UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

GameTVD: UMA PROPOSTA DE ARQUITETURA PARA *FRAMEWORK* DE JOGOS 2D PARA TV DIGITAL

LADY DAIANA DE OLIVEIRA MAIA

Manaus
2010

UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

LADY DAIANA DE OLIVEIRA MAIA

GameTVD: UMA PROPOSTA PARA *FRAMEWORK* DE JOGOS 2D PARA TV DIGITAL

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Amazonas, como requisito parcial para a obtenção do título de Mestre em Informática, área de concentração em Sistemas Embarcados.

Orientador: Prof. Dr. José Pinheiro de Queiroz Neto
Co-orientador: Prof. Dr.-Ing. Vicente Ferreira de Lucena Junior

Manaus
2010

GameTVD: UMA PROPOSTA DE ARQUITETURA PARA *FRAMEWORK* DE JOGOS DIGITAIS PARA TV DIGITAL

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Amazonas, como requisito para a obtenção do título de Mestre em Informática, área de concentração em Sistemas Embarcados.

Aprovado em 23 de agosto de 2010

BANCA EXAMINADORA

Prof. Dr. José Pinheiro de Queiroz Neto
Instituto Federal do Amazonas - IFAM
Campus Manaus Distrito Industrial

Prof. Dr. Raimundo da Silva Barreto
Universidade Federal do Amazonas – UFAM
Departamento de Ciência da Computação

Prof. Dr. Cícero Augusto Mota Cavalcante
Universidade Federal do Amazonas – UFAM
Instituto de Ciências Exatas - Itacoatiara

Prof. Dr.-Ing. Vicente Ferreira de Lucena Junior
Universidade Federal do Amazonas
Faculdade de Tecnologia

Ao meu Pai, Sr. Luiz Edilson F. Pinto, pelo incentivo aos meus estudos, pela dedicação a família e por ter me dado a oportunidade que ele não recebeu.

AGRADECIMENTOS

Primeiramente a Deus pela sabedoria dada para superação de mais esse obstáculo.

A minha família o meu muito obrigado, minha mãe Aldaci e meu amado marido Luciano pela compreensão e o apoio em mais essa etapa na minha vida.

Ao meu orientador Dr. José Pinheiro e co-orientador Dr. Vicente Lucena por compartilharem comigo a sua competência e me auxiliarem durante esse projeto.

Aos colegas do Laboratório de TV digital da Universidade Federal do Amazonas – UFAM: Vandermi, Antônio, Marcelo, Ricardo, Alexandre, Orlewilson, Luciano e Wanderlan que muitas vezes me ajudaram tirando dúvidas. Em especial para o aluno de graduação Adriano, que contribuiu para o desenvolvimento dos jogos.

A todos os professores do Departamento de Ciência da Computação da UFAM que contribuíram diretamente ou indiretamente para formação dessa pesquisa.

Você não sabe
O quanto eu caminhei
Prá chegar até aqui
Percorri milhas e milhas
Antes de dormir
Eu nem cochilei
Os mais belos montes
Escalei
Nas noites escuras
De frio chorei.

Comp.: Toni Garrido / Lazão / Da Gama / Bino.

RESUMO

A TV digital está em fase de implantação e iniciando a sua popularização no Brasil e por isso são poucas as experiências consolidadas no desenvolvimento de aplicações interativas para o modelo brasileiro de TV digital, configurando-se assim, um desafio para pesquisadores, técnicos e empresários, principalmente no que diz respeito a jogos digitais, pois a indústria de jogos cresceu no último ano 31% nos diversos tipos de plataforma existentes. A proposta desse trabalho é investigar o ambiente de TV digital, fazendo um levantamento de requisitos para jogos dentro desta plataforma, propondo em seguida uma arquitetura para um *Framework* com o objetivo de auxiliar o processo de desenvolvimento de jogos. Para tanto, segue-se o conceito de um *Framework*, que tem como objetivo auxiliar a criar padrões de projeto típicos de jogos, facilitando a construção da interface gráfica, do tratamento de eventos e do controle da navegação. Para atender ao propósito do trabalho, são apresentados os conceitos de jogos eletrônicos no contexto de convergência digital, metodologias de engenharia de software para construção de *Framework*s de jogos e as especificações do padrão televisivo brasileiro, levando em consideração as limitações típicas de aplicações para TV digital. Faz parte do escopo da proposta identificar e detalhar as etapas necessárias para a elaboração do *Framework*, bem como apresentar um estudo de caso como prova de conceito, um jogo para TV digital desenvolvido utilizando o *Framework* proposto. Desta forma, pretende-se evidenciar que a plataforma contribui de forma positiva para os desenvolvedores de jogos para a TV digital, evitando a preocupação com as características específicas da plataforma e da implementação de rotinas necessárias em quase todos os jogos digitais.

Palavras chave – TV digital, jogos digitais, *Framework,* engenharia de software, desenvolvedor, interface gráfica.

ABSTRACT

The Digital TV is in its consolidation phase and has begun its popularization in Brazil. Therefore the experiences acquired in the development of interactive applications for the Brazilian model of Digital TV are few, and constitute a challenge for researchers, technicians and entrepreneurs, especially in regard to Digital Games, since the game industry has grown 31% last year in various types of existing platforms. The proposal of this study is to investigate the environment of Digital TV, conducting a survey of the requirements for games to be run in this platform, then propose an architecture for a framework aiming at assisting the process of games development. For this, it follows the concept of a framework for games that has as its objective to assist in the creation of design patterns typical of games, thus facilitating the construction of graphical user interfaces, event handling and navigation control. To meet the purpose of the paper, the concepts of electronic games in the context of digital convergence are presented, along with software engineering methodologies for building game frameworks and the specifications of the Brazilian television standard, which could impose to the project the limitations of a typical application for Digital TV. This proposal also encompasses the identification and detailing of the stages necessary for the elaboration of a framework, as well as the presentation of a case study as proof of concept, namely a game for Digital TV developed using this framework on a PC. With this goal, it is intended to demonstrate that the platform contributes positively to the software development for digital TV, protecting the developers from specific characteristics of the platform and the implementation of routines needed in almost all digital games.

Keywords - Digital TV, digital games, framework, software engineering, developer, graphical user interface.

LISTA DE FIGURAS

Figura 1.1: Cronograma de implantação da TV digital no Brasil. 14
Figura 2.1: Ilustração comparativa de *Framework*s: Extremamente geral ou específica. 26
Figura 2.2: Modelo de um sistema de televisão digital. 26
Figura 2.3: Camadas de Tecnologia do ISDTV-T. 27
Figura 2.4: Modelo de referência do middleware Ginga. 30
Figura 2.5: Cenário A da arquitetura do Ginga-J com GEM 1.1. 31
Figura 2.6: Arquitetura do Ginga-J com JavaDTV 1.3. 32
Figura 2.7: Especificação JavaDTV 1.3. 33
Figura 2.8: Ciclo de ação do *xlet*. 36
Figura 2.9: *Xlet Pong* executado no *XletView*. 40
Figura 2.10: Diagrama de pacotes do Ginga Game. 42
Figura 2.11: Arquitetura do *Framework* wGem. 44
Figura 2.12: Representação em relação a especialização e utilização. 46
Figura 2.13: Interface RummiTV. 46
Figura 2.14: Inerface do jogo T-lerning Hackerteen Prototype. 47
Figura 3.1: Fases do desenvolvimento do *Framework* GameTVD. 48
Figura 3.2: Diagrama de blocos do *Framework* GameTVD. 49
Figura 3.3: Ciclo de vida de um jogo. 51
Figura 3.4: Modelo da arquitetura conceitual do *Framework*. 52
Figura 3.5: *Framework* GameTVD 54
Figura 3.6: Jogos de Plataforma: *PAC-MAN* (a) e Mário Brós (b). 55
Figura 3.7: Diagrama de classes do *Framework* GameTVD 58
Figura 4.1: Interface Principal e a New Project. 62
Figura 4.2: Estrutura de pastas do sistema. 62
Figura 4.3: Arquivo descritor da estrutura do jogo. 63
Figura 4.4: Interface para criação de sprites. 65
Figura 4.5: Interface gráfica da classe *createSound*. 66
Figura 4.6: Interface gráfica *Create Room*. 67
Figura 4.7: Interface Create Object. 68
Figura 4.8: Opções de Eventos. 69
Figura 4.9: Ações de um objeto. 70
Figura 4.10: Processo de conversão do jogo para TV digital. 73
Figura 5.1: Criando *room* inicial. 77
Figura 5.2: *Room* inicial do jogo no emulador. 77
Figura 5.3: Sprites do jogo *Pac-Man*. 78
Figura 5.4: *Room* desenvolvido no GameTVD. 78
Figura 5.5: Objeto *Pac-Man* 79
Figura 5.6: *Room* do jogo *Pac-Man* for TV. 80
Figura 5.7: *Room* de ajuda do jogo *Pac-Man* for TV. 80
Figura 5.8: *Room* final do jogo *Pac-man* for TV. 81
Figura 5.9: Interface inserir background. 81
Figura 5.10: Interface inserir evento no objeto nave. 82

Figura 5.11: Jogo *Space Invaders*. 83

LISTA DE QUADROS

Quadro 1: Criando os diretórios. 63
Quadro 2: Criando descritor para carregar a arvore posteriormente. 63
Quadro 3: Classe Copia utilizada para adicionar a imagem no diretório *Sprites*..... 65
Quadro 4: Código do *xlet* gerado. 72
Quadro 5: Compilar *xlet*..... 73
Quadro 6: Executar o emulador..... 74

LISTA DE ABREVIAÇÕES

2D	Duas Dimensões
3D	Três Dimensões
ARIB	*Association of Radio Industries and Busines*
ATSC	*Advanced Television System Committee*
AWT	*Abstract Window Toolkit*
BARB	*Brodcast Audience Research Board*
BML	*Broadcast Markup Language*
CDC	*Conected Device Configuration*
DASE	*DTV Application Software Enviromente*
DAVIC	*Digital Áudio-Visual Council*
DVB	*Digital Video Broadcasting*
EPG	Guia de Programação Eletrônica
FP	*Foundation Profile*
GBF	*Gamework's Brazilian Framework*
GEM	*Globally Executable MHP*
GNU	*General Public License*
HAVI	*Home Audio Video*
HTML	*HyperText Markup Language*
IPTV	*Internet Protocol TV*
ISDB	Sistema Integrado de Radiodifusão
ISDTV-T	Sistema Internacional de Televisão Digital – Terrestre
JMF	*Java Virtual Machine*
LAVID	Laboratório de Aplicações de Vídeo Digitais
LWUIT	*LightWeight User Interface Toolkit*
MHP	*Multimedia Home Plataforma*
NCL	*Nested Context Language*
OpenGL	*Open Graphics Library*
PBP	*Personal Basis Profile*
PC	Computador Pessoal;
PDA	P*ersonal Digital Assistant*
PUC-Rio	Pontifícia Universidade Católica do Rio de Janeiro
PVR	*Personal Video Recoder*
SI	Informação de Serviços
STB	*Set-Top Box*
TVDI	TV digital Interativa
UFAM	Universidade Federal do Amazonas
UFPB	Universidade Federal da Paraíba.
VRML	*Virtual Reality Modeling Language*
XHTML	*Extensible Hypertext Markup Language*
XML	*Extensible Murkup Language*

SUMÁRIO

1 INTRODUÇÃO 14

1.1 Motivação/Justificativa 17

1.2 Problema 18

1.3 Objetivos 19

1.3.1 Objetivo Geral 19

1.3.2 Objetivos Específicos 19

1.4 Metodologia da pesquisa 19

1.5 Organização do Trabalho 20

2 *FRAMEWORK*S DE JOGOS E TELEVISÃO DIGITAL: PLATAFORMAS DE DESENVOLVIMENTO 21

2.1 Jogos para computadores e TVs 21

2.2 Framework 22

2.2.1 Classificação dos *Framework*s 23

2.2.2 Desenvolvimento de *Frameworks* de Aplicação 24

2.2.3 Vantagens e Desvantagens de *Framework* de Jogos 25

2.3 Sistema e Padrão da TV digital 26

2.3.1 *Set-Top Box* - STB 28

2.3.2 Interface entre *software* e *hardware: Middleware* 28

2.3.3 GINGA: NCL ou Java 29

2.3.4 API JAVATV 35

2.3.5 *Xlets* – Linguagem em GINGA-J 36

2.3.6 Classificação das aplicações para TV digital 37

2.4 Trabalhos Relacionados 38

3 ESPECIFICAÇÃO DO DESENVOLVIMENTO DO *FRAMEWORK* GAMETVD 48

3.1 Levantamento da arquitetura 48

3.1.1 Limitações de Interfaces gráficas para TVD. 53

3.1.2 Arquivo gerado *xlet.* 54
3.1.3 Especificação dos jogos 2D 55
3.2 Análise e Modelagem 56
3.2.1 Bibliotecas para TVDI 57
3.2.2 Diagrama de Classe 57
3.3 Ferramentas de Desenvolvimento 59
3.3.1 Plataforma *Eclipse* e *Netbeans* 59
3.3.2 Emulador *XletView* 60
4 GAMETVD: *FRAMEWORK* DE JOGOS DE PLATAFORMA PARA TV DIGITAL 61
4.1 Processo Principal 61
4.2 Processos de adicionar objetos 64
4.2.1 Criando *Sprites*. 65
4.2.2 Criando Sons 66
4.2.3 Criando Cenários 66
4.3 Processo de tratamento de eventos 68
4.3.1 Criando objetos 68
4.4 Processo de conversão do jogo para TV digital 71
5 DESENVOLVENDO JOGOS COM O GAMETVD: ESTUDO DE CASO 75
5.1 Jogos Desenvolvidos 76
5.1.1 Pac-Man 76
5.1.2 Space Invaders 81
5.2 Testes e Resultados 83
6 CONSIDERAÇÕES FINAIS 85
6.1 Dificuldades encontradas 86
6.2 Trabalhos Futuros 86
7 REFERÊNCIAS BIBLIOGRÁFICAS 87
APENDICE A 92

1 INTRODUÇÃO

Atualmente no Brasil, a Televisão Digital Interativa – TVDI é aberta e está se tornando uma realidade, pois o governo Federal vem promovendo Políticas Públicas visando à implantação de infra-estrutura necessária para que até o ano de 2016 esta tecnologia esteja funcionando em todos os estados. Em relação ao estado do Amazonas, apenas a capital Manaus está recebendo o sinal digital e ainda não são todas as emissoras que estão com a tecnologia de TV Digital.

Por enquanto, DTV (2009) afirma que são vinte e um estados, principalmente as capitais, que estão transmitindo o sinal digital com prazos estipulados referente ao período de início e término do processo de implantação da televisão digital em diferentes regiões do território nacional, conforme disposto pelo Ministério das Comunicações (FÓRUM SBTV, 2006) e ilustrado no cronograma da Figura 1.1.

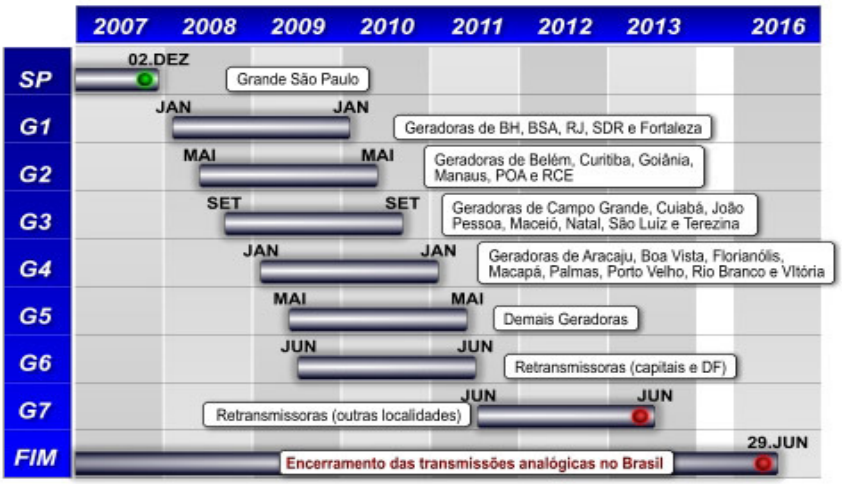


Figura 1.1: Cronograma de implantação da TV digital no Brasil.
Fonte: (DTV, 2009).

O ISDB-TB - Sistema Internacional de Rádio Difusão Terrestre Brasileiro é o padrão brasileiro, grátis, livre e que permite imagens e som com alta qualidade, apresentando características como portabilidade e mobilidade, execução de aplicações interativas e de múltiplos programas simultâneos. O ISDB-TB é baseado no Sistema Integrado de

Radiodifusão Digital - ISDB desenvolvido no Japão e adotado pelo Brasil com atualizações tecnológicas nos padrões de áudio, vídeo e interatividade. A seleção do padrão adotado pelo Brasil aconteceu após ampla disputa entre os três padrões existentes mundialmente, que são o *Advanced Television System Committee* - ATSC (padrão americano), o *Digital Video Broadcasting* - DVB (padrão europeu) e o ISDB (padrão japonês) considerado pelos especialistas o mais adequado para a nossa realidade socioeconômica (FÓRUM SBTV, 2009).

Entre os objetivos da TVDI está proporcionar a democratização do acesso à informação para a população nos próximos anos e ser uma importante aliada na educação à distância, sendo possível distribuir conteúdos educacionais a populações geograficamente distribuídas e permitir uma maior interatividade entre os seus usuários, se tornando uma ferramenta de inclusão social devido sua penetração nos lares brasileiros. O conceito de TV interativa diz que o telespectador pode passar a interagir com a programação em tempo real, escolhendo ângulos de câmera, acessando aplicativos interativos, tendo uma comunicação bidirecional entre a emissora e o telespectador, que passa a ter uma participação mais efetiva no conteúdo. A interatividade na TVDI envolve também aspectos financeiros como o aumento da qualidade e quantidade dos serviços oferecidos (SILVA et. al., 2008).

Em se tratando de jogos para TV digital, Ferreira e Souza (2008) afirmam que as iniciativas são menores ainda, pois em um jogo eletrônico são necessários vários recursos computacionais para que seja possível atender as expectativas do usuário e, além disso, existem algumas limitações impostas pelo ambiente, como por exemplo, a interface com o jogo através de um controle remoto, a escassez de ferramentas para auxiliar o desenvolvimento e ainda a falta de uma estratégia para projeto de jogos.

Apesar disso tudo, o fato é que uma das maneiras interessantes de se utilizar a TV digital para fins educacionais é a utilização de jogos, que deve ser considerado como um recurso didático pedagógico, favorecendo o processo ensino-aprendizagem. Segundo Tarouco

(2004), ao se pensar na utilização dos jogos educacionais é necessário primeiramente defini-lo como uma ferramenta que facilita o aprendizado e aumenta a capacidade de retenção do que foi ensinado. A sua utilização possui o intuito de despertar o interesse do educando no aprendizado, ou seja, se tornar um elemento de estímulo motivador no processo de ensino-aprendizagem em crianças e jovens.

Um jogo tem a função de consolidar os esquemas já formados e dar prazer ou equilíbrio emocional a criança. Existe uma série de vantagens na utilização dos jogos em ambiente escolar, o despertar da motivação, curiosidade e estímulos, além de contribuir para o crescimento intelectual e afetivo, pois o conhecimento é adquirido de maneira prazerosa e lúdica.

Segundo Clua e Bittencourt (2004) o jogo pode ser educacional ainda que não possua objetivos pedagógicos explícitos, pelo fato de desenvolverem habilidades cognitivas importantes para o processo de aprendizagem como resolução de problemas, percepção, criatividade, raciocínio rápido, entre outras habilidades. Quando um jogo é criado especificamente com o propósito de conteúdos pedagógicos e para ser utilizado dentro do âmbito escolar, denomina-se de jogo didático.

Nesse contexto, observa-se a necessidade da criação de recursos para suporte ao desenvolvimento dessas aplicações, pois além dos jogos exercerem um poder atrativo e motivador, eles podem possibilitar a inclusão social, que é um dos objetivos do Sistema de TV digital do Brasil.

Neste trabalho, será apresentada a tecnologia de TV digital brasileira e os requisitos necessários para o desenvolvimento de jogos para esse ambiente, levando em consideração as especificações do *middleware* Ginga, a fim de propor uma arquitetura para *Framework* de jogos digitais capaz de gerar jogos para serem executados em sistemas televisivos digitais.

1.1 Motivação/Justificativa

As pesquisas realizadas sobre as tecnologias de TV Digital estão em fase de crescimento e existem diversos trabalhos consolidados em países que possuem essa tecnologia há mais tempo que o Brasil, por isso as pesquisas no Brasil sobre o tema Sistema Brasileiro de TV digital é importante, já que o país encontra-se em fase de implantação dessa tecnologia.

Empreendimentos voltados para a TVDI configuram um desafio para pesquisadores, técnicos e empresários, lembrando que a indústria brasileira é hoje responsável por 0,16% do faturamento mundial com jogos eletrônicos, um número muito modesto para a indústria brasileira. Por outro lado, aconteceu um crescimento de 31% da indústria de jogos entre 2008 e 2009 o que é um sinal claro do fortalecimento da indústria brasileira de entretenimento digital (ABRAGAMES, 2009). Esse cenário justifica a utilidade desse trabalho, pois se verifica que a indústria está demandando claramente iniciativa deste tipo.

Por se tratar de uma tecnologia recente, foi percebida a necessidade da criação de recursos para suporte ao desenvolvimento de aplicações para a plataforma de TV digital, sendo assim, este trabalho pretende proporcionar ampliação do nível de conhecimento sobre a tecnologia necessária para o desenvolvimento de *Framework*s de jogos para TV digital, onde apresentará os conceitos relevantes da tecnologia, visto que se trata de um assunto em estado da arte.

Além disso, por ser uma tecnologia adotada por outros países é necessário evitar a importação de aplicações e promover o desenvolvimento de *softwares* voltados para a necessidade do modelo brasileiro de TV digital, a fim de despertar na comunidade científica o interesse no desenvolvimento de aplicativos interativos para TV digital na Educação, incentivando a criação da infra-estrutura necessária para o desenvolvimento de *software*.

1.2 Problema

A produção de um jogo envolve várias atividades, como produção artística, elaboração de roteiros, implementação em uma linguagem de programação. Sendo necessário o conhecimento em diversas áreas como linguagem de programação, sistemas operacionais, computação gráfica, engenharia de software e até inteligência artificial. Por isso é importante a criação de ferramentas de suporte ao desenvolvimento desses jogos para diversos tipos de plataformas, sejam plataformas móveis, computadores, consoles e até mesmo para TV digital (COSTA et. al. 2008).

O desenvolvimento de jogos para TVD no cenário brasileiro está sendo pouco explorado, e um dos motivos é o fato de que para desenvolver estes jogos é necessário um profissional experiente com qualificações variadas, que domine as tecnologias associadas ao desenvolvimento de aplicações para TVD.

Essa carência acontece principalmente devido às dificuldades quanto ao desenvolvimento destes jogos, tais como a falta de suporte à manipulação de recursos gráficos, sonoros e sua otimização, a falta de ferramentas para auxiliar no desenvolvimento do sistema, e ainda o fato de não haver uma padronização na construção dos jogos (FERREIRA e SOUZA, 2008).

