplos promissores e motivadores de uso de Ambientes Virtuais Colaborativos.

Na educação, a utilização de AVCs possibilita ao aluno a experimentação, o estímulo ao raciocínio e o trabalho colaborativo, e é defendida como um dos principais meios para a aprendizagem efetiva **[Coscarelli, 1998]**.

Entretanto, as tecnologias existentes atualmente para distribuição de conteúdo tridimensional na Web ainda caminham a passos lentos. A dificuldade e o tempo necessário para o desenvolvimento de uma aplicação de Realidade Virtual são algumas das principais limitações para sua distribuição em larga escala. Isto se deve ao fato de que os sistemas existentes atualmente focam em um domínio específico de aplicação, fazendo com que ambientes ou situações não previstas no projeto original não possam ser facilmente suportadas **[Oliveira et al., 2000]**.

Outro problema, que diz respeito principalmente à extensão de ambientes, é que o modelo que rege o ciclo da simulação está tão sintonizado com a aplicação que alterações neste modelo implicam invariavelmente em alterações na aplicação em si **[Oliveira et al., 2000]**. Alguns pesquisadores dedicaram esforços para superar essa limitação, originando sistemas extensíveis, dentre os quais é possível citar Bamboo **[Watsen e Zyda, 1998]** e JADE **[Oliveira et al., 2003]**. Embora estes sistemas superem ou, ao menos, minimizem os problemas relacionados à extensibilidade de AVCs, eles ainda dependem de uma arquitetura própria que, de forma geral, ou não pode ser

adaptada para a Web ou o processo de adaptação é tão dispendioso que o torna inviável **[Oliveira et al., 2003]**.

Por outro lado, a utilização de estruturas que possibilitem o direcionamento preventivo do fluxo da simulação vem sendo utilizada há algum tempo na área de Drama Interativo (ID – *Interactive Drama*). Estas estruturas são abstrações de uma estória não-linear. Uma estória não-linear consiste basicamente da descrição de pontos chaves de uma, que pode ter o seu fluxo alterado mediante a intervenção do usuário e/ou sistema, através da execução de eventos pré-definidos **[Laaksolahti e Boman, 2002]**.

A utilização destas estruturas para o gerenciamento das ações do usuário e das reações geradas por elas em um AVC abre portas para se estender o ambiente não através da alteração de códigos que descrevem o comportamento de cada objeto, como acontece nos sistemas mencionados anteriormente, mas sim através da alteração da estória prevista para o AVC em questão **[Boukerche et al., 2004b]**.

Neste trabalho é apresentada uma proposta de utilização de estórias interativas não lineares como forma de possibilitar a extensão de AVCs baseados na WWW. Uma estória interativa não linear descreve o modelo da simulação, ou seja, o ciclo de eventos que o usuário pode realizar dentro de um AVC e suas reações correspondentes, através de simulações atômicas **[Boukerche et al., 2004b]**. Desta forma é possível separar a especificação geométrica do ambiente da especificação comportamental e funcional do mesmo, facilitando o desenvolvimento e a extensibilidade de aplicações de AVCs para WWW. Para possibilitar a utilização de estórias não lineares foi desenvolvida uma arquitetura que permite a extensibilidade de AVCs em tempo de execução, foi desenvolvida uma arquitetura que permite a substituição da estória não-linear que gerencia as ações do usuário neste ambiente sem que seja necessária a interrupção do seu funcionamento. Esta arquitetura é composta, basicamente, dos seguintes módulos:

interpretador de entradas, responsável por capturar e interpretar as ações do usuário, a partir de um dispositivo ou da interface de um navegador; *motor da estória*, responsável por armazenar a estória não-linear do ambiente e verificar possíveis mudanças no fluxo da mesma mediante as interações do usuário; e *motor da cena*, responsável por reunir e sincronizar estas informações para apresentá-las ao usuário.

A descrição da estória é feita através de uma linguagem de marcação que define os pontos de interação presentes no ambiente, as ações que o usuário pode realizar sobre estes pontos e as reações correspondentes. Esta linguagem "empacota" especificações geométricas do mundo virtual (tais como geometria, cor, textura, etc.), permitindo a definição do modelo da estória sem se preocupar com a descrição geométrica da cena e dos objetos.

1.1. Justificativa e relevância do tema

Estudos têm sido feitos desde 1996, para se desenvolver AVCs dinâmicos que pudessem refletir as ações do usuário no ambiente na forma de estórias sendo narradas por meio de simulação 3D, tal como o ambiente Alladdin da Disney **[Pausch, 1996]**.

Departamento de Storytelling Digital da universidade de Rundeturmstraβe, na Alemanha, vem despendendo esforços no sentido de desenvolver uma arquitetura que possibilite o suporte à contagem de estórias interativas utilizando como veículo as mídias digitais, incluindo ambientes virtuais 3D **[Schneider, 2003]**.

O foco destas aplicações, entretanto, é a narração automática de estórias, ou seja, o objetivo do modelo da estória é guiar o usuário por uma linha virtual do tempo, sobre a qual eventos ocorrem dentro do ambiente de simulação. A ocorrência destes eventos pode, eventualmente, ser reflexo das ações dos usuários. Neste caso, temos uma

estória interativa que, invariavelmente, é não-linear. Ainda assim, a utilização de um modelo que seja capaz de coordenar os acontecimentos em uma simulação que, eventualmente, possa refletir as ações do usuário, é um aspecto interessante para se utilizar em AVCs, principalmente porque este modelo pode desacoplar a aplicação da simulação, permitindo alterações na cena, sem que seja necessário interromper a simulação.

O crescimento da indústria de jogos de computador e treinamento comercial e industrial observado nos últimos anos serve de motivação para este trabalho. Motivação ainda maior é a aplicação de AVCs com estórias não lineares nas áreas de ensino à distância. Muitos professores têm idéias de estórias que podem ser utilizadas de forma educativa, mas não têm os conhecimentos necessários, e muitas vezes tempo, para desenvolver um ambiente de simulação com o qual o aluno possa interagir para participar da estória de uma forma mais ativa. A ferramenta proposta neste trabalho pode auxiliar estes professores na transposição da estória criada por eles para uma representação mais próxima da linguagem do computador, facilitando assim, o desenvolvimento de estórias em ambientes virtuais pelos professores.

Em um shopping virtual, com várias lojas, um profissional de marketing pode compor diferentes estórias para expressar campanhas de marketing que a loja deseje lançar. Por exemplo, no dia dos namorados, uma estória pode ser adicionada para uma loja em particular para que quando um usuário entrar na loja, ele veja corações vermelhos sobre os produtos em oferta. Usuários que tocarem nestes corações entram automaticamente em um concurso para concorrer a um prêmio. Neste caso, a adição de novos objetos 3D, alteração de textura, adição de novos pontos de interação (corações vermelhos) e de novas respostas às ações dos usuários, podem ser parte da campanha de marketing e, portanto, fazer parte da estória que foi adicionada ao cenário.

Por fim, a arquitetura apresentada neste trabalho, pode auxiliar também aplicações de treinamento, ou na reconstrução de ambientes de simulação, quando há a necessidade de alterações de caráter funcional. Por exemplo, uma simulação de treinamento de montagem de um motor de carro, em que as peças que constituem o motor foram alteradas, em vez de mudar o ambiente virtual, através, por exemplo, de reconfiguração manual, o roteiro pode ser alterado e refletido no ambiente de maneira mais automática, mudando-se apenas o roteiro do treinamento.

1.2. Objetivos

O objetivo principal deste trabalho é possibilitar a extensão de AVCs baseados na tecnologia da WWW. A definição de uma estrutura que, através de um modelo da estória fornecido por um autor, suporte a construção de simulações de Ambientes Virtuais Colaborativos com estórias não lineares são frutos deste projeto. Com isso, pretende-se contribuir para o desenvolvimento em maior escala deste tipo de aplicações, que vem sendo utilizadas com sucesso nas áreas de educação, treinamento, entretenimento, etc.

Na estrutura proposta o modelo da estória é descrito através de uma linguagem de marcação denominada VEML (*Virtual Environment Markup Language*). Estas definições podem ser criadas idealmente por um roteirista para descrever a disposição dos objetos na cena gráfica e as possibilidades interações destes objetos entre si, incluindo possíveis eventos que são gerados em decorrência destas interações. Desta forma pretende-se diminuir os esforços para desenvolvimento de AVCs para a Web e facilitar a extensão de aplicações já existentes bem como o desenvolvimento de novas aplicações.

1.3. Organização do trabalho

No capítulo 2 é apresentado um estudo detalhado sobre AVCs, apresentando os requisitos deste tipo de aplicações, em especial o requisito de extensibilidade. Neste capítulo também são apresentados sistemas existentes atualmente que suportam AVCs, bem como um modelo genérico para suporte de aplicações deste tipo. No terceiro capítulo são descritos os padrões existentes para distribuição de conteúdo 3D na Web, tais como VRML/X3D, Java3D e MPEG-4, e apresentados os sistemas Bamboo e JADE, que possibilitam a extensão de AVCs e fazem uso destes padrões para a descrição geométrica do mundo virtual.

No capítulo 4 são apresentados os trabalhos relacionados a drama interativo e storytelling, que procuram gerar narrativas automáticas através de um modelo de estória. Este modelo pode ser utilizado para descrever pontos de interação em um AVC.

A união do conceito de estórias interativas não lineares com os padrões de suporte à distribuição de conteúdo 3D, através da utilização de uma arquitetura modular que permita um intercâmbio entre as várias tecnologias existentes, objeto deste projeto, é apresentada nos capítulos 5 e 6. O capítulo 5 apresenta uma arquitetura modular que dá suporte à utilização de estórias não lineares em AVCs e o capítulo 6 descreve a linguagem VEML, que é utilizada para descrever estas estórias não lineares. Neste capítulo também é apresentado o conceito de simulações atômicas, que é utilizada na linguagem VEML e é seguido das conclusões, referências bibliográficas e anexos.

2. Ambientes Virtuais Colaborativos

A Realidade Virtual (RV) pode ser definida como uma interface natural e poderosa de interação homem-máquina que permite ao usuário interação, navegação e imersão em um ambiente tridimensional sintético, que é gerado pelo computador, através de canais multi-sensoriais, tais como visão, audição, tato, olfato e paladar **[Burdea e Coiffet, 1994]**.

As interfaces desenvolvidas com base nas tecnologias de RV estão entre as mais avançadas atualmente, permitindo visualizar, manipular e explorar o conteúdo de ambientes virtuais tridimensionais complexos, compostos por elementos gráficos capazes de representar praticamente qualquer tipo de informação. Um ambiente virtual tridimensional pode ser visto então como um cenário gráfico 3D que representa uma situação imaginária, ou não, sintetizada pelo computador **[Frery, 2003]**.

Existem pelo menos dois tipos de realidade virtual: imersiva e não imersiva. O primeiro tipo é baseado na utilização de capacetes visualizadores (*HMD – Head Mounted Display*) ou em salas de proteção nas paredes (CAVEs). Além do fator visual, os dispositivos ligados aos demais sentidos humanos também são importantes para a sensação de imersão como, por exemplo, o som, rastreadores de posicionamento automático da pessoa e dos movimentos da cabeça, controles reativos, etc. O segundo tipo, também denominado realidade virtual *Desktop*, baseia-se na utilização de

monitores dos PCs como dispositivos de apresentação do mundo virtual. Neste tipo de realidade virtual a interação pode ser obtida através de apontadores como o *mouse* **[Begault, 1994]**. Em ambos os casos, o sistema tem que ser capaz de interagir com o usuário, detectando suas ações e refletindo-as no mundo virtual.

Nas tecnologias de realidade virtual são encontrados três termos que estão relacionados, mas que possuem significados distintos: mundo virtual, ambiente virtual e sistemas de realidade virtual. Um mundo virtual corresponde à modelagem tridimensional geométrica de objetos, já um ambiente virtual refere-se não só a modelagem geométrica, mas também a simulação em si, permitindo a interação do usuário. Os sistemas de realidade virtual, por sua vez, são aqueles que fornecem o suporte adequado para a apresentação e gerenciamento dos ambientes virtuais.

2.1. Definição de AVC e características

Ambientes Virtuais Compartilhados (AVCs) podem ser definidos como sendo um ambiente tridimensional sintético compartilhado entre múltiplos usuários que interagem com o ambiente colaboram entre si, em tempo real, para atingir um objetivo comum. Segundo Singhal e Zyda **[Singhal e Zyda, 1999],** AVCs têm as seguintes características:

- **Compartilhamento de espaço**: todos os participantes têm a ilusão de estarem localizados em um mesmo espaço que, por sua vez, apresenta as mesmas características para todos os participantes;
- **Presença compartilhada**: **C**ada participante é representado dentro do ambiente virtual por um objeto geométrico, denominado avatar, e pode ver as representações dos outros usuários que estejam localizados no mesmo ambiente compartilhado;

- **Compartilhamento de tempo**: um participante deve ser capaz de ver o comportamento dos outros no momento em que isto ocorre, ou seja, o ambiente virtual deve suportar interação em tempo real;
- **Comunicação**: o sistema deve suportar a comunicação entre os participantes para manter, entre outros aspectos, a consistência do ambiente entre todos os usuários participantes.

Segundo o projeto proposto pela Living World **[Living World, 1997]**, as características das ações que um usuário pode realizar em um AVC são:

- Modificações na cena que podem ser iniciadas por qualquer cliente;
- Modificações que devem, potencialmente, ser sincronizadas em todos os clientes;
- Para todas as modificações, cada objeto, ou é responsável por refletir atualização localmente e enviar mensagens de sincronização para as instâncias deste mesmo objeto em todas as máquinas de usuários participantes, ou é um receptor destas mensagens de sincronização de outros usuários participantes e deve realizar a ação apropriada;
- Nem todas as modificações devem obrigatoriamente ser realizadas no instante em que é recebida; a otimização do desempenho depende de saber-se qual objeto é mais provável se ser alterado posteriormente;
- Nem todas as notícias de modificação precisam ser comunicadas; a otimização depende também de ter-se o conhecimento de quem necessita saber o quê e como.

2.2. Requisitos de AVCs – Extensibilidade

Um dos maiores desafios encontrados nos AVCs refere-se sobre a preservação da **consistência** do ambiente entre todos os participantes. Este requisito consiste em garantir que cada terminal compartilhando a mesma simulação seja capaz de trocar dados de atualização do mundo virtual de modo consistente com os demais participantes para garantir que seja apresentado exatamente o mesmo ambiente a todos os usuários. Aplicações desse tipo são mais eficientes quando podem usufruir de uma **conectividade** que possibilite comunicação em grupo, ou comunicação por difusão seletiva. A Internet, inicialmente, não possibilitava, entretanto, este tipo de conectividade. Sendo assim, muitos sistemas foram desenvolvidos inicialmente utilizando uma arquitetura cliente-servidor, baseado no paradigma "um-para-um", e esta implementação acabou se estendendo até os dias atuais.

Entretanto, este modelo apresenta limitações óbvias quanto à **escalabilidade**: quando a demanda de usuários é muito grande, o servidor torna-se facilmente um ponto de gargalo, já que toda a comunicação entre os participantes do ambiente deve inevitavelmente passar por ele.

Uma alternativa para possibilitar uma maior escalabilidade em AVCs baseados em redes ponto-a-ponto é apresentado por Boukerche e colegas **[Boukerche et al., 2004b]** e consiste da utilização de componentes especificados pela extensão do padrão MPEG-4 multi-usuário ainda em andamento que provê controle e gerenciamento de sessão e sincronização de AVCs na rede ponto-a-ponto híbrida Gnutella. Esta solução minimiza as desvantagens da utilização de uma arquitetura cliente-servidor ou puramente ponto-a-ponto.

Além dos requisitos de consistência, conectividade e escalabilidade apresentados anteriormente, são também requisitos de AVCs, segundo Boukerche e Araújo **[Boukerche e Araújo, 2004]**:

- **Persistência:** O estado do sistema tem que ser mantido, independente se o usuário está presente no ambiente ou não;
- **Reusabilidade:** Deve ser possível que elementos de outros ambientes possam ser reutilizados/combinados;
- **Interoperabilidade:** Definições e padrões devem ser utilizados para permitir a interação entre vários sistemas heterogêneos;
- **Adaptabilidade:** Um grande número de usuários conectados a partir de redes e dispositivos heterogêneos deve ser suportado;
- **Segurança:** a segurança deve ser garantida.

Tentativas atuais de se desenvolver AVCs que atendam os requisitos citados anteriormente estão baseadas em metodologias de projeto de componentes, engenharia de software e princípios de políticas de segurança. Mas o maior obstáculo para os sistemas atuais está em se tentar transferir as experiências obtidas em um sistema para outro. A origem deste problema está no fato do sistema ser fortemente relacionado em termos de implementação. Como conseqüência disto, os critérios do projeto moldam todos os mecanismos e processos de um sistema em particular. Por isto é difícil extrair qualquer parte relevante de código de um sistema e adaptá-lo para as especificações de um outro. Em muitos casos as incompatibilidades tornam ineficazes quaisquer tentativas de integração **[Oliveira et al., 2000]**.

Extensibilidade de AVCs

A extensibilidade é um requisito importante de AVCs e consiste da possibilidade de alterar ou melhorar as funcionalidades de uma aplicação sem que haja a necessidade de interromper o seu funcionamento. A possibilidade de estender-se um sistema já existente provê uma solução com capacidades quase ilimitadas.

Estender um sistema pode ser entendido como o processo de adição de capacidades ou funcionalidades a uma estrutura já existente. No contexto da computação em particular, a extensibilidade pode ferir-se à habilidade de alterar-se **[Watsen e Zyda, 1998]**:

- Um único objeto ou uma classe inteira;
- O programa em execução ou suas estruturas de suporte;
- O comportamento da execução de um programa.

Nos sistemas de RV apresentados anteriormente, qualquer modificação que se faça necessária no sistema exige invariavelmente que a aplicação seja finalizada para que a reconfiguração possa ser realizada. Estender-se uma aplicação de AVC sem que haja a necessidade da interrupção de seu funcionamento é um requisito importante quando se pensa em sistemas baseados na web, como, por exemplo, servidores de jogos, que devem oferecer serviço 24 horas por dia, 7 dias por semana. Neste tipo de aplicações, a interrupção do funcionamento do sistema para a realização de alterações é algo inadmissível. Desta forma, não é possível se pensar na existência de aplicações de RV sendo oferecidas com sucesso por servidores de aplicações comerciais de larga escala sem que este sistema possua suporte à extensibilidade.

2.3. Aplicações de AVCs

Ambientes Virtuais Colaborativos podem beneficiar uma infinidade de aplicações, como ambientes educacionais, vídeo-conferência, entre outras. Essa seção apresenta alguns exemplos de aplicações de AVCs.

2.3.1 Ambientes Educacionais

Nestes ambientes o professor pode iniciar a aplicação e os estudos conectando-se ao ambiente virtual de maneira local ou remota. Tais ambientes podem existir continuamente ou durarem o tempo de uma aula tradicional; podem oferecer livre acesso a todos os usuários ou serem restritos aos alunos da disciplina. O professor poderá guiar os alunos no interior do ambiente, explicando detalhes que julgue ser importante ou auxiliando os alunos nas dificuldades encontradas. A comunicação entre professores e alunos pode ser feita via *Chat*, fórum, vídeo ou áudio. Os estudantes podem gravar a aula ou realizar novas anotações para futura consulta. A Figura 1 mostra um exemplo de AVC para ambientes educacionais.

Figura 1 - AVC Educacionais

2.3.2 Vídeo-Conferência 3D

Aplicações de vídeo-conferência 3D requerem a representação realística do usuário em um ambiente virtual. A vídeo-conferência 3D é uma extensão das aplicações de teleconferência 2D, em que os usuários podem interagir tanto com os objetos virtuais quanto com os outros participantes. Os usuários podem ser representados através de um corpo virtual com o uso de faces realistas, como ilustrado na Figura 2.

2.3.3 Desenvolvimento de Protótipos

Os ambientes colaborativos oferecem meios de modificar objetos no interior de um ambiente virtual, bem como criar novos objetos. Um bom exemplo de aplicações colaborativas são os projetos de desenvolvimento de protótipos, em que os engenheiros participantes estão localizados em diferentes lugares, como ilustrado na Figura 3. A

Figura 2– AVC de vídeo conferência 3D

colaboração pode ser síncrona ou assíncrona. Na colaboração síncrona todos os participantes do ambiente podem atuar ao mesmo tempo; na assíncrona a participação de cada usuário pode acontecer em tempos diferentes.

Figura 3 – AVC para desenvolvimento de protótipos

2.3.4 Shopping Virtual

Um usuário pode entrar em uma loja virtual e interagir com produtos que estão à venda. Caso ele queira mais detalhes sobre o produto, poderá assistir a algum vídeo, ler um texto ou mesmo escutar as declarações dos clientes que já compraram aquele produto. Ele pode, ainda, chamar um vendedor, representado por um outro *avatar* ou um *bot* – um objeto simulado. A comunicação entre o cliente e o vendedor poderá ser feita através de texto, áudio ou vídeo. Na Figura 4 pode-se ver um exemplo de um shopping virtual utilizando AVC.

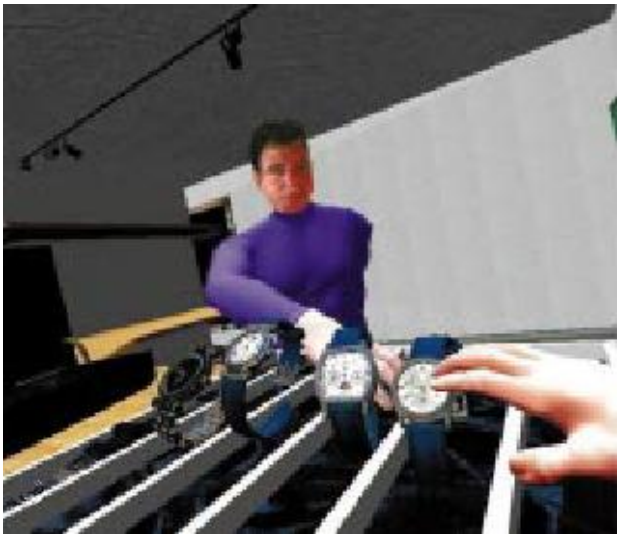


Figura 4 – Shopping Virtual

2.3.5 Comunidades Virtuais

As comunidades virtuais diferem das aplicações de ambientes virtuais colaborativos pelo fato de os ambientes existirem ao longo de um período determinado e de as aplicações existirem de forma permanente. Devido às comunidades virtuais poderem ser habitadas por uma grande quantidade de usuários, verdadeiras cidades podem ser formadas, com a existência de prefeito, polícia e centro de informações, tal

Figura 5 – Comunidade Virtual em um AVC

qual as cidades reais. Cada usuário pode ter um espaço próprio, onde possa realizar alterações, conforme as leis prescritas na "cidade virtual". A Figura 5 mostra um exemplo de uma comunidade virtual com três usuários *on-line*.

