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

Athus: um framework para o desenvolvimento de jogos para TV Digital utilizando Ginga

RICARDO MENDES COSTA SEGUNDO

JOÃO PESSOA - PB
~~SETEMBRRO DE 2011~~

RICARDO MENDES COSTA SEGUNDO

Athus: um framework para o desenvolvimento de jogos para TV Digital utilizando Ginga

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Centro de Ciências Exatas e da Natureza da Universidade Federal da Paraíba, como parte dos requisitos para a obtenção do título de Mestre em Informática.

Orientadora: Profª. Dra. Tatiana Aires Tavares

JOÃO PESSOA - PB
SETEMBRRO DE 2011

RESUMO

O Ginga, middleware do Sistema Brasileiro de TV Digital, tem ganhado importância internacional, sendo adotado por diversos países da América do Sul e ao se tornar recomendação ITU para IPTV. Com isso, o sinal digital, irá ser recebido por mais televisores, levando vídeo de alta definição e áudio de alta qualidade aos telespectadores, como também a possibilidade de interagir com as aplicações televisivas. As possibilidades de interatividade destas aplicações são inúmeras: enquetes, acompanhamento de estatísticas em eventos esportivos, informações adicionais e complementares sobre novelas e seriados, entre outras. Dentre estas aplicações, um tipo que poderá tomar espaço são os jogos, os quais vêm ganhando destaque no mercado mundial em diversas plataformas. Eles possuem um caráter de entretenimento e desafio, variando de jogos simples, acessíveis a todos, a jogos mais complexos voltados a jogadores experientes. Criar uma ferramenta que auxilie no desenvolvimento de jogos para o Ginga se torna um passo importante para facilitar a produção de jogos que possam atingir milhões de pessoas não só no Brasil, mas em todos os países que adotarem o padrão brasileiro. Assim sendo, este trabalho apresenta o Athus, um framework de desenvolvimento de jogos voltado para a Televisão Digital. Para isto foi feito um estudo sobre os diversos aspectos dos jogos, tais como a indústria, formas de produção e categorias; e também, uma pesquisa acerca da produção dos jogos para a TV Digital no Brasil e no mundo. No decorrer do trabalho é mostrado como os jogos poderão ser utilizados sobre a plataforma Ginga, a construção e uso do framework e os resultados obtidos com sua utilização (jogos desenvolvidos e testes realizados com programadores).

Palavras-chave: Jogos, Ginga, Framework, SBTVD, TV Digital;

ABSTRACT

Ginga, the Brazilian System of Digital TV middleware, has gained international importance, being adopted by several countries in South America and become a ITU recommendation for IPTV. Thus, the digital signal will be received by most televisions, leading high-definition video and high-quality audio to viewers, but also the possibility to interact with television applications. The possibilities of interactivity are many: surveys, monitoring statistics at sport events, and additional information about soap operas and sitcoms, among others. In these applications, a type that may take place are the games, which are gaining prominence in the world market on multiple platforms. They have a character of entertainment and challenge, ranging from simple games, accessible to all, the more complex games geared toward experienced players. Create a tool that helps in the development of games for the Ginga becomes an important step to facilitate the production of games that can reach millions of persons not only in Brazil but in all countries that adopt the Brazilian standard. Therefore, this paper presents the Athus, a game development framework targeted for Digital Television. To achieve this, many studies were done on the various aspects of games, such as industry, production forms and categories, and also a research on the production of games for digital TV in Brazil and worldwide. During the work is shown how the games can be used on the platform Ginga, the construction and use of the framework and the results obtained with its use (and games developed tests to programmers).

Keywords: Games, Ginga, Framework, SBTVD, Digital TV;

Sumário

Capítulo

1

Introdução

A televisão no Brasil assume papéis importantes, sendo fonte de informação e de entretenimento para muitos brasileiros. O advento da Televisão Digital no Brasil traz aos usuários não apenas uma melhor experiência em termos de som e imagem, mas possibilita a interatividade.

A interatividade por si só não é algo novo na TV, programas como *Wink Dink and You (Winki Dink, 1953), Você Decide (Você Decide, 1992) e Big Brother (Big Brother, 2000)* possuem recursos de interatividade. Porém apenas no primeiro programa citado existe interação direta com a televisão, todos os outros a interação ocorre por meios externos como telefone e Internet. Já o programa *Wink Dink and You* apesar de interagir através da televisão, o faz de forma analógica. O que a TV Digital traz de novo é a forma de interagir com o telespectador, ele não terá que utilizar outros dispositivos (celular, computador, giz de cera,...) como intermediários para a interação.

Sobre a interatividade, é possível pensar em um tipo de aplicação que leva este termo ao extremo, os jogos. Os jogos possuem a característica de envolver os jogadores de tal forma que fazem com que eles se sintam dentro da aplicação, ou seja, eles não se sentem apenas controlando os personagens, sentem fazer parte deste (Schell, 2008). E este tipo de experiência tem na maioria das vezes o objetivo de entreter, fazendo dos jogos uma forma de entretenimento bastante difundida nos dias de hoje.

1.1 Motivação

A primeira transmissão televisiva no Brasil ocorreu em 1950. Era o início da TV analógica, uma grande novidade, mesmo com a imagem em preto e branco, com pouca nitidez e muito chuvisco (Menezes, 2009), e acessível aos poucos que podiam comprar um aparelho de TV.

Em 1972, ocorreu outra mudança, o mundo preto e branco da TV ganhou cores. Graças ao padrão adotado, PAL-M (Iano, 1994), os aparelhos antigos continuaram a funcionar sem problemas, porém sem a possibilidade de ver as grandes mudanças que as cores traziam a programação.

Em dezembro de 2007, a forma de ver televisão começou a mudar mais uma vez, o Brasil entrou na era da TV digital, com melhor qualidade de som e imagem e a possibilidade da interatividade (Menezes, 2009).

