UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA ESTRUTURA DE SUPORTE PARA ADAPTAÇÃO EM JOGOS 3D MULTIUSUÁRIO

Alessandro Rodrigues e Silva

SÃO CARLOS
2003

UMA ESTRUTURA DE SUPORTE PARA ADAPTAÇÃO EM JOGOS 3D MULTIUSUÁRIO

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA ESTRUTURA DE SUPORTE PARA ADAPTAÇÃO EM JOGOS 3D MULTIUSUÁRIO

Alessandro Rodrigues e Silva

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para obtenção do Título de Mestre em Ciência da Computação.
Orientadora: Dr Regina Borges de Araújo

SÃO CARLOS
2003

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

S586es

Silva, Alessandro Rodrigues e.
Uma estrutura de suporte para adaptação em jogos 3D multiusuário / Alessandro Rodrigues e Silva. -- São Carlos : UFSCar, 2009.
139 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2003.

1. Ambientes virtuais colaborativos. 2. Ambientes 3D. 3. Adaptação de conteúdo. 4. MPEG-4. 5. Multiusuário. 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

UMA ESTRUTURA DE SUPORTE PARA ADAPTAÇÃO EM JOGOS 3D MULTIUSUÁRIO

ALESSANDRO RODRIGUES E SILVA

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 titulo de Mestre em Ciência da Computação.

Membros da Banca:

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

Prof. Dr. Sérgio Donizetti Zorzo
(DC/UFSCar)

Profa. Dra. Líria Matsumoto Sato
(POLI/USP)

São Carlos
Agosto/2003

“...the world isn't run by the laws written on paper, it's run by people, some according to laws, others not.....”

Said by Thomas 'Tommy' Angelo in "MAFIA: The city of Lost heaven (2002)", voice by Mike Sorvino, written by Daniel Vavra.

Dedico este trabalho a minha falecida Mãe

AGRADECIMENTOS

Á minha família, principalmente a minha madrinha que me ajudou em todo o percurso de minha vida mais do que ninguém, ao meu pai pelo apoio e carinho, sem os quais eu não teria chegado até aqui.

Á minha namorada, Lea Marques de Jesus, pela compreensão, paciência e amor.

À Profª. Dra. Regina Borges de Araújo pela amizade, compreensão, paciência, orientação, apoio e credibilidade depositados em mim.

Aos grandes amigos, especialmente ao BB, BG, Pia Fino, Japulha, Laffrancha, Fernando, Taci, Pentium, Baby, Matheus, Chyrstymmas, Gislaine, Xorume, Clô, Biriguey e Pureza, pelos momentos de muita alegria, descontração e palhaçadas e companheirismo.

E é claro, ao AEL, que nunca reclamou em me dar uma mão, esteve presente comigo nos mais difíceis momentos, me acompanhou dia e noite na busca pelos meus objetivos, e raramente me passou raiva. Afinal sempre cuidei dele com carinho e upgrades.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRÁFICOS

LISTA DE ABREVIAÇÕES E SIGLAS

3D	*Three-dimensional*
API	*Application Programming Interface*
AC	*Adaptador de Conteúdo*
BIFS	*Binary Format for Scene*
CDMA	*Code Division Multiple Access*
CRT	*Cathode Ray Tube*
CPU	*Central Process Unit*
CGI	*computer generated imagery*
DLL	*Dynamic Link Libraries*
DMIF	*Delivery Multimedia Integration Framework*
EAI	*External Authoring Interface*
ES	*Elementary Stream*
ED	*Especificador de Desempenho*
FPS	*Frames per Second*
FPSs	*First Person Shooters*
GPS	*Global Position System*
GPRS	*General Packet Radio Service*
GPU	*Graphic Processor Unit*
GUI	*Graphic User Interface*
GSM	*Groupe Speciale Mobile/Global System for Mobile Communications*
HTTP	*Hyper Text Transfer Protocol*
IrDA	*Infrared Data Association*
IP	*Internet Protocol*
IA	*Inteligência Artificial*
ISDN	*Integrated Service Digital Network*
JVM	*Java virtual Machine*
Kbps	*Kilobits per second*
LAN	*Local Area Network*
LCD	*Liquid Crystal Display*
MAI	*Multi-user Application Interface*
MBK	*Mutech Bookkeeper*
MCM	*Mutech Channel Manager*
MMH	*MUTech Message Handler*
MMGAME	*Multi-User MPEG-4 Game Adaptation Manager*
MPEG	*Moving Picture Experts Group*
MPEG-4 MU	*MPEG-4 Multi-user*
MPEG-J	*MPEG-4 Java*
MR	*Monitor de Recursos*
MSC	*Mutech Session Controller*
MU	*Multiusuário*
MUD	*Multi-user Dungeons*
NPC	*Non Personal Character*
NPM	*Non Persistent Multiplayer*
OD	*Object Descriptor*
PC	*Personal Computer*

PDA	*Personal Digital Assistant*
PM	*Persistent Multiplayer*
QPS	*Quadros por Segundo*
Qos	*Quality of Service*
QdS	*Qualidade de Serviço*
RAM	*Read Access Memory*
RPG	*Role Playing Game*
RTS	*Real Time Strategy*
RV	*Realidade Virtual*
SDRV	*Sistemas Distribuídos de Realidade Virtual*
SL	*Sync Layer*
STB	*Set-Top-Box*
TCP	*Transmission Control Protocol*
TDMA	*Time Division Multiple Access*
TTL	*Time To Live*
UDP	*User Datagram Protocol*
VRML	*Virtual Reality Modeling Language*
VO	*Video Object*
WAN	*Wide Area Network*
WG11	*Work Group 11*
WWW	*World Wide Web*
X3D	*Extensible 3D*
XML	*Extensible Markup Language*

RESUMO

Com a maior disseminação e confiabilidade das redes de comunicação sem fio, e a emergência de dispositivos com o potencial de processamento e de comunicação cada vez maior, aplicações antes restritas aos computadores pessoais estão sendo vislumbradas em dispositivos tão heterogêneos quanto relógios com localizadores, geladeiras que acessam a Internet, celulares, PDAs (computador digital pessoal), set-top-boxes e consoles de jogos. A integração dessa miríade de dispositivos e de tecnologias de redes, com capabilidades variadas, exige, hoje, dos programadores e projetistas de software, atenção especial na construção de aplicações - especialmente quando essas aplicações são compartilhadas por usuários em diferentes dispositivos, cada um produzindo e consumindo informações de acordo com a capacidade do dispositivo. Neste sentido, a adaptação da aplicação, que permite a um software reagir a variações nos recursos utilizados por ele, permitindo melhor adequação de suas funções e dados para uma determinada configuração, é um processo importante e está sendo, cada vez mais, considerada como parte importante de sistemas que atuam em ambientes heterogêneos de computação e de rede. Uma grande quantidade de trabalhos tem sido devotada à adaptação da entrega de formatos tradicionais de multimídia, como texto, imagens, áudio e vídeo, entretanto, pouca atenção tem sido dada ao conteúdo digital 3D, primeiro por causa da complexidade das questões envolvidas na adaptação de aplicações 3D, maior do que as de outras mídias, e também pelo fato que oportunidades verdadeiras de marketing para conteúdo 3D estão apenas começando a surgir. Mais ainda, iniciativas que promovem a padronização da adaptação devem ser consideradas, para que seja promovida a interoperabilidade em futuros mecanismos de adaptação. Este trabalho investigou o padrão MPEG-4 e sua extensão MPEG-J que tratam, de forma superficial ainda, da questão da adaptação, e propôs um *framework* de adaptação de aplicações 3D, denominado MMGAME, mais especificamente de jogos multiusuário em que, elementos que compõem uma aplicação de jogos possam ser adaptados em função de flutuações nos recursos do dispositivo que executa a aplicação e da rede onde este dispositivo está conectado.

ABSTRACT

With the growing dissemination and reliability of wireless networks and the emergence of devices with more and more processing and communication power, applications up to now restricted to PCs are being envisaged to run on devices as heterogeneous as wrist clocks with GPS locators, refrigerators with internet access, up to mobile phones, PDAs, set-top-boxes and game consoles. The integration of this myriad of devices and network technologies, with different capabilities, demand special attention from the software programmers and designers– specially when these applications are shared among multiple users from different devices, each one producing and consuming information according to each device capacity. In this sense, application adaptation, which allows a software to react upon the resources variations used, is an important process towards fitting the application to a certain device configuration. This adaptation is being considered as an important part of the systems that act upon heterogeneous processing and communication environment. Therefore, a large amount of work has been done in the traditional multimedia adaptation, such as text, images, audio and video, less attention has been focused on 3D digital content – first because of the complexity involved in the 3D applications adaptation issues, and also because true marketing opportunities for 3D graphics in heterogeneous devices have just began to emerge. Moreover, initiatives that promote adaptation standardization must be considered, so that interoperability among adaptation mechanisms can become a reality. This work investigated the MPEG-4 standard and its extension MPEG-J to handle adaptation issues, and proposes an adaptation framework for 3D multi-user game applications, entitle MMGAME. With this framework, elements composing a game application can be adapted according to devices resources variations where the application is running as well as in the network where is connected.

1 INTRODUÇÃO

Nos jogos 3D multiusuário - MU, múltiplos usuários compartilham, de forma síncrona, um mesmo ambiente virtual tridimensional. Os jogos 3D MU são exemplos relevantes de ambientes virtuais tridimensionais multiusuário, pois exigem desempenho e sincronização rígida de dados e de controle, além de ser uma das classes de aplicações de maior apelo econômico a serem oferecidas em dispositivos móveis. Até recentemente, somente computadores pessoais eram usados para jogos on-line, mas isso mudou radicalmente com a capacidade da nova geração de consoles para executar jogos em rede, além da evolução da computação móvel, resultado do surgimento de novos dispositivos leves de maior poder computacional e do crescimento das redes de comunicação sem fio. Apesar da evolução da computação móvel oferecer aos usuários acesso a jogos compartilhados de qualquer lugar e a qualquer momento, vários desafios devem ser superados, entre eles o alto custo das linhas de conexão sem fio, a alta latência e as interrupções freqüentes. Mais ainda, na computação móvel, uma aplicação segue o usuário, mas o seu modelo não consegue, de modo transparente e sem o auxílio do usuário, ajustar-se a eventuais mudanças no ambiente. Para isso, mecanismos de adaptação devem ser projetados que se adaptam às características, num determinado momento, do dispositivo, da rede e do usuário.

Na literatura, sistemas que tratam da adaptação de aplicações multimídia, mais especificamente vídeo e áudio, são vários [Coda00], [Cadmium00], [Fox97]. Alguns estudos tratam da adaptação 3D apenas no contexto da interface [Eisenstein00], porém pouca atenção tem sido focada no conteúdo digital 3D [Martim02].

O padrão MPEG-4 oferece uma solução independente de dispositivos e de rede para construção das mais diversas aplicações multimídia, entre elas vídeo interativo, filmes com várias faixas de áudio, complexas apresentações e aplicações 3D. Uma extensão ao padrão MPEG-4, denominada MPEG-4 multiusuário, define o suporte ao compartilhamento de um mesmo ambiente 3D entre múltiplos participantes. Outra extensão ao padrão MPEG-4, MPEG-J, permite a inserção de código Java na aplicação, o que aumenta a funcionalidade, controle e eficiência, permitindo a inserção de mecanismos de adaptação. Esta solução, entretanto, deixa ao projetista da aplicação a responsabilidade de projetar todas as facetas da adaptação, o que pode resultar em soluções apenas parciais.

Este trabalho compreendeu a análise dos parâmetros a serem considerados para a adaptação de jogos 3D, tanto de seu conteúdo quanto de sua funcionalidade, necessários para o uso em dispositivos e redes heterogêneos, o que resultou na concepção de um

framework de auxílio para implementação de mecanismos de adaptação em aplicações MPEG-J no o uso em Jogos 3D, ou qualquer outra aplicação 3D que necessite de interação em tempo real.

Este *framework,* denominado MMGAME, permite ao programador a inserção de mecanismos de adaptação para satisfazer a necessidade de adequação à variação de recursos do dispositivo, da rede e da aplicação. Selecionando os melhores formatos para a execução da aplicação, garantindo a independência de dispositivo, juntamente com o monitoramento dos recursos, reagindo apropriadamente às suas mudanças e mantendo as preferências do usuário, garante-se à aplicação o desempenho necessário para a execução de Jogos 3D em tempo real.

Para facilitar a inserção dos mecanismos de adaptação, o *framework* permite que sejam capturados, em tempo real, todos os tipos e formatos de dados que a aplicação utiliza, bem como as possibilidades de adaptação que o determinado dado suporta, proporcionando facilidade e liberdade tanto para a programação da adaptação quanto para a programação do próprio Jogo 3D.

Mecanismos em tempo real são disparados, calculando o impacto de cada mecanismo de adaptação na configuração específica do usuário, assegurando que a política de adaptação será válida para todas as plataformas.

1.1 Motivação

Existe, hoje, uma grande diversidade em termos de dispositivos e redes heterogêneas disponibilizadas para a sociedade. Jogos multiusuário, por exemplo, têm grande potencial de tornarem-se aplicações de larga escala em dispositivos que vão de celulares, a PDAs, set-top-boxes e até a consoles embarcadas em utensílios domésticos, como uma geladeira que acessa a Internet, inclusive.

Frente à emergência de aplicações executadas em dispositivos e redes heterogêneas, que se movem junto com o usuário, a adaptação da aplicação, que permite a um software reagir a variações nos recursos utilizados por ele, permitindo melhor adequação de suas funções e dados para uma determinada configuração, é um processo importante e está sendo, cada vez mais, considerada como parte importante de sistemas que atuam em ambientes heterogêneos de computação e de rede. Entretanto, para que as aplicações se adéquem às variações de recursos, elas necessitam ter ciência do contexto ao seu redor e como devem reagir a mudanças possíveis de ocorrer, mantendo sua funcionalidade, mesmo

que reduzida, para promover seu serviço ao usuário. Tal comportamento é obtido através da incorporação da adaptação dinâmica (em tempo-real) à aplicação.

As aplicações que não sofrem adaptação, em geral, atendem aos usuários de forma homogênea, não garantindo a qualidade do serviço durante toda a sua execução.

Jogos são exemplos de aplicações encontradas na maioria das plataformas, como set-top-boxes, celulares, PCs, PDAs etc. Além de possuírem enorme apelo comercial, o que estimula a produção em massa deste tipo de aplicações, jogos são exemplos de aplicações que necessitam de garantia de recursos para o devido funcionamento, tornando-as candidatas em potencial para o uso de adaptação.

Um relatório divulgado pelo NPD Group [NPD02], no início de 2003, indica que dos 10 produtos de software mais vendidos para o computador de 2002, três são jogos, perdendo apenas para produtos de gerência empresarial.

A indústria de consoles de jogos rendeu, em 2002, mais de 10,5 bilhões de dólares [NPDb03], somando vendas de hardware, software, e periféricos, somente nos EUA. Para efeito de comparação, a indústria de Hollywood, em 2001, rendeu 8.41 bilhões de dólares [Bjork02]. O mercado mundial do entretenimento digital é estimado em 18 bilhões de dólares com perspectivas de crescimento de até 26.7 bilhões de dólares até o ano de 2005. In-Start/MDR estima que a indústria de jogos sem fio deva crescer até 2.8$ bilhões no mundo inteiro até 2005[IDGA02].

O forte apelo comercial leva os desenvolvedores a propor soluções que promovem a comunicação entre usuários de diversos dispositivos para competições em rede. Esta abordagem maximiza os lucros, ao mesmo tempo em que fornece aos usuários várias opções de entretenimento. Empresas como a SquareSoft [Square03] e TerraForge[Terra03] possuem redes que interligam vários usuários de dispositivos diferentes em um mesmo jogo.

A necessidade eminente de manter o desempenho em dispositivos suscetíveis a variação de recursos durante a execução de jogos multiusuário 3D motivou este trabalho na construção de uma solução para garantir este desempenho com o uso da adaptação.

1.2 Estrutura da Dissertação

No segundo capítulo introduz-se o conceito de Jogos, suas variações quanto ao tipo, requisitos técnicos e modelos de comunicação. Também são discutidas as características particulares apresentadas pelos jogos em relação a outras aplicações 3D, principalmente quanto ao requisito de interação em tempo real. São identificados os elementos que impactam

no desempenho e que podem sofrer adaptação. No terceiro capítulo são descritas as principais questões relacionadas à adaptação, e suas técnicas, encontradas na literatura. No quarto capítulo apresenta-se o padrão MPEG-4 e MPEG-J, e justifica a escolha deste padrão como base para a construção de um *framework* de suporte à adaptação de mídia 3D para jogos multiusuário. No quinto capítulo discute-se o projeto, a especificação e a implementação parcial de um framework de suporte a adaptação de jogos 3D multiusuário para uso em ambientes heterogêneos. No sexto capítulo são apresentados: o resultado do trabalho; as conclusões; trabalhos futuros e referências bibliográficas.

2 JOGOS: APLICAÇÕES GRÁFICAS 3D EM TEMPO-REAL

De acordo com [Sabadello01] os jogos de computador têm evoluído por um longo tempo, quase ao início do desenvolvimento de software. Hoje, os jogos são populares entre várias faixas etárias e gêneros oferecendo, dentre outros recursos, gráficos 3D e avançada inteligência artificial. Este capítulo descreve os tipos de jogos existentes e identificam os elementos, os quais usados na construção dos jogos, os requisitos básicos, a aceleração do hardware usada para atender esses requisitos e as configurações atualmente utilizadas para prover o desempenho necessário nestas aplicações. O objetivo é mostrar os parâmetros que podem ser adaptados em função de flutuações nos recursos de software, hardware e de rede em diferentes dispositivos que participam de jogos multiusuários.

O termo jogos refere-se a passatempos, empregados com a finalidade de entreter o usuário (jogador). Os jogos necessitam de um conjunto de regras e, se competitivo, apresentar alguma forma de medir qual a possibilidade de um jogador ganhar, de acordo com o objetivo principal. Adicionalmente, a maioria dos jogos requer alguma habilidade por parte dos jogadores.

Em uma visão macro, os jogos em geral se dividem em tabuleiro, cartas, atléticos, infantis e de computadores (videogames e jogos digitais) [Svedjedal02]**.** Jogos digitais são utilizados em vários tipos de dispositivos, como *arcades*, consoles, PCs, e dispositivos de mão. *Arcades,* também chamadas de fliperama, em que o jogador paga para jogar apenas uma partida, promove grande qualidade sonora e *feedback* real para o usuário, por meio de uso de cabines, porém são feitos para uma especifica placa controladora gráfica. Consoles são dispositivos computacionais desenvolvidos especialmente, mas não exclusivamente, para acomodar complexas aplicações de jogos, exemplos mais recentes são: Sony Playstation 2™, Nintendo GameCube™ e Microsoft XBox™. Jogos de PCs usam o computador, e todos os recursos nele contido, para proporcionar uma melhor experiência ao jogador, os jogos são retro-compativeis, diferentes da maioria dos consoles, onde um jogo é desenvolvido especialmente para um dispositivo em particular e não funciona em outra plataforma, mesmo superior. No entanto, para todas as plataformas e dispositivos usa-se o mesmo processo de criação e desenvolvimento, doravante quando utilizado o termo jogos de computador, não se refere somente a PCs, mas também a videogames (consoles) ou qualquer forma de jogo digital.

De acordo com [Bettner01], os jogos digitais são jogados contra, ou moderados por um computador. Em casos raros, eles podem ser jogados entre computadores.

Existem, hoje, diversas categorias de jogos digitais. *Wolf* [Wolf00] classificou-os da seguinte maneira:

- Ação/Tiros
- Aventura/ Ficção Interativa
- Simulação
- RPG (Role Playing Games)
- Estratégia
- Esportes
- Lutas
- Tabuleiros/ Puzzles

Os tipos de jogos, em maior detalhe, são descritos a seguir.

2.1 Classificação de Jogos

De acordo com [Laird03], não há uma linha para classificação dos tipos de jogos, sendo que muitos deles apresentam múltiplos gêneros. Existem jogos de estratégia (um exemplo é o *Dungeon Keeper*) que permitem a humanos realizar ações como em jogos de ação. Também há jogos de ação em que se devem gerenciar recursos e múltiplas unidades, característica de um jogo de estratégia, como é o caso do Giants –Citzen Cabuto. Apesar da existência de vários gêneros de jogos, Laird e Lent [Laird03] e também [Wolf00] apresentam as definições desses gêneros de jogos. Filtrando apenas os jogos em tempo real, temos a seguinte divisão:

2.1.1 Ação

Os jogos de ação são um dos gêneros mais populares de jogos e envolve o jogador humano controlando um personagem num ambiente virtual, caracterizado por possuir vários elementos de ação, como correr, saltar, combater, fugir, etc. Esses jogos variam na perspectiva que o jogador tem de seus personagens, a qual pode ser: primeira pessoa, em que o jogador enxerga apenas o campo de visão de seu personagem; terceira pessoa, onde é

visualizado o personagem e a cena ao seu redor. Exemplos populares incluem *Quake III*, *Half-Life*, *Unreal Tournament 2003, MAFIA-The City of Lost Heaven* e *Halo.*

2.1.2 Role Playing Games(RPG)

Em RPG´s, um jogador controla diferentes tipos de personagens tais como guerreiros, magos ou mesmo um ladrão. O jogador questiona, coleciona e vende itens, luta com monstros e melhora as capacidades do seu personagem (tais como força, magia, rapidez, etc.), tudo num mundo virtual estendido. Exemplos desse tipo de jogo incluem, *Baldur's Gate, Diablo, Final Fantasy X,* entre outros. Recentemente os RPG´s multiusuário em massa, (*massive multiplayer role-playing games*) têm sido introduzidos, milhares de pessoas jogam e interagem num mesmo mundo, como exemplo temos *Ultima Online, Everquest,* e *Asheron's Call*.

2.1.3 Estratégia em Tempo Real

Em jogos de estratégia, os jogadores controlam várias unidades como tanques de guerra ou unidades militares, realizando as batalhas através da visão de todo o campo de batalha, contra um ou mais oponentes. Os jogos de estratégia incluem diferentes tipos de batalha: histórica (*Close Combat, Age of Empires*), realidades alternativas (*Command and Conquer*), futuro da ficção (*Starcraft*), e lendárias (*Warcraft*). O jogador é colocado em face de problemas como alocação de recursos, escalonamento da produção e organização de defesas e ataques [Davis 1999]. Existem jogos de estratégia baseados em turnos, como *Civilization III,* que não são disputados em tempo real, o jogador possui um tempo para premeditar com cuidado suas estratégias e ataques.

2.1.4 Esportes

Os jogos de esportes cobrem as atividades tradicionais utilizando times, como futebol ou baseball, a esportes individuais como os olímpicos. Jogos de times têm o jogador como uma combinação de técnico e jogadores (personagens) ao mesmo tempo, tais como futebol americano *(NFL 2003)*, [Whatley 1999], basquete (EA NBA 2003), futebol(FIFA 2003) entre outros.

2.1.5 Jogos de Corrida *(Racing games)*

Os jogos de corrida englobam os seguintes tipos de veículos: carros, caminhões, motos, barcos, aviões, skates, entre outros. Alguns dos primeiros jogos de computador envolveram corridas de carro, evitando obstáculos e tentando vencer outros participantes. Em jogos de corrida o computador oferece uma simulação de esporte em primeira ou terceira pessoa. O jogador controla um personagem num jogo cujos competidores podem ser outros jogadores ou mesmo computadores.

Baseado em todos os gêneros de jogos acima se obtêm os principais requisitos técnicos de jogos, características necessárias e que devem ser atendidas não apenas na construção, mas também no gerenciamento do jogo. Estes requisitos são descritos a seguir.

2.2 Requisitos Técnicos de Jogos Digitais Multiusuário

A seguir estão descritos os requisitos técnicos de jogos multiusuário, incluindo tanto jogos peer-to-peer quanto cliente servidor.

2.2.1 Consistência

O requisito de consistência implica, mesmo que cada participante compute sua própria visão local do jogo, na existência de uma técnica de sincronismo distribuído necessária para garantir que os participantes computem estados do jogo tão similares quanto possível. As entidades devem também se recuperar de informações perdidas ou atrasadas [daRosa99].

2.2.2 Sincronismo

Jogos distribuídos os quais utilizam a Internet precisam ainda resolver o problema do sincronismo. Como os atrasos da rede são diferentes para qualquer par de participantes na Internet, mecanismos de sincronização precisam ser introduzidos para permitir que pacotes enviados “ao mesmo tempo” sejam processados “juntos” por qualquer participante. Outro aspecto importante da sincronização é que todos os participantes devem exibir o mesmo estado global do jogo em certo instante de tempo [daRosa99].

2.2.3 Escalabilidade

Escalabilidade é a propriedade na qual um sistema pode crescer, com novas partes se necessário, sem a necessidade de reescrever sessões inteiras a cada vez que algum componente é adicionado, bem como acomodar uma grande quantidade de usuários na medida em que entram no ambiente. A escalabilidade somente é vista em jogos cliente-servidor.

2.2.4 Persistência

Jogos persistentes são caracterizados por salvar o contexto no qual o usuário está submetido, permitindo ao mesmo, recuperar o estado anterior em um momento futuro. A persistência é usada em jogos multiusuário massivos, onde milhares de usuários se interagem em um mundo virtual grande e complexo. O estado do usuário compreende o de seu personagem, seus objetos, em alguns casos, até a residência no mundo virtual.

2.2.5 Interatividade

Interação são descrições, ilustrações, representações ou manifestações de ações engatilhadas por jogadores ou pelo ambiente de jogo.

Interatividade é o termo usado para medir a influência do usuário. Os jogos devem apresentar suporte para grandes mundos com um alto grau de interatividade com o personagem, ambiente, NPCs (Personagens Controlados por Computador) e outros usuários.

Se há uma palavra com muitos significados nos jogos, essa é "interatividade". Em jogos de aventura, "interatividade" significa ser capaz de manipular o ambiente. Em jogos de ação, significa ter controle sobre o personagem principal. Alguns jogos combinam ambos os tipos de interatividade com excelentes resultados[Kent01]. Uma interatividade rica significa o uso de formas mais naturais da utilização da interação.

2.2.6 Extensibilidade

Uma aplicação extensível é aquela que apresenta várias formas de comunicação, tornando-se assim viável a diferentes modelos de comunicação, como por exemplo, *peer-to-peer*, cliente/servidor e o modelo rede de servidores [Aspects of net MCGs].

A extensibilidade para outras plataformas e outros SO's também são contempladas neste requisito, visto que os jogos multiusuário podem ser jogados por qualquer usuário, em qualquer rede e em qualquer dispositivo.

2.2.7 Segurança

A segurança *online* tem se tornado muito importante na indústria do entretenimento. Em jogos multiusuário métodos de segurança devem ser utilizados, destacando: bloqueio de acesso a usuários não autorizados; técnicas que impeçam que um determinado usuário altere o estado do jogo para ganho próprio.

Em [Kirmse97] dois tipos de objetivos de segurança foram reconhecidos para jogos *online*: Proteção de informações sensíveis, por exemplo, número do cartão de crédito, e o oferecimento de um campo de jogo limpo, sem trapaças. A segurança também envolve outros fatores que são:

Proteção a servidores de jogos, devido à invasão de trapaceiros ou *hackers*, podendo causar problemas de autenticação de outros usuários e até atrasos no andamento do jogo;

- Proteção dos clientes, pois os *hackers* ou trapaceiros podem agir como um usuário válido, alterando e destruindo todas as informações e perfis de um usuário invadido;
- Negação de um serviço, ou "denial-of-service" é quando os trapaceiros impedem usuários legais de acessarem os jogos ou suas informações;
- O servidor passa a controlar as ações dos clientes, por exemplo, exigindo com que o cliente se atualize, fazendo assim um *download* da atualização, que sem o cliente perceber, traz consigo *trojans* ou outros arquivos inválidos;
- Os trapaceiros ainda podem modificar o software cliente para que ele faça coisas inesperadas, como por exemplo, permitir invasão de outros usuários, e alterar o estado do personagem ilegalmente.

Para esses problemas, [Kato & Zojonc] oferecem soluções como sistema de detecção de intrusos, rígidos métodos de controle de autenticação, assinaturas digitais, melhores e mais seguros métodos de autenticação cliente/servidor, com uso de criptografia,

armazenamento de informações dos clientes em servidores seguros e ferramentas de monitoramento do sistema.

2.2.8 Latência

A Latência indica o atraso que ocorre quando uma mensagem segue de uma origem para um destino. Também a variância da latência sobre um determinado tempo, isto é, *Jitter*, é outra característica que afeta uma aplicação interativa. Para sistemas interativos em tempo real, como os jogos multiusuário, a latência entre 0.1 e 0.5 segundos é aceitável. Por exemplo, o padrão DIS (*Distributed Interactive Simulation*) especifica que a latência da rede deve ser menor que 100 ms [Neyland97].

O limiar de quando a latência fica inconveniente para o usuário depende do tipo de aplicação. Em um jogo de estratégia em tempo-real (RTS), uma latência mais alta (até mesmo 500 ms) é aceitável contanto que permaneça estática [Bettner01]. Por outro lado, jogos que requerem muita atenção do usuário como os atiradores de primeira pessoa (FPSs) precisam de uma latência de no máximo 250 ms.

Devemos considerar também latência no processamento, podendo haver usuários em dispositivos de uma menor capacidade de computação, jogando com usuários em super computadores com uma alta capacidade de processamento.

2.2.9 Largura de Banda

Largura de banda refere-se à capacidade de transmissão de uma linha de comunicação, tal como uma rede. É definido como uma proporção da quantidade de dados transmitidos por unidade de tempo. Em redes WANs, a largura de banda varia de dezenas de kbps até 44.7 Mbps dependendo da rede. Em LANs, a largura de banda apresenta-se muito maior variando de 10 Mbps até 10 Gbps. Entretanto, LANs tem um tamanho limitado e suportam um número limitado de usuários, enquanto que WANs permitem conexões globais.

Os requisitos de largura de banda dependem muito do número e distribuição dos usuários e é claro, dependem também das técnicas de transmissão. A transmissão pode ser:

- *Broadcast*, onde as mensagens são enviadas a todos os participantes [Macedonia97]. Obviamente, essa técnica pode gerar problemas caso o número de usuários venha a crescer;

- *Unicast*, onde a comunicação é feita diretamente entre dois *hosts*, e a mensagem é enviada diretamente. Porém, uma vez que a maioria das mensagens é enviada a múltiplos receptores, o *Unicast* desperdiça largura de banda;
- *Multicasting*, onde a comunicação é feita entre uma origem e um conjunto de receptores que pertencem a um mesmo grupo *multicast*. Essa técnica permite entrada em grupos que os interessam, ou seja, fazer parte de um grupo *multicast* [Macedonia95].

A largura de banda pode variar dependendo da rede utilizada. Com base nesse argumento, os jogos multiusuários devem possuir um suporte para tratamento da largura de banda. Podemos encontrar usuários em uma rede *wireless*, por exemplo, com uma largura de banda de até 256kbps, jogando com outros usuários que estejam em uma LAN, com largura de banda disponível de até 1 Gbps. O fluxo de informação deve ser constante nas duas redes para impedir altas latências e falhas na comunicação.

2.2.10 QoS

Jogos que utilizam a infra-estrutura existente na Internet enfrentam a falta de suporte a requisitos de *Quality of Service* (QoS). Em redes de comunicação de dados, QoS implica em um compromisso dos dois lados: o usuário deve especificar e controlar com rigor seu tráfego, e o provedor de serviço de rede fornecer garantias de QoS, descartando o excesso de demanda por parte do usuário. A Internet oferece o outro extremo ao aceitar qualquer demanda, dando uma QoS na base do “melhor esforço” e confiando que os usuários serão comportados para garantir justiça na alocação de recursos de rede [daRosa99].

2.3 Jogos Multiusuário

Jogos multiplayer, ou multiusuário, são jogos executados por computador, ou outros dispositivos, onde vários indivíduos jogam simultaneamente e onde a interação de um jogador afeta o modo que o jogo trata os outros jogadores.

Se um jogo é multiusuário, há várias alternativas: um grupo pequeno de pessoas jogando na mesma máquina ou dispositivo todos na mesma sala; várias pessoas jogando individualmente em suas estações por meio de uma LAN; várias pessoas, podendo até chegar a milhares, jogando em uma rede pública de comunicação. A primeira categoria de jogos multiusuário tende a ser somente uma extensão do jogo *single-player (monousuário)*,

pois não usam nenhum mecanismo de comunicação. O número de jogadores é limitado para poupar o processamento do dispositivo, tipicamente, não ultrapassa quatro usuários. Em redes locais, a quantidade de usuários pode chegar a dezenas, e em redes públicas até milhares.

Para que haja interação em um jogo multiusuário deve haver uma sensação de presença repassada de cada jogador para os demais, permitindo ações, com ou contra os adversários e companheiros. Outra característica considerada essencial para um jogo multiusuário é a capacidade do envio de mensagens em uma linguagem natural, podendo ser em forma textual ou sonora e, futuramente, serem representadas em vídeo. Não menos importante, os jogos multiusuário devem possuir um conjunto de regras e um objetivo definido para a atribuição de uma vitória e uma recompensa (mesmo que seja apenas pessoal), bem como em situações que acontece a derrota de um jogador e suas conseqüências. [Caroll03]

A capacidade de proporcionar a interação multiusuário é uma característica importante em grande parte dos jogos de computador desenvolvidos hoje, sendo altamente considerada no momento da compra de um jogo. Não importa qual o tipo de jogo, jogar contra oponentes humanos reais é mais desafiador e motivante. Porém, a realização de tais jogos requer a comunicação entre os jogadores, e os programadores têm que lidar com algumas limitações causadas pela rede. [Sabadello01]

Para que haja a consistência em um ambiente de jogos multiusuário é necessária a comunicação entre os clientes os quais participam da mesma partida. Clientes de jogos em rede periodicamente emitem eventos para serem recebidos por um subconjunto de outros clientes, existem duas abordagens para a transmissão de eventos, o modelo cliente-servidor e o modelo peer-to-peer.

No cliente-servidor, os jogadores não trocam os dados diretamente uns com os outros, mas enviam e recebem toda a informação solicitada por meio de um servidor ou servidores, como ilustra a Figura 1. Servidores de jogos são mais complexos que servidores de outros serviços, como http, cada informação enviada pelos clientes deve ser comparada, por motivos de consistência. A vantagem dessa arquitetura é que o jogador precisa manter apenas uma simples conexão com o servidor não importando a quantidade de usuários conectados, isso reduz o tráfego de mensagens na rede uma vez que os usuários as enviam apenas uma vez (para o servidor). A desvantagem acontece quando há um aumento no número de conexões de usuários, o servidor começa a formar um gargalo, impedindo

conexões e possivelmente aumentando a latência na distribuição dos eventos. Outra desvantagem é quando há uma falha no servidor, todos os usuários perdem seu estado no jogo e não há como prosseguir a partida. Para a solução de ambos os casos recomenda-se o uso de replicação de estados entre servidores e a distribuição de carga. Porém com a inclusão destas características, o serviço passa a ter um custo elevado forçando o provedor a cobrar pelo uso da infra-estrutura. [Fitzek02]

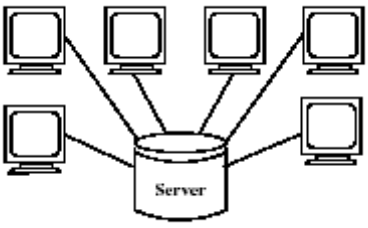

Figura 1 Modelo de distribuição de eventos Cliente-Servidor

O modelo peer-to-peer tem como principal característica as trocas de mensagens diretamente entre os usuários conectados. Com isso, não há necessidade de um servidor central para as mensagens e muito menos para alguma influência no jogo. Toda comunicação e cálculos são feitos pelos próprios usuários e isso remove o gargalo da rede, porém são limitados a grupos pequenos de usuários, tipicamente 16[Bauer02]. Uma ilustração do modelo se encontra na Figura 2.