2.4. Sistemas de suporte à AVCs

A maioria dos sistemas existentes atualmente que suportam aplicações de AVCs está baseado em Sistemas Distribuídos de Realidade Virtual (SDRV). Um SDRV pode ser caracterizado por ambientes virtuais tridimensionais extensíveis que podem ser compartilhados por milhares de usuários que interagem no ambiente da aplicação. O ambiente é habitado por entidades que podem ser controladas pelos usuários ou por simulações (*scripts* ou agentes). O ambiente contém também entidades estáticas, como árvores, construções, etc., que podem sofrer modificações durante a execução da simulação. Quando um usuário executa uma ação, esta deve ser refletida localmente (fornecer uma resposta visual no terminal do usuário) assim como para todos os usuários remotos. Tudo isto deve ocorrer com uma latência aceitável, assim todos os usuários podem compartilhar da mesma visão do ambiente e interagir com o sistema, mantendo a sensação de imersão. Quando o número de usuários cresce significantemente, a atualização do ambiente através da rede se torna um grande desafio para os projetistas deste tipo de aplicação **[Boukerche e Araújo, 2004]**.

A primeira implementação bem sucedida de um sistema para suporte à AVCs em tempo real de larga escala foi o projeto SIMNET descrito na próxima sessão.

2.4.1 SIMNET

O SIMNET é uma tecnologia suportada pela rede ARPA que foi concebida pelo departamento de defesa dos Estados Unidos que utiliza redes locais e

processamento distribuído para simulações distribuídas de sistema para fins militares [Miller e Thorpe, 1995]. O sistema foi demonstrado inicialmente por simulações de treinamento dos tanques de guerra Abrams M1 e Bradley M2/3.

Na comunidade acadêmica estes sistemas começaram a ser desenvolvido utilizando-se aplicações baseadas em padrões e protocolos experimentais da Internet.

2.4.2 NPSNET

O NPSNET (*Naval Postgraduate School Networked Vehicle Simulator*) foi o sucessor do SIMNET e foi originalmente desenvolvido para simulações de campo de batalha. O grupo NPSNET esteve também envolvido no desenvolvimento de um protocolo de transferência para realidade virtual (*VRTP – Virtual Reality Transfer Protocol*) **[Singhal e Zyda, 1999]**.

O objetivo do grupo de pesquisa do NPSNET é explorar tecnologias de hardware e software para construção de mundos virtuais gerados por computador nos quais usuários e veículos possam se mover e interagir. As tecnologias de hardware estão focadas em estações para processamento de gráficos tridimensionais, redes locais e distribuídas, HMDs (*Head Mounted Displayers)* e monitores de alta resolução e novos dispositivos de entrada tridimensional. Neste sistema é possível dirigir um veículo através de um terreno, parar este veículo em frente a um prédio, descer do veículo, entrar a pé no prédio e interagir com os usuários que também estejam neste prédio. A máquina de cada participante pode ser configurada para simular um veículo militar ou pessoal de infantaria, e podem ser suportados milhares destas entidades interagindo **[Zyda, 1996]**.

No NPSNET os objetos do mundo virtual são representados por entidades. Uma entidade contém atributos que armazenam informações de estado – localização,

velocidade, cor, orientação, etc. – e mantém uma lista dos protocolos que ela é capaz de entender. Estes protocolos representam um mapeamento entre o conjunto de mensagens da rede e o conjunto de ações e comportamento da entidade. O protocolo recebe uma mensagem da rede, determina seu conteúdo e então altera apropriadamente o estado da entidade. Os protocolos controlam as entidades. Um protocolo é uma instância de um comportamento de uma entidade. Adicionar um novo comportamento para uma entidade requer a adição ou alteração de um protocolo. Nas simulações distribuídas, algumas entidades contêm informações de estado proprietárias, enquanto outras apenas irão refletir o estado e um objeto em uma outra máquina **[Capps et al., 2000]**.

A Figura 6 mostra a arquitetura do sistema NPSNET. O Universo Persistente é responsável pelo armazenamento e manutenção do estado da simulação, mesmo quando não há usuário presentes. AVC escalável representa uma instância em execução de um ambiente. Os Protocolos Dinâmicos são definidos por entidades, são intercambiáveis e podem ser transferido diretamente para a máquina cliente através da Internet. Bamboo é um mecanismo independente de linguagem e plataforma para permitir a reconfiguração dinâmica de aplicações.

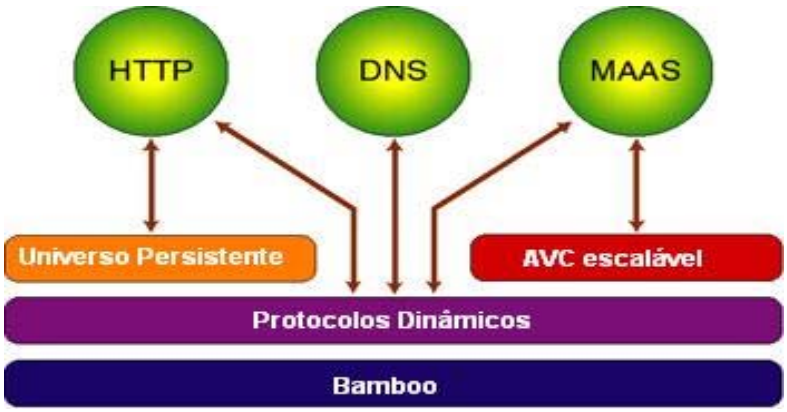


Figura 6 - Arquitetura do NPSNET-V

2.4.3 DIVE

O Ambiente Virtual Distribuído Interativo (*DIVE – Distributed Interactive Virtual Environment*) foi desenvolvido inicialmente para colaboração e teleconferência pelo Instituto Sueco de Ciência da Computação. Posteriormente o projeto europeu de tecnologias e serviços avançados de comunicação *AC040 Collaborative Virtual Environments (COVEN)* o adotou com o objetivo de desenvolver uma plataforma de propósito geral para aplicações comerciais baseadas em Realidade Virtual **[Hagsand, 1996]**.

DIVE suporta o desenvolvimento de ambientes virtuais multi-usuário compartilhados, interfaces de usuários e aplicações. Permite que participantes distribuídos sejam representados e naveguem livremente dentro de mundos virtuais 3D e comuniquem em tempo-real através de áudio, vídeo, texto, gestos gráficos simples e ferramentas de apoio a reuniões. Usuários também podem manipular objetos no mundo virtual (movendo e virando objetos ou adicionando objetos novos).

Um participante de um mundo virtual DIVE pode ser ou um usuário ou uma aplicação. Os usuários navegam em um espaço tridimensional e podem ver, conhecer e colaborar com outros usuários e aplicações do ambiente. Os usuários são representados por objetos gráficos denominados *body-icons*, que podem ser utilizados como modelos a partir dos quais os dispositivos de entrada do usuário é modelado no espaço 3D. O usuário vê e interage com o ambiente através de uma interface denominada *visualizador*. Um visualizador pode ser configurado para trabalhar com uma gama de dispositivos de entrada e saídas, como HMDs, *dataglovers*, etc. O visualizador lê as entradas geradas pelo usuário através dos dispositivos e as mapeia no sistema DIVE.

Isto inclui navegação no espaço 3D, seleção e movimentação de objetos, etc. **[Carlsson e Hagsand, 1993]**.

DIVE provê tanto uma arquitetura quanto um modelo de programação. A arquitetura está focada em soluções de software e rede que possibilita interações altamente compreensivas por cada ponto participante, isto é, os resultados de uma interação são exibidos, sempre que possível, imediatamente no ponto local, mas um pouco atrasados nos pontos remotos. O modelo de programação esconde detalhes da rede, permitindo ao programador focar-se no ambiente, seu conteúdo e sua lógica. Um ponto refere-se a um processo sendo executado em um computador específico conectado à Internet. Uma aplicação ou processo é qualquer programa ativo interfaceando com o ambiente virtual através da exibição e modificação de entidades, monitorando e reagindo a diferentes tipos de eventos, e assim por diante **[Frecon, 2004]**. A Figura 7 mostra a arquitetura de execução e a comunicação entre processos em DIVE.

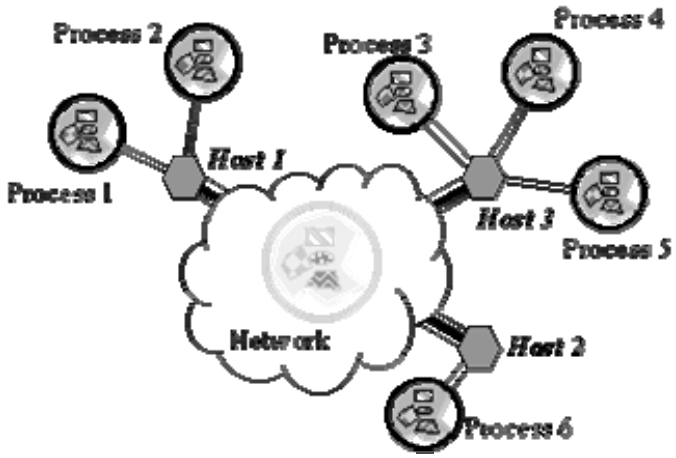


Figura 7 - Arquitetura de execução DIVE (Adaptado de [Frecon, 2004]**)**

Uma das principais características da arquitetura DIVE é o compartilhamento distribuído da base de dados do mundo. As interações dos usuários e das aplicações utilizam-se deste meio comum. A base de dados do mundo atua como uma abstração,

desta forma as aplicações DIVE operam somente como a base de dados e não se comunicam diretamente umas com as outras. Esta técnica permite uma clara separação entre as interfaces de aplicação e rede.

2.4.4 MASSIVE

MASSIVE (*Model, Architecture and System for Spatial Interaction in Virtual Environments*) é um outro sistema de realidade virtual distribuído que provê facilidades para se suportar interações de usuários e cooperação através de texto, áudio e gráficos. A interação é controlada por um modelo espacial de interação. A arquitetura de comunicação está baseada em processos de comunicação que apresentam interfaces em ambos os fins e que integra RPC, atributos e fluxos em um mesmo contexto. Um serviço de troca de interface espacial, gerenciador de aura, foi implementado para suportar a troca de interfaces em um contexto espacial **[Greenhalg e Benford, 1995]**.

O aspecto central do MASSIVE é o seu modelo espacial de interação, que tem o objetivo de facilitar a escalabilidade do AVC, no qual se utiliza o conceito de *aura* de um objeto que caracteriza a sua vizinhança imediata. Cada objeto em um mundo virtual possui uma aura específica de interação com outros objetos, que depende da mídia utilizada (por exemplo, gráficos, textos, áudio, etc.). Essa aura define uma vizinhança espacial na qual é possível a interação com outros objetos. Assim, uma aura determina quais outros objetos um determinado objeto poderá perceber visualmente, auditivamente, por meio de colisão, etc. A dimensão e forma de uma aura em princípio podem variar dinamicamente e podem ser associadas em função da posição e, possivelmente, de outros atributos do objeto, bem como o grau de visão do objeto. Define-se que a interação entre dois objetos é possível somente quando existe uma intersecção de suas auras ou um objeto se encontra dentro da aura do outro. Neste caso,

ocorre o que é denominado *colisão de auras*. A Figura 8 mostra este sistema de negociação espacial envolvendo a *aura* de cada objeto.

O sistema MASSIVE é dirigido por dois requisitos principais: escalabilidade (possibilitar o compartilhamento do ambiente entre milhares de usuários) e heterogeneidade (possibilitar que diferentes dispositivos, com diferentes capacidades, possam compartilhar o mesmo AVC). MASSIVE suporta mundos virtuais conectados através de portais. Cada mundo pode ser habitado por vários usuários que podem interagir através de combinações *ad-hoc* de interfaces de gráficos, áudio e texto. A interface gráfica renderiza objetos visíveis em um espaço tridimensional e permite ao usuário navegar neste espaço com total liberdade de movimento em todos os eixos. A interface de áudio permite ao usuário ouvir objetos e suporta tanto a execução de áudio em tempo real como sob demanda. A interface de texto provê uma visão tipo MUD através de uma janela que parece fechar-se no infinito.

Uma das características principais do MASSIVE é que estas interfaces podem

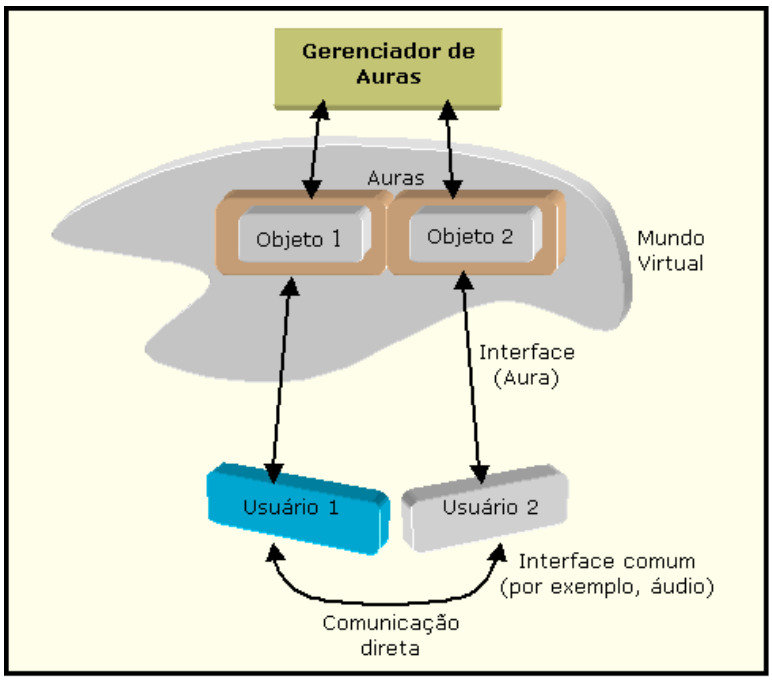


Figura 8 - Negociação espacial envolvendo pares de objetos

ser combinadas arbitrariamente de acordo com as capacidades do equipamento do usuário. Isto permite que o usuário de uma estação gráfica poderosa possa utilizar as interfaces de texto, áudio e gráfica, e compartilhar este ambiente com o usuário em uma estação menos poderosa que estaria, por exemplo, utilizando apenas a interface de texto.

2.4.5 VLNet

O sistema Rede para Vida Virtual (*Virtual Life Network – VLNet*) foi desenvolvido pela MiraLab da Universidade de Genova, pelo Laboratório de Computação Gráfica, e pelo Instituto Federal de Tecnologia Suíço. O sistema VLNet tenta integrar técnicas de vida artificial com técnicas de realidade virtual para criar ambientes virtuais realísticos, compartilhados por pessoas reais, e formas de vidas virtuais autônomas, com comportamento próprio, capazes de perceber o ambiente e interagir com os participantes **[Capin et al., 1997]**. A Figura 9 demonstra a comunicação entre processos durante a execução de um ambiente VLNet.

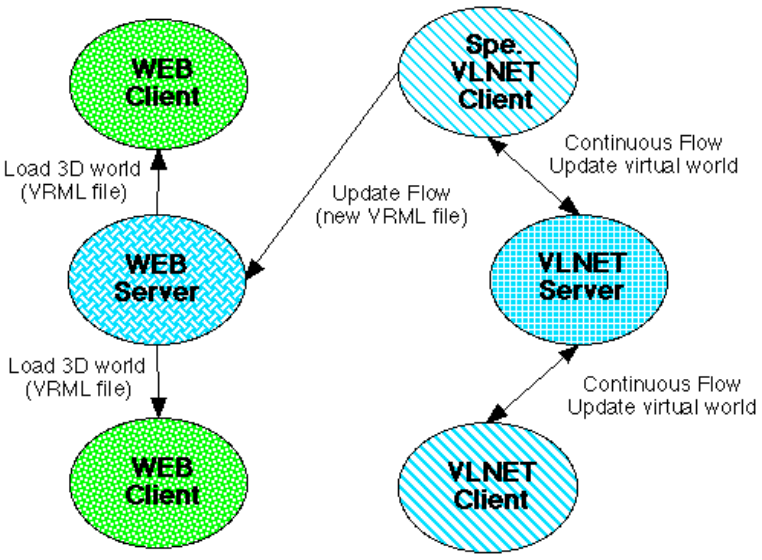


Figura 9 - Estrutura de funcionamento VLNet

O VLNet apresenta uma arquitetura modular baseada numa abordagem de multi-processos, que basicamente pode ser dividido nos processos fundamentais (*core*) e nos processos de dispositivos externos. No *core* do VLNet o processo principal executa a simulação básica e provê serviços básicos dos elementos do ambiente virtual para aplicações externas, denominadas *controladores*. O processo principal é composto por quatro motores:

- **Motor de Comportamento de Objetos:** Armazena requisições de alterações dos objetos ou pesquisa por definição e comportamento do objeto na cena, além de detectar colisão entre os objetos;
- **Motor de Navegação:** É responsável por conectar as entradas do usuário a operações de navegação, seleção e manipulação de objetos. Uma entrada consiste de uma matriz global de posições relativas ou absolutas de um objeto ou requisições de seleção ou liberação de um objeto;
- **Motor de representação facial:** Conecta-se a dispositivos externos de face que captam imagens de vídeo ou parâmetros de um modelo de face e atualiza a memória interna compartilhada do VLNet e atualiza a lista de mensagens.
- **Motor de representação corporal:** possui uma interface externa com a postura corporal, incluindo parâmetros de ângulo das juntas e posicionamento global, e parâmetros de alto nível para animação corporal. Este motor permite definirem-se vários níveis de controle sobre o corpo e combinar a saída de diferentes dispositivos corporais externos em uma única postura final.

A personagem virtual, no VLNet, é equipada com sensores audiovisuais e táteis que servem como base para implementação de qualquer comportamento humano.

2.4.6 Um framework geral para AVCs de Matijasevick e colegas

Os vários sistemas de suporte aos AVCs existentes atualmente, de maneira geral, buscam soluções para um domínio especifico de problema, pois o desenvolvimento de aplicações que possam ser utilizadas de uma forma genérica implica em um grande número de restrições. Com o intuito de minimizar estas restrições impostas aos AVCs pelos sistemas existente atualmente, Matijasevic e colegas [**Matijasevic, 2002]** propuseram um *framework* adequado para uma variedade de aplicações que incorpora aspectos de funcionalidade e comunicação comuns à maioria dos sistemas de AVCs.

A funcionalidade de um sistema, ou seja, o que o usuário faz no ambiente e como isto é interpretado pela aplicação, torna-se um aspecto decisivo quando se começa a investigar a qualidade do serviço percebida pelo usuário. A avaliação do desempenho da rede é apenas uma parte do processo completo de avaliação feito pelo usuário, por isto este *framework* considera dois aspectos:

- **Aspectos de aplicação:** como uma interação no nível da aplicação afeta os aspectos de comunicação?
- **Aspectos de comunicação:** como os eventos do nível de comunicação são refletidos no nível da aplicação?

O *framework* é ainda dividido em dois modelos: funcional e interconexão, que representam duas visões complementares e inter-relacionadas de um ambiente virtual.

O modelo funcional concebido por Matijasevick e colegas [**Matijasevic, 2002]** apresenta as funcionalidades necessárias para que uma aplicação de AVC possa realizar o propósito para o qual foi desenvolvida. A Figura 10 mostra este modelo.

Os módulos de Entrada do Usuário, Computação, Rede e Exibição compõem o modelo básico. Estes módulos são sub-divididos em módulos mais especializados para representar o funcionamento de sistemas multi-usuários. O modulo de Entrada do Usuário é dividido em Usuário–AV, que inclui navegação, objetos controlados pelo usuário e interações entre usuários, e Usuário-Grupo, que inclui as interações relacionadas à grupos de usuários.

Dentro deste modelo a computação é o processo principal. Este processo é responsável por receber as mensagens de requisição de atualizações do ambiente geradas pelos usuários, executá-las, detectar colisões, atualizar posicionamento e ponto de vista do usuário e representar graficamente todas estas alterações. As funções deste

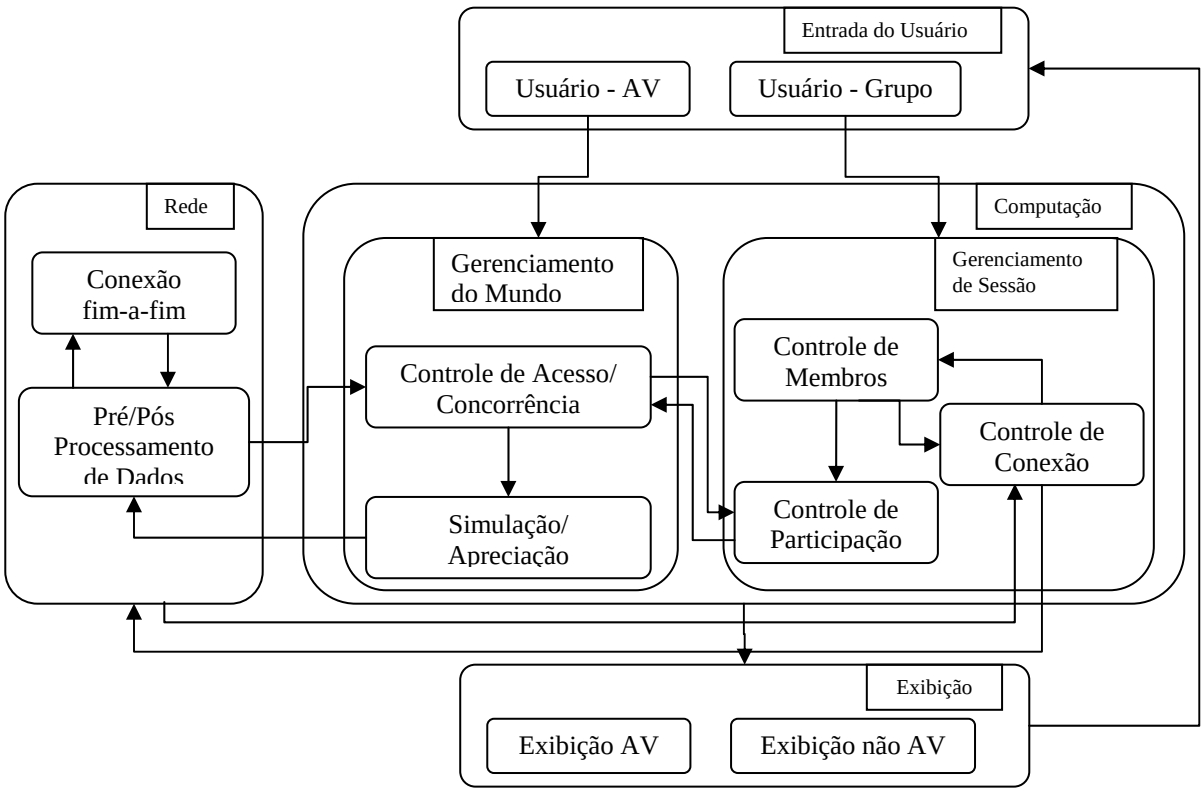


Figura 10 - Modelo Funcional de um AVC (adaptado de [Matijasevick, 2002])

módulo são separadas em Gerenciamento do Mundo e Gerenciamento de Sessão. O Gerenciamento do Mundo trata principalmente das ações dos usuários e dos efeitos que elas causam no mundo virtual, enquanto o Gerenciamento de Sessão foca-se na associação de usuários, participação e na comunicação entre os processos distribuídos que compõe o mundo virtual.

