UNIVERSIDADE FEDERAL DA PARAÍBA
DEPARTAMENTO DE INFORMÁTICA
CENTRO DE CIÊNCIAS EXATAS E DA NATUREZA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Daniel Faustino Lacerda de Souza

INCORPORANDO TECNOLOGIAS 3D A TV DIGITAL E INTERATIVA: UM ESTUDO DE ESTRATÉGIAS DE INTEGRAÇÃO BASEADAS NO *MIDDLEWARE* GINGA

JOÃO PESSOA
2010

Daniel Faustino Lacerda de Souza

INCORPORANDO TECNOLOGIAS 3D A TV DIGITAL E INTERATIVA: UM ESTUDO DE ESTRATÉGIAS DE INTEGRAÇÃO BASEADAS NO *MIDDLEWARE* GINGA

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Ciências da Computação da Universidade Federal da Paraíba como requisito para a obtenção do Título de Mestre em Informática.

Orientadoras: Liliane dos Santos Machado
Tatiana Aires Tavares

JOÃO PESSOA
2010

Ata da Sessão Pública de Defesa de Dissertação de Mestrado do DANIEL FAUSTINO LACERDA DE SOUZA, candidato ao Título de Mestre em Informática na Área de Sistemas de Computação, realizada em 05 de julho de 2010.

Aos cinco dias do mês de julho do ano dois mil e dez, às nove horas, na Sala de Reunião do Centro de Ciências Exatas e da Natureza da Universidade Federal da Paraíba, reuniram-se os membros da Banca Examinadora constituída para examinar o candidato ao grau de Mestre em Informática, na área de "*Sistemas de Computação*", na linha de pesquisa "*Processamento de Sinais e Sistemas Gráficos*", o Sr. Daniel Faustino Lacerda de Souza. A comissão examinadora composta pelos professores doutores: Tatiana Aires Tavares (DI - UFPB), Primeiro Orientador e Presidente da Banca Examinadora, (DI-UFPB, Liliane dos Santos Machado, Segundo Orientador, (DI-UFPB), Guido Lemos de Souza Filho (DI-UFPB), como examinador interno e Luiz Marcos Garcia Gonçalves (UFRN) como examinador externo. Dando início aos trabalhos, a Profª. Tatiana , cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse, oralmente, a exposição do trabalho de dissertação intitulado "*INCORPORANDO TECNOLOGIAS 3D a TV DIGITAL E INTERATIVA: UM ESTUDO DE ESTRATÉGIAS DE INTEGRAÇÃO BASEADOS NO MIDDLEWARE GINGA*". Concluída a exposição, o candidato foi argüido pela Banca Examinadora que emitiu o seguinte parecer: "aprovado" Assim sendo, deve a Universidade Federal da Paraíba expedir o respectivo diploma de Mestre em Informática na forma da lei e, para constar, o professor Lucídio dos Anjos Formiga Cabral, Sr. Vice-Coordenador do PPGI, lavrou a presente ata, que vai assinada por ele, e pelos membros da Banca Examinadora. João Pessoa, 05 de julho de 2010.

Lucídio dos Anjos Formiga Cabral

Profª. Dra. Tatiana Aires Tavares
Primeiro Orientador (DI-UFPB)

Profª. Drª.Liliane dos Santos Machado
Segundo Orientador (DI-UFPB)

Prof. Dr. Guido Lemos de Souza Filho
Examinador Interno (DI-UFPB)

Prof. Dr. Luiz Marcos Garcia Gonçalves
Examinador Externo (UFRN)

AGRADECIMENTOS

Pensei que chegando a este ponto de minha jornada o mais fácil estaria por vir: os Agradecimentos. Somente agora, me preparando para escrever este tópico é que percebo tamanha responsabilidade e o cuidado que se faz necessário ao escolher estas palavras. Cuidado este não relacionado a quem mencionar, mas sim a quem não mencionar dado o vasto número de pessoas que me apoiaram nestes dois anos. Por esta razão, tomando a devida precaução de não cometer injustiça com ninguém gostaria de agradecer:

Em primeiro lugar, a Deus pelo dom da vida, inspiração e amparo concedido nos dias de atribulação e tristeza.

Em especial, os meus pais, José Geraldo de Souza e Idalva Lacerda de Souza, por toda força e motivação nos momentos difíceis. Pelo carinho concedido e pelo belo exemplo de família que me foi passado ao longo de minha vida.

As minhas orientadoras, Liliane dos Santos Machado e Tatiana Aires Tavares, pela paciência, amizade e orientação na minha caminhada em busca do saber. À professora Liliane, agradeço ainda, por ter confiado em mim quando era apenas um aluno em início do curso, me oferecendo a oportunidade de ingressar na vida de pesquisador.

Aos meus amigos de infância, em especial Táles, por sempre estarem presentes em minha vida, oferecendo seu apoio e sua confiança em quaisquer assuntos de ordem pessoal.

A todos os meus amigos do Laboratório de Tecnologia para o Ensino Virtual e Estatístico e Laboratório de Vídeo Digital, em especial ao professor Ronei e colegas de mestrado Alysson, Bruno e Alana, por me suportarem e me apoiarem no que foi preciso em todos os dias de trabalho.

Agradeço também a todos os professores, por todo apoio e ajuda nesta caminhada, me auxiliando através de sugestões, ensinamentos e companheirismo. Agradeço em especial ao professor Guido, por compartilhar de seu conhecimento na área de minha pesquisa, auxiliando no desenvolvimento das idéias e teorias abordadas neste trabalho.

Muito obrigado a todos!

RESUMO

Recentemente o Brasil definiu seu padrão de TV Digital. Uma das novidades neste novo modelo é a digitalização do conteúdo transmitido pelas emissoras, que permite a convergência de aplicações e serviços de internet em serviços interativos para TV. Neste sentido, o *middleware* para TV é o software responsável por oferecer suporte à execução de tais aplicativos. A integração de tecnologias 3D a ambientes de TV Digital estende as possibilidades de interatividade e entretenimento. Através do uso deste tipo de tecnologia é possível ampliar o conjunto de possibilidades de pesquisa e negócios no âmbito da TV Interativa.

Este trabalho apresenta uma arquitetura de suporte à execução de aplicações tridimensionais em ambientes de TVDI. Tal arquitetura é baseada no *middleware* brasileiro de TVDI: Ginga. Neste trabalho são discutidas as estratégias de integração de tecnologias 3D através dos ambientes de execução e apresentação do Ginga. Além da definição da arquitetura, este trabalho apresenta uma prova de conceito para o novo modelo apresentando, bem como uma análise de desempenho das soluções propostas. Por fim, é feita um estudo comparativo entre as soluções desenvolvidas, a fim de definir qual estratégia é mais viável.

Palavras Chave: Televisão Digital, Tecnologias 3D, Ginga

ABSTRACT

Recently, Brazil has defined its Digital Television standard. One of news in this model is the digitization of content transmitted by the broadcasters, which enables the convergence of applications and Internet services in interactive services for TV. In this sense, middleware for TV is the software which supports the implementation of such applications. The integration of 3D technologies in Digital Television environments extends the possibilities of interactivity and entertainment. Thus, the use of this technology can expand the range of research and business opportunities within the Interactive TV area.

This work presents an architecture to support the execution of 3D applications on Digital Television environments. This architecture is based on DTV Brazilian middleware: Ginga. In this work are discussed strategies for integration of 3D technologies through execution and presentation environments of Ginga. Besides the definition of the architecture, this work presents a proof of concept for the new model presented as well as a performance analysis of the proposed solutions. Finally, a comparison is made between the solutions developed in order to determine which strategy is more feasible.

Keywords: Digital Television, 3D Technologies, Ginga

Sumário

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

1.1 APRESENTAÇÃO

A Televisão Digital Interativa (TVDI) possibilita a combinação do estilo tradicional de assistir TV com a possibilidade de interação que atualmente é provida por computadores pessoas e Internet [MAAD, 2003a]. Para LÉVY (2005) a interatividade é a participação ativa do beneficiário em uma transação de informação. Desta forma, Televisão Digital Interativa (TVDI) pode ser entendida como aquela que possibilita ao receptor transformar as mensagens e não simplesmente recebê-las passivamente, de forma que o usuário torna-se um co-autor desta mensagem [TONIETO, 2006].

A digitalização dos meios de comunicação agiu diretamente sobre a cadeia de entrega de produções de TV. Esta digitalização possibilita a convergência de aplicações que são transmitidas digitalmente via *broadcast* e os serviços de internet em serviços multimídia interativos que, por sua vez, através da audiência da TV atingem uma parcela significativa da população [MAAD, 2003b]. Com o passar dos anos, novos formatos, gêneros e formas de conteúdo vêm surgindo [JENSEN, 2005]. Notícias personalizadas, guias de programação, jogos interativos, bem como aplicações até então existentes apenas na Internet como serviços de e-mail, *homebanking*, *homeshopping* e Educação a Distância (t-*learning*) são exemplos de aplicações que demonstram esta evolução.

Ao mesmo tempo em que novos paradigmas de interação surgem no âmbito da TVDI, novas tecnologias para a construção de aplicações tridimensionais também avançam no sentido de integração com sistemas embarcados. Estas tecnologias são conjuntos de padrões e API's (*Application Programming Interface*) que tem por objetivo oferecer suporte a construção de aplicações tridimensionais diversas. Atualmente tais tecnologias vêm avançando de forma a diminuir o alto custo computacional demandado por aplicações 3D. Tais avanços possibilitam a utilização de técnicas desta natureza dentro de um ambiente de TVDI, fornecendo um conjunto novo de possibilidades e idéias no que diz respeito à interatividade [OLAIZOLA et al., 2006].

Este trabalho é uma proposta para o suporte e execução de aplicações tridimensionais em um ambiente de TVDI. Para tanto, discutimos e analisamos tecnologias 3D e a viabilidade de incorporação de tais tecnologias a ambientes de TV Digital, bem como procuramos avaliar como estas tecnologias poderão corroborar na construção de aplicações interativas e intuitivas.

1.2 MOTIVAÇÃO

A interação entre humanos e computadores, através de algum meio de interfaceamento, está presente em qualquer sistema computacional, desde o mais simplista, como uma calculadora eletrônica, até os mais sofisticados, como a operação de grandes maquinários industriais. Em um sistema gráfico, as questões de interface são muito importantes e no caso de programas interativos a interface gráfica é essencial [VELHO e GOMES, 2001]. A utilização de tecnologias 3D para a construção de tais programas interativos tem sua eficácia comprovada nestes seus 30 anos de estudo e utilização [BRUTZMAN e DALY, 2007].

Diversos esforços são feitos para incorporar tecnologias 3D em diversas áreas, como indústria, educação, arquitetura e mais recentemente na web [BRUTZMAN e DALY, 2007]. Outro fator interessante é a incorporação destas tecnologias a sistemas embarcados, mais especificamente dispositivos móveis. Hoje em dia existem dezenas de dispositivos móveis que incorporam placas gráficas e/ou possuem poder de processamento capaz de trabalhar com ambientes 3D [FADI et al., 2005]. As empresas que possuem este tipo de dispositivos exploram principalmente as áreas de interfaces 3D e jogos. A construção de jogos em particular é uma das áreas que mais foi beneficiada pelo desenvolvimento das tecnologias 3D. Muitas das técnicas de iluminação, sombreamento e texturização surgiram com o propósito de ampliar a qualidade gráfica de certos jogos. Um fator interessante neste sentido é o desenvolvimento dos chamados Jogos Sérios (*Serious Games*), jogos que tem como finalidade colaborar com a formação educacional do indivíduo [MICHAEL e CHEN, 2006].

A convergência digital é o termo utilizado para designar a tendência de utilizar uma mesma infra-estrutura tecnológica com o objetivo de prover diferentes serviços, que anteriormente requeriam canais de comunicação e padrões diferentes. Esta tendência mostra-

se como outro fator motivante, uma vez que, a TV Digital é uma das candidatas a participar desta convergência como uma das tecnologias de comunicação moderna.

A grande importância que é dada hoje em dia às tecnologias 3D nos diversos campos, bem como, o seu uso como ferramenta que amplia a capacidade de interação dos indivíduos com aplicações, aliada ainda à tendência de convergência que as tecnologias modernas estão inseridas e a necessidade que a TVDI possui de ampliar e explorar seu potencial no que diz respeito aos meios de interagir com o usuário, são os principais fatores motivantes que nos levam a perceber a necessidade imediata de integrar tais tecnologias.

1.3 RELEVÂNCIA

Atualmente, o Brasil passa por um período de transição entre um sistema de TV analógica para o Sistema Brasileiro de TV Digital (ISDB-Tb). Com esta transição surgem problemas e estudos em diversas áreas que tendem a colaborar com esta implantação. As áreas de infra-estrutura de transmissão e recepção, sistemas computacionais para gerenciamento do *hardware*, ambiente de produção nas emissoras, modelo de negócio e conteúdo interativo são áreas onde tais problemáticas e estudos estão sendo discutidos.

Uma agravante é a diversidade socioeconômica brasileira, que torna a transição do antigo modelo analógico para um digital mais complexa do que em outros países. Dentro desse contexto, é relevante considerar algumas questões pertinentes:

- Como oferecer uma infra-estrutura tecnológica homogênea em um país de distâncias continentais?
- Como oferecer conteúdo e promover a inclusão social para milhares de telespectadores de diferentes perfis, acessos e classes sociais?
- Como utilizar esta ferramenta de forma sustentável, favorecendo o ensino não presencial?

A questão do favorecimento do ensino não-presencial foi, inclusive, uma das principais áreas de interesse do governo quando instituiu o Sistema Brasileiro de Televisão Digital [BRASIL, 2006]. Neste sentido a integração de tecnologias 3D ao SBTVD pode ser

vista como um fator relevante, uma vez que, tais tecnologias podem ser utilizadas como poderosas ferramentas para a concepção e desenvolvimento de aplicações que tenham como propósito a inclusão social. Ambientes virtuais para discussão, jogos educacionais e sistemas de apoio ao telespectador são temas que se utilizam destas tecnologias para oferecer meios mais intuitivos de se interagir com o usuário. Existem ainda fatores relacionados ao próprio padrão como a transmissão e apresentação de conteúdo em formato de LIBRAS (Linguagem Brasileira de Sinais). A adoção de tal tecnologia pode ajudar na definição de formatos para tratamento deste tipo de conteúdo dentro do ambiente de TV.

Outro fator relevante a ser citado está relacionado ao fato de nenhum dos atuais padrões mundiais suportarem a execução de aplicações tridimensionais diretamente. O que existe atualmente são alguns estudos para a inclusão deste tipo de suporte no padrão europeu de televisão digital (DVB) [CESAR, 2005] [PULLES e SASNO, 2004]. Como o ISDB-Tb é relativamente novo e ainda está em processo de padronização existe a possibilidade de inclusão de suporte 3D a este padrão.

1.4 OBJETIVOS

Como objetivo geral deste trabalho pretende-se desenvolver um estudo exploratório de tecnologias 3D e como estas tecnologias podem ser incorporadas em um ambiente de TVDI. Como padrão de referência para o estudo será utilizado o ISDB-Tb (Sistema Brasileiro de TV Digital) [BRASIL, 2006]. O Ginga, *middleware*, através de sua API de inovação é utilizado como ponto de entrada para a avaliação de viabilidade e integração desta tecnologia ao sistema brasileiro.

Pretende-se como objetivos específicos pesquisar conteúdo teórico e prático referente a cada uma das etapas para se alcançar o objetivo geral. Assim, dividimos os objetivos específicos da seguinte forma:

- Estudar das tecnologias 3D potencialmente candidatas à integração com o *middleware* Ginga. Dentre estas tecnologias se destacam JAVA3D [SELMAN, 2002], openGL ES [KHRONOS, 2009], LWJGL [RYCHLIK-PRINCE et al., 2009] e M3G [SUN, 2009] como APIs para geração de ambientes 3D, além do

X3D [BRUTZMAN e DALY, 2007] e O3D [GOOGLE, 2009] como padrões para definição e mapeamento destes ambientes virtuais.

- Estudar *middleware* para TVDI. Pesquisar informações sobre os *middleware* existentes, analisar suas propostas, peculiaridades e capacidade de adaptação a proposta apresentada. Analisar em particular o *middleware* Ginga que compõe o ISBD-Tb.

- Desenvolver uma estratégia que permita a integração e execução de aplicações 3D no Ginga.

- Estudo de Caso: Desenvolver aplicações piloto, com o objetivo de validar a solução construída, bem como apontar direções para exploração destas tecnologias como forma de favorecer a inclusão digital e ensino não presencial.

1.5 CONTRIBUIÇÕES

Este trabalho traz como contribuição a especificação de uma arquitetura que provê o suporte à execução de aplicações tridimensionais a um ambiente de TVDI. Em particular possibilita a execução deste tipo de aplicações no Sistema Brasileiro de Televisão Digital. Com este suporte pretendemos oferecer mais um diferencial entre o ISDB-Tb e outros sistemas de televisão digital, visto que, nenhum outro padrão mundial possibilita a exploração deste tipo de tecnologia diretamente. O referencial teórico e os sistemas desenvolvidos como parte deste trabalho também são uma contribuição relevante para futuras pesquisas nesta área.

No Brasil a instituição do ISDB-Tb possui como principal apelo a inclusão social das camadas mais pobres da população. Desta forma, a integração de um ambiente de TVDI com tecnologias 3D pode contribuir de forma a fomentar soluções para os desafios relacionados à inclusão social através de aplicações mais sofisticadas e que possam favorecer o aprendizado e a disseminação de informação. Jogos educacionais, ambientes virtuais de aprendizagem, exploração de museus virtuais e campi virtuais são exemplos de aplicações que poderiam ser desenvolvidas para este tipo de ambiente.

Outra contribuição importante está relacionada à convergência digital de tecnologias de forma que seja possível desenvolver aplicações tridimensionais que passem a ser executadas, por exemplo, através da plataforma PC, celulares e também através da TV. A construção de ambientes virtuais adaptativos é um exemplo de aplicação que poderia ser explorado através da convergência destas tecnologias. Aplicações tridimensionais multiplataforma poderiam, agora, ser desenvolvidas também para um ambiente de TVDI.

1.6 CENÁRIO DE PESQUISAS CORRELATAS NA INSTITUIÇÃO

Este trabalho se insere no contexto das pesquisas desenvolvidas pelo Laboratório de Tecnologias para o Ensino Virtual e Estatístico (LabTEVE) e pelo Laboratório de Vídeo Digital (LAVID), ambos da Universidade Federal da Paraíba (UFPB). O LabTEVE tem como foco as pesquisas nas áreas de jogos eletrônicos e simuladores para ensino e aprendizagem médica baseados em Realidade Virtual (RV). O LAVID, por sua vez, trabalha na pesquisa e desenvolvimento de tecnologias para TV Digital e Interativa (TVDI), bem como serviços de distribuição vídeo e outras áreas correlatas a estas.

No contexto do LabTEVE, trabalhos relacionados à presente pesquisa se inserem no campo de jogos eletrônicos, especificamente na área de jogos 3D para sistemas embarcados. O trabalho de Virgínio Filho (2009) trata sobre o desenvolvimento de jogos tridimensionais para sistemas embarcados, com foco em dispositivos móveis. Neste trabalho o autor analisa as técnicas e ferramentas existentes atualmente para esta área, além de desenvolver um jogo educacional 3D para celulares. Outra pesquisa correlata é o CyberMed [MACHADO, 2009], um conjunto de bibliotecas para o apoio no desenvolvimento de aplicações baseadas em Realidade Virtual. Uma das bibliotecas contidas no CyberMed é justamente um módulo para desenvolvimento de ambientes tridimensionais. Baseado nestes estudos o laboratório desenvolve outras pesquisas relacionadas as áreas de RV, computação gráfica e jogos eletrônicos para diversas plataformas.

O LAVID possui experiência na área de tecnologias para TVDI. Os trabalhos de Soares [SOARES, 2006] e Leite [LEITE et al., 2005] tiveram como objetivo desenvolver plataformas de referência para o *middleware* procedural e declarativo, respectivamente, do

padrão brasileiro de TV Digital. O OpenGinga[1], software livre mantido pelo LAVID, tem como objetivo o desenvolvimento de uma implementação de referência do *middleware* do padrão brasileiro de TV (*middleware* Ginga) Por fim, o LAVID é um dos laboratórios que coordena o GingaCDN[2] (*Ginga Code Development Network*), projeto que tem como principal objetivo explorar a pesquisa e desenvolvimento de ferramentas que tornem possível a criação, gerenciamento e operação de uma rede de desenvolvedores de código para o *middleware* Ginga.

[1] OpenGinga - http://www.openginga.org/

[2] GingaCDN - http://www.gingacdn.lavid.ufpb.br/

2 ESTADO DA ARTE EM APLICAÇÕES DE TVDI UTILIZANDO TECNOLOGIAS 3D

Neste capítulo apresentamos as tecnologias 3D atuais, bem como as soluções específicas para sistemas embarcados. Outro ponto discutido são os conceitos básicos sobre TV Digital Interativa, fundamentais para o entendimento do trabalho ora proposto. Também ressaltamos as abordagens existentes para *middleware* nos sistemas de TV Digital. Por fim, serão discutidos os trabalhos relacionados ao objeto desta pesquisa.

2.1 TECNOLOGIAS 3D

A utilização de sistemas gráficos nos oferece o meio mais natural de comunicação com sistemas computacionais, visto que, a grande capacidade para reconhecimento de padrões 2D e 3D permite aos seres humanos perceber e interpretar dados de imagens com rapidez e eficiência [FOLEY et al., 1997]. Ao longo dos anos uma série de pesquisas na área de computação gráfica foram desenvolvidas e, como resultado, observa-se hoje o surgimento de uma série de técnicas e dispositivos que permitem interagir com aplicações tridimensionais em tempo real [MÖLLER e HAINES, 2002] [WATT, 2000]. Em paralelo com o surgimento destas técnicas surgiram também várias ferramentas e APIs que permitem o desenvolvimento de sistemas tridimensionais [MCREYNOLDS e BLYTHE, 2005]. Este conjunto de técnicas, ferramentas e APIs gráficas é chamado, neste trabalho, de tecnologias 3D.

A utilização de tecnologias 3D está presente em todas as partes. Desde os mais simples jogos eletrônicos até o projeto dos mais modernos equipamentos para viagens espaciais, passando também pela publicidade, com as vinhetas eletrônicas, e pela medicina, onde a criação de imagens de órgãos internos ao corpo humano possibilita o diagnóstico que em outros tempos somente seria possível com intervenções cirúrgicas [VELHO e GOMES, 2001].

Dois fatores que, ao longo do tempo, contribuíram para o surgimento e crescimento das tecnologias 3D foram [FOLEY et al., 1997]:

- O desenvolvimento da tecnologia de circuitos integrados que permitiu o barateamento e a consequente popularização das máquinas;

- A popularização de certos aplicativos (planilhas, editores de texto, editores gráficos, processadores de imagem, bancos de dados, etc) permitiram a popularização destas tecnologias na medida em que possibilitaram que o usuário comum sem conhecimento ou tempo para desenvolver aplicativos gráficos pudessem se utilizar das facilidades da mesma.

2.1.1 Áreas de Aplicação

O avanço nas pesquisas relacionadas a tecnologias 3D tornou a computação gráfica uma poderosa ferramenta para o desenvolvimento rápido e econômico de sistemas gráficos. Atualmente, quase não existem áreas onde não podemos imaginar algum uso vantajoso deste tipo de tecnologia. Hoje em dia encontramos Computação Gráfica em diversas áreas como ambientes de realidade virtual, visualização de dados, educação e treinamento, entretenimento, etc [HEAR e BAKER, 2004], conforme detalhado nas subseções seguintes.

2.1.1.1 Ambientes de Realidade Virtual

Uma das áreas onde a computação gráfica e as tecnologias 3D são aplicadas é a área de aplicações baseadas em Realidade Virtual. Neste tipo de aplicação o usuário pode interagir com objetos tridimensionais, utilizando dispositivos específicos desenvolvidos para navegação em ambientes tridimensionais [BURDEA e COIFFET, 2003]. As simulações em ambientes de realidade virtual são geralmente utilizadas para treinamento e entretenimento.

Alguns exemplos de aplicações nesta área são os simuladores para treinamento médico. Estes simuladores têm por objetivo oferecer uma forma de estudantes e profissionais da área médica simularem procedimentos que, em no mundo real são de difícil execução, por escassez dos componentes necessários a simulação, por exemplo. Na Figura 1 podemos observar um simulador para coleta de medula óssea em crianças desenvolvido por MACHADO (2003). Este simulador utiliza os conceitos de Realidade Virtual associado a tecnologias 3D para construir um ambiente virtual de treinamento.

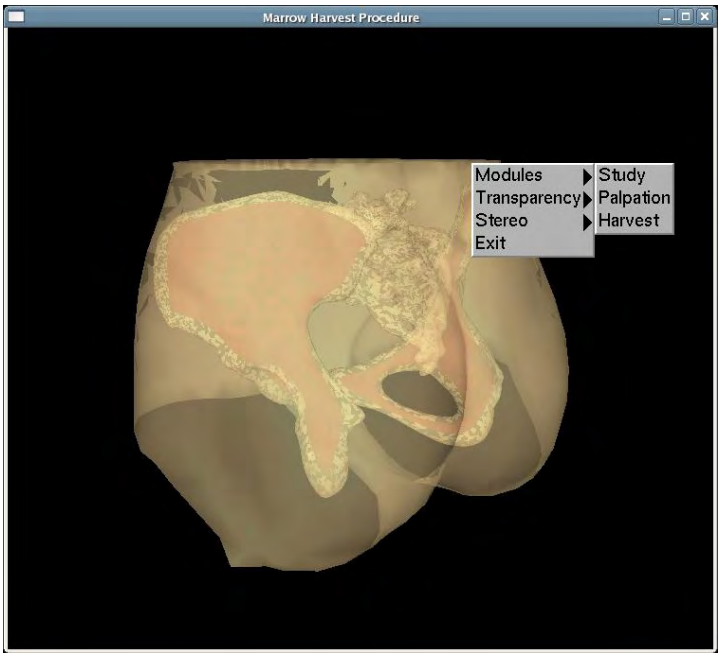


Figura 1. Simulador para a Coleta de Medula Óssea [MACHADO, 2003].

2.1.1.2 Visualização de dados

Na área de visualização de dados a computação gráfica é utilizada de forma a produzir a representação gráfica de dados científicos, de engenharia, medicina, etc. Existem diferentes conjuntos de dados e uma descrição efetiva destes dados de forma gráfica depende das características destes dados. Algumas técnicas são utilizadas também na análise e entendimento destes dados, principalmente quando estamos trabalhando com funções matemáticas complexas.

Um exemplo de uma aplicação desta natureza é o sistema *EducaView* que tem por objetivo utilizar técnicas de computação gráfica para auxiliar no processo de ensino e aprendizagem de classificação de imagens [DIAS, 2009]. Neste sistema as imagens são classificadas e o volume de dados gerado pelo simulador apresentado em um ambiente tridimensional, que facilita o entendimento por parte do aluno (ver figura 2).

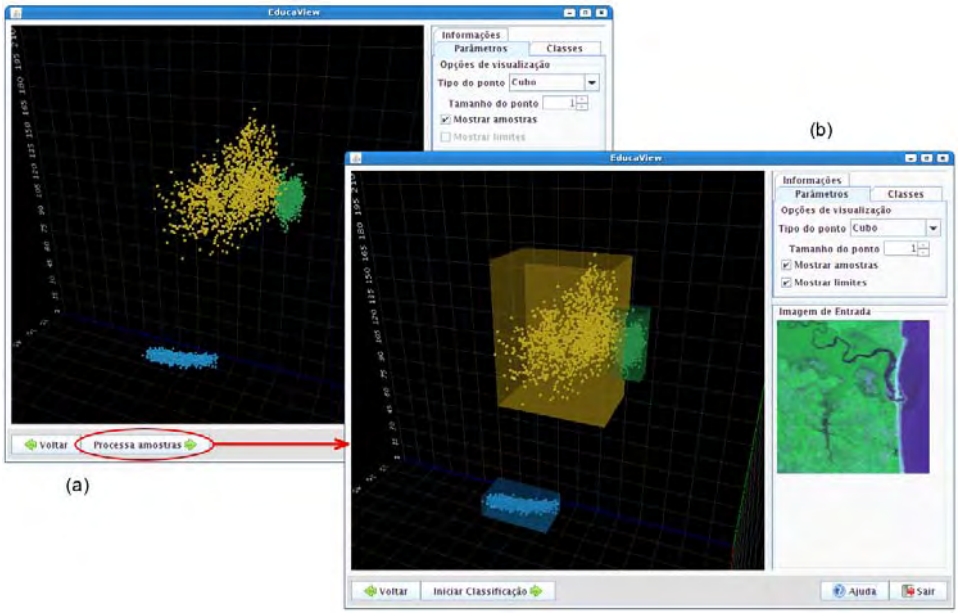


Figura 2. Sistema de visualização de dados a partir de classificação de imagens (EducaView) [DIAS, 2009].

2.1.1.3 Educação e Treinamento

Modelos físicos, financeiros, econômicos, médicos são utilizados com frequência com o objetivo de auxiliar no ensino e treinamento. Alguns exemplos de aplicações para esta área são os simuladores para navegação aérea, simuladores para treinamento médico, aplicações para simulação de eventos físicos, etc. Em alguns casos *hardware* específico é desenvolvido para o propósito da simulação. Um exemplo de aplicação nesta área é o *CyberMed*: uma ferramenta para o desenvolvimento de aplicações baseadas em Realidade Virtual para treinamento médico [MACHADO, 2009].

2.1.1.4 Entretenimento

Esta é uma das áreas que mais se beneficiou com os avanços da computação gráfica. Produções de televisão, filmes, clips de músicas são alguns dos programas de entretenimento que utilizam tecnologias 3D para criarem efeitos especiais ou melhorarem a aparência dos personagens, por exemplo. Estes efeitos especiais se traduzem em cenários completos, cidades antigas que são recriadas, ações humanamente impossíveis e que com o advento da computação gráfica puderam estar presentes nas produções de cinematográficas.

Uma das áreas do entretenimento que mais utiliza de recursos gráficos é sem dúvida a área de jogos. Jogos são a maior aplicação da computação gráfica, e a grande motivação para seu desenvolvimento. Os jogos eletrônicos vêm, desde o seu surgimento, experimentando grandes avanços. Atualmente o atrativo financeiro desta área tem sido o fator motivante para a sua evolução no que diz respeito ao desenvolvimento de novos estilos e novas técnicas computacionais. [ALVES et al., 2004].

2.1.1.5 Outras áreas

Além das áreas citadas acima, existem uma diversidade de outras áreas onde a computação gráfica atua. Algumas delas mais antigas como no caso da área de entretenimento e em outras de forma mais recente, como é o caso da Internet e da TV. Outros exemplos de áreas onde estas tecnologias são empregadas são:

- Artes: Para expressão artística utilizando os ambientes gráfico-computacionais como meio ou fim, tais como gravura digital, arte digital, web arte.
- Arquitetura e Design de Produto: desenvolvimento gráfico dos projetos de forma visual e com a aplicação dos cálculos matemáticos para os testes dos projetos quanto a resistência, a variação de luz e ambientes.
- Engenharia: simulação de todas as espécies de eventos físicos, químicos dos materiais envolvidos nos projetos em elaboração.
- Geoprocessamento: Para geração de dados relacionados à cidades, regiões e países.
- Design Visual: para o desenvolvimento de mídias visuais, desde a impressa (como propagandas em revistas e outdoors) quanto para o auxílio cinematográfico dos comerciais televisivos.