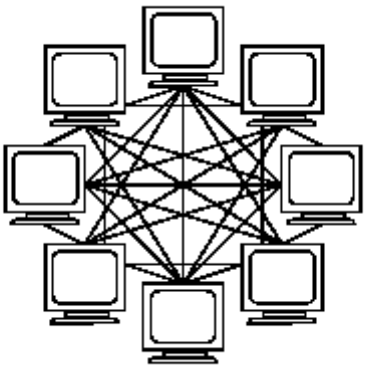

Figura 2 Modelo de distribuição de eventos peer-to-peer

O modelo *peer-to-peer* apresenta algumas vantagens:

- Maior agilidade com relação ao modelo cliente/servidor, pois as mensagens são entregues diretamente de usuário a usuário evitando o gargalo na distribuição das mensagens
- Caso um usuário perca a conexão os outros usuários permanecem inalterados;

Mas também apresentam desvantagens que são:

- Não apresentam muita segurança, tornando-se vulneráveis a trapaceiros e *hackers*.
- Numa arquitetura puramente peer-to-peer, ou seja, sem a presença de um ponto central, o tráfego na rede é muito maior que no cliente-servidor, pois a quantidade de mensagens na rede cresce pelo quadrado do número de jogadores.
- Falta de escalabilidade, devido ao crescente número de mensagens à medida que se cresce o número de usuários.

Um problema de proporção maior é quando há a necessidade de sincronização dos eventos em uma aplicação peer-to-peer, onde todos os jogadores devem realizar o mesmo comando de certo jogador ao mesmo tempo, como nos casos de jogos de estratégia em tempo real, como *Age of Mithology* e *Warcraft III*. Sendo assim a performance no tratamento de mensagens para todo o grupo é reduzida ao nível do peer mais fraco, caso este possua uma alta latência ou baixo processamento, os demais usuários também enfrentaram os mesmos retardos, comumente este retardo é chamado de *lag*.

2.4 Pipeline3D

Segundo Crawford [Crawford97] os elementos fundamentais comuns a quaisquer jogos de computador são representação, interação, desafios e recompensas. A representação indica como o jogo representa um subconjunto da realidade, a interação descreve como, o que, e quando, a mudança do subconjunto de realidade acontece e a maneira na qual ela será representada. O desafio de um jogo surge automaticamente quando se usa interatividade, o jogador é persuadido a atingir um objetivo. Obstáculos devem ser estipulados para prevenir acesso fácil ao mesmo, caso este desafio consistir de um agente inteligente que reage diferentemente dependendo da reação do usuário, diz-se ser controlado por uma inteligência artificial, este agente é denominado NPC (Non Personal caracter; Personagem não humano). Recompensas são atribuídas ao jogador no decorrer do jogo, ajudando-o a completar mais desafios, ou auxiliando em partidas disputadas entre outros jogadores. O sistema de recompensa depende muito do tipo do jogo, podendo ser dinheiro, itens, segredos,

etc. O ponto foque do projeto desenvolvido é a representação do mundo em que o jogo é ambientado, como são realizadas, do que são compostas e o que é necessário para acelerar a representação em tempo real, portanto, esta seção detalha a representação de um jogo 3D.

A representação gráfica em três dimensões é caracterizada pela construção de um modelo geométrico usando as três coordenadas cartesianas X, Y e Z - altura, largura e profundidade, respectivamente, para posterior exibição em uma tela de computador, na forma 2D. Um sistema gráfico 3D é organizado, de forma simplificada, em três componentes: especificação da cena, renderização e exibição [House96].

Ainda segundo [House96], a especificação da cena constitui-se de uma representação interna, em três dimensões, da imagem virtual a ser exibida, a qual será construída a partir de uma interface visual interativa ou por meio de primitivas. A cena necessita conter a descrição da geometria (pontos, vetores de orientação, linhas e polígonos), o tipo do material (atributo de um objeto que define como ele reflete e refrate luzes) e a iluminação (fontes de luzes e simulação da propagação das mesmas).

A renderização é o processo de transformação da descrição de uma cena 3D em uma imagem 2D, que será exibida na tela. A renderização de uma cena realista de forma eficiente é um desafio. O renderizador é o motor (*engine*) que impulsiona o processo de transformação de uma cena 3D em uma imagem 2D. O processo de renderização é complementado ainda estabelecendo-se a intensidade da superfície dos objetos de acordo com as condições de iluminação na cena (algoritmos *shader*), tais como intensidade e posição das fontes de luz e iluminação geral da cena, grau de transparência e suavidade das superfícies, de acordo com as características da superfície dos objetos. Procedimentos podem ser aplicados para gerar regiões de sombra corretas para a cena [Hearn97]. Wolberg [Wolberg90] resume os passos básicos para o processo de renderização, a saber:

- Orientar a cena 3D para a posição da câmera virtual (ponto de vista do jogador)
- Projetar os pontos da cena 3D sobre um plano virtual 2D
- Decidir que superfícies projetadas no plano virtual serão visíveis
- Fixar um conjunto de amostras da cena 3D
- Determinar a cor transmitida ou refletida para a tela de cada ponto visível no conjunto de amostras, por meio dos algoritmos *shader*.
- Construir uma imagem digital a partir de uma filtragem e amostragem dos pontos usados nos algoritmos de *shader*.

- Armazenar a imagem em um buffer de quadros de cena *(frame buffer)* para exibição.

O *frame buffer* é parte da memória de vídeo usada para representar exatamente os pontos exibidos na tela. Caso estes pontos mudem, a imagem exibida na tela também muda.

A renderização, especificamente em jogos 3D, segue basicamente os mesmos passos descritos acima. Porém, modificações devem ser realizadas para que o processo resulte em altas taxas de exibição de buffers de quadros de cena *(frame buffers)* por segundo - esta taxa é conhecida como quadros por segundo (QPS). A necessidade de altas taxas de QPS justifica-se pois jogos são aplicações que necessitam de interação em tempo real e conseqüentemente de alto desempenho.

2.5 Jogos 3D – Interação em tempo real

Existem diversas áreas da computação gráfica que utilizam a representação 3D como solução para problemas específicos. Estas soluções diferem-se na qualidade da imagem, taxa de quadros atingida em tempo real e custo [Helman96]. As mais relevantes destas aplicações são:

- Imagem gerada pelo computador – CGI *(computer generated imagery)*. Usadas para a geração de filmes, possui as seguintes características: altíssima qualidade de imagem, baixa taxa de quadros e alto custo.
- Modelagem, animação, CADs e visualização de dados. Possuem alta qualidade de imagem, média taxa de QPS e médio custo.
- Jogos possuem alta taxa de QPS, baixo custo e qualidade de imagem mediana.
- Simuladores possuem alta taxa de QPS, qualidade de imagem de média a alta e alto custo.

Esta divisão é representada através de uma relação entre custo, qualidade de imagem e taxa de QPS, como ilustra a Figura 3.

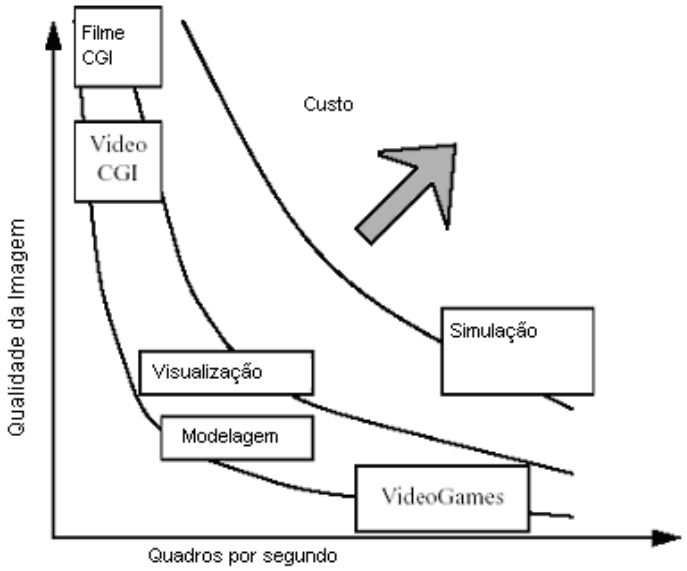

Figura 3 Qualidade de imagem x Custo x QPS [Helman96]

Clay [Clay96] discute que, até a alguns anos atrás, para se construir cenas 3D realistas tinha-se duas opções: produzir a imagem com qualidade suficientemente alta que atendesse os padrões de Hollywood, com tempos de renderização medidos em **minutos por quadro,** ou usar equipamentos de visualização de topo de linha, para produzir imagens na taxa de 12 a 60 QPS, pelo custo variando entre $200,000 a $1,000,000 de dólares.

Mas tecnicamente este estado mudou, devido à evolução dos computadores e do hardware para processamento gráfico, resultando no decremento da curva de custo em direção à origem, para uma combinação específica de qualidade e taxa de quadros. Isto resulta na possibilidade de uso de sistemas de tempo real para uma ampla gama de proporcionando maior qualidade de imagem para as aplicações implementadas em tempo real. Porém, novos requisitos de qualidade são estabelecidos e o ciclo recomeça, mas a curva de aplicações permanece a mesma, visto que as empresas podem investir a mesma quantia que anterior para proporcionar as aplicações de qualidade ainda maior. Assim, aplicações de alta qualidade de imagem, alta taxa de QPS e com custo reduzido podem ser vislumbradas. A

Tabela 1 exibe aplicações 3D e respectivas taxas de QPS atingidas.

Tabela 1 Divisão por quadros por segundo de aplicações 3D

Taxa de QPS	Aplicação	Qualidade	Custo
Quadro-a-quadro	Filme CGI	Altíssima	Altíssimo
0.001-1	Vídeo CGI	Alta	Alto
Interativo	Ferramentas de Modelagem	Baixa	Médio
5-10	Captura de Movimentos	Baixa	Médio
	Visualização de dados	Baixa	Médio
Tempo Real	Simulação	Média	Alto
15-60	Videogames	Baixa	Baixo

Aplicações que se enquadram entre 15 e 25 QPS atendem parcialmente a sensação de interatividade em tempo real. A quantidade de interações realizadas irá ditar a sensação de imersão do jogador. Acima de 30 quadros por segundo, a interação é considerada em tempo real. Acima de 60 QPS, mudanças na taxa de quadros por segundo são imperceptíveis.

O desafio de manter uma taxa de quadros que se sustente na maioria das configurações é de grande importância em aplicações de jogos. Com o hardware promovendo, no mínimo, aceleração básica para renderização, outros pontos de latência devem ser considerados para que seja mantida a taxa de quadros que proporcione interação em tempo real.

A latência de uma interação é o tempo medido desde o momento em que o usuário emitiu uma entrada até o momento em que esta entrada é refletida na tela - quanto maior a latência, menor a taxa de quadros renderizada por segundo. Vários fatores contribuem para a latência: o tipo do dispositivo de entrada, a arquitetura do programa, processamento de efeitos gráficos, tempo de renderização, processamento sonoro, etc. Diferentes partes de um sistema podem possuir diferentes latências, como por exemplo, a resposta visual da mudança do ponto de visão do usuário provavelmente irá diferir do tempo de resposta de um tratamento de colisão.

O desempenho completo do jogo dependerá de vários elementos, incluindo o hardware, software e os dados da aplicação. Os componentes de uma plataforma de jogo são, tipicamente, estruturados em níveis. Os componentes de nível mais baixo não são específicos

de nenhum jogo em particular. Entretando, os componentes de nível superior tornam-se mais específicos, por exemplo, um gerenciador que controla o sistema para uma classe específica de jogos. A camada final é formada pelos conteúdos específicos de um jogo, como: personagens, modelos geométricos 3D, organização da cena, lógica do jogo, comportamento, animação, inteligência de NPCs, ou seja, a aplicação em si e os conteúdos associados.

Cada uma destas camadas necessita de elementos de hardware para melhorar o desempenho. Juntos esses elementos formam um conjunto comum de capacidades que são usadas para a execução de tipos diferentes de jogos.

A Figura 4 ilustra um conjunto comum de elementos de hardware de diferentes plataformas. A Tabela 2 descreve exemplos de hardware e software, com a indicação de qual componente realiza uma determinada função em um jogo.

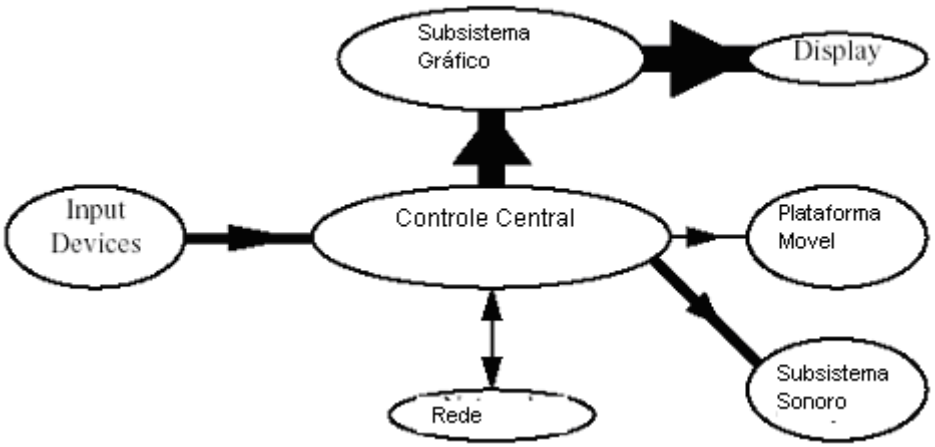

Figura 4 Interconexão de elementos de hardware.

Tabela 2 Funções de baixo nível e exemplos de respectivos componentes

Função	Exemplo de Hardware	Exemplo de Software
Processamento Central	CPU(s)	Unix™
Visual	Subsistema gráfico Monitores Capacetes	IRIS Performer ™
Áudio	Sintetizador MIDI	*Drivers* de áudio
Movimento	Plataforma de movimentação	Modelos dinâmicos *Drivers* de movimentação
Entrada	Controles	*Drivers* de dispositivos

	Rastreadores	

Um jogo é constituído, basicamente, dos componentes listados na

Tabela 3.

Tabela 3 Divisão do tipo de dados usados em jogos[Hoffman96]

Componentes	Exemplos
Base de dados Visual	Geometria, texturas, modelos de animação
Base de dados Sonora	Amostras, dados para sintetização
Modelos de comportamento	Controle de animação, colisão e inteligência artificial
Interface com o usuário	Modelos de interface
Lógica do Jogo	Manipulação do Cenário
Aplicação	Controle geral, história, coordenação, continuidade.

A base de dados geométrica é tipicamente construída usando uma ferramenta de modelagem 3D. A saída é uma coleção de polígonos, definição de materiais, arquivos de textura e transformações matriciais (necessárias para animação de um objeto) destinados a renderização em tempo real. Ferramentas de modelagem comerciais tendem a se enquadrar em duas categorias: modeladores com origem em simulação visual ou origem em animação computacional, não realizada em tempo real. O primeiro tenta focar no suporte a construção de objetos, polígono por polígono, ou por meio de definições simples. Este método de construção é apropriado para sistemas em tempo real, tendo em vista que a contagem de polígonos deve ser reduzida para que seja atendida razoável taxa de QPS. Os modeladores destinados à produção de animações quadro-a-quadro normalmente possuem um conjunto rico de ferramentas de construção de superfícies, mas perdem em não fornecer métodos de

renderização em tempo real. Alto nível de primitivas, como esferas, cilindros e NURBS[1], são recursos poderosos, porém perigosos para sistemas em tempo real, pois geram muitos polígonos a mais do que o aceitável para renderização em tempo real.

Como resultado da saída da modelagem dos dados, tem-se o terreno do jogo, os modelos de objetos, texturas, materiais e informação da aplicação.

Os **terrenos** são ambientes 3D projetados para serem navegados e consistem de todos os planos: chão, piso e parede, além de características, tais como: oceanos, estradas, arbustos, e tudo o que permanece estático na aplicação.

Os **modelos** compreendem quaisquer objetos da base de dados que possuem animação, seja ela fixa ou de comportamento variado, e são, na sua maioria, modelados com um alto nível de detalhe *(LOD – Level of Details)* com subseqüentes representações com menor nível de detalhes (menos polígonos) usados para controlar a densidade da cena.

Texturas são arranjos retangulares de dados RGB (textels) que são aplicados a uma superfície geométrica para criar a ilusão de detalhes. São geralmente derivados de fotografias e sintetizados por programas especializados ou qualquer ferramenta que produza como saída uma imagem passível de mapeamento em um polígono. O mapeamento de texturas é um recurso usado em processos de renderização em tempo real. Uma textura de ótima qualidade possui um resultado tão bom quanto à representação correspondente em polígonos, porém consumindo uma quantidade menor de recursos.

Os **materiais** controlam as propriedades reflexivas e refrativas de um polígono, como a transparência, e propriedades gerais como a reação a luzes ambientais, difusas, emissivas, e especular.

A informação associada à geometria ajuda a aplicação a controlar a cena, com constantes de movimento que controlam a animação do objeto (estado estático ou dinâmico), *bounding boxes,* usados para detecção de colisão entre objetos, dentre outras.

Alguns dados não são possíveis de serem gerados por modeladores, e precisam ser inseridos na forma de programação, ou por meio de ferramentas específicas, como efeitos especiais, poeira, fumaça, faíscas e demais. Outra exceção é o tratamento de movimentação e animação que são gerados por seqüências de animação, feitas no mundo real, que levam a um método de interpolação, resultando no comportamento do objeto.

Com o conteúdo da aplicação definido, o próximo passo é a renderização, a qual precisa ser acelerada para que as aplicações respondam, para o usuário, em tempo real.

[1] Non-uniform rational B-spline: é um modelo matemático usado regularmente em programas gráficos para gerar e representar curvas e superfícies

Esta aceleração é obtida por meio de hardware apropriado - este processo será detalhado na próxima seção.

2.6 Suporte a Aceleração gráfica via hardware

Uma vez definidos os requisitos de desempenho e conteúdo de um jogo, a arquitetura do hardware deve ser projetada para suportar as operações usadas pelos componentes de software de maneira otimizada. A maioria dos dispositivos renderiza a cena rastreando os polígonos, em vez de realizar técnicas complexas como *Ray tracing* ou renderização volumétrica, como é feito em sistemas de visualização computacional ou animação em CGI. A maioria das técnicas usadas em aplicações que necessitam de alta qualidade de imagem, não é usada em sistemas de jogos de entretenimento em tempo real, a menos que exista suporte de hardware para tal técnica, e uma grande base instalada deste hardware exista nos equipamentos dos consumidores.

Como existem diferentes tipos de aceleradores gráficos 3D, os desenvolvedores devem adequar suas aplicações a fim de serem executadas no maior número de configurações possível sem prejudicar a qualidade do jogo.

Atualmente se usa o termo GPU *(Graphic Processor Unit* – Unidade de processamento gráfico) a fim de referir-se ao *chip* responsável pela renderização, aceleração, cálculos e demais funções de um acelerador 3D. A GPU armazena na memória de vídeo todos os dados necessários para realizar suas funções. A comunicação entre a GPU e a memória de vídeo deve ser extremamente rápida, superior ao tempo de acesso à memória RAM. Da mesma maneira, a comunicação entre a memória RAM e a memória de vídeo também requer altas taxas de transmissão.

Uma típica questão relacionada à aceleração em hardware é a seguinte: “Exatamente o que é acelerado?”. As repostas são muitas e dependem tanto da aplicação a ser usada quanto da geração em que foi concebida. Os elementos que, tipicamente, são considerados para serem submetidos à aceleração em aplicações 3D em tempo real são ilustrados na Figura 5 [Johnst96].

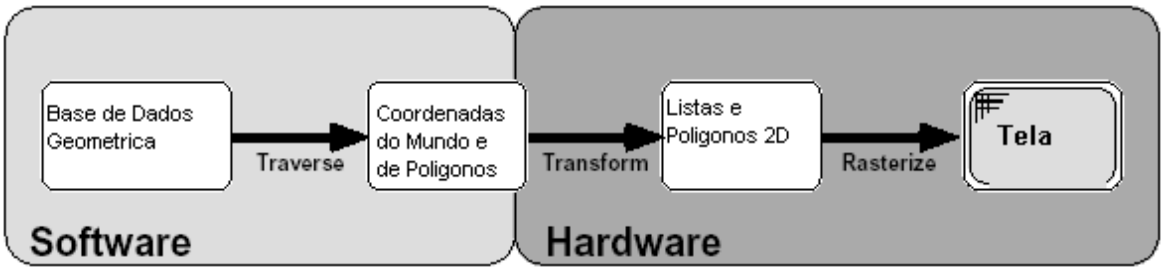

Figura 5 Aceleração típica em hardware

No caso da Figura 5, o hardware contém o seguinte: uma pilha de modelos organizados em matrizes, uma pilha de visualização, e alguns atributos de renderização. São então passados ao hardware, polígonos, vértice por vértice, nas coordenadas do mundo. As vantagens são velocidade, transformação e iluminação, realizadas em hardware, além de outros benefícios. A desvantagem é o custo da solução de hardware, o qual é atualmente, de no mínimo cinqüenta dólares.

Várias técnicas antes realizadas via software, migraram para o hardware, como por exemplo, *clipping* (remoção de partes do objeto não visíveis), *culling* (exibição dos objetos somente contidos no campo de visão do usuário), tratamento de oclusão (não renderização de objeto cuja representação esteja bloqueada por outros objetos) dentre outras [Johnst96]

Para que um jogo seja executado sem que a percepção de desempenho pelo usuário seja comprometida, deve ser mantida uma taxa mínima de quadros por segundo. Caso esta taxa esteja abaixo do valor mínimo necessário para a percepção de interação o usuário perceberá a queda de desempenho, podendo resultar em distração, perda de concentração, e até mesmo tonturas e náuseas, o que, conseqüentemente, levará à perda de interesse pelo jogo.

Atingir uma determinada taxa de quadros por segundo para uma cena é simples - a dificuldade está em manter esta taxa quando o conteúdo da cena varia arbitrariamente em termos de complexidade, uma situação corriqueira neste tipo de aplicação. Soluções propostas tentam avaliar o pior caso em que a complexidade de uma cena pode atingir, e otimizar o processo de renderização para tal situação - apesar desta solução se ser viável em alguns casos, geralmente resulta em perda inaceitável de recursos do sistema [Hoffman96].

Neste trabalho de mestrado é proposta a manutenção de uma taxa mínima de quadros por segundo, renderizando a cena com a maior qualidade possível, conseguido por meio da adaptação da cena sendo renderizada, via um *framework* de adaptação, a ser descrito no quinto capítulo.

Técnicas mais apropriadas, as quais resultam em melhor desempenho, são normalmente, deixadas a cargo dos motores de jogos. Porém, recursos de configuração devem ser oferecidos ao usuário para que ele possa sintonizar a sua aplicação em função da configuração existente em sua plataforma, fazendo uso apropriado de possíveis acelerações existentes de hardware. Os parâmetros que influenciam no desempenho de jogos são descritos a seguir.

2.7 Parâmetros configuráveis que influenciam no desempenho de Jogos

Na seção anterior foi visto que as acelerações usadas em hardware contribuem para o aumento do desempenho em aplicações de jogos 3D. Entretanto, existem também configurações destinadas à adequação da aplicação em diferentes plataformas. As opções de configurações dependem tanto do hardware quanto da forma em que o jogo foi programado, bem como dos dados usados pela aplicação. Estas opções são fornecidas por meio de uma interface com o usuário desenvolvida pelos projetistas do jogo [Bigos96].

O usuário então seleciona a opção desejada, por exemplo, manter a qualidade de imagem e realismo ou obter desempenho. Estas opções são normalmente genéricas para maioria dos jogos - caso a plataforma não possua suporte para as configurações oferecidas, por não possuir aceleração em hardware ou não suportar o nível estabelecido, as opções ficam desabilitadas na interface. As opções relacionadas ao software são fornecidas sem a necessidade de hardware especifico, porém dependem de recursos, tais como processamento e memória, para a execução com uma taxa de QPS aceitável, e variam de aplicação para aplicação [Simpson02].

Dentre as opções dependentes de hardware, para o processamento de imagens e áudio, podemos destacar as seguintes:

Resolução: Determina quantos *pixels* (pontos que formam uma imagem na tela) são exibidos, por exemplo, uma resolução de 800x600 implica na exibição de 480000 *pixels*. Quanto maior a resolução melhor detalhada é a imagem gerada. A resolução máxima é determinada pela GPU, juntamente com a capacidade do monitor ou *display* de suportá-la. A modificação da resolução impacta principalmente na GPU, mas também tem um bom impacto sobre a CPU, pois é necessário que mais vértices e coordenadas sejam geradas.

Profundidade de cores: Determina qual a possibilidade máxima de cores que será representada pelo sistema. Este parâmetro varia de 2^8 a 2^{32} e é determinado pela GPU

bem como pelo monitor ou *display*. A mudança deste parâmetro durante o jogo acarreta maior carga sobre a GPU e memória de vídeo, bem como sobre a memória principal.

Qualidade de textura: Este parâmetro normalmente modifica a resolução e a quantidade de cores que a textura possui. Maior resolução e melhor profundidade de cores tornam a imagem mais definida, menos borrada, e com transição entre cores mais suave. Quanto maior a qualidade de uma textura mais memória de vídeo, bem como memória principal, são necessárias.

Mip-Mapping: A alteração da textura de um objeto, plano ou terreno, em tempo real, é obtida por meio da técnica de *Mip-mapping*, que define a textura a ser usada, a partir da distancia entre o jogador e o objeto texturizado. O custo desta técnica é o uso de memória, pois é necessário armazenar todas as variações de uma mesma textura causando um aumento de 33% do uso da memória.

Filtragem de Texturas: Determina o método no qual a textura é corrigida após ser aplicada. Esta opção geralmente contém os seguintes valores: *Bilinear, Trilinear, Anisotropic(2x, 4x, 8,16x).* Quanto maior o valor, maior o detalhe e menos borradas as texturas aparecem quando vistas de longa distância, ou ângulos diferentes do frontal. A melhoria na qualidade da imagem é atingida por meio da correção do mapeamento do *textel* para *pixel* por meio do uso de amostras ao redor do *textel* para promover melhor definição de cores. Apesar da possibilidade desta opção ser aplicada via software (CPU), o desempenho ficaria inaceitável, portanto os jogos utilizam-se do suporte via hardware (GPU) para a ativação desta opção. Mesmo implementada via hardware, esta opção impacta no desempenho da GPU, principalmente quando uma opção de melhor qualidade está ativada (por exemplo, a *Anisotropic 16X).*

***Antialiasing*:** O uso desta opção tenta remover as linhas tracejadas que formam as bordas dos polígonos, tornando-as retas, suavizando as bordas de polígonos de cores diferentes, aumentado drasticamente, a qualidade e definição dos objetos. Esta opção é normalmente apresentada nos formatos 2x, 4x e 6x, que são o número de amostras ao redor de um determinado *pixel* que é usado para obtenção dos novos valores de cores a serem usados na suavização. Assim como o filtro de texturas, a implementação desta função via software é inviável e depende de suporte de hardware para ativação. *Antialiasing* causa um grande impacto na GPU, principalmente quando escolhida a opção 6x.

Qualidade do áudio: Esta opção permite modificar a qualidade do áudio para obter pequenos ganhos de desempenho, principalmente em decodificadores sonoros não direcionados a jogos. Ao diminuir a qualidade de áudio reduz-se a taxa de amostragem do

sinal (que varia ente 8khz até 48 kHz), e a quantidade de bits usada para codificação de uma amostra (8 ou 16). Para uma plataforma suportar uma determinada qualidade sonora ela deve possuir um software de decodificação sonora e suporte ao hardware para a saída gerada.

Número de Canais ou Vozes: Quantidade máxima de canais que deverão ser reproduzidos ao mesmo tempo. Entenda-se por canais simplesmente como sons independentes. Um arquivo de som *mono* utiliza um canal para ser reproduzido, no entento, um som estéreo utiliza dois canais. A capacidade de uma plataforma reproduzir uma determinada quantidade de canais dependerá do decodificador sonoro (placa de som) além de possuir a capacidade de acesso a diferentes *buffers*. A quantidade de canais reproduzidos impacta diretamente na CPU resultando em grandes diferenças de desempenho.

Áudio 3D: processamento de áudio onde se obtém a "sensação" de localidade e profundidade por meio da definição de intensidade e localização das fontes sonoras. Caso a fonte de um determinado som esteja, no mundo virtual, do lado esquerdo do usuário, o som será reproduzido no alto-falante esquerdo com mais intensidade do que no alto-falante direito (caso a plataforma use som estéreo). Caso o usuário se distancie desta fonte sonora, a intensidade diminuirá proporcionalmente nos alto-falantes até que não seja ouvida. O processamento utilizado para proporcionar o áudio 3D pode ser feito tanto pela CPU, com pouco impacto, quanto pelo codificador sonoro, caso este possua suporte de hardware para tal.

Além das opções que necessitam de suporte a hardware para serem ativadas, existem também as opções dependentes da aplicação. Apesar de certas opções estarem vinculadas à implementação do jogo, elas são projetadas parametricamente para que o usuário modifique-as, a seu critério. Apesar do suporte a estas opções variar de jogo para jogo, ou mesmo não existir em determinados sistemas, este tem papel importante na questão de desempenho. Dentre os aspectos mais comumente encontrados em ambientes virtuais 3D que tratam do desempenho, destacam-se:

Detalhes do Mundo: Determina quantos objetos ao redor do usuário serão mostrados. Numa floresta, por exemplo, alterações nesta opção levariam à exibição de menos árvores e flores, tornando a renderização mais rápida. O impacto maior é notado na GPU, mas a CPU sofre um pequeno aumento de carga também.

Detalhes do Avatar: Decide qual o nível de qualidade aos quais todos os avatares, representação geométrica dos usuários no mundo virtual 3D, serão submetidos. Maior nível de detalhe significa maior número de polígonos e texturas aplicadas com maior resolução. O impacto de mudanças nesta opção refletirá sobre a carga da GPU e da memória.

Detalhes de Física: Modifica o nível de detalhes da simulação da física em uma cena, como, por exemplo, gravidade, centro de gravidade de um objeto, mudanças de velocidade, inércia, peso, atrito, tratamento de colisões, entre outros. Os efeitos mais realistas são os primeiros a serem retirados por causarem um maior impacto no desempenho. Esta opção causa grande impacto sobre a CPU, pois são necessários cálculos complexos para a exibição de efeitos mais realistas. O mínimo que um jogo deve simular é a detecção de colisão. As demais possibilidades dependem dos programadores decidirem se serão implementadas ou não.

Sombras Dinâmicas: Sombras dinâmicas reagem a uma fonte de luz variante. Por exemplo, a sombra do usuário ao passar ao lado de uma lâmpada na parede, muda de orientação e intensidade. Também influenciam no reflexo causado por uma luz em outra superfície. A GPU tende a ficar mais ocupada à medida que mais efeitos deste tipo são usados.

Detalhes de Sombras: Um refinamento da opção anterior. A mudança de detalhes causa alterações na resolução da sombra, gerando contornos mais definidos e suaves. Assim como na opção anterior, a GPU sofre maior impacto com o aumento do número de detalhes.

Decals: Modificações feitas dinamicamente em superfícies como resultado de ações engatilhadas por objetos no mundo virtual, incluindo o usuário, bem como a inclusão de novos objetos na cena dinamicamente. Para realizar e manter estes efeitos é necessário ter potencial de processamento da CPU relativo à quantidade e duração dos mesmos.

Tempo de IA: Quantidade do tempo da CPU destinado ao processamento da inteligência artificial que comanda os objetos dirigidos por simulação, quando presentes. Quanto mais tempo de IA é concedido à aplicação, melhor a reação dos avatares comandados por computador, maior o desafio fornecido ao jogador e um alto grau de realismo é oferecido ao usuário.

Partículas: Na criação de efeitos especiais como fumaça, faíscas e fogo, usam-se vários polígonos, chamados de partículas, além de processamento para controle dos mesmos, podendo ser submetidos a processamento resultantes da física e de interações com o usuário. A quantidade de polígonos que é gerada e o comportamento dos objetos definirão a qualidade e realidade dos efeitos apresentados. O impacto é observado tanto no processamento da CPU quanto no da GPU, e dependendo da quantidade e qualidade dos efeitos, este impacto trará severas conseqüências no desempenho.

2.8 Considerações Finais

Nesse capítulo descreveram-se os tipos de jogos existentes, em especial os jogos 3D multiusuário, seus requisitos e modelos de comunicação. Foram também identificados os elementos que impactam no desempenho de um jogo 3D, que podem ser selecionados e/ou sintonizados, pelo usuário, para melhor desempenho ou realismo, inclusive. Em ambientes heterogêneos de processamento e/ou de comunicação, alguns destes elementos podem sofrer adaptação para que as opções do usuário possam ser mantidas frente a flutuações nas capacidades de processamento e de armazenamento, bem como modificações no estado da rede de comunicação a qual o dispositivo do usuário está conectado. Os elementos que compreendem jogos virtuais 3D MU e que podem impactar nos recursos dos dispositivos de acesso e da rede são inúmeros e complexos. Por isso, em termos de adaptação de mídia, na literatura, está voltado ao tratamento de mídias contínuas, como áudio e vídeo. São poucos os trabalhos encontrados sobre adaptação de mídia 3D. Desta forma, apenas alguns destes elementos foram selecionados como parâmetros que podem sofrer adaptação em ambientes de jogos 3D multiusuário, para servir como uma base inicial de mecanismos de adaptação de mídia 3D em ambientes de processamento e de rede heterogêneos. O próximo capítulo discute os mecanismos de adaptação existentes na literatura.

3 ADAPTAÇÃO

O termo “adaptação” é visto, na literatura, em vários contextos: adaptação de mídia, adaptação de conteúdo, adaptação de interface, adaptação de aplicação, adaptação em função de requisitos de qualidade de serviço – QdS etc. De uma maneira geral, adaptação significa mudar o estado de uma mídia, ou uma aplicação, para que esta se ajuste às condições encontradas na rede e/ou dispositivo em que se encontra. Este trabalho se concentra na adaptação de aplicações, particularmente as aplicações de jogos multiusuário. A adaptação da aplicação significa, neste trabalho, sintonizar os dados com o estado dos recursos do dispositivo e da rede. A seguir, é dada uma visão geral de adaptação encontrada na literatura.

3.1 Uma Visão Geral de Adaptação

Tendo em vista o recente uso do termo “aplicação adaptativa”, há um longo tempo os programas de computador são projetados de modo a se acomodarem em diferentes ambientes, pois existe uma variedade de combinações de sistemas e usuários para quais os programas são destinados. Esta é, na verdade, uma maneira natural das empresas aumentarem o apelo comercial de seus produtos. As inúmeras opções, escolhas, parâmetros, preferências disponíveis nos mais populares produtos (processadores de texto, antivírus e sistemas operacionais) são sinais do esforço para adaptar as aplicações às diferentes necessidades dos usuários.

Programas configuráveis permitem incluir somente módulos que são necessários em contextos particulares. Os programas também se “adaptam” a respectivos hardwares e sistemas operacionais, por exemplo, a quantidade disponível de memória, a presença de instruções particulares em processadores, a dispositivos de entrada e saída. Alguns programas detectam os recursos disponíveis no dispositivo, como características da tela, quando não, a escolha é feita pelo usuário durante a instalação. No que tange a infra-estrutura de comunicação vislumbra-se negociação entre terminais, dependendo das características presentes nos mesmos, reconfiguração de sistemas tolerantes a falhas, detecção e correção de erros em protocolos de comunicação tradicionais (HDLC, X.25) [Gecse99].

A adaptação tratada neste trabalho, seguindo a tendência do mercado, é fornecida como um **serviço** central, oferecendo suporte para várias aplicações diferentes,

resultando em maior eficiência e qualidade, além de ser uma candidata em potencial para uso em ambientes móveis [Baggio01].