Já os jogos são milenares ferramentas de entretenimento para os seres humanos, também servindo como uma ferramenta para o seu crescimento intelectual. O jogo também é uma forma de superarmos nossos limites, já que eles em essência possuem um caráter de desafio e para um jogo existir ele deve dar ao jogador a possibilidade de resolver problemas gerados pela dinâmica do jogo. Sendo assim, temos uma definição simples de jogo: jogo é uma atividade de resolução de problemas, que proporciona divertimento (Schell, 2008). Analogamente, os jogos eletrônicos, ou vídeo games, também possuem essa definição como base, porém eles possuem algo a mais: a interação com o jogo pelo jogador é feita através de algum tipo de aparelho eletrônico, que gera imagens e símbolos que são manipulados e modificados de acordo com as ações do jogador. Logo, jogos para TV Digital podem ser categorizados como jogos eletrônicos e compartilham importantes características de entreter e nos estimular intelectualmente.

1.2 Objetivos

Este trabalho tem como **objetivo principal** o desenvolvimento de um framework para desenvolvimento de jogos para Televisão Digital através do middleware Ginga, de forma a possibilitar ao desenvolvedor customizar o framework para desenvolver aplicações específicas utilizando instancias das classes do framework.

Para concretizar o desenvolvimento do framework proposto, temos os seguintes **objetivos específicos**:

- Investigar os modelos de jogos para TV Digital, especialmente para o SBTVD.
- Estudar as tecnologias envolvidas (para TV Digital e para desenvolvimento de Jogos).
- Propor, especificar e modelar um framework para o desenvolvimento de jogos para TV Digital com Ginga.
- Testar o framework proposto em termos de funcionalidades oferecidas.
- Validar o framework proposto para testar sua aceitação com programadores.

1.3 Trabalhos Correlatos na Instituição

O Laboratório de Aplicações de Vídeo Digital (LAVID) é internacionalmente reconhecido por seus estudos e desenvolvimento acerca do Sistema Brasileiro de Televisão Digital, assim, existem diversos trabalhos relacionados ao desenvolvimento de aplicações para TV Digital, como o OpenGinga, o Ginga App Store, a Estação-Escola de Televisão Digital (EETVD) e o BEACON.

1.3.1 OpenGinga

OpenGinga (OpenGinga, 2011) é uma plataforma que permite executar aplicações Ginga num computador pessoal, incluindo o sistema operacional, uma implementação de referência do middleware Ginga e aplicações exemplo. Ele tem o objetivo de oferecer uma implementação completa (não emulada) de código aberto de um middleware de TV Digital compatível com o padrão SBTVD e executável em ambiente de PC, oferecendo suporte às APIs definidas nas especificações Ginga-J e GingaNCL.

O OpenGinga foi projetado para plataformas PC com o sistema operacional Linux. Com o OpenGinga instalado, um computador com suporte a hardware específico pode ser usado como um receptor de TV Digital.

Como um dos resultados do projeto, desenvolvedores de aplicações podem baixar o ambiente OpenGinga e instalá-lo nos seus computadores, de forma que estes irão simular o funcionamento do middleware Ginga, executando aplicações para TV Digital, incluindo fluxos de vídeo compatíveis com os padrões Ginga.

1.3.2 Ginga AppStore

A partir da implantação do Sistema Brasileiro de TV Digital – SBTVD, surgiram inúmeras possibilidades de aplicações que não se restringem apenas a melhoria de qualidade áudio-visual. A finalidade precípua do processo de implantação do SBTVD é prover um padrão que proporcione tanto a transmissão de vídeo com alta qualidade quanto à "inclusão digital" para parte da população que vive hoje à margem da "sociedade da informação".

Neste cenário, o SBTVD é um recurso fundamental, já que visa disponibilizar um padrão capaz de proporcionar serviços, como: telessaúde, teleeducação, além de interatividade para realizar, por exemplo: enquetes, compras on-line, videoconferência, e etc. Entretanto, atualmente ainda há um conjunto de entraves quanto à disponibilização de conteúdo e implementação de canal de retorno para o usuário final do SBTVD, notadamente em regiões com pouca infraestrutura de telecomunicações. Este projeto apresenta uma alternativa conjunta para viabilizar a implementação de um repositório público de aplicações interativas (denominado Ginga AppStore), que utiliza a infraestrutura para canal de retorno a partir da rede pública do NAVEGAPARÁ. Como estudo de caso, apresenta-se uma aplicação típica de telessaúde (ZéGotinhaTV) para controle de vacinação, que possibilita o suporte à decisão para políticas de saúde pública (Ginga App Store, 2011).

1.3.3 EEDTV

Este projeto visa uma rede de compartilhamento da infraestrutura física e equipamentos disponíveis em uma Estação-Escola de Televisão Digital. Para este compartilhamento é utilizada uma ferramenta que viabiliza o uso remoto e compartilhado dos equipamentos da EETVD, o VirtuaLabTV, que é um laboratório virtual para testes a distância de conteúdos audiovisuais interativos. Essa ferramenta configura os equipamentos de transmissão e recepção e reproduz as interações do usuário no ambiente real, além de mostrar ao usuário o

conteúdo sendo executado no ambiente real, tudo isso de forma simples e remota (EETVD, 2011).

Outro objetivo do projeto foi capacitar onze produtoras audiovisuais de forma que elas pudessem criar de forma autônoma produções interativas. As equipes das onze produtoras consorciadas receberam treinamento sobre os fundamentos da TV Digital, criação de roteiros interativos e produção de aplicações para TV Digital. Como resultado deste trabalho onze programas de televisão digital interativos foram gerados, constando de enquetes em programas de auditório a interações que trazem conteúdos extras ao telespectador que opte por interagir com o programa.

