RANGEL JUNGLES DOS SANTOS

AMBIENTES VIRTUAIS BASEADOS NA WEB - ALGUNS ASPECTOS FUNDAMENTAIS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. André Luiz Battaiola

CURITIBA

2007

RANGEL JUNGLES DOS SANTOS

AMBIENTES VIRTUAIS BASEADOS NA WEB - ALGUNS ASPECTOS FUNDAMENTAIS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. André Luiz Battaiola

CURITIBA

2007

DEDICATÓRIA

Dedico este trabalho àqueles que me suportaram por tantos anos e que com seu apoio, carinho e atenção permitiram que alcançasse esse objetivo. Aos meus pais Sebastião e Maria; irmãos, irmã e cunhado Edson e Marsal, Nadir e Ricardo; e também às minhas amigas Adriana, Joice e Danielle.

AGRADECIMENTOS

Meus especiais agradecimentos a Deus e também a todos aqueles que de alguma forma contribuíram para a pesquisa e para a escrita dessa dissertação. Ao meu orientador Dr. André Luiz Battaiola e aos meus colegas de pesquisa Msc. Flávio Eduardo Martins, Bel. Rafael Pereira Dubiela, Bel. Daniel Avilez, César Kohl e a Vanessa Nitta.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Em função da popularização da Internet as denominadas aplicações baseadas na Web vêm ganhando espaço frente as convencionais, propiciando tanto aos desenvolvedores quanto aos usuários maior flexibilidade para a construção e utilização de aplicações. Contudo os ambientes virtuais, que podem ser empregados em várias atividades, ainda não estão completamente consolidados nesse novo ambiente.

Isso ocorre porque sua construção e operação ainda apresentam um conjunto vasto de problemas sem soluções definitivas, os quais carecem de maiores estudos e pesquisa. Motivo pelo qual na pesquisa de mestrado descrita nessa dissertação buscou-se encontrar respostas para três questões preponderantes no desenvolvimento desse tipo de aplicação:

- Quais são as linguagens e padrões mais adequados para a criação de aplicações gráficas bi e tridimensionais, interativas e inter-comunicáveis, baseadas na Web?
- Como a execução dessas aplicações em máquinas que dispõem de recursos limitados de processamento e de memória pode ser viabilizada?
- Como as dificuldades enfrentadas pelos usuários para a instalação e operação dessas aplicações podem ser reduzidas?

O objetivo era o de fornecer subsídios técnicos que permitissem a progressão do uso dos ambientes virtuais na Web, tendo em vista que há um grande potencial de emprego dessas aplicações em domínios como desenho auxiliado por computador, visualização científica e entretenimento digital, por exemplo.

Por se tratar de uma pesquisa de cunho qualitativo e caráter exploratório, o método empregado na pesquisa foi o de Estudo de Casos Múltiplos com Unidades Incorporadas.

Os principais resultados obtidos na pesquisa foram:

i) A definição de um conjunto das tecnologias adequadas à criação de Ambientes Virtuais Baseados na Web;

ii) Um pacote para a extensão de uma dessas tecnologias, o qual permite que ela seja usada para a criação de aplicações com processamento gráfico local ou remoto,

iii) Técnicas que permitem a instalação automatizada de plugins e bibliotecas necessárias para execução das aplicações.

O autor deste trabalho acredita que este conhecimento cria um viés tecnológico que pode auxiliar no desenvolvimento de Ambientes Virtuais Baseados na Web, bem como facilitar a utilização dessas aplicações pelos usuários.

ABSTRACT

Due the Internet's popularization, the called Web based applications come gaining space over the conventional ones. In consequence, as much developers as users have been benefited by the attainment of a more flexible way for construction and use of applications.

Despite this, the virtual environments, that can be used in the most varied fields of knowledge, are not yet completely consolidated in this new environment. Its construction and operation still present an extensive set of problems without definitive solutions, which ones lack study and research. For this reason, this master's research tries to find answers to three of these preponderant questions:

- What are the languages and standards more adjusted for the creation of Web based bi and three-dimensional, interactive, inter-communicable graphical applications?
- How the execution of these applications in machines that make use of limited resources of processing and memory can be possible?
- How the difficulties faced by the users for installation and operation of these applications can be reduced?

The goal with the attainment of answers to these questions was to supply subsidies to the development of these applications and its use in some domains (*CAD*, scientific visualization, digital entertainment, etc.), that already use conventional graphical applications and can be benefited by the Web use.

Being a research of qualitative nature and exploratory character, the employed method was the Multiple Cases with Incorporated Units Study, which can be described as being the observation of particular aspects of a problem by the analysis of objects / elements / situation where they are found.

The main deriving results of this research had been:

i) The definition of a set of the more adjusted technologies to the creation of Web Based Graphical Applications. The target of these applications varies of simplest (primi-

tive geometric only) until most complex ones (counting on interactive animations, sound equipment, etc.), being both hardware and software platform independent and capable to be Web based used,

ii) A package for extension of one of these technologies, which allows its use for the creation of applications with local graphical or remote processing,

iii) Techniques that allow the automatized installation of plugins and necessary libraries for the applications' execution.

The author of this work believes that this knowledge creates a technological bias that can assist futures developers in the development of Web Based Virtual Environments, as well as facilitate the use of these applications by the users.

CAPÍTULO 1

INTRODUÇÃO

Este capítulo aborda as razões principais consideradas para o desenvolvimento de Ambientes Virtuais Baseados na Web (AVBW). Apresenta especialmente àquelas que fundamentam o interesse por uma sub-categoria específica dessas aplicações: a de processamento de conteúdo gráfico interativo. Alguns dos muitos campos de conhecimento em que podem ser utilizadas também são apresentados como parte da fundamentação teórica, a qual conta ainda com uma seção a respeito dos objetivos e da organização geral do texto desta dissertação.

1.1 Aplicações Baseadas na Web

Em razão de políticas governamentais, redução dos preços de componentes de hardware e de software, mudanças culturais e também em função da crescente necessidade de inclusão digital, o acesso a Internet vem crescendo em todo o mundo [7]. Com isso as aplicações tradicionais (que rodam localmente) vêm se tornando cada vez menos atrativas, dando espaço a aplicações mais poderosas baseadas na Web. [83]

Uma das principais vantagens trazidas por essas novas aplicações é a maior liberdade para a utilização dada aos usuários. Pois, para o seu uso, não é preciso instalar (no sentido mais estrito do conceito) softwares, bibliotecas e demais elementos necessários para a sua execução. Esse benefício se acentua pelo fato de que estas aplicações podem ser usadas em qualquer máquina, em qualquer lugar do mundo, sem que para isso sejam necessários hardware e/ou softwares específicos, mas tão somente computadores com acesso à Internet.

Outra grande vantagem é a possibilidade de se considerar, durante o desenvolvimento da aplicação, um único ambiente de execução (o ambiente Web) ao invés de vários. Graças a isso, os desenvolvedores desse tipo de aplicação não precisam gerar infindáveis versões para diferentes sistemas operacionais, como ocorre nas aplicações convencionais.

Ao contrário, podem se ater ao desenvolvimento de uma única versão que poderá alcançar diferentes populações de usuários e até mesmo ser atualizada simultânea para todas elas. Algo impensável para as aplicações tradicionais. [92]

Como ilustração apenas, pode-se citar como bons exemplos desse novo tipo de software, os discos virtuais e as páginas com manipulação de bases de dados, mas mais especificamente os *Webmails*. Esses softwares contém todas as funcionalidades básicas dos cliente de e-mail convencionais e operam na Internet em navegadores Web. São executados livremente por seus usuários em diferentes máquinas, independentemente das suas plataformas, sem que para isso seja requerido nenhum esforço adicional de instalação e/ou configuração.

1.1.1 Ambientes Virtuais Baseados na Web (AVBW)

Dentro deste contexto de mudança do ambiente desktop para a Web, os ambientes virtuais baseados na Web merecem especial atenção, não somente pelo fato de trabalharem em ambiente distribuído, sob a arquitetura cliente-servidor, diferentemente das suas equivalentes tradicionais, tampouco por isso implicar na necessidade de respeito (sempre que possível) ao modo de exploração típico da Internet, no qual o usuário recebe o conteúdo instantaneamente após clicar em um link[90], mas pelas muitas possibilidades que o seu desenvolvimento pode trazer.

Segundo Lau e Kunni, atualmente há um quadro bastante favorável ao desenvolvimento desse tipo de aplicação. Particularmente devido a redução dos preços e ao aumento da disponibilidade dos hardwares para aceleração de gráficos 3D em PC's [92]. Contudo essas condições comerciais e técnicas favoráveis são apenas alguns dos fatores que justificam o investimento em pesquisa para o desenvolvimento desses softwares.

A vasta gama de áreas em que podem ser empregadas é a principal razão que sustenta o maior interesse nessa categorias de aplicações, que a exemplo das suas equivalentes tradicionais podem ser usadas nas mais variadas tarefas. [51]

Engenharias, medicina, administração e física com suas ferramentas de desenho auxiliado por computador (*Computer-Aided Design - CAD*), aplicações visuais de diagnóstico,

esquemas de representação de dados financeiros e simulações de fenômenos naturais são, respectivamente, ótimos exemplos de domínios que utilizam com sucesso ambientes virtuais convencionais e que podem lucrar muito com o desenvolvimento de equivalentes baseadas na Web.

1.1.2 Áreas Mais Beneficiadas

Dos muitos campos que podem fazer uso dessas aplicações, o ensino, a visualização científica e o entretenimento digital talvez sejam os que podem receber o maior impacto pelo emprego dessa nova tecnologia.

Em *"Improving Visualization Skills in Engineering Education"*(2005) [67] Contero e outros relatam a experiência de utilização de AVBW como ferramentas auxiliares ao ensino. Eles demonstram que os AVBW oferecem ao ensino um conjunto novo de recursos que podem ser empregados na confecção de materiais didáticos mais eficazes, dinamizando o ensino e aprendizado.

Esta idéia é sustentada indiretamente também por Schär e Krüger em *"Using New Learning Techonologies with Multimedia"*(2000)[95]. Segundo esses autores ferramentas para ensino auxiliado por computador (*Computer-Aided Learning - CAL*) realmente eficazes devem possuir, além de um método pedagógico adequado, representações de conteúdo capazes de reduzir a carga cognitiva dos estudantes. O que pode ser feito pela combinação de informação verbal (narrativa em voz ou texto) e não-verbal (imagens, vídeos, animações, esquemas, etc.), de representações visuais e auditivas e de estáveis (compostas por texto e/ou figuras) e variáveis (vídeo em movimento, som, animações). Elementos comumente encontrados nos AVBW, o que comprova a sua afinidade com o ensino.

De maneira semelhante, a visualização científica também tem muito a lucrar com esses novos sistemas. Ying, Gracanin e Lu em " *Web Visualization of Geo-Spatial Data using SVG and VRML/X3D*"(2004)[46], Gill, Caris e Smith em " *Interactive Web-Based Visualisation of Block Model Data*"(2004) [106] e Gelautz e outros em " *Web-based Visualization and Animation of Geospatial Data Using X3D*"(2004)[71] relatam, entre outras coisas, os

benefícios que a utilização via Web pode trazer à visualização científica. Não somente no que diz respeito a visualização de dados de geoprocessamento, mas a visualização científica como um todo.

Eles argumentam principalmente que a modelagem e utilização via Web permitem uma rápida disseminação de resultados de pesquisas recentes para a comunidade científica e provêem uma poderosa ferramenta amigável ao usuário para a apresentação de produtos científicos com potencial de exploração comercial.

O imenso potencial comercial dessa solução também é a aresta que liga o entretenimento digital aos (AVBW). Internet e entretenimento digital já possuem um extenso relacionamento já há algum tempo. Jogos em rede são um bom exemplo de aplicações de entretenimento digital na Internet bastante populares. Contudo a sua utilização baseada na Web especificamente ainda está galgando seus primeiros passos.

Segundo Crandall e Sidak[23], componentes da *Entertainment Software Association* (*ESA*)[1], os *Massively Multiplayer Online Role Playing Game* (*MMORPG's*)[2], tem feito muito sucesso entre os jovens e adolescentes nos últimos anos. Iniciados originalmente somente com texto, com o avanço da tecnologias da Internet e o aumento da banda disponível, esses jogos já incluem imagens estáticas.

Apesar disso alguns, *MMORPG's*) como o *Warcraft*, dispõem de 5.5 milhões de jogadores que pagam cerca de 15 dólares por mês para ter acesso ao jogo, dando lucro de vários milhões de dólares aos seus desenvolvedores. Situação que dá uma idéia do quanto bem sucedidos jogos baseados na Web com gráficos interativos podem ser.

[1] *Entertainment Software Association*(*ESA*) - Associação que existe nos Estados Unidos desde 1994 e é formada por gigantes do entretenimento digital como *Atari*, *Buena Vista Games*, *Eidos Interactive*, etc.

[2] *Massively Multiplayer Online Role Playing Game* (*MMORPG's*) - jogos de interpretação de papéis, multi-usuários, em rede.

CAPÍTULO 2

PROBLEMAS ABORDADOS NA PESQUISA

O desenvolvimento dos *AVBW*, assim como o de sistemas convencionais, possui um universo bastante vasto de particularidades e de problemas. Contudo, devido ao surgimento bastante recente deste tipo de aplicação, muitos desses problemas ainda carecem de soluções definitivas, sendo portanto tópicos interessantes de pesquisa.

Os três tópicos abordados na pesquisa descrita nessa dissertação são: a) A insuficiência de capacidade de processamento e de memória das máquinas onde se deseja executar as aplicações, b) A dificuldade de escolha por linguagens e padrões adequados a sua criação e c) O baixo nível de usabilidade oriundo das dificuldades existentes no processo de instalação desses softwares.

A opção pela abordagem deles foi determinada não somente pelo seu caráter recorrente no processo de desenvolvimento, mas principalmente por eles representam barreiras reais à consolidação dessas aplicações.

Nas seções seguintes, cada um desses problemas é apresentado mais detalhadamente e se aponta a sua relevância para a consolidação das AGBW.

2.1 Escolha das Linguagens e Padrões Adequadas à Criação de AGBW

Um dos primeiros desafios encontrados na criação de AVBW é a escolha das linguagens e padrões a serem utilizados. Atualmente, muitas das tecnologias existentes para a produção de softwares baseados na Web não possuem recursos mínimos necessários a criação de ambientes virtuais, o que impossibilita o seu uso em AVBW. [87]

O controle de diversas mídias (tais como texto, imagens, sons e animações), assim como a detecção de colisão tão necessária à criação de jogos de computador e outros tipos de softwares para entretenimento digital, descrito por Dalgarno [12] como fundamental

para a criação de material didático altamente interativo na Web, por exemplo, não é realizado pela maioria das tecnologias disponíveis.

Além disso, a necessidade de completa independência de plataforma, decorrente do compromisso de rodar em qualquer *host* da rede, é outro fator a dificultar a escolha das linguagens e padrões a serem adotados. Já que algumas tecnologias que possuem os recursos necessários à criação de conteúdos gráficos são atreladas a sistemas operacionais ou navegadores específicos, o que as torna inadequadas a construção de AVBW para públicos com sistemas heterogêneos.

Esse é o caso da *Vector Markup Language* (*VML*), linguagem de representação de gráficos vetoriais bidimensionais interpretada no navegador Web desenvolvida pela *Microsoft*, *Macromedia* e outras companhias, por exemplo. [17] Com recursos suficientes para a criação de conteúdos 2D complexos, ela não é indicada para a produção de AVBW para quaisquer usuários.[16]

Isso ocorre porque aplicações construídas com essa tecnologia só podem ser executadas no navegador *Microsoft Internet Explorer* (*MSIE*), único que possui suporte a essa linguagem. Como conseqüência usuários de outros navegadores Web populares, tais como *Mozilla Firefox* e *Opera*, não têm vias para utilizar essas aplicações em seus navegadores preferidos, já que além de não haver suporte a essa linguagem nesses navegadores não há *plugins*, bibliotecas, pacotes ou outros componentes que possam ser instalados a fim de provê-lo. [15]

Neste contexto a definição das tecnologias a serem usadas é mais do que uma mera questão de escolha, é também a definição de quais conteúdos poderão ser implementados e a quais usuários o software poderá ser disponibilizado. Assim dependendo das opções feitas, a manipulação de certos ambientes virtuais em alguns navegadores pode pode ser simplesmente inviável. Razão pela qual este foi um dos aspectos considerados na pesquisa por afetar tanto os conteúdos manipulados pela aplicação, quanto a distribuição desses sistemas, determinando, conseqüentemente, os campos em que podem ser utilizados e também o público a que estarão acessíveis.

2.2 Insuficiência na Capacidade de Processamento e Memória

A imensa capacidade em prover comunicação entre máquinas com diferentes configurações é certamente uma das características mais admiráveis da Internet. Fruto de uma família de protocolos cuja implementação independe de configurações de hardware e de software, a Internet permite que máquinas com os mais variados recursos computacionais se comuniquem. [35]

Contudo, apesar de favorável em muitos contextos, essa característica possui também um lado negativo ao uso de AVBW. Isso porque esses software, de um modo geral, demandam considerável capacidade de processamento e memória, recursos que nem sempre estão disponíveis nos *hosts* da Internet.

Como conseqüência, a idéia de que um AVBW poderá ser executado somente em um conjunto restrito de *hosts* da rede (os que possuem os recursos mínimos necessários), apesar de ideologicamente inaceitável, não é nem um pouco absurda, mas ao contrário, é uma realidade presente na utilização desse tipo de software.[1]

Porém, esta situação não deverá persistir por muito tempo. O aumento de disponibilidade e a redução dos preços dos componentes de hardware para aceleração de gráficos 3D em *PC's* são bons indícios de que em um futuro próximo essa realidade poderá mudar [92]. O mesmo pode-se dizer com relação ao surgimento de novos padrões para a rede, tais como o *IPV6*, e a crescente popularização das redes gigabit.

Em termos práticos, estes novos recursos propiciam o aumento da banda disponível para comunicação em rede e, conseqüentemente, a possibilidade da adoção de técnicas de processamento distribuído para a superação dos problemas de processamento[82, 21, 94].

A visão dessas perspectivas torna clara a noção de que os problemas relativos aos recursos computacionais insuficientes serão contornadas em um futuro próximo. Mas mais do que isso, que há a necessidade de pesquisas mais elaboradas sobre o assunto. Tendo em vista que as técnicas citadas só recentemente começaram a ser empregadas para a resolução desse problema específico. Motivo pelo qual esta questão foi escolhida

[1]Note-se que, o fato de uma aplicação de visualização científica (que utiliza vários megabytes de memória e um grande volume de processamento) não poder ser executada por um computador médio ligado à Internet não é algo que cause espanto.

para exploração no contexto dessa dissertação.

2.3 Baixa Usabilidade Causada pelas Dificuldades de Instalação e Operação

Um dos conceitos mais importantes existentes no estudo de Interfaces Homem-Computador é a da usabilidade de software. Segundo Andreas Holzinger, usabilidade é freqüentemente definida como sendo a facilidade de uso e aceitabilidade de um sistema por uma classe particular de usuários ao realizar tarefas específicas em um ambiente específico. [9]

Segundo a definição oficial dada pela Norma ISO 9241-11 porém, usabilidade é "a extensão para a qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos em um específico contexto de uso". [44]

Mas mais do que um conceito com várias definições possíveis, a usabilidade é uma componente fundamental a ser considerada em todo o processo de desenvolvimento de software. Não somente nas etapas finais e na construção de interfaces gráficas, mas realmente em todo o processo. [117, 100, 5]

Em outros termos, a busca pela redução na dificuldade de uso deve envolver todas as questões relacionadas a utilização do software e não somente a interação com o usuário. A preocupação com o "tornar mais fácil"precisa estar presente também em outras tarefas, como à confecção de manuais, a criação de ferramentas de atualização e à instalação do software.

Nielsen em seu livro *"Usability Engineering"* descreve um caso em que apesar do bom nível de usabilidade da aplicação em si, dificuldades na instalação foram responsáveis pela baixa aceitação do sistema e o conseqüente fracasso do projeto. Este relato, apesar de ser a respeito de um sistema convencional, apresenta um grande desafio a ser enfrentado no desenvolvimento de AVBW.[45]

Apesar de, por definição, AVBW não necessitarem de instalação, na prática a situação é um pouco diferente. *Cookies*, bibliotecas e *plugins* são apenas alguns exemplos de elementos requeridos por alguns desses softwares e que necessitam instalação.

Porém, o problema que atenta contra a usabilidade, entretanto, não está na necessidade de instalação de tais elementos, mas sim nos fatos de que esse processo costuma exigir vários passos e de que eles podem variar de sistema operacional para sistema operacional e/ou de navegador Web para navegador Web. Isto resulta em grandes diferenças nos procedimentos a serem adotados para a instalação desses artefatos em diferentes sistemas operacionais e, pior, em diferentes navegadores Web em um mesmo sistema operacional.

Tendo em conta essa situação e considerando o objetivo da pesquisa de fornecer subsídios à consolidação dessas aplicações, a busca por técnicas capazes de reduzir a dificuldade existente na instalação dos AVBW foi tomada como ponto a ser explorado no âmbito dessa dissertação. Não somente por representar uma medida para aumentar a satisfação dos eventuais usuários, mas por ser um fator crucial à consolidação desse tipo de software.