Tipicamente, um dispositivo móvel é escasso de recursos, conforme visto no capítulo 2, além de possuir um poder de processamento (inteiro e ponto flutuante) inferior aos dispositivos fixos, como computadores pessoais. Complementando, a conexão de rede é suscetível a constantes variações de largura de banda, latência, conectividade e taxas de erro. Problemas relacionados ao gerenciamento de energia, como preservação de bateria, fazem com que várias ações sejam evitadas, retardadas, ou até mesmo recusadas. Finalmente, os clientes móveis necessitam de tratamento diferenciado em relação à segurança e robustez.

Toda aplicação, em qualquer dispositivo, está sujeita a concorrência com outros processos do sistema, com prioridades pré-estabelecidas, variações da largura de banda e latência, decorrente de congestionamentos, e heterogeneidade de recursos [Rao01]. Como conseqüência, as aplicações necessitam adaptar-se a mudanças no ambiente ao seu redor e também a mudanças nos níveis de recursos internos do dispositivo.

A adaptabilidade é possível em vários níveis:

Nível Físico: o mecanismo de adaptação resulta em uma seleção apropriada do canal de transmissão, controle do nível de energia, detecção de erros etc.

Camada de Rede: Mecanismos de rotas devem adaptar-se para mobilidade, como exemplo, o padrão IP Móvel.

Camada de Transporte: O protocolo desta camada deve distinguir entre enlaces sem fio e com fio, fornecendo maior suporte à correção de erros em enlaces com taxas mais altas de erro, como é o caso dos enlaces sem fio.

Camada de Aplicação: As aplicações devem adaptar-se a variações da largura de banda e de recursos do dispositivo.

3.2 Onde fazer a Adaptação?

A adaptação da aplicação pode ser feita do ponto de vista dos dados e/ou do controle. A adaptação de dados implica numa transformação dos dados usados com freqüência pela aplicação para versões que se adaptam aos recursos disponíveis, sem a perda da representação. Por exemplo, uma textura pode ser transformada em uma versão com menor resolução. Entretanto, a adaptação de controle implica na modificação do fluxo da aplicação,

ou seja, no seu comportamento. Uma aplicação pode, a partir da notificação de aumento de atraso na rede, introduzir um buffer de recepção de dados, em tempo de execução, para a diminuição da latência [Lara2001].

Segundo [Noble98] a adaptação, quanto à localização, é classificada como:

Transparente para a aplicação ou baseada em Sistema – o Sistema Operacional toma para si a responsabilidade integral da adaptação, ficando entre a aplicação e o dado. Esta abordagem fornece compatibilidade retroativa com as aplicações existentes, e não necessita de modificação na aplicação. Por promover um controle centralizado, várias aplicações podem ser adaptadas seguindo uma política geral do sistema. Entretanto, as políticas de adaptação, neste caso, são dependentes do tipo dos dados e também do tipo da aplicação. Assim, para cada tipo de aplicação uma nova política deve ser estabelecida, não sendo possível o uso de políticas genéricas.

***Laissez-faire* ou baseada na Aplicação** – independente do sistema, somente a aplicação é alterada. Nesta estratégia, não há um gerenciador central de adaptação – apenas a aplicação é alterada, o sistema não está ciente da adaptação. Isto torna mais difícil a escrita do código das aplicações, devendo ser capaz de detectar, por exemplo, outras aplicações que estejam concorrendo pelo mesmo recurso, porém, a adaptação é feita sob medida, logo, um melhor resultado final.

Ciente da aplicação – promove uma parceria entre a aplicação e o sistema para prover a adaptação para todas as aplicações que façam uso da mesma. Esta estratégia envolve muitas vezes a existência de uma camada entre a aplicação e o sistema operacional, que é responsável pelo monitoramento e controle de recursos, bem como as decisões sobre a aplicação de políticas de adaptação. Esta camada pode ser um Middleware ou framework [Baggio98], [Gecse99].

Baseada em Componente - Esta abordagem possibilita um controle específico da aplicação e de políticas para adaptação de dados sem a necessidade de modificação na aplicação. Isto é feito através de APIs e de tratamento de formatos, que proporcionam a adaptação de controle. Este tipo de adaptação tenta unir os benefícios das abordagens *Laissez-faire* e *transparente para a aplicação*.

As estratégias de adaptações *Laissez-faire*, *ciente de aplicação* e *baseada em componentes,* permitem a adaptação tanto do controle quanto dos dados. Entretando, a adaptação *transparente para a aplicação* é limitada apenas aos dados.

Após a escolha do tipo de adaptação a ser promovida e como será implementada, vários fatores devem ser considerados no projeto da aplicação, estes são discutidos na próxima seção.

3.3 Projeto de aplicações adaptativas

Uma aplicação adaptativa é aquela que modifica seu comportamento de acordo com as mudanças percebidas no ambiente, mantendo assim a semântica (ou comportamento esperado) para o usuário [McIlhagga98]. Geralmente, a adaptação, em função de flutuações nos níveis de recursos dos dispositivos, é de responsabilidade do sistema. A adaptação que envolve, por exemplo, modificação nas funções da aplicação, é visível para o usuário, assim, o usuário deve estar envolvido no projeto da aplicação.

A Figura 6 ilustra um exemplo de adaptação em camadas, baseada na aplicação *(Laissez-faire).* Na figura é ilustrado também o grau de satisfação do usuário.

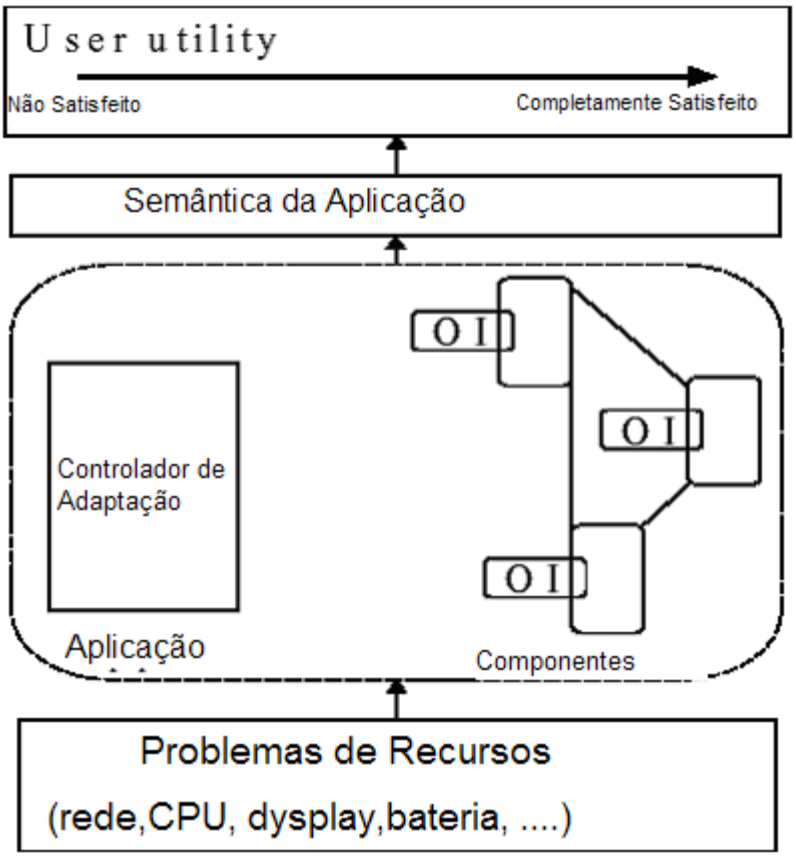

Figura 6 Visão de Adaptação em camadas [adaptada de McIlhagga98]

O primeiro passo no projeto de uma aplicação adaptativa é identificar os recursos e a semântica da aplicação. As escolhas no projeto da aplicação devem incluir uma implementação aberta *(OI- Open implementation)* da tecnologia que expõe os recursos a serem monitorados. O controlador de adaptação pode ajustar, por exemplo, as necessidades de rede dos componentes que lhe solicitaram o serviço mediante uma política de adaptação. O controlador da adaptação pode ser implementado como parte de um componente.

Implementações ortogonais, que oferecem várias maneiras de se fornecer o mesmo resultado, são usadas para aumentar o nível de liberdade da aplicação para se adequar a mudanças nos recursos, modificando a funcionalidade da aplicação ao mesmo tempo em que se mantém a semântica da aplicação.

Muitos usuários não estão preocupados com as características intrínsecas da rede ou visor, mas sim no funcionamento do software. Neste caso, a presença do usuário durante a fase de projeto faz-se necessária. A escolha de incluir ou não a intervenção do usuário no processo de adaptação depende dos recursos considerados e também do comportamento da adaptação. Por exemplo, a oferta ao usuário de diferentes rumos que uma aplicação pode tomar, mediante carga baixa de bateria.

Para tratar da flutuação no nível dos recursos, a aplicação pode ser executada em trajetórias ou caminhos pré-determinados. Determinar todos os contextos possíveis nos quais uma aplicação será executada durante o projeto pode não ser plausível, sendo assim, alguns contextos devem ser determinados em tempo de execução. Neste caso, o usuário ajudará a construir um perfil que defina as mudanças de comportamento da aplicação que não puderam ser pré-estabelecidas.

As configurações disponíveis para o usuário no tangente aos recursos devem ser interpretadas e apresentadas de forma metafórica, proporcionando fácil entendimento para a escolha de uma decisão correta.

3.4 Políticas de Adaptação

A adaptação é orientada por uma política. O tipo da aplicação, bem como os dados, muitas vezes dita a escolha da política utilizada. Um modelo de adaptação deve levar a política em consideração (quando adaptar, o que adaptar, quanto adaptar, considerações de custo etc.), além de outras questões, tais como detecção de conflito de interesses e como resolvê-los, grau de intervenção do usuário na política de adaptação etc. [Rao00].

As seguintes questões devem ser consideradas quando se desenvolve sistemas de adaptação, pois estas devem reger as políticas a serem estabelecidas[*Cadmium00*]:

Quando adaptar? Devem ser pré-determinadas as fronteiras, isto é, o limite superior e inferior, para cada recurso considerado. Ex: Se Largura de banda for menor que 256kbps, as mensagens de voz devem sofrer maior compactação.

O que adaptar? Escolha dos recursos que deverão ser considerados em uma política de adaptação.

Quanto adaptar? Como escolher a granularidade da mudança quando o nível de um ou mais recursos mudam. Por exemplo, o tipo de compressão que as mensagens de voz devem ser submetidas quando a largura de banda cai de 328kbps para 256kbps.

Qual o custo? De que forma o serviço de adaptação impacta em serviços *pay-per-view* ou serviços pré-pagos.

Como detectar e resolver conflitos na política? Problemas podem surgir em decorrência do atendimento de políticas conflitantes. Por exemplo, se a disponibilidade de processamento aumenta de 50% para 90%, a política de adaptação pode determinar que sejam usadas mensagens de voz no lugar de mensagens de texto, entretanto a largura de banda diminui para 64 kbps impossibilitando o tráfego de mensagens de voz.

Qual deve ser o grau de intervenção do usuário? O grau de intervenção do usuário na determinação da granularidade da política de adaptação deve ser definido. Por exemplo, o usuário pode intervir na escolha do nível de qualidade em uma aplicação de visualização de imagens.

Quão ciente de arquitetura deve ser a adaptação? Como a aplicação adaptará a segurança, memória disponível, e níveis de computação em várias plataformas.

3.5 Classificação de recursos

A escolha dos recursos que deverão ser considerados para eventual adaptação é uma questão importante para que seja definida uma política de adaptação. A Figura 7 ilustra os tipos de recursos existentes. Num dispositivo, dois tipos podem ser identificados: as capabilidades do dispositivo e as capabilidades de comunicação. Cada recurso pode ser estático ou mudar dinamicamente no decorrer do processamento da aplicação.

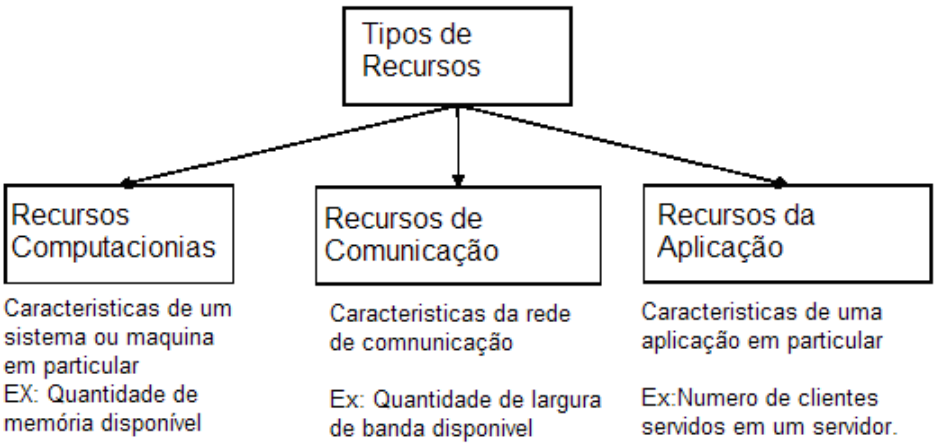

Figura 7 Classificação de recursos quanto ao tipo [Thades02]

A falta de um tipo particular de recurso pode causar uma demanda extra de outro. Por exemplo, em uma aplicação multimídia, a falta de recursos de comunicação introduz um custo adicional *(overhead)* nos recursos de computação – limitações na largura de banda podem requerer compressão que, por sua vez, requer maior potencial de processamento. Desta forma, é freqüentemente necessário comprometer uma categoria de recurso em função da escassez de outro.

Além de recursos de computação e de comunicação, muitas aplicações têm entidades embutidas que agem como provedores de recursos. Um exemplo simples é o número de clientes suportados por um servidor, que representa a carga sobre o servidor. Quanto mais clientes são servidos, maior a carga. Um programa cliente deve ser capaz de optar entre interagir com um servidor leve ou com um servidor carregado.

A

Tabela 4 classifica os recursos em estáticos ou dinâmicos e em recursos de computação ou comunicação.

Tabela 4 Classificação de recursos

	Recursos de computação	**Recursos de Comunicação**
Estáticos	Número de processadores Clock do processador Capacidade de memória Capacidade de disco	Máximo de largura de banda disponível
Dinâmicos	Espaço em disco disponível Número de processos ativos Utilização de CPU Número de quadros de página e tamanho de cada quadro	Carga de bateria Latência e *Jitter* Taxa de erro Conectividade Caracterização de Tráfego

3.6 Trabalhos relacionados

Nesta seção serão apresentados alguns trabalhos relevantes na área de adaptação de aplicações, tais como o CODA, CADMIMU e Odyssey, além de trabalhos relacionados à solução proposta nesta dissertação de mestrado.

Uma grande quantidade de trabalhos tem sido devotada à adaptação da entrega de formatos tradicionais de multimídia, como texto, imagens, áudio e vídeo. Pouca atenção tem sido dada ao conteúdo digital 3D, primeiro por causa da complexidade das questões envolvidas na adaptação de aplicações 3D, mais complexas que outras mídias, e também pelo fato que oportunidades de marketing para conteúdo 3D estão apenas começando a surgir [Martim02].

3.6.1 Adaptação em aplicações Multimídia

A adaptação de aplicações multimídia normalmente se baseia no monitoramento dos recursos de rede, de hardware e de software, e na conseqüente modificação de tipos de dados, tais como vídeo, imagens, áudio, texto [Fox97].

Os recursos de rede que podem sofrer flutuação incluem: largura de banda, latência e taxa de erro na rede. Os recursos de hardware dos dispositivos que podem variar incluem: tamanho e resolução da tela, quantidade de cores, memória, poder de processamento, carga da bateria etc. Os recursos de software que variam se restringem às capacidades do sistema operacional ou da aplicação para suportar um determinado tipo de dado (por exemplo, *Post script* ou extensões HTML não padronizadas) ou extensões de protocolo, como o IP multicast.

Modificações no áudio, por exemplo, inclui alterações do número de canais (*Surround*, estéreo ou mono), redução da freqüência de amostragem do sinal sonoro e algoritmos de compressão. Em situações mais estritas, como no caso de mensagens de voz, um sintetizador é utilizado para converter áudio em texto.

Várias ações de adaptação podem ser aplicadas aos tipos de dados multimídia: a compressão pode ser usada tanto em quadros de vídeo quanto em quadros de imagens, resultando ou não em perda da informação. A redução da dimensão/resolução do vídeo ou imagem pode ser feita durante o processo de compressão ou aplicada separadamente. Filtros são aplicados a vídeos para descartar quadros semelhantes [McIhagga98].

Quanto aos textos, formatações e *aliasing* de fontes podem ser retirados para uma representação mais dinâmica na tela. Fontes muito complexas serão descartadas em prol do uso de alternativas mais simples.

3.6.2 CODA

CODA [Coda98] é um sistema de arquivos usados para permitir a continuidade de operações durante desconexões, incorporando políticas de *caching* e *prefetching.* Promove aplicação-transparente para adaptação por meio de acesso ao sistema de arquivos. Um ponto central do projeto é o gerenciador de cache *Venus*. Sempre que uma chamada call() para um arquivo é executada, *Venus* busca antecipadamente todo o arquivo e o armazena localmente no cliente - somente atualizações são comunicadas ao servidor.

CODA fornece dois modelos de operação no cliente: *modo conectado,* em que o servidor pode ser contatado de todos os pontos, e *modo desconectado,* em que o servidor não é alcançado durante a transação - neste caso, todas as atualizações devem ser sincronizadas com o servidor, assim que as conexões forem re-estabelecidas novamente.

O CODA é usado principalmente em dispositivos móveis e sem-fio, visto que conexões intermitentes são inerentes a estes ambientes.

3.6.3 Odyssey

Odyssey [Odyssey00] é formado por um conjunto de extensões para o Unix para suportar mobilidade. As extensões são fornecidas no nível da API e internamente ao sistema operacional - a adaptação é ciente da aplicação, permitindo-as mapear níveis de recursos a níveis de fidelidade, porém, o monitoramento e o controle dos recursos são feitos pelo próprio sistema.

Existem três componentes de suporte na API do Odyssey: suporte para as aplicações trocarem características presentes no ambiente; para manter controle das mudanças; e para requisitar uma mudança de política baseada no estado atual do ambiente.

3.6.4 Cadmium

Cadmium [Cadmium00] é um modelo ciente de rede que identifica e usa recursos disponíveis localmente, adaptando a mudanças, e promovendo a ilusão de continuidade de serviços, no que diz respeito a conexões fracas, desconexões e reconexões. Para manter suporte à intermitência das conexões, Cadmium promove a replicação de dados e de código (cache) em ambos os lados da rede, fixo e móvel. Suporte adicional é oferecido para tratar com as questões relativas à replicação de dados.

Cadmium fornece informações do nível de recursos e também realiza um *upcall* (*feedback*) sempre que os níveis de recurso ultrapassam os limites de tolerância permitidos, proporcionando que as aplicações mudem dinamicamente de acordo com o estado do ambiente.

3.6.5 Virtual execution environment

Chang e Karamcheti [Chang00] descreveram um *framework* de adaptação independente de aplicação, que garante um nível de desempenho através de adaptação

contínua a mudanças de recursos. Essa solução simplifica o projeto de aplicações cientes de recursos, pois os mecanismos de adaptação não precisam ser explicitamente programados. Esta solução consiste dos seguintes componentes:

Tunability Inerface: expõe escolhas de adaptação na forma de configurações alternativas da aplicação, enquanto promove o encapsulamento das funções principais do sistema. Uma aplicação que deseja ser adaptada deve ser escrita fazendo o uso da interface que promoverá a construção de agentes assim como controle de mensagens, permitindo acesso externo a execução da aplicação.

Virtual Excution Enviroment: emula a execução da aplicação sobre diversas situações de disponibilidade de recursos, permitindo a coleta *off-line* de informações sobre um comportamento resultante. Este ambiente, por meio da interceptação de chamadas a API do sistema, consegue simular o comportamento da variação dos recursos.

Juntos, estes componentes permitem que decisões sejam tomadas de forma automática em tempo real sobre as seguintes questões: “quando adaptar” (através do monitoramento contínuo das condições de recursos e o progresso da aplicação) e “como adaptar” (por meio da escolha dinâmica da configuração mais apropriada, de acordo com a preferência do usuário).

A solução acima monitora somente a carga da CPU e a variação de largura de banda da aplicação, desconsiderando os recursos da aplicação. A execução do *Virtual Excution Enviroment* é realizada antecipadamente e os valores dos resultados são mapeados via uma ferramenta de *benchmark* comercial que é utilizada em outras máquinas, o que torna a execução em outras plataformas possível, mas com resultados imprevisíveis. Porém, a solução serve para qualquer aplicação que deseja usar adaptação.

3.6.6 ARTE

ARTE - *Adaptive Rendering and Transmission Environment for 3D Graphics [Martin 02],* é um dos poucos sistemas de adaptação de mídia 3D encontrados na literatura. Ele facilita a entrega de modelos 3D em ambientes heterogêneos monitorando os recursos disponíveis e selecionando a transmissão apropriada de modalidades para adequarem as capacidades e estados temporários de cada cliente.

ARTE consiste de quatro componentes chaves: um monitor que mantém o controle das mudanças nas características dinâmicas do ambiente no qual a transmissão do

modelo e renderização são realizadas; um esquema de organização hierárquica para modelos 3D que permite a representação de várias modalidades a serem associadas a cada componente do modelo; um mecanismo de seleção adaptativo que leva em consideração características do ambiente, estrutura do modelo, e preferências do usuário para determinar a modalidade mais apropriada para cada componente de um modelo 3D que se enquadra em uma determinada configuração de recursos; e finalmente um motor de trans-codificação que incorpora um número de módulos os quais realizam as conversões de dados 3D para as modalidades selecionadas pelos mecanismos de adaptação.

A geometria dos modelos é exibida como coleções de componentes. A geometria de cada componente é convertida, ou seja, trans-codificada em modalidades com diferentes qualidades de imagem e características de transmissão. Associado a cada componente, encontra-se o valor percentual da importância que a adaptação gera para a percepção final na renderização do modelo.

ARTE propõe o monitoramento da taxa de quadros por segundo, ou seja, considera os recursos da aplicação, mas desconsidera os recursos do dispositivo, pois a taxa de quadros é o resultado de um estado da aplicação e dos recursos do sistema. Como a adaptação não leva em consideração o tipo de modalidade que é mais apropriada para um recurso em escassez, não há como obter esta informação, a renderização pode não ter a melhor qualidade possível para uma determinada configuração de recursos.

O mecanismo de *benchmark* realizado no cliente tem como objetivo determinar o impacto no desempenho do uso de modalidades genéricas. Este tipo de *benchmark* consegue, com precisão, adequar as políticas de adaptação às capacidades do dispositivo. Porém é necessário carregar todo um conjunto de modalidades especificamente para o teste, além dos dados da aplicação.

Outro ponto negativo, o sistema é projetado para uma determinada aplicação de visualização, apesar de o mecanismo poder ser implantado em diversas aplicações deste gênero.

3.6.7 Motores de Jogos

Uma abordagem semelhante ao comportamento de um componente do *framework* proposto é implementada no jogo Grand Prix 4 [GP402], em que o motor do jogo testa o jogo durante uma cena pré-determinada, com uma quantidade de configurações pré-

estabelecidas, até que a mínima taxa de quadros por segundo atinja entre 25-35 QPS. Estas opções de qualidade são fixadas nas configurações do jogo, porém não existe adaptação. O problema desta abordagem é que o teste realizado não leva em consideração momentos da aplicação que podem ser renderizados com uma qualidade maior de imagem, pois são apresentados a altas taxas quando executadas com as opções *default*.

3.7 Considerações Finais

Neste capítulo foi dada uma visão geral de adaptação, políticas de adaptação e tipos de recursos que devem ser considerados, bem como alguns sistemas de adaptação encontrados na literatura. Como a adaptação é um processo considerado importante em sistemas que atuam em ambientes heterogêneos de computação e de rede, iniciativas que promovem a padronização devem ser consideradas. O padrão MPEG-4 e sua extensão MPEG-J tratam, de forma superficial ainda, a questão da adaptação. Desta forma, o projeto de um *framework* de adaptação de aplicações 3D, realizado como parte deste trabalho de mestrado, foi baseado no padrão MPEG-4. O padrão MPEG-4 e sua extensão MPEG-J são descritos no próximo capítulo.

4 O PADRÃO MPEG-4 E SUA EXTENSÃO MPEG-J

O MPEG-4 [ISO02] é um padrão da ISO/IEC (ISO/IEC JTC1/SC29/WG11) desenvolvido pelo MPEG (*Motion Picture Experts Group*), responsável também pelo desenvolvimento dos padrões MPEG-1 (usado para criação de vídeo interativo em CD-ROM) e MPEG-2 (usado em DVD e televisão digital). O MPEG-4 é um esforço internacional que envolve centenas de pesquisadores e engenheiros, movidos pelo desafio da composição audiovisual interativa, com a possibilidade de taxas de transmissão muito baixas.

O grupo MPEG vem trabalhando no padrão MPEG-4 desde 1993, e finalizou a sua primeira versão no final de Fevereiro de 1999. O MPEG-4 provê serviços de televisão digital, aplicações gráficas interativas e multimídia interativa (em especial na WWW), satisfazendo as necessidades de autores, provedores de serviços e usuários finais. Diferentemente dos padrões MPEG-1 e MPEG-2, que foram divididos em 3 partes (Sistema, Vídeo e Áudio), o MPEG-4 é dividido em 6 frentes de trabalho conforme demonstrado na Tabela 5.

Tabela 5 Divisão do Plano de Trabalho do MPEG-4

Parte	Título	Padrão
1	Sistema	ISO/IEC 14496/1
2	Vídeo	ISO/IEC 14496/2
3	Áudio	ISO/IEC 14496/3
4	Testes de Conformidade	ISO/IEC 14496/4
5	Software de Referência	ISO/IEC 14496/5
6	DMIF(Delivery Multimedia Integration Framework)	ISO/IEC 14496/6

As partes 1, 2 e 3 possuem uma variedade de grupos de ferramentas, disponibilizando um determinado tipo de funcionalidade, como por exemplo, ferramentas de recuperação de erros, codificação de formato, etc. As diferentes ferramentas dentro de cada parte são agrupadas conforme as características, complexidade e os seus requisitos de funcionalidade. Cada grupo de ferramentas forma um *object profile* (perfil de objetos) dentro de cada parte. *Profiles* são divididos em níveis que são pontos de operação ou subgrupos de ferramentas.

4.1 Tecnologias de suporte às aplicações 3D

Os ambientes virtuais disponibilizados na Internet, no fim da década de 90, utilizavam a integração das linguagens VRML (*Virtual Reality Modeling Language*) e Java, ou ainda soluções proprietárias. Uma nova geração de tecnologias 3D para a criação de ambientes virtuais multiusuário, independente de plataformas, tem surgido, tais como Java3D, MPEG-4 e X3D. Estas tecnologias são brevemente descritas abaixo.

4.1.1 A evolução da linguagem VRML

VRML é uma linguagem independente de plataforma que permite a criação de ambientes virtuais 3D, permitindo-se navegar, visualizar e interagir com objetos por meio de um navegador web. A VRML 1.0, desenvolvida em 1994, descreve somente ambientes estáticos. Em 1996, a VRML 2.0 foi especificada para permitir a representação de comportamentos dinâmicos. A capacidade do VRML 2.0 é ampliada para o desenvolvimento de ambientes interativos multiusuário com a Interface de Autoria Externa (EAI). A EAI é usada por um *applet* para controlar um *browser* VRML bem como o seu conteúdo [ROEHL97], de tal forma que a manutenção da consistência de ambientes virtuais compartilhados podia ser realizada, através de *applets* Java - vários ambientes virtuais multiusuário foram criados desde então com esta tecnologia. Em 1997, a linguagem VRML 2.0 foi formatada em conformidade com os requisitos da ISO/IEC, tornando-se reconhecida como padrão internacional. A partir daí, a VRML97 substituiu as versões VRML2. 0 e VRML 1.0 que se tornaram obsoletas.

VRML é designada para codificar modelos poligonais, suportando fontes de luz, propriedades de materiais, mapeamento de texturas, transformações hierárquicas, pontos de visão e animações.

A atualização da linguagem VRML97 levou à VRML New Generation (VRML NG), rebatizada em 1999 como X3D *(Extensible 3D),* a ser descrita na próxima seção.

4.1.2 X3D - *Extensible 3D*

O X3D foi desenvolvido pelo *Web 3D Consortium* com o apoio de várias empresas desenvolvedoras de navegadores para a Web, considerado como sendo a próxima geração da VRML.

Embora o X3D seja baseado na VRML 97, o objetivo do X3D é bem mais ambicioso. Enquanto a VRML97 presta-se a suportar ambientes virtuais via Internet, o X3D ambiciona o suporte a esses ambientes a partir de qualquer dispositivo (*3D everywhere*): de celulares a set-top-boxes na TV interativa. O X3D pretende atingir este objetivo, mantendo uma estreita relação com a linguagem XML, relação esta que simplifica a autoria de mundos virtuais a ponto de poderem ser gerados através de rótulos (tags) XML. Com a simplificação da programação exigida para a construção de mundos virtuais, e a emergência de ferramentas de autoria X3D, é esperado que aplicações 3D possam ser criadas em escala, por um grande número de autores que não precisam ser proficientes em programação. Outras características que serão futuramente adicionadas ao X3D são: suporte gráfico 2D, *streaming* de áudio e *streaming* de vídeo. O desenvolvimento do X3D está sendo liderado pelo Grupo Tarefa X3D do Consórcio Web3D *(Web 3D Consortium)*, uma organização formada por mais de 50 empresas da área de alta tecnologia com o objetivo de estabelecer um conjunto de padrões 3D compatíveis para a Internet (www.web3d.org).

4.1.3 Java 3D

Java 3D é uma extensão da plataforma Java 2 JDK introduzida em meados do ano de 1997 pela *Sun Microsystems*, possui um conjunto de Interfaces de Programação de Aplicação - APIs que são usadas para a criação e manipulação de ambientes virtuais na forma de aplicações isoladas ou *applets* 3D na Web.

A integração natural do Java 3D não só com a linguagem Java, mas também com toda a família Java Media (Java Media Framework, Java Sound, Java Advanced Imaging e outras), aliado ao fato do Java 3D suportar funções as quais melhoram o desempenho e de usar aceleração em hardware, cria o potencial para a geração de ambientes virtuais multiusuário, integrados a multimídia, com alto grau de complexidade, que são construídos pela comunidade de desenvolvedores Java, sem muito esforço.

Como qualquer API de alto nível, Java 3D é projetada para explorar APIs tradicionais de baixo nível como OpenGL, e Direct3D. Recentemente, por meio de interações entre a SUN e comunidade VRML, foi concebido o Xj3D, um conjunto de interfaces,

extensões de bibliotecas e aplicações, todas baseadas em Java, as quais permitem VRML 2.0 e X3D serem exibidos em um ambiente Java 3D. **[http://www.xj3d.org/]**

4.1.4 Middleware

O recente aumento da complexidade de programação em consoles de jogos, aliado ao fato da expectativa dos jogadores pelo lançamento de produtos inovadores em um curto espaço de tempo, mostra que o método tradicional de desenvolvimento de jogos "faça tudo você mesmo" já não é mais viável comercialmente. Para manterem-se competitivas e lucrativas, as empresas desenvolvedoras de jogos devem terminar seus produtos em um período relativamente curto de tempo, mas com um alto grau de qualidade. Uma solução para este problema é o uso de componentes genéricos prontos para o uso e projetados para realizar uma específica tarefa em um jogo, como exibição de imagens 3D ou simulação do comportamento físico de objetos. Estes componentes são desenvolvidos por outras companhias e fornecem soluções confiáveis e seguras para um domínio de aplicações, além de suportar várias plataformas, tanto de consoles domésticos como PCs e MACs.

Esta abordagem deixa os desenvolvedores livres para concentrar em outros aspectos do projeto, como diversão e "jogabilidade". Em contrapartida, este método de desenvolvimento inevitavelmente não atinge resultados finais tão bons quanto seriam se fosse construída uma solução específica para a necessidade dos desenvolvedores. Em um mercado onde originalidade e distinção significam muito para o sucesso do jogo, o uso de *middlewares* pode resultar em semelhanças no produto final com soluções de competidores que utilizam o mesmo *software*. [Rico02]

Antes de 2002, soluções "pseudomiddleware" eram usadas somente em construções de jogos do mesmo gênero, por meio de motores *(engines)* licenciados, como exemplo, Id software Quake III Engine, utilizado em vários jogos de tiro em primeira pessoa (*Soldier of Furtune II, Medal of Honor Allied Assault, Jedi Kngth II*), e destinado somente à plataforma PC. Durante o ano de 2002 vários jogos de alta qualidade e de grande sucesso foram desenvolvidos usando *middlewares*, por exemplo, Morrowind**,** GTA: Vice City, Tom Halk Pro Stalkers 4[Ground02]. Os dois jogos mais vendidos em 2002 foram concebidos usando a RenderWare Plataform, uma solução Middleware que suporta as plataformas PCs, Playstation 2 e XBox.

4.1.5 O padrão MPEG-4

O MPEG-4 é uma solução padronizada da *ISO/IEC (ISO/IEC JTC1/SC29/WG11),* desenvolvida pelo *MPEG (Moving Picture Experts Group),* para a codificação e entrega de diferentes formatos de mídia sobre uma ampla variedade de redes e plataformas computacionais. O potencial do MPEG-4 consiste na codificação e entrega separada ou integrada de diferentes conteúdos (áudio, vídeo, 2D, 3D, classes Java etc.) por meio de Fluxos Elementares (*Elementar Streams* – ES) levando à construção de complexas aplicações multimídia. Parte do MPEG-4 define a estrutura BIFS (*BInary Format Scene*) que codifica cenas de maneira extremamente compacta e eficiente (10 a 15 vezes menores que arquivos VRML correspondentes) [KAN01].

Como o MPEG-4/BIFs é baseado no VRML97, todo o conteúdo desenvolvido nesta linguagem poderá ser acessado e visualizado em diferentes terminais MPEG-4 (TV interativa, celulares de 3ª. geração), bastando para isso converter o VRML97 para o formato BIFS - a conversão é simples e existem conversores eficientes para isso [Cortez02]. O padrão MPEG-4 também adiciona várias especificações de nós, em relação ao VRML (112 VS 54).

Soluções *Middlewares*, apesar de apresentarem melhores resultados gráficos que as tecnologias discutidas no início deste capítulo, são soluções proprietárias. Dentre as demais tecnologias, o padrão MPEG-4 se destaca pelo objetivo de suporte Multimídia e por possuir mecanismos eficientes de entrega e representação de dados. Mais ainda, o MPEG-4 é um padrão bem conhecido do grupo de pesquisadores do LRVNet (Laboratório de Realidade Virtual em Rede do DC/UFSCar), onde este trabalho foi desenvolvido. Desta forma, o padrão MPEG-4 foi escolhido como base para a construção de jogos 3D e, através de sua extensão MPEG-J, suportar adaptação de jogos para diferentes dispositivos e redes.

4.2 Padrão MEPG-4 – Camada de Sistema

Dentro da especificação do MPEG-4, a camada mais relacionada à produção e apresentação de ambientes 3D/2D é denominada Sistemas, onde as funcionalidades suportadas são:

- Descrição da cena para composição (sincronização espaço/temporal) de múltiplos objetos de mídia. A descrição da cena provê um rico grupo de nós para operadores de composição 2D/3D e primitivas gráficas;
- Texto com suporte a linguagens internacionais e seleção de estilo de fonte;

- Interatividade, que inclui: interação cliente/servidor; um modelo genérico de evento para tratar as ações do usuário, etc.;
- Ferramentas para intercalar múltiplos fluxos, incluindo informações de temporização (FlexMux);
- Independência na camada de rede;
- Mecanismos de sincronização, recuperação, temporização, etc.