Atualmente outra grande área onde este tipo de tecnologia está sendo empregada é a *Web*. O consórcio WEB 3D [WEB3D, 2009], é uma iniciativa para o desenvolvimento de tecnologias 3D que possam dar suporte a criação e acesso de ambientes 3D através da *internet*, de forma prática e rápida. Através destas tecnologias pretende-se integrar uma série de serviços através da rede, oferecendo suporte e acesso para qualquer usuário que deseje

navegar. Uma das iniciativas que vem crescendo é o X3D Earth [WEB3D, 2009], um projeto baseado na tecnologia X3D [BRUTZMAN e DALY, 2007] e que tem como objetivo criar uma comunidade virtual colaborativa, possibilitando que cada usuário integre seus serviços a este sistema virtual e possa colaborar para a construção deste ambiente virtual.

2.1.2 Tecnologias 3D e Sistemas Embarcados

Recentemente, estamos observando um aumento no desempenho dos dispositivos com sistemas embarcados. Este fato é resultado da evolução do *hardware* e *software* que compõe estes dispositivos. Os dispositivos com sistemas embarcados, em particular os dispositivos moveis estão evoluindo de tal modo, que estão suportando aplicações cada vez mais complexas [NADALUTTI et al., 2006]. Contudo a renderização de ambientes tridimensionais neste tipo de dispositivo ainda é considerada uma tarefa difícil. Algumas das características que implicam em sérias limitações para estes tipo de sistema são:

- CPUs limitadas;
- Baixa capacidade de armazenamento das memórias;
- Ausência ou performance limitada de aceleradores gráficos;
- Ausência ou performance limitada de unidade de processamento de ponto flutuante;
- Baixa capacidade para consumo de energia; e
- Ausência de ambientes sofisticados para o desenvolvimento e depuração de aplicações.

Várias empresas de microprocessadores começaram a desenvolver linhas de produtos que viabilizam a renderização de gráficos 3D em dispositivos móveis. A *NVIDIA*®, por exemplo, desenvolveu a placa de vídeo *GoForce™ 3D* que oferece funcionalidades para o desenvolvimento de jogos 3D para dispositivos moveis [NVIDIA, 2009]. Além da *NVIDIA*® outras empresas da área estão lançando seus produtos, como é o caso da ARM com seu produto MBX 3D ou a ATI™ com o seu processador IMAGEON [ATI, 2009][ARM, 2003]

Na área de Televisão Digital os primeiros avanços no sentido de integrarem *hardware* específico para renderização de gráficos 3D partiu da AMD com o lançamento do processador *AMD Xilleon™ 220* [AMD, 2009], um processador baseado na arquitetura MIPS com suporte a renderização de gráficos 2D/3D e decodificação de vídeo MPEG-2. Um pouco depois a Broadcom lançou seu chip *BCM7030* com uma arquitetura muito parecida com o processador da AMD. O *BCM7030* oferece suporte as APIs OpenGL, Direct3D e BroadCastCL [BROADCOM, 2009]. Este chip vem integrado com algumas das famílias de set top box (dispositivos que embarca o *middleware*) da Broadcom e são um dos primeiros esforços de oferecer suporte em sistemas embarcados para TVD renderização eficiente de aplicações 3D.

2.1.3 APIs Gráficas 3D

Com a evolução da computação gráfica e das técnicas de renderização de cenas tridimensionais, surgiu a necessidade de criar ferramentas que reduzissem o tempo de desenvolvimento das aplicações tridimensionais. Em função desta necessidade surgiram as APIs gráficas que tem como objetivo oferecer uma série de funcionalidades que facilitem o desenvolvimento e execução de aplicações tridimensionais em ambientes diversos. Atualmente existe uma série de APIs, que dependem em sua maioria da plataforma para as quais foram desenvolvidas, bem como do propósito a que servem.

Existem *APIs* mais genéricas, que servem para objetivos mais gerais com é o caso da especificação *OpenGL* (*Open Graphics Library*) [SHREINER et al., 2005] e do *Java3D* [SELMAN, 2002]. Existem outras bibliotecas, porém, com objetivos mais específicos que são construídas uma camada acima das *APIs* genéricas citadas anteriormente. Um exemplo deste tipo de biblioteca são as *engines* gráficas utilizadas para o desenvolvimento de Jogos 3D. Estas *engines* possuem técnicas de computação gráfica avançadas, além de funcionalidades próprias da área de desenvolvimento de jogos, o que favorece os seus usuários. Como exemplos deste tipo de biblioteca existem a Ogre [OGRE, 2009] e o Panda3D[PANDA3D, 2009].

2.1.3.1 OpenGL

A OpenGL[3] é uma API gráfica livre utilizada para a concepção e desenvolvimento de aplicativos tridimensionais, ambientes 3D, jogos entre outros tipos de aplicações [SHREINER, 2005]. Esta biblioteca foi especificada pelo grupo Khronos, um consórcio de empresas e instituições que objetivam gerar um padrão para desenvolvimento de aplicações gráficas. O principal objetivo desta especificação é servir como padrão de referência para desenvolvimento de software e hardware para trabalhos no campo das diversas áreas correlatas à computação gráfica. Atualmente a OpenGL e o DirectX[4] da Microsoft são os principais padrões para o desenvolvimento de aplicações gráficas

2.1.3.2 Java 3D

O Java3D é uma API desenvolvida pela *Sun MicroSystems* para a rendenrização de Gráficos 3D interativos utilizando a linguagem de programação Java. O Java3D é uma API Java com foco em aplicações clientes. Outros exemplos de APIs Java deste tipo são a AWT e o Swing, APIs para o desenvolvimento de gráficos 2D [SELMAN, 2002]. O Java3D utiliza a OpenGL ou o DirectX de forma nativa para renderizar as cenas tridimensionais. Porém, a descrição da cena, a lógica da aplicação e os elementos de interação residem em código Java. Enquanto que em OpenGL a nível de descrição da cena consiste de pontos, linhas e polígonos no Java 3D podemos descrever uma cena como uma coleção de objetos.

O diferencial do Java3D em relação a OpenGL é o fato dela ser orientada a objeto, facilitando a análise e desenvolvimento das aplicações. Outra facilidade desta API é que podemos aumentar e diminuir o nível de abstração com que manipulamos a cena tridimensional. Além de aplicar os conceitos da orientação a objetos a Sun inseriu otimizações na API de forma a reduzir o *overhead* gerado pelas chamadas nativas a OpenGL. A API permite a criação de aplicações gráficas tridimensionais que podem ser desktops ou acessadas via web através de *applets* [SELMAN, 2002].

---

[3] OpenGL - http://www.opengl.org/

[4] DirectX - http://www.microsoft.com/windows/directx/

2.1.3.3 M3G

A M3G (*Mobile 3D Graphics API*) é uma especificação que tem por objetivo definir uma API Java para o desenvolvimento de Gráficos 3D para dispositivos móveis. Esta API estende as funcionalidades da plataforma *Java Micro Edition* (JME), uma versão da plataforma Java para sistemas embarcados. A API M3G foi desenvolvida utilizando o modelo de processo Java Community Process estabelecido pela Sun de forma que ela é identificada como JSR 184, referente a versão 1.1 da especificação. A versão 2.0 é sua versão mais recente e está em desenvolvimento como JSR 297.

A M3G foi desenvolvida com de forma a tentar contornar os conhecidos problemas de desenvolvimento que são enfrentados quando desenvolvemos para plataformas com sistemas embarcados. Dessa forma a API é flexível o bastante para permitir que o desenvolvedor implemente seu código completamente baseado nos recursos de software da API, ou que o usuáro utilize as vantagens do hardware específico de um dispositivo para desenvolver aplicações mais robustas [HAMER, 2007].

2.1.3.4 OpenGL ES

A OpenGL ES (*Open Graphic Library for Embedded Systems*) é um padrão de API 3D para sistemas embarcados. Ela foi especificada pelo grupo Khronos, através do grupo de trabalho intitulado OpenGL ES. Esta API oferece um mecanismo multiplataforma portáveis para acesso ao *pipeline* gráfico dos dispositivos gráficos embarcados [PULLI et al., 2008]. Esta biblioteca é vista como uma das grandes apostas do futuro por ter surgido da iniciativa de mais de 75 empresas da indústria de sistemas embarcados [TREVETT, 2004]. Vendedores de dispositivos móveis estão incorporando suporte em hardware e em software à API em seus dispositivos.

A OpenGL ES é um sub-conjunto bem definido da API OpenGL. As principais diferenças entre a OpenGL ES e a OpenGL são:

- Algumas funcionalidades da OpenGL foram removidas devido ao seu alto custo computacional;

- Foram introduzidos tipos de dados menores, de forma a ocuparem menos espaço na memória escassa dos dispositivos.

- Foi adicionado suporte a operações matemáticas de ponto-fixo.

Atualmente, alguns dispositivos móveis já oferecem suporte a OpenGL ES. Bem como sistemas operacionais como o SymbianOS e o BREW já oferecem suporte a API. De forma que desenvolvedores podem escrever código na linguagem nativa do sistema operacional e aproveitar a alta performance das aplicações.

2.1.3.5 Linguagens de descrição de Ambientes

Além das APIs gráficas já citadas, existem ainda outras tecnologias que permitem a especificação de ambientes tridimensionais interativos, bem como permitem que desenvolvedores programem modos de interagir com estes ambientes. As linguagens de descrição de ambientes são um exemplo. Estas linguagens utilizam uma sintaxe bem definida para descrever objetos tridimensionais no que diz respeito ao seu posicionamento na cena, suas propriedades materiais e seu comportamento. Na sua maioria elas precisam de um browser que interprete a descrição presente em um documento e renderize o ambiente descrito além de executar os eventos pré-definidos. Como exemplos destas linguagens podem citar o VRML, X3D e o O3D.

- VRML – O VRML (*Virtual Reality Modeling Language*) é um padrão para descrição de ambientes tridimensionais que pode ser utilizado tanto em um ambiente Desktop como através da Internet. Esta linguagem possui uma sintaxe específica e através desta sintaxe podemos definir malhas poligonais, cor, transparência, textura, etc. O padrão VRML é considerado, relativamente antigo, de forma que outros padrões mais modernos surgem com o objetivo de tomar o seu lugar.

- X3D – O X3D (eXtensible 3D) é um padrão aberto para distribuir conteúdo 3D. De fato ele não é uma API ou mesmo uma linguagem de descrição de ambientes em si. Seu objetivo é tentar integrar ambos, uma API e uma linguagem de descrição para ambientes e comportamentos através de um

arquivo simples com sintaxe baseada em XML. Para manter a compatibilidade o X3D possui funcionalidades que permitem a integração entre ele e o padrão VRML, de forma que podemos utilizar ambientes descritos neste último padrão dentro de ambientes X3D [BRUTZMAN e DALY, 2007].

- O3D – O O3D é uma API open-source desenvolvida em JavaScript que permite a criação de aplicações 3D que são executadas em navegadores web, jogos, visualizadores de modelos 3D, etc. A API O3D maximiza a performance de suas aplicações utilizando-se dos benefícios das placas gráficas que um determinado dispositivo venha a possuir, em vez de utilizar apenas renderização por software. Esta API é a mais recente entre as três tecnologias citadas e é desenvolvida e mantida pelo Google Labs[5].

2.2 TV DIGITAL

Desde o surgimento do primeiro canal de TV (BBC de Londres) a televisão vem evoluindo a forma de como disseminar seus serviços, bem como descobrindo maneiras de satisfazer seu público alvo [WU et al., 2006]. Atualmente vivemos mais uma revolução deste meio de comunicação, a adoção de um padrão de TV Digital em território brasileiro. A TV Digital tem como principal característica a união do que há de melhor do mundo da televisão e do computador. Neste contexto podemos enxergar a TV como um terminal de aprendizado, lazer e serviços.

Um sistema básico de Televisão Digital (TVD) consiste de uma estação transmissora, um meio físico sobre o qual o sinal é transmitido, que pode ser o ar ou meio físico guiado (cabo coaxial, fibra óptica etc.), e um receptor responsável por receber o sinal transmitido, decodificá-lo e exibi-lo. Como a transmissão é feita através de um fluxo de bits, há a possibilidade de se transmitir uma maior quantidade de informação multiplexada, em comparação ao sistema analógico [JONES et al., 2006]. Isso é possível principalmente graças ao desenvolvimento de técnicas de compressão, através das quais se podem produzir vídeos

[5] Google Labs - http://www.googlelabs.com/

com taxas em bits de 1/4 a 1/10 do original puro (vídeo digital não comprimido [MAIOR, 2002].

Graças a esta característica, os sistemas de TV Digital tendem a adotar padrões de codificação de vídeo que suportam resolução superior às disponíveis nos padrões de TV analógica, assim como padrões de codificação de áudio que suportam codificação de um maior número de canais. A transmissão digital viabiliza também a transmissão de múltiplos formatos simultaneamente, de forma que o conteúdo possa ser transmitido em diferentes resoluções, para diferentes dispositivos. Para tanto, é necessário que sejam estabelecidos padrões que normatizem todo o processo de captura, compressão, modulação e transmissão dos sinais de vídeo, além de todas as interfaces físicas entre os equipamentos envolvidos no processo, para garantir a compatibilidade entre os elementos envolvidos. Isso é feito através da definição de um conjunto de padrões, um para cada uma dos componentes (Figura 3) que integram um sistema de televisão digital [MORRIS e SMITH-CHAIGNEAU, 2005].

Para que a interatividade possa ser explorada ao máximo os receptores de TV Digital vêm equipados com o que denominamos de canal de retorno ou canal da interatividade. Este canal de retorno tem como objetivo oferecer a possibilidade do usuário se comunicar com serviços remotos a fim de efetivar transações interativas. No caso do SBTVD o objetivo do governo é possibilitar o acesso a interatividade para a grande maioria da população. Desta forma uma séria de estudos relacionados a que tipo de infra-estrutura de redes utilizar estão em andamento.

Em relação ao nível de interatividade que a TVDI pode nos oferecer podemos classificá-los de acordo com as características mostradas abaixo e citadas em [OLIVEIRA, et al., 2008]:

- Nível 0 - Exposição de imagens em preto e branco, poucos canais. O controle do usuário está limitado a ligar, desligar e manipular o brilho e contraste.

- Nível 1 - Imagem colorida, maior número de canais e controle remoto.

- Nível 2 – Surgimento de aparelhos periféricos como vídeo cassete, DVD players, console para jogos e etc.

- Nível 3 – Existe interatividade através de ligações telefônicas, envio de mensagens por celular ou e-mail.

- Nível 4 – Neste nível o telespectador pode escolher, em tempo real, ângulos de câmeras ou mesmo manipular que informação deseja visualizar.

- Nível 5 – Neste nível o telespectador pode participar da programação, por exemplo, enviando seus próprios vídeos.

- Nível 6 – O mesmo tipo de interação do nível seis, só que com vídeos em alta definição.

- Nível 7 – Alcance da interatividade plena, gerando conteúdo ao mesmo tempo em que a emissora. Desta forma o monopólio de produção veiculado as emissoras seria rompido.

2.2.1 Componentes de uma Rede de TV Digital

Podemos definir uma rede de TV Digital como sendo um conjunto de componentes necessário para a transmissão e recepção de sinal de TV de acordo com um sistema de TV Digital. [YAMADA et al., 2004] [LEITE et al., 2005]. Uma rede de TV Digital pode ser decomposta em uma série de componentes conforme pode ser observado na figura 3.

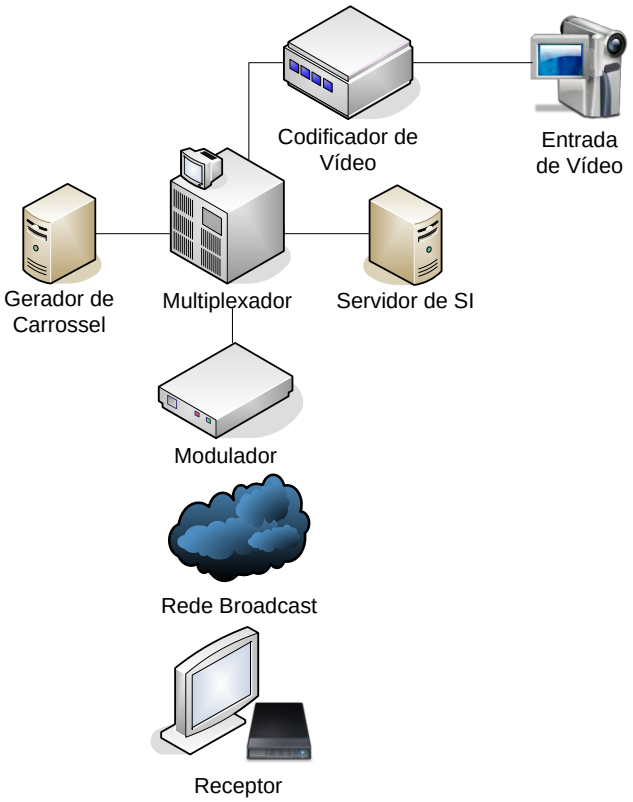


Figura 3. Esquema de uma Rede de TV Digital [BATISTA, 2008].

Basicamente, esta rede é constituída por um codificador de vídeo, um gerador de carrossel, um Servidor de SI, um multiplexador e um receptor, cujas as funcionalidades estão descritas abaixo.

- *Codificador de Vídeo* – Responsável por condificar o vídeo em um fluxo digital utilizando um determinando padrão de codificação como, por exemplo, MPEG-2 Vídeo [ISO/IEC 13818-1, 2000] ou H.264 [RICHARDSON, 2003].

- *Gerador de Carrossel* – Este componente é responsável por multiplexar no fluxo principal, que está sendo transmitido, um sistema de arquivos. Este sistema de arquivos geralmente representa as aplicações interativas e os arquivos necessários para a perfeita execução destas aplicações. Estes arquivos são serializados e enviados de forma cíclica por um determinado tempo. O gerador de carrossel segue as especificações do padrão [ISO/IEC TR 13818-6, 1998] [ISO/IEC 13818-10, 1999],

Digital Storage Media Command and Control (DSM-CC), que é responsável pelas definições do gerador de carrossel, bem como de seu processamento no receptor.

- *Servidor de SI* – Componente responsável por gerenciar as tabelas que carregam informações sobre serviços oferecidos pela transmissora. Estes serviços correspondem a informações sobre canais e a programação destes.

- *Multiplexador* – Este componente é responsável por unir todos os fluxos que serão transmitidos para o receptor. Este fluxos incluem o fluxo de vídeo, o fluxo de áudio e o fluxo de dados. O formato adotado pela maioria dos sistemas é o *MPEG-2 Transport Stream* (MPEG-2 TS) [ISO/IEC 13818-1, 2000].

- *Receptor* – É componente responsável por receber e tratar todas as informações multiplexadas no fluxo da transmissora. Um receptor deve ser capaz de demultiplexar o fluxo enviado e processar cada um dos fluxos específicos. Esse receptor, que na sua forma desacoplada é conhecido como *set-top box*, pode ser visto como um computador adaptado para as necessidades do ambiente televisivo, possuindo processador, memória, sistema operacional, etc. Geralmente é neste receptor que está instalado o middleware de TV, software responsável por abstrair as características específicas do hardware de cada receptor, gerenciar e executar as aplicações residentes, bem como as aplicações enviadas pela transmissora [SOUZA FILHO et al., 2007] [SOARES et al., 2007]. Na figura 4 podemos observar como seria a arquitetura básica de um receptor de TV Digital.

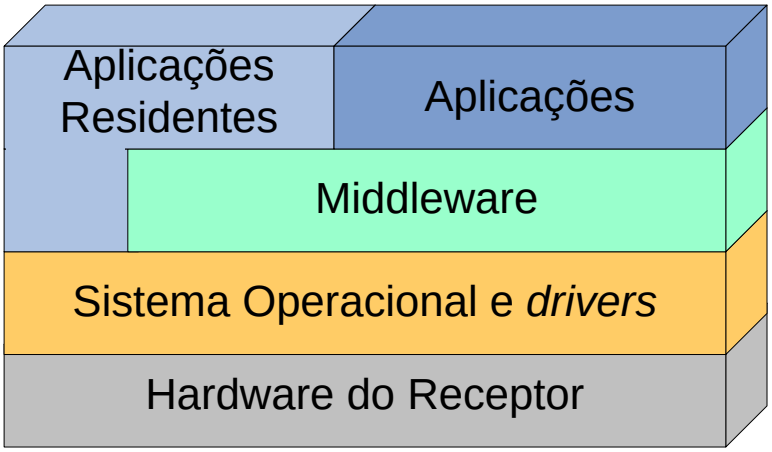


Figura 4. Arquitetura básica de um receptor de TV Digital [BATISTA, 2008].

2.2.2 Padrões de TV Digital

Atualmente existem quatro padrões mundiais de sistema de televisão digital interativa, quais sejam ATSC, DVB, ISDB e o SBTVD. Estes padrões se diferenciam pela diversidade de soluções tecnológicas que podem ser adotadas [SOUZA FILHO et al., 2007] [SOARES et al., 2007] [MORRIS e SMITH-CHAIGNEAU, 2005].

2.2.2.1 ATSC – (*Advanced Television System Committee*)

Padrão adotado pelos Estados Unidos, o ATSC [ATSC, 2009] permite diversas configurações para a camada de transmissão uma vez que define diferentes esquemas de modulação para transmissão terrestre, via cabo e via satélite [HENDERSON et al., 2006]. Opera com canais de 6, 7 ou 8 MHz utilizando modulação 8VSB com taxa de transmissão de 19,8 Mbps [BRETL et al., 2006]. Embora o padrão permita diversos modos de transmissão com diferentes níveis de resolução da imagem, o modo de alta definição (HDTV) é o utilizado. Na camada de codificação de áudio o padrão utilizado é o Dolby AC-3. Para a multiplexação de áudio, vídeo e dados o padrão utilizado é o MPEG-2 [DAVIDSON et al., 2006].

2.2.2.2 DVB – (*Digital Video Broadcasting*)

O padrão DVB [DVB, 2009] foi adotado na Europa. Este padrão adota esquema de modulação COFDM (*Coded Orthogonal Frequency Division Multiplexing*) com uma taxa de

transmissão que pode variar entre 5 e 31,7 Mbps. O padrão DVB-T (*DVB Terrestrial*) suporta seis modos de transmissão com resoluções que variam de 1080 a 240 linhas podendo ser utilizados tanto para sistema de alta definição (HDTV) quanto para sistemas de baixa definição (LDTV). Atualmente os europeus utilizam o modo de transmissão com qualidade de vídeo padrão (SDTV) o que permite que até 6 programas sejam transmitidos em um mesmo canal [REIMERS, 2006]. Nas camadas de codificação de áudio e vídeo, além da camada de transporte de transporte o padrão utilizado é o MPEG-2.

2.2.2.3 ISDB – (*Integrated Services Digital Broadcasting*)

Também conhecido como padrão japonês de televisão digital, o ISDB [UEHARA, 2006] é adotado em sua plenitude apenas no Japão. Este padrão utiliza o esquema de modulação COFDM e foi projetado para suportar sistemas hierárquicos com múltiplos níveis. No esquema utilizado no Japão são transmitidos programas em HDTV e LDTV para dispositivos móveis. O áudio é codificado utilizando MPEG-2 AAC e o vídeo é codificado utilizando MPEG-2 Vídeo [UEHARA, 2006]. Assim como os outros dois padrões citados o ISDB utiliza o MPEG-2 na camada de transporte.

2.2.2.4 ISDB-Tb – (*Integrated Services Digital Broadcasting, Terrestrial, Brazilian version*)

O ISDB-Tb é o padrão brasileiro de televisão digital. Com o decreto lei 4.901 de 26 de novembro de 2003 deram-se início os trabalhos para a análise, desenvolvimento e implantação do sistema. Como foco das pesquisas foram propostas várias inovações com o objetivo de seguir o propósito do projeto e de adaptar o modelo a realidade brasileira. Neste contexto era inviável a adoção pura e simples de um dos padrões estrangeiros já existentes. De fato o que foi adotado dos padrões estrangeiros, mais precisamente do japonês, foi o sistema de modulação. No demais, os pesquisadores brasileiros procuraram desenvolver um sistema que atendesse as necessidades de nosso país e que tinha como requisitos básicos a robustez, flexibilidade, interatividade, inclusão social e baixa complexidade para o usuário. Em termos de inovação, o ISDB-Tb criou seu próprio modelo de *middleware* que é compatível com as normas internacionais, mas que, além disso, possui certas características

existentes apenas nele. Tais características é o que difere o Ginga [SOUZA FILHO et al., 2007] [SOARES et al., 2007] do restante dos *middlewares* para TVDI.

Em termos de esquema de modulação, o ISDB-Tb utiliza o mesmo esquema japonês, o COFDM, podendo, da mesma forma, suportar sistemas hierárquicos de múltiplos níveis. Seguindo o modelo de negócio japonês os programas serão transmitidos em dois formatos de definição, *HDTV* para as residências e *LDTV* para dispositivos móveis. Na camada de codificação de vídeo é utilizado o MPEG-4 Vídeo (H.264), na camada de codificação de áudio é utilizado o *MPEG-4 AAC* (*Advanced Audio Coding*) e para a camada de transporte de áudio, vídeo e dados utilizamos o MPEG-2 TS (*Transport Stream*) [ABNT NBR 15601, 2008].

2.2.3 Middleware para TV

Middleware é uma camada de software intermediária entre o código das aplicações e o sistema operacional ou uma plataforma de hardware [SOARES, 2008]. Um middleware para televisão digital consiste de todos os recursos necessários para suportar o desenvolvimento e execução das aplicações. Neste sentido ele é constituído pelas máquinas de execução das linguagens suportadas pelo padrão de TV, as bibliotecas e funções que permitem o desenvolvimento com estas linguagens.

Muitas empresas têm lançado modelos de middleware proprietários para os seus usuários de TV. Esta tendência é mais visível nas empresas de serviço de TV a cabo. Estas empresas oferecem seus middlewares embarcados nos set top box que alugam aos seus consumidores, com o objetivo de oferecer um certo nível de interatividade. Dentre os middlewares para TV proprietários podemos destacar o OpenTV Core (OpenTV), NDS Core (NDS), MediaHighway (Canal+), PowerTV e o MicrosoftTV (Microsoft).

Atualmente, porém, o que se verifica é uma tendência a utilização de middlewares baseados em padrões abertos. Este fato ganhou força com a criação da TV Digital aberta, de forma que, os governos dos países que têm este tipo de tecnologia, além de produtores de conteúdo e fabricantes de equipamentos eletrônicos, trabalham juntos com o objetivo de encontrar soluções abertas para middleware de TVD, visando especialmente a interatividade.

Os principais padrões de middleware aberto são o *Multimedia Home Plataform* (MHP) [TS 102 812, 2003] do padrão europeu (DVB), o *OpenCable Application Plataform* (OCAP) e o True2Way do *CableLabs*, o *Advanced Common Application* (ACAP) [ATSC, 2005] e o *Digital TV Applications Software Enviroment* (DASE) do padrão americado (ATSC), o ARIB B.23 [ARIB, 2004], do padrão japonês (ISDB) e por fim, o Ginga, middleware do padrão brasileiro de TVD (ISDB-Tb) [SOUZA FILHO et al., 2007].

Para tornar possível a harmonização das diferenças entre os *middlewares* dos diferentes sistemas tornando possível o desenvolvimento de aplicações que pudessem ser executadas em diferentes sistemas, foi elaborada uma norma desenvolvida com base no *middleware* MHP, do padrão europeu, denominada GEM (*Globally Executable MHP*) [GEM, 2009], que identifica e dá suporte a APIs independentes de sistema específico. Essa especificação é atualmente adotada pelos padrões de *middleware* procedural japonês (ARIB B.23), americano (ATSC - ACAP) e o europeu (DVB MHP). Basicamente, o GEM composto pela união das principais API's utilizadas em ambientes de TV Digital. Na figura 5 podemos observar como o GEM foi concebido e de quais padrões a sua especificação foi originada.

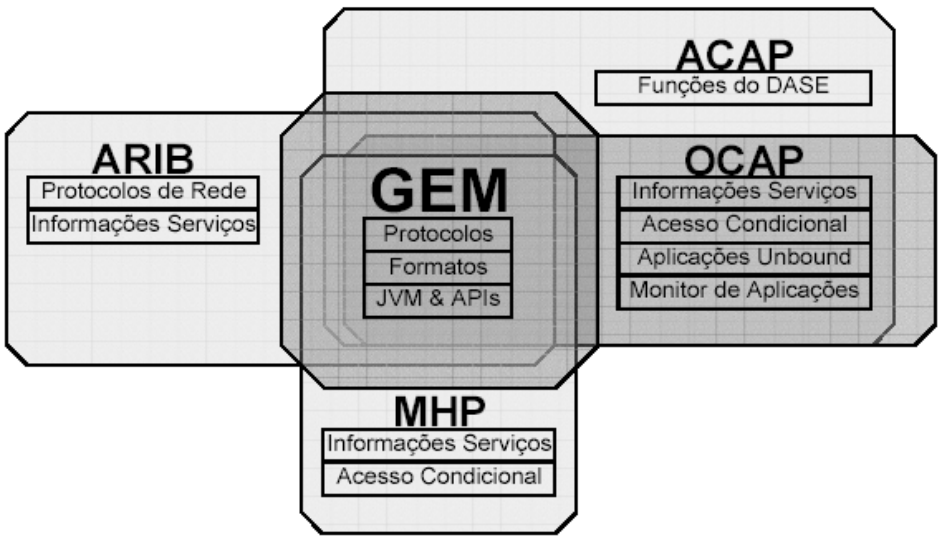


Figura 5. Especificação do GEM (Globally executable middleware) [GEM, 2009].

Outra compatibilização relacionada ao desenvolvimento de middlewares para TV partiu da ITU (*Internacional Telecommunication Union*), que publicou um conjunto de

recomendações através dos documentos ITU-T ITU-T: J.200 [ITU, 2001], J.201 [ITU, 2003a] e J.202 [ITU, 2003b]. Essas recomendações visam oferecer uma especificação para middlewares de TV Digital abordando vários aspectos do projeto de middleware, como os diferentes níveis de arquitetura do *middleware* e as definições para *middleware* declarativo e procedural [BATISTA, 2006]. Na figura 6 podemos observar a arquitetura proposta pelas recomendações da ITU, relacionadas à arquitetura do *middleware* e seus níveis.

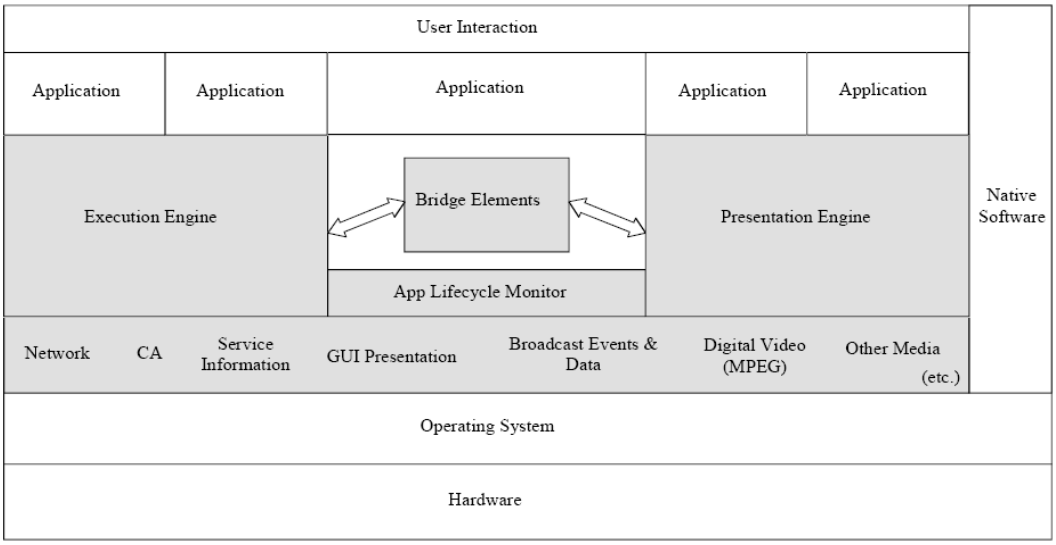


Figura 6. Arquitetura recomendada pela ITU para middlewares de TV Digital [ITU, 2003a] [ITU, 2003b].