Por fim a exibição é dividida na exibição dos dados que compões efetivamente o mundo virtual, como objetos, texturas, avatares, entre outros, e dos dados que são necessário para sincronização do mundo, como *status* dos objetos e mensagens de erro e configuração.

2.4.7 Comparação entre os sistemas apresentados

Os sistemas apresentados anteriormente tendem a serem bons em tarefas específicas e não atenderem satisfatoriamente a condições não previstas no projeto inicial, limitando, desta forma, o projeto geral e, conseqüentemente, a infra-estrutura da implementação **[Oliveira et al., 2000]**. A Tabela 1 mostra um comparativo entre alguns destes sistemas.

Tabela 1 Comparativo funcional entre sistemas de suporte à AVC (Adaptado de [Matijasevic, 2002]**)**

Parâmetro	NPSNET	DIVE	MASSIVE	VLNet
Propósito	Treinamento militar	Prototipação colaborativa	Conferência colaborativa	Colaboração, jogos, shopping
Modelo de dados	Replicado	Compartilhado, distribuído	Replicado	Compartilhado, distribuído
Nº de usuários	> 1000	20	60	8
Replicação de dados	Total	Parcial	Total	Parcial
Inicialização	Base de dados de inicialização	"ponto mais próximo"	Estado transferido por agente principal	Mundo inicial
Entrada/Saída	Transparente	Transparente	Explicito	Explicito
Controle de Acesso	Nenhum	Nenhum	Leve, pesado e controle de travas	Controlado pelo servidor
Interação	Espacial (AOIM, Gerenciamento de áreas de interesses)	Espacial (aura, focus, nimbus)	Espacial (locais, limites, aspectos)	Espacial (AOIM, salas, portais)
Reconhecimento de saída	Sim	Sim	Não	Sim

Outro ponto negativo destes sistemas é que eles são intimamente ligados ao projeto para o qual eles foram definidos, e, desta forma, os critérios do projeto moldam todos os mecanismos e processos de um sistema em particular. Como conseqüência disto, torna-se difícil extrair qualquer código relevante de um sistema e adaptá-lo para outro.

Estas limitações tornam o desenvolvimento de ambientes virtuais uma tarefa dispendiosa além de praticamente impossibilitar a realização de alterações nas aplicações já existentes.

2.5. Comunicação multi-usuário em AVCs

A distribuição de usuários é uma propriedade intrínseca dos AVCs, e o controle de usuários é baseado em conceitos de comunicação em grupo, no qual um grupo corresponde a um conjunto de processos distribuídos compartilhando uma mesma sessão do ambiente. Uma das maiores preocupações em AVCs é manter-se a consistência enquanto se permite a interação de múltiplos usuários. Para o suporte a este tipo de sistema são necessárias três componentes **[Living World, 1997]**:

- **Gerenciamento de grupos:** cria e destrói processos remotos para um grupo. Opcionalmente pode manter um registro dos membros do grupo que fizeram requisições de entrada e saída;
- **Comunicação de grupos:** mecanismo para informação de trocas entre membros do grupo;
- **Gerenciamento de replicação:** utiliza a comunicação de grupos para manter a consistência entre as cópias de dados associados com os múltiplos processos de um grupo.

O gerenciamento de grupos pode ser implementado através da utilização do conceito de sessões. A replicação pode ser implementada através do conceito de zonas (ou regiões ou localidades) **[Living World, 1997][Greenhalg e Benford, 1995][Barrus et al., 1996]**. Uma sessão corresponde a uma instância do ambiente sendo compartilhada por um sub-grupo de usuários. O controle de usuários é realizado por um gerenciador de sessão, que armazena uma lista dos usuários participantes desta sessão. Usuários enviam mensagens ao controlador de sessão solicitando sua adição na lista de participantes. Caso a entrada do usuário seja autorizada, o controlador de sessão fornece uma identificação única a este usuário. Em implementações ponto-a-ponto, o controlador de sessão pode também fornecer a lista dos demais participantes.

Após o usuário ser aceito em uma sessão, o Gerenciador de Sessão comunicação com o Gerenciador de Zonas, que implementa o serviço de gerenciamento de replicação, para fornecer ao usuário todos os arquivos de representação e descrição do mundo virtual.

2.6. Limitações dos sistemas de suporte à AVC apresentados quanto à Extensibilidade

Conforme visto nas seções anteriores que descrevem os sistemas de suporte à AVCs, e nas quais são feitas comparações entre eles, a maioria não suporta o requisito não funcional de extensibilidade. Isto porque estes sistemas são fortemente sintonizados com a aplicação, tornando difícil tanto o reaproveitamento de código quanto a extensão da aplicação sem a interrupção do sistema.

2.7. Considerações Finais

Nos últimos anos, um crescente interesse tem surgido em Ambientes Virtuais Colaborativos, com aplicações na área de treinamento militar, medicina, ensino a distância, comunidades virtuais, jogos multi-usuários, dentre outras. Vários sistemas de suporte a AVCs foram implementados, dentre eles podemos citas: NPSNET, VLNet, DIVE e MASSIVE. O problema com estes sistemas é que por serem fortemente sintonizados com a aplicação, tornam-se mais inflexíveis e não atendem a um importante requisito não funcional que é a extensibilidade, que consiste basicamente da possibilidade de alterar-se um AVC durante sua execução. Como forma de solucionar, ou ao menos minimizar, esta limitação, alguns sistemas foram propostos como Bamboo e JADE. O funcionamento destes sistemas é apresentado no próximo capítulo, bem como um estudo dos padrões de suporte a AVCs para a Web e como estes padrões tratam a extensibilidade de AVCs.

3. Tecnologias de Suporte à AVCs na Web

A rede mundial de computadores (Internet) está disponível hoje para grande parte da população mundial e, por isto, tornou-se um campo promissor para aplicações de AVCs. Atualmente as tecnologias mais relevantes para este tipo de aplicações são mantidas pelo *Web3D Consortium* **[Walsh e Sévenier, 2001]**, que também é responsável por promover a interoperabilidade entre as tecnologias padronizadas por ele e entre tecnologias de infra-estrutura da Web, tais como *Dynamic HTML* (DHTML), *Document Object Model* (DOM) e *eXtensible Markup Language* (XML).

Atualmente, as tecnologias 3D para Web oferecem aos desenvolvedores a possibilidade de criar conteúdo tridimensional interativo para a WWW que pode ser apresentado em diferentes plataformas, sem considerar o mecanismo de baixo nível que renderiza os resultados experimentados pelos usuários finais. As tecnologias de suporte a AVCs na Web mais conhecidas e importantes incluem: VRML, X3D, Java 3D e MPEG4. Embora a VRML *(Virtual Reality Mark Up Language)* tenha sido altamente explorado nos últimos anos, X3D, Java3D e MPEG-4 têm surgido como formas poderosas de tecnologias 3D baseadas na WWW que, mais do que competir, complementam o VRML.

Dentre os benefícios providos pelas tecnologias 3D baseadas na Internet, é possível citar:

- Abstração dos detalhes de plataforma, como motores de renderização, *drivers* de dispositivos, placas aceleradoras, etc., que devem ser especificados pelos desenvolvedores de conteúdo quando fazem uso de APIs de baixo nível, como, por exemplo, *OpenGL* e *Direct3D*;
- Utilização de estruturas de dados que refletem o grafo da cena, em que os objetos são organizados de forma hierárquica. Essas estrutura é encontrada em tecnologias 3D de alto nível, as quais auxiliam os desenvolvedores na descrição e codificação da cena.

Uma visão geral destas tecnologias é apresentada nas seções seguintes.

3.1.1 VRML (*Virtual Reality Modeling Language*)

O VRML é uma linguagem padronizada para a construção de objetos e cenas 3D. Uma cena VRML é descrita em um arquivo de texto ASCII, que deve ser completamente interpretado por um navegador VRML antes de a mesma cena poder ser visualizada pelo usuário final.

Da mesma maneira que um arquivo HTML, um arquivo VRML é acessível em qualquer plataforma computacional, bastando ao usuário apenas ter um navegador Internet-VRML instalado.

O VRML foi a primeira linguagem de desenvolvimento de conteúdo 3D a obter status de padrão internacional.

A primeira especificação oficial surgiu em 1995 como um subconjunto de características do *Open Inventor*, da *Silicon Graphicas Inc.* (SGI), mas com novas características de rede; esta especificação descrevia como cenas 3D baseadas na Internet deviam ser construídas.

No entanto esta versão era extremamente limitada quanto à criação de mundos virtuais com qualidade de realismo e interatividade, devido ao fato de ela suportar apenas nós estáticos de descrição de cena. A especificação foi então atualizada para suportar nós dinâmicos com capacidade de proverem suporte a áudios, vídeo e animações. Estas alterações originaram uma nova versão do VRML (VRML 2.0 ou 97).

Algumas das funcionalidades providas pela arquitetura básica do VRML 2.0 são:

- **Composabilidade:** permite que várias cenas e objetos sejam compostos independentemente um dos outros e armazenados em seus próprios arquivos;
- **Escalabilidade:** permite que ambientes de tamanhos arbitrários sejam construídos com uma escala de visão que pode ser dinamicamente alterada;
- **Extensibilidade:** permite aos desenvolvedores estenderem as capacidades e características não suportadas pela especificação 2.0, como, por exemplo, suporte a AVCs.

Embora a linguagem VRML seja muito utilizada, ela apresenta várias limitações, como, por exemplo, não suportar formato de arquivo binário, programação difícil para desenvolvedores que não possuam uma boa noção de geometria, os navegadores existentes não suportam satisfatoriamente texto, áudio e vídeo, pois o texto é representado como um objeto 2D no ambiente virtual de tal forma que não possui uma representação realística, o vídeo é mapeado como textura em um objeto 3D e deve ser completamente carregado para a memória da estação local antes de ser apresentado para o usuário.

3.1.2 X3D (*Extensible 3D*)

Em 1998 a especificação do VRML 97 começou a sofrer modificações. Tais esforços foram originalmente conhecidos como *VRML Next Generation* (VRML-NG), e tiveram tal denominação modificada para *eXtensible 3D* (X3D) devido à seu intimo relacionamento com XML **[Abreu, 2001]**.

X3D é um padrão de software escalável e aberto para definição e comunicação de conteúdo 3D interativo em tempo real para efeitos visuais de modelagem comportamental. Ele pode ser utilizado em vários tipos de dispositivos e para um abrangente leque de aplicações, incluindo CAD, simulação visual, visualização médica, GIS, entretenimento, educação e apresentações multimídia **[Web3D, 2005]**.

O objetivo do X3D é disseminar conteúdo 3D por toda parte, através da Internet, Intranets e sistemas locais. Também se identifica com dispositivos de baixo custo, tais como PDAs, televisão digital, *set-up boxes* e consoles de jogo.

Embora o X3D esteja baseado em VRML ele não é simplesmente um novo caminho de disseminação de conteúdo VRML. Ao invés disto o X3D realmente realça o VRML em muitos aspectos, enquanto preserva-se compatível com a padronização desta linguagem. A utilização de NURBS (*Non Uniform Rational BSplines*) para descrição de superfícies e curvas, multi-textura, H-Anim (*Human Animation*), gráficos 2D e *streaming* de áudio e vídeo são exemplos de tecnologias adicionadas ao X3D que não estavam presentes inicialmente no VRML **[Web3D, 2005]**.

3.1.3 Java3D

Java3D é definido como sendo um conjunto completo de APIs com suporte a características gráficas 3D que estende a linguagem de programação Java para permitir aos desenvolvedores a criação de aplicações gráficas 3D de alto desempenho independentes de plataforma que são executados por uma Máquina Virtual Java (*JVM – Java Virtual Machine*) residente no terminal do cliente **[Java3D, 2005]**.

A API Java3D provê um conjunto de interfaces orientadas a objetos que suporta um modelo de programação simples e de alto nível que pode ser utilizado para construir, renderizar e controlar o comportamento de objetos gráficos tridimensionais e ambientes visuais. Com esta API é possível incorporar gráficos de alta qualidade, escaláveis e independentes de plataformas a uma aplicação baseada na tecnologia Java.

VRML/X3D e Java3D possuem similaridades e diferenças que podem ser verificadas na Tabela 2.

Tabela 2 – Similaridades e Diferenças entre VRML e Java3D (adaptado de [Abreu, 2001]**)**

Similaridades	Diferenças
Tecnologia e especificação disponíveis livremente	Criação e entrega de conteúdos 3D
Alto-Nível: sem complexidades encontradas nas linguagens 3D tradicionais (*OpenGL* e *Direct3D*)	O escopo de suas capacidades como linguagem de programação: VRML é uma linguagem de desenvolvimento de conteúdo 3D relativamente simples; Java3D é um conjunto de APIs 3D que estende a linguagem de programação Java.
Java3D pode ler arquivos VRML através de um mecanismo denominado *Loader*	VRML tem seu código fonte interpretado por um navegador enquanto que Java3D tem seu código fonte compilado no formato *bytecodes* antes de ser apresentado.

3.1.4 MPEG-4

O MPEG-4 é uma coleção integrada de tecnologias multimídia desenvolvidas pelo *Moving Pictures Experts Group* (MPEG), é um padrão ISO/IEC para a codificação e entrega de diferentes formatos de mídias sobre uma ampla variedade de redes e

plataformas de computação. Suporta vários tipos de mídias, como áudio, vídeo, texturas, conteúdos 2D/3D, etc. As mídias chegam ao seu destino separadamente (como Fluxos Elementares, que são dados elementares recebidos da saída do *buffer* de um codificador, independentemente do seu conteúdo) e são integradas, produzindo uma rica experiência multimídia no visualizador de conteúdo MPEG-4 **[N3850, 2000]**.

O MPEG-4 utiliza um formato binário para construir a cena, o BIFS – que é um completo *framework* para codificação de dados da cena em MPEG-4. A utilização de BIFS permite que mídias MPEG-4 sejam combinadas com conteúdos 2D/3D, além de tratar a interações do usuário com o conteúdo e o gerenciamento de alterações locais e remotas dos dados da cena (Comandos BIFS).

A Internet é somente um dos muitos mecanismos de entrega suportados pelo MPEG-4. Sistemas de entrega *broadcast*, tais como aqueles utilizados na entrega de conteúdos de televisão digital e *set-up boxes*, também são suportados por esta tecnologia **[Walsh e Sévenier, 2001]**. Sendo assim, o MPEG-4 pode ser visualizado em uma infinidade de aplicações, como, por exemplo, jogos de entretenimento 3D para celulares de terceira geração.

Através do BIFS o MPEG-4 estende a linguagem VRML em vários aspectos importantes, cujos principais são: compressão binária – a cena BIFS é armazenada em um formato binário comprimido que é de 10 a 20 vezes menor, em tamanho de arquivo, que a equivalente à VRML comprimida com *GZip*; combinação de mídias – integra vários tipos de mídia de forma mais satisfatória que o VRML; e compressão de áudio – capacidade de combinar sons naturais, sintéticos e efeitos espaciais.

3.2. Comparação entre as tecnologias de suporte a AVCs na WWW

Os padrões de suporte à AVCs na Web apresentados possuem pontos diferentes que caracterizam e evidenciam os seus principais aspectos. A Tabela 3 apresenta uma análise de algumas características importantes para a disseminação e desenvolvimento de ambientes virtuais com o uso dessas tecnologias 3D.

Tabela 3 - Comparação VRML/X3D, Java 3D e MPEG-4

	Ferramentas de Autoria	Existência de Plug-ins	Sincronização de Mídia
VRML/X3D	**Poucas**	**Poucos**	**Sim**
JAVA 3D	**Poucas**	**Não**	**Sim**
MPEG-4	**Uma**	**Não**	**Sim**

Os itens especificados na tabela são discutidos em detalhes a seguir:

- **Ferramenta de autoria:** As ferramentas de autoria facilitam a modelagem de objetos tridimensionais e, além de facilitarem a criação de mundos geométricos, muitas delas fornecem uma simulação do mundo virtual em tempo de criação, em que o usuário pode incluir, modificar, transladar, rotacionar, entre outras coisas, um determinado objeto com movimentos do *mouse*.

- **Existência de Plug-ins:** Um *plug-in* é uma aplicação ou um recurso específico incorporado em um *browser* tal como *Netscape*, usado para a visualização de conteúdos *VRML*, ou outros formatos específicos.

- **Sincronização de Mídia:** No contexto de multimídia, o conceito de sincronização é amplo, incluindo o planejamento do acesso a recursos compartilhados, e a especificação e o controle de atividades conjuntas de processos cooperantes. A integração temporal de diversas mídias em

uma aplicação requer a utilização de técnicas de sincronização. Mais restritamente, técnicas de sincronização são mecanismos destinados a coordenar a ordenação temporal dos eventos relacionados à aplicação.

3.3. Tratamento comum de interações nas tecnologias de suporte a AVCs na WWW

Como visto anteriormente, as tecnologias apresentadas para construção de AVCs na Web são baseadas em VRML, pois estendem suas funcionalidades. Por isto o modelo de execução de ambas é praticamente o mesmo. Este modelo é baseado em eventos, que é uma forma de encapsular um pedaço de código e transmiti-lo a um outro nó. Estes eventos podem ser gerados por nós especiais denominados *sensores*. Os sensores captam pequenos trechos de entradas que não pertençam ao padrão utilizado e as converte em eventos que podem ser disseminados pela cena gráfica **[Roehl et al., 1997]**. A Tabela 4 mostra os sensores mais comuns e suas funções.

Tabela 4 – Principais Sensores das Tecnologias de AVCs para Web (adaptado de [Roehl et al., 1997]**)**

Sensor	Descrição
CylindreSensor	Traduz as entradas do usuário em uma movimentação cilíndrica
PlaneSensor	Traduz as entradas do usuário em uma movimentação pelo plano X-Y
ProximitySensor	Detecta quando o usuário entra em uma área ao redor de um dado objeto
SphereSensor	Traduz as entradas do usuário em um formato de uma espera para rotação de objetos
TimeSensor	Detecta a hora e fornece à cena uma noção do tempo decorrido; utilizado em animações.
TouchSensor	Detecta quando o usuário toca em um determinado objeto
VivibilitySensor	Detecta se um objeto esta atualmente visível ao usuário

O fluxo dos eventos é determinado através de um mecanismo de *roteamento*. Este mecanismo relaciona um campo de entrada de um nó com um campo de saída de um outro. Desta forma, quando um evento é gerado, o valor do campo de saída é atualizado no campo de entrada. Entretanto, mesmo com as evoluções fornecidas no padrão X3D, ainda não é foco destas tecnologias a especificação de um padrão para o

gerenciamento das ações do usuário. Geralmente, é requerido que os AVCs alterem-se dinamicamente em resposta às entradas do usuário, aos eventos externos e ao estado corrente do ambiente. A proposição "*SE* a tranca está fechada *E* a senha correta foi digitada, libere a tranca" ilustra um tipo de problema que não pode ser representado em um AVC utilizando-se apenas sensores.

Para permitir o tratamento de condições como estas, as tecnologias fornecem uma interface que permite a aplicações externas acessarem e, eventualmente, alterarem o conteúdo ou a estrutura da cena gráfica sendo visualizada pelo usuário.

3.3.1 Interface de Acesso à Cena (SAI – *Scene Acess Interface*)

Originalmente, em VRML, a interface de acesso à cena era chamada EAI (*External Application Interface*). A interface EAI permite ao projetista do ambiente definir capacidades de interação com a cena não suportadas originalmente pelo conjunto de recursos disponíveis na especificação do padrão utilizado, através de *scripts* Java (javascripts) ou de aplicações externas. Esta interface é um conjunto de protocolos para manipulação da cena gráfica apesar de não fazer parte diretamente da cena em si.

A SAI é uma evolução da interface EAI. A SAI compreende uma interface comum que permite tanto a manipulação do navegador quanto a cena gráfica por uma aplicação externa ou interna à cena, por meio de um *script node* (nó de script). Entretanto, não é possível que códigos escritos para uma aplicação externa sejam diretamente usados como scripts. Os dois ambientes possuem diferentes requisitos e habilidades para acessar e interagir com a cena gráfica.

Conceitualmente a interface provê cinco tipos de acesso à cena:

- Acesso às funcionalidades do navegador;

- Recepção de notificações de ações do navegador, como endereço incorreto, início e término da simulação;
- Envio de eventos para campos com capacidades de entrada dos nós da cena;
- Leitura do último valor enviado por campos com capacidade de saída dos nós da cena; e
- Recebimento de notificações quando eventos alteram o valor de campos de algum nó da cena.

3.3.2 MPEG-J

O MPEG-J foi introduzido na segunda versão do padrão MPEG-4 e consiste de um sistema programático que especifica um conjunto de APIs (*Application Program Interface*) para o funcionamento em conjunto de visualizadores de mídia MPEG-4 com código Java, dando suporte à manipulação da cena e permitindo respostas apropriadas para os eventos da rede, do servidor ou das entradas do usuário. Por combinar mídia MPEG-4 e código executável seguro, os criadores de conteúdo podem embutir controles complexos e mecanismos de processamento de dados junto às suas informações de mídia para inteligentemente gerenciar as operações da sessão áudio-visual **[Bressan, 2002]**.

Uma aplicação MPEG-J pode ser local ou remota. Quando uma aplicação é remota, ela deve implementar a interface *MPEGLet*, da mesma maneira que ocorre com uma *applet* Java. A aplicação é distribuída através de um seguimento elementar, da mesma maneira que áudio, vídeo, BIFs de descrição da cena, etc. Isto significa que uma aplicação MPEG-J remota deverá possuir um descritor de objeto, para que possa ser

multiplexada e distribuída dentro de um arquivo no formato MPEG-4. Entre as APIs especificadas do MPEG-J, a API S*cene Graph Manager* é a mais importante no tratamento da interface do usuário com a aplicação por permitir modificar e controlar a cena gráfica **[Todesco e Zuffo, 2004]**.

3.4. Limitações dos sistemas de suporte a AVCs na Web quanto à Extensibilidade

Os padrões atuais de realidade virtual para a Web apresentados são suficientes para construir sistemas passivos de AVCs, isto é, sistemas que utilizam a tecnologia de realidade virtual para visualizar cenas pré-definidas. Uma vez que o projetista cria a cena virtual, o sistema armazena esta descrição em um arquivo com um formato pronto para ser utilizado. O usuário pode apenas visualizá-lo em seu formato original. Os objetos virtuais da cena e suas posições iniciais e atributos originais são fixos.

Os nós de script em VRML/X3D e os comandos de atualização da cena em MPEG-4 (*BIFS Command*), que podem ser utilizados por aplicações para alterar a cena através da interface MPEG-J ou SAI, suportam animações e modificações do conteúdo da cena, mas o projetista ainda precisa pré-programar estas mudanças, que são definidas quando a cena virtual é projetada. O usuário pode navegar na cena, mas esta navegação não altera a estrutura da cena, apenas o ponto de vista do usuário **[Walckzapk e Cellary, 2003]**.

