UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Projeto de um Ambiente 3D de Visualização e Reprodução de Eventos Capturados e Interpretados a Partir de Ambientes Físicos Cientes de Contexto para Aplicações de Preparação para Emergência

Altieres Ribeiro Lopes

São Carlos - SP

Maio – 2006

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

L864pa

Lopes, Altieres Ribeiro.
Projeto de um ambiente 3D de visualização e reprodução de eventos capturados e interpretados a partir de ambientes físicos cientes de contexto para aplicações de preparação para emergência / Altieres Ribeiro Lopes. -- São Carlos : UFSCar, 2006.
95 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2006.

1. Ambientes virtuais colaborativos. 2. Ambientes 3D. 3. Ambientes cientes de contexto. 4. Preparação para emergência. 5. Gravação. I. Título.

CDD: 006 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Projeto de um Ambiente 3D de Visualização e Reprodução de Eventos Capturados e Interpretados a Partir de Ambientes Físicos Cientes de Contexto para Aplicações de Preparação para Emergência"

ALTIERES RIBEIRO LOPES

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

Membros da Banca:

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

Prof. Dr. Rudinei Goularte
(ICMC/USP)

Prof. Dr. João Batista Camargo Junior
(Poli/USP)

São Carlos
Maio/2006

“Porque o Senhor dá a sabedoria, da sua boca vem a inteligência e o entendimento.”
Provérbios 2.6

Agradecimentos

Agradeço a Deus que tem me sustentado a cada dia, que me deu a oportunidade de realizar este trabalho e que constantemente me ensina.

À maravilhosa família que Deus me deu. Ao meu pai que, embora já ausente, me ensinou a ter dignidade. À minha mãe, sempre confiante em mim, que tem me surpreendido demonstrando uma força que sobeja minhas expectativas. À minha avó, muito sentimental, que se emocionou a cada partida.

À minha namorada que, embora longe, esteve sempre presente me dando força e confiando na minha capacidade.

Ao meu primo e colega de quarto. Além da companhia nunca inconveniente também proporcionou grandes gargalhadas.

À grande família em Cristo, principalmente aos membros da IPI de Campestre e da IPB Filadélfia de São Carlos. Às famílias do pastor Sebastião Silvestre e do presbítero João Casolli, que muitas vezes acolheram a mim e ao meu primo como se fossemos seus filhos.

A todos os professores que contribuíram para minha formação, principalmente à profª Drª Regina Borges de Araujo, que muito me incentivou e orientou nos caminhos da pesquisa.

Ao prof Ms Leonardo Andrade e ao aluno Francisco Gaspar do depto de Artes da UFSCar, que prontamente contribuíram para o desenvolvimento desse trabalho.

A todos os integrantes do *LRVNet* que conheci, Taciana, Gislaine, Richard, Diego, Fernando, Néstor, Gilson, Leandro e Ricardo. Também aos alunos do GAPIS e às funcionárias Cristina e Mirian, sempre dispostos a ajudar.

Resumo

Sistemas de apoio à preparação para emergências, em especial o monitoramento preciso de ambientes físicos sujeitos a situações de emergência, são recursos importantes para empresas e órgãos públicos de defesa civil, pois podem ajudar a evitar e/ou reduzir riscos de perda de vidas e de patrimônio. A maioria dos sistemas de monitoramento de ambientes físicos descritos na literatura apresenta limitações, tais como: não permitem visualização posterior da situação de emergência; são limitados a determinados tipos de aplicação; não permitem a identificação precisa das situações de risco; etc. Neste trabalho foi proposto e avaliado um sistema que visa superar essas limitações, através do uso integrado de redes de atuadores e de sensores sem fio, computação ciente de contexto e de realidade virtual. O trabalho consistiu da criação, implementação e avaliação de um sistema de gravação e reprodução de mídia 3D em que ambientes físicos sujeitos a situações de emergência são munidos de sensores com potencial computacional e de comunicação. Esses sensores capturam e interpretam contextos que são mapeados em uma linguagem visual, que é refletida em um ambiente de realidade virtual que simula o ambiente físico. O uso de realidade virtual para fins de visualização e acesso em tempo- real[1] ou posteriormente de um ambiente que apresenta características de segurança crítica visa superar as limitações de interfaces do tipo hipermídia, ou mídia contínua, como vídeo, quando as experiências do mundo real são muito complexas. No sistema criado como resultado deste trabalho, usuários podem reproduzir as experiências capturadas do mundo real para análise, avaliação, monitoramento e treinamento. As inovações deste trabalho residem em dois aspectos: utiliza uma técnica otimizada de gravação que economiza tempo de processamento e espaço de armazenamento; e os comandos de atualização de cena são independentes de navegador, permitindo a visualização do ambiente virtual colaborativo (AVC) através de diferentes navegadores 3D para Web. Com a colaboração do Departamento de Artes e Comunicação (DAC) da FSCar, foi criada também uma linguagem visual para a pronta identificação de situações emergenciais e uma interface para melhor navegação do usuário em sistemas complexos. Exemplos de uso incluem o monitoramento de diferentes plantas industriais, aeronaves em situações de ensaios de vôo e solo, plataformas de exploração de petróleo, etc. Este trabalho é parte de um projeto colaborativo entre o Laboratório de Realidade Virtual em Rede (LRVNet) do Departamento de Computação da UFSCar e o PARADISE Lab do SITE da University of Ottawa.

[1] Toda a complexidade dos sistemas de tempo-real não será abordada nesta dissertação, sendo que o termo tempo-real será utilizado indicando apenas que os eventos devem ser apresentados ao cliente o mais rápido possível.

Abstract

Systems for emergency preparedness support, especially those for accurate monitoring of physical environments subjected to emergency situations, are valuable resources for companies and civil defense public institutions, since these systems can help avoiding and/or reducing lives and patrimony losses. Most of the existing monitoring systems described in the literature have limitations, such as: no posterior visualization of emergency situations that have occurred; limited to specific types of applications; inaccurate identification of risk situations; etc. In this work, a system was proposed and evaluated that aims to overcome these limitations through the integrated use of wireless actor and sensor networks, context aware computing and virtual reality. The work consisted on the creation, implementation and evaluation of a recording and playing 3D media in which physical environments subjected to emergency situations are deployed sensors with processing and communication resources. These sensors capture and interpret contexts, which are mapped, through a visual language, on a 3D virtual environment that mimics the physical environment. The use of virtual reality for visualization and access in real-time[2] or afterwards of situations that are occurring in the physical environment, through a 3D Virtual Environment, can overcome the limitations of hypermedia interfaces or continuous media, like video, when the experiences of the real world are very complex. This work describes the project of a recording and playing system, which allows users to play live experiences gathered from the real world for analysis, evaluation, monitoring and training. The novelty of the system resides in two aspects: it uses an optimized recording technique that saves processing time and storage space; it records scene updating commands independent from 3D Players, allowing the visualization of the collaborative virtual environment (CVE) through any existing 3D web players. In collaboration with the Arts and Communication Department (DAC) of UFSCar, a visual language to prompt identification of emergency situations was created as well as an interface to complex systems. Examples of use include the monitoring of industrial plants, flight rehearsals, petrol exploration platforms, etc. This work is part of a collaborative Project between the Networked Virtual Reality Lab (LRVNet) of the Computer Science Department at UFSCar and PARADISE Lab of SITE at University of Ottawa.

[2] All complexity of real-time systems don't will be considered in this dissertation, real-time will be used indicating that the events must be shown faster as possible.

Lista de Figuras

FIGURA 1 - MONITORAMENTO PRECISO DE UMA AERONAVE 2
FIGURA 2 - SISTEMA *FIRE* 6
FIGURA 3 - VISÃO GERAL DO PROJETO DE PREPARAÇÃO PARA EMERGÊNCIAS DO *LRVNET* 9
FIGURA 4 - VISÃO GERAL DO PROJETO 29
FIGURA 5 - INTERAÇÃO ENTRE OS MÓDULOS DO PROJETO 32
FIGURA 6 - A) CENA 3D, B) GRAFO QUE REPRESENTA ESTA CENA E C) DESCRIÇÃO DESTA CENA EM X3D 34
FIGURA 7 - INTERFACE DE MONITORAMENTO 36
FIGURA 8 - ESCALA DE CORES - TEMPERATURA 37
FIGURA 9 - SÍMBOLOS - LÍQUIDOS E GASES 38
FIGURA 10 - SÍMBOLOS - FALHA NO CONDUTOR 39
FIGURA 11 - SÍMBOLO - CAMPO MAGNÉTICO 39
FIGURA 12 - ESCALA DE CORES - PRESSÃO 40
FIGURA 13 - SÍMBOLOS - PRESSÃO 40
FIGURA 14 - CÓDIGO PARA INSERÇÃO DE NÓ 41
FIGURA 15 - CÓDIGO PARA REMOÇÃO DE NÓ 41
FIGURA 16 - CÓDIGO PARA ALTERAÇÃO DE PARÂMETRO 41
FIGURA 17 - CÓDIGO PARA SUBSTITUIÇÃO DE NÓ 42
FIGURA 18 - LINHA DO TEMPO: *EVENT-REPLAY* 44
FIGURA 19 - LINHA DO TEMPO: GRAVAÇÃO DE CENAS COMPLETAS 45
FIGURA 20 - LINHA DO TEMPO: GRAVAÇÃO DE CENAS PARCIAIS 46
FIGURA 21 - LINHA DO TEMPO: GRAVAÇÃO UTILIZANDO O MÉTODO OTIMIZADO 48
FIGURA 22 - MODELO ER DO BANCO DE DADOS PARA ARMAZENAMENTO DAS GRAVAÇÕES 49
FIGURA 23 - INTERFACE DO NAVEGADOR CLIENTE 51
FIGURA 24 - INTERFACE AVANÇADA DO NAVEGADOR 51
FIGURA 25 - PALETA PARA MAPEAMENTO DE TEMPERATURA EM COR 54
FIGURA 26 - NÚMERO DE GRAVAÇÕES (EM 1 HORA) 59
FIGURA 27 - EVENTOS POR GRAVAÇÃO 60
FIGURA 28 - LATÊNCIA DE ACORDO COM O TIPO DE COMANDO 62
FIGURA 29 - LATÊNCIA AO ACESSAR ALEATORIAMENTE CENAS GRAVADAS 63
FIGURA 30 – ECONOMIA DE ESPAÇO DE ARMAZENAMENTO COM O USO DE CENAS COMPLETAS 66
FIGURA 31 - CASO DE USO – ACESSAR O SISTEMA 75
FIGURA 32 - CASO DE USO – ESPECIFICAR A LINGUAGEM VISUAL 75
FIGURA 33 - DIAGRAMA DE ATIVIDADES 76
FIGURA 34 - DIAGRAMA DE CLASSES 77
FIGURA 35 - PSEUDOCÓDIGO - *TTRADUTOR.TRADUZIR(CNT)* 77
FIGURA 36 - PSEUDOCÓDIGO - *TGRAVADOR.COMANDORECEBIDO(COM)* 78
FIGURA 37 - PSEUDOCÓDIGO - *TGRAVADOR.GRAVARCC()* 78
FIGURA 38 - PSEUDOCÓDIGO - *TGRAVADOR.GRAVARCP()* 78
FIGURA 39 - PSEUDOCÓDIGO – *TREPRODUTOR.INICIAR(Z)* 79
FIGURA 40 - PSEUDOCÓDIGO - *TREPRODUTOR.BUSCARANTERIORES()* 79
FIGURA 41 - PSEUDOCÓDIGO – *TREPRODUTOR.RUN()* 79
FIGURA 42 - PSEUDOCÓDIGO - *TSERVIDOR.CLIENTECONECTOU()* 79
FIGURA 43 - DIAGRAMA DE SEQÜÊNCIA 80

Lista de Tabelas

TABELA 1 - COMPARAÇÃO ENTRE OS SISTEMAS DE MONITORAMENTO 7
TABELA 2 - COMANDO DA LV PARA SENSORES DO TIPO TEMPERATURA, DLV 53
TABELA 3 - EXEMPLOS DE LEITURAS REALIZADAS PELOS SENSORES 53
TABELA 4 - COMANDOS APÓS TRADUÇÃO 54
TABELA 5 - COMPUTADORES UTILIZADOS PARA REALIZAÇÃO DOS EXPERIMENTOS 60
TABELA 6 - MODELOS 3D E SUAS CARACTERÍSTICAS 61
TABELA 7 - CARACTERÍSTICAS DOS MODELOS 3D DO AMBIENTE DE MONITORAMENTO DO AVIÃO 61
TABELA 8 - COMPLEXIDADE DOS OBJETOS X LATÊNCIA DE EXECUÇÃO 62

Sumário

1. INTRODUÇÃO 1

1.1. OBJETIVOS E JUSTIFICATIVA 3
1.2. ORGANIZAÇÃO DA DISSERTAÇÃO 3

2. SISTEMAS DE MONITORAMENTO DE AMBIENTES FÍSICOS SUJEITOS A SITUAÇÕES DE EMERGÊNCIA 5

2.1. O SISTEMA SIREN 5
2.2. POSTO DE COMANDO DO FUTURO 5
2.3. O SISTEMA FIRE 6
2.4. LIMITAÇÕES DOS SISTEMAS APRESENTADOS 6
2.5. PROJETO DE UM SISTEMA DE PREPARAÇÃO PARA EMERGÊNCIAS DO LRVNET 8
2.6. CONSIDERAÇÕES FINAIS 10

3. AMBIENTES VIRTUAIS 3D (AV3D) E TECNOLOGIAS DE VISUALIZAÇÃO 3D PARA A *WEB* .. 11

3.1. TECNOLOGIAS DE SUPORTE A AMBIENTES VIRTUAIS 3D NA WEB 12
3.1.1. VRML 12
3.1.2. JAVA-3D 13
3.1.3. MPEG-4 14
3.1.4. X3D 16
3.2. O XJ3D 18
3.3. CONSIDERAÇÕES FINAIS 19

4. GRAVAÇÃO E REPRODUÇÃO DE MÍDIA 3D 22

4.1. RPBIMS 22
4.2. RRCVRS 23
4.3. DIVE 23
4.4. MASSIVE-3 24
4.5. WHEREWEREWE 25
4.6. ANALOGIA À COMPACTAÇÃO DO MPEG. 25
4.7. CONSIDERAÇÕES FINAIS 26

5. LINGUAGEM VISUAL E PROJETO DE VISUALIZAÇÃO 29

5.1. DESCRIÇÃO DAS CENAS 33
5.2. LINGUAGEM VISUAL 35
5.2.1. Interface Gráfica para o Monitoramento de AV3D 35
5.3. TRADUÇÃO PARA A LINGUAGEM VISUAL 40
5.4. CONSIDERAÇÕES FINAIS 42

6. IMPLEMENTAÇÃO E AVALIAÇÃO DO MECANISMO DE GRAVAÇÃO E REPRODUÇÃO DE AV3D 43

6.1. GRAVAÇÃO DE MÍDIA 3D 43
6.1.1. Técnica Event-Replay 43
6.1.2. Gravação de Cenas Completas 45
6.1.3. Gravação de Cenas Parciais 46
6.1.4. Gravação Utilizando o Método Otimizado 47
6.2. ARMAZENAMENTO 48
6.3. IMPLEMENTAÇÃO 49
6.3.1. Lado Cliente 50
6.3.2. Lado Servidor 52
6.4. EXPERIMENTOS REALIZADOS E RESULTADOS OBTIDOS 57
6.4.1. Cenas Completas e Cenas Parciais 57
6.4.2. Ambiente de Experimentação do Sistema de Gravação 60
6.4.3. Avaliação do Sistema de Gravação 60
6.5. CONSIDERAÇÕES FINAIS 66

7. CONCLUSÕES 67

7.1. CONTRIBUIÇÕES GERADAS 67
Artigos Publicados *67*
7.2. TRABALHOS FUTUROS 68
7.3. CONCLUSÕES 68
GLOSSÁRIO 70
ANEXO A – SEMIÓTICA 72
ANEXO B – MODELAGEM UML DO SISTEMA 75
CASOS DE USO 75
DIAGRAMA DE ATIVIDADES 76
DIAGRAMA DE CLASSES 76
DIAGRAMA DE SEQÜÊNCIA 80
REFERÊNCIAS BIBLIOGRÁFICAS 82

1. Introdução

Aplicações para preparação para emergências ganharam especial atenção após os acontecimentos de 11 de setembro de 2001 nos Estados Unidos. Nesta data, os ataques terroristas ao Pentágono e ao *World Trade Center* fizeram com que governos de todo o mundo investissem quantidades consideráveis de recursos na criação de planos de ação durante situações emergenciais. Além das situações emergenciais que ocorrem como resultado da ação dos seres humanos, como ataques terroristas, diversas outras situações de emergência podem ocorrer, seja devido a ações da natureza, tais como inundações, tempestades, terremotos etc., seja devido a falhas ou combinação de diferentes tipos de falhas (mecânicas, elétricas, de resistência de materiais etc). A preparação para emergências requer um processo de planejamento, treinamento e exercício acompanhado da aquisição de equipamentos e de instrumentos para suporte às ações de emergência. Para isso, o uso de dados históricos, coletados de incidentes passados é de extrema importância. Além do uso de dados históricos, dados sintéticos (gerados por computador) podem ser utilizados com o mesmo fim. A visualização em tempo-real de incidentes em progresso, e a posteriori de dados coletados ou sintéticos, são dois grandes desafios na criação de um sistema de monitoramento de ambientes físicos sujeitos a situações de emergência. Um outro desafio, é o monitoramento preciso dos ambientes sob situações de risco, já que a maioria dos sistemas de monitoramento de ambientes físicos atualmente disponíveis não oferece a precisão necessária.

De modo a superar os desafios mencionados acima, um projeto de preparação para emergência para monitoramento preciso de ambientes físicos sujeitos a situações de risco que integra realidade virtual, redes de sensores e computação ciente de contexto, está sendo desenvolvido como parte de um projeto colaborativo entre o Laboratório de Realidade Virtual em Rede (*LRVNet*) do Departamento de Computação da UFSCar e o PARADISE Lab do SITE da *University of Ottawa*. Um exemplo de monitoramento preciso de uma aeronave é mostrado na Figura 1. Eventos são capturados do ambiente físico da aeronave e interpretados, através de mecanismos que vão desde lógica simples até *Redes Bayesianas*. A interpretação dos contextos pode indicar a uma situação de emergência que é mapeada, através da linguagem visual, em um modelo 3D que imita o ambiente físico. Na figura, a alta temperatura na turbina esquerda do avião é mostrada através da cor vermelha. Os contextos interpretados são gravados como comandos que modificam o grafo de cena do ambiente virtual, sempre que novos eventos ocorrem no ambiente físico. Este ambiente virtual pode ser

visualizado em tempo-real ou posteriormente, podendo ser utilizado qualquer navegador 3D para visualização do ambiente, uma vez que os comandos gravados podem ser convertidos para diferentes formatos.

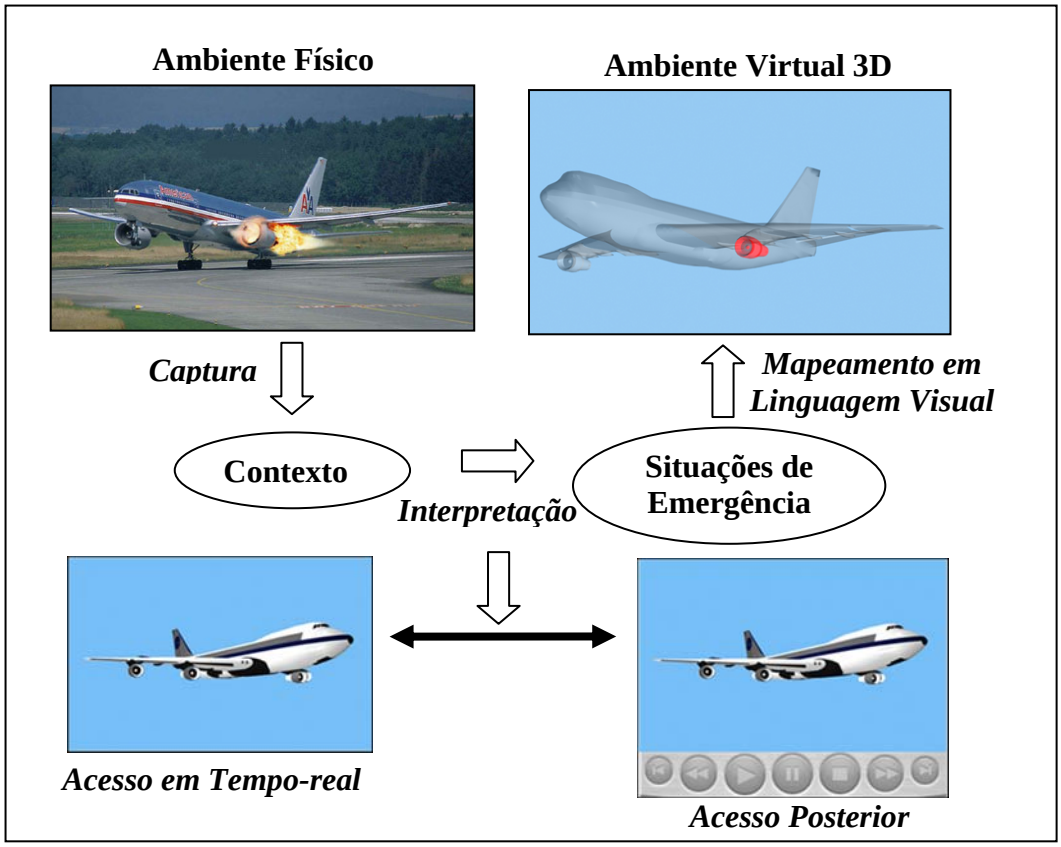


Figura 1 - Monitoramento preciso de uma aeronave

Se o acesso ao ambiente virtual é em tempo-real, ou seja, as informações apresentadas indicam a situação atual do ambiente naquele momento, eventuais situações de risco podem ser detectadas antes mesmo de se tornarem acidentes (prevenção). Já em uma situação de emergência, o acesso em tempo-real pode ser importante no auxílio à tomada de decisões por parte das equipes de salvamento.

Para que os eventos permaneçam disponíveis para acesso a qualquer momento, posterior à ocorrência dos mesmos, foi desenvolvido o sistema para visualização a posteriori. Este sistema pode ajudar no treinamento de equipes de resgate, bem como auxiliar equipes de segurança e de seguradoras durante a realização de perícias.

No monitoramento de situações críticas alguns requisitos não funcionais, como confiabilidade, eficiência e integridade das informações, deverão ser considerados, uma vez que a sua não observância poderá levar a situações de risco para seres vivos, patrimônio ou meio ambiente [MIC 04].

1.1. Objetivos e Justificativa

O principal objetivo deste trabalho é apresentar, desenvolver e avaliar uma solução de preparação de emergência para visualização, gravação e reprodução de ambientes virtuais tridimensionais (3D) que reflete situações ocorridas em ambientes físicos cientes de contexto. Esta solução inclui: um sistema de visualização em tempo-real que é utilizado como forma de apoio às equipes de prevenção; uma linguagem visual criada para representar, de forma intuitiva e clara, os acontecimentos do ambiente físico (é com base nesta linguagem que os eventos notificados pela rede de sensores são traduzidos em alterações na cena tridimensional do usuário); um sistema de gravação/reprodução dos eventos que ocorreram no ambiente físico; e uma interface que facilita a rápida identificação de situações de emergência. Como o sistema de gravação e reprodução permite a visualização pós-ocorrência, este pode ser uma ferramenta útil também no treinamento de diferentes equipes (apoio, prevenção, resgate, etc.).

Este projeto visa superar as limitações de interfaces do tipo hipermídia, ou mídia contínua, como vídeo, quando as experiências do mundo real são muito complexas. Mais ainda, os ambientes virtuais para *Web* não possuem uma forma padrão de gravação de suas sessões e as soluções existentes não suprem as necessidades do problema aqui apresentado. Entre as principais limitações dessas soluções é que algumas são proprietárias e outras não permitem, ou acessam de forma lenta, posições aleatórias da gravação. O acesso aleatório é de extrema importância, uma vez que o monitoramento deverá ser contínuo e, portanto, as gravações terão durações, teoricamente, infinitas.

Este trabalho apresenta o projeto e implementação e avaliação de protótipo de uma solução para monitoramento de condições de emergência em ambientes físicos cientes de contexto como contribuição para a classe aplicações de preparação para emergências.

1.2. Organização da Dissertação

Este trabalho está organizado da seguinte forma: o capítulo 2 faz uma revisão bibliográfica a respeito dos sistemas de monitoramento de ambientes físicos sujeitos a situações de emergência existentes. O capítulo 3 trata dos ambientes virtuais 3D e suas respectivas tecnologias de suporte para a *Web*. O capítulo 4 descreve alguns sistemas de gravação e reprodução de mídia 3D existentes. Uma visão geral do projeto de visualização e da linguagem visual para aplicações de preparação para emergência é dada no capítulo 5, sendo que a implementação e avaliação do mecanismo de gravação e reprodução de

Ambientes Virtuais 3D (AV3D) é descrita no capítulo 6. No capítulo 7 são apresentadas as conclusões do trabalho, seguidas de anexos e referências bibliográficas.

2. Sistemas de Monitoramento de Ambientes Físicos Sujeitos a Situações de Emergência

Existem alguns sistemas de monitoramento disponíveis atualmente, porém, nenhum que fizesse uso de uma interface tridimensional foi encontrado, tampouco sistemas com interfaces gráficas que levassem em consideração a exploração do potencial de imagens como forma rápida de identificação de situações de emergência. Esses sistemas objetivam monitorar condições do ambiente físico e auxiliar em decisões que necessitem ser tomadas rapidamente, durante eventuais incidentes, entretanto apresentam várias limitações que serão discutidas a seguir. São exemplos desses tipos de sistemas o *Siren* [CHE 04], o Posto de Comando do Futuro (*The Command Post of the Future*) [DAR 04] e o FIRE (*Fire Information and Rescue Equipment*) [WRI 04].

2.1. O Sistema Siren

O *Siren* é um sistema para auxiliar exclusivamente bombeiros em operações de combate a incêndio. Faz uso de uma interface bidimensional que roda em PDAs levados pelos bombeiros. Esta característica possui certa desvantagem se forem imaginadas situações em que a sala se encontra cheia de fumaça, o que irá dificultar ou mesmo impossibilitar a visualização por parte do bombeiro. Na solução aqui proposta, apenas uma equipe de apoio, exterior ao ambiente que se encontra em perigo faz uso do sistema. Informações podem ser trocadas com a equipe que está atuando no incidente através de rádio, por exemplo.

Esse sistema se preocupa apenas com os acontecimentos em tempo-real, não tratando da persistência dos dados. Também se pode dizer que ele trabalha com um monitoramento de menor precisão, uma vez que indica apenas a sala que se encontra em perigo, não permitindo uma análise mais precisa do problema. Trata de apenas três contextos: localização relativa (“Sala tal”), temperatura do ambiente e nível de oxigênio. As mensagens de alerta são cinco: “lugar perigoso”, “perigo para si mesmo”, “próximo a um lugar perigoso”, “outros em perigo” e “instruções” [CHE 04].

2.2. Posto de Comando do Futuro

O Posto de Comando do Futuro (*The Command Post of the Future*) se concentra na criação de táticas e estratégias para auxiliar nas tomadas de decisões através de técnicas de

inteligência artificial, principalmente em tempo de execução. É constituído de vários projetos e permite a criação de *displays* interativos que ajudam nas tomadas de decisões. Facilita a interação de várias formas, a visualização de informações do ambiente e a formação de estratégias baseada em conhecimento [DAR 04]. Este sistema é constituído de vários projetos e seu uso é restrito à área militar, uma vez que sua complexidade e funcionalidade não se aplicam a ambientes de monitoramento predial ou industrial, por exemplo.