Um dos objetivos do MPEG é evitar o aparecimento de tecnologias proprietárias, visualizadores de conteúdos (*players*) e formatos que sejam dependentes de plataformas específicas. Para isso, as seguintes medidas devem ser seguidas:

- Representar as unidades que contêm áudio, vídeo ou ambos, por meio de "objetos de mídia". Esses objetos de mídia podem ser de origem natural ou sintética, isto é, serem gravados, por meio de câmera, microfone, ou gerados por um computador;
- Descrever a composição desses objetos para gerar componentes de objetos de mídias que formam as cenas audiovisuais; multiplexar e sincronizar os dados associados aos objetos de mídia, de maneira que eles possam ser transportados, através de um canal de comunicação, com qualidade de serviço (QoS) adequada à natureza específica de cada objeto de mídia;
- Permitir a interação com a cena audiovisual gerada na máquina do usuário.

A Figura 8 ilustra um exemplo de uma cena MPEG-4. As cenas audiovisuais são compostas por alguns objetos de mídia, organizados de forma hierárquica, como uma estrutura em árvore, conforme ilustrado na Figura 9 (a árvore da Figura 8 refere-se à Figura 9). Os objetos de mídia como imagens, vídeo, áudio, etc., são representados nas folhas da árvore. A árvore não é necessariamente estática - nós podem ser adicionados, substituídos ou removidos, e as propriedades dos nós serem alteradas, como por exemplo, os parâmetros de posicionamento.

O MPEG-4 implementa a descrição da cena no formato BIFS (*BInary Format for Scenes*). As funcionalidades as quais podem ser inseridas dentro de uma cena incluem:

- Posicionamento de objetos de mídia em qualquer espaço dentro do sistema de coordenadas;

- Aplicação de transformações para modificação de atributos de objetos de mídia; Agrupamento de objetos de mídia, que possibilita a formação de objetos de mídia compostos;
- Aplicação de fluxo de dados nos objetos de mídia;
- Alteração do campo de visão do usuário;
- Disparo de eventos quando da seleção de um objeto específico, por exemplo, a iniciação ou interrupção de um vídeo. Tipos complexos de comportamento podem ser disparados, como por exemplo, uma cena onde um telefone virtual toca, o usuário atende e um elo de comunicação é estabelecido;
- Seleção do tipo de linguagem desejada quando múltiplas linguagens estão disponíveis.

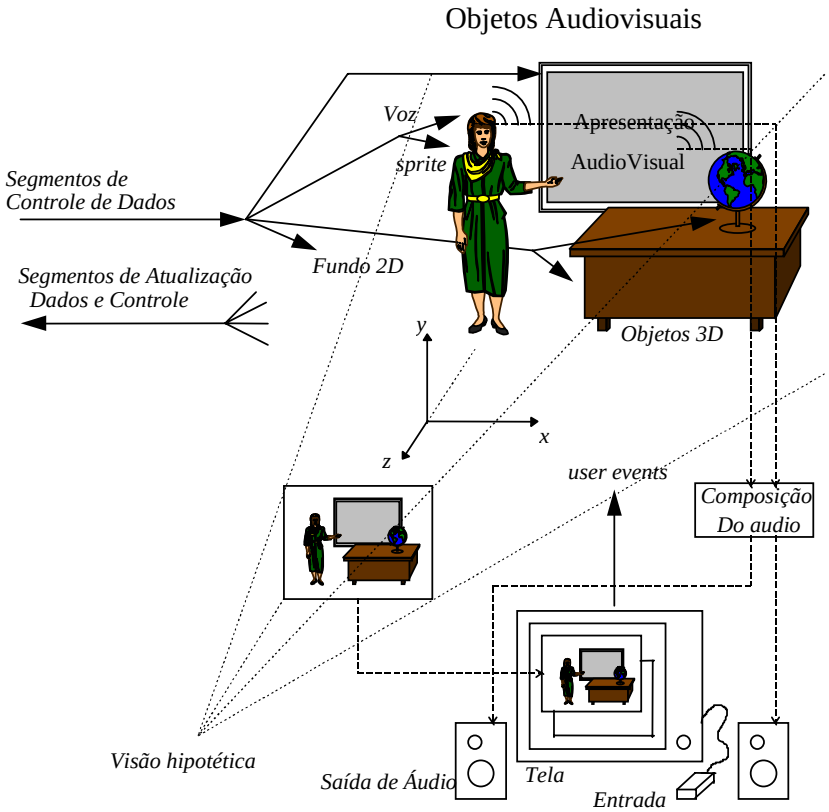

Figura 8 Exemplo de uma Cena MPEG-4

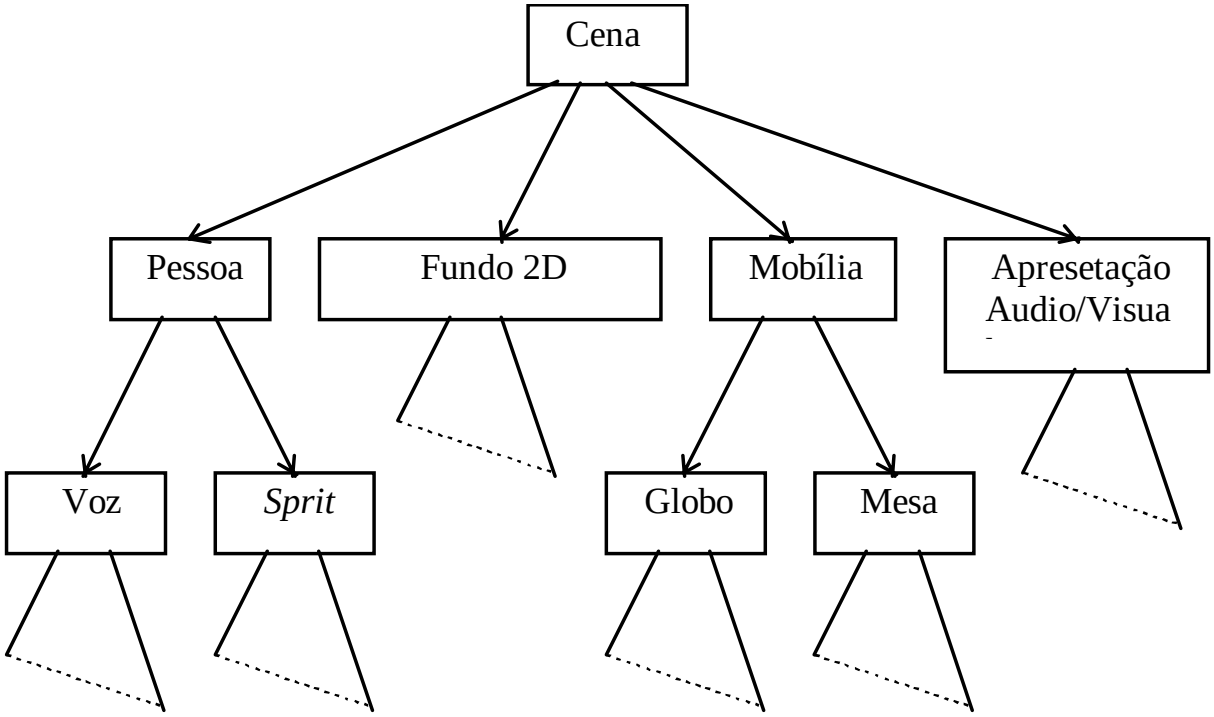

Figura 9 Estrutura Hierárquica de uma Cena

A descrição de uma cena em MPEG-4 estende os conceitos da linguagem VRML para permitir as funcionalidades descritas acima. A Figura 10 ilustra os componentes de um sistema MPEG-4.

4.2.1 Fluxos Elementares

Objetos de mídia e BIFs são todos convertidos em fluxos de dados (*streams*) que são transportados em um ou mais fluxos elementares (ES - *Elementary Stream*). No MPEG-4, qualquer tipo de dado é transportado através de um fluxo contínuo (*streaming*). O termo ES refere-se ao dado que contém totalmente ou parcialmente a representação codificada de um simples áudio ou vídeo, descrição da cena ou informação de controle. Todos os ES são identificados e caracterizados por um descritor de objetos, o que permite manipular os dados de forma hierárquica, bem como associar meta informações sobre o conteúdo dos dados e direitos autorais.

Cada fluxo elementar contém somente um tipo de dado, por exemplo, fluxo de imagens (JPEG) ou áudio (G723). Os fluxos elementares são decodificados usando seus decodificadores de fluxo específicos.

Cada fluxo de dados é caracterizado por um grupo de descritores que trazem informações de configuração, como por exemplo: determinação dos decodificadores necessários e o tempo de decodificação. Os descritores trazem também, indicações de QoS

necessárias para a transmissão da informação Multimídia, como prioridade, taxa máxima de velocidade, etc.

4.2.2 Sincronização

A sincronização dos ESs é realizada através de marcas de tempos (*time stamping*) de unidades de acesso individuais dentro dos próprios ESs. A identificação de tais unidades de acesso e a sincronização das marcas de tempo são executadas na camada de sincronização. Independente do tipo de mídia, esta camada permite a identificação das unidades de acesso (quadros de vídeo ou áudio e comandos de descrição da cena) em fluxos elementares, recriando os tempos base dos objetos de mídia ou a descrição da cena, permitindo assim a sincronização entre eles.

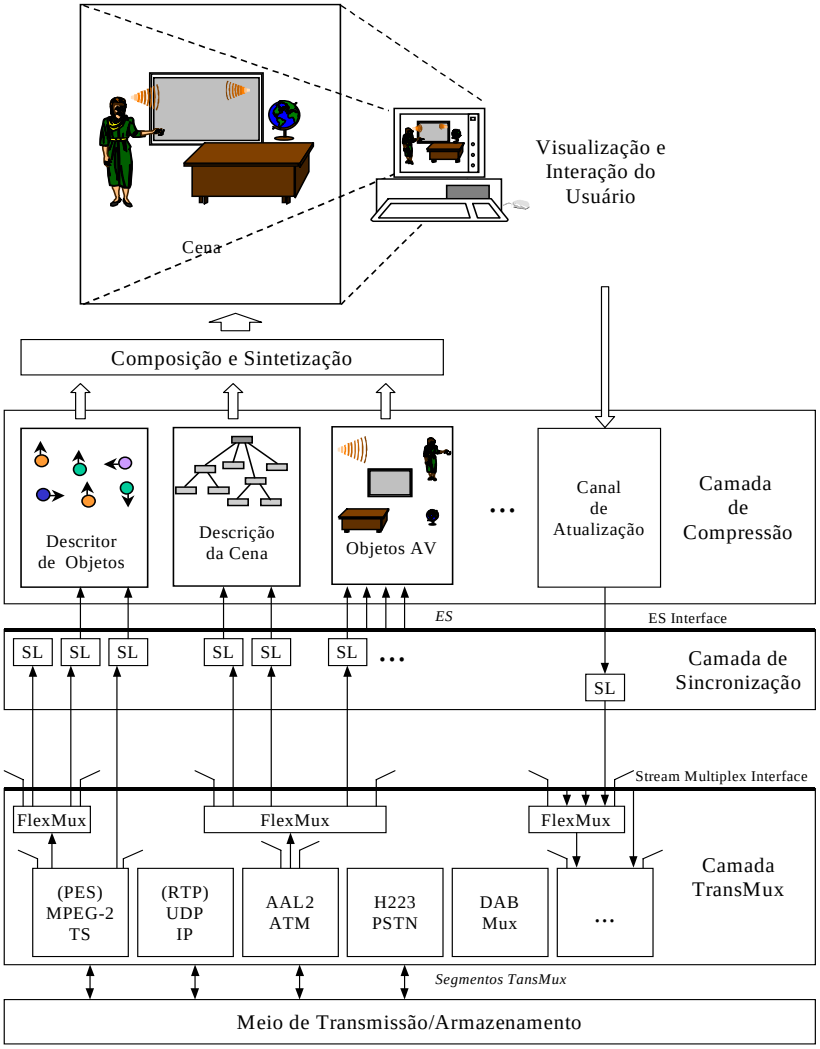

Figura 10 COMPONENTES DE UM SISTEMA MPEG-4

Na camada de distribuição, os fluxos de dados podem explorar diferentes QoS que estiverem disponíveis. Esta camada contém dois níveis de multiplexação: o primeiro nível, denominado FlexMux, e o segundo nível denominado TransMux.

O FlexMux permite agrupar ESs com baixa sobrecarga de multiplexação. A multiplexação neste nível pode ser usada, por exemplo, para agrupar ESs com requisitos similares de QoS, reduzindo o número de conexões ou o atraso fim-a-fim.

O segundo nível, TransMux, oferece os serviços de transporte e os requisitos de QoS. Somente a interface deste nível é especificada pelo MPEG-4, tornando possível utilizar qualquer tipo de protocolo existente como RTP, UDP, TCP, ATM ou MPEG-2 Transport Stream, para criar uma instância do TransMux, permitindo que o MPEG-4 seja usado em uma grande variedade de plataformas.

Tanto o FlexMux, como o TransMux, são definidos e coordenados de acordo com a especificação do DMIF – *Delivery Multimedia Intregration Framework*. O DMIF define os mecanismos de gerenciamento de sessão, canais de transporte e o fluxo dos ESs através destes canais. O gerenciamento das sessões e dos canais de transporte é visível para o MPEG-4, através unicamente da interface DAI (*DMIF Interface*). Uma aplicação que usa esta interface, não precisa conhecer como as camadas inferiores interpretam esses comandos. Cada rede necessita, por exemplo, definir somente como mapear os fluxos MPEG-4 aos seus canais de transporte.

4.2.3 Objetos de Vídeo e Áudio

No MPEG-4 o conceito de *Vídeo Object* (VO), corresponde a um segmento de *bits* que um usuário pode acessar ou manipular (como as operações de *cut* e *paste*). No lado do codificador, a informação da codificação é enviada para indicar onde e quando cada VO será exibido. Do lado do decodificador o usuário pode mudar a composição da cena exibida e interagir com ela.

Os *Video Objects* superam as limitações de padrões como JPEG ou MPEG, onde as imagens ou vídeo são representados como matrizes retangulares e comprimidas neste formato. Um vídeo MPEG-2 é representado como uma seqüência de quadros de tamanho fixo, este modo de representação não possibilita a capacidade de distinguir elementos diferentes que compõem cada quadro. Para aplicações em que não existe interatividade, este formato mostrou-se bastante eficiente, mas não para aplicações que necessitam de interatividade. Por

exemplo, o usuário pode apontar para uma região particular, "*hot regions*", para obter informações a respeito de um assunto, tornando necessário compor a cena com vários objetos.

Este tipo de representação requer que o esquema de compressão possa lidar com formas arbitrárias bem como o receptor tenha também a capacidade de compor a cena.

A camada 2 do MPEG-4 trata a forma como o vídeo é utilizado dentro do padrão MPEG-4, permitindo codificar tanto imagens ou vídeos naturais quanto imagens sintéticas. As principais funcionalidades suportadas na parte 2 são:

- Taxa de *bits* entre 5 *kbit*/s e 4 *Mbits*/s;
- Compressão eficiente, a qual inclui codificação de texturas com qualidade ajustada entre aceitável, para altas taxas de compressão, até as que não sofrem qualquer perda;
- Acesso aleatório, etc.

O MPEG-4 padroniza a codificação de áudio em taxas que variam entre 2 *kbit*/s até 64 *kbit*/s. O padrão NBC do MPEG-2, que é incorporado no MPEG-4, trata a codificação acima dos 64 *Kbit*/s. Várias funcionalidades estão disponíveis para cobrir os vários tipos de aplicações, como controle de velocidade ou recuperação de erros.

4.3 A extensão MPEG-J do padrão MPEG-4 para suporte a adaptações.

O MPEG-J [ISO02] foi introduzido na versão 2 do MPEG-4, e consiste de um conjunto de APIs (*Application Programming Interface*) os quais possibilitam que aplicações e *applets* Java possam acessar os componentes de um *player* MPEG-4. O MPEG-J suporta a manipulação da cena, permitindo respostas apropriadas para os eventos da rede, do servidor ou das entradas do usuário.

Uma aplicação MPEG-J pode ser local ou remota. Quando uma aplicação é remota, ela deve implementar a interface MPEGLet. Da mesma maneira que ocorre com os *applets* Java, aspectos relacionados à segurança devem ser considerados quando a aplicação for remota. Uma aplicação é distribuída da mesma maneira que os demais fluxos o são (vídeo, áudio, descrição da cena – BIFs, etc.), através de um Segmento Elementar - ES, ou seja, uma aplicação MPEG-J remota possui um descritor de objeto, para que possa ser multiplexada e distribuída dentro de um arquivo no formato MPEG-4.

A arquitetura do MPEG-J, incorporado a um sistema MPEG-4, é ilustrada na Figura 11. As aplicações Java podem acessar todos os componentes de um *player* MPEG-4,

através das APIs. Tais componentes incluem os Recursos de Apresentação e Execução, Decodificadores, Recursos de Rede e a Cena Gráfica.

Os pacotes que compõem atualmente as APIs do MPEG-J são:

- *org.iso.mpeg.mpegj.scene* - Acesso à cena gráfica, que permite a criação e modificação dos nós da cena;
- *org.iso.mpeg.mpegj.resourceManager* - Gerenciamento dos recursos do sistema, acesso estatístico e as características dinâmicas do *player*;
- *org.iso.mpeg.mpegj.network* - Acesso e controle dos decodificadores usados;
- *org.iso.mpeg.mpegj.decoder* - Acesso e controle dos componentes de rede de um *player* MPEG-4.

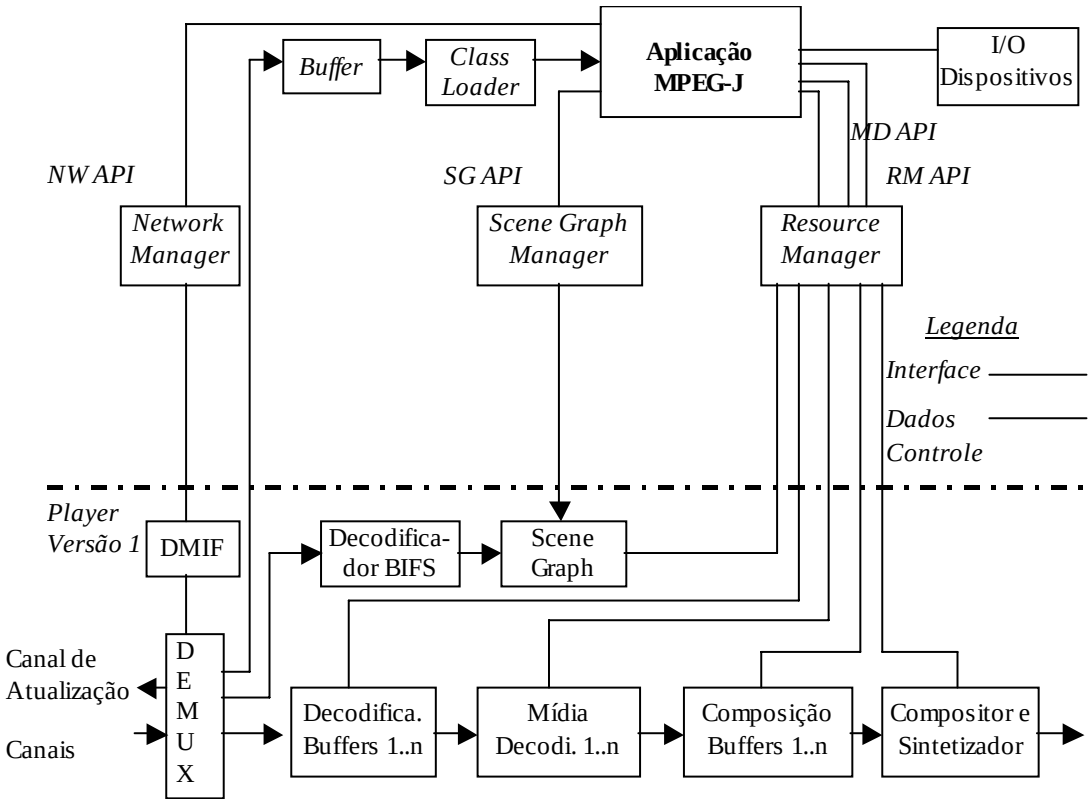

Figura 11 Interação dos Componentes MPEG-4 com o MPEG-J

Para que uma aplicação MPEG-J possa acessar os gerenciadores implementados em cada terminal, ela deve fazer uso da classe *MPEG-J Terminal*, que possui os métodos específicos para o acesso a cada gerenciador (g*etNetworkManager( ), getResourceManager( )* e *getrSceneManager ( )*).

Entre as atuais APIs especificadas pelo MPEG-J, a API *Scene Graph Manager* é a parte mais importante para a implementação de sistemas multiusuário. Através do acesso à

cena gráfica, uma aplicação modifica e controla o modo em que a cena é apresentada. Por exemplo, quando um usuário entra em um ambiente virtual multiusuário, é necessário criar a sua representação dentro do ambiente virtual (avatar). Isto significa que cada aplicação participante do ambiente virtual terá que acessar a cena gráfica para criarem novos nós referentes ao novo usuário (avatar) que entrou no ambiente virtual.

Para uma aplicação MPEG-J fazer uso de mecanismos de adaptação, ela deve fazer uso da API Resource Manager. Estas APIs são usadas para monitorar os recursos do Terminal MEPG-4, sejam eles estáticos ou dinâmicos. Suas principais classes são:

- Renderer: usado para fornecer informações de recursos do Renderizador, como a quantidade de quadros por segundo
- Capability Manager: usado para fornecer recursos do terminal, sejam eles estáticos, como Memória total, ou dinâmicos, como a carga da CPU.
- MPDecoder: usado para acesso a mídias, codificadores, e eventos do terminal MPEG-4.

4.3.1 Iniciando uma aplicação remota MPEG-J

Uma sessão MPEG-J não é iniciada enquanto o *player* MPEG-4 não receber um descritor de objeto MPEG-J. Após o recebimento, o *player* segue os seguintes passos: Abre um novo canal para o fluxo elementar - ES da aplicação MPEG-J. O fluxo é tratado da mesma maneira que os demais fluxos (vídeo, áudio, etc.). O fluxo elementar é dividido em Unidades de Acesso. As Unidades de Acesso MPEG-J são enviadas para o *Class Loader*, que efetua a carga de todas as classes da aplicação. Um “decodificador” MPEG-J é responsável por receber as Unidades de Acesso e tomar as decisões para executar a aplicação. Podem existir um ou mais pontos de entrada em um fluxo MPEG-J. A cada ponto de entrada, um novo *thread* é disparado, iniciando a sua execução.

O ciclo de vida de um MPEGLet, é muito similar a uma *applet* Java. A interface MPEGLet possui os métodos *init(), run(), stop()* e *destroy()*. Antes de iniciar um MPEGLet, o método *run()* é ativado em um *thread* separado. Os métodos *stop()* e *destroy()* são similares a um *applet* Java. Se outro fluxo de *bits* MPEG-J é recebido pelo *player*, ele é iniciado e tratado em um *thread* diferente.

4.3.2 Distribuição de Dados MPEG-J

Uma aplicação MPEG-J é distribuída para um *player* MPEG-4 como um fluxo elementar específico. Os dados MPEG-J podem ser classes ou objetos serializados, enviados dos servidores para os clientes. Os dados MPEG-J devem ser distribuídos conforme os seus requisitos de tempo de execução - isto é assegurado por meio de arquivos de classes ou objetos serializados usados juntamente a Arquivos de Cabeçalho Java (*Java Stream Header*). O Arquivo de Cabeçalho Java é anexado a cada classe ou objeto antes de ser passado para a Camada de Sincronização durante o processo de codificação. Após este empacotamento, qualquer mecanismo de transporte que trate dos aspectos relacionados ao tempo, como FlexMux, RTP e MPEG-2 *Transport Stream*, pode ser usado para transportar os dados para o cliente.

Problemas de perdas de pacotes podem ocorrer durante o envido dos dados MPEG-J para os clientes, causando atrasos na execução dos programas. As possíveis soluções para tratar a perda de pacotes são:

- Retransmissão de toda a classe em intervalos regulares quando o canal de atualização (*back channel*) não estiver presente no *player*. Esta solução é inviável quando existir um amplo número de clientes ou quando classes ou objetos possuírem grandes tamanhos.
- Sinalização da perda para o servidor que então retransmite o pacote perdido, quando o canal de atualização estiver presente.

O MPEG-J não designa qualquer tipo de esquema específico para o tratamento de perdas de um fluxo de dados MPEG-J, pois as perdas de pacotes não são tratadas na camada do MPEG-J. É assumido que a camada de transporte garante a entrega confiável dos dados.

Opcionalmente, mecanismos de compressão podem ser usados para melhorar a eficiência na transmissão dos dados. Arquivos Java utilizam o pacote *java.util.zip* para realizar a compressão e descompressão dos dados.

4.4 O Uso do padrão MPEG-4 no suporte a ambientes virtuais multiusuário em Dispositivos Heterogêneos

A tecnologia de suporte a ambientes virtuais multiusuário é baseada em mecanismos de compartilhamento. Na especificação do padrão MPEG-4 [ISO02], duas abordagens principais de compartilhamento são identificadas:

- ***Mecanismos baseados em protocolos*** (como o H323 e VRTP). Estes mecanismos definem um conjunto de mensagens os quais fluem entre os objetos multimídia que implementam os mecanismos de compartilhamento;
- ***Mecanismos baseados em APIs*** (como o CORBA). Estes mecanismos definem uma hierarquia de objetos, bem como APIs de comunicação entre os objetos comunicantes do ambiente virtual compartilhado. Os bits que fluem pelo meio de transmissão são tratados por uma camada inferior de *middleware.*

Ambos os mecanismos possuem pontos fortes e fracos que satisfazem diferentes mercados. Segundo a especificação do MPEG-4 [ISO02], os mecanismos baseados em APIs estão amadurecidos em termos de padronização, mas não são adaptáveis para suportar ambientes que possuem baixa largura de banda e processamento limitado, como é o caso de dispositivos ubíquos do tipo celulares e *PDA`s*. Esta limitação poderia ser superada por meio da definição de protocolos otimizados de comunicação, bem como a seleção cuidadosa de conjuntos de objetos disponíveis num determinado momento da aplicação. Entretanto essas soluções remeteriam à primeira abordagem: mecanismos baseados em protocolos, cuja solução de padronização não atende a todos os requisitos de suporte a ambientes virtuais compartilhados.

O padrão emergente MPEG-4 MU foi gerado a partir de uma rigorosa Chamada para Propostas (ISO/IEC JTC1/SC29/WG11/N3574) que endereça um conjunto completo de requisitos para o suporte a ambientes virtuais compartilhados. Os ambientes virtuais multiusuário permitem que vários usuários, representados por avatares, compartilhem o mesmo ambiente, interagindo entre si e com os objetos ali presentes.

O Padrão MPEG-4 MU trata ambientes virtuais 3D compartilhados, onde existem mais de um usuário interagindo com o ambiente e com outros usuários em tempo real. As ações multiusuário são bem simples e o seu relacionamento estrutural pode ser definido como [Living97]:

- Modificações na cena podem ser iniciadas por qualquer cliente;
- Modificações devem, potencialmente, serem sincronizadas em todos os clientes;
- Para todas as modificações, cada objeto, ou é a sua origem, com a responsabilidade para espalhar a sua notícia de modificação, ou é um receptor de eventos de modificação, e deve realizar a ação apropriada;

- Nem todas as modificações são realizadas de uma única vez; a otimização do desempenho depende do conhecimento de prováveis mudanças que podem acontecer em seguida;
- Nem todas as notícias de modificação precisam ser comunicadas, a otimização depende também do conhecimento de quem necessita saber, o que, e como.

Um dos maiores desafios presente nos ambientes virtuais 3D compartilhados é manter a consistência das informações entre os múltiplos usuários. Nesses ambientes estão compartilhados não somente o espaço virtual, mas também os objetos nele contidos. Sendo assim, pertencem a um ambiente virtual 3D compartilhado, seções e zonas de compartilhamento e objetos com estado compartilhado ou não [ISO02].

De acordo com a especificação do MPEG-4 MU [ISO02], uma seção é um recipiente para uma ou mais zonas de compartilhamento; uma zona é um recipiente para um ou mais objetos compartilhados; um objeto compartilhado é definido como sendo qualquer objeto no ambiente virtual 3D que possua um estado de compartilhamento. Dessa forma, um ambiente é declarado como sendo compartilhado através da definição de uma seção de compartilhamento no ambiente.

Após a inserção de uma seção de compartilhamento, várias zonas podem ser definidas, contendo vários objetos compartilhados. A inserção de zonas de compartilhamento em um ambiente virtual 3D garante benefícios, sendo os principais: [ROEH97]:

- Melhora o desempenho de renderização;
- Reduz os requisitos de memória, até mesmo para mundos extensos;
- Agiliza o carregamento dos mundos;
- A habilidade para designar propriedades (iluminação, nevoeiro, etc.) diferentes para partes diferentes do mundo;
- Maior gerenciamento na detecção de colisão;
- Auxilia na limitação de atualizações num sistema multiusuário;

Dessa forma, a delimitação de ambientes virtuais 3D em seções e zonas de compartilhamento auxilia tanto no projeto e desenvolvimento de ambientes com grandes extensões, quanto nas questões de gerenciamento de atualizações.

Mundo, Cena, Zona, Sessão e Objeto Compartilhado são termos do padrão MPEG-4MU que caracterizam um ambiente virtual multiusuário. Tais termos são descritos como:

- *Mundo:* Uma ou mais cenas conectadas tecnicamente e conceitualmente.

- *Cena:* Um conjunto de objetos ou nós com comportamento específico e uma lista opcional de rotas responsáveis por propagar eventos entre objetos. Uma cena é representada por um grafo de cena onde as folhas são nós [Roehl97].
- *Nó:* é um objeto básico no padrão MPEG-4, cujo agrupamento é usado para organizar e gerenciar cenas com múltiplas mídias.
- *Zona:* Uma porção contígua de uma cena; um recipiente para Objetos Compartilhados.
- *Sessão:* um recipiente para zonas compartilhadas de uma cena.
- *Objeto* Compartilhado: Qualquer objeto cujo estado e comportamento são sincronizados entre múltiplos clientes.

O suporte para Ambientes Virtuais multiusuário no MPEG-4 está em fase de padronização. No padrão emergente MPEG-4 MU, a manutenção da consistência do ambiente colaborativo é realizada, através do mecanismo Pilot/Drone e do protocolo *BIFS-Command*, usados para refletir as mudanças de estado dos objetos compartilhados entre todos os participantes.

A arquitetura multiusuário do padrão emergente MPEG-4 MU é baseada no mecanismo Pilot/Drone proposta pela especificação Living Worlds [Living99]. De acordo com a especificação citada, os *Pilots* são cópias mestres de nós em clientes ou em servidores, onde mudanças de estados ou comportamentos são replicadas para outras instâncias, isto é, os *Drones* (réplicas de Pilots) que correspondem diretamente aos seus Pilots. A Figura 12 ilustra este mecanismo: os círculos cinza representam os *Pilots*, correspondentes a objetos compartilhados, e os brancos, os *Drones.*

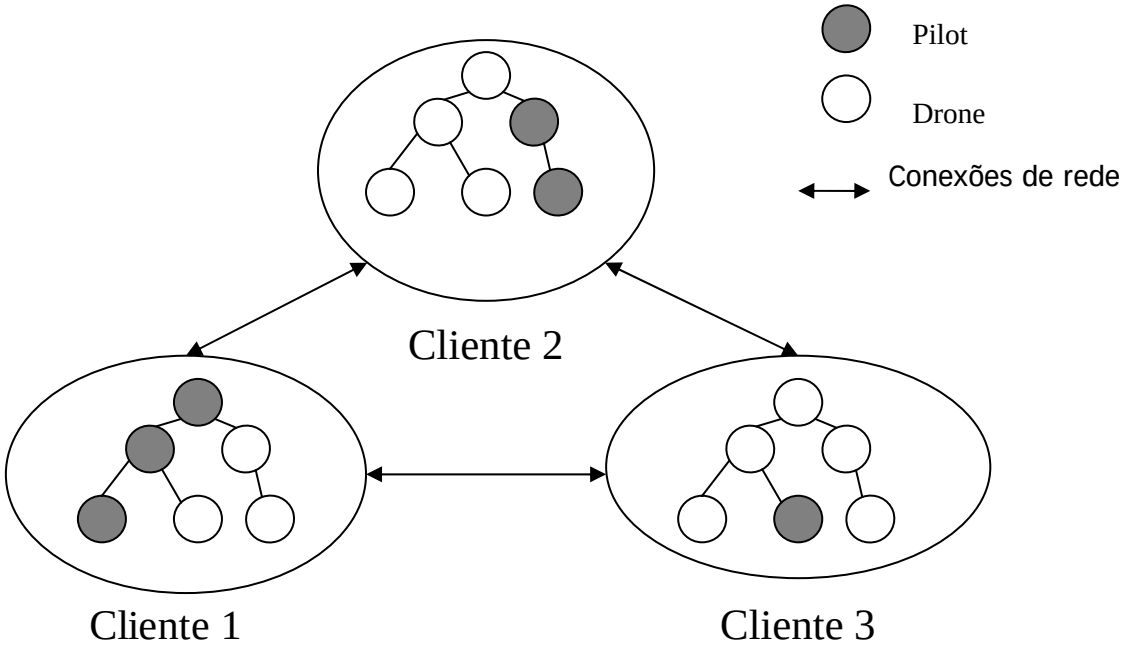

Figura 12 Representação Pilot/Drone e grafos de cenas em terminais clientes

Todo objeto compartilhado possui seu *Pilot* o qual é capaz de originar comportamentos e mudanças de estado. A melhor definição para o par Pilot/Drone é o próprio avatar, que é a representação tridimensional de um usuário em um ambiente virtual interativo.

4.4.1 Componentes da Arquitetura do MPEG-4MU

Para a sincronização de objetos compartilhados no MPEG-4MU, vários componentes controlam e gerenciam as mensagens de solicitação e atualização dos estados/comportamentos dos objetos compartilhados, enviados pelo protocolo *BIFS Command*. A Figura 13 ilustra os componentes da arquitetura MPEG-4 MU (A parte pontilhada representa um terminal cliente).

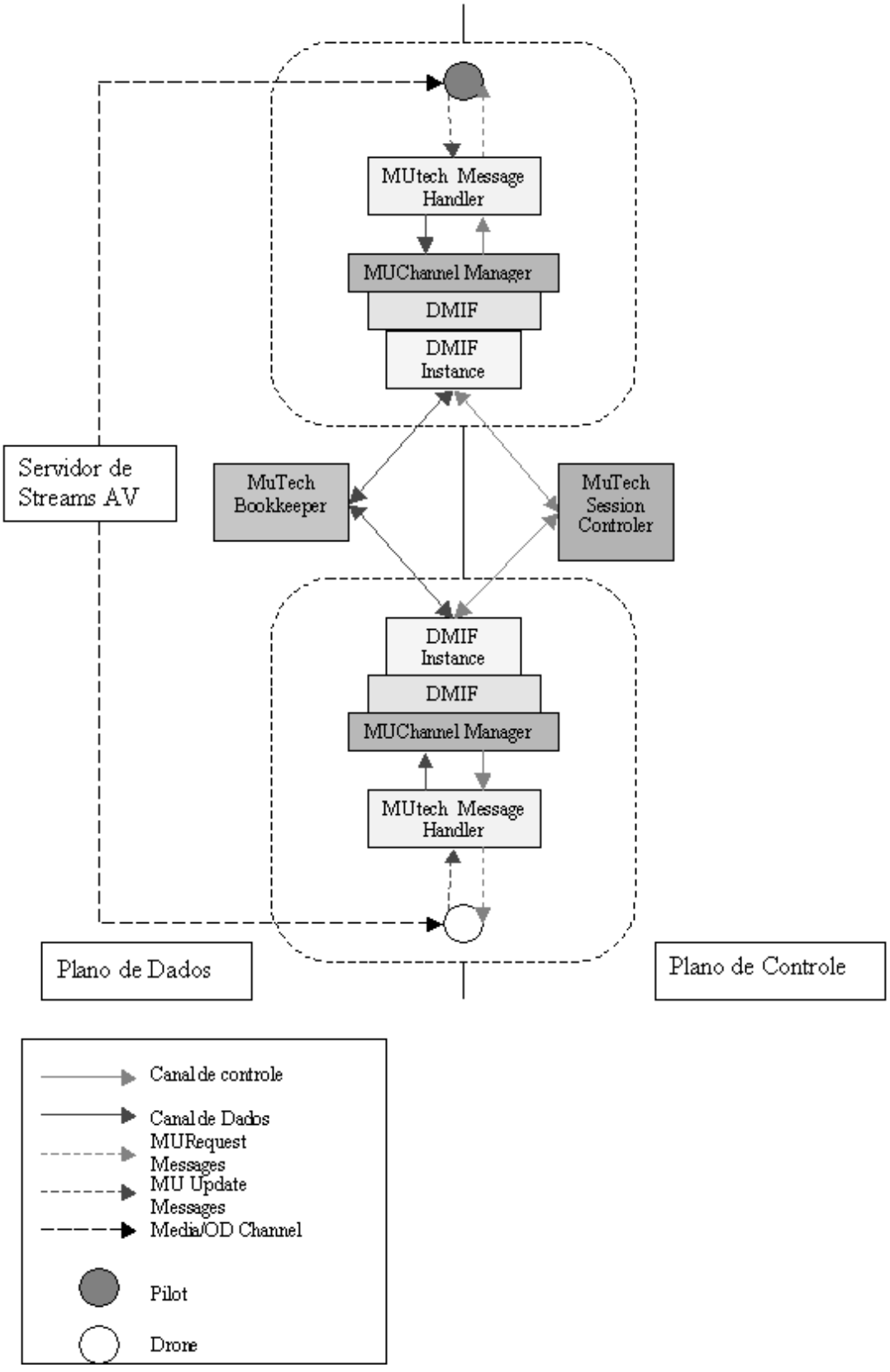

Figura 13 Arquitetura MPEG-4 MU [ISO01]

Os principais componentes da arquitetura, ilustrada na figura, são: O *MUTech Session Controller* (MSC, Controlador de Sessão), *MUTech Bookkeeper* (MBK), *MUTech Message Handler* (MMH, Controlador de Mensagem), e o *MUTech Chanel Manager* (MCM).