CAPÍTULO 3

REVISÃO BIBLIOGRÁFICA

Muitos trabalhos têm sido realizados com respeito a utilização de conteúdos gráficos na Web. Contudo são poucos aqueles que têm se concentrado simultaneamente nas três questões principais desse estudo. A visão geral de alguns desses trabalhos demonstra bem isso:

Em "*A Java 3D-Enabled Cyber Workspace*", Wang e outros [58] apresentam um *framework* composto por três módulos (apresentação, controle e modelagem) para a criação de aplicações com conteúdo tridimensional na Web. Criado com tecnologias tais como *Java Server Pages* (*JSP*), *Java 3D* e *Virtual Reality Modelating Language* (*VRML*), tal *framework* utiliza processamento local e seria uma opção viável para a criação de AGBW, contudo, apesar de estar sujeito aos problemas de insuficiência de processamento e dificuldade de instalação, estes problemas não são tratados no âmbito desse trabalho.

Em "*A Java Web Application for Allowing Multiuser Collaboration and Exploration of Existing VRML Worlds*", Presser [20] apresenta outro *framework* para a manipulação de ambientes tridimensionais na Web. Também criado a partir de tecnologias como *JSP* e *VRML*, esse *framework* também utiliza processamento local para os conteúdos tridimensionais, razão pela qual é, igualmente, suscetível aos problemas de dificuldade de instalação e de insuficiência de processamento. Estes problemas também não tem soluções apontadas neste texto.

Em "*A Multiuser 3D Web Browsing System*", Huang e outros [47] descrevem uma arquitetura para aplicações tridimensionais baseadas na Web usando *Common Gateway Interface* (*CGI*) e *VRML*. Apesar de bastante claros quanto a promoção de comunicação entre múltiplos usuários, eles não apresentam as formas como as aplicações criadas sob essa arquitetura podem ter os problemas de insuficiência de processamento e dificuldade de instalação superados.

Em "*A VRML-Based Anatomical Visualization Tool for Medical Education*", Warrick e Funnell [81] descrevem uma ferramenta educacional para visualização de informações de Anatomia. Composta por módulos 2D e 3D, tal ferramenta utiliza tecnologias como *VRML* e *Java 3D* para a criação de ambientes gráficos interativos. Apesar disso e do seu uso claramente utilizar processamento local, não são descritas medidas para a resolução das questões da insuficiência de processamento e de dificuldades de instalação eventualmente existentes.

Em "*A Web-based System for Interactive Visualization of Scientific Concepts*", Neo, Lin e Gay [55] descrevem um sistema para a visualização tridimensional interativa de conceitos científicos variados. Criado com tecnologias como *Adobe Shockwave Flash*, *VRML*, e *Extensible 3D* (*X3D*) esse sistema permitiria a fácil publicação de materiais tridimensionais animados sobre conceitos científicos de diversas áreas, utilizando para isso uma arquitetura com processamento local. Por outro lado não se cita nenhuma medida quanto a resolução dos problemas de dificuldade de instalação e superação de problemas de insuficiência de processamento.

No tutorial "*Building Virtual Worlds with VRML*", Nadeau [28] descreve como a criação de ambientes tridimensionais pode ser feita usando *VRML*. A tecnologia *Java 3D* é citada como passível de ser usada nesse tipo de aplicação na Web. Contudo qualquer menção é feita sobre a forma como as questões do processamento e de instalação podem ser tratadas.

Em "*Interactive Web-Based Visualisation of Block Model Data*", Gill, Caris e Smith [106] descrevem como as tecnologias *VRML* e *X3D* podem ser usadas para a visualização de dados complexos. Eles apresentam uma arquitetura com processamento local, mas não se atém a resolução de problemas de processamento e/ou de usabilidade que podem surgir em função da utilização dessa topologia de processamento.

Em "*jGL and its Applications as a Web3D Platform*", Chen e Nishita [13] propõem uma nova plataforma para gráficos 3D baseada em uma derivação da *OpenGL* criada por eles. Apesar de abordar questões de processamento e de facilidade de instalação, não aponta alternativas para contornar situações em que a capacidade de processamento

encontrada é insuficiente.

Em "*SVG for Educational Simulations*", Bogaard, Vullo e Cascioli [25] descrevem como *Scalable Vector Graphics* (*SVG*), *Synchronized Multimedia Integration Language* (*SMIL*), *ECMAScript* e *Perl Hypertext Processor* (*PHP*) foram utilizados em simulações para o ensino de conceitos de redes de computadores. Eles também fazem comparações dessas tecnologias com a *Adobe Shockwave Flash* e indicam suas vantagens sobre essa tecnologia, contudo, não deixam claro como a questão da instalação pode ser trabalhada em circunstâncias similares.

Em "*The Java 3D API and Virtual Reality*", Rosenblum e Macedonia [56] explicam como *Java 3D* pode ser utilizado em ambientes virtuais. Eles relatam as vantagens da utilização dessa tecnologia nesses sistemas, em especial no que diz respeito a promoção da interatividade, contudo não se atém às questões de insuficiência de processamento e de instalação, quando da utilização dessa solução em aplicações para a Web.

Em "*The Virtual Reality Modeling Language and Java*", Brutzman [33] descreve como as tecnologias *Java 3D* e *VRML* podem ser empregadas para a criação de conteúdo tridimensional interativo para a Web, entretanto, também não deixa claro como as questões da instalação e da insuficiência de processamento, causadas pelo processamento local proposto, podem ser contornadas.

Em "*Web Visualization of Geo-Spatial Data using SVG and VRML/X3D*", Ying, Gracanin e Lu [46] descrevem como as tecnologias *SVG* e *VRML/X3D* foram utilizadas na construção de um *framework* para a visualização na Web de dados de geoprocessamento. Eles descrevem como a intercomunicação foi conseguida entre os conteúdos bidimensionais e tridimensionais, mas não explicam como os problemas de processamento e usabilidade, decorrentes da utilização do processamento local, podem ser superados.

Em "*Web-based Visualization and Animation of Geospatial Data Using X3D*", Gelautz [71] e outros também apresentam um framework para a visualização de dados de geoprocessamento na Web. Construído usando *X3D*, as aplicações criadas também teriam o processamento centrado no cliente (processamento local), porém, apesar disso representar possíveis problemas com a usabilidade em face da dificuldade de instalação e com o

processamento em razão da grande demanda de recursos para processamento, nenhuma medida para remediar tais problemas é citada.

CAPÍTULO 4

MÉTODO UTILIZADO NA PESQUISA

Na pesquisa realizada, em virtude do seu caráter exploratório e qualitativo, seguindo a classificação dada por Robert K. Yin[89], foi utilizado o método de Estudo de Casos Múltiplos com Unidades Incorporadas.

Em termos práticos, os processos de desenvolvimento de dois softwares foram analisados e neles as respostas dadas as questões de pesquisa foram observados em maiores detalhes. Para isso, conjuntos de procedimentos previamente estabelecidos foram adotados e especial atenção foi dada à escolha das tecnologias.

Por se tratar de uma atividade de cunho mais teórico, ela precisou ser realizada separadamente, além de ser realizada com base ambos os aplicativos em análise, diferentemente das demais que foram realizadas no contexto de cada caso.

Os procedimentos adotados para essa tarefa foram os seguintes:

- Inicialmente, as especificações das aplicações foram analisadas e delas foram retirados os requisitos mínimos exigidos às tecnologias a serem empregadas.
- Uma pesquisa bibliográfica preliminar foi realizada com a finalidade de apontar demais requisitos desejáveis e indicar as tecnologias mais promissoras para uso na criação das aplicações.
- Os resultados obtidos nessa fase preliminar do estudo foram utilizados em uma nova pesquisa bibliográfica mais aprofundada para a determinação das linguagens e padrões a serem adotados.
- Os resultados dessa nova pesquisa foram então convertidos em tabelas que permitem fácil verificação e comparação dos requisitos das tecnologias citadas na literatura como mais promissoras.

- O critério utilizado para a escolha das linguagens e padrões mais adequados, foi o da melhor adequação aos requisitos previamente estabelecidos. Na única situação em que esse critério foi insuficiente, outros aspectos do contexto do desenvolvimento da pesquisa foram considerados para a realização da escolha.

Para a observação dos demais aspectos foi utilizado outro conjunto de procedimentos. A metodologia de desenvolvimento em espiral proposta por Boehm[97] e a técnica de prototipagem foram adotadas para auxiliar o desenvolvimento das aplicações. Com isso, os passos de definição de objetivos, avaliação e redução de riscos, desenvolvimento e validação e planejamento foram executados continuamente, em ciclos de desenvolvimento[97].

Para o primeiro ciclo de desenvolvimento de ambas as aplicações o objetivo estabelecido foi a criação de um conjunto de protótipos simples, relativos as questões investigadas. O plano inicial era utilizar esses protótipos como base para a geração de uma primeira versão completa das aplicações em um segundo ciclo do processo. Na qual os problemas investigados seriam simultaneamente tratados.

Mas infelizmente não houve tempo hábil para execução do segundo ciclo para uma das aplicações. Pois em razão do grande número de peculiaridades encontradas e da complexidade da implementação das soluções idealizadas o cronograma inicial estabelecido não pode ser cumprido.

Apesar disso, é importante ressaltar, os resultados da pesquisa não foram afetados, já que os dados conseguidos com a elaboração dos protótipos no primeiro ciclo de desenvolvimento foram suficientes para a obtenção de resultados consistentes. Mesmo sem a sua integração prevista como objetivo para o segundo ciclo, ter sido realizada.

Os critérios utilizados para avaliar os resultados obtidos com a observação dessas questões foram: a adequação às demandas dos casos e a possibilidade de extensão para casos semelhantes. A extensão para casos semelhantes, em particular, foi o critério mais considerado na análise e discussão dos resultados feita nessa dissertação. Pois dela depende o valor científico dos resultados conseguidos.

CAPÍTULO 5

CASOS EM ESTUDO

Este capítulo apresenta os casos estudados e as estratégias usadas neles para a resolução dos problemas investigados na pesquisa.

5.1 O Edugraph

O Edugraph é um jogo educacional concebido no âmbito do projeto de pesquisa *Ludic-Learning*, financiado pelo CNPq, voltado ao ensino de conceitos básicos de computação gráfica para alunos de graduação dos cursos de Engenharia, Arquitetura e Design. Composto por cinco fases, o jogo combina três ambientes bidimensionais com dois tridimensionais.

Seu enredo constitui-se da saga do Edu, personagem estilizado que em um mundo fictício se vê preso em um labirinto. Para sair deste labirinto que o aprisiona, ele precisa superar uma série de desafios que constituem as 5 fases do jogo. A cada fase concluída, o personagem retoma uma das suas cinco cores perdidas quando do seu aprisionamento no labirinto[**?**].

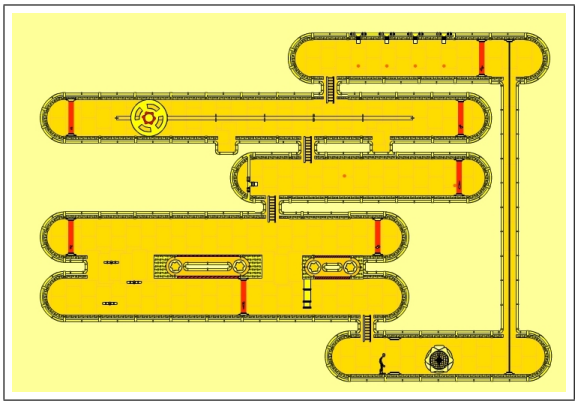

Figura 5.1: Fase Amarela.

A primeira fase, denominada fase amarela, é um jogo de aventura típico, composto

por um ambiente bidimensional no qual pedaços de uma chave estão distribuídos. Para encerrar a fase, o personagem (controlado pelo jogador via teclado) deve ultrapassar obstáculos diversos e reunir todos os pedaços da chave da fase, em um tempo máximo limite e sem ser "morto". Somente após realizar essa tarefa, o jogador poderá remontar (por meio de operações com o mouse) a chave que encerra a fase, usando as operações booleanas bidimensionais, conteúdo educacional objeto do aprendizado.

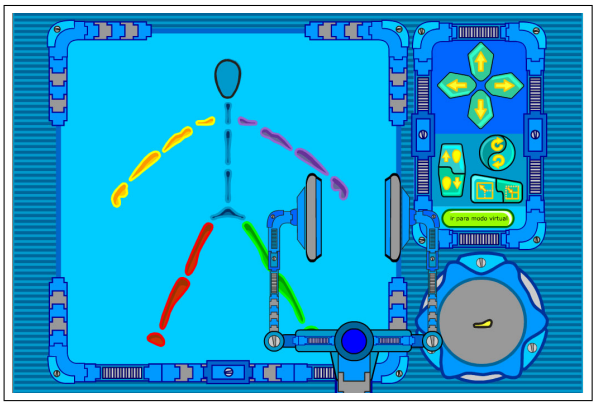

Figura 5.2: Fase Azul.

A segunda fase, denominada fase azul, é um jogo de quebra-cabeça. Composto também por um ambiente bidimensional, ela exige que o jogador remonte o personagem Edu em um molde, a partir dos pedaços do seu corpo. Para isso, ele pode se utilizar tanto do teclado, quanto do mouse para fazer a movimentação dos pedaços do corpo e as transformações necessárias. O conteúdo educacional, objeto de aprendizado abordado nessa fase, são as transformações geométricas bidimensionais de rotação, translação e mudança de escala. Elas são executadas no jogo durante a remontagem do personagem, em função do fato de que os pedaços do seu corpo são apresentados ao jogador em um local à parte do molde e em posição e escalas diferentes das que devem ser usadas[**?**].

A terceira fase, denominada fase verde, é um jogo de labirinto 2D. Nele, o personagem representado por um veículo esférico é deslocado dentro de um labirinto em busca de pedaços da chave que encerram a fase. "Bolas assassinas"sempre em movimento dentro do labirinto são os obstáculos a realização da tarefa. O conteúdo educacional abordado nessa fase é o das transformações geométricas de rotação e translação, dado que para se

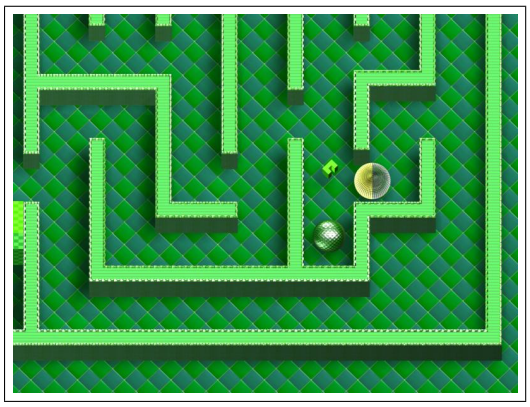

Figura 5.3: Fase Verde.

movimentar dentro do labirinto é necessária a execução dessas transformações.

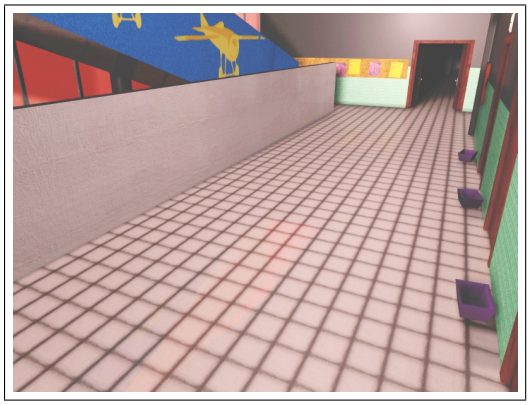

Figura 5.4: Fase Vermelha.

A quarta fase, denominada fase vermelha, é um jogo de estratégia 3D. O personagem, agora aprisionado em um dos andares de um prédio (representação do Departamento de *Design* da UFPR), necessita, assim como na primeira fase, encontrar e capturar pedaços da chave necessária para o encerramento da fase. Contudo, nessa fase, os desafios a serem superados são desafios lógicos e de raciocínio, dado que o jogador precisará observar e analisar pistas deixadas em diferentes locais do ambiente. Essas pistas orientarão o jogador quando da sua busca por pedaços da chave, a qual permitirá acesso ao computador "Lai_01"existente no Laboratório de Animação Interativa (LAI) também representado no cenário. Os objetivos de aprendizado explorados nessa fase são as operações booleanas, em especial a intersecção.

A quinta e última fase, denominada fase roxa, é um quebra-cabeça tridimensional.

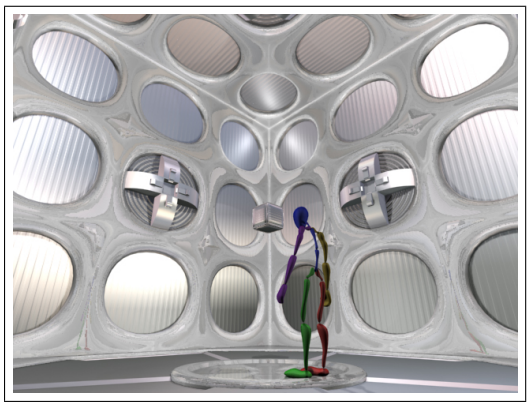

Figura 5.5: Fase Roxa.

O personagem aprisionado em uma sala denominada cubo, precisa remontar um cubo a partir de peças tridimensionais disponíveis. Para tanto, faz uso de um painel existente no centro da sala. O conteúdo educacional abordado nessa fase é o das transformações geométricas tridimensionais. A relação entre esse conteúdo e a tarefa da fase se dá em função da necessidade do jogador em usar tais transformações para posicionar as peças que constituem o cubo.

5.2 O SignType

SignType é um projeto desenvolvido no Laboratório de Animação Interação do Departamento de Design da UFPR. O texto abaixo, extraído de partes do projeto submetido ao CNPq pelos professores Dr. André Luiz Battaiola e Marcel Pereira Pauluk (atualmente engajado em programa de doutorado na Alemanha), define a motivação para o seu desenvolvimento, bem como suas principais características.

Constata-se entre as diversas áreas do conhecimento científico, um generalizado interesse pelo potencial da semiótica aplicada, em especial daquela fundada na obra do filósofo e polímata norte-americano Charles S. Peirce (1839-1914). Apesar deste interesse multíplice, a maioria dos cientistas e dos pesquisadores envolvidos com a semiótica não está metodologicamente capacitada para explorar com eficiência os conceitos instrumentais desta disciplina (Santaella 2002: xi-xvii); como conseqüência desta limitação, os resultados obtidos em pesquisas nas quais a compreensão de processos sígnicos faz-se

necessária têm ficado aquém das capacidades heurística, elucidativa e crítica da semiótica.

Por ser uma ciência de insigne caráter transdisciplinar (Nöth 1995: 125), a semiótica necessita ser retroalimentada com os resultados de sua aplicação e de sua efetiva interação com outras ciências. Resultados incorretos ou insatisfatórios obtidos pela aplicação pouco informada ou desassistida de seu instrumental teórico comprometem não apenas os resultados em pesquisa, mas a própria evolução da semiótica enquanto ciência geral dos processos sígnicos ou semiose.[48, 50]

Os esforços para reverter este quadro estiveram tradicionalmente concentrados na produção de manuais introdutórios e avançados de semiótica geral e aplicada (e.g. Coelho Neto 1980[48]; Deely 1982[50]; Eco 1976[107]; Nöth 1990[112, 114, 113], 1995; Santaella 1983, 1995, 2002[60, 61, 62]; etc.). Apenas recentemente, aliando o amadurecimento das discussões em torno das teorias de base da semiótica às possibilidades hipermidiáticas de ensino à distância, aprendizado interativo e engenharia semiótica, foram desenvolvidos os primeiros protótipos digitais de diagramas semióticos interativos: 10cubes e 3N3 (2001 Farias, Queiroz & Gomes). Na esteira destes aplicativos, SignType - Tipificador Semiótico Assistido para Dados Fenomenológicos - propõe-se como um instrumento facilitador e didático na tarefa de classificação semiótica de extração peirceana de qualquer espécie de objeto ou evento.

Os diagramas digitais interativos supracitados trabalham em um nível avançado de complexidade teórica. Apesar de contarem com um dispositivo de ajuda (Help), este se restringe a uma página explicativa das funções e atribuições dos ícones e mecanismos do aplicativo. Seu público-alvo é, portanto, restrito a especialistas da área. SignType, ao contrário, trabalha em um nível médio de complexidade teórica; tendo em vista uma gama de usuários mais ampla, o aplicativo prevê um assistente (Wizard) didático que auxiliará o usuário na utilização do programa, na compreensão de suas próprias ações e na interpretação e aplicabilidade dos resultados obtidos.

SignType também possui, quanto à metodologia de classificação semiótica empregada, um diferencial fundamental: ela se apóia nos tipos mais comuns de erro e/ou subaproveitamento comumente efetuados por pesquisadores pouco versados no assunto (Pauluk 2003:

147 passim [69]) para refinar o resultado obtido ao final do processo classificatório. Ao permitir que o usuário faça um levantamento prévio dos aspectos específicos do fenômeno analisado reportando-se às três tricotomias que dão origem aos dez tipos finais de signo (Peirce 1903[18, 19]) sem aplicar nenhuma das regras geradoras destes dez tipos (cf. Queiroz 1997, 2002[85, 86]), o aplicativo pode ampliar o resultado final apresentando algumas das classes impossíveis determinadas pelo usuário como sendo, na verdade, aspectos semióticos relevantes do fenômeno analisado que normalmente ficariam sem qualquer destaque nos diagramas tradicionalmente utilizados para modelar as relações entre as 10 classes. Esta metodologia foi aplicada experimentalmente e com sucesso em Arqueologia Pré-Histórica (Pauluk 2002[68]), na crítica da classificação de grafismos rupestres proposta por Anati (1994)[38].