Dadas as dificuldades na etapa de implementação do jogo ou da aplicação para cada tipo de plataforma, é possível apontar uma preocupação predominante quanto à elaboração e o uso de ferramentas que facilitem o processo de criação de jogos específicos para plataforma TVD de forma ágil e fácil. Portanto, o problema tratado neste trabalho consiste em como reduzir as dificuldades associadas à carência de ferramentas relacionadas a tecnologia de TVD para o desenvolvimento de jogos.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral dessa proposta é investigar os elementos tecnológicos do modelo brasileiro de TV digital e propor o desenvolvimento de um *Framework* baseado nesta arquitetura que facilite a criação de interfaces gráficas de jogos;

1.3.2 Objetivos Específicos

Os objetivos específicos são:

a) Identificar os elementos necessários para o desenvolvimento de jogos para TV digital;

b) Implementar as funcionalidades de suporte ao tratamento de eventos e controle de navegação no ambiente da Televisão Digital em um *Framework* para o desenvolvimento de jogos;

c) Desenvolver um jogo com o *Framework* como estudo de caso para verificar a funcionalidade do ambiente proposto.

1.4 Metodologia da pesquisa

O processo adotado para o desenvolvimento do projeto envolveu atividades iniciais como a concepção e fundamentação da teoria relacionada, com o estudo da metodologia para o desenvolvimento do *Framework* levando em consideração os conceitos de engenharia de software e o levantamento da arquitetura necessária para o desenvolvimento de jogos para o modelo brasileiro de TV digital. O objetivo dessa fase foi adquirir o conhecimento teórico necessário para o desenvolvimento do trabalho.

Em seguida, implementou-se o *Framework* com a arquitetura necessária para desenvolvimento de jogos no ambiente de TV digital, levando em consideração os padrões,

arquiteturas e protocolos do Sistema Brasileiro de TV digital. Esta fase teve o objetivo de implementar as classes necessárias do *Framework* embasadas no domínio da aplicação. A próxima fase foi à avaliação funcional do ambiente proposto com um estudo de caso, no qual foi desenvolvido um jogo com o *Framework e* aplicados alguns testes utilizando um emulador de TV digital.

1.5 Organização do Trabalho

O trabalho é constituído por seis capítulos, onde no primeiro capítulo é apresentada à introdução do trabalho com o contexto no qual o mesmo se insere, a justificativa, e os objetivos do trabalho e a metodologia utilizada.

A fundamentação teórica sobre os conceitos necessários para o desenvolvimento de um *Framework* e o detalhamento das tecnologias para o entendimento do sistema brasileiro de TV digital são mostrados no Capítulo 2, além disso, é apresentada uma revisão dos trabalhos relacionados com o tema da proposta.

No Capítulo 3 é apresentada a metodologia para o desenvolvimento de um *Framework*, que consiste no levantamento da arquitetura, os principais requisitos, análise e modelagem do software e as ferramentas de desenvolvimento, levando em consideração os *Framework*s estudados no capítulo anterior.

No Capítulo 4 apresenta a ferramenta desenvolvida com as suas principais funcionalidades, em seguida, o estudo de caso é descrito no Capítulo 5, no qual são detalhados os resultados e discussões referentes ao jogo desenvolvido utilizando o *Framework* proposto e no Capítulo 6 são apresentadas as considerações finais e proposta de trabalhos futuros.

2 *FRAMEWORK*S DE JOGOS E TELEVISÃO DIGITAL: PLATAFORMAS DE DESENVOLVIMENTO

Para o desenvolvimento do trabalho é necessário a descrição de alguns conceitos referentes ao desenvolvimento de um *Framework* e detalhes de como é formado o Sistema de Televisão Digital no Brasil.

2.1 Jogos para computadores e TVs

Existem diversas definições para jogos, e geralmente são conceituados como formas de entretenimento, porém, segundo Jaccober (2007) é difícil uma definição padrão sobre jogos, e uma das definições afirma que o jogo é uma transgressão sobre a diversão, acrescentando elementos não convencionais de estrutura e regras a uma atividade lúdica.

Em se tratando de jogos de computadores, pode-se conceituar como um programa de entretenimento onde a plataforma é um computador pessoal ou ainda, como um sistema interativo que permite ao usuário experimentar, sem riscos, uma situação de conflito, sendo formados pelo enredo que consiste no tema, nas regras e os objetivos a serem alcançados (BATTAIOLA, 2000).

É interessante pensar em jogos no sentido de estarem em um novo ambiente de execução, o qual os brasileiros ainda não estão acostumados, configurando a união de dispositivos como TV, celular, Computador Pessoal - PC, P*ersonal Digital Assistant* - PDA, consoles e outros, a fim de que o jogador possa ter um jogo a qualquer momento e lugar.

A convergência da televisão com o universo dos jogos digitais pode aumentar ainda mais os índices de desenvolvimento das indústrias de jogos e das emissoras de TVs, especialmente pelas vantagens que a TV digital oferece. O Reino Unido é líder em termos de penetração de TV digital e desenvolvimento de serviços para TV Interativa, tendo apresentado em março de 2003, índices de 1.300 milhões de utilizadores dos jogos disponíveis na Sky

Digital, sendo que 40% eram mulheres. O canal de jogos *PlayJam* da Sky Digital, consta entre o 8º e o 15º lugar dos canais mais vistos do serviço da Sky, de acordo com o *Broadcast Audience Research Board* - BARB, o órgão oficial no Reino Unido para medição de audiências (QUICO, 2009).

2.2 Framework

Na literatura existem diversas definições sobre o conceito de *Framework*. Neste trabalho, utilizou-se a definição de (JOHNSON, 1991) e (GAMMA et. al., 1995), onde um *Framework* é definido como um conjunto de objetos que colaboram com o objetivo de atender a um conjunto de responsabilidades para uma aplicação específica ou um domínio de aplicação.

Um *Framework* tem o objetivo de dar suporte ao desenvolvimento de aplicações pertencentes ao mesmo domínio de problema, ou seja, a um domínio específico, sendo assim, pode-se conceituar um *Framework* de *software* como uma implementação de um conjunto de classes em uma linguagem específica, com o intuito de auxiliar no processo de desenvolvimento de outros *softwares* e aumentar o reuso.

Em (SILVA e OLIVEIRA, 2006), afirmam que o papel do *Framework* é prover uma orientação arquitetural ao desenvolvedor apresentando um modelo com classes abstratas e definindo suas responsabilidades e colaborações. Em relação ao papel do desenvolvedor, tem-se a customização do *Framework* para uma aplicação particular, pois este fornece uma implementação para as funções básicas e constantes e ainda inclui um mecanismo para permitir que o desenvolvedor se conecte às diversas funções.

No geral, os *Framework*s devem possuir algumas características básicas, como ser fácil de usar (usabilidade), permitir a reutilização (reusabilidade), permitir implementação (extensível) e principalmente atender ao domínio do problema. Essas características mostram

os benefícios de se utilizar um *Framework*. Porém, existem alguns obstáculos no desenvolvimento de um *Framework* devido à alta complexidade envolvida no projeto.

2.2.1 Classificação dos *Framework*s

Segundo Barreto (2006) os *Framework*s podem ser classificados como *Framework*s de aplicação orientados a objetos e *Framework*s de componentes. O primeiro define uma estrutura para o desenvolvimento de aplicações orientadas a objetos enquanto que o *Framework* de componentes define uma infra-estrutura de execução, onde componentes podem ser interligados. Além disso, os *Framework*s de aplicação podem ser classificados de acordo com o seu escopo, ou seja, com o modo que será empregado, são classificados como (ASSIS e SUZANO, 2003):

a) ***Framework* de Infra-estrutura de Sistemas**: Denominados como *Framework*s horizontais, são utilizados para facilitar a construção de sistemas operacionais, interfaces gráficas e ferramentas de processamento de linguagens, buscando portabilidade e eficiência;

b) ***Framework*s de Integração de *Middleware***: Utilizados na comunicação de ambientes para integrar aplicações e componentes distribuídos. Os *Framework*s ORB (*Object Request Brocker*) e bancos de dados transacionais são exemplos deste tipo de *Framework*;

c) ***Framework*s de Aplicações Corporativas**: Conhecidos como *Framework*s verticais, são empregados no desenvolvimento de aplicações comerciais para usuários finais e são os mais caros para serem desenvolvidos quando comparados com outros tipos de *Framework*. Como exemplo, tem-se os *Framework*s para aplicações financeiras ou sistemas contábeis.

Os *Framework*s podem ainda serem classificados em caixa-branca e caixa-preta ou caixa-cinza, quanto ao modo usado para constituí-lo. Os *Framework*s de caixa-branca são

baseados nas características de herança e ligações dinâmicas presentes na orientação ao objeto, sendo necessário que o desenvolvedor tenha conhecimento da estrutura interna do *Framework* e são mais difíceis de usar. Quanto aos de caixa-preta baseiam se em composição de objetos, possuindo uma maior facilidade de uso, mas são menos flexíveis. Finalmente, são classificados como caixa-cinza, a união das características dos *Framework*s caixa-branca e caixa-preta, para diminuir as desvantagens dos dois (ASSIS e SUZANO, 2003).

2.2.2 Desenvolvimento de *Frameworks* de Aplicação

No desenvolvimento de um *Framework* busca-se por generalidade em relação aos conceitos e funcionalidades do domínio, pois um *Framework* é uma abstração de um domínio de aplicações. Além disso, são necessárias as características de alterabilidade (alterar as funcionalidades presentes) e extensibilidade (ampliar a funcionalidade presente) para o projeto de um *Framework* (ASSIS e SUZANO, 2003).

Para a construção de um *Framework* deve-se produzir uma estrutura de classes tendo a capacidade de adaptar-se a um conjunto de aplicações diferentes, porém no mesmo domínio, tendo em vista que é necessário dispor de modelagens de um conjunto significativo de aplicações do domínio, como no caso da metodologia "Projeto Dirigido por Exemplo" (SILVA, 2000).

O Projeto Dirigido com o método "Por Exemplo" específica que o desenvolvimento de um *Framework* para um domínio de aplicação é decorrente de um processo de aprendizado a respeito deste domínio, que se obtém a partir do estudo de aplicações desenvolvidas. Pois, o pensamento humano realiza-se de maneira concreta e não abstrata, por isso o *Framework* é obtido por meio da generalização de casos concretos (aplicações), sendo o próprio *Framework* uma abstração de domínio. As abstrações são obtidas de um modo *bottom-up*, a partir de exemplos concretos são criadas as classes (BULARMAQUI, 2004).

A metodologia “Projeto Dirigido Por Exemplo” é um processo de desenvolvimento que utiliza etapas de análise, projeto e teste. Segundo (SILVA, 2000) a etapa de análise é referente à coleta de exemplos de programas que poderiam ser desenvolvidos a partir do *Framework* e avaliar a adequação de cada exemplo. A próxima fase trata do projeto das classes para abranger os exemplos estudados. Na fase de teste, é necessário testar o *Framework* usando-o para desenvolver uma aplicação.

2.2.3 Vantagens e Desvantagens de *Framework* de Jogos

O desenvolvimento de jogos para plataformas com limitações, como é o caso do celular e da TV digital, é uma tarefa desafiadora, pois além das dificuldades encontradas para o seu desenvolvimento, devem-se levar em conta as características inerentes a estes dispositivos, tais como: memória, processamento, formato da tela e dispositivos de entrada.

A utilização de *Framework*s de jogos permite fornecer um conjunto de componentes projetados especialmente para um domínio específico, visando a reusabilidade de código e ainda a diminuição do tempo de desenvolvimento dos jogos. Os componentes comuns de um jogo estão implementados previamente, permitindo a reutilização dos mesmos e com isso um desenvolvimento mais rápido de novos jogos. Outra vantagem da utilização de um *Framework* de jogos é o desenvolvimento de jogos similares, fazendo com que o desenvolvedor torne-se familiar ao tipo do jogo e ao *Framework*, aumentando sua produtividade.

No entanto, existem algumas desvantagens na utilização de *Framework*. Segundo Barros (2003) uma das grandes desvantagens no uso dos *Framework*s ocorre devido ao fato de que os jogos podem ser completamente diferentes uns dos outros.

Desta forma, utilizar uma arquitetura muito geral, que busque atender a todos os tipos de jogos fatalmente irá se tornar muito complexo, o que poderá inviabilizar a construção de jogos, ou ainda criar uma arquitetura especialista que servirá apenas para um determinado tipo

de jogo, o que foge ao propósito do *Framework*. Portanto, é preciso estar atento no momento do desenvolvimento do *Framework*, evitando-se construir algo muito geral ou específico, conforme analogia do elefante sendo um *Framework* extremamente geral e o robô específico apenas a um item do jogo, demonstrado na Figura 2.1.

Figura 2.1: Ilustração comparativa de *Framework*s: Extremamente geral ou específica.
Fonte: (Barros, 2003).

2.3 Sistema e Padrão da TV digital

Um Sistema de Televisão é composto por um Difusor, Meios de Difusão e um Sistema de Recepção, que consistem respectivamente na informação gerada pelo estúdio, a transmissão dessas informações e a recepção dos sinais na televisão de maneira digital. A Figura 2.2 ilustra esse cenário.

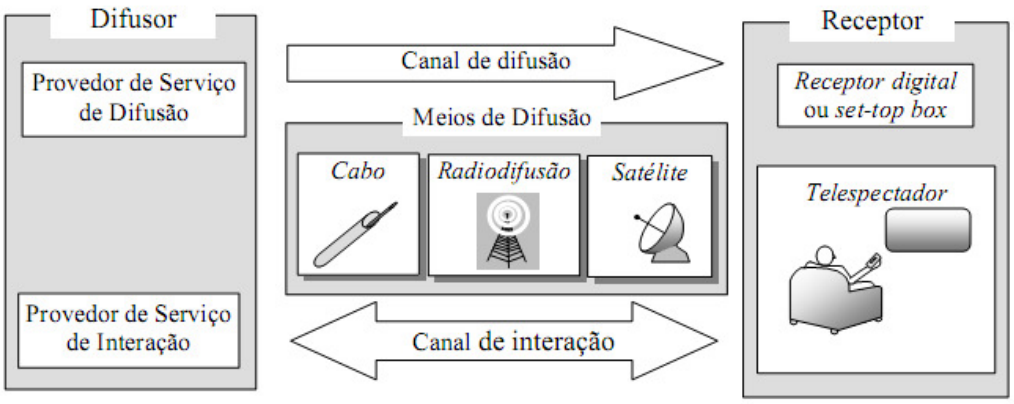


Figura 2.2: Modelo de um sistema de televisão digital.
Fonte: (Becker e Montez, 2005).

O ISDTV-T é baseado no padrão ISDB, tecnologia que reúne os recursos de mobilidade, portabilidade, alta definição, transmissão de dados e segmentação que é a subdivisão do canal permitindo a transmissão simultânea de programas no mesmo canal. É formado por um conjunto de componentes relacionados à aplicação, codificação do sinal, multiplexação e transmissão, comum ao padrão ISDB.

Em (PINTO et. al., 2007) são descritas as camadas de aplicação, *middleware*, compressão, transporte e modulação. A camada de aplicação é onde acontecem as entradas de informações de vídeo, áudio e novos serviços interativos. Na camada de codificação do sinal, os sinais de áudio e vídeo são compactados por meio dos sistemas MPEG-4 ACC e o MPEG-4 H.264, de acordo com o que expressa a Figura 2.3.

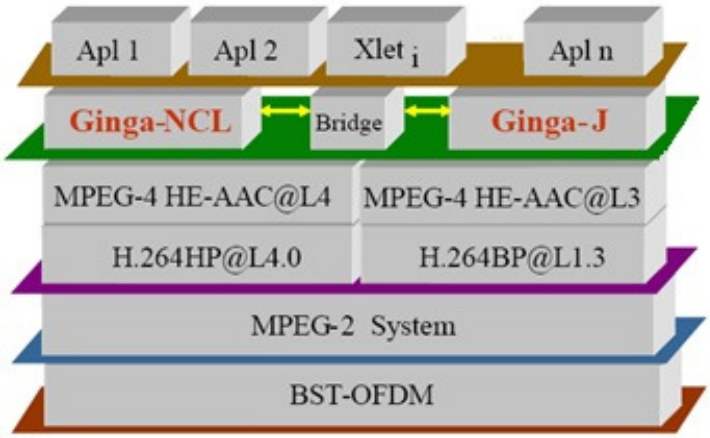


Figura 2.3: Camadas de Tecnologia do ISDTV-T.
Fonte: (SOARES e CASTRO, 2008).

A interface responsável pela implementação de interatividade e novos serviços interativos, chamada de *middleware,* é o GINGA, e foi constituído pelo desenvolvimento de projetos de pesquisa coordenados pelos laboratórios Telemídia da Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio e o Laboratório de Aplicações de Vídeo Digital - LAVID da Universidade Federal da Paraíba - UFPB. Na camada de multiplexação ocorre a união dos sinais em um feixe único, o qual será transmitido pela camada de radiodifusão utilizando a modulação BST-OFDM.

2.3.1 *Set-Top Box* - STB

A TV analógica não será instantaneamente eliminada com a introdução da TV digital no Brasil, esta transição se dará em longo prazo, devido fatores econômicos e sociais. Para amenizar esta transição, a solução encontrada foi utilizar-se de o Set-Top Box - STB que em português significa "caixa que fica sobre a TV", é um aparelho eletrônico digital é constituído por hardware e *software* dedicados com o objetivo de decodificação do sinal recebido para a exibição em televisores analógicos, com capacidade de processamento de áudio, vídeo e execução de programas.

Portanto, os STBs possuem duas funções principais, o suporte a imagens e áudios digitais e a comunicação de dados bidirecionais, ou seja, levará a interatividade entre o usuário e o aparelho de TV, por meio de um canal de retorno que possibilitara o uso de serviços interativos como *pay-per-view* (vídeos sob demanda), jogos e comércio eletrônico.

2.3.2 Interface entre *software* e *hardware: Middleware*

O *Middleware* é uma interface entre as aplicações e o sistema operacional, tem o objetivo de oferecer um padrão para as aplicações, sem se importar com as diferenças da camada do sistema operacional e do hardware que oferecem suporte aos serviços de decodificação, modulação e transporte (SILVA, 2004).

A principal vantagem da utilização do *middleware* é a portabilidade que proporciona para as aplicações, permitindo que estas sejam utilizadas em qualquer receptor digital (STB) que ofereça suporte ao *middleware* adotado. Existem no mercado padrões de *middleware* para os sistemas de TV digital, oferecido pelos diversos órgãos de padronização citados em (LEMOS et. al., 2004), são eles:

a) DASE - DTV *Application Software Enviromente* é oferecido pelo padrão ATSC, permite a execução de aplicações e conteúdo interativos, suas aplicações são

executadas em modelo procedural e declarativo, para aplicações Java e aplicações em HTML- *HyperText Markup Language* e *JavaScript*;

b) MHP - *Multimedia Home Plataform* foi proposto pelo padrão DVB, é utilizado em receptores digitais, tendo a finalidade de executar aplicações em linguagem procedural usando Java ou linguagem declarativa utilizando o desenvolvimento em HTML;

c) ARIB - *Association of Radio Industries and Business* desenvolvido no padrão ISDB, é utilizada em serviços multimídia e interativo para TV digital, é formado por duas especificações, ARIB STD-B24 - *Data Coding and Transmission Specification for Digital Broadcasting* e ARIB-STD B23 - *Aplication Execution Engine Platform for Digital Broadcasting*, que utilizam respectivamente uma linguagem declarativa denominada *Broadcast Markup Language* - BML baseada em XML - *Extensible Murkup Language* e uma linguagem baseada no MHP desenvolvendo em Java;

d) GINGA é desenvolvido pelas universidades PUC-Rio e UFPB, é o *middleware* do Sistema Brasileiro de TV digital, ele pode ser dividido em duas categorias de acordo com o paradigma da programação: procedural e declarativa. Ginga-J provê a infra-estrutura necessária para a execução de aplicações procedurais na linguagem Java e o Ginga-NCL para aplicações declarativas baseadas em documentos hipermídia escritos na linguagem NCL - *Nested Context Language*.

2.3.3 GINGA: NCL ou Java

De acordo com as funcionalidades requeridas no projeto de cada aplicação, um paradigma será mais adequado que o outro. No *middleware* Ginga existe dois subsistemas chamados de Ginga-NCL, para aplicações declarativas em NCL e Ginga-J, para aplicações procedurais em Java.

A arquitetura do Ginga pode ser dividida em três módulos principais: Ginga-CC (*Ginga Common Core*) é o subsistema lógico que provê todas as funcionalidades comuns ao suporte dos ambientes declarativo Ginga-NCL e o módulo imperativo Ginga-J.; os módulos Ginga-NCL e Ginga-J que compõem a camada de serviços específicos do Ginga. Na Figura 2.4 está representado o modelo de referência do *middleware* Ginga.

Figura 2.4: Modelo de referência do *middleware* Ginga.
Fonte: (Fórum SBTV, 2010).

Para Soares (2007), "Ginga-NCL é o subsistema 'Lógico' do *middleware* Ginga que processa documentos NCL" e de acordo com (ZACANARO et. al., 2009), o formatador NCL recebe um documento NCL e controla a sua apresentação, sincronizando os objetos de mídia, fazendo com que eles sejam apresentados.

As linguagens declarativas seguem um alto nível de abstração, normalmente ligada a um domínio ou objetivo específico. Neste modelo, o programador fornece um conjunto de tarefas que deverão ser executadas, não se preocupando com os detalhes de como o interpretador, compilador, máquina real ou virtual fará este procedimento.

Dentre as linguagens declarativas mais comuns para a TV digital, pode-se citar a NCL e o *Extensible Hypertext Markup Language* - XHTML. Em relação às linguagens procedurais, o programador tem maior poder sobre o código, à linguagem utilizada para

desenvolver sistemas de TVD nesse paradigma é a linguagem Java (BARBOSA e SOARES, 2008).

Inicialmente a plataforma Ginga-J foi projetada adotando o GEM - *Globally Executable MHP*, especificação européia para suprir todas as funcionalidades necessárias para a implementação de aplicativos para televisão digital utilizando APIs para o processamento de classes Java compiladas, sendo consideradas como componentes, cada uma definida para um tipo de serviço e representado na Figura 2.5.

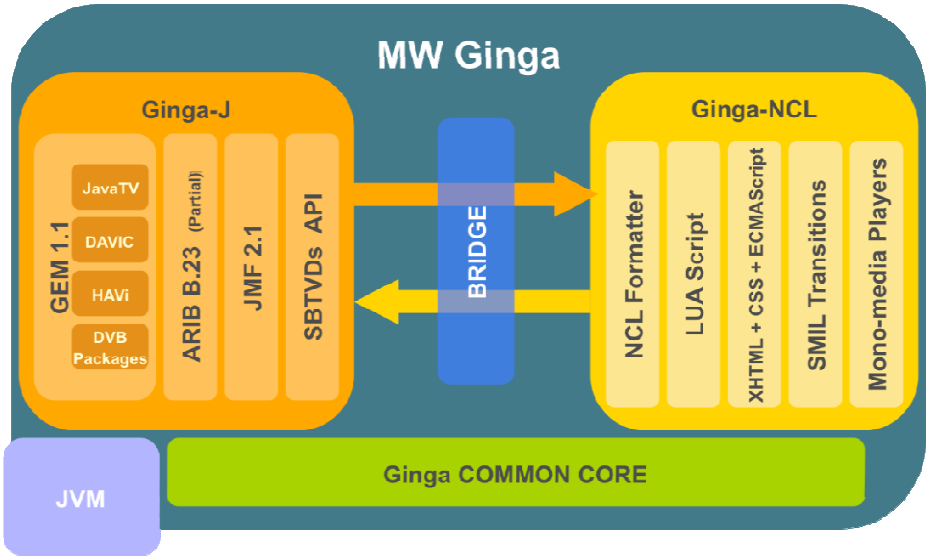


Figura 2.5: Cenário A da arquitetura do Ginga-J com GEM 1.1.
Fonte: (LAVID, 2010).