O MSC é responsável pelo gerenciamento de uma sessão multiusuário e sua principal função é fornecer a relação das zonas existentes na sessão, a instância do *Bookkeeper* associado e o gerenciamento de permissões de entrada/saída de participantes numa sessão ou adição/remoção de zonas.

O componente *MUTech Bookkeeper* (MBK) é o responsável por controlar todas as zonas compartilhadas e os objetos nela contidos. Este controle se dá através do gerenciamento de mensagens enviadas pelo protocolo *BIFS-Command.* Estas mensagens são propagadas para os terminais cliente subscritos nas zonas.

O *MUTech Message Handler* (MMH) provê uma interface, no terminal cliente, para o envio e recepção de mensagens de atualização de estado dos objetos compartilhados, etiquetando estas mensagens com a identificação (*id*) do cliente antes delas serem enviadas.

O MCM é um componente auxiliar para configurar fluxos de dados e controle entre cliente e MSC, por meio do DMIF. Mensagens multiusuário são enviadas por meio de canais abertos no servidor DMIF e módulos do cliente.

O servidor de fluxos (AV Stream Server) supre os nós da cena com fluxos MPEG-4 apontados pelo descritor de objetos. Isso permite o uso de *unicast* ou *multicast*, arquivos locais, entre outros fatores para distribuição e compartilhamento de fluxos entre os terminais clientes, não sendo necessárias mudanças na descrição da cena.

Para exemplificar melhor, quando um participante deseja ingressar no ambiente compartilhado, o MSC é responsável por aceitar ou recusar a sua entrada. Se aceito, o participante é alocado para uma zona da sessão multiusuário e habilitado para interagir com os demais participantes e com os objetos compartilhados.

Dessa forma, se o participante efetuar alguma ação que modifique o estado do seu avatar (por exemplo, caminhar pelo ambiente) ou de algum objeto compartilhado, esta modificação irá disparar o envio de mensagens de atualização, gerenciado pelo *Bookkeeper*

De modo a otimizar a implementação do mecanismo *Pilot/Drone*, nós adicionais foram incluídos na arquitetura do MPEG-4 que geram ações imediatas de atualização do ambiente virtual, mediante eventos advindos do usuário e/ou da aplicação. Estes nós incluem:

- *MUSession.* Nó recipiente para o depósito de cenas compartilhadas. Ele deve conter, obrigatoriamente, uma ou mais zonas compartilhadas.
- *MUZone.* É um recipiente para nós com compartilhamento de semânticas. Assim, objetos compartilhados ficam relacionados diretamente com suas zonas.

4.5 Considerações Finais

Neste capitulo foram apresentadas as tecnologias de suporte a ambientes 3D multiusuário, e as justificativas para a escolha do padrão MPEG-4 para o uso no *framework* de suporte a adaptação desenvolvido como parte deste trabalho.

No que se refere ao padrão MPEG-4, descreveu-se sua composição bem como a definição das especificações presentes na divisão do Sistema padrão, bem como sua extensão MPEG-J para incorporação de código Java na aplicação MPEG-4, e a extensão MPEG-4 MU usada para o suporte multiusuário a aplicações 3D em rede.

Uma importante consideração quanto à especificação da extensão MPEG-J é que, uma vez sendo o MPEG-J um padrão, a API descrita apenas define e especifica, mas naturalmente não é implementada, pois é dependente de plataforma e deve ser programada usando JNI - *Java Native Interface.* Mais ainda, a API somente está habilitada para o tratamento 2D, sendo que o tratamento 3D ainda está em processo de padronização.

Desta forma, a API MPEG-J não foi utilizada em sua total potencialidade neste trabalho. Para as demais funções do *framework* foram usadas primitivas desenvolvidas pelo autor do projeto.

As classes relativas à adaptação da API MPEG-J foram estendidas uma vez que as funções de monitoramento de recursos do sistema eram muito limitadas. Por exemplo, a função de monitoramento de CPU retorna a carga total do sistema, e não somente a carga da aplicação, o que não se enquadraria nas necessidades do projeto, caso houvesse concorrência entre aplicações, a informação não seria confiável.

Uma das contribuições para trabalhos futuros é submeter uma lista de atualização das APIs MPEG-J para serem incorporadas no padrão. O próximo capítulo descreve o *framework* proposto, bem como sua implementação e avaliação de resultados.

5 MMGAME - UM *FRAMEWORK* DE SUPORTE À ADAPTAÇÃO DE JOGOS 3D MULTIUSUÁRIOS, INTEGRADO À EXTENSÃO MPEG-J DO MPEG-4.

Este capítulo especifica um *framework* para uso em aplicações MPEG-J que auxilia o programador na inserção de mecanismos de adaptação para resolver três potenciais problemas:

- Portabilidade de jogos 3D em diversas plataformas;
- Variação dos níveis de recursos do ambiente que cerca o jogador, ocasionando perda de desempenho;
- Variação da complexidade da cena durante a execução de Jogos 3D, ocasionando perda de desempenho.

Adequando os dados da aplicação, consegue-se a portabilidade da aplicação, monitorando a flutuações dos níveis de recursos de sistema e aplicação, tomando decisões apropriadas, mantém-se o desempenho desejado pelo usuário. O *framework* foi chamado de MMGAME (**M**ulti-user **M**PEG-4 **G**ame **A**daptation **M**anag**E**r).

Com este *framework* o programador define os recursos a serem monitorados e quais mecanismos serão acionados para garantir as métricas de desempenho estipuladas pelo usuário. Estes mecanismos de adaptação são identificados dinamicamente e seu resultado é precisamente calculado para cada cliente.

O *framework* foi concebido para o uso em aplicações de entretenimento 3D multiusuários, no entanto pode ser usado em qualquer aplicação 3D que demande de interatividade em tempo real e performance, ou seja, que necessite uma mínima taxa de quadros por segundo. O *framework* proporciona também a execução dos jogos 3D MU em diversos dispositivos mesmo que a aplicação não foi concebia para o uso em ambientes ubíquos.

Os recursos monitorados pelo *framework* compreendem recursos de sistema, como largura de banda, carga de processador, memória disponível e recursos da aplicação, como taxa de quadros renderizada em um determinado instante, promovendo a adaptação de dados e controle.

O *framework* de suporte a adaptação em jogos tridimensionais multiusuários é composto dos seguintes componentes: Especificador de Desempenho(ED), Adaptador de

Conteúdo(AC) e Monitor de Recursos(MR), cada componente é representado por um processo. O ED e MR são processos presentes no terminal do cliente, sendo que o AC está presente no MSC (*MUtech Session Controller*).

O ED possui a função de determinar as capacidades do terminal, no tangente à execução de conteúdo 3D multiusuário e impacto dos mecanismos de adaptação, fornecendo todos os resultados para o programador. O *feedback* do monitoramento dos recursos em tempo real é de responsabilidade do MR, o AC promove os métodos de adaptação antes que o dado chegue ao terminal do usuário. Durante a execução da aplicação as adaptações realizadas no cliente são engatilhadas pela aplicação MPEG-J. A Figura 14 exibe a divisão dos componentes da arquitetura entre um terminal cliente e a rede de serviço, a comunicação entre os componentes também estão ilustradas.

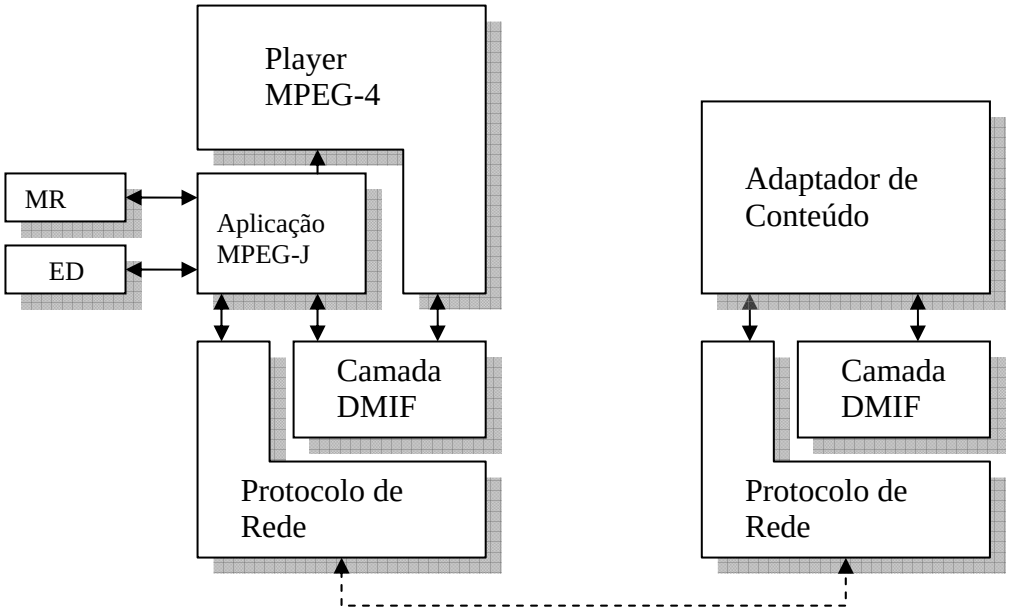

Figura 14 Comunicação entre componentes da estrutura

O *framework* utiliza de mecanismos de adaptação que respondem aos problemas relacionados ao projeto de aplicações adaptativas mencionados na seção 3.3 da seguinte maneira:

Quando adaptar?

A adaptação é feita antes dos dados serem transmitidos ao terminal cliente e durante a execução da aplicação.

Quando o ED verifica o potencial do terminal onde está sendo executando, ele decide ou não pela ativação de alguma adaptação, caso positivo, é um indicador que o terminal nãoé capaz de executar as mídias com a máxima qualidade de suas representações e define a necessidade de adaptação antes dos dados serem recebidos pelo terminal.

Por exemplo, quando o display de um PDA não suporta uma profundidade de cores maiores que 16 bits, todo o conteúdo de texturas é convertido para 16 bits de profundidade. Esta adaptação é feita antes dos dados do jogo serem recebidos do servidor, ou posteriormente, quando o usuário entra em outra seção MPEG-4, onde é feito o *download* do próximo ambiente 3D, e possivelmente mais mídias para aquela seção, todas estas adaptações são realizadas pelo AC.

A adaptação durante a execução da aplicação é acionada logo que o MR detecta que certo recurso ultrapassou um limite pré-estabelecido, seja este um limite inferior ou superior. Caso a disponibilidade de um recurso se torne maior, a política pode chegar à conclusão que não há mais necessidade de certa adaptação ou por escolher uma adaptação mais branda. Estas adaptações são realizadas quando a taxa de quadros por segundo ou a largura de banda no cliente está comprometida.

O que adaptar?

A aplicação MPEG-J requisita ao Adaptador de Conteúdo a adaptação de qualquer tipo de dado, desde que este tenha suporte para tal, a regra também é respeitada para adaptações realizadas no cliente.

Foram considerados mecanismos de adaptação que acarretassem mudanças nos dados e no controle da aplicação para alcançar maior desempenho e flexibilidade na escolha de mecanismos, proporcionando ao usuário, opções que preferencialmente adéqüem as suas necessidades.

Os mecanismos os quais adaptam os dados da aplicação são usados tanto no cliente quanto no AC, e podem ser divididos em dois grupos.

Mídias de trafego entre usuários: Mensagens de Voz e Vídeo quando aplicadas (tanto na transmissão, quanto na recepção).

Mídias de trafego entre o servidor e o usuário: Texturas, Arquivos de som, vídeo e o arquivo MPEG-4 que contém os dados geométricos 3D.

Os mecanismos que adaptam o controle são usados somente no terminal cliente para controlar a taxa de quadros por segundo renderizada. Um pequeno conjunto foi escolhido para tratamento no *framework*, os quais são mostrados a seguir.

- Diminuição da quantidade de objetos(não compartilhados) renderizados na tela
- Diminuição na produção de efeitos visuais baseados em partículas
- Exclusão, em alguns casos, do tratamento da física.
- Diminuição da quantidade de sons reproduzidos ao mesmo tempo.
- Exclusão, em alguns casos, de alterações feitas no ambiente.
- Diminuição do tempo da CPU concedida a IA dos jogadores controlados pela CPU.

Onde Adaptar?

Dependendo da ocasião a adaptação será feita no Cliente ou no AC.

Adaptações em função da largura de banda e capacidades do dispositivo são realizadas no AC, localizado no MSC, situado na rede de serviço ao usuário. Adaptações em função da variação dos recursos locais, excluindo a largura de banda, e taxa de quadros por segundo são realizadas no terminal do usuário.

Um cliente não realiza uma adaptação considerando os demais participantes, caso uma mensagem de áudio que o cliente A envia para o cliente B for transmitida em uma capacidade não suportada pelo destino, seja momentaneamente devido a largura de banda ou permanentemente devido a limitações de hardware, a função de sua adaptação fica a cargo do AC, porém quando o cliente A está com largura de banda comprometida ou carga da CPU reduzida, a adaptação é feita no cliente A, pois a mensagem deve ficar de acordo com suas capacidades, não importando se o cliente B suporta um formato superior ao enviado. Como os mecanismos de adaptação de dados acarretam perdas, não é possível aumentar a qualidade da mensagem quando ela passa pelo AC.

Como adaptar?

Um dado ou mídia pode ser decodificado usando várias configurações. A adaptação dos dados em uma aplicação MPEG-4 é realizada pela alteração da configuração ou mesmo pela troca de um decodificador/codificador associado a um determinado tipo de dado.

Uma mídia enviada de usuário para usuário pode ser recodificada no AC para adequar-se aos requisitos de largura de banda do usuário de destino. Caso haja uma troca de

codificadores no lado do servidor (AC), a mesma troca deve ser refletida no decodificador cliente para que não haja incompatibilidades.

O cliente deve possuir suporte para um determinado codificador/decodificador além de possuí-lo instalado junto ao Player MPEG-4. Doravante quando se falar em mecanismos de adaptação significa a troca de um codificador no AC, de um decodificador no dispositivo cliente, podendo ser acompanhado da alteração da configuração no decodificador.

Existem diversas maneiras de adaptar os dados em uma aplicação 3D, a existência destes mecanismos na aplicação dependerá de sua presença no Adaptador de Conteúdo, no terminal cliente, e a existência no jogo do tipo de dado que os mecanismos suportam.

Para facilitar o processo de decisão de como adaptar, o *framework* fornece métodos que examinam a capacidade destes mecanismos, em tempo de execução, quanto ao desempenho conseguido. A forma que um determinado mecanismo trabalha não é de preocupação do *framework*, o importante é que seja capturado em tempo real, o ganho de performance em um jogo particular em uma plataforma específica.

Uma aplicação 3D varia extremamente, por este motivo parte dos testes realizados pelo ED devem ser realizados para toda nova aplicação 3D.

A adaptação de controle necessita que o mecanismo seja disponibilizado por meio de uma interface, porém não é necessário nenhum decodificador ou codificador, pois alterar o controle significa modificar o fluxo lógico da aplicação, e não alterar um dado. Por exemplo, um jogo pode fazer o uso de tratamento de física ou simulação de IA nos personagens comandados pela CPU. Por este motivo a aplicação deve ser escrita de uma forma padronizada para o reconhecimento de adaptações de controle pelo *framework*.

Apesar de existir várias formas de se adaptar os dados em uma aplicação 3D, os métodos usados são praticamente os mesmos. Nesta especificação foram considerados decodificadores e codificadores básicos, presentes na maioria dos sistemas, que realizam as seguintes funções:

Dados Geométricos 3D: Compressão antes da transmissão e descompressão na recepção dos dados. Alta largura de banda pode significar que não seja necessária a compressão dos dados, pois o tempo necessário para a descompressão mais o tempo da transmissão pode ser mais longo do que o total necessário para o tráfego do dado não comprimido.

- Textura: Diminuição da resolução, quantidades de cores, alteração do formato da textura e do nível de compressão.

- Vídeo: Diminuição da taxa de quadros, resolução, cores, alteração na compressão, eliminação de quadros, eliminação completa da faixa de imagens, preservando apenas a faixa de áudio.
- Mensagens de Voz, Áudio e Áudio em vídeo: Diminuição do número de canais de saída, freqüência de amostragem do sinal, número de bits para codificação, e alteração na Compressão e formato.

Quanto da aplicação precisa ser modificado?

Para inserir mecanismos de adaptação em uma aplicação MPEG-J o programador deve inserir chamadas aos componentes do *framework* e parametrizar sua aplicação para tratamento de adaptação de controle.

Com a visão geral do funcionamento do *framework* apresentado, as seções seguintes apresentarão em detalhes os componentes do *framework*, suas respectivas implementações, o jogo usado como teste para validação do *framework* e os resultados obtidos.

5.1 Ciclo de vida de uma aplicação usando o MMGAME

Para melhor compreensão dos passos que são realizados desde o download inicial da aplicação até a sua finalização, esta sessão apresenta um ciclo de vida dos componentes do *framework* para ilustrar quando um componente é ativado e descartado.

Partindo do pressuposto que o usuário possua em seu terminal um player MPEG-4, os codificadores e decodificadores instalados, os passos a seguir descrevem a entrada de um usuário em um portal de Jogos 3D MPEG-4. A Figura 15 ilustra o ambiente usado na execução destes passos.

1. Usuário realiza download de um arquivo MPEG-4 (.mp4), este arquivo contém o endereço do MSC responsável pela seção inicial do jogo.
2. Player MPEG-4, ao processar o arquivo, inicia o download da Aplicação MPEG-J e os componentes do *framework* (caso o terminal cliente não possuir estes componentes armazenados), ambos são recebidos em forma de MPEG-let. Originalmente estas MPEG-lets estão no terminal onde se encontra o MSC e o AC da seção inicial do jogo.
3. A aplicação MPEG-J inicia os componentes do Framework: ED, e MR.
4. ED examina as capacidades do dispositivo e envia o resultado para a aplicação MPEG-J.

5. Recebendo os dados do ED a aplicação MPEG-J decide se existe a necessidade de adaptação inicial das mídias que irão ser recebidas do AC e qual o método de adaptação mais adequado, caso positivo, uma requisição é enviada ao AC.
6. A Aplicação MPEG-J então inicia o *download* do jogo, esta requisição é feita para o Adaptador de Conteúdo que entrega para o cliente uma versão adaptada, caso haja necessidade.
7. O ED realiza sua segunda bateria de testes, desta vez com os dados da própria aplicação recebida, que são submetidos a testes de adaptação para verificação do ganho de desempenho que cada mecanismo infere.
8. O usuário entra com suas preferências.
9. Com base nas preferências do usuário e na segunda bateria de testes do ED. Os métodos de adaptação são então mapeados tanto no cliente como no adaptador de conteúdo.
10. O Jogo é iniciado.
11. Detectando se o nível de algum recurso atingir um patamar de alerta, o MR envia estes dados para a aplicação MPEG-J.
12. Ao receber as informações dos recursos comprometidos, a aplicação MPEG-J ativa umas das políticas de adaptação definidas no item 9.
13. O usuário finaliza a aplicação ou entra em outra seção MPEG-4, neste caso o ciclo se repete a partir do passo 6, porém os passos 7 e 8 não são executados.

Quando um determinado usuário muda de sessão, ele automaticamente muda de MSC, se o mesmo estiver em outro terminal acarreta na mudança do AC, sendo necessário o envio da configuração anteriormente obtida.

Quando o usuário envia uma mensagem de atualização da cena não há adaptação a ser feita, a mensagem é multiplexada pelo MBK, enviada para cada usuário presente na mesma zona, juntamente com os demais fluxos que estão sendo enviados.

Quando o usuário envia uma mensagem de voz ou vídeo a mídia é capturada pelo MBK o qual repassa para o adaptador de conteúdo, juntamente com o identificador dos clientes que devem receber aquela mídia, as adaptações são feitas paralelamente, e assim que finalizadas, são entregues aos clientes de destino.

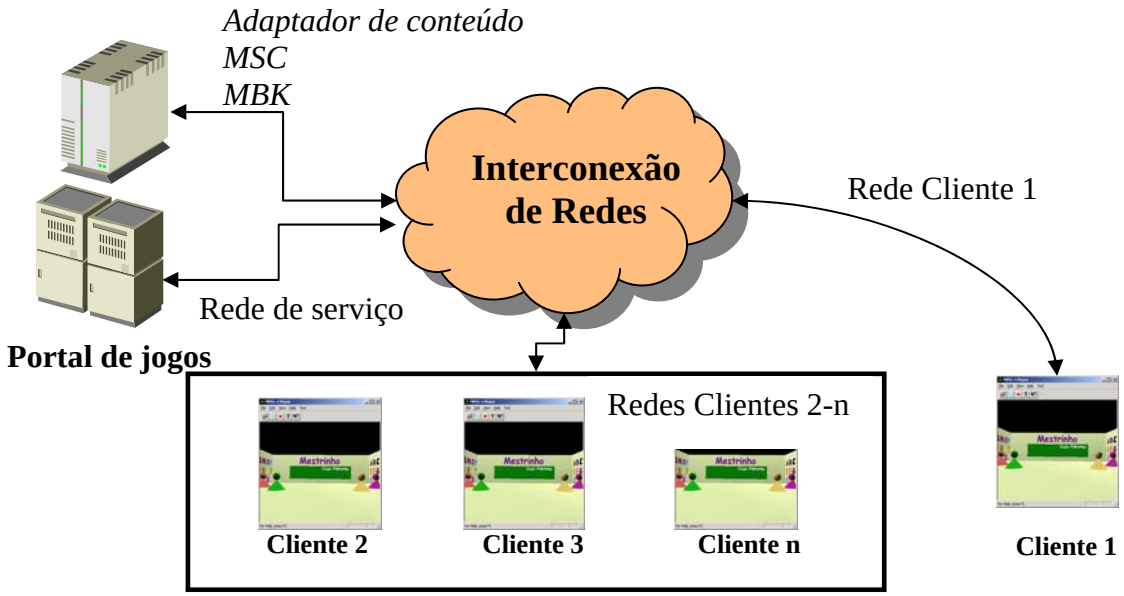

Figura 15 Comunicação de clientes de várias redes com a rede de serviço

5.2 Interface com o usuário

Um usuário seleciona, através de uma interface de alto nível, opções que determinam as ações a serem executadas quando há a necessidade de adaptação. Estas opções incluem, mas não se limitam, a indicadores quantitativos de qualidade e desempenho segundo a Figura 16.

É oferecida ao usuário a opção de determinar a quantidade de QPS mínima que o jogo possa atingir, caso a taxa fique abaixo deste valor os mecanismos de adaptação serão ativados para que a taxa volte ao nível definido pelo usuário.

Alternativamente é fornecida a escolha quanto à conservação da CPU ou a largura de banda, quando não há necessidade de adaptação, durante o download do conteúdo MPEG-4.

Por fim, uma política de prioridades determinará as formas de adaptação que irão garantir a taxa de quadros por segundo especificada pelo usuário. As opções fornecidas são “Qualidade de Imagem”, “Efeitos”, ”Som” e “IA”. A escolha dos mecanismos de adaptação dependerá da prioridade escolhida, as primeiras características a serem diminuídas serão as que possuem menor prioridade. Na configuração ilustrada da Figura 16, por exemplo, a inteligência artificial será a primeira a ser diminuída, seguido do som, efeitos e por último a qualidade da imagem, isto significa que esta configuração garantirá o melhor desempenho com a melhor qualidade de imagem possível.

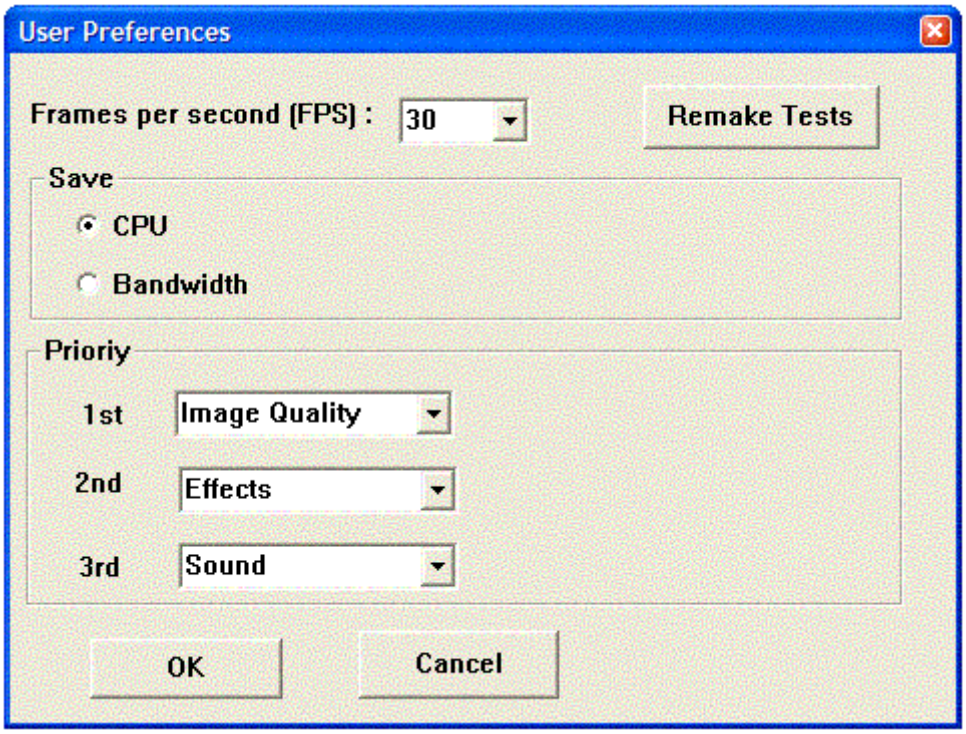

Figura 16 Software de entrada do usuário

Não existe a possibilidade de o usuário selecionar uma quantidade de quadros por segundo que ele não consiga renderizar, pois os dados são limitados segundo os resultados da segunda bateria realizada pelo ED. Caso o usuário deseje refazer os testes, por exemplo, caso ele notou que uma aplicação externa comprometeu o desempenho dos testes, ele pode fazê-lo clicando no bobão “Remake Tests”.

A interface sempre estará disponível ao usuário no menu da aplicação MPEG-J caso seja necessário mudar suas preferências em outro momento. As mudanças das preferências acarretam na mudança das políticas de adaptação.

5.3 Especificador de Desempenho

O Especificador de Desempenho é iniciado no dispositivo do usuário antes do jogo ser recebido e carregado. Sua primeira função é realizar uma estimação inicial da capacidade máxima dos recursos do dispositivo para estabelecer os limites nos quais a aplicação será executada, fornecendo dados para a aplicação MPEG-J determinar quais modificações devem ser feitas para a adequação do jogo ao dispositivo do usuário, ou mesmo se o dispositivo tem os requisitos mínimos para a aplicação.

Verificar a disponibilidade mínima de recursos no cliente impede o *download* desnecessário dos dados da aplicação para o terminal do usuário, caso ele não tenha a capacidade de executar o jogo com um mínimo de desempenho necessário estipulado pelo programador.

Como função estendida o ED testa todos os métodos de adaptação disponíveis no cliente, ou seja, testar as configurações dos decodificadores para determinar a influência exata de um mecanismo de adaptação no desempenho da plataforma do usuário. Para que o ED consiga testar a aplicação é necessário que o programador forneça um arquivo contendo detalhes de como serão conduzidos os testes. Sendo assim dois testes são realizados, o primeiro para determinar as capacidades estáticas do dispositivo, e o segundo para determinar como o dispositivo reage aos usos de mecanismos de adaptação para aquela determinada aplicação.

Os resultados, tanto do primeiro teste (teste inicial) quanto do segundo (teste estendido), são armazenados em um arquivo para posterior consulta, o que impede a execução desnecessária de ED caso não haja modificação no hardware do cliente. O primeiro teste é o mesmo para qualquer aplicação e seu resultado não terá variação com o tempo, sendo assim, somente existe um resultado. Como o segundo teste é especifico para cada jogo, e seus resultados variaram de aplicação para aplicação, é necessário a gravação em arquivo de todos os testes aplicados a diferentes jogos. Caso um determinado jogo tenha sido testado não é necessário realizar testes novamente, a não ser que o programador determine a necessidade de um novo teste estendido.

Assim que os resultados do teste estendido, ou teste inicial (caso não haja suporte do programador para a execução do segundo), forem enviados à aplicação MPEG-J o programador inicia a interface com o usuário. Em um segundo passo os limites são mapeados, impedindo o usuário de selecionar alguma configuração que seu dispositivo não suporte, no entanto esta capacidade somente é suportada se o teste estendido for realizado.

Os dados coletados pelo especificador de desempenho em seu primeiro teste incluem:

- Capacidade de processamento: calcula a relação entre a freqüência do processador e a capacidade de execução de instruções por segundo, tanto em ponto flutuante quanto em inteiros. Esta abordagem é mais realista do que a simples identificação do *clock* máximo do processador.
- Teste de Memória: Largura de Banda máxima (em GB por segundo) de transferência entre o processador e suas memórias, caso o dispositivo possuir mais que um tipo,

como é o caso dos celulares que possuem memória Flash e Memória RAM. A quantidade total também é verificada.

- Capacidade de processamento gráfico: número de polígonos renderizados por segundo, quantidade de pontos de texturas processados por segundo, quantidade de memória de vídeo (se disponível), resolução máxima suportada, quantidade de cores do *display*, freqüência de atualização do *display*, identificação das capacidades de aceleração gráfica via hardware.
- Largura máxima de banda: quantidade de tráfego suportado pelo cliente, através da medição das velocidades de *download* e *upload.*
- Capacidade de processamento sonoro: quantidade de canais de saída de áudio, número máximo de sons reproduzidos ao mesmo tempo, taxa de amostragem do sinal sonoro, etc.

O resultado do teste inicial é comparado com o requisito mínimo estipulado pelo programador, se não forem atendidos, a aplicação é abortada, caso contrário, serão comparados com os valores necessários para execução das mídias em suas configurações máximas.

Se houver alguma divergência quanto à execução das mídias em suas configurações máximas, versões adaptadas serão selecionadas para download. Esta abordagem visa adequar as mídias especificamente para cada dispositivo, evitando o armazenamento de conteúdo que nunca será usado, desperdiçando armazenamento local e forçando mecanismos de adaptação a serem usados continuamente.

O primeiro teste adquire a especificação de alguns recursos estáticos, como resolução máxima do *display*, e outros dinâmicos, como a quantidade máxima de polígonos que são renderizados por segundo. Apesar de serem recursos que indicarão quão bem uma aplicação 3D pode ser executada naquele terminal, seus valores são genéricos para qualquer aplicação, porém devido à variação da complexidade de uma cena 3D, isto impede que valores exatos de desempenho sejam deduzidos para uma determinada aplicação.

Caso o programador conheça bastante sua aplicação, no que se refere à consumação dos recursos, o mesmo pode estipular uma razoável política de adaptação que funcione para diversas plataformas. Porém mesmo para o programador da aplicação é difícil prevenir como será o desempenho da mesma em plataformas que não são consideradas inicialmente ou que não reagem bem aos mecanismos de adaptação propostos.

Para solucionar este problema o ED foi estendido como uma ferramenta de *benchmark* específico para cada aplicação que verificará o impacto de cada método de

adaptação disponível na execução da aplicação. Em linhas gerais o teste estendido simulará a ação de um usuário jogando em uma cena da aplicação, esta simulação será repetida uma vez para cada método de adaptação disponível e de cada execução será extraído o ganho de QPS conseguido pela ativação daquele respectivo método de adaptação. Como o teste estendido é uma peça chave para a decisão de políticas, sua composição será detalhada na próxima seção.

5.3.1 Teste Estendido

Sem respaldo em referências bibliográficas que indicam qual o método mais adequado para adaptar-se a uma determinada mudança de recurso quando utilizados jogos 3D, os mecanismos de adaptação podem não apresentar os resultados mais adequados, e muita carga é colocada nos ombros do programador.

Segundo Bowers [Bowers00], a escolha de um método errôneo para adaptação pode gerar piores resultados do que o problema no qual ela foi aplicada para solucionar. Situações não previstas podem resultar na ativação de adaptações inadequadas proporcionando nenhum beneficio para a aplicação.

Tendo em vista que diversas configurações são oferecidas atualmente em aplicações 3D, citadas na seção 2.7, os possíveis métodos de adaptação ficam mais claros, porém o impacto no qual estes métodos resultam no desempenho do sistema são obscuros e variam dependendo da plataforma e do sistema de renderização desenvolvido.

A solução proposta usa a tecnologia MPEG-4 para promover adaptação, por ser um padrão ISO- IEC, maior diversidade de sistemas de renderização são encontrados. Cada fabricante de um Player MPEG-4 pode utilizar de métodos de renderização completamente distinto dos demais, resultando em diferenças de desempenho em uma mesma aplicação, em um mesmo terminal, usando Players diferentes. Por este motivo os métodos escolhidos para adaptação devem impactar não importando o Player. É de alta importância que os desenvolvedores de Players MPEG-4 utilizem, ao máximo, os recursos disponíveis no dispositivo para melhor execução da aplicação.

Devido aos vários problemas citados, foi desenvolvido um modo para que as políticas de adaptação sejam testadas, antes do uso da aplicação pelo cliente, conseguindo valores precisos do resultado da adaptação. A responsabilidade do teste fica a cargo do ED, que já executa um teste inicial para recolher valores estáticos, por isso o nome de teste estendido. Não faria sentido executar o teste estendido, que necessita dos dados da aplicação

no cliente e possui maior duração, caso o terminal do cliente não se submeter pelo teste inicial que definirá os requisitos mínimos da aplicação.

A opção da execução do teste estendido é facultativa, porém altamente recomendada, as políticas de adaptação serão automaticamente geradas e escolhidas na medida correta, alcançando maior desempenho, e proporcionando menor trabalho para o programador. Caso o programador não opte pelo teste estendido ele terá que manualmente especificar quais serão os limites dos recursos a serem monitorados e quais métodos serão aplicados para a adaptação. O resultado desta abordagem pode ser imprevisível, o programador pode escolher um método erroneamente provocando um grande impacto na qualidade da imagem, ou mesmo na satisfação do usuário, sem aumentar o desempenho da aplicação. Porém a inserção de mecanismos extras aos gerados automaticamente contribui para uma melhor configuração da aplicação.