SignType pode, em suma, fazer com que pesquisadores de quaisquer áreas interessados na exploração experimental e aplicada da semiótica peirceana sejam capacitados teórica e metodologicamente em uma importante área desta disciplina simultaneamente ao processo de operação do aplicativo e obtenção de resultados qualitativamente superiores.

...

Tendo como proposta ser simultaneamente um software didático e aplicável, os principais objetivos do SignType são a capacitação científica de pesquisadores e estudiosos de quaisquer áreas do conhecimento na metodologia de classificação semiótica de extração peirceana e a conversão de input fenomenológico (descrições impressionísticas) em output semiótico (tipos de signo), otimizando o aproveitamento do potencial científico da semiótica pelo usuário. Para isso ele apresenta aos usuários os aspectos teóricos da semiótica de Charles S. Peirce, na forma de em um conjunto de diagramas, os quais têm como principais características a:

- Configurabilidade: visualização ou não de certas informações, determinada por critérios hierárquicos variáveis;
- Alternabilidade: substituição dos conceitos por modelos diversos e
- Modularização: facilitação da comparação entre as classificações em curso

Esses diagramas podem ser divididos em duas categorias, uma composta por modelos gráficos estritamente bidimensionais e outra composta por um diagrama tridimensional. Ou, alternativamente em quatro tipos básicos, utilizados para:

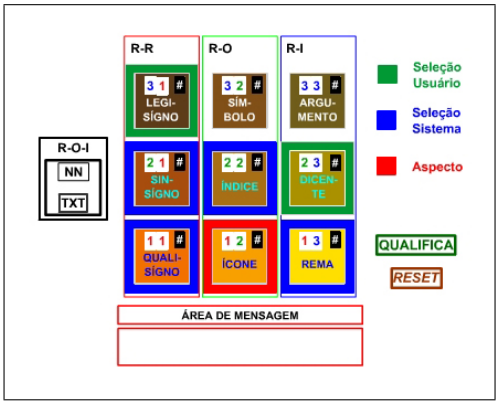

Figura 5.6: Levantamento Aspectual

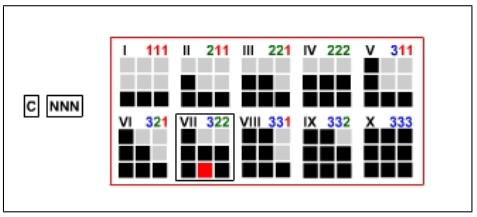

Figura 5.7: Qualificação

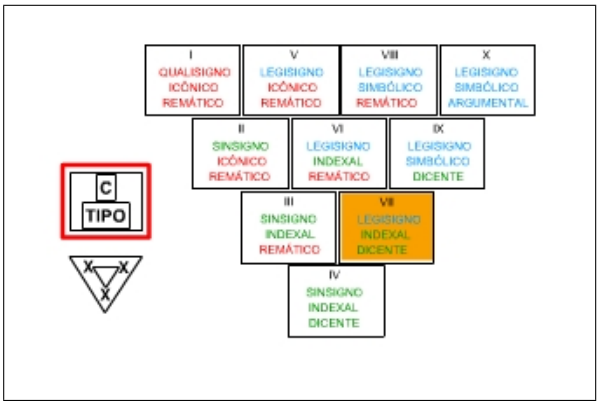

Figura 5.8: Classificação

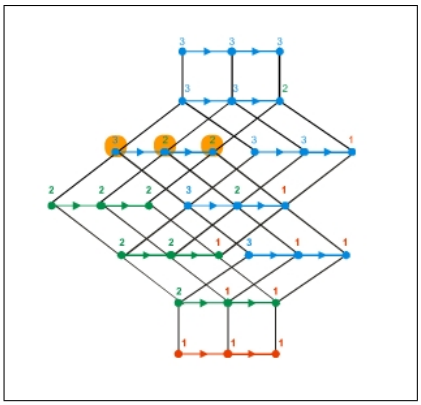

Figura 5.9: Diagramação (2D)

- Levantamento Aspectual: obtenção de dados por meio de um conjunto de botões com informações configuráveis, que representam as unidades básicas de análise (figura 5.6);
- Qualificação: apresentação dos resultados da primeira etapa de processamento via um diagrama 2D (figura 5.7).
- Classificação: apresentação dos resultados da qualificação e obtenção de informações para a etapa seguinte do processamento (classificação). (figura 5.8)
- Diagramação: apresentação final dos resultados da qualificação e classificação. (figuras 5.9 e 5.10)

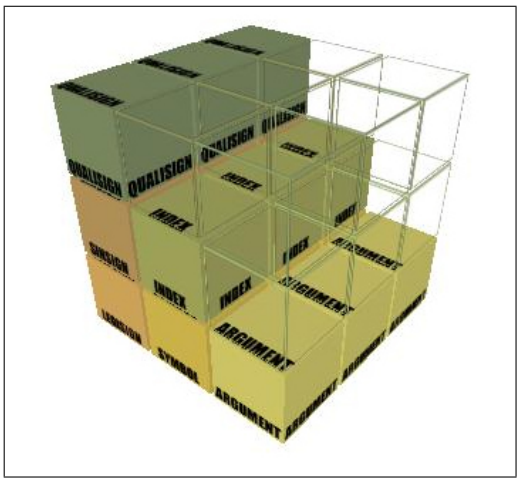

Figura 5.10: Diagramação (3D).

Uma versão inicial do SignType foi implementada utilizando-se as tecnologias estudadas neste trabalho. Esta versão carece ainda de alguns ajustes, bem como a adição de uma página de Ajuda, no entanto, no entanto, mesmo neste estado atual, ela é perfeitamente adequada para justificar a aplicação das tecnologias estudadas.

CAPÍTULO 6

ESCOLHA DAS LINGUAGENS E PADRÕES

Em *"Emerging Web Graphics Standards and Techonologies"* (2003)[91], Lau, Li, Kunii e outros autores discutem as vantagens da utilização de gráficos vetoriais e de ambientes virtuais distribuídos e apontam o que consideram as tecnologias mais promissoras para a criação de conteúdos gráficos 2D e 3D interativos para Web. Segundo eles, o *Scalable Vector Graphics (SVG)* em conjunto com a *Synchonizaded Multimedia Integration Language (SMIL)* constituem a solução mais indicada para a elaboração de conteúdos 2D baseados na Web. As características apontadas, que corroboram esta indicação, são:

- Facilidade de integração dos documentos com outras tecnologias da Web, tais como, *HyperText Markup Language (HTML)* e *Cascadin Style Sheets (CSS)*;
- O compartilhamento natural de dados com outros documentos derivados da *Extensible Markup Language (XML)*; e
- A possibilidade de trabalho em conjunto com outras linguagens de programação para Web, tais como, *Active Server Pages (ASP)*, *Perl Hypertext Pages (PHP)*, *Java Server Pages (JSP)* e *Javascript (ECMAScript)*.

Para conteúdos 3D, interativos, baseados na Web, eles apontam a *Extensible 3D (X3D)* e a *Java 3D (J3D)* como sendo as tecnologias mais adequadas. A primeira por ser um melhoramento da *Virtual Reality Modelating Language (VRML)*, uma linguagem que já foi amplamente utilizada para a criação desse tipo de conteúdo. Melhoramento este que inclui alterações na interface de programação de aplicações avançadas, formatos de codificação adicionais e um arquitetura baseada em componentes que permite a modularização dos conteúdos.

Já a segunda, a *Java 3D*, por prover ao programador uma interface de programação de aplicações (*Application Programming Interface - API*) de alto-nível, com uma poderosa

estrutura de dados para a manipulação de ambientes tridimensionais, denominada de grafo de cena, além de todas as facilidades da linguagem de programação Java. A independência de plataforma, com a possibilidade de finalização dos gráficos (rendering) em diferentes interfaces de programação de aplicações (*API*) de baixo-nível (*DirectX* e *OpenGL*) são outras características dadas como preponderantes pelos autores.

Porém, além dessas tecnologias, vale lembrar que em termos de gráficos 2D a *Macromedia Shockwave Flash* (ou simplesmente *Flash*) também possui requisitos interessantes para a produção de AGBW[79]. O que pode-se comprovar pelo fato de que esta tecnologia constitui atualmente um padrão de mercado, sendo amplamente utilizada nos mais diferentes conteúdos para Web, o que abrange de simples *banners* a complexos e extensos sites e cartões virtuais.

O fato de possuir uma ferramenta que engloba tanto a criação dos elementos gráficos, quanto a elaboração da programação necessária para manipulá-los, bem como muitos recursos para a criação de animação (animação por quadros-chave, por trajetória, por *morphing*, etc.) e manipulação de sons, são os principais, mas não únicos, atrativos dessa tecnologia.

A presença de suporte interno (*built-in*) nos navegadores mais populares e forma inteiramente automatizada de detecção e instalação dos softwares necessários para sua operação (*plugin's*) em diferentes navegadores também merecem especial destaque[22]. Por isso para a escolha das linguagens e padrões realizada no estudo de casos descrito nessa dissertação, uma pesquisa bibliográfica mais foi criteriosa foi feita. Visando identificar com melhor precisão quais tecnologias eram mais adequadas aos softwares que seriam construídos.

Os requisitos utilizados e também as análises feitas das linguagens e padrões podem ser encontradas nas próximas seções. Os dados obtidos na pesquisa bibliográfica que deram origem às informações constantes nas análises, por outro lado, podem ser encontrados no Apêndice A dessa dissertação.

6.1 Requisitos de Tecnologia

Dentre as tecnologias existentes atualmente para a produção de software, apenas um conjunto relativamente pequeno possui os requisitos mínimos necessários para o desenvolvimento de AVBW. Por este e outros fatores, como por exemplo a necessidade de independência de plataforma, a tarefa de optar por um conjunto de tecnologias está longe de ser trivial.

No estudo de caso realizado, para facilitar a escolha das linguagens e padrões a serem adotados, estabeleceu-se, com base nas especificações dos softwares, um conjunto de requisitos a ser respeitado. O objetivo era conseguir uma via para realizar a escolha mais criteriosamente, de maneira que no futuro, com o advento de novas tecnologias e/ou alteração das existentes, uma comparação mais fácil pudesse ser realizada.

Os requisitos definido como necessários para a produção das aplicações em estudo foram:

- Em termos da produção de conteúdo:
  - O suporte a manipulação de diversas mídias. Tais como: texto, imagens, sons e animações.
- Em termos de recursos para programação:
  - Vasta capacidade de controle por programação;
  - Criação de estruturas de dados complexas;
  - Obtenção de dados de dispositivos de entrada convencionais (teclado, mouse, etc); e
  - Integração com outras tecnologias.
- No que diz respeito a operação no ambiente Web:
  - Independência de navegador e de sistema operacional;

  - Suporte interno (*built-in*) ou formas de instalação automática nos navegadores mais populares;
  - Formas otimizadas de compactação dos dados geométricos e/ou de criação de mecanismos de gerenciamento de fluxos de dados; e

- Quanto as facilidades de utilização:
  - Baixo custo de utilização;
  - Padronização internacional;
  - Liberdade para modificação (código aberto);
  - Vasta documentação; e
  - Ferramentas de criação, edição e publicação.

A relação desses requisitos com as aplicações em estudo é clara. O Edugraph, como a maioria dos jogos, é constituído por elementos variados (texto, imagens, animações). Sua lógica de funcionamento não é simples o que requer recursos sofisticados para a programação, como a criação de estruturas de dados complexas, integração com outras tecnologias, etc. Sua execução via Web por usuários não especializados em computação suscita à promoção de facilidades de instalação (a qual pode ser fornecida pelo suporte interno ou formas de instalação automática em navegadores populares) e à necessidade de independência de plataforma.

Da mesma forma o SignType, que apesar de ser uma aplicação bem mais simples, além dos recursos anteriormente citados, exige também condições diferenciadas para a execução. As situações idealizadas para o seu uso prevêem a sua operação em máquinas de baixa capacidade de processamento.

6.1.1 Suporte a Diferentes Topologias de Processamento - Um Requisito à Parte

A organização da aplicação, ou seja, a forma como ela é executada e como é realizado o processamento que transporta o resultado visual da manipulação do ambiente virtual ao

usuário, é um fator na elaboração de AVBW. Este fator é mais relevante na implementação de conteúdos tridimensionais, já que conteúdos bidimensionais costumam ser bem menos complexos e exigir menos recursos de processamento.

A sua influência na produção desse tipo de software é tão importante quanto a consideração dos requisitos anteriormente citados. A opção por uma determinada topologia influencia tanto a forma como será organizado e implementado o código, quanto à escolha da tecnologia a ser utilizada.

O fato dos AVBW possuírem a arquitetura cliente-servidor e de que a Internet é uma rede composta por hosts sob os quais não se pode fazer qualquer asserção em termos de capacidade de processamento e memória, contribui por tornar crítica a opção por uma determinada topologia.

A utilização de uma topologia com execução centrada no servidor, por exemplo, acarreta muito provavelmente as seguintes situações: a) o uso intenso de banda de rede, b) problemas com interatividade causada pela latência na comunicação pela rede e c) necessidade de configurações de hardware com alto poder de processamento para o servidor.

Note-se que para a implementação de aplicações com essa topologia, as linguagens e padrões utilizados devem permitir a criação de servidores capazes de manipularem ambientes gráficos, processar dados de entrada enviados pelo cliente, gerar e enviar na forma de imagens compactadas ao cliente pelo menos 10 quadros por segundo (taxa de quadros mínima para a apresentação de animações contínuas[98]).

Também devem permitir a criação de clientes que possam ser executados em navegadores Web, os quais devem ser capazes de obter e enviar dados de dispositivos de entrada, além de exibir as imagens produzidas pelo servidor.

Como vantagens da adoção dessa topologia, pode-se citar a possibilidade de uso de um cliente que necessite de configurações de hardware e de capacidades de processamento e de memória mínimas, o que facilita a obtenção da independência de plataforma.

Por outro lado, a adoção da topologia centrada no cliente implica, possivelmente nas seguintes situações: a) maior latência para o início da execução da aplicação, b) problemas para a promoção de independência de plataforma e c) necessidade de configurações de

hardware com alto poder de processamento para a execução no cliente.

As vantagens dessa topologia são: a) o baixo uso de banda de rede, b) a possibilidade de uso de servidores Web convencionais, c) a obtenção e processamento de dados oriundos de dispositivos de entrada e d) geração e exibição do ambiente gráfico.

Tecnologias para criação de aplicações nessa topologia devem permitir a alteração da configuração de software do cliente (navegador Web) e a solução de forma automatizada de eventuais problemas de permissão de uso de recursos do sistema e de instalação de bibliotecas e de outros artefatos de software necessários à execução da aplicação.

Assim, torna-se claro que a utilização de uma determinada topologia implica diretamente na escolha das tecnologias a serem empregadas, vez que os requisitos para cada uma delas são bastante distintos.

No estudo de caso realizado, em razão das aplicações possuírem características de uso e necessidades de interatividade bastante distintas, ambas as topologias foram empregadas.

No caso do Edugraph, por ser uma aplicação na qual a resposta a interação é fundamental, optou-se pela execução centrada no cliente. Evidentemente que isso implica na restrição ao uso de máquinas com baixa capacidade de processamento. Contudo, considerando-se que o compromisso com a funcionalidade da aplicação é mais importante do que com a máxima flexibilidade de uso, esse quadro não se torna tão crítico.

No SignType, ao contrário, a topologia escolhida foi a de processamento distribuído. Como esta é uma aplicação para auxilio no ensino acadêmico, o qual será realizado em laboratórios com máquinas com conhecida limitações de processamento, esta alternativa foi considerada como mais adequada. Note-se que as exigências quanto a velocidade da resposta a interação nessa aplicação não são críticas.

No SignType, eventuais atrasos no processamento de um evento de entrada não comprometem a utilização do aplicativo. Ao contrário, no Edugraph, um atraso pode representar a perda de execução de ações importantes durante o transcurso do jogo.

Por essas razões, para a determinação das linguagens e padrões mais adequados, a possibilidade de operação tanto na topologia centrada no cliente quanto na com processamento distribuído foi considerada. Esta é uma questão tecnológica importante para a

resolução dos demais problemas.

6.2 Tecnologias Utilizadas no Estudo de Caso

Com base na análise comparativa das tecnologias 2D emergentes (Tabela 6.1), pode-se observar que o *SVG* e a *SMIL* constituem a solução mais interessante do ponto de vista tecnológico, pois além de respeitarem os requisitos relativos à confecção de conteúdos, também possuem características que facilitam o estudo e a utilização, tais como baixo custo, padronização internacional e código aberto.

Requisito	SVG & SMIL	Flash
Manipulação de diversas mídias	Sim	Sim
Vasta capacidade de programação	Sim	Sim
Estruturas de dados complexas	Sim	Sim
Tratamento a dispositivos de entrada	Sim	Sim
Integração com outras tecnologias	Sim	Sim
Independência de plataforma	Sim	Sim
Suporte interno (built-in)	Não	Sim
Compactação de dados	Não	Sim
Baixo custo de utilização	Sim	Não
Padronização internacional	Sim	Não
Código aberto	Sim	Não
Vasta documentação	Sim	Sim
Ferramentas para edição e publicação	Não	Sim

Tabela 6.1: Tecnologias 2D

Contudo, ao se considerar questões de mercado e a necessidade das aplicações serem mais amigáveis ao usuário (*user-friendly*), a *Flash* se torna mais adequada. Pois ao contrário do *SVG* e da *SMIL*, ela não requer a instalação explícita do seu *plugin*. Dado que faz parte da instalação padrão de muitos sistemas e possui formas automáticas de atualização. Além disso, tanto os conteúdos bidimensionais do Edugraph, quanto os do SignType já haviam sido implementados com essa tecnologia, o que resultaria em retrabalho, no caso da adoção da sua concorrente.

Com respeito ao 3D, a análise comparativa das tecnologias emergentes (Tabela 6.2) revela um quadro mais favorável à solução formada por *X3D* e *Xj3D*. Principalmente porque essas tecnologias possuem padronização e fácil integração com outras tecnologias, requisitos não encontrados na sua concorrente, *Java 3D*.

É evidente, porém, que esse padrão e essa *toolkit Java* também possuem suas limitações. A principal delas certamente é a inexistência de uma estrutura previamente concebida para

Requisito	X3D & Xj3D	Java 3D
Manipulação de diversas mídias	Sim	Sim
Vasta capacidade de programação	Sim	Sim
Estruturas de dados complexas	Sim	Sim
Tratamento a dispositivos de entrada	Sim	Sim
Integração com outras tecnologias	Sim	Não
Independência de plataforma	Sim	Sim
Suporte interno (built-in)	Não	Não
Compactação de dados	Sim	Sim
Baixo custo de utilização	Sim	Sim
Padronização internacional	Sim	Não
Código aberto	Sim	Sim
Topologias de processamento variadas	Não	Não
Vasta documentação	Não	Sim
Ferramentas de edição e publicação	Sim	Sim

Tabela 6.2: Tecnologias 3D

a criação de aplicações com topologias de processamento variadas. Como visto anteriormente, esse é um requisito importante para a superação das limitações de processamento que podem ser encontradas em máquinas onde se deseja executar as aplicações.

Felizmente, esse é um problema contornável. A possibilidade de livre alteração dada pela licença de uso (código aberto) é o grande trunfo dessa *toolkit*. Graças a essa facilidade, módulos inteiros podem ser criados e incluídos na *toolkit*, permitindo assim o seu emprego em topologias de processamento para as quais não foi originalmente concebida.

Requisito	Java Media Framework	ECMAScript
Manipulação de diversas mídias	Sim	Sim
Estruturas de dados complexas	Sim	Sim
Tratamento a dispositivos de entrada	Sim	Sim
Independência de plataforma	Sim	Sim
Suporte interno (built-in)	Não	Sim
Baixo custo de utilização	Sim	Sim
Padronização internacional	Não	Sim
Vasta documentação	Sim	Sim
Comunicação com *SVG* e *SMIL*	Não	Sim
Comunicação com *Flash*	Não	Sim
Comunicação com *X3D* e *Xj3D*	Sim	Sim
Comunicação com *Java 3D*	Sim	Sim

Tabela 6.3: Tecnologias de Integração

No que diz respeito à tecnologia de integração dos conteúdos 2D e 3D, a análise comparativa (Tabela 6.3) revela que a *JavaScript / ECMAScript* é, dentre as tecnologias investigadas, a mais adequada à realização da tarefa. A extensão *Java Media Framework*, além de não ser suportada pelos navegadores Web mais populares, não permite a manipulação de conteúdos 2D em *SVG* com *SMIL*. Condição não problemática se for considerado que a tecnologia adotada para a confecção desse tipo de ambiente é a *Flash*.

Contudo, também para essa tecnologia 2D há severas restrições quanto à possibilidade

de manipulação. Somente conteúdos gerados sob a versão 2 do formato *swf* do *Flash* podem ser utilizados pela *JMF*, o que torna o seu uso inviável, dado que, atualmente, a versão do formato em uso é a 8. A *JavaScript / ECMAScript* ao contrário, se revela mais adequada para a aplicação em questão, pois conta com recursos suficientes para suprir a todos os requisitos anteriormente estabelecidos.

CAPÍTULO 7

SUPERANDO A INSUFICIÊNCIA DA CAPACIDADE DE PROCESSAMENTO

O problema de insuficiência de capacidade de processamento que pode ser encontrado na execução das AVBW foi enfrentado de formas distintas na produção dos softwares analisados durante a pesquisa. Não somente porque essas aplicações foram concebidas para uso em condições diferentes, mas, em especial, porque elas apresentam necessidades distintas em termos de resposta à interatividade.