2.3. O Sistema Fire

O FIRE (*Fire Information and Rescue Equipament*) é um projeto do departamento de engenharia mecânica da *Berkeley*, na Universidade da Califórnia, em conjunto com o corpo de bombeiros de Chicago. Destina-se ao auxílio no combate a situações de incêndio, fornecendo informações importantes para operações de resgate. O gerenciamento é feito por um comandante através de uma interface bidimensional. Um visor chamado *FireEye*, mostrado na Figura 2, fica acoplado à mascara do bombeiro, facilitando a visualização de informações provenientes da rede de sensores.

Figura 2 - Sistema *Fire*

2.4. Limitações dos Sistemas Apresentados

Na Tabela 1 são relacionados os sistemas apresentados e suas principais características, inclusive o sistema proposto pelo LRVNet. Dos sistemas apresentados, apenas o posto de comando do futuro permite que seu monitoramento seja gravado. Este é um importante ponto negativo desses sistemas, dadas as vantagens apresentadas anteriormente do

uso da reprodução posterior dos eventos ocorridos, seja para perícias, seja para o treinamento de equipes. O posto de comando do futuro, embora permita gravação, é muito complexo e voltado à área militar, o que dificulta sua implantação em outros ambientes. Além disso, os sistemas *Siren* e *Fire* são específicos para auxílio ao corpo de bombeiros, o que também limita a utilização deles em outros ambientes. Para superar as limitações apresentadas dos outros sistemas, este trabalho propõe uma solução para monitoramento de ambientes físicos sujeitos a situações de risco através de uma interface 3D. Esta interface 3D proposta também permite a identificação de situações de emergência de forma rápida e precisa. O monitoramento pode ser feito por objeto ou mesmo por parte de um mesmo objeto, bastando para isso que hajam sensores nos locais desejados e o ambiente virtual seja modelado com isso em vista. Assim, apenas o objeto ou a parte do objeto sendo monitorado por sensor refletirá as alterações relacionadas aos eventos capturados por aquele sensor. Além desta flexibilidade no monitoramento do ambiente, este sistema também dispõe de módulos para gravação e reprodução dos eventos. Este sistema possibilita a visualização de um ponto aleatório da gravação, bem como pausa, avanço quadro-a-quadro ou avanço rápido, como nos reprodutores de mídia contínua comuns.

Tabela 1 - Comparação entre os Sistemas de Monitoramento

Sistema	Interface	Precisão	Persistência	Usuário	Aplicação	Complexidade
Siren	2D	Menor	Não	Bombeiro (PDA)	Bombeiros	Baixa
Fire	2D	Menor	Não	Bombeiro (Máscara)	Bombeiros	Baixa
P. C. F.	2D	Maior	Sim	Equipe	Militar	Alta
LRVNet	3D	Maior	Sim	Equipe de Apoio	Qualquer	Baixa

A próxima seção apresenta a descrição do projeto de um sistema de preparação para emergências que está sendo desenvolvido no Laboratório de Realidade Virtual em Rede (*LRVNet*) do Departamento de Computação da UFSCar. Esse projeto objetiva superar os limites dos ambientes discutidos acima, propondo um sistema que trata da captura e interpretação de eventos em ambientes cientes de contexto com maior precisão, através de redes de atuadores e de sensores sem fio. Este trabalho de dissertação está inserido nesse projeto que será descrito a seguir.

2.5. Projeto de um Sistema de Preparação para Emergências do LRVNet

O presente trabalho é parte de um projeto que está sendo desenvolvido pelo Laboratório de Realidade Virtual em Rede (*LRVNet*) do departamento de computação da Universidade Federal de São Carlos em conjunto com o PARADISE Lab da *University of Ottawa*. Este projeto integra as áreas de realidade virtual, rede de sensores e computação ubíqua [ARA 03] e tem como objetivo oferecer soluções para que ambientes sujeitos a condições de emergência possam ser monitorados de forma precisa, ajudando na prevenção, tomada de decisões e treinamento de equipes, bem como avaliações posteriores. Diversos trabalhos foram e outros estão sendo desenvolvidos no *LRVNet*, tratando das diferentes necessidades que as redes de sensores sem fio apresentam, como por exemplo, o roteamento eficaz das mensagens, a economia de energia, a ordenação de eventos e a interpretação de contextos [KUD 04] [MIC 04] [PAZ 04] [SIL 05]. Este projeto, ilustrado na Figura 3, está organizado como segue: Algoritmos para redes de atuadores e de sensores (RASSF) que atendam às necessidades da classe de aplicações de monitoramento, isto é, protocolos tolerantes a falhas, cientes de consumo de energia, baixa latência, seguros, com qualidade de serviço e etc.; Mecanismos de interpretação de contextos (internamente às redes RASSF e também externamente em "*sinks*", dispositivos de maior capacidade de processamento, armazenamento e comunicação, quando a interpretação é mais complexa) e linguagem visual que permite identificar de maneira rápida e precisa situações de emergência; e Estruturas para visualização e acesso em tempo real e posterior que permitem a identificação precisa de situações de risco, através de ambientes virtuais 3D que imitam o ambiente físico [SIL 05].

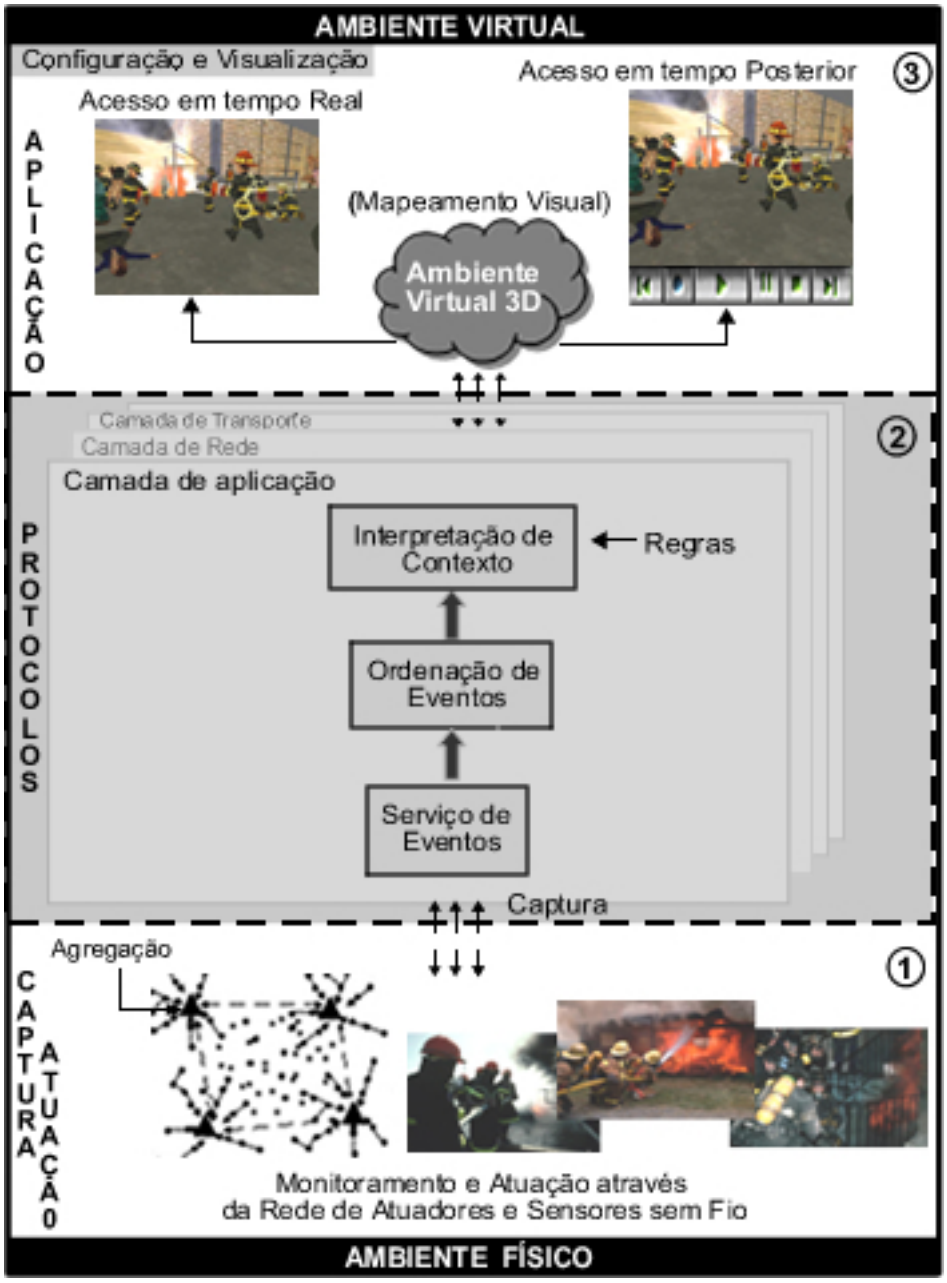


Figura 3 - Visão geral do projeto de Preparação para Emergências do *LRVNet*

A proposta de utilizar redes de sensores sem fio, conforme mostrado na Figura 3 (1), visa, entre outros fatores, facilitar a implantação em lugares de difícil acesso. Estas redes podem ser de diferentes topologias e também podem utilizar números variáveis de sensores, o que torna necessária a utilização de protocolos que tratem estas variações em busca de uma melhor utilização da rede de sensores como um todo [PAZ 04] [SIL 05]. Entre os protocolos desenvolvidos, conforme mostrado na Figura 3 (2), estão os de agregação, ordenação e interpretação de eventos. Os eventos coletados pela rede de sensores sem fio são interpretados e mapeados em linguagem visual, o que facilita a identificação de situações de emergência. A interpretação pode ser simples e ser usada para tratar ambigüidades e repetição de informação

ou tão complexa quanto se deseja. Exemplo de interpretação simples é o cálculo da média de diferentes valores de temperatura, que resulta em um único valor mapeado na cor vermelha no objeto alvo de monitoramento. Interpretações de contextos mais complexos podem exigir o uso de técnicas de inteligência artificial na correlação de eventos (na própria rede de sensores ou através de um *middleware* [KUD 04] [SIL 05]) como, por exemplo, na identificação das causas que levaram a asa de uma aeronave a rachar. Este trabalho está inserido no contexto mostrado na Figura 3 (3).

2.6. Considerações Finais

Sistemas de monitoramento com câmeras de vídeo e sensores de intrusão, normalmente utilizados em casas ou escritórios, não podem ser usados em ambientes que necessitem de um monitoramento mais preciso, uma vez que não fornecem informações a respeito de temperatura, pressão, presença gases e outros. Câmeras de vídeo são sensores cujo contexto capturado são imagens. Como diversos sistemas de ambientes virtuais 3D permitem que vídeos sejam mapeados como textura em seus ambientes, estes podem ser usados como complemento dentro do próprio ambiente virtual, se achado conveniente.

Conforme discutido acima, a maioria dos sistemas de monitoramento descritos na literatura apresentam limitações, tais como: não permitem visualização posterior da simulação; são específicas para um determinado tipo de aplicação, portanto não facilmente adaptáveis; não permitem a identificação precisa das situações de risco; etc. Neste trabalho é proposto um sistema que visa superar estas limitações, através do uso integrado de redes de atuadores e de sensores sem fio, ciência de contexto e ambientes de realidade virtual. Este sistema de preparação para emergência poderá ser utilizado em diferentes ambientes físicos (ou mesmo artificiais). Exemplos de uso deste sistema incluem o monitoramento de diferentes plantas industriais, aeronaves em situações de ensaios de vôo e solo, plataformas de exploração de petróleo, meios de transporte em geral (trens, ônibus etc), etc. Para isso, sistemas de visualização de ambientes virtuais 3D via *Web* serão utilizados, permitindo o uso de uma multiplicidade de dispositivos e redes, bem como a aplicação de semiótica na interface a ser gerada. Ambientes virtuais 3D e respectivas tecnologias de suporte para a *Web* serão discutidos no próximo capítulo.

3. Ambientes Virtuais 3D (AV3D) e Tecnologias de Visualização 3D para a *Web*

A Realidade Virtual (RV) pode ser definida como uma interface natural e poderosa de interação homem-máquina, por permitir ao usuário interação, navegação e imersão no ambiente tridimensional sintético gerado pelo computador, através de canais multisensoriais (visão, audição, tato, olfato, paladar etc.). Um Ambiente Virtual Colaborativo (AVC) é definido aqui como um ambiente de realidade virtual em rede em que usuários participantes compartilham um mesmo espaço 3D com o objetivo de trabalho colaborativo, jogos, treinamentos, aprendizagem etc [BOU 05].

Os participantes em um AVC são representados por personagens gráficos chamados de *avatares*, que disponibilizam suas identidades, presenças, localizações e atividades entre si. Os *avatares* interagem com o mundo virtual (mundo geométrico) e também com outros *avatares*, através de diferentes mídias como áudio, vídeo, gestos gráficos e texto. Estes mundos virtuais podem ter várias formas, e as representações usadas para o mundo virtual e para os *avatares* são inúmeras [BEN 01].

Neste trabalho, serão considerados importantes os ambientes virtuais 3D (ambientes de RV), i.e., ambientes que refletem o mundo físico ciente de contexto sendo monitorado através de sensores.

A visualização de AV3D é promovida através de visualizadores (motores gráficos) tipicamente construídos na forma de *software* cliente instalado na máquina do usuário. Quando a visualização é feita através da *Web*, navegadores 3D são utilizados para renderizar, tipicamente, mundos virtuais descritos na linguagem VRML ou X3D.

Alguns navegadores para a *Web* são de código proprietário. São exemplos: *Blaxxun*, *Flux Player*, *Avalon*, *Bitmanagement* e *Octaga*. Exemplos de navegadores 3D de software livre incluem: *CyberX3D*, *FreeWRL*, *X3D Toolkit* e *Xj3D*. Neste trabalho, considera-se importante que a visualização de AV3D que refletem ambientes físicos cientes de contexto seja feita através de navegadores 3D da *Web* para que seja mais fácil o monitoramento desses ambientes de qualquer lugar, a qualquer momento, com a adaptação do sistema para diferentes tipos de dispositivos (*Personal Digital Assistants* – PDAs, celulares, etc). Assim, a próxima seção descreve as principais tecnologias de suporte a AV3D na *Web*.

3.1. Tecnologias de suporte a Ambientes Virtuais 3D na Web

Existem várias tecnologias de padrão aberto para ambientes virtuais (AVs) 3D para *Web*, tais como VRML, X3D, Java3D e MPEG-4. Não serão feitas definições e descrições detalhadas dessas tecnologias, até porque cada uma delas é bastante extensa, mas sim as principais características de cada uma serão apresentadas. Um estudo mais aprofundado sobre o padrão X3D será feito e, ao final, será justificada a sua escolha.

O VRML foi altamente explorado nos últimos anos, porém, o X3D, o Java3D e o MPEG-4 têm surgido como formas poderosas de tecnologias 3D baseadas na Internet que não só competem, mas complementam o VRML.

Atualmente as tecnologias 3D para *Web* provêm a possibilidade de se criar conteúdo tridimensional interativo para a Internet, que podem ser apresentados em diferentes plataformas sem que seja levada em conta a aplicação que irá apresentá-lo.

Dentre os benefícios providos pelas tecnologias 3D baseadas na Internet, pode-se destacar:

- Abstração dos detalhes da plataforma, como mecanismo de representação (*Render Engine*), *drivers* de dispositivo, placas aceleradoras, etc., que devem ser especificadas quando se usa APIs de baixo nível, como o OpenGL e o Direct3D;
- Utilização de estruturas de dados que refletem o grafo da cena, em que os objetos são organizados de forma hierárquica. Essa estrutura é encontrada em tecnologias 3D de alto nível, auxiliando na descrição e codificação da cena.

Uma visão geral de cada uma destas tecnologias será apresentada nas seções seguintes.

3.1.1. VRML

O VRML foi criado inicialmente para permitir o desenvolvimento de ambientes virtuais 3D complexos para a Internet. Chris Marrin [CAM 97] identifica vários problemas que advêm da complexidade da especificação do VRML. Estas limitações resultam na geração de navegadores complexos e instáveis, o que é refletido diretamente sobre o desempenho e a qualidade da apresentação da cena, bem como na integração com multimídia. O VRML não possui um formato binário para descrição de suas cenas, estas são descritas em um arquivo no formato texto ASCII e, portanto, tipicamente são de tamanho grande. Em

decorrência disso e do fato de que a cena deve ser completamente interpretada pelo navegador, antes da apresentação de cenas VRML pode-se observar uma alta latência.

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

Embora a linguagem VRML seja muito utilizada, ela apresenta várias limitações: navegadores existentes não suportam satisfatoriamente texto, áudio e vídeo. O texto é representado como um objeto 2D no ambiente virtual de forma que não possui uma representação realística; o vídeo é mapeado como textura em um objeto 3D e tem de ser, assim como o áudio, completamente carregado para a memória da estação local antes de ser apresentado para o usuário. Como já citado anteriormente, o VRML não suporta um formato binário de arquivo, o que deixa seus arquivos com um tamanho maior.

3.1.2. JAVA-3D

O Java-3D define APIs de alto nível para desenvolvimento 3D em JAVA. A integração da linguagem Java com o Java-3D permite o uso de todo potencial do JAVA em conjunto com a cena Java 3D, a saber, independência de plataforma, controle de janelas, rede e acesso a bancos de dados. Para visualização de cenas são necessários os ambientes J2RE (*Java 2 Runtime Environment*) e J3D (*Java 3D*), que podem estar integrados.

É definido como sendo um conjunto completo de APIs com suporte a características gráficas 3D que estendem a linguagem de programação Java. Com isso, permite a criação de aplicações gráficas 3D de alta performance e independentes de plataforma, por serem executadas através de uma Máquina Virtual Java (JVM – *Java Virtual Machine*) residente no cliente.

A API Java-3D provê um conjunto de interfaces orientadas a objetos que suportam um modelo de programação simples e de alto nível que pode ser utilizado para construir, representar (*render*) e controlar o comportamento de objetos gráficos tridimensionais e ambientes virtuais. A principal vantagem do Java 3D é o seu rico conjunto de APIs disponível em todas as distribuições Java. Com esta API é possível incorporar gráficos de alta qualidade, escaláveis e independentes de plataformas a uma aplicação baseada na tecnologia Java. Outra característica importante do Java 3D é que seus carregadores de classe lêem arquivos de cenas 3D (não arquivos Java) e criam representações em Java 3D do conteúdo dos arquivos, que podem ser seletivamente adicionados a um mundo Java 3D ou aumentados por outro código

em Java 3D. Existem pelo menos 14 carregadores públicos disponíveis, entre eles *3D-Studio*, *AutoCAD*, VTK e VRML. Também é possível escrever carregadores personalizados para o Java 3D. Ele é capaz de gerar visão estérea automaticamente e fazer uso de diferentes dispositivos de entrada sem conhecimento do programador, através do uso de dispositivos de entrada genéricos [SEV 00].

Dentre suas principais desvantagens, pode-se citar o tempo de aprendizado, uma vez que há a necessidade de se conhecer a linguagem Java, e a necessidade da presença da máquina virtual, inclusive em dispositivos com recursos de hardware e software restritos, como PDAs e celulares.

O Java 3D foi criado pela *Sun Microsystems*, mas tornou-se uma API de domínio público em junho de 2004. Atualmente é desenvolvido em comunidade pelo java.net, através da contribuição de diversas pessoas [JAV 06].

Promete alto desempenho, pois utiliza Open-GL ou Direct-3D (que necessitam ser pré-instalados) para fazer uso de placas aceleradoras e, além disso, é capaz de dividir suas tarefas em múltiplos processadores. Em ambos os casos o processo é transparente para o usuário. O Java 3D se escala exibindo gráficos com a melhor qualidade possível, de acordo com o ambiente em que se encontra, usando um único código fonte.

Assim como em JAVA, um arquivo Java 3D é compilado, gerando um *bytecode* de tamanho reduzido. Para programadores avançados, oferece comandos de representação de baixo nível (alterar o número de triângulos da cena, por exemplo).

3.1.3. MPEG-4

O uso da comunicação multimídia (texto, gráficos, áudio e vídeo integrados) atualmente tem crescido muito, com isso, torna-se essencial o uso de padrões, tanto para a indústria quanto para os usuários [TOD 04].

O MPEG (***M****oving* ***P****icture* ***E****xperts* ***G****roup*) é um grupo de trabalho da ISO/IEC responsável pelo desenvolvimento de padrões para representação codificada de áudio e vídeo. Entre os outros padrões que desenvolve, pode-se destacar o MPEG-1 e o MPEG-2, que permitiram a criação de vídeos interativos em CD-ROM, DVD e Televisão Digital [ISO 02a].

O MPEG-4 é um outro trabalho deste grupo que provê elementos tecnológicos padronizados, permitindo a integração dos paradigmas de produção, distribuição e acesso a conteúdo [ISO 02a]. Permite uma grande interação por parte dos usuários, uma vez que sua

cena é uma composição de objetos audiovisuais independentes, sendo possível a inserção, remoção ou alteração de objetos na cena [TOD 04].

Para a produção, ou seja, para os autores, provê um grau de reuso e flexibilidade muito superior às hoje encontradas em tecnologias individuais como Televisão Digital, gráficos animados, páginas da WWW (*World Wide Web*) e suas extensões, ainda especifica uma interface para gerenciamento de propriedade intelectual e proteção, o IPMP (*Intellectual Property Management and Protection*).

Oferece a provedores de serviço informações transparentes que, com a ajuda dos padrões, podem ser traduzidos para mensagens de sinalização nativas de cada rede, também provê um descritor genérico de QoS (*Quality of Service*) para diferentes mídias.

Permite aos usuários uma grande interatividade e também permite seu uso em redes com relativa baixa largura de banda e redes móveis [ISO 02a].

Sua força vem principalmente do seu radical paradigma Orientado a Objeto, permitindo que áudio e vídeo sejam facilmente manipulados. As cenas, que podem ser bidimensionais ou tridimensionais, são compostas por objetos independentes ou agrupados. Os objetos podem ser Naturais (gravados com uma câmera ou microfone) ou Sintéticos (gerados por computador) [SEV 00].

O MPEG-4 busca evitar a multiplicidade de formatos e *players* proprietários e não compatíveis. Para isso, provê caminhos padronizados para [ISO 02a]:

- Representar unidades aurais, visuais ou conteúdo audiovisual, sendo que estes objetos de mídia podem ser naturais ou sintéticos;
- Descrever a composição destes objetos para criar objetos de mídia compostos que formam as cenas audiovisuais;
- Juntar as várias partes (*multiplex*) e sincronizar os dados associados com os objetos de mídia, tal que possam ser transportados através de canais de rede que provêem um apropriado QoS para a natureza dos objetos de mídia específicos;
- Interagir com a cena audiovisual gerada no receptor.

Interações do usuário podem ser feitas local ou remotamente, sendo que alterações na posição do usuário são feitas em seu próprio terminal, característica extremamente importante para respostas rápidas (especialmente em redes lentas), ou quando não existe canal de retorno, como em situações de *broadcast*.

O MPEG-4 provê ainda meios para proteção de direitos autorais, e APIs para acesso a elementos da cena, recursos de rede, de terminal e dispositivos de entrada. Estas APIs são chamadas de MPEG-J. Suporta cenários cliente/servidor (incluindo *broadcast*) e armazenamento em massa (CD, HD, DVD, etc.).

Atualmente existem algumas versões de navegadores MPEG-4 (terminais clientes MPEG-4) disponíveis para *set-top-boxes* interativos. O desenvolvimento do padrão MPEG-4 em telefones celulares e dispositivos móveis em geral está em curso e é apenas uma questão de tempo antes que possam ser vistas aplicações multimídia complexas nestes dispositivos, através de navegadores MPEG-4 [SEV 00].

3.1.4. X3D

O X3D é um sistema de software que integra gráfico 3D e multimídia prontos para serem distribuídos pela rede. Cada aplicação X3D é um universo 3D baseado no tempo que contém gráficos e objetos aurais que podem ser modificados dinamicamente através de uma variedade de mecanismos [WEB 06a].

Após anos de uso do VRML, alguns aspectos da especificação foram revisados e melhorados, resultando no que inicialmente foi chamado de VRML-NG (*New Generation*). Posteriormente passou a ser responsabilidade do *Web3D Consortium*, sob a denominação X3D (*eXtensible 3D*), devido à sua íntima relação com o XML (*eXtensible Markup Language*). Muitas das premissas básicas do VRML foram estendidas, permitindo uma maior flexibilidade. Outras vantagens incluem maior consistência, devido à alteração de alguns nomes de campos, e maior precisão com a iluminação e com o modelo de eventos.

Também são aceitos múltiplos formatos de codificação: a forma “Textual”, que é semelhante ao VRML, a forma “XML” e a forma “Binária”. O formato binário está atualmente em desenvolvimento e uma de suas vantagens é que ele oferecerá melhor rendimento na transferência de dados [WEB 06b].

A arquitetura genérica do X3D tem em vista a importância de que qualquer novo componente integre-se totalmente com os componentes existentes e estenda efetivamente as capacidades da especificação atual [CAR 04]. O X3D é um padrão de software aberto. Permite a especificação de objetos e cenas gráficas tridimensionais em XML, para a comunicação de dados 3D em tempo-real e através de uma ampla variedade de aplicativos. Possui um rico conjunto de características para uso no desenvolvimento de aplicativos para

engenharia e visualização científica, CAD e arquitetura, visualização médica, treinamento e simulação, multimídia e entretenimento, educação e mais.

A codificação XML permite integração estável com *web services* e transferência de dados entre aplicações ou mesmo plataformas [WEB 06b]. Também se integra facilmente a redes distribuídas. Em tempo-real, permite gráficos de alta qualidade, interatividade, áudio, vídeo e dados 3D. Pronto para *broadcast* e aplicações embarcadas, funciona de celulares a supercomputadores.

Dá suporte, além dos já citados gráficos 3D e 2D, a animações, áudio e vídeo espaciais, interação (clicar, arrastar e eventos com o teclado), navegação (movimento de câmeras), detecção de colisão, proximidade e visibilidade, definição de objetos pelo usuário, capacidade de alterar a cena dinamicamente através de programação e linguagens de script, uso de *hiperlinks* e simulações físicas (animação humana, dados geoespaciais, e integração com protocolos de simulação interativa distribuída).

Seu núcleo reduzido proporciona uma melhor eficiência, facilidade de desenvolvimento e manutenção. O X3D pode ser estendido através do uso de componentes (*components*), e perfis (*profiles*). Devido a esta arquitetura modular, o X3D permite que diferentes perfis sejam adotados e suportados pelos diferentes tipos de mercado. Um perfil é uma coleção de componentes para um nível de suporte específico. Os perfis básicos da especificação estão listados a seguir [WEB 06a]:

- Intercâmbio (*Interchange*): perfil básico para comunicação entre aplicações. Suporta geometria, texturas, iluminação básica e animação.
- Interativo (*Interactive*): permite interação básica com um ambiente 3D adicionando vários nós sensores para navegação e interação do usuário, determinação melhorada de tempo e iluminação adicional.
- Imersivo (*Immersive*): dá suporte total a gráficos 3D e interações, incluindo suporte a áudio, colisão, neblina e scripts.
- Completo (*Full*): inclui todos os nodos definidos, incluindo componentes NURBS, H-Anim e GeoSpatial.
- MPEG-4 Interativo (MPEG-4 *Interactive*): é um perfil adicional, uma versão pequena do perfil interativo, desenvolvido para transmissão, dispositivos portáteis e telefones móveis.