O primeiro passo do teste estendido é carregar uma configuração de uma cena, escolhida e projetada pelo programador, para realizar o teste. Esta configuração representa um estado em particular da arvore da cena MPEG-4 que pode ser atingida enquanto o usuário está jogando, esta cena de teste é fornecida com a aplicação.

Após carregar o estado da cena no Player MPEG-4, o ED simula uma atividade do usuário que está definida em outro arquivo criado pelo programador. Este arquivo contém uma linha temporal que especificará, para cada instante de tempo discriminado, uma ação a ser realizada.

Durante cada execução dos testes um método de adaptação será testado. O ED automaticamente entra em contato com o adaptador associado ao MSC da respectiva aplicação e consulta os métodos de adaptação, ou seja, as configurações de decodificadores e codificadores, para um determinado tipo de dado. Após a execução do teste estendido o ED, com base nas preferências do usuário, define quais as adaptações mais apropriadas para o aumento da taxa de QPS, em função da alteração de recursos, ou da variação da complexidade da aplicação.

Para a adaptação de controle o ED deve consultar parâmetros expostos pela aplicação MPEG-J definidas manualmente pelo programador. Este processo é dificilmente automatizado, pois o controle é realizado por meio de código inerente a aplicação. Estes parâmetros são testados usando os mesmos procedimentos dos testes com a adaptação dos dados da aplicação.

5.3.2 Considerações

A complexidade dos dois arquivos fornecidos pelo programador afetará nas políticas de adaptação, cabe a ele escolher uma cena com maior probabilidade de ocorrer e simular as ações do usuário apropriadamente. Caso seja criada uma cena que não ocorra com freqüência, a escolha automática decidi por métodos de adaptação que não serão totalmente coerentes na maior parte da execução da aplicação. A escolha de uma cena extremamente detalhada, a qual aconteça raramente durante a execução do jogo, resulta em um ganho considerável de QPS caso seja executado uma adaptação que diminua a qualidade da imagem. Porém, se durante a execução da aplicação for mais comum o uso de efeitos de partículas a diminuição da qualidade de imagem acarretará em uma queda não tão significante quanto à diminuição da apresentação dos efeitos.

Seguindo a analogia acima, caso seja criado uma simulação do usuário com um comportamento incomum o resultado não será condizente com a maioria dos jogadores. Um usuário que se movimenta lentamente experimentará uma taxa de QPS maior que um jogador ativo que está constantemente correndo e modificando seu campo de visão, pois não interage com freqüência com a aplicação.

Os resultados analisados levam em consideração o impacto de cada tipo de alteração na configuração dos decodificadores e codificadores os quais afetam na quantidade de quadros por segundo renderizados, carga no processador e memória ocupada.

É muito importante que nenhuma aplicação esteja sendo executada durante a realização dos testes, caso haja concorrência entre outros processos os resultados não serão coerentes com o desempenho do terminal, por isso é altamente recomendado o encerramento de qualquer aplicação que faça uso do processador ou da rede. A duração dos testes depende da capacidade do dispositivo e da quantidade de testes requisitados.

5.4 Monitor de Recursos

O monitor de recursos inspeciona a variação dos recursos do dispositivo, da rede e da aplicação, lendo parâmetros adquiridos em tempo real e comparando com valores predefinidos. Caso estes valores ultrapassarem estes limites previamente estabelecidos, uma solicitação de adaptação é acionada, via aplicação MPEG-J.

A aplicação MPEG-J registra no Monitor de Recursos um canal usado para a recepção dos eventos, que são acionados logo que haja uma variação do recurso para um valor entre limites definidos, seja no inferior ou no superior, por exemplo, defini-se o recurso

monitorado como sendo a CPU e limites 25, 50, 75, 100 (valores tomados como porcentagem do uso de CPU da aplicação), o MR então notifica a aplicação MPEG-J caso o nível da CPU, detida pela aplicação, alcance valores como 60 ou 40. Caso o nível da CPU se modifique a aplicação MPEG-J é novamente notificada, desde que este valor ultrapasse um limite.

Os dados coletados pelo monitor de recursos são: quantidade de memória disponível no dispositivo, quantidade de processamento detido pela aplicação, latência da rede, largura de banda disponível e número de quadros por segundo sendo renderizados no momento, este último é permitido o monitoramento por apenas uma aplicação.

5.4.1 Considerações

Considerações importantes devem ser tomadas na utilização do Monitor de Recursos. O processo deve ser iniciado com prioridade baixa para que não ocupe desnecessariamente o processamento da CPU. Isto significa que, como toda aplicação em execução, ele deterá parte do processamento da CPU, diminuindo assim o desempenho do sistema. Os testes mostraram que esta queda de desempenho fica em torno de 1% a 5%. Os Resultados apresentados no capitulo 6 levaram em consideração esta perda de desempenho para avaliar a adaptação como uma técnica adequada para o fornecimento de um serviço de qualidade.

No momento da notificação, todos os níveis de recursos são passados para a aplicação MPEG-J, não somente aquele monitorado. A escolha desta abordagem visa proporcionar maior versatilidade na escolha do método de adaptação pelo programador que terá uma visão geral dos recursos do dispositivo naquele momento.

O Monitor de Recursos é implementado de modo a fornecer a leitura de recursos para várias aplicações. Caso mais de uma aplicação se registre para um determinado recurso, diante de uma modificação em um valor monitorado, o MR compara com todos os limites das demais aplicações registradas para aquele recurso. O MR monitora cada recurso separadamente para cada aplicação registrada, pois o valor pode ser diferente. Como exemplo, uma aplicação A detém 40% da CPU sendo que uma aplicação B detém 30 %, o MR deve monitorar separadamente para cada recurso valores diferentes estipulados por aplicações distintas.

5.5 Adaptador De Conteúdo

Na infra-estrutura do servidor, a adaptação é realizada pelo AC. O conteúdo inicial carregado no cliente é adaptado em função de suas capacidades, as quais são detectadas pelo ED. Todo dado adaptável, destinado a algum cliente, passa pelo AC que realiza adequações no mesmo antes de ser enviado para a aplicação cliente. Estes dados são texturas, mensagens de voz, sons de ambiente e vídeos.

O AC funciona como um servidor de conteúdo MPEG-4, armazenando todos os segmentos elementares necessários para a composição de uma cena definida pela seção controlada pelo MSC.

O Adaptador de Conteúdo possui para cada cliente o nível máximo suportado por uma determinada mídia, e o valor da largura de banda do cliente (último monitoramento enviado pela aplicação MPEG-J) para estipular a adaptação necessaria para adequar as mídias que serão enviadas.

5.5.1 Considerações

Como a aplicação MPEG-J não tem a consciência da quantidade de mensagens que um usuário receberá durante o jogo, colocar a adaptação da largura de banda sob responsabilidade dela tornará o resultado da adaptação imprevisivel, o resultado pode acarretar atrasos das mensagens transmitidas em tempo real.

A política de adaptação da largura de banda se concentra em dar prioridade para mensagens de atualização em relação às mídias em tempo real enviadas para o cliente. Todos os dados enviados para o usuário serão multiplexados para que a taxa de bits/s da stream seja igual ou inferior a largura de banda atual do cliente. As mídias em tempo real serão adaptadas, a princípio, para se adequarem a largura de banda restante, já computadas a taxa necessária para recebimento das mensagens de atualização. As mídias nos padrões máximos suportados pelo usuário serão enviadas caso a largura de banda seja suficiente. Vale ressaltar que em casos da adaptação em função da largura de banda o cliente recebe a mídia em um formato inferior ao máximo suportado pelo terminal.

A adaptação em função de recursos locais depende do dado que é recebido pela rede. Como a carga sobre a CPU varia muito com o tempo, não é adequado colocar a adaptação em função da CPU do cliente a cargo do Adaptador de Conteúdo pois várias requisições seriam enviadas para adaptação e o trafego na rede sofreria um aumento considerável devido ao tráfego das mídias adaptadas.

Ao receber a mídia da rede o terminal aplicará a adaptação quando necessário, haverá o caso em que largura de banda esteja tão escassa que a adaptação não ocorrerá pois o dado foi transmitido abaixo do padrão em que a adaptação seria feita.

Todo MSC possui um Adaptador de Conteúdo sendo executado em conjunto, no mesmo host, porém um AC fornece o servico para varios MSCs, desde que estas instâncias estejam no mesmo terminal. Esta escolha foi feita por medidas de performance, o tempo gasto para o envio de uma stream pelo MSC para um AC que esteja em outro terminal pode acarretar em um atraso ao usuário.

Existe a preocupação de sobrecarga do AC, pois há a possibilidade de numerosos pedidos de adaptação para várias mídias de vários usuários. Para este fim o AC muda sua política de adaptação escolhendo o armazenamento das mídias pré-definidas em todos os formatos possíveis, deixando para a adaptação em tempo real, somente as mídias enviadas entre usuários.

Um AC realiza a adaptação de uma mídia enviada em tempo real apenas uma vez para cada tipo de adaptação e repassa para todos os clientes que necessitam de uma mídia adaptada com este respectivo tipo. Como exemplo, vamos considerar uma mídia sendo enviada com o nível máximo de qualidade(tipo de adaptação 4). Dois clientes necessitem desta mídia no tipo de adaptação 2 e um cliente necessita da mídia no tipo 3, os demais não necessitam de adaptação para aquela mídia. O AC então adapta a mídia para o tipo 3 paralelamente a adaptação para o tipo 2 e entregando aos usuários assim que o processo finaliza.

5.6 Aplicação MPEG-J

A aplicação MPEG-J é usada para diversos fins em uma aplicação MPEG-4, como descrito na seção 4.3**,** a adaptação é uma delas. O código deve ser estruturado o bastante para que seja fácil a discriminação da adaptação com o código de tratamento de interação do usuário, comportamento de personagens, inteligência artificial, mensagens multiusuário etc.

A aplicação MPEG-J tem como responsabilidade a comunicação com o AC, MR, ED, assim como promover uma interconexão entre eles. É na aplicação MPEG-J que é definido o método que será executado na recepção dos valores da variação de recursos enviado pelo MR, este método tem papel fundamental, pois define as políticas de adaptação em tempo real.

Outra função da aplicação MPEG-J é iniciar o ED para coletar informações sobre o dispositivo e posteriormente verificar se o cliente possui os requisitos mínimos para a aplicação. Esta verificação deve ser realizada pelo programador que necessita ter os conhecimentos para a inserção apropriada destes valores.

A ativação do segundo teste do ED é opcional, caso o programador tenha suprimido, junto à aplicação, os arquivos necessários para o teste estendido.

Durante a programação devem ser estipuladas as funções de modificação de controle da aplicação, caso seja de interesse do programador fornecer adaptações de controle, mapeando funções genéricas de adaptação para funções específicas dentro da aplicação. Alguns controles não precisam ser mapeados, como a quantidade de sons reproduzidos ao mesmo tempo, pois existem meios de definir diretamente esta opção por meio de remoção de nós da cena.

5.7 Implementação dos Componentes do Framework

Esta seção apresenta as primitivas usadas na implementação, o modelo de código o qual deve ser seguido durante o uso do framework, no qual inclui um guia de uso das primitivas chamadas pela interface do *framework*, bem como regras a serem obedecidas para a ativação apropriada de mecanismos de adaptação.

A implementação do *framework* se encontra parcialmente concluída, sua finalização depende da conclusão de um jogo 3D, a ser discutido na seção 6.1, usado nos testes dos mecanismos de adaptação. Por este motivo os métodos discutidos podem sofrer pequenas alterações no futuro para se adequarem a possíveis modificações.

Quanto à natureza da implementação, o AC é escrito em Java, o ED e MR foram desenvolvidos em C++. Esta abordagem foi escolhida pela falta de suporte na Máquina Virtual Java (JVM™-Java Virtual Machine) para monitoramento de recursos em tempo real. Como a implementação da JVM™ varia de uma plataforma a outra, isto forçaria o sistema de monitoramento a não ser uniforme através das implementações, por isso a especificação das JVMs™ abrem mão das características do sistema a favor da portabilidade da linguagem [Baschieri02]. Isto significa que o ED e MR devem ser desenvolvidos individualmente para cada plataforma, no *framework* implementado, estes componentes foram desenvolvidos para a plataforma Windows 2000 ™ e suas interfaces são disponíveis por meio de *Dynamic Link Libraries*(dlls) acessadas via Java Native Interface [JNI03]. A chamada aos serviços oferecidos pelos três componentes é feita pela aplicação MPEG-J.

Particularidades da implementação de cada componente serão apresentadas e discutidas a seguir.

5.7.1 ED

Para iniciar o Especificador de Desempenho a primitiva abaixo é invocada, a qual possui como retorno uma estrutura chamada ED_Results contendo os resultados dos testes.

ED_Start(Value1,Value2,….,ValueN-1, ValueN);

Os valores passados como parâmetros são constantes definidas na aplicação indicando quais dos testes deverão ser realizados. A primitiva cria um construtor da Classe ED no código dependente de plataforma. A quantidade dos parâmetros é variável, ou seja, o programador pode inserir tanto um como vários parâmetros. Os tipos possíveis de testes correspondem aos valores da Tabela 6:

Tabela 6 Identificação dos parâmetros de ativação do ED.

ID	TIPO
1	Teste de CPU
2	Teste de VIDEO
3	Teste de MEMÓRIA
4	Teste de Rede
5	Teste de Áudio

O código dependente de plataforma executa cada teste usando APIs do Sistema Operacional presentes no terminal

À medida que os testes são realizados uma estrutura é preenchida, e retornada ao programador com o resultado dos testes. Esta estrutura é apresentada a seguir.

```
struct _EDResults
{
        struct _CPU;
        struct _video;
        struct _audio;
        struct _net;
        struct _Memory;

}EPResults;
```

```
struct _Net
{
        int BandWidth;
        int ErrorRate;
        int Ping;
}Net;
```

```
struct _Memory
{
        int BandWidth;
        int Total;
}Memory
```

```
struct _Audio
{
        int Channels;
        int SampleRate;
        int Voices;
}Audio;
```

```
struct _Video
{
        int VideoMem;
        int PolyCount;
        int PolyCountLight;
        int Fill Rate;
        bool DX8;
        bool DX9;
        int OpenGLVer;
        int DysplayType;
        int NumberOfColors;
        int MaxResolution;
}Video;
```

```
struct _CPU
{
        int FPoint;
        int Integer;
        int PentiumRate;
        int Multimidia;
        bool MMX;
        bool 3DNow;
        bool 3DNow2
        bool SSE;
        bool SSE2;
}CPU;
```

Figura 17 Estruturas que formam o ED_Results

Estrutura Net:

- *BandWidth:* Largura de banda máxima do dispositivo com o provedor do serviço de adaptação. O valor está expresso em Kb/s (Kilo bits por segundo)
- *ErrorRate:* Taxa média de erro da Rede. Valor expresso como porcentagem de perda de pacotes.
- *Ping:* Latência entre o usuário e o provedor de serviço. Valor em milissegundos.

Estrutura CPU:

- *FPoint:* Número de instruções em ponto flutuante processadas por segundo.
- *Integer:* Número de instruções inteiras processadas por segundo.
- *PentiumRate: Benchmark* que retorna a equivalência com um processador Pentium original. Valor expresso em MHz.
- *Multimídia:* Número que avalia as capacidades Multimídia do dispositivo conseguidos via benchmark
- *MMX:* Presença de instruções MMX. Valor Booleano.
- *SSE:* Presença de instruções SSE. Valor Booleano.
- *SSE2:* Presença de instruções SSE2. Valor Booleano.
- *3DNow:* Presença de instruções 3DNow. Valor Booleano.
- *3DNow2:* Presença de instruções 3DNow Professional. Valor Booleano.

Estrutura Áudio:

- *Channels:* Número de canais presentes no dispositivo.
    - Valor = 1: Mono
    - Valor = 2: Stereo
    - Valor = 3: Surround
    - Valor = 4: 4.1
    - alor = 5: 5.1
- *SampleRate:* Amostragem máxima do sinal em kHz.
- *Voices:* Quantidade máxima de sons que podem ser reproduzidas ao mesmo tempo.

Estrutura Vídeo

- VideoMem: Quantidade máxima de memória de vídeo. Valor em MegaBytes
- *PolyCount:* Milhões de polígonos renderizados por segundo.
- *PolyCountLight:* Milhões de polígonos, com uma fonte de luz, renderizados por segundo.
- *FillRate:* Milhões de Textels (pontos de polígonos com Texturas) renderizados por segundo .
- *DX8:* Presença de aceleração em hardware para DirectX8. Valor booleano
- *DX9:* Presença de aceleração em hardware para Directx9 Valor booleano.
- *OpenGLVer:* Versão do OpenGL suportado via hardware.
- *DysplayType:* Tipo do Display
    - Valor = 1: CRT
    - Valor = 2: LCD
    - alor = 3: Plasma
- *NunberOfColors:* Número máximo de cores do display. Valor em bits
- *MaxResolution:* máxima resolução suportada pelo display
    - Valor = 1: 320x240
    - Valor = 2: 640x480
    - Valor = 3: 800x600
    - Valor = 4:1024x768
    - Valor = 5:1280x1024
    - Valor = 6:1600x1200

Estrutura Memory

- *BandWidth:* Largura de banda com a memória RAM. Valor em MB/Segundo
- *Total*: Total de Memória RAM Disponível. Valor em MegaBytes

5.7.1.1 Execução do Teste estendido

Para a execução do teste estendido faz se uso da primitiva *ED_EX("Scenesample.scene", "SceneSimulation.sim")* na aplicação MPEG-J. Onde *Scenesample.scene* é a cena usada para testes e *SceneSimulation.sim* é a descrição da simulação a ser representada, seu retorno é um objeto instanciado da classe DynAdapt.

Automaticamente o ED utiliza a função GetRequiredCodecs () que retorna uma lista de codificadores disponíveis em um vetor de referências para objetos do tipo Codec. Para cada entrada ele realiza a função GetAdaptParameters(PtCodec[X]) que pesquisa no AC os parâmetros para cada adaptação suportada por aquele Codec. Os parâmetros são armazenados numa lista de execução. Após ser completada a lista de execução a chamada para a realização dos testes é feita por Execute_EX (ExecutionList);

A lista de execução possui, para cada entrada, o nome do Codec e uma string de parâmetros que será atribuída ao correspondente decodificador presente no MPEG-4 Player.

Execute_EX inicia o processo de testes de todas as configurações e retorna, para cada linha na lista de execução, o ganho em QPS, memória e memória de vídeo, enviando um objeto DynAdapt a aplicação MPEG-J

5.7.1.2 Execução do Teste estendido para adaptações de controle

A maneira de capturar as possibilidades de adaptação de controle é diferente dos métodos utilizados para adaptação de conteúdo, pois não são utilizados decodificadores. A classe estática ControlAdaptations é preenchida durante a codificação da aplicação MPEG-J, durante a captura dos parâmetros dos codificadores disponíveis é também realizado a chamada ControlAdaptations.GetParameters(), inserindo na lista de execução as possíveis adaptações de controle.

5.7.1.3 Chamada da interface com o usuário

Para que a interface do usuário seja exibida, a função, *UserInterface (DynAdapt)* é executada. O objeto *DynAdapt* pode conter o valor *null*, caso não for realizado o teste estendido pelo ED. A interface é exibida com o campo “Frames per second” limitado a máxima taxa de quadros medida durante o teste estendido.

5.7.1.4 Estrutura do arquivo de simulação do usuário

O comportamento na cena que deve ser simulado durante o teste estendido realizado pelo ED está descrito em um arquivo junto com a aplicação, neste arquivo é simulado a interação com a aplicação por meio de códigos ascii significando uma determinada tecla. Esta abordagem apesar de extensiva permite independência de aplicação, pois todas deverão possuir mecanismos de tratamento de entrada por teclado.

O arquivo pode ser gerado automaticamente por meio de ferramentas de keyboard *logging* como *spylock* [SpyLock03].

O arquivo deve conter, para cada linha, um tempo em milissegundos, e o código ASCII da tecla pressionada naquele instante. O formato do arquivo é do tipo texto.

O arquivo da cena de simulação não tem nenhuma diferença de um arquivo normal de representação de cena.

5.7.1.5 Estrutura do arquivo de saída do ED

O arquivo onde o ED armazena os resultados dos testes é de grande utilidade para identificar os procedimentos realizados com todas as aplicações executadas no cliente de forma a minimizar o tempo de testes em uma aplicação antes de ser iniciada.

O arquivo de saída gravado pelo ED contendo os resultados do teste inicial e de cada teste executado com as aplicações segue o formato abaixo:

```
First Test
;CPU values
;video values
;audio values
;Net values
;Memory values

App1: dd:mm:yy  hh:mm:ss
CodecID1: fps  Mem  VidMen
CodecID2: fps  Mem  VidMen
.
.
.
CodecIDN: fps  Mem  VidMen
```

Após a palavra reservada **First Test** os valores contidos na estrutura ED_Results são gravados, logo após são armazenados para uma determinada aplicação executada em certo dia, mês, e ano, hora minuto e segundo, os valores dos ganhos em quadros por segundo, memória e memória de vídeo para cada configuração de um ou mais *codecs* disponíveis.

5.7.2 Monitor de Recursos

O MR monitora determinados recursos e quando os níveis de disponibilidade destes ultrapassam limites pré-estabelecidos por uma aplicação registrada, esta é notificada, com o envio dos valores de todos os recursos monitorados, não somente o que ela pediu para ser monitorado, isto é feito pois para uma escolha adequada de uma política de adaptação deve-se conhecer o estado atual dos demais recursos.

Para que uma aplicação se registre no Monitor de Recursos ela deve usar a primitiva a seguir.

MR_Reg(Resource,EventHandlerFunction,Bounds);

O campo *resource* segue os valores da Tabela 7 e indica o recurso a ser monitorado, o campo *EventHandlerFunction* indica a função que tratará o evento de resposta do MR quando o recurso monitorado ultrapassar os limites estipulados no campo *Bounds*, o tipo deste campo é definido como um vetor de inteiros contendo os valores limites para cada a notificação. EX: O valor do campo *Resource* com o valor 1 indicando que a CPU está sendo monitorada e atribuindo ao campo *Bounds* um vetor V com os elementos definidos como

V={20,40,60,80}, significa que o MR deve monitorar a CPU notificando a aplicação quando a porcentagem detida pela mesma atingir valores entre 00-20,21-40,41-60,61-80,81-100. A aplicação não é notificada caso o nível da CPU detida passe de 45 para 55, pois não ultrapassou nenhum limite estabelecido.

Tabela 7 Recursos monitorados pelo MR

RECRUSO	Valor
CPU	1
Memória	2
Largura de banda	3
Latência	4
Quadros por segundo	5

Para o registro do monitoramento da taxa de quadros por segundo é usado o valor de User.FPS, parâmetro que o usuário estipulou durante a escolha das preferências.

Para monitoramento da carga sobre a CPU e memória foram utilizadas chamadas a API Win32 que fazem uso do PDH- Performance Data Helper, uma interface que tem como objetivo principal coletar dados de performance. O processo de monitoramento é simples, cria-se uma *query* e adiciona *performance counters*, que coletam os dados de performance, quando estes dados são processados e a query é finalizada.

Para calcular a latência foi utilizada a chamada Net::Ping da API Win32. Para o cálculo da largura de banda foi utilizada a taxa de Kbytes/segundo da ultima transmissão recebida. Como os quadros por segundo são enxergados como recursos da aplicação, o monitoramento não é realizado por APIs do sistema. A API MPEG-J já fornece o suporte para a leitura de quadros por segundo, portanto foi utilizado a API MPEG-J **org.iso.mpeg.mpegj.resource. MPRendererFrameEvents.**

5.7.3 Aplicação MPEG-J

A aplicação MPEG-J é responsável pela ativação do ED e do registro com o MR e AC.

Após a ativação do ED, e posteriormente ao recebimento dos dados coletados, o programador deve analisar os resultados para compará-los e decidir quais adaptações usar ou mesmo decidir que não é possível a execução da aplicação.

Por meio da primitiva *ED_Start* que retornará *ED_Results* um simples conjunto de sentenças de fluxo de código são necessárias para determinar a adaptação.

```
ED_Start(1,2,3,4,5);

Vm = ED_Results.Video.VideoMem;

if (Vm <= 4M)
{
        AbortOperation("1");
}
```

Caso não haja nenhum requisito que impeça a aplicação de executar o próximo passo é a escolha de algum mecanismo de adaptação, e conseqüentemente o mapeamento desta opção no AC.

5.7.3.1 Registro no AC

O Registro no AC é feito logo após a detecção dos requisitos mínimos da aplicação, e deve ser feito manualmente. O Registro é necessário para o estabelecimento de um canal entre o cliente e o AC. Um host ao se registrar deve especificar os mecanismos de adaptação iniciais.

Para se registrar no AC a aplicação envia a seguinte primitiva:

AC_Register(Client_IP(),AdaptType);

Client_IP() é uma função que retorna o IP do cliente e *AdpatType* é uma estrutura que define os tipos de adaptação que o cliente deve sofrer para que as streams sejam adequadas a largura de banda do usuário, ou mesmo para impedir que seja enviada uma mídia em um formato não suportado pelo usuário. AdaptType é definido como descrito na Figura 18.

```
struct _ AdpatType
{
        Codec Compression3D;
        Codec TextureAdap;
        Codec VideoAdp;
        Codec AudioAdap;
        int Bandwidth
} AdpatType;
```

Figura 18 Estrutura AdaptType

O campo *Bandwidth* indica largura de banda disponível no cliente, para uso durante a adaptação em tempo real, os demais indicam a adaptação durante o carregamento do jogo especificado por codificadores que serão substituídos no AC para o envio de conteúdo 3D comprimido, texturas, vídeos, arquivos e mensagens de áudio.

5.7.3.2 Definição da classe tratadora de eventos

No momento do registro no MR a aplicação MPEG-J indica o objeto que instancia a classe de evento a qual será notificada caso haja a ultrapassagem de um dos limites estipulados. A classe definidora deste objeto deve implementar a Interface *AdaptEventHandler*.

A classe deverá realizar o *Override* do método *ResourceChange* o qual será chamado toda a vez que for detectado a variação para fora dos limites estabelecidos. Neste método o programador deve inserir as condições para tratamento da variação do recurso e, quando necessário, invocar a seguinte API MPEG-J:

Resourcemanager.changeDecoder(node,Decoder);

Esta primitiva usa uma classe definida no padrão MPEG-J para modificar um decodificador associado a um nó MPEG-4. Este decodificador deve ser compatível com o codificador usado pelo AC para aquele respectivo nó. Uma exceção é o nó de renderização que é usado somente no cliente.

Caso o programador executou o teste estendido, basta acionar os métodos que realizam a funções automáticas. Como exemplo: DynAdapt.Control, que realizará adaptações de controle de acordo com a preferência do usuário.

A primitiva *MR_REG* pode ser usada novamente, caso seja do interesse do programador, mudar os níveis do recurso a serem monitorados, uma prática bem vinda, mas

não necessária. Esta ação é útil nos casos em que o MR registre constante mudança de um recurso variando entre dois limites adjacentes, por exemplo, o nível de largura de banda flutuando entre 19Kb/s e 21Kb/s, com o MR monitorando o recurso com um limite de 20Kb/s o que sugere que toda vez que esteja abaixo ou acima de 20kb/s uma mensagem deverá ser enviada ao AC, ocasionando um fluxo alto de mensagens, que comprometeria o desempenho da rede e da aplicação.

5.7.3.3 Definição de mecanismos de adaptação de controle

Uma adaptação de controle será utilizada caso a aplicação forneça parâmetros a serem modificados que alterem seu comportamento. Com algumas exceções, a adaptação de controle pode ser executada sem o conhecimento prévio do programador, como no caso da diminuição da quantidade de sons reproduzidos ao mesmo tempo. Esta adaptação é feita retirando-se os sons, com prioridade menor, da cena gráfica.

Para as demais adaptações é necessário usar a classe estática *ControlAdaptation*, por meio de uma definição de uma interface JAVA e implementação do método *SetParameter*, cujo corpo da função deve ser uma chamada à uma método da aplicação que cause alguma mudança no controle da mesma.

5.7.4 AC

O AC possui duas primitivas principais: *AddCodecConfiguration(Codec)*, necessária para posterior consulta pelo ED da configuração disponível de cada codec.
Sendbuffer(AudioBuffer,MSGBuffer) usada para tratar a adaptação da largura de banda para os clientes.

6 EXECUÇÃO E AVALIAÇÃO DO *MMGAME*

Para avaliar o *framework* proposto foi concebido um ambiente de teste que representasse, da melhor forma possível, os resultados das políticas de adaptação para compará-los aos dados de uma execução sem adaptação. Para este fim foi projetado um jogo 3D multiusuário com os dados necessários para a execução apropriada dos testes realizados pelo ED.

Nos ambientes de testes preliminares foi usado o Player desenvolvido pelos próprios integrantes do processo de definição do padrão, que é publicamente disponível para desenvolvedores por meio de FTP no site do padrão MPEG-4, o desempenho em todos os demais players será de igual escala ou superior.

Quanto ao ambiente para a execução da aplicação, foi escolhida uma plataforma PC de nível médio, o qual possui hardware apropriado para conduzir a aplicação em uma qualidade de imagem e desempenho mediano, para fazer justo o uso de adaptação. A plataforma é descrita abaixo:

- *Processador AMD Atlhon 800 MHz*
- *Memória DDR PC 2100(266 MHz) 256 MB*
- *Placa de vídeo RIVA TNT2 8 MB – Versão do Driver 42.01*
- *HD ATA 66 2 MB de buffer.*
- *Monitor com resolução máxima de 1024x768 32bits de cores*
- *Windows XP*.

A escolha desta plataforma para testes se justifica por várias razões. Como componente principal, e de maior impacto em aplicações 3D, a placa de vídeo possui uma GPU de nível mediano para execução de aplicações 3D, possui acelerações básicas, porém uma quantidade pequena de memória de vídeo. A quantidade de memória RAM é suficiente para armazenamento de todo o conteúdo do jogo sem a necessidade de swap com o conteúdo armazenado na memória virtual. O processador por sua vez é adequado para aplicações 3D, com um *clock* suficiente para suportar um bom processamento sonoro e comportamento simulado.

6.1 O Jogo desenvolvido

Em seguida temos o jogo 3D usado. Quanto à qualidade gráfica, optou-se por uma quantidade satisfatória de texturas com resoluções bem detalhadas, o que proporciona uma razoável qualidade visual. Quanto à qualidade sonora, um conjunto de som foi escolhido para proporcionar maior realismo ao jogo, com formatos e tamanhos totalmente coerentes com os demais jogos da atualidade. Uma preocupação no desenvolvimento do jogo foi a semelhança com os jogos atuais, em nível de qualidade visual e sonora. O desenvolvimento de um jogo com recursos gráficos sofisticados, para maior realismo, necessita de uma grande base instalada de hardware com aceleração para tais recursos, o que não se justificaria neste projeto, devido à necessidade do suporte para uma heterogeneidade de dispositivos. Um terminal ou dispositivo que suporte a execução de jogos 3D deve possuir características mínimas, definidas pelo desenvolvedor do jogo, como por exemplo, a presença de um processador para aceleração 3D e uma quantidade adequada de processamento. Sendo assim o jogo foi projetado para ser executado em dispositivos móveis e desktop de recursos menores labelados com o suporte a aplicações 3D.

Decidiu-se, em relação o tipo do jogo, pelo tiro em primeira pessoa (FPS – First Person Shooter). O usuário não visualiza seu próprio avatar, pois a tela do computador simula a visão do usuário. É possível visualizar tudo ambiente ao seu redor com apenas o movimento do mouse, o qual simula a orientação, e com a interação com o teclado, que simula a movimentação para frente, atrás, direita e esquerda.

O jogo é titulado Moving Target onde o objetivo dos usuários é acertar um alvo dirigido por uma inteligência artificial. O ambiente é uma arena, um local fechado de tamanho relativamente pequeno, o que caracteriza a visita de uma área pelo usuário diversas vezes durante uma partida, visto a constante procura pelo alvo para conseguir aumentar sua pontuação.

A implementação desde jogo se encontra parcialmente concluída. As principais características estão funcionando, dentre elas a movimentação no ambiente e a simulação de agentes dirigidos por computador com movimentos pré-definidos e o controle sonoro. A inteligência artificial, interação com objetos, efeitos especiais e alteração do ambiente está em fase de implementação.

Para a modelagem do ambiente usou-se um editor de cenários que acompanha o jogo *Quake III Arena* desenvolvido pela *Id Software*[Quake99], o mapa resultante foi convertido para um formato de intercambio (extensão .OBJ) e aberto no modelador *Maya 4.5*

Personal Edition, uma versão para uso pessoal e grátis de um dos melhores modeladores 3D do mercado. Neste modelador foi inserido tratamento de colisão, e alguns parâmetros VRML, o resultado foi exportado para o padrão VRML 2.0, que por sua vez, foi convertido em MPEG-4 3D, usando um conversor desenvolvido por um aluno de iniciação cientifica do nosso grupo de pesquisa[CORTEZ 2002]. Demais modificações foram feitas direto no arquivo MPEG-4. O resultado final está ilustrado na Figura 19 e Figura 20.

Figura 19 Tela do Jogo

Figura 20 Tela do jogo.

Neste jogo foram utilizados os seguintes formatos de texturas, PNG (Portable Network Graphics), JPG e TGA (TARGA), com várias resoluções, profundidade de cor fixada em 24bits, algumas em formato de quadrados e outras em retângulo. A escolha por tanta diversidade é a manutenção da coerência com os jogos atuais.

Existem casos em que os vários formatos de textura influenciam em uma aplicação 3D, um deles, o tamanho dos arquivos em disco. Uma mesma imagem com resolução 512x512x24b possui no formato JPEG, 97KB, no formato PNG, 302 KB e no TGA, 768 KB, sendo a qualidade do formato JPEG pouco inferior aos demais. Tendo em vista que a aplicação usa, em média, 200 texturas, a quantidade de espaço em disco necessário para o armazenamento pode variar até dezenas de megabytes. A escolha do formato no qual a imagem é armazenada dependerá do programador, uma vez que as texturas são carregadas na memória não há diferença entre os formatos, pois o conteúdo residente em memória é uma versão não comprimida em formato RAW. O formato do arquivo altera não somente o espaço ocupado no armazenamento local, como também o tempo para carregar as texturas em memória. O formato JPEG, devido ao seu alto grau de compressão, é o que apresenta maior tempo para ser carregado, seguindo do formato PNG. Texturas em TGA são carregadas mais rapidamente em memória, uma vez que a representação no armazenamento local é idêntica ao armazenado em memória.

Cabe ao programador escolher um formato que seja adequado a qualquer dispositivo, por exemplo, o formato JPG ou PNG, que permite o nível de compressão ser ajustado.

Como o jogo está em fase de implementação, não foi possível o teste de todas as opções disponíveis pelo *framework*. Com as características já implementadas, conseguem-se alguns parâmetros para avaliar os resultados da adaptação, estes parâmetros incluem: configuração da textura, configuração do som e configuração das mensagens de voz. A próxima seção descreve os testes que foram realizados em função destes parâmetros e a discussão em torno dos resultados obtidos.

6.2 Testes Realizados pelo ED

A aplicação MPEG-4 contendo o jogo foi disponibilizada em um servidor de *streams* MPEG-4 e os passos descritos na seção 5.1 foram seguidos.

O teste inicial do ED foi executado, os parâmetros definidos para monitoramento incluíram o teste de vídeo, áudio, CPU e de rede. Os resultados obtidos foram

comparados com os requisitos mínimos da aplicação definidos na aplicação MPEG-J, são eles:

- Memória de vídeo maior que 4 MB
- Processador com Rate (PentiumRate) maior 300;
- Largura de banda maior que 33kbps;
- Suporte a reprodução de sons com 8Khz de taxa de amostragem;

Os valores acima foram obtidos usando todas as políticas de adaptação em suas formas mais agressivas. Caso não for possível a execução das mídias em sua capacidade mínima, não haverá como executar a aplicação com uma performance aceitável.