Nas seções seguintes, são apresentadas as medidas adotadas para a solução desse problema no processo de criação de conteúdos bidimensionais e tridimensionais de ambos os softwares.

7.1 A Questão do Processamento nas Fases e Módulos 2D do Edugraph e do SignType

Para a criação dos conteúdos bidimensionais, em função da adoção da tecnologia *Adobe Shockwave Flash*, o processamento gráfico remoto (estratégia vista como mais adequada para a redução dos efeitos da insuficiência de capacidade de processamento) não pode ser empregada. O principal motivo para isso é que essa tecnologia dispõe de recursos limitados para a o estabelecimento de conexões com outros *hosts* e não tem código aberto, o que impossibilita o aprimoramento de tais recursos e, conseguentemente, a adoção de processamento distribuído. Por este motivo, a questão do processamento só foi considerada de fato na elaboração dos conteúdos tridimensionais, onde este problema é bem mais crítico.

7.2 A Questão do Processamento nas Fases 3D do Edugraph

A grande demanda por processamento das fases tridimensionais é claramente um dos maiores problemas existentes no Edugraph, mas certamente não o único. Concebido para ser executado em computadores particulares, esse jogo tem como um dos requisitos centrais de sua jogabilidade a necessidade de oferecer respostas rápidas a ação do usuário [1]. Comportamento que não é facilmente obtido em arquiteturas com processamento distribuído (tidas como as mais adequadas para a resolução dos problemas de insuficiência de processamento), tendo em vista que os atrasos na comunicação (inevitáveis nessa arquitetura) costumam interferir negativamente na reação do sistema aos eventos do usuário, causando um efeito de retardamento entre a ocorrência dos eventos e a apresentação das suas conseqüências.

Por esse motivo uma arquitetura com processamento local foi adotada para essas fases, dado que os níveis de interatividade exigidos por esses módulos da aplicação podem ser atingidos mais facilmente através dessa estratégia.

Essa opção, entretanto, teve conseqüências importantes em vários aspectos preponderantes. As principais delas estão relacionadas ao tratamento de eventos de entrada, à alterações na configuração do sistema, ao tráfego de informações pela rede e à execução da aplicação.

No tratamento de eventos de entrada ela reduziu as dificuldades encontradas. Não somente porque a latência entre o processamento e a exibição dos eventos deixou de existir, mas porque, neste caso, a tecnologia usada isenta o programador da aplicação de desenvolver a comunicação com os dispositivos de entrada.

Esse efeito ocorre porque a *Xj3D* (*toolkit Java* utilizada) implementa o controle dos dispositivos de entrada usuais, (mouse e teclado) e não usuais (tais como *game pads*), tornando direto o relacionamento entre a cena 3D e os eventos de entrada gerados pelo usuário. [2].

[1] Note-se que em muitas das fases deste jogo, o jogador precisa demonstrar bons reflexos, além de um raciocínio ágil e habilidade, o que requer, da parte do sistema, reações imediatas aos eventos de entrada gerados pelo usuário

[2] Em outras palavras, ao se criar uma aplicação gráfica 3D usando *Xj3D*, o programador não precisa se preocupar com o tratamento dos eventos de entrada gerados pelo usuário. Essa tarefa é feita pela

Na configuração do sistema a opção feita teve impacto negativo, tendo em vista que levou a necessidade de instalação de pacotes adicionais, além da Máquina Virtual *Java*. Situação que ocorre devido às aplicações com *Xj3D* utilizarem classes e bibliotecas externas para a sua execução, as quais não pertencem a instalação padrão do ambiente de execução *Java* (*Java Runtime Environment - JRE*).

Como bons exemplos de bibliotecas externas necessárias pode-se citar a *Java Open Audio Library*(*JOAL*) e a *Open Dynamics Engine for Java* (*ODEJava*), que são responsáveis pelo controle e manipulação de elementos de áudio em ambientes tridimensionais e pela promoção de recursos como detecção de colisão entre elementos da cena, respectivamente[3].

No tráfego de informações pela rede, ao contrário de na configuração do sistema, a opção feita teve impacto positivo. Como nessa topologia a aplicação não é dividida em cliente-servidor o uso da rede para a execução da aplicação não é necessário tal como ocorre naquelas que utilizam o processamento distribuído. Ao invés disso, devido ao processamento local da cena 3D, somente o código da *Applet* e da página *html* em que está contida precisam ser comunicados pela rede efetivamente. A instalação de pacotes adicionais na Máquina Virtual (citada no outro tópico) é um procedimento realizado uma única vez, na primeira execução da aplicação, o que não é uma condição permanente para a utilização do software.

Já no que diz respeito a execução da aplicação, o impacto do processamento local foi negativo, pois levou a necessidade do uso de assinaturas digitais nas *Applets*. Necessidade existente porque a Máquina Virtual *Java*, por razões de segurança, impõem várias restrições a esse tipo de software, as quais só podem ser revertidas pela identificação da procedência da aplicação e/ou pela autorização explícita do usuário para sua execução.

Um exemplo dessas restrições é o bloqueio, pela Máquina Virtual, ao acesso a funções de baixo nível do sistema (funções de hardware), com o intuito de preservar a integridade do sistema operacional. Este procedimento obstruiu a geração dos conteúdos tridimen-

própria *toolkit* e coordenada pelo criador da cena tridimensional em *X3D*, que determina, por meio de scripts de comportamento internos ao código X3D, quais são as ações a serem executadas em função da ocorrência de cada evento gerado pelo usuário.

[3]Essas bibliotecas precisam ser incluídas na Máquina Virtual *Java* antes do início da execução das aplicações com Xj3D, pois do contrário esta operação é abortada automaticamente pela Máquina Virtual.

sionais do jogo, pois eles são exibidos utilizando o pipeline de renderização da *OpenGL*[4] e só pode ser revertido com o uso de assinaturas digitais[5]

7.3 A Questão do Processamento no Módulo 3D do SignType

Diferentemente das fases tridimensionais do Edugraph, o módulo 3D do SignType não possui requisitos críticos de interação, razão pela qual, na sua criação, a topologia de processamento local utilizada nas fases 3D do Edugraph não foi empregada. Além disso, a topologia com processamento distribuído permite que o sistema possa ser executado em um cliente com baixa capacidade de processamento gráfico, bem como facilita a divulgação de novas versões e assegura uma maior relação entre o usuário e a página do sistema, dado que ele sempre acessa o sistema via página Web.

Caracterizada pela obediência ao modelo cliente-servidor, esta topologia é composta por quatro elementos básicos: servidor Web, servidor 3D, clientes Web e clientes 3D. As atribuições de cada um são:

- Servidor Web - servidor de páginas html convencional. É responsável por hospedar o cliente 3D (*Applet Java*) e a página em que está embutida;
- Clientes Web - navegadores Web convencionais. Dão acesso à pagina html onde se encontra o cliente 3D e comportam a sua execução por meio do *plugin Java*;
- Servidor 3D - servidor de processamento da cena tridimensional. É responsável por receber os comandos de interação enviados pelos clientes 3D e por enviar continuamente para eles fluxos de imagens compactadas como resultado do processamento da cena;
- Clientes 3D - *Applet Java* responsáveis por exibir as imagens da cena processada remotamente. Capturam os eventos de entrada gerados pelo usuário e os enviam para o servidor 3D.

[4] *OpenGL* (biblioteca gráfica utilizada pela *Xj3D* para a renderização das cenas 3D), o qual utiliza recursos diretos do hardware. Para se contornar este problema é necessário o uso de assinaturas digitais.

[5] Os procedimentos necessários para a realização de tais assinaturas digitais em *Applets* são melhor expostos no capítulo 8 que trata de questões associadas ao aumento da usabilidade da aplicação pela redução das dificuldades de instalação e operação pelo usuário.

Apesar de simples conceitualmente, esses elementos se relacionam de maneira peculiar durante a execução da aplicação. Isto porque, em especial, o cliente 3D é carregado diretamente da rede e executado no *plugin Java*, internamente ao cliente Web. Este comportamento condiz perfeitamente com o conceito de operação baseada na Web, já que o software é acessado pelo usuário pela Internet dentro do navegador Web convencional, sem que para isso necessite de configurações de hardware e de software específicas.

Uma visão da seqüência de eventos que ocorrem na execução da aplicação ajuda a entender melhor este processo:

1. Inicialmente, os servidores Web e 3D são iniciados e permanecem a espera de conexões com seus respectivos clientes;

2. O usuário, por meio do navegador Web (cliente Web), acessa a página na qual a aplicação tridimensional (cliente 3D) está embutida;

3. Se o *plugin Java* não estiver previamente instalado no navegador Web, a instalação é realizada automaticamente, com o acompanhamento do usuário;

4. A aplicação tridimensional (cliente 3D) é iniciada e a sua procedência é conferida pelo usuário por meio de caixa de aviso referente a assinatura digital[6];

5. A aplicação tridimensional (cliente 3D) estabelece conexão com o servidor 3D;

6. Um fluxo contínuo de imagens passa a ser enviado do servidor 3D ao cliente 3D;

7. O cliente 3D passa a apresentar as imagens da cena processada remotamente e a capturar os eventos de entrada gerados pelo usuário;

8. As informações de interação recebidas do cliente 3D são processadas pelo servidor 3D, gerando alterações na cena;

9. Os passos 7 e 8 são repetidos continuamente até o encerramento da aplicação;

[6]A assinatura digital só é conferida pelo usuário nos casos em que não é homologada por instituição de certificação digital internacionalmente reconhecida, como a VeriSign. Ela é desnecessária quando os servidores Web e 3D são mantidos no mesmo *host*.

10. No encerramento a conexão com o servidor é fechada e a aplicação gráfica (cliente 3D) é retirada de execução.

Os diagramas 7.1 e 7.2 auxiliam na compreensão de como o processamento da cena 3D é realizado remotamente para vários clientes e como cada cliente se relaciona com seu respectivo servidor.

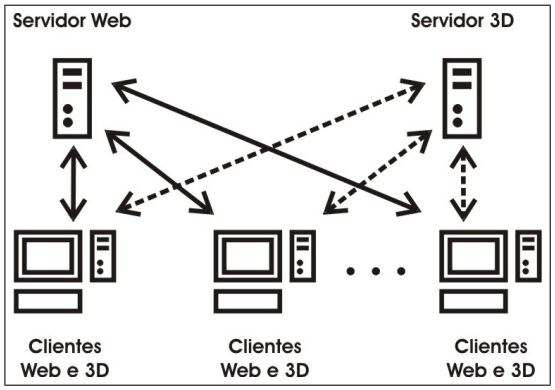

Figura 7.1: Arquitetura de processamento

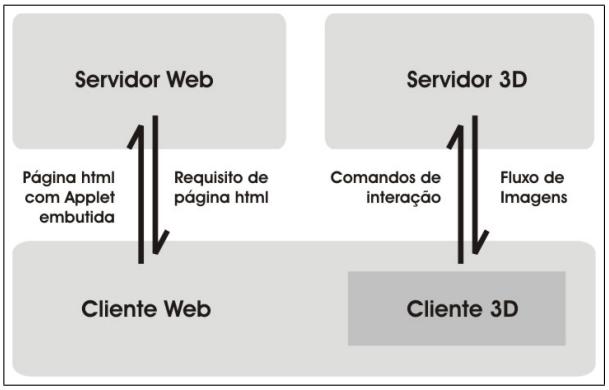

Figura 7.2: Cliente 3D e cliente Web

7.3.1 A Implementação da Arquitetura

O trabalho para a implementação da arquitetura anteriormente descrita foi um desafio à parte na criação do módulo tridimensional do SignType. Isto, não somente pelo envio de fluxos de imagens compactadas pela rede, mas, principalmente, porque a *toolkit Xj3D* precisou ser adaptada para prover essa funcionalidade.

Isso ocorre porque a extensão *Java* para a manipulação de cenas tridimensionais no padrão *X3D* não foi concebida originalmente para ser utilizada em softwares com processamento distribuído, mas sim para a criação de aplicações gráficas*stand-alone*. Isto que implica que os quadros provenientes do processamento da cena 3D são diretamente passados do *pipeline* de renderização para o componente da interface gráfica que os exibe localmente.

Em outros termos, nas aplicações criadas com essa biblioteca, as imagens resultantes do processamento da cena são diretamente apresentadas na interface da aplicação, na máquina em que ela estava sendo executada, sem que seja possível com isso, a sua manipulação e envio pela rede a eventuais clientes da aplicação. Por esse motivo, para que

elas pudessem ser usadas com esse intuito, um estudo mais aprofundado da *toolkit* foi realizado. O objetivo desse estudo era identificar uma maneira viável para mandar os quadros gerados, diretamente do pipeline gráfico para um pós-processamento, ao invés de para um componente de interface como ocorre originalmente, sem que com isso, porém, as demais funcionalidades da *Xj3D* fossem afetadas.

Em função de uma leitura minuciosa da documentação disponível da *Xj3D*, foi possível alterar os códigos dessa biblioteca e da *Aviatrix3D*[7], *API* para criação de grafo de cena do qual essa *toolkit* é derivada. Como um resultado relevante deste trabalho, se obteve um pacote para a extensão da *Xj3D* que permite que os quadros gerados sejam capturados ainda no *pipeline* gráfico e utilizados para outros fins que não seja a exibição na interface gráfica local.

Ressalte-se, porém, que apesar do sucesso nessa tarefa, outras questões igualmente importantes não puderam ser resolvidas definitivamente com esse estudo. Este é o caso, por exemplo, do tráfego de imagens pela rede, que pode ser realizado de diversas maneiras (via conexão convencional com *TCP* ou *UDP*, via *Web Services*, etc.). Como estas diversas alternativas demanda um estudo detalhado, o que requer tempo, essas pesquisas fora, sugeridas como trabalho futuro.

7.3.2 Conseqüências do Processamento Distribuído

Como esperado, a opção pela adoção do processamento distribuído trouxe implicações em diversas características do sistema. As principais, assim como nas fases tridimensionais do Edugraph, foram observadas no tratamento dos eventos de entrada, na configuração do sistema, no tráfego de informações na rede e na execução da aplicação.

No tratamento dos eventos de entrada, em razão do processamento da cena não se dar na mesma máquina em que ocorrem as ações do usuário, uma estrutura de comunicação precisou ser montada para o tratamento dos eventos de entrada no módulo do SignType.

Caracterizada pelo envio unidirecional de comandos, em um esquema mestre-escravo, ela é dividida em duas partes: uma no cliente e outra no servidor 3D. A porção situada

[7] *Aviatrix3D - API Java* para a criação de grafos de cena usando *OpenGL* e da *OpenAL*

no cliente tem como atribuições a captura dos eventos de entrada gerados pelo usuário e a codificação de tais eventos em comandos de alteração da cena 3D, a serem enviados ao servidor. Enquanto que a porção no servidor, por outro lado, é responsável pela decodificação dos comandos recebidos e a sua transformação em ações de alteração na cena 3D. O diagrama 7.3 mostra de uma forma mais simples como este processo ocorre.

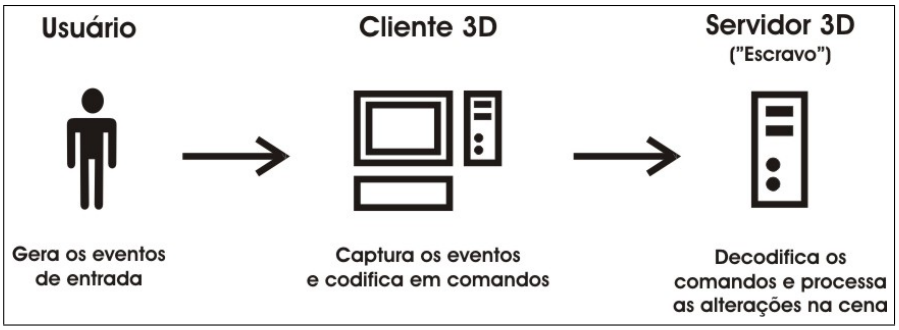

Figura 7.3: Esquema mestre-escravo.

Apesar da simplicidade conceitual, essa estrutura exigiu um grande esforço para ser implementada, pois a transformação dos comandos em alterações na cena 3D precisou ser realizada pela interface de acesso a cena (*Scene Access Interface - SAI*) ao invés da forma convencional, pela *toolkit Xj3D*. Em outros termos, ao invés de usar a extensão *X3D* para tratar os eventos de entrada internamente a cena tridimensional, um processador de comandos precisou ser criado via *SAI* para promover as alterações na cena.

Além disso, em razão da comunicação pela rede, a adoção dessa estrutura trouxe também problemas na resposta do sistema aos eventos de entrada por requerer que o tratamento indireto dos eventos de entrada seja feito através da comunicação entre o cliente e o servidor pela rede. Em função da complexidade deste estudo, ele foi considerado como excedente ao escopo da pesquisa realizada, e, assim, deixado como proposta para trabalhos futuros.

No que diz respeito a configuração do sistema, no módulo 3D do SignType, não foi necessária a instalação de pacotes adicionais a Máquina Virtual *Java*. Tendo em vista que a própria linguagem *Java* (e conseqüentemente o seu ambiente de execução) possui as funcionalidades necessárias para a execução da aplicação. Funcionalidades como descompactação e apresentação de imagens, estabelecimento de conexões com outros hosts

e comunicação com navegadores Web via *ECMAScript* pelo recurso *liveconnect*, por exemplo, que são utilizadas no cliente 3D do SignType para o recebimento das imagens do servidor e exibição ao usuário, bem como na comunicação com o conteúdo 2D pelo navegador Web.

Por este motivo, nos sistemas em que a Máquina Virtual *Java* está previamente instalada, o usuário pode executar esse módulo diretamente, sem precisar esperar que pacotes adicionais sejam baixados da Internet e instalados no Ambiente de Execução *Java*, o que torna o seu uso bastante mais simples do que o das fases 3D do Edugraph.

No tráfego de informações pela rede, por conta do processamento remoto da cena tridimensional, o uso da rede durante a execução da aplicação no módulo 3D do SignType é intenso. Esta condição é bem diferente da encontrada nas fases tridimensionais do Edugraph, em que a rede não é utilizada pela aplicação para essa atividade.

A direção do maior fluxo de mensagens é do servidor para o cliente, pois os dados enviados do cliente para o servidor se restringem a poucos bytes a cada ação do usuário. Devido a sua grande complexidade, alternativas para redução do uso da banda de comunicação não foram buscadas no âmbito desta pesquisa. Motivo pelo qual ele é proposto para resolução em trabalhos futuros.

Ao contrário da questão principal na execução da aplicação, a necessidade de uso de assinaturas digitais. Como citado anteriormente, na seção sobre as soluções utilizadas no Edugraph, por questões de segurança, a Máquina Virtual *Java* impõe um conjunto extenso de restrições às *Applets*, as quais só podem ser revertidas com a identificação da procedência do software via assinatura digital. Razão pela qual no módulo 3D do SignType, apesar da adoção de uma topologia de processamento completamente diferente, a reversão dessas restrições também foi trabalhada. Dessa vez, entretanto, devido ao uso de funções de hardware, como no Edugraph, mas pelo estabelecimento de conexão para envio e recebimento de dados de outro host da Internet.

CAPÍTULO 8

FACILITANDO A INSTALAÇÃO E O USO

Nielsen no seu livro "Usability Engineering"[45] é taxativo em relação a necessidade de se facilitar o uso das aplicações. Ele ressalta que esta facilidade deve ser considerada não somente na criação da aplicação em si, mas também em outros aspectos relacionados, como na escrita de manuais e na criação de mecanismos de instalação e atualização.

Contudo, como descrito anteriormente, o processo de instalação dos elementos necessários à execução dos AVBW costuma ser bastante variável, dependendo do sistema operacional e até mesmo do navegador Web utilizado, situação que prejudica a utilização desse tipo de software. Neste contexto, nos casos estudados, medidas foram tomadas para reduzir as dificuldades de instalação das aplicações.

A primeira destas medidas foi a escolha por linguagens e padrões que oferecessem mecanismos para a instalação automática de todos os elementos necessários a execução das aplicações. Esta escolha ocorreu ainda na fase de definição das tecnologias a serem utilizadas e foi um dos motivos pelos quais o *Adobe Shockwave Flash* e a *Xj3D* foram selecionadas para a criação dos conteúdos bidimensionais e tridimensionais, respectivamente.

A segunda medida foi a determinação, ainda na fase de desenvolvimento das aplicações, dos fatores capazes de trazer reflexos negativos à usabilidade. Em especial, foram analisados os fatores que não dependem do conteúdo específico da aplicação em si, mas que podem ser encontrados também no desenvolvimento de aplicações semelhantes.

A terceira foi a busca por soluções genéricas capazes de reduzir o efeito desses fatores na usabilidade desses softwares. Os fatores observados e os procedimentos apurados estão descritos nas seções seguintes.

8.1 Instalação dos *Plugins Java* e *Shockwave Flash*

Uma das características mais básicas das aplicações baseadas na Web é o fato de uso se dar através de navegadores Web convencionais. Contudo, em alguns casos, isso só é possível após a instalação de *plugins* nos navegadores, pois eles nem sempre oferecem suporte na sua configuração padrão para a execução dessas aplicações.

Nos casos analisados, em função das opções tecnológicas feitas, esse problema foi facilmente resolvido. Pois a instalação automática dos *plugins Flash* e *Java*[1] necessários, pode ser promovida por páginas html com código ECMAScript adequado.