- CDF (CAD *Distillation Format*): outro perfil adicional, está em desenvolvimento e permite a tradução de dados CAD para um formato aberto para publicação e mídia interativa.

Possui uma única API unificada, diferentemente do VRML que possui uma API interna e outra API externa. O uso de uma única API no X3D resolveu alguns problemas do VRML e tornou mais robusta e confiável sua implementação [WEB 06a].

O X3D é um padrão aberto que não inclui direitos autorais (*royalties*), ou seja, pode ser usado livremente. O *Web3D Consortium* não requer nenhum limite de Propriedade Intelectual para a tecnologia e possui um acordo com a ISO para publicação da especificação sem nenhum custo.

Ele não é uma API, tampouco um formato de arquivo apenas. Um arquivo X3D combina descrições de geometria e de comportamento e estende o VRML resolvendo suas limitações [WEB 06a]:

- Potencialidade de cena gráfica expandida;
- Modelo de programação de aplicação revisada e unificada;
- Múltipla codificação de arquivos, descrevendo o mesmo modelo abstrato, incluindo XML;
- Arquitetura modular, permitindo uma escala de níveis de adoção e suporte para muitos tipos de mercado diferentes;
- Estrutura de especificação expandida.

O X3D tem conseguido significativo suporte de grandes empresas como *Sun* e *Microsoft*. Existem softwares de referência abertos e também proprietários disponíveis (autoria de conteúdo, ferramentas de edição, visualizadores, navegadores, *plug-ins*, *toolkits* de desenvolvimento, etc). Entre os *softwares* abertos de referência está o Xj3D [XJ3 06] que define um *toolkit* para construção de conteúdo.

3.2. O Xj3D

O *toolkit* Xj3D é um projeto do *Web3D Consortium* escrito totalmente em Java. Pode ser usado para importar conteúdo VRML ou X3D, através de seu navegador padrão ou para criar um outro navegador, totalmente novo. O projeto começou como apenas um carregador (*loader*) para o Java 3D, foi crescendo e atualmente está sendo usado como uma das

principais bases de teste para verificar o trabalho na nova especificação do X3D [XJ3 06]. Embora inicialmente tenha usado apenas o Java 3D como mecanismo de representação, atualmente também suporta OpenGL e um motor de representação específico para dispositivos móveis, entre outros.

Seu código fonte está licenciado sob a licença LGPL (GNU *Lesser General Public License*), que diz que as *"as GPLs (incluindo a LGPL) se destinam a garantir sua liberdade de distribuir e alterar software livre, para garantir que o software é livre para todos os seus usuários, em contraste às licenças para muitos softwares que são feitas para tirar sua liberdade de distribuir e alterá-lo."* [FOU 99].

O Xj3D não trata apenas da implementação de códigos para mostrar arquivos VRML e X3D, também trata da criação do primeiro modelo (*prototype*) e teste de várias partes da especificação antes que ela se torne parte dos processos da ISO (*International Organization for Standardization*) [XJ3 06]. Dessa forma, os códigos contidos no Xj3D sempre estão atualizados, contendo o que há de mais atual quanto à especificação do X3D, e constantemente são corrigidos eventuais *bugs*.

Também é um conjunto de ferramentas (*toolkit*) bastante flexível, podendo ser estendido de várias formas. Isto inclui a possibilidade de permitir carregamento de novos formatos de arquivos, criação de novos tipos de nodos e perfis, entre outros. Também permite a manipulação do grafo de cena através da SAI (*Scene Access Interface*). Esta interface de acesso à cena fornece uma forma compatível para executar e modificar cenas X3D para diferentes implementações do X3D. Algumas funções específicas do Xj3D também estão disponíveis, para aumentar as possibilidades no desenvolvimento de aplicações. Através da SAI é possível carregar arquivos VRML ou X3D e ter completo acesso ao grafo de cena, podendo inserir ou remover nodos, modificar campos e etc.

3.3. Considerações Finais

De forma geral, pode-se dizer que o VRML foi a primeira linguagem de descrição de cenas 3D de alto nível desenvolvida, o que facilitou a criação de conteúdos, livrando as pessoas que desenvolvem da necessidade de conhecerem detalhes de implementação.

Por outro lado, o foco principal do Java 3D são os programadores de aplicações, suas capacidades (*loaders* e escalabilidade, por exemplo) ajudam bastante àqueles que já possuem mundos virtuais em outros padrões. Outra grande vantagem é sua total integração com a

linguagem Java, ou seja, toda a portabilidade e capacidade de acesso (ao SO, à Rede, à Banco de Dados, etc) desta linguagem estão disponíveis para auxiliar na criação das cenas gráficas.

Diferente dos outros padrões, o MPEG-4 é um completo *framework*, trata da composição, representação, compressão, sincronização e distribuição de seus objetos. Juntamente com seu formato de codificação BIFS, provê uma grande capacidade de compactação e distribuição em *broadcast*. Já se pode encontrar navegadores MPEG-4 em PDAs e celulares, também cogita-se a hipótese de sua adoção como padrão para a TV Digital do Brasil [LAM 04][LEM 04][MAR 04]. O comitê que irá decidir o padrão a ser implantado solicitou novo adiamento para que possam finalizar as negociações comerciais. Esta decisão deveria ter sido tomada dia 10 de fevereiro de 2006, sofreu prorrogação por 30 dias e agora teve nova prorrogação por prazo indeterminado [MAR 06].

Por sua vez, o X3D provê grafos de cena codificados em XML e uma interface para autoria de cenas, a SAI (*Scene Authoring Interface*), que permite acesso total ao grafo de cena. É um formato de arquivo poderoso e extensível para efeitos visuais 3D, modelagem comportamental e interação. Sua codificação em XML facilita a mobilidade de cenas entre aplicativos e permite seu uso em *web services*. A SAI permite que conteúdo em tempo-real e controles sejam integrados em uma ampla gama de aplicações e, além disso, não cobra direitos autorais.

Dentre as necessidades que foram supridas pelo X3D, pode-se destacar:

- Criação de cenas numa linguagem de alto nível, possivelmente através de ferramentas de exportação em aplicativos como o *3D Studio Max* ou *Blender*;
- Suporte em vários ambientes, uma de suas implementações, o Xj3D, é desenvolvido em Java, o que provê grande portabilidade;
- Alteração da cena por eventos possível através da SAI;
- Distribuição de conteúdo, como possui o formato XML permite portabilidade, podendo utilizar inclusive *web services*.

Destacando-se a implementação Xj3D do padrão X3D, podem-se acrescentar as seguintes características:

- Sempre atualizado, é uma versão que contém o que há de mais novo em relação à especificação X3D, uma vez que é desenvolvida pelo próprio criador do padrão e uma

de suas aplicações é servir como ambiente de teste para novas características do padrão.

- Flexibilidade, uma vez que as APIs da SAI permitem a criação de um navegador através da linguagem Java, todas as funcionalidades de rede e outras disponíveis nesta linguagem também podem ser usadas para criação das cenas. Sua interface também pode ser estendida, provendo novas funcionalidades, através de componentes do próprio Java.

Para este trabalho, a tecnologia X3D foi considerada como a melhor opção, porque além de suprir todos os requisitos da aplicação, disponibiliza o *toolkit* Xj3D. Este *toolkit* facilita a implementação e garante conformidade com o padrão X3D. Uma vez apresentadas as ferramentas para visualização de ambientes virtuais 3D, a próxima seção apresenta os sistemas existentes para gravação e reprodução destes, mostrando suas principais características.

4. Gravação e Reprodução de Mídia 3D

Os ambientes monitorados muitas vezes necessitam, além da possibilidade de monitoramento em tempo-real, ter suas cenas gravadas para posterior reprodução. Exemplos de uso destas gravações é o treinamento de equipes, como no caso do treinamento dos membros do corpo de bombeiros, e a realização de perícias, onde uma empresa seguradora pode analisar de forma precisa todos os acontecimentos anteriores, que deram origem ao problema, para descobrir sua causa. Nas próximas seções serão descritos e discutidos alguns sistemas de destaque na literatura para gravação e reprodução de mídia 3D.

4.1. RPBIMS

Effelsberg et al. (2004) descreve o RPBIMS (*Recording and Playing Back Interactive Media Streams*), algoritmos de gravação que foram implementados no IMoD (*Interactive Media on Demand*) e que são uma solução para gravação e reprodução aleatória de mídia interativa. Seu modelo para gravação de mídias interativas depende do uso do protocolo RTP/I para a distribuição da mídia. O RTP/I foi inicialmente inspirado no protocolo RTP [APP 05], que faz uso de muitas funcionalidades deste e está adaptado para atender as necessidades específicas das mídias interativas distribuídas. Define alguns campos de cabeçalho, por exemplo: Marca de tempo; Identificador de componente e Tipo de dado (evento, estado-delta ou estado). Define também alguns mecanismos comumente necessários, como a capacidade de requisição do estado de um subcomponente.

O não uso do protocolo RTP/I causa a necessidade da adaptação do algoritmo de gravação, para que este seja capaz de extrair do protocolo em uso algumas informações sobre a semântica do fluxo de dados.

É necessário que todos os estados a serem gravados sejam transmitidos pela rede, gerando grande desperdício de largura de banda e limitando a quantidade de gravações de estados possíveis de serem realizadas. Como parte do protocolo, uma requisição de cena também pode ser ignorada, caso a máquina que deve enviar o estado esteja ocupada no momento. Isto reduz a qualidade da gravação, embora não coloque em perigo sua integridade [EFF 04].

4.2. RRCVRS

Georganas e Hosseini (2002) em seu artigo RRCVRS (*MPEG4 Based Recording and Replay of Collaborative Virtual Reality Sessions*) apresentam uma solução para a gravação de sessões de realidade virtual colaborativa, fazendo uso da tecnologia MPEG-4.

O gravador é inserido na seção colaborativa apenas como um receptor, ou seja, recebe os eventos durante a seção como qualquer outro colaborador. Estes eventos são então armazenados, juntamente com uma marca de tempo [GEO 02]. Como os dados gravados são constituídos apenas de eventos e nenhuma informação para reconstrução da cena é gravada, este sistema não permite acesso aleatório às suas gravações. Uma vez que só é possível acesso seqüencial, o início da reprodução de um instante avançado da gravação é custoso e demorado [EFF 04].

Para a sincronização dos diversos fluxos faz uso de alguns mecanismos do próprio MPEG-4, como o OCR (*Object Clock Reference*) e o DTS (*Decoding Time Stamp*). As gravações são armazenadas em arquivos locais que posteriormente poderão ser reproduzidas diretamente por qualquer navegador MPEG-4.

4.3. DIVE

O DIVE (*Distributed Interactive Virtual Environment*) é um sistema de RV distribuído e heterogêneo, baseado no UNIX e em protocolos de rede da Internet. É uma arquitetura de software para a realização e implementação de uma ampla variedade de ambientes virtuais multiusuários baseados na Internet. Cada processo participante tem uma cópia do banco de dados replicado e as alterações são propagadas aos outros processos através de protocolos *multicast* confiáveis [CAR 93].

Existe um conjunto de ferramentas, o *Dive-toolkit*, que, entre outras ferramentas, possui o *DiveBone*, que permite a análise visual da arquitetura e do tráfego da rede. Também existem navegadores para visualização e aplicações que geram *logs* [FRE 99]. Embora estes *logs* tenham sido criados inicialmente para análise do padrão de uso da rede por parte dos usuários, o sistema foi estendido para permitir que estes *logs* sejam reproduzidos, ou seja, é possível rever uma sessão através dos *logs* gravados [BEN 02]. Desta forma, o mesmo problema do RRCVRS pode ser notado: permite apenas acesso seqüencial às suas gravações, dispensando muito processamento e tempo no início da apresentação de um instante aleatório da gravação.

4.4. MASSIVE-3

O MASSIVE-3 é um sistema multiusuário de RV para AVCs que suporta mundos virtuais povoados e interativos, combinando gráficos 3D, áudio em tempo-real e fluxo de vídeo. Seus mundos virtuais podem ser estruturados a partir de vários locais (espaços virtuais arbitrários) com seu próprio sistema de coordenadas.

Suas gravações são feitas num formato específico e permitem acesso aleatório. Uma gravação captura todos os movimentos, interações e a fala do usuário em um local.

O MASSIVE-3 permite que locais sejam estendidos, uma vez que podem ser ligados a gravações de outros locais. Neste cenário, um usuário que está no local corrente é capaz de observar as ações virtuais gravadas acontecendo ao seu redor, e pode se mover livremente (o usuário pode interagir somente com o local atual). Este tipo de ligação é chamado de ligação temporal (*temporal link*).

A característica chave das ligações temporais é sua flexibilidade, no que diz respeito a como os locais correntes são relacionados aos locais gravados. Um local pode ser ligado a uma sessão passada dele próprio. Por exemplo, o usuário pode estar andando pelo ambiente com seu *avatar* atual, ao mesmo tempo em que vê seu *avatar* gravado caminhando por onde ele passou na sessão gravada. Como mencionado anteriormente, também permite que a gravação de um local diferente do atual seja usada.

As gravações podem ser inseridas com deslocamento temporal, em velocidade diferente, ou ainda, adiantando ou retrocedendo. Relacionamentos espaciais também são possíveis, como posição, rotação e tamanho. A aparência do material gravado pode sofrer algum efeito como, por exemplo, estar transparente (efeito “fantasma”), sem cor (escala de cinzas) ou normal [BEN 02].

Em suma, as gravações podem ser reproduzidas através das ligações temporais. A flexibilidade destas permitem que uma ou várias gravações, do mesmo ou de diferentes locais sejam reproduzidos ao mesmo tempo, sendo que cada um pode ter suas propriedades, já apresentadas, alteradas independentemente.

Embora disponha de funções poderosas, é um *software* comercial. Sendo assim, a agregação de novas funcionalidades, como o monitoramento de ambientes físicos, é bastante limitada.

4.5. WhereWereWe

O *WhereWereWe* é uma API para desenvolvimento de programas que utilizem objetos distribuídos. Contém a ILU (*Inter-Language Unification*), um sistema de objetos distribuídos que permite aos usuários construírem objetos facilmente. Estes existem em um espaço de endereços em uma máquina, mas têm objetos "substitutos" que existem no mesmo ou em outro espaço de endereços, na mesma ou em outra máquina da rede. As cópias dos objetos fazem chamadas RPC (*Remote Procedure Call*) aos objetos reais, os quais, de fato, realizam as operações.

Programas que desejam gravar ou reproduzir dados através do *WhereWereWe* devem implementar esta API, que possui sete abstrações básicas. Dentre elas, *sessões*, *fluxos* e *eventos*, que podem ser entendidos da seguinte forma: sessões são coleções nomeadas de fluxos que correspondem a eventos semânticos, como "Aula XYZ". Os fluxos podem conter áudio, vídeo ou um *log* de atividades de um programa. E os eventos são acontecimentos que se sucedem no fluxo [HAR 95]. Os dados dos fluxos normalmente são um simples arquivo em disco que podem ser reproduzidos posteriormente.

Embora o sistema *WhereWereWe* forneça uma infra-estrutura na qual serviços de reprodução de mídia podem ser inseridos e utilizados [HAR 95], nenhum serviço que realize tal tarefa foi encontrado. Sua utilização em conjunto com sistemas de AV3D também é custoso, uma vez que estes necessitariam ser adaptados para suportarem a infra-estrutura do *WhereWereWe*.

4.6. Analogia à compactação do MPEG.

Deseja-se gravar o fluxo de mídia 3D de forma que sua posterior reprodução possa ser iniciada em um ponto aleatório. A compactação de vídeo 2D no padrão MPEG tem algumas características que poderão ser úteis no desenvolvimento de técnicas para a gravação da mídia 3D. De modo geral, na gravação de vídeo são gravadas figuras estáticas, os chamados quadros, de forma que ao serem exibidos seqüencialmente geram a sensação de movimento. No padrão MPEG existem 3 tipos de quadros comprimidos, chamados de I (*Intraframe*), P (*Predicted*) e B (*Bidirectionally Predicted*).

Nos quadros do tipo I apenas a redundância espacial é utilizada para compressão, permitindo o acesso a estes quadros diretamente, uma vez que são independentes dos outros. Os quadros do tipo P e B utilizam algoritmos para redução da redundância temporal através

da predição de movimento, ou seja, sozinhos não possuem informação suficiente para exibição de um quadro. Para reconstrução de um destes quadros é necessário processá-lo juntamente aos seus quadros anterior (P ou B) e posterior (somente B).

Embora os quadros do tipo B sejam os que apresentam a maior taxa de compressão, seguido pelos do tipo P, necessita-se intercalar quadros do tipo I de tempos em tempos para que a qualidade das imagens seja restaurada e, como já citado, para permitir acesso aleatório aos quadros do filme [CHE 95].

Utiliza diversos métodos matemáticos para a compressão, entre eles: Transformada Discreta do Cosseno (DCT), Quantização, *Run Length Encoding* (RLE), *Motion Estimation* e *Motion Compensation* (MEC) e Codificação de *Huffman*.

Ao se considerar que fluxos de mídia 3D podem conter comandos de alteração de cena, pode-se fazer uma analogia à compactação MPEG. O comando que realiza a troca completa da cena pode ser comparado aos quadros I, uma vez que é suficiente para construir toda a cena sem nenhuma informação adicional. Os outros comandos, que alteram apenas partes da cena, podem ser comparados aos quadros do tipo P, pois dependem do estado anterior em que a cena se encontrava, para que gerem o efeito desejado.

4.7. Considerações Finais

Um esquema de gravação simples é o *event-replay* (melhor discutido na Seção 6.1.1) que, como o próprio nome sugere, consiste em apenas armazenar os eventos, juntamente com seu tempo de execução. Para reprodução, tais eventos são executados de acordo com os valores de tempo também armazenados e, para acesso aleatório, todos os eventos, desde o início, são reproduzidos até o ponto que se deseja. Um dos problemas desta abordagem é que se houver um grande histórico de eventos, será gasto muito processamento para se encontrar o ponto desejado e, conseqüentemente, poderão acontecer grandes esperas até que a reprodução se inicie. Um outro problema que pode ser observado é que objetos que não são mais usados, mas que foram usados no passado, serão carregados e descarregados desnecessariamente, consumindo recursos [EFF 04].

O DIVE é um exemplo deste tipo de abordagem e que exemplifica a possibilidade de se reproduzir um ambiente a partir de um simples *log*, simples no sentido de que não possui nenhuma informação adicional a não ser os eventos que foram ocorrendo. O *WhereWereWe* é outro exemplo, este armazena todos os fluxos, inclusive os eventos, na forma de arquivos. É

flexível no tocante à possibilidade, devido à sua API (note-se que o programa deverá fazer uso desta), de gravar eventos de diferentes aplicativos, como de um aplicativo de edição de texto ou imagem, por exemplo. O RRCVRS também utiliza a técnica *event-replay* e suas gravações podem ser reproduzidas diretamente por um aplicativo compatível com o formato gravado. Nestes três casos pode-se observar que a abordagem *event-replay* é usada e, portanto, os problemas apresentados anteriormente se aplicam a estes.

A técnica de *rollback*, que é freqüentemente usada em banco de dados e aplicações distribuídas interativas, também pode ser usada para possibilitar a reprodução dos eventos em sentido inverso. Para isso, necessita-se que no *log* seja armazenado, para cada evento, seu correspondente contra-evento. Por exemplo, o contra-evento de uma inserção de um nó é a remoção do mesmo e, a alteração do valor de uma propriedade, voltá-la para seu valor anterior. O MASSIVE-3 é um exemplo de aplicação que possui a possibilidade de reprodução das cenas gravadas em sentido inverso.

O RPBIMS e o *WhereWereWe* exigem que aplicações já desenvolvidas sejam adaptadas para serem capazes de fornecer as informações para este novo protocolo. Apesar dessa necessidade, é um pouco mais flexível que o MASSIVE-3, uma vez que este utiliza um formato proprietário.

A análise da codificação de vídeo usando MPEG também é interessante, uma vez que as características de sua compactação se assemelham às características da gravação de mídia 3D, sendo esta última sem compactação.

Como visto, as diferentes aplicações têm seus enfoques, atendendo às necessidades que se propõe. Neste trabalho, deseja-se a possibilidade de acesso aleatório, uma vez que as gravações terão durações que podem ser longas (teoricamente a gravação se inicia e não pára mais, uma vez que o monitoramento pode ser contínuo), deseja-se também que as cenas sejam portáveis, ou seja, possam ser vistas em diferentes sistemas operacionais e arquiteturas, incluindo dispositivos móveis.

Para a gravação das cenas e dos comandos de alteração destas, um banco de dados será utilizado. Desta forma será possível que a aplicação servidora acesse aleatoriamente os eventos.

Uma vez que as características destes sistemas de gravação foram apresentadas, o próximo capítulo descreve o sistema de visualização em tempo-real utilizado no projeto. Este sistema é de grande importância para a compreensão do sistema de gravação proposto, o qual

será utilizado na gravação dos ambientes virtuais 3D utilizados no projeto de preparação para emergência do *LRVNet*.

5. Linguagem Visual e Projeto de Visualização

No projeto de preparação para emergência, para o monitoramento de ambientes físicos cientes de contexto, as cenas são vistas em tempo-real e também posteriormente, através das gravações/reproduções de situações que ocorreram no ambiente físico. Essas situações são interpretadas e mapeadas, através de linguagem visual, criada como parte deste projeto. Neste capítulo será descrito o sistema de gravação/reprodução projetado, bem como a linguagem visual desenvolvida em conjunto com o departamento de Artes e Comunicação (DAC) da UFSCar, além da descrição dos comandos que alteram a cena no navegador do cliente e a tradução dos eventos capturados pelos sensores para esses comandos. A Figura 4 mostra todos os módulos envolvidos no sistema de gravação/reprodução projetado como parte deste trabalho.

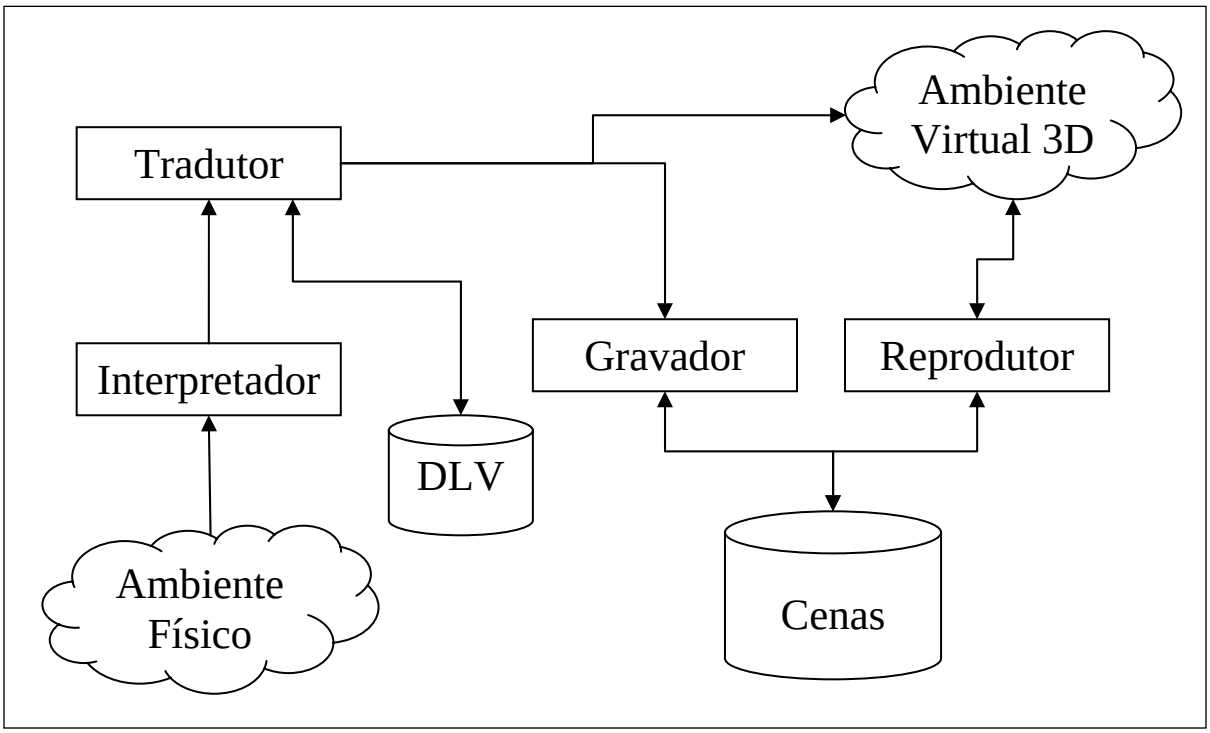


Figura 4 - Visão Geral do Projeto

Os módulos do sistema são descritos a seguir:

- Ambiente físico – elemento físico (prédio, avião, sala etc) povoado de sensores, que é o objeto de monitoramento.
- Interpretador – transforma dados capturados pela RASSF em informações úteis ao tradutor, constitui-se de um *middleware* ou da própria RASSF.

- Tradutor – traduz as informações vindas do Interpretador em comandos que podem ser executados por um navegador cliente. Utiliza o DLV para a tradução.
- Dicionário da Linguagem Visual (DLV) – Dicionário que contém informações para tradução. Estas informações devem ter como base a linguagem visual, para que tenham relevância e clareza.
- Gravador – módulo responsável por gravar os comandos de alteração da cena no repositório de Cenas.
- Reprodutor – módulo responsável por receber requisições dos navegadores dos clientes e enviar a eles os dados do repositório de Cenas.
- Cenas – banco de dados ou outra estrutura responsável por armazenar as Cenas e comandos de alteração destas. Contém as gravações realizadas pelo sistema.
- Ambiente virtual – ambiente de RV que imita o ambiente físico sendo monitorado.

De acordo com o que pode ser observado na Figura 4, depois que os dados são coletados pela rede de sensores e são interpretados, as informações obtidas são enviadas ao tradutor. Este tradutor é responsável por traduzir estas informações em linguagem visual através do Dicionário da Linguagem Visual (DLV). Basicamente, a linguagem visual constitui-se de comandos para alteração da cena em conseqüência de um evento sentido pela rede de sensores. Uma vez que a cena pode ser alterada arbitrariamente (parâmetros podem ser alterados, nós podem ser inseridos ou removidos), existe a necessidade de se especificar exatamente o que irá acontecer no ambiente virtual em resposta a cada evento possível de ser gerado na rede de sensores. Isto é feito através do DLV, sendo que cada um destes eventos deve ter uma entrada correspondente neste dicionário. Mais informações a respeito da Linguagem Visual e do DLV podem ser vistas nas Seções 5.2 e 5.3.

Como resultado da tradução, são obtidos comandos para alteração da cena. Estes comandos, como visto, devem estar em conformidade com a linguagem visual. Os comandos de alteração da cena podem ser enviados diretamente ao navegador do cliente, para serem exibidos, ao gravador, para serem armazenados, ou a ambos. O gravador armazena estes dados em um repositório para que possam ser re-exibidos futuramente, através do reprodutor. A tarefa básica do reprodutor é receber as requisições dos navegadores dos clientes e processá-las. Seu processamento consiste basicamente em enviar ao navegador do cliente as

informações para reconstrução da cena inicial e os sucessivos comandos que aconteceram durante o intervalo de tempo solicitado.

Por sua vez, o Ambiente Virtual 3D contém a cena 3D que representa o ambiente físico, em tempo-real ou que foi gravado para reprodução posterior. Esta cena é exibida através de um navegador na máquina do cliente. Como será visto mais adiante, existe uma flexibilidade quanto ao navegador utilizado: pode-se utilizar diferentes navegadores e diferentes tecnologias de AVs para *WEB*; os testes realizados neste trabalho utilizam-se de um navegador construído a partir do *toolkit* Xj3D, já discutido na Seção 3.2.