1.3.4 BEACON

BEACON (BEACON, 2011) é a sigla para Consórcio Brasileiro-Europeu para Serviços de Televisão Digital Terrestre. Este projeto visa o desenvolvimento de serviços inovadores na área de "*t-learning*" (aprendizado à distância através de televisão digital interativa), ligados à inclusão social. Os serviços desenvolvidos nesse projeto são baseados em pesquisa pioneira na interoperação entre o sistema de televisão digital terrestre europeu (DVB) e o sistema brasileiro (SBTVD). Além da faceta tecnológica, o projeto visa o desenvolvimento de metodologias eminentemente pedagógicas para promoção da educação à distância utilizando a TV Digital como ferramental didático.

O projeto BEACON tem por objetivos:

- Realizar pesquisa nos serviços inovadores de Televisão Digital Terrestre (TDT) endereçados para o domínio de ensino à distância (*t-learning*), notadamente customizados para as necessidades específicas do Brasil e relacionados a aspectos de inclusão social;
- Estabelecer um consórcio Brasileiro-Europeu que irá gerenciar a exploração dos artefatos e os serviços implementados pelas atividades do projeto;
- Disseminar os resultados do projeto e prover os resultados da pesquisa e análises dos dados resultantes da execução do piloto para a Administração Pública Brasileira, de forma a contribuir para o processo de construção de políticas na área de desenvolvimento da tecnologia e serviços de TDT.

1.4 Estrutura da Dissertação

Esta dissertação está organizada em sete capítulos. O Capítulo 1 aborda a introdução onde é realizada uma breve descrição do problema tratado, a motivação para o seu desenvolvimento e os principais objetivos. O segundo capítulo apresenta a fundamentação teórica necessária ao desenvolvimento do trabalho, abordando aspectos de TV Digital e Jogos. O terceiro capítulo descreve os trabalhos correlatos encontrados e uma análise dos mesmos. O quarto capítulo apresenta detalhes de implementação, arquitetura e outros aspectos do framework Athus. O quinto capítulo apresenta os experimentos realizados para verificação das funcionalidades do framework. O sexto capítulo apresenta testes realizados com o framework, apresentando a metodologia e os resultados obtidos. O último capítulo apresenta as considerações finais e o futuro da pesquisa.

Capítulo

2

Fundamentação Teórica

Neste capítulo são apresentados os diversos aspectos dos jogos e da televisão digital e as principais características dos games, suas classificações e formas de produção. Também são abordados os conceitos de interação na TV Digital, os tipos de aplicativos possíveis nesta plataforma e o funcionamento do middleware brasileiro.

2.1 Jogos Eletrônicos

Os jogos eletrônicos tomam a cada ano mais espaço na sociedade, nos Estados Unidos a sua evolução chega a confrontar outras grandes mídias, como a indústria do cinema. Segundo dados da *Entertainment Software Association* (ESA, 2010) a venda de videogames nos Estados Unidos atingiu a marca em 2010 de 10,5 bilhões de dólares, valor muito próximo a renda da indústria do cinema no mesmo país, que alcançou no mesmo ano, segundo a *Motion Picture Association of America* (MPAA, 2010), 10,6 bilhões de dólares, com a diferença de incluir nesta estatística o Canadá.

2.1.1 História dos Jogos Eletrônicos

Os jogos eletrônicos surgiram junto com a Guerra Fria, pois com o avanço da tecnologia nesta época, tornou-se possível fazer programas que não apenas fizessem cálculos, mas também diversas outras funcionalidades. O primeiro jogo eletrônico surgiu em 1958, criado por William A. Higinbotham, físico, que criou o jogo intitulado Tênis para Dois. O

jogo era bem simples, rodando em um osciloscópio, e foi criado com o propósito de entreter os visitantes do laboratório onde trabalhava. Em 1961, com maior influência da Guerra Fria, Steve Russel desenvolveu o *SpaceWar*, que era um jogo de naves onde os jogadores competiam entre si para destruir a nave do adversário. Este jogo tem outro aspecto interessante, pois ele foi distribuído entre os diversos laboratórios que possuíam equipamento semelhante ao de Russel, sendo assim jogado por centenas de pessoas, e também foi com o *SpaceWar* que surgiu o primeiro *Joystick*, com o intuito de melhorar a jogabilidade, substituindo os botões dos consoles que eram utilizados anteriormente (Discovery, 2007).

Surgiu nesta época então o pensamento de que você apesar de possuir uma TV em casa, não tinha a capacidade de interferir nesta, criando-se assim os primeiros consoles de jogos. O pioneiro foi o *MagnaVox* em 1972, que podia ser conectado a TV, possibilitando aos usuários jogar os jogos da época, dos quais se destacava um jogo que se tornou um marco da popularização dos vídeo games na sociedade, o Pong. O Pong era um jogo de dois jogadores, imitando uma partida de tênis, que se tornou febre não apenas nos consoles caseiros, mas também em fliperamas localizados em diversos estabelecimentos. Um fato interessante sobre o Pong é o de ele surgir junto ao movimento feminista nos EUA, tornando-se um jogo jogado por todos, onde mulheres podiam competir de igual para igual contra os homens (Discovery, 2007).

Porém, os jogadores e criadores começam a sentir a falta de algo que os ligasse aos jogos, ou seja, faltava uma narrativa. Foi então que outro clássico surgiu, o *Space Invaders* em 1978 criado por Tomohiro Nishikado, que tinha uma história simples de invasão alienígena a terra. Entretanto, esta pequena historia revolucionou os demais jogos que viriam a surgir a seguir, pois agora não bastava ser jogável, o jogo deveria prender a atenção do jogador de alguma forma. Algum tempo depois outra revolução ocorreu, o surgimento de *PacMan* (vide Fig.1) em 1980 criado por Toru Iwatani, que trouxe aos jogos o primeiro herói, pois antes não havia no jogo um personagem com quem o jogador pudesse se identificar, podendo agora imergir no jogo ao participar deste se pondo no lugar de tal personagem.