Além disso, essas tecnologias também dispõem de ferramentas para auxiliar a criação dessas páginas Web especiais, o que facilita em muito o trabalho dos desenvolvedores.

Um bom exemplo dessas ferramentas é o *HTMLConverter*, que faz parte do kit de desenvolvimento *Java* (*Java Development Kit - JDK*). Ela recebe como entrada páginas html convencionais com *Applets* incorporadas e produz novas páginas semelhantes com código *ECMAScript* capazes de instalar o *plugin Java* em diferentes navegadores de diferentes sistemas operacionais, tornando desnecessária a formulação de esquemas especiais para a verificação da disponibilidade dos *plugins* e a instalação automática nesses sistemas.

8.2 Instalação de Pacotes Adicionais à Máquina Virtual Java

Nas fases 3D do Edugraph, uma das conseqüências mais negativas do uso do processamento local foi a necessidade de instalação de pacotes adicionais ao Ambiente de Execução *Java* (*Java Runtime Enviromment - JRE*). Em situações convencionais essa tarefa é realizada pelo usuário da aplicação, que precisa baixar "manualmente"as versões corretas dos instaladores e proceder a instalação das bibliotecas necessárias seguindo passos determinados. Isto representa um fator de redução da satisfação do usuário e conseqüentemente da usabilidade.

Nas fases tridimensionais do Edugraph, essa realidade foi diferente, pois nelas a tarefa de instalação das bibliotecas necessárias foi inteiramente delegada a Máquina Virtual

[1] A *toolkit Xj3D* é uma extensão da linguagem *Java*

Java. Isto só foi possível graças a adoção dos métodos descritos por Wijkman e outros em "*Transparent Java Standard Extensions with Native Libraries on Multiple Platforms*"[84].

Esses métodos são baseados, em grande parte, na confecção adequada de descritores de conteúdo (*manifests*) de arquivos compactados *Java* (*Java Archives - JAR*). Isto permite que as aplicações *Java* instalem automaticamente na Máquina Virtual as bibliotecas que precisam para serem executadas, eliminando, em termos práticos, a necessidade da ação do usuário.

Ainda assim, a questão da inclusão de pacotes adicionais não foi resolvida completamente. Como citado anteriormente, o tamanho total das bibliotecas a serem instaladas em um sistema com configuração padrão pode chegar a vários mega bytes, o que pode exigir um tempo considerável para a carga em redes com pouca disponibilidade de banda. Esta espera pode reduzir a satisfação do usuário e atentar contra a usabilidade.

Entretanto, em "*Library-Based Java Applications Extracting: Reducing the size of Java applications by creating an application extractor.*"[39], Tip e outros apresentam um conjunto de ferramentas capazes de remover classes, métodos e até mesmo atributos não utilizados em bibliotecas que fazem parte de aplicações *Java*. Segundo eles, estas ferramentas podem, em alguns casos, reduzir o tamanho dos pacotes a serem instalados para cerca de 10% do tamanho original, o que, certamente, diminuiria o tempo de instalação.

Essa possibilidade não pode ser averiguada nesta pesquisa em razão, principalmente, de falta de tempo, motivo pelo qual foi deixada como proposta a ser investigada em trabalhos futuros.

8.3 Assinatura Digital de Applets

Em razão das limitações de acesso impostas pela Máquina Virtual Java, tanto no conteúdo 3D do SignType quanto nos do Edugraph, foi analisada a necessidade de assinatura digital nas *Applets*. Fruto da exigência da identificação da procedência do software pelo usuário, ela é um procedimento que segundo Pistoia e outros[70], Gong e outros[57] e Knudsen[54], pode ser realizado em 5 passos básicos [2]:

[2]As ferramentas citadas pertencem ao Kit de Desenvolvimento Java (*Java Development Kit- JDK*)

1. Compilação do código fonte pelo compilador *javac* e geração das classes interpretáveis;;

2. Compactação das classes geradas e geração de um único pacote com toda a aplicação pela ferramenta *jar*;

3. Geração das chaves criptográficas necessárias para a assinatura da aplicação, pela ferramenta *keytool*;

4. Assinatura propriamente dita da aplicação, por meio da ferramenta *jarsigner*;

5. 5. Exportação do certificado digital criado, pela ferramenta *keytool*, para registro em autoridade certificadora internacionalmente reconhecida.

A redução da usabilidade é um impacto claro do processo de assinatura. Quando inicia uma aplicação assinada digitalmente no *plugin Java* em seu navegador Web, o usuário é evocado por meio de uma janela de aviso a autorizar a exibição do conteúdo, o que nem sempre é facilmente entendido por ele, pois os motivos que fundamentam o pedido de autorização nem sempre são evidentes. Esse problema pode ser facilmente resolvido pelo registro do certificado digital gerado em autoridade certificadora internacionalmente reconhecida, tal como a *VeriSign*. Quando o certificado digital é emitido por uma fonte confiável a autorização para a execução não é requisitada, o que reduz os transtornos oferecidos ao usuário.

CAPÍTULO 9

RESULTADOS

A realização dos estudos de casos trouxe diversos resultados significativos. O mais evidente deles é a obtenção de uma primeira versão do SignType, pois demonstra a utilização prática das técnicas descritas. Contudo, outros resultados importantes, os quais podem trazer benefícios diretos ao desenvolvimento de AVBW, merecem destaque e, por isto, são descritos abaixo.

- A avaliação de que as tecnologias *Flash*, *X3D/Xj3D* e *ECMAScript* são atualmente as mais adequadas para a criação de conteúdos bidimensionais, tridimensionais e para a realização da comunicação entre os ambientes bi e tridimensionais gerados, respectivamente. Este dado é uma solução passível de resolver o problema encontrado pelos desenvolvedores na definição das linguagens e padrões mais adequados a criação de AVBW.

- As tabelas 6.1, 6.2 e 6.3, criadas como parte do estudo a respeito das tecnologias mais promissoras para a criação dos AVBW. Estas tabelas expõem de forma concisa, comparativa e de fácil compreensão as principais características destas tecnologias. Isto tanto ajuda o desenvolvedor de AVBW quanto abre caminho para novas discussões e pesquisas.

- Os dados coletados nas observações feitas sobre as conseqüências trazidas pelas topologias de processamento adotadas, os quais são expostos no corpo da dissertação e sintetizados na tabela 9.1. Estes dados podem ajudar os desenvolvedores a identificar a topologia mais adequada para a criação de um determinado AVBW, com base em seus requisitos de interação, de uso de rede, de operação e atualização de sistemas e da capacidade de processamento e de armazenamento das máquinas.

- O pacote para a extensão da *Xj3D* permite o seu uso na criação de aplicações com processamento gráfico tanto local, quanto remoto. Esta determinação oferece aos desenvolvedores uma opção para a criação de AVBW. Ele pode optar entre duas topologias com características distintas de interatividade, de uso de banda e de disponibilidade de recursos para processamento. Esta opção é parte do processo de definição de uma solução para a questão da insuficiência de processamento em determinados tipos de ambientes.

- A técnica que permite a instalação assistida dos *plugins* necessários para a execução das aplicações. Pois reduz o esforço exigido do usuário para a realização da tarefa e conseqüentemente para utilização da aplicação. Implicação que faz parte das soluções encontradas para a questão da dificuldade de instalação.

- As técnicas que automatizam a instalação na Máquina Virtual *Java* das bibliotecas necessárias para a execução do conteúdo tridimensional processado localmente. Elas eliminam os transtornos oferecidos aos usuários para a instalação dessas bibliotecas. Esta facilidade reduz o número de erros cometidos pelos usuários na realização dessa tarefa, o que aumenta o nível de usabilidade da aplicação.

	Topologia de Processamento	
Implicações	**Local**	**Remoto**
Na configuração do sistema	Pacotes adicionais precisaram ser instalados na Máquina Virtual *Java*	Nenhum pacote adicional precisou ser instalado na Máquina Virtual Java
Na execução da aplicação	As *Applets* precisaram ser assinadas digitalmente	A *Applet* precisou ser assinada digitalmente
No uso de banda de comunicação	A rede não foi utilizada durante a execução da aplicação	A rede foi intensamente usada durante a execução da aplicação
No tratamento de eventos de entrada	Foram obtidas facilidades para o tratamento dos eventos	Grandes dificuldades foram enfrentadas na resposta do sistema aos eventos de entrada

Tabela 9.1: Topologias de Processamento Utilizadas

9.1 Resultados Colaterais

Além dos resultados principais anteriormente descritos, outros adicionais foram obtidos a partir de problemas secundários enfrentados no desenvolvimento das aplicações analisadas na pesquisa. Problemas tais como a impossibilidade de comunicação síncrona entre os ambientes 2D e 3D, necessidade de redução do tamanho das bibliotecas adicionais e restrições a execução impostas pela Máquina Virtual *Java*, por exemplo, apesar de não pertencerem ao foco principal desta pesquisa, também são importantes na criação de AVBW.

A seguir há uma descrição desses problemas adicionais e das soluções encontradas para eles:

- A instalação de bibliotecas adicionais na Máquina Virtual *Java*, mesmo quando procedida automaticamente, pode reduzir o nível de usabilidade da aplicação, pois pode exigir um tempo longo para ser realizada. Note-se que o tamanho somado das bibliotecas pode chegar a 12MB, dependendo do sistema operacional em que serão instaladas.

  - Softwares como *JShrink*, *mBird* e *IBM Jax* podem ser utilizados para reduzir o tamanho de aplicações Java[39]. Eles removem classes, métodos e até mesmo atributos não utilizados em bibliotecas, o que pode, em alguns casos extremos, diminuir o tamanho total das aplicações a cerca de 1/10 do seu tamanho original. Esta estratégia é capaz de amenizar o problema do tempo gasto para a instalação das bibliotecas.

- Devido ao fato de que os ambientes bidimensionais e tridimensionais são criados como aplicações separadas (e executadas em *plugins* distintos), não é possível fazer o envio síncrono de informações entre eles, pois ao enviar dados a um ambiente, o outro não tem garantias de que eles serão recebidos adequadamente. Esta situação pode acarretar falha de comunicação, pois o ambiente gráfico de destino pode perder as informações recebidas por elas não estarem em um estado de processamento adequado ao perfil do ambiente.

  - Uma maneira simples encontrada para contornar o problema da comunicação foi a utilização do conceito de *Mailboxes*. Este conceito, oriundo da teoria da comunicação entre processos em sistemas operacionais, pode ser empregado para promover a comunicação assíncrona entre os ambientes bi e tridimensionais, pois pode ser facilmente implementado com os recursos existentes na *ECMAScript*.

- Realizar a comunicação via *ECMAScript* , as aplicações 2D e 3D em *Flash* e *Java* (*Xj3D*) convertem em texto (*strings*) os dados que serão enviados. Assim, para que os dados sejam utilizados pela aplicação de destino, ela precisa processar o texto recebido para recuperá-los. Contudo, dependendo do volume e do tipo de informações comunicadas, essa decodificação pode ser bastante complexa, pois alguns tipos de dados existentes na linguagem Java não estão disponíveis também nas linguagens *ActionScript* e *ECMAScript* da qual ela é derivada.

  - O processamento do texto com os dados da comunicação pode ser facilitado pela adoção da *XML*. Tanto o *Flash* quanto o *Java* possuem bibliotecas que permitem o *parsing* de arquivos gerados a partir dessa metalinguagem. Este recurso, além de eficaz, não acarreta problemas de instalação de pacotes e/ou extensões, tendo em vista que elas fazem parte da distribuição padrão dessas tecnologias.

- A adoção do processamento distribuído para a solução do problema da insuficiência de processamento acarreta o uso acentuado de banda de comunicação, o que impede o uso dessas aplicações em redes com largura de banda reduzida. Este fato se contrapõe ao princípio de aplicações poderem ser executadas em qualquer máquina.

  - A utilização de protocolos de envio de mídias dependentes do tempo, tais como o *Real-Time Protocol* (*RTP*), e de padrões de compactação de vídeo, tais como o H.323, ambos implementados na *Java Media Framework API*, pode reduzir a quantidade de dados que trafegam na rede e, conseqüentemente, se adequar aos níveis atuais de largura de banda. Os recursos destas tecnologias se mantêm

invisíveis para o usuário e evitam a transmissão de dados desnecessários, pois dispensam a retransmissão de dados recebidos com atraso expressivo ou erro pelo fato de explorarem as redundâncias existentes entre os quadros (imagens) adjacentes.

CAPÍTULO 10

DISCUSSÃO

O objetivo primordial da pesquisa de encontrar soluções eficientes para algumas das questões preponderantes na criação de AVBW foi alcançado. Contudo, é importante ressaltar que em alguns contextos, ainda que as soluções descritas nessa dissertação sejam implementadas adequadamente não é possível assegurar a correta execução dessas aplicações. Tal fato ocorre porque há fatores nas soluções descritas que podem limitar ou até mesmo impedir a sua ação.

Alguns desses fatores estão relacionados às próprias técnicas empregadas para resolver os problemas e são, portanto, intrínsecos a elas. Outros contudo, são devidos a aspectos externos e por essa razão, muitas vezes, não podem ser contornados por estratégias alternativas.

Como fatores intrínsecos às técnicas utilizadas, além daqueles já descritas no corpo do texto e nos resultados, pode-se citar:

a) A vulnerabilidade a ataques e utilização indevida por usuários mal-intencionados,

b) A impossibilidade de redimensionamento dos conteúdos tridimenionais da aplicação sem perda de informação e/ou de qualidade visual e

c) A possibilidade de impedimento de execução dos conteúdos tridimensionais.

Os quais são resultantes do uso da linguagem *ECMAScript* na comunicação entre os ambientes gráficos 2D e 3D, do emprego de imagens convencionais compostas por pixels no processamento remoto e do possível bloqueio ao estabelecimento de conexões imposto por serviços tais como *proxys* e *firewalls*, respectivamente.

Para amenizá-los, entretanto, há a opção de:

a) Reduzir ao máximo a realização de processamento em *ECMAScript* pela transferência das funcionalidades de comunicação principais para os ambientes gráficos,

b) Buscar gerar imagens vetoriais a partir da cena tridimensional,

c) Empregar tecnologias para envio de mensagens que não estejam sujeitas a barramento por firewall, tais como Web Services.

Entretanto essas estratégias podem também não ser completamente efetivas na resolução desses problemas e demandam bastante trabalho para implementação. Além, é claro, de poderem resultar em novas limitações e problemas que podem influenciar negativamente a execução da aplicação.

Como fatores externos às técnicas utilizadas, pode-se citar a impossibilidade de uso em navegadores cujo *plugin Java* tenha sido desabilitado pelo usuário e a ineficácia da instalação automática de bibliotecas, causada pelo impedimento de acesso a arquivos de sistema definido por políticas de permissões em sistemas operacionais multi-usuários. Os quais só podem ser amenizados, possivelmente, pela inclusão de manuais de utilização e/ou de recursos instrucionais que orientem o usuário acerca das configurações do sistema necessárias para a execução da aplicação. Trabalho que, espera-se, será realizado em pesquisa futura já programada.

CAPÍTULO 11

CONCLUSÕES

As dificuldades enfrentadas nesta pesquisa demonstraram que a obtenção de soluções que abarcam todos as questões relacionadas à implementação das AGBWs exigiria, pela sua complexidade e extensão, um tempo de trabalho muito além do regulamentar disponível para o desenvolvimento de uma dissertação de mestrado. No entanto, apesar das restrições das soluções encontradas, as considerações abaixo ressaltam a relevância deste trabalho.

A correta definição do problema e do seu alcance, bem como o gerenciamento das diversas atividades envolvidas na pesquisa mostraram a correta aplicação de conceitos de metodologia científica. Por outro lado, o caráter exploratório e até inovador da pesquisa dificultou a aplicação rigorosa do método.

Usualmente, a obtenção na pesquisa científica de uma solução para um dado problema não implica necessariamente em um ponto final. Muitas vezes, surgem vários novos questionamentos os quais irão demandar novas pesquisas. O autor deste trabalho vislumbra várias formas de se otimizar soluções encontradas nesta pesquisa, as quais não foram desenvolvidos em função de limitação de tempo. O caráter inovador de algumas destas novas pesquisas (vetorização de imagens renderizadas no servidor para envio ao cliente de forma mais compactada, por exemplo) abre campo para trabalhos futuros a serem desenvolvidos em um curso de doutorado, o qual o autor espera iniciar em breve.

O autor também pretende continuar interagindo com o Laboratório de Animação Interativa do Departamento de Design da UFPR de forma a participar no desenvolvimento do projeto EEHouse[1] , cujo objetivo é desenvolver um jogo para ensino de conceitos de eficiência energética. Este projeto tem suporte financeiro da FINEP e deverá ser iniciado em abril de 2007. Note-se que toda a pesquisa resultante deste trabalho será de extrema importância para o desenvolvimento deste projeto.

[1]EEHouse - projeto de pesquisa cujo objetivo é criar um jogo educacional, baseado na Web, para o ensino de conceitos de eficiência energética.

O autor considera que os resultados obtidos neste trabalho dão margem a publicações de peso. Alguns artigos já foram publicados em eventos nacionais, no entanto, o grande tempo investido na conclusão deste trabalho, bem como a obtenção dos dados mais importantes somente na fase final da pesquisa, dificultaram a elaboração de artigos passíveis de submissão a veículos de maior relevância. No entanto, o autor já traçou uma meta ambiciosa de publicações com o intuito tanto de divulgar o trabalho quanto de conquistar um currículo que lhe abra as portas para um curso de doutorado.

REFERENCIAS BIBLIOGRAFICAS

[1] Adobe Systems Incorporated, Disponível em: <http://www.adobe.com/products/golive/overview.html> Acesso em: 04 abr. *Adobe Go Live CS2*, 2006.

[2] Adobe Systems Incorporated, Disponível em: <http://www.adobe.com/svg/pdfs/illustrator_svg.pdf> Acesso em: 04 abr. *Adobe Illustrator*, 2006.

[3] Adobe Systems Incorporated, Disponível em: <http://http://www.adobe.com/svg/viewer/install/main.html> Acesso em: 04 abr. *Adobe SVG Viewer*, 2006.

[4] Adobe Systems Incorporated, Disponível em: <http://www.macromedia.com/br/buy/> Acesso em: 04 abr. *Opções de Compra de Produtos Adobe*, 2006.

[5] Seffah Ahmed e Metzker Eduard. The obstacles and myths of usability and software engeneering. *Communications of the ACM*, 47(12):71–77, dez de 2004.

[6] Terrazas Alejandro, Ostuni John, e Barlow Michael. *Java Media APIs: Cross-Platform Imaging, Media, and Visualization*. Sams Publishing, 2002. 848p.

[7] Magalhães Alexandre. *O que esperar da Internet nos próximos anos?* IBOPE Inteligência, IBOPE//NetRatings, Internet, Notícias 2006, Disponível em: <http://www.ibope.com.br/> Acesso em: 04 abr., 2006.

[8] Battaiola André Luiz e Jungles dos Santos Rangel. Implementação das fases 2d do jogo educacional edugraph. *Simpósio Brasileiro para Jogos de Computador e Entretenimento Digital*, páginas 75–83, nov de 2005.

[9] Holzinger Andreas. Usability engeneering methods for developers. *Communications of the ACM*, 48(1):71–76, jan de 2005.

[10] Autodesk, Inc., Disponível em: <http://images.autodesk.com/adsk/files/gmk_maya7_overview.pdf> Acesso em: 04 abr. *Autodesk Maya*, 2006.

[11] Autodesk, Inc., Disponível em: <http://www.autodesk.com/3dsmax> Acesso em: 04 abr. *Discreet 3D Studio Max*, 2006.

[12] Dalgarno Barney. Technologies supporting highly interactive learning resources on the web: A analysis. *Jornal of Interactive Learning Research*, páginas 153–171, 2001.

[13] Chen Bing-Yu e Nishita Tomoyuki. jgl and its applications as a web3d platform. *SIGWEB: ACM Special Interest Group on Hypertext, Hypermedia, and Web*, páginas 85–91, 2001.

[14] The Blender Foundation, Disponível em: <http://www.blender.org/cms/Home.2.0.html> Acesso em: 04 abr. *Blender - Open Source 3D Graphics Creation*, 2006.

[15] Mathews Brian, Lee Daniel, Dister Brian, Bowler John, Cooperstein Howard, Jindal Ajay, Nguyen Tuan, Wu Peter, e Sandal Troy. *Introduction to Vector Markup Language (VML).* MSDN Library, Disponível em: <http://msdn.microsoft.com/workshop/author/vml/default.asp> Acesso em: 22 jan., 2007.

[16] Mathews Brian, Lee Daniel, Dister Brian, Bowler John, Cooperstein Howard, Jindal Ajay, Nguyen Tuan, Wu Peter, e Sandal Troy. *Vector Markup Language Reference.* MSDN Library, Disponível em: <http://msdn.microsoft.com/workshop/author/VML/Shape/ e_shape.asp> Acesso em: 22 jan., 2007.

[17] Mathews Brian, Lee Daniel, Dister Brian, Bowler John, Cooperstein Howard, Jindal Ajay, Nguyen Tuan, Wu Peter, e Sandal Troy. *Vector Markup Language (VML).* Submission to the World Wide Web Consortium, Disponível em: <http://www.w3.org/TR/NOTE-VML> Acesso em: 22 jan., 2007. NOTE-VML-19980513.

[18] Pierce Charles S. *Semiótica*, capítulo (Fragmento manuscrito). Perspectiva, São Paulo, 1903.

[19] Pierce Charles S. *Semiótica*, capítulo Nomenclature and divisions of triadic relations, as far as they are determined. Perspectiva, São Paulo, 1903.