Definiu-se que as gravações devem ocorrer após a tradução, ou seja, serão gravados comandos já no formato X3D. Uma outra possibilidade seria efetuar a gravação antes dos eventos serem traduzidos para a linguagem visual. Se esta opção fosse escolhida, haveria a necessidade de se desenvolver uma arquitetura para determinação do estado atual da cena, tarefa esta já executada pelo Xj3D. Este, além de gerenciar o grafo da cena através da SAI, fornece acesso total a este grafo. Sendo assim, para se saber o estado atual da cena basta utilizar o grafo de cena do Xj3D. É óbvio que este grafo deverá ser atualizado a cada comando, assim como o navegador do cliente, para que represente o estado atual de forma correta.

Como armazenamento é utilizado um banco de dados *MySQL*. Uma outra forma seria utilizar um formato de arquivo próprio em que fosse possível e eficiente o acesso aleatório a cenas, assim como determinar qual o próximo evento a ser executado. É provável que o desempenho desta forma seja superior ao adotado, que utiliza banco de dados. Tal comparação será proposta como trabalho futuro.

Pode ser observada a interação entre os diferentes módulos do projeto mais detalhadamente na Figura 5. Esta interação também é vista através da rede, possivelmente a Internet, entre o servidor e os navegadores dos clientes.

As informações vindas da Rede de Atuadores e Sensores Sem Fio (RASSF) vão diretamente para o servidor que, após processá-las (basicamente apenas tradução), as envia aos clientes conectados para visualização em tempo-real. Este servidor pode disponibilizar as cenas em diferentes formatos, como os já apresentados VRML, X3D e MPEG4, bastando utilizar um simples conversor, capaz de converter de um dos formatos disponíveis para o formato desejado.

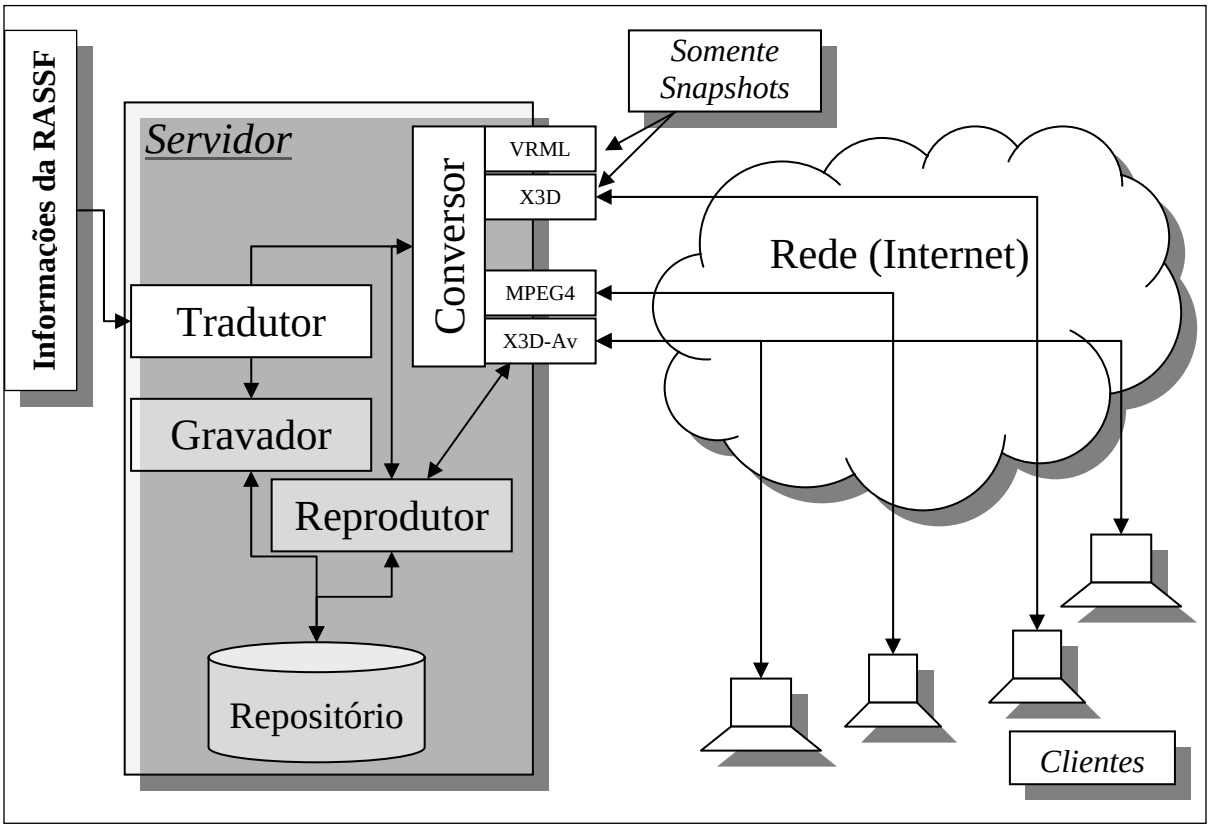


Figura 5 - Interação entre os módulos do projeto

Embora não esteja sendo utilizado, um exemplo desse tipo de conversor é o MP4Box, disponibilizado junto ao GPAC (*GPAC Project on Advanced Content*). O GPAC é uma estrutura (*framework*) multimídia baseada no padrão de Sistemas do MPEG-4 (MPEG-4 Systems - ISO/IEC 14496-1) e é desenvolvido desde o início em ANSI C, provendo portabilidade de código para diferentes plataformas. Existem versões disponíveis para *Windows*, *Linux*, *WindowsCE* e *PocketPC*, sendo possível utilizar seu código em plataformas embarcadas e DSPs (*Digital Signal Processing*). É licenciado sob licença LGPL e permite a conversão entre os formatos VRML, X3D e MPEG-4, e variações destes últimos (textual, XML e binário).

A vantagem de se criar um servidor independente de tecnologia é a possibilidade de suporte futuro para diferentes tecnologias, por exemplo, a possibilidade de disponibilizar um fluxo de mídia MPEG-4 através do protocolo RTSP na porta 554 e simultaneamente disponibilizar cenas no formato X3D através da X3D-Av (cenas no formato X3D e suporte a controles avançados), utilizando uma outra porta qualquer. Atualmente apenas o formato MPEG-4 permite *streaming* de suas cenas, sendo provável que o padrão X3D também passe a suportar quando seu formato binário estiver concluído. Apenas navegadores que suportem *streaming* de dados podem receber as atualizações. Navegadores que não suportem *streaming*,

podem visualizar apenas uma versão estátiva da cena atual (*snapshot*), a menos que seu código seja alterado ou um *plug-in* seja instalado. Neste trabalho, o Xj3D foi estendido para a criação de um navegador com suporte a *streaming*. No servidor, apenas a interface X3D-Av foi desenvolvida, disponibilizando cenas no formato X3D, comandos de atualização e controles avançados. Os comandos de atualização foram desenvolvidos utilizando XML e sua sintaxe foi baseada nos BIFS-COMMANDS [GEO 01] do MPEG-4 (formato XMT [ISO 02b]). Os controles avançados permitem que o usuário, através da interface avançada existente no navegador, escolha entre monitorar o ambiente em tempo-real ou visualizar cenas previamente gravadas. No caso da visualização de cenas gravadas ainda é possível pausar a reprodução, avançar evento-a-evento ou avançar rapidamente. Estes controles serão vistos em maiores detalhes na Seção 6.3.1.

Quanto à arquitetura do servidor, já foi visto que o processamento se inicia no tradutor, que converterá as informações vindas da RASSF em linguagem visual. Os comandos que compõem a linguagem visual são então enviados simultaneamente aos clientes conectados para visualização em tempo-real e ao gravador.

O gravador pode ser parte integrante do servidor ou uma aplicação separada que recebe os eventos da mesma forma que os navegadores dos clientes. Uma vez recebido um evento, realiza a gravação de acordo com uma das técnicas apresentadas na Seção 6.1. Estas gravações são armazenadas no repositório para serem utilizadas pelo reprodutor em resposta às solicitações dos clientes.

Somente os clientes que utilizam o serviço de controles avançados podem utilizar o reprodutor. Este serviço não está disponível aos outros clientes, uma vez que é preciso especificar o instante em que a reprodução deve iniciar e deve-se controlar a reprodução. O controle da reprodução é feito através dos botões de pausa e avanço rápido, por exemplo, e não pode ser realizado sem as funcionalidades disponibilizadas por tal serviço.

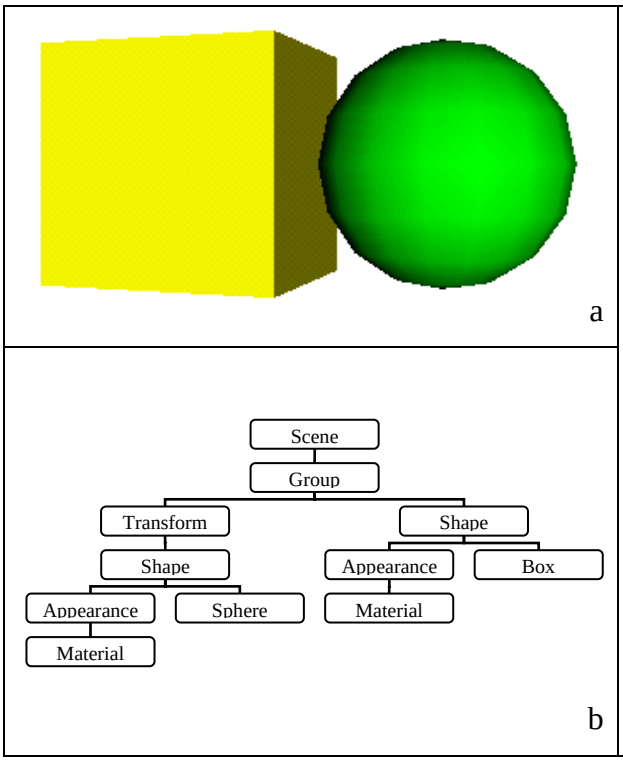
5.1. Descrição das cenas

As tecnologias de ambientes virtuais 3D para *Web* apresentados na Seção 3.1 descrevem suas cenas através de grafos acíclicos e conexos (árvores). O grafo de cena representa a hierarquia dos elementos da cena, sendo que existem dois tipos de nós neste grafo: os nós folha e os nós de agrupamento. Os nós folha contêm a definição das formas geométricas: luz, neblina, sons, etc. Os outros, como o próprio nome sugere, agrupam nós

filhos. Cada nó pode ter somente um nó pai e possui campos que especificam suas propriedades.

A maioria das tecnologias de AV3D permite que o grafo da cena seja lido e alterado em tempo-real, sendo que as alterações neste grafo são refletidas diretamente na cena apresentada pelo navegador. A leitura do grafo da cena é importante, pois como será visto adiante, no momento da gravação é necessário gravar o estado atual da cena. Esta gravação é necessária para que se tenha um melhor desempenho durante o acesso a um instante aleatório. O estado atual da cena nada mais é do que o grafo da cena em um dado instante de tempo, sendo que a informação necessária para a gravação do estado atual de um ambiente virtual 3D pode ser capturada diretamente deste grafo.

Na Figura 6 é apresentada uma cena 3D, o grafo que a representa e a sua descrição em X3D.




```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN"
"http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
xsd:noNamespaceSchemaLocation="http://www.web3d.org/specificat
ions/x3d-3.0.xsd" version="3.0">
<Scene>
 <Group>
  <Transform translation="2.1 0 0">
   <Shape>
    <Appearance>
     <Material DEF="M1" diffuseColor="0 1 0"/>
    </Appearance>
   <Sphere/>
  </Shape>
 </Transform>
 <Shape>
  <Appearance>
   <Material DEF="M2" diffuseColor="1 1 0"/>
  </Appearance>
  <Box/>
 </Shape>
</Group>
</Scene>
</X3D>
```

c

Figura 6 - a) Cena 3D, b) Grafo que representa esta cena e c) descrição desta cena em X3D

Considerando-se gravações sem compressão, a gravação e a reprodução de mídia 3D é mais complexa do que a de mídia contínua, por causa desses estados de cena. Embora existam várias ferramentas para gravação de mídia 2D, existem poucas para mídia 3D, provavelmente devido às suas características mais complexas [EFF 04]. Devido à dependência entre os eventos, é necessário saber sobre o estado da mídia antes de iniciar a reprodução (em

gravações de mídia contínua, sem compactação, é possível iniciar a reprodução a partir de qualquer ponto, pois informações anteriores ao ponto desejado são irrelevantes).

Para acesso aleatório aos nós, sem a necessidade de se percorrer toda a árvore, os nós podem ter nomes que os identifique. De acordo com a Figura 6.c, isto é feito especificando o nome dos nós *Material* através do campo *DEF*. O posterior acesso no Xj3D pode ser feito através do comando *getNamedNode(nodeName)*.

A seção a seguir trata da linguagem visual. É esta linguagem que define como as informações vindas da RASSF serão traduzidas em comandos compreensíveis pelo navegador.

5.2. Linguagem Visual

Uma linguagem visual foi criada que considera conceitos de semiótica para a representação imagética de situações de emergência que podem ocorrer em ambientes físicos cientes de contexto. A vantagem de se usar uma linguagem visual utilizando-se desses conceitos é a facilidade de identificação de situações de emergência de forma mais rápida. Fundamentos de semiótica são apresentados no Anexo A.

Ainda tratando da visualização das cenas em tempo-real, a Linguagem Visual (LV) é a responsável por representar os comandos que serão utilizados pelo navegador do cliente, em resposta às informações vindas da RASSF. Isto deve ser feito de forma que as alterações na cena sejam claras e representativas do que está acontecendo no ambiente físico.

Para a especificação desta linguagem contou-se com a colaboração do Departamento de Artes da UFSCar, mais especificamente do professor Leonardo Andrade e do aluno Francisco Gaspar. Como resultado das reuniões realizadas, foram especificados desde a interface do navegador, botões necessários, posicionamento de componentes e cores até a especificação da LV. A próxima seção descreve a interface gráfica implementada.

5.2.1. Interface Gráfica para o Monitoramento de AV3D

A linguagem visual desenvolvida buscou representar da melhor forma alguns eventos passíveis de ocorrerem em qualquer ambiente de condição crítica, sendo que algumas representações foram desenvolvidas especificamente para o monitoramento de aeronaves. A construção de uma interface gráfica coerente e que faça a função de comunicar de forma objetiva é regulamentada pelas características de composição imagética que já foram citadas, como o equilíbrio, o contraste, as formas geométricas, as cores e a similaridade.

No caso das interfaces de monitoramento, existem outros fatores que contribuem para sua composição, como a função da interface, o que irá ser monitorado e quem irá monitorar. É importante que se saiba o motivo pelo qual essa interface está sendo utilizada e quem irá fazer o monitoramento, pois, toda a informação visual deve ser coerente com a proposta do monitoramento e com o imaginário de quem monitora.

A interface do AVC para monitoramento de uma aeronave é exibida na Figura 7. Sua identidade visual deve ser contida, não possuindo extravagâncias que possam vir a comprometer a troca de informação. As cores utilizadas devem ser neutras, em tom de cinza, indicando que não há nenhum evento. No caso de acontecer algum, a cor utilizada deve chamar a atenção, indicando estado de alerta, como o vermelho e o laranja.

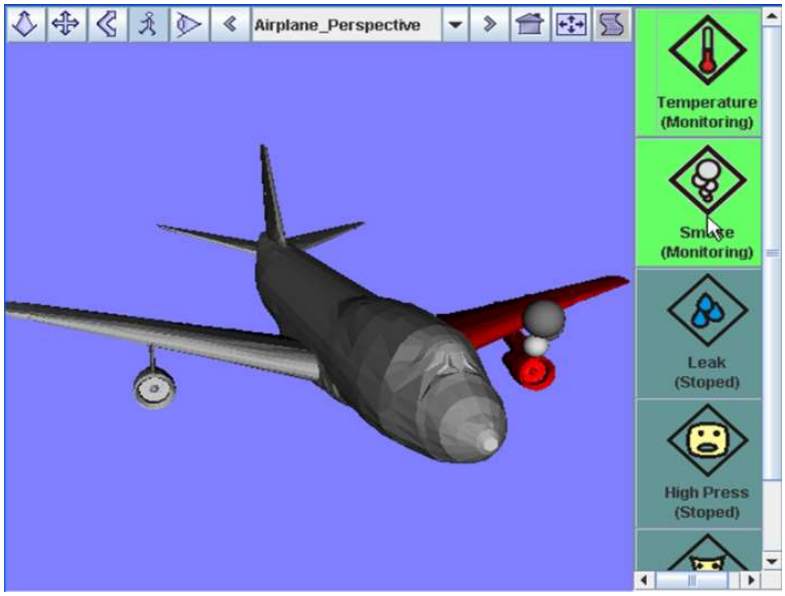


Figura 7 - Interface de Monitoramento

A organização da informação gráfica deve estar em equilíbrio com o menu de seleção de eventos a serem monitorados, localizado na borda lateral, e com a barra de controle da mídia, localizada na borda inferior. Esta última permite a escolha entre visualizar a situação atual do ambiente, bem como navegar nas gravações, da mesma forma como é feita para vídeos bidimensionais. Assim, estas duas barras compreendem as linhas verticais e horizontais, como discutidas anteriormente, facilitando o acesso ao menu e não prejudicando a visualização do gráfico de monitoramento. Os botões do menu devem estar em uma cor verde fosca quando não estiverem em funcionamento e em verde claro “vivo” para indicar que o botão foi ativado. A cor verde foi escolhida para indicar o funcionamento ou não, pois ela possui um pequeno espectro, não sendo uma cor agressiva, porém, que chama a atenção quando colocado em uma interface em que se predomina o cinza.

Na borda lateral, já se encontra o menu de acesso aos eventos que serão monitorados, como o fogo, pressão, temperatura, fumaças, gases, etc. Para monitorar um desses eventos, basta selecionar o símbolo respectivo na barra de menu. Uma outra melhoria na interface é a interligação entre estes botões e os eventos. Dessa forma, se alguma parte da aeronave pegar fogo, o símbolo de fogo no menu de eventos começa a piscar em vermelho, chamando a atenção do usuário.

A simbologia utilizada para representar os eventos, tanto nos menus quanto no desenho do objeto monitorado, devem fazer parte do universo imagético comum de todos, para que a interpretação seja feita de forma fácil, sem complicações e por qualquer pessoa. Foi escolhido utilizar símbolos e não imagens icônicas, pois os ícones podem ser muito peculiares para indicar alguma parte ou evento. Sendo assim, a compreensão pode ser prejudicada caso a pessoa que faça o monitoramento não conheça aquele ícone. A representação por símbolos é de compreensão mais fácil, pois se comunica com o imaginário da pessoa; muitas vezes essa representação é mais eficiente do que o próprio ícone do objeto.

Figura 8 - Escala de Cores - Temperatura

A temperatura pode ser representada acordo com uma escala de cores, apresentada na Figura 8. Os objetos sendo monitorados têm suas cores alteradas de acordo com a temperatura mapeada nesta escala. As variações cromáticas, em ordem crescente de temperatura, são as seguintes: azul – verde – amarelo – laranja – vermelho. Essa escala foi baseada na teoria de Max Planck, do Espectro de Radiação do Corpo Negro. Ele é comumente utilizado para representar a emissão de temperatura por corpos.

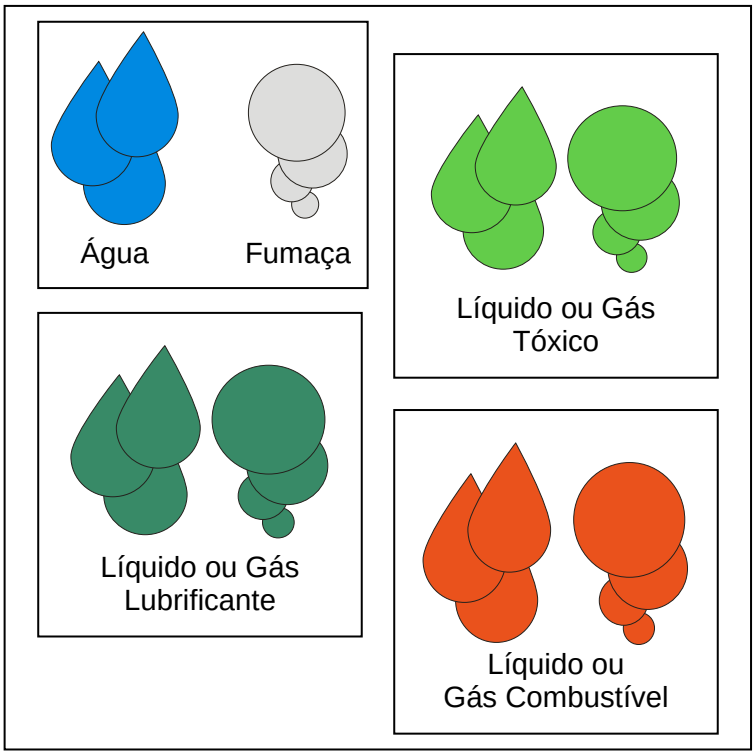


Figura 9 - Símbolos - Líquidos e Gases

Apenas a temperatura é representada através da alteração da cor do objeto, sendo que os outros eventos necessitam o mapeamento de uma textura no objeto ou mesmo a inserção de um objeto 3D no ambiente virtual. Eventos como o vazamento de líquidos ou gases, vistos na Figura 9, se utiliza de símbolos que representem o evento, como gotas representando líquidos e círculos arranjados de forma a representar gases. Para representar vazamentos de água, a cor azul clara, padrão nas cartas geográficas, foi escolhida. Da mesma forma, foi estabelecido que a cor mais indicativa para fumaça seja a cinza.

A diferenciação entre as diferentes classes de líquidos e gases pode ser feita pela alteração da cor do objeto, sendo que para a representação de líquidos ou gases tóxicos estabeleceu-se como padrão a cor verde clara, comumente utilizada para tal representação. Lubrificantes devem ser representados pela cor verde escura. Por fim, combustíveis utilizarão a cor vermelha, uma cor de temperatura quente, ideal para representação de gases ou líquidos inflamáveis. Além disso, a cor azul clara pode ser utilizada para a representação de gases inofensivos, como o oxigênio. Diferentes outras classes de líquidos e/ou gases podem ser especificadas e cores indicativas de suas características podem ser especificadas para sua representação.

Para indicação do tipo de líquido ou gás, pode-se utilizar uma letra dentro do símbolo. Por exemplo, o vazamento de diferentes líquidos inflamáveis (representados por gotas de cor

vermelha) pode ter as letras Q (querosene), G (gasolina), A (álcool) ou D (diesel) no seu símbolo, o que aumenta a precisão do monitoramento.

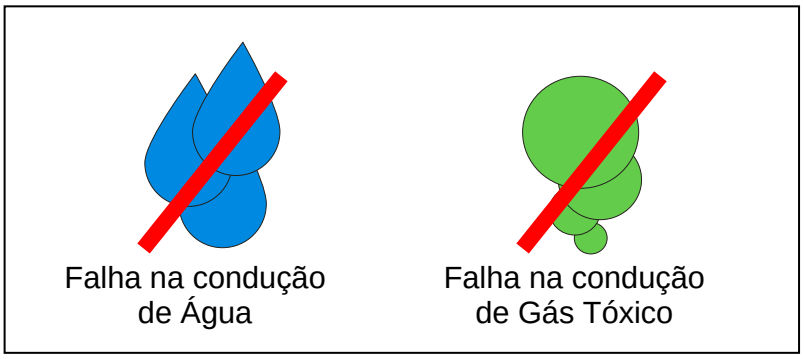


Figura 10 - Símbolos - Falha no condutor

Como pode ser observado na Figura 10, no caso de falha no condutor de qualquer um destes líquidos ou gases, será adicionado um corte vermelho sobre o símbolo já especificado para aquele evento, indicando a negação do evento.

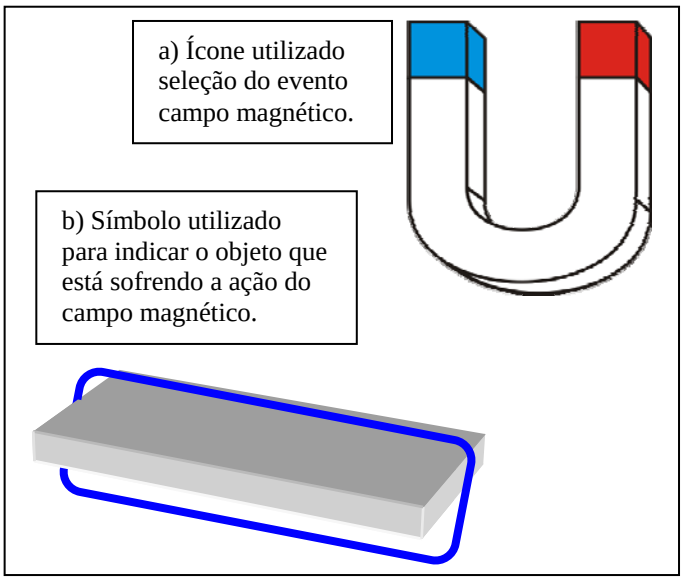


Figura 11 - Símbolo - Campo Magnético

Para os eventos já apresentados, como líquidos e gases, a mesma forma geométrica pode ser usada tanto para representar o evento no AVC quanto na barra de seleção de eventos a serem monitorados. Para a representação de eventos do tipo campo magnético, apresentado na Figura 11, pode ser utilizado um ímã em forma de ferradura na barra de seleção de eventos e uma linha azul em torno dos objetos que estiverem sofrendo a ação do campo magnético. O critério utilizado para essa escolha foi a própria característica do imã de gerar campo magnético. A forma de ferradura é bem simbólica para sua representação e comumente faz parte do imaginário das pessoas. O próprio imã utilizado na barra de seleção de eventos poderia ser utilizado no AVC, porém, foi definido que uma linha azul (que é uma cor que não

chama muita a atenção, mas que corresponde à representação imagética de um campo magnético) permite uma melhor representação da ação do campo magnético; mais uma vez o imaginário imagético foi utilizado.

Figura 12 - Escala de Cores - Pressão

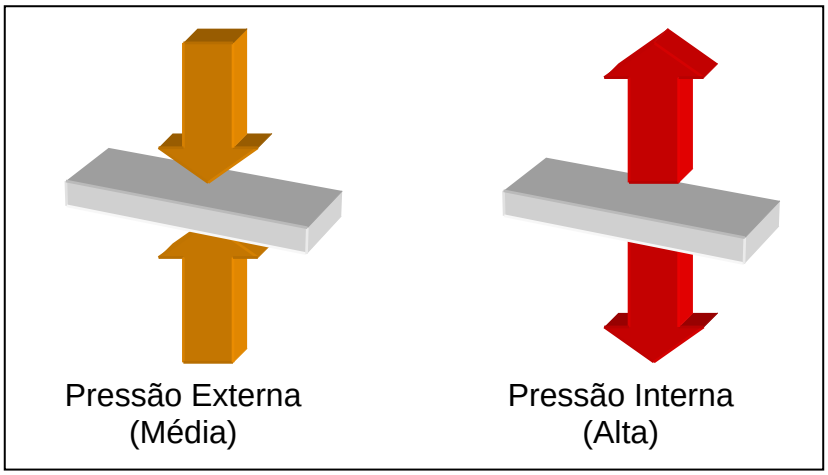


Figura 13 - Símbolos - Pressão

Assim como para a representação de diferentes intensidades de campos magnéticos pode-se utilizar diferentes tons de azul, para a representação de diferentes níveis de pressão pode-se utilizar uma escala de cores como a da Figura 12. Além desta escala cromática, uma forma geométrica representativa deve ser utilizada, como a apresentada na Figura 13. O sentido das setas indicativas de pressão é que determinarão se a pressão é interna ou externa. Setas apontando para o objeto e fazendo movimentos centrípetos indicam pressão externa. Setas apontando para o ambiente e fazendo movimentos centrífugos indicam pressão interna. A localização das setas indica onde está havendo variação de pressão e sua cor o nível desta.