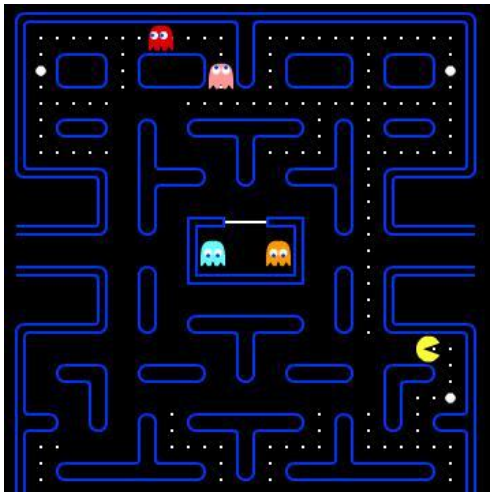


Figura 1 - Jogo PacMan.

Então com a massificação e popularização dos jogos nos anos 80, surgem diversos empresas de consoles, como Master-System, Nes e Atari (IGN Retro), e também uma diversificação dos jogos que começam a incorporar personagens e histórias cada vez mais envolventes. Isto fez com que os jogadores passassem a jogar com mais seriedade, e não apenas como um mero passa tempo, elevando ainda mais a indústria dos jogos. Foi com esta evolução que surgiram dois divisores de águas: em 1981 o jogo Mario e Zelda. Mario apresentou um herói que não apenas seria popularizado, mas se tornaria a marca padrão da Nintendo, uma das maiores indústria de jogos. O jogo *Legend of Zelda* em 1986 trouxe aos *gamers* uma história fictícia, porém bem trabalhada e complexa, algo que ainda não se tinha visto na área, pois fazia com que o jogador viajasse para o mundo virtual em sua história.

A partir de então não houve mudanças significativas na indústria nem nos jogos, até o surgimento em 1993 do jogo DOOM, que ficou marcado não apenas pela revolução de trazer um novo estilo de jogo, mas pelas diversas polêmicas que o jogo causou. Como novo estilo de jogo, DOOM trouxe ao público o estilo FPS (*First Person Shooter*), onde o jogador tinha a perspectiva do personagem de jogo e o principal objetivo do jogo era sair atirando em seus inimigos. Socialmente DOOM causou polêmica devido a seu aspecto de jogo violento e sangrento, pois autoridades e pais acreditavam que tal jogo poderia causar má influência nos jovens. Tal fato foi acentuado com a tragédia de Columbine, onde dois jovens armados invadiram a escola onde estudavam, e armados mataram e feriram diversos estudantes, logo após cometendo suicídio. Tal acontecimento foi creditado por algumas autoridades e pessoas ao fato, dentre outros, dos dois jovens jogarem o jogo DOOM. Após este fato surgiu então a classificação etária dos jogos, assim como temos nos filmes, para que as crianças não sofressem más influências.

Com o passar do tempo, as tecnologias evoluíram, e tornou-se possível trabalhar com ambientes tridimensionais, fazendo com que os jogos se tornassem mais realistas. Esta realidade levou os jogos a outros fins que não fossem o de entretenimento, como por exemplo, simuladores.

Os simuladores permitiam treinar pilotos e outros profissionais em um ambiente virtual mais barato e seguro, antes que eles realmente entrassem em ação no mundo real. Mas, com os jogos se tornando cada vez mais realistas, tem-se a preocupação de como separar o *videogame* da realidade. Não podemos negar que com a evolução tecnológica melhorando a qualidade gráfica dos jogos e com cada vez mais profissionais na área tornando os jogos mais atrativos para os jogadores, os jogos eletrônicos se tornam uma mídia cada vez mais poderosa.

Na última década ocorreram duas grandes mudanças: a massificação de jogos online, os quais abriram diversas portas para a interação entre jogadores que antes estavam limitados a disputas somente dentro de sua casa e também a novos tipos de negócios, pois agora itens de jogos podiam ser vendidos e comercializados neste mundo cibernético, e também uma mudança gradativa na forma de interagir com os consoles.

Esta mudança vem do fato da popularização de dispositivos (vide Fig. 2) que se baseiam na movimentação do jogador para o controle dos personagens no videogame, ou seja, o jogador não mais deve apenas apertar botões para controlar o jogo, ele agora irá trabalhar o corpo inteiro para interagir com o videogame.

(a) (b) (c)

Figura 2 - Dispositivos de Interação baseados na movimentação do corpo do jogador. Topo - Kinect do Microsoft Xbox 360. em (a) e (b) PS Move do Play Station 3 e em (c) o Nintendo Wii Move.

2.1.2 Indústria dos Jogos Eletrônicos

Como vimos, os jogos sofreram grandes transformações desde o Tênis para dois, até jogos mais recentes como *Assassin's Creed*. As mudanças que nos referimos não são apenas em relação à evolução nos gráficos, mas estão também relacionadas ao poder dos jogos dentro da sociedade. Uma forma de vermos isto é analisando os números da indústria dos jogos eletrônicos.

A indústria mundial dos jogos eletrônicos alcançou em 2006 um nível que chegou ao mesmo patamar da indústria do cinema, neste ano como podemos ver na reportagem da redação UOL sobre a Eletronic Arts Games (UOL, 2010), a indústria dos jogos chegou a uma arrecadação de US$ 9,66 bilhões, enquanto a indústria do cinema arrecadou US$ 9,5 bilhões no mesmo ano, porém enquanto o crescimento do cinema foi de 1,9% de 2005 para 2006, a indústria dos jogos teve um crescimento de 28,4% no mesmo período.

Números mais recentes vistos em (ESA, 2010) mostram que a indústria dos jogos alcançou apenas nos EUA a arrecadação de US$ 10,5 bilhões, apesar da pirataria, que é uma dos grandes problemas enfrentados por esta indústria.

No Brasil, muito mais que nos EUA, como visto em (BSA, 2010), a indústria dos jogos é afetada pela pirataria, alcançando 56% de seu mercado interno de softwares em produtos piratas. Um sinal disto é o fato do Brasil representar apenas 0,16% do faturamento mundial com jogos eletrônicos, segundo pesquisa da ABRA (Associação Brasileira das Desenvolvedoras de Jogos Eletrônicos) (ABRAGAMES, 2008). Porém aos poucos isto vem mudando, pois entre 2007 e 2008 a indústria de jogos teve um crescimento de 31% segundo a mesma pesquisa.