Devido a esta limitação dos padrões de RV para Web, possibilitar a extensibilidade das aplicações é uma tarefa extremamente complexa, pois a codificação do ambiente deve ser totalmente carregada na memória para que, então, a aplicação possa ser executada. Sendo assim, quanto uma alteração é feita no ambiente, ele deve

ser novamente totalmente carregado para que estas alterações possam ser refletidas no AVC.

Como forma de solucionar, ou ao menos minimizar, esta limitação foram propostos os sistemas Bamboo **[Watsen e Zyda, 1998]** e JADE **[Oliveira et al., 2003]**. Estes sistemas fornecem meios de se substituir o código em execução na memória que descreve o comportamento de um objeto, sem que haja a necessidade de se interromper a execução do AVC. Estes sistemas são descritos mais detalhadamente nas próximas sessões.

3.5. Sistemas que tratam extensibilidade de AVCs

Para que um sistema atenda o requisito não funcional de extensibilidade, este deve ser baseado em uma arquitetura modular que permita a alteração de módulos individuais sem prejudicar o funcionamento global do sistema. Além disto, o sistema deve também possibilitar a reusabilidade, além de outros requisitos não funcionais de engenharia de software, descritos no capítulo 2 **[Oliveira et al., 2003]**. Tendo estas exigências em mente, foram desenvolvidos sistemas para possibilitar a extensibilidade de AVCs, tais como Bamboo e JADE.

3.5.1 Bamboo

O Bamboo, dependendo do enfoque utilizado, pode ser considerado como sendo um *toolkit*, um *framework*, ou um sistema. *Toolkits*, em geral, são APIs[1] desenvolvidas para prover funcionalidades que reduzam os esforços dos programadores. *Toolkits* que reforçam as estruturas de execução de um programa são denominados

[1] Do inglês *Application Programmer's Interface*. Consiste da especificação de funções de programação que podem ser utilizadas em um domínio de aplicação.

Frameworks. Quando um *framework* possui uma rotina principal, possibilitando sua execução em um dado equipamento, ele é chamado de Sistema. Bamboo apresenta todas esta facetas **[Watsen e Zyda, 1998]**. O projeto Bamboo foi desenvolvido pelo grupo de pesquiso do NPSNET e é utilizado no NPSNET-V para permitir extensibilidade em tempo de execução.

O mecanismo do Bamboo é provido de um componente principal (*Core*) necessário para suportar o desenvolvimento de aplicações que sejam dinamicamente extensíveis. O foco principal deste *core* é possibilitar coexistência de diferentes *plug-ins* em um sistema multi-thread. Este mecanismo possibilita a substituição do código endereçado na memória por um processo em execução.

Entretanto, possibilitar a extensão de uma aplicação em execução tem mais implicâncias do que simplesmente trazer um novo código para o mesmo endereço de memória do processo corrente. Bamboo provê também um *framework* no qual o novo código deve ser explicitamente inserido. Neste *framework* as classes implementam uma interface de *callback*, que representa um dos principais mecanismos da estrutura do Bamboo. Através deste mecanismo é possível parar ou pausar a execução de uma *thread* para que o seu conteúdo seja substituído na memória.

3.5.2 JADE (*Java Adaptative Dinamic Envrionments*)

JADE e Bamboo são duas tecnologias que se confrontam na discussão Java vs. C++. JADE é uma alternativa de implementação de uma plataforma universal para AVCs que segue os mesmos princípios de engenharia de software utilizados em Bamboo, mas que faz uso da grande variedade de funcionalidades já implementadas nas máquinas virtuais Java (*JVM – Java Virtual Machine*). A natureza multi-plataforma do Java provê o JADE com amplas capacidades de extensão, enquanto evita a utilização de

bibliotecas adicionais que, além de serem muitas vezes demasiadamente genéricas, podem ser muito incômodas **[Oliveira et al., 2000]**.

O projeto do JADE é focado em um núcleo pequeno e flexível com suporte a componentes dinâmicos de Sistemas de Realidade Virtual de Larga Escala (*LSVE – Large Scale Virtual Reality Systems*). O principal componente do *framework* JADE é denominado *module*. Este componente implementa a interface mínima para que qualquer outro componente possa ser gerenciado pelo *kernel* do JADE. Basicamente um módulo JADE pode ter duas especializações distintas: *RunnableModule*, que consiste de um módulo associado a uma *thread* distinta, e *ModuleManager*, que é basicamente um módulo especializado no gerenciamento de outros módulos.

O *kernel* do JADE é basicamente um *ModuleManager* que contém os mecanismos necessários para suportar a extensibilidade da aplicação. Para este fim, o núcleo do JADE contém vários objetos, dentre os quais os mais relevantes são:

- **Alocador de Recursos:** responsável por recuperar módulos, classes ou bibliotecas de sistemas de um local especificado, tal como o disco local, ou um endereço de um computador remoto, e liga-las dinamicamente ao *kernel* do sistema.
- **Compilador:** responsável por realizar a compilação dos módulos que são carregados dinamicamente. O resultado final desta compilação é dependente na natureza do compilador. O *kernel* do JADE possui um compilador para processar os arquivos de configuração do sistema e comandos *in-line*.
- **Gerenciador de recarga:** este objeto contém as políticas para gerenciamento de um módulo quando seu conteúdo é substituído.

O *kernel* do JADE deve ser ainda capaz disponibilizar uma interface simples de acesso, sem que seja necessário a utilização de mecanismos de passagem de referência para acesso a estas funcionalidades. Para isto, o *kernel* do JADE implementa o padrão *Singleton*, que garante que apenas uma única instância do *kernel* estará disponível dentre de uma mesma máquina virtual Java.

3.6. Considerações Finais

Um requisito não funcional importante em AVCs é o da extensibilidade, em especial quando se pensa em sistemas que devem fornecer serviço ininterruptamente.

Visando atender este requisito, foi desenvolvido o projeto Bamboo, um sistema de suporte à extensibilidade de ambientes virtuais. Por ser implementado utilizando C++ e OpenGL++, Bamboo não é inerentemente multi-plataforma. Para possibilitar que as aplicações sejam executadas em sistemas de arquiteturas diferentes, Bamboo utiliza a tecnologia NSPR (*NetScape Portable Run-time*). Considerando esta particularidade do Bamboo, Oliveira e colegas **[Oliveira et al., 2000]** propõem o projeto JADE, um sistema que, assim como o Bamboo, provê extensibilidade em tempo de execução de ambientes virtuais, mas que está baseado na tecnologia multi-plataforma Java, possibilitando que o suporte a múltiplas plataformas seja transparente ao programador.

Entretanto, a extensibilidade fornecida tanto pelo Bamboo quanto pelo JADE, é extremamente relacionada à programação. Isto exige que os especialistas no desenvolvimento de aplicações de AVCs para a WWW tenham um profundo conhecimento da tecnologia utilizada. Este requisito implica em uma limitação ainda maior para o desenvolvimento de AVCs pois, em geral, estes demandam um esforço em conjunto de projetistas gráficos (responsáveis pela modelagem do ambiente),

programadores (responsáveis pela implementação dos módulos), e conteudistas (especialistas responsáveis pelo conteúdo do AVC, que pode ditar os eventos e as possíveis interações da simulação em questão).

Por outro lado, as pesquisas na área de Drama Interativo (ID) têm mostrado um esforço no sentido de construir um modelo que seja capaz de gerir os eventos que ocorrem em uma simulação. Este modelo permite que o desenvolvimento da "estória" que ocorre no ambiente virtual durante uma simulação seja especificado independentemente da implementação da geometria dos objetos e da programação do ambiente.

No próximo capítulo os conceitos de estórias não lineares e de Drama Interativo são apresentados como uma forma de abordagem mais simples para o desenvolvimento de aplicações de AVC.

4. Descrição de AVCs através de Estórias não Lineares

As tecnologias existentes atualmente para o desenvolvimento de AVCs atendem satisfatoriamente as necessidades dos projetistas quando estes desejam criar ambientes que tenham um conjunto de ações que devem ocorrer de forma pré-definidas, isto é, quando o projetista tem um conhecimento, ao menos parcial, da ordem dos eventos que irão ocorrer no ambiente. Quando o número de usuários começa a aumentar e eles se tornam participantes mais ativos na simulação, tendo liberdade de realizar qualquer ação que desejarem, o gerenciamento de todas estas possibilidades torna-se extremamente complicado utilizando-se os recursos disponíveis atualmente.

Outro problema é que, de maneira geral, a especificação tanto da geometria do ambiente quando da *estória* dele – entenda-se por estória do ambiente a especificação do conjunto de ações do usuário que terão implicações na simulação – estão descritas em um arquivo que deve ser carregado totalmente na memória antes da simulação ser iniciada. Esta realidade tem duas implicações:

- Quando um sistema é alterado, o arquivo de especificação deve ser totalmente carregado novamente para que estas alterações possam ser contempladas; e

- O processo de alteração é complicado, devido ao fato da especificação da estória muitas vezes estar intimamente relacionada com a especificação geométrica da cena.

Mesmo diante deste problema, a comunidade acadêmica de computação não tem dado muita atenção para este aspecto nos últimos anos, sendo as pesquisas voltadas mais para distribuição e comunicação entre ambientes distribuídos. Entretanto, outras áreas têm soluções com enfoques diferenciados, mas que podem ser interessantes para o problema apresentado.

4.1. Drama Interativo

O termo Drama Interativo (ID – *Interective Drama*) pode referenciar às formas existentes de ficção interativa, jogos de computador, apresentações teatrais com a participação da platéia, etc., bem como a sistemas imersivos futuristas, como o Holodek descrito na série de TV *Star Trek*, e consiste de um drama no computador no qual o usuário é uma personagem. Ser uma personagem significa ser capaz de realizar, no mundo fictício, qualquer ação que outras personagens sejam capazes de realizar também. Esta definição mostra que ID não existe ainda, pois nos sistema de ficção interativa ou jogos baseados em estórias existentes, as personagens simuladas realizam ações que o usuário não pode realizar, então o usuário não é uma personagem, mas ele guia uma personagem através de um conjunto limitado de ações **[Szilas, 2003]**.

4.1.1 Interactive Drama Architecture (IDA)

Magerko e Laird **[Magerko e Laird, 2003]** desenvolveram uma arquitetura para drama interativo chamada IDA (*Interactive Drama Architecture*). Esta arquitetura conta com um componente central chamado *Director* que recebe o estado inicial do ambiente e a trama pré-escrita pelo autor como dados iniciais. O papel do *Director* é monitorar o comportamento do usuário e a trama, garantindo que a narrativa ocorra o mais diretamente possível. As relações básicas entre os componentes humanos (usuário e autor) e os componentes de software da IDA (*Director*) são mostrados na Figura 11.

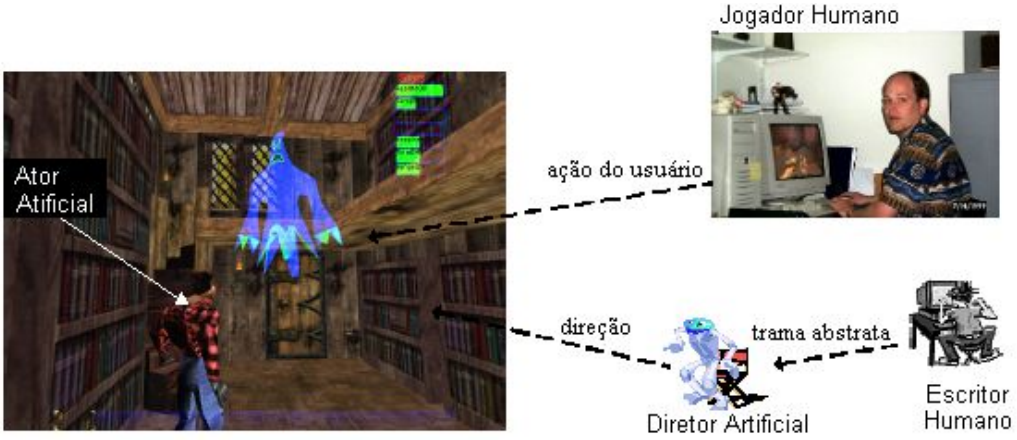


Figura 11 – Arquitetura da IDA

Magerko e Laird propõem também a utilização de um modelo baseado em estados para representação da estória, ao invés dos tradicionalmente utilizados modelos baseados em planejamentos e sistemas de geração de estória.

No modelo baseado em estados há uma diferença inerente entre representar as experiências do usuário desejadas pelo autor (o que o usuário *deve* fazer) e o comportamento do usuário (o que o usuário *quer* fazer). Cada cena da trama é representada por um gráfico de estados que é desejado que sejam alcançados. O estado final marca o fim da cena assim como o estado inicial descreve como a cena deve começar. Se o autor quiser escrever uma trama mais complexa ele deve incluir vários

estágios com o objetivo de adicionar um alto grau de variabilidade à narrativa. Cada estado possui uma transição para um outro estado, provendo uma ordem relativa da seqüência de estados em relação ao tempo. Cada transição provê informações temporais de como os eventos da trama devem fluir.

O conteúdo de cada estado será uma conexão de causas lógicas e/ou restrições, descrevendo como o mundo deve apresentar-se para o usuário.

Por fim, a arquitetura faz um tratamento do comportamento dos erros do usuário para evitar que a estória siga para um estado indesejado. Um erro pode ser visto como qualquer comportamento do usuário que não seja adequado ao contexto da estória.

4.1.2 Direcionamento prevenido da trama

Outra forma de se conseguir atingir uma narração interativa é proposta por Laaksolahti e Boman **[Laaksolahti e Boman, 2002],** através da utilização de agentes inteligentes que atuam como personagens da estória, e a representação da estória por meio de um autômato finito. Os autores descrevem o comportamento de cada personagem por meio de uma lista de possíveis ações que podem ser tomadas pela personagem para se atingir seus objetivos individuais. Quando há uma ação “falha”, a personagem realiza uma outra ação da sua lista. Uma falha pode ser vista como uma ação que a personagem pretende realizar, mas que não é possível por algum motivo, por exemplo, a personagem deseja ir ao shopping de carro, mas o carro não está na garagem. Falhas podem ser causadas pela interação do usuário ou das outras personagens.

A função do gerente da estória, como mostrado na Figura 12, é receber as informações da interface do usuário, tratá-las e refletir o estado atual da estória. O coração desta aplicação é a separação entre o gerente da estória e a interface do usuário, permitindo que vários tipos diferentes de interfaces possam ser utilizadas, como, por exemplo, apresentações em Flash®[2] e ambientes virtuais.

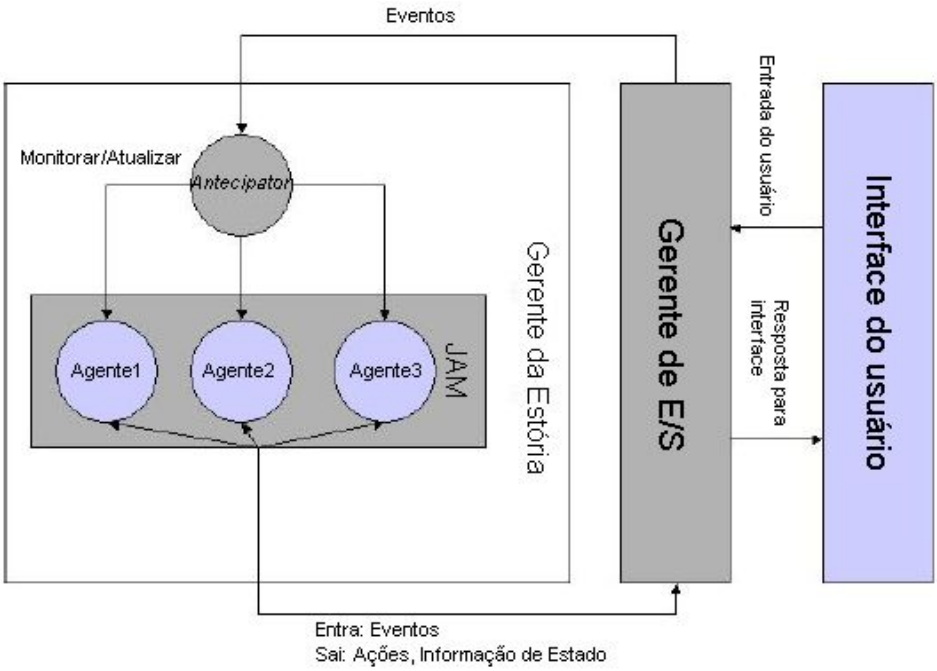


Figura 12 – Visão Geral do sistema (adaptado de [Laaksolahti e Boman, 2002]**)**

O gerenciador da estória contém todos os agentes do ambiente, incluindo o *anticipator*, cujo propósito é atuar como o diretor da estória. O *anticipator* incorpora os mecanismos de planejamento preventivo do sistema. Sua função é monitorar o progresso da estória e decidir um curso apropriado de ações que devem ser tomados pelas personagens.

A estória é modelada em um autômato finito em que cada estado corresponde a uma cena e cada transição pode ser interpretada como um conjunto de pré e pós-requisitos. Os pré-requisitos são aqueles descritos pelo estado anterior adicionado das

[2] Flash é um produto da Macromedia Inc.

condições estipuladas para que a transição seja realizada e os pós-requisitos são descritos pelo estado atual.

A alteração do estado da estória é feita por executores que alteram os valores dos dados da trama que são mantidos em uma estrutura de armazenamento como, por exemplo, um banco de dados. Os executores podem ser classificados de acordo com o escopo dos parâmetros que eles estão alterando: ambiental, global, local ou nenhum.

O processo para se modelar um drama interativo é realizado seguindo os seguintes passos:

1. Descrever o cenário inteiro como um autômato finito;
2. Listar os estados resultantes para pares de estados/transições;
3. Dividir os estados em desejáveis e indesejáveis. Estados indesejáveis são aqueles a partir dos quais a estória não pode ter prosseguimento.
4. Dividir os estados desejáveis em ordinários e finais.
5. Revisar o gráfico do autômato e repetir o processo se necessário.

4.1.3 IDtension

O IDtension é um protótipo de um sistema para drama interativo proposto por Szilas, Marty e Réty **[Szilas, 2003]** que estende a simulação para a narrativa em si. Ele se inspira na lingüística, cujo objetivo é descrever aberturas temporais de uma estrutura atemporal, ou seja, permitir que eventos gerados em um dado momento do tempo gerenciem uma narração que seja independente do tempo para acontecer.

O modelo estrutural de narrativa do IDtension, que é mostrado na Figura 13, inclui um sistema básico de regras que dita quais as ações possíveis na estória e como

elas podem ser encadeadas ao mesmo tempo. Um modelo para seqüenciamento das ações do usuário também é incluído. O sistema é composto dos seguintes módulos:

- **A narrativa lógica**: este módulo calcula as ações que são possíveis em um dado estado da narrativa. Ele é composto por cerca de quarenta regras que são genéricas para todas as narrativas. Cada regra implementa uma condição lógica para que uma ação seja possível. As ações incluem: informar, encorajar, dissuadir, aceitar, recusar, executar, cumprimentar, conceder, etc.
- **Mundo da estória**: define as entidades básicas da estória: personagens, objetos, lugares, objetivos, tarefas, sub-tarefas ou segmentos, obstáculos, estados das personagens e fatos a cerca da situação do mundo da estória (uma porta estar fechada, por exemplo).
- **Modelo do usuário**: armazena um histórico das ações e valores da personagem, as seqüências de narrativa, iniciadas, etc. Isto possibilita estimar o impacto das ações do usuário na estória.
- **Seqüenciador da narrativa**: é o "diretor" do sistema. Ele recebe possíveis ações da narrativa lógica, e pergunta ao modelo de usuário qual o impacto de cada ação, e então envia à interface de exibição a ação que será executada.
- **Interface de exibição**: é responsável por exibir as ações e gerenciar a interação entre o computador e o usuário. Atualmente a interface usada no sistema é a geração de textos básicos como saída e a seleção das possíveis ações como entrada. Futuramente a interface incluirá um navegador.

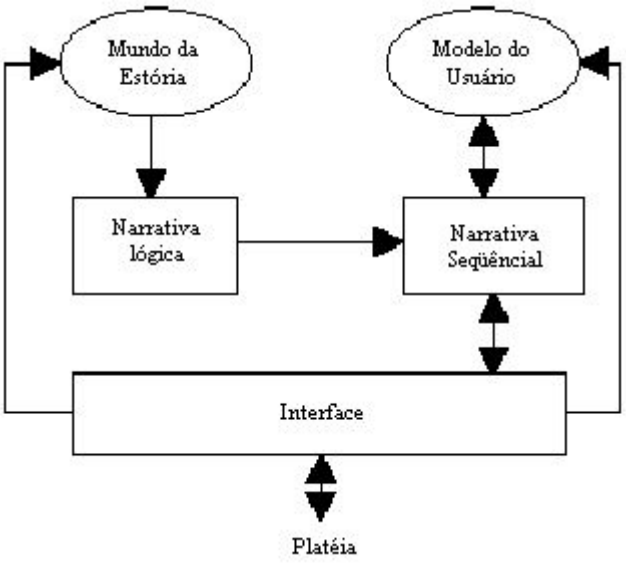


Figura 13 – Modelo da estrutura do Idtension (adaptado de [Szilas, 2003]**)**

4.1.4 NIDGE

O NIDGE (*Narrative Interactive and Dramatic Game Extension*) é um *framework* para se projetar narrativas interativas e dramáticas para jogos de computador [Kurniawan, 2001].

O fundamento do NIDGE é permitir a criação de jogos de computador com narrativas interativas e dramáticas. Para alcançar este objetivo o projeto do NIDGE é inspirado no mecanismo de produção de sonhos. Os sonhos são uma forma de narrativas nas quais o "sonhador" tem alguma quantidade de controle sobre o que acontece. Além disto, a narrativa do sonho pode parecer para o "sonhador" real e dramática durante a sua duração.

O funcionamento do NIDGE acontece na seguinte ordem:

1. As ações do jogador e o estado do universo do jogo disparam algum interesse;

2. O interesse pode desenvolver-se e formar um tema. Este tema dispara alguma narrativa armazenada no banco de dados do NIDGE com este tema;
3. NIDGE forma a narrativa do jogo (durante a sessão do jogo) mudando de uma narrativa para outra. Desta forma a narrativa completa do jogo se torna um aglomerado destes fragmentos de narrativas existentes.

O processo de troca de narrativa é transparente para o usuário para evitar que a narração seja percebida de forma estranha como pode acontecer nos sonhos.

4.2. Considerações Finais

Um sistema de realidade virtual é composto por uma grande variedade de processos, cada qual com suas funcionalidades. Entretanto a literatura atual tem dado mais ênfase a funcionalidades de comunicação, suporte multi-usuário e composição e gerenciamento da cena gráfica, deixando a área de controle das interações do usuário pouco explorada.

Trabalhos na área de Drama Interativo têm oferecido alternativas interessantes para o gerenciamento destas interações, entretanto, o foco desta linha de pesquisa é permitir o desenvolvimento de sistema que reflitam uma estória, independente do meio de narrativa (livro, TV, AVC, teatro, etc.), e que possa adaptar esta estória em decorrência das decisões tomadas pela platéia em pontos pré-definidos.