5.3. Tradução para a Linguagem Visual

Uma vez que a linguagem visual (LV) foi definida, será descrito o módulo que efetua a tradução das informações vindas da RASSF para esta LV. Os comandos especificados a seguir foram baseados nos comandos do padrão MPEG-4, os chamados *BIFS-Commands*. Tiveram que ser definidos porque o X3D não especifica comandos para atualização de sua cena. Basicamente existem quatro tipos de comandos:

- Inserção de nó:

```
<insertNode nodeName="">
  <Shape DEF="S1">
    <Appearance>
      <Material DEF="M1" diffuseColor="0 1 0"/>
    </Appearance>
    <Box />
  </Shape>
</insertNode>
```

Figura 14 - Código para Inserção de nó

Para a inserção de um nó deve-se especificar o nome de seu nó-pai em *nodeName*. O valor deste campo também pode ser vazio como na Figura 14, indicando que o nó deve ser um *root-node*, ou seja, deve ser inserido diretamente no nó raiz (*scene*). A descrição contida entre estas *tags* indica a subárvore a ser inserida como filha do nó-pai especificado.

Esse comando permite que qualquer objeto seja inserido na cena, o que em um ambiente de monitoramento possivelmente será utilizado para representar eventos que ocorram. Por exemplo, se for definido na linguagem visual que fogo deve ser representado por um objeto que tenha o formato de chamas em 3D, este objeto deve ser inserido nas posições em que os sensores detectarem fogo.

- Remoção de nó:

```
<removeNode nodeName="S1"/>
```

Figura 15 - Código para Remoção de nó

Apenas um campo pode ser especificado para a remoção de um nó. Este campo, *nodeName*, especifica o nome do nó a ser removido e não pode ser vazio, como observado na Figura 15. Caso esse nó tenha nós filhos, todos serão automaticamente removidos.

Voltando ao exemplo anterior, se os sensores detectam que o fogo acabou, o objeto chama, previamente inserido, necessita ser removido. Isto pode ser feito utilizando-se esse comando.

- Alteração de parâmetro:

```
<changeValue nodeName="M1" nodeField="diffuseColor" nodeValue="1 0 0"/>
```

Figura 16 - Código para Alteração de parâmetro

Muitas vezes há a necessidade de se alterar o valor de um campo de um nó, como posição (*translation*) ou cor (*diffuseColor*). Um código para alteração de parâmetro é mostrado na Figura 16.

Um exemplo simples, presente na linguagem visual já especificada, é alteração da cor de um objeto de acordo com sua temperatura, utilizando uma escala de cores que vai desde o azul (indicando muito frio) até o vermelho (indicando muito quente).

- Substituição de nó:

```
<changeNode nodeName="S1">
  <Shape DEF="S1">
    <Box />
  </Shape>
</changeNode>
```

Figura 17 - Código para Substituição de nó

É implementado internamente como a remoção do nó especificado em *nodeName* seguida da inserção do nó contido entre as *tags* no mesmo nó pai daquele removido anteriormente. É normalmente utilizado em situações em que diferentes objetos são usados para representar o mesmo evento, um para cada intensidade do evento. Um exemplo de código é dado na Figura 17.

De posse desta especificação, o navegador cliente foi estendido para realizar as modificações na cena segundo os comandos recebidos do servidor. Também foi criado o servidor, permitindo que diversos parâmetros destes comandos pudessem ser alterados de acordo com os valores recebidos da rede de sensores. Estes valores podem ser o nome do nó (de acordo com o nome do sensor) ou o valor do campo cor (*diffuseColor*, de acordo com o valor da temperatura), por exemplo. O módulo responsável por efetuar a integração das informações vindas da rede de sensores com os comandos definidos no DLV é o módulo de tradução.

5.4. Considerações Finais

Este capítulo descreveu uma visão geral do projeto de um sistema de gravação/reprodução de AV3D e os módulos que o constituem. Foi aqui discutida também a importância da imagem e de como uma linguagem visual pode facilitar a compreensão, de forma mais rápida e precisa, do que está ocorrendo em um ambiente físico sendo monitorado. Características de composição imagética como o equilíbrio, o contraste, as formas geométricas, as cores, e a similaridade, foram utilizadas na concepção da linguagem visual aqui descrita. Finalmente, foram descritos os comandos responsáveis pela tradução das informações vindas da RASSF para a Linguagem Visual concebida.

No próximo capítulo serão discutidos os algoritmos de gravação e reprodução existentes, sendo detalhado e avaliado o método otimizado desenvolvido neste trabalho.

6. Implementação e Avaliação do Mecanismo de Gravação e Reprodução de AV3D

Diversos sistemas já existentes para gravação e reprodução de mídia 3D foram discutidos na Capítulo 2, sendo que nenhum deles se mostrou viável para a gravação de mídia para aplicações de monitoramento de ambientes de segurança crítica. Naquele capítulo também foi visto o projeto da aplicação de preparação para emergências do *LRVNet* e em qual parte desta aplicação este trabalho se encontra. Uma vez estudada a linguagem visual e o sistema de visualização em tempo-real no Capítulo 5, este capítulo descreverá o sistema de gravação e reprodução desenvolvido neste trabalho.

6.1. Gravação de Mídia 3D

A gravação de mídia 3D tem suas dificuldades, como já discutido, principalmente no que se refere aos seus estados. Também foi visto que o estado de uma cena 3D pode ser resumido como sendo o grafo daquela cena em um dado instante de tempo.

Gravar o estado da cena é importante para que a posterior reprodução possa acessar aleatoriamente as gravações. Algumas ferramentas para gravação de mídia 3D existentes não gravam estes estados, o que inviabiliza o acesso aleatório. Nas subseções seguintes serão descritas técnicas de gravação de mídia 3D, as primeiras adaptadas de técnicas já existentes na literatura, destacando suas vantagens e desvantagens. Por fim será apresentada a técnica otimizada que reúne as principais vantagens das outras técnicas.

6.1.1. Técnica *Event-Replay*

Esta técnica é muitas vezes chamada de *event-replay* por apenas reapresentar os eventos da mesma forma que ocorreram. Embora seja a mais simples de ser compreendida e implementada, esta técnica não permite acesso aleatório. Por isso, bastante tempo e processamento podem ser necessários antes que uma reprodução seja iniciada.

O *event-replay* consiste em gravar os eventos à medida que acontecem juntamente com uma marca de tempo (*timestamp*), sem se preocupar com o estado da mídia. Esse arquivo, uma vez gravado, se assemelha a um *log* de eventos. A posterior reprodução é feita de acordo com as marcas de tempo e exige que o usuário inicie a visualização desde o início da gravação, para que todos os eventos anteriores à posição desejada sejam executados.

Na Figura 18 é apresentada a linha de tempo para uma gravação em que ocorreram 20 eventos. Cada evento é representado por um traço vertical. Tanto na visualização em tempo-real quanto na posterior reprodução é necessário que o estado inicial da cena seja recuperado. O chamado estado inicial da cena normalmente contém todas as geometrias básicas para a criação da cena, sem a representação de nenhum evento. No exemplo de um monitoramento industrial, o estado inicial da cena contém todas as pessoas, paredes, pisos, máquinas e demais objetos necessários para a simulação do ambiente real, todos em suas formas e cores "neutras", sem nenhuma informação do estado do ambiente.

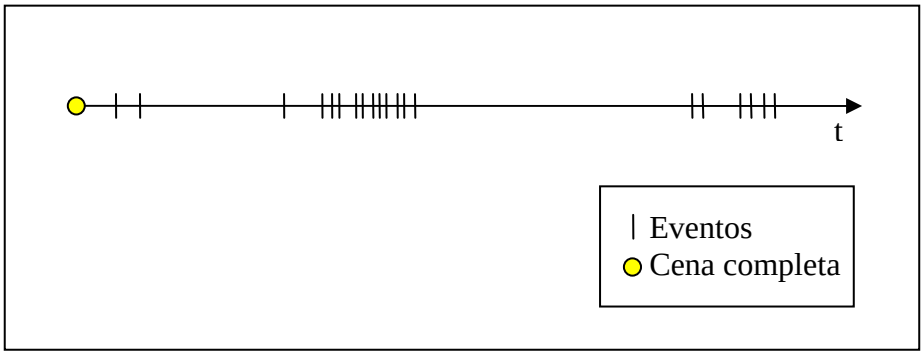


Figura 18 - Linha do tempo: *event-replay*

Após o carregamento do estado inicial da cena no acesso em tempo-real, os eventos são apresentados no navegador à medida que acontecem. Já no acesso posterior, as marcas de tempo são usadas para se calcular o tempo entre a execução de dois eventos.

O acesso aleatório não é possível nesta técnica, pois não se pode recuperar o estado do grafo da cena no ponto desejado. A execução a partir do ponto desejado da gravação também não é viável, uma vez que as informações se tornam inconsistentes. Se, por exemplo, o primeiro evento de uma gravação indica uma alteração de temperatura e é solicitada a reprodução após este momento, o navegador necessita carregar a cena inicial e executar este comando, de forma a reconstruir o estado no momento desejado.

É possível simular acesso aleatório para o usuário, executando todos os eventos anteriores à posição desejada de uma vez. Assim, embora o acesso seja seqüencial, o usuário tem a impressão de que o acesso foi direto ao ponto desejado, uma vez que os eventos anteriores ao ponto desejado são processados rapidamente. Porém, caso exista um grande número de eventos neste intervalo, será necessária uma grande quantidade de processamento, causando longos atrasos no início da reprodução.

Sendo assim, para que se possa realizar acesso verdadeiramente aleatório e com isso ter um bom desempenho no início dessas reproduções, é necessário que se grave o estado

atual da cena periodicamente. Esta técnica foi chamada de gravação de cenas completas e é explicada em detalhes na próxima seção.

6.1.2. Gravação de Cenas Completas

Nesta técnica, assim como na *event-replay*, os eventos são gravados à medida que acontecem. Todavia, de tempos em tempos o estado atual da cena também é armazenado. Gravando-se o estado atual da cena, esta descrição armazenada conterá toda informação necessária para reconstruir a cena, desde os objetos estáticos como paredes e portas até as informações visuais dos eventos, como alterações de cor, forma e presença de novos objetos. Conforme a Figura 19.

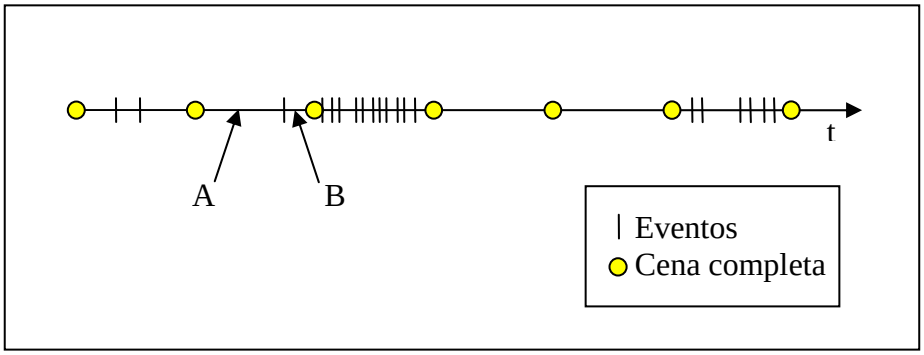


Figura 19 - Linha do tempo: Gravação de cenas completas

Uma vez gravado periodicamente o estado atual da cena, o acesso aleatório durante a reprodução é conseguido carregando-se a cena completa imediatamente anterior ao ponto desejado. Algumas vezes, entre a cena completa carregada e o ponto desejado não haverá nenhum evento (caso A). Nestes casos, somente o carregamento da cena completa é suficiente para reconstruir o estado da cena no ponto desejado. Caso haja algum evento neste intervalo (caso B), será necessário executar os eventos da mesma forma que na técnica *event-replay*, reproduzindo seqüencialmente todos os eventos, do início (cena completa imediatamente anterior ao ponto desejado) até o ponto requisitado (neste caso, ponto B), em tempo-real. A grande quantidade de processamento e os longos tempos de espera são superados, desde que um número suficiente de cenas completas seja gravado.

Surgem algumas questões: Não seria possível economizar espaço de armazenamento, uma vez que as cenas completas são normalmente grandes? Quanto tempo esperar entre uma gravação e outra? A Seção 6.1.3 responde à primeira pergunta, gravando apenas os objetos que foram modificados na cena. A outra pergunta é respondida na Seção 6.1.4, onde um método otimizado de gravação é proposto como parte deste trabalho.

6.1.3. Gravação de Cenas Parciais

Como foi visto na Seção 4.6, para diminuir a quantidade de informação necessária para representar uma animação (seqüência de eventos), a compressão de vídeo bidimensional tenta eliminar a redundância temporal. Similarmente, o objetivo de se gravar cenas parciais é não gravar informações já contidas em gravações anteriores.

Existem duas formas principais de se gravar periodicamente cenas parciais: A primeira delas é gravando a diferença entre o estado da cena inicial e o estado da cena atual. Sendo assim, em uma posterior reprodução necessita-se apenas carregar a cena inicial seguido da cena parcial, uma vez que esta última contém toda informação de alterações que ocorreram desde o início. A segunda forma é gravar as diferenças entre o estado em que a cena se encontrava quando a última gravação de cena parcial ocorreu e o estado da cena atual. Nesta abordagem são gravadas as diferenças entre cenas parciais. O problema desta técnica é semelhante ao da técnica *event-replay*: para acessar aleatoriamente uma cena é necessário recuperar seqüencialmente todas as cenas parciais.

Analisando-se a primeira abordagem, em que toda a diferença entre a cena inicial e a cena atual é gravada, pode-se concluir que haverá economia na quantidade de espaço necessário para armazenar as gravações, uma vez que objetos estáticos como paredes e mesmo objetos dinâmicos que não sofreram alteração desde o início da gravação não serão armazenados.

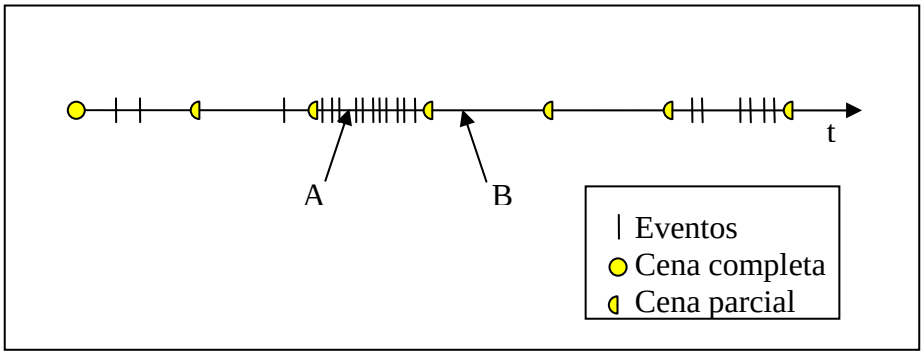


Figura 20 - Linha do tempo: Gravação de cenas parciais

Através da Figura 20 pode ser observada a existência dos estados parciais. Como discutido anteriormente, estes eventos parciais contêm a diferença entre o estado atual da cena e o estado inicial desta. De acordo com esta especificação, para reproduzir uma gravação é necessário recuperar o estado inicial da cena (cena completa), atualizá-la de acordo com a cena parcial imediatamente anterior ao ponto desejado (suficiente para reprodução a partir do

ponto B) e, caso existam, executar os eventos entre a cena parcial e o ponto desejado (3 eventos no caso A).

O ponto negativo desta abordagem é observado quando um sensor detecta alguma alteração, permanecendo neste novo estado por muito tempo. Como a gravação de cenas parciais é cumulativa, ou seja, grava todas as alterações desde a cena inicial (cena completa), esta alteração estará presente em todas as gravações de cenas parciais, embora a alteração tenha ocorrido uma única vez.

A próxima seção apresenta uma solução que elimina a redundância que surge na gravação de cenas parciais e responde à pergunta sobre quanto tempo esperar entre uma gravação e outra.

6.1.4. Gravação Utilizando o Método Otimizado

O método otimizado, proposto neste trabalho, utiliza um *log* dos eventos que ocorreram, gravação de cenas completas e gravação de cenas parciais, buscando aproveitar as melhores vantagens de cada uma destas abordagens.

Gravar apenas cenas completas exige muito espaço de armazenamento. Gravar apenas cenas parciais faz com que, a partir da ocorrência de um evento, este esteja presente em todas as gravações de cenas parciais; isto desperdiça espaço de armazenamento. Se utilizadas as gravações de cenas completas e a de cenas parciais em conjunto, a gravação de cenas completas pode ser feita menos vezes que a de parciais, porque esta última é de tamanho reduzido. A gravação de cenas parciais mais freqüentemente melhora o desempenho no início de uma reprodução aleatória. Também, a redundância existente nas cenas parciais, que faria o tamanho destas crescer com o passar do tempo, é eliminada a cada gravação de cena completa realizada.

O tempo entre uma gravação e outra é relativo, uma vez que algumas variáveis se fazem presentes. O tamanho da cena e dos comandos que realizarão as alterações influenciam diretamente na quantidade de processamento gasto para processá-los e na latência de transmissão. Definir que uma cena completa seja gravada em intervalos de *t* segundos não é uma boa opção, visto que neste intervalo podem acontecer muitos eventos quando o ambiente se encontra em uma situação emergencial. Esta quantidade elevada de eventos causa o mesmo problema que a técnica *event-replay*, onde ocorrem grandes atrasos no início de uma reprodução aleatória. O outro extremo também é possível, onde nenhum evento ocorre em um destes intervalos. Se isto ocorrer, duas cenas exatamente iguais serão gravadas, o que não é

justificável. citadas (tamanho da cena e dos comandos) influenciem, o desempenho do sistema se torna superior quando se aguarda um número determinado de eventos ao invés de um tempo especificado. Respondendo à pergunta proposta, na verdade o melhor não é esperar um intervalo de tempo, mas um determinado número de eventos. Como ilustra a Figura 21, várias gravações acontecem em momentos de pico, ao passo que em situações estáveis nenhuma gravação é realizada.

Observando-se a Figura 21 pode-se notar a gravação de cenas parciais a cada 3 eventos, sendo que cenas completas são gravadas a cada 3 cenas parciais, ou seja, a cada 9 eventos. Este número reduzido de eventos permite uma melhor compreensão da figura, sendo que na prática a gravação de cenas parciais a cada 50 eventos e de cenas completas a cada 10 parciais se mostrou suficiente. Estes valores foram obtidos nos testes realizados e, observando-se a Figura 29, notou-se que com estes valores o início da reprodução teria uma latência inferior a 500ms.

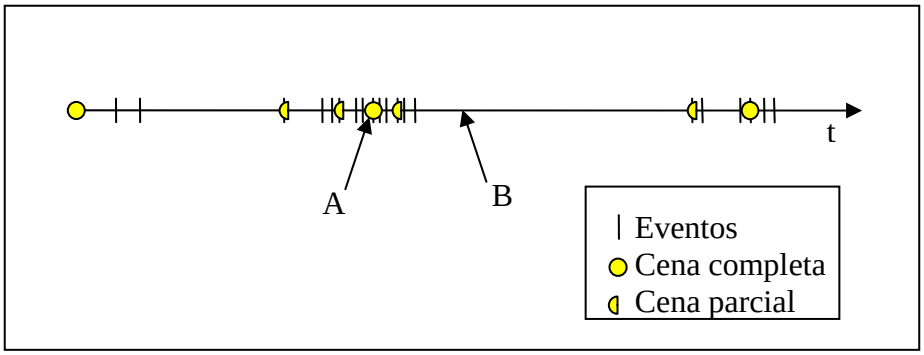


Figura 21 - Linha do tempo: Gravação utilizando o método otimizado

O acesso ao ponto A é facilmente realizado da mesma forma que na abordagem que utiliza apenas cenas completas, ou seja, basta apenas carregar a cena completa correspondente àquele ponto. O acesso ao ponto B, por outro lado, é o pior caso. Nele é necessário o carregamento da cena completa imediatamente anterior (neste caso, a mesma do ponto A), a cena parcial também imediatamente anterior e ainda executar os dois outros eventos existentes entre a cena parcial e o ponto desejado.

6.2. Armazenamento

Para o protótipo desenvolvido foi usado um banco de dados *MySQL* para armazenamento das gravações, cujo modelo Entidade Relacionamento (ER) é apresentado na Figura 22.

A tabela de comandos armazena o *log* dos eventos recebidos da rede de sensores, na forma de comandos que podem ser executados pelo navegador. Nesta tabela existe o campo *cmMomento*, responsável por armazenar o instante em que o evento ocorreu. As outras duas tabelas não possuem um campo determinando o momento em que as cenas foram gravadas, mas sim qual a ID do último comando contido nesta cena.

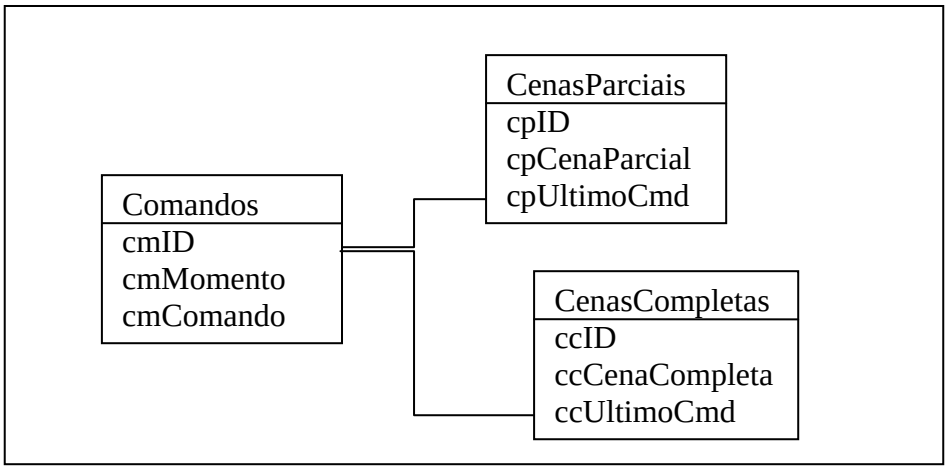


Figura 22 - Modelo ER do banco de dados para armazenamento das gravações

Um formato de arquivo próprio pode ser utilizado em lugar do banco de dados para se conseguir um melhor desempenho. Este formato deve possuir características que permitam navegar eficientemente comando a comando (na hora da reprodução) e encontrar rapidamente a cena completa, a cena parcial e os comandos necessários para reconstrução do estado da cena em um momento solicitado. Ponteiros indicando o próximo evento e índices são opções a serem consideradas. Na próxima seção são discutidos os principais detalhes da implementação dos módulos do sistema.

6.3. Implementação

Como dito anteriormente, foi escolhido o *toolkit* Xj3D como base para o desenvolvimento do sistema proposto. Um navegador que suporta cenas no formato VRML e X3D está disponível neste conjunto de ferramentas, que também disponibiliza APIs em linguagem Java para acesso à cena. O anexo B descreve a especificação UML do sistema implementado.

Na Seção 6.3.1 serão discutidas as adaptações realizadas no navegador que será utilizado pelo cliente.

6.3.1. Lado Cliente

Embora muitas funções estejam disponíveis para acesso à cena através das APIs do Xj3D, algumas de grande importância para este trabalho não existem. O acesso à cena é possível através de funções que permitem a seleção dos nós raízes (*getRootNodes*) ou a seleção de um nó nomeado (*getNamedNode*). Os campos de um nó podem ser acessados (*getField*), assim como seus nós filhos (contidos em um campo, normalmente denominado *children*, cujo tipo é *SFNODE* ou *MFNODE*), em caso de nós de agrupamento. Apesar destas funções existirem, não há nenhuma forma disponível para alterar a cena através de comandos, o que levou à necessidade de se especificar alguns comandos, descritos na Seção 5.3. Para que estes comandos pudessem ser executados foi criada uma classe denominada *MSScene* (cena do sistema de monitoramento) que além de ser capaz de interpretar esses comandos, também possibilita a recuperação dos nós em formato X3D, outra função ausente nas APIs do Xj3D. Para esta última função, todo o grafo da cena necessita ser percorrido, iniciando pelos nós raízes e recursivamente analisando seus nós filhos.

Uma vez que estas funções-base foram concluídas, desenvolveu-se a parte cliente para comunicação com o servidor, utilizando soquetes.

Alguns botões também foram adicionados na interface do navegador, para permitir maior controle por parte do usuário. A interface com todas as modificações realizadas é mostrada na Figura 23.

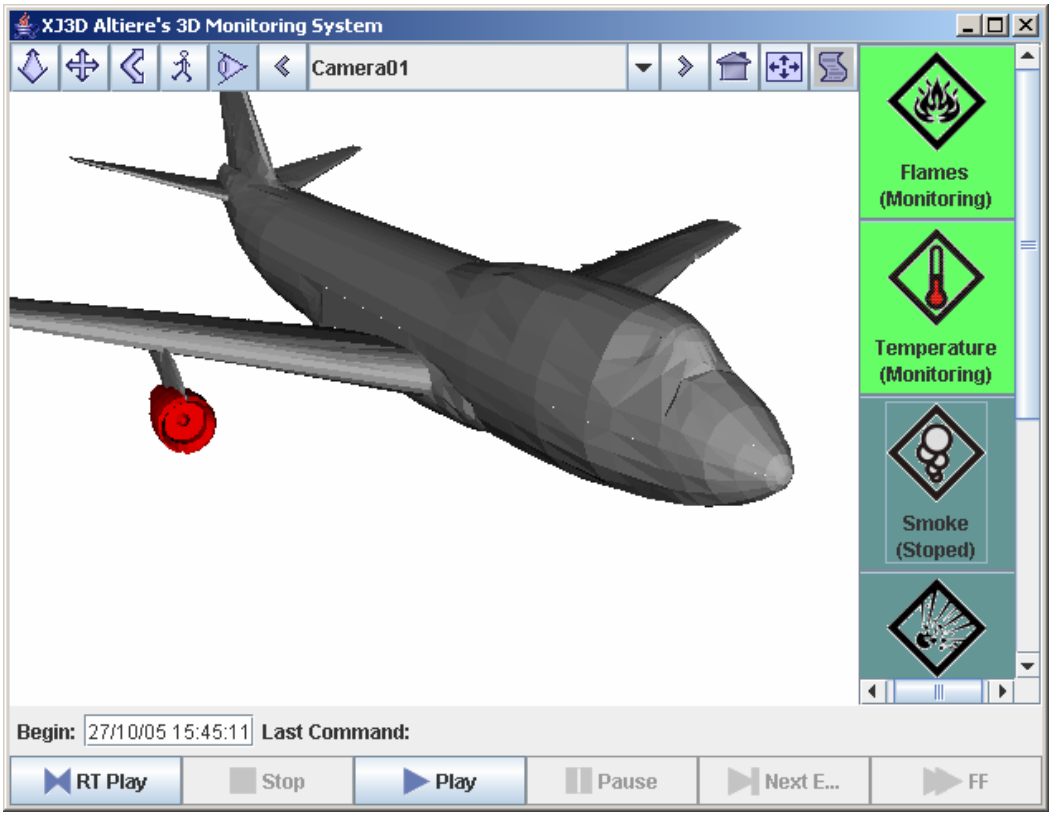


Figura 23 - Interface do navegador cliente

Podem ser observadas nesta figura duas barras não existentes na interface padrão, uma se encontra na parte inferior e a outra na lateral direita.