As APIs encontradas no GEM 1.1 para o processamento das classes Java são (SILVA, 2004):

a) API JavaTV definida pela Sun: Utilizada para apresentação, seleção de serviços, controle dos gráficos na tela, que será explicada no item seguinte;

b) API DAVIC - *Digital Áudio-Visual Council*: Criada pela associação DAVIC, essa API especifica formatos de conteúdo para objetos como áudio, vídeo, textos e hipertexto e ainda controla o acesso ao aplicativo e a língua adotada (áudio e legenda);

c) API HAVI - *Home Audio Video*: Criada por uma associação de companhias de produtos eletrônicos, com objetivo de atuar na apresentação e interface gráficas do usuário. É uma API que possuí um padrão para interconexão e interoperação de áudio e vídeo digital, a fim de interagirem entre si na rede. Além disso, pode gerenciar a rede, a interface do usuário e a comunicação dos componentes. Essa API permite que usuários controlem a aplicação por meio de botões de um controle remoto.

d) API definida pelo DVB - *Digital Video Broadcasting*: É uma API, relacionada ao padrão DVB, necessária para segurança, acesso de dados e para dispositivos de I/O (entrada/saída).

Como algumas dessas APIs não são livres, posteriormente foram aprovadas mudanças no Ginga-J, onde antes era utilizado o GEM, agora foi adotado o JavaDTV, sem cobrança de *royalties.* A Figura 2.6 mostra a arquitetura do Ginga-J com JavaDTV.

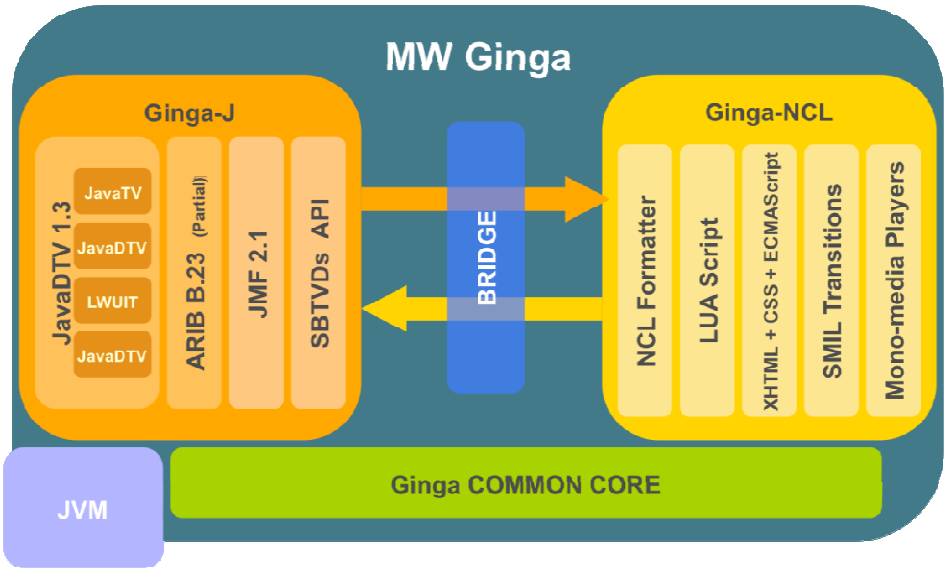


Figura 2.6: Arquitetura do Ginga-J com JavaDTV 1.3.
Fonte: (Fórum SBTV, 2010).

O JavaDTV propõe novas APIs como API gráfica LWUIT- *LightWeight User Interface Toolkit* (FÓRUM SBTV, 2010). A especificação é composta pelas API JavaDTV e

JavaTV adicionadas à base dos componentes do ambiente de execução Java, incluindo ainda CDC- Connected *Device Configuration*, FP- *Foundation Profile* e PBP - *Personal Basis Profile*.

JavaDTV é uma especificação que oferece funcionalidades de um receptor de televisão digital para o desenvolvimento de aplicações baseadas em Java conforme mostra a Figura 2.7 (ORACLE, 2010).

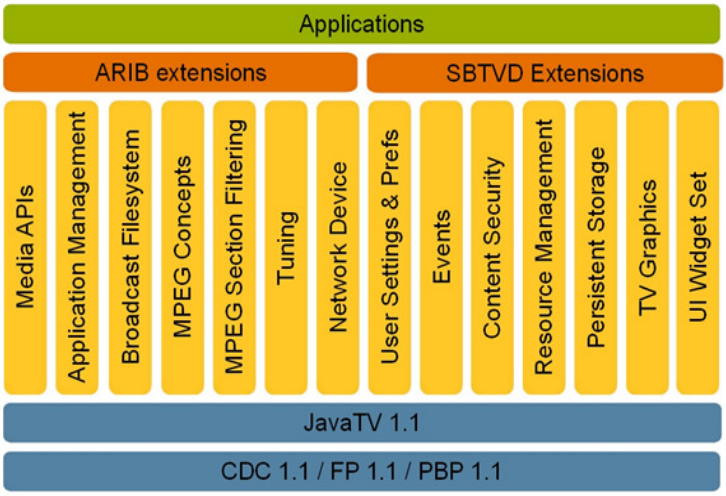


Figura 2.7: Especificação JavaDTV 1.3.
Fonte: (SUN, 2010).

A API LWUIT define os componentes gráficos de alto nível incorporados no JavaDTV . Os pacotes que definem estes componentes gráficos são (ABNT, 2010):

a) com.sun.dtv.ui – define os componentes gráficos especificamente ligados a televisão;

b) com.sun.dtv.lwuit – contém os componentes gráficos que dão suporte a criação de interfaces gráficas de usuário;

c) com.sun.dtv.lwuit.animations – habilita tanto componentes gráficos como transições animadas entre contêineres;

d) com.sun.dtv.lwuit.geom – define os elementos geométricos básicos para desenho;

e) com.sun.dtv.lwuit.layouts – define tipos úteis de layoults gráficos;

f) com.sun.dtv.lwuit.list – define estruturas de lista customizáveis, utilizadas em componentes de outros pacotes, como o com.sun.dtv.lwuit;

g) com.sun.dtv.lwuit.painter – permite desenhar arbitrariamente elementos gráficos a partir de imagens planas, escaladas e/ou *tiled*;

h) com.sun.dtv.lwuit.plaf – permite customizar a aparência dos componentes gráficos;

i) com.sun.dtv.lwuit.util – pacote de utilidades.

O Ginga-NCL e o Ginga-J são inovações mundiais, e criar um comparativo entre as linguagens é algo polêmico, principalmente quando não há comprovação bibliográfica, isso acontece porque as duas possuem características e paradigmas de programação diferentes. Porém, comparando as duas linguagens de acordo com as norma ABNT NBR - 15604, é possível destacar alguns parâmetros relevantes para cada linguagem (ZACANARO et. al.,2009) (BARBOSA, 2009) (OLIVEIRA et. al, 2009):

a) A comunidade de desenvolvedores Java, que é a linguagem do Ginga-J é maior do que NCL;

b) A API Java suporta todos os recursos necessários para o desenvolvimento de aplicações, sem a necessidade de integração com outra linguagem;

c) No que se refere ao suporte de desenvolvimento de aplicações, Java traz várias APIs padrão que permitem a criação de programas em vários domínios de aplicação, como por exemplo, som, vídeo, segurança, banco de dados, etc.;

d) A Linguagem NCL necessita de integração com outras linguagens para o desenvolvimento de aplicações com diversos domínios;

e) NCL é simples de se compreender e também de ser utilizada na criação de documentos hipermídia.

2.3.4 API JAVATV

Sun Microsystems criou a API JavaTV e desenvolvida no ambiente J2ME – *Plataforma Java 2 Micro Edition*, sendo uma extensão da plataforma Java, é uma API utilizada no desenvolvimento de conteúdo para Televisão Digital Interativa, e provê as funcionalidades necessárias para um receptor de TVD ou STB (LEMOS et. al., 2004).

Esta API apresenta um bom nível de abstração, isso é uma característica que facilita no desenvolvimento, pois o desenvolvedor não se preocupa com as camadas mais baixas, que se refere aos protocolos de serviços, transmissão e rede. Funciona como um *middleware*, pois se situa entre o sistema operacional e as aplicações.

A API JavaTV é composta pela JVM - *Java Virtual Machine* e várias bibliotecas destinadas à TVDI, permitindo ao desenvolvedor escrever o código apenas uma única vez, pois a JVM torna compatível com os diversos receptores, sem se preocupar em saber qual *hardware* será usado.

JavaTV oferece serviços e informações de serviços - *SI*, onde serviço pode ser considerado um programa de televisão, ou seja, um conjunto de conteúdo (vídeo, áudio e dados) para apresentação no STB. E SI é uma coleção de informações que especificam o conteúdo dos serviços, que são armazenadas em uma base de dados denominada SI database.

Além disso, conforme Loureiro (2004), a linguagem Java especifica pacotes que são utilizados para o desenvolvimento de interfaces, navegação, serviços e transportes, alguns deles são: Javax.tv.carousel : fornece acesso a arquivos de radio difusão e diretório de dados; Javax.tv.graphics: permite que *xlets* possam obter seu repositório principal; Javax.tv.locator: oferece maneiras para referenciar dados ou aplicativos acessíveis pela API JavaTV; Javax.tv.*xlet*: provê interfaces para o desenvolvimento e comunicação entre aplicações, oferecendo um gerenciamento.

2.3.5 *Xlets* – Linguagem em GINGA-J

As aplicações JavaTV, que são desenvolvidas na linguagem Java são denominadas *xlets*. Um *xlet* é similar a um *applet* (web) ou *midlet* (celular), e possuí um ciclo de vida que é composto por quatro estados, conforme (LEMOS et. al., 2004).

a) Carregado (*loaded*): instanciado ou criado pelo gerente de aplicação;

b) Pausado (*paused*): não utiliza os recursos compartilhados, precisando liberar recursos;

c) Ativo (*active*): ativa as funcionalidades e executar normalmente;

d) Destruído (*destroyed*): liberação de recursos e finaliza a execução.

Para o gerenciamento do ciclo de vida de um *xlet*, JavaTV utiliza um gerente de aplicação (*application manager),* que pode modificar o seu estado, assim como também o próprio *xlet*, que notifica o gerente de aplicação quando muda de estado.

O ciclo de vida de um *xlet,* se inicia quando instanciado pelo gerente de aplicação usando o método *intXlet( )*; Nesse estado *Loaded*, o gerente passa ao *xlet* o objeto *XletContext* (canal de comunicação entre o *xlet* e o gerenciador de aplicação) que define o contexto de execução, podendo enviá-lo novamente para o gerenciador notificando sobre transição do seu estado, conforme mostra a Figura 2.8.

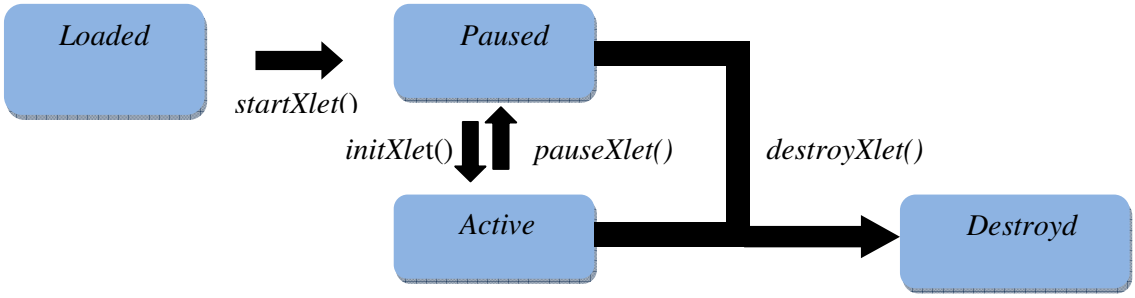


Figura 2.8: Ciclo de ação do *xlet*.

Após a inicialização, a aplicação passa para o estado *paused*, que pode ser modificado para o estado ativo, através do método *startXlet( )*, que por sua vez pode voltar ao estado

paused utilizando o método *pauseXlet( )* e por fim vai para o estado *destroyd*, usando o método *destroyXlet( ).*

2.3.6 Classificação das aplicações para TV digital

Segundo (BECKER e MORAES, 2009) a TVDI abrange uma série de aplicações, serviços e tecnologias, muitas ainda nem inventadas. Estas podem ser classificadas em sete grupos, são eles:

a) *Enhanced TV*: grupo que abrange o conteúdo televisivo de vídeo e elementos gráficos, como fotos e animações;

b) *Internet on TV*: refere-se ao acesso à internet usando um televisor, ou seja, todas as funções da internet estão disponíveis. Ex: *T- commerce, T-gov, T-bank*;

c) *Individualized TV*: permite a adaptação total da TV ao telespectador de maneira individual, possibilitando a personalização da TV, isso inclui escolha dos ângulos de câmera em transmissões esportivas ou espetáculos teatrais;

d) *Video-on-demand*: A grade de programação fica disponível pela emissora, com exceção dos programas ao vivo, para serem assistidos em qualquer horário;

e) *Personal Video Recorder - PVR:* permite a gravação digital de programas apenas especificando alguns dados do filme, como o título, o horário ou o assunto, por exemplo. Neste grupo pode-se pausar a transmissão, mesmo que ela esteja acontecendo ao vivo, e retomar a reprodução do ponto em que foi parado, pulando os comerciais;

f) *Walled Garden:* São revistas televisivas que esclarecem ao usuário o que é possível fazer, o que está disponível, e quais as aplicações residentes. Ex. EPG - Guia de Programação Eletrônica;

g) *Game Console*: permite o uso da TV para jogos, seja contra a própria TV, computador, ou em rede, contra outros jogadores. Ex: Jogo *T-lerning*.

2.4 Trabalhos Relacionados

Com a implantação da TV digital no Brasil cresce a demanda de aplicações para TVD, sendo necessária a criação de ferramentas que auxiliem os desenvolvedores na implementação de *softwares* para esta nova tecnologia de maneira simples e otimizada. Foram realizadas pesquisas bibliográficas, na qual localizou alguns trabalhos sobre *Framework* para plataforma PC, celular e TV digital, relacionados com o tema dessa pesquisa comentados a seguir.

Em (SILVA e OLIVEIRA, 2006) é apresentado um projeto de *Framework* de desenvolvimento de interfaces gráficas para TV digital baseado na arquitetura HAVI, que especifica técnicas de projetos de *Framework* e define os padrões de projeto (*design patterns*). O estudo mostra as etapas para criação de um *Framework* e os principais problemas relativos à programação de interfaces gráficas em plataformas com recursos limitados.

Este trabalho contribuiu com as suas definições sobre o padrão de projeto para o desenvolvimento do *Framework* GameTVD, pois também utilizará, para o desenvolvimento das interfaces dinâmicas para TV um arquivo XML. Este arquivo XML define os componentes presentes em cada tela, além de definir outros atributos como título, cor, tamanho e os eventos que serão lançados pelo acionamento de um determinado componente.

A diferença entre este trabalho e a pesquisa em questão, é que a pesquisa envolve tratamentos de eventos mais complexos como, por exemplo, movimentação na tela e aceleração de objetos para criação de jogos e não apenas interfaces estáticas. Contudo, foi utilizada a biblioteca HAVI no GameTVD para a criação dos componentes gráficos, assim como o trabalho de Silva e Oliveira (2006).

Outro *Framework* para o desenvolvimento de aplicações para TVDI é o AppTV (HATTORI et. al., 2005), sua principal função é abstrair e encapsular parte dos conceitos da TV digital a fim de padronizar a criação de modelos de aplicativos para o funcionamento em diversos *middlewares*. Por ser um *Framework* possuí três características fundamentais: é

flexível, possibilita adaptações e facilita sua utilização nos diversos *middlewares* que suportem a plataforma JavaTV; é extensível, admite que se possam criar classes para aumentar as suas funcionalidades; é baseado em componentes, ou seja, apresenta componentes, oferece suporte ao acoplamento de novos componentes e fornece a base para criação de novos componentes.

Apesar da ferramenta proposta não está validada, o interessante do trabalho descrito em (HATTORI et. al., 2005) é a criação das interfaces gráficas utilizando a tecnologia XML, que permite o armazenamento de dados relacionados à edição de propriedades visuais da aplicação. Essa característica do trabalho foi utilizada na criação dessa pesquisa.

O trabalho descrito em (FERREIRA e SOUZA, 2008) é proposto um *middleware* para suporte ao desenvolvimento de jogos em TVDI, disponibilizando uma estrutura para execução de jogos e uma API de desenvolvimento de aplicações para esse ambiente. O *middleware* TuGA fornece suporte aos sistemas gráfico, sonoro e de entrada. Sendo o sistema gráfico responsável pela integração das imagens com o vídeo, dando um *feedback* visual ao jogador, o sistema sonoro responsável por permitir o efeito de imersão e permite o suporte à ambientação sonora durante a atividade interativa e o sistema de entrada responsável por permitir a interação do jogador com o dispositivo.

Essa arquitetura inclui um *Framework* denominado GBF.TV (*Gamework's Brazilian Framework*) que fornece os recursos de gerenciador de *sprites*, fontes, personagens, sons, interface gráfica e tempo. Como estudo de caso foi implementado o jogo *Xlet Pong* utilizando o GBF.TV, como pode ser visto a interface na Figura 2.9.

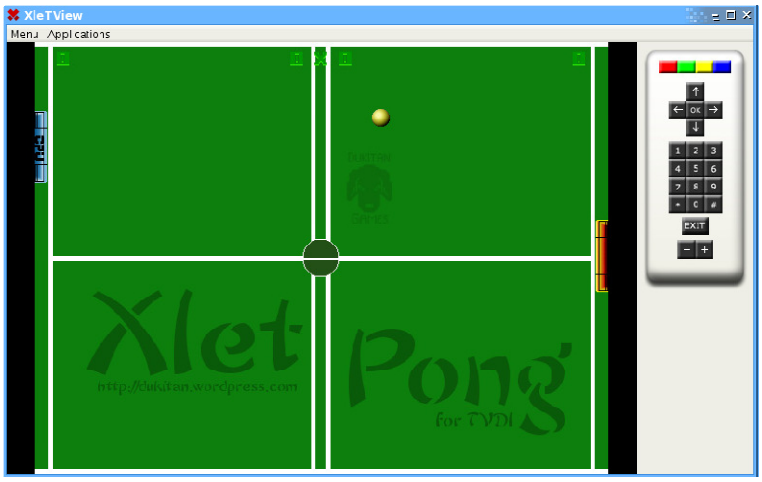


Figura 2.9: *Xlet Pong* executado no *XletView*.
Fonte: (FERREIRA e SOUZA, 2008)

O interessante da proposta é que a criação do jogo e do *middleware* TuGA aconteceu paralelamente, sendo que, quando era detectada uma característica particular do funcionamento do jogo no ambiente, essa era analisada para saber se deveria fazer parte do escopo do *Framework* ou do *middleware*.

O trabalho do (FERREIRA e SOUZA, 2008) é inicial e exibe apenas uma transformação do modelo de jogos de PC para TV, trata-se de um *Framework* que mantém as características e as familiaridades encontradas no desenvolvimento de jogos para PC. Diferente da proposta desta dissertação que propõe um *Framework* especificamente para criação de jogos para TVD.

Outro trabalho similar ao que foi apresentado é o Barboza e Clua (2009) que utiliza a idéia do uso do *Framework* para o desenvolvimento de games para TV digital usando o Ginga-J, o *Framework* Ginga Game é um conjunto de classes que simplifica o desenvolvimento de jogos para a TV digital, bem como abstrai o processo de desenvolvimento relacionado à plataforma específica, como validação do Ginga Game um simples jogo de *Quiz* foi desenvolvido.

Os autores Barboza e Clua (2009), dizem que entre as aplicações possíveis para este novo ambiente estão os jogos digitais, que a cada ano atraem um público cada vez maior em todo o mundo. No entanto, o desenvolvimento de um jogo não é uma tarefa simples, e fazê-lo em uma plataforma limitada, tais como os receptores digitais pode ser um processo complicado. Sendo assim, eles propõem um *Framework* de desenvolvimento de jogos para TV digital, sem preocupações sobre questões técnicas ou tarefas comuns relacionadas com o desenvolvimento do jogo.

O *Framework* Ginga Game é subdividido em três diferentes pacotes Java. O pacote Ginga Game fornece algumas interfaces abstratas que devem ser implementadas em um pacote de plataforma específica. Neste pacote estão definidos os conceitos básicos: como objetos e componentes do jogo, cenas e o modelo de aplicativo que gerencia esses objetos.

A classe Game.GameComponent é o pacote que possui um conjunto de componentes prontos para uso. Estes componentes devem ser adicionados a objetos em uma cena do jogo. Entre os componentes desenvolvidos estão *AnimatedSprite* (que permite o desenho de imagens animadas), *StaticSprite* (para desenhar imagens estáticas), e *BoundingBox* (para verificar colisão usando retângulos).

O pacote Ginga Game JavaTV trata da plataforma específica JavaTV. Um exemplo de plataforma de recurso específico é o gerenciador de janelas. A API JavaTV usa a classe *HScene* (do pacote HAVI) para acessar as janelas do aplicativo, essas classes são colocados para tornar fácil a mudança de plataforma de execução, se for necessário. Os módulos são representados na Figura 2.10.

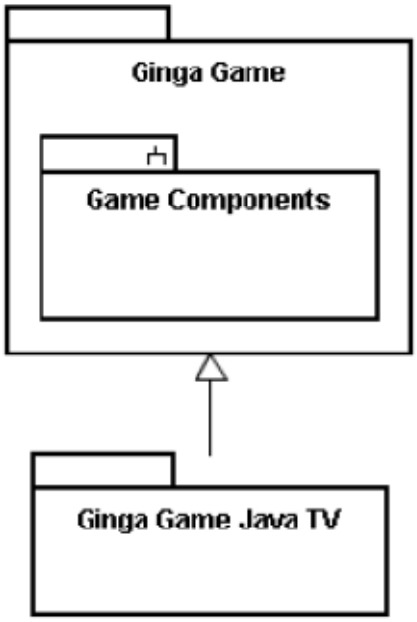


Figura 2.10: Diagrama de pacotes do Ginga Game.
Fonte: (Barboza e Clua, 2009)

Os jogos desenvolvidos com Ginga Game são criados por uma classe que estende da classe Game, é a classe principal do *Framework*. Ela é o ponto inicial para a implementação de novos jogos, sendo responsável pela execução do modelo de aplicação e pelo gerenciamento de componentes. Os jogos podem ser decomposto em unidades lógicas chamadas de cenas que são instâncias da classe *Scene*.

Cada cena é independente de outras cenas e pode ser inicializada e finalizada a qualquer momento. É possível também realizar vários componentes do jogo e objetos que são carregados e removidos do Game, ao mesmo tempo da cena. Esta divisão permite que uma organização mais simples do jogo, onde cada tela, palco ou nível pode ser descrito como uma cena.

Uma cena pode conter uma coleção de objetos do jogo, onde um objeto do jogo é uma instância da classe *GameObject.* Estas entidades interagem entre si e essas interações são o que dão vida ao jogo.