Figura 3 - Evolução da audiência nas TVs entre 2005 – 2009. Fonte: (UOL, 2011)

Na Fig. 3 é mostrado que desde 2005 aparelhos como DVDs e consoles de jogos vêm ganhando espaço na audiência brasileira, com um aumento de mais de 100% em cinco anos, superando em horário nobre (18h-0h) grandes emissoras como Bandeirantes e RedeTV!.

Este aumento de audiência mostra que o público brasileiro começa a procurar conteúdos diferentes dos proporcionados pelas emissoras de TV, como jogos através dos consoles, shows e filmes através de DVD players, Bluerays e outros.

2.1.3 Tipos de Jogos Eletrônicos

A classificação mais ampla dos tipos de jogos separa-os em jogos para entretenimento, jogos sérios (*serious games*) e simuladores. Em (Narayanasamy, 2006) é dado o foco da diferença entre estes três tipos de jogos na forma como são desenvolvidos, ou seja, a diferença esta no foco de seu desenvolvimento, onde jogos de entretenimento seriam jogos desenvolvidos para a diversão do usuário, jogos sérios para aprendizagem de algo especifico e simuladores para o treinamento. Porém em (Johnston e Whitehead, 2009) é apresentada uma proposta um pouco diferente e mais interessante entre estas classificações. É proposto que estes jogos não se diferenciam em sua produção, mas sim pelos jogadores que o jogam. Uma forma de vermos isto é pensarmos em um simulador de vôo. Se colocarmos uma criança nele, esta irá se

divertir e considerar aquilo como um jogo de entretenimento, enquanto um piloto profissional o consideraria um simulador.

Jogos sérios não possuem como foco o entretenimento do jogador. Este tipo de jogo objetiva um meio para promover atividades como educação, saúde, políticas governamentais, publicidade, dentre outros.

Na publicidade, os jogos recebem o nome de *advergames*. Estes jogos são produzidos exclusivamente para a divulgação de produtos, como o jogo T-Racer (FIAT, 2010) promovido pela FIAT para promoção de um de seus carros, ou até mesmo a promoção de outro jogo, como no caso de *Mirro's Edge* (Ubisoft, 2010), que teve um *advergame* do jogo lançado antes do lançamento do jogo real.

Nas demais áreas se destacam projetos como o "*Games for Change"* (Games For Change, 2010) e "*Games for Health*" (Games For Health, 2010), que têm respectivamente o intuito de conscientização das pessoas para os problemas globais e para os problemas de saúde humanitária. No "*Games for Change"* são disponibilizados jogos para conscientização, e também dicas e oportunidades de projeto para pessoas que se interessem em participar destas mudanças.

Jogos de entretenimento são os mais comuns e populares. São eles os responsáveis pela movimentação de bilhões de dólares no mercado dos jogos e também por diversas polêmicas que já foram citadas anteriormente. Algumas destas polêmicas levaram a criação de um sistema de recomendação dos jogos por idade, o *Entertainment Software Rating Board* (ESRB, 2009), que classifica os jogos em seis categorias:

- Infantil: têm conteúdo que pode ser adequado para as idades de três anos e mais velhos. Não contém nenhum material que os pais possam considerar impróprio;
- Todos: tem conteúdo que pode ser adequado para as idades de seis anos e mais velhos. Títulos nesta categoria podem conter um mínimo de violência moderada e uso frequente de linguagem moderada;
- Maiores de 10: têm conteúdo que pode ser adequados para as idades de dez anos e mais velhos. Títulos nesta categoria podem conter mais violência e linguagem moderadas;
- Jovens: têm conteúdo que pode ser adequado para as idades de treze anos e mais velhos. Títulos nesta categoria podem conter violência, temas sugestivos, humor bruto, sangue mínimo, apostas simuladas, e uso frequente de linguagem forte;

- Madura: têm conteúdo que é adequado para pessoas com idades de dezessete anos e mais velhas. Títulos nesta categoria podem conter violência intensa, sangue, conteúdo sexual e linguagem forte;
- Apenas para adultos: têm conteúdo que só deve ser jogado por pessoas maiores de dezoito anos. Títulos nesta categoria podem incluir cenas prolongadas de intensa violência e conteúdo gráfico sexual e nudez.

O processo de categorização dos jogos é feito da seguinte forma: durante a fase final de desenvolvimento do jogo, os desenvolvedores submetem um questionário sobre o jogo à *Entertainment Software Ranting Board* (ESRB) especificando exatamente quais tipos de conteúdo estarão contidos no jogo, junto com uma demonstração do jogo para analise pela própria instituição, incluindo nesta demonstração: jogabilidade, missões, linguagem utilizada, conteúdos extras, e outros. Após o envio, a equipe da ESBR analisa o material e repassa tal para especialistas que irão analisá-los novamente, comparar com outros jogos e ao final entram em consenso sobre a categoria do jogo, mandando o resultado a ESRB, que ira revisar a decisão. Ao final a produtora do jogo é notificada da classificação e antes de disponibilizar o jogo a venda, envia uma copia do jogo em formato final para ESRB para uma verificação com o que foi anteriormente dito.

Outra forma comum de classificação de jogos de entretenimento é relacionada ao gênero do jogo, ou seja, as características de jogabilidade do jogo. Os mais comuns são:

- Ação: é um gênero de jogos que enfatiza desafios físicos, incluindo a coordenação olho-mão e reação em tempo integral. O gênero inclui diversos subgêneros, tais como jogos de luta, jogos de tiro e jogos de plataforma.
- Arcade: são em geral aqueles que têm a ação definida por comandos simples e evolução linear do cenário, como Sonic, Mario e Castlevania.
- Aventura: é um jogo no qual o jogador assume o papel de um protagonista de uma história interativa que é impulsionado pela exploração e quebra-cabeças em vez de desafios físicos, tais como combate.
- Corrida: é um gênero, seja na perspectiva de primeira pessoa ou de terceiros, no qual o jogador participa de uma competição de corrida com qualquer tipo de ambiente (terra, ar ou mar) em qualquer tipo de veículo.
- Esporte: são jogos onde o jogador participa de uma competição baseada em alguma atividade esportiva do mundo real.