A barra inferior, vista em maiores detalhes na Figura 24, permite ao usuário selecionar se deseja visualizar o ambiente em tempo-real (*RT Play*), para analisar a situação em que o mesmo se encontra atualmente, ou se ele deseja visualizar alguma cena previamente gravada. Para acesso às gravações, é necessário especificar a data e a hora desejada antes de iniciar a reprodução (*Play*). Uma vez reproduzindo, é possível pausar (*Pause*), fazer um avanço evento-a-evento (*Next Event*) ou avançar rapidamente (*FF, Fast Forward*).

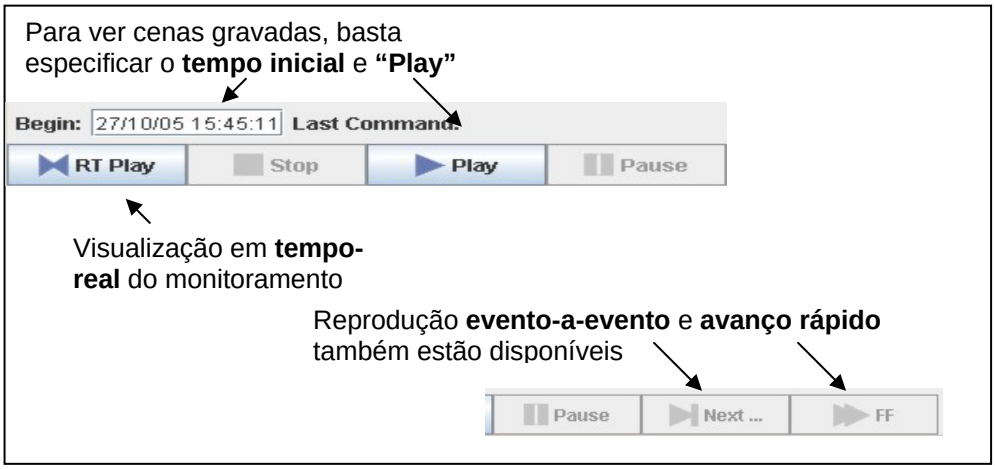


Figura 24 - Interface avançada do navegador

A barra lateral exibe botões que indicam os diferentes tipos de monitoramento que podem estar sendo realizados. Através destes é possível selecionar quais eventos serão monitorados (verde claro) e quais serão ignorados (verde escuro), com o objetivo de despoluir visualmente o ambiente.

Todo o controle realizado pelo navegador cliente através da barra inferior se resume ao envio de solicitações ao servidor. Desta forma, um clique no botão pausar faz com que o navegador cliente solicite a pausa para o servidor que, por sua vez, interrompe o envio dos comandos de atualização. Um posterior clique no botão reproduzir envia outra solicitação ao servidor, que retoma a execução de onde havia parado. As opções de pausar, ir para o próximo evento e avanço rápido estão disponíveis apenas na reprodução posterior. Um protocolo foi especificado para esta comunicação cliente/servidor, tratando destas solicitações e do envio de comandos de atualização de cena.

Por outro lado, os botões da barra lateral realizam suas funções localmente, procurando no grafo da cena pelos objetos ou alterações que sejam de sua competência e, de acordo com a ação de liberar ou pressionar o botão, suprime ou re-exibe, respectivamente, tais alterações.

6.3.2. Lado Servidor

Como já visto na Seção 3, o servidor pode ser dividido em três módulos principais, o de tradução, o de gravação e o de reprodução, sendo que estes dois últimos utilizam o sistema de armazenamento descrito na seção 6.2. A seguir será analisado cada um deles.

6.3.2.1. Módulo de Tradução

Uma vez especificada a forma de se realizar a tradução para a linguagem visual, como mostrado na Seção 5.3, será discutido como é realizada a tradução de uma informação vinda da RASSF para um comando que é enviado ao cliente, através da linguagem visual. Será utilizado o exemplo da alteração de cor do objeto de acordo com a temperatura para explicar como isto ocorre. Neste exemplo serão considerados dois sensores, chamados Temperatura_Caixa e Temperatura_Esfera, cada um responsável por um objeto, respectivamente, pelo objeto Caixa e pelo objeto Esfera. Cada pacote contendo informações da rede de sensores contém os seguintes campos: Nome, Tipo e Valor. Na Tabela 2 pode ser visto o comando especificado pela LV para sensores que monitoram temperatura, uma entrada do DLV.

Tabela 2 - Comando da LV para sensores do tipo temperatura, DLV

Tipo de Sensor	**Comando da LV**
...	...
Temperatura	<changeValue nodeName="%Nome%" nodeField="diffuseColor" nodeValue="%Valor%"/>
...	...

Parâmetros são indicados pelo nome do campo a que correspondem entre dois sinais de porcentagem (%xxxxx%). A Tabela 3 mostra os valores coletados pelos sensores em três momentos.

Tabela 3 - Exemplos de leituras realizadas pelos sensores

Evento	**1**	**2**	**3**
Nome	Temperatura_Caixa	Temperatura_Esfera	Temperatura_Caixa
Tipo	Temperatura	Temperatura	Temperatura
Valor	10	50	30

A tradução de eventos para a linguagem visual é realizada buscando-se diretamente no DLV (Tabela 2) pela linha que corresponde ao Tipo do sensor que realizou a leitura, através de uma consulta SQL à base de dados. No exemplo, as três leituras são do tipo Temperatura, o que fará o tradutor utilizar a única entrada mostrada na Tabela 2, substituindo os parâmetros pelos correspondentes valores de campo, utilizando-se funções de manipulação de *strings* do Java.

O campo Nome do sensor pode ser usado diretamente no parâmetro que indica o nome do nó que deve ser alterado, desde que os nomes dos nós da cena inicial tenham os mesmos nomes dos sensores. Caso não seja possível realizar este mapeamento direto, é necessária uma tabela de dois campos, Nome do Sensor e Nome do Objeto, para fazer a correspondência.

Para o mapeamento do Valor sentido pelo sensor em uma cor correspondente, pode-se utilizar a seguinte fórmula:

R = Valor * 2,55

G = 0

B = 255 - R

Onde o R, o G e o B representam as intensidades relativas de vermelho (Red), verde (Green) e azul (Blue), respectivamente, que formam uma determinada cor. O valor de cada componente (R, G e B) deve ser um número inteiro entre 0 e 255. Por exemplo, o valor RGB “255 0 0” representa a cor vermelha. Por motivo de simplificação, foi assumido nas formulas

valores de temperatura entre 0 e 100 graus e cores variando do azul ao vermelho, de acordo com a Figura 25.

Figura 25 - Paleta para mapeamento de temperatura em cor

Na Tabela 4 são mostrados os três comandos da Tabela 3 após a tradução.

Tabela 4 - Comandos após tradução

Evento	Comando
1	<changeValue nodeName="Temperatura_Caixa" nodeField="diffuseColor" nodeValue="25 0 230"/>
2	<changeValue nodeName="Temperatura_Esfera" nodeField="diffuseColor" nodeValue="127 0 128"/>
3	<changeValue nodeName="Temperatura_Caixa" nodeField="diffuseColor" nodeValue="76 0 179"/>

Este módulo de tradução é utilizado para a visualização dos clientes em tempo-real. É também necessário para a realização das gravações, uma vez que os comandos são gravados após a tradução.

6.3.2.2. Módulo de Gravação

Para gravação é utilizado o esquema otimizado proposto na Seção 6.2. Foi escolhida a gravação pós-tradução, uma vez que é necessária a gravação dos estados da cena. Se optado pela gravação antes do processamento, não seria possível recuperar o estado da cena, uma vez que a cena ainda não existiria.

Após a tradução, o próprio *toolkit* Xj3D é utilizado para processar os comandos, como se fosse um cliente, para que o estado da cena seja mantido. Utilizando-se as funções desenvolvidas (Seção 6.3.1), pode-se recuperar o estado da cena e gravá-lo nos momentos determinados pelo método otimizado.

Embora a gravação das cenas completas seja relativamente simples, a gravação das cenas parciais tem alguns fatores que a tornam mais custosa. Enquanto a cena completa é a descrição de toda a cena, a cena parcial envolve apenas as modificações realizadas desde a gravação da última cena completa. Então, como saber o que foi modificado? Duas opções foram levadas em conta.

A primeira opção seria marcar, no grafo da cena, aqueles nós que fossem alterados durante a execução dos comandos. Todos os nós seriam desmarcados no momento da gravação de uma cena completa. Ao se gravar uma cena parcial todo o grafo seria percorrido, sendo gravadas apenas as informações necessárias para reconstruir o estado atual a partir da última cena completa gravada. A implementação desta solução mostrou-se inviável, pois necessitaria que um campo fosse adicionado nos nós do grafo para marcação dos nós alterados. Também seu desempenho seria comprometido quando cenas grandes fossem utilizadas, uma vez que é necessário percorrer todo o grafo da cena a cada gravação de cena parcial.

Como alternativa, uma classe para gerenciamento das alterações foi desenvolvida, levando em consideração certas peculiaridades. A seguir são listadas algumas de suas características:

- Possui quatro vetores, para armazenar informações sobre os nós inseridos, removidos e alterados e para os valores alterados.
- A cada gravação de cena completa, todos os vetores são zerados.
- No momento de uma inserção, todas as entradas nos vetores, relativas ao objeto inserido, são zeradas e, então, é adicionada uma entrada no vetor de nós inseridos. É feito assim porque a inserção de um nó, mesmo que ele já exista, substitui todo aquele galho do grafo da cena.
    - Se o objeto havia sido removido, não é necessário removê-lo para depois fazer a inserção, uma vez que só a inserção já substitui a versão antiga, anterior à remoção.
    - Se havia sido inserido anteriormente, a nova inserção substitui a versão anterior.
    - Se havia sido substituído, idem à inserção.
    - Se algum dos valores de seus campos havia sido alterado, a inserção substitui todos eles.

  Desta forma, qualquer alteração prévia de um nó pode ser desconsiderada.
- Ao remover um nó, todas as entradas nos vetores, relativas ao objeto inserido, são zeradas, uma vez que não é necessário inseri-lo ou alterá-lo para depois removê-lo.

Uma entrada é adicionada no vetor de objetos removidos somente se este objeto existia na última cena completa gravada, caso contrario não haverá o que remover.

- A alteração de um nó é similar à inserção.
- Se o valor de um campo é alterado e ele ainda não foi alterado, uma entrada é inserida no vetor de valores alterados. Caso já tenha sido alterado, a entrada no vetor é substituída.

Desse modo os vetores conterão todas as informações do que foi alterado na cena. Estas informações serão gravadas para que possam ser posteriormente, na reprodução, utilizadas para reconstruir o estado da cena.

Os comandos, como já dito, são gravados da mesma forma que chegam. Para que fosse possível visualizar a reprodução em sentido inverso (*rewind*) seria necessário gravar os chamados contra-eventos, como acontece no caso do Massive-3 [BEN 02]. Os contra-eventos são eventos que desfazem seu respectivo evento, ou seja, o contra-evento de:

- inserir de um nó é removê-lo.
- remover um nó é inseri-lo.
- alterar um nó é voltar sua cópia antiga.
- alterar o valor de um campo é voltar seu valor antigo.

Como não foi observada necessidade imediata desta função, ela não foi implementada e será proposta como trabalho futuro.

6.3.2.3. Módulo de Reprodução

Uma vez gravadas as cenas completas e parciais e os eventos, o navegador do cliente pode solicitar a reprodução desta gravação.

Para se determinar qual cena completa, qual cena parcial e quais comandos necessitam ser enviados ao navegador, a seguinte lista de passos foi desenvolvida:

- Selecionar o evento de maior ID, cujo momento de gravação (*cmMomento*) seja inferior ao momento solicitado. Este valor é armazenado em *maxID*.
- Enviar ao navegador do cliente que realizou a solicitação a última cena completa anterior ao momento solicitado, ou seja, que tenha maior ID e *ccUltimoCmd* seja menor que a variável *maxID*.

- Da mesma forma que no passo 2 seleciona-se a cena parcial. Todavia, esta só é enviada ao cliente caso:

  CenasParciais.ccUltimoCmd > CenasCompletas.ccUltimoCmd

  Uma vez que podem não existir cenas parciais entre a cena completa e o ponto solicitado.

- Por fim, enviam-se todos os comandos entre a cena parcial enviada (ou completa, caso não tenha sido enviada a cena parcial no item 3) e o ponto solicitado.

Após estes passos, a cena no navegador do cliente é reconstruída exatamente como se encontrava no momento solicitado. Uma vez restabelecido o estado da cena, os comandos continuam sendo enviados em intervalos de tempo estabelecidos pela seguinte fórmula:

Intervalo = ProximoComando.cmMomento – ComandoAtual.cmMomento

O que permitirá a execução em intervalos de tempo iguais aos que realmente existiram. Para reprodução evento-a-evento, esta fórmula é ignorada, sendo executado um evento a cada solicitação de novo evento (feito pelo navegador). O avanço rápido é conseguido dividindo-se o valor obtido pela fórmula anterior por algum número maior que um, nos testes o valor 5 foi utilizado. Embora não implementado, a reprodução em "câmera lenta" (*slow motion*) pode ser obtida da mesma forma, bastando aumentar o valor do intervalo ao invés de diminuir. Na seção a seguir serão abordados os experimentos realizados e os resultados obtidos.

6.4. Experimentos Realizados e Resultados Obtidos

Esta seção descreve os experimentos que foram realizados. O primeiro deles compara a gravação de cenas completas e parciais de acordo com um intervalo de tempo e de acordo com um número de eventos, validando, assim, o que foi proposto no método otimizado da Seção 6.1.4. O segundo experimento avalia o desempenho do sistema de gravação proposto.

6.4.1. Cenas Completas e Cenas Parciais

Para assegurar um bom desempenho na operação de acesso aleatório é necessário que um número suficiente de estados seja gravado. Como este número de gravações é altamente dependente da aplicação e também porque as gravações de estados consomem processamento e tempo consideráveis, deseja-se gravar o menor número possível de cenas completas e parciais, mantendo um bom desempenho na operação de acesso aleatório. Como o

processamento dos eventos se relaciona mais intimamente ao número de eventos ocorridos do que ao tempo decorrido, pode-se notar uma maior eficiência deste, tanto nos casos em que a rede se encontra estável, gerando poucos eventos, quanto naqueles de situação crítica, em que a rede gera um número elevado de eventos.

Como foi visto na Seção 6.1.4, caso seja utilizado um intervalo de tempo fixo entre as gravações, se um número extremamente baixo (ou nulo) de eventos ocorrer, a gravação é feita mesmo assim. Estas gravações conterão praticamente os mesmos dados que as gravações anteriores. Também nas situações em que um número elevado de eventos ocorre, as gravações continuam ocorrendo naquele intervalo previamente estabelecido, o que poderá prejudicar o desempenho da operação de acesso aleatório. A solução Otimizada supera este impacto no desempenho balanceando o número de gravações efetuadas de acordo com o número de eventos que estão ocorrendo. Em situações em que poucos eventos ocorrem, também são efetuadas poucas gravações. Quando um número elevado de eventos acontece, as gravações são efetuadas em intervalos de tempo menores.

Considerando uma situação onde eventos são gerados durante uma hora, uma comparação entre as duas abordagens é apresentada na Figura 26, considerando o número de gravações realizadas. Pode ser visto, a partir desta figura, que as gravações na abordagem que utiliza um intervalo de tempo fixo grava o mesmo número de cenas, independentemente do número de eventos gerados. Por outro lado, a abordagem Otimizada se adapta ao número de eventos ocorridos. São gravadas mais cenas parciais que completas principalmente para economia de espaço de armazenamento, de acordo com a Seção 6.1.4.

A geração deste gráfico se deu da seguinte forma: Cenas Completas com Intervalo Fixo gravadas a cada minuto, o que dá um total de 60 gravações independentemente do número de eventos gerados na rede; da mesma forma, as Cenas Parciais também de Intervalo Fixo são gravadas a cada 5 segundos, num total de 720 gravações; quanto à abordagem otimizada, estabeleceu-se uma gravação de Cena Parcial a cada 10 eventos e de Cena Completa a cada 100 eventos. O número de gravações utilizando a técnica otimizada foi calculado simplesmente através das fórmulas:

Completas = Eventos / 100

Para cenas completas e:

Parciais = Eventos / 10 – Completas

Para cenas parciais, uma vez que é desnecessário gravar uma cena parcial quando uma completa é gravada.

Como mencionado anteriormente, o número a mais de gravações realizadas na abordagem otimizada após determinado limiar é bom, uma vez que isto garante o desempenho, ou tempo de resposta, durante acessos a pontos aleatórios das gravações.

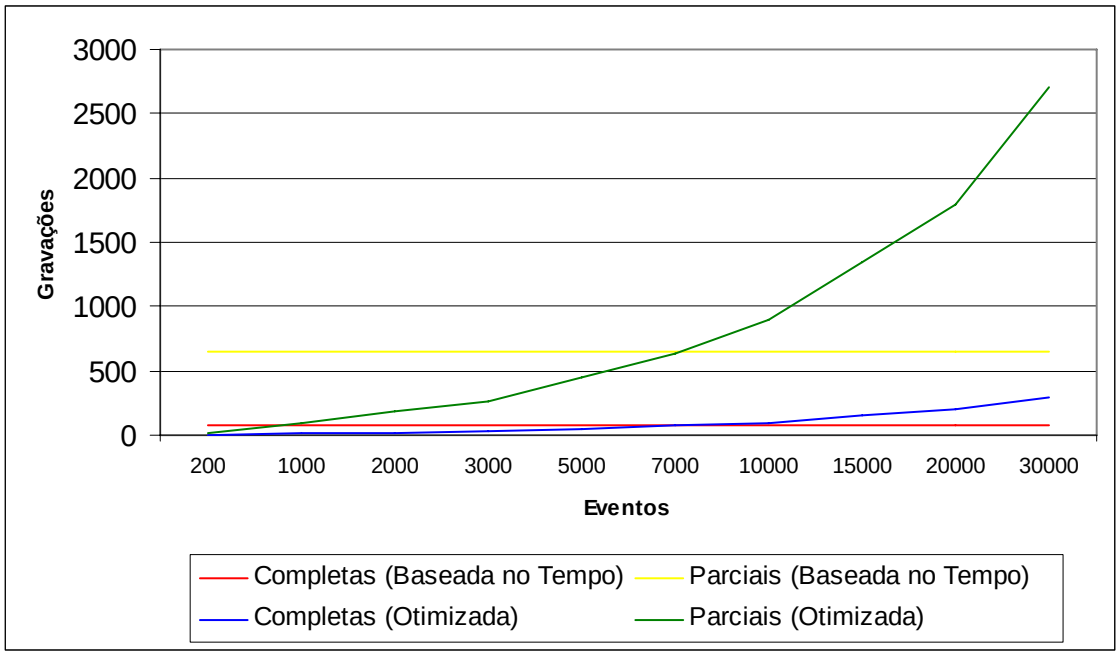


Figura 26 - Número de Gravações (em 1 hora)

Ainda comparando estas duas abordagens, os momentos em que ocorreram gravações de estados podem ser vistos na Figura 27, juntamente à quantidade de eventos gravados. No método que utiliza um intervalo tempo fixo observa-se que a quantidade de gravações baseia-se apenas no tempo, quanto maior a carga da rede, maior o número de eventos contidos na gravação. Por outro lado, pode ser visto que o método otimizado se adaptou à quantidade de eventos que estavam ocorrendo, gravando mais cenas no momento de pico (por volta de 5 à 20) e não realizando gravações quando eventos não ocorreram. O número de eventos gravados por gravação permaneceu constante.

Para geração deste gráfico, um número aleatório de eventos foi gerado (carga da rede), sendo que na técnica baseada no tempo todos os eventos gerados neste intervalo foram gravados a cada 5 segundos. A técnica otimizada, por sua vez, gravou a cena completa a cada três eventos gerados na rede.

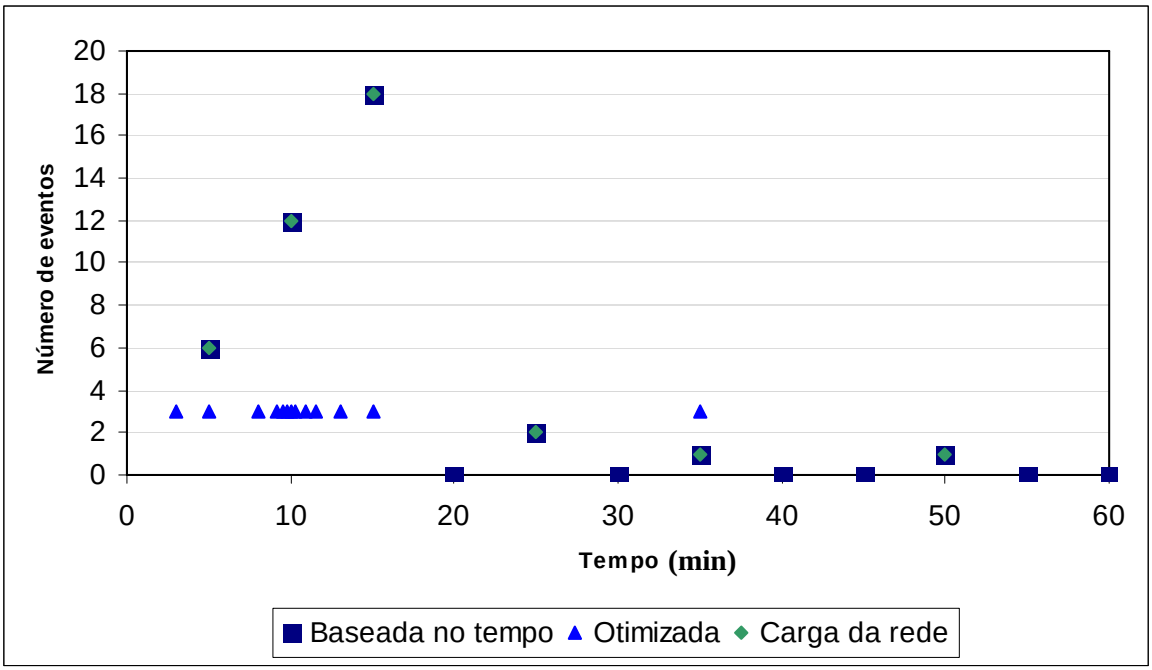


Figura 27 - Eventos por Gravação

6.4.2. Ambiente de Experimentação do Sistema de Gravação

Para realização dos experimentos a seguir foram utilizados dois computadores, um para execução do programa servidor e outro para execução do programa cliente. As configurações de *hardware* destes computadores são apresentadas na Tabela 5.

Tabela 5 - Computadores utilizados para realização dos experimentos

Computador	Processador	Memória	Vídeo	Sistema Operacional
Servidor	Athlon 2600+	512MB	NVidia FX5200	Windows XP
Cliente	Celeron 2.5GHz	512MB	NVidia MX4000	Windows 2000

Para conexão dos dois computadores foi utilizada uma rede local (LAN) de 100Mb e para armazenamento das informações de tradução para a Linguagem Visual e das gravações dos comandos e das cenas foi utilizado o banco de dados MySQL Server 5.0.15. A próxima seção mostra os experimentos realizados e os resultados obtidos, sendo que os gráficos exibem a média dos resultados obtidos em 20 execuções.

6.4.3. Avaliação do Sistema de Gravação

Os experimentos foram divididos em duas etapas, uma para teste do desempenho em tempo-real e outra para avaliação do sistema de gravação.

Os testes em tempo-real, apresentados na Figura 28, utilizaram os objetos mostrados na Tabela 6 para indicar a média de tempo que um comando leva para ser apresentado ao

cliente, desde o momento em que é recebido da rede de sensores. Esta latência pode ser descrita de acordo com a fórmula a seguir:

$$L_{Total} = L_{Tradução} + L_{Transmissão\ pela\ Rede} + L_{Processamento\ no\ navegador}$$

Tabela 6 - Modelos 3D e suas características

Objeto	**Nº Polígonos**	**Tamanho (bytes)**	**Observação**
Objeto 1	0	273	Forma básica
Objeto 2	378	12.194	Indexed Face Set
Objeto 3	512	56.187	Indexed Face Set
Objeto 4	3432	119.853	Indexed Face Set
Objeto 5	6604	189.697	Indexed Face Set

Além dos objetos apresentados, também se mediu a latência para comandos simples, aqueles que apenas alteram uma propriedade do objeto, como a cor, posição ou rotação, por exemplo, indicados no gráfico como “Alteração de parâmetro”.

Tabela 7 - Características dos Modelos 3D do ambiente de monitoramento do avião

Objeto	**Tamanho**
Avião	202.331
Vazamento	6.079
Pressão	32.501
Fumaça	1.075

A Tabela 7 apresenta as características dos modelos 3D utilizados no ambiente de monitoramento do avião apresentado anteriormente. Comparando-se estas duas tabelas pode-se ter uma idéia do tempo gasto para exibição dos objetos em um cenário real.

Pode-se observar, ainda com base na Figura 28, que a alteração de parâmetros tem um custo muito inferior à inserção de um objeto na cena. Também pode ser observada a influência direta que a complexidade do objeto tem sobre a latência, por exemplo, um objeto que contém por volta de 378 polígonos demora cerca de 108,85ms, o que permite a inserção de 9 objetos deste na cena a cada segundo.

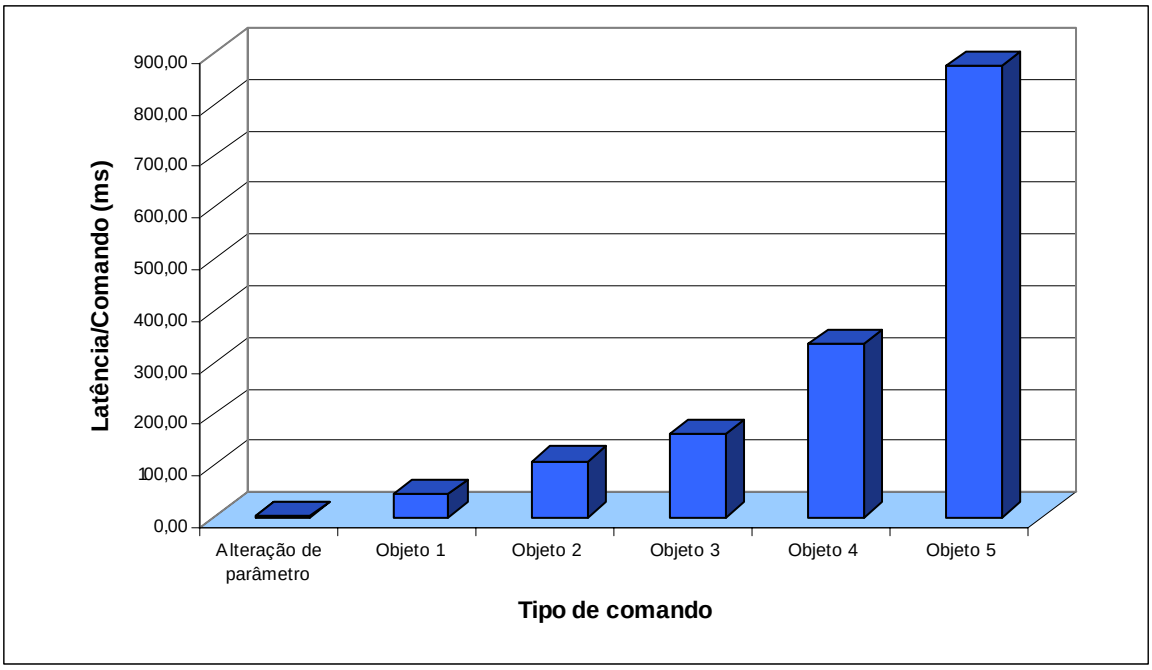


Figura 28 - Latência de acordo com o tipo de comando

Uma comparação com os outros objetos é mostrada na Tabela 8.