Compete ao desenvolvedor usar os componentes disponíveis do *Framework* ou ainda criar novas classes para compor o *Framework*. As classes *GameObject* e *GameComponent* podem ser estendidas para o desenvolvedor poder criar seu próprio conteúdo. Porém, para o desenvolvimento dos jogos é necessário um conhecimento sobre programação em linguagem JavaTV, pois é necessário um trabalho de implementação através de código.

A diferença principal entre esse trabalho e o *Framework* GameTVD é exatamente a interface gráfica, que propõem uma interface amigável para a criação dos jogos, não necessitando o usuário ter conhecimentos da linguagem de programação do Ginga-J. Além disso, poderá criar os eventos dos objetos utilizando apenas a interface gráfica, sem a necessidade de conhecer qualquer linguagem de programação. Para isso, o GameTVD propõem armazenar a estrutura dos jogos em arquivos XML, que poderá ser facilmente montados no ambiente de TVD pelo *Framework*.

Na Tabela 2.1 é mostrada uma comparação entre os trabalhos relacionados com o presente trabalho, levando em consideração os *Framework*s quer foram desenvolvidos para aplicações e jogos, interface gráfica, utilização de XML e tratamento de eventos.

Tabela 2.1 Comparativo entre os trabalhos analisados e o presente trabalho

	Silva e Oliveira (2006)	Hattori et. al. (2005)	Ferreira e Souza (2008)	Barboza e Clua (2009)	*Framework* GameTVD Maia (2010)
Desenvolvimento de jogos	Não	Não	Sim	Sim	Sim
Interface gráfica	Não	Não	Não	Não	Sim
Leitura de XML	Sim	Sim	Não	Não	Sim
Tratamento de eventos	Não	Não	Sim	Sim	Sim

Os trabalhos descritos em (PESSOA, 2001; ROCHA, 2003; VALENTI, 2005) não estão diretamente relacionados ao desenvolvimento de jogos para a TV digital, mas possui conteúdo interessante sobre o desenvolvimento de jogos, como por exemplo, definições de

gêneros e componente de jogos. Além disso, apresentam motores de jogos e tecnologias utilizadas em computadores para o desenvolvimento dos mesmos, arquiteturas e definições de animação úteis no entendimento dessa aplicação.

Em (PESSOA, 2001) foi definido alguns conceitos importantes para o desenvolvimento de jogos, entre estes o princípio de animação:

> "O princípio da animação, que é usado para dar a impressão de movimento, é usado continuamente enquanto o jogo é executado. O processo é simples, e consiste na rápida sucessão de imagens. Se a diferença entre imagens consecutivas for pequena suficiente, o olho humano entende essa sucessão como se fosse um movimento contínuo"

O trabalho em questão propõe uma arquitetura para criação de jogos para dispositivos móveis e é implementado um *Framework* denominado wGem, levando em consideração os recursos limitados dessa plataforma. Como estudo de caso dois jogos foram elaborados: o *BreakOut*, onde o jogador ter que destruir um conjunto de blocos e *Ship*, que é um jogo de nave com o objetivo de destruir as naves inimigas. A Figura 2.11 apresenta a arquitetura do wGem.

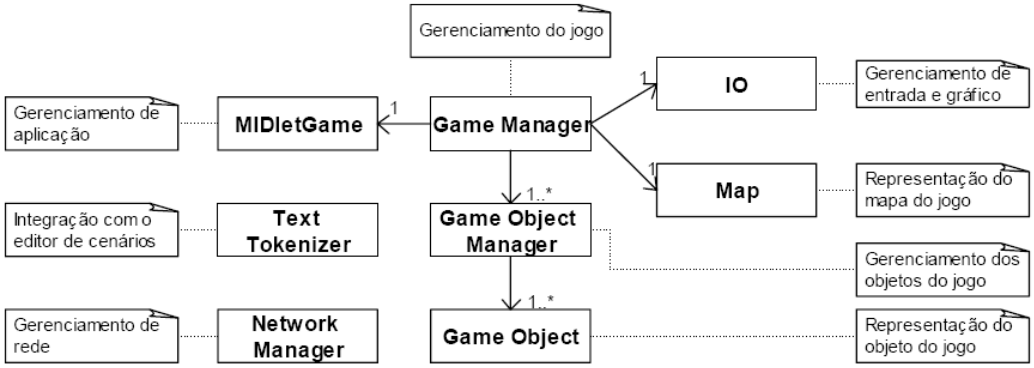


Figura 2.11: Arquitetura do *Framework* wGem.
Fonte: PESSOA, 2001

O *Framework* desenvolvido em (ROCHA, 2003) é o *Forge* 16V, destinado ao desenvolvimento de jogos isométricos, com ele foram implementados três protótipos,

apresentando desempenho e o reuso esperado. Além disso, o trabalho descreve a importância do uso de motores no desenvolvimento de jogos:

> "Com o uso dos motores para jogos, ficou mais fácil para uma empresa desenvolver mais de um jogo em paralelo e fazer com que a equipe focasse mais na lógica do jogo, devido ao maior nível de abstração e um maior grau de reuso, deixando de lado a maioria dos detalhes técnicos. Outro ponto positivo é que os motores diminuem o risco do projeto dar errado. Isto ocorre porque quem desenvolve os motores são especialistas em jogos, e o código e arquitetura resultante é bastante testado antes de entrar em produção, levando a um código mais robusto e otimizado. Além disso, o programador do jogo precisa escrever bem menos código, pois a estruturação do jogo é ditada, na maioria das vezes pela arquitetura estabelecida no motor" (ROCHA, 2003).

Em (VALENTE, 2005), descreve que a diferença entre toolkits e bibliotecas é definida através da forma que são projetados segundo o seu paradigma, porém ambos permitem o reuso do código. Toolkits e bibliotecas não impõem uma estrutura à aplicação, sendo apenas ferramentas auxiliares para a solução de um determinado problema.

Em relação à diferença entre os *Framework*s e os *game engines* é que o ultimo representa aplicações concretas, pois um *Framework* fornece um esqueleto do sistema ao dividir o projeto em classes abstratas, definindo suas responsabilidades e colaborações, o *game engine* implementa uma arquitetura, aceitando dados e configurações que determinam os aspectos do jogo, mostrando claramente as diferentes soluções em termos de reusabilidade para desenvolvimento de jogos, tais como: bibliotecas, toolkits, *Framework*s e *engines*.

Neste trabalho define-se que a diferença existente é o nível de especialização apresentado por elas, onde bibliotecas são coleções de rotinas projetadas para oferecer algum tipo de funcionalidade e os *engines* são altamente especializados, sendo que, um *game engine* pode ser construído a partir de um *Framework*, que por sua vez pode ser projetado através da aplicação de vários toolkits e bibliotecas. Estes conceitos estão na Figura 2.12.

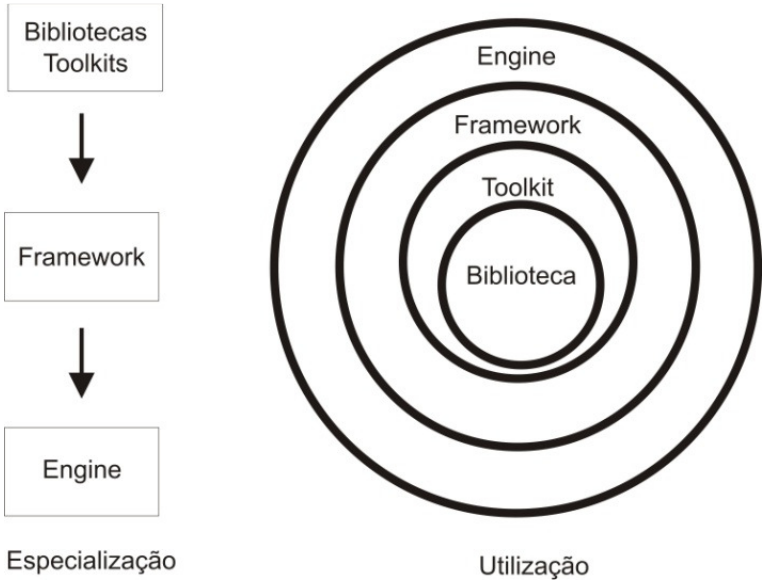


Figura 2.12: Representação em relação a especialização e utilização.
Fonte: (VALENTE, 2005).

Em relação aos jogos pra TV digital existem alguns projetos no Brasil, como o *RummiTV* (SANTOS et. al., 2008), um jogo baseado no *Rummikub*, que é um jogo matemático de propósito educativo, que visa estimular a capacidade de raciocínio de crianças e jovens. Foi desenvolvida na linguagem JavaTV, sendo compatível com os padrões mundiais, inclusive o padrão brasileiro.

O projeto encontra-se em fase final de desenvolvimento e a Figura 2.13 representa a interface do *RummiTV*.

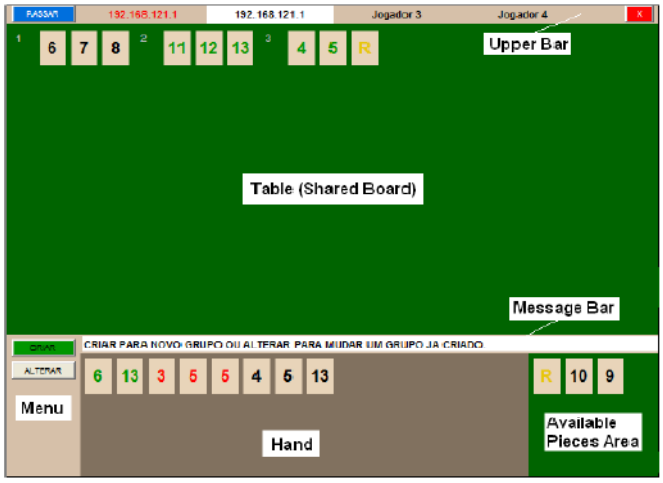


Figura 2.13: Interface RummiTV.
Fonte: (SANTOS et. al., 2008)

Outro exemplo de *T-lerning* é o jogo *Hackerteen Prototype* (FÓRUM SBTV, 2009), uma aplicação de interface amigável, conforme Figura 2.14 que utiliza o raciocínio lógico e matemático. O jogo foi desenvolvido por uma parceria entre a empresa 4Linux e o Laboratório TeleMídia da PUC – Rio, utilizando a linguagem NCL-Lua. A maneira de interação do jogo é por meio do controle remoto, além disso, possui interatividade, pois pode-se ter como adversário o próprio *software* ou outros jogadores conectados.

Figura 2.14: Interface do jogo *T-lerning Hackerteen Prototype*.
Fonte: (FÓRUM SBTV, 2009)

Os jogos apresentados são de gêneros diferentes dos jogos que deverão ser criados pelo GameTVD, pois não foi encontrado nenhum exemplo de jogos de plataforma desenvolvidos no Brasil, mas a idéia é criar jogos com interfaces amigáveis igualmente aos jogos apresentados. Sendo que, o desenvolvimento dos jogos será de maneira mais simples utilizando o GameTVD.

3 ESPECIFICAÇÃO DO DESENVOLVIMENTO DO *FRAMEWORK* GAMETVD

O estudo abordado neste capítulo relaciona-se com as etapas necessárias para a criação do *Framework* GameTVD. As etapas são o levantamento da arquitetura, análise e modelagem, implementação do *Framework* e as atividades de cada fase respectivamente. Tais etapas são fundamentais para o desenvolvimento do trabalho, pois se trata da forma conceitual da modelagem do *Framework*, seus requisitos e as ferramentas utilizadas no seu desenvolvimento, como apresentado na Figura 3.1.

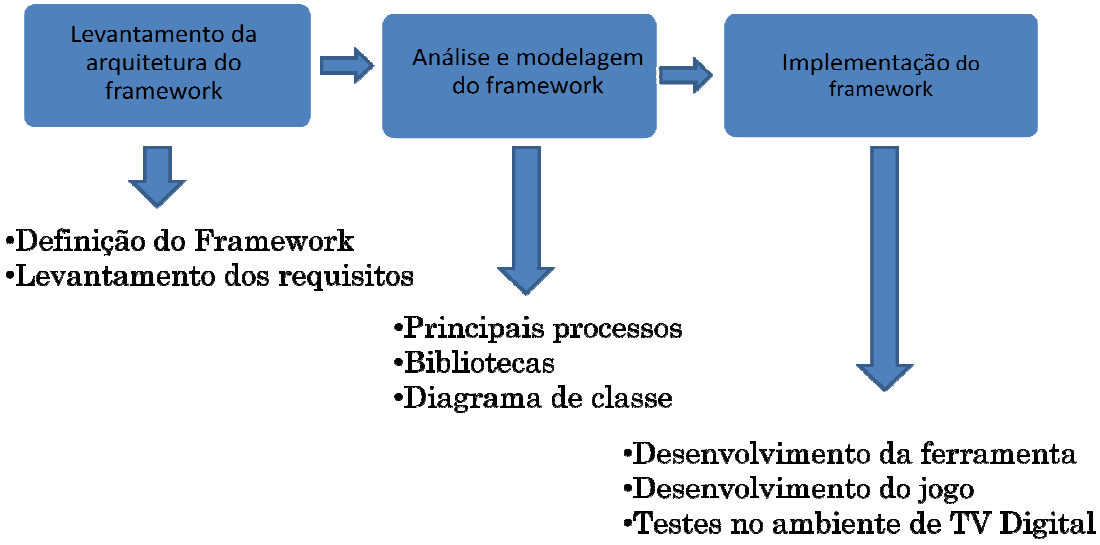


Figura 3.1: Fases do desenvolvimento do *Framework* GameTVD.

3.1 Levantamento da arquitetura

O *Framework* GameTVD é uma ambiente de trabalho para geração de jogos, consiste em um conjunto de classes básicas que permitem a criação de novos jogos, apenas sendo necessário ao usuário adicionar os objetos e cenários para a criação dos jogos, através de uma interface gráfica intuitiva.

As principais funcionalidades oferecidas pelo GameTVD são a exibição de imagens, utilizando mecanismo de visualização em duas dimensões – 2D, a reprodução de música e som, o tratamento de eventos gerados pelo controle remoto e o processamento da lógica do

jogo através de ações. As ações representam modificações sob os objetos do jogo. O foco do *Framework* é facilitar a criação de interfaces gráficas dos jogos e prover suporte para execução de jogos na TVD. Além disso, não é necessário que o usuário conheça alguma linguagem de programação, o que facilita a sua utilização.

O *Framework* apresenta três blocos responsáveis para atender o objetivo proposto: O primeiro bloco é o *Framework*, que consiste na ferramenta gráfica de criação de jogos e o *xlet* gerado (aplicativo para TVD) para a TV digital, o segundo bloco representado pelo padrão de TV digital, com as camadas de aplicações, *middleware*, sistema operacional e hardware, sendo o *xlet* gerado pelo *Framework* uma aplicação que residirá na camada de Aplicações. O terceiro bloco apresenta o ambiente emulado de TV digital representando pelo *XletView*.

Sendo assim, o usuário cria a aplicação (jogo) na ferramenta gráfica proposta pelo *Framework*, o qual gera o *xlet*. Para transmitir o código *xlet* a plataforma de TVD é utilizado o protocolo IPTV - *Internet Protocol TV*. A Figura 3.2 demonstra a caracterização do *Framework*.

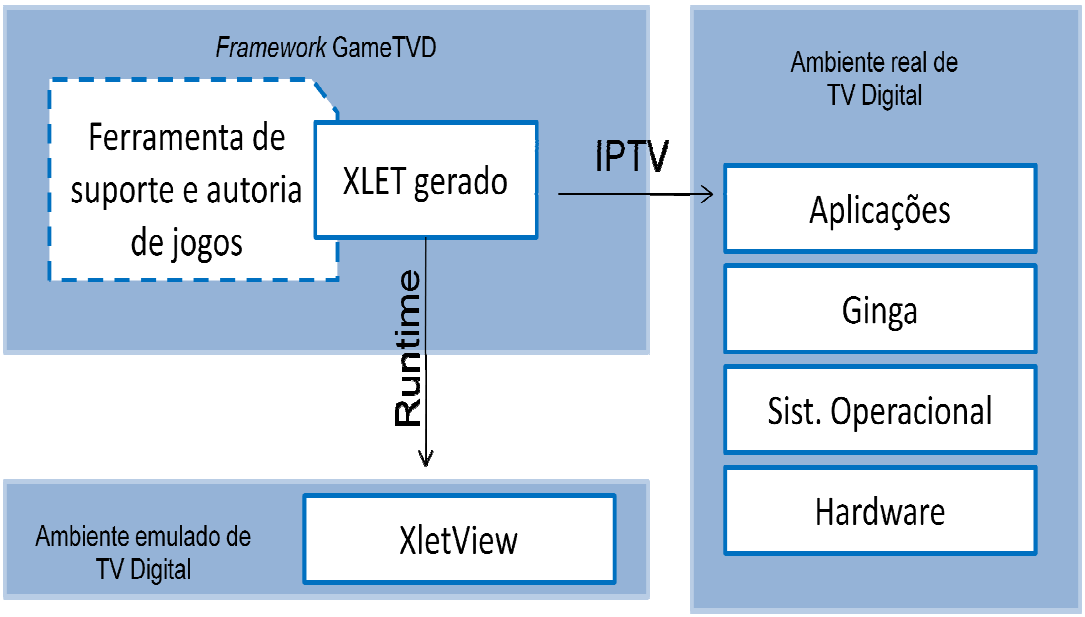


Figura 3.2: Diagrama de blocos do *Framework* GameTVD.

O serviço de transmissão IPTV, ou TV sobre IP, permite a transmissão de programas e conteúdo através de uma rede IP. De um modo geral, o usuário acessa as aplicações na TV através da internet. Nesse modelo, a oferta de conteúdo pode ser realizada por qualquer pessoa ou empresa, passando a ser um provedor de conteúdo (RODRIGUES, 2006).

A elaboração do projeto da arquitetura do *Framework* para jogos em TV digital consistiu no estudo dos requisitos necessários e as características que um *Framework* deveria oferecer, dentro das limitações impostas pelo ambiente da TV digital. Através desse levantamento foi possível identificar características e propor uma arquitetura conceitual para o GameTVD.

Os requisitos definidos para o software consistem em:

a) Prover suporte à implementação de jogos a serem executados em aparelhos de TV digital ou STBs;
b) Desenvolver o *Framework* utilizando o paradigma de orientação a objetos e técnicas de engenharia de desenvolvimento de software a fim de poder usufruir das vantagens e facilidade desse paradigma;
c) Possuir uma licença de software livre para compartilhar os conceitos e experiências adquiridas com a comunidade;
d) O *Framework* funcionará inicialmente na plataforma Windows, mas poderá ser ampliado para suporte ao sistema operacional *General Public License* - GNU/Linux;
e) A implementação do *Framework* será na Linguagem Java, devido a mesma oferecer portabilidade, alto desempenho, ser orientada objetos e dinâmica (HORSTMANN e CORNELL, 2001);

f) Pretende-se desenvolver os mais variados gêneros de jogos com a visualização em 2D, para isso optou-se por uma abordagem mais geral, sem se preocupar com os detalhes de cada gênero.

Para compreender melhor a arquitetura do *Framework* é necessário entender o ciclo de vida de um jogo em geral, segundo (ROCHA, 2003) consiste de um laço continuo que executa uma determinada lógica e apresenta o resultado ao usuário, conforme pode ser verificado na Figura 3.3.

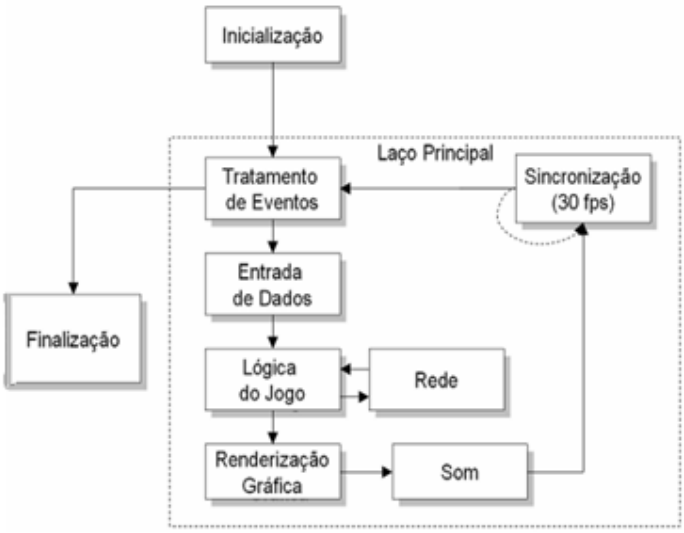


Figura 3.3: Ciclo de vida de um jogo.
Fonte: (Rocha,2003).

O ciclo de vida de um jogo possui a inicialização como responsável pela preparação para executar o jogo, o laço é responsável pelo gerenciamento e execução dos componentes da aplicação, como o tratamento de eventos, a entrada de dados, a lógica do jogo e a renderização gráfica, até a finalização do jogo, onde são liberados todos os recursos que foram empregados.

A arquitetura definida para o *Framework* GameTVD foi baseada em uma arquitetura de *Framework* de jogos para dispositivos móveis, o qual também possuí algumas limitações sobre a plataforma. Foi observada a necessidade de definição dos módulos e a interação entre

eles. Esses módulos são responsáveis por gerenciar todos os eventos que acontecem durante um jogo, o conjunto dos módulos forma a arquitetura do *Framework*.

Não existe uma regra para criação de arquiteturas de *Framework* e nem um consenso, porém é possível afirmar que é importante que a arquitetura seja clara para que se torne possível sua implementação. A arquitetura sugerida para o *Framework* de jogos GameTVD está explicita na Figura 3.4, composta pelos seus módulos e uma breve descrição da responsabilidade de cada um.

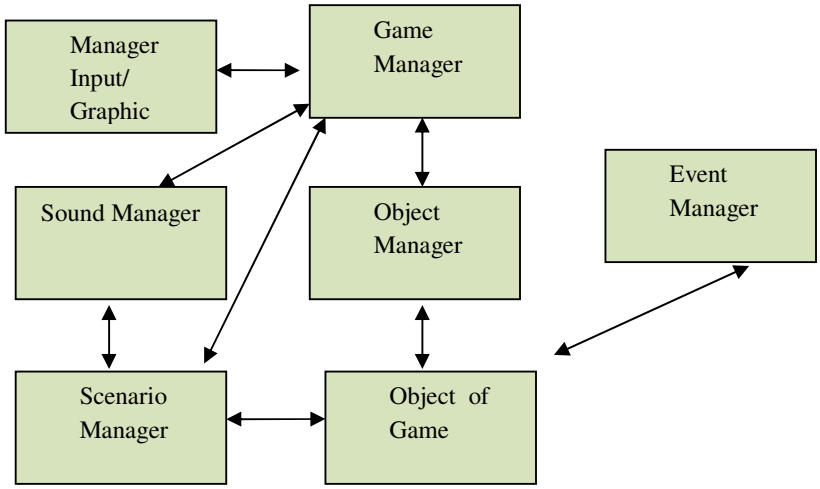


Figura 3.4: Modelo da arquitetura conceitual do *Framework*.