- Estratégia: estilo de jogo em que a habilidade de tomada de decisão dos jogadores tem um significado elevado em determinar o resultado.
- *Role Playing Game* (RPG): é um gênero que utiliza elementos de jogabilidade encontrada em jogos RPG de mesa (classes de personagem, níveis).

Cada um destes gêneros pode ser estendido a partir de conexões Web, podendo ser apenas jogos onde o jogador se conecta a alguns outros para disputas especificas de um jogo, ou cooperação entre eles, como acontecem nos jogos *Need for Speed Most Wanted* e *Splinter Cell Double Agent*, até jogos com suas experiências totalmente voltadas para a competição entre jogadores na internet, como o caso de *World of Warcraft.*

Simuladores são programas que replicam artificialmente as condições reais normalmente encontradas em um tipo de operação. Servem basicamente para treinamento, pois em muitos casos o uso do ambiente real para tal se torna muito caro ou arriscado.

Os simuladores também são utilizados em diversas áreas, como aeronáutica, construção civil, para treinamento de operadores de guindastes, e também no exército, onde os soldados podem participar de guerras virtuais.

2.1.4 Desenvolvimento de Jogos Eletrônicos

Um jogo de videogame se difere dos demais softwares pela sua etapa de pré-produção e pela integração intensiva de seus diversos recursos multimídia (gráficos, texturas, áudio, modelos 3d, etc.) (Kanode e Haddad, 2009). A fase de pré-produção é semelhante à fase de especificação de requisitos a partir do cliente, exceto que o cliente (*game designer*) por sua vez cria constantemente protótipos do jogo estabelecendo novas visões dele.

As principais etapas do desenvolvimento de um jogo são pré-produção, produção e testes (Kanode e Haddad, 2009). Como já dito, na pré-produção acontece a concepção do jogo, tendo como resultado o documento de especificação dele, o chamado GDD (*Game Development Document*). Ao decorrer do desenvolvimento do jogo este documento pode ser atualizado de acordo com as limitações encontradas e protótipos testados.

Na fase de produção ocorre a geração dos conteúdos, incluindo a codificação. Esta fase esta mais propicia a problemas, como atrasos, problemas na interpretação dos requisitos, objetivos não alcançados e outros. Nesta fase os desenvolvedores criam protótipos interativos

para que a experiência do jogador com o jogo possa ser testada, podendo ocorrer mudanças bruscas no projeto caso a ideia inicial do GDD não seja satisfatória, podendo afetar recursos do projeto.

A fase de testes é a última antes do lançamento oficial do jogo. Nela ocorrem testes de qualidade do jogo, testes para encontrar possíveis erros de desenvolvimento e também testes de stress, onde os jogos são colocados a seus limites de configurações em busca de problemas.

Estas são as fases genéricas na produção de um jogo, de forma que diferentes metodologias podem ser utilizadas em cada uma dessas fases, ou até adaptar metodologias para se encaixarem aos pontos específicos do desenvolvimento de jogos.

Exemplos podem ser encontrados em um dos maiores sites de notícias da indústria dos jogos, Gamasutra.com (Gamasutra, 2011). Neste site são encontrados casos de desenvolvimento de jogos utilizando diferentes metodologias, como o caso da Crystal Dynamics, criadora do jogo *Tomb Raider: Anniversary*, que diz ter utilizado metodologias ágeis no desenvolvimento do jogo. Porém, por serem empresas com fins lucrativos, não há divulgação aberta das metodologias utilizadas, e quando isto ocorre, faltam detalhes.

Do ponto de vista dos profissionais envolvidos (*stakeholders*) os jogos evoluíram muito, sendo necessários diferentes perfis de profissionais (International Game Developers Association, 2010) para abarcar todas as habilidades necessárias para a produção de um jogo.

Os *designers* são responsáveis pelo conceito do jogo, desde a concepção de idéias até a representação do jogo no mercado. Entre os designers, existe o game designer, que se preocupa com todos detalhes do jogo, temos o designer de níveis, que cria as fases e estruturas do jogo, e também roteiristas e escritores que trabalham para reunir a forma e o conteúdo do jogo da melhor maneira possível.

Os programadores lidam com os códigos e por isso são necessários em muitas das etapas da produção de um jogo.

A equipe de arte se preocupa com o visual do jogo, que é um dos elementos chave para o sucesso ou fracasso do mesmo. O visual nasce dos rabiscos de ilustradores, da criação de imagens conceituais e da pesquisa para adequar o visual a essência do jogo. Enquanto

artistas fazem as animações de uma cena narrativa, modeladores desenvolvem modelos que o jogador utilizará durante o jogo.

A equipe de som cuida dos efeitos sonoros nos jogos, os quais evoluíram de simples efeitos em chips eletrônicos para trilhas sonoras gravadas por orquestras. Grandes produções utilizam especialistas para captura de efeitos sonoros realistas, atores para dublagem de personagens e compositores próprios.

Por fim, os produtores são os líderes do projeto, coordenando as equipes e administrando o desenvolvimento do jogo. Na produção há o apoio de equipes de controle de qualidade, marketing, vendas e outros executivos envolvidos no projeto.

2.2. TV Digital

A TV Digital é uma plataforma pouco explorada para desenvolvimento de softwares no Brasil, encontrando assim diversos desafios a serem superados e diversos conceitos a serem definidos. Por esta razão, será feita uma revisão sobre o que é TV Digital, seus desafios e possibilidades.