Como podemos observar, estão presentes o ambiente declarativo e o procedural, bem como uma ponte entre os ambientes, de forma a permitir a interoperabilidade. Os padrões de *middleware* que seguem esta recomendação definem linguagens para ambiente declarativo e procedural, conforme pode ser observado na figura 7.

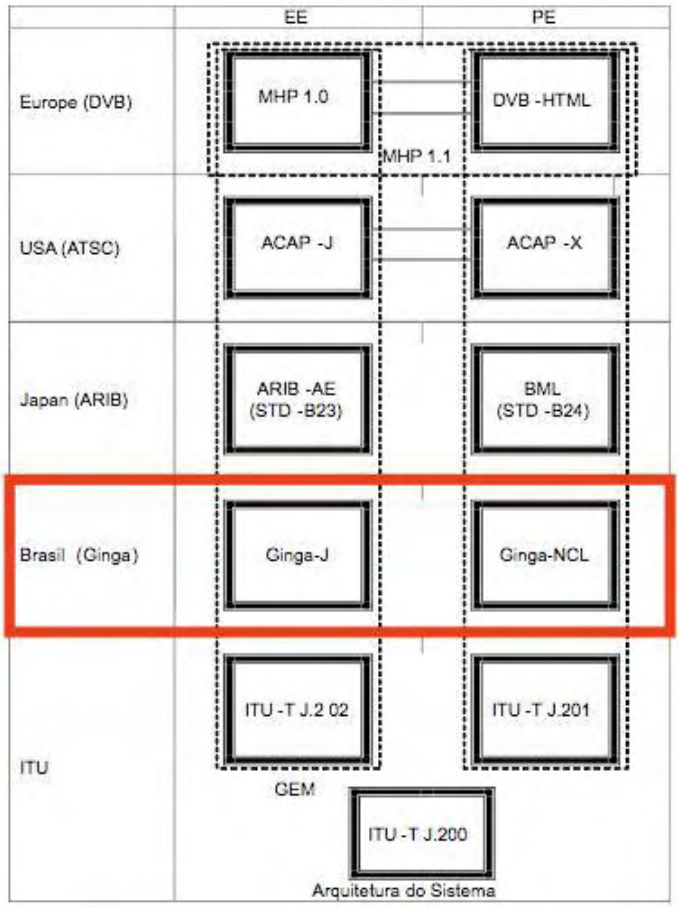


Figura 7. Liguagens do ambiente declarativo e procedural dos padrões que seguem as definições da norma ITU-T J.200.

2.2.3.1 Middleware Ginga

O Ginga é o padrão de *middleware* do Sistema Brasileiro de Televisão Digital. Ele é responsável por dar suporte às aplicações declarativas e procedurais, sendo compatível com as definições internacionais da ITU [ITU, 2001]. Uma das características que diferencia o Ginga dos outros *middlewares* existentes é justamente os requisitos sob os quais ele foi concebido. Quando o governo brasileiro iniciou as pesquisas para o desenvolvimento do *middleware* brasileiro, ele determinou alguns requisitos baseados, em sua maioria, em particularidades do contexto social brasileiro. As funcionalidades inovadoras do Ginga foram desenvolvidas de forma que seja possível a criação de aplicações avançadas que procurem explorar a integração com outras tecnologias afim de facilitar o acesso à toda a população. Estas funcionalidades estão inseridas na API de inovação do *middleware*.

O Ginga é resultado de alguns anos de pesquisa na área de middleware para TV Digital. Sua especificação teve origem na união de dois trabalhos de ambientes para TVDI, o MAESTRO [SOARES, 2006] e o FlexTV [LEITE et al., 2005], o primeiro com foco em um ambiente declarativo e o segundo com foco em um ambiente procedural. Na figura 8 é apresentada a arquitetura do *middleware* Ginga.

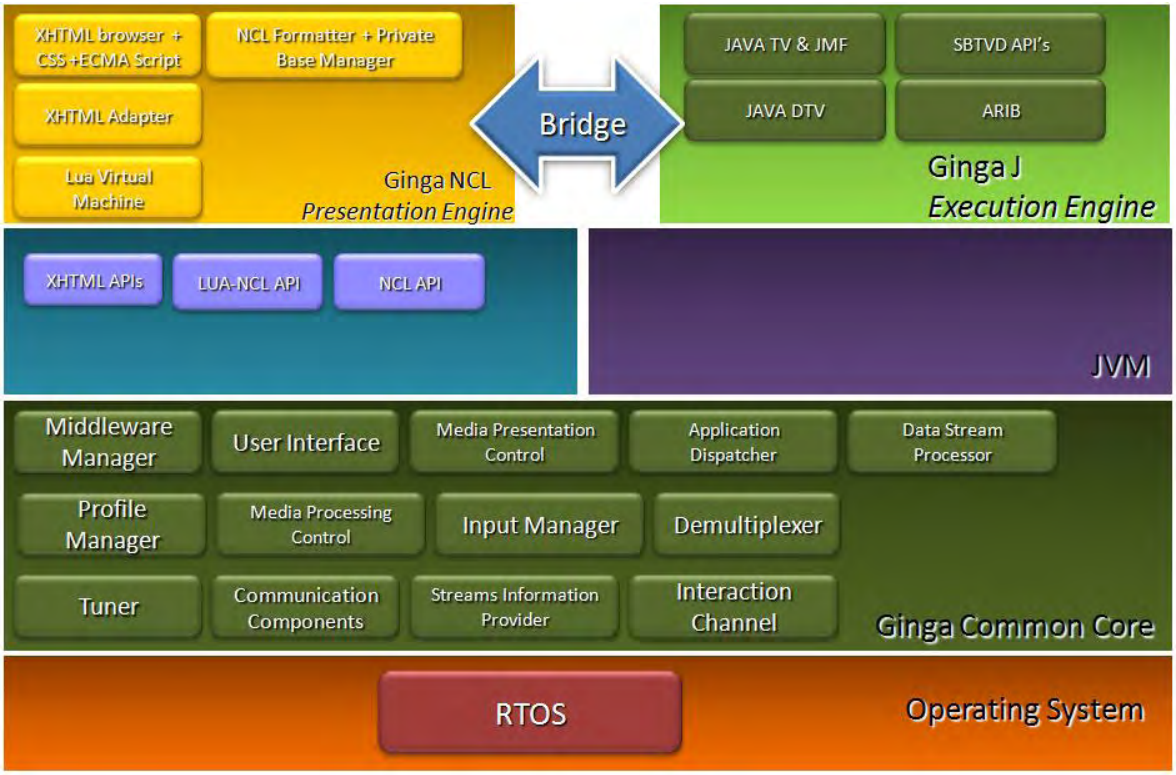


Figura 8. Arquitetura do *middleware* Ginga.

O Ginga-NCL [SOARES et al., 2007] é o ambiente declarativo do middleware Ginga. A linguagem utilizada neste ambiente é o NCL[6] (*Nested Context Language*) [ANTONACCI, 2000], uma linguagem de descrição e sincronização de mídias desenvolvida pela PUC-Rio. Na camada do Ginga-NCL estão presentes os formatadores de conteúdo NCL (*NCL Formatters*), que são os mecanismos responsáveis pela decodificação dos documentos NCL.Outros componentes presentes nesta camada são os baseado em XHTML, que incluem uma linguagem de descrição de estilos (CSS) e um interpretador para ECMAScript [ECMA-

[6] Linguagem NCL - http://www.ncl.org.br/

262, 2009]. Por fim, esta camada ainda possui uma máquina virtual para a linguagem de script Lua[7].

O Ginga-J [SOUZA FILHO et al., 2007] é o ambiente procedutal do *middleware* Ginga. A linguagem utilizada neste ambiente é o Java da Sun. Este ambiente é constituído por um conjunto de API's Java que viabilizam o desenvolvimento de aplicações para TVDI. Basicamente, este conjunto de API's é composto pela API' de Serviços de Informação (SI) do padrão ISDB ARIB B.23, pela especificação JAVA DTV da Sun e pela API JMF 2.1 e por outras API's adicionais, conforme pode ser observado na figura 9.

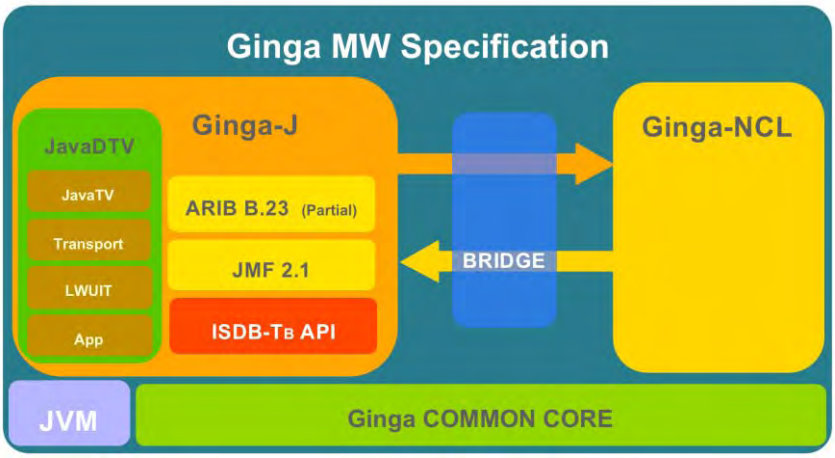


Figura 9. Arquitetura do Ginga-J.

As API Java DTV [SUN, 2008] foram desenvolvidas em comum acordo entre a Sun Microsystem e o Governo Federal brasileiro com o objetivo de evitar a cobrança de *royalties* que outras API's definidas pelo GEM possuíam. Algumas das API's do GEM que foram substituídas pelo Java DTV são o DAVIC (*Digital Audio Video Council*), o HAVi (Home Audio Video Interoperability) e o pacote DVB Application. Os equivalentes funcionais destas API's no Java DTV são, respectivamente, os pacotes Transport, LWUIT e Application. Na figura 10, podemos observar o conjunto de API's disponíveis para desenvolvimento no ambiente Ginga-J. Vale ressaltar que as API's do Java TV também estão inclusas no ambiente Ginga-J, além da API's *Connected Device Configuration, Foundation Profile e Personal Basis Profile*. Uma das principais inovações incorporadas a este novo conjunto de APIs é o

[7] Linguagem Lua - http://www.lua.org/

LWUIT[8]. Esta API gráfica foi desenvolvida inicialmente para a utilização em dispositivos móveis, porém foi adaptada para atender aos requisitos de um ambiente de TVDI. Ela é baseada na API gráfica Swing da Sun e possui uma série de componentes gráficos, gerenciadores de layout e mecanismos para a transição animada de telas. Outra característica interessante desta API é sua ponte com a API 3D M3G, de forma que a LWUIT oferece alguns recursos 3D para a transição de tela.

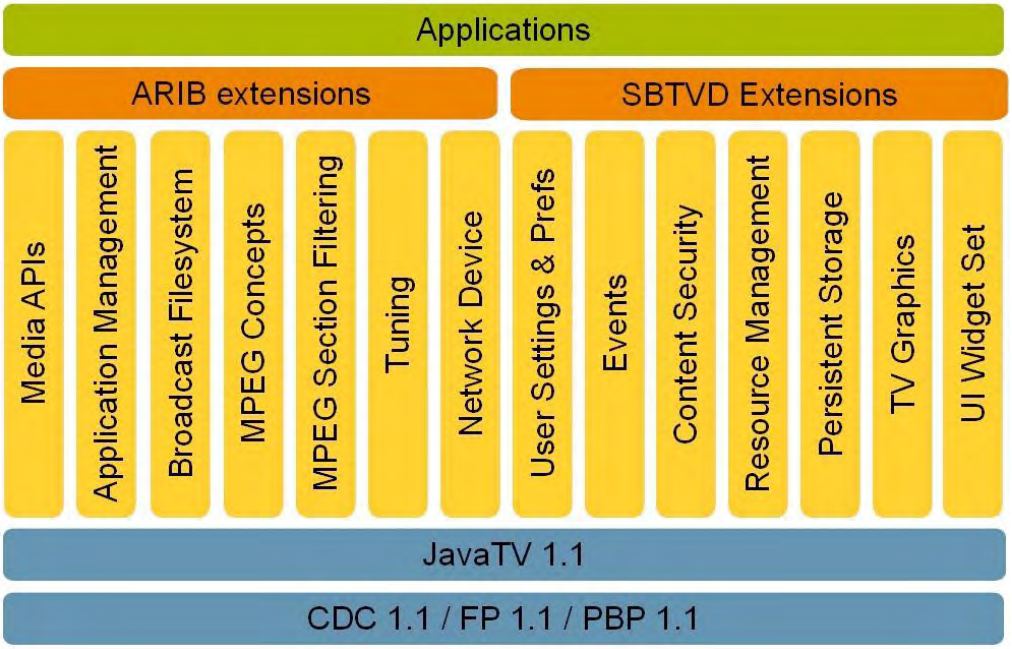


Figura 10. API's disponíveis para desenvolvimento utilizando o Java DTV.

O *Ginga Common Core* (Figura 8) é a camada do *middleware* Ginga responsável por oferecer suporte, em baixo nível, as API's dos ambientes declarativo e procedural. Ele é composto por um conjunto de componentes nativos que utilizam as funcionalidades do hardware específico sobre o qual estão sendo executado para atender as solicitações das camadas superiores. Alguns dos componentes que estão presentes nesta camada são os decodificadores de áudio e vídeo, os sintonizadores e bibliotecas para acesso ao sistema de informações do fluxo de dados *MPEG-2 System*, responsável por coletar as informações sobre a programação que está sendo transmitida, bem como informações sobre aplicações interativas.

[8] LWUIT - https://lwuit.dev.java.net

Uma das grandes novidades do *middleware* Ginga é a capacidade de interação com múltiplos dispositivos. Os outros padrões mundiais só prevêem a interação do usuário com o *middleware* utilizando o controle remoto da TV. O Ginga, porém, prevê a utilização de dispositivos móveis para a interação com as aplicações interativas. Mais do que isto, permite que múltiplos usuários possam interagir ao mesmo tempo com tais aplicações. Desta forma, saímos de um ambiente onde só existe um usuário e o *set top box* para um ambiente onde vários dispositivos podem embarcar o *middleware* Ginga e muitas de suas funcionalidades. Estes dispositivos são chamados por SILVA (2008) de dispositivos Ginga.

Através desta característica inovadora. o telespectador pode interagir com o dispositivo Ginga através de dispositivos de interação que podem conter componentes de software Ginga. Para que um dispositivo de interação possa ser utilizado, ele deve se registrar junto ao *middleware*, e durante esse processo o dispositivo de interação pode receber o componente de software necessário para viabilizar a comunicação com este dispositivo. Como resposta à informação enviada pelo telespectador, o Ginga apresenta a saída de vídeo e áudio utilizando seu próprio monitor e auto-falantes ou os dos dispositivos de interação. Um único dispositivo pode ter capacidade de entrada e saída simultâneas. Na figura 11, podemos observar como se dá esta comunicação entre sistemas que embarcam o *middleware* Ginga e os múltiplos interadores.

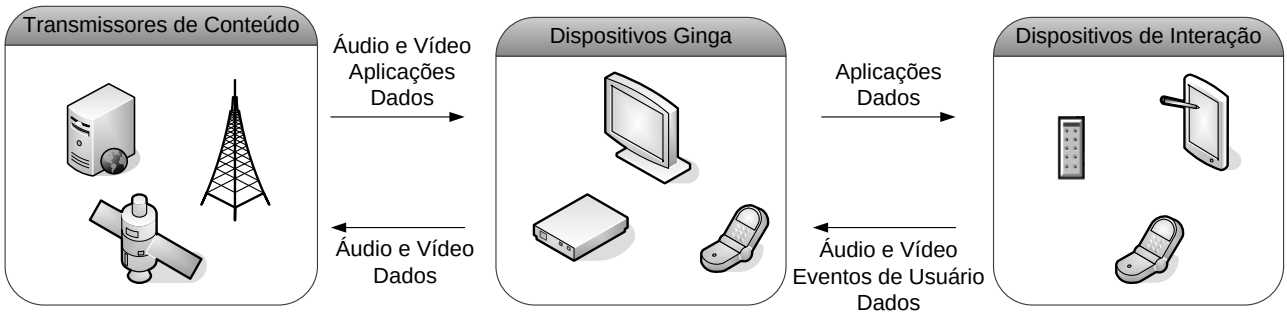


Figura 11. Comunicação entre dispositivos Ginga e os dispositivos de interação [SILVA, 2008].

2.3 APLICAÇÕES PARA TELEVISÃO DIGITAL E INTERATIVA

Com o advento da TV Digital, surgiram uma série de mudanças no que diz respeito à transmissão e apresentação de conteúdo para o telespectador. Uma das principais mudanças proporcionadas por esta nova tecnologia é, sem dúvida, a possibilidade de transmissão e apresentação de conteúdo interativo [BECKER, 2009]. Este conteúdo interativo nada mais é do que softwares que serão executados pelo *middleware* embarcado nos STBs (set top boxes).

Dentro de um ambiente de TVDI, estes softwares, ou aplicações interativas, possuem algumas características particulares em relação às aplicações convencionais. Estas diferenças estão relacionadas ao público que esta aplicação irá atender, bem como o seu tempo de vida [VEIGA, 2006]. Outras características que o diferenciam são o fato de eles exigirem uma infra-estrutura de transmissão e um modelo de negócio, que hoje em dia, ainda não está totalmente definido.

Os desenvolvedores de aplicações interativas podem optar por um dos dois paradigmas de desenvolvimento de aplicações existentes dentro dos padrões: o procedural ou declarativo. A maioria dos padrões mundiais oferece suporte a linguagens em ambos os paradigmas. Os padrões que seguem as recomendações da ITU (ITU-T J.200) são orientados a oferecer estas duas possibilidades. Para saber quais linguagens cada padrão suporta basta observar a figura 7 descrita no tópico 2.2.3. No caso do padrão brasileiro, as aplicações podem ser desenvolvidas em Java no caso do paradigma procedural ou em NCL, no caso do paradigma declarativo. Existe ainda a possibilidade de desenvolvermos aplicações NCL que executem código Lua. Ou mesmo aplicações NCL que invoque código Java e vice-versa, através da ponte de comunicação que é descrita no padrão [ABNT NBR 15606-2, 2008].

Quanto à finalidade das aplicações, elas podem servir para os diversos propósitos e áreas. Aplicações mais simples para a realização de enquetes é são bastante exploradas. Neste tipo de aplicação interativa o usuário é chamado a responder algum tipo de pergunta sobre um assunto em particular. Espera-se que este tipo de aplicação sejam as primeiras a surgir dado o seu baixo nível de complexidade, além do grande número de enquetes que as emissoras elaboram em seus programas.

Um exemplo de enquete é a aplicação para a escolha da musa do carnaval carioca, desenvolvida em Java pelo LAVID[9] (Laboratório de Vídeo Digital) da Universidade Federal da Paraíba (UFPB). Nesta aplicação o usuário é convidado a eleger a musa do carnaval carioca através da seleção de uma das candidatas que são mostradas pela enquete (ver figura 12). O usuário utiliza o controle remoto para navegar entre as candidatas, uma descrição simples de cada uma é apresentada e então o usuário utiliza um dos botões do controle remoto para selecionar a sua favorita. Esta informação seria enviada para o servidor de contagem de votos através do canal de retorno. De mãos destes dados a emissora pode calcular o resultado de sua enquete.

Figura 12. Aplicação do tipo enquete. Neste caso o usuário é convidado a escolher a musa do carnaval carioca.

Outro tipo de aplicação que pode ser desenvolvida para ambientes de TVDI são as relacionadas à educação (*t-learning*). O governo brasileiro, em particular, através do decreto lei que institui o Sistema Brasileiro de TV Digital (SBTVD), deixa claro que um dos focos do novo sistema de TV deve ser a sua utilização com o objetivo de promover a inclusão social através do ensino à distância. Desta forma, o campo de aplicações voltadas ao ensino e aprendizagem tem uma grande importância.

9 LAVID – www.lavid.ufpb.br

Um exemplo de aplicação relacionada ao ensino é o VestibaTV (figura 13) [PALMEIRA, 2009]. O VestibaTV é uma aplicação de t-*learning* que tem por objetivo auxiliar o aluno nos estudos preparatórios para ingresso em curso de nível superior através de vestibular (ver figura 13). Suas principais características são o uso de linguagem acessível aos adolescentes, apresentação de conteúdo interativo, apresentação de exercícios e simulados, além da apresentação de tópicos relevantes os estudantes e a comunidade como um todo [REY-LOPEZ et al., 2007]. O vestibaTV foi desenvolvido utilizando a linguagem declarativa NCL.

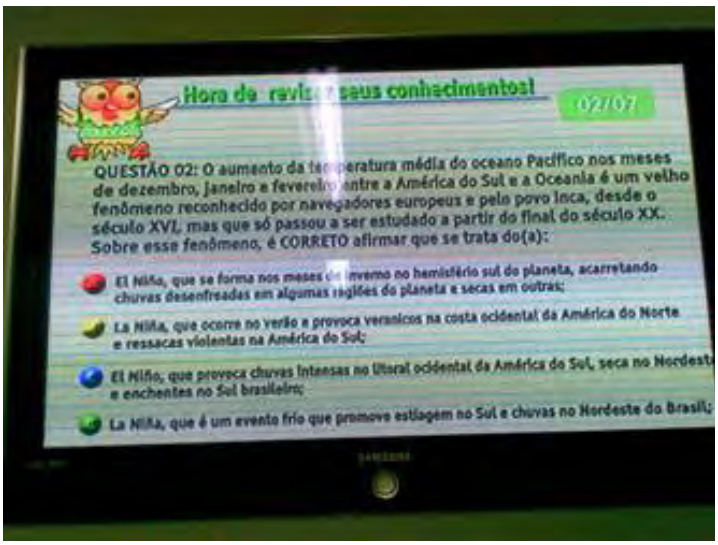


Figura 13. VestibaTV: uma aplicação educativa para TVDI (t-learning) [PALMEIRA, 2009].

Especificamente no Brasil, outro tipo de aplicação que deverá ser explorada são as que utilizam a possibilidade de interação multiusuário. Esta característica é um dos diferenciais do padrão brasileiro em relação aos outros padrões mundiais. Neste sentido um campo muito grande de possibilidades se abre. Aplicações na área de jogos, entretenimento, colaboração poderão ser exploradas. A possibilidade de utilizar dispositivos móveis para interação com a TV nos permite desenvolver aplicações que podem ser executadas tanto em um ambiente de TV como no próprio dispositivo de interação. Esta característica favorece, por exemplo, aplicações em que o usuário necessite de uma certa privacidade, como no caso das aplicações bancárias (*t-banking*).

A Torcida Virtual [TAVARES et al., 2004] é um exemplo de aplicação que utiliza os benefícios das inovações presentes no padrão brasileiro de TV Digital (ver figura 14). Nesta

aplicação o usuário tem a possibilidade de interagir com amigos dispostos geograficamente em locais distintos através de um ambiente acústico compartilhado. A aplicação da torcida virtual apresenta a arquibancada de um estádio de futebol. O usuário autorizado escolhe uma posição onde deseja sentar e utilizando o seu dispositivo móvel pode interagir através de áudio com as pessoas ao seu redor, dentro do ambiente virtual. Um servidor de áudio é responsável por receber os dados de áudio de cada usuário do ambiente e multiplexar a informação levando em consideração a distância entre as cadeiras dos usuários e a intensidade de suas vozes.

Figura 14. Torcida Virutal, um ambiente acústico compartilhado onde os usuários torcem para o seu time de futebol favorito [TAVARES et al., 2004].

A Torcida Virtual é um exemplo de aplicação complexa que demonstra o potencial do padrão brasileiro, quando falamos de desenvolvimento de aplicações para TVDI. Existem outras áreas em que aplicações interativas podem ser desenvolvidas. A área de comércio através da TV (*t-commerce*), transações bancários, ensino à distância, jogos, economia e cidadania são apenas outras que podem ser citadas (ver figura 15). De fato, existe um longo caminho inexplorado relacionado ao verdadeiro potencial que estas aplicações interativas têm.

A possibilidade de unir estas atuais características com tecnologias 3D amplia mais ainda este potencial.

Figura 15. Aplicações interativas nas áreas de entretenimento, t-banking e economia, desenvolvidas pela HXD[10].

2.4 APLICAÇÕES 3D PARA DISPOSITIVOS COM SISTEMAS EMBARCADOS

Os avanços nas pesquisas em hardware permitiram a criação de dispositivos para sistemas embarcados com maior poder de processamento. Recentemente, muitos destes dispositivos são equipados com placas gráficas que permitem renderizar gráficos com mais eficiência e sofisticação. Outra fator que contribuiu para os avanços nesta área foi a especificação de novos padrões e APIs específicas para esta área como é o caso do M3G e OpenGL ES e a JSR 239[11] (*bind* em linguagem java para a OpenGL ES) [KAMEYAMA et al., 2003] [SOHN et al, 2004].

Os dispositivos móveis são o tipo de sistemas embarcados que mais avança em direção a suportarem tecnologias 3D. Atualmente, muitos celulares e PDAs (*personal digital*

[10] HXD - http://www.hxd.com.br/

[11] *Java Binding for OpenGL ES* - http://jcp.org/en/jsr/detail?id=239

assistant) já vêm com placas de vídeo integradas e com suporte ao desenvolvimento de aplicações tridimensionais. Através deste tipo de suporte é possível desenvolvermos a aplicações tridimensionais para diversos propósitos. Porém, como na indústria de PCs, os primeiros investimentos são voltados para a área de entretenimento, mais especificamente jogos eletrônicos. O mercado de jogos 3D para dispositivos móveis é recente. Porém vem crescendo à medida que novos modelos com suporte 3D são lançados no mercado. Quake é um exemplo de jogo que já está sendo distribuído para alguns modelos de celulares com suporte 3D. Quake é um jogo de tiro em primeira pessoa, sua primeira versão foi lançada em 1996, sendo bem recebido pelo público, de forma que, faz sucesso até hoje. Na figura 16 podemos observar o jogo Quake S60 com aceleração OpenGL ES pronto para execução na plataforma *Symbian* com *chipset* OMAP 2420, incorporado a modelos de celulares como o N95 e E90 da Nokia.

Figura 16. Jogo Quake S60 com aceleração OpenGL ES.

Um trabalho com um foco mais educacional é desenvolvido por [VIRGÍNIO FILHO, 2009]. O autor propôs um jogo na modalidade *serious game* que utiliza aceleração OpenGL ES. O GeoEspaço Mobile é um jogo educacional que tem por objetivo apresentar para o jogador desafios de geometria espacial em um contexto de um jogo de aventura (ver figura 17). O usuário deve utilizar seus conhecimentos sobre geometria espacial para resolver os desafios lançados pelo jogo e seguir pelas diversas fases. A opção por contextualizar os desafios em um jogo de aventura proporciona um maior fator atrativo ao jogador sem, no entanto, prejudicar o jogo no que diz respeito ao limite de processamento do dispositivo.

Figura 17. GeoEspaço Mobile, um serious game para dispositivos móvies que utiliza tecnologias 3D [VIRGÍNIO FILHO, 2009].

Outra abordagem interessante foi a seguida por [NADALUTTI et al., 2006]. O autor integrou um *browser* X3D para dispositivos moveis utilizando aceleração OpenGL ES. O MobiX3D como é denominado o *browser* suporta modelos clássicos de iluminação, além de alguns algoritmos mais sofisticados. Atualmente o MobiX3D suporta um subconjunto das funcionalidades do *profile* interativo definido pelo padrão X3D para dispositivos com baixa capacidade de processamento. O MobiX3D oferece também suporte completo ao padrão H-Anim descrito pelo padrão para animação de humanóides.O *browser* foi desenvolvido em C++ para plataforma *PocketPC*. Na figura 18 pode-se ver alguns ambientes X3D sendo executados no *browser*.

Figura 18. MobiX3D: browser X3D para dispositivos moveis. Utiliza OpenGL ES para renderização das cenas 3D [NADALUTTI et al., 2006].

2.5 INTEGRAÇÃO DE TECNOLOGIAS 3D COM TV DIGITAL

Embora exista uma série de avanços com relação ao suporte de aplicações 3D em dispositivos móveis, este fato não se repete quando falamos de TV Digital. As pesquisas nesta área estão em fase inicial. de forma que existem poucos trabalhos que tem por objetivo analisar o potencial da integração destas duas tecnologias. Porém, o suporte deste tipo de tecnologia, bem como, o uso de Inteligência Artificial [MAAD, 2002] e Realidade Virtual [ZUFFO, 2001] podem contribuir para a evolução do conceito de interação com o conteúdo interativo [MAAD, 2003b]. Alguns trabalhos relacionados já estão surgindo nesta área. Alguns com objetivos específicos, como o desenvolvimento de uma aplicação específica [MAAD, 2003a] [KAMMANN, 2005]. Outras buscam uma forma de prover suporte ao transporte, processamento e apresentação de gráficos 3D [FEHN et al., 2006].

2.5.1 Aplicações 3D para TV

No que diz respeito a aplicações 3D, alguns pesquisadores desenvolveram trabalhos específicos que resolvem o problema de utilização de tecnologias 3D pontualmente, para o seu caso em particular. Alguns destes trabalhos estão relacionados às áreas de RV (Realidade Virtual), RA (Realidade Aumentada) e interfaces 3D assistidas por sistemas inteligentes.

O trabalho de MAAD (2003a) apresenta uma plataforma com a capacidade de prover interação inteligente dentro de um ambiente de TVDI. O autor discute a utilização de um *avatar* virtual 3D para auxiliar a interação entre o usuário e a TV. Embora o foco deste trabalho seja o serviço inteligente que está sendo provido, o autor faz uma discussão sobre as vantagens de utilizar um avatar virtual 3D para a interação com o usuário [MAAD, 2003b]. O autor utilizou um avatar 3D baseado na face de uma mulher, a qual chamou de Marylin (ver figura 19).Os usuários do sistema poderiam utilizar os serviços do avatar para acessarem informações sobre conteúdo personalizado.

Figura 19. Avatar virtual Marylin, sistema inteligente para auxilio na busca de conteúdo personalizado [MAAD, 2003a].

O trabalho de Kammann (2005), por sua vez, aborda o uso de Realidade Aumentada no ambiente de TV. O autor analisa como utilizar ambientes de Realidade Aumentada na TV. Como estudo de caso o autor implementa um jogo baseado em um esporte chamado *pelota*. O jogo gera conteúdo de vídeo modificado para incluir as cenas gráficas super postas, oferece vários ângulos de visão e interatividade (ver figura 20). O autor utiliza o estudo de caso para avaliar as restrições, problemas e o potencial de utilizar integrar estes dois tipos de tecnologias. O autor se preocupa também em discutir sobre as dificuldades no desenvolvimento, em relação a concepção do sistema, restrição do hardware e das APIs gráficas estudadas.

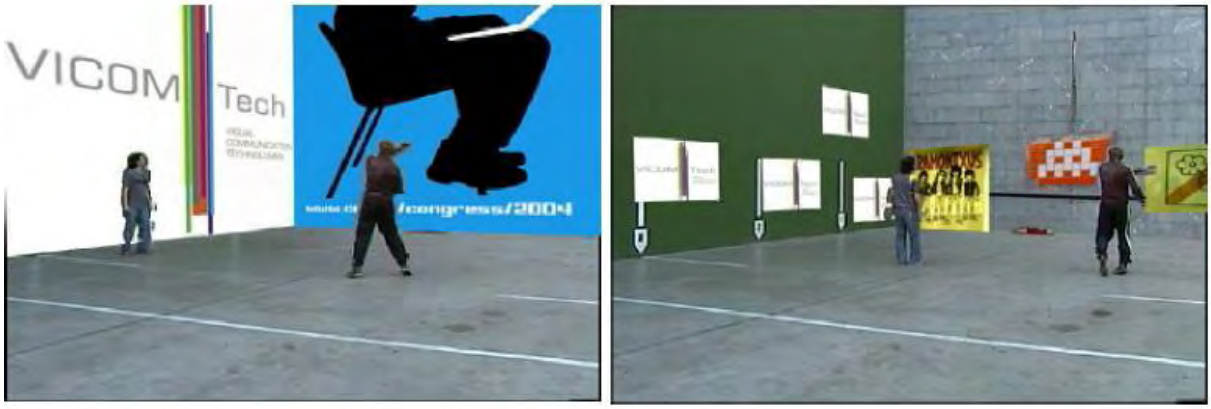


Figura 20. Execução do jogo baseada em Realidade Aumentada Pelota [KAMMANN, 2005].