Tabela 8 - Complexidade dos objetos x Latência de execução

Tipo de comando	Nº Polígonos	Latência	Comandos em 1s
Alteração de parâmetro	0	2,61	383,69
Objeto 1	0	44,52	22,46
Objeto 2	378	108,85	9,19
Objeto 3	512	159,84	6,26
Objeto 4	3432	335,97	2,98
Objeto 5	6604	875,71	1,14

Para avaliação do sistema de gravação e reprodução e a técnica otimizada proposta, pode-se relacionar o número de eventos processados contra a latência, conforme a Figura 29, medida desde a requisição de um ponto da gravação até o momento em que a cena se torna disponível para o cliente.

Para realização deste experimento foi inserido código no navegador do cliente, de modo a cronometrar o tempo desde o clique do usuário até o momento em que o evento foi processado no cliente. Cada teste, para cada uma das técnicas, foi repetido 10 vezes. O momento da gravação que se desejava reproduzir foi buscado na tabela de gravações, onde se tem a ID e o Tempo em que cada evento foi gravado. Desta forma, uma vez determinado o número do evento que se desejava reproduzir, através desta tabela pode-se determinar o tempo que deveria ser solicitado.

Ainda de acordo com a Figura 29, o primeiro grupo de barras, de rótulo 0, indica que foi carregada apenas a cena inicial (uma cena completa), o segundo grupo, de rótulo 20, indica que 20 comandos aconteceram desde o início da gravação e assim por diante. Como o número de eventos cresce com o passar do tempo, o desempenho da técnica *event-replay* piora à medida que novos eventos são gerados. Isto pode ser facilmente observado uma vez que esta técnica tem que processar todos os eventos, desde o primeiro até aquele imediatamente anterior ao ponto desejado. Portanto, quanto mais eventos, maior a latência de processamento e transmissão.

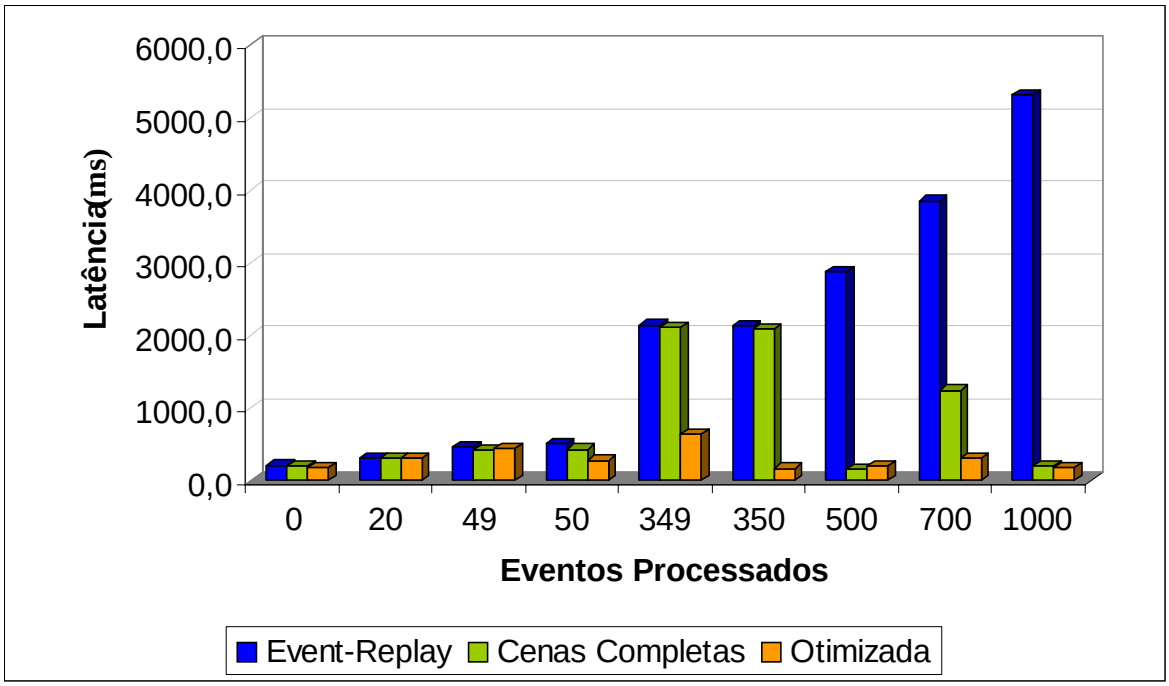


Figura 29 - Latência ao acessar aleatoriamente cenas gravadas

Para a análise dos dois outros casos é necessário observar que as cenas são gravadas após um número fixo de eventos. Dessa forma, requisições feitas imediatamente após uma gravação de uma cena completa ou parcial certamente terão uma latência muito inferior àquelas que fizerem um acesso a um ponto imediatamente anterior a este ponto (onde se tem o maior número de eventos a serem processados).

Na abordagem otimizada, as cenas parciais foram gravadas a cada 50 eventos, sendo que em ambas abordagens as cenas completas foram gravadas a cada 500 eventos (ou seja, a cada 10 cenas parciais na abordagem otimizada). O momento de gravação de uma cena completa, neste caso, coincide com o momento de gravação de uma cena parcial. Assim, quando uma cena completa é gravada, a gravação de cena parcial não é realizada.

As gravações de cenas parciais gastam menos espaço de armazenamento, como poderá ser visto adiante. Por isso pode-se gravá-las a uma taxa mais elevada que a gravação de cenas completas, com um custo de armazenamento igual ou mesmo inferior.

Analisando o acesso à posição em que nenhum evento foi gravado (evento 0), pode-se observar que as três abordagens têm aproximadamente o mesmo atraso. Fica clara esta proximidade nos resultados, visto que todas as três abordagens dependem apenas do carregamento da cena inicial, que é uma cena completa. Algo semelhante acontece no acesso ao evento de número 1000. Como no caso anterior, as abordagens que fazem uso da gravação de cenas completas apenas carregam a cena completa correspondente àquela posição (lembrando que existem cenas completas gravadas nos instantes em que os eventos de número 500 e 1000 foram gravados, daí a necessidade de se carregar apenas a cena completa). Em contrapartida, a técnica *event-replay* tem seu pior resultado neste ponto, por ter que processar os 1000 eventos de uma vez. Também deixa claro que seu desempenho continuaria a piorar a medida que mais eventos fossem gerados. Analisando o acesso ao ponto em que o evento de número 1000 foi gravado, pode-se observar a vantagem do uso de cenas completas, porém, não o de cenas parciais.

Para analisar a vantagem do uso de cenas parciais serão tomados por base os momentos dos eventos de número 349 e 350. A gravação de Cenas Completas tem um desempenho muito próximo nestes dois pontos, certamente porque apenas um evento a mais teve que ser processado. Se comparadas suas latências com as da abordagem Otimizada, pode-se dizer que esta última possui um desempenho 69,7% melhor ao acessar a posição do evento 349 e 91,7% melhor ao acessar a posição do evento 350. As cenas parciais (gravadas a cada 50 eventos) apresentam seu melhor desempenho exatamente neste ponto, e em qualquer outro múltiplo de 50. Pode-se observar que a latência caiu de 640,4ms para 173,8ms, aproximadamente 73% de melhora no desempenho. O desempenho da abordagem Cenas Completas melhora somente ao acessar a posição do evento 500, posição esta em que uma nova cena completa foi gravada. Imediatamente antes deste ponto é onde ele teria seu pior desempenho (posição do evento de número 499) e, portanto, porcentagens de melhoria ainda maiores seriam obtidas pelo uso de cenas parciais.

Com base nestes resultados obtidos cogitou-se a possibilidade de criação de um *cache* na máquina cliente. Como muitas vezes o mesmo objeto será reutilizado, se este *cache* for suficientemente grande para guardar todos os objetos em uso, a latência na inserção de objetos será bastante reduzida, visto que na transmissão pela rede, a latência de inserção de um objeto

passará a ter valores próximos ao da latência de alteração de parâmetros, que é bastante inferior. A criação deste *cache* e de um sistema de predição também serão propostos como trabalhos futuros.

A escolha de se utilizar intervalos de 50 eventos para a gravação de cenas parciais pode ser justificada ainda analisando-se a Figura 29. Com esta taxa de gravação, garantiu-se tempo de resposta médio inferior a meio segundo. Até mesmo nos piores casos (início de reprodução exatamente antes de uma gravação) os resultados foram próximos a este valor, sendo que o pior caso levou 640ms e o melhor apenas 180ms para iniciar a reprodução de um ponto aleatório da gravação. Aguardar este tempo para o início da reprodução não pareceu tedioso aos usuários.

Por fim, um teste para comparação do tamanho das cenas parciais e completas também foi realizado. Para isto utilizou-se um modelo 3D representando uma pequena casa de quatro cômodos. Foram utilizados quatro sensores: monitorando a temperatura de uma sala, a temperatura de um objeto, a presença de chamas em um dos cômodos e a presença de fumaça em outro. As cenas completas tiveram em média 21,8KB de tamanho, enquanto as cenas parciais foram de apenas 0,646KB. É claro que estes valores dependem do cenário de uso, principalmente da quantidade de objetos sendo monitorados. No entanto fica clara a grande diferença de quantidade de espaço de armazenamento necessário para cada tipo de cena.

Neste teste, o ambiente monitorado sofreu alterações, da mesma forma que nos testes anteriores; porém, desta vez os eventos foram controlados para que alguns ocorressem em intervalos aleatórios e outros fossem mais estáveis, ocorrendo em intervalos de tempo maiores. Conforme a Figura 30, os resultados obtidos demonstraram que quando mais de 12 eventos têm essa característica estável, a quantidade de espaço de armazenamento necessário para armazenar somente cenas parciais ultrapassa a quantidade para armazenamento das cenas completas e parciais (otimizado). Assim, o uso de cenas completas é vantajoso quando a quantidade de eventos estáveis ultrapassa certo limiar (que depende dos tamanhos da cena completa e dos eventos), economizando espaço de armazenamento e processamento.

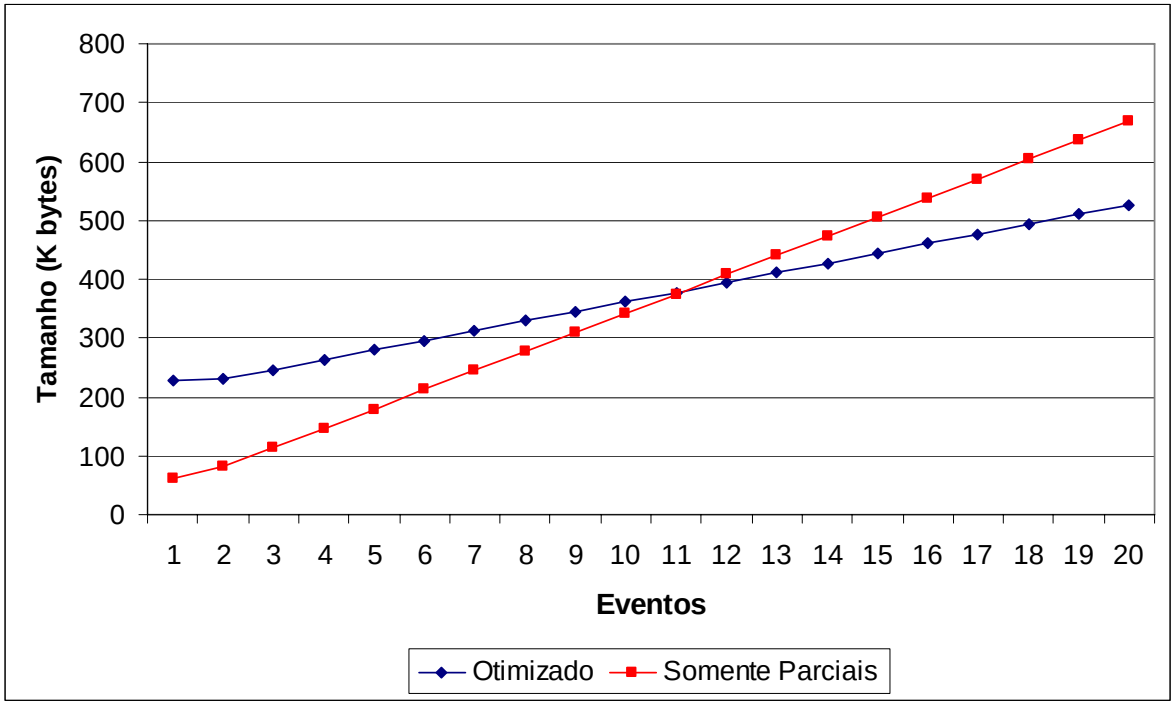


Figura 30 – Economia de espaço de armazenamento com o uso de cenas completas

6.5. Considerações finais

Este capítulo descreveu o método otimizado de gravação proposto, implementado e avaliado. Este método demonstrou algumas vantagens sobre os métodos relatados na literatura, podendo ser destacada a existência do módulo de gravação.

Algumas implementações, como a do RPBIMS, necessitam que o estado seja transmitido pela rede, gerando grande desperdício de largura de banda e limitando a quantidade de gravações de estados realizadas [EFF 04]. Como o módulo de gravação gerencia localmente o estado da cena, ele mesmo recupera e grava os estados sem gerar nenhum tráfego excedente na rede. O RPBIMS também permite que estados sejam descartados caso a máquina que está respondendo à solicitação esteja ocupada [EFF 04]. Mais uma vez, por ser local e não depender de um protocolo externo, o sistema aqui proposto garante a gravação de todos os estados necessários, o que garante um tempo de resposta mais confiável.

Os resultados obtidos se mostraram satisfatórios, uma vez que os testes comparativos entre diferentes técnicas apontaram para o método otimizado como melhor solução. O próximo capítulo tece as considerações finais a respeito do trabalho.

7. Conclusões

Este capítulo descreve as contribuições geradas com a realização deste trabalho, bem como os trabalhos futuros e as considerações finais.

7.1. Contribuições Geradas

O monitoramento de ambientes físicos cientes de contexto sujeitos a situações de emergência que utiliza interface de realidade virtual é algo pouco explorado atualmente, assim como a gravação e a reprodução de mídia tridimensional. Em ambos os casos as contribuições geradas serão úteis para outros trabalhos do *LRVNet* e demais pesquisadores do grupo.

A relevância do sistema proposto pôde ser observada pelo manifesto interesse da Empresa Brasileira de Aeronáutica, EMBRAER, em utilizar o sistema de preparação para emergência sendo desenvolvido no *LRVNet* no monitoramento de ensaios de vôo, bem como gravá-los para posterior análise. Um projeto de colaboração entre a Embraer e o *LRVNet* está em andamento.

Artigos Publicados

Um artigo foi publicado e outros dois estão em fase de análise, são eles:

- LOPES, A. R.; ARAUJO, R. B.; BOUKERCHE, A. *Recording and Playing 3D Collaborative Virtual Environment Simulations as Continuous Media for Critical Security Applications*. In: 4th International Information and Telecommunication Technologies Symposium (I2TS). Florianópolis, Brasil. 14-16 de Dezembro, 2005.
- LOPES, A. R.; ARAUJO, R. B.; BOUKERCHE, A. *Recording and Playing Events for the Emergency Preparedness Class of Applications*. Submetido ao 10-th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, October 2-6, 2006, Torremolinos, Malaga, Spain – resultado da avaliação a ser divulgado em 30/06/2006.
- LOPES, A. R.; ARAUJO, R. B.; BOUKERCHE, A.; Andrade, L.; Gaspar, F. *A Visual Interface to a Complex Emergency Preparedness System*. Submetido ao 19th annual ACM Symposium on User Interface Software and Technology, October 15-18, 2006, Montreux, Switzerland - resultado da avaliação a ser divulgado em 12/06/2006.

7.2. Trabalhos Futuros

Os seguintes trabalhos futuros podem ser destacados:

- Desenvolvimento de um formato de arquivo próprio, eficiente para gravação e reprodução de eventos e cenas;
- Reprodução em sentido inverso (*rewind*);
- Criação de um *cache* para armazenamento de objetos e texturas utilizados para representar o monitoramento;
- Criação de um sistema de predição para que durante o monitoramento sejam indicados possíveis incidentes antes mesmo que estes se manifestem;
- Criação de uma rede de atuadores e sensores sem fio (ou simulador) que interaja com este sistema, visto que foram utilizados apenas dados fictícios até o momento;
- Modificar o servidor para que diferentes cenas sejam utilizadas, com diferentes níveis de detalhes. Desta forma, pode-se ter uma cena que apresente um avião inteiro, em que a ocorrência de um evento em uma turbina seja apresentada de forma geral. Ao se clicar nesta turbina, o navegador do cliente pode ser direcionado a uma outra cena, contendo apenas ela e onde cada uma de suas partes é exibida com maiores detalhes;
- Inserção de uma barra de navegação que indique diferentes pontos de vista, identificados pelos objetos aos quais correspondam. Embora o próprio navegador já disponibilize uma, nela não podem ser vistos todos os pontos de vista de uma vez. Além de esta barra permitir a troca do ponto de vista, seus botões podem indicar a situação do objeto que representam, por exemplo, piscando caso um evento necessite de atenção. Embora possa ser utilizada transparência nos objetos, observou-se a necessidade de uma barra como esta porque em alguns casos objetos podem ficar oclusos.

7.3. Conclusões

O monitoramento de ambientes físicos tem grande importância, principalmente em ambientes cuja segurança é crítica. Nestes ambientes, desconhecer o que está ocorrendo pode levar a perdas de vidas e de patrimônio.

Um sistema de monitoramento preciso, visualizado através de um ambiente de realidade virtual, apresenta vantagens sobre os sistemas que utilizam interfaces

bidimensionais. Entre estas vantagens está o maior potencial para representação de objetos complexos, bem como maior precisão para identificação de pontos de origem de situações que podem levar a emergências.

Nesta dissertação de mestrado foi apresentada uma linguagem visual para o monitoramento de ambientes de segurança crítica, com enfoque na área de aviação. Esta linguagem visual foi concebida com a colaboração do Departamento de Artes e Comunicação da UFSCar. Ela provê uma interface visual, baseada nos conceitos da semiótica, que permite a identificação de situações de emergência de forma mais rápida e com maior precisão. Também foi visto o projeto de um sistema de gravação/reprodução, que grava AV3D como mídia contínua.

O fator de inovação deste sistema reside em dois pontos principais: utiliza uma técnica otimizada de gravação que economiza tempo de processamento e espaço de armazenamento; e os comandos de atualização de cena são independentes de navegador, permitindo a visualização do AVC independentemente do navegador 3D adotado pelo usuário. O mecanismo otimizado foi contrastado com o mecanismo de gravação de cenas totais e parciais, mostrando melhor desempenho para a classe de aplicações que sofrem a ocorrência de um número arbitrariamente grande de eventos gerados a qualquer momento (uma característica da classe de aplicações de monitoramento de ambientes sujeitos a situações de emergência)

Além do monitoramento em tempo-real de ambientes físicos cientes de contexto, a utilização deste sistema pode ser estendida também para perícias feitas por especialistas em segurança e/ou seguradoras, bem como em situações de treinamento de equipes.

Por fim, os testes realizados demonstraram que o sistema proposto tem potencial para uso eficaz na classe de aplicações de preparação para emergência. Um projeto foi submetido à FAPESP pelo *LRVNet* com o apoio da EMBRAER. Assim, as pesquisas realizadas neste trabalho terão continuidade e serão focadas, num primeiro momento, em situações específicas de monitoramento de aeronaves em ensaios de solo e vôo, cujos dados para simulação utilizarão dados reais a serem fornecidos pela EMBRAER.

Glossário

API	Application Program Interface – Interface para Programação de Aplicativos.
Avatar	Representação geométrica de um usuário participante de um ambiente de Realidade Virtual. Pode variar de um sofisticado modelo 3D até um simples ícone.
Bug	Qualquer defeito encontrado em um programa de computador.
EMBRAER	Empresa Brasileira de Aeronáutica SA.
FAPESP	Fundação de Amparo à Pesquisa do Estado de São Paulo.
GeoSpatial	Inclui como associar localizações do mundo real com elementos no mundo X3D, bem como especificar nós particularmente direcionados para aplicações que utilizam regiões geográficas.
H-Anim	Humanoid Animation – Padrão ISO que define o modelo de articulações de um corpo igual ao humano.
Middleware	Software de interface que permite interação de diferentes aplicações de softwares, geralmente sobre diferentes plataformas de hardware e infra-estrutura, para troca de dados.
NURBS	Nonuniform Rational B-Splines – Representação matemática para curvas e superfícies lisas. Utilizado em sistemas CAD (Computer Aided Design - desenho com ajuda do computador) de superfícies complexas.
Toolkit	Conjunto de ferramentas para desenvolvimento ou grupo de programas e rotinas usadas como base para programação em um novo sistema ou de um hardware especial.
VRML	Virtual Reality Modelating Language – Linguagem para Modelagem de Realidade Virtual. Normalmente utilizado na Internet.
Web Services	Serviços Web. Componente de software capaz de ser acessado através de protocolos de rede como SOAP sobre HTTP. Utiliza um sistema de mensagens XML padronizado.

X3D	eXtensible 3D – 3D extensível. Sucessor do VRML.
Xj3D	Navegador e Toolkit para X3D escrito em linguagem Java.
XML	eXtended Markup Language – Linguagem de Marcação Extensível.
XMT	eXtensible MPEG-4 Textual Format – Formato para descrição de cenas MPEG-4 em XML.

Anexo A – Semiótica

As imagens têm se mostrado como um meio de comunicação bastante eficiente e coerente com a sociedade dinâmica em que nos encontramos. A necessidade de se obter informação de uma forma rápida, precisa e de fácil compreensão, juntamente com todo o potencial comunicativo das imagens, fez com que elas fossem implantadas e modeladas conforme o contexto. Desta forma pode ser feita uma troca de informação veloz, explorando toda a potencialidade de significado que a imagem possui e não a subjugando a ilustrar fatos descritos por uma notícia ou uma legenda [DON 97].

As cores, as formas geométricas e a organização das imagens, a forma como elas são apresentadas através de um suporte (como um papel, uma placa de informação ou uma interface gráfica) e até mesmo a linguagem verbal escrita utilizada de forma sucinta, são alguns itens que fazem parte da composição imagética e que passaram a ser produtores individuais de significado para o significado maior que a imagem deve ter [PED 77].

No caso da linguagem verbal escrita, é interessante que ela faça parte da comunicação imagética, porém como mais um produtor de significado e não, apenas, para descrever a imagem. Os *outdoors*, os *banners*, as fachadas luminosas, as placas de trânsito, as interfaces gráficas, entre outros, são exemplos de como a imagem pode ser utilizada para fazer uma comunicação rápida, precisa e com alto teor de informação.

A importância da imagem como meio de comunicação não é privilégio da sociedade atual, apesar de estar sendo bastante utilizada atualmente. Antes mesmo da utilização da linguagem verbal escrita, as imagens eram utilizadas para a troca de informação [DAY 85]. A maior eficiência na comunicação que foi conseguida através da linguagem verbal, agora está revertendo-se para a linguagem imagética. Isso acontece devido ao nível de complexidade das linguagens.

Anteriormente, a comunicação verbal escrita desenvolveu-se a partir dos pictogramas e passou a ser mais completa, eficiente e com um nível de complexidade superior à linguagem imagética; para o tipo de comunicação que na época era necessário ela se fazia coerente. Hoje, a linguagem verbal escrita não suporta a comunicação dinâmica e direta. Sua complexidade dificulta a troca de informação rápida, prejudicando a comunicação. Já a linguagem imagética comunica de uma forma mais dinâmica, simples e direta. Por exemplo: observa-se nos meios de comunicação atuais, como nas propagandas de revistas e outdoors, a preferência pela

comunicação através de imagens. Nesses casos, muitas vezes, ocorre uma inversão de valor: anteriormente a imagem ilustrava o texto escrito, hoje, a imagem é que chama a atenção para o anúncio e seduz quem o vê; o texto verbal serve de complemento para ela, com informações adicionais do produto.

Numa interface gráfica de um *site* por exemplo, onde o usuário procura por alguma informação, as imagens devem ser mais indicativas do que as palavras. Essas se reduzem a palavras chave [PED 77]. Todo o direcionamento do usuário dentro do *site* é sugerido pelas imagens. Segundo Dondis [DON 97] e Caleb Gattegno, todo o direcionamento do usuário dentro de um *site* é sugerido pelas imagens.

No entanto, toda essa simplicidade característica da imagem serve como uma máscara para toda a complexidade de construção dela. O processo de confecção imagético envolve uma série de fatores, dos quais são relevantes serem citados a forma geométrica, cor, equilíbrio e tensão da imagem, organização imagética, similaridade gráfica, simetria, contraste e a harmonia, entre outros.

A forma geométrica pode conter uma informação de estabilidade ou instabilidade. Um quadrado, um triângulo ou um círculo representados num plano cartesiano, estão centrados sob o eixo vertical e horizontal e, portanto, estão em equilíbrio. Esses eixos regem o equilíbrio da imagem e não são agressivas à percepção humana.

No entanto, quando a forma geométrica foge dos parâmetros da base horizontal e da coluna vertical, ela pode causar um desconforto para quem vê. Esse desconforto é causado por uma tensão da imagem. Quando ela não se “encaixa” nas linhas verticais e horizontais, há uma tendência de se formar uma imagem tensa, desequilibrada. Outro fator que contribui para essa característica imagética citada é a assimetria da imagem. Uma forma geométrica irregular, que foge dos padrões das linhas verticais e horizontais, pode gerar uma imagem tensa [DAY 85]. Pensando-se em comunicação visual, não é interessante produzir uma imagem com um certo grau de tensão, pois isso pode parecer estranho à pessoa que deve receber a mensagem, a menos que essa seja a intenção. É interessante a uma imagem comunicativa que ela esteja em harmonia consigo mesma. Por exemplo: que as formas geométricas sejam parecidas e que estejam agrupadas por semelhança; que a informação verbal escrita seja sucinta, pontual; que as cores utilizadas não sejam muito contrastantes. O contraste deve ser usado de forma moderada, sendo utilizado mais intensamente quando há a necessidade de chamar a atenção para alguma coisa [DON 97].

As cores, particularmente, possuem características físicas que alteram a percepção fisiológica ocular. Elas em estado de luminância possuem diferentes tamanhos de espectros. O vermelho é a cor de maior espectro visível, portanto sua percepção ocular é bastante intensa. Ela é mais agressiva à percepção fisiológica do olho. Já o azul possui um dos menores espectros de cor, por isso, sua percepção ocular não é tão intensa e é pouco agressiva a quem vê. Assim, justifica-se o porquê da cor vermelha ser bastante usada para indicar algo perigoso, ou chamar a atenção para algo.

As cores do semáforo, por exemplo, têm fundamento no tamanho do espectro delas. O vermelho que tem uma percepção mais intensa chama bastante a atenção e indica que se deve parar. O amarelo, que possui um espectro menor que o do vermelho, mas maior que o do verde, chamar a atenção indicando que o sinal irá ficar vermelho. Já o verde, possui o menor espectro de cor e portanto não é agressivo à percepção da visão, indicando para prosseguir.