Sistemas de ID são baseados em um modelo da estória que é responsável por gerenciar a narrativa sendo apresentada na aplicação, isto é, é através do modelo da estória que o autor define os acontecimentos que irão dirigir o fluxo da narrativa. Em

narrativas interativas, ou seja, nas quais o usuário tem um papel ativo, este modelo de estória deve apresentar estruturas de controle capazes de inserir ponto no universo não linear da estória a partir dos quais eventos linearizados pré-definidos pelo autor possam ocorrer. Alternativas para estas estruturas são apresentadas em **[Kurniawan, 2001]**, **[Laaksolahti e Boman, 2002]**, **[Magerko e Laird, 2003]** e **[Szilas, 2003]**.

A utilização destes modelos de estória em AVCs permite que os projetistas destas aplicações definam pontos de interação no ambiente que podem, eventualmente, gerar eventos na simulação, especificados na estória, em decorrência das ações do usuário ou de configurações do ambiente.

O próximo capítulo apresenta uma arquitetura que possibilita a utilização de estórias interativas não lineares para definir o modelo de eventos de uma AVC. A utilização desta arquitetura possibilita a redefinição dos eventos que ocorrem no ambiente em resposta às interações do usuário, sem que haja a necessidade de interromper o funcionamento da aplicação.

5. Proposta de uma Arquitetura para AVCs que suporta Extensibilidade, através de Estórias Interativas não Lineares

Nos últimos anos as tecnologias de suporte à de Realidade Virtual evoluíram consideravelmente, impulsionadas principalmente pelas indústrias de jogos e de treinamento militar e industrial. Entretanto, a evolução das tecnologias para utilização de AVCs na Web não tem acompanhado este crescimento. Dentre os principais motivos para este acontecimento, podemos citar a dificuldade de modelagem dos ambientes, e também pelo fato de não ser fácil a reutilização de partes de um ambiente já configurado para outras aplicações, além das limitações dos padrões atualmente definidos, tais como VRML/X3D e MPEG-4. A maioria destes padrões focam apenas no desenvolvimento de sistemas que utilizam a tecnologia de RV para visualização de cenas pré-definidas, permitindo um limitado conjunto de interações. Embora seja permitida a captura de certos tipos de ações do usuário, através da utilização de sensores e de interpoladores e/ou comandos de atualização da cena para fornecer uma resposta a estas ações, eles ainda precisam ser pré-programados pelo projetista do ambiente quando a aplicação é projetada. Além disto, estes mecanismos não fornecem um suporte suficientemente abrangente para que os ambientes possam ser modificados dinamicamente em resposta às entradas do usuário, eventos externos e ao estado corrente do ambiente **[Walckzapk e Cellary, 2003]**.

Outro fator limitante para a utilização em larga escala de AVCs na Web é que a maioria dos sistemas não suporta a extensibilidade da aplicação. A extensibilidade em tempo de execução consiste de capacidade de se adicionar ou remover funcionalidades do ambiente sem ter que interromper sua execução e é um requerimento importante de AVCs, principalmente quando um provedor de conteúdos deve fornecer serviços ininterruptamente **[Boukerche et al., 2004a]**. Neste capítulo é apresentada uma solução para o problema da extensibilidade, através da utilização de estórias não-lineares para descrever o comportamento do sistema em decorrência das interações do usuário.

Foi desenvolvida uma arquitetura modular de suporte a AVCs que permite a migração de aplicações e a integração de módulos de sistemas distintos. Esta arquitetura fornece o suporte necessário para que se possa descrever o comportamento do ambiente através de estórias não-lineares. Um dos pontos fortes da utilização da abordagem de estórias para representar o modelo que descreve as reações que são geradas no sistema em decorrência das interações do usuário é a possibilidade de que cada profissional envolvido no processo de criação de uma aplicação de AVC pode trabalhar focado unicamente na sua área de conhecimento, isto é, o projetista do ambiente pode descrever a estória sem se preocupar com a especificação da geometria dos objetos nem com a programação dos eventos. O mesmo ocorre com o *designer gráfico* e com o programador da simulação.

5.1. Uma Arquitetura para AVCs Extensíveis

Como dito anteriormente, os padrões existentes atualmente para desenvolvimento de AVCs para a web focam principalmente na especificação da geometria dos objetos que compõe o mundo virtual, fornecendo poucos recursos para que seja possível se realizar um gerenciamento efetivo das ações do usuário.

Fornecer um modelo genérico para este tipo de gerenciamento não é uma tarefa trivial, pois, devido à natureza não linear que um ambiente virtual assume durante uma simulação, não é possível ao projetista prever a ordem das ações que o usuário irá tomar no ambiente. Estabelecer uma seqüência de ações tira a liberdade do usuário, diminuindo sua motivação para permanecer no ambiente.

A arquitetura apresentada na Figura 14 foi desenvolvida para suportar a utilização de técnicas de Drama Interativo - ID para descrever um modelo das ações que o usuário pode executar no ambiente e que são relevantes para reger o fluxo dos acontecimentos da simulação. A utilização destes módulos permite ao projetista do ambiente a definição das ações do usuário através do conjunto de simulações atômicas que constituem este modelo.

A arquitetura compreende os seguintes módulos: Interface do Usuário, Interpretador de Entradas, Motor da Estória, Detecção de Colisão, Motor da Cena,

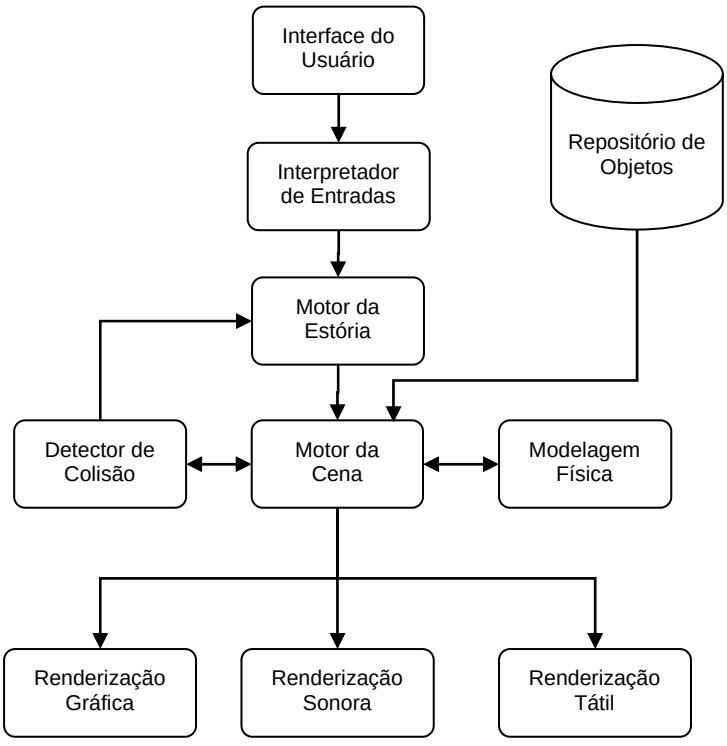


Figura 14 - Arquitetura modular para ambientes virtuais

Modelagem Física, Repositório de Objetos, além de Renderização Gráfica, Sonora e Tátil.

A **Interface do Usuário** provê uma forma intuitiva de se transmitir ao sistema as intenções do usuário de modificação do ambiente, através da movimentação de seu avatar ou da interação com os objetos ou outros usuários do AVC. O **Interpretador de Entradas** é responsável por converter as ações capturadas pela **Interface do Usuário** em comandos para o **Motor da Cena** e o M**otor da Estória**. O módulo de **Modelagem Física** é responsável por simular o comportamento dos objetos que, dentro do AVC, estejam sujeitos a leis físicas, como gravidade, ondulações, reflexão e refração da luz, entre outros. O **Motor da Cena** utiliza os serviços deste módulo para atualizar os atributos dos objetos que compõem a cena gráfica, como posição, textura, forma, etc. O módulo de **Detecção de Colisão** existe para evitar que haja sobreposição de objetos devido à movimentação do avatar dos usuários ou mesmo às suas interações com os objetos do mundo virtual. Quando uma colisão é detectada, é envida uma mensagem ao **Motor da Cena**, solicitado a atualização da posição dos objetos envolvidos. O **Repositório de Objetos** armazena representações geométricas 3D do mundo virtual como um conjunto de objetos, cada qual com um identificador único, permitindo ao **Motor da Cena** referenciar qualquer objeto sem ter conhecimento do seu conteúdo. A **Renderização Gráfica** provê uma representação visual dos objetos 3D ao usuário. Da mesma forma, a **Renderização Sonora** oferece efeitos de som e áudio no AVC, e a **Renderização Tátil** é utilizada quando existem dispositivos capazes de oferecerem ao usuário uma resposta sensitiva às suas ações. Todos estes módulos estão relacionados com o **Motor da Cena**, que utiliza seus serviços para fornecer ao usuário uma representação gráfica 3D o ambiente que possa refletir suas ações e reações correspondentes.

O **Motor da Cena** e o **Motor da Estória** são os principais módulos desta arquitetura e, por isto, são descritos mais detalhadamente nas próximas seções.

5.1.1 Motor da Cena

Em ambientes virtuais é necessário haver uma estrutura para gerenciamento da cena gráfica, que disponibilize meios de se inserir, modificar e alterar objetos. O **Motor da Cena** é a componente responsável por esta tarefa. Além de prover uma interface bastante ampla para manipulação da cena gráfica sendo apresentada, o **Motor da Cena** deve ainda manter uma estrutura, geralmente uma árvore, para referenciar os objetos da cena. Uma vez que a árvore de representação da cena gráfica está pronta, ela é decomposta para fornecer as entradas para os motores de renderização apropriados (gráfica, sonora e tátil).

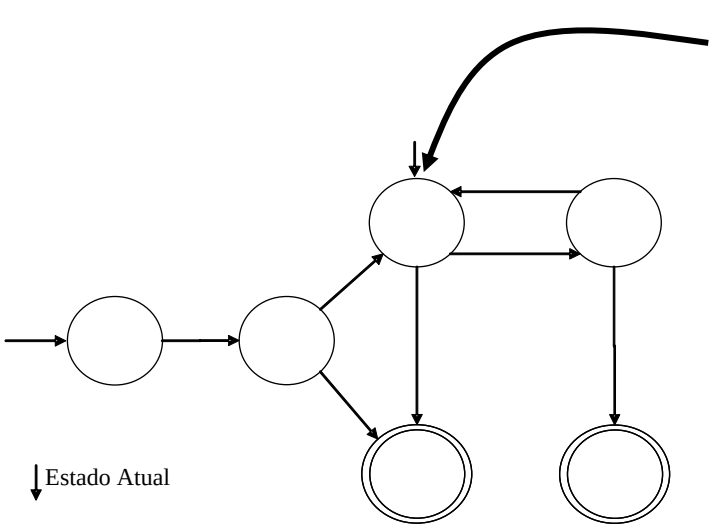
Em suma, o **Motor da Cena** é responsável por prover ao usuário uma representação gráfica que expresse as conseqüências de suas ações no AVC. Entretanto, nem todas as ações do usuário necessitam de um processamento específico do **Motor da Cena** para se calcular as alterações que serão geradas na simulação. Muitas ações podem ser simuladas através da modelagem física. Por exemplo, se o usuário segura um objeto, este objeto deve acompanhar o movimento da mão do usuário. Da mesma forma, se o usuário solta este objeto, ele deve cair, a não ser que haja sua superfície para aparálo. Naturalmente, estes fatos dependem do ambiente sendo modelado, a gravidade, por exemplo, deve ser maior em uma simulação de uma ambiente na Terra do que em uma na Lua. Mesmo assim, o modelo físico é suficiente para descrever as reações causadas pela ação do usuário nestes casos. Algumas ações do usuário, entretanto, têm um impacto na simulação que não pode ser descrito apenas com o modelo físico, como, por exemplo, em uma simulação de treinamento para se desarmar bombas, na qual o usuário

corta o fio vermelho ao invés do azul. Dispor de um módulo separado para tratar estas ações possibilita a reconfiguração da simulação sem que seja necessário alterar os demais módulos do sistema.

5.1.2 Motor da Estória

O **Motor da Estória** é o módulo responsável por tratar as interações do usuário que não possam ser modeladas por um modelo físico. A possibilidade de se tratar estas ações em um módulo separado torna possível alterar-se o tratamento destes eventos sem que haja a necessidade de se alterar também a arquitetura básica do sistema.

O **Motor da Estória** abstrai a simulação como sendo uma instância de uma estória, dividindo-a em atos, na qual cada ato narra uma parte da estória global. Internamente, ela é representada como uma máquina de estados, em que cada ato representa um estado e cada transição é um conjunto de pré-requisitos para que possa haver a mudança de um ato para outro **[Laaksolahti e Boman, 2002]**. O conjunto de requisitos é uma seqüência de configurações de um ou mais objetos e/ou personagens da simulação. Segundo Magerko e Laird **[Magerko e Laird, 2003]** cada estado pode, ainda, conter um conjunto de ações que são disparadas quando este estado é alcançado. Então, quando uma transição de estado ocorre, o **Motor da Estória** pode gerar mensagens de atualização da cena, que são enviadas para o Motor da Cena. A Figura 15 mostra um exemplo da construção de uma máquina de estados para o Motor da Estória.



Object	Attributes
Extintor	[use: no; pos: 0:150:30]
Chama	[pos: 0:115:0]
Sala	null
Bombeiro	[pos: XYZ]
Falha	null
Sucesso	null

Figura 15 - Máquina de estados armazenada pelo Motor da Estória

Para evitar que a máquina de estados esteja monitorando constantemente a configuração do ambiente para verificar se uma transição é possível, ela armazena uma tabela contendo as configurações dos objetos que são relevantes para a progressão da estória, o **Interpretador de Entrada**, gera, a partir das entradas do usuário, comandos para modificação das configurações destes objetos.

A Figura 19 mostra o modelo UML completo da arquitetura desenvolvida para possibilitar a utilização de estórias não lineares como forma de descreverem-se as ações do usuário que tem implicações diretas no fluxo da simulação e suas respectivas reações. *VEMLEventDispatcher* é a classe base que contém as operações para envio de eventos entre os módulos registrados. A classe *VEMLEvent* é uma abstração destes eventos. *InputListener*, *StoryListener* e *ColisionListener* são interfaces que contém os métodos para tratamento destes eventos. *Properties* descreve uma relação das propriedades dos objetos que podem ser alteradas. As demais classes implementam as funções dos módulos da arquitetura descrita anteriormente nesta sessão.

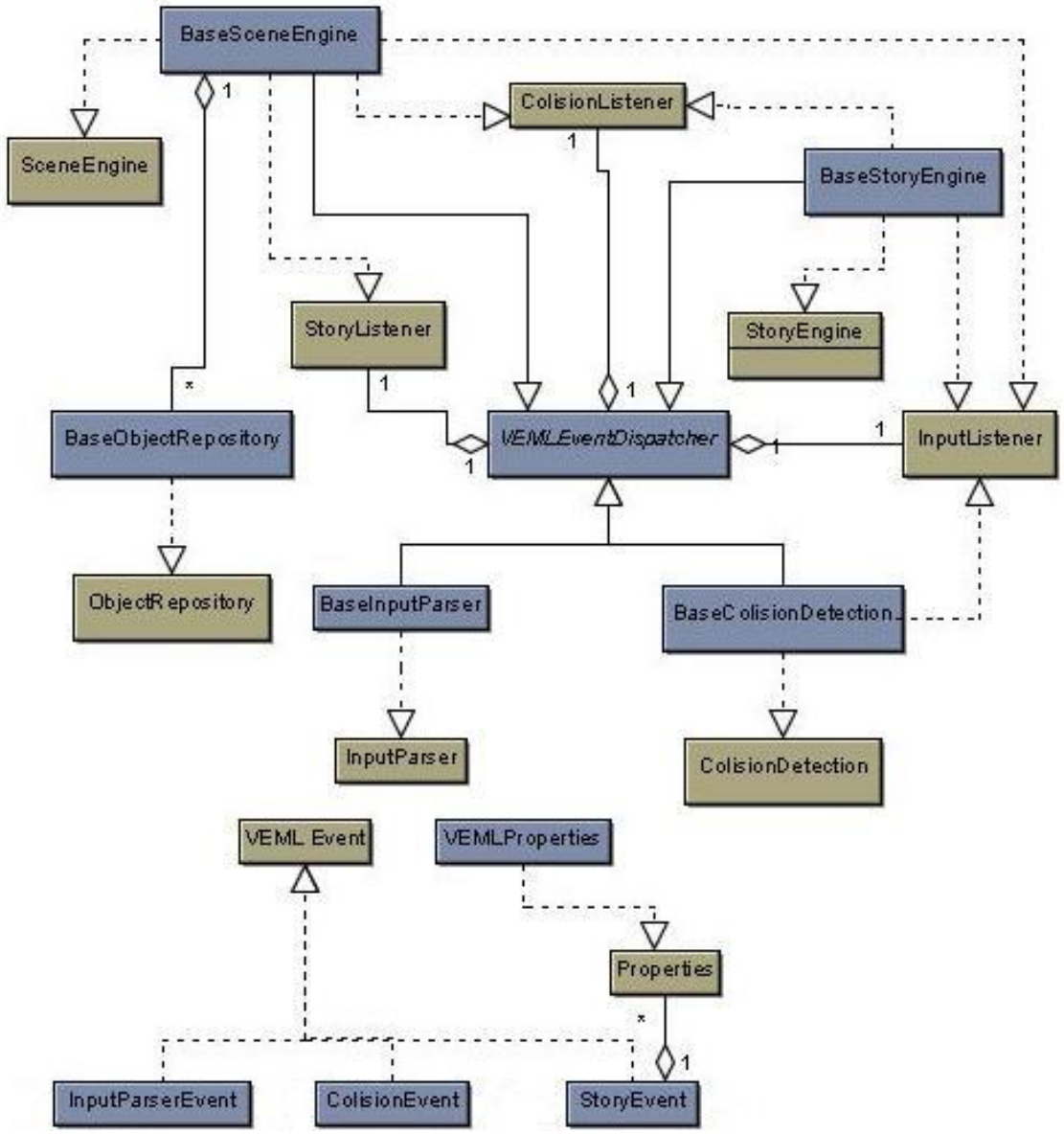


Figura 16 - Diagrama UML da Arquitetura

5.2. Considerações Finais

A utilização do conceito de estórias não lineares para possibilitar a extensão de AVCs é uma alternativa à utilização de sistemas tais como Bamboo e JADE, que são relacionados com a programação, exigindo dos desenvolvedores um profundo conhecimento do framework disponibilizado por estas tecnologias.

Este conceito permite uma separação entre a descrição geométrica do mundo virtual, a programação dos eventos e a definição do relacionamento entre estes eventos, permitindo que os profissionais responsáveis pelo desenvolvido do AVC nestas áreas possam trabalhar independentemente.

Para facilitar a descrição destas estórias não-lineares, foi criada uma linguagem de marcação, denominada VEML. Esta linguagem é apresentada no próximo capítulo.

6. VEML (Virtual Environment Mark Up Language) – Uma Linguagem de Descrição de AVCs

A Linguagem de Marcação eXtensível (*eXtensible Markup Language* – XML) é uma tecnologia que, embora não traga nenhuma revolução, disponibiliza formas de se realizar, de maneira mais prática, operações que já eram realizadas antes. A nova forma de se realizar estas operações é, ainda, portátil, independente de linguagem de programação ou sistema operacional e baseadas em padrões abertos.

Um arquivo XML possibilita a organização de informações de uma forma fácil de ser interpretada tanto para um leitor humano quanto para o computador. Este é um dos principais motivos desta tecnologia estar sendo cada vez mais utilizada em aplicações computacionais, principalmente na Web e para armazenamento temporário de informações, como, por exemplo, informações trocadas entre aplicações integradas.

Este capítulo apresenta uma linguagem de marcação baseada em XML denominada VEML (*Virtual Environment Markup Language*) capaz de representar o gerenciamento das ações do usuário em um AVC. Esta representação é feita, através do conceito de simulações atômicas, elaborado neste trabalho, e descrito a seguir.

6.1. O Conceito de Simulações Atômicas

Uma simulação em um AV é naturalmente não linear, pois não se pode dizer com antecedência a ordem em que os eventos irão ocorrer no ambiente. Esta característica dos AVCs dificulta a construção de um modelo de gerenciamento das ações do usuário, justamente por não ser possível especificar-se um tempo *t* no qual o usuário realizará uma ação *a*.

Os trabalhos de Szilas **[Szilas, 2003]** na área de Drama Interativo (ID) podem oferecer uma solução para este problema. O ID é uma forma de utilização de AVCs para representação de estória. Entretanto, estórias, ao contrário de AVCs, são lineares, ou seja, possuem uma seqüência pré-definida de eventos que ocorrem em uma ordem também definida. Szilas propõe formas de se linearizar uma estória em pontos determinados para que eventos seqüenciais possam ocorrer, e depois retornar ao estado não linearizado.

Uma forma de se inserir pontos de linearização em uma estória é a utilização de estórias atômicas. Uma estória atômica *AS* é uma tupla (*P*, *E*, *N*), na qual *P* são os eventos pré-requesitados que devem ocorrer para que *AS* seja contada, *E* são os eventos que são disparados contando-se *AS*, e *N* é a narrativa **[Szilas, 2003]**

As estórias atômicas, entretanto, estão baseadas nas interações do narrador com a platéia. Em AVCs não há narradores, apenas as interações do usuário com os objetos e outros usuários no mundo virtual. É cunhado aqui o termo “simulação atômica”, que é definido em função destas interações dos usuários. Uma simulação atômica poderia ser definida, da mesma forma que uma estória atômica, isto é, como a tupla (*P*, *E*, *S*), na qual *P* representa o conjunto de ações dos usuários e configurações do ambientes que são pré-requisitos para que a simulação aconteça; *E* é o conjunto de eventos que são

realizados no ambiente quando a simulação é iniciada; e *S* consiste da representação gráfica tridimensional destes acontecimentos dentro do AVC **[Boukerche et al., 2004b]**.

Os pré-requisitos podem ser tanto ações do usuário realizadas sobre os pontos de interação imersos no ambiente virtual, como configurações do ambiente que, eventualmente, também podem ser alteradas em decorrência das ações do usuário ou pela execução de outras simulações atômicas.

As ações disparadas pela execução de uma simulação atômica podem tanto alterar as configurações do ambiente, como as operações e/ou os atributos dos objetos que compõem o mundo virtual, tais como translação, rotação, cor, textura, forma, etc.

Permitir ao projetista do ambiente virtual especificar o tratamento das ações do usuário na forma de simulações atômicas lhe fornece um meio mais intuitivo de realizar sua tarefa, pois ele não é obrigado a passar conhecer os detalhes de funcionamento das estruturas de gerenciamento das ações do usuário fornecidas por algum padrão, tal como VRML/X3D ou MPEG-4. Além disto, facilita a manutenção e extensão dos ambientes, uma vez que a especificação deste gerenciamento é independente das especificações geométricas dos objetos, facilitando a localização das opções que devem ser alteradas. Adicionalmente, é possível estender a aplicação, através da adição de novas simulações atômicas ou do remanejamento das já existentes.