Os módulos propostos possuem as seguintes funcionalidades:

a) *Game Manager*: O módulo gerenciador é o modulo principal sendo responsável por criar, gerenciar, destruir os demais módulos. E ainda gerar o ciclo de vida do *xlet* para a execução no STB;

b) *Manager Input/ Graphic*: É responsável por identificar os eventos do controle remoto e ainda permitir uma representação gráfica, desenhando os elementos gráficos do jogo na tela da TV, permitindo as imagens 2D;

c) *Sound Manager*: É o módulo responsável pela a reprodução dos sons e efeitos sonoros;

d) *Event Manager*: Possui a responsabilidade de gerenciar os eventos que estão relacionados aos objetos do jogo;

e) *Object Manager*: Responsável por gerenciar os objetos do jogo e controlar o seu ciclo de vida e a detecção de colisão;

f) *Object of game*: Contém os atributos dos objetos pertinentes ao jogo, como posição, velocidade, etc;

g) *Scenarios Manager:* Responsável pelo cenário do jogo contém os personagens e os objetos de cada fase.

3.1.1 Limitações de Interfaces gráficas para TVD.

Uma das principais diferenças entre o desenvolvedor dos jogos para PC e jogos para a TV digital é a maneira de lidar com a plataforma gráfica. Desenvolvedores de ambientes de TVD devem conhecer os diferentes modelos gráficos, as questões envolvidas na configuração dos dispositivos de visualização e integração de vídeo em suas aplicações, assim como a necessidade de conceber uma interface que funcione bem dadas às limitações de uma tela de TV (MORRIS e SMITH-CHAIGNEAU, 2005).

A interface é fundamental para que um jogo possa ser bem aceito, por isso há necessidade de considerar alguns pontos relevantes para o desenvolvimento de interfaces gráficas relacionas a TV (MORRIS e SMITH-CHAIGNEAU, 2005):

a) Pontos na tela: As imagens exibidas na TV são formadas por *pixels* enquanto as geradas pela API gráfica são pontos quadrados, ocasionando uma variação de dimensão e uma mudança na resolução do receptor;

b) Variação da exibição: O formato de exibição em uma TVDI pode variar entre o padrão 4:3 e o widescreen 16:9 ou ainda 14:9, gerando alterações no tamanho dos objetos gráficos;

c) Transparência: A utilização de gráficos transparentes para ver o que está por trás;

d) Cores: O espaço de cor do sinal digital é o espaço YUV enquanto gerados pela API gráfica é o espaço RGB.

A escolha da API gráfica para a criação das interfaces foi determinada pela análise dos problemas citados acima e suas possíveis soluções utilizando o conjunto das APIs HAVI (HAVI, 2010) e *Abstract Window Toolkit* -AWT, que é responsável pela criação e gerência do uso de interface gráfica no ambiente de TV digital.

3.1.2 Arquivo gerado *xlet*.

Um *xlet* é composto por uma classe principal que implementa a interface *xlet* e possivelmente outras classes, sendo carregada por um construtor default que não contém nenhuma operação, estando sempre pronto pra ser pausado ou destruído. O arquivo *xlet* é o jogo gerado pelo *Framework* automaticamente e esse arquivo deverá conter o ciclo de vida do *xlet* e todas as rotinas inseridas no jogo. Os itens são inseridos de acordo com a inserção na ferramenta e depois codificados para linguagem JavaTV no *xlet*.

O código é gerado utilizando as classes *New* e *CreateRoom*, a classe *New* cria o esqueleto do código com todos os seus ciclos e a *CreateRoom* adiciona os demais códigos referente aos objetos adicionados ao cenário, tornando o processo mais simples e reduzindo a quantidade de código fonte que deve ser gerado. Isto é feito de maneira automática pelo *Framework* e o código gerado é compilado pela biblioteca JavaTV criando o *xlet* que será executado no compilador *XletView*, conforme a Figura 3.5:

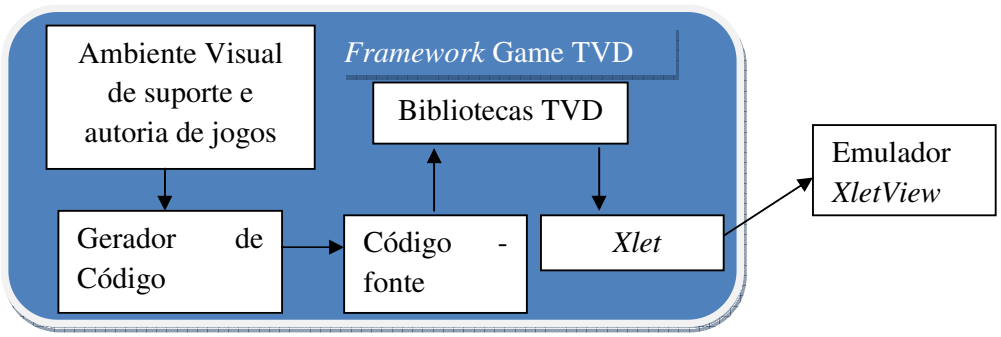


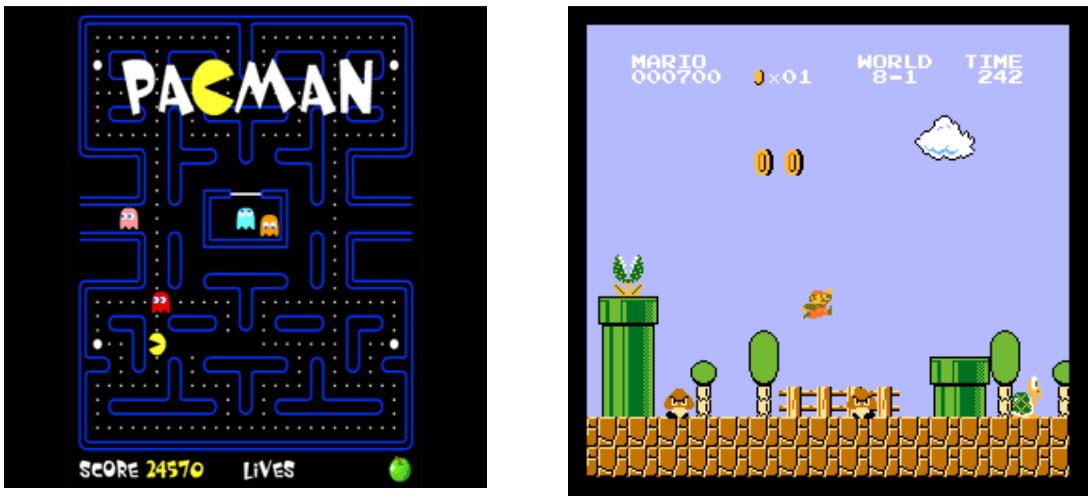
Figura 3.5: *Framework* GameTVD

3.1.3 Especificação dos jogos 2D

Em computação gráfica, 2D é chamado de objetos com duas dimensões, tendo assim largura e comprimento. Sobre a tecnologia de jogos 2D é possível especificar texturas de textos e imagens representadas em jogos através de objetos do tipo s*prite*.

Os *sprites* consistem em uma série de imagens que ao serem exibidas rapidamente em seqüência produzem efeitos de animação. Tais *sprites* podem representar os personagens do jogo que interagem uns com os outros e com o cenário, podendo inserir ações de pular, voar, nadar, lutar e atirar. Geralmente, o jogo fornece um *sprite* que representa o personagem principal controlado pelo jogador. Os outros *sprites* podem ser monstros e inimigos que perseguem o jogador, ou objetos que o jogador pode pegar, tal como moedas e armas (ALMEIDA, 2004).

Os jogos que utilizam *sprites* também são conhecidos como jogos de plataforma, é o nome dado a um gênero de jogos de videogame em que o jogador corre e pula entre plataformas e obstáculos, enfrentando inimigos e coletando objetos bônus. Entre os jogos de plataforma podemos destacar o *Pac-Man* e o Super Mario, que pode ser visualizado na Figura 3.6.



(a) (b)

Figura 3.6: Jogos de Plataforma: *PAC-MAN* (a) e Mário Brós (b).
Fonte: (BITTENCOURT e CLUA, 2004)

A idéia de *sprites* está relacionada ao princípio da animação que consiste na criação de um movimento contínuo no jogo. Esse processo consiste em uma seqüência de imagens unidas, onde o olho humano entende como um movimento contínuo (PESSOA, 2001). De acordo com Almeida (2004) um *sprite* pode executar diversos tipos de movimentos que devem ser representados visualmente para dar mais realismo ao jogo.

3.2 Análise e Modelagem

Com o intuito de auxiliar o processo de desenvolvimento da arquitetura foi realizado o levantamento dos principais processos em relação as funcionalidade do *Framework*. Tais processos apresentam uma simplificação da funcionalidade do sistema, sendo que cada processo poderá envolver uma ou mais classes de controle de acordo com análise realizada. Na fase de análise definiram-se quatro processos básicos:

- **Processo principal:** É o processo de inicialização do sistema e coordenador dos demais processos, responsável pela criação do ciclo de vida do *xlet*.
- **Processo de adicionar objetos:** é o processo responsável pela criação de objetos como texto, imagem, som. Permitindo o carregamento dos objetos e manipulação em memória.
- **Processo de tratamento de eventos:** responsável por todos os eventos de entrada e saída do sistema, pela interação do usuário com o jogo. Ele que permite colocar ações aos objetos.
- **Processo de conversão do jogo para TV digital:** é o processo fundamental, ele que permite a inserção de cada objeto e realiza o processamento deles no *xlet* para execução do jogo no STB.

3.2.1 Bibliotecas para TVDI

Para o desenvolvimento das aplicações *xlets* realizou-se um levantamento sobre as bibliotecas necessárias para o ambiente de TVDI, as quais funcionariam de acordo com as especificações do Ginga-J. Tais especificações são utilizadas para garantir a compatibilidade com o padrão GEM. Desta maneira não foram utilizadas as novas especificações entregues pela *SUN Microsystem*, visto que o trabalho já estava em fase de conclusão.

Em um primeiro momento o Ginga-J foi especificado de forma compatível com o padrão GEM, além disso, possui um conjunto de APIs específicas para o ISDTV-T. Desse modo é possível a construção de aplicações que possam ser executadas em qualquer *middleware*, desde que estas utilizem o conjunto de APIs especificadas pelo GEM. Por isso, o trabalho em questão foi baseado no GEM, pois somente no mês de agosto de 2010 foi aprovado a Norma da ABNT que especifica o Ginga-J, adotando o JavaDTV (FÓRUM SBTV, 2010). No entanto, migrar para as novas especificações não exigirá muito esforço, pois o Ginga-J continua utilizando a API JavaTV.

3.2.2 Diagrama de Classe

O paradigma orientado a objeto não é apenas um termo para denotar uma abordagem de desenvolvimento de software, mas pode ser definido como um enfoque completo da engenharia de software, o qual segue os mesmos passos de uma abordagem convencional, onde o domínio do problema é caracterizado como um conjunto de objetos que possuem comportamentos e atributos próprios, sendo que os objetos podem se comunicar entre si e serem manipulados. Esses objetos caracterizam-se como classes e subclasses (PRESSMAN, 2002).

A Figura 3.7 mostra o diagrama de classe simplificado do *Framework* proposto.

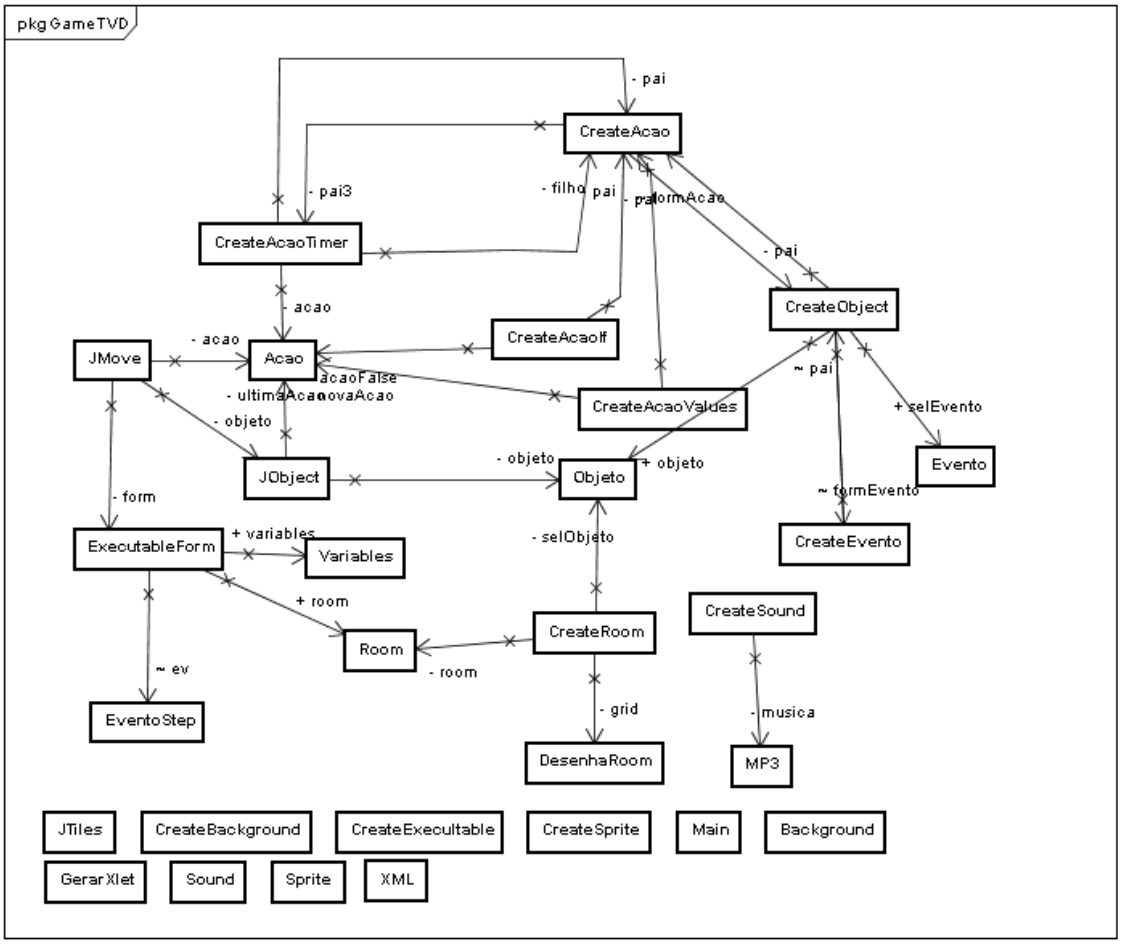


Figura 3.7: Diagrama de classes do *Framework* GameTVD

É possível fazer uma relação entre os módulos da arquitetura do *Framework* e o diagrama de classes (Figura 3.4 apresentada na página 50 e Figura 3.7 acima), onde se observa a mesma estrutura operacional, de acordo com a Tabela 3.1.

Tabela 3.1 - Relação entre os módulos da arquitetura do *Framework* e o diagrama de classe.

Game Manager	*ExecutableForm*
InputGraficManager	*DesenhaRoom*
Sound Manager	*CreateSound/MP3*
Event Manager	*Action/Event*
Object Manager	*CreateObject/JObject*
Object of game	*Object*
Scenarios Manager	*CreateRoom/Room/*

Este trabalho estabelece as seguintes funcionalidades para as classes criadas:

- Classe *ExecutableForm* será responsável pela inicialização das demais classes, e o gerenciamento do laço principal, controlando os estados do jogo.
- Classe *DesenhaRoom* será instanciada pela classe GameManager e responsável pelo gerenciamento gráfico que utilizara API HAVI e também pelo gerenciamento dos dispositivos de entrada, detectando os dispositivos que existem.
- Classe *CreateSound/MP3* permitirá a reprodução de sons e músicas no jogo no formato WAV e Midi.
- Classe *Action* e *Classe Event* são responsáveis pela animação na tela.
- Classe *CreateObject/Object* gerenciará os *Sprites* (objeto gráfico) do jogo, controlando o ciclo de vida dos mesmo.
- Classe *Object* é o objeto do jogo mais básico, contém vários atributos que permite ser gerenciado.
- Classe *CreateRoom/Room* é a classe responsável pelo nível do jogo e o que conterá na cena.

3.3 Ferramentas de Desenvolvimento

3.3.1 Plataforma *Eclipse* e *Netbeans*

Para o desenvolvimento do *Framework* foram utilizadas duas plataformas de desenvolvimento. O *Eclipse*, que é uma plataforma para integração de ferramentas para o desenvolvimento, tal plataforma apresenta as seguintes vantagens: desenvolver e compilar aplicações Java; depuração integrada em tempo real, ajudando a encontrar os erros de compilação; facilidade na reunião de bibliotecas para a compilação; ambiente amigável; plataforma portável, aberto e extensível, podendo adicionar outras ferramentas.

A outra plataforma utilizada foi o *Netbeans*, um ambiente integrado de desenvolvimento - IDE que permite ao programador criar programas utilizando recursos gráficos. Esta Ferramenta auxiliou na criação das interfaces gráficas. Sendo as duas ferramentas livres e necessitam para a compilação ter uma versão do JDK (Java) instalado no computador.

3.3.2 Emulador *XletView*

Para execução do jogo e validação de testes é necessário que um emulador esteja instalado no computador, é um software para testar aplicações televisivas interativas no PC, fornecendo uma simulação de um ambiente de TV digital. Após realização de um pesquisa sobre os emuladores existentes, optou-se pelo emulador *XletView*, que possui *Open Source* (código aberto) licenciado pela GNU *Public Licence*, é compatível com o padrão MHP, pois visa impulsionar o desenvolvimento de aplicações baseada em especificações GEM.

Para utilizar o *XletView* é necessário apenas ter familiaridade com a linguagem Java. E pode ser utilizado no Windows XP, Vista, linux e Mac. Em relação ao *Framework*, a ferramenta *XletView* já está embutida podendo ser utilizada em tempo de execução do aplicativo.

4 GAMETVD: *FRAMEWORK* DE JOGOS DE PLATAFORMA PARA TV DIGITAL

Esse capítulo apresenta o desenvolvimento da ferramenta de interface gráfica GameTVD para a criação dos jogos relacionados com os principais processos funcionais correspondentes aos módulos citados na arquitetura do *Framework* proposto.

O GameDTV foi desenvolvido em uma plataforma PC 32bits, sistema operacional Windows Vista , o *hardware* utilizado foi um *Pentium Dual-core* 1.7Ghz, 3 GB de memória. A implementação foi realizada através do desenvolvimento dos seguintes módulos:

- *Game Manager*
- *Manager Graphic*
- *Sound Manager*
- *Event Manager*
- *Object Manager*
- *Object of Game*
- *Scenarios Manager*

A interface do *Framework* é intuitiva, baseando-se em um motor de jogos para computadores denominado *GameMaker* (GAMEMAKER,2009) , o qual possui uma interface amigável. Pretende-se futuramente disponibilizá-lo a usuários, permitindo que os mesmos possam criar jogos nesta plataforma, e em seguida aplicar um questionário para análise do *Framework* incluindo suas funcionalidades e interfaces.

4.1 Processo Principal

O processo principal envolve a classe Principal necessária para a inicialização do sistema. Através dela todas as demais classes são instanciadas. Nesse processo também é utilizada a classe que é responsável pela interface *New Project* e por criar logicamente e fisicamente o novo projeto no diretório escolhido conforme mostra a Figura 4.1.

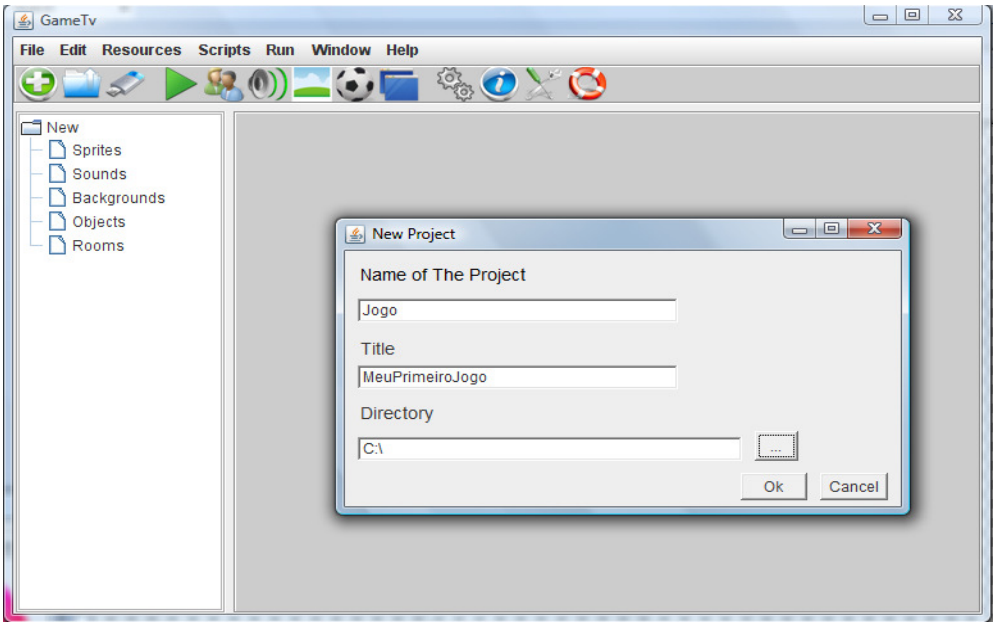


Figura 4.1: Interface Principal e a New Project.

Ao iniciar um novo projeto deverá ser inserido o nome do projeto, título e o diretório onde será armazenada a estrutura de pastas semelhante à árvore apresentada no aplicativo. O nome do projeto é o nome da pasta que o sistema cria para armazenar todos os objetos do jogo e o código gerado em *xlet*, o título é dado ao *root* da árvore do aplicativo e ao *xlet* que será construído pelo sistema. Caso o usuário não digite o nome do projeto, o *Framework* utilizará uma estrutura de pastas *default*, de acordo com a Figura 4.2. Onde se tem a pasta " temp" que armazenara a pasta *Backgrounds*, *Objects, Rooms, Sounds, Sprites e Xlets*, contendo os *backgrounds*, objetos, cenários, sons, *sprites* e o *xlet* do jogo respectivamente.

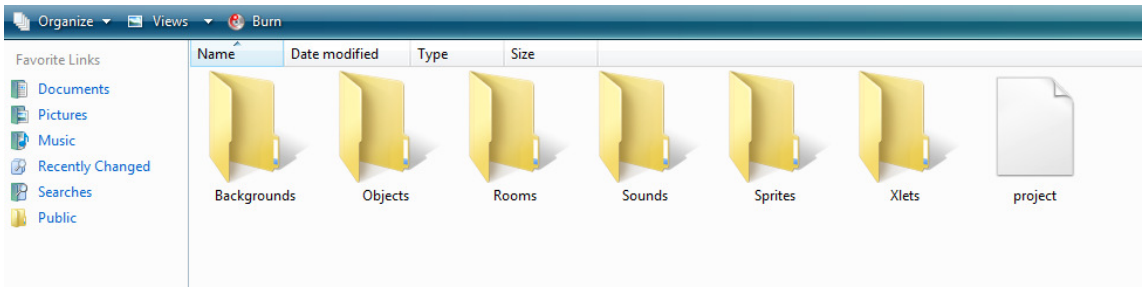


Figura 4.2: Estrutura de pastas do sistema.

Para a criação das pastas é utilizado o seguinte método apresentado no Quadro 1:

```
public static void criandoDiretorios(String texto) {
      File caminho = new File(texto);
      caminho.mkdirs();
   }
```

Quadro 1: Criando os diretórios.