[20] Presser Clifton G.M. A java web application for allowing multiuser collaboration and exploration of existing vrml worlds. *SIGGRAPH: ACM Special Interest Group on Computer Graphics and Interactive Techniques*, páginas 85–92, 2005.

[21] Partridge Craig. *Gigabit Networking (Paperback)*. Addison-Wesley Professional, 1994.

[22] Murray Craig S. e Church Justin Everett. *Macromedia Flash MX Game Programming*. Premier Press, 2003.

[23] Robert W. Crandall e J. Gregory Sidak. Video games serius business for america's economy. *Entertainment Software Association (ESA)*, páginas 1–48, 2006.

[24] Selman Daniel. *Java 3D Programming*. Manning Publications Co., 2004.

[25] Bogaard Daniel S., Vullo Ronald P., e Cascioli Christopher D. Svg for educational simulations. *ACM: Association for Computing Machinery*, 2004.

[26] Goodman Danny. *JavaScript Bible*. IDG Books Worldwide, New York, 3. ed edition, 1998. 1015p.

[27] Flanagan David. *JavaScript: The Definitive Guide*. O'Reilly and Associates, Sebastopa, 3. ed. edition, 1998. 776p.

[28] Nadeau David R. Building virtual worlds with vrml. *IEEE Computer Graphics and Applications*, 19(2):18–29, abr de 1999.

[29] Bouvier Dennis J. *Getting Started with the Java 3D API*. Sun Microsystems, Inc., Disponível em: <http://java.sun.com/developer/onlineTraining/java3d/j3d_tutorial_ch1.pdf> Acesso em: 04 abr., 2006. 2000 v. 1.5.1.

[30] Bulterman Dick. *Synchronized Multimedia Integration Language (SMIL 2.1)*. World Wide Web Consortium (W3C), Disponível em:

<http://www.w3.org/TR/2005/CR-SMIL2-20050513/> Acesso em: 04 abr., 2006.

[31] Document Engeneering Lab - The University of Nottingham, Disponível em: <http://www.ep.cs.nott.ac.uk/projects/SVG/flash2svg/> Acesso em: 04 abr. *Macromedia Shockwave Flash to SVG Conversion*, 2006.

[32] Document Engeneering Lab - The University of Nottingham, Disponível em: <http://www..eprg.org/research/SVG/ps2svg/> Acesso em: 04 abr. *PostScript to SVG Conversion*, 2006.

[33] Brutzman Don. The virtual reality modeling language and java. *Communication of the ACM*, 41(6):57–65, jun de 1998.

[34] Twilleager Doug e et al. Java technologies for games. *ACM Computers in Entertainment*, 2(2):1–9, abr de 2004.

[35] Comer Douglas E. *Internetworking with TCP/IP*. Prentice Hall, Upper Sidder River, New Jersey, 4th edition, 2000.

[36] The Eclipse Foundation, Disponível em: <http://www.eclipse.org/> Acesso em: 04 abr. *Eclipse*, 2006.

[37] ECMA General Assembly, Disponível em: <http://java.sun.com/products/java-media/jmf/2.1.1/guide/index.html> Acesso em: 04 abr. *Standard ECMA-262 (ECMAScript Language Specification)*, 3. ed. edition, 2006. ano 1999, 188p.

[38] Anati Emmanuel. *Constants in 40,000 years of art*. Origins of semiosis: sign evolution in nature and culture., Winfried Nöth, Berlin, 1994. New York: Mouton de Gruyter.

[39] Tip Frank, Sweeney Peter F., e Laffra Chris. Extracting library-based java applications. *Commun. ACM*, 46(8):35–40, 2003.

[40] Free Software Foundation, Inc., Disponível em: <http://www.gnu.org/licenses/gpl.txt> Acesso em: 04 abr. *GNU General Public License (GPL)*, 2006.

[41] Flammia Giovanni. Smil makes web applications multimodal. *IEEE Intelligent Systems*, páginas 12–13, ago de 1998.

[42] IBM Inc., Disponível em: <http://www.gnu.org/licenses/gpl.txt> Acesso em: 04 abr. *Common Public License Version 1.0 (CPL)*, 2006.

[43] Inria, Disponível em: <http://http://wam.inrialpes.fr/software/limsee2/> Acesso em: 04 abr. *The cross-platform SMIL 2.0 authoring tool*, 2006.

[44] ISO/DIS 9241-11. *Guidance on Usability. Ergonomic Requirements for Office Work with Visual Display Terminals (VDT).*

[45] Nielsen Jakob. *Usability Engineering.* Morgan Kaufmann, 1993.

[46] Ying Jianghui, Gracanin Denis, e Lu Chang-Tien. Web visualization of geo-spatial data using svg and vrml/x3d. IEEE Computer Society Press, editor, *Proceedings of the Third*, páginas 497–500, Hong Kong, 2004. International Conference on Image and Graphics. n. 3,.

[47] Huang Jiung-Yao, Fang-Tso Chao-Tsou, e Chang Jia-Lin. A multiuser 3d web browsing system. *IEEE Internet Computing*, 2(5):70–81, nov de 1998.

[48] Coelho Neto João Teixeira. *Semiótica, informação e comunicação: diagrama da teoria do signo.* Perspectiva, São Paulo, 1980.

[49] Bowler John. *Scalable Vector Graphics (SVG) 1.0 Specification.* World Wide Web Consortium (W3C), Disponível em: <http://www.w3.org/TR/2001/REC-SVG-20010904/> Acesso em: 04 abr., 2006.

[50] Deely John. *Semiótica básica.* Ática, São Paulo, 1982 (1990).

[51] Lehman John A. Business graphics: A taxonomy for information systems managers. *Data Base Fall*, páginas 24–31, 1986.

[52] Barrilleaux Jon August. *3D User Interfaces with Java 3D*. Manning Publications Co., Greenwich, 2000.

[53] Gay Jonathan. *The History of Flash*. Macromedia Graphics Inc., Disponível em: <http://<http://www.macromedia.com/macromedia/events/john_gay/index.html> Acesso em: 04 abr., 2006.

[54] Knudsen Jonathan. *Java Cryptography*. O'Reilly Publications, 1 ed. edition, mai de 1998.

[55] Neo Ker Sin, Lin Qingping, e Gay Robert K.L. A web-based system for interactive visualization of scientific concepts. *SIGGRAPH: ACM Special Interest Group on Computer Graphics and Interactive Techniques*, páginas 155–158, 2004.

[56] Rosenblum Lawrence e Macedonia Michael. The java 3d api and virtual reality. *IEEE Computer Graphics and Applications*, páginas 12–15, jun de 1999.

[57] Gong Li, Ellison Gary, e Dageforde Mary. *Inside Java 2 Platform Security: Architecture, API Design, and Implementation*. Prentice Hall, 2ed. edition, 2001.

[58] Wang Lihui, Wong Brian, Shen Weiming, e Lang Sherman. A java 3d-enabled cyber workspace. *Communications of the ACM*, 45(11):45–50, nov de 2002.

[59] deCarmo Linden. *Core Java Media Framework*. Prentice Hall, New Jersey, 1999.

[60] Santaella Lucia. *O que é Semiótica*. Brasiliense, São Paulo, (1983) 1987.

[61] Santaella Lucia. *Teoria Geral dos Signoss: como as linguagens significam as coisas*. Pioneira, São Paulo, (1995) 2000.

[62] Santaella Lucia. *Semiótica Aplicada*. Thompson, São Paulo, 2002.

[63] Soares Luciano Pereira. *X3D Frequently Asked Questions.* Web 3D Consortium, Disponível em: <http://www.lsi.usp.br/ lsoares/x3d/faq.html#process-5> Acesso em: 04 abr., 2006.

[64] Macromedia Flash, Disponível em: <http://http://en.wikipedia.org/wiki/Macromedia_Flash> Acesso em: 04 abr. *Macromedia Flash*, 2006.

[65] Macromedia Graphics, Inc., Disponível em: <http://livedocs.macromedia.com/flash/8/main/Part7_Extending.html> Acesso em: 04 abr. *Flash 8 Documentation*, 2006.

[66] Macromedia Graphics, Inc., Disponível em: <http://livedocs.macromedia.com/flex/20beta1/docs/ 00002555.html> Acesso em: 04 abr. *Flash 8 Documentation*, 2006.

[67] Contero Manuel et al. Improving visualization skills in engineering education. *IEEE Computer Graphics and Applications*, páginas 24–31, out de 2005.

[68] Pauluk Marcel. Distinções analíticas e hipóteses classificatórias da tipologia dos signos da.

[69] Pauluk Marcel. Sistemas de escrita: Abordagens, tipologias, perspectivas em semiótica. Dissertação de Mestrado, Pontificia Universidade Católica de São Paulo - PUC-SP, 2003.

[70] Pistoia Marco, Reller Duane F., Gupta Deepak, Nagnur Milind, e Ramani Ashok. *Java 2 Network Security.* IBM International Technical Support Organization, 1999.

[71] Gelautz Margrit, Brandejski Michael, Kilzer Florian, e Amelung Falk. Web-based visualization and animation of geospatial data using x3d. IEEE Computer Society Press, editor, *Proceedings. 2004 IEEE International*, páginas 4773–4775, Baltic, 2004. Geoscience and Remote Sensing Symposium. n. 5.

[72] Mercury Computer Systems Inc., Disponível em: <http://www.tgs.com/support/oiv_doc/index.htm> Acesso em: 04 abr. *Open Inventor*, 2006.

[73] Microsoft Corporation, Disponível em: <http://www.microsoft.com/products/expression/en/interactive_designer/id_free_trial.aspx> Acesso em: 04 abr. *Microsoft Expression Interactive Designer*, 2006.

[74] Microsoft Corporation, Disponível em: <http://office.microsoft.com/pt-br/FX010858021046.aspx> Acesso em: 04 abr. *Microsoft FrontPage*, 2006.

[75] Microsoft Inc., Disponível em: <http://http://office.microsoft.com/en-us/assistance/HP010501811033.aspx?mode=print> Acesso em: 04 abr. *Save as Web Page browser and output format compatibility*, 2006.

[76] Mozzila Project, Disponível em: <http://http://www.mozilla.org/projects/svg/> Acesso em: 04 abr. *Mozilla SVG Project*, 2006.

[77] Open Source Scalable Vector Graphics Editor, Disponível em: <http://www.inkscape.org/> Acesso em: 04 abr. *Inkscape*, 2006.

[78] OSFlash, Disponível em: <http://www.osflash.org/open_source_flash_projects> Acesso em: 04 abr. *Open Source Flash Projects*, 2006.

[79] Steinmeyer Paul. *Development Platforms for Casual Games*. O'Reilly Media Inc., 2002.

[80] Liu Peiya. Scalable vector graphics. *IEEE MultiMedia*, páginas 99–102, set de 2003.

[81] Warrick Philip A. e Funnell W. Robert J. A vrml-based anatomical visualization tool for medical education. *IEEE Transactions on Information Technology in Biomedicine*, 2(2):55–63, jun de 1998.

[82] Loshin Philiph. *IPv6 Clearly Explained*. Morgan Kaufmann Publishers Inc., 1999.

[83] Fraternali Piero. Tools and approaches for developing data-intensive web applications: A survey. *ACM Computing Surveys*, 31(3):227–263, set de 1999.

[84] Wijkman Pierre A. I., Mitra Wijkman, e Dissanaike Suru. Transparent java standard extensions with native libraries on multiple platforms. *Proceedings...*, páginas 41–43, Kilkenny City, Ireland, 2003. Internation Conference on Principles and Practice of Programming in Java (PPPJ'03).

[85] João Queiroz. Sobre as 10 classes de signos de c.s. peirce. Dissertação de Mestrado, Pontificia Universidade Católica de São Paulo - PUC-SP, 1997.

[86] João Queiroz. *Modelos das relações sígnicas na semiose segundo C. S. Peirce: evidências empírico-teóricas.* Tese de Doutorado, Pontificia Universidade Católica de São Paulo - PUC-SP, 2002.

[87] Jungles dos Santos Rangel e Battaiola André Luiz. Análise de tecnologias para a implementação de jogos web. Sociedade Brasileira de Computação (SBC), editor, *Anais do Simpósio Brasileiro de Jogos para Computador e Entretenimento Digital*, páginas 137–147, São Paulo, nov de 2005. Universidade de São Paulo, Sociedade Brasileira de Computação (SBC).

[88] RealNetworks, Disponível em: <http://http://www.real.com/international/player/> Acesso em: 04 abr. *RealPlayer*, 2006.

[89] K. Yin Robert. *Estudo de Caso: Planejamento e Métodos.* Artmed Editora S.A., São Paulo, 2 ed. edition, 2001.

[90] Fielding Roy T. e Taylor Richard N. Principled design of the modern web architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, mai de 2002.

[91] Lau Rynson W. H. e Kunii Tosiyasu L. Emerging web graphics standards and techonologies. *IEEE Computer Graphics and Applications*, páginas 66–75, fev de 2003.

[92] Lau Rynson W. H. e Kunii Tosiyasu L. Web graphics. *IEEE Computer Graphics and Applications*, páginas 26–27, fev de 2003.

[93] Konno Satoshi. *CyberX3D Virtual Reality Development Package for C++*. Disponível em: <http://www.cybergarage.org/vrml/cx3d/cx3dcc/index.html> Acesso em: 04 abr., 2006.

[94] Hagen Silvia. *IPv6 Essentials*. O'Reilly Media Inc., 2002.

[95] Schar Sissel G. e Krueger Helmut. Using new learning techonologies with multimedia. *IEEE MultiMedia*, páginas 40–51, set de 2000.

[96] Sodipodi Development Group, Disponível em: <http://www.sodipodi.com/index.php3> Acesso em: 04 abr. *Sodipodi*, 2006.

[97] Ian Sommerville. *Engenharia de Software*. Pearson Addison Wesley, São Paulo, 6 ed. edition, 2003. Tradução André Maurício de Andrade Ribeiro.

[98] Bryson Steve. Virtual reality in scientific visualization. *Communications of the ACM*, 39(5):62–71, 1996.

[99] Probets Steve e et al. Vector graphics: From postscript and flash to svg. *Proceedings ACM...*, páginas 135–143, Atlanta, 2001. ACM Press.

[100] Suziah Sulaiman. Usability and the software production life cycle. *CH196*, páginas 13–18, jan de 1996.

[101] Sun Microsystems, Disponível em: <https://java3d.dev.java.net/> Acesso em: 04 abr. *Java3d Project home*, 2006.

[102] Sun Microsystems, Inc., Disponível em: <https://j3dfly.dev.java.net/> Acesso em: 04 abr. *Java 3D Fly Through*, 2006.

[103] Sun Microsystems, Inc., Disponível em: <http://java.sun.com/products/java-media/jmf/2.1.1/guide/index.html> Acesso em: 04 abr. *Java Media Framework API Guide*, 2006.

[104] Sun Microsystems, Inc., Disponível em: <http://www.netbeans.org/about/legal/spl.html> Acesso em: 04 abr. *Sun Public License Version 1.0 (SPL)*, 2006.

[105] SWiSHzone.com Pty Ltd, Disponível em: <http://www.swishzone.com> Acesso em: 04 abr. *The Ultimate in Flash Authoring*, 2006.

[106] Gill Tony, Caris Con, e Smith Guy LeBlanc. Interactive web-based visualisation of block model data. SIGGRAPH: ACM Special Interest Group on Computer Graphics e Interactive Techniques : The Web3D Consortium, editors, *Proceedings of the ninth international conference on 3D Web technology*, páginas 23–30, New York, NY, USA, 2004. ACM Press.

[107] Eco Umberto. *Tratado geral de semiótica*. Perspectiva, São Paulo, 1976 (1997).

[108] Web 3D Consortium, Disponível em: <http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification/Part01/Architecture.html > Acesso em: 04 abr. *Abstract Specification*, 2006.

[109] Web 3D Consortium, Disponível em: <http://www.web3d.org/applications/tools/utilities_and_translators/> Acesso em: 04 abr. *File Translators and Utilities*, 2006.

[110] Web 3D Consortium, Disponível em: <http://www.web3d.org/news/archives/> Acesso em: 04 abr. *X3D Headlines News Archieves*, 2006.

[111] Web 3D Consortium, Disponível em: <http://www.web3d.org/x3d/overview.html> Acesso em: 04 abr. *X3D Overview*, 2006.

[112] Nöth Winfried. Handbook of semiotics. Indiana Univ. Press, editor, *Kartosemiotik/Kartosemiotika: Internationales Korrespondenz-Seminar*, páginas 7–21, Bloomington, jun de 1990. Allgemeine Semiotik und Kartosemiotik, Indiana Univ. Press.

[113] Nöth Winfried. Panorama da semiótica: De platão a peirce. *Visualidade, urbanidade, intertextualidade*, (Annablume):119–133, (1995) 1998. A.C. de Oliveira & Y. Fechine, Hacker, São Paulo.

[114] Nöth Winfried. Cartossemiótica. A.C. de Oliveira & Y. Fechine, editor, *Visualidade, urbanidade, intertextualidade*, páginas 119–133. Hacker, 1998.

[115] World Wide Web Consortium (W3C). *W3C SMIL Document License*.

[116] World Wide Web Consortium (W3C), Disponível em: <http://http://www.w3.org/Consortium/Legal/2002/copyright-documents-20021231> Acesso em: 04 abr. *W3C SVG Document License*, 2006.

[117] Ferré Xavier, Juristo Natalia, Windl Helmut, e Constantine Larry. Usability basics for software developers. *IEEE Software*, páginas 22–31, jan de 2001.

[118] Xinox JCreator, Disponível em: <http://www.jcreator.com/> Acesso em: 04 abr. *Autodesk Maya*, 2006.

[119] XStream Software, Disponível em: <http://xstreamsvg.com> Acesso em: 04 abr. *Rapid SVG*, 2006.

Apêndice

APÊNDICE A

TECNOLOGIAS GRÁFICAS EMERGENTES

A.1 *Scalable Vector Graphics* (*SVG*) e *Synchronized Multimedia Integration Language* (*SMIL*)

A.1.1 Histórico

Resultado do esforço do *SVG Working Group*[1] do *W3C*[2], o *SVG* é um padrão que descreve gráficos vetoriais 2D, interativos e animados por meio de uma gramática derivada da *Extensible Markup Language (XML)* [49]. A versão 1.0 da sua especificação foi divulgada em setembro de 2001 e corresponde ao objetivo que motivou o seu desenvolvimento: ser uma solução orientada a documento genérica para elementos gráficos capaz de ser adaptada para mídia moderna [80]. A sua versão mais recente (versão 1.1) foi divulgada em janeiro de 2003 e apresenta poucas diferenças se comparada com a original.

Assim como o *SVG,* a *Synchronized Multimedia Integration Language (SMIL)* é uma recomendação do *W3C* para a criação de documentos para a Web. Sua primeira especificação foi publicada em 1998 e a sua versão mais recente (versão 2.1), cerca de 15 vezes maior do que a original, foi publicada em dezembro de 2005. A sua construção foi motivada pela demanda por uma linguagem de integração de multimídia genérica que permitisse a autoria de forma simples de apresentações audiovisuais interativas [30].

A.1.2 Estrutura interna e recursos

Assim como qualquer outro padrão de representação de gráficos, o *SVG* possui suporte ao desenho de todas as primitivas gráficas básicas (linhas, polígonos, retângulos, círculos

[1] *SVG Working Group* - Grupo de trabalho formado por membros da *World Wide Web Consortium* (*W3C*) e de representantes de empresas líderes de mercado, tais como a *Adobe*, *Corel*, *Apple*, *Macromedia*, *Microsoft*, *Sun*, *Hewlett-Packard*, *Canon*, *Kodak* e *ILOG*.

[2] O *World Wide Web Consortium (W3C)* é um consórcio internacional que desde 1994 desenvolve tecnologias inter operantes (padrões, guias, softwares e ferramentas) para a Web.

e elipses). Também permite a criação de caminhos por meio de linhas, curvas de Bezier e arcos elípticos, bem como o preenchimento de polígonos com cores uniformes e/ou gradientes e a aplicação de filtros. As transformações afins de rotação, translação e mudança de escala, assim como o enviesamento (*skew*), são passíveis de aplicação.[49].

A sua descrição completamente textual por meio de tags *XML,* o torna uma solução especialmente interessante, pois permite fácil integração dos seus documentos com diferentes tecnologias. *Cascading Style Sheets (CSS)*[3] e *Extensible Style Language (XSL)*[4], são bons exemplos de tecnologias para descrição de estilos que podem ser utilizadas com o *SVG*. O seu suporte ao *Document Object Model (DOM)*[5] permite que os documentos *SVG* sejam utilizados em conjunto com as mais variadas linguagens de programação de conteúdos para Web. Dentre elas, pode-se citar a *JavaScript / ECMAScript*, *Perl Hypertext Preprocessor (PHP)* e *Active Server Pages (ASP)* [91].

Em razão do esforço realizado pelo *W3C* pela sua adoção em novos dispositivos de hardware e também devido à sua definição com base no estabelecido pelo *International Color Consortium (ICC)*[6], o que busca viabilizar o uso do *SVG* em múltiplas plataformas sem alterações nas cores dos gráficos, uma versão mais leve do *SVG* foi criada para utilização em dispositivos com baixo poder de processamento e espaço de memória. O *SVG-Tiny* permite que gráficos *SVG* sejam gerados em *PDA's* e telefones celulares, entre outros dispositivos [25].