Durante o tempo de vida de uma simulação atômica, a simulação entra em um estado linear, no qual as ações pré-definidas em *E* ocorrem seqüencialmente. Após isto, ela volta para seu estado não-linear, no qual não há ações pré-definidas, somente as ações realizadas pelos usuários, as quais não têm uma ordem pré-estabelecida.

6.2. A Linguagem VEML

Como visto anteriormente, uma simulação atômica possui três componentes (a tupla (*P, E, S*) descrita na seção anterior), onde o último componente consiste da simulação em si. Uma linguagem XML poderia representar parcialmente uma simulação atômica contemplando apenas seus dois primeiros componentes, mas não a sua representação gráfica em tempo-real.

Descrever separadamente todas as simulações atômicas em XML pode não ser interessante, pois pode haver casos em que mais de um conjunto de pré-requisitos levem ao mesmo conjunto de eventos. Por exemplo, em um ambiente de shopping virtual, no qual o usuário navega em um ambiente virtual 3D, podendo fazer compras nas lojas, as ações de selecionar uma roupa, um sapato ou um perfume podem disparar um mesmo evento que debita o valor da mercadoria no cartão de crédito do cliente. Neste caso, pode ser conveniente que mais de um conjunto de condições possam estar relacionadas ao mesmo conjunto de eventos para evitar que o projetista do ambiente tenha que codificar várias vezes o mesmo conjunto de pré-requisitos. Para atender a este requisito, as simulações atômicas são subdivididas, na linguagem VEML, em “condições” e “eventos”. As “condições” representam os pré-requisitos necessários para que a simulação atômica possa ocorrer, e os “eventos” representam as ações que são disparadas no ambiente quando esta simulação ocorre. O conjunto de condições é armazenado em uma “transição”, que indica qual é o conjunto de “eventos” que deve ser disparado quando estas condições são atendidas. Desta forma, é possível ao projetista reutilizar eventos que são disparados por conjuntos diferentes de pré-requisitos.

As simulações atômicas são armazenadas pelo módulo **Motor da Estória** da arquitetura descrita no capítulo 5. Como dito, este módulo armazena uma estória abstrata para a simulação na forma de uma máquina de estados. Assim sendo, cada conjunto transição-estado da máquina pode ser visto como uma simulação atômica, uma vez que as transições representam o conjunto de configurações necessário que o ambiente virtual deve apresentar para que ela possa ser realizada (pré-requisitos) e os estados contêm um conjunto de eventos que são executados no ambiente quando ele é alcançado (ações). A linguagem obtida por esta construção é mostrada na Figura 17.

```
<simulation>
    <state number="1">
        ...
        <conditions toState="2">
            <condition name="touchSensor">
                                ...
        <condition>
    </conditions>
    ...
    </state>
    <state numver="2">
        <events>
            <event name="updateNode">
                                ...
            </event>
            ...
        </events>
        ...
    </state>
    ...
</simulation>
```

Figura 17 - Simulação atômica descrita em VEML

Cada elemento `state` corresponde a um estado da máquina. O atributo `number` define um número para este estado que é utilizado pelos conjuntos de pré-requisitos para dizer qual é o estado da máquina que contém o conjunto de eventos que devem ser disparados no ambiente. Esta associação é feita através do atributo `toState` do elemento `conditions`.

Os elementos entre as tags `<condition>` e `</condition>` definem cada pré-requisito, que é especificado pelo atributo `name`. Opcionalmente, estes pré-requisitos podem requerer algum parâmetro, que é passado pelo elemento

<parameter> do qual o atributo `value` define o valor deste parâmetro. Da mesma forma, as ações que ocorrem no ambiente quando a simulação atômica é disparada são definidas entre as tags `<event>` e `</event>`. A função dos atributos é idêntica à dos pré-requisitos.

Pode ser interessante também ao projetista do ambiente, especificar o posicionamento dos objetos dentro do mundo virtual, algum modelador que gerencie o comportamento do objeto em resposta a algum modelo matemático, qual o objeto irá representar o usuário dentro do ambiente (avatar), ou alguma outra propriedade dos objetos. Ou ainda, alterar estar informações para uma aplicação diference, mas que seja baseada na já existente. Para que isto seja possível, a linguagem proposta contém uma seção de configurações preliminares que contemplam estas informações, como é mostrado na Figura 18.

```
<declaration>
      <object name= "Sala" url= "Room.3ds" />
      <object name= "Avatar" url= "Bombeiro.3ds" avatar= "yes" />
      <object name= "Extintor" url= "Extintor.3ds" >
             <translation x= "10" y= "0" z= "15" />
      </object>
      <object name= "Chamas" url= "Fogo.3ds" modeler= "Chamas.java" />
             ...
</declaration>
<simulation>
      <scene number= "01" >
             <basis object= "Sala" >
                    <put object= "Avatar" />
                    <put object= "Extintor" />
             </basis>
      </scene>
      ...
</simulation>
```

Figura 18 - Especificação do mundo virtual em VEML

Cada elemento `object` representa um objeto do mundo virtual e possui dois atributos obrigatórios: `name`, que define um identificador único para este objeto, e `url`, que especifica o endereço do arquivo que contém a especificação geométrica deste objeto, que pode ser um código VRML/X3D ou MPEG-4, um áudio ou vídeo, ou qualquer outra representação do objeto. As propriedades `isAvatar`, `controledByUser` e `modeler` são opcionais e definem, respectivamente, se o

objeto é um avatar, se será controlado pelo usuário e a função que modela do comportamento do objeto.

Para se especificar a disposição dos objetos no mundo virtual utiliza-se as tags `<basis>` e `<put>`. `basis` define uma base sobre a qual os objetos pertencentes aos seus elementos filhos estarão posicionados e `put` posiciona um objeto nesta base. Se o objeto tiver a propriedade `translation` ou `rotation` definida ele deverá ser inserido com estas especificações relativas a base. Um tutorial completo sobre a linguagem VEML é apresentado no Anexo II.

6.3. Um Exemplo de AVC utilizando VEML

Esta seção mostra um exemplo de ambiente de treinamento de oficiais do corpo de bombeiros descrito em VEML. Primeiramente, o objetivo do treinamento consiste do combate a incêndio. O processo de desenvolvimento é realizado por três profissionais: um designer gráfico (que faz a modelagem dos objetos que irão compor o mundo virtual); um programador (que implementa os métodos de gerenciamento das ações do usuário, caso seja necessário); e um conteudista (que especifica a "estória" como uma abstração da simulação, isto é, a integração dos objetos no ambiente e o conjunto de simulações atômicas que serão responsáveis por gerar os eventos causados no sistema em decorrência das ações do usuário). Uma possível especificação VEML é apresentada na Figura 19.

No caso apresentado, o usuário está em uma sala em chamas. Nesta sala existem extintores que devem ser utilizados para se apagar o incêndio. Foram especificados dois pontos de interação neste ambiente, uma no extintor e outro nas chamas. A ação do usuário sobre o ponto de interação das chamas causa o encerramento da simulação com o usuário tendo falhado no treinamento. Este fato pode ser observado

```
<?xml version= "1.0" encouding= "UTF-8"?>
<veml xsl:link= "..." >
    <declaration>
        ...
    </declaration>
    <simulation>
        ...
        <state number= "1" >
            <conditions toState= "2" >
                <condition name= "cross" >
                        <parameter value= "Avatar" />
                        <parameter value= "Chamas" />
                </condition>
            </conditions>
            <conditions toState= "3" >
                <condition name= "touch" >
                        <parameter value= "Avatar" />
                        <parameter value= "Extintor" />
                </condition>
            </conditions>
        </state>
        <state number= "2" >
            <events>
                <event name= "show" >
                        <parameter value= "FallMsg" />
                </event>
            </events>
        </state>
        <state number= "3" >
            <conditions toState= "2" >
                <condition name= "cross" >
                        <parameter value= "Avatar" />
                        <parameter value= "Chamas" />
                </condition>
            </conditions>
            <conditions toState= "4" >
                <condition name= "use" >
                        <parameter value= "Avatar" />
                        <parameter value= "Extintor" />
                </condition>
            </conditions>
        </state>
        <state number= "4" >
            <conditions toState= "5" >
                <condition name= "near" >
                        <parameter value= "Avatar" />
                        <parameter value= "Chamas" />
                </condition>
            </conditions>
            <conditions toState= "3" >
                <condition name= "unhold" >
                        <parameter value= "Avatar" />
                        <parameter value= "Extintor" />
                </condition>
            </conditions>
        </state>
        <state number= "5" >
            <events>
                <event name= "show" >
                        <parameter value= "SucessMsg" />
                </event>
            </events>
        </state>
    </simulation>
</veml>
```

Figura 19 - Exemplo de um Arquivo VEML

analisando-se a simulação atômica formada pelo conjunto da transição `<conditions toState= "2" >` e do conjunto de eventos do segundo estado. O usuário pode realizar ações sobre o ponto de interação encontrado no extintor. Quando o usuário toca no extintor, é alterada uma configuração do ambiente dizendo que a partir deste momento, o extintor também passa a ser controlado pelo usuário. Uma vez controlado o

extintor, o usuário pode agora utilizá-lo, alterando uma outra configuração do ambiente. Agora a ação do usuário sobre o ponto de interação das chamas tem um resultado diferente, devido à nova configuração do ambiente (extintor em uso), finalizando a simulação com sucesso do usuário no treinamento. Do mesmo modo que anteriormente, todos estes fatos podem ser observados analisando-se os elementos `conditions` e os eventos disparados pelos respectivos estados apontados pelo atributo `toState`. Uma vez definida a simulação, o projetista do ambiente pode disponibilizar esta especificação em um *WebService* que fornecerá a estória e a configuração do ambiente para os usuários que desejarem realizar o treinamento. Ao receber a especificação VEML, o cliente valida o arquivo e o submete a uma aplicação intermediária, responsável por extrair as informações necessárias para cada módulo da arquitetura. Uma vez que os dados de todos os módulos tenham sido preenchidos, a nova simulação passa a ser executada.

6.4. Uma estrutura de Suporte à extensibilidade em tempo de execução

Quando uma extensão do ambiente é realizada, é necessário que haja uma sincronização entre os ambientes para se garantir a estabilidade entre todos os participantes. Para que esta sincronização seja realizada, dois componentes foram criados: apresentados na Figura 17, Controlador de Sessão (SC – *Session Controler*) e o Sincronizador de Estória e Cena (3S – *Scene & Story Synchronizator*).

A função do SC é basicamente manter as informações sobre as sessões, zonas e os usuários em cada uma delas, através do recebimento de requisições de conexão e verificando se estas têm condições para serem controladoras de sessões **[Boukerche et al., 2004b]**.

O controle de sessão é implementado através de mensagens de controle que são divididas em *control request messages*, *control replay messages* e *control command messages*. Estas mensagens são utilizadas para troca de informações entre os controladores de sessão e os participantes das sessões controladas por cada um deles.

O 3S é responsável por garantir a sincronização dos ambientes nas máquinas de cada participante de uma sessão. Quando um usuário realiza uma ação, ela é capturada pelo Interpretador de Entradas, que atualiza a configuração do ambiente do Motor da Estória. O Motor da Estória então verifica se alguma simulação atômica pode ser disparada e, em caso positivo, os eventos disparados por esta simulação são enviados para o 3S, caso contrário, é enviado ao 3S apenas o evento capturado pelo Interpretador de Entradas. O 3S, por sua vez, envia um comando de atualização para que a simulação local reflita a ação do usuário (e suas possíveis reações), verifica com o SC as máquinas que estão compartilhando a mesma seção, e envia às mesmas comandos de atualização, possibilitando que estas alterações sejam realizadas em todas as máquina que estejam dentro desta seção. O diagrama de seqüência apresentado na figura 20 ilustra este procedimento.

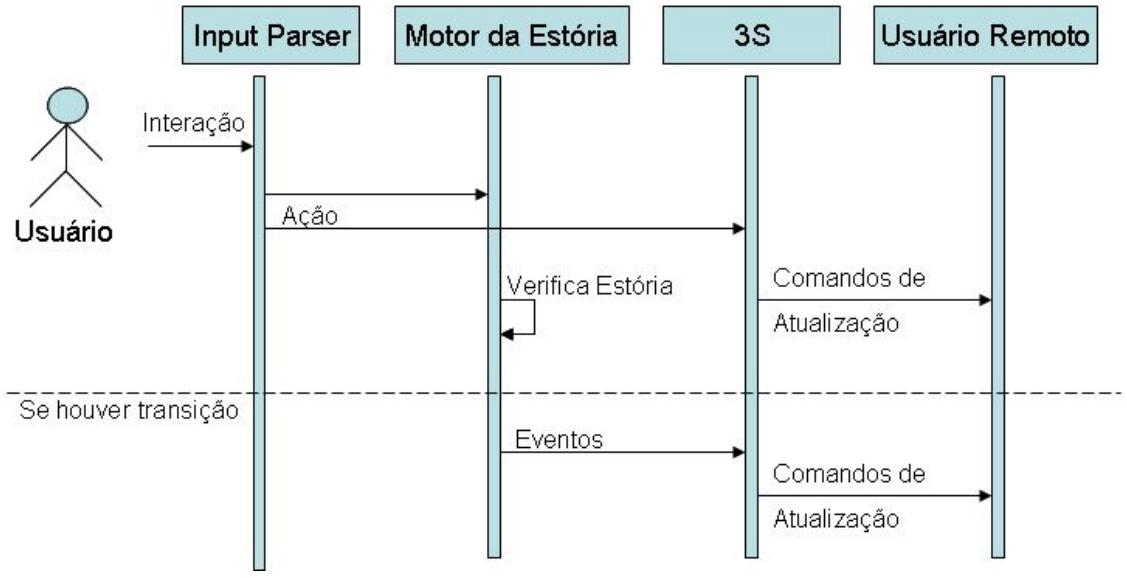


Figura 20 – Diagrama de Seqüência

Para que o usuário tenha acesso ao AVC, a especificação VEML deste deve estar disponível em um servidor de conteúdo. O navegador do usuário, ao acessar este servidor irá fazer uma solicitação da especificação do ambiente que, quando recebida, será decodificada pelo *Parser* e transmitida para Gerador de Informações dos Módulos (MIG - *Module Information Generator*) para que sejam geradas as informações necessárias a cada módulo.

MIG então faz a solicitação da entrada do usuário na sessão ao SC e a atualização da cena e da estória, através da inserção do avatar do novo usuário no ambiente. Se a entrada do usuário na sessão for possível, estes componentes então se responsabilizam por enviar notificações de atualização aos usuários que estão participando desta sessão.

A questão principal da extensibilidade em tempo de execução diz respeito aos usuários que já estejam participando do ambiente quando uma alteração é realizada, uma vez que novos usuários já estarão recebendo do servidor de conteúdo a especificação atualizada do ambiente. Para que o sistema possa manter sua consistência após uma atualização, duas abordagens distintas são possíveis:

- Criar-se uma sessão não atualizada para manter os usuários que já estavam participando do ambiente antes da atualização; e
- Enviar-se notificações de alteração aos usuários, solicitando que este atualizassem as definições locais do ambiente.

Cada abordagem é discutida com mais detalhes nas seções seguintes.

6.4.1 Sincronização através de sessões não atualizadas

A idéia desta abordagem é manter usuários antigos e novos em sessões distintas do ambiente. Esta é a solução mais simples, uma vez que não exige acesso às informações locais no terminal dos usuários.

Quando uma atualização é feita, o sistema criaria uma nova sessão *S* do ambiente que estaria visível a novos usuários, mantendo os usuários que já estivessem participando do ambiente antes da atualização em uma sessão *S'* que não seria visível. Quando todos os usuários participantes da sessão *S'* deixassem o ambiente, esta sessão deixaria de existir.

Esta abordagem apresenta três pontos que devem ser considerados:

- Os novos usuários não seriam capazes de interagir com os usuários que já estavam presentes no ambiente antes da alteração;
- O número de estruturas de controle necessárias para o gerenciamento da aplicação cresce exponencialmente em relação ao número de sessões alteradas;
- É possível que o projetista do ambiente não deseje manter usuário em uma instância não atualizada o sistema, como, por exemplo, um shopping virtual que teve sua campanha de marketing atualizada.

6.4.2 Sincronização através de notificações de atualização

Esta abordagem exige a atualização dos dados locais no terminal dos usuários participantes do ambiente. Uma vez que as alterações na especificação do ambiente estejam disponíveis no servidor de conteúdo, este enviaria uma notificação aos terminais dos usuários informando a existência de uma nova especificação. Os terminais, por sua vez, fariam uma nova solicitação da especificação ao servidor de

conteúdo. Esta especificação seria novamente decodificada pelo *Parser* e transmitida ao MIG que, neste caso, geraria apenas informações de atualização para que os módulos refletissem a nova especificação do ambiente, ao invés de gerar todas as informações, como ocorre quando o usuário acessa o ambiente inicialmente.

Estas mensagens de atualização seriam propagadas através das componentes SC e 3S, que enviariam mensagens de sincronização aos demais terminais participantes da sessão, desta forma, o servidor não precisaria manter um registro de todos os terminais participantes do ambiente, mas apenas de alguns, preferencialmente aqueles com maior capacidade de processamento. Esta implementação é semelhando à proposta por Boukerche e colegas **[Boukerche et al., 2004b]**.

Esta abordagem, entretanto, também carece de consideração em relação aos seguintes aspectos:

- A alteração do ambiente pode causar a diminuição da motivação do usuário, uma vez que pontos de interação já conhecidos podem deixar de estar presentes;
- Como é possível alterar-se também a composição da cena na especificação do ambiente, é possível que objetos sejam abruptamente inserido no ambiente, ferindo sua consistência.

Uma solução híbrida das duas abordagens propostas anteriormente parece ser a que mais atende aos requisitos de extensibilidade em tempo de execução.

Uma vez que alterações na especificação VEML fossem realizadas, o sistema criaria uma nova instância do ambiente para abrigar os usuários participantes do ambiente neste momento e seria enviada a estes usuários uma mensagem visual de que o sistema foi alterado. Desta forma os usuários que quisessem participar do ambiente mais

atualizado deveriam realizar uma ação explicita que seria interpretada como a confirmação desta decisão, como, por exemplo, passar por um portal no ambiente.

O projetista poderia também definir um tempo limite de vida para que a instância não atualizada do sistema, que seria informado aos usuários, caso as alterações sejam prioritárias.

6.5. Estudo de Caso: Homo Anonimous

Homo Anonimous é um projeto do Departamento de Artes da Universidade Federal de São Carlos (UFSCar) que busca defrontar a pessoa com uma nova realidade vivida no cotidiano das grandes cidades, em que cada indivíduo forma sua personalidade única, ou baseada em padrões comportamentais adotados por uma comunidade, ou pela aversão a estes mesmos padrões.

O projeto prevê a construção de uma instalação composta por cinco cômodos separados entre si por cortinas, possibilitando a transição entre cômodos em qualquer ponto da instalação. Em cada cômodo é apresentada uma composição de imagens, sons e objetos que caracterizam certo estereótipo comportamental. Cada cômodo contém também um conjunto de atores que realizam interpretações relacionadas ao tema comportamental escolhido para o cômodo. As ações dos participantes em um cômodo têm reflexão nos componentes e na interpretação dos atores em outros cômodos.

Com o intuito de promover a visitação da instalação mencionada anteriormente, é previsto também no projeto a especificação de um AVC no qual um grupo de usuários possa interagir entre si e, através destas interações e da experimentação do ambiente, construir sua personalidade. O ambiente virtual deve adaptar-se segundo as opções de personalidade feitas por cada usuário.

Como estudo de caso foi implementada uma parte deste ambiente: a Loja de Bonecos. Nesta loja o usuário tem a possibilidade de "comprar" um boneco que, segundo ele, melhor represente a sua personalidade. Dependendo da escolha feita pelo usuário, algumas opções de interação no ambiente são mantidas, outras são retiradas e outras ainda podem ser adicionadas. Neste teste inicial, o ambiente altera seu padrão de cor para a cor do boneco selecionado. A codificação completa em VEML do ambiente é apresentada no Anexo I.

Quando o usuário toca em um dos bonecos, ele é elevando verticalmente, para representar que esta é a possível opção de aquisição feita pelo usuário. Tocar no mesmo boneco faz com que ele retorne à sua posição original. Tocar em outro objeto altera a seleção, elevando o novo objeto e retornando o anterior à sua posição original. A Figura 211 mostra o ambiente para este caso.

Figura 20 - Seleção do boneco verde no ambiente Homo Anonimous

Quando o usuário toca no vendedor, após ter selecionado um boneco, ele confirma sua opção de aquisição do boneco selecionado. O boneco então é removido do ambiente e o usuário não pode mais interagir com os demais bonecos. A Figura 222 mostra um exemplo de configuração do ambiente ao selecionar-se o boneco verde.

A codificação do AVC Homo Anonimous através da linguagem VEML possibilita que o ambiente possa ser alterado, como, por exemplo, pela inserção de novas personalidades, alteração das implicações da escolha de uma personalidade, sem que seja necessária a interrupção do funcionamento do ambiente. A Figura 233 mostra uma extensão do AVC para disponibilizar ao usuário mais uma opção de personalidade, que se reflete na adição de mais um boneco.