Além disso, é criado um arquivo descritor que contém a estrutura do jogo, ou seja, esse descritor irá conter o nome do projeto, o título do jogo, a quantidade de *sprites*, sons, *bakgrounds*, objeto e cenário (*room*) do jogo, conforme mostra a Figura 4.3.

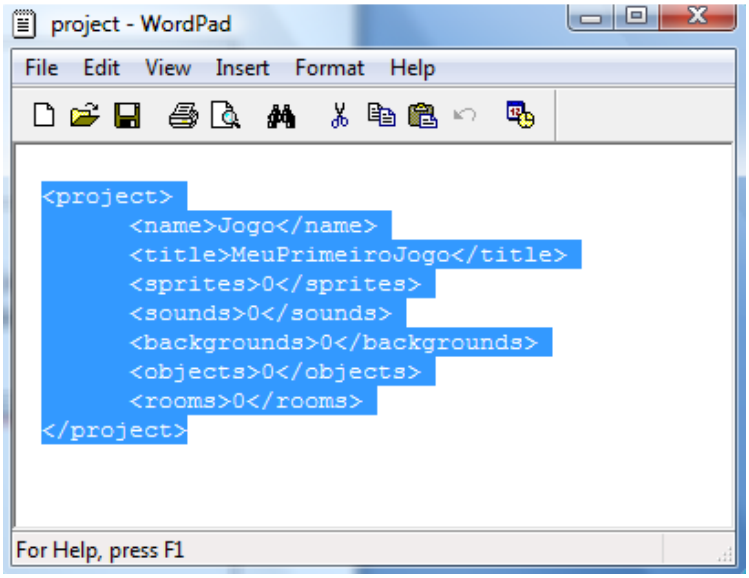


Figura 4.3: Arquivo descritor da estrutura do jogo.

Para a criação do arquivo descritor é utilizado método do Quadro 2:

```
public static void criandoDescritor(String nomeProjeto, String diretorio) throws IOException {
      File descritor = new File(diretorio, "project.bin");
      OutputStream out = new FileOutputStream(descritor);
      String texto = nomeProjeto + "\n" + Principal.name_titulo;
      out.write(texto.getBytes(), 0, texto.length());
      out.close();
      descritor.setReadOnly();
   }
```

Quadro 2: Criando descritor para carregar a arvore posteriormente.

O Arquivo descritor é um arquivo XML, é importante para identificar a quantidade de objetos e itens que o jogo possuirá, facilitando no momento da busca pelos itens a serem carregados no jogo.

O processo principal está relacionado com o módulo *GameManager* da arquitetura proposta, o qual é encarregado de lidar com a instanciação e destruição dos demais módulos.Através do *GameManager* é possível acessar todos os outros módulos do *Framework*, com isso essa é a principal interface entre o usuário e o jogo em si.

4.2 Processos de adicionar objetos

O processo de adicionar objetos para a criação do jogo é o responsável por permitir que as imagens sejam desenhadas na tela do computador, carregando as imagens e sons e manipulando em memórias. Esse processo é representado pelas classes createSprite, *createSound*, c*reateBackground*, *createObject* e *createRoom* que correspondem respectivamente a inserir *sprites*, sons, *backgrounds,* objetos e cenários.

Além disto, o processo envolve o módulo *Manager Graphic* para inserir os *sprites*, *backgrounds* no cenário do jogo, gerando uma representação gráfica, e ainda o módulo *Sound Manager* para inserir os arquivos de áudio. Além disso, no cenário poderá inserir textos e objetos criados pelos *sprites*. Para executar esse módulo, que apresenta os elementos gráficos, é necessário executar o *Game Manager*.

O módulo Game Manager gerencia os objetos, carregando-os em um vetor. Os objetos ficam armazenados em um arquivo XML, contendo as posições iniciais e os códigos dos efeitos relacionados. Essa estrutura permite que todos os objetos sejam carregados no PC e também que possam ser carregados na TV, pois irão utilizar o mesmo arquivo XML.

4.2.1 Criando *Sprites*.

A Figura 4.4 representa a interface *Create Sprites*, com ela é possível adicionar as imagens com formato “.jpg”, “.jpeg”, “.png “e “.gif” que poderão ser utilizadas no jogo. Após a inserção da imagem escolhida, a mesma aparecerá do lado esquerdo do aplicativo na estrutura de pasta correspondente ao diretório *Sprites*.

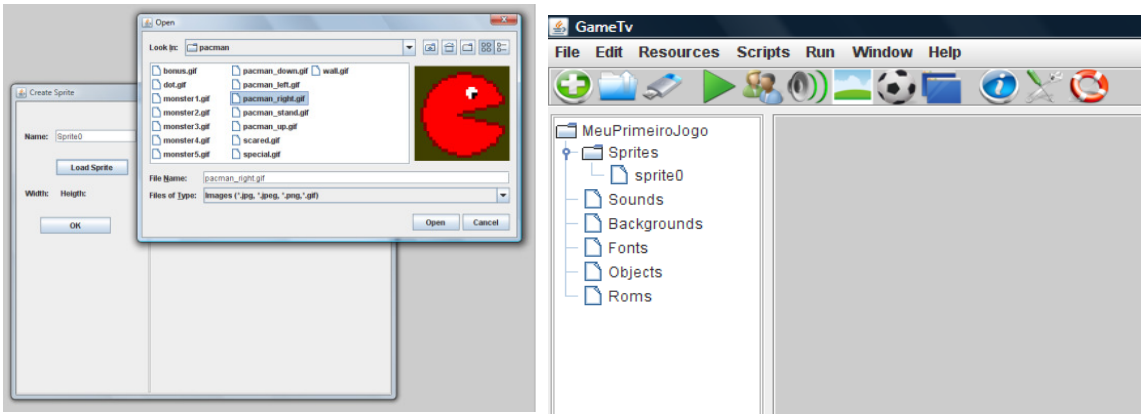


Figura 4.4: Interface para criação de sprites.

Para atualizar a imagem adicionada basta dar um duplo clique no nome dela e o formulário aparecerá novamente para mudar o *sprite*. A imagem será adicionada fisicamente no diretório *Sprite* através da utilização da classe Copia, mostrado no Quadro 3:

```
class Copia {
    public void Copia(File entrada, File saida) throws FileNotFoundException, IOException {
     FileInputStream input = new FileInputStream(entrada);
     FileOutputStream output = new FileOutputStream(saida);
    int c;
    do {
       c = input.read();
       if (c!=-1) {
          output.write(c);
       }
    } while (c != -1);
    input.close();
    output.close();
    }
}
```

Quadro 3: Classe Copia utilizada para adicionar a imagem no diretório *Sprites*.

4.2.2 Criando Sons

Com a intefarce *Create Sound* é possível inserir os arquivos sonoros, a classe *createSound* é responsável por permitir a leitura e reprodução dos sons. Além disso, controla o parâmetro volume e é representado pelo módulo *Sound Manager*. A Figura 4.5 apresenta a interface gráfica da classe *createSound*.

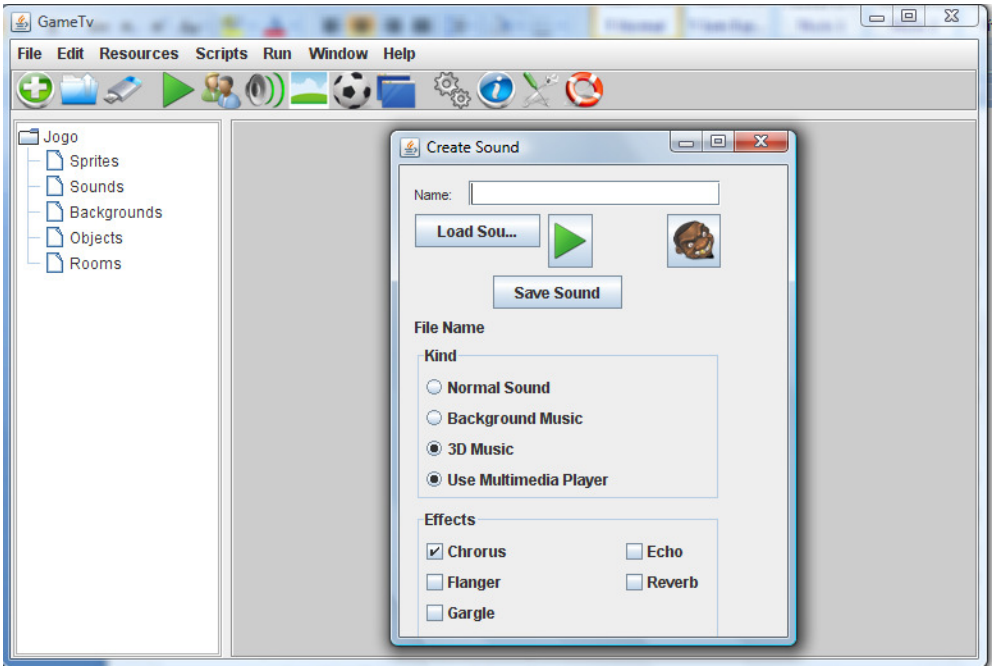


Figura 4.5: Interface gráfica da classe *createSound*.

4.2.3 Criando Cenários

A interface *Create Room* é utilizada para definir os níveis do seu jogo ou os cenários. No cenário é possível ter um fundo, conhecido como *background*, que pode ser uma cor ou uma imagem, podendo ser carregado a partir de um arquivo na interface *Create Background*. Ao criar um cenário, esta interface permite posicionar os objetos na tela, correspondentes aos objetos do jogo, assim como gerar várias instâncias de um mesmo objeto do jogo. Por exemplo, pode-se adicionar um objeto “pedra” e usá-lo em vários locais, ou ainda ter várias instâncias de um objeto “nave”, e todas elas terão o mesmo comportamento.

Os cenários são as entidades onde o jogo acontece. Todo jogo necessita de pelo menos um cenário onde são colocados os objetos iniciais de um jogo e o plano de fundo. Além disso, podem-se adicionar textos no cenário. Na janela *Create Room*, correspondente a Figura 4.6, temos as propriedades da *room*, onde são visualizadas as abas *Settings, Objects, Background* e *Titles*.

As propriedades são definidas da seguinte maneira: em *Settings* contém as configurações da janela; em *Objects* são escolhidos os objetos a serem adicionados; em *Backgrounds* é inserido um fundo ou cor de fundo; e em *Titles* é permitido inserir textos e posicioná-los.

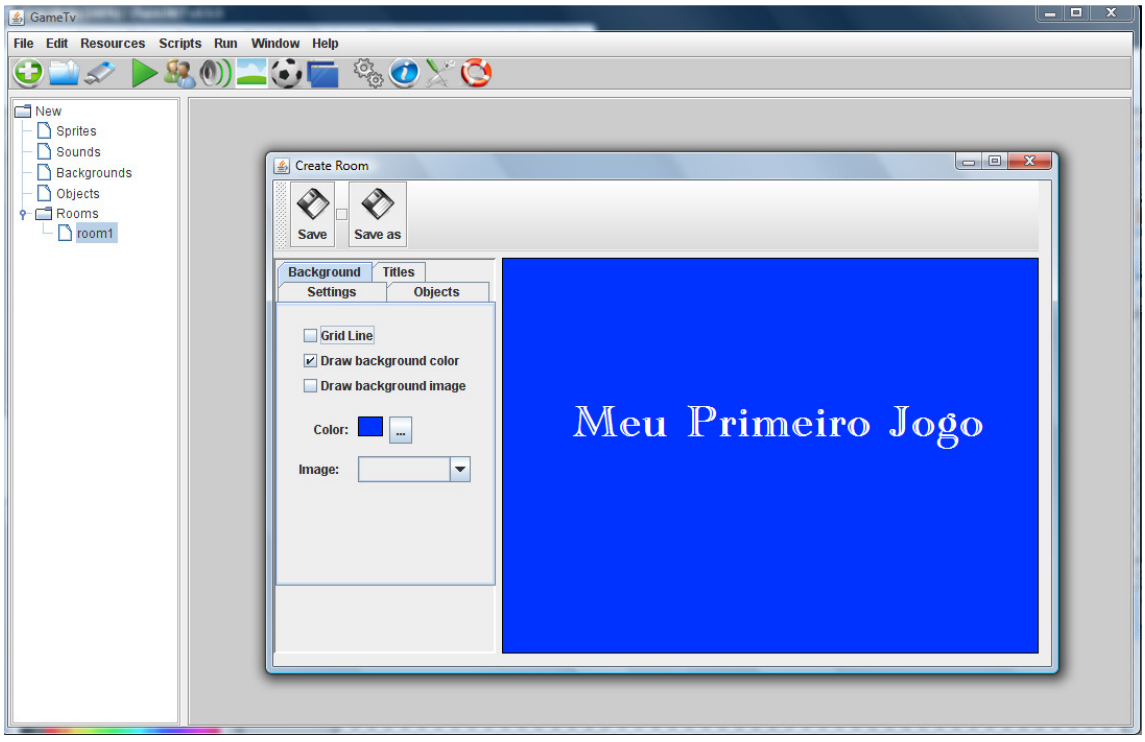


Figura 4.6: Interface gráfica *Create Room*.

Na Figura 4.6, observamos a aba *Background* com as opções de utilizar um *background* já existente ou uma cor de fundo desejada, conforme a escolha é selecionada as opções de "*Draw background color*" ou "*Draw background image*". À medida que vai modificando o cenário, por exemplo, uma configuração de tela, inserção de uma cor de

background ou a inserção de um objeto, o painel ao lado vai recebendo as modificações simulando uma tela.

4.3 Processo de tratamento de eventos

4.3.1 Criando objetos

Os objetos são recursos que permitem a geração de movimentos para os *sprites.* Com a interface *Create Object* é possível escolher qual o *sprite* que se deseja adicionar, seus eventos e ações, ou seja, é possível programar o comportamento do objeto, tendo a percepção de animação e dinamismo.

A diferença entre os objetos e os *sprites*, é que os *sprites* são apenas imagens animadas ou não, enquanto os objetos possuem comportamentos e são representados pelos *sprites*. Os jogos são formados pelos objetos, sendo que um objeto descreve uma entidade, e a partir dele são criadas as instâncias que executam ações e que participam do jogo. Conforme Figura 4.7.

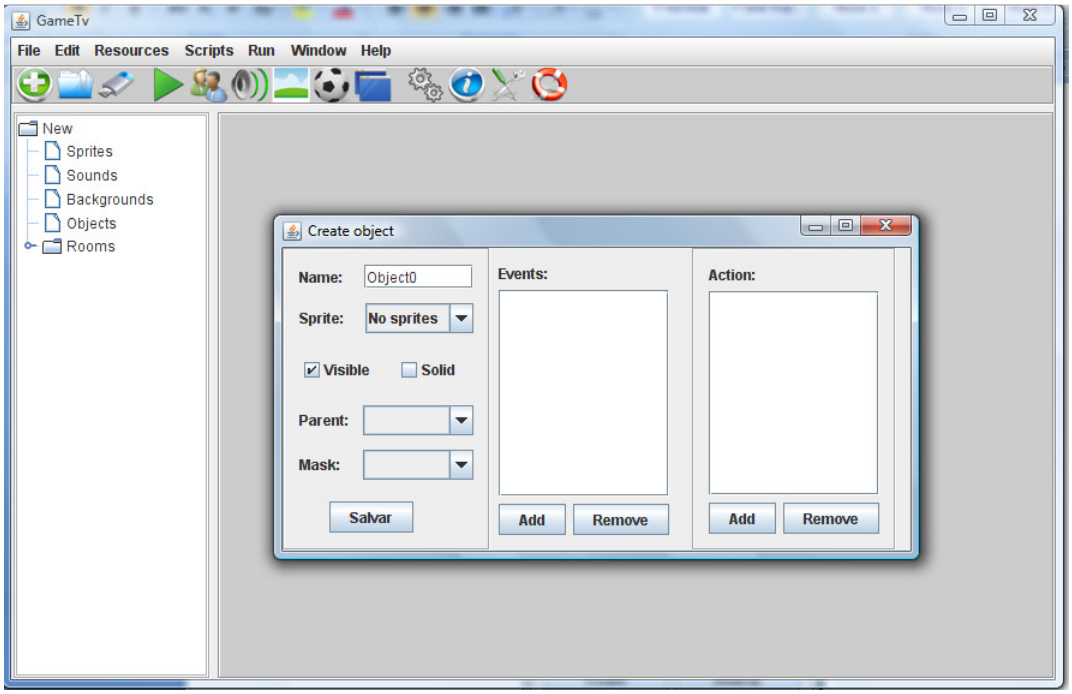


Figura 4.7: Interface *Create Object*.

A idéia de adicionar objetos no jogo é para que haja interação, para isso é necessário indicar quais eventos a serem adicionados ao objeto e que ações devem ser tomadas após este

evento. A animação do jogo é criada através de uma interface amigável, onde é possível adicionar e excluir os eventos e ações.

As opções de eventos serão apresentadas em uma nova janela, demonstrada na Figura 4.8.

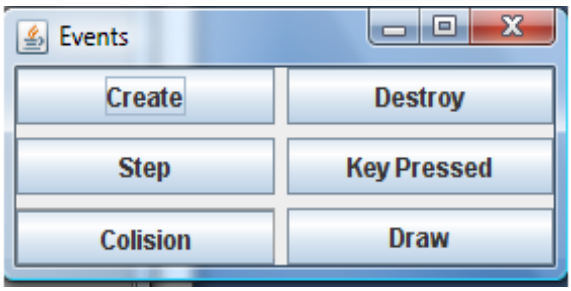


Figura 4.8: Opções de Eventos.

Foram considerados os seguintes eventos:

- *Create*: acontece quando uma instância é criada;
- *Destroy*: sucede um pouco antes de uma instância ser destruída;
- *Step*: realiza um evento a cada passo do objeto;
- *Key Press*: ao se pressionar uma tecla do controle remoto;
- *Colision*: ocorre quando uma instância colide com outra instância de objeto;
- *Draw*: ao desenhar a instancia do objeto;

As ações são executadas de acordo com os eventos, por exemplo, existe o evento *create* que é emitido quando uma instância do objeto é criada, podendo adicionar ações ao objeto criado, sendo possível atribuir a ele a ação de mover-se.

Quando dois objetos se chocam, tem-se o evento *collision*, neste caso pode-se executar a ação de parar ou mudar a direção do movimento. Além disso, é possível ainda executar a ação de "emitir um som" por ocasião do evento *collision*. Quando o jogador pressiona uma tecla o evento *Key Pressed* é emitido, sendo executada a ação do personagem em se movimentar na direção da tecla pressionada.

A idéia geral é que para cada objeto é possível indicar ações para diversos eventos e, assim, definir o comportamento do objeto. É possível inserir as seguintes ações, conforme a Figura 4.9.

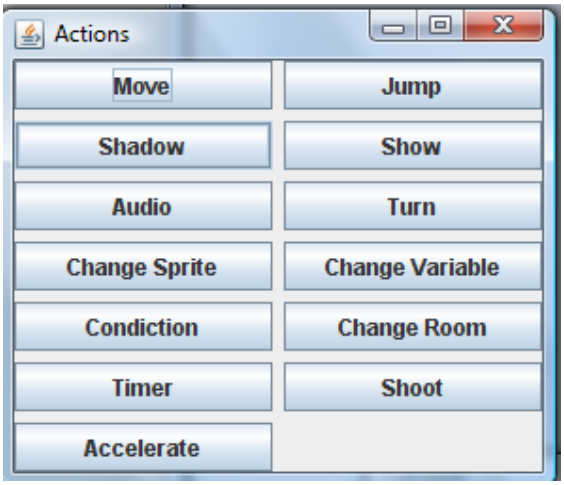


Figura 4.9: Ações de um objeto.

As funções de ações são:

- *Move*: o objeto pode mover-se para direita, esquerda, na horizontal e vertical, ir e voltar, movimentar-se randomicamente, andar em uma direção ou dar passos naquela direção e voltar para a sua posição inicial;
- *Jump:* o objeto possui a capacidade de pular*;*
- *Shadow:* o objeto fica invisível;
- *Show:* o objeto aparece;
- *Audio:* insere um áudio no objeto;
- *Turn:* direciona o objeto para direita, esquerda, na horizontal e vertical;
- *Change Sprite:* muda a imagem do objeto;
- *Change Variable:* muda o valor de uma variável adicionada;
- *Condiction:* usa-se condições para manipular ações;
- *Change Room:* troca a *room* em exibição;
- *Timer:* Ação que após um tempo estipulado acontece;

- *Shoot:* atirar um objeto;
- *Accelerate:* acelerar a movimentação de um objeto.

4.4 Processo de conversão do jogo para TV digital

O *Framework* é composto pelo processo de geração do código-fonte para o jogo em ambiente de TV digital. O código é gerado utilizando as classes e métodos do *Framework* GameTVD, o que torna o processo mais simples e reduz a quantidade de código fonte que deve ser gerado.

A criação do código é realizada de maneira automática em tempo de execução com a estrutura básica de um *xlet* contendo o seu ciclo de vida, conforme Quadro 4, e é possível simular o ambiente de TV digital com o *XletView* chamado pelo *Framework* também em tempo de execução.

```
//importar bibliotecas utilizadas

import Java.awt.Color;
import Java.awt.Container;
import Java.awt.event.KeyEvent;
import Java.awt.event.KeyListener;
import Javax.tv.xlet.Xlet;
import Javax.tv.xlet.XletContext;
import Javax.tv.xlet.XletstateChangeException;
import org.havi.ui.HContainer;
import org.havi.ui.HScene;
import org.havi.ui.HSceneFactory;
import org.havi.ui.HSceneTemplate;
import org.havi.ui.HText;
import Java.awt.Graphics;

public class New extends Container implements Xlet, KeyListener {

        private XletContext context;
        private HScene cena;
        private HContainer container;
        private HText texto;

        public New()
             {

             }

   public void destroyXlet(boolean arg0) throws XletstateChangeException {
      System.out.println("Destroi");
      HSceneFactory.getInstance().dispose(cena);
      context.notifyDestroyed();
   }

   public void initXlet(XletContext context) throws    XletstateChangeException {
         System.out.println("Iniciar");
```

```
        this.context= context;

        //cria um template pra requerer uma cena
        HSceneTemplate template = new HSceneTemplate();
        template.setPreference(HSceneTemplate.SCENE_SCREEN_LOCATION,
                new org.havi.ui.HScreenPoint(0, 0), HSceneTemplate.REQUIRED);
         cena = HSceneFactory.getInstance().getBestScene(template);
         cena.setBackgroundMode(HScene.BACKGROUND_FILL);
         this.setSize(cena.getSize());
         cena.setVisible(false);
         cena.setBounds(0,0,720,576);
         cena.setLayout(null);
         cena.setBackgroundMode(HScene.BACKGROUND_FILL);
                cena.addKeyListener((KeyListener)this);
         cena.add(this);

    }

    public void pauseXlet() {

    }

    public void startXlet() throws XletstateChangeException {
           System.out.println("Start");
       cena.setVisible(true);

    }

    public void keyTyped(KeyEvent e) {
       throw new UnsupportedOperationException("Not supported yet.");
    }

    public void keyPressed(KeyEvent e) {
          System.out.println(e.getKeyCode());
    }

    public void keyReleased(KeyEvent e) {
       throw new UnsupportedOperationException("Not supported yet.");
    }
```

Quadro 4: Código do *xlet* gerado.