2.5.2 Utilizando MPEG-4 para construção de gráficos 3D

Alguns trabalhos utilizam o padrão de compressão de vídeo MPEG-4 para a descrição das cenas tridimensionais. Isto é possível porque o padrão MPEG-4 define um conjunto de ferramentas para a construção de conteúdo gráfico mais sofisticado. O padrão permite trabalharmos com objetos de mídia naturais e sintéticos [PULLES e SASNO, 2004]. O padrão permite a criação e manipulação de imagens animadas 2D e 3D. Como mecanismo de sincronização dos objetos de mídia, o padrão dispõe de uma linguagem de descrição de cena chamada BIFS (*Binary Format for Scene*) [SINGÈS et al., 2000]. Através destas linguagens é possível descrevermos nós de mídia dentro de arquivos MPEG-4. Estes nós podem ser arquivos de áudio, vídeo, imagens ou objetos tridimensionais.

Um dos primeiros trabalhos que utiliza o padrão MPEG-4 como estratégia para a integração de gráficos 3D com TV é o de PULLES e SASNO (2004). Os autores demonstram neste trabalho como o padrão MHP pode ser utilizado em conjunto com o padrão MPEG-4 para prover conteúdo interativo mais complexo. A proposta dos autores é tentar unir o melhor dos dois padrões para oferecer serviços mais complexos. Como estudo de caso os autores desenvolveram uma plataforma capaz de tratar conteúdo baseado no padrão MPEG-4 e no padrão MHP. Em seu experimento a plataforma desenvolvida é dividida em duas partes lógicas: conteúdo e transmissão. Na figura 21 podemos observar a arquitetura do experimento proposto.

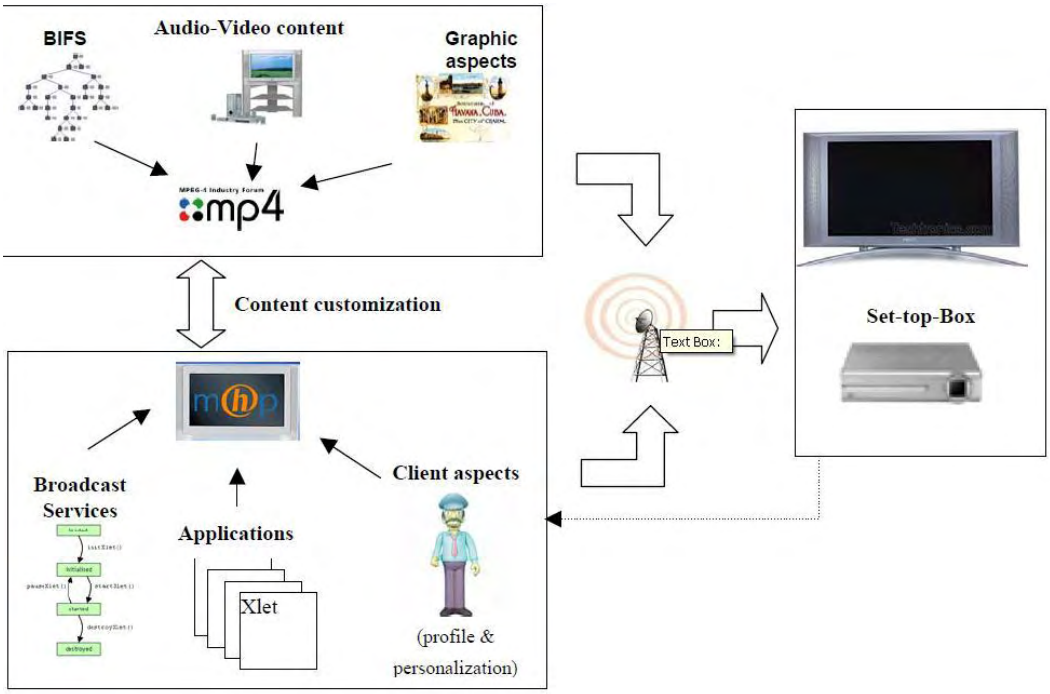


Figura 21. Arquitetura sproposta por [PULLES e SASNO, 2004] para integração entre os padrões MPEG-4 e o MHP.

A parte de conteúdo é tratada pelas funcionalidades do MPEG-4. Esta parte está relacionada ao tratamento do áudio, vídeo e cenas descritas com a linguagem BIFS. A parte de transmissão é gerenciada pelo MHP. O MHP gerencia os eventos de transmissão, sistemas de informação, as aplicações que são enviadas pela emissora, além dos eventos de entrada que são recebidos via controle remoto. Baseado nesta arquitetura e considerando o MHP como

elemento de controle é possível manipular o conteúdo transmitido e customizá-lo dependendo da necessidade do usuário.

O trabalho de BOYLE et al. (2001) utiliza a mesma estratégia para a geração de cenas tridimensionais. O autor se utiliza dos benefícios do padrão MPEG-4 e a linguagem de descrição de cena BIFS para gerar conteúdo 3D. O foco do trabalho, porém é a utilização do padrão MPEG-4 para a reconstrução de cenas 3D realistas. Esta reconstrução leva em consideração a movimentação das câmeras de TV e o movimento dos personagens no ambiente. O autor utiliza como estudo de caso a reconstrução ambientes 3D relacionados a eventos esportivos [MALERCZYK, 2003]. O trabalho em questão faz parte de um projeto maior chamado PISTE, um projeto Europeu que tem como objetivo estudar o processo de criação, transmissão e recepção de conteúdo interativo durante transmissão de eventos esportivos. Na figura 22 podemos observar a reconstrução tridimensional de algumas cenas reais que serão transmitidas aos telespectadores.

Figura 22. Cenas 3D reconstruídas a partir de cenas reais. Após a reconstrução estas informações são enviadas para os telespectadores [MALERCZYK, 2003].

Existem outros projetos que tem como foco a utilização de MPEG-4 para a geração de conteúdo interativo mais complexo. O projeto *CustomTV*[12] analisou a viabilidade de utilizar o padrão MPEG-4 para a customização e transmissão de conteúdo interativo. O projeto *Sambits* propôs uma plataforma que integra o MHP, os padrões MPEG-4 e MPEG-7, além de suporte a HTML [ILLGNER e COSMAS, 2001]. De fato, existem alguns esforços foram feitos para analisar formas de utilizar o MPEG-4 com o objetivo de trabalhar com conteúdo interativo mais avançado, principalmente em conjunto com a plataforma MHP do padrão europeu. O que se percebe, porém, é que o MHP não adotou esta estratégia, possivelmente pela falta de maturidade e robustez do MPEG-4 na época das pesquisas [CESAR, 2005].

2.5.3 Extensão do Middleware de TV para Integração de Tecnologias 3D

Outra estratégia para a integração de tecnologias 3D ao ambiente de TVDI está relacionada a extensão dos padrões de *middleware* existente para suportarem o desenvolvimento e execução de aplicações tridimensionais. Esta estratégia permitiria que conteúdo 3D fosse apresentado aos telespectadores através de aplicações tridimensionais. Através desta integração um número enorme de possibilidades de aplicações interativas surge em diversas áreas.

O trabalho de CESAR (2005) analisa esta estratégia do ponto de vista de um novo nível de interatividade dentro do ambiente de TV. O autor também propõe uma arquitetura para a plataforma de execução destas aplicações, baseada no padrão Europeu de TV Digital. O autor analisa também as principais APIs gráficas existentes e propõe a inclusão de uma destas APIs no conjunto de bibliotecas que seriam utilizadas pelos desenvolvedores de aplicações interativas para a geração de conteúdo 3D. Outra característica abordada pelo trabalho é a infra-estrutura do ambiente nativo onde estas aplicações serão executadas. Especificamente, o autor analisa as funcionalidades que seriam necessárias agregar ao *CommonCore* do *middleware* para viabilizar a execução das aplicações. Como estudo de caso o autor propõe duas plataformas de TV baseadas em sua pesquisa: a OTADIGI e o UBIK.

[12] *CustomTV* - http://www.custom.tv/

A plataforma OTADIGI [CESAR, 2005] foi implementada com o objetivo de oferecer suporte a execução de aplicações DVB-J, a linguagem procedural do MHP, além de permitir o uso de uma linguagem de descrição de cena de alto nível: o SMIL [HOSCHKA, 2001]. Uma das características do Otadigi era permitir que os usuários executassem aplicações que utilizavam o canal de retorno para interação. O objetivo era explorar ao máximo o potencial interativo que o padrão Europeu havia definido. Este estudo de caso foi o primeiro passo para analisar a possibilidade de incrementar o nível de interatividade do conteúdo transmitido. A plataforma em questão suportava o tipo de interação, até então mais avançado, dentro do MHP. Algumas aplicações [PENG et al., 2001] [HERRERO et al, 2003] foram desenvolvidas para este ambiente como podemos observar na figura 23 e figura 24

Figura 23. Player SMIL executando uma aplicação de teletexto Artigo [LAMADON et al., 2003].

Figura 24. Aplicação DVB-J sendo executada na plataforma OTADIG [HERRERO et al, 2003].

A outra plataforma desenvolvida por [CESAR, 2005] foi o UBIK [CESAR et al, 2006]. Esta plataforma foi desenvolvida baseada nos estudos sobre a extensão do *middleware* para TV de forma a prover suporte a execução de aplicação tridimensionais no receptor de TV. Para tanto, o autor propôs uma plataforma que incluía uma API gráfica para o desenvolvimento de aplicações tridimensionais, além de suporte nativo a execução destas aplicações. O UBIK utilizava a API *OpenGL* para a construção das aplicações. Na camada nativa a plataforma utilizava *SDL* para gerenciar o contexto das aplicações e DirectFB[13] para apresentar a cena resultante para o usuário.

O objetivo do UBIK era avaliar os requisitos necessários para gerar cenas 3D em um ambiente de TV. O autor analisou recursos de memória e processamento necessários para executar o ambiente e comparou os resultados com os dispositivos existentes à época [CESAR et al, 2006]. Em relação aos níveis de interação do usuário, o autor propôs um novo nível visto que as aplicações 3D ampliavam o nível de interatividade entre o telespectador e a TV. Ao invés de interagir com interfaces 2D o usuário tinha agora a possibilidade de manipular interfaces 3D, executar jogos no ambiente de TV, navegar por ambientes virtuais, etc. Na figura 25 podemos observar alguns dos testes executados pelo autor utilizando a plataforma UBIK.

Figura 25. Execução de aplicações 3D em um ambiente de TVDI utilizando o UBIK. À esquerda um objeto gráfico no vídeo, à direita o jogo Tux Racer [CESAR, 2005].

[13] *DirectFB* - http://www.directfb.org/

3 GINGA3D – EXTENSÃO DA ARQUITETURA DO MIDDLEWARE GINGA

3.1 INTRODUÇÃO

Um dos fatores que torna a TV Digital um atrativo é a excelente qualidade de som e imagem. De fato, ela nos oferece esse avanço, mas as inovações agregadas a esse novo paradigma não se resumem apenas a alta definição. Também temos a questão da multiprogramação, guia eletrônico de programação, mobilidade e portabilidade; ou seja, passa a ser possível, por exemplo, assistir TV com um celular dentro de um ônibus em movimento. Além da interatividade que se apresenta como um dos principais atrativos e, ao mesmo tempo, desafios do sistema brasileiro de televisão digital terrestre.

O suporte a tecnologias 3D em um ambiente de TVDI amplia o conjunto de possibilidades de entretenimento, interatividade e como conseqüência, novas possibilidades de negócios. No contexto do sistema brasileiro de TVDI estas tecnologias podem auxiliar o governo e a iniciativa privada a alcançar os objetivos traçados pela instituição do SBTVD no nosso país. Em termos de governo tais tecnologias podem suportar a construção de ambientes virtuais que auxiliem na educação à distância, desenvolvimento cultural e disseminação de informação de forma mais atrativa. No âmbito da iniciativa privada uma das áreas que pode se beneficiar destas tecnologias é o mercado de jogos e derivados. Outro fator que pode ser explorado por ambas as iniciativas (pública e privada) é a convergência tecnológica, através da construção e adaptação de plataformas virtuais que venham a oferecer múltiplos meios de acesso.

Na seção 2.1 pode-se constatar a utilização de tecnologias 3D em diversas áreas, quais sejam: realidade virtual, visualização de dados, educação, entretenimento entre outras. Na seção 2.1.2 foram relatados os esforços para padronização e integração de tecnologias 3D e sistemas embarcados, bem como o *status* atual das pesquisas nesta área. A convergência destas áreas, os padrões advindos destas pesquisas e as API’s gráficas construídas em função desta padronização são o ferramental utilizado para criação de uma arquitetura que permita a execução de sistemas 3D em um ambiente de TVDI.

A seção 2.5 apresentou alguns dos esforços feitos com o objetivo de integrar sistemas 3D e Televisão Digital e Interativa. Como podemos observar nesta seção, a maioria dos trabalhos possuem um foco específico em certos tipos de aplicações não oferecendo uma especificação de propósito geral [MAAD, 2003a] [KAMMANN, 2005] [PULLES e SASNO, 2004]. O trabalho de Cesar (2005), por sua vez, tem como objetivo principal oferecer uma arquitetura genérica para o suporte a execução de sistemas 3D, bem como oferecer a infra-estrutura necessária para o desenvolvimento de tais sistemas. Esse trabalho foi desenvolvido baseado no padrão europeu de TV Digital.

O presente trabalho analisou e desenvolveu estratégias para a integração de tecnologias 3D ao ISDB-Tb com o objetivo de prover uma arquitetura que suporte a construção e execução de sistemas tridimensionais para este ambiente. Nesse sentido a arquitetura do *middleware* brasileiro de TV (Ginga) foi analisada e estratégias baseadas nesta especificação para integrarmos tais tecnologias foram desenvolvidas. Desta análise, surgiu a proposta de uma arquitetura que explora os ambientes procedural e declarativo do *middleware* a fim de verificar a viabilidade de incorporação de tecnologias 3D através de uma destas duas camadas do Ginga.

3.2 INTEGRAÇÃO DE TECNOLOGIAS 3D AO MIDDLEWARE GINGA

Como apresentado na seção 2.2.3.1, o *middleware* Ginga é composto por uma *engine* de apresentação (Ginga-NCL) e uma *engine* de execução (Ginga-J). Estes dois módulos do *middleware* permitem a execução de aplicações baseados em dois paradigmas diferentes de concepção e desenvolvimento de aplicações. O Ginga-NCL utiliza uma linguagem declarativa enquanto o Ginga-J utiliza uma linguagem procedural. Neste sentido, integrar tecnologias 3D ao *middleware* brasileiro significa encontrar meios de oferecer suporte a concepção e desenvolvimento de aplicações 3D baseado em uma destes dois paradigmas, ou mesmo em ambos. Porém a utilização de linguagens procedurais e declarativas possuem suas facilidades e dificuldades.

O Ginga-J utiliza a linguagem de programação Java e suas APIs específicas para TV no desenvolvimento de aplicações. Na adoção de uma estratégia de integração que contemple a *engine* de execução as APIs 3D devem ser baseadas nesta linguagem. Em termos de

desenvolvimento, a linguagem Java é de fácil aprendizagem e bastante difundida atualmente nas comunidades de desenvolvedores. Aplicações desenvolvidas no Ginga-J facilitam a utilização de camadas de negócio, de forma que aplicações como jogos e outros tipos de aplicações interativas se encaixam muito bem. Outra característica interessante é o suporte em termos de APIs gráficas ao desenvolvimento de aplicações. Já existe uma série de bibliotecas escritas em linguagem Java com o propósito análogo ao pretendido com este trabalho, como por exemplo, as APIs gráficas para dispositivos móveis. Um fator negativo relacionado à utilização da linguagem Java é a sincronização de mídias. Neste ponto, embora existam APIs Java para a sincronização de mídia, este suporte não é sofisticado, de forma que o desenvolvimento de aplicações com foco na descrição e sincronização de múltiplas mídias não é uma tarefa fácil.

O Ginga-NCL, por sua vez, utiliza a linguagem NCL (*Nested Context Language*), que tem como principal característica mecanismos que permitem a sincronização de múltiplas mídias. Esta facilidade permite a apresentação de objetos de mídias 3D baseado no fluxo de vídeo da emissora, ou mesmo a sincronização destes objetos com outras mídias que, eventualmente, estejam sendo apresentadas. A utilização da linguagem NCL também facilita a apresentação de ambientes tridimensionais baseados em linguagens de descrição de ambientes como o X3D e o VRML. As aplicações 3D construídas neste contexto teriam um foco na apresentação de conteúdo gráfico, porém um mínimo de interatividade também é possível. Outra característica interessante da linguagem NCL é sua capacidade de extensão, de forma que a inclusão de descritores para objetos de mídia 3D é facilitada. Uma dificuldade advinda da utilização do Ginga-NCL está relacionada à construção de aplicações com uma camada de negócio mais robusta, visto que a linguagem NCL não permite de forma simplificada a incorporação de uma lógica de negócio a suas aplicações. Para minimizar o impacto deste fator, a norma brasileira permite a utilização da linguagem Lua em conjunto com a linguagem NCL, de forma a permitir o desenvolvimento de aplicações que tenham foco no negócio.

A integração de tecnologias 3D a sistemas de TVDI pode ser analisada em função de dois outros contextos que apresentam limitação: hardware e software. Em termos de hardware, sistemas de TVDI são sistemas embarcados com baixa capacidade de processamento e recursos limitados para o armazenamento de dados, tanto em memória

quanto em disco rígido. Uma vez que aplicações 3D requerem um pouco mais de poder de processamento, é inviável sua utilização em ambientes com configurações de hardware tão restritas. Um fator, porém, que vem contribuindo para a disseminação de tais aplicações neste ambiente é a evolução do hardware para sistemas embarcados, além do desenvolvimento de chips gráficos específicos para estas plataformas como pôde ser observado na seção 2.1.2.

Em termos de software, como discutido no início do capítulo, o padrão brasileiro não oferece suporte em sua norma à execução de aplicações 3D, dificultando assim o desenvolvimento, execução e comercialização de aplicações desta natureza. A expansão da norma para permitir o suporte a este tipo de aplicação deve ser baseada na extensão da arquitetura do *middleware*. Neste sentido, a arquitetura do Ginga especifica que todos os serviços básicos sejam providos pelo núcleo comum (*Ginga Common Core*) que faz o interfaceamento com o sistema operacional. As camadas que estão acima do núcleo comum utilizam os serviços providos por esta para desempenharem seus papeis. Seguindo esta metodologia, para que fosse possível dar suporte à execução de aplicações 3D nas camadas superiores dos ambientes de execução e apresentação, foi necessário prover este suporte no núcleo comum. Feito isto, foi possível adotar estratégias de desenvolvimento de aplicações baseada na *engine* de execução (linguagem procedural) e na *engine* de apresentação (linguagem declarativa). A arquitetura especificada, conhecida como Ginga3D, é baseada no modelo arquitetural em camadas utilizado pelo próprio *middleware* e estende o Ginga de forma a prover suporte a aplicações 3D no Ginga-J e no Ginga-NCL.

3.3 O GINGA3D

O Ginga é composto por uma série de módulos que tem por objetivo prover serviços de decodificação de áudio, vídeo e dados. Além de módulos básicos para os serviços citados, o Ginga possui a capacidade de executar e gerenciar aplicações de TVDI através dos módulos de gerenciamento e execução de aplicações. Com o advento do Ginga3D o *middleware* abrirá a possibilidade de desenvolvimento e execução de uma nova classe de aplicações tridimensionais, como jogos ou objetos de mídia mais realistas, criando um novo nicho de mercado dentro desta área.

Basicamente, o Ginga3D atua nas diversas camadas do *middleware* oferecendo a abstração necessária tanto para o usuário desenvolvedor de aplicações 3D quanto para o usuário que irá utilizar as aplicações. Na figura 26 é apresentada a atual arquitetura do *middleware* Ginga com a inclusão dos componentes do Ginga3D em suas diversas camadas. Esta foi a configuração utilizada para a integração deste tipo de tecnologia ao *middleware*.

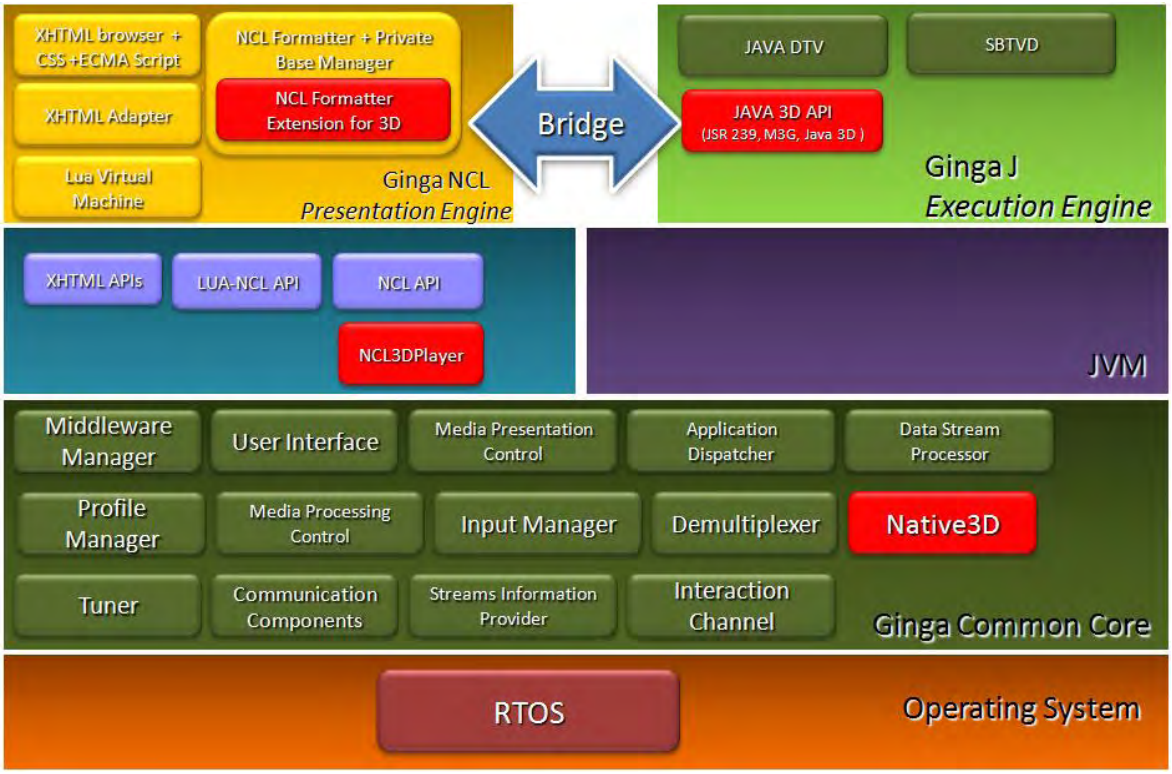


Figura 26. Arquitetura do *middleware* Ginga com adição dos módulos (em vermelho) da especificação Ginga3D.

Na camada do *Ginga Common Core* encontra-se um módulo nativo responsável pela renderização dos objetos e cenas gráficas. Este módulo é responsável pela integração dos módulos das camadas superiores oferecendo um meio bem definido de transação de informações entre as camadas para a renderização dos objetos gráficos bem como o tratamento de eventos. Nas camadas mais acima foram adicionadas APIs utilizadas por desenvolvedores tanto no Ginga-J (*Engine* de Execução) quanto no Ginga-NCL (*Engine* de Apresentação). Na camada intermediária à *Engine* de Apresentação encontra-se o *player* 3D que será executado quando objetos de mídia 3D forem descritos em documentos NCL. Este *player* segue as especificações da norma que define a utilização de *players* para a execução de

mídias descritas pela linguagem NCL [ABNT NBR 15606-2, 2008]. O módulo 3D para a *Engine* de Apresentação ainda estende a API de formatação do NCL para permitir a inclusão de nós de mídia 3D sincronizados com as demais mídias suportadas pelo padrão. Na parte do Ginga-J foi adicionada API 3D OpenGL ES, mais especificamente seu *bind* para a linguagem Java (JSR 239). Esta API foi escolhida, devido a sua popularidade, esforços das empresas para fortalecimento deste padrão, bem como o seu foco em sistemas embarcados [KAMEYAMA et al., 2003].

O Ginga3D se comunica com o hardware específico através do próprio *middleware*, de forma que é transparente ao componente das camadas superiores como as bibliotecas nativas tratam as rotinas de renderização gráfica em nível de SO. Como é de costume em sistemas embarcados, uma camada de abstração de hardware (HAL), sobre a qual o *middleware* é implementado, sempre é utilizada. Isto permite que os fabricantes de *middleware* possam utilizar quaisquer tecnologias para desenvolver os módulos de renderização 3D baseado na arquitetura proposta.

3.3.1 Módulo 3D Nativo

O *Ginga Common Core*, uma das camadas do *middleware* Ginga, tem como propósito oferecer uma camada de código nativo onde as funcionalidades do *middleware* sejam implementadas baseadas em um determinado *hardware*. A esta camada foi adicionado o módulo 3D nativo chamado Native3D. Neste módulo é que estão descritos o gerenciador de contexto (*ContextManager*) e os exibidores de mídia que são definidos no sub-módulo *Renderer*. Na figura 27 é apresentada a arquitetura do módulo Native3D.

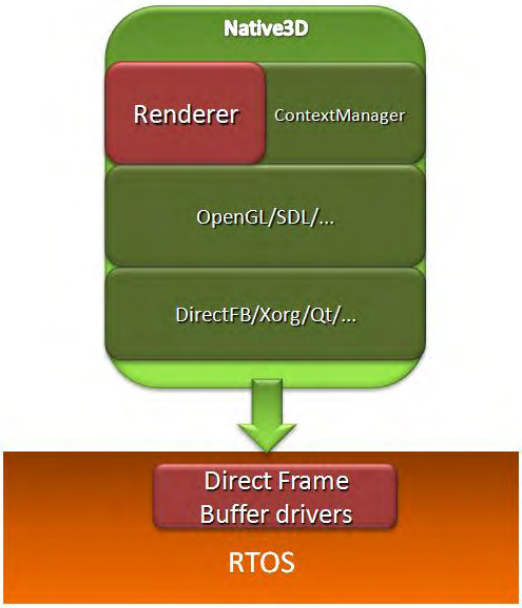


Figura 27. Native3D: módulo responsável pela execução nativa das funções de renderização.

O gerenciador de contexto (*ContextManager*) tem por finalidade gerenciar as aplicações tridimensionais que estarão em execução no *middleware* garantindo acesso unificado ao contexto nativo do ambiente de renderização. Este módulo armazena as propriedades relacionadas à determinada aplicação. O gerenciador de contexto exerce papel fundamental em quaisquer bibliotecas 3D, uma vez que, além de armazenar os dados de uma determinada aplicação, como por exemplo, matrizes de transformação, ele define como o ambiente virtual será desenhado na janela gráfica. Estas propriedades devem ser passíveis de armazenamento e recuperação quando da sobreposição de aplicações que estão sendo executadas pelo *middleware*. Esta característica é importante, uma vez que podemos parar uma determinada aplicação e reiniciá-la a partir do estado que entrou em espera.

Os usuários do gerenciador de contexto são componentes das camadas acima do *Ginga Common Core*. Estes componentes solicitam o contexto das aplicações e o gerenciador de contexto retorna um ponto de acesso para este contexto. Em relação à manipulação do gerenciador (criação, destruição e tratamento de erros), esta é feita pelo próprio módulo nativo. Caso algum problema ocorra na criação ou manipulação do gerenciador de contexto o módulo Native3D retorna informações de erros para os componentes das camadas acima. Uma vez que se está trabalhando sobre uma plataforma onde a execução de aplicações não é o

principal evento, este tipo de tratamento de erros é essencial para não prejudicarmos a experiência de reprodução de mídias ao usuário.

O sub-módulo *Renderer* tem por finalidade encapsular renderizadores nativos tais como *browsers* ou exibidores de mídia 3D. Este sub-módulo oferece um ponto de acesso para que as APIs em alto nível (NCL e Java) possam acessar suas implementações nativas. O *Renderer* gera as cenas tridimensionais através de APIs nativas como OpenGL, Direct3D, SDL e etc. As funcionalidades destas APIs são acessadas através de um ponto comum que seria o gerenciador de contexto. O *Renderer* responde as chamadas das APIs em alto nível, de forma a executar a tarefa solicitada pelo método em alto nível. Para tanto, ele foi construído de tal forma que permite a sua especialização para as diversas APIs que venham a ser integradas aos componentes em camadas superiores. No caso dos exibidores de mídias 3D necessários aos *players* NCL, o *Renderer* oferece acesso a implementações de *browser* que irão renderizar os ambientes virtuais descritos por linguagens de descrição como o X3D e o VRML.

Quanto à comunicação entre o *Renderer* e os componentes da camada superior, estes solicitam uma determinada tarefa através de um protocolo específico. A especialização do *Renderer* para aquele componente em alto nível executa a tarefa específica, através de uma interface definida para os renderizadores baseados na especificação do *Renderer*. Tal interface é composta basicamente pelos métodos *play*, *pause*, *stop* e *destroy* especificados pela própria norma Ginga para a construção de player de mídia. Caso não haja implementação de um determinado player solicitado pelas camadas superiores um erro é lançado para o componente informando que o tipo de mídia 3D não é suportado.

Na base da arquitetura do módulo Native3D encontram-se as APIs responsáveis pela renderização das cenas tridimensionais. Os desenvolvedores de *middleware* têm a responsabilidade de definir as bibliotecas e padrões que irão utilizar, baseados na conveniência e plataformas de hardware que utilizam. Na camada de mais baixo nível do componente são mantidas as referências ao sistema gráfico utilizado para apresentar as cenas. O sistema gráfico também é uma escolha dos fabricantes de *middleware*. Alguns exemplos de

sistemas que podem ser utilizados são o DirectFB, X Window System[14], a plataforma Qt[15] entre outros.

3.3.2 Players 3D para a Engine de Apresentação

Na camada intermediária do Ginga estão as APIs responsáveis pelos serviços específicos do *middleware*. As APIs Lua, NCL e de XHTML são algumas das entidades contidas nesta camada. Com a adoção do Ginga3D o módulo NCL3DPlayer (ver figura 28) será adicionado a esta camada. Este módulo tem por objetivo executar as mídias 3D que foram solicitadas pela *engine* de apresentação através de comandos pré-definidos pela API NCL. O Componente NCL3DPlayer é composto por um player 3D genérico. Este player genérico é especializado para o tipo de linguagem de descrição de ambiente que se deseja suportar através Ginga. Por exemplo, caso desejássemos que o Ginga fosse capaz de renderizar ambientes X3D deveríamos ter um exibidor X3D que quando solicitado pela *Engine* de Apresentação do Ginga fosse instanciado e utilizado para renderizar as cenas descritas neste formato

Figura 28. Arquitetura do módulo NCL3DPlayer.