6.6. Considerações Finais

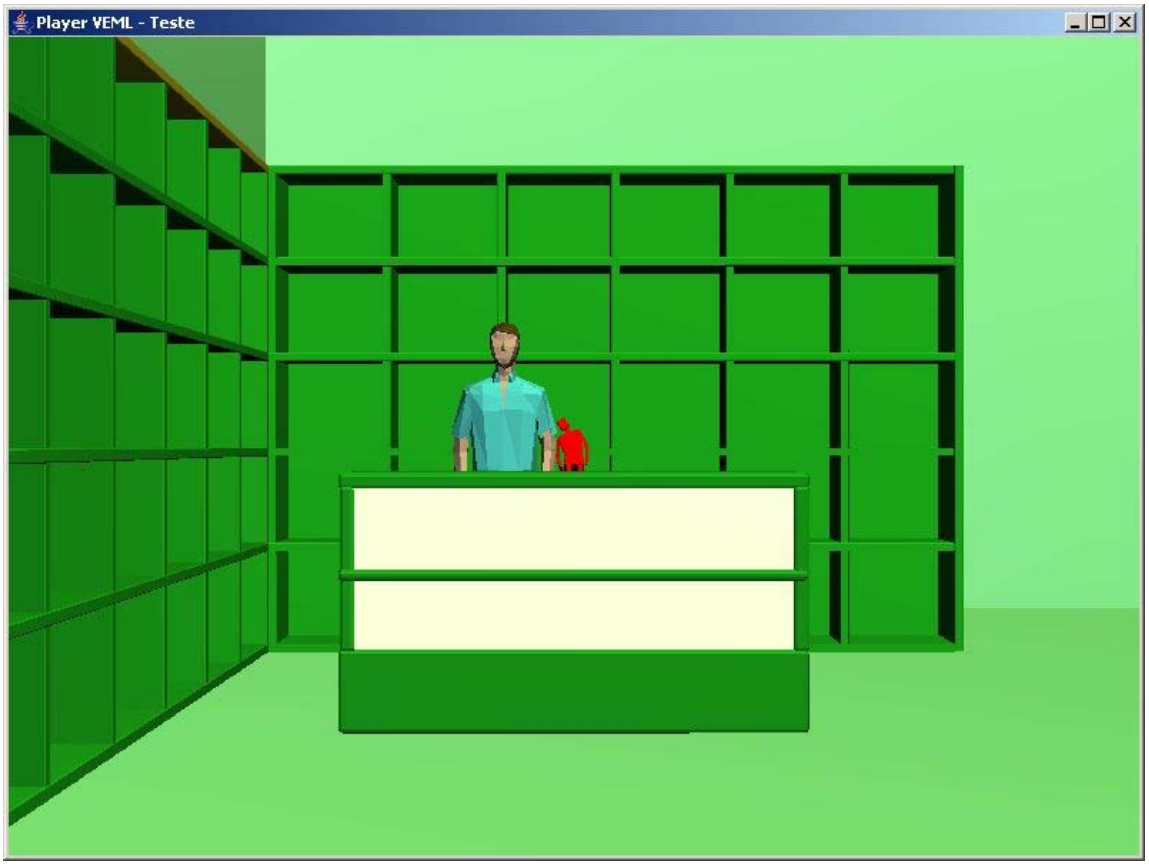


Figura 21 - Ambiente Homo Anonimous após seleção do boneco verde

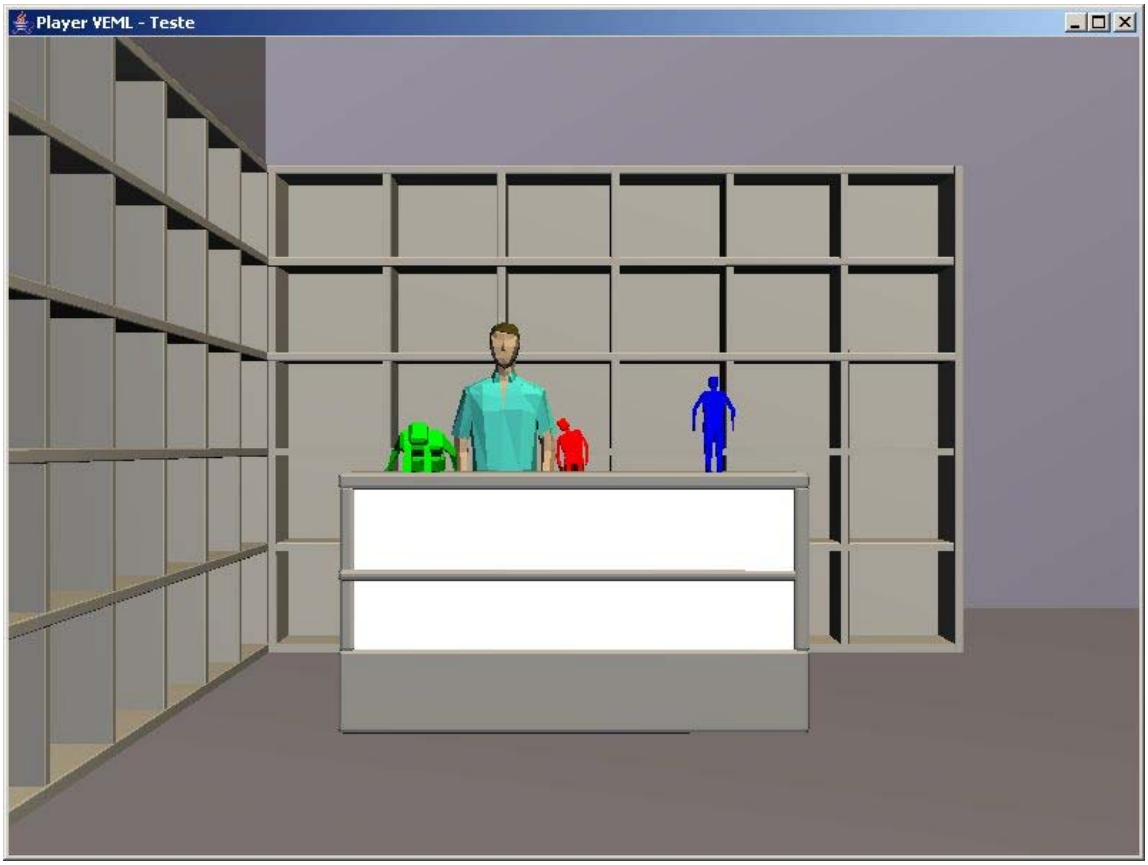


Figura 22 - Homo Anominous estendido para três opções de personalidade

Aplicações de AVCs, sobretudo da Web, têm sido cada vez mais aceitas e utilizadas. Entretanto, o tempo gasto no desenvolvimento deste tipo de aplicações é um fator que ainda inibe a utilização, em maior escala, desta tecnologia. Além disto, muitas das soluções existentes atualmente são intimamente ligadas à aplicação, gerando três problemas básicos: (1) A arquitetura de suporte não possibilita aplicações fora do domínio tratado por ela; (2) A utilização de uma aplicação em uma arquitetura diferente, ou de parte de uma arquitetura em outra, é inviável e muitas vezes impossível; (3) Alteração na aplicação muitas vezes se reflete em alterações na arquitetura de suporte.

Outra limitação para este tipo de aplicação é a carência de mecanismos eficazes que possibilitem a extensão do ambiente em tempo de execução. Em grande parte destas aplicações, senão todas, é desejável que o sistema permaneça no ar ininterruptamente. Os padrões existentes atualmente não permitem que alterações sejam realizadas nos

sistema sem que seja necessário interromper o seu funcionamento, pois a especificação do sistema é salva em um formato de arquivo pronto para ser utilizado, que é interpretado pela aplicação de visualização apenas no momento em que o ambiente é carregado.

Neste trabalho foi apresentada a utilização de uma arquitetura que permite a utilização de simulações atômicas para gerenciamento das interações do usuário juntamente com componentes de Controle de Seção e Sincronização da Estória e Cena para permitir que seja realizada extensão em tempo real de um AVC de modo a garantir seu funcionamento mesmo durante a atualização destas mudanças.

7. Conclusões

A utilização de Ambientes Virtuais Colaborativos teve um crescimento nas últimas décadas impulsionado principalmente pela indústria de jogos de computador e treinamento militar e industrial. Entretanto, mesmo com o investimento destes setores, a construção de AVCs em larga escala ainda está distante da realidade. Isto porque os sistemas de suporte a AVCs existentes atualmente são desenvolvidos para atender a finalidades muito específicas, impossibilitando o desenvolvimento de ambientes que requeiram características não previstas no projeto inicial e inviabilizando a extensão destas aplicações.

Mesmo os sistemas que possibilitam a extensão de AVCs em tempo de execução dependem muito do conhecimento dos programadores, que tratam a extensibilidade através da substituição do código do processo em execução que controla o comportamento de um ou mais objetos do mundo virtual.

Quando o foco das aplicações passa a ser a WWW, esta carência de sistemas de suporte se torna ainda maior. Os padrões de suporte a AVCs na WWW existentes, disponibilizam uma interface de acesso à cena para possibilitar que aplicações externas possam alterar dinamicamente a simulação. Entretanto, esta interface exige que os atributos ou as propriedades dos objetos que serão controlados por esta aplicação sejam mapeados na área de dados da aplicação quando o ambiente é iniciado. Esta restrição

impossibilita, por exemplo, a inserção de novos objetos que estejam também sob o controle da aplicação.

Visando atender o requisito não funcional de extensibilidade de Ambientes Virtuais Colaborativos na Web, foi apresentada, neste trabalho, uma solução que utiliza o conceito de estórias interativas não-lineares como forma de descrever AVCs. Este tipo de estórias já vem sendo utilizado na área de Drama Interativo para possibilitar a geração autônoma de narrativas a uma platéia. Ao se descrever as ações do usuário que terão um impacto decisivo no decorrer da simulação e de suas respectivas reações, através de uma estória interativa não-linear, torna-se possível alterar completamente uma simulação substituindo-se apenas esta estória, sem a necessidade de envolver profissionais capacitados para realizar a reprogramação do ambiente, além de permitir que o sistema continue em execução durante a realização das alterações, o que é uma vantagem de suma importância para servidores de aplicações que exigem um funcionamento ininterrupto durante todos os dias do ano.

Para possibilitar a utilização da solução proposta, foi apresentada uma arquitetura modular que pode ser adaptada para funcionar com diferentes tipos de visualizadores. Esta arquitetura possui módulos com funcionalidades bem definidas, dentre os principais, os módulos responsáveis pela captura e interpretação das ações do usuários, pelo gerenciamento da estória não linear, pelo armazenamento dos objetos que compõem o mundo virtual.

A descrição da estória é feita, através de uma linguagem de marcação denominada VEML *(Virtual Environment Mark Up Language)*. Esta linguagem possui um perfil mínimo que deve ser implementado por todos os interpretadores. A utilização desta abordagem de perfis permite a compatibilidade da linguagem com implementações distintas da arquitetura. Para auxiliar o projetista de AVCs no processo

de desenvolvimento da estória, está sendo desenvolvida uma ferramenta de autoria em VEML (editor gráfico de estórias).

A solução proposta possibilita a extensibilidade de AVCs com um alto grau de independência de programação, facilidade de integração com outros padrões de AVCs para Web, representação do AVC e codificação de seus eventos de forma tal que é de fácil interpretação tanto pelo desenvolvedor do ambiente quanto pela máquina (já que utiliza o conceito de estórias codificadas, através de uma linguagem de marcação baseada em XML) e com foco principal para a WWW. Devido a estas vantagem, a utilização de VEML pode ser uma alternativa promissora para a codificação e extensão de AVCs na Web.

7.1. Contribuições Geradas

Este trabalho teve como resultado, até o momento, as seguintes publicações:

Boukerche, A., Araujo, R. B. and Duarte, D. D. - 3D Virtual Environments Extensibility Based on Interactive non-Linear Stories In: The 8-th IEEE International Symposium on Distributed Simulation and Real Time Applications - DS_RT´2004, Budapest, 21-23 October, 2004.

A. Boukerche, D. D. Duarte and R. B. Araujo - VEML: A Mark Up Language to describe Web-Based Virtual Environment through Atomic Simulations. In: The 8-th IEEE International Symposium on Distributed Simulation and Real Time Applications - DS_RT´2004, Budapest, 21-23 October, 2004 (short paper).

Duarte, D. D., Boukerche, A., Araujo, R. B. A Language for Building and Extending 3D Virtual Environments in the Web In: WebMedia/LA-Web 2004, Ribeirão Preto. Published at the IEEE Proceedings of the Joint Conference of the Second Latin American Web Congress and 10th Brazilian Symposium on Multimedia and the Web (LA-Webmedia 2004).

7.2. Trabalhos Futuros

Os seguintes trabalhos futuros deverão ser realizados:

- Aprimoramento da estrutura de suporte à extensibilidade em tempo-real (controle de sessões e zonas);
- Especificação e implementação de um editor gráfico de estórias interativas não lineares em VEML;
- Construção de um estudo de caso completo que compreende o treinamento de oficiais do corpo de bombeiros, através de AVCs representados em VEML (atualmente já existe uma parceria entre o LRVNet - Laboratório de Realidade Virtual em Rede, no qual este trabalho foi desenvolvido, e o Corpo de Bombeiros de São Carlos em projeto relacionado a monitoramento de ambientes cientes de contexto para aplicações de segurança crítica).

Referências Bibliográficas

[Abreu, 2001] Abreu, S. *A Tecnologia MPEG-4 no Suporte a Ambientes Virtuais Multiusuário*, dissertação de mestrado do PPG-CC, Universidade Federal de São Carlos, São Carlos-SP, Julho de 2001.

[Araújo e Oliveira, 1998] Araújo, R.B., Oliveira, M.A.M.S., *Ambientes 3D Interativos Multiusuário na WEB: Uma Avaliação Realista*, I SICON – Simpósio de Sistemas e Controle, Rio de Janeiro, Setembro, 1998, Anexo I, pp.19-26.

[Barrus et al., 1996] J.W. Barrus, R.C. Waters and D.B. Anderson. *Locales and Beacons: Efficient and Precise Support for Large Multi-User Virtual Environments*. IEEE Computer Graphics and Applications, 16(6):50-57, November 1996.

[Begault, 1994] Begault, D.R. *3D Sound for Virtual Reality and Multimedia*, Academic Press, Cambridge, MA, 1994.

[Boukerche e Araújo, 2004] Boukerche, A. and Araújo, R.B. titulo, submetido à Parallel Simulation, 2004.

[Boukerche et al., 2004a] Boukerche, A., Araújo, R.B., Laffranchi, M.M., *A Hybrid Solution to Support Multiuser 3D Virtual Simulation Environments in Peer-to-Peer Networks*, Proceedings of the Eighth IEEE International Symposium on Distributed Simulation and Real-Time Application (DS-RT 2004), p. 61-68. Budapest, Hungria, 21-23, Outubro, 2004.

[Boukerche et al., 2004b] Boukerche, A., Duarte, D.D., Araújo, R.B., *VEML: A Mark Up Language to Describe Web-Based Virtual Environment through Atomic Simulations*, In. DISTRIBUTED SIMULATION AND REAL-TIME APPLICATIONS, 2004. DS-RT 2004, Eighth International Symposium on, p. 214-217, Budapest, Hungary, 2004.

[Bressan, 2002] Bressan, C.M. *Uma Contribuição ao Padrão Internacional Emergente MPEG-4 Multiusuário*, dissertação de mestrado do PPG-CC, Universidade Federal de São Carlos, São Carlos-SP, Janeiro de

2002

[Bryson, 1994] Bryson, E., *Invited Talk – Virtual Environments in Scientific Visualization*, Virtual Reality Software & Technology, Proceedings of the VRST'94 Conference, p. 201-220, Singapura, 23-26 Agosto, 1994.

[Burdea e Coiffet, 1994] Burdea, G. C., Coiffedt, P. *Virtual Reality Technology*. Editora John Wiley & Sons, 1994.

[Capin et al., 1997] Capin, T.K., Noser, H., Thalmann, D., Pandzic, I.S., Thalmann, N.M., *Virtual Human Representation and Communication in VLNet*, IEEE Computer Graphics and Applications, vol. 17, ed. 2, p. 42-53, Março-Abril, 1997.

[Capps et al., 2000] Capps, M., McGregor, D., Brutzman, D., Zyda, M.J., *NPSNET-V. A new beginning for dynamically extensible virtual environments*. IEEE Computer Graphics and Applications, vol. 20, ed. 5, p. 12-15, Set.-Out., 2000.

[Carlsson e Hagsand, 1993] Carlsson, C., Hagsand, O., *DIVE A multi-user virtual reality system*, IEEE Virtual Reality International Symposium, 18-22 Setembro, 1993.

[Coscarelli, 1998] Coscarelli, C. V. *O uso da informática como instrumento de ensino-aprendizagem*. Presença Pedagógica. Belo Horizonte, março/abril, 1998, pp. 36-45.

[Deering, 1991] Deering, S., Multicas routing in a datagram internetwork, Ph.D. dissertation, Stanford University, Dezembro, 1991.

[Frecon, 2004] Frecon, E., *DIVE: communication architecture and programming model*, IEEE Communications Magazine, vol. 42, ed. 4, p. 34-40, 2004.

[Frery, 2003] Frery, A., Kelner, J., *Realidade Virtual e Multimídia*. Disponível: site Centro de Informática da Universidade Federal de Pernambuco (1998). URL: http://www.cin.ufpe.br/~if124, consultado em 16/12/2003.

[Greenhalg e Benford, 1995] Greenhalg, C., Benford, S., *MASSIVE: a distributed virtual reality system incorporating spatial trading*, Proceedings of the 15th International Conference on Distributed Computing Systems, p. 27-34, 30 de Maio à 2 de Junho, 1995.

[Hagsand, 1996] Hagsand, O., Interactive Multiuser VE in DIVE systems, IEEE Multimedia, vol. 3, no. 1, pp. 30-39, 1996

[Java3D, 2005] Java3D Project Home, on-line: https://java3d.dev.java.net/, visitado em Janeiro de 2005.

[Kurniawan, 2001] Kurniawan, S. *Overview of NIDGE: A Framework to Desing Narratively Interactive and Dramatic Videogames*, Proceedings in PACLING 2001

[Laaksolahti e Boman, 2002] Laaksolahti J., Boman, M. *Anticipatory Guidance of Plot.*
In proceedings of Workshop of Adaptive Behaviour in Anticipatory Learning Systems (ABiALS) 2002 Edinburgo, Escócia

[Living World, 1997] Living Worlds Proposal Draft 2. On-line: http://www.vrml.org/WorkingGroups/livingworlds/draft_2/
index.html, Abril, 1997.

[Magerko e Laird, 2003] Magerko, B., Laird, J. E. *Building an Interactive Drama Architecture*, 1ª Conferência Internacional de Interactive Storytelling e Entretenimento, Darmstadt, Alemanha, 24 – 26 de Maio de 2003

[**Matijasevic, 2002]** Matijasevic, M., Gracanin, D., Valavanis, K.P., Lovrek, I., *A Framework for Multiuser Distributed Virtual Environments*, IEEE Transactions on Systens, Man, and Cybernetics – Part B: Cybernetics, Vol 32, No. 4, Agosto, 2002.

[Miller e Thorpe, 1995] Miller, D.C. e Thorpe, J.A., *SIMNET: The advent of simulator network*, Proceedings IEEE, v. 83, p. 1114-1123, Agosto, 1995.

[N3850, 2000] Information Technology – Coding of audio-visual objects – Part 1: Systems, ISO/IEC JTC 1/SC 29/WG 11 – 14496-1, Outubro, 2000.

[Oliveira et al., 2000] Oliveira, M., Crowcroft, J., Slater, M., *Component Framework Infrastructure for Virtual Environment*, ACM Collaborative Virtual Environments, p. 136-146, 2000.

[Oliveira et al., 2003] Oliveira, M., Crowcroft, J., Slater, M., *An Inovative Design Approach to Build Virtual Environments Systems*, Internationa Immersive Projection Technologies Workshop. Proceedings of the Workshop on Virtual Environments 2003, Zurich, Suiça, 2003.

[Pausch, 1996] Pausch, R., Snoddy, J., Taylor, R., Watson, S., Haseltine E. *Disney's Aladdin: First Steps Toward Storytelling in Virtual Reality*, Proceedings in SIGGRAPH'96, Nova York, NY, USA, 1996, pp. 193-203

[Roehl et al., 1997] Roehl, B., Couch, J., Reed-Ballreich, C., Rohaly, T, Brown, G., *Late Night VRML 2.0 with Java*, Emeryville, Calofornia, editora Ziff-Davis Press, ISBN 1-56276-504-3, 1997.

[Schneider, 2002] Schneider, O. *Storyworld creation: Authoring for interactive storytelling*. In Proceedings of the 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Plzen-Bory, República Tcheca, 4 à 8 de fevereiro de 2002

[Schneider, 2003] Schneider, O., Braun, N., Habinger, G. *Storylining Suspense: An Authoring Environment for Structuring Non-Linear Interactive Narratives*, Journal of WSCG, Vol. 11, Plzen, República Tcheca, 3 a 7 de Fevereiro de 2003

[Shin-Ting e Malheiros, 2002] Shin-Ting, W. e Malheiros, M.G., *Interactive 3D Geometric Modelers with 2D UI*, Proceedings of The 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Plzen-Bory, República Tcheca, 4 à 8 de fevereiro de 2002

[Singhal e Zyda, 1999] Singhal, S. e Zyda, M. *Networked Virtual Environments: Design and Implementation*, (Siggraph Series), ACM Press, New York, 1999.

[Szilas, 2003] Szilas, N., Marty, O., Réty, J. *Authoring highly generative Interactive Drama*, Proceedings in International Conference on Virtual Storytelling, Toulouse, França, 20 a 21 de Novembro de 2003, pp. 37-46.

[Todesco e Zuffo, 2004] Todesco, G., Zuffo, M., *MPEG-4: Teoria e Prática*, Tópicos em Tecnologia Web & Multimídias, WebMedia & LA-Web 2004 Joint Conference, p. 1-38, Ribeirão Preto, Setembor, 2004.

[Todesco, 2000] Todesco, G. *Criação de Ambientes Virtuais Multiusuário através do Padrão MPEG-4*. Dissertação de Mestrado, PPG-CC, Universidade Federal de São Carlos, São Carlos-SP, 2000.

[Walckzapk e Cellary, 2003] Walckzapk, K. and Cellary, W. “X-VRML for Advanced Virtual Reality Applications”, IEEE Computer, Volume 36, Issue 3, page 89-92, March 2003

[Walsh e Sévenier, 2001] Walsh, A. e Sévenier, M., *Core Web3D*, editora Prentice Hall PTR, ISBN 0-13-085728-9, 2001

[Watsen e Zyda, 1998] Watsen, K. e Zyda, M., *Bamboo – A Portable System for Dynamically Extensible, Real-Time, Networked, Virtual Environments*, Proceedings for the 1998 IEEE

Virtual Reality Annual International Symposium (VRAIS'98), Atlanta, Estados Unidos, 14 à 18 de março de 1998.

[Web3D, 2005] Web3D Consortium, on-line: http://www.web3d.org, visitado em Janeiro de 2005.

[Zyda, 1996] Zyda, M.J., *Networking large-scale virtual environments*, Proceedings Computer Animation' 96, p. 1-4, 3-4 Junho, 1996.