Uma vez que não é necessário definir os métodos que serão invocados, a idéia do *Framework* é criar o jogo no computador através da interface gráfica, tendo uma usabilidade amigável ao usuário e o *Framework* deve se encarregar de gerar o código para TVD. Para isso, o *Framework* insere os dados em arquivos XML, onde para cada objeto e *room* (cenário) possui um arquivo XML para que o jogo possa ser consultado e montado para o ambiente de TV digital. O XML do cenário contém todos os objetos do jogo, e no XML do objeto são encontrados todos os eventos e ações relacionados àquele objeto e o *sprite* que representa aquele objeto.

A Figura 4.10 representa o processo de conversão do Framework GameTVD, e possui os módulos Criação da Interface (Sistema gera o arquivo XML) e Conversão do Jogo (Sistema ler XML e gera *xlet*), sendo o *xlet* enviado para TVD.

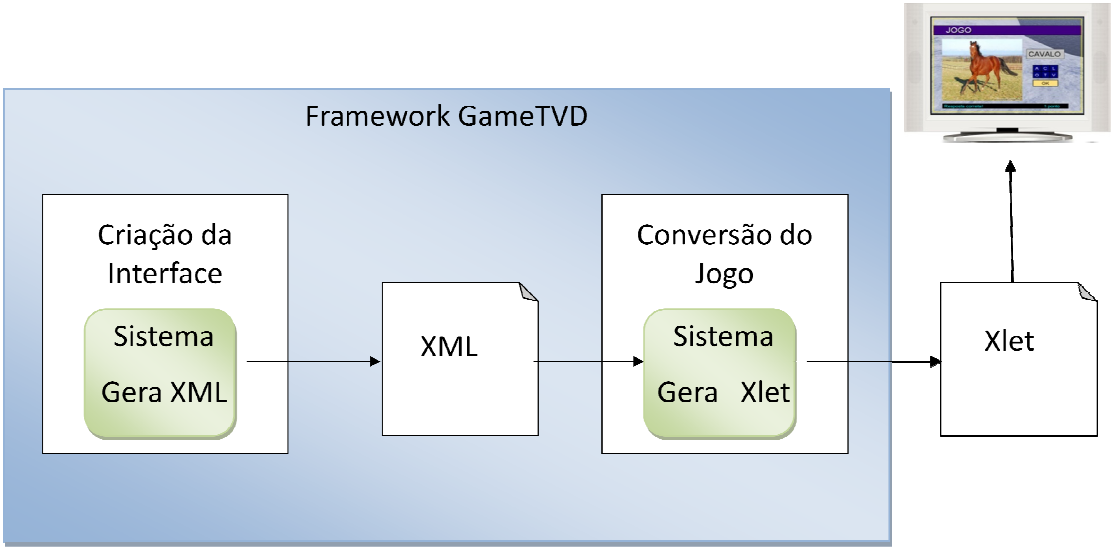


Figura 4.10: Processo de conversão do jogo para TV digital.

Após a criação do código fonte (*xlet*), ainda falta que ele seja compilado utilizando a API JavaTV e as bibliotecas comentadas anteriormente. O método Main.compile é usado através do import com.sun.tools.Javac.Main para que haja a compilação do código; Para isso, é utilizado o método, demonstrado no Quadro 5.

```
public void compilar(){

  int resultadoCompilacao = Main.compile(new String[]{Principal.path+ "\\" + Principal.name_titulo+ ".Java"});
   if (resultadoCompilacao == 0) {
      System.out.println("Compilado com sucesso: "+Principal.path+ "\\" + Principal.name_titulo+ ".Java");
   }
   else{
       System.out.println("Error.");
      System.out.println(Principal.path+"\\"+ Principal.name_titulo+ ".Java");
   }
}
```

Quadro 5: Compilar *xlet*.

O próximo passo é executar o emulador do ambiente de TV digital, para isto o *Framework* adiciona o nome do projeto e o endereço do *xlet* no arquivo XML referente à

estrutura da aplicação do *XletView,* utilizando o arquivo "*applications.xml*', através do método representado no Quadro 6.

```
public void executar(String n_tit, String n_end) throws FileNotFoundException, IOException{

    FileReader leitura = new FileReader("C:/Rebuild/xletview-0.3.6/config/applications.xml");
   // File original= new File("C:/Rebuild try/xletview-0.3.6/config/applications.xml");
    File rename=new File("C:/Rebuild/xletview-0.3.6/config/applications.xml");
    File arq_l= new File("C:/Rebuild/xletview-0.3.6/config/applications2.xml");
    BufferedReader in_l= new BufferedReader(leitura);
    String proucura="/APPLICATIONS>";
    String digita= "     <APPLICATION>\n"+
              "        <NAME>"+Principal.name_titulo+"</NAME>\n"+
              "        <PATH>"+Principal.path+"</PATH>\n"+
              "        <XLET>"+Principal.name_titulo+"</XLET>\n"+
              "    </APPLICATION>\n"+
              "</APPLICATIONS>";

    escreve(arq_l, in_l,n_end,proucura,digita,1);

    rename.delete();
    arq_l.renameTo(rename);
          try {
        Runtime.getRuntime().exec("cmd.exe /C start xletview.bat");
     } catch (IOException ex) {
        Logger.getLogger(escreverArquivo.class.getName()).log(Level.SEVERE, null, ex);
     }

}
```

Quadro 6: Executar o emulador.

O desenvolvimento dos jogos para *set-top boxes* por meio GameTVD é realizado de forma idêntica aos jogos criados para Desktop com Java SE, em nível de estruturação do código e abstração, mas em cima de uma máquina virtual reduzida, pois os set-top boxes tem menor capacidade de processamento e armazenamento.

Com a possibilidade de utilizar esta ferramenta apresentada, um desenvolvedor de jogos para TVD estará apto, com um treinamento relativamente simples e sem a necessidade de escrever um código, a utilizar o GameTVD e desenvolver jogos para TVD com algumas, ou todas as características: *backgrouns*, s*prites* ou imagens; Gráficos animados; Funções básicas de som e efeitos especiais; E jogos de plataforma.

5 DESENVOLVENDO JOGOS COM O GAMETVD: ESTUDO DE CASO

O *Framework* GameTVD não pretende ser uma ferramenta definitiva, e sim uma base sólida que possa ser melhorada e ampliada com o tempo e a interação de outros pesquisadores, e portanto este capítulo não tem a intenção formalmente validar o *Framework*, inclusive considerando que esta não é uma tarefa simples, necessitaria do desenvolvimento de alguns jogos, levando em considerações variáveis como desempenho, tempo de desenvolvimento, facilidades e limitações do *Framework*. Além disso, seria importante a criação de um jogo utilizando o *Framework* e o desenvolvimento do mesmo jogo em paralelo por um outro desenvolvedor, para que fosse possível uma comparação entre os projetos.

Portanto, a abordagem de validação levaria um tempo maior de análise e por possuir essa quantidade de variáveis, se tornaria um processo complexo e não muito confiável. A idéia do estudo de caso não é validar o *Framework* e sim criar uma prova de conceito, mostrando que o mesmo funciona a contento atendendo os objetivos propostos.

Os testes de execução do *Framework* GameDTV e a execução do jogo no emulador *XletView* foram realizados em uma plataforma PC 32bits e 64 bits, sistema operacional *Windows Vista* e *Seven*, o *hardware* utilizado foi um *Pentium Dual-core* 1.7Ghz, 3 GB de memória. Não foi possível realização de testes em ambientes reais, devido o Ginga-J ainda não estar implementado para STBs e TVs Digitais comerciais.

Para a implementação do estudo de caso, foi escolhido o desenvolvimento de dois jogos, sendo o primeiro criado nos anos 80, pela empresa Namco, denominado *Pac-Man* e o segundo *Space Invaders,* criado por *Tomohiro Nishikado* para a *Taito Corporation* no Japão e foi um dos primeiros videogames de tiro com gráfico bidimensional.

Ambos os casos, os jogos foram desenvolvidos por um aluno de graduação da área de ciência da computação, com conhecimentos em programação, mas sem experiência em jogos

e tecnologias associadas à TVDI, sob a supervisão da pesquisadora que desenvolveu o Game TVD. Na próxima seção serão mostrados os jogos desenvolvidos utilizando o *Framework* sem a utilização de linhas de código adicionais e, logo após serão apresentados os testes realizados e resultados obtidos.

5.1 Jogos Desenvolvidos

5.1.1 Pac-Man

O jogo *Pac-Man* foi um fenômeno de popularidade e entrou para o livro dos recordes como ícone de vídeo-game mais lembrado do mundo (CLUA e BITTENCOURT, 2004). O objetivo do desenvolvimento é possibilitar a realização de testes de funcionalidades implementadas para o GameTVD.

O jogo desenvolvido consiste em um personagem, *Pac-Man* controlado por um usuário, posicionado em um labirinto simples, repleto de pastilhas pequenas e grandes e 4 (quatro) fantasmas que o perseguem neste labirinto. O objetivo é "comer" todas as pastilhas sem o *Pac-Man* ser alcançado pelos fantasmas, cada pastilha "comida" conta pontos (*scores*), e quando for a pastilha grande, durante alguns segundos, ele poderá até "comer" os fantasmas.

Poderia até ser feito um *redesign* do jogo, porém, opção foi desenvolver a versão tradicional utilizando as mesmas regras do jogo, os movimentos dos personagens acontecem sempre em uma das direções, vertical ou horizontal, para cima, para baixo, para a direita ou para a esquerda.

O Jogo é composto por quatro telas, denominadas *room.* A primeira *room* é a tela inicial do jogo, que apresenta botões de ínicio para iniciar o jogo, ajuda onde contém as instruções para jogar e sair, para finalizar o jogo.

A Figura 5.1 representa a *room* inicial com os seus respectivos objetos.

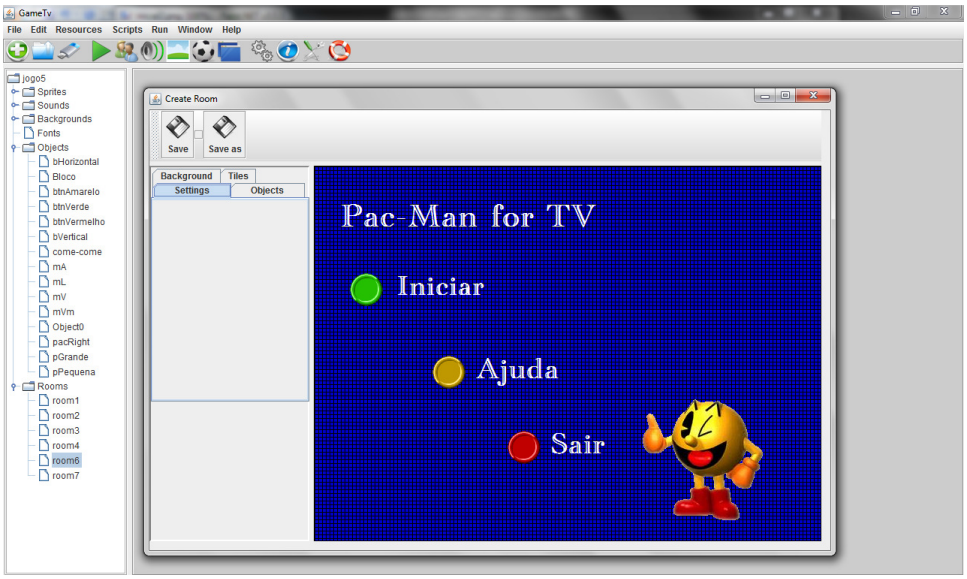


Figura 5.1: Criando *room* inicial.

As cores dos botões foram escolhidas de acordo com as teclas coloridas especiais do controle remoto, que se encontra no emulador do lado direito. A *room* de inicio pode ser visualizada no emulador correspondente a Figura 5.2.

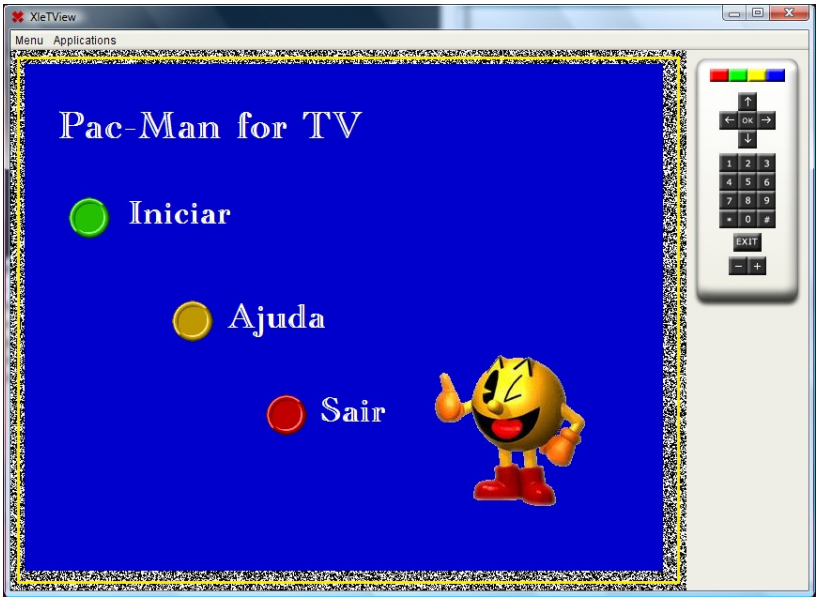


Figura 5.2: *Room* inicial do jogo no emulador.

Na Figura 5.3 são apresentados os componentes do cenário principal do jogo, representados pelos *sprites*.

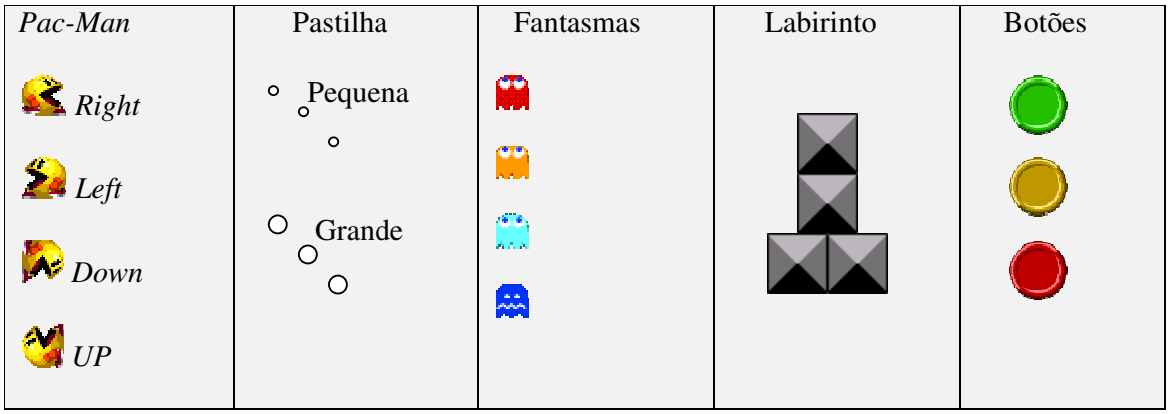


Figura 5.3: Sprites do jogo *Pac-Man*

De acordo com a abordagem proposta, após a inserção dos *sprites*, devem-se criar os objetos do jogo. Cada jogo deve ter diversos gerenciadores de objetos, sendo responsável pelo gerenciamento dos objetos similares. Dessa forma os seguintes objetos foram criados para comporem o cenário do jogo, conforme mostra a Figura 5.4.

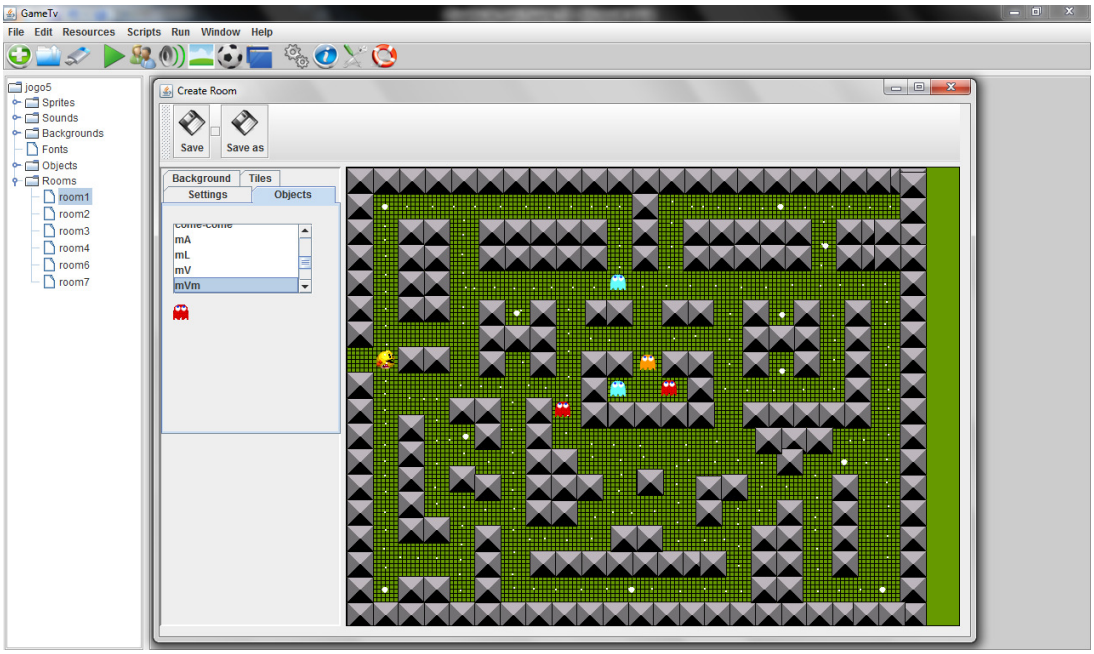


Figura 5.4: *Room* desenvolvido no GameTVD.

Os objetos do cenário criado possuem ações inseridas pelo framework, estas ações foram detalhadas no item 4.3, as quais permitem as seguintes responsabilidades para os objetos:

- *Pac-Man*: responsável pelo gerenciamento do mesmo e pelas suas ações tomadas, caso o usuário utilize a seta para esquerda mudará para o desenho *Left* e irá para esquerda, seta para baixo mudará pra o desenho *Down* e para baixo, seta para cima mostra desenho *UP* e para direção acima, seta para direita mudará apresenta o *Pac-Man Right* e direção direita. Caso colide com outro objeto da tela também gerenciará sua ação, caso seja uma das paredes do Labirinto fica parado, sendo uma pastilha continua andando, caso seja um fantasma perde uma vida do jogo.
- Fantasma: responsável pelo tratamento de colisão do fantasma com os outros objetos da cena, a ação dada a esse *sprite* é movimentar-se aleatoriamente na tela até encontrar o *Pac-Man.*
- Pastilha pequena: ao colidir com o *Pac-Man* a pastilha sumirá e contará pontos ao jogo, denominado *Score*.
- Pastilha grande: ao colidir com o *Pac-Man* a pastilha sumirá e permitirá que o *Pac-Man* possa "comer" os fantasmas, por um determinado tempo.

A Figura 5.5 representa o objeto do *Pac-Man* contendo os eventos e ações.

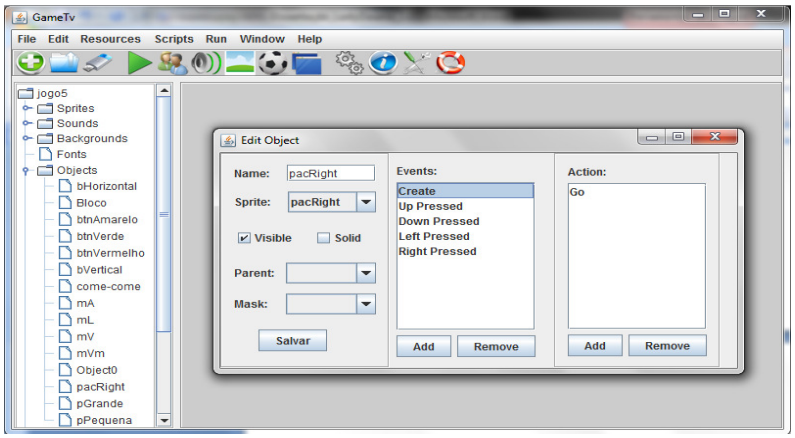


Figura 5.5: Objeto *Pac-Man*

A Figura 5.6 mostra a *room* principal do jogo executando no emulador, o labirinto foi montado com os *sprites* de forma aleatória e o mesmo pode ser modificado, ou criado outra *room* caso deseje outros níveis para o jogo. O jogo em questão possui apenas um nível de dificuldade e apresenta na tela a pontuação obtida pelo *Pac-Man* através dos *scores* e ainda as vidas através dos *lives*.

Figura 5.6: *Room* do jogo *Pac-Man* for TV.

A tela de ajuda é muito importante em um jogo, por isso, ao selecionar o botão vermelho de ajuda, aparecerá a tela de ajuda com as instruções do jogo, conforme pode ser observado na Figura 5.7. As cores relacionadas ao background das telas foram definidas de modo a combinar com as cores dos botões.

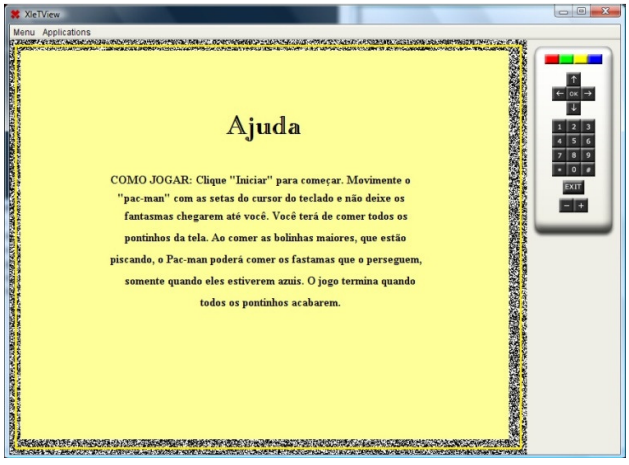


Figura 5.7: *Room* de ajuda do jogo *Pac-Man* for TV.

O fim do jogo acontece quando o *Pac-man* “come ” todas as pastilhas ou quando ele é “comido” por um fantasma por três vezes, terminando com as suas vidas (*lives*), ou ainda quando na Figura 5.2 é selecionado o botão vermelho, aparecendo a última *room* do jogo, representada pela Figura 5.8.

Figura 5.8: *Room* final do jogo *Pac-man* for TV.

5.1.2 Space Invaders

O *Space Invaders* é um jogo clássico de plataforma, trata-se de uma nave espacial que possui o objetivo de destruir outras naves para ganhar o maior número de pontos possíveis. Para a construção do jogo, foi adicionado um background, como mostra a Figura 5.9.

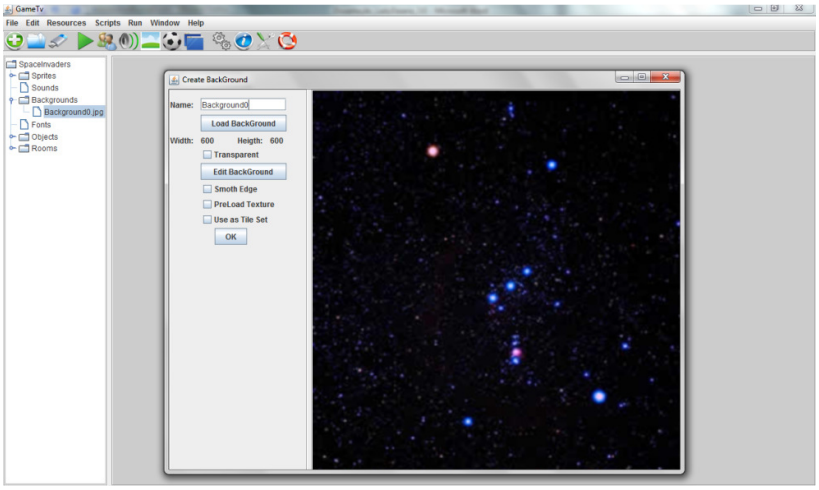


Figura 5.9: Interface inserir background.