O sistema oferece uma interface para um *player* 3D que a API NCL irá utilizar para controlar os exibidores de mídia. Este *player* possui as funções básicas especificadas nas normas ABNT [ABNT NBR 15606-2, 2008] [ABNT NBR 15606-5, 2008]. Este componente

[14] *Xorg Foundation* - http://www.x.org/

[15] Qt - http://qt.nokia.

permite que vários tipos de exibidores sejam utilizados oferecendo um acesso único a estes exibidores. O formatador NCL que está na camada mais acima irá utilizar este *player* para executar chamadas padrões que estão definidas na própria norma (*start*, *stop*, *resume* e etc). Estas chamadas devem ser mapeadas de forma coerente para os exibidores de mídias 3D que deverão interpretar de forma específica cada um dos comandos.

A *engine* de apresentação invocará o *player* 3D através de algum dos métodos pré-definidos pela API NCL. O *player* por sua vez acessa uma implementação de exibidor para aquele determinado tipo de mídia 3D. Uma vez que acessou este exibidor, ele utilizará o mesmo para executar as ações solicitadas pela *Engine* de Apresentação baseado nos comandos presentes no documento NCL que está sendo interpretado pelos formatadores da linguagem. Como pré-requisito para a perfeita execução do ambiente tridimensional descrito no documento NCL, deverá haver uma implementação de exibidor compatível com a mídia desejável. Não é necessário que os exibidores de mídias 3D estejam em um mesmo contexto no nível do componente nativo, uma vez que a norma permite que mais de um *player* seja executado em paralelo para exibir mídias específicas. Porém, uma vez que mais de uma mídia 3D esteja descrita em um mesmo documento NCL elas devem estar inseridas em um mesmo contexto em nível de aplicação declarativa.

Além de uma interface para um *player* 3D este módulo é composto por um adaptador de mídias 3D que tem por objetivo oferecer uma camada de abstração em relação ao tipo de *player* 3D que será executado. Com a utilização destes adaptadores o sistema é capaz de integrar vários exibidores de mídias 3D diferentes. Poderíamos integrar por exemplo mídias X3D [Brutzman, 2007] ou O3D [Google, 2009]. A especificação de um adaptador abstrato nos dá um ponto de comunicação unificado entre estes exibidores e o *player* 3D permitindo a especificação de um conjunto básico de funções que alinhem os métodos definidos no player e os métodos necessários nos exibidores. Os adaptadores de mídias 3D devem partir desta especificação abstrata e se especializarem para cada tipo de exibidor.

Quanto ao funcionamento deste componente, o *player* 3D requisitará um adaptador específico indicando o tipo de mídia 3D que deseja executar. O método fábrica irá retornar um adaptador específico para aquele tipo de mídia. Os comandos enviados pelo interpretador do documento NCL serão repassados para o adaptador que se encarregará de mapeá-los em

comandos equivalentes no exibidor 3D. Caso não haja uma implementação para aquele tipo de mídia 3D o método responsável pela construção dos adaptadores deverá retornar um código de erro definido ao *player*.

3.3.3 Hierarquia de Superfícies

Quanto à apresentação das cenas 3D, o Ginga3D segue a mesma estratégia que outras bibliotecas gráficas seguem atualmente a exemplo do Havi [16]. Basicamente, estas bibliotecas entendem que existem três tipos de superfícies que compõe a cena: o *background*, a superfície de vídeo e a superfície de gráficos. Na figura 29 é possível observar o esquema de composição da cena através das superfícies combinadas. Neste esquema de composição através de superfícies a cena final é gerada em função de uma lógica baseada em uma hierarquia de superfícies. Como é possível observar na figura 29 a superfície *background* é a de mais baixa hierarquia, logo só é apresentada caso não exista nada sobrepondo a região da mesma. A superfície de vídeo tem prioridade sobre o *background* e a superfície gráfica é a de mais alta prioridade, de forma que irá sobrepor todas as outras quando houver alguma aplicação gráfica. Outra característica desta estratégia é que só existe uma superfície *background*, um número determinado de superfícies de vídeo, enquanto podemos ter várias superfícies gráficas. Na figura 30 podemos observar o esquema de composição de um vídeo com uma aplicação gráfica, além do grafo de cenas que pode ser criado, exemplificando a capacidade de criação de múltiplas superfícies como foi citado anteriormente.

[16] *HAVi (Home Audio Video Interoperability) - www.havi.org*

Figura 29. Esquema de superposição de superfícies para a composição da cena final que será apresentada ao usuário.

Figura 30. Esquema de composição de cenas baseado em hierarquia de superfícies.

3.3.4 Interatividade no Ginga3D

A interatividade é outra característica proveniente de um ambiente de TVDI que deve ser suportado pelo Ginga3D. Este suporte se dá através do tratamento dos eventos de entrada que são especificados para um ambiente de TV. Em sua forma mais simplificada esta interação é feita através da utilização de controle remoto que possui uma série de botões que um teclado convencional não possui. Estes botões são mapeados de forma que o Ginga3D possa entender os eventos e tratá-los quando necessário. Em termos de interação com as

aplicações 3D, a utilização de controles remotos será similar ao uso do teclado para interação com aplicações 3D em PCs.

A utilização de um gerenciador de eventos é uma boa estratégia para o tratamento dos comandos enviados pelo usuário. Algumas implementações de middleware, como por exemplo, o OpenGinga, utilizam este tipo de recurso. Para tanto um novo componente é incluído ao núcleo comum do middleware: o *InputManager*. O *InputManager* faz o papel de gerenciador de eventos. A principal função deste componente é delegar aos módulos e aplicações nativas interessadas os eventos que estão sendo enviadas ao usuário. Basicamente, os outros módulos do *middleware* devem ser registrar a este componente e passar um mapa de eventos que desejam tratar. Quando o *InputManager* recebe um determinado evento ele busca em sua fila de prioridade qual entidade do *middleware* receberá o determinado evento. Esta estratégia permite a criação de eventos diversos, de forma que podemos mapear não só os eventos de um controle remoto convencional, mas de outros dispositivos não-convencionais a um ambiente de TVDI.

Associado a estratégia de uso de um gerenciador de eventos temos o fato de o padrão brasileiro permitir a interação com múltiplos dispositivos [SILVA, 2008], de forma que mais de um usuário poderia, por exemplo, interagir em um ambiente virtual, jogo ou mesmo algum objeto gráfico mais simples. O padrão brasileiro possui uma API específica para o tratamento de eventos lançados por dispositivos móveis, mapeando botões e comandos específicos de aparelhos como celulares e PDAs. Além do mapeamento dos eventos, a API de múltiplos dispositivos define mecanismos para a interação por mais de um usuário, de forma a tratar os múltiplos eventos que chegam ao *middleware*. Estas regras a nível de núcleo podem ser implementadas no *InputManager*.

Através da extensão do gerenciador de eventos e expansão do mapa de eventos tratados podemos integrar dispositivos não-convencionais ao Ginga3D. Poderíamos utilizar dispositivos com três graus de liberdade (3 DOF) para interagirmos com as aplicações tridimensionais, por exemplo. O mapeamento destes eventos seria feito pelo *InputManager*. Como estamos lidando com aplicações para TV, o *InputManager* deve delegar os eventos recebidos ao gerenciador de aplicações que tem o controle sobre o estado de cada aplicação. Cabe a cada aplicação tratar ou não os eventos enviados pelo *InputManger* através do

gerenciador de aplicações. Na figura 31 podemos observar o fluxo dos eventos enviados pelos usuários da TV com a expansão da arquitetura do Ginga para a incorporação da especificação Ginga3D. O módulo nativo do Ginga3D deverá se registrar junto ao *InputManager* para tratar os eventos específicos de dispositivos não-convencionais que, por ventura, possam ser utilizados para interagir com as aplicações tridimensionais.

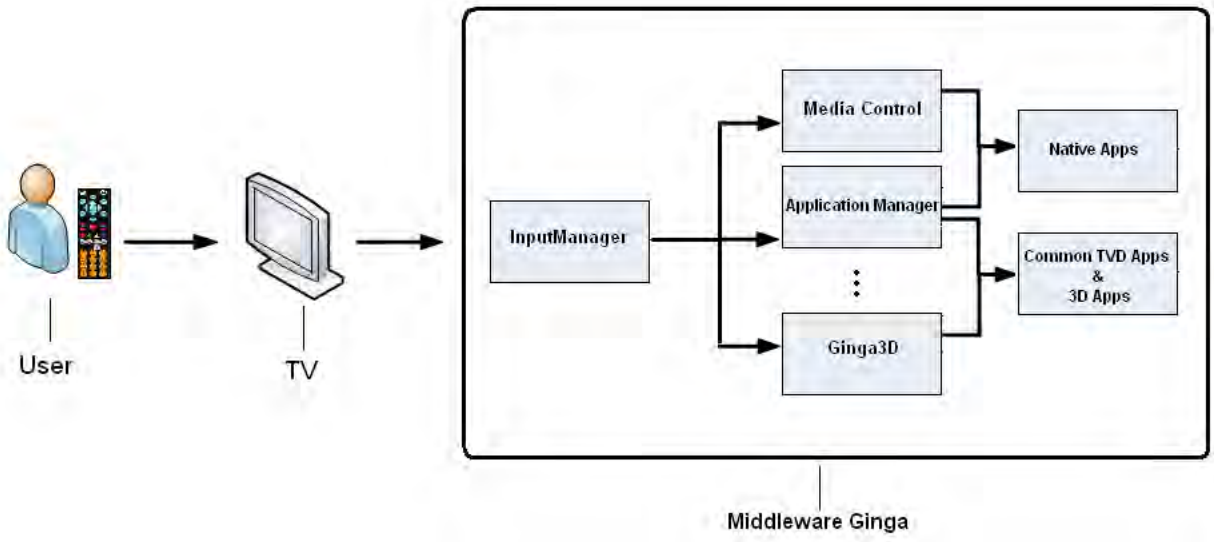


Figura 31. Tratamento de eventos no Ginga3D.

3.3.5 Outras Considerações sobre o Ginga3D

Uma vez que estamos lidando com sistemas tridimensionais o desempenho destas aplicações é um fator chave. O sistema deve possuir requisitos de performance que permitam que o usuário tenha uma experiência contínua de interatividade. Neste sentido um dos indicadores de desempenho que deve ser explorado é a taxa de atualização da cena tridimensional mensurada em quadros por segundo (*frames per second* – FPS). Taxas de atualização da cena gráfica ideais giram em torno de aproximadamente 30 fps, valores muito abaixo disto tornam a experiência de interação do usuário muito reduzida. Outro fator relacionado ao desempenho e experiência de interatividade é o tratamento das entradas pelo usuário e a respectiva resposta por parte da aplicação. Este fator é fundamental e deve ser trabalhado de forma que a interação receba a maior prioridade. O fato que torna esta característica tão importante é a coerência temporal que deve existir entre a cena apresentada e o que o usuário está interagindo.

Alguns fatores que são relevantes quando estamos desenvolvendo sistemas 3D para dispositivos com baixa capacidade de processamento devem ser observados. Estes requisitos não dizem respeito apenas ao Ginga3D, mas sim a todas as aplicações que serão construídas baseadas no mesmo. Neste sentido, vale citar alguma das recomendações descritas por [FADI et al., 2005] sobre construção de aplicações 3D para dispositivos móveis e que são perfeitamente aplicáveis ao ambiente de TVDI:

- Evite utilizar números de ponto-flutuante;
- Utilizar sempre que possível tipo de dados pequenos (short, char e etc);
- Não passar vetores de dados utilizando cópia. Sempre utilizar ponteiros;
- Evitar ao máximo utilizar operações de multiplicação e divisão complexas;
- Minimizar o número de vértices duplicados nos objetos;
- Reduzir ao máximo o número de polígonos;
- Usar, sempre que possível, primitivas simples (triângulos e quadrados);
- Reduzir o tamanho das texturas que serão utilizadas;
- Utilizar ferramentas de compressão de imagem e dados para reduzir o tamanho dos dados;

Por fim, em termos de qualidade implementações do Ginga3D devem ser construídas de forma que facilite o seu reuso. Outros fatores como os requisitos de segurança, privacidade e capacidade de resposta devem levar em consideração as recomendações descritas na norma do padrão brasileiro. Os requisitos de performance também relacionados à qualidade foram discutidos no início desta seção. Os componentes do Ginga3D devem passar por baterias de testes que atestem sua integridade e confiabilidade e, por conseguinte, sua qualidade.

3.4 CONSIDERAÇÕES FINAIS

A fim de estender a atual arquitetura do *middleware* Ginga propomos uma especificação que incorpora a esta arquitetura módulos que possam viabilizar a execução e desenvolvimento de aplicações tridimensionais. Esta arquitetura é chamada de Ginga3D. Seu objetivo é oferecer a capacidade de executarmos aplicações 3D dentro de um ambiente de TVDI, seguindo um padrão bem definido, e integrando APIs gráficas e players para objetos de mídia tridimensionais que possibilitem o desenvolvimento destas aplicações.

Como discutido na seção 3.2, cada um dos paradigmas utilizados por estes dois módulos do *middleware* possuem característica que favorecem ou limitam a sua utilização. Desta forma, a especificação de uma arquitetura que prevê o suporte a sistemas 3D nos dois ambientes oferece a flexibilidade necessária ao desenvolvedor de aplicações para a escolha da tecnologia que deseja utilizar. Esta flexibilização é um dos fatores que diferencia este trabalho dos trabalhos correlatos apresentados. Outro fator interessante é que esta estratégia possibilita a utilização do ambiente declarativo para a especificação de mídias 3D. Todos os trabalhos correlatos apresentados utilizam apenas o ambiente procedural.

4 DESENVOLVIMENTO

A extensão da arquitetura do *middleware* Ginga, introduzida neste trabalho, foi validada através da elaboração de uma prova de conceito, ou seja, construímos um modelo prático para a proposta teórica apresentada no Capítulo 3. Para o modelo prático foram desenvolvidos os componentes propostos na especificação do Ginga3D. A fim de simular um ambiente próximo do ambiente de TV foi utilizado uma implementação de referência do *middleware* Ginga. Conhecida como OpenGinga, tal implementação de referência foi desenvolvida utilizando o conceito de código aberto de forma que, o seu uso e acesso ao código fonte foram facilitados. O modelo prático estende os módulos relacionados à geração de gráficos e tratamento de eventos de entrada, além de incluir as inovações da nova especificação ao OpenGinga. Com o propósito de validar a extensão implementada, foram desenvolvidas algumas aplicações piloto para avaliar requisitos não funcionais como desempenho e interação.

Neste capítulo é apresentada uma descrição do OpenGinga, bem como o modelo de componentes utilizado para o gerenciamento dos seus diversos módulos. Em seguido são discutidos as modificações que foram feitas no OpenGinga a fim de suportar a renderização de primitivas gráficas 3D. Uma vez feitas estas considerações, são apresentados e discutidos os componentes que compõe a arquitetura do Ginga3D, bem como sua integração com o Ginga-J e o Ginga-NCL. Por fim, é apresentado o browser X3D que foi desenvolvido a fim de permitir a apresentação de cenas 3D descritas com esta linguagem.

4.1 OPENGINGA

OpenGinga[17] é uma plataforma que permite executar aplicações Ginga em um computador pessoal. Esta implementação de referência do *middleware* brasileiro foi desenvolvida pelo LAVID/UFPB, sob plataforma Linux. A sua distribuição é livre e junto com o pacote de execução do ambiente estão inclusos o sistema operacional, uma implementação de referência do *middleware* baseada em componentes e aplicações exemplo. A versão utilizada neste trabalho possui suporte apenas as funcionalidades básicas do Ginga-

[17] http://dev.openginga.org/

CC e a execução de aplicações Java baseada. O objetivo foi trabalhar com uma versão básica, fácil de entender e manter do *middleware* e partir daí iniciar o desenvolvimento dos novos componentes do Ginga3D.

O ambiente de execução Ginga-J do OpenGinga utiliza a implementação *Advanced* do projeto phoneME[18] que contém: CDC 1.1.1, *Foundation Profile* 1.1, *Personal Basis Profile* 1.1, *Personal Profile* 1.1 e um gerenciador simples de *xlets*. Este ambiente é originalmente integrado ao DirectFB para possibilitar a execução de aplicações Java sem a necessidade de um servidor X, porém, devido a ausência de uma implementação de uma API 3D para o ambiente gráfico do DirectFB o OpenGinga foi adaptado para execução sobre a plataforma X *Window*. Esta versão do *middleware* utiliza a distribuição Linux Ubuntu 9.10[19] como provedor de serviços de software básico de segurança, gerenciamento de memória, gerenciamento de processos, protocolos de rede e modelo de drivers.

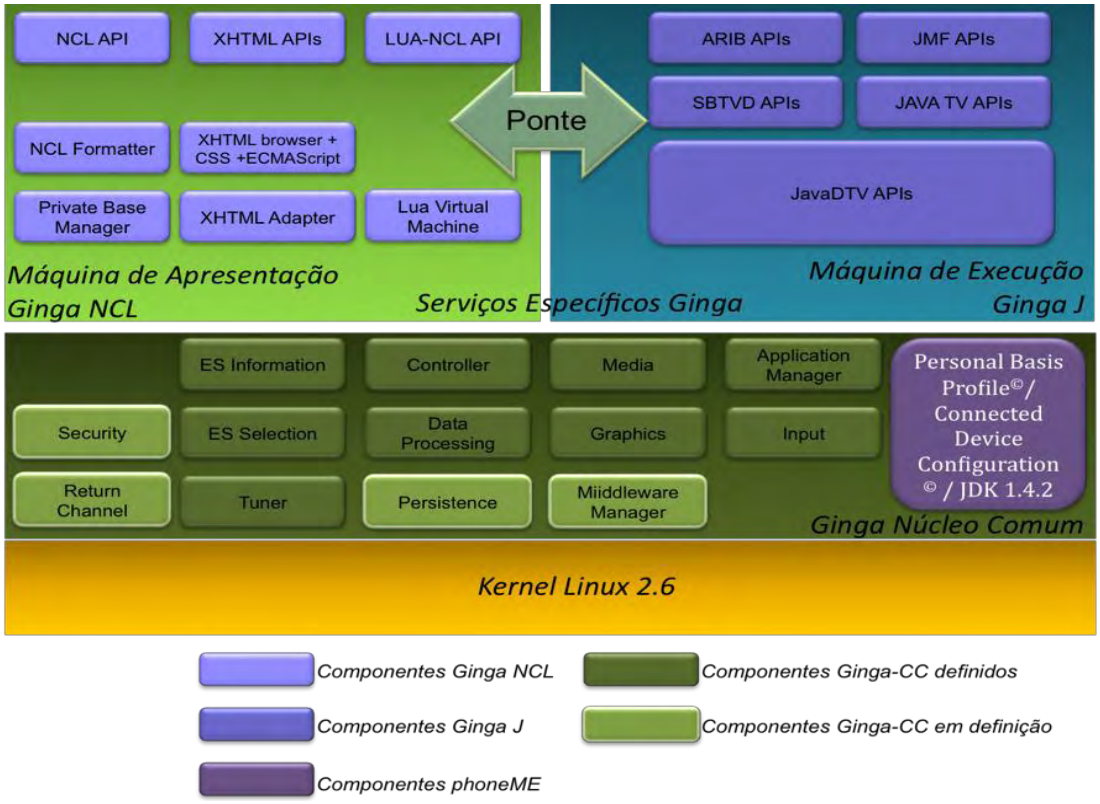


Figura 32. Arquitetura do OpenGinga.

[18] http://phoneme.dev.java.net

[19] www.ubuntu.com

A Figura 32 apresenta a arquitetura para o OpenGinga que foi utilizada no presente trabalho. No caso, a maioria dos elementos do núcleo comum (*Common Core*) já estava definida e possuía uma implementação de referência para a maioria dos componentes. Para os componentes Ginga-J buscou-se fazer o máximo de reuso de código já existente no projeto OpenGinga ou outras implementações de outros projetos disponíveis. Uma característica interessante do OpenGinga é que ele foi desenvolvido baseado no paradigma de orientação a componentes de modo que esta implementação utiliza-se de um modelo de componentes para a comunicação entre as suas diversas partes. Tal modelo foi primordial na análise e desenvolvimento dos novos componentes proposto pelo Ginga3D.

4.1.1 Modelo de Componentes FlexCM

Um modelo de componentes é a especificação que define todos os conceitos envolvidos no desenvolvimento e execução dos componentes. Por exemplo, é o modelo que define como um componente é carregado dinamicamente em memória, como está organizado o ciclo de vida dos componentes, quais interfaces os componentes devem implementar. Já um ambiente de execução é a implementação responsável por gerenciar os componentes de acordo com as especificações definidas pelo modelo. São exemplos de atividades realizadas por um ambiente de execução de componentes: carregar componentes dinamicamente em memória, inicializar componentes, prover implementações para as interfaces requeridas de um componente, etc. pelo menos pra modelo de componentes...

O modelo de componentes FlexCM [MIRANDA FILHO et. al.] foi projetado para ser independente de plataforma de execução e de linguagem de programação. Entretanto, no contexto deste trabalho, foi utilizada a implementação na linguagem C++ do seu ambiente de execução, possibilitando a execução de componentes escritos em tal linguagem. A principal característica do modelo FlexCM é a montagem automática de arquiteturas baseadas em componentes através da representação explícita das conexões entre cada componentes através de um arquivo de descrição arquitetural escrito em XML. Nesta abordagem, os componentes devem declarar explicitamente suas interfaces providas e requeridas. É papel do ambiente de execução decidir qual implementação de uma interface requerida será carregada no sistema. As interfaces base definidas pelo modelo FlexCM estão representadas na Figura 33.

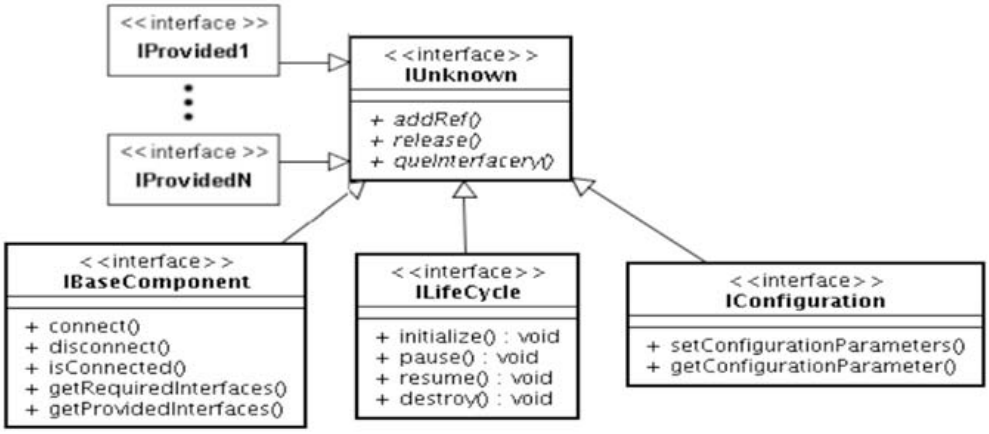


Figura 33. Interfaces base do modelo FlexCM.

As interfaces bases do modelo definem certas características que serão herdadas pelos componentes desenvolvidos. Basicamente, a interface *IBaseComponent* define as funcionalidades necessárias para que um dado componente possa ser conectado a outros componentes dentro do ambiente do FlexCM. A interface *ILifeCycle*, por sua vez, descreve as instruções básicas de cada componente relacionadas ao seu ciclo de vida. Por fim, a interface *IConfiguration* oferece um conjunto de métodos que possibilita a passagem de parâmetros de configuração para um dado componente. Ainda em relação às interfaces, no FlexCM interfaces e implementações são identificados através de GUIDs (*Globally Unique Identifiers*). Estes GUIDs são números de 128 bits gerados de forma a garantir sua unicidade global. Desta forma, um componente pode declarar os GUIDs de suas interfaces providas e requeridas e o ambiente de execução poderá identificar unicamente estas interfaces.

Como dito anteriormente, o OpenGinga utiliza este modelo de componentes, de forma que cada módulo do *Common Core* possui a sua implementação baseada no FlexCM. Cada componente do OpenGinga possui um conjunto de interfaces providas e interfaces requiridas. As interfaces providas indicam os serviços que um determinado componente do *Common Core* tem a oferecer. As interfaces requeridas por sua vez, indicam os serviços necessários para que o componente possa ser carregado e funcionar corretamente. Desta forma, criamos um esquema de dependência baseado nos serviços providos e requeridos por cada componente do OpenGinga. Na Figura 34 podemos observar o esquema de dependências entre os diversos componentes do OpenGinga.

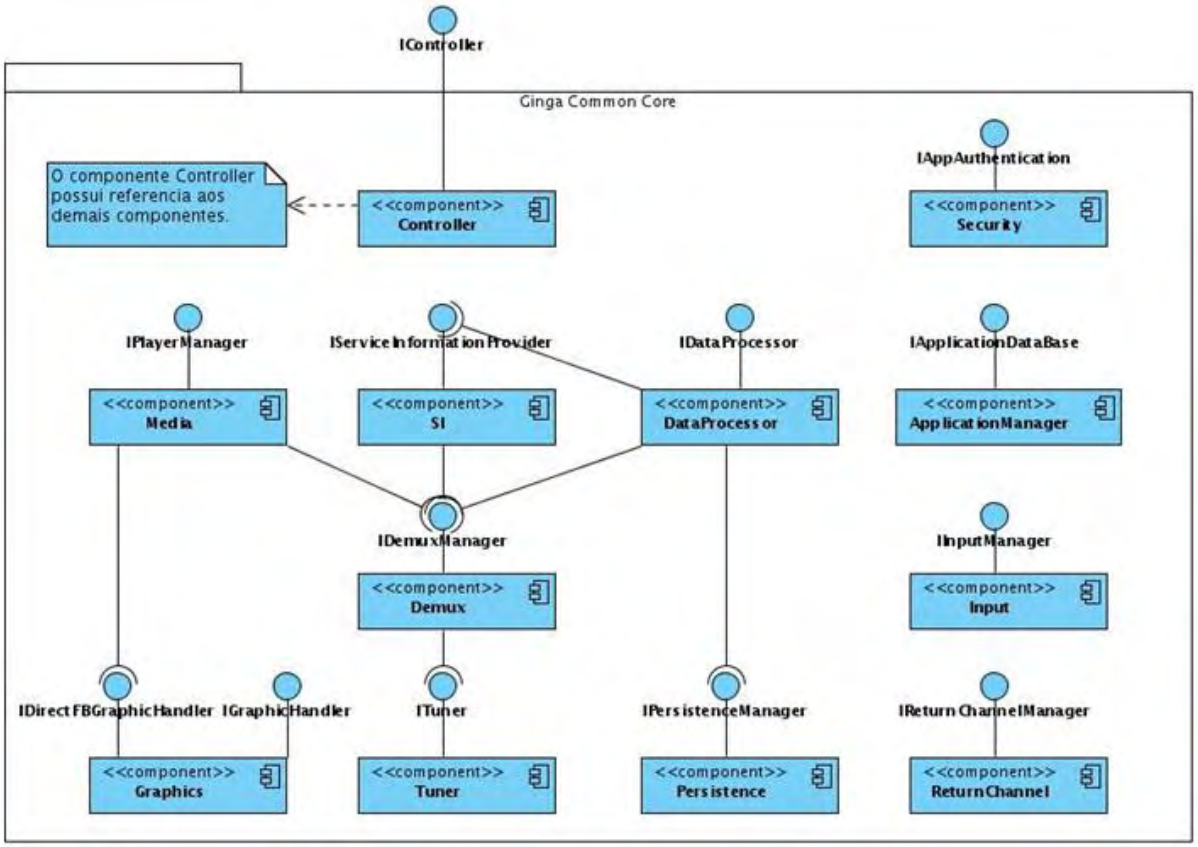


Figura 34. Interfaces base do modelo FlexCM.

Outra característica do FlexCM é a forma como são registrados e definidas as diversas conexões entre os componentes. Para tanto existem dois arquivos base que são utilizados pelo modelo para definição destas características. No arquivo *registry.xml* estão definidos todos os componentes que serão carregados dentro do ambiente do FlexCM. Em sua sintaxe o desenvolvedor de componentes indica informações sobre nome , *alias*, local onde a biblioteca dinâmica está localizada, etc. Na Figura 35 é apresentado um exemplo do arquivo de registros do FlexCM. O segundo arquivo, denominado *architecture.xml*, é responsável por definir a forma como os componentes vão se conectar e as interfaces que serão utilizadas por cada cliente. A sintaxe deste arquivo define os IDs dos componentes que serão conectados, bem como a interface provida que está oferecendo um determinado serviço ao componente cliente. Na figura 36 é apresentado um exemplo de arquivo de arquitetura utilizado pelo FlexCM.

```
<?xml version="1.1" encoding="UTF-8" ?>
<!-- registry components from middleware -->
<registry relative_path="./src">
        <component alias="tuner-ufpb" guid="906ca0a0-5000-11db-b1de-0800200c9a66">
                <path>tuner-ufpb/bin/libtuner.so</path>
                <name>libtuner.so</name>
                <provides>758f082d-8637-4542-abbb-8df5bba94061</provides>
        </component>

        <component alias="demux-ufpb" guid="906ca0a0-5111-11db-b1de-0800200c9a66">
                <path>demux-ufpb/bin/libdemux.so</path>
                <name>libdemux.so</name>
                <provides>855e2320-3bc8-11dd-ae16-0800200c9a66</provides>
                <requires>758f082d-8637-4542-abbb-8df5bba94061</requires>
        </component>

        <component alias="graphics-ufpb" guid="906ca0a0-5333-11db-b1de-0800200c9a66">
                <path>graphics-ufpb/bin/libgraphics.so</path>
                <name>libgraphics.so</name>
                <provides>883f0184-e9c5-4d7a-abb6-6a4f5954d024</provides>
        </component>

        <component alias="media-ufpb" guid="906ca0a0-5222-11db-b1de-0800200c9a66">
                <path>media-ufpb/bin/libmedia.so</path>
                <name>libmedia.so</name>
                <provides>418afb1a-6199-480c-b598-d4cc0069d13b</provides>
                <requires>855e2320-3bc8-11dd-ae16-0800200c9a66</requires>
                <requires>883f0184-e9c5-4d7a-abb6-6a4f5954d024</requires>
        </component>

</registry>
```

Figura 35. Arquivo registry.xml que define os componentes no modele FlexCM

```
<?xml version="1.1" encoding="UTF-8"?>
<!-- dependency mapping from middleware -->
<architecture>
        <!--conectar o componente demux-ufpb ao componente tuner-ufpb
        passando o IID da interface ITuner para o metodo IDemuxManager::connect()
        -->
        <connect guid1="906ca0a0-5111-11db-b1de-0800200c9a66"
                 guid2="906ca0a0-5000-11db-b1de-0800200c9a66"
                 iid="758f082d-8637-4542-abbb-8df5bba94061"></connect>

        <!--conectar o componente media-ufpb ao componente demux-ufpb
        passando o IID da interface IDemuxManager para o metodo IPlayerManager::connect()
        -->
        <connect guid1="906ca0a0-5222-11db-b1de-0800200c9a66"
                 guid2="906ca0a0-5111-11db-b1de-0800200c9a66"
                 iid="855e2320-3bc8-11dd-ae16-0800200c9a66"></connect>

        <!--conectar o componente media-ufpb ao componente graphics-ufpb
        passando o IID da interface GraphicHandler para o metodo IPlayerManager::connect()
        -->
        <connect guid1="906ca0a0-5222-11db-b1de-0800200c9a66"
                 guid2="906ca0a0-5333-11db-b1de-0800200c9a66"
                 iid="883f0184-e9c5-4d7a-abb6-6a4f5954d024"></connect>

</architecture>
```

Figura 36..Arquivo architecture.xml. Define as conexões entre os diversos componentes do *middleware*.