Os resultados também são comparados com os requisitos de execução de cada mídia em seu formato máximo, para verificar a necessidade de adaptação inicial das mídias, são eles:

- Resolução das texturas de 640x480 pixels;
- Profundidade de cores das texturas de 24bits;
- Reprodução sonora suportando taxa de amostragem de 22 kHz.

Caso a plataforma não possua as capacidades máximas, as mídias deverão ser adaptadas antes de recebidas no cliente, para poupar tanto largura de banda como espaço de armazenamento local, no entanto nesta arquitetura não foi necessário.

Como a plataforma passou pelos testes de requisitos mínimos e máximos, o ED iniciou o download dos dados da aplicação e acionou o teste estendido, fornecendo os arquivos necessários para a simulação do comportamento no jogo.

Durante o teste estendido o principal parâmetro de comparação será a quantidade de quadros renderizados por segundo. Notas quanto à qualidade visual e sonora também são levadas em consideração.

O procedimento de teste é idêntico para qualquer circunstância, ou seja, caso seja realizado o mesmo teste duas vezes o resultado será idêntico ou com diferenças imperceptíveis e irrisórias.

Para a realização dos testes definiu-se, no arquivo necessário, um caminho pré-estabelecido que a simulação percorrerá em toda execução. Nesta simulação somente a movimentação do usuário (devido ao próprio estado atual do jogo) está implementada, cada movimentação possui velocidade constante e a câmera muda constantemente de foco. Várias áreas do jogo foram inspecionadas, algumas com grandes detalhes e outras com poucos, tudo com o intuito de gerar a maior quantidade de dados possível para minimizar a possibilidade de

erro nas decisões tomadas a partir dos testes. O tempo total do teste é de aproximadamente 4 minutos e 50 segundos e tem como objetivo principal simular o comportamento de um usuário na arena do jogo, porém o tempo pode ser minimizado. A escolha de um teste tão longo justifica-se por conseguir grandes quantidades de dados implicando em decisões mais apropriadas.

Existem dois tipos de decodificadores presentes na máquina cliente necessários para a execução da aplicação, um decodificador para as texturas e outro para o áudio. Durante o teste estendido foram realizadas as detecções na configuração destes decodificadores, obtidas em temo real pela pesquisa das mesmas no AC, e executados os testes. As seções posteriores apresentarão e avaliarão os resultados. No Anexo A encontra-se um teste simulando a queda da carga da CPU e suas conseqüências.

Deseja-se testar os dois tipos de adaptação, de dados e de controle. Para adaptação de dados foi escolhida a adaptação da textura e qualidade sonora, para controle foi escolhida a diminuição da quantidade de sons reproduzidos por segundo.

6.3 Variação na textura

Este teste procurou definir qual o impacto da variação de uma textura na aplicação 3D executada no cliente. Quando carregadas pela aplicação, as texturas são armazenadas na memória RAM e copiadas para a memória de vídeo. A Figura 21 apresenta uma cena não texturizada, a Figura 22 apresenta a mesma cena texturizada.

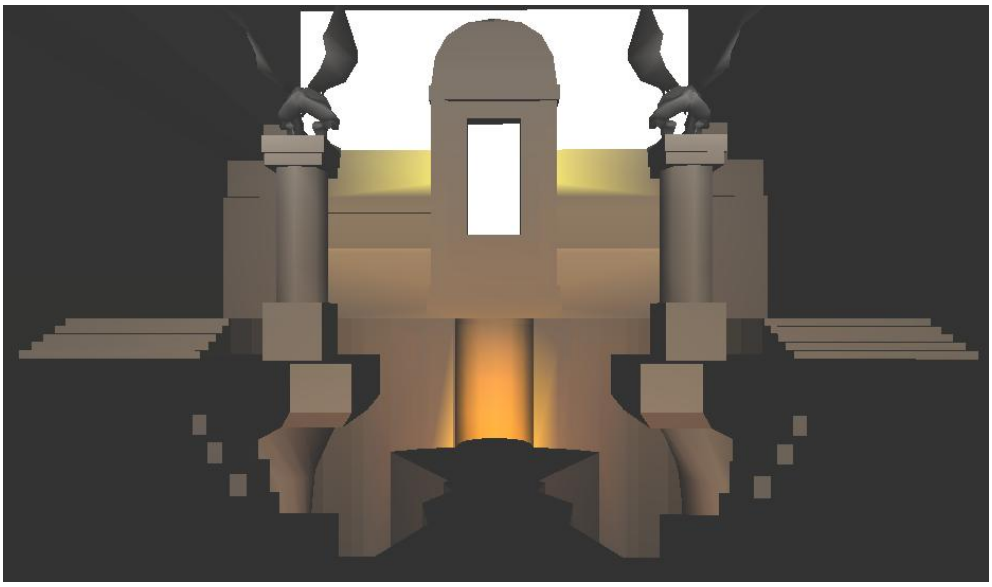

Figura 21 Imagem do jogo sem texturização

Figura 22 Imagem do jogo com texturização

No início do desenvolvimento de jogos, e ambientes virtuais 3D em geral, era muito comum a utilização de poucas, ou até nenhuma, textura em uma cena, atualmente não se cogita esta abordagem por provocar uma degradação exagerada da imagem.

O ED realizou o teste com duas configurações do decodificador, cada uma resultou em suas respectivas séries de dados. A configuração original do decodificador decodifica as texturas em seus formatos padrões, com resoluções estão variando entre 32x32 e 512x512, sendo que a maioria se encontra na resolução de 256x256. A profundidade de cor foi mantida constante, no formato de 24bits. Durante o segundo teste as dimensões das texturas foram limitadas a 64x64 *pixels* pela modificação na configuração do decodificador. O aspecto da imagem foi mantido, por exemplo, imagens com o formato 256x512 foram convertidas para 32 x 64. Nos dois testes o processamento sonoro foi desabilitado. O Gráfico 1**Erro! Fonte de referência não encontrada.** apresenta o tempo em relação à quantidade de quadros renderizados por segundo.

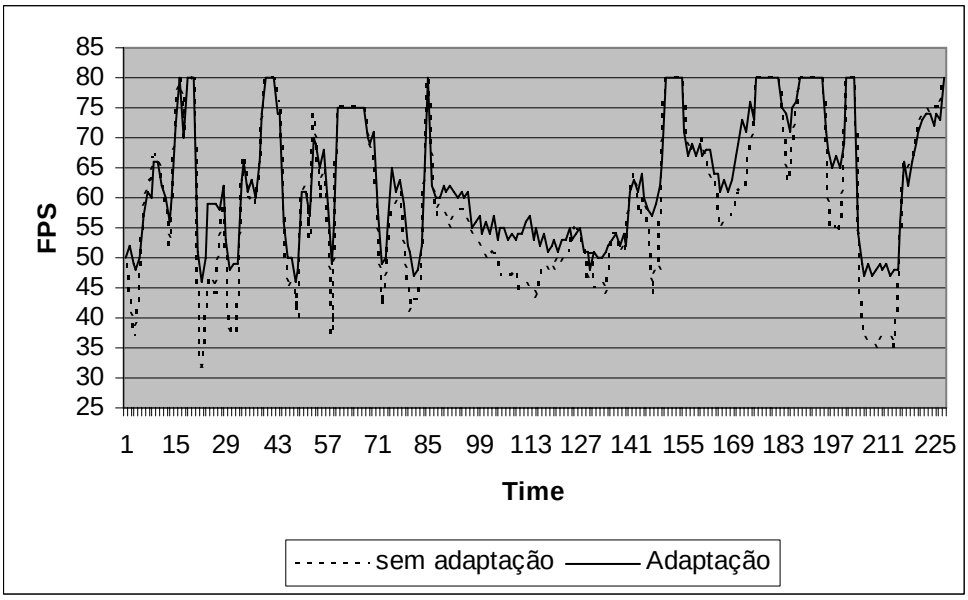

Gráfico 1 Resultado da execução das texturas convertidas

Este gráfico ilustra que a diferença, quanto ao desempenho, ocorre em cenas com alta variedade de texturas, pelo menos quanto a esta plataforma. O impacto sobre o processador não é notado, mesmo que este parâmetro não é direcionado para diminuir a carga do processador. Este parâmetro é afetado pela GPU utilizada, especificamente pela capacidade de processamento de *textels* por segundo, na literatura denominado *Fill Rate*. O recurso mais afetado por esta modificação é a quantidade de memória utilizada.

A queda de performance causada pela falta de memória para armazenamento de texturas acarretou em baixa taxa de quadros renderizados por segundo em cenas com uma variedade maior de texturas, pois neste caso elas foram buscadas na memória principal. Carregar as texturas da memória principal para a memória de vídeo acarreta em queda na taxa de quadros por segundo, principalmente caso estas trocas sejam constantes. A busca deste conteúdo na memória virtual (armazenamento local) causa momentos de congelamento da imagem apresentada devido à baixa velocidade de acesso do *hard drive*.

A Tabela 8 apresenta os valores da quantidade ocupada em memória com relação aos padrões utilizados nas duas séries de testes. O armazenamento necessário se refere à quantidade de espaço usado no armazenamento local de todas as texturas do jogo, a memória de vídeo refere-se às texturas usadas na seção de teste do jogo.

Tabela 8 Consumo de memória pelas Texturas

	Memória de vídeo	Armazenamento Necessário
Nenhuma textura	4.203	0
Texturas Convertidas	7.735	1.92
Texturas Padrões	14.071	16.2

Os dados acima demonstram que o uso de texturas convertidas diminui em quase três vezes a memória necessária para armazenamento. Vale ressaltar que este ganho é também refletido na memória principal, visto que o conteúdo da memória de vídeo é um subconjunto da memória principal. Uma adaptação que empregasse esta transformação poderia ser usada na adaptação do conteúdo antes mesmo do envio ao usuário, caso o dispositivo possua pouca memória de vídeo e/ou memória principal. Tal transformação também pode ser implementada em tempo real como contramedida de um provável aumento no uso da memória causada por outras aplicações. A diferença de qualidade que o uso de texturas com limite de resolução na aplicação é comparada pela Figura 23 e Figura 24.

Figura 23 Imagem com texturas em seu formato padrão

Figura 24 Imagem com texturas de resoluções limitadas

Nota-se que a imagem perde definição, ficando com aparência borrada, mas a qualidade da imagem mantém-se em um nível bem aceitável. Em contrapartida esta mudança beneficia dispositivos com pouca memória e com baixo processamento gráfico.

Este teste apresentou ganhos de performance na aplicação, pois a memória necessária para armazenar as texturas em seu formato padrão é superior a quantidade de memória de vídeo, forçando a busca na memória principal, o que não acontece com as texturas em seu formato convertido. Para manter a performance obtida é necessário que a aplicação mantenha detido certa quantia da memória principal.

6.4 Variação no Som

Este teste identificou o impacto na performance da diminuição da quantidade de sons reproduzidos ao mesmo tempo e a variação da qualidade do som reproduzido na aplicação 3D. A qualidade do som a ser reproduzido influencia diretamente na percepção da realidade que o jogo proporciona - e quanto mais sons, melhor imersão é fornecida. A diminuição da quantidade de sons emitidos constitui uma adaptação de controle, e a diminuição da qualidade uma adaptação do dado.

O processamento sonoro é feito, em sua maior parte, pela CPU que descomprime e/ou decodifica parte ou a totalidade do arquivo e armazena em um buffer para a reprodução, o qual é processado pelo controlador sonoro.

Todos os sons usados no teste estão no formato PCM, que necessita de pouco esforço do processador para decodificação. Os arquivos ocupam entre 1k e 55k e são de curto tempo de reprodução, entre um e cinco segundos.

O procedimento usado nos testes consiste em reproduzir uma determinada quantidade de sons em uma determinada qualidade sonora constantemente durante todo o caminho percorrido durante o teste automático. O tamanho do arquivo não traz grandes impactos, diferentemente do teste realizado com texturas, uma vez que os arquivos reproduzidos possuem tamanho pequeno, logo - não existe preocupação quanto ao armazenamento local.

O primeiro teste realizado avaliou o impacto que a qualidade do som afeta na aplicação, os tipos de qualidade usados estão listados na Tabela 9.

Tabela 9 Propriedades dos tipos de arquivos sonoros usados

	Canais	Taxa de Amostragem(Khz)	Bits de codificação	Largura de Banda Necessária(Kbps)
Tipo 1	1	8	8	88
Tipo 2	1	16	16	256
Tipo 3	1	22	16	352

Foram executadas quatro séries neste teste. Em cada série foi utilizado somente arquivos de um determinado tipo de qualidade, o quarto teste foi realizado sem som, por motivo de comparação. O número de sons reproduzidos simultaneamente foram 64 e a qualidade de textura está inalterada. O resultado é apresentado no Gráfico 2.

O ganho de performance em relação à qualidade é imperceptível, ao menos para esta plataforma. Caso os arquivos sonoros usados por jogos fossem de comprimento maior, a qualidade influenciaria no tamanho final do arquivo, e conseqüentemente no espaço

em memória RAM necessário para armazenamento. Usualmente os arquivos sonoros são pequenos e reproduzidos continuamente para fornecer a sensação de continuidade, com exceção da musica de background que costuma ser de comprimento e qualidade superior. O ganho de memória principal é irrisório, 1,9 MB do tipo 1 em relação ao tipo 3.

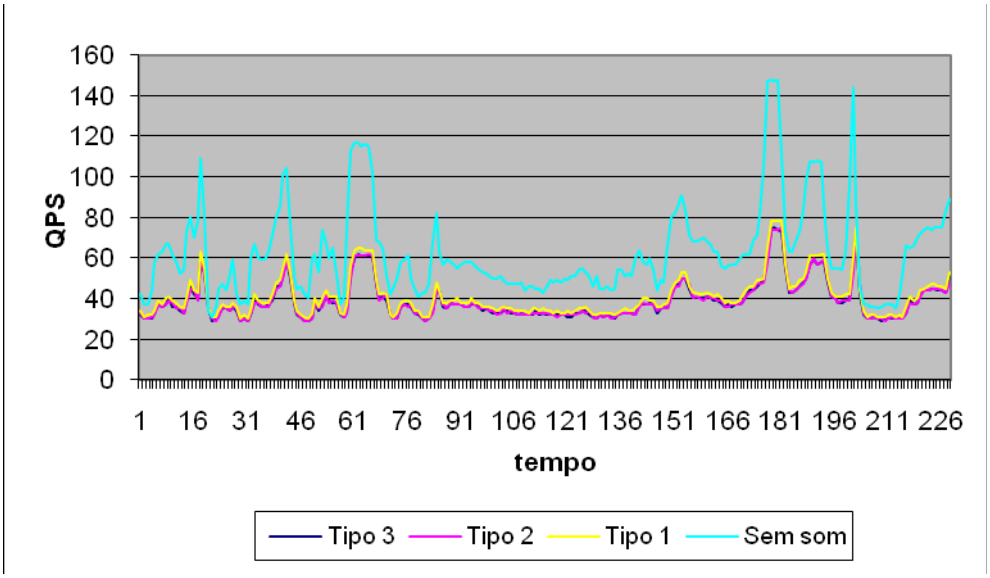

Gráfico 2 Diferença na qualidade de som em relação a performance.

Uma política de adaptação engatilhada pelo ED que use a diminuição da qualidade de arquivos sonoros antes que sejam carregados para o dispositivo é utilizada no caso d dispositivo não suportar a reprodução de certa taxa de amostragem ou quantidade de canais. A adaptação das mensagens de áudio transmitidas em tempo real de jogador para jogador são vistas como um ótimo exemplo para utilizar este mecanismo de mudança de qualidade sonora. Uma típica mensagem de 6 segundos que um usuário envia a outro sofre um atraso de acordo com os dados da Tabela 10.

Tabela 10 Tempo em segundos da transmissão de uma mensagem de voz

	Largura de banda disponível (Kbps)					
	56	**64**	**128**	**256**	**512**	**1024**
Tipo 1	9.4	8.3	4.1	2.1	1.0	0.5
Tipo 2	27.0	23.6	11.8	5.9	3.0	1.5
Tipo 3	37.7	33.0	16.5	8.3	4.1	2.1

Como no padrão MPEG-4 as mensagens são enviadas como streams, o conteúdo não necessita ser totalmente recebido para que seja possível a reprodução, ela pode ser iniciada logo que sejam recebidos os primeiros dados. Sendo assim, um atraso máximo para a mensagem de voz seria de 6 segundos, mais do que isto, o Player necessitaria armazenar a stream por completo para que seja possível a reprodução sem a perda de

continuidade. Um atraso na chegada de uma mensagem de voz urgente pode comprometer o desempenho de um jogador.

De acordo com a largura de banda disponível no usuário uma política de adaptação, acionada no AC, converte uma mensagem de voz em uma qualidade inferior e enviada ao usuário, impedindo a reprodução atrasada da mensagem.

O segundo teste teve como objetivo verificar o impacto que a quantidade de sons influencia em uma aplicação 3D. Foram executadas cinco séries de testes no ambiente de simulação, cada uma com respectivamente 64, 32, 16, 8 e nenhum som. Os arquivos possuem a qualidade definida no tipo 3 e as texturas estão em seu formato de qualidade padrões. O resultado é apresentado no Gráfico 3.

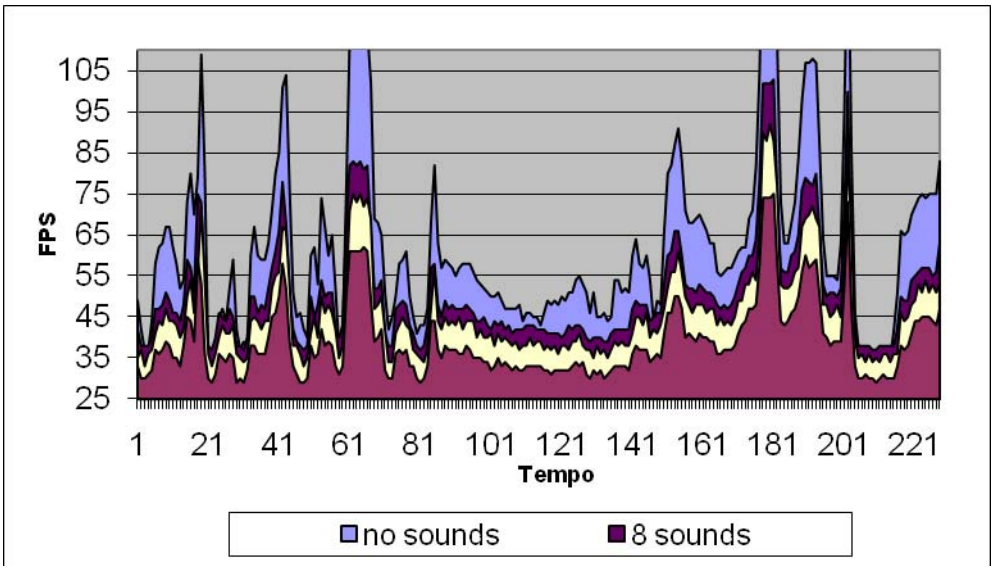

Gráfico 3 Desempenho da quantidade sonora

Neste teste observa-se ganhos de performance de todas as séries de testes em relação à série com 64 sons. Os ganhos são discretos e pequenos nesta plataforma, porém a diferença é bem mais significativa em plataformas com menor poder de processamento. Os picos no gráfico representam áreas menos complexas do jogo, por isso são renderizadas em uma taxa bem maior que as demais. Não existe ganho de memória pois todos os sons são carregados, e em tempo de execução, é decidido qual som deverá ser reproduzido.

Um quadro comparativo contendo a média do ganho de performance medido em taxa de quadros em relação à série “64 sons” juntamente com seu respectivo desvio padrão é observado na Tabela 11.

Tabela 11 Ganho de performance em relação ao teste reproduzindo com 64 sons

	32 sons	16 sons	8 sons	Nenhum som
Média (QPS)	7	10	12	23
Desvio Padrão	1.47	2.94	3.1	3.4

O pequeno desvio padrão apresentado se deve a própria concorrência entre os processos, os quais durante os testes influenciam para uma maior ou menor medida do valor. O gráfico também justifica o fato de que o processamento sonoro depende da CPU, não importando a cena renderizada, pois a diferença de porcentagem entre uma série e outra permanece praticamente constante no decorrer do teste.

Uma política de adaptação é engatilhada pelo ED impedindo a execução de mais sons que o dispositivo pode suportar. A ativação da mesma política pode ser feita pela aplicação MPEG-J.

Os testes realizados indicam que o uso de um arquivo sonoro de qualidade inferior resulta em um som abafado e diferenças de timbre perceptíveis. A diminuição da quantidade de sons reproduzidos acarreta na perda da imersão, o usuário pode estar vendo um objeto que emita um som, quando este não está sendo reproduzido.

6.5 Políticas de adaptação

Depois de realizados os testes foram inseridos na GUI do usuário o valor mínimo de 30 QPS. O ED então gera facilmente as políticas de adaptação, baseados nos resultados obtidos.

Como o objetivo do *framework* é fornecer o melhor desempenho possível, o dispositivo deve ser capaz de executar o jogo com as qualidades no mínimo, por este motivo, não será considerada uma política de adaptação que remova o processamento de texturas, pois a degradação na qualidade de imagem é exageradamente notável. Esta regra não será aplicada ao áudio, visto que costumeiramente, a opção de torná-lo desabilitado é fornecida na maioria dos jogos para aumento de performance.

Para o tratamento de texturas a seguinte política será adotada: Como a plataforma não possui memória suficiente para armazenar todas as texturas em seu formato padrão durante toda a execução da aplicação, o que acarreta perdas na performance, o ED decide por acionar a adaptação para a conversão das texturas. Esta adaptação é feita antes do carregamento do jogo e localmente no cliente, pois não é possível alterar o conteúdo em memória a não ser que a aplicação seja carregada novamente.

Caso o programador não fornecer os arquivos de configuração do teste estendido ele deve inserir a política manualmente. Como exemplo, caso a quantidade de memória de vídeo for menor do que 8 MB limitar a dimensão da textura em *64x64pixels*, definindo a adaptação do Tipo 1. Caso esteja entre 8M e 16MB, definir a adaptação do tipo 2 (limitar a dimensão da textura a no máximo 256x256 pixels), caso for maior que 16MB, não haverá necessidade de adaptação. Esta política será mapeada em código segundo a Figura 25.

```
Vm = EPResults.Video.VideoMen;

if (Vm <= 8M)
{
        AdaptType.TextureAdap=2;

}
else if( (Vm > 8) AND (Vm <= 16))
{
        AdaptType.TextureAdap=2;
}
else    AdaptType.TextureAdap=0;

…….…
AC_Register(Client_IP(),AdaptType);
```

Figura 25 Checagem de portabilidade

Para a adaptação em função da carga sobre a CPU, e quantidade de quadros renderizados por segundo, será utilizada a quantidade máxima de sons reproduzidos simultaneamente. O efeito da quantidade de sons afeta tanto a quantidade de quadros quanto a carga sobre a CPU.

Automaticamente o ED mapeia os valores da tabela Tabela 11 para se adequar a taxa de quadros por segundo. Caso o programador necessitasse inserir o código manualmente, o resultado estaria de acordo com a Figura 26, porém é difícil saber o comportamento da máquina do cliente tão perfeitamente.

```
int AdaptHandler (event in fps, event in cpu, event in
mem)
{       int fps_d = (User.FPS – fps);
        If  (nsound= 64)
        {       if ( fpd_d < 0) nsound = 64;
                else if ( fps_d < 7) nsound = 32;
                else if ( fps_d < 10) nsound = 16;
                else if ( fps_d < 12) nsound = 8;
                else nsound = 0;
        }
        else if (nsound= 32)
        {       if ( fpd_d < -7 ) nsound = 64;
                else if ( fpd_d < 0) nsound = 32;
                else if ( fps_d < 10) nsound = 16;
                else if ( fps_d < 12) nsound = 8;
                else  nsound = 0;
        }
        else if (nsound=16)
        {
                else if ( fpd_d < -10 ) nsound = 64;
                else if ( fpd_d < -3 ) nsound = 32;
                else if ( fpd_d < 0) nsound = 16;
                else if ( fps_d < 12) nsound = 8;
                else nsound = 0;
        }
        else if(nosund=8)
        {
                else if ( fpd_d < -12 ) nsound = 64;
                else if ( fpd_d < -5 ) nsound = 32;
                else if ( fpd_d < -2) nsound = 16;
                else if ( fps_d < 0) nsound = 8;
                else if ( fps_d < 23) nsound = 0;
        }
        else if (nsound=0)
        {
                else if ( fpd_d < -23 ) nsound = 64;
                else if ( fpd_d < -16 ) nsound = 32;
                else if ( fpd_d < -13) nsound = 16;
                else if ( fps_d < -11) nsound = 8;
                else nsound = 0;
        }
        ChangeNSound(nsound);
```

Figura 26 Adaptação em função da CPU e taxa QPS

Para o tratamento da variação da largura de banda, usaremos a qualidade do áudio para diminuirmos o tráfego recebido, seguindo os valores da Tabela 9 a política de adaptação analisa se a largura de banda total é maior que a taxa necessária para o envio de todos os fluxos que estão sendo destinados ao cliente em certo momento, caso não seja, ele adapta todos os fluxos para possibilitar o envio dentro da largura de banda suportado pelo cliente. Vale ressaltar que as mensagens de sincronização terão prioridade para ocupar toda a largura de banda, se necessário, e caso não haja largura de banda suficiente o buffer de

mensagens de áudio será descartado. A reflexão em código desta adaptação é observada na Figura 27. O ED não influencia na adaptação quanto à largura de banda, a adaptação fica a cargo do AC e não do cliente, responsável por enviar a taxa atual para o AC quando detectado uma variação que ultrapasse algum limite.

```
//no cliente
int ThourghPutHandler (event in BW_aval)
{       AdpatType.Bandwidth= BW_aval;
        AC_Register(Client_IP(),AdaptType);
}

//no AC
int sendbuffer( AudioBuffer,MsgBuffer)
{       itn kbps_d, kbps_r;
        kbps_d = AdaptType(ClientId).bandwidth -
        MsgBuffer.TotalKbps;
        if (AudioBuffer.Tkbps > kbps_d)
        {       kpbs_r = kbps_d / AudioBuffer.nstreams;
                if ( kbps_r > 256) send(AudioBuffer ,256);
                else if (Kbps_r > 88) send(AudioBuffer ,88);
                else send (Audiobuffer,0);
        }
}
```

Figura 27 Tratamento de Eventos

6.6 Execução da Aplicação

Após a execução do ED e definidas as políticas de adaptação, o jogo é executado, juntamente como a aplicação MPEG-J e MR.

A execução foi conduzida em um ambiente de simulação composto de três maquinas clientes (Cliente_1, Cliente_2 e Cliente_3) e um servidor. Os Clientes_2 e Cliente_3 possuem seus recursos estáveis e processamento local suficiente para execução do Jogo, por isso não foi considerado nenhum método de adaptação. O Cliente_1, objeto de estudo dos resultados, onde foi executado o ED, possui recursos em condições instáveis, a carga concorrente sobre o processador será simulada por uma aplicação que realiza laços infinitos em certos intervalos de tempo. A variação da largura de banda é feita na saída do AC para o Cliente_1.

Durante a execução da aplicação, o ambiente foi explorado pelos três clientes, gerando mensagens de atualização entre ambos, juntamente com mensagens de voz enviadas

do Cliente_2 e Cliente_3 ao Cliente_1. A Adaptação em função da largura de banda, executada pelo AC, é realizada adaptando-se o conteúdo das mensagens de voz enviadas para o Cliente_1. Durante a execução da aplicação a largura de banda variou entre 800, 600, 400 e 200 kbps, como demonstração, os Clientes 2 e 3 enviaram mensagens de voz para o Cliente_1 ao mesmo tempo, cada mensagem a uma taxa de 352Kbps. A adaptação é acionada quando não existe largura de banda suficiente para o envio dos fluxos de áudio para o Cliente_1, ou seja, quando a largura de banda disponível está abaixo de 704Kbps. O Gráfico 4**Erro! Fonte de referência não encontrada.** apresenta o tempo resultante desde o início do envio de uma mensagem de áudio de 1(um) segundo dos Clientes 2 e 3 até o seu completo armazenamento no buffer de reprodução do Cliente 1.

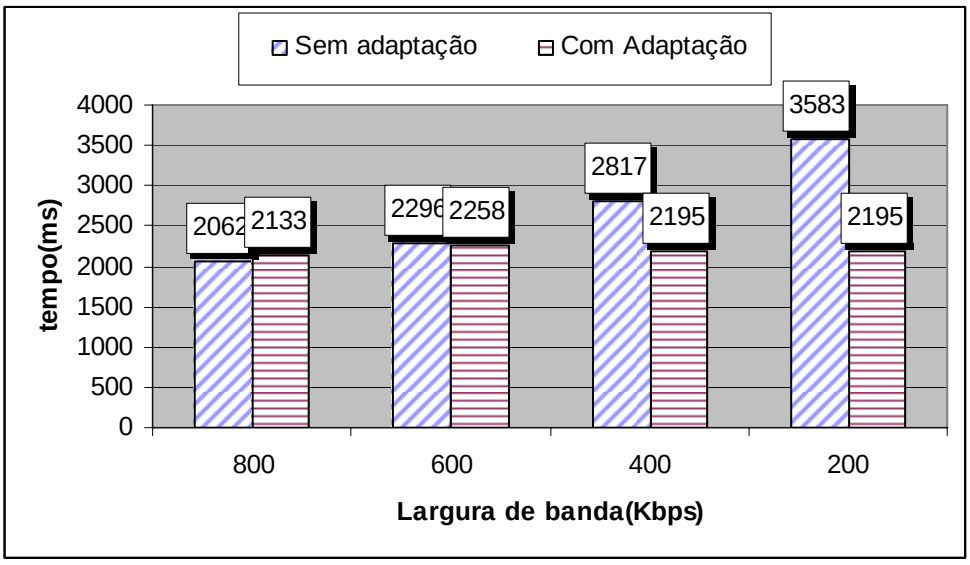

Gráfico 4 Adaptação em função da largura de banda.

O tempo, em milissegundos, de atraso é computado como o tempo de envio até o AC somado com o tempo de entrega ao Cliente_1. Quando ocorre adaptação este tempo é somado ao gasto pelo processamento no AC. Nos testes a mídia atravessa um servidor, existindo um AC é verificada a necessidade de adaptação e posteriormente realizada.

A soma dos fluxos enviados totaliza 704 kbps. A adaptação foi aplicada quando a largura de banda esteve em 600kbps, resultando em os dois fluxos de 256 kbps(512kbps no total), em 400kbps e 200kbps, resultando em dois fluxos de 88kbps(totalizando 176 kbps). A configuração dos fluxos resultantes segue os descritos na Tabela 9.

A interpretação do Gráfico 4indica que para larguras de banda onde não existe necessidade de adaptação, acima de 704kbps, o processo sofre atraso de 71 ms devido ao

processamento pelo AC. Em casos em que a adaptação é realizada, o tempo do processamento é somado com o tempo da adaptação, o que explica a diferença maior para entrega do fluxo adaptado em 600kbps em relação ao entregue em 800kbps. Em 400 e 200 kbps o tempo é o mesmo, pois é realizada a mesma adaptação, e inferior ao realizado em 600kbps, resultado de uma taxa de bits de saída menor.

O processo de adaptação, apesar de gerar um pequeno atraso quando a largura de banda é suficiente para o trafego de dados em seu formato padrão, é viável nos casos em que existe a necessidade de adaptação. Para velocidades baixas seu uso é completamente justificado, chegando a conseguir 1388ms de vantagem na recepção do fluxo em relação ao envio convencional em seu formato padrão. O objetivo da adaptação, no caso da largura de banda, foi manter um atraso constante do fluxo recebido, não importando a taxa de bits atual.

Durante a execução do jogo o Cliente_1 explorou o ambiente encontrando situações em que a cena se tornava complexa, atingindo uma taxa de quadros menor. Foi realizada uma primeira execução sem adaptação-onde foram gravados em arquivo todos os comandos do participante- logo em seguida realizado uma reprodução do comportamento do jogador com a adaptação ativada, para não haver desigualdade na medida dos valores. O resultado é exposto no Gráfico 5, a execução é feita com 64 sons reproduzidos simultaneamente.

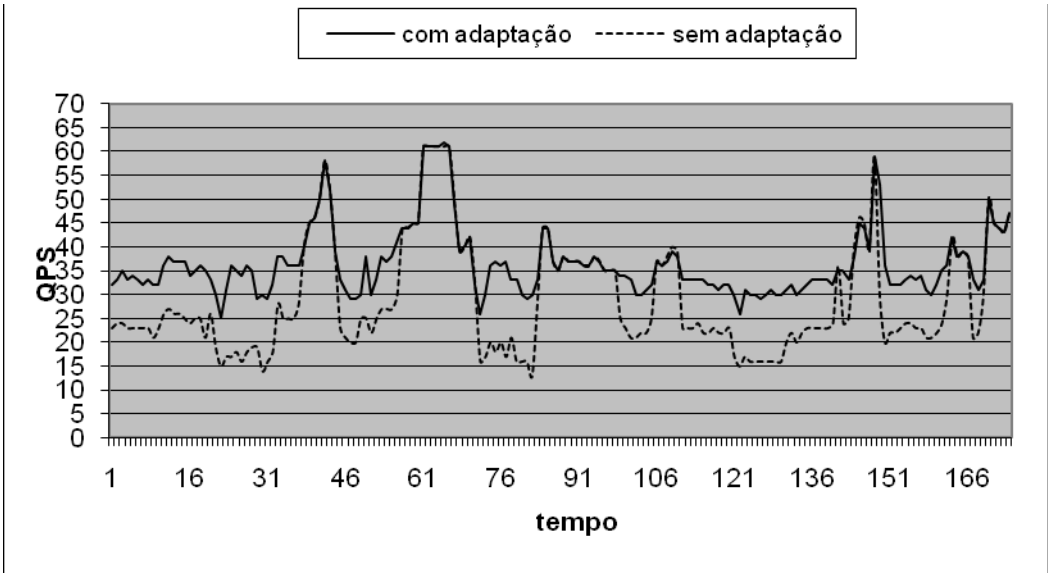

Gráfico 5 Adaptação em função da CPU e QPS

Além da complexidade da cena foi introduzida uma aplicação para aumentar a carga do processador durante a execução do jogo nas seguintes faixas de tempo: 19-32, 72-82 e 121-131, resultando na perda de 25% do processamento para esta aplicação. O resultado é visto claramente no gráfico, analisando a curva sem adaptação. Para melhor visualização dos

resultados, criaram-se mais dois gráficos, Gráfico 6 e Gráfico 7, a partir do Gráfico 5. As barras indicam que a exibição de uma taxa de quadros entre um intervalo [X, X+5] ocorreu Y% do tempo.