2.2.1 Televisão Digital Interativa

Segundo Parker (1999), a TV Digital Interativa (TVDI) é algo mais do que apenas melhor qualidade de som e imagem, ela é uma convergência entre televisão, telefonia, Internet e computador pessoal através de uma simples caixa, o *set-top-box* (STB), com a promessa de acesso extraordinário a todos os tipos de informação e comunicação interativa inimagináveis. TVD se refere à transmissão de sinais em uma forma digital.

A transmissão digital permite mesclar conteúdos de áudio e vídeo com dados binários. Assim, é possível entregar aplicações de software em um aparelho digital de TV ou em um STB. O set-top-box é responsável por receber o sinal digital usando um sintonizador digital e então convertê-lo para um formato analógico para ser visto em um aparelho analógico de TV, ou pode já estar embutido na TV em modelos mais novos.

A TV Digital ganhou destaque nos últimos anos no Brasil, principalmente após a criação o padrão nacional baseado em alguns parâmetros do padrão japonês (ISDB-T), chamado de Ginga®. Essa padronização nacional é formada por um conjunto de tecnologias e inovações brasileiras que tornam a especificação de middleware mais avançada e a melhor solução para os requisitos do país. Para tornar mais eficiente o desenvolvimento de aplicações, bem como reduzir os custos associados, favorecendo assim a consolidação da TVDI, os fabricantes e provedores de conteúdo perceberam que a solução é adotar mecanismos que tornem portáveis as aplicações e os serviços nos diversos tipos de STBs. Um aspecto importante que deve ser considerado na interatividade, diz respeito à interação humano computador. Antes do surgimento da TV Digital a concepção de interação humano computador já se fazia presente através de Weiser (1991), que afirmou estar vivendo a era da ubiqüidade, onde os computadores estão em todos os lugares, inseridos intrinsecamente ao cotidiano humano. Desta forma, podemos ver que a TVDi se enquadra neste conceito, pois adiciona a uma atividade cotidiana (assistir TV) mais uma plataforma com as características de um computador.

A interatividade televisiva, com o advento da TV Digital Interativa, está passando por um processo de evolução. Lemos e Elias (2004) classificam a interatividade em níveis de interação, que vão desde ligar e desligar a TV, fazer a troca de canais pelo controle remoto, usar videocassete e videogame, opinar a respeito de um conteúdo televisivo por telefone ou correio, até entrar em um primeiro estágio de TV interativa, que permite escolher ângulos de câmeras e navegar pelas informações. Montez e Becker (2005) estendem essa definição, propondo novos níveis de interatividade nos quais o usuário pode enviar seu próprio conteúdo, chegando a um estágio similar ao que ocorre na Internet hoje, onde qualquer pessoa pode ter seu próprio website e até fazer a difusão de seu próprio conteúdo audiovisual. Gawlinski (2003) define o que chamamos de TV interativa como algo que permite o estabelecimento de um diálogo entre o usuário com o programa ou serviço.

Segundo Peng (2002), existem basicamente três níveis de interatividade: interatividade local, interatividade de uma via e interatividade de duas vias.

Na interatividade local há o suporte a aplicações onde os telespectadores interagem apenas com a aplicação que esta sendo executada no STB. Não há comunicação entre o STB e servidores de dados. Os códigos e informações para execução destas aplicações são baixados da transmissão em broadcast dos sinais de TV da emissora, armazenadas e

montadas na memória do STB e então executadas. Exemplos são a EPG (*Electronic Programming Guide), Closed Caption*, jogos locais, dentre outros.

Com a interatividade de uma via temos a necessidade de um canal de retorno para comunicação entre a aplicação que esta executando no STB e um servidor para o processamento dos dados colhidos pela aplicação. Neste nível temos apenas informações sendo enviadas do telespectador para o servidor. Exemplos são pesquisas e votações.

Interatividade de duas vias corresponde às aplicações que podem enviar informações aos servidores, porém também podem individualmente baixar novas informações a partir do canal de retorno. Exemplos são: acesso a email, T-Banking, jogos interativos multiplayers, dentre outros.

2.2.2 Programas Interativos

EPGs, *Enhanced TV*, conteúdo sobre demanda, TV personalizada, Internet na TV, jogos e apostas, publicidade interativa e t-commerce são oito categorias que Jensen (2005) classifica como os tipos de aplicações para TV.

EPG é a Interface gráfica que possibilita a navegação pelas várias possibilidades de programação que o usuário encontrará na TV Digital, sendo o equivalente aos guias de horários de televisão publicados nos jornais, com funções e operação análoga a de um portal de internet, contendo informações como horários de inicio e fim de um programa, recomendação etária e sinopse.

Enhanced TV é, de forma simplificada, conteúdo extra, normalmente na forma de texto e imagens, que são sobrepostos aos vídeos que estão sendo exibidos, passando ao telespectador informações adicionais sobre o programa que esta sendo exibido, sendo este conteúdo alcançado pelo usuário de forma interativa através das aplicações interativas. Exemplos comuns incluem estatísticas em eventos esportivos, informações sobre personagens e artistas de TV. Os conteúdos podem ser independentes da programação como serviços de notícias, email e catálogos de venda.

Conteúdo sobre demanda se refere a conteúdos que você pode obter através de serviços disponibilizados, como vídeos sob demanda, música e jogos, pagando taxas para poder obter tais serviços.

TV personalizada pode ser vista como formas de gerir informação. Temos então dois tipos: personalização e customização. Personalização é feito pelo sistema de mídias, onde este envia informações individualizadas ao usuário. Já customização é dirigida pelo usuário, ou seja, ele seleciona opções e obtém conteúdos personalizados a partir daquelas escolhas.

Internet na TV permite aos usuários realizar muitas das atividades normalmente desempenhadas em um computador conectado à Internet através do aparelho de televisão, por exemplo, ler e escrever e-mails e mensagens instantâneas, participar de bate-papos e discussões de grupos, navegarem na web ou fazer pesquisas na internet.