4.1.2 Adaptação dos componentes do OpenGinga

Alguns dos componentes do OpenGinga tiveram que ser adaptados para a construção do modelo prático exigido pela prova de conceito. A maior modificação foi reflexo da mudança do sistema gráfico empregado. A versão original do OpenGinga foi implementada utilizando o sistema gráfico DirectFB. Porém, como discutido anteriormente, este sistema gráfico não possui suporte a API gráficas 3D. Desta forma, a biblioteca gráfica Qt foi utilizada para o desenvolvimento das primitivas gráficas desta nova versão do OpenGinga. O Qt foi escolhido devido à diversidade de funcionalidades oferecidas pela API para a construção de gráficos, sua compatibilidade com o sistema gráfico X *Window* e principalmente, devido a sua integração com a APIs OpenGL e OpenGL ES, facilitando o desenvolvimento do componente 3D.

Uma vez que o sistema gráfico foi modificado, alguns componentes tiveram que ser reimplementados. De fato, todos os componentes que possuíam algum tipo de dependência do sistema de janelas foram modificados, gerando novas versões. Os componentes que foram reimplementados foram:

- ✓ *Graphics;*
- ✓ *Media;*
- ✓ *Input Manager*.

Em relação ao componente *graphics*, todas as funções relacionadas ao desenho de primitivas gráficas foram reimplementadas. Funções de desenho de elipses, quadrados, circunferências e outras relacionadas a preenchimento de primitivas e clipping foram refeitas baseadas na API Qt. Em relação ao componente *graphics*, outra modificação feita foi na estratégia para desenho da cena. Uma vez que o Qt trabalha com sistemas de eventos para todas as suas funcionalidades, foi necessário desenvolver um sistema customizado para envio de eventos de desenho ao núcleo da API, a fim de controlar com mais precisão todas as funcionalidades relacionadas ao desenho de janelas gráficas. Outra característica do Qt é que o mesmo só trata eventos gerados pela *thread* principal do programa, de forma que qualquer *thread* intermediária que gerar eventos para a sua *engine* gráfica, não terão suas solicitações

atendidas. Desta forma, foi necessário criar um esquema para registro de eventos na *thread* principal, a fim de que todas as tarefas solicitadas por quaisquer *threads* do OpenGinga fossem processados. Vale salientar, que o esquema de hierarquia de camadas apresentado na seção 3.3.3 continuou seguindo a mesma estratégia.

O Componente *media*, por sua vez, foi reimplementado a fim de suportar a apresentação do vídeo utilizando janelas gráficas desenvolvidas sobre o Qt. A grande modificação no componente *media* foi a substituição do seu provedor de vídeo. Na versão original do OpenGinga o provedor de vídeo utilizando pelo componente *media* foi o *DirectFB Video Provider*[20], um conjunto de funções oferecidas pelo próprio DirectFB para decodificação e manipulação de vídeo. Como substituto, foi escolhido o *Video Lan*[21] (VLC) como provedor de vídeo padrão para esta nova versão do componente. O VLC foi o escolhido devida a sua facilidade de integração com o Qt, além do enorme conjunto de funções oferecido por sua API de desenvolvimento para a construção de aplicativos de vídeo. O componente *media* instancia um novo *player* VLC sempre que solicitado pelo *middleware*. A cada instância do VLC é associado uma janela gráfica criada através do componente *graphics*.

Em relação ao componente *InputManager*, foi necessário modificar a forma como os eventos nativos eram recebidos pelo mesmo. Sua implementação original utilizava eventos do *DirectFB* para tratar entradas do teclado, mouse e controle remoto. Uma vez que o sistema de janelas gráficas foi modificado, se fez necessário programar um componente *InputManager* que fosse capaz de tratar os eventos enviados pelo Qt. Desta forma, uma nova versão do componente foi desenvolvida utilizando o sistema de tratamento de eventos do Qt para o processamento de eventos vindos dos dispositivos de interação como mouse, teclado e controle remoto. Em relação à forma como os eventos eram delegados a outros componentes do *middleware*, a estratégia continuou a mesma: recebe o evento o primeiro componente que se registrar para aquele dado evento. Uma *thread* associada ao *InputManager* é responsável por capturar quaisquer eventos e delegar a seus respectivos ouvintes.

[20] www.directfb.org

[21] http://www.videolan.org/

4.2 EXTENSÃO DO COMPONENTE *GRAPHICS*

Um dos problemas encontrados em relação à adaptação da arquitetura do Ginga foi o fato do componente *graphics* somente permitir o desenho de janelas gráficas com suporte a desenho 2D. A fim de resolver este problema foi necessário ampliar o conjunto de funcionalidades previstas na arquitetura do Ginga para o componente *graphics*. Originalmente este componente foi concebido com o objetivo de oferecer funções básicas para o desenho de gráficos 2D, bem como, possibilitar a criação de janelas gráficas mais simples, onde estas primitivas deveriam ser desenhadas. Na Figura 37 podemos observar a especificação original do componente *graphics*, suas classes e interface providas para acesso por parte de outros componentes.

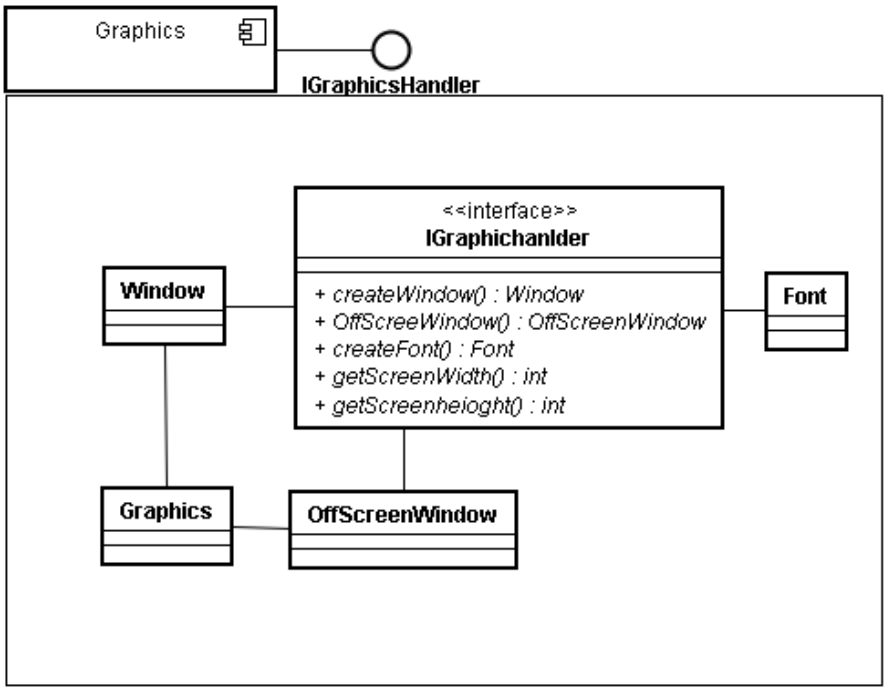


Figura 37. Digrama de classes do Componente Graphics.

A fim de oferecer suporte à construção de gráficos tridimensionais a especificação do componente *graphics* foi ampliada através da adição de novas funcionalidades. As novas funcionalidades dizem respeito à possibilidade de criação de janelas gráficas baseadas no padrão OpenGL, gerenciador de contexto gráfico e um modelo de interface para componentes

que desejam implementar aplicações 3D. As novas classes e seus relacionamentos com as outras classes do componente *graphics* podem ser observadas na Figura 38.

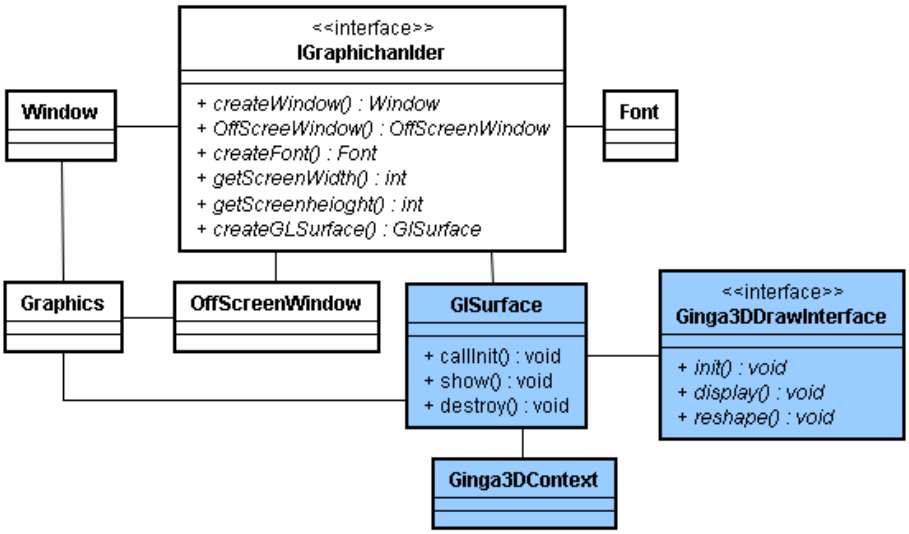


Figura 38. Digrama de classes do Componente Graphics, após adição do suporte a criação janelas OpenGL.

A classe *GLSurface* tem como finalidade oferecer um novo tipo de janela gráfica que seja capaz de renderizar primitivas gráficas 3D. Esta classe é composta de quatro chamadas básicas utilizadas pela *engine* gráfica para invocar aplicações OpenGL. Os métodos *callInit()*, *reshape()*, *show()* e *destroy()* são invocados pela *thread* principal responsável pelo desenho das janelas gráficas dentro do *middleware*. Associado a uma *GLSurface* temos um contexto gráfico, responsável por manter as propriedades relacionadas a uma janela OpenGL como, por exemplo, valores das matrizes de iluminação, propriedades relacionadas a textura e iluminação, listas de visualização, etc. O contexto gráfico associado a uma *GLSurface* está representado através da classe *Ginga3DContext*. Outra característica da *GLSurface* é sua capacidade de registrar diversas interfaces de desenho. Desta forma, é possível que mais de uma aplicação manipule a janela representada por uma *GLSurface*. Internamente a classe *GLSurface* armazena uma janela gráfica Qt, assim como as janelas gráficas convencionais. Porém, esta janela só é acessada através da *thread* responsável pelo desenho.

A interface *Ginga3DDrawInterface* descreve uma série de métodos que devem ser implementados por aplicações que desejem se registrar a uma *GLSurface*, de forma a ter a

capacidade de desenhar primitivas gráficas. Basicamente, existem três métodos principais que as aplicações clientes devem implementar, quais sejam: *init*, *display*, *reshape*. Quando uma *GLSurface* é invocada pela primeira vez pela *engine* gráfica, ela invoca o método *init* de todas as interfaces de desenho associadas a ela. Imediatamente após e durante modificações no tamanho da janela gráfica, ela invoca o método *reshape* de todas estas entidades. O método display é invocado sempre que for necessário desenhar novamente a cena gráfica para àquela *GLSurface*.

Por fim, outra modificação feita ao componente *graphics* foi a ampliação do seu conjunto de serviços providos através da interface *IGraphicHandler*. Originalmente era permitida apenas a criação de janelas gráficas através dos métodos *createWindow* e *createOffScreenWindow*, além da possibilidade de construção de fontes. Com a expansão do componente a possibilidade de criação de janelas OpenGL foi adicionado a interface IGraphicHandler através do método *createGLSurface*. Desta forma, quaisquer componentes interessados, notadamente o componente Ginga3D, tem a possibilidade de criar janelas deste tipo e registrar aplicações 3D através da interface *Ginga3DDrawInterface*.

4.3 O COMPONENTE NATIVE3D

Como apresentado na seção 3.3.1 o módulo Native3D tem por objetivo prover suporte a execução de aplicações 3D dentro do *Common Core* do *middleware* Ginga. Neste módulo foram incorporadas as funcionalidades básicas para o desenvolvimento tanto de aplicações 3D no ambiente do Ginga-J, bem como no ambiente do Ginga-NCL. Seguindo o modelo de desenvolvimento baseado em componentes utilizado pelo OpenGinga, o módulo Native3D foi especificado de tal forma que representasse um componente dentro do *Common Core* do *middleware*, bem como, foi adaptado para ser estar de acordo com o modelo de componentes do FlexCM. A Figura 39 mostra do componente *Native3D* e sua interface de comunicação.

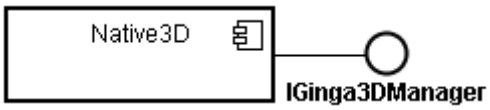


Figura 39. Componente Native3D e sua interface provida IGinga3DManager.

Em termos de comunicação o componente *Native3D* se conecta a dois outros componentes do OpenGinga: o *Graphics* e o *InputManager* (ver Figura 40). O *Native3D* acessa a superfície GL através do componente *Graphics*. O componente utiliza esta superfície para desenhar todas as primitivas gráficas indicadas pelo desenvolvedor de aplicações 3D, quer seja no ambiente do Ginga-J ou Ginga-NCL. Vale salientar que, em nível de *Common Core*, no contexto do *Native3D*, tanto aplicações desenvolvidas em Java como aplicações NCL são tratadas da mesma forma. Isto se dá, devido ao fato do *Native3D* utilizar um único mecanismo para desenho das primitivas gráficas. O que diferencia de fato um ambiente do outro é como este mecanismo é utilizado nas camadas mais acima do *Native3D*. Quanto ao componente *InputManager*, o *Native3D* se registra como um ouvinte dos eventos de entrada do *middleware* no momento que alguma aplicação 3D é executada. Para tanto, o Nat*i*ve3D deve manter uma referência do gerenciador de eventos de entrada no escopo de seu componente. O FlexCM, através do seu mecanismo para conexão de componentes garante a existência desta instância do componente *InputManager* no contexto do *Native3D*. De fato, no ambiente do FlexCM qualquer componente cliente tem acesso a instâncias dos componentes que oferecem algum serviço.

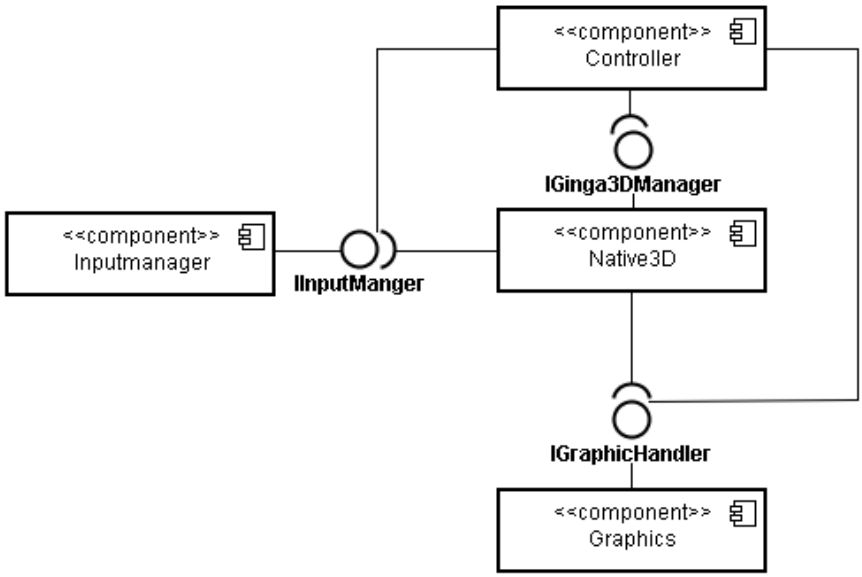


Figura 40. Comunicação entre o Native3D e outros componentes do OpenGinga.

A fim de exemplificar como a conexão entre tais componentes é feita dentro do FlexCM, na Figura 41 e 42 são mostradas, respectivamente, o registro do componente

Native3D e o esquema de conexão deste componente com os outros componentes do OpenGinga.

```
<?xml version="1.1" encoding="UTF-8" ?>
<!-- registry components from middleware -->
<registry relative_path="../">
<component alias="ginga3d-ufpb" guid="906ca0a0-5888-11db-b1de-0800200c9a66">
         <path>ginga3d-ufpb-daniel/bin/libginga3d.so</path>
         <name>libginga3d.so</name>
         <provides>972585fc-33fe-45ba-a39b-415632d81dca</provides>
</component>

<component alias="graphics-ufpb" guid="906ca0a0-5333-11db-b1de-0800200c9a66">
        <path>graphics-ufpb-daniel/bin/libgraphics.so</path>
        <name>libgraphics.so</name>
        <provides>883f0184-e9c5-4d7a-abb6-6a4f5954d024</provides>
</component>

<component alias="inputmanager-ufpb" guid="906ca0a0-5444-11db-b1de-0800200c9a66">
         <path>inputmanager-ufpb-daniel/bin/libinputmanager.so</path>
         <name>libinputmanager.so</name>
         <provides>fb273d9e-5d2c-49e5-a345-50a7298b878b</provides>
</component>

</registry>
```

Figura 41. Arquivo de registro dos componentes que serão utilizados pelo Ginga3D.

```
<?xml version="1.1" encoding="UTF-8"?>
<!-- dependency mapping from middleware -->
<architecture>
        <!--conectar o componente ginga3d-ufpb ao componente graphics-ufpb-daniel
         passando o IID da interface IGraphicHandler para o metodo IGinga3DManager::connect()
        <connect guid1="906ca0a0-5888-11db-b1de-0800200c9a66"
                 guid2="906ca0a0-5333-11db-b1de-0800200c9a66"
                 iid="883f0184-e9c5-4d7a-abb6-6a4f5954d024"></connect>
        <!--conectar o componente ginga3d-ufpb ao componente inputmanager-ufpb
         passando o IID da interface IInputManager para o metodo IGinga3DManager::connect()
        <connect guid1="906ca0a0-5888-11db-b1de-0800200c9a66"
                 guid2="906ca0a0-5444-11db-b1de-0800200c9a66"
                 iid="fb273d9e-5d2c-49e5-a345-50a7298b878b"></connect>

</architecture>
```

Figura 42. Definição do esquema de conexão entre os componentes do OpenGinga e os componentes do Ginga3D.

Basicamente, o *Native3D* é composto por quatro classes principais que definem as funcionalidades deste componente. Na figura 43 é mostrado o relacionamento entre as classes

do *Native3D*. A classe *GLCanvas* tem por objetivo definir um contêiner para uma aplicação 3D. Este container agrega uma superfície GL a ele, bem como define o comportamento das aplicações 3D que são definidas utilizando o mesmo. Para que seja possível desenhar alguma informação gráfica dentro de um *GLCanvas* uma aplicação cliente deve registrar uma entidade *drawable* ao container. Esta entidade *drawable* deve herdar as regras da interface Ginga3DDrawInterface definida dentro do componente *Graphics*. Internamente o *GLCanvas* utiliza estas propriedades do *Ginga3DDrawInterface* para desenhar os dados passados pela aplicação cliente.

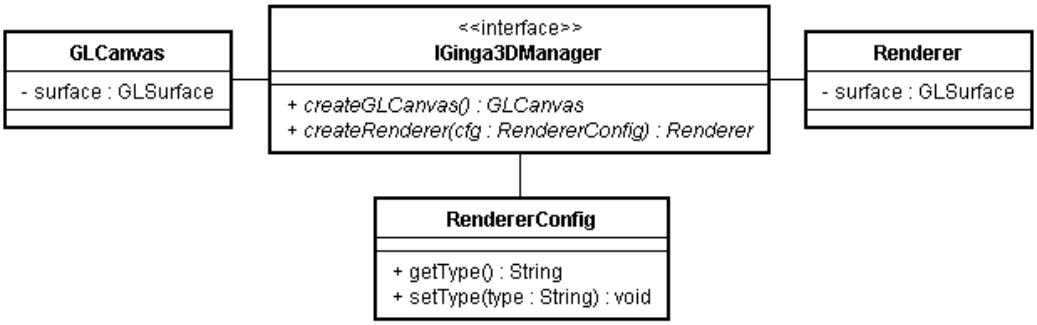


Figura 43. Diagrama de classes do Componente Native3D.

A classe *Renderer* também pode ser vista como um container para aplicações 3D. Porém, no caso do *Renderer* ele é definido de forma a representar *players* de mídia. Esta classe segue as definições do sub-módulo *Renderer* apresentado na seção 3.3.1. Como definido anteriormente, esta classe possui um comportamento bem definido, através de métodos que definem o ciclo de vida das instâncias da mesma. O ciclo de vida é determinado pelos métodos: *play(), pause(), resume()* e *destroy()*. Através destes quatro métodos os componentes das camadas superiores podem definir *players* para exibição de mídias 3D. Para a definição destes *players* é necessário programar estes quatro métodos para um determinado tipo de tecnologia específica como as linguagens de descrição (X3D ou VRML). A classe *Renderer* também agrega uma superfície GL que é utilizada internamente pelo *Native3D* para desenhar os ambientes 3D definidos pelas linguagens de descrição de ambientes. A classe *RenderConfig* encapsula os dados referentes aos exibidores de mídia definidos através do *Renderer*. Através do *RenderConfig* o *Native3D* sabe qual tipo de exibidor de mídia deverá ser construído, bem como a localização de todos os arquivos com a descrição dos ambientes que serão processados e exibidos.

Por fim, o componente *Native3D* ainda é composto pela interface *IGinga3DManager*. Esta interface é responsável por oferecer os serviços deste componente aos outros componentes do *middleware*, bem como as camadas superiores. Basicamente os serviços oferecidos são a capacidade de construir *GLCanvas* e exibidores de mídias (*Renderer*) para apresentação de ambiente 3D. No caso de *GLCanvas*, estas são construídas através do método *createGLCanvas*. Através deste método é possível configurar o tamanho e a posição da *GLCanvas* no *display*. Para a criação de exibidores de mídia é utilizado o método *createRenderer*. Assim como no método anterior é possível configurar a posição e tamanho do exibidor. Porém, além disso, este método recebe uma instância da classe *RendererConfig*. Através desta instância é que o componente *Native3D* saberá como construir um novo exibidor de mídia. Esta construção segue a seguinte estratégia: a entidade cliente indica ao componente *Native3D* que tipo de exibidor ela deseja através do método *setType* da instância do *RendererConfig*. Internamento o *Native3D* acessa o tipo de exibidor e constrói uma instância do mesmo ou lança um erro informando que aquele determinando exibidor não é suportado pelo fabricante do *middleware*.

4.4 INTEGRAÇÃO COM O GINGA-J

No Ginga-J o suporte às aplicações 3D inlcui APIs baseadas na linguagem Java, que tenham como propósito específico a construção de gráficos 3D. Como discutido na seção 3.3, no âmbito deste trabalho, a API JSR 239 (*bind OpenGL ES* para JAVA) foi a escolhida. Uma vez definida a API gráfica, é necessário apenas que os desenvolvedores de *middleware* ofereçam uma implementação desta API para suas plataformas. No contexto deste trabalho, foi desenvolvida uma implementação das funcionalidades básicas desta API a fim de validar sua utilização como API padrão para desenvolvimento de aplicações 3D no ambiente de TVDI.

Para implementação da API JSR 239, utilizamos as funcionalidades providas pelo componente *Native3D* do *Common Core*. A interface de comunicação entre este componente e o código Java foi o padrão JNI (*Java Native Interface*) conforme ilustrado na Figura 44. O JNI permite a chamada de código nativo dentro do contexto de uma aplicação gráfica. Desta forma, foi utilizada esta tecnologia para o mapeamento direto das funcionalidades da API Java para as funcionalidades providas pelo *Native3D*. A comunicação entre as diversas

camadas é feita em uma estrutura *top-down*, ou seja, de cima para baixo. O código Java cliente solicita através de chamadas sucessivas a métodos nativos que o componente *Native3D* execute certas operações. Estas solicitações são enviadas via JNI ao componente *Native3D* que, por sua vez, executa cada instrução solicitada. Como a API JSR 239 é muito semelhante a sua versão em C++, o processo de mapeamento das funções da OpenGL ES diretamente para as funções correspondentes nativas foi simplificado.

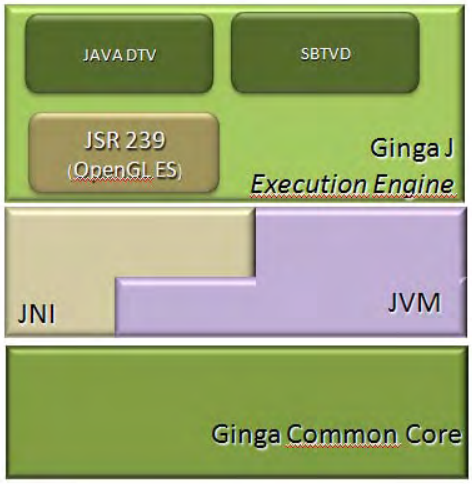


Figura 44. Comunicação entre a API Java e o *Common Core* utilizando JNI.

Em termos de implementação, o código nativo para a API Java acessa a interface do componente *Native3D* e instância *GLCanvas* à medida que solicitado pela aplicação cliente. Geralmente, apenas uma *GLCanvas* é criada e utilizada para desenhar a cena tridimensional. Quando o código Java solicita a criação de uma nova superfície para renderização, o código nativo também cria uma entidade *drawable* associada àquele *GLCanvas* a fim de utilizar esta implementação para centralizar as chamadas vindas do código Java. Em termos de interatividade, o próprio padrão brasileiro já define classes responsáveis pelo tratamento de eventos de entrada, de forma que o desenvolvedor de aplicações pode utilizá-las para a inclusão de tratamento de eventos em suas aplicações.

Em termos de execução de aplicações, o esquema é o mesmo dos outros tipos de aplicações interativas. Uma vez que uma determinada aplicação interativa chega ao *middleware*, ela é decodificada pelo processador de dados. Finalizado o processamento o componente *ApplicationManager* é invocado através do componente *Controller*, a fim de

registrar uma nova aplicação Java. Quando o *middleware* solicita a execução da aplicação, o *ApplicationManager* invoca a máquina virtual Java, passando os dados necessários para execução da aplicação interativa, como classe principal e *classpath*.

4.5 INTEGRAÇÃO COM O GINGA-NCL

Para a integração de aplicações 3D no ambiente Ginga-NCL é preciso definir exibidores de mídia que sejam capazes de executar uma mídia 3D quando solicitado pela *Engine* de Apresentação. Como apresentado na seção 3.3.2 a API NCL processa o documento NCL e, baseado em uma série de eventos descritos, executa as mídias descritas no documento. O padrão brasileiro define que a exibição da mídia é feita através de um media player com chamadas padrão: *play(), pause(), resume()* e *destroy()*. No caso das mídias 3D este procedimento também deve ser seguido independente do tipo de mídia que for utilizada.

Nesse sentido, a linguagem de descrição de ambientes virtuais X3D foi designada para ser utilizada como objeto de mídia nos documentos NCL. Para tanto foi necessário definir um *player* X3D para a execução de tais mídias quando solicitado pela API NCL. Este player denominado X3DRenderer é baseado na classe *Renderer* definida no componente *Native3D*. De fato, o objetivo da classe *Renderer* era definir um exibidor abstrato para *player* de objetos de mídia 3D. O *X3DRenderer* possui chamadas padrões definidas pela norma brasileira, de forma que pode ser invocado pela API NCL para apresentar os nós X3D descritos no documento. Instanciando a arquitetura descrita na seção 3.3.2, para o caso da utilização do X3D temos a estrutura apresentada na figura 45. Vale salientar que, esta mesma estratégia seria adotada caso outra tipo de documento de mídia 3D fosse utilizado, como é o caso do VRML.

Figura 45. Arquitetura do *NCL3DPlayer* instanciada para a tecnologia X3D.

Uma vez que o *X3DRenderer* é baseado na classe *Renderer*, ele apenas define os métodos básicos para execução do *player*. Porém, para apresentação de uma cena X3D outros requisitos são necessários. De fato, é necessário que tenhamos um *browser* X3D responsável pelo *parser* do documento e tratamento de todos os nós descritos pelo X3D. Desta forma, para prova de conceito foi necessário desenvolver um *browser* X3D capaz de processar documentos escritos neste padrão. O *browser* desenvolvido segue a norma X3D descrita para exibidores que devem ser executados em ambientes de baixo poder de processamento. Vale salientar que por questões de simplificação, nem todos os nós descritos no padrão X3D foram implementados, porém um conjunto relevante de funcionalidades foi adicionado à prova de conceito.

4.5.1 Desenvolvimento do Browser X3D

Como apresentado na seção anterior, o desenvolvimento de um browser X3D foi necessário de forma a ser possível processar documentos baseados neste padrão dentro do OpenGinga. O desenvolvimento deste *browser* exigiu a tomada de certos cuidados referentes à plataforma sobre a qual estamos trabalhando. Todos os requisitos descritos para as aplicações interativas devem ser seguidos também por esta ferramenta, uma vez que ela está sendo executada no contexto de um *set top box*. Características como desempenho, tratamento de números de ponto flutuante, estrutura de dados bem definidas são cuidados que devem ser

tomados quando do desenvolvimento deste ou de quaisquer outros *browsers* que venham a ser integrados ao *middleware*. No caso específico do X3D, o anexo C de sua norma [BRUTZMAN e DALY, 2007] define um perfil para documentos X3D que deve ser utilizado caso se deseje desenvolver aplicações para dispositivos de baixo poder de processamento. Conhecida como *Interactive Profile*, esta parte da norma, define as limitações de um *browser* em relação à quantidade de polígonos que podem ser processados, tipos de modelos de iluminação e texturização que podem ser utilizados dentre outras características.

Basicamente, o *browser* desenvolvido é composto por uma classe que processa o arquivo X3D e monta o grafo de cena que será desenhado pela rotina de *display* do renderizador. A classe de processamento segue a estratégia de delegar a cada método específico um determinado nó que será processado. Cada método da classe de processamento recebe o nó a ser processado e o nó ao qual ele deve pertencer, de forma que ela seja capaz de adicionar o nó atual ao grafo da cena X3D. Caso algum tipo de erro ocorra no processamento de um dado nó, este é abandonado, não sendo integrado ao grafo principal. Esta característica possibilita o processamento e visualização de documentos X3D mesmo que eles contenham algum erro de sintaxe. Na figura 46 podemos observar o diagrama de classes do *browser* X3D desenvolvido. Neste diagrama a classe X3D *parser* é responsável pela análise do documento X3D, bem como o processamento dos nós.

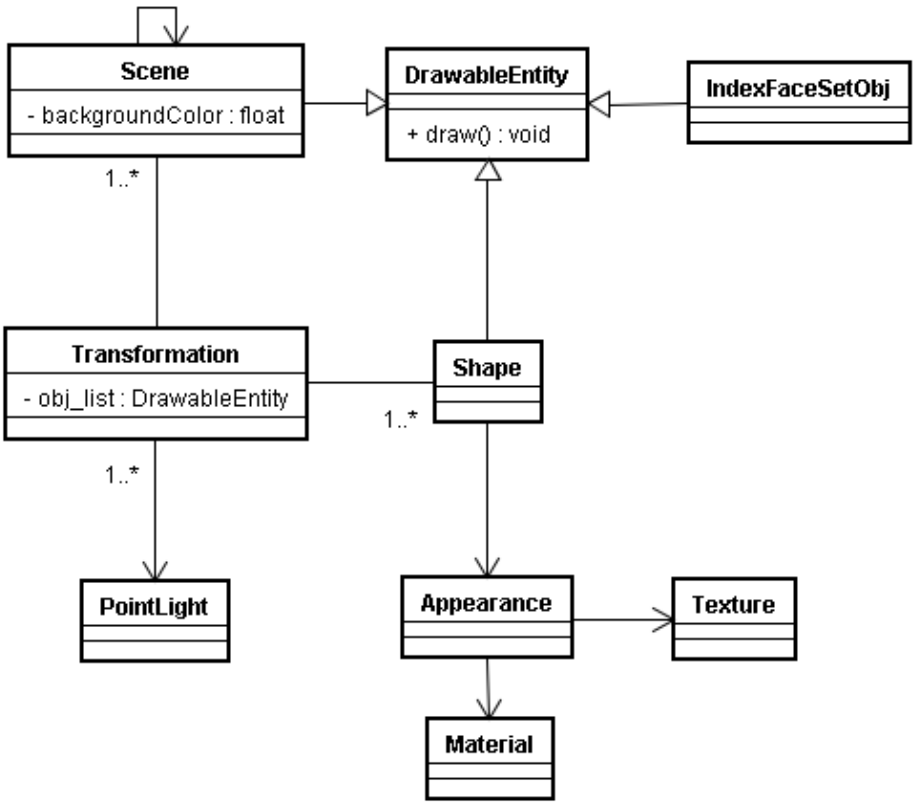


Figura 46. Diagrama de classes do *browser* X3D.