Em seguida, foram criados os *sprites* do jogo e os objetos com seus respectivos eventos e ações, conforme mostrado no item 4.3. O jogo em questão foi de fácil implementação, necessitando apenas entender as regras do jogo e utilizar as ações implementadas no *Framework*.

Como exemplo de regra do jogo, a nave só poderia atirar caso o seu tiro anterior estivesse desaparecido na tela. Tal situação foi tratada com uma variável. Se a variável estiver com o valor lógico um, a nave pode atirar caso contrário não poderá atirar. Ou seja, quando o tiro colidir com a grade ou outro objeto essa variável recebe um, e enquanto o tiro não colidir essa variável permanecia com o valor zero.

A nave principal apresenta os eventos de movimentos verticais para direita e esquerda, conforme a tecla pressionada e atira quando se pressiona a tecla <enter> no controle remoto. Caso a nave tenha uma colisão com um dos tiros das naves inimigas deve desaparecer, perdendo assim o jogo. Os eventos relacionados à nave podem ser visualizados na Figura 5.10.

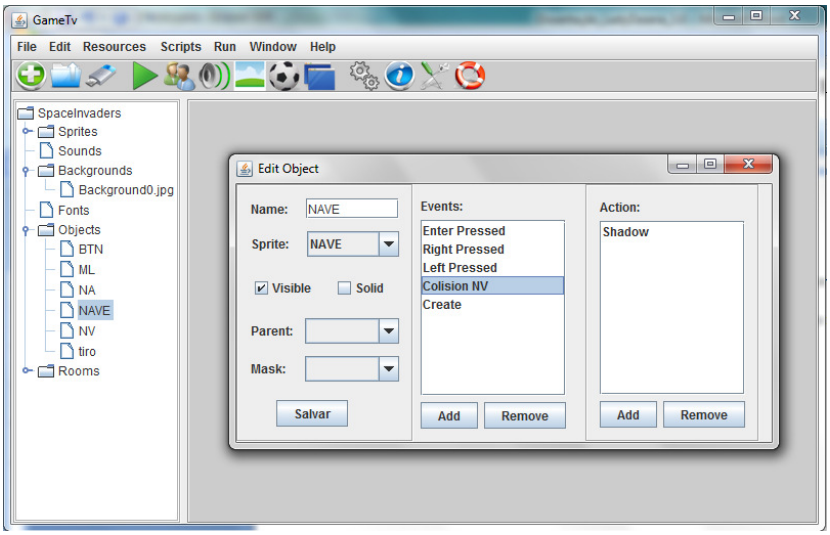


Figura 5.10: Interface inserir evento no objeto nave.

A Figura 5.11 representa o cenário principal do jogo *Space Invaders*, O cenário possui alguns tipos de naves, tendo a nave principal o objetivo de eliminar as outras naves

apresentadas na *room*. A cada nave eliminada, o jogador ganha *scores*, caso a nave seja acertada por uma nave inimiga perderá vidas, após perder três vidas em *lives*, o jogo termina.

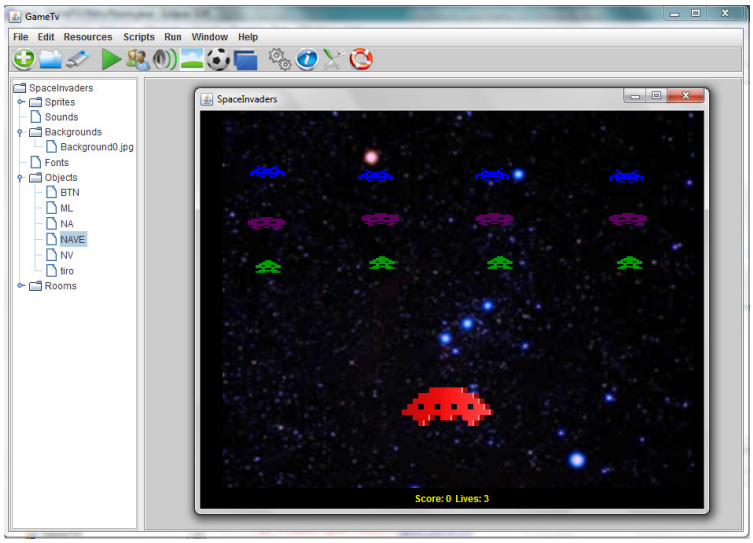


Figura 5.11: Jogo *Space Invaders*.

5.2 Testes e Resultados

O estudo de caso realizado possuiu o objetivo de verificar as funcionalidades do *Framework*, identificando as qualidades, defeitos e futuros módulos para o desenvolvimento de jogos para TVD. Observou-se aspectos como desempenho, usabilidade e portabilidade para o padrão brasileiro de TV digital. Visto que, no desenvolvimento desse projeto o Ginga-J ainda não estava pronto, tendo apenas as especificações que ainda estão sendo desenvolvidas.

O *Framework* cria e executa os jogos na plataforma PC, depois de compilado, gera o código compatível com o ambiente de TVDI, testando no emulador, sendo que, o código gerado não apresentou nenhuma incompatibilidade. Observou-se que à medida que se insere os objetos na *room* (cenário), o jogo passa a demorar mais para ser carregado. Isso acontece pelo fato da necessidade de ler no arquivo XML todos os objetos que serão carregados em memória, aumentado o tempo de carregamento do *xlet*.

O GameTVD facilitou a criação de interfaces e o tratamento de eventos, pois não foi necessário conhecer, a priori, nenhuma linguagem de programação para poder utilizá-lo.

Permitiu ainda, uma navegação entre telas no desenvolvimento e conseqüente aumento na percepção das diversas partes de desenvolvimento do jogo. Entretanto, um dos fatores primordiais do uso do *Framework,* que se relaciona com ao tempo de desenvolvimento do jogo, não pôde ser quantitativamente mensurado, embora a percepção que se obteve é de que o mesmo permitiu uma economia de tempo e diminuição dos esforços no momento da criação do jogo, uma vez que o *Framework* dispõe de classes com os principais eventos realizados em um jogo, como andar, pular, parar, etc., e com isso foi possível ter reusabilidade e ganho de tempo.

Nos testes, percebeu-se um atraso de tempo maior em relação a uma ação do jogador e a visualização deste no jogo no emulador de TVDI, o que não acontecia no PC. Isso ocorreu devido a limitações do emulador que simula as limitações reais de memória, capacidade de processamento específico deste tipo de plataforma.

Sem a intenção de ser uma solução definitiva, o GameTVD ainda necessita de adaptações e testes em ambiente real de TV digital, que ainda se encontra em implantação, bem como padronizações que sociabilizem sua utilização. Além disso, para implementações futuras seria interessante um módulo de implementação de algoritmos básicos de inteligência artificial e suporte a uma linguagem de script.

6 CONSIDERAÇÕES FINAIS

Comparando a TVDI ao mercado de vídeo games observa-se que há poucos jogos para a TV, sendo mais freqüentes jogos simples e com interatividade local, que imitam os antigos jogos árcades. No entanto, o STB pode ser considerado como uma plataforma com potencial para jogos, pois os recursos de hardware em breve estarão disponíveis, como e-mail, telefone através do canal de retorno, podendo a TVDI ser um ambiente para jogos com participação massiva e ainda competir com as outras plataformas.

Observou-se a falta de referência a respeito dos padrões mundiais de TVDI suportarem placas gráficas aceleradoras e de texturização três dimensões - 3D e nem a linguagens 3D como *Open Graphics Library* - OpenGL, *Virtual Reality Modeling Language* - VRML e Java3D, no que se refere a um STB padrão. A existência de tal suporte permitirá um melhor desempenho dos STBs para jogos. A TVDI atualmente pode utilizar gráficos 2D, transparências, animações, botões e textos, mas não é padrão contar com teclados e joystick.

O *Framework* proposto neste trabalho contribuiu com o aumento da demanda da criação de jogos para TVDI, facilitando e incentivando os desenvolvedores de software e provedores de conteúdo televisivo no desenvolvimento de jogos para essa nova plataforma, com o objetivo de garantir rotinas funcionais necessárias em quase todos os jogos digitais. A utilização do *Framework* para o desenvolvimento de jogos para TV digital é uma alternativa viável e torna o processo mais acessível para desenvolvedores iniciantes e estudantes e facilita o trabalho de profissionais da área. Esse estudo contribuiu para criação de jogos para TVDI e pode ser usado futuramente como base para outros projetos que se destinam a desenvolver jogos para esta plataforma.

6.1 Dificuldades encontradas

A principal dificuldade enfrentada foi a escassez de documentação relacionada ao desenvolvimento de jogos para TVDI. Além disso, a especificação do Ginga-J que previa utilizar a API HAVI para criação de interfaces, foi modificada durante a etapa de implementação do *framework* e passou a utilizar a API LUWIT. O projeto continuou utilizando as APIs HAVI e AWT, contudo a mudança para a API Gráfica a ser escolhida no sistema brasileiro de TV digital não é uma tarefa complexa.

A falta das normas relacionadas às especificações do Ginga-J constituíram também uma outra dificuldade, desde a escolha das APIs à necessidade de implementação de algumas funcionalidades relacionadas aos jogos, assim como isso gerou uma limitação dos testes em ambientes reais, visto que as especificações foram aprovadas somente no mês de agosto de 2010, não havendo no momento STBs comerciais com o Ginga-J.

6.2 Trabalhos Futuros

Como trabalhos futuros são apresentadas as pesquisas que poderão dar continuidade a esse trabalho relacionados a aperfeiçoamentos do *Framework* GameTVD, através de extensões ou módulos. As principais possibilidades são:

- Desenvolver um *game engine* para integrar com o *Framework* aumentando o escopo;
- Criar módulos voltados para inteligência artificial, gerenciador de múltiplos jogadores e canal de retorno;
- Aplicabilidade do *Framework* GameTVD para jogos educacionais;
- Otimizações em relação ao tempo de execução e carregamento dos objetos para melhorar a performance do GameTVD.

7 REFERÊNCIAS BIBLIOGRÁFICAS

ABNT. Associação Brasileira de Normas Técnicas. **Parte 4: Ginga-J - Ambiente para a execução de aplicações procedurais**: NBR-15606-4: 2010.

ABRAGAMES, 2009**. A indústria brasileira de jogos eletrônicos. Um mapeamento do crescimento do setor nos últimos 4 anos**. Pesquisa da Associação Brasileira das Desenvolvedoras de Jogos Eletrônicos. Disponível em: www.abragames.org. Acessado em: 12 de ag. de 2009.

ALMEIDA, M. R. G. **Programação de Jogos 2D usando o MRDX e Fundamentos de Jogos 3D**. 2004. 73f. Monografia (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Federal do Pará, Belém. 2004.

BARBOSA S.D.J.; SOARES, L.F.G. **TV digital interativa no Brasil se faz com Ginga: Fundamentos, Padrões, Autoria Declarativa e Usabilidade**. Em T. Kowaltowski & K.Breitman (orgs.) Atualizações em Informática 2008. Rio de Janeiro, Rj: Editora PUC-Rio, 2008. pp.105-174.

BARBOZA, D. C.; CLUA, E. W. G. **Ginga Game: A *Framework* for Game sevelopment for the interactive Digital Television**. Simpósio Brasileiro de Games – SBGAmes, Rio de Janeiro, 2009.

BARROS, T. G. F., 2003. ***SYMBG(R)AF – Symbian Games Framework.*** Trabalho de Conclusão de Curso, Universidade Federal de Pernambuco, CursoCiência da Computação. Recife, 2003.

BATTAIOLA, A. L.. 2000. **Jogos por Computador – Histórico, Relevância Tecnológica e Mercadológica, Tendências e Técnicas de Implementação** In: XIX Jornada de Atualização em Informática. Curitiba:SBC, Julho/2000, v. 2. pp. 83-122.

BECKER, V.; MONTEZ, C., 2005. **TV digital Interativa: Conceitos, desafios e perspectivas para o Brasil.** Florianópolis: Editora da UFSC, 2005.
BITTENCOURT, J. R.; CLUA, E.W.G. **Desenvolvimento de Jogos 3D: Concepção, Design e Programação.** XXV Congresso da Sóciedade Brasileiro de Computação – SBC, Jornadas de Atualização em Informática - JAI, 2005.

BURLAMAQUI, A. M. F.. **N2N *Framework*: Uma Plataforma para Desenvolvimento de Sistemas Colaborativos Distribuídos**. Dissertação de mestrado do Programa de Pós-Graduação em Sistemas e Computação - PPgSC/UFRN, Rio Grande do Norte, 2004.

CLUA, E.W.G, BITTENCOURT, J. R. **Uma Nova Concepção para Criação de Jogos Educacionais**. XV Simpósio Brasileiro de Informática na Educação- SBIE, Anais vol. II. Manaus: UFAM, 2004.

COSTA, B. R.; SANTOS, J. F., ARTHUR, R., 2008. ***Framework*** **Multiplataforma para Desenvolvimento de Jogos 2D e aplicações Multimidias.** Anuário da Produção de Iniciação Científica Discente. Vol. XI, n. 12, Ano 2008.

DTV. **Site Oficial da TV digital Brasileira**. Disponível em http://www.dtv.org.br/materias.asp?menuid=3&id=11. Acessado em: 09. fev. 2010.

FERREIRA, D. A; SOUZA, C. T. **TuGA: Um *Middleware* para o Suporte ao Desenvolvimento de Jogos em TV digital Interativa**. NASH – Núcleo Avançado em Engenharia de Software Distribuído e Sistemas Hipermídia. Centro Federal de Educação Tecnológica do Ceará. Infobrasil2008. Fortaleza, 2008.

FÓRUM SBTV. **Sistema Brasileiro de TV digital**. Disponível em http://www.forumsbtvd.org.br/ . Acessado em: 25. Jan. 2010.

GAMEMAKER. **Gamemaker Lite.** Disponível em :http://www.yoyogames.com/make. Acessado em: Julho. 2009.

GAMMA, E.; HELM, R.; JOHSON, R. E.; VISSIDES, J. (1995): **Design patterns: elements of reusable object-oriented software**. Addison-Wesley, Reading, MA, 1995.

HATTORI, L. P.;MARQUES NETO, M. C. ; SILVA, S. S. ; SANTOS, C. A. S.; TAVARES, T. A. **Utilizando o *Framework* AppTV no Desenvolvimento de Aplicacoes para TV digital Interativa.** In: Brazilian Symposium on Computer Graphics and Image Processing, SIBGRAPI, 2005, NATAL. Workshop de TV digital e Interativa. Natal: National Library, 2005. v. 1. p. 1-6.

HAVI. **Home Audio Video**. Disponível em: http://www.havi.org. Acessado em: Junho,2010.

JACCOBER, E. C.. **Proposta e implementação de uma interface para motores de jogos interativa e concentrada no usuário**. Dissertação de mestrado apresentada a Escola Politécnica da Universidade de São Paulo, 2007

LAVID. **Laboratório de Aplicações de Vídeo Digital**. Disponível em: http://www.lavid.ufpb.br. Acessado em: Junho, 2010

LEMOS, G.; FERNANDES, J.; SILVEIRA, G. **Introdução à Televisão Digital Interativa:**

Arquitetura, Protocolos, Padrões e Práticas. In: Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Comunicação, 2004, Salvador. Anais. Salvador: [s.e.], 2004.

LOUREIRO, J. A**. Interfaces de Programação para o Desenvolvimento de Aplicações para TV digital**. Pernambuco: UFPE, 2004. Monografia, Faculdade em Ciência da Computação, Centro de Informática, Universidade Federal de Pernambuco, 2004.

OLIVEIRA, B. J. D.; SILVA, J. C. F; TAVARES, A. T.. **Um estudo entre Ginga-J e Ginga-NCL no âmbito de aplicações interativas residentes**. I Simpósio Internacional de TV digital, Baurú – SP, 2009.

ASSIS, S. R.; SUZANO, R. ***Framework*: Conceitos e Aplicações.** Científico. Ano III, V. II. Salvador, 2003.

BARRETO JR., C. G. **Agregando *Framework*s de Infra-Estrutura em uma Arquitetura Baseada em Componentes: Um Estudo de Caso no Ambiente AulaNet.** Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2006.

HORSTMANN, C.S.; CORNELL, G. 2001. **Core JAVA 2 Volume I – Fundamentos**. MAKRON Books, São Paulo.

JOHNSON, R. E. (1991): Reusing Object-Oriented Design, University of Illinois.
MORRIS, S.; SMITH-CHAIGNEAU, A. 2005. **Interactive TV Standards.** Elsevier Inc: 2005.

ORACLE, **Java DTV API 1.0 Specification for digital terrestrial TV related to the SBTVD** . Disponível em: http://www.oracle.com/technetwork/Java/Javame/tech/Javatv-136131.html. Acessado em: Junho, 2010

PESSOA, C. 2001. **wGEM: Um Framewok de Desenvolvimento de Jogos para Dispositivos móveis**. Dissertação de Mestrado. Universidade Federal de Pernambuco-Graduação em Ciência da Computação. Pernambuco, 2001.

PINTO, L. D. O, QUEIROZ-NETO, J. P., SILVA, K. N. **Tecnologias utilizadas para o desenvolvimento de aplicações educacionais através da TV digital**. II Congresso de Pesquisa e Inovação da Rede Norte Nordeste de Educação Tecnológica – CONNEPI, Paraíba, 2007.

PRESSMAN, R. S. 2002. **Engenharia de Software**. 5. Ed – Rio de Janeiro: McGraw-Hill. 2002.

QUICO, C.,2009. **It came from outer space: Jogos de computador invadem programação e serviços de televisão digital.** Texto da Biblioteca Online de Ciências da Comunicação, 2003.

ROCHA, J. T. S., 2003. **Forge 16V: Um *Framework* para Desenvolvimento de Jogos Isométricos.** Dissertação de Mestrado. Coordenação da Pós-Graduação em Ciências da Computação do Centro de Informática. Universidade Federal de Pernambuco. Pernambuco, 2003.

RODRIGUES, L. M.. **IPTV: Conceitos, Padrões e Soluções**. Trabalho de Conclusão de Curso, Ciência da Computação, No. 05/06 ISSN: 0103-9741, PUC-Rio, Rio de Janeiro, 2006.

SANTOS, J. A. F. ; MARTINS, E. R. ; ARRUDA, J. P. S. ; DANTAS, M. ; SANCHES, M. L. A. ; SAADE, D. C . **RummiTV: An Interactive Game for the Brazilian Digital TV System.** In: XXVI Simpósio Brasileiro de Telecomunicações, 2008, Rio de Janeiro. SBRT 2008, 2008.

SILVA, A. R.; OLIVEIRA, M. R. **Projeto de um *Framework* de Desenvolvimento de Interfaces Gráficas para TV digital.** Trabalho de Conclusão de Curso, Universidade de Brasília – UnB. Brasília, 2006.

SILVA, G. M.; SADZEVICIUS, J. E. S.; MATEUS, N. S.. **Corrida do Conhecimento: Um Estudo De Caso Sobre o Uso da Tvdi Na Educação**. 14° Congresso Internacional ABED de Educação a Distância - "Mapeando o Impacto da EAD na Cultura do Ensino-Aprendizagem" - Santos (SP), 2008

SILVA, J. Q. **TV digital Interativa.** São Leopoldo: UVRS, 2004. Monografia, Curso de Especialização em Redes de Computadores, Centro de Ciências Exatas e Tecnológicas, Universidade do Vale do Rio dos Sinos.

SILVA, R. P; **Suporte ao desenvolvimento e uso de *Framework*s e componentes**. Tese. Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, 2000.

SOARES, L. F. G., 2007. **Ambiente para desenvolvimento de aplicações declarativas para a TV digital brasileira.** TV digital: qualidade e interatividade / IEL.NC.– Brasília : IEL/NC, 2007.

SOARES, L. F. G; CASTRO, P. H., 2008. **As Múltiplas Possibilidades do Middleware Ginga**. Disponível em: http://www.gingancl.org.br/resources/SBTVD-Ginga.pdf. Acessado em: fev. 2009.

TAROUCO, L. M. R. et. al. **Jogos Educacionais**. Novas Tecnologias na Educação-CINTED-UFRGS, V.2, Nº 1, Florianópolis, 2004.

VALENTI, L., 2005. **GUFF: Um *Framework* para Desenvolvimento de Jogos.** Dissertação de Mestado. Programa de Pós-Graduação em Computação, Universidade Federal Fluminense, Rio de Janeiro, 2005.

ZANCANARO, A.; SANTOS, P.; TODESCO, J. L., 2009. **Ginga-J ou Ginga-NCL: características das linguagens de desenvolvimento de recursos interativos para a TV digital.** 1º Simpósio Internacional de Televisão Digital (SIMTVD) – 18 a 20 de novembro, Bauru/SP. 2009. Acessado em: fev. 2010. Disponível em: http://www2.faac.unesp.br/pesquisa/lecotec/eventos/simtvd/trabalhos.php.

APENDICE A

PUBLICAÇÕES REALIZADAS DURANTE O MESTRADO

PINTO, L. D. O. ; QUEIROZ-NETO, J.P ; LUCENA, V. F. . **Uma Proposta de Arquitetura Para Framework de Jogos Digitais Para TV Digital**. In: IV Congresso de Pesquisa e Inovação da Rede Norte Nordeste de Educação Tecnológica - CONNEPI, 2009, Belém- PA. Connepi 2009, 2009.

PINTO, L. D. O. ; SILVA, V. J. ; ROSA, R. E. V. S. QUEIROZ-NETO, J.P; LUCENA, V. F. **An experience to use robotics to improve computer science learning.** In: Frontiers in Education - 39th Annual, 2009, San Antonio, TX. IEEE Frontiers in Education - 39th Annual, 2009.

PINTO, L. D. O. ; QUEIROZ-NETO, J.P ; LUCENA, V. F. . **An engineering educational application developed for the Brazilian digital TV system**. In: Frontiers in Education - 38th Annual, 2008, Saratoga Springs - New York. IEEE Frontiers in Education - 38th Annual, 2008. p. S2F-14-S2F-19.

PINTO, L. D. O. ; SILVA, V. J. ; ROSA, R. E. V. S. ; LUCENA, V. F. ; QUEIROZ-NETO, J.P. **A Robótica como Ambiente de Programação Utilizando o Kit Lego Mindstorms**. In: XIX SBIE - Simpósio Brasileiro de Informática na Educação, 2008, Fortaleza- CE. II Workshop de Ambientes de Apoio à Aprendizagem de Algoritmos e Programação, 2008.

PINTO, L. D. O. ; SILVA, V. J. ; ROSA, R. E. V. S. ; QUEIROZ-NETO, J.P; LUCENA, V. F. . **Desenvolvimento de Aplicações Embarcadas para Lego RCX-2.0 Utilizando Software Livre**. In: III CONNEPI - Congresso de Pesquisa e Inovação da Rede Norte Nordeste de Educação Tecnológica, 2008, Fortaleza -CE. Anais do III CONNEPI, 2008.