O tamanho dos espectros e de percepção fisiológica ocular se dá na seguinte ordem: Ultra-violeta – violeta (400 mu)– azul (480 mu) – verde (560 mu) – amarelo (580 mu) – laranja (620 mu) – vermelho (700 um) – infra-vermelho.

Anexo B – Modelagem UML do Sistema

Este anexo apresenta a modelagem em UML do sistema de monitoramento e gravação de ambientes virtuais 3D, que permite visualização em tempo-real e posterior, sendo as reproduções realizadas da mesma forma que em mídia contínua. Inicialmente serão apresentados os principais casos de uso, seguido do diagrama de atividades, diagrama de classes e, por fim, o diagrama de seqüência.

Casos de Uso

É apresentado na Figura 31 o caso de uso em que o usuário do sistema tem duas opções: visualizar os acontecimentos do mundo físico em tempo-real ou reproduzir aqueles que foram anteriormente gravados.

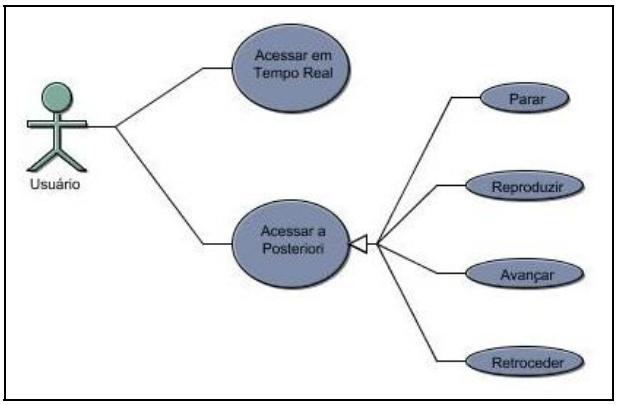


Figura 31 - Caso de uso – Acessar o sistema

Se ele optar por uma visualização em tempo-real, nenhuma interação a mais será feita por ele, apenas os comandos de alteração da cena vindos do ambiente físico é que alterarão a cena.

Por outro lado, a visualização posterior permitirá ao usuário iniciar a reprodução a partir de determinado tempo, podendo, por exemplo, avançar ou retroceder na gravação.

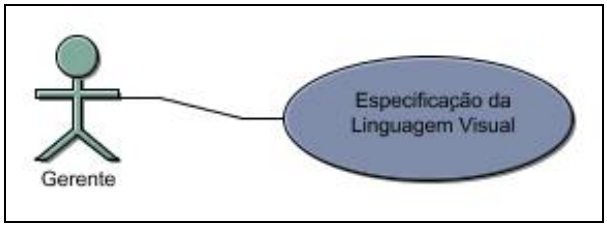


Figura 32 - Caso de Uso – Especificar a Linguagem Visual

Como já mencionado, a definição da linguagem visual é responsabilidade do gerente, como mostrado na Figura 32.

Diagrama de Atividades

É mostrado na Figura 33 o diagrama de atividades. Deve-se lembrar que o usuário terá duas opções básicas: visualizar o ambiente em tempo-real ou posteriormente.

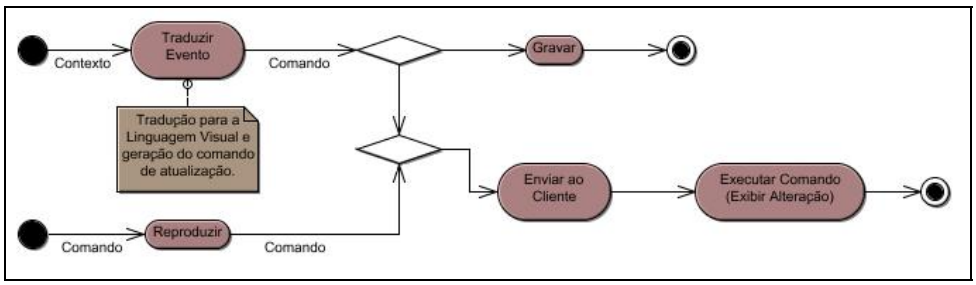


Figura 33 - Diagrama de Atividades

Também poderão existir vários clientes acessando em tempo-real e posteriormente ao mesmo tempo. Ações são desencadeadas a partir de um contexto recebido do ambiente físico. Este contexto é traduzido para a Linguagem Visual e então é gerado um comando para alteração da cena no navegador. Então, o comando é enviado ao navegador do cliente que, quando executá-lo, apresentará visualmente o contexto recebido.

Simultaneamente, estes contextos estarão sendo gravados para posterior acesso e, também simultaneamente, outros clientes poderão estar fazendo acessos a cenas previamente gravadas.

Este processo de visualização posterior não necessita traduzir os contextos, uma vez que os comandos é que foram gravados. Os comandos armazenados em disco são acessados pelo reprodutor e enviados ao cliente, como no caso anterior.

Não se deve esquecer que no acesso às gravações eventuais interações do usuário, como avanço ou retrocesso, necessitam de um tratamento por parte do reprodutor.

Diagrama de Classes

As quatro principais classes utilizadas para o desenvolvimento deste trabalho são apresentadas na Figura 34.

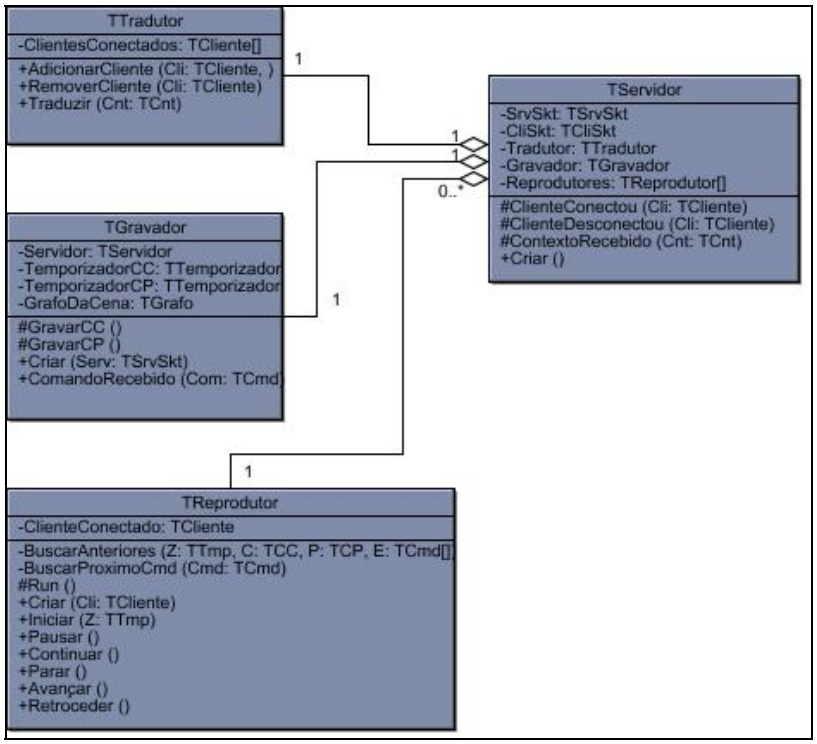


Figura 34 - Diagrama de Classes

A classe *TTradutor* é a responsável por traduzir os contextos capturados pela rede de sensores para a Linguagem Visual, compor comandos para alteração da cena no navegador e os enviar aos clientes conectados para acesso em tempo-real. Esta tarefa é executada pelo evento *Traduzir()*, cujo pseudocódigo é mostrado na Figura 35. A lista dos clientes conectados para acesso em tempo-real estão contidos no vetor *ClientesConectados* desta mesma classe.

```
Traduzir(Cnt: TCnt)
    Consultar a Linguagem Visual
    Se NecessitaObjeto3D, Então
        Recupera Obj3D do Repositório de Objetos
    FimSe
    Constrói o Comando de Atualização
    Envia este Comando aos ClientesConectados
Fim
```

Figura 35 - Pseudocódigo - *TTradutor.Traduzir(Cnt)*

Em certos casos, para a construção do comando há a necessidade de inserir um objeto 3D na cena. Estes objetos se encontram armazenados num repositório de objetos 3D. Um exemplo destes objetos pode ser a chama que será usada em casos de incêndio.

A classe *TGravador* é a responsável por gravar os comandos a medida que são recebidos. Ele se conecta ao *TTradutor* como se fosse um cliente comum, utilizando o parâmetro *Servidor* para especificar as informações de conexão.

A cada mensagem recebida, esta conexão executa o evento *ComandoRecebido(Com)*, conforme ilustra a Figura 36. Este evento é responsável por alterar o grafo da cena (parâmetro *GrafoDaCena*) e gravar tal comando.

```
ComandoRecebido(Com)
    Gravar Com
    Alterar GrafoDaCena, segundo Com.
    Para cada nó alterado por Com
        Nó.Alterado = Verdadeiro
Fim
```

Figura 36 - Pseudocódigo - *TGravador.ComandoRecebido(Com)*

Os temporizadores *TemporizadorCC* e *TemporizadorCP* são usados apenas para disparar os eventos *GravarCC* e *GravarCP*, respectivamente.

A gravação de uma cena completa (*GravarCC*, Figura 37) é feita gravando todo o grafo da cena. Uma vez gravada uma cena completa, deve-se "zerar" o grafo da cena. Isto é feito para que cenas parciais não gravem comandos que já constam na cena completa.

```
GravarCC()
    CC = vazio
    Para cada Nó faça
        CC = CC + Nó
        Nó.Alterado = Falso
    FimPara
    Gravar CC
Fim
```

Figura 37 - Pseudocódigo - *TGravador.GravarCC()*

Por sua vez, as cenas parciais (*GravarCP*, Figura 38) gravam apenas os nós alterados pelos comandos.

```
GravarCP()
    CP = vazio
    Para cada Nó faça
        Se Nó.Alterado Então
            CP = CP + Nó
        FimSe
    FimPara
    Gravar CP
Fim
```

Figura 38 - Pseudocódigo - *TGravador.GravarCP()*

A terceira classe, a *TReprodutor*, é a responsável por estabelecer conexões com os clientes que desejam visualizar cenas gravadas previamente. É responsável por recuperar os dados gravados e os enviar ao cliente. Também fornece funcionalidades como reprodução aleatória, avanço e retrocesso.

Para permitir que clientes acessem as gravações de forma independente um do outro (em relação ao tempo), devem existir instâncias diferentes (ou *threads*) do reprodutor para cada cliente. Na criação da instância (*Criar(Cli)*) já será determinado o cliente de sua responsabilidade, e este ficará armazenado em *ClienteConectado*.

Depois de conectado, o cliente envia uma mensagem indicando o tempo de início. Este comando invocará o método *Iniciar(Z)*. Quando o reprodutor recebe esta mensagem, recupera a cena completa e a parcial, os comandos entre a cena parcial e o ponto desejado e envia tudo isso ao cliente. Seu pseudocódigo se encontra na Figura 39.

```
Iniciar(Z)
    BuscarAnteriores(Z, CC, CP, Cmd[])
    ClienteConectado.Enviar(CC)
    ClienteConectado.Enviar(CP)
    ClienteConectado.Enviar(Cmd[])
Fim
```

Figura 39 - Pseudocódigo – *TReprodutor.Iniciar(Z)*

O evento *BuscarAnteriores()* é apresentado na Figura 40.

```
BuscarAnteriores(Z, CC, CP, Cmd[])
    CC = Última Cena Completa tal que seu tempo seja
menor que Z
    CP = Última Cena Completa tal que seu tempo esteja
entre CC.Tempo e Z
    Para cada Cmd entre CP.Tempo e Z Faça
        Cmd[] = Cmd[] + Cmd
    FimPara
Fim
```

Figura 40 - Pseudocódigo - *TReprodutor.BuscarAnteriores()*

Durante uma sessão, o método *Run* deve constantemente obter o próximo comando armazenado em disco e, no tempo certo, disponibilizar ao cliente. Conforme a Figura 41.

```
Run()
    Enquanto Verdadeiro Faça
        CmdAnterior = Cmd
        BuscarProximoCmd(Cmd)
        Aguardar(Cmd.Tempo - CmdAnterior.Tempo)
        ClienteConectado.Enviar(Cmd)
    FimEnquanto
Fim
```

Figura 41 - Pseudocódigo – *TReprodutor.Run()*

O evento *BuscarProximoCmd ()* apenas recupera do disco comandos seqüencialmente.

Por fim, a classe *TServidor* que trabalha como um "gerente". Ela basicamente é responsável por criar as outras classes e gerenciar as conexões dos clientes, verificando se é uma conexão em tempo-real ou posterior e tomando as decisões necessárias.

Será mostrado apenas o pseudocódigo do evento *ClienteConectou()* desta classe. Como ilustra a Figura 42.

```
TServidor.ClienteConectou(Cli)
    Se Cli.Tipo = TempoReal Então
        Tradutor.AdicionarCliente(Cli)
    Senão
        Reprodutor[] = TReprodutor.Criar(Cli)
    FimSe
Fim
```

Figura 42 - Pseudocódigo - *TServidor.ClienteConectou()*

Na seção a seguir será exibido um diagrama de seqüência, mostrando a interação entre os objetos do sistema.

Diagrama de Seqüência

Uma seqüência de mensagens trocadas entre as classes apresentadas na seção anterior é mostrada no diagrama da Figura 43. Trata-se apenas de um exemplo que deverá ajudar na compreensão do funcionamento global do sistema.

A execução do sistema pelo *Administrador* instancia um objeto da classe *TServidor*, que por sua vez cria um objeto da classe *TTradutor* e outro da *TGravador*. Existirá apenas um objeto de cada uma destas três classes durante toda a execução do programa.

O *Gravador*, que acaba se ser criado, se conecta imediatamente ao *Servidor*, como se fosse um cliente. O *Servidor*, ao verificar que se trata de uma conexão para visualização em tempo-real, adiciona este cliente ao módulo de *Tradução*. A partir daqui, todos os eventos provenientes da rede de sensores já serão recebidos pelo gravador e, conseqüentemente, gravados em disco.

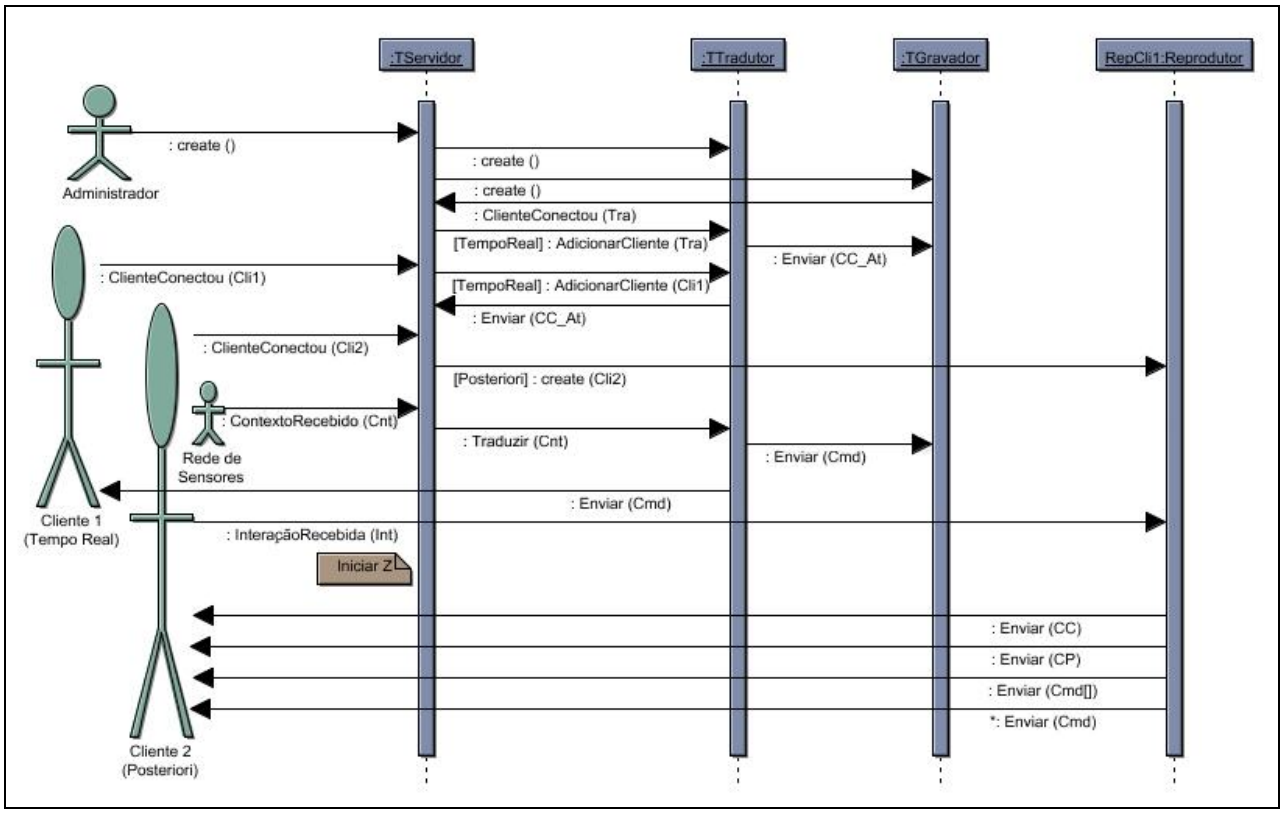


Figura 43 - Diagrama de Seqüência

Também deve ser observado que quando um cliente se conecta para monitorar em tempo-real, o primeiro comando enviado contém a cena inicial, com todo o grafo de cena atualizado.

Depois de determinado tempo o *Cliente 1* se conecta ao servidor, desejando visualizar os eventos em tempo-real. Da mesma forma que o gravador, todos os clientes que desejarem visualizar as cenas em tempo-real serão adicionados às conexões do tradutor e receberão a cena atual.

Diferentemente, o *Cliente 2* se conecta ao servidor desejando fazer um acesso a posterior. Ao invés de serem adicionados à lista de conexões do Tradutor, clientes que desejam este tipo de conexão fazem com que o servidor instancie um objeto *TReprodutor*. Lembrando que para cada um destes clientes existirá uma instância diferente da classe *TReprodutor*.

Quando a rede de sensores enviar um evento ao servidor, este imediatamente chamará o evento *Traduzir* do tradutor. Após a tradução do evento para a linguagem visual e construção do comando de atualização da cena, o tradutor envia mensagens (ou uma mensagem em *broadcast*) a todos os clientes conectados a ele. Neste exemplo, ao *Gravador* e ao *Cliente 1*. Será assim com todos os eventos vindos da rede de sensores.

Mensagens dos usuários conectados para acesso a posterior são destinadas diretamente aos seus correspondentes reprodutores. Neste exemplo supõe-se uma mensagem solicitando o início da reprodução em determinado tempo (Z).

Como consta no pseudocódigo da Figura 39, o *Reprodutor* envia a cena completa, a parcial e os eventos entre a cena parcial e o tempo desejado. Por fim, são enviados os eventos seguintes, espaçados de acordo com o tempo em que ocorreram.

Referências Bibliográficas

[APP 05] Apple. “QuickTime - Tutorials – RTSP”. Disponível: http://www.apple.com/quicktime/tools_tips/tutorials/rtp.html [consulta: Fevereiro de 2005].

[ARA 03] Araújo, R. “Computação Ubíqua: Princípios, Tecnologias e Desafios”. São Carlos, UFSCar. 2003.

[BEN 01] Benford, S.; Greenhalgh, C.; Rodden, T.; Pycock, J. “Collaborative virtual environments”. In: Communications of the ACM. Volume 44, Number 7. 2001. p. 79-85.

[BEN 02] Benford, S.; Flintham, M.; Greenhalgh, C. E Purbrick, J. “Applications of temporal links: recording and replaying virtual environments”. In: Virtual Reality Conference. Proceedings. IEEE. 2002. p. 101-108.

[BOU 05] Bouras, Ch.; Panagopoulos, A.; Tsiatsos, Th. “Advances in X3D multi-user virtual environments”. Seventh IEEE International Symposium on Multimedia. Irvine, California. Dezembro de 2005. p. 136 – 142.

[CAM 97] Campbell, B. E Marrin, C. “Teach Yourself Vrml 2 in 21 Days”. Indiana: Sams, 1997. p. 479.

[CAR 04] Carvalho, G. N. M.; Gill T.; Parisi T. “X3D programmable shaders”. Proceedings of the ninth international conference on 3D Web technology. Monterey, Califórnia, 2004. p. 99 – 108.

[CAR 93] Carlsson, C. E Hagsand, O. “DIVE A multi-user virtual reality system”. In: Virtual Reality Annual International Symposium. Proceedings, IEEE. 1993. p. 394-400.

[CHE 04] Chen, N.; Hong, J.; Jiang, X., Wang K., Takayama, L. e Landay, J. “Siren: Context-aware Computing for Firefighting”. In: Pervasive’ 2004, 2004.

[CHE 95] Chen, M. S.; Kandlur, D.D. “Downloading and Stream Conversion: Supporting Interactive Playout of Videos in a Client Station”. In: Multimedia Computing and Systems. Proceedings of the International Conference on. Washington, DC, Maio de 1995. p. 73-80.

[DAR 04] Darpa Information Exploitation Office. “Command Post of the Future”. Disponível: http://dtsn.darpa.mil/ixo/programdetail.asp?progid=18 [consulta: Fevereiro de 2004].

[DAY 85] Dayan, D. “O Código-Matriz do Cinema Clássico”. In: GEADA, Eduardo (Org.). Estética do Cinema. Lisboa: Dom Quixote, 1985.

[DEY 01] Dey, A. “Understanding and Using Context”. In: Personal and Ubiquitous Computing, 2001. p. 4-7.

[DON 97] Dondis, D. A. “Sintaxe da Linguagem Visual”. 2a Edição. São Paulo: Martins Fontes, 1997.

[EFF 04] Effelsberg, W; Hilt, V.; Mauve, M. E Vogel, J. “Recording and Playing Back Interactive Media Streams”. In: Transactions on Multimedia. IEEE, 2004.

[FOU 99] Foundation, Free Software Inc. “GNU Lesser General Public License Version 2.1”. Boston, MA, USA, Fevereiro de 1999.

[FRE 99] Frécon, E.; Greenhalgh, C. E Stenius, M. “The DiveBone – an application-level network architecture for Internet-based CVEs”. In: Virtual reality software and technology. Proceedings. ACM, 1999. p. 58-65.

[GEO 01] Georganas, N. E Hosseini M. “Suitability of MPEG4’s BIFS for Development of Collaborative Vitual Environments”. In: International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Proceedings. IEEE, 2001. p. 299 – 304.

[GEO 02] Georganas, N. E Hosseini M. “MPEG4 Based Recording and Replay of Collaborative Virtual Reality Sessions”. In: Virtual Reality Conference. Proceedings. IEEE, 2002. p. 271.

[HAR 95] Harrison, S.; Janssen, B.; Kurtenbach, G.; Minneman, S.; Moran T. E Smith, I. “A confederation of tools for capturing and accessing collaborative activity”. In: Third International Conference on Multimedia. Proceedings. ACM. São Francisco, 1995. p. 523-534.

[ISO 02a] International Organisation For Standardisation, “MPEG-4 Overview”. Março de 2002. Disponível: http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm [consulta: Fevereiro de 2005].

[ISO 02b] International Organisation For Standardisation. “MPEG-4 Systems. Coding of audio-visual objects – Part 1: Systems - Amendment 2: Textual Format”. Março de 2002. Disponível: http://www.chiariglione.org/mpeg/working_documents/mpeg-04/systems/3rd_edition_wd.zip [consulta: Fevereiro de 2005].

[JAV 06] Java.net. “Java 3D”. Disponível: https://java3d.dev.java.net/ [consulta: Abril de 2006].

[KUD 04] Kudo, T. “Computação Ciente de Contexto Aplicada ao Monitoramento de Condições Críticas em Ambientes Físicos”. Dissertação (Mestrado em Ciência da Computação) – Departamento de Computação. Universidade Federal de São Carlos, São Carlos, 2004. p. 98.

[LAM 04] Lamar, M. “Projeto SBTVD - Sistema Brasileiro de Televisão Digital”. 2004. Disponível: http://www.eletrica.ufpr.br/lamar/Projetos/CODIS-II/PT_Compressao_UFPR.pdf [consulta: Fevereiro de 2005].

[LEM 04] Lemle, M. “Pesquisas em TV Digital avançam no Rio de Janeiro”, Agosto de 2004. Disponível: http://www.faperj.br/boletim_interna.phtml?obj_id=1367 [consulta: Fevereiro de 2005].

[LYY 02] Lyytinen, K. E Yoo, Y. “Issues and Challenges in Ubiquitous Computing”. In: Communications of the ACM, Vol. 45, No. 12, Dezembro de 2002. p. 62-65.

[MAR 04] Marca, R.; Rasia, M. E Souza, U. “MPEG-4 x MPEG-2 na Implantação do Sistema Brasileiro de TV Digital”. Março de 2004. Disponível: http://www.eletrica.ufpr.br/lamar/Projetos/CODIS-II/Projeto_Final2003/MPEG4xMPEG2.pdf [consulta: Novembro de 2004].

[MAR 06] Marques, G. “TV digital: debate ainda vai longe”. Jornal O Estado de São Paulo. São Paulo, SP. Abril de 2006.

[MIC 04] Micheloti, G. “Proposta de um Middleware para Monitoramento de situações de emergência em Ambientes Físicos ou Lógicos Cientes de Contexto”. Dissertação (Mestrado em Ciência da Computação) – Departamento de Computação. Universidade Federal de São Carlos. São Carlos, SP. 2004. p. 114.

[PAZ 04] Pazzi, R. W. N. "Especificação e Implementação de um Protocolo Tolerante a Falhas e de Baixa Latência para Redes de Sensores Sem Fio". Dissertação (Mestrado em Ciência da Computação) – Departamento de Computação. Universidade Federal de São Carlos. São Carlos, SP. 2004. p. 88.

[PED 77] Pedrosa, I. “Da Cor à Cor Inexistente”. Editora: Universidade de Brasília. Brasília, 1977.

[SEV 00] Sévenier, M E Walsh, A. “Core Web 3D”, Prentice Hall, 2000. p. 1152.

[SIL 05] Silva, Fernando H. S.; Boukerche, A.; Araujo, R. B. “A Low Latency and Energy Aware Event Ordering Algorithm for Wireless Actor Sensor Networks”. In: 8-th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems. Montreal. 2005.

[TOD 04] Todesco, G. e Zuffo, M. “MPEG-4 Teoria e Prática”. In: Teixeira, C. e Goularte, R. Tópicos em Tecnologias Web & Multimídia. Ribeirão Preto: SBC. Capítulo 1. 2004. p. 1-38.

[WEB 06a] Web3D Consortium. “X3D Overview”. Disponível: http://www.web3d.org/x3d/overview.html [consulta: Abril de 2006].

[WEB 06b] Web3D Consortium. “X3D FAQ”. Disponível: http://www.web3d.org/x3d/faq/index.html [consulta: Abril de 2006].

[WEI 91] Weiser, M. “The computer for the 21st century”. In: Scientific American. vol. 265, no. 3, Setembro 1991. p. 94-104.

[WRI 04] Wilson, J.; Pelt, A. V.; Snydal, J.. “FireEye: Needs and Usability Assessment of a Head-Mounted Display for Firefighters”. Berkeley’s Mechanical Engineering department. Chicago. Maio de 2005. Disponível: http://www.offhanddesigns.com/jon/docs/FireEye.pdf [consulta: Abril de 2006].

[XJ3 06] Xj3D Project. “Reference Sotfware X3D”. Disponível: http://www.xj3d.org [consulta: Abril de 2005].