Outra classe importante dentro do contexto do *browser* desenvolvido é a *X3DView*. Ela implementa a interface *Ginga3DDrawInterface* de forma que esta classe é quem define as regras para desenho do grafo de cena armazenado. Esta classe é registrada no componente *Graphics* através da *GLSurface* que representa o *browser* X3D. A classe *X3DView* contém o grafo de cena que será desenhado, bem como uma instância do *parser* que será utilizado para processar o documento. Esta classe recebe como entrada o documento X3D que deve ser processado. Este documento é então passado para o *parser* que se encarrega de processá-lo. Uma vez que documentos X3D podem fazer referência a outros documentos X3D, os métodos de processamento da classe *X3DParser* podem fazer chamadas recursivas a fim de processar nós que descrevem uma cena em um documento diferente. As outras classes do *browser* X3D encapsulam as características de cada um dos nós que foi implementado. Por exemplo, a classe *Box* encapsula um nó X3D que descreve um objeto gráfico na forma de uma caixa. A classe *Transform* define nós que aplicam transformações de escala, rotação e translação a um objeto ou um conjunto de objetos gráficos. As classes *Light* e *Material* definem as luzes e

propriedades materiais dos objetos respectivamente. A classe *IndexFaceSetObj*, por sua vez, descreve objetos gráficos mais complexos definidos por um conjunto de pontos e vértices definidos dentro do próprio documento X3D.

Quanto à interação, o *browser* X3D solicita o registro do componente *Native3D* como ouvinte dos eventos de entrada. Os eventos são então passados ao *browser* que responde as entradas de uma forma pré-definida. Basicamente, os eventos de entrada são utilizados no *browser* com o objetivo de navegação, modificação do ponto de visualização e ativação de eventos e sensores descritos no documento, como abrir portas, interagir com objetos gráficos, etc. Uma vez que o *browser* for destruído, ele indica ao componente *Native3D* que não serão mais tratados eventos, de forma que este possa remover seu registro do componente *InputManager*.

Por questões de simplificação, nem todos os nós X3D foram suportados no *browser* X3D desenvolvido. Optou-se pela definição apenas dos nós mais utilizados, uma vez que, através deles já é possível avaliar o desempenho do *browser* no tocante a exibição de ambientes virtuais. A tabela 01 apresenta o nós que são suportados pelo *browser* X3D desenvolvido no âmbito deste trabalho.

Tabela 1. Nós X3D suportados pelo *browser* desenvolvido.

Nó X3D Implementado	Status
Box	Suportado
Sphere	Suportado
Inline	Parcialmente Suportado
Scene	Parcialmente Suportado
Transform	Suportado
Material	Suportado
Texture	Suportado

IndexFaceIbject	Parcialmente Suportado
Light	Parcialmente Suportado (modelo *Flat*)
Shape	Parcialmente Suportado
Background	Suportado
NavigationInfo	Parcialmente Suportado

4.5.2 Mídia 3D e documentos NCL

A descrição das mídias 3D em documentos NCL é viável devido ao próprio caráter da linguagem. O modelo sobre o qual o NCL foi concebido permite sua extensão para a descrição e apresentação de novos tipos de mídias. Desta forma, a descrição de documentos X3D ou VRML pode ser integrada de forma simples. Uma vez que estamos tratando a mídia 3D em um ambiente definido por uma janela 2D, as definições de região e descrição de mídia são similares a definição de uma mídia do tipo vídeo. Utilizando uma abordagem onde os nós de mídias 3D realizam as operações similares a um player de vídeo (*play, pause, resume* e *stop*) a definição de elos entre estas mídias e outros tipos de mídia também é facilitado. É possível definir, por exemplo, quanto tempo aquele ambiente virtual estará ativo, passível de interação ou em estado de espera (*pause*). No caso deste trabalho, foi proposto apenas a apresentação destas mídias de forma simplificada, em um contexto 2D, como é comumente utilizado nos sistemas convencionais: PCs, vídeo games, celulares.

4.6 CONSIDERAÇÕES FINAIS

Este capítulo apresentou a análise e desenvolvimento do modelo prático utilizado para validar a arquitetura do Ginga3D. Com o objetivo de simular um ambiente próximo do real foi utilizado o OpenGinga, uma implementação de referência do *middlewarde* Ginga. Esta ferramenta foi concebida para execução sobre a plataforma PC. Para prova de conceito foi

escolhida a versão do OpenGinga que utiliza os conceitos de desenvolvimento baseado em componentes, de forma que, os módulos definidos para o Ginga3D foram transformados em componentes e colocados de acordo com o modelo FlexCM para componentes.

O modelo prático desenvolvido possibilitou a execução de aplicações tanto no ambiente do Ginga-J, com a integração da API JSR 239 ao *middleware*, bem como no ambiente do Ginga-NCL com o desenvolvimento de players de mídia 3D. No caso do Ginga-J foi utilizada a comunicação via JNI como ponte entre as funcionalidades do componente *Native3D* e o código Java. Para o ambiente NCL foi necessário desenvolver um player para um tipo de mídia 3D a fim de validar a arquitetura proposta. A linguagem de descrição de mídia X3D foi escolhida. Para tanto, foram desenvolvidos um player X3D, bem como um *browser* responsável por processar e apresentar o conteúdo dos documentos X3D. Na abordagem apresentada, os ambientes 3D são manipulados pela API NCL como entidades gráficas descritas em uma janela 2D. Esta abordagem facilita a extensão da linguagem para a descrição de mídias 3D.

5 RESULTADOS

Neste capítulo são apresentadas algumas aplicações desenvolvidas utilizando o Ginga3D. Como plataforma de execução foi utilizada a implementação apresentada no capítulo 4. Além das aplicações tridimensionais, serão apresentados alguns dados referentes ao desempenho das tecnologias utilizadas por cada ambiente para executar tais aplicações no OpenGinga. Por fim, é apresentado um comparativo relacionado às funcionalidades oferecidas por cada ambiente. Este comparativo diz respeito às características oferecidas por cada ambiente como técnicas suportadas, flexibilidade para desenvolvimento, tempo de desenvolvimento de aplicações, etc.

5.1 DESENVOLVIMENTO E EXECUÇÃO DE APLICAÇÕES 3D

A fim de testar os componentes desenvolvidos foram construídas quatro aplicações 3D, sendo duas para execução no ambiente Ginga-J e duas para execução utilizando o player 3D desenvolvido para o ambiente Ginga-NCL. As aplicações para o Ginga-J procuraram explorar o conjunto de funções contidas na biblioteca JSR 239 com o objetivo de validar a implementação desta API sobre o componente Native3D. As aplicações desenvolvidas para o ambiente Ginga-NCL procuram avaliar as funcionalidades implementadas pelo player 3D para documentos X3D. Para tanto, foram desenvolvidos dois ambientes que oferecem possibilidade de navegação e utilizam os nós X3D que foram incorporados ao browser X3D desenvolvido.

5.1.1 Aplicações no Ginga-J

A primeira aplicação desenvolvida foi o clássico exemplo *GLGears* que tem por objetivo apresentar três engrenagens tridimensionais conectadas uma as outras executando um movimento sincronizado (ver Figura 47). Neste exemplo foram utilizadas algumas das funções responsáveis pelo tratamento do modelo de iluminação e aplicação de propriedades materiais sobre os objetos gráficos. Foi utilizado o modelo de iluminação constante, de forma a melhorar o desempenho da aplicação reduzindo o número de cálculos. Nesta aplicação foi

testado também as funções para manipulação das matrizes da OpenGL bem como as funções para a geração de uma projeção perspectiva.

Figura 47. Execução da aplicação Java GLGears no OpenGinga.

A aplicação é construída em uma superfície hierarquicamente superior a superfície do vídeo, em conformidade com o esquema de sobreposição de superfícies apresentado na seção 3.3.3. Esta aplicação em particular é executada por um tempo determinado, no caso 10 segundos. Ao fim da aplicação todos os recursos utilizados são liberados e a máquina virtual é destruída. Devido a baixa quantidade de primitivas gráficas utilizadas, bem como um modelo de iluminação simplificado, esta aplicação é executada dentro do ambiente do *middleware* a uma taxa média de 200 fps (*Frames per Second*).

A segunda aplicação desenvolvida utilizando a API OpenGL ES é uma animação com três cubos que executam um movimento aleatório dentro do ambiente tridimensional (ver Figura 48). São apresentadas também outras primitivas gráficas no background da janela. O objetivo desta aplicação é testar funções mais complexas da API integrada ao *middleware*. Notadamente são testados os recursos para importação e aplicação de texturas 2D. Outro recurso testado é a capacidade de manipulação do *buffer* de profundidade, como por exemplo,

a permissão para escrita neste *buffer*. Outras funções testadas estão relacionadas ao desenho de primitivas utilizando as funções *client-side* da OpenGL.

Figura 48. Execução da aplicação Java de cubos com movimentos aleatórios.

O esquema de apresentação da imagem é análogo a aplicação GLGears. A manipulação dos *buffers* de profundidade não influencia o desempenho da aplicação. Porém, como já era esperado a aplicação de texturas reduz a performance, bem como aumenta o consumo de memória devido a necessidade armazenamento dos dados referentes as imagens carregadas. A utilização das funções *cliente-side* da OpenGL ES não causam nenhum tipo de modificação na performance uma vez que todos os cálculos são feitos no próprio dispositivo que embarca o *middleware*. Em termos de taxa de apresentação, esta aplicação apresentou uma média de aproximadamente 146 fps.

5.1.2 Aplicações no Ginga-NCL

Os testes das aplicações 3D no ambiente Ginga-NCL tiveram como foco a apresentação de objetos de mídia 3D utilizando o player 3D construído sob o *browser* X3d. Foram desenvolvidos dois tipos de testes. O primeiro teste, análogo aos testes com o Ginga-J, estavam relacionados a avaliação da performance do *browser* X3D e como os objetos de mídia eram apresentados no ambiente. O segundo teste teve como objetivo avaliar a interface

do player 3D quanto ao seu comportamento quando solicitada uma determinada ação (play, pause, resume e destroy). Quanto aos resultados do primeiro teste, são apresentadas abaixo algumas das aplicações desenvolvidas, em relação ao segundo teste o player se comportou da forma prevista quando solicitado algum tipo de comando através de sua interface padrão.

Para o teste de apresentação de objetos de mídia, no primeiro instante foi utilizado um documento X3D simples que descreve um birô e uma cadeira (ver Figura 49). O usuário tem a possibilidade de explorar estes objetos através da interação com a cena 3D. Para apresentação desta cena, o *player* 3D é invocado e utilizando a implementação do *browser* X3D executa o *parser* do documento, processa os nós e apresenta a cena para o usuário. O usuário da aplicação tem a possibilidade de explorar o objeto de mídia apresentado através da navegação pelo ambiente. Embora o padrão X3D permita diversas formas de exploração e navegação, este *browser* suporta apenas o modo "*ANY*" que permite o usuário navegar livremente pela cena. Em termos de frequência de atualização, este ambiente é executado a uma taxa de 227 fps.

Figura 49. Browser X3D apresentando o conteúdo de um documento X3D que descreve um gabinete simples.

Outro teste realizado utilizou um documento X3D que descreve um ambiente virtual mais complexo. Este documento exibe um pequeno templo chinês envolvido por uma cerca de madeira (ver Figura 50 e Figura 51). Neste documento foram utilizados todos os nós X3D implementados pelo *browser*. A aparência do ambiente foi descrita utilizando o nó *Material*, os objetos foram descritos utilizando o nó *IndexFaceSetObject*. O templo foi dividido em ambientes menores, os quais foram carregados na mesma cena utilizando o esquema nós *Inline* do padrão X3D. Através deste tipo de nó é possível importar um documento X3D inteiro para dentro do ambiente que se está manipulando. Outros nós para aplicação de textura, adição de objetos pré-definidos e múltiplos pontos de luz também foram utilizados. Para este ambiente a taxa de atualização foi de 58 fps.

Figura 50. Browser X3D apresentando o conteúdo de um documento X3D que descreve um templo chinês.

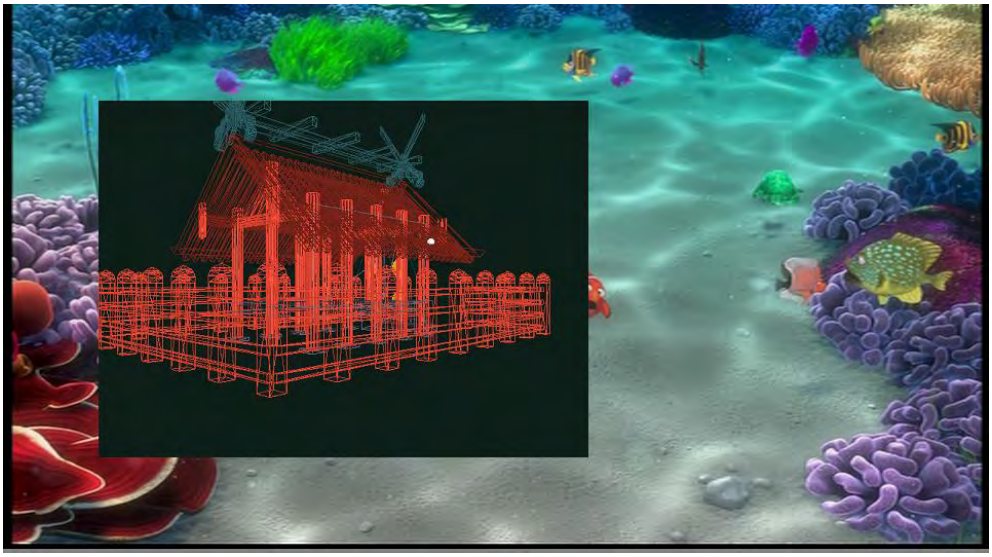


Figura 51. Browser X3D apresentando o conteúdo de um documento X3D que descreve um templo chinês em modo *wireframe*.

5.2 ANÁLISE DE DESEMPENHO

A fim de analisar a performance do OpenGinga, após a inclusão dos componentes do Ginga3D foram executados alguns testes. Tais testes tiveram como objetivo a análise de certos como taxa de atualização das cenas 3D, consumo de memória e uso da CPU. Para a execução dos testes foi utilizado a mesma metodologia em ambos os ambientes. Foi desenvolvida uma aplicação simples que apresenta um cubo composto por primitivas. Neste caso o tipo de primitiva gráfica utilizada foi o quadrado. Á medida que os testes eram executados, o número de primitivas que definia o cubo aumentava (ver Figura 52).

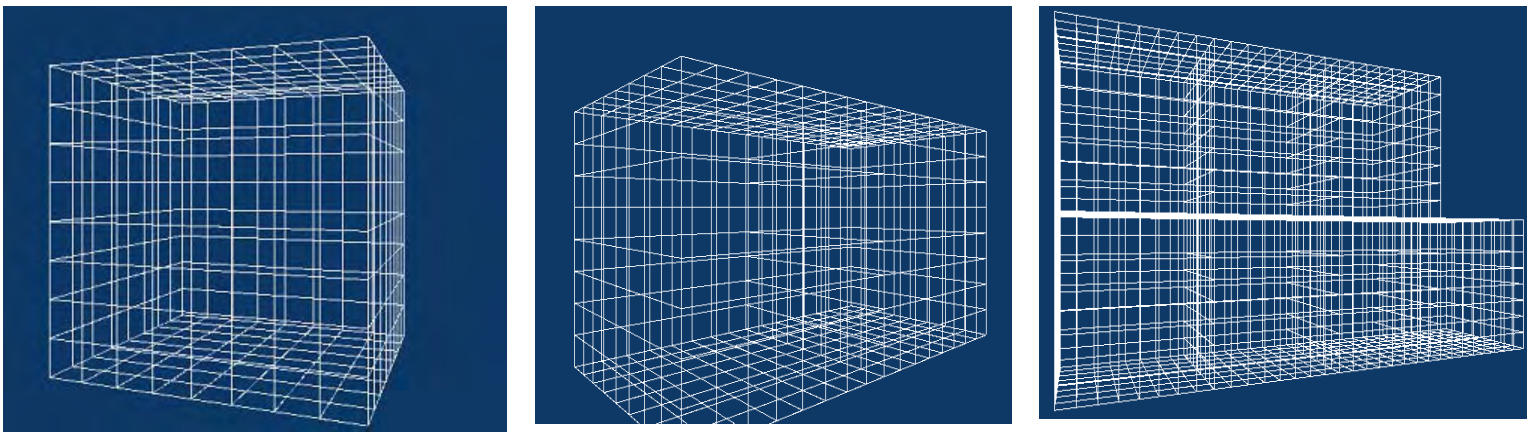


Figura 52. Objetos gráficos utilizados para análise de desempenho.

O primeiro teste executado avaliou a taxa de atualização da cena tridimensional em cada ambiente. No ambiente Ginga-J esta taxa diz respeito ao número de frames gerados pela JSR 239 em um segundo. No caso do Ginga-NCL foi avaliado o número de frames gerados em um segundo pelo browser X3D que apresenta a cena 3D. Na Figura 53 é apresentada a variação da taxa de atualização para cada ambiente à medida que o número de primitivas gráficas varia.

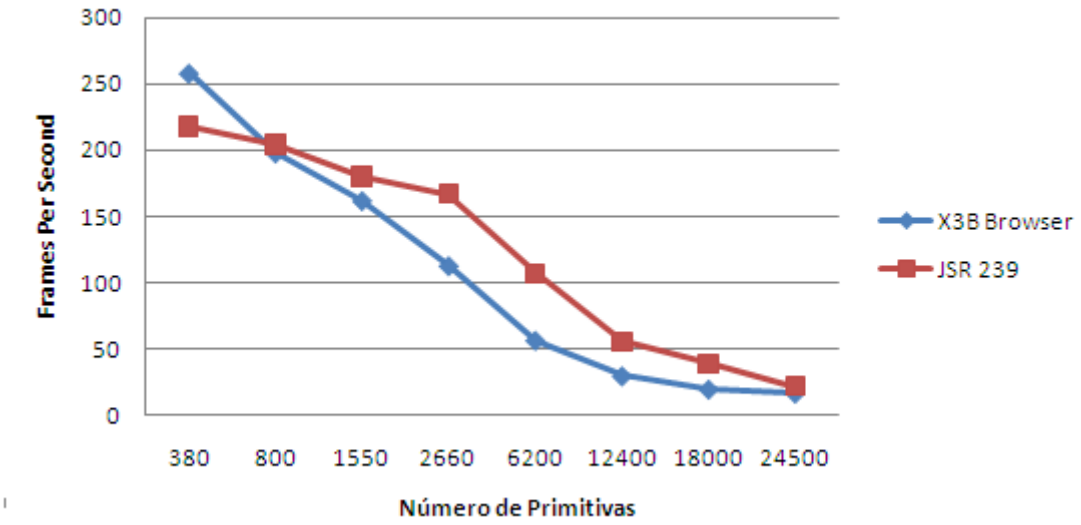


Figura 53. Variação da taxa de atualização da cena gráfica em função do número de primitivas utilizadas na aplicação.

Como pode ser observado na Figura 53 para um número baixo de primitivas ambos os ambientes apresentam taxas de atualização elevadas. O browser X3D apresenta taxa mais elevadas que a API JSR 239. Isto se deve a necessidade de execução da aplicação Java sobre a máquina virtual, de forma que a perda em desempenho decorrente deste fato é perceptível quando estamos lidando com aplicações mais simples. À medida que o número de primitivas vai aumentando ambos os ambientes declinam em desempenham. Outro fato observado é que à medida que o número de primitivas aumenta a aplicação Java passa a ganhar em desempenho em relação aos ambientes apresentados no browser X3D, porém para valores muito altos ambas possuem desempenho parecido. Até 25000 primitivas os dois ambientes apresentam taxas aceitáveis de aproximadamente 30 fps. Quando se passa muito deste limite, as taxas caem para 20 fps.

Quanto ao uso da CPU, na Figura 54 é observado um gráfico comparativo entre os serviços básicos do Ginga e as tecnologias 3D empregadas para execução de aplicativos tridimensionais. Os serviços básicos do Ginga, como decodificação de vídeo, sistema de informação e *tuner*, utilizam aproximadamente 75% para executarem suas tarefas básicas. Dentro estes serviços o decodificador de vídeo é o processo mais crítico dentro do ambiente do OpenGinga. Isto se deve ao fato do OpenGinga ser executado sob a plataforma PC, de forma que não é utilizado nenhum tipo de placa decodificadora dedicada. Quanto aos serviços do Ginga3D, quando o browser X3D é invocado o uso da CPU sobre para aproximadamente 75%, uma vez que o mecanismo de renderização é ativado, bem como, os mecanismos de tratamento de eventos do próprio browser. Quando executamos uma aplicação Java o consumo fica em terno dos 87%, uma variação similar a utilização do browser X3D. De fato, ambos os ambientes utilizam o mesmo componente nativo para renderização das cenas. O que diferencia ambos é a necessidade que a API JSR 239 fazer chamadas via JNI para acessar as funcionalidades do componente Native3D, enquanto o browser X3D faz isto de forma nativa.

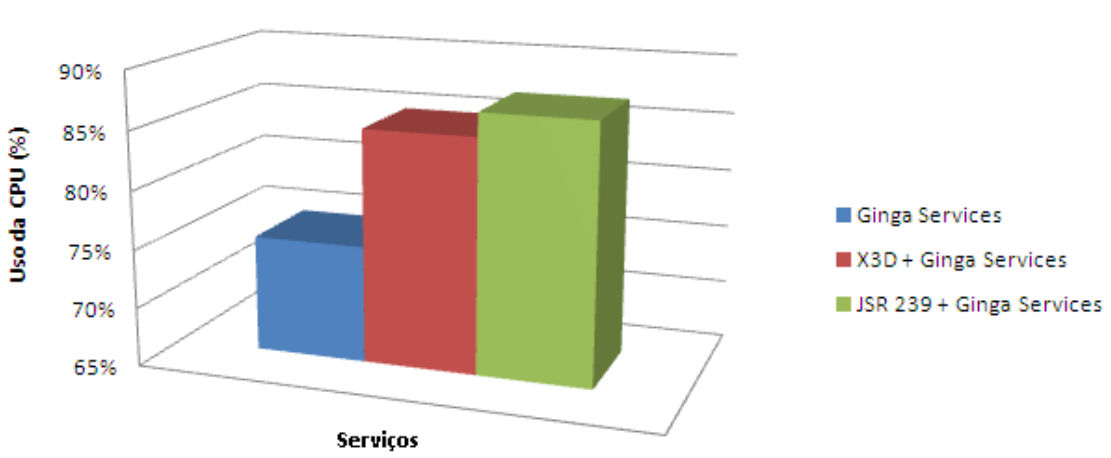


Figura 54. Porcentagem de uso da CPU pelos serviços oferecidos pelo *middlewar*e.

Outro aspecto avaliado foi o consumo de memória pelos serviços do *middleware*, bem como pelos serviços oferecidos pelo Ginga3D (ver Figura 55). Os serviços básicos do *middleware* consomem em média 60 Mb de memória para executarem suas tarefas. No caso do browser X3D, quando executamos um ambiente simples o consumo de memória sobe para aproximadamente 68.17 Mb, representando um aumento de 8.17 Mb uso de memória pelos

serviços do *middleware*. No caso da aplicação Java este consumo aumenta para aproximadamente 74.3 Mb. A justificativa para um maior consumo, como nos outros casos se deve a utilização da máquina virtual para execução das aplicações 3D.

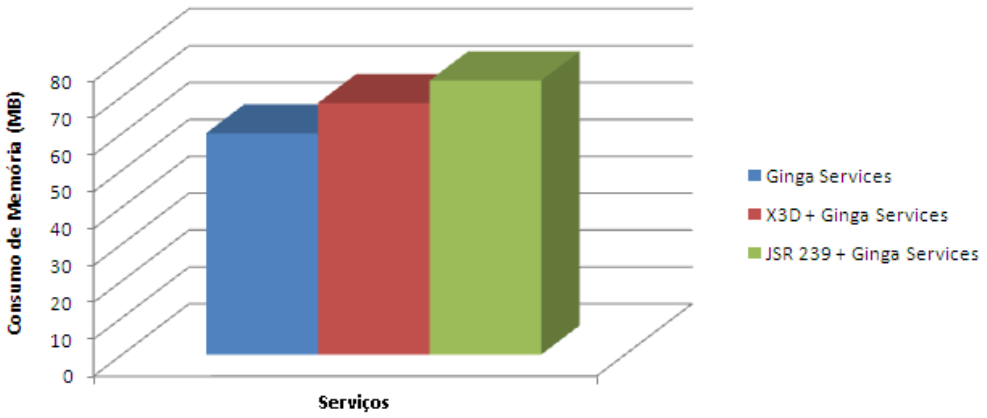


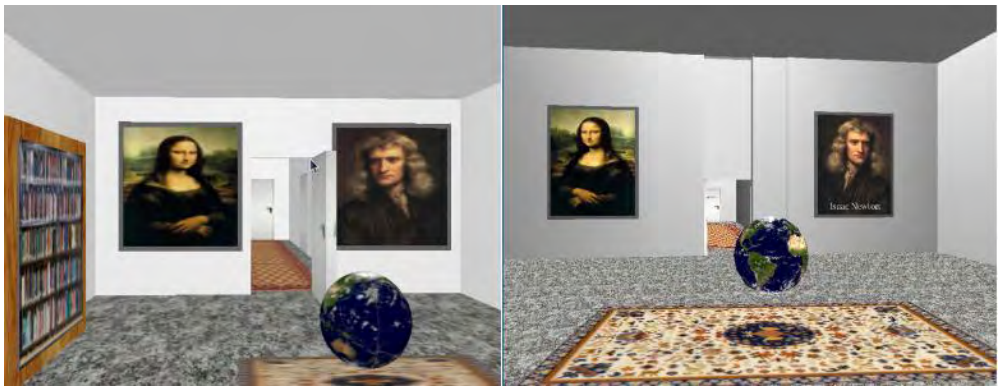
Figura 55. Porcentagem de uso da CPU pelos serviços oferecidos pelo *middlewar*e.

Vale salientar que os valores apresentados no gráfico da Figura 55 estão dentro da faixa de valores previstos por CESAR et. al. (2005). Em seu trabalho o autor afirma que um ambiente de *middleware* com suporte a aplicações interativas 3D consome em média 90Mb de memória. No caso deste trabalho, o consumo médio apresentado está abaixo dos limites verificados pelo autor. É evidente que este consumo pode variar dependendo do tipo de aplicação, porém o que de fato pode aumentar o consumo de memória em termos de aplicação é a utilização de texturas e o uso exagerado de primitivas gráficas para descrever os objetos tridimensionais. Desta forma, é necessário estabelecer limites para o uso destas funcionalidades na construção de aplicativos tridimensionais para TV. Estas restrições devem ser definidas na própria especificação do padrão à exemplo da norma X3D que identifica estes limites para cada perfil de interatividade da linguagem.

5.3 GINGA-J X GINGA-NCL

A fim de comparar o suporte a aplicações 3D nos ambientes Ginga-J e Ginga-NCL, foi desenvolvido um ambiente virtual simplificado. Este protótipo foi importante para análise de

certos aspectos relacionados ao desenvolvimento de aplicações 3D em ambos os ambientes. A mesma aplicação foi utilizada de forma que o processo de desenvolvimento foi avaliado com base nos mesmos requisitos. Em termos de desenvolvimento foram observadas as técnicas de aplicação de textura, tratamento de colisão e eventos. Em termos de arquitetura, a aplicação desenvolvida utilizando o Ginga-J foi executada sob a implementação da API OpenGL ES, acessando as funcionalidades do componente Native3D. A aplicação desenvolvida para o Ginga-NCL foi executada com o auxílio do browser X3D desenvolvido neste trabalho (ver Figura 56).



(a) **(b)**

Figura 56. Ambiente Virtual desenvolvido. Em (a) utilizando o X3De em (b) utilizando o JSR 239.

O uso de *browsers* torna desnecessário a implementação de certas funcionalidades uma vez que eles já incorporam estas características. Um exemplo são os importadores de malhas. Contudo, quando utilizamos uma API de mais baixo nível, como a OpenGL ES os desenvolvedores precisam incorporar, através de implementação ou utilizando alguma biblioteca, estes importadores a suas aplicações. A utilização de browsers também torna desnecessário que o usuário desenvolvedor tenha conhecimentos profundos sobre técnicas de computação gráfica. Por outro lado, o uso de API de baixo nível permite a construção de aplicações customizadas que utilizam técnicas mais sofisticadas que, por alguma razão, possam não ser suportadas pelos browsers.

Na Tabela 02 é apresentada uma comparação entre as funcionalidades oferecidas pela API gráfica em Java e o uso da linguagem NCL com X3D através de um browser específico. O desenvolvimento de aplicações utilizando uma linguagem de descrição de ambientes como

X3D ou VRML é mais rápido. Isto está relacionado ao fato de que browsers abstraem parte da complexidade das técnicas de computação gráfica que geralmente são utilizadas em aplicações. Como resultado, a aplicação usando o X3D foi desenvolvida em tempo menor que a desenvolvida utilizando OpenGL ES. Contudo, como apresentado na Tabela 2, o uso de bibliotecas gráficas permite uma maior flexibilidade quanto ao desenvolvimento de aplicações 3D, uma vez que é possível escolher as técnicas utilizadas baseada na necessidade do desenvolvedor. Neste sentido, aplicações desenvolvidas com tecnologias que necessitam de um browser para serem executadas estão restritas as funcionalidades oferecidas por aquele browser. Desta forma, cada ambiente pode oferecer certos benefícios, dependendo dos requisitos necessários para um dado tipo de aplicação. A estratégia de suporte a tecnologias 3D em ambos os ambientes permite que desenvolvedores sintam-se livres para escolher o ambiente em que desejam trabalhar.

Tabela 2. Comparação entre a API JSR 239 e o X3D.

Características	**Ginga-J (Java)**	**Ginga-NCL (NCL+X3D)**
Tratamento de Colisão	Implementado pelo desenvolvedor	Suportado no *browser*
Tratamento de Eventos	Implementado pelo desenvolvedor	Suportado no *browser*
Textura	Implementado pelo desenvolvedor	Suportado no *browser*
Técnicas de Iluminação	Completamente suportado	Parcialmente suportado
Técnicas Avançadas (*Ray trace, Fog, Bump Mapping*, etc)	Completamente suportado	Parcialmente ou não suportado
Tempo de Desenvolvimento	~5 dias	~1 dias

5.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou resultados obtidos na utilização do modelo prático desenvolvido para validar a arquitetura do Ginga3D. Neste sentido foram desenvolvidas algumas aplicações interativas 3D utilizando as tecnologias 3D integradas ao OpenGinga. Tais aplicações serviram para avaliar o desempenho do ambiente proposto, bem como analisar as funcionalidades oferecidas pelo mesmo. Além das aplicações, foi apresentada uma análise de desempenho da API JSR 239, bem como do browser X3D. Foi verificado que os ambientes Ginga-J e Ginga-NCL necessitam de quantidades parecidas de recursos para execução de aplicativos tridimensionais. Por fim, foi feita uma comparação entre os ambientes em termos de funcionalidades oferecidas, flexibilidade e tempo de desenvolvimento de aplicações. Concluiu-se que cada ambiente possui suas limitações e benefícios e que a estratégia que oferece suporte tanto no Ginga-J quanto no Ginga-NCL oferecer uma maior flexibilidade para os desenvolvedores de aplicações interativas.