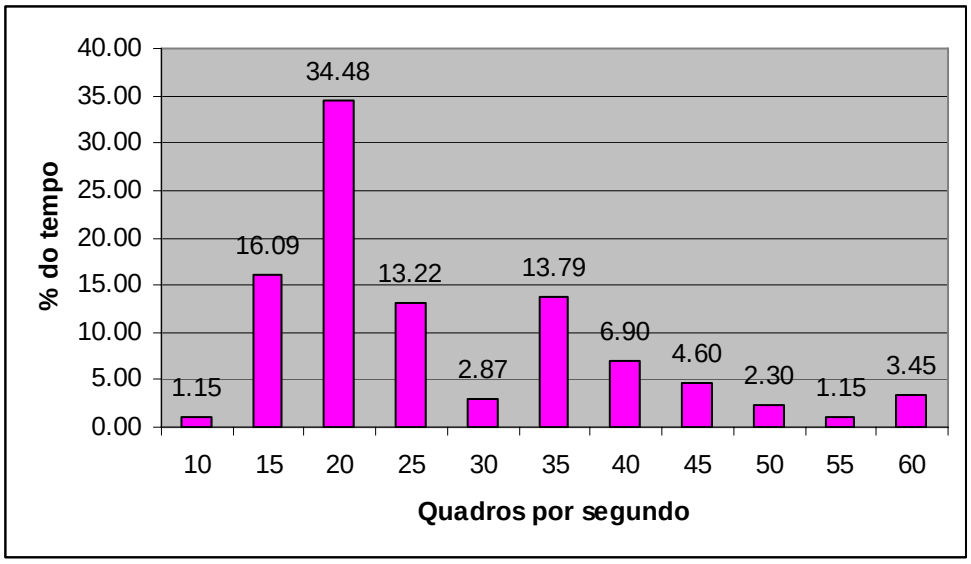

Gráfico 6 Dispersão de QPS durante a execução sem adaptação

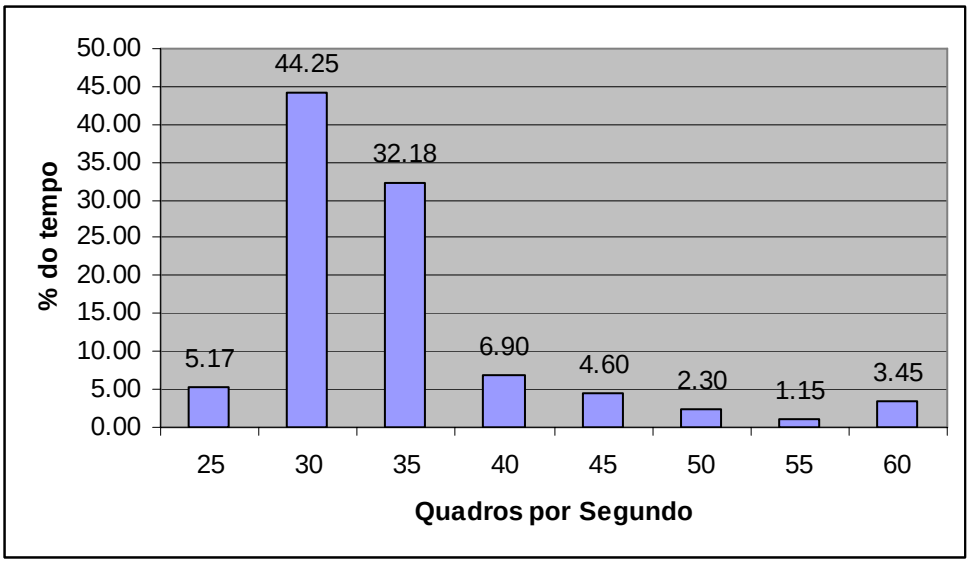

Gráfico 7 Dispersão de QPS durante a execução com adaptação

Os dois gráficos exibem claramente o comportamento da aplicação. A execução com adaptação apresentou em quase 95% do tempo uma taxa de quadros maiores do que 30, em contrapartida, a execução sem adaptação manteve a taxa de quadros acima de 30 em apenas 35% do tempo, o que torna o uso da adaptação justificado, por não dizer essencial para uma melhor experiência do usuário. Os demais 5% do tempo em que os quadros são

renderizados abaixo da taxa estipulada no início da aplicação são provenientes do tempo em que o MR detecta a queda até a sua normalização resultante da ativação da adaptação.

No Gráfico 5 nota-se que em certos casos a execução da aplicação com a adaptação presente é idêntica a execução da aplicação sem a presença da adaptação, isto se deve ao fato de não haver nenhuma adaptação ativada. Neste caso as duas execuções compartilham a mesma configuração da aplicação, ou seja, 64 sons reproduzidos simultaneamente.

O objetivo da adaptação, no caso da carga sobre a CPU e da complexidade da cena, é proporcionar uma mínima taxa de quadros por segundo, neste caso, 30 QPS. Tal objetivo foi alcançado com excelentes resultados, proporcionando ao usuário uma experiência livre de *slowdowns*, com pouca perda da sensação de imersão ao ambiente.

Na próxima sessão são discutidas as considerações finais sobre o trabalho, bem como trabalhos futuros que enriquecerão a qualidade do *framework* MMGAME.

7 CONCLUSÕES

Este trabalho especificou e implementou uma estrutura de adaptação de aplicações que envolvem mídia 3D, 2D e áudio para o uso em Jogos 3D. Com esta estrutura, aplicações se tornam cientes de adaptação e pouco de seu código necessita de mudanças. Apesar de o programador estar ciente da adaptação, esta tarefa é facilitada com a inserção de métodos que proporcionam a obtenção em tempo real da performance alcançada em cada terminal cliente, aliviando o programador para a concentração nos demais pontos do projeto do Jogo. Para atingir o desempenho desejado pelo usuário, é especificada a quantidade mínima de quadros por segundo, juntamente com as opções de degradação.

Uma vantagem desta estrutura é a habilidade de adaptar aplicações para diferentes dispositivos, de baixa capacidade de processamento como os telefones móveis, até set-top-boxes e PCs. Isso é feito por meio de um componente de avaliação de capabilidades que identifica, de antemão, se o dispositivo é capaz de processar uma determinada aplicação 3D. Alterações nos recursos do sistema, tais como carga de CPU, capacidade de memória, largura de banda da rede, atraso na rede etc., engatilham adaptação. A adaptação pode ser feita localmente ou no servidor de conteúdo.

Uma aplicação escrita com MMGAME livra o usuário da preocupação com a variação do comportamento que sua aplicação apresenta no decorrer de uma partida, seja ela causada pelo aumento da complexidade de uma cena ou devido à variação de algum recurso usado no momento pela aplicação, fornecendo sempre a mesma performance sobre as mais variadas possibilidades.

A estrutura se integra à especificação de adaptação definida pelo padrão MPEG-4 e seu padrão emergente MU, o que permite ao framework maior campo de atuação, atingindo um nicho maior de dispositivos não se prendendo a soluções proprietárias e conseqüentemente pagas.

Um dos aspectos únicos do trabalho é a adaptação de mídia 3D, ao contrário dos trabalhos normalmente encontrados na literatura que tratam a adaptação basicamente no âmbito de vídeo, áudio, textos e páginas da web. Outro aspecto importante foi considerar múltiplos usuários no acesso a conteúdo adaptável, bem como a troca deste conteúdo entre participantes, na maioria das soluções propostas as aplicações são apenas mono-usuário e não fornecem o suporte para a comunicação multiusuário.

7.1 Considerações Finais

Os jogos virtuais 3D multiusuário são aplicações que exigem sincronismo no compartilhamento de um mesmo ambiente virtual, o que torna a sua adaptação em ambientes ubíquos, um grande desafio. A adaptação de aplicações é normalmente tratada na literatura considerando-se apenas aplicações multimídia e tipos de dados como vídeo, áudio e texto. Este trabalho identifica os tipos de dados que sofrem adaptação em ambientes de jogos virtuais 3D multiusuário e propõe um *framework* que, integrado ao sistema MPEG-J, suporta a adaptação de dados e controle em ambientes virtuais tridimensionais. Este trabalho diferencia-se dos demais descrevendo um *framework* de adaptação que, integrado ao sistema MPEG-J, tem o potencial para suportar complexas aplicações em ambientes virtuais tridimensionais, acessadas de diferentes dispositivos.

A adaptação mostrou ser uma ótima solução para a portabilidade de uma aplicação bem como possível resolução para o problema da manutenção de uma taxa de quadros constante, em uma aplicação 3D, proporcionando ao usuário interação em tempo real.

O padrão MPEG-4 demonstrou ter potencial para se destacar dentre as demais soluções disponíveis no mercado por permitir independência de plataforma, controle total da aplicação, extensões multiusuário e fácil desenvolvimento.

A área de jogos 3D, além ser de grande interesse do autor, proporcionou grande desafio para obtenção dos resultados, provindo em uma solução que não somente se provou concreta como também recompensadora.

7.2 Trabalhos Futuros

Vários trabalhos futuros são vislumbrados a partir do projeto desenvolvido visando tornar o *framework* uma solução completa. Nas seções seguintes será detalhada cada uma destas extensões.

7.2.1 Finalização da Implementação do Jogo Moving Target

O Jogo Moving Target encontra-se ainda em processo embrionário, pois foi construído pelo autor apenas para testar alguns mecanismos de adaptação. Para a total

validação do *framework* é necessário a construção de um jogo que utilize variadas técnicas de realismo e proporcione a mesma experiência ao usuário que um jogo comercial. Aplicando o framework em uma solução completa não só demonstraria a potencialidade da ferramenta como também provaria que o padrão MPEG-4, uma solução padrão para todas as plataformas sem custo para o desenvolvedor, consegue alcançar os mesmos níveis de qualidade das demais opções disponíveis no mercado, que no entanto, são pagas.

7.2.2 Mecanismo para auxilio na criação de testes de simulação

Atualmente o programador não dispõe de uma ferramenta auxiliar para construção de cenas e comportamento simulado usados durante o teste estendido no ED.

Esta ferramenta auxiliaria o programador a montar, a partir dos dados do próprio jogo 3D, um estado que consiga representar melhor tanto a complexidade da cena quanto o comportamento de vários usuários em um ambiente 3D, conseguindo assim melhores resultados durante o teste estendido.

7.2.3 Execução do *framework* em demais dispositivos

A implementação atual do MMGAME funciona apenas no sistema operacional Windows™, para a execução do *framework* em demais ambientes é necessário que se modifique a parte dependente de plataforma dos componentes ED e MR, além de contar com a presença na plataforma e um Player MPEG-4.

A verificação do comportamento do *framework* em dispositivos móveis é essencial para justificar a necessidade do uso da adaptação em conter variações de recursos, tanto de rede como de sistema. Infelizmente como o projeto não obteve apoio financeiro, não foi possível a aquisição de dispositivos para este fim.

7.2.4 Histórico

Haverá casos em que o programador não conseguirá representar de maneira correta o comportamento do usuário no teste estendido, seja por falta de experiência ou pela escolha de uma abordagem incorreta.

O uso de um mecanismo de histórico, que gravaria as taxas de quadro durante a execução da aplicação, seja com a habilitação da adaptação ou não, seria de grande utilidade

para melhor adequar as políticas de adaptação ao específico usuário que detêm posse de um determinado dispositivo.

Um método deveria ser criado possibilitando que a aplicação aprenda a se comportar de maneira personalizada com o uso constante pelo usuário, o qual resultaria na definição de políticas de adaptação especializadas para aquele determinado usuário.

Este método seria muito útil em aplicações em que o usuário pode escolher diferentes papéis, ou profissões, para alcançar o objetivo do jogo.

7.2.5 Criação de uma linguagem para adaptação

Mesmo que o ED consiga gerar as políticas de adaptação em tempo real, ainda é necessária a intervenção do programador para a escolha de uma melhor política frente à escassez de um determinado recurso.

O uso de uma linguagem voltada ao uso da adaptação, a qual seja interpretada em tempo real, ou gere código Java para o uso na aplicação MPEG-J, facilitaria ainda mais o papel do programador em definir o melhor método para a escolha da adaptação que será ativada.

Este método seria de grande utilidade quando várias constantes devem ser analisadas para a determinação de um mecanismo de adaptação.

7.2.6 Contribuição formal ao padrão MPEG-4

Conforme mencionado na sessão 4.5, a MPEG-J fornece mecanismos de suporte a monitoramento de recursos, porém não são adequados para determinados tipos de aplicação, além de falta de outros métodos necessários em uma determinada situação.

Como uma contribuição do trabalho, será redigido um documento de sugestão para alteração e complementação da API MPEG-J, a ser avaliado em um dos vários encontros da ISO destinado ao padrão MPEG-4.

7.2.7 Experimento

Em um passo final, após a finalização da implementação do jogo e o correto complemento do *framework*, o pacote será disponibilizado para testes de prateleira tanto para

o uso pelo programador quanto no nível de satisfação do usuário alcançado pelo uso de uma aplicação que usa do artifício da adaptação para promover melhor serviço.

No ponto de vista do programador será avaliada a necessidade de uma melhor documentação ou adequação das classes para melhor compreensão do *framework*. No lado do usuário, será avaliada se a adaptação atende a necessidade da maioria dos jogadores, os quais necessitam de interação em tempo real em seus jogos, e as modificações que o *framework* deverá sofrer para cumprir este papel.

REFERÊNCIAS

[AIM03] "Radio Frequency Identification (RFID) home page". Disponível em http://www.aimglobal.org/technologies/rfid/.[consulta: março 2003]. Association for Automatic Identification and Data Capture Technologies.

[Baggio01] Bagio, Aline. "Design and Early Implementation of the Cadmium Mobile and Disconnectable Middleware Support", INRIA Research Report 3515.

[Baschieri02] Baschieri, Francesco; Bellavista, Paolo; Corradi, Antonio. "Mobile Agents for QoS Tailoring, Control and Adaptation over the Internet: the ubiQoS" Video on Demand Service Proceedings of the 2002 Symposium on Applications and the Internet (SAINT.02)

[Buer02] Daniel Bauer, Sean Rooney, Paolo Scotton, IBM Research- Zurich Research Laboratory, "Network Infrastructure for Massively Distributed Games" ,Netgames 2002 April 1617, 2002, Braunschweig,

[Bettner01] Bettner, P. e Terrano, M. "1500 archers on a 28.8: Network programming in Age of Empires and beyond", in The 2001 Game Developer Conference Proceedings, San Jose, CA, Mar. 2001.

[Bigos96] Andy Bigos "Exploiting Consumer Class 3D Hardware Acceleration for Real-Time Entertainment" Designing Real-Time 3D Graphics for Entertainment SIGGRAPH '96 Course

[Bjork02] Staffan Bjork, Jussi Holopainen, Peter Ljungstrand an. Regan Mandryk - Special Issue on Ubiquitous Games."Personal and Ubiquitous Computing" (2002) 6:358–361 Springer-Verlag London Ltd 2002

[Bowers00] Shawn Bowers, Lois Delcambre,David Maier, Crispin Cowan,Perry. Wagle,Dylan, McNamee, Anne-Françoise Le Meur,Heather Hinton. "Applying Adaptation Spaces to Support Quality of Service and Survivability". Defense Advanced Research Projects Agency, DARPA

[Burkhard02] Burkhard, T J., Henn, H., Hepper, S., Rindtorff, K., Schack, T. (2002) "Pervasive Computing: Technology and Architecture of Mobile Internet Applications". Ed. Addison-Wesley. 410 pags.

[Carroll03] Eddy Carroll "Interactive Multi-User Computer Games". Disponível em http://iol.ie/~ecarroll/ [consulta: março 2003]

[Clay96] Sharon Rose Clay, Silicon Graphics Computer Systems. "Optimization for Real-Time Entertainment Applications on Graphics Workstations".

Designing Real-Time 3D Graphics for Entertainment. SIGGRAPH '96 Course

[Chang00] Fangzhe Chang and Vijay Karamcheti. "Automatic Configuration and Run-time Adaptation of Distributed Applications". Ninth IEEE Symposium on High Performance Distributed Computing (HPDC), August 2000.

[Cadmium00] Cadmium Project (2000) "The Cadmium Project". Disponível em www-sor.inria.fr/projects/cadmium/index.html [consulta: Novembro 2000]

[Coda00] Coda and Odissey (2000) "Mobile Information Access" Disponível em www.cs.cmu.edu/afs/cs/project/coda/Web/coda.html,[consulta: Novembro 2000]

[Cortez02] Cortez R. e Araujo R. B.," Implementação de um conversor VMRL/MPEG-4", Relatório Técnico, DC-UFSCar, Maio 2002.

[Crawford97] Chris Crawford – "The Art of Computer Game Design – Chapter 1" 1996-7 Washington State University

[Dornan01] Dornan, A. (2001) "Wireless Communication: o guia essencial de comunicação sem fio". Editora Campus, Rio de Janeiro.

[Eisenstein00] Eisenstein, J., Vanderdonckt, J. and Puerta, A. (2000) "Adapting to Mobile Contexts with User-Interface Modelling". IEEE Computer

[Fitzek02] Frank Fitzek, Gerrit Schulte, Martin Reisslein. "System Architecture for Billing of Multi–Player Games in a Wireless Environment using GSM/MUTS and WLAN Services". NetGames2002, April 16–17, 2002, Braunschweig, Germany.

[Fox97] Fox, S., Gribble, E., Brewer, A. and Elan, A. (1997). "Adapting to Network and Client Variability via On-Demand Dynamic Distillation" University of California at Berkeley, July.

[Gecse99] Jan Gecsei, "Adaptation in Distributed Multimedia Systems" Université de Montréal.

[GSMW02] GSM World (2002) "The Wireless Revolution" Disponível em http://www.gsmworld.com. [consulta 20/12/ 2002].

[GP402] Greoff Crammond's "Grand Prix 4" – Distribuido pela Infogrames – Desenvolvido pela Simergy – PC GAME - US Relese 2002

[Harte02] Harte, L., Levine, R. e Kikta, R.(2002) "3G Wireless Demystified". McGraw-Hill, 496 pags.

[Helman96] James Helman Silicon Graphics Computer Systems."Architecture and Performance of Entertainment Systems" Designing Real-Time 3D Graphics for Entertainment SIGGRAPH '96 Course

[Hoffman96] Wes Hoffman.”Simulation Database Design for Visual Simulation and Entertainment” Designing Real-Time 3D Graphics for Entertainment SIGGRAPH ‘96 Course

[House96] DONALD H. HOUSE “Overview of Three-Dimensional Computer Graphics Visualization Laboratory”, College of Architecture, Texas A&M University ACM Computing Surveys, Vol. 28, No. 1, March 1996

[IDGA02] Interactive Digital Software Association. Disponível em http://www.idsa.com [consulta: novembro 2002]

[ISO02] ISO/IEC JTC1/SC29/WG11 N5285 “CODING OF MOVING PICTURES AND AUDIO “MPEG-4 Systems - October 2002

[JNI02] Java Native Interface Disponível em http://java.sun.com/docs/books/tutorial/native1.1/ [consulta: julho 2002]

[Johnst96] Eric Johnston LucasArts Entertainment “Prototyping and Portability of Hardware-Assisted 3D Games”. Designing Real-Time 3D Graphics for Entertainment SIGGRAPH ‘96 Course

[KAN01] Kan, H. Y., Duffy, V. G., Su, C. “An Internet Virtual Reality Collaborative Environment for Effective Product Design”, Computers in Industry, Vol. 45, 2001, p. 197-213.

[Lara01] Lara, E., Wallach D. S. and Zwaenepoel, W. (2001) “Puppeteer: Component-based Adaptation for Mobile Computing”, Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems, San Francisco, California.

[Limber96] Michael Limber Angel Studios”Creating Compelling Real-Time Content” Real-Time 3D Graphics for Entertainment SIGGRAPH ‘96 Course

[Living97] Living Worlds Proposal Draft 2,Disponível em http://www.vrml.org/WorkingGroups/livingworlds/draft_2/index.htm [consulta: Fev 2001]

[Martim02] Ioana M. Martin, “ARTE - An Adaptive Rendering and Transmission Environment for 3D Graphics” IBM T. J. Watson Research Center

[Massaud00] Massaud, E. M.(2000) “Estudo de técnicas de alocação dinâmica de recursos e sincronismo para serviços de multimídia num sistema móvel celular CDMA de banda larga”. Dissertação de Mestrado. Departamento de Engenharia de Telecomunicações e Controle. Escola Politécnica, USP, São Paulo.

[McIlhagga98] Malcolm McIlhagga, Ann Light, and Ian Wakeman, “Towards a design methodology for adaptive applications”, The fourth annual ACM/IEEE

international conference on Mobile computing and networking October 25 - 30, 1998, Dallas, TX USA, pp 133-144.

[Miller01] Miller, M. (2001) “Descobrindo o Bluetooth”, Editora Campus, 289 pags.

[NPD03] The NPD Group – “The NPD Group Reports Increase in PC Software Sales for 2002” - Finance and Business Categories Experience Double-Digit Growth - NPD Press Release January 27, 2003 disponível em http://www.npd.com/ [consulta: fevereiro 2003]

[NPDb03] The NPD Group – “The NPD Group Reports Annual 2002 U.S. Video Game Sales Break Record” - NPD Press Release - January 28, 2003. Disponível em http://www.npd.com/. [consulta:fevereiro 2003]

[Noble98] Noble, B. (1998) “Mobile Data Access”, doctoral thesis, School of Computer Science, Carnegie Mellon university, May, CMU-CS-98-118.

[Octave03] “Sistema de entrada Octave” E-Acute. Disponível em www.e-acute.fr [consulta: abril 2003]

[Odyssesy00] B.D. Noble and M. Satyanarayanan. “Experience with adaptive mobile applications in Odyssey” Mobile Networks and Applications 4 (1999) 245–254 Baltzer Science Publishers BV

[Quake99] Quake III Arena – Desenvolvido por Id Software, Distribuído por Activision. PC GAME - US RELEASE 1999

[Rao01] Depak Rao. “Efficient and Portable Middleware for application-level Adaptation”, Master of Science thesis, Department of Computer Science and Applications Virginia Tech, Blacksburg, Virginia. 2001.

[Rohf96] John Rohlf and James Helman ,Silicon Graphics Computer Systems,” IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics” Designing Real-Time 3D Graphics for Entertainment
SIGGRAPH ‘96 Course

[ROEHL97] Roehl, B. et alii. “Late Night VRML 2.0 with Java”. Ziff-Davis Press, 1997, 710 páginas

[Sabadello01] Sabadello M. “Small Group Multiplayer Games”. Viena University of Technology, Austria. Abril/Maio 2001

[Simpson02] Jake Simpson. “Game Engine Anatomy” Extreme Tech April 12, 2002 disponível em http://www.extremetech.com/category2/0,3971,23466,00.asp [consulta: outubro 2002]

[Spylock03] SpyLock – “Logging keyboards events” Disponível em www.spylock.com [consulta: junho 2003]

[Square03] Final Fantasy XI Online PC Press – News Section, Disponível em www.squaresoft.com [consulta: abril 2003]

[Svedjedal02] Johan Svedjedal - Uppsala University- "The digital theory of games" Gaming for the academy ESSAY

[Steven97] Armando Fox Steven D. Gribble Eric A. Brewer Elan Amir."Adapting to Network and Client Variability via On-Demand Dynamic Distillation" University of California at Berkeley jul 1997

[T902] Sistema de Entrada T9. Disponível em www.t9.com/demo_page2.html [consulta outubro 2002]

[Terra03] Asgard technology Press –"Linking Plataforms" Disponível em http://www.terraforge.com/technology.asp?page=asgard [consulta: janeiro 2003]

[Tran02] Lan Tran. "Difference Between Gaming Consoles (PS/PS2, GameCube, Xbox) and PC" May 13, 2002

[MUTS01] MUTS (2001) "Universal Mobile Telecommunications System" (MUTS). Web Proforum Tutorials. Disponível em http://www.iec.org [consulta: agosto 2001].

[Underhill01] Underhill, W. E Dickey, C. (2001) "Lost in a mobile maze". Newsweek. pp. 17. May.

[Waldman97] Waldman, H. e Yacoub, M. D.(1997) "Telecomunicações - Princípios e Tendências", Série Universidade, Editora Erica, 287 págs

[Wolberg90] WOLBERG, G. 1990. "Digital Image Warping". IEEE Press, Los Alamitos, CA.

[Wolf00] Mark J. P. Wolf "Genre and the Video Game, Chapter 6" of The Medium of the Video Game 2000 University of Texas Press

ANEXO A – SIMULAÇÃO DA VARIAÇÃO DA CPU

Este teste, diferente dos apresentados na seção 6, não se baseará na alteração da configuração do decodificador usado pela aplicação, e sim na variação da porcentagem de CPU detida pelo jogo. Com este teste pretendeu-se estipular o quanto e de que forma a CPU influencia em uma aplicação 3D, precisamente na renderização da cena, como também discutir a necessidade de adaptação em um ambiente 3D de constante mudança, como notados em jogos.

Este teste responde a perguntas, até o momento, não profundamente comentadas. Quando existe a necessidade de uma adaptação? Quanto tempo uma adaptação será aplicada? Como resolver a necessidade da adaptação em função da aplicação ? Estas perguntas serão respondidas no decorrer do teste.

O teste submeteu o ambiente de simulação a uma execução com menor quantidade de CPU disponível, simulando a concorrência entre processos de maior prioridade, ou até mesmo, simulando plataformas de menor poder de processamento. Este teste consistiu de três séries, a primeira com 100% do processador disponível, a segunda com 80% e a terceira com 20%. Não foi processado nenhum som justamente para demonstrar o quanto a CPU influencia somente na renderização. O resultado é mostrado no Gráfico 8.

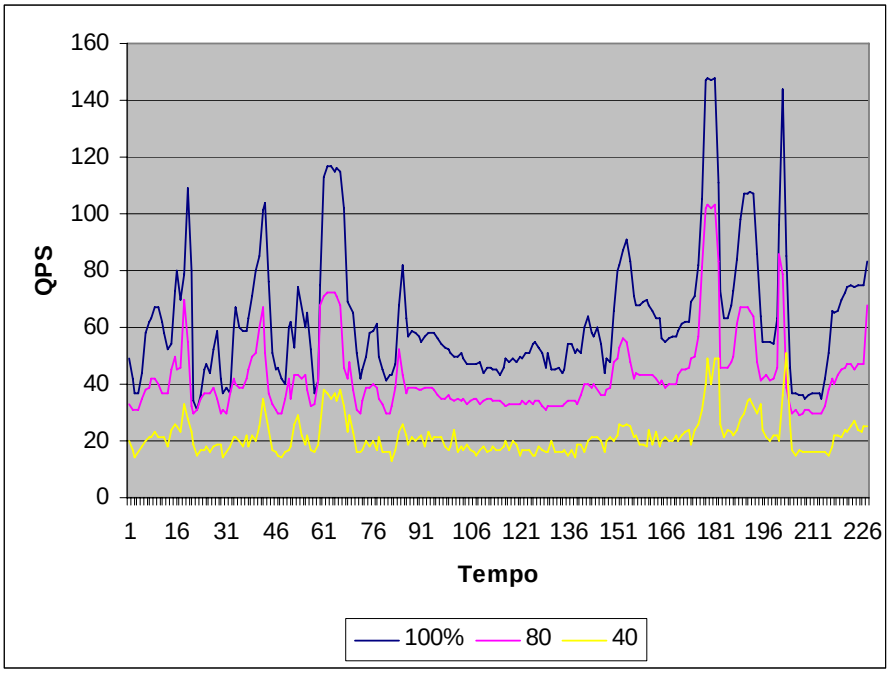

Gráfico 8 Resultado da variação da CPU na performance

Nota-se que a CPU influencia diretamente na renderização, a diferença entre duas determinadas séries permanecem, em sua maior parte, constante. Analisando uma série em particular percebe-se que mesmo sem a flutuação de recursos, a performance da aplicação varia. Em cenas mais simples, picos do gráfico, o desempenho é maior em todas as séries, em cenas mais complexas, a taxa é relativamente baixa. Analisando todas as séries chega-se a conclusão que quanto menor CPU disponível para uso, menor desempenho será alcançado.

Para termos uma noção de quando uma adaptação será acionada, tendo em vista somente a flutuação de performance da aplicação, gráficos de dispersão foram construídos. O Gráfico 9 ilustra a dispersão dos dados da série executada com 100% do processador em relação à freqüência em que ocorre uma determinada exibição em um intervalo de quadros por segundo. As barras indicam que a exibição de uma taxa de quadros entre um intervalo [X,X+5] ocorreu Y% do tempo.

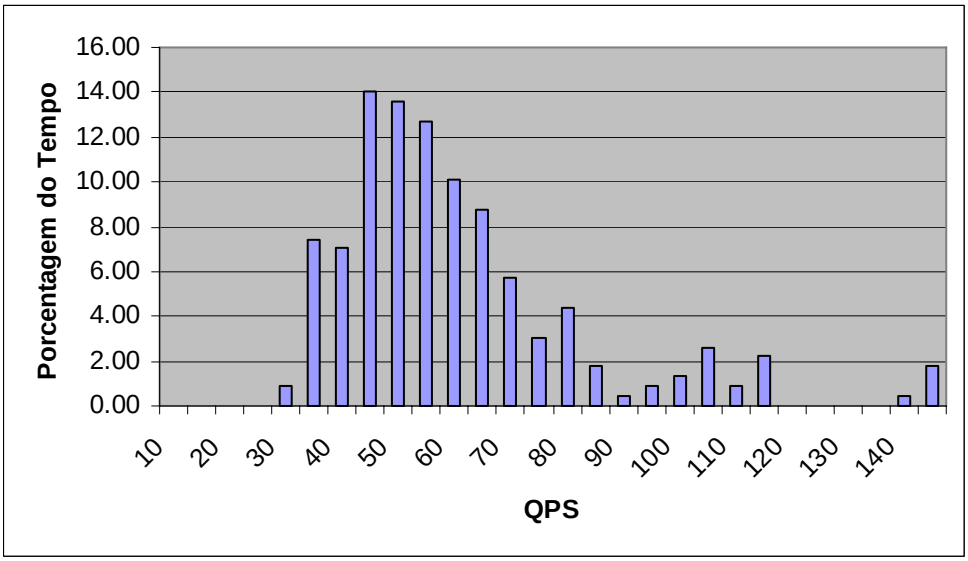

Gráfico 9 Dispersão da taxa de QPS utilizando 100% da CPU

A adaptação será ditada pelo usuário, caso ele deseje a performance em seu extremo, uma taxa de 60 quadros por segundo deve ser assegurada, o que implica que em 55.7% do tempo haverá adaptação. Para conseguir este valor somaram-se todas as porcentagens dos intervalos que apresentaram exibição menor do que 60. Este valor justifica que não somente a flutuação dos recursos deve ser considerada, mas também a variação da própria aplicação. Caso o intervalo usado para a decisão da ativação de uma determinada adaptação for dez QPS, três mecanismos de adaptação deverão ser estipulados, cada um para tratar os seguintes intervalos: [30,40[; [40,50[; [50,60[. A adaptação acionada quando a taxa de quadros por segundo se estabelecer nos intervalos acima acontecerá por certa quantidade

de tempo, como descrita na Tabela 12. Mesmo que a adaptação executada para atender o intervalo [30,40[acontecer em apenas 8,33 % do tempo, não significa que em toda aplicação 3D este valor será válido. Mesmo que o ambiente de simulação seja projetado para reproduzir um comportamento de um usuário, não significa que todo usuário se comportará desta maneira, o tempo pode ser diferente de 8,33 %. Porém, a curva será bem semelhante à apresentada no Gráfico 9, ou seja, uma pirâmide com o lado esquerdo menor do que o direito.

Tabela 12 Intervalo de possibilidade de adaptação

Intervalo	Porcentagem do Tempo
[30.40[	8.33
[40,50[	21.05
[50,60[	26.32

O Gráfico 10 ilustra a dispersão dos dados extraídos da série com “80 %” da CPU disponível. Seguindo o mesmo raciocínio do gráfico anterior conclui-se que haverá menos intervalos para o tratamento da adaptação caso seja escolhido um valor mediano pelo o usuário, como 40, pois a quantidade de intervalos diminuiu, apesar da distribuição dos valores ser diferente, a forma da curva é semelhante.

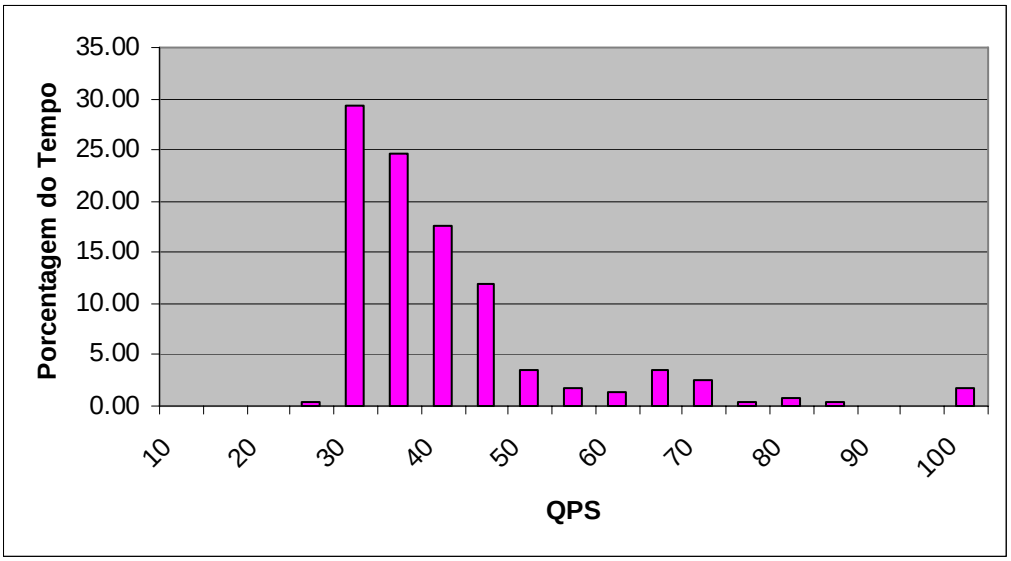

Gráfico 10 Dispersão da taxa de QPS utilizando 80% da CPU

Por último, a análise do Gráfico 11, dispersão dos valores da série com 20% de CPU ocupada, mostra que quanto menos CPU, menor a freqüência de adaptação. Os gráficos 19,20 e 21 possuem, respectivamente, 20, 14 e 9 intervalos de dados, caso fossem feitos mais testes com menor ocupação de CPU, teríamos uma quantidade ainda menor de intervalos. Uma menor freqüência em adaptações indica que plataformas com menor poder de

processamento terão menor possibilidades de adaptações, além de a qualidade apresentada ser bem menor da vislumbrada em dispositivos com maior capacidade.

Uma menor freqüência em adaptações não justifica o fato da necessidade de adaptações. Em contrapartida quanto menor o processamento, por mais tempo a cena é renderizada com baixos quadros por segundo, comprovado pelo deslocamento da curva de dispersão para a esquerda, do gráfico 18 para o gráfico 20 e do 20 para o 21. No Gráfico 11 em 89.47 % do tempo a cena é renderizada abaixo dos 30 QPS, ou seja, abaixo do ideal para um jogo, daí a verdadeira necessidade da adaptação, isto significa que em quase 90% do tempo haverá adaptação.

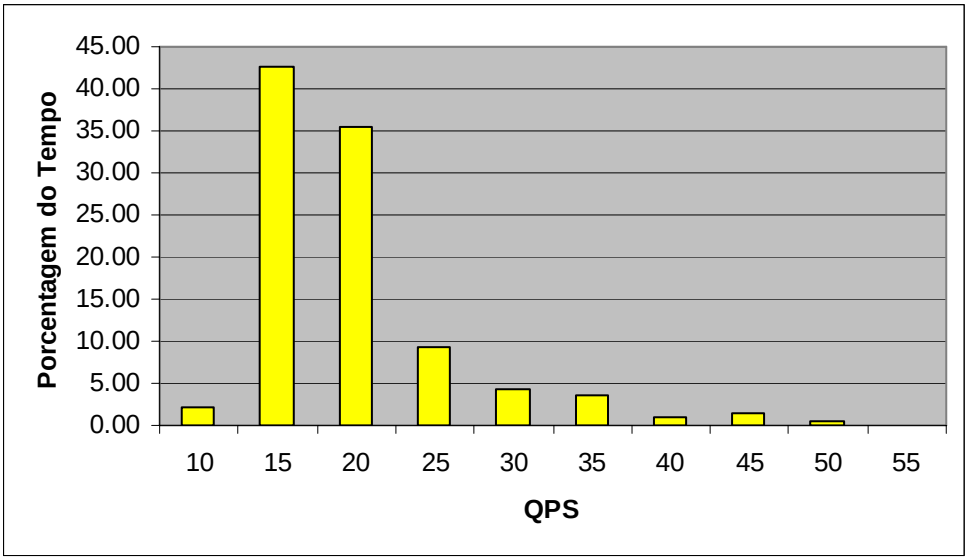

Gráfico 11 Dispersão da taxa de QPS utilizando 20% da CPU

Os testes realizados nesta seção foram realizados apenas para comprovação prática dos valores utilizados como parâmetros de configuração dos níveis a serem monitorados pelo MR.