Segundo Probets, Mong, Evans e Brailsford (2001) [99], animações podem ser criadas em *SVG* por três maneiras basicamente:

- Usando o modelo declarativo de animação do próprio *SVG*
- Manipulando o *Document Object Model* (*DOM*)
- Usando a *Synchronized Multimedia Integration Language* (*SMIL*)

[3] *Cascading Style Sheets* (*CSS*) - Padrão desenvolvido e recomendado pelo *W3C* para promover um mecanismo simples para adição de estilos (fontes, cores, tabulações, etc.) a documentos Web.

[4] *Extensible Style Language* (*XSL*) - Linguagem derivada da *XML*, desenvolvida e recomendada pelo W3C para a definição de transformações em documentos *XML* e apresentações.

[5] *Document Object Model* (*DOM*) - Interface neutra de plataforma e linguagem que permite programas e scripts acessar dinamicamente e atualizar o conteúdo, estrutura e estilo dos documentos Web.

[6] *International Color Consortium* (*ICC*) - Consórcio internacional criado em 1993 por indústrias de hardware para a criação de um sistema universal para gerenciamento de cor.

Destas maneiras, a mais completa é a usando a *SMIL.* As demais (pelo modelo declarativo de animação do *SVG* e pela manipulação do *DOM*) são adequadas apenas para casos em que é desnecessária a sincronização de diferentes tipos de mídias, o que não ocorre na elaboração de conteúdos multimídia complexos. Por essa razão, recomenda-se a utilização da linguagem *SMIL* em conjunto com o *SVG* para a criação de conteúdos multimídia.

Usada para a criação apenas da mídia e/ou de apresentações multimídia integradas com fluxos de áudio, vídeo, imagens e texto, a *SMIL* possui uma gramática derivada da *Extensible Markup Language (XML).* Essa condição a torna adequada a operar em conjunto, a exemplo do *SVG*, com os padrões *CSS* e *XSL* [30] [41].

Em "*SMIL 2.0, XML for Web Multimedia*", Rutledge apontas os 5 pontos mais relevantes dessa linguagem:

1. Integração de mídias: *SMIL* é uma linguagem que permite somente a integração de mídias em uma única apresentação, o que exclui a criação das mídias. Para manipular mídias, a *SMIL* faz referências a arquivos de diversos formatos. O navegador então localiza os itens de mídia e a *SMIL* constrói os controles sobre eles. Além dos controles, a *SMIL* também possui formas de criar efeitos de transição, tais como, desvanecimento e surgimento para acompanhar o início e / ou o término da exibição das mídias.

2. Disposição: A segunda atribuição de grande importância da *SMIL* é a definição da disposição dos itens de mídia dentro da apresentação. Parâmetros como as dimensões e o posicionamento dos itens são bons exemplos de informações de disposição utilizadas na *SMIL.*

3. Sincronismo: Sem a utilização da *SMIL*, os itens de mídia definidos em *XML* para Web (por exemplo, o *SVG*) se mantêm estáticos. Ela permite que tais elementos mudem em função do tempo, o que adiciona dinamismo à apresentação. Além disso, devido à forma como ela manipula as modificações dos itens pelo tempo, a *SMIL* permite a sincronização de diferentes itens de mídias, associando, por exemplo,

textos como legenda de filmes, sons como explicações de animações, etc.

4. Ligação: Assim como a *HTML*, a *SMIL* possui formas de realizar ligações com outros documentos Web. Os *hyperlink*'s permitem que as apresentações *SMIL* façam referências para elementos em outros endereços Web ou mesmo em outras partes da própria apresentação. Com isso, as apresentações podem ser divididas em páginas, permitindo um desenvolvimento modularizado do conteúdo.

5. Adaptabilidade: Talvez esta seja a mais interessante das características da *SMIL*, pois ela permite que as apresentações sejam desenvolvidas considerando não apenas um, mas vários ambientes de execução. Como os documentos Web podem ser disponibilizados para pessoas em todo o mundo, é de se supor que um mesmo documento seja utilizado por pessoas que falam diferentes línguas e que tenham máquinas com as mais variadas configurações de hardware e capacidade de processamento. Para lidar com esses problemas, a *SMIL* possui elementos que permitem a especificação de itens de mídia diferenciados para usuários com as mais diversas línguas e máquinas. Um exemplo relevante para demonstrar essa possibilidade é o de que uma mesma apresentação *SMIL* pode possuir textos em línguas diferentes para pessoas de diferentes nacionalidades, assim como disposições de itens e mesmo mídias com graus de qualidade diversos para máquinas com maior ou menor poder de processamento.

A.1.3 Ferramentas de autoria

A criação de documentos *SVG* pode ser realizada por um vasto conjunto de aplicações. Dentre elas há de se citar as de código aberto *Sodipodi*[96] e *Inkscape*[77] e as comerciais *RapidSVG*[119] e *Adobe Illustrator* [2]. Probets, Mong, Evans e Brailsford (2001) [99] apresentam formas alternativas para a criação de documentos *SVG*. A confecção via programação em *Perl* com uso da biblioteca *SVG-PL* é uma delas. Sua principal vantagem é a possibilidade de criar gráficos dinâmicos de forma relativamente simples.

Outra forma possível para a criação de documentos *SVG* citada por eles é a conversão

dos formatos *PostScript (PS)*[7], *Portable Document Format (PDF)*[8] e *Macromedia Shockwave Flash* para o formato *SVG*. A conversão dos dois primeiros tipos de documentos é possível porque parte do esforço do projeto de desenvolvimento do *SVG* é dado para a confecção de um driver para o *Ghostscript*[9], o qual é capaz de manipular tanto documentos *PS* como documentos *PDF* [32]. A do terceiro pode ser realizada de forma limitada pela ferramenta *Flash2Svg* desenvolvida pelos próprios autores do artigo[31].

Em razão do seu formato textual, a criação e edição de documentos *SMIL* pode ser realizada por editores de texto convencionais. Contudo, para maior facilidade de manipulação, recomenda-se a utilização de ferramentas gráficas. Duas bastante interessantes são a de código aberto *Limsee2*[43] e a comercial *Adobe GoLive CS2*[1]. Além destas, um conjunto bastante vasto pode ser encontradas na página da *SMIL* no *W3C* (http://www.w3.org/AudioVideo/#Authoring).

A.1.4 Potenciais barreiras a sua utilização

Dois são os mais claros possíveis empecilhos a utilização do *SVG* e da *SMIL* para a criação de aplicações gráficas 2D interativas para Web. O primeiro deles se refere à dificuldade de elaboração das aplicações, enquanto que o segundo é relativo à sua distribuição.

A possível dificuldade na elaboração das aplicações se deve ao fato de que os principais softwares para criação de documentos *SVG* não possuem funcionalidades que permitam a criação de documentos *SMIL*. O mesmo acontecendo com os softwares para a criação dos documentos *SMIL* em relação ao *SVG*. Essa característica pode representar um empecilho ao se considerar que o desenvolvimento de aplicações é um processo cíclico em que as várias fases são continuamente revisadas pelos desenvolvedores. Em termos práticos a utilização de pelo menos duas ferramentas (uma para a criação de documentos *SVG* e outra para a criação de documentos *SMIL*) é indesejável por causar uma intensa mudança de contexto

[7] *PostScript (PS)* - Linguagem interpretada para a descrição de gráficos, desenvolvido pela Adobe Systems. Usa notação pós-fixa para descrição de gráficos paginados bidimensionais.

[8] *Portable Document Format* (*PDF*) - Padrão de descrição de páginas desenvolvido pela *Adobe* e que possui um modelo de imagens semelhante ao *PostScript* nível 2.

[9] *Ghostscript* - Interpretador *Postscript* (*PS*) disponível sob a licença *GNU Public Licence* para vários sistemas operacionais.

de desenvolvimento.

No que diz respeito à distribuição, uma barreira existente é a falta de suporte interno nos navegadores a essas tecnologias. O *Microsoft Internet Explorer (MSIE)* por exemplo, navegador mais popular do mercado segundo dados de sites tradicionais especializados em estatísticas sobre a Internet tais como TheCounter.com Global Statistics (www.thecounter.com/stats/), OneStat.com (www.onestat.com/), AdTech (adtech.info/) e Net Applications (www.netapplications.com/), não possui suporte interno (*built-in*) a aplicações nos formatos *SVG* e *SMIL* [75].

Uma solução possível para esse problema é a instalação de *plugin*'s. Porém, tanto o *Adobe SVG Viewer* [3], quanto o *RealPlayer* [88] (*plugin*'s *SVG* e *SMIL* para *MSIE* mais populares) não possuem mecanismos automáticos de detecção e instalação. Tal situação obriga os usuários a baixá-los e instalá-los antes da utilização de aplicações nessas tecnologias, o que é indesejável do ponto de vista da Interface Humano-Computador por não ser amigável ao usuário (*user-friendly*).

Quando considerado o *Mozilla Firefox* porém, esse problema é menos grave. Posto que a partir da sua versão 1.5 o suporte interno (*built-in*) é provido para ambas as tecnologias [76]

A.1.5 Custo de utilização

Como aventado no breve histórico, tanto o *SVG* quanto a *SMIL* são padrões abertos desenvolvidos pelo *W3C*. As suas licenças de utilização permitem cópia, utilização e alteração, tanto para fins particulares quanto para comerciais, dos documentos de sua especificação, softwares e demais conteúdos sem qualquer ônus desde que sejam preservados os créditos aos seus autores e desenvolvedores [116] [115]. Desta forma e considerando-se a existência de ferramentas de código aberto para o desenvolvimento de documentos *SVG* e *SMIL*, pode-se dizer que estas tecnologias não possuem custo de utilização no que se refere a licenças de software e ferramentas.

A.2 *Macromedia Shockwave Flash* (*Flash*)

A.2.1 Histórico

Em dezembro de 1996 a *Macromedia* adquiriu um software de animação baseado em vetores chamado *FutureSplash* e o transformou na versão 1.0 do *Flash*. A versão 2.0 foi lançada em 1997 com funcionalidades tais como suporte a som stereo e integração com bitmap realçada.

Inicialmente o *plugin Flash Player* não era distribuído com navegadores Web populares e os usuários precisavam visitar o site da *Macromedia* para baixá-lo. Entretanto a partir de 2000, ele começou a ser distribuído com todos os navegadores mais populares (*AOL, Netscape e Internet Explorer*) que, por questões de mercado, começaram a prover suporte interno (*built-in*) ao *player*. Dois anos mais tarde, passou a ser também distribuído juntamente com todas as atualizações do *Windows XP*.

Em setembro de 2001, um exame feito pela *Media Metrix* mostrou que dos 10 maiores sites dos Estados Unidos, 7 faziam uso de conteúdos *Flash*. Em 15 de março de 2002, a *Macromedia* anunciou a disponibilidade do *Macromedia Flash MX* e *Macromedia Flash Player 6*, com suporte para vídeo, componentes de aplicação e acessibilidade.

O *Flash MX 2004* foi atualizado em setembro de 2003, com funcionalidades tais como: desempenho em tempo de execução 8 vezes mais rápido e o novo *Macromedia Flash Player 7*, hábil para criar cartas, grafos, e adicionais efeitos de texto com novo suporte para extensões (vendidas separadamente), alta fidelidade para importação de *PDF* e arquivos *Adobe Illustrator 1.0*, desenvolvimento para dispositivos móveis e ambientes de desenvolvimento baseado em formulários [53] [64].

Em dezembro de 2005, a *Adobe* adquiriu a *Macromedia* e o seu portfólio de produtos, incluindo o *Flash*. Atualmente, seu desenvolvimento está caminhando no sentido de prover soluções de apresentação de conteúdo, campo atualmente dominado por tecnologias como *JSP* e *ASP*.

A.2.2 Estrutura interna e recursos

Macromedia Flash é um ambiente de desenvolvimento integrado (*integrated development enviroment, IDE*) e *Flash Player* é uma máquina virtual usada para executar arquivos *Flash*. Na linguagem coloquial, entretanto, há uma mistura de conceitos. *Flash* pode ser tanto o ambiente de autoria, o *player* ou o arquivo da aplicação. Os arquivos *Flash*, normalmente chamados de filmes *Flash*, usualmente tem a extensão *.swf* e podem aparecer como elementos de páginas Web ou ser executados separadamente no *Flash Player*.

Os filmes *Flash* são constituídos por elementos de mídia, tais como, imagens, sons, vídeos, objetos gráficos, texto, por scripts que definem o seu comportamento e por um tipo especial de símbolo, o clipe de filme (*movie clip*).

Clipes de filmes são mini-filmes *Flash*, o que equivale a dizer que são constituídos, como o filme principal, de elementos de mídia, scripts e por clipes de filme. Assim como elementos de mídia convencionais, eles possuem propriedades, tais como coordenadas, brilho, ângulo de rotação, altura e largura, que podem ser alteradas por meio dos *scripts* e pela interface da *IDE Flash*. Analogamente, os filmes *Flash* equivalem às aplicações em si e os clipes de filme a sub-programas especializados em tarefas específicas do sistema [8].

A interface da *IDE Flash* representa metaforicamente o ambiente de produção de um filme. Com isso, o conceito de cena é amplamente aplicado. Os atores atuantes, nesse contexto, são representados pelas mídias manipuladas. Enquanto que a locação, o camarim, a cenografia, a marcação de cena e os scripts que orientam a atuação dos atores são representados por painéis.

O painel que representa a locação onde se passa o filme é denominado dentro da aplicação de "cena". Neste painel é possível visualizar o conteúdo que está sendo desenvolvido e seus controles permitem iniciar e interromper a execução do filme, entre outras coisas. O camarim é representado pelo painel denominado "biblioteca", onde é apresentado o elenco de atores do filme. Eventualmente, em caso de cenas muito complexas, a divisão dos atores em vários elencos é recomendada. Com isso, é possível ter mais de uma biblioteca para uma mesma cena.

A estrutura cenográfica é composta por painéis de criação e edição de conteúdos

gráficos. Eles contêm as ferramentas básicas de desenho, como traçado de retas, desenho de curvas e de polígonos, ferramentas de preenchimento e paletas para manipulação de cores. Ferramentas para a produção de efeitos visuais mais sofisticados como degrades e reflexo também estão disponíveis.

A marcação das cenas do filme é representada por um painel denominado "linha do tempo". Nele a entrada e a saída dos personagens da cena é representada por barras horizontais. Como um mesmo ator pode representar vários personagens em uma mesma cena, é comum ter diferentes barras para demonstrar a sua entrada e saída. Essa situação em que um mesmo ator incorpora vários personagens numa mesma cena é denominado dentro do *Flash* de "instanciação"[**?**].

As ações que cada personagem deve executar são determinadas, assim como nos filmes convencionais, por *scripts*. "Ações"é o nome dado ao painel onde são editados os *scripts* de cada personagem. As linguagens utilizadas para a criação de *scripts*, a partir da versão 8 da *IDE*, são a *JavaScript* e a *ActionScript*. *Javascript / ECMAScript* é uma linguagem de programação de aplicações para a Web executada no lado do cliente (navegador Web). Seu histórico, estrutura e demais características serão vistas em mais detalhes nas seções seguintes [65].

Considerada um dialeto do *Javascript / ECMAScript*, a *ActionScript* é uma linguagem orientada a objetos, com sintaxe semelhante a do *Java* e suporte a programação dirigida por eventos e ao *DOM*. Com isso, a tarefa de elaborar *scripts* para determinar o comportamento dos elementos de um filme pode ser definida como a construção de uma hierarquia de objetos que trocam mensagens entre si em função, principalmente, de eventos ocorridos.

Dentre os eventos padrão tratados pela *ActionScript* estão os externos e internos. Exemplos de eventos externos são: acionamento / pressionamento e liberação de teclas do teclado e acionamento / deslocamento e pressionamento de botões do mouse. Exemplos dos internos: aqueles que ocorrem em função de colisões entre elementos, entrada em novo quadro (*frame*) do filme, etc. O suporte ao *DOM* permite a sua comunicação com outras linguagens e padrões *Javascript / ECMAScript*, *JSP*, *ASP*, *CSS* e *XSL* são bons exemplos de linguagens e padrões que podem se comunicar com *ActionScript* via *DOM* [66].

A.2.3 Ferramentas de autoria

Além da *IDE* padrão desenvolvida pela *Macromedia*, existe um conjunto considerável de ferramentas (muitas delas livres e de código aberto) que manipulam e / ou produzem aplicações *Flash*. Dentre elas, pode se citar as livres desenvolvidas no projeto *OSFlash* [78] e as comerciais *Swish Max* [105] e *Microsoft Expression Interactive Designer* [73].

A.2.4 Potenciais barreiras a sua utilização

Mesmo considerando-se os esforços para a criação de ferramentas livres e de código aberto, o fato do *Flash* ser uma tecnologia fechada é sem dúvida o principal empecilho para a sua utilização na implementação de aplicações gráficas baseadas na Web. Como citado no histórico, recentemente as definições, softwares e direitos sobre o *Flash* foram vendidos juntamente com a *Macromedia* para a *Adobe*. Tal fato pode representar um problema, tendo em vista as mudanças de foco que podem ser dadas à tecnologia em função dos interesses da sua nova empresa detentora.

A.2.5 Custo de utilização

Se considerada a existência de ferramentas livres e de código aberto, pode-se dizer que o custo de produção de aplicações *Flash* é desprezível. Contudo, se tomado em consideração que há uma *IDE* padrão e que ela é a ferramenta mais adequada ao desenvolvimento de aplicações nessa tecnologia, o custo passa a ser um fator preponderante. O preço de uma cópia profissional completa e licenciada da *IDE* era de cerca de 699 reais em 04 de abril de 2006, segundo o site da empresa no Brasil [4].

A.3 *Extensible 3D* (*X3D*) com *XJ3D*

A.3.1 Histórico

Fruto do esforço do *Web 3D Consortium*, grupo de trabalho do *W3C*, responsável pela elaboração de padrões abertos para a comunicação 3D em tempo real, o *X3D* teve sua

primeira versão lançada em 2000. Em 2002, parte da sua especificação foi aceita pelo *Moving Picture Experts Group (MPEG)* como a base de gráficos interativos 3D para o padrão multimídia *MPEG-4*. Ainda nesse ano, foi criado o *Java Rendering Working Group*, responsável pela criação das atribuições da *API* gráfica para manipulação do *X3D* usando *Java*.

A *XJ3D* (*toolkit X3D* de código aberto baseado em *Java*) teve a sua primeira versão com capacidade para manipular todas as funcionalidades *X3D* (versão M8) lançada em 2004. Ano este em que também foram lançadas versões iniciais do *Vizx3d* e do *Flux*, aplicações pioneiras na visualização de conteúdos *X3D*. Em 2005, exportadores foram lançados para as ferramentas *Blender*, *3D Studio Max* e *Maya*, além da *H3D* (*API* C++ para desenvolvimento de aplicações gráficas com *X3D*).

Atualmente, está em curso o desenvolvimento de editores e demais ferramentas para a composição de mundos virtuais e elaboração de aplicações baseadas nessa tecnologia [110].

A.3.2 Estrutura interna e recursos

X3D é um padrão aberto de formato de arquivo baseado em *XML* criado para a comunicação de dados 3D tanto para aplicações convencionais, quanto para aplicações em rede. Possui um conjunto rico de funcionalidades para uso em engenharia, visualização científica, desenho auxiliado por computador (*CAD*), arquitetura, visualização de dados médicos, treinamento e simulação, multimídia, entretenimento e educação, entre outros [111].

Construído a partir da *VRML*, o *X3D* apresenta um conjunto de novas funcionalidades, tais como interface avançada de programação de aplicações, formatos de codificação de dados adicionais, conformação mais estrita e arquitetura dividida em componentes, o que permite o suporte modular ao padrão. Além destas, possui também os recursos básicos da *VRML* [108]:

- Gráficos 3D - geometrias poligonais e paramétricas, transformações hierárquicas, iluminação, materiais e mapeamento de texturas;

- Gráficos 2D - texto, formas vetoriais e planares 2D apresentadas com a hierarquia de transformações 3D;
- Animação - marcadores de tempo e interpoladores para dirigir animações contínuas, animação de humanóides e *morphing*;
- Áudio e vídeo distribuídos no espaço - fontes audiovisuais mapeadas na geometria da cena;
- Interação com o usuário - seleção e arrastamento baseados no mouse e na entrada do teclado;
- Navegação - câmeras, movimentação do usuário na cena 3D, colisão, proximidade e detecção de visibilidade;
- Objetos definidos pelo usuário - habilidade de estender a funcionalidade do navegador interno pela criação de tipos de dados definidos pelo usuário;
- Escrita de *scripts* - habilidade de mudar dinamicamente a cena via programação e linguagens de *script*;
- Operação em rede - habilidade de compor uma única cena *X3D* com recursos localizados na rede, ligando objetos de outras cenas ou recursos localizados na Web;
- Simulação física - animação de humanóide, conjuntos de dados geoespaciais, integração com protocolos de simulação interativa distribuída.

O *X3D* conta também com uma arquitetura que permite, ao contrário da *VRML*, a utilização de subconjuntos da especificação. Os denominados perfis (*profiles*) são compostos de blocos modulares de funcionalidades denominados componentes. São quatro os perfis básicos existentes em *X3D*:

- Intercâmbio (*Interchange*) - é o perfil básico para comunicação entre aplicações. Suporta geometrias, texturas, iluminação básica e animações;

- Interativo (*Interactive*) - habilita a interação básica com o ambiente 3D pela adição de vários nodos sensores para a navegação e a interação com o usuário (por exemplo, *PlanseSensor*, *TouchSensor*, etc.), sincronismo realçado e iluminação adicional;

- Imersivo (*Immersive*) - habilita todas as funcionalidades gráficas 3D e interação, incluindo suporte a áudio, colisão, fumaça e escrita de *scripts*;

- Completo (*Full*) - inclui todos os nodos definidos, até mesmo *NURB*'s, animação de humanóides e componentes geoespaciais.