6 CONCLUSÃO

A redução do custo computacional de certas APIs gráficas e o melhoramento do hardware gráfico para sistemas embarcados contribui para que sistemas 3D possam ser integrados aos dispositivos de baixo poder de processamento. Tais fatores possibilitam a utilização de técnicas para a geração de cenas 3D dentro de um ambiente de TVDI, fornecendo um conjunto de novas possibilidades e idéias no que diz respeito às aplicações interativas. Neste sentido, propomos o Ginga3D, uma arquitetura baseada no *middleware* Ginga. O objetivo do Ginga3D é estender o *middleware* brasileiro de forma a permitir o desenvolvimento e execução de aplicações tridimensionais em um ambiente de TV. Através do Ginga3D serão exploradas as possibilidades de utilizar o ambiente procedural ou declarativo do *middleware* para suportar tais aplicações.

Neste capítulo evidenciamos os resultados obtidos a partir da realização do presente trabalho em âmbito teórico e prático, como também, os principais desdobramentos previstos em trabalhos futuros.

6.1 RESULTADOS OBTIDOS

Como resultado do trabalho foi concebida uma arquitetura, baseada na extensão do middleware Ginga, a fim de incorporar o suporte ao desenvolvimento e execução de aplicações tridimensionais em um ambiente de TV Digital. Com o objetivo de validar a arquitetura proposta foi desenvolvida uma prova de conceito baseada em uma implementação livre do *middleware* brasileiro: o OpenGinga. A prova de conceito desenvolvida expandiu o OpenGinga de forma a suportar a execução de aplicações 3D em ambos os ambientes do *middleware*, como proposto na arquitetura. No ambiente Ginga-J foi adicionado a API OpenGL ES, enquanto que, no ambiente do Ginga-NCL foi adicionado o suporte a *player* 3D. Ainda como parte da prova de conceito, a arquitetura do *player* 3D foi instanciada para uma tecnologia específica, no caso, o X3D. Desta forma foi incorporada à prova de conceito um *browser* X3D compatível com as normas definidas para sistemas com baixo poder de processamento.

Além da especificação da arquitetura e da prova de conceito, foram desenvolvidas algumas aplicações interativas a fim de analisar o comportamento destas dentro do ambiente do *middleware*. Estas aplicações também serviram para analisar as possibilidades de interação com as aplicações 3D, bem como avaliar a integração de tais tecnologias em cada um dos ambientes do Ginga. Para tanto foram feitos alguns testes que avaliaram o desempenho das aplicações e foram feitas análises relacionadas às funcionalidades oferecidas em cada ambiente.

Como principal diferencial em relação a outros trabalhos o Ginga3D oferece um ambiente flexível para o desenvolvimento e execução de aplicações, de forma que o usuário desenvolvedor pode optar pelos benefícios provenientes do ambiente procedural ou do ambiente declarativo do *middleware*. Neste contexto, outro diferencial do presente trabalho é o suporte a descrição e sincronização de mídias 3D através da linguagem NCL. A sincronização de ambientes tridimensionais descritos por linguagens como X3D e VRML com outras mídias também é possível, visto que a linguagem NCL possibilita a extensão de seus descritores.

No contexto social, a incorporação desta proposta ao *middleware* brasileiro pode auxiliar a construção de aplicativos que contribuam para o desenvolvimento de áreas como a Educação à Distância, inclusão social através do acesso a informação, bem como a resolução de problemas emergentes relacionados ao próprio sistema brasileiro de TV, como é o caso da transmissão e apresentação de conteúdo em LIBRAS. Em termos de mercado, o suporte a aplicações 3D pode gerar uma nova área de atuação por parte das empresas, desenvolvendo produtos que possam ser beneficiados através do uso deste tipo de tecnologia. A indústria de games, notadamente, é uma das áreas com maior potencial quando falamos de aplicações 3D. Um novo mercado para jogos de TV com um público relativamente considerável poderia impulsionar esta área da mesma forma que a indústria de games para dispositivos móveis foi impulsionada quando do suporte de tecnologias 3D a estes dispositivos.

6.2 TRABALHOS FUTUROS

A estratégia adotada no trabalho segue o modelo tradicional de apresentação de aplicações 3D em sistemas de visualização 2D. Desta forma, características relacionadas a

apresentação do ambiente virtual na TV ainda podem ser descritas em fator de termos bidimensionais, como posição e tamanho 2D da janela. Esta metodologia facilita a integração com as tecnologias já existentes para TV. Porém, outra abordagem pouco explorada seria aquela em que todo o ambiente de TV fosse visto de um contexto tridimensional, de forma que o ambiente 3D das aplicações pudesse se confundir com o próprio ambiente da TV. Neste sentido, poderíamos ter uma referência tridimensional para qualquer conteúdo áudio-visual apresentado. Esta abordagem não é tratada neste trabalho, porém aparece como uma solução paralela que poderia ser estudada mais afundo a fim de verificar sua viabilidade.

Outro trabalho futuro está relacionado ao suporte na linguagem NCL a mecanismos de sincronização de mídia em um contexto tridimensional. Embora a linguagem permita sua expansão a fim de descrever novas mídias e a sincronização entre elas, esta não descreve estas mídias em um contexto tridimensional, onde cada conteúdo teria uma posição específica neste ambiente. Talvez, uma abordagem semelhante a de padrões como o MPEG-4 que utiliza uma linguagem para descrição de nós de mídia (BIFS). Como apresentado na seção 2.5.2, esta linguagem oferece um meio de sincronizar diversos tipos de mídia, inclusive mídias 3D. É permitido também fazer referências a nós de vídeo dentro do ambiente 3D, porém como discutido, a sua implementação em codificadores e decodificadores MPEG-4 inviabiliza a sua popularização, devido ao alto custos. A extensão do NCL a fim de oferecer funcionalidades semelhantes poderia viabilizar a utilização deste tipo de estratégia.

Em termos de aplicações para TV, um dos principais focos de trabalho seria o desenvolvimento de um mercado para a produção e comercialização de jogos para TV Digital. Ainda relacionado à área de jogos, mas em um contexto de pesquisa, estes ambientes poderiam ser utilizados para o desenvolvimento de jogos educativos promovendo, por exemplo, a área de educação a distância. Em relação a ambientes virtuais, através da integração de tecnologias como X3D e VRML é possível estendermos as pesquisas nesta área para dentro do contexto da TV, de forma que estes ambientes possam se popularizar através dos segmentos mais populares. A própria pesquisa na área de interpretação de sinais de LIBRAS ainda está em sua fase inicial podendo ser ampliada através da utilização desta nova proposta.

Em relação ao ambiente Ginga-NCL, mais precisamente os player para linguagem X3D é possível ampliar o seu suporte através de mecanismos do próprio X3D para comunicação com código Java. Como o padrão brasileiro especifica uma ponte de comunicação entre o Ginga-J e o Ginga-NCL é possível usar informações da SAI (*Scene Acess Interface*) do padrão X3D para manipulação mais robusta destes ambientes. Atualmente o browser desenvolvido não possui suporte a tratamento de eventos que disparam código Java, justamente pelo fato de estarem sendo executados sob o ambiente Ginga-NCL. Este fato limita o potencial deste padrão para ambientes virtuais.

Outro trabalho futuro nesta área é a análise da utilização de dispositivos não-convencionais para interação com tais ambientes 3D. A utilização de controles remotos pode restringir de certa forma as possibilidades de interação do usuário dentro destes ambientes, de forma que a análise de novas formas de se interagir pode beneficiar a utilização de tais tecnologias dentro de um ambiente de TV. Neste sentido, a inclusão de dispositivos com retorno de força, sistemas de rastreamento dentre outras técnicas são campos a serem explorados.

6.3 DISCUSSÃO

A fim de transformar esta pesquisa em produto é preciso que o trabalho seja adaptado aos moldes de um documento de proposta para extensão da especificação do *middleware* Ginga existente atualmente. As definições e possibilidades aqui apresentadas necessitem de aprovação do fórum SBTVD e órgãos gestores de forma a compor as normas ABNT que definem o padrão de TV Digital no Brasil. Uma vez aprovado este novo padrão abre um conjunto novo de possibilidades na área de TV Digital, ampliando consideravelmente o número de trabalhos que podem ser desenvolvidos com tal tecnologia. Em relação à própria arquitetura, existem outros aspectos a serem observados e novas possibilidades quanto a forma de integrarmos tecnologias 3D ao *middleware*. Neste sentido, é coerente acreditar que nos próximos anos, com a queda nos preços do hardware para TV, além do desenvolvimento de sistemas mais robustos, a área de aplicações tridimensionais pode ganhar grande importância dentro do cenários de aplicativos interativos para TV Digital.

Este trabalho foi desenvolvido em uma parceria entre o Laboratório de Desenvolvimento de Tecnologias para o Ensino Virtual e Estatística (LabTEVE) e o Laboratório de Vídeo Digital (LAViD), ambos instalados na Universidade Federal da Paraíba. O desenvolvimento do trabalho permitiu a troca de conhecimentos e tecnologias entre os dois laboratórios promovendo, acima de tudo, uma rica discussão sob o ponto de vista de cada grupo de pesquisa.

Como resultados desta pesquisa também foram publicados alguns trabalhos científicos em congressos na área de TV Digital e Realidade Virtual. O primeiro trabalho intitulado *Incorporating 3D Technologies to the Brazilian DTV Standard: Integration Strategies and Proposed Architecture* foi publicado no *8th European Conference on Interactive TV and Video* (EUROITV'2010) que ocorreu na cidade de Tampere, Finlândia. Um segundo trabalho intitulado *Extending Brazilian DTV middleware to Incorporate 3D Technologies* foi publicado no *XII Symposium on Virtual Reality (SVR'2010)*, que ocorreu na cidade de Natal, Brasil.

REFERÊNCIAS

[ABNT NBR 15601, 2008] ABNT NBR 15601. Televisão digital terrestre — Sistema de transmissão. 2008.

[ABNT NBR 15606-2, 2008] ABNT NBR 15606-2. Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações. 2008.

[ABNT NBR 15606-5, 2008] Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital - Parte 5: Ginga-NCL para receptores portáteis - Linguagem de aplicação XML para codificação de aplicações.

[ALVES et al., 2004] ALVES, L. G. R., Sena, G., Rettori, A., Guimarães, H. Ensino on-line, Jogos Eletrônicos e RPG: Construindo Novas Lógicas. In: Conferência eLES'04, 2004, Aveiras. Anais da Conferência eLES'04.

[AMD, 2009] AMD Advanced Micro Devices, Inc. AMD Xilleon™ 220 – Technical Overview. Disponível em: <http://ati.amd.com/products/xilleon220/index.html>. Último acesso em agosto de 2009.

[ANTONACCI, 2000] ANTONACCI, M. J., NCL: Uma Linguagem Declarativa para Especificação de Documentos Hipermídia com Sincronização Temporal e Espacial. Dissertação de Mestrado. Departamento de Informática, PUC-Rio, 2000.

[ARM, 2003] ARM Ltd. and Imagination Technologies. ARM MBX HR-S 3D Graphics Core – Technical Overview, 2003.

[ATI, 2009] ATI™. Imageon Overview. Disponível em: <http://ati.amd.com/products/handheld.html>. Último acesso em agosto de 2009.

[ATSC, 2009] ATSC - Advanced Television Systems Committee. Disponível em: <www.atsc.org>. Último acesso em agosto de 2009.

[ATSC, 2005] ATSC Standard, Advanced Common Application Platform (ACAP) – 2005.

[BATISTA, 2006] BATISTA, C. E. C. F. TV Digital - Java na sala de estar. Revista MundoJava, Edição 17. 2006.

[BATISTA, 2008] BATISTA, C. E. C. F. TVGrid: Computação em *grid* em uma Rede de TV Digital. Dissertação de Mestrado, Departamento de Informática, UFPB, João Pessoa, 2008.

[BECKER, 2009] BECKER, V. Concepção e Desenvolvimento de Aplicações Interativas para Televisão Digital. Dissertação de Mestrado, UFSC, Florianópolis, 2006.

[BRASIL, 2006] BRASIL. Ato Presidencial 4091/2006: Instituição do SBTVD, 26 de Novembro de 2003. Brasil. Disponível online em http://www.planalto.gov.br. Último acesso em 17/08/2009.

[BRETL et al., 2006] BRETL, W., MEINTEL, W. R., SGRIGNOLI, G., WANG, X., WEISS, S. M., SALEHIAN, K. ATSC RF, Modulation, and Transmission. In: Global digital television: Technology & Emerging Services Proceeding of the IEEE, vol 94, nº 01, 2006.

[BROADCOM, 2009] BROADCOM CORPORATION. BCM7030 - High Definition Video UMA Subsystem with 2D/3D Graphics. Disponível em: <http://www.broadcom.com>. Último acesso em agosto de 2009.

[BRUTZMAN e DALY, 2007] BRUTZMAN, D., DALY, L. X3D – Extensible 3D Graphics for Web Authors. Editora Elsevier, 1ª Ed, San Francisco, 2007.

[BOYLE et al., 2001] Boyle E., et al. The creation of MPEG-4 content and its delivery over DVB infrastructure. In Proceedings of the first Joint IEI/IEEE Symposium on Telecommunications Systems Research, Dublin, Ireland, 2001.

[BURDEA e COIFFET, 2003] BURDEA, G., C., COIFFET , P. Virtual Reality Technology. New York, Editora John Wiley & Sons, 2003.

[CESAR, 2005] CESAR, P. A Graphics Software Architecture for High-End Interactive TV Terminals. Dissertação de Doutorado, Helsinki University of Technology, Helsinki, 2005.

[CESAR et al, 2006] CESAR,P., VUORIMAA, P., VIERINEN, J. A graphics architecture for high-end interactive television terminals. ACM Transactions on Multimedia Computing, Communications, and Applications, vol. 2, pp. 343-357, 2006.

[DAVIDSON et al., 2006] DAVIDSON, G. A., ISNARDI, M. A., FIELDER, L. D., GOLDMAN, M. S., TODD, C. C. ATSC Video and Audio Coding. In: Global digital television: Technology & Emerging Services Proceeding of the IEEE, vol 94, nº 01, 2006.

[DIAS, 2009] Dias, G.P. Um Sistema de Realidade Virtual para Apoio ao Ensino de Classificação de Imagens. Dissertação de Mestrado. Departamento de Informática, UFPB, João Pessoa, 2009.

[DVB, 2009] DVB - Digital Video Broadcasting Project. Disponível em: <www.dvb.org>. Último acesso em agosto de 2009.

[ECMA-262, 2009] ECMA-262. ECMAScript Language Specification. Disponível em: <http://www.ecma-international.org>. Último acesso em agosto de 2009.

[FADI et al., 2005] FADI, C., COULTON, P., EDWARDS, R. Evolution of 3D Games on Mobile Phones. In: Proceedings of the IEEE Fourth International Conference on Mobile Business, Sydney, Austrália, pp 173-179, 2005.

[FEHN et al., 2006] FEHN, C., BARRE, R. and PASTOOR, S., **"**Interactive 3-DTV-concepts and key technologies**"**. Proceeding of. IEEE, vol. 94, nº 3 pp. 524-538, 2006.

[FOLEY et al., 1997] FOLEY, J., D., DAM, A. V., FEINER, S. K., HUGES, J., F. Computer Graphics: Principles and Practice. Boston, Editora Addison Wesley, 1997.

[GEM, 2009] GEM - Globally Executable MHP. DVB Document A140 – *Globally Executable MHP (GEM) Specification 1.1*". DVB Bluebook, 2009. Disponível em: < http://www.mhp.org/mhpgem12.htm >. Último acesso em agosto de 2009.

[GOOGLE, 2009] Google Inc. O3D Technical Overview. Disponível em: <http://code.google.com/apis/>. Último acesso em agosto de 2009.

[HAMER, 2007] HAMER, C. Creating Mobile Games. New York, NY, Editora Apress, 2007.

[HEARM e BAKER, 2004] HEARN, D., BAKER, M.P. Computer Graphics with OpenGL. Upper Saddle River, Editora Prentice Hall, 2004.

[HENDERSON et al., 2006] HENDERSON, J. G. N., BRETL, W. E., DEISS, M. S., GOLDBERG, A., MARKWALTER, B., MUTERSPAUGH, M., TOUZNI, A. ATSC DTV Receiver Implementation. In: Global digital television: Technology & Emerging Services Proceeding of the IEEE, vol 94, nº 01, 2006.

[HERRERO et al, 2003] HERRERO, C.,CESAR, P., VUORIMAA, P. Delivering MHP Applications into a Real DVB-T Network, Otadigi. In: Proceedings of the 6st IEEE International Conference on Telecommunications in Modern Satellite, Cable, and Broadcasting Services, Nis, Serbia and Montenegro, pp. 231-234, 2003.

[HOSCHKA, 2001] P. HOSCHKA et al. Synchronized multimedia integration language (SMIL) 2.0. W3C Recommendation, 2001.

[ILLGNER e COSMAS, 2001] ILLGNER, K., COSMAS, J. System concept for interactive broadcasting consumer terminals. In: Proceedings of the International Broadcasting Convention (IBC), Amsterdam, The Netherlands, 2001.

[ISO/IEC TR 13818-6, 1998] ISO/IEC TR 13818-6. Information technology — Generic coding of moving pictures and associated audio information - Part 6: Extensions for DSM-CC, 1998.

[ISO/IEC 13818-10, 1999] ISO/IEC 13818-10. Information technology - Generic coding of moving pictures and associated audio information — Part 10: Conformance extensions for Digital Storage Media Command and Control (DSM-CC), 1999.

[ISO/IEC 13818-1, 2000] ISO/IEC 13818-1. Information technology - Generic coding of moving pictures and associated audio information: Part 1: Systems, 2000.

[ITU, 2001] ITU Recommendation J.200. Worldwide common core – Application environment for digital interactive television services. 2001.

[ITU, 2003a] ITU Recommendation J.201. Harmonization of declarative content format for interactive television applications. 2003.

[ITU, 2003b] ITU Recommendation J.202. Harmonization of procedural content formats for interactive TV applications. 2003.

[JENSEN, 2005] JENSEN, J. F. Interactive television: New Genres, New Format, New Content. Second Australasian Conference on Interactive Entertainment, Sydney, Australia. ACM International Conference Proceeding Series, Vol. 123, pp. 89-96, 2005.

[JONES et al., 2006] JONES, G., DEFILIPPIS, J. M., HOFFMANN, H., WILLIAMS, E. A. Digital Television Station and Network Implementation. In: Global digital television: Technology & Emerging Services Proceeding of the IEEE, vol 94, nº 01, 2006.

[KAMEYAMA et al., 2003] KAMEYAMA, M., KATO, Y., FUJIMOTO, H., NEGISHI, H., KODAMA, Y., INOUE, Y., KAWAI, H. 3D Graphics LSI Core for Mobile Phone ”Z3D”. In HWWS ’03: ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland. Proceedings of the HWWS ’03, pp. 60–67, 2003.

[KAMMANN, 2005] KAMMANN, T. D. Interactive Augmented Reality in Digital Broadcasting Environments. Dissertação de Mestrado, Universitat Kolenz Landau, Koblenz, 2005.

[KHRONOS, 2009] Khronos Group. OpenGL ES Specification. Disponível em: < http://www.khronos.org/opengles/>. Último acesso em Agosto de 2009.

[LAMADON et al., 2003] J.L. LAMADON J. L., CESAR, P., HERRERO, C., Vuorimaa, P. Usages of a SMIL player in digital television. In Proceedings of the 7th IASTED International Conference on Internet and Multimedia Systems and Applications, Honolulu, Hawaii, pp. 579-584, 2003.

[LEITE et al., 2005] LEITE, L. E. C., BATISTA, C. E. C. F., SOUZA FILHO, G. L., KULESZA, R., ALVES, L. G. P., BRESSAN, G., RODRIGUES, R. F., SOARES, L. F. G. FlexTV – Uma Proposta de Arquitetura de Middleware para o Sistema Brasileiro de TV

Digital. In Revista de Engenharia de Computação e Sistemas Digitais, São Paulo , v. 2, pp 29-50, 2005.

[LÉVY, 2005] LÉVY, P. Cibercultura. São Paulo, Editora 34, 2005

[MAAD, 2002] Maad, S., Universal Access For Multimodal ITV Content: Challenges and Prospects, In: 7 th ERCIM Workshop on User Interfaces for All, Paris, France. *SPRINGER LNCS proceedings of the* ERCIM , 2002.

[MAAD, 2003a] MAAD, S. MARILYN: a novel platform for intelligent interactive TV (IITV). In Jacko, J.A., Stephanidis, C. (Org.),Human-Computer Interaction, Theory and Practice (Part 1), Lawrence Erlbaum Associates, Mahwah, NJ, Vol. 1 pp.1041-5, 2003.

[MAAD, 2003b] MAAD, S., The potential and pitfall of interactive TV technology: an empirical study. In: International Conference on Television in Transition, Inglaterra, 2003. Disponível em: < http://web.mit.edu/cms/mit3/papers/maad.pdf >. Último acesso em Agosto de 2009.

[MACHADO, 2003] MACHADO, L. S. A Realidade Virtual no Modelamento e Simulação de Procedimentos Invasivos em Oncologia Pediátrica. Tese de Doutorado. Escola Politécnica da Universidade de São Paulo, USP, 2003.

[MACHADO, 2009] MACHADO, L, S,; MORAES, R, M, ; SOUZA, D, F L. ; SOUZA, C. L. ; CUNHA, Í. L. L. "A Framework for Development of Virtual Reality-Based Training Simulators". Studies in Health Technology and Informatics, v. 142, p. 174-176, 2009.

[MAIOR, 2002] MAIOR, M. S., TV interativa e seus caminhos. Dissertação de mestrado profissional, UNICAMP, Campinas, 2002.

[MALERCZYK, 2003] MALERCZYK, C., KLEIN, K., WEIBEISEK, T., 3D reconstruction of sports events for digital TV. In Proceedings of the 11th International Conference in Central Europe on Computer Graphics, Visualization, and Computer Vision, Plzen, Czech Republic, 2003.

[MCREYNOLDS e BLYTHE, 2005] MCREYNOLDS, T., BLYTHE, D. Advanced Graphics Programming Using OpenGL. São Francisco, Editora Elsevier, 2005.

[MICHAEL e CHEN, 2006] MICHAEL, D., CHEN, S. Serious Games: Games that Educate, Train, and Inform. Thomson Course Technology, Boston, MA, 2006.

[MIRANDA FILHO et. al.] MIRANDA FILHO, S., LEITE, L. E. C., SOUZA FILHO, G. L., MEIRA, S., FLEXCM - A Component Model for Adaptive Embedded Systems. 31st Annual International Computer Software and Applications Conference, vol. 1, pp.119-126 2007.

[MORRIS e SMITH-CHAIGNEAU, 2005] MORRIS, S., SMITH-CHAIGNEAU, A. Interactive TV Standards – A Guide to MHP, OCAP and JavaTV. Burlington, MA, USA, Elsevier, Focal Press, 2005.

[MÖLLER e HAINES, 2002] MÖLLER, T., A., HAINES, E. Real-Time Rendering. Massachusetts, Editora A.K. Peters, 2002.

[NADALUTTI et al., 2006] NADALUTTI, D., CHITTARO, L., BUTTUSSI, F. Rendering of X3D Content on Mobile Devices with OpenGL ES. In: 9th International Conference on 3D web technology. Columbia, Maryland. Proceeding of 9th International Conference on 3D web technology. Columbia, 2006.

[OGRE, 2009] OGRE. OpenSource 3D Graphics Engine. Disponível em: <www.ogre3d.org>. Último acesso em agosto de 2009.

[OLAIZOLA et al., 2006] OLAIZOLA, I., G., MARTIRENA, I., B., KAMMANN, T., D. Interactive Augmented Reality in Digital Broadcasting Environment. In Proceeding of 4th European Interactive TV Conference (EuroITV),Atenas, Grécia, 2006.

[OLIVEIRA, et al., 2008] OLIVEIRA, F., TAVARES., T., CHAGAS., A., BURLAMAQUI, A., BRENNAD, E., SOUZA, G.,L. Experiences from the use of shared multimedia space for e-learning in Brazil primary schools. In: 3rd International Conference on Digital Interactive Media in Entertainment. And Arts (DIMEA 2008), Athenas. Proceeding of 3rd Internacional Conference on Digital Interactive Media in Entertainment. And Arts, New York, NY, vol. 349, pp. 91-98, 2008.

[PALMEIRA, 2009] PALMEIRA, A. F., SOUSA, M. F., TAVARES, T. A., SEGUNDO, R. M. C., FILHO, G. L. S.VestibaTV – An Interactive Program for Vestibular Training. Proceeding of EUROITV2009, Leuven, Bélgica, 2009.

[PANDA3D, 2009] PANDA3D. Free 3D Game Engine. Disponível em: <www.panda3d.org>. Último acesso em agosto de 2009.

[PENG et al., 2001] PENG, C., Cesar, P., Vuorimaa, P. Integration of Applications into Digital Television Environment. In: Proceedings of the 7th International Conference on Distributed Multimedia Systems, Taipei, Taiwan, pp 266-272, 2001.

[PULLES e SASNO, 2004] PULLES, R., SASNO, P. A set top box combining MHP and MPEG-4 interactivity. In: 2nd European Union Symposium on Ambient Intelligence. Eindhoven, The Netherlands, Proceedings of the 2nd European Union Symposium on Ambient Intelligence, pp. 31-34, 2004.

[PULLI et al., 2008] PULLI, K., AARNIO, T., VILLE, M., ROIMELA, K., VAARALA, J. Mobile 3D Graphics with OpenGL ES and M3G. Boston, Morgan Kaufmann, 2008.

[REIMERS, 2006] REIMERS, U. H. DVB—The Family of International Standards for Digital Video Broadcasting. In: Global digital television: Technology & Emerging Services Proceeding of the IEEE, vol 94, nº 01, 2006.

[REY-LOPEZ et al., 2007] REY-LOPEZ, M., DIAZ-REDONDO, R., FERNANDEZ-VILAS, A., PAZOS-ARIAS, J. Entercation: Engaging Viewers in Education Through TV. ACM Entertaint. Vol. 5, No. 2, Article 7, 2007.

[RICHARDSON, 2003] RICHARDSON, I. E. G. H.264 and MPEG-4 Video Compression. Aberdeen, Reino Unido, Editora John Wiley & Sons, 2003.

[RYCHLIK-PRINCE et al., 2009] RYCHLIK-PRINCE, C., MATZON, B., NAUR, E., TSAKPINIS, I. LWJGL – Lightweight Java Game Library. Disponível em: < http://lwjgl.org/ >. Último acesso em agosto de 2009.

[SELMAN, 2002] SELMAN, D. Java 3D Programming. Greenwich, Editora Manning, 2002.

[SHREINER et al., 2005] SHREINER, D., WOO, M., NEIDER, J., DAVIS, T. OpenGL Programming Guid (Red Book). Boston, Addison Wesley, 2005.

[SILVA, 2008] SILVA, L. D. N. Uma Proposta de API Para desenvolvimento de aplicações Multiusuário e Multidispositivo para TV Digital Utilizando o Middleware Ginga. Dissertação de Mestrado, João Pessoa, Departamento de Informática, UFPB, 2008.

[SINGÈS et al., 2000] SIGNÈS, J., FISHER, Y., ELEFTHERIADIS, A. MPEG-4's binary format for scene description, In: Signal Processing: Image Communication, vol. 15, pp. 321-345, 2000.

[SOARES, 2006] SOARES, L. F. G. MAESTRO: The Declarative Middleware Proposal for the SBTVD. Proceedings of the 4th European Interactive TV Conference. Atenas, 2006.

[SOARES et al., 2007] SOARES, L. F. G., RODRIGUES, R. F., MORENO, M. F. Ginga-NCL: the Declarative Environment of the Brazilian Digital TV System. In: Journal of the Brazilian Computer Society. Porto Alegre, RS, nº. 4, Vol. 13. pp.37-46. ISSN: 0104-6500, 2007.

[SOHN et al, 2004] SOHN, J.-H., WOO, R., AND YOO, H.-J. 2004. A Programmable Vertex Shader with Fixed-point SIMD Datapath for Low Power Wireless Applications. In HWWS '04: ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware. ACM Press, New York, NY, USA. Proceedings of the HWWS '04, pp 107-114, 2004.

[SOUZA FILHO et al., 2007] SOUZA FILHO, G. L., LEITE, L. E. C., BATISTA, C. E. C. F. Ginga-J: The Procedural Middleware for the Brazilian Digital TV System. In: Journal of the Brazilian Computer Society. Porto Alegre, RS, nº. 4, Vol. 13. pp.47-56. ISSN: 0104-6500, 2007.

[SUN, 2008]. Sun MicroSystems . Java DTV API 1.0 Specification. s.l : Sun Microsystems, 2008.

[SUN, 2009] Sun MicroSystems. JSR 184: Mobile 3D Graphics API for J2ME™. Disponível em: < http://jcp.org/jsr/detail/184.jsp>. Último acesso em agosto de 2009.

[TAVARES et al., 2004] TAVARES, T. A., BURLAMAQUI, A. M. F., ALBINO, D. A. S., SIMONETTI, C., LEITE, L. E. C, FERNANDES, J. H. C., FILHO, G. L. S. Sharing Virtual Acoustic Spaces over Interactive TV Programs: Presenting Virtual Cheering Application. In: International Conference on Multimedia and Expo (ICME), 2004, Taipei. ICME 2004 Proceedings, 2004.

[TONIETO, 2006] TONIETO, M. T.; Sistema Brasileiro de TV Digital – SBTVD - Uma Análise Política e Tecnológica na Inclusão Social; Mestrado Profissional em Computação Aplicada, UECE/CEFET, 2006;

[TREVETT, 2004] TREVETT, N. Khronos and OpenGL ES. Proceeding of Siggraph 04. Tokyo, Japão, 2004.

[TS 102 812, 2003] TS 102 812:2003. Digital video broadcasting (DVB) multimedia home platform (MHP). 2003.

[UEHARA, 2006]. UEHARA, M. Application of MPEG-2 Systems to Terrestrial ISDB (ISDB-T). In: Global digital television: Technology & Emerging Services Proceeding of the IEEE, vol 94, nº 01, 2006.

[VEIGA, 2006] VEIGA, E. G. Modelo de Processo de Desenvolvimento de Programas para TV Digital e Interativa. Dissertação de Mestrado, UNIFACS. 2006.

[VELHO e GOMES, 2001] VELHO, L., GOMES, J. Sistemas Gráficos 3D. Instituto de Matemática Pura e Aplicada – IMPA. Série de Computação e Matematica. Rio de Janiero, IMPA, 2001.

[VIRGÍNIO FILHO, 2009] VIRGÍNIO FILHO, R.T. Conteúdos Tridimensionais em Dispositivos Móveis: Um Estudo Aplicado ao Desenvolvimento de Jogos Educacionais para Celulares. Dissertação de Mestrado, Departamento de Informática, UFPB, João Pessoa, 2009.

[WATT, 2000] WATT, A. 3D Computer Graphics. Inglaterra, Editora Addison Wesley, 2000.

[WEB3D, 2009] WEB 3D Consortium. X3D-EARTH. Disponível em: http://www.web3d.org/x3d-earth/>. Último acesso em agosto de 2009.

[WU et al., 2006] WU, Y., HIRAKAWA, S., REIMERS, U.H., WHITAKER, J. Overview of Digital Television Development Worldwide. In: Global digital television: Technology & Emerging Services. Proceeding of the IEEE, vol 94, nº 01, pp 8-21, 2006.

[YAMADA et al., 2004] YAMADA, F., SUKYS, F., BEDICKYS, G, AKAMINE, C., RICHARDS JUNIOR, C., DANTAS, C. E. S. Sistema de TV Digital, Procedimento de Medidas. In: Revista Mackenzie de Engenharia e Computação. São Paulo, Editora Mackenzie, vol. 5, p. 13-268, 2004.

[ZUFFO, 2001] ZUFFO, M. K., A convergência da realidade virtual e Internet Avançada em novos paradigmas de TV Digital Interativa, Tese de Livre Docência, USP, 2001.