Anexo I: Especificação VEML do ambiente Homo Anonimous

```
<?xml version="1.0" encoding="UTF-8" ?>

<veml profile="minimal">
    <declaration>
        <object name="Loja" url="..\\..\\Exemplos\\Loja\\loja.wrl">
            <transformation>
                <translation axisX="140" axisY="-37" axisZ="-100" />
            </transformation>
        </object>
        <object name="Vendedor" url="..\\..\\Exemplos\\Loja\\vendedor.wrl">
            <transformation>
                <translation axisY="-10" axisZ="-200" />
            </transformation>
        </object>
        <object name="Bancada" url="..\\..\\Exemplos\\Loja\\bancada.wrl">
            <transformation>
                <translation axisY="-20" axisZ="-180" />
            </transformation>
        </object>
        <object name="Estante" url="..\\..\\Exemplos\\Loja\\estante_l.wrl">
            <transformation>
                <translation axisY="5" axisZ="-180" />
            </transformation>
        </object>
        <object name="Boneco1" url="..\\..\\Exemplos\\Loja\\boneco_caido.wrl">
            <transformation>
                <translation axisX="-20" axisY="6" axisZ="-185" />
            </transformation>
        </object>
        <object name="Boneco2" url="..\\..\\Exemplos\\Loja\\boneco_dormindo.wrl">
            <transformation>
                <translation axisY="2" axisZ="-176" />
            </transformation>
        </object>
        <object name="Boneco3" url="..\\..\\Exemplos\\Loja\\boneco_em_pe.wrl">
            <transformation>
```

```
                <translation axisX="18" axisY="7" axisZ="-176" />
            </transformation>
        </object>
    </declaration>
    <simulation>
        <scene number="1">
            <basis object="Loja">
                <put object="Vendedor" />
                <put object="Bancada" />
                <put object="Estante" />
                <put object="Boneco2" />
            </basis>
            <state number="1">
                <events>
                    <event name="veml.actions.MoveTo">
                        <parameter value="-20" />
                        <parameter value="6" />
                        <parameter value="-185" />
                        <parameter value="Boneco1" />
                    </event>
                    <event name="veml.actions.MoveTo">
                        <parameter value="0" />
                        <parameter value="2" />
                        <parameter value="-176" />
                        <parameter value="Boneco2" />
                    </event>
                    <event name="veml.actions.MoveTo">
                        <parameter value="18" />
                        <parameter value="7" />
                        <parameter value="-176" />
                        <parameter value="Boneco3" />
                    </event>
                </events>
                <conditions toState="2">
                    <condition name="veml.conditions.Touch">
                        <parameter value="Boneco1" />
                    </condition>
                </conditions>
                <conditions toState="3">
                    <condition name="veml.conditions.Touch">
                        <parameter value="Boneco2" />
                    </condition>
                </conditions>
                <conditions toState="4">
                    <condition name="veml.conditions.Touch">
                        <parameter value="Boneco3" />
                    </condition>
                </conditions>
            </state>
            <state number="2">
```

```
            <events>
                <event name="veml.actions.MoveTo">
                    <parameter value="-20" />
                    <parameter value="16" />
                    <parameter value="-185" />
                    <parameter value="Boneco1" />
                </event>
                <event name="veml.actions.MoveTo">
                    <parameter value="0" />
                    <parameter value="2" />
                    <parameter value="-176" />
                    <parameter value="Boneco2" />
                </event>
                <event name="veml.actions.MoveTo">
                    <parameter value="18" />
                    <parameter value="7" />
                    <parameter value="-176" />
                    <parameter value="Boneco3" />
                </event>
            </events>
            <conditions toState="1">
                <condition name="veml.conditions.Touch">
                    <parameter value="Boneco1" />
                </condition>
            </conditions>
            <conditions toState="3">
                <condition name="veml.conditions.Touch">
                    <parameter value="Boneco2" />
                </condition>
            </conditions>
            <conditions toState="4">
                <condition name="veml.conditions.Touch">
                    <parameter value="Boneco3" />
                </condition>
            </conditions>
            <conditions toState="5">
                <condition name="veml.conditions.Touch">
                    <parameter value="Vendedor" />
                </condition>
            </conditions>
        </state>
        <state number="3">
            <events>
                <event name="veml.actions.MoveTo">
                    <parameter value="-20" />
                    <parameter value="6" />
                    <parameter value="-185" />
                    <parameter value="Boneco1" />
                </event>
                <event name="veml.actions.MoveTo">
```

```
                <parameter value="0" />
                <parameter value="12" />
                <parameter value="-176" />
                <parameter value="Boneco2" />
            </event>
            <event name="veml.actions.MoveTo">
                <parameter value="18" />
                <parameter value="7" />
                <parameter value="-176" />
                <parameter value="Boneco3" />
            </event>
        </events>
        <conditions toState="2">
            <condition name="veml.conditions.Touch">
                <parameter value="Boneco1" />
            </condition>
        </conditions>
        <conditions toState="1">
            <condition name="veml.conditions.Touch">
                <parameter value="Boneco2" />
            </condition>
        </conditions>
        <conditions toState="4">
            <condition name="veml.conditions.Touch">
                <parameter value="Boneco3" />
            </condition>
        </conditions>
        <conditions toState="6">
            <condition name="veml.conditions.Touch">
                <parameter value="Vendedor" />
            </condition>
        </conditions>
    </state>
    <state number="4">
        <events>
            <event name="veml.actions.MoveTo">
                <parameter value="-20" />
                <parameter value="6" />
                <parameter value="-185" />
                <parameter value="Boneco1" />
            </event>
            <event name="veml.actions.MoveTo">
                <parameter value="0" />
                <parameter value="2" />
                <parameter value="-176" />
                <parameter value="Boneco2" />
            </event>
            <event name="veml.actions.MoveTo">
                <parameter value="18" />
                <parameter value="17" />
```

```
                <parameter value="-176" />
                <parameter value="Boneco3" />
            </event>
        </events>
        <conditions toState="2">
            <condition name="veml.conditions.Touch">
                <parameter value="Boneco1" />
            </condition>
        </conditions>
        <conditions toState="3">
            <condition name="veml.conditions.Touch">
                <parameter value="Boneco2" />
            </condition>
        </conditions>
        <conditions toState="1">
            <condition name="veml.conditions.Touch">
                <parameter value="Boneco3" />
            </condition>
        </conditions>
        <conditions toState="7">
            <condition name="veml.conditions.Touch">
                <parameter value="Vendedor" />
            </condition>
        </conditions>
    </state>
    <state numbe="5">
        <events>
            <event name="remove">
                <parameter value="Boneco1" />
            </event>
            <event name="margeChildrensColor">
                <parameter value="Loja" />
                <parametar value="0" />
                <parametar value="1" />
                <parametar value="0" />
            </event>
        </event>
    </state>
    <state numbe="6">
        <events>
            <event name="remove">
                <parameter value="Boneco2" />
            </event>
            <event name="margeChildrensColor">
                <parameter value="Loja" />
                <parametar value="1" />
                <parametar value="0" />
                <parametar value="0" />
            </event>
        </event>
```

```
            </state>
            <state numbe="7">
                <events>
                    <event name="remove">
                        <parameter value="Boneco3" />
                    </event>
                    <event name="margeChildrensColor">
                        <parameter value="Loja" />
                        <parametar value="0" />
                        <parametar value="0" />
                        <parametar value="1" />
                    </event>
                </event>
            </state>
        </scene>
    </simulation>
</veml>
```

Anexo II: Tutorial VEML

VEML (*Virtual Environment Markup Language*) é uma linguagem de marcação baseada em XML (*eXtensible Markup Language*) definida para a especificação de Ambientes Virtuais Colaborativos (AVCs). Esta linguagem encapsula definições das geometrias dos objetos que compõe o mundo virtual através da metáfora de utilização de um repositório para estes objetos, que são referenciados através de uma *alias* na descrição do ambiente.

Neste breve tutorial são apresentadas a estrutura da linguagem e as possibilidades de construção e extensão de AVCs que ela possibilita.

1. Estruturas básicas da linguagem

Uma especificação VEML é composta por duas partes principais: a definição dos objetos que compõe o mundo virtual e a especificação dos pontos de interação do ambiente, das ações que o usuário deve realizar sobre eles para ativá-los e dos eventos que são disparados no ambiente mediante a ativação de um ponto de interação. O conjunto ponto_de_interação-ações-eventos é denominado “estória do AVC”. Esta denominação deve-se ao fato deste modelo estar baseado em estórias não-lineares, que são estórias em que são definidos apenas pontos chaves da estória em que a platéia pode interagir e, em virtude destas interações, o narrador define o novo rumo da estória.

A especificação de objetos é demarcada pela *tag* `<declaration>`. Dentro desta *tag* são declarados os objetos que compõe o mundo virtual, suas propriedades e atributos.

A especificação da estória do AVC é definida pela *tag* `<simulation>`. Nesta *tag* são definidos os estados nos quais o AVC pode estar durante sua execução. Cada estado pode possuir um conjunto de eventos que são realizados no ambiente quando ele é alcançado. Para que seja realizada a transição de um estado para outro o usuário deve realizar ações pré-definidas sobre pontos de interações também definidos. Todos estes aspectos são especificados dentro da *tag* `<declaration>`.

1.1. *Declaração de objetos*

A declaração dos objetos que compõe o mundo virtual é feita em VEML através da *tag* `<object>`. A apresentação mínima de uma declaração de objetos é:

```
<object name="Casa" url=http://localhost/vrml/casa.wrl />
```

Neste exemplo está sendo declarado um objeto que é referenciado no ambiente pelo *alias* **Casa**. Como se pode perceber, a especificação do *alias* é feita pela definição da propriedade `name` de `object`. Da mesma forma, a propriedade `url` define o local onde se encontra a especificação da geometria deste objeto.

1.1.1 Definindo um Avatar

Avatar é um objeto que representa o usuário dentro de AVC. Para especificar que um objeto ira assumir esta função, deve-se configurar a propriedade `isAvatar` de `object` como verdadeira

```
<object name="Avatar" url="./man.wrl" isAvatar="true" />
```

1.1.2 Definindo quem controla um objeto

Dentro de um AVC geralmente os objetos encontram-se imóveis em suas posições pré-definidas, sendo movidos apenas em decorrência de ações específicas do usuário sobre eles. Algumas vezes, entretanto, pode ser interessante que alguns objetos não apresentem esta configuração.

Uma configuração alternativa para o controle do objeto é possibilitar que este modele algum comportamento, como, por exemplo, uma fonte, as folhas de uma árvore ao vento, etc. A especificação desta configuração é feita definindo-se o modelador do comportamento do objeto pela propriedade `modeler`.

```
<object name="Tree" url="tree.wrl" modeler="wind.class" />
```

Outra possível configuração é permitir que o objeto movimente-se juntamente com o usuário. Esta configuração é feita através da definição da propriedade `controledByUser` como verdadeira.

```
<object name="Roupa" url="ts.wrl" controledByUser="true" />
```

1.1.3 Definindo propriedades do objeto

Dentro de `object` é possível definir-se as propriedades do objeto sendo declarado. O perfil mínimo da linguagem VEML contempla apenas as propriedades de translação de rotação. Perfis mais avançados podem contemplar outros tipos de propriedades previstos em padrões como VRML e X3D, tais como textura, escala, cor, etc.

Translação e rotação são especificadas, respectivamente, pelas *tags* `<translation>` e `<rotation>`, que são definidas dentro da *tag* `<transformation>`. O valor de cada uma destas propriedades é definido

separadamente para cada eixo através da configuração dos atributos `axisX`, `axisY` e `axisZ` para, respectivamente, os eixos X, Y e Z. Caso alguma dessas propriedades seja omitida, o sistema atribui automaticamente o valor 0 (zero) para ela.

```
<object name="Mesa" url="./mesa.wrl" >
    <transformation>
        <translation axisX="10" axisZ="-120">
        <rotation axisY="1.17">
    </transformation>
</object>
```

Os valores para `<translation>` são dados na unidade de medida adotada pelo ambiente. Os para `<rotation>` são dados em radianos.

1.2. *Especificação da Estória*

Após declarar os objetos que irão compor o mundo virtual, o projetista do ambiente deve configurar as cenas, definindo a relação entre estes objetos, e definir os pontos de interação e as ações-reações relacionados a eles.

Um AVC completo pode ser comporto por várias cenas relacionadas entre si, por isto, cada subconjunto da estória é definido em uma cena. Uma cena é compreendida pelo objetos pertencentes à *tag* `<scene>`.

```
<simulation>
    <scene>
        ...
    </scene>
</simulation>
```

1.2.1 Inserindo objetos no ambiente

Há duas formas de inserir-se um objeto já declarado no ambiente virtual sendo projetado. A primeira é inserir diretamente no mundo virtual, desta forma o objeto será inserido nas coordenas absolutas (X, Y, Z) especificadas na declaração. Neste caso a

inserção do objeto é feito pela *tag* `<put>`. A propriedade `objet` deve ser configurada com o *alias* do objeto que se deseja inserir.

```
<scene>
    <put object="Tree" />
</scene>
```

Outra forma é inserir um objeto que pertença a outro objeto previamente inserido no ambiente. Neste caso o objeto será inserido nas coordenadas especificadas relativas à posição do objeto base. Para realização desta forma de inserção deve-se primeiramente definir qual será o objeto base. Este processo é feito através da utilização da *tag* `<basis>`, que tem a mesma estrutura de `<put>`.

```
<scene>
    <basis object="Casa">
        <put object="Mesa" />
    </basis>
</scene>
```

Todos os objetos adicionados entre as *tag* `<basis>` serão inseridos no ambiente dentro do objeto especificado no atributo `object`. É possível também fazer um aninhamento de uma ou mais bases.

1.2.2 Definindo os estados do AVC

Uma cena pode assumir vários estados durante a execução do AVC. Cada estado pode conter um conjunto de eventos que são executados no ambiente e um conjunto de transições para próximos estados. Cada transição é composta por um conjunto de condições que devem ser mutuamente verdadeiras para que a mudança de estado possa ocorrer.

Estados são especificados pela *tag* `<state>`. O atributo `number` é utilizado para referenciar-se a este estado em uma transição.

```
<scene>
```

```
        ...
        <state number="1">
            ...
        </state>
    </scene>
```

Caso o estado contenha um conjunto de eventos a serem executados no ambiente, estes eventos são definidos dentro da *tag* `<events>`.

```
<scene>
    ...
    <state number="1">
        <events>
            ...
        </events>
    </state>
</scene>
```

As transições para próximos estados são definidos dentro da *tag* `<conditions>`. O atributo `toState` permite especificar-se qual o estado destino da transição.

```
<scene>
    ...
    <state number="1">
        <events>
            ...
        </events>
        <conditions toState="2">
            ...
        </conditions>
    </state>
</scene>
```

1.2.3 Definido os eventos e condições

A forma de definição dos eventos e das condições para mudança de estado adotadas em VEML é um dos principais meios para possibilitar a fácil extensão do ambiente. Um evento consiste de um trecho de código de um programa que, quando executado, recebe como parâmetro as informações da ação do usuário sendo executada e dos objetos recipientes desta ação.

A definição deste evento é feito pela *tag* `<event>`, que possui um atributo `name` para definir o nome do processo responsável pela execução deste evento. Os valores para inicialização do processo são passados pelos atributos `value` das *tags* `<parameter>`.

```
<events>
    <event name="veml.events.MoveTo">
        <parameter value="Mesa" />
        <parameter value="10" />
        <parameter value="0" />
        <parameter value="-120" />
    </event>
</events>
```

A definição das condições segue a mesma linha geral de raciocínio.

```
<conditions toState="2">
    <condition name="veml.conditions.Touch">
        <parameter value="Mesa" />
    </condition>
</conditions>
```

Tanto para a definição de um evento quando para definir uma condição o projetista do ambiente deve unicamente conhecer a API do processo responsável por sua execução.

Adicionalmente, uma condição pode ser configurada para que seja realizada uma transição não apenas de estado, mas também de cena. Desta forma, todo o ambiente é substituído pela nova cena e o estado atual do AVC passa a ser o estado iniciar desta cena. Este processo é feito substituindo-se o atributo `toState` de `<conditions>` por `toScene`.

```
<conditions toScene="2">
    ...
</conditions>
```

2. Criando eventos e condições de VEML em Java

Como dito anteriormente, tanto os eventos como as condições definidas na estória de um AVC utilizando VEML são processos responsável por exercer alguma operação, ou de alteração de cena gráfica, no caso dos eventos, ou de verificação de atributos e propriedades dos objetos, no caso das condições.

O *toolkit* do VEML disponibiliza um conjunto de classes Java© que, além de fornecerem a arquitetura que dá suporte à utilização desta tecnologia, define alguns eventos e algumas condições padrões.

Via de regra, todo evento implementa a interface *action* e toda condição implementa a interface *condition*. Para implementar um novo evento ou uma nova condição, o programador deve conhecer a API do *toolkit* VEML.

Anexo III: Editor VEML

Por ser baseada em XML, a linguagem VEML possibilita uma fácil interpretação tanto pelo leitor humano quanto pelo computador. Entretanto o processo de especificação de um AVC diretamente na linguagem é custoso e desgastante.

De modo a facilitar a criação de estórias não-lineares em VEML para descrição de AVCs foi projetado um editor VEML que permite a criação/alteração de estórias,

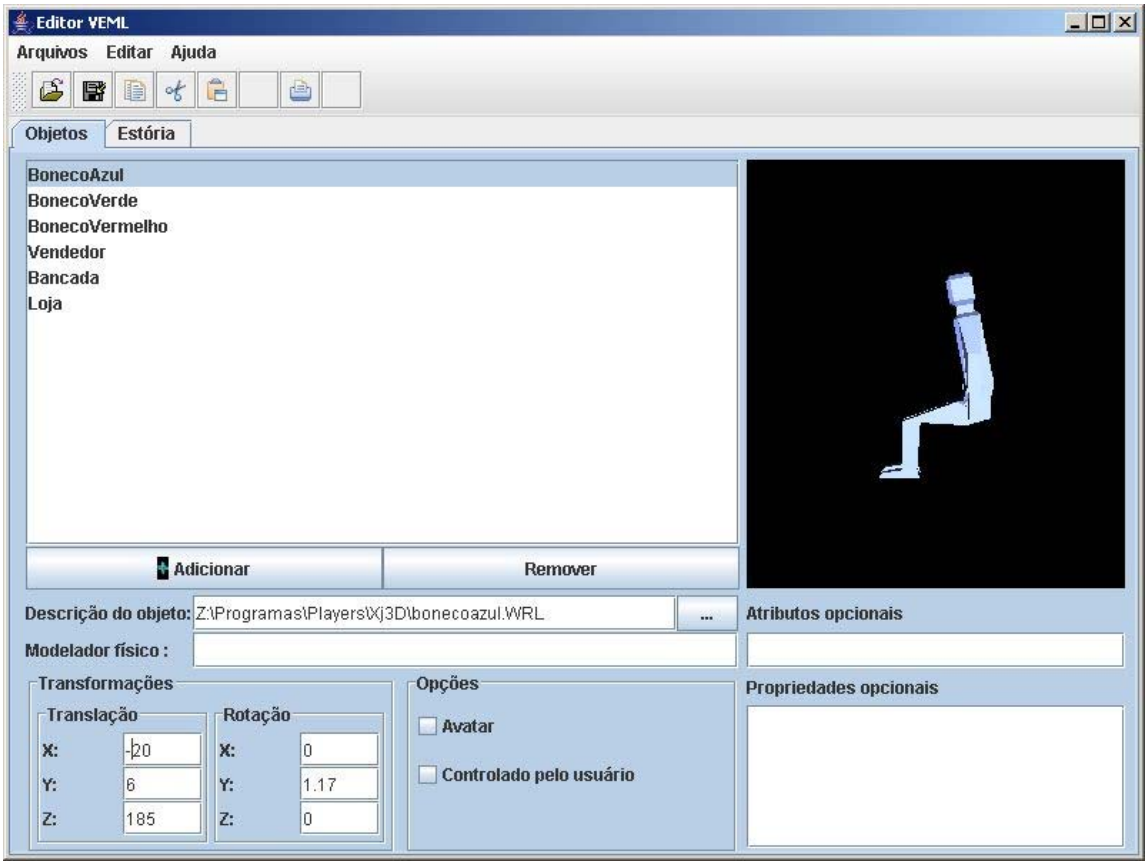


Figura 23 - Protótipo do Editor VEML: declaração de objetos

adição/remoção de pontos de interação, adição/remoção de objetos do ambiente, etc.

As Figuras 24 e 25 mostram as interfaces com o usuário do protótipo pensado para esta ferramenta.

Na aba de objetos apresentada na Figura 25 o projetista do ambiente declara os objetos que irão compor o mundo virtual. Nesta aba é possível especificar todos os atributos e propriedades do objeto contemplados no perfil mínimo do VEML e ainda usuários avançados podem especificar atributos e propriedades adicionais para perfis mais complexos.

Na aba de estória o projetista define os conjuntos de simulações atômicas do ambiente na forma de estórias não-lineares. Cada estado pode possuir um conjunto de eventos a serem realizados e um conjunto de condições para mudança de estado. Cada evento e cada condição são configurados para, quando selecionado, solicitar ao usuário

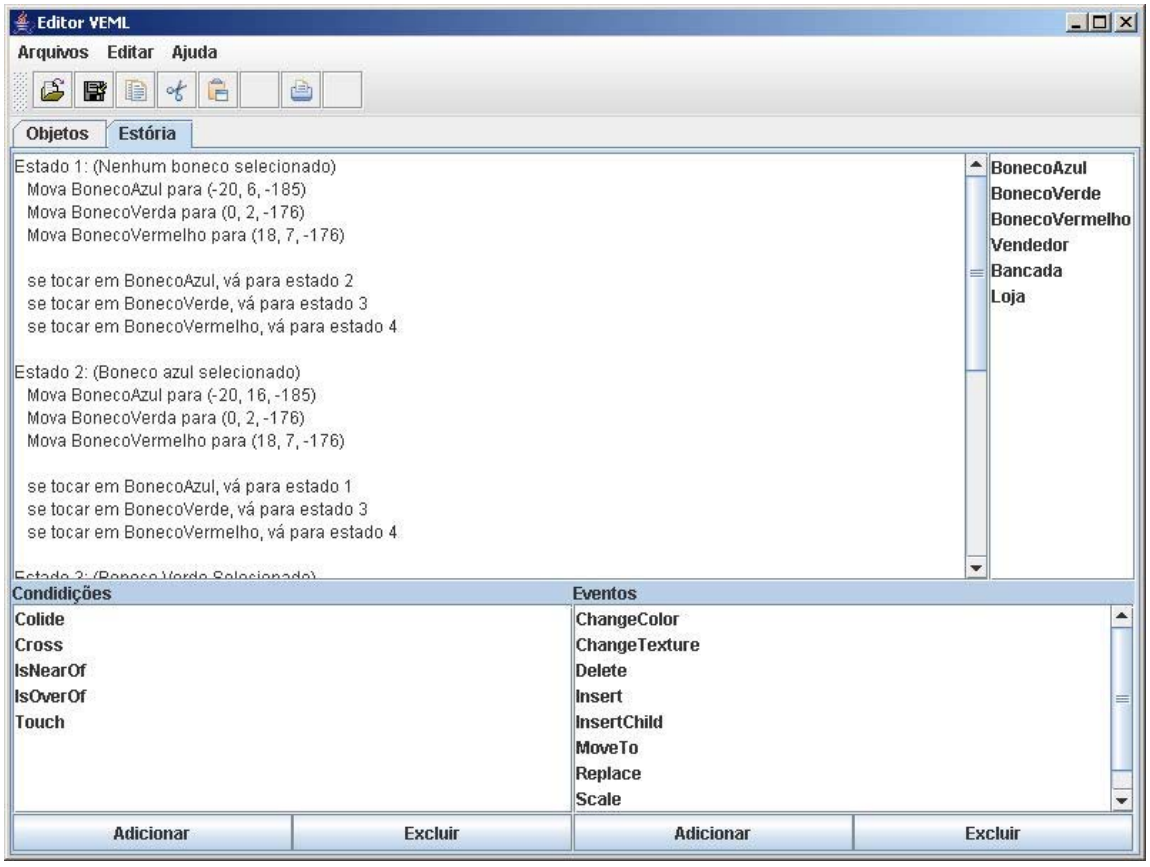


Figura 24 - Protótipo do Editor VEML: Especificação da Estória

os parâmetros necessários para sua inicialização. Estes parâmetros podem ser objetos declarados para o ambiente ou valores passados na forma de texto.

O protótipo ainda está sendo especificado com maior riqueza de detalhes e será implementado como trabalho futuro.