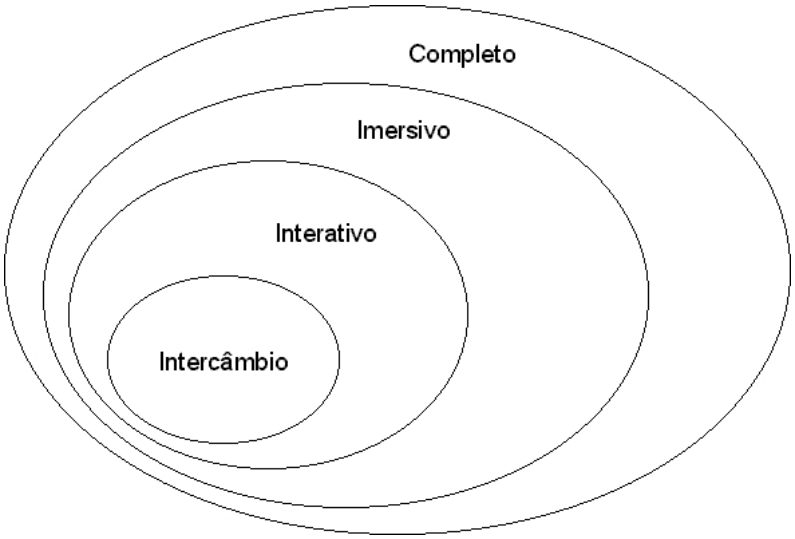

Figura A.1: Hierarquia de perfis *X3D*.

A organização dos conteúdos tridimensionais acontece no *X3D* através de uma estrutura de dados básica. O grafo de cena *X3D* é uma estrutura que contém todos os objetos do ambiente tridimensional e seus relacionamentos. Os quais são descritos através das arestas que ligam os vértices que representam os elementos e transformações.

A sintaxe derivada da *XML* permite a integração dos conteúdos com outros padrões Web. Web services e *SVG* são bons exemplos de tecnologias que podem operar perfeitamente integradas com o *X3D*.

A.3.3 Ferramentas de autoria

A composição de modelos animados em *X3D* pode ser feita a partir da ferramenta livre *X3D-Edit*[33], disponível para as plataformas *Windows*, *MacOS* e *Linux*. Contudo, por se tratar de uma ferramenta puramente textual, o *X3D-Edit* não é adequado para a criação de conteúdos complexos. Para isso, recomenda-se a utilização de ferramentas como as comerciais *3D Studio Max* e *Maya* e a livre *Blender*. Um conjunto de exportadores gratuitos para elas tem sido desenvolvido pela *Web 3D Consortium* e já está disponível para download no site oficial do *X3D* [109].

As alternativas são bem variadas para a inserção de comportamentos via interface de programação e criação de aplicações. Códigos *Javascript / ECMAScript* podem ser facilmente criados usando-se a comercial *Mercury Open Inventor*[72]. Para a formulação de aplicações *X3D* em C++ destaca-se a excelente *API* livre *CyberX3D for C++*[93], a qual também possui uma versão disponível na linguagem *Java*. A *CyberX3D for Java* é, ao lado da também livre *XJ3D*, uma ótima alternativa para composição de aplicações nessa linguagem, contudo, a segunda, por se tratar da *toolkit* oficial de teste e desenvolvimento do padrão *X3D*, é a mais recomendada.

A.3.4 Potenciais barreiras a sua utilização

Apesar da sua natural vocação para a elaboração de aplicações baseadas na Web, o *X3D* não é suportado internamente por nenhum dos navegadores Web populares. A instalação de *plugin*'s é, nesse caso, uma alternativa. Porém, por se tratar de um processo pouco amigável ao usuário, essa possibilidade não é considerada. Isto, em especial, porque a utilização de *API*'s, tais como, a *XJ3D* e a *CyberX3D* para *Java*, permitem a criação de aplicações *X3D* em *Applets Java*, semelhantes as que podem ser feitas com base na extensão *Java 3D*.

A.3.5 Custo de utilização

O *X3D* é um padrão aberto que não tem *royalties* associados a ele. O Consórcio *Web3D* tem a política de não requerer nenhuma restrição de propriedade intelectual para a tecnologia e há um acordo com a *ISO*[10] para se publicar a especificação para o público sem nenhum custo [63]. Além disso, considerando-se a existência de *API*'s para a programação e as ferramentas de edição gratuitas, pode se concluir que não há custo para a utilização dessa tecnologia para a criação de aplicações.

A.4 *Java 3D* (*J3D*)

A.4.1 Histórico

Fruto dos esforços conjuntos do Grupo de Tecnologias para Jogos da *Sun* (*Sun's Game Technologies Group*) e de empresas líderes do mercado de hardware gráfico (*Intel, Silicon Graphics* e *Apple*), a *Java 3D* teve sua primeira versão completa publicada em dezembro de 1998.

A versão mais recente (versão 1.4) foi publicada no final de fevereiro desse ano (2006)[101]. A *Java 3D API* é uma interface para a escrita de programas para *desktop*. Estes programas são baseados na Web e permitam a apresentação e a interação com gráficos tridimensionais[29]. Para tanto, ela provê mecanismos para a finalização (*rendering*) de gráficos e sons tridimensionais, a execução comportamental e a extensão do modelo de visualização.

Por questão de desempenho, ela utiliza funções de *API*'s de baixo-nível, tais como, a *OpenGL* e a *DirectX*, para a manipulação dos dados gráficos. Contudo, sua hierarquia de classes é de alto-nível, escondendo completamente as funções de baixo-nível do programador[34].

[10] *ISO* (*International Organization for Standardization)*- Organização não-governamental internacional responsável pela elaboração de padrões para as mais variadas áreas.

A.4.2 Estrutura interna e recursos

Para facilitar a composição e o controle de cenas, a *Java 3D* é constituída de uma estrutura de dados, denominada grafo de cena, que é capaz de englobar desde dados geométricos a comportamentos dos modelos. O grafo de cena é, tecnicamente falando, um grafo acíclico direcionado, no qual uma estrutura hierárquica de relacionamentos entre os objetos da cena é representada[52][58].

Ele é composto geralmente por um nodo "*VirtualUniverse*", o qual define um universo. Como filho, esse nodo deve possuir ao menos um nodo do tipo "*Locale*", o qual define um sistema de coordenadas onde estará situado o subgrafo que contém os elementos da cena. Este é composto tipicamente por dois "*BranchGroup*'s", nodos especiais utilizados para agrupar nodos com características semelhantes. Um destes nodos agrupa os elementos em si, ou seja, os modelos geométricos, as aparências, os comportamentos, as localizações, os sons e as luzes, pertencentes à cena. Outro modo determina os parâmetros de visualização da cena, tais como, posicionamento, volume de visualização, orientação, etc[29].

Os modelos geométricos animados podem ser inseridos na cena através de duas formas básicas: pela instanciação de primitivas existentes na *Java 3D* (cubo, elipse, esfera, etc) ou pela carga de modelos elaborados por ferramentas de modelagem. Como o método de instanciação é inviável para a elaboração de modelos complexos, a carga de modelos é a via mais recomendável para a criação de cenas. Por essa razão, a *Java 3D API* possui carregadores internos (*built-in*) para modelos nos formatos *VRML* e *Object*, além de uma classe abstrata que permite a implementação de carregadores para os demais formatos de modelos geométricos[24].

Atualmente, há carregadores eficientes implementados usando essa classe abstrata para modelos animados em *VRML* e *X3D*. Uma lista completa de carregadores de modelos geométricos para *Java 3D* é mantida permanentemente no endereço (http://java3d.j3d.org/utilities/loaders.html).

Além desses recursos, a *Java 3D*, por ser uma extensão do padrão da linguagem *Java*, conta com todos os recursos dessa linguagem para a construção de estruturas de dados complexas, comunicação em rede, formulação de aplicações complexas, interfaces, etc. A

independência de plataforma e as formas automatizadas para a instalação e a atualização dos componentes de software necessários[84] para a execução permitem que suas aplicações sejam utilizadas sem maiores dificuldades por usuários de diferentes sistemas operacionais e navegadores.

Outra característica importante derivada do relacionamento da *Java 3D* com os demais conteúdos *Java* é a possibilidade de utilização de dispositivos de entrada não usuais, como *joysticks*. Em "*The Java 3D API and Virtual Reality*"(1999), Rosemblun e Macedonia[56] salientam o fato de aplicações *Java 3D* poderem possuir um número indeterminado de dispositivos. Eles ressaltam também a necessidade de tais dispositivos respeitarem ao denominado acesso em 6 graus de liberdade (*six-degrees-of-freedom*), o qual é suportado pelas 9 operações de acesso genérico existentes na *Jinput* (*Java Input Controller API*), extensão *Java* destinada ao controle de dispositivos de entrada não usuais.

A.4.3 Ferramentas de autoria

A produção de modelos animados para utilização em cenas Java 3D pode ser realizada por um conjunto vasto de softwares. Basicamente, qualquer software de modelagem 3D capaz de exportar conteúdos para arquivos *VRML* e/ou *X3D* válidos pode ser utilizado para a criação de modelos para cenas *Java 3D*.

Contudo, dentre as aplicações de modelagem mais recomendadas estão as comerciais *AutoDesk 3D Studio Max*[11] e *Maya*[10] e a de código aberto *Blender*[14]. Para a escrita de código *Java* recomenda-se a utilização de ferramentas como a gratuita *Eclipe*[36] e a comercial *Xinox JCreator* [118]. Porém, tanto uma quanto a outra não possuem recursos específicos para a criação de conteúdo gráfico tridimensional.

Uma alternativa para a visualização e a edição de cenas em *Java 3D* são as ferramentas de código aberto *Java 3D Fly Through* e a *Java 3D Editor*[102]. Elas permitem a criação e a manipulação de cenas de forma mais fácil e simples do que pela programação direta.

A.4.4 Potenciais barreiras a sua utilização

Em princípio, a maior barreira existente para a não utilização da *Java 3D* é a dificuldade para a importação de modelos animados. Apesar da existência de carregadores eficientes para essa tarefa, nem sempre as ferramentas de modelagem geram arquivos *VRML* e *X3D* válidos. Com isso, não são raras as situações em que animações e elementos geométricos são importados inadequadamente ou mesmo não são importados pelas aplicações *Java 3D*. Esse problema, porém, não é causado pela tecnologia *Java 3D* em si, mas pelas ferramentas utilizadas para a criação de modelos.

A.4.5 Custo de utilização

Se considerada a criação de cenários e modelos animados usando o *Blender*, software de modelagem 3D mantido sob a licença *GNU General Public License (GPL)*[40], a edição de código em *Eclipse*, software regido pela *Common Public License (CPL)*[42], a visualização e a edição de cenas pelo *Java 3D Fly Throgh* e *Java 3D Editor*, respectivamente, ambos softwares regidos pela *Sun Public License (SPL)*[104], pode-se dizer que não há custo de utilização. Isto porque todos os softwares necessários, incluindo a própria *Java 3D API*, possuem licenças que não prevêem a cobrança de *royalties* pela utilização e comercialização de softwares elaborações a partir do seu uso.

A.5 *Java Media Framework* (*JMF*)

A.5.1 Histórico

Fruto do esforço conjunto da *Sun Microsystems Inc.*, *Silicon Graphics Inc.* e *Intel Corporation*, o *Java Media Framework (JMF)* foi desenvolvida com foco na execução de fluxos de mídia gravada, ou seja, o *JMF* foi desenvolvida para prover suporte a sincronização, controle, processamento e apresentação de fluxos de mídia. A versão 1.0 dessa extensão *Java* foi publicada em 1998. Atualmente, na versão 2.1.1, a *JMF API* é uma solução viável para a manipulação de diferentes mídias em aplicações *Java* convencionais e *Ap-*

plets Java[59].

A.5.2 Estrutura interna e recursos

Fundamentalmente, a *JMF* é uma extensão da *Java* para a manipulação de áudio e vídeo. Mais formalmente, a *Java Media Framework Application Programming Interface (JMF API)* é uma *API* opcional para a extensão das funcionalidades do núcleo da linguagem *Java*. Suas principais funcionalidades são[6]:

- Independência de plataforma;
- Manipulação integrada de áudio e vídeo como objetos de mídia;
- Suporte para um significativo número de tipos de conteúdo de áudio e vídeo e codec's[11];
- Execução de mídia;
- Gravação de mídia para arquivo;
- Captura de mídia a partir de dispositivos como câmeras e microfones;
- Recebimento de fluxos de mídia transmitidos através da Internet;
- Transmissão de fluxos de mídia através da Internet;
- Multiplexação / Demultiplexação (combinação ou divisão) de mídias;
- Transcodificação (alteração para um formato diferente) de mídias;
- Unificação em um quadro de trabalho (*framework*) do suporte a todas as operações em mídias (por exemplo, efeitos) como processamento;
- Extensibilidade para suportar outros formatos e *plugin*'s[12];
- Integração natural com a *API Java* existente.

[11] Codec - acrônimo de Codificador/Decodificador, dispositivo de hardware ou software que codifica/decodifica sinais.

[12] *Plugin* - Extensão de software que adiciona novas funcionalidades a uma dada aplicação.

Dirigida a eventos, a *JMF* também possui manipulação de exceções e suporte a sub-programas (*thread*'s). Dentre os formatos de áudio manipulados pela *JMF* estão:

- ***AIFF* (.aiff)**;
- *AVI* (.avi);
- *GSM* (.gsm);
- *HotMedia* (.mvr);
- *MIDI* (.mid): Tipo 1 e 2;
- *MPEG-1* Video (.mpg);
- *MPEG Layer II* Audio (.mp2);
- *QuickTime* (.mov);
- *Sun Audio* (.au);
- *Wave* (.wav).

O seu modelo de gravação, processamento e apresentação de mídias é semelhante ao usado nos equipamentos convencionais, ou seja, a manipulação de mídias com *JMF* possui, a exemplo da manipulação convencional, fases bem distintas. A captura de vídeo para uma fita feita a partir de uma câmera de vídeo, a execução do vídeo pelo vídeo-cassete e a apresentação do vídeo nos dispositivos de saída (televisor e alto-falantes) são representadas na *JMF* pelas fases de obtenção da mídia a partir de uma fonte de dados (captura a partir de um dispositivo de entrada, leitura de arquivo ou mesmo recebimento a partir de um fluxo de mídia a partir da Internet), processamento (mixagem, edição ou mesmo simples execução da mídia) e apresentação (envio da mídia processada para os dispositivos de saída adequados)[103].

A.5.3 Ferramentas de autoria

Por se tratar, assim como o *Java 3D*, de uma extensão do padrão *Java*, a *Java Media Framework API* conta com ferramentas *freeware* e comerciais para a geração de código. Em especial, se destacam a gratuita *Eclipse*[36] e a comercial *Xinox JCreator*[118], porém, ambas não possuem recursos específicos para pré-visualização de aplicações construídas com *JMF*.

A.5.4 Potenciais barreiras a sua utilização

JMF possui suporte a manipulação de um vasto conjunto de mídias, porém a sua comunicação com conteúdos *SVG* e *SMIL* é bem complexa. *JMF* não implementa comunicação com o *DOM*, modelo usado por esses padrões para comunicação com outras tecnologias.

No que diz respeito a comunicação com conteúdos *Flash*, a situação é menos problemática. *JMF* permite manipulação de conteúdos *Macromedia Flash* e a sua inclusão em aplicações *Java*, dentre elas *Applets*, no entanto, tal manipulação possui severas restrições. A maior delas se refere à versão dos conteúdos *Flash* manipulados. O sistema *Macromedia Flash* está, presentemente, em sua oitava versão, mas somente a segunda versão do formato *SWF* é suportado pela *JMF*.

Esse quadro torna problemática a adoção da *JMF* como tecnologia de integração entre conteúdos 2D e 3D interativos, baseados na Web.

A.5.5 Custo de utilização

A exemplo da *J3D*, a *JMF* não possui custo de utilização. Licenciada sob a *Sun Public License (SPL)*[104] que exime os desenvolvedores de pagamento de qualquer *royalty* pelo uso, publicação e comercialização de aplicações, a *JMF* pode ser usada sem qualquer custo adicional com ferramentas de edição livres, tais como a *Eclipse*[36].

A.6 *Javascript / ECMAScript*

A.6.1 Histórico

Em dezembro de 1985, a *Netscape* e a *Sun* publicaram as especificações de uma nova linguagem idealizada por Brendan Eich, denominada de *JavaScript*. Com elementos de sintaxe semelhantes aos do *C* e de *C++*, essa linguagem em pouco tempo se tornou bastante popular entre os desenvolvedores de conteúdo para a Web. Anos mais tarde, devido ao grande sucesso da linguagem produzida pelas concorrentes, a *Microsoft* publicou as especificações da *JScript*. Linguagem que, a exemplo da *JavaScript*, possuía sintaxe semelhante ao *C* e ao *C++* e rodava no lado do cliente (navegador Web)[26].

A elaboração de um padrão conjunto, baseado nas duas linguagens, teve início em novembro de 1996. A primeira edição *ECMA* foi adotada pelo *ECMA General Assembly* em junho de 1997. O padrão *ECMA* foi submetido ao *ISO/IEC JTC 1* e aprovado como padrão internacional *ISO/IEC 16262* em abril de 1998.

Da sua primeira para a segunda edição ocorreram apenas mudanças editoriais. Atualmente, o padrão está em sua terceira edição e ele inclui: expressões regulares, melhor manipulação de palavras (*string*'s), novas estruturas de controle, manipulação de exceções pelas diretivas *try / catch*, definição firme de erros e formatação para a saída numérica, entre outras coisas.

Contudo, o trabalho sobre a linguagem ainda não está completo. O comitê técnico está trabalhando para promover melhorias significativas, tais como melhorias na inter operação com padrões do *W3C*[37].

A.6.2 Estrutura interna e recursos

ECMAScript é uma linguagem baseada em objetos (*objects*), em outras palavras, todas as facilidades das máquinas em que será executada são providas por objetos. Desta forma, um programa *ECMAScript* é um conjunto de objetos que se comunicam. Formalmente, um objeto *ECMAScript* é uma coleção não ordenada de propriedades (*properties*) que podem conter zero ou mais atributos (*attributes*) que determinam como cada propriedade

pode ser usada. Propriedades são *container*'s que mantém outros objetos, valores primitivos (*primitive values*) ou métodos (*methods*)[37].

A linguagem define 5 tipos internos (*Undefined, Null, Boolean, Number* e *String*) e uma coleção de objetos internos (*built-in objects*), os quais incluem objetos para a representação de funções (*function object*), vetores (*array object*), palavras (*string object*), números (*number object*), datas (*date object*), etc. Também define os operadores e estruturas de controle e iteração clássicas e possui uma sintaxe semelhante a do *Java*[27].

A hierarquia de objetos da *ECMAScript* constitui uma representação dos elementos comuns em páginas Web. Quadros, botões e campos de formulário são alguns desses elementos representados nessa hierarquia. Cada navegador Web que suporta a execução de códigos *ECMAScript* implementa esse conjunto de elementos, ao qual é dado o nome de *Document Object Model (DOM)*. Graças a esse modelo, páginas *HTML* com código *ECMAScript* podem ser visualizadas nos mais diferentes navegadores.

Outro benefício advindo da utilização do *DOM* é a natural comunicação com padrões baseados nesse modelo. *CSS*, *Dinamic HTML (DHTML)*, *XLS*, *SVG*, a linguagem *Java* (nas *Applets Java*) e também os conteúdos *Flash* são bons exemplos de padrões que podem ser utilizados em conjunto com a *ECMAScript*[13][26].

A.6.3 Ferramentas de autoria

Por se tratar de código inserido em arquivos *HTML*, os programas *ECMAScript* podem ser editados por um grande número de ferramentas. Editores de texto convencionais podem ser utilizados para essa tarefa. Além deles, ferramentas de autoria de páginas *HTML*, que na maioria das vezes possuem recursos para a visualização e a edição de *scripts*, podem ser utilizados para a escrita de código nessa linguagem. Um exemplo simples de ferramenta de edição com essa característica é o *Microsoft FrontPage*[74].

A visualização das páginas criadas com *ECMAScript* pode ser feita diretamente nesse tipo de ferramenta ou alternativamente em navegadores com suporte a *ECMAScript*.

[13]Outros tipos de documentos (imagens, sons, vídeos, etc.) também são manipulados por essa linguagem.

A.6.4 Potenciais barreiras a sua utilização

O fato da *ECMAScript* ser uma linguagem executada pelo cliente da comunicação Web (navegador) pode ser apontado como uma fator em potencial de desencorajamento de uso. Se comparada a linguagens executadas pelo servidor (servidor Web), tais como, *PHP*, *ASP* e *JSP*, essa linguagem é mais suscetível a manipulação indevida por parte de usuários mal intencionados, o que torna os sistemas desenvolvidos nessa linguagem mais vulneráveis a ataques.

A.6.5 Custo de utilização

Por se tratar de uma linguagem de padrão aberto regida pela *SPL*[104] e por existir ferramentas de autoria gratuitas, tais como, editores de texto convencionais, pode-se dizer que não há custo na utilização dessa linguagem para o desenvolvimento de aplicações.