Publicidade interativa é a comunicação de marketing e de mensagens de marcas utilizando a interatividade da transmissão digital, permitindo ao usuário solicitar mais informações sobre produtos apresentados, e também em alguns casos permite que o espectador compre o produto diretamente do anúncio. Neste aspecto a publicidade junta fronteiras com T-commerce.

T-commerce é simplesmente o fenômeno do e-commerce conhecido na Internet e transferido para o meio televisivo. Enquanto a televisão tradicional cria conteúdos criativos, colocando anúncios entre os segmentos, a televisão interativa com T-commerce gera conteúdos também criativos, apoiando as vendas durante o conteúdo. Por outras palavras, T-commerce permite ao telespectador comprar produtos e serviços que ele vê na tela da televisão. Desta forma, os telespectadores se tornam consumidores à medida que eles fazem transações através da televisão, que serão realizadas de diversas maneiras.

Jogos possuem na TV diversas possibilidades, tanto na forma de interação com o conteúdo quanto à forma de distribuição do mesmo. O conteúdo de um jogo pode estar relacionado ou não ao programa de TV que esta sendo exibido, ou até pode não haver programas sendo exibidos, sendo o jogo o foco de entretenimento naquele momento. Na forma de distribuição, os jogos podem ser enviados via broadcasting pela emissora, podem ser baixados pela internet ou ate por serviços Pay-per-play, onde o jogador paga pelos jogos que queira jogar na TV. Veremos mais detalhes sobre este tipo de aplicação nas próximas páginas.

2.2.3 Sistema Brasileiro de Televisão Digital

A principal inovação do Sistema Brasileiro de TV Digital Terrestre (SBTVD-T) é o seu middleware de referência, denominado Ginga. Além de ser compatível com diversos dispositivos e suportar vários protocolos de comunicação, o Ginga oferece suporte ao desenvolvimento de aplicações de diferentes paradigmas de programação, declarativas e imperativas. Assim, a arquitetura do Ginga é subdividida em três subsistemas: o GingaNCL, o Ginga-J e o Ginga-CC.

O GingaNCL, inovação completamente brasileira, desenvolvido pela PUC - Rio, é o subsistema lógico responsável pelo processamento de aplicações declarativas NCL (*Nested Context Language*). NCL (Soares & Barbosa, 2009) é uma linguagem de aplicação XML que separa conteúdo e estrutura, facilitando o suporte a múltiplos dispositivos, a interatividade, sincronismo espaço-temporal, etc. Além desta linguagem também é usada a linguagem de script Lua (Ierusalimschy, 2006), para quando for necessária a geração dinâmica de conteúdo.

O Ginga-J é composto por um conjunto de APIs definidas para atender todas as funcionalidades necessárias para o desenvolvimento de aplicativos para TVD, desde a manipulação de dados multimídia até protocolos de acesso. Sua especificação é formada por uma adaptação da API de acesso a informação de serviço do padrão japonês, pela especificação Java DTV, além de um conjunto de APIs adicionais de extensão ou inovação. As APIs adicionais incluem um conjunto de classes disponíveis para a ponte entre os aplicativos escritos nas linguagens NCL e Java, funcionalidades adicionais para sintonia de canais, envio de mensagens assíncronas pelo canal de interatividade e integração de dispositivos externos ao middleware, viabilizando o suporte a recursos multimídia e interação simultânea de múltiplos usuários em aplicações de TVD .

O Ginga-CC, *Ginga Common Core*, é o núcleo comum do middleware com suporte aos dois subsistemas específicos. Este subsistema depende da plataforma na qual está embarcada, fornecendo abstração do hardware e do sistema operacional ao Ginga-J e ao Ginga-NCL, através de APIs. Assim o Ginga-CC é responsável pelos exibidores de áudio, vídeo, imagem e texto, pelos dados obtidos através do canal de interatividade, ou retorno, o gerenciador de dados e outros componentes que dependem da implementação de cada receptor.

A Fig. 4 mostra a estruturada da arquitetura do middleware Ginga.

Figura 4 - Arquitetura do Ginga. Fonte: (Lemos et. al, 2005)

Atualmente o Sistema Brasileiro de TV Digital esta sendo adotado em outros países (vide Fig. 5) da América Latina (DTV.org, 2011). Desta forma o Ginga esta ganhando âmbito internacional, sendo assim acessado por mais telespectadores e aumentando o numero de interessados na plataforma.

Como estes países estão adotando o SBTVD, todas as aplicações desenvolvidas no Ginga serão compatíveis com a TV Digital de todos estes países, sendo assim, uma ferramenta de auxilio ao desenvolvimento de aplicações poderá ser utilizada por programadores em todos estes países.

Figura 5 - Países que adotaram o padrão ISDB-Tb – Fonte: (Mapa ISDB-Tb, 2010)

2.3 Possibilidades de Jogos na TV Digital

A programação de aplicações para TV Digital possui diversos desafios relacionados a questões de interface que devem considerar a TV como saída das aplicações e questões de sobreposição do vídeo que é enviado junto à aplicação. Desafios relacionados ao público alvo das aplicações produzidas, pois como a distribuição destas aplicações é feita em broadcast, ou seja, todos os telespectadores sintonizados na emissora de TV irão receber a mesma aplicação, mesmo possuindo perfis muito diferentes. E por fim, desafios de Interação Humano Computador (IHC), tanto no aspecto de usabilidade dos dispositivos de interação (controle-remoto), quanto desafios de aceitabilidade das aplicações.

Tais desafios podem limitar as possibilidades de jogos executáveis na plataforma, porém, podemos levantar algumas possibilidades que deverão ser opções para os desenvolvedores de jogos.

A primeira possibilidade é o uso da plataforma como um console de jogos, onde o jogador pode adquirir o jogo através do canal de retorno conectado a internet e executá-lo em sua TV, ou até mesmo jogar jogos enviados pela emissora que não sejam relacionados à grade de programação.

Outro modelo possível é o surgimento de canais exclusivos de jogos. Nestes canais não há uma programação tradicional, em vez de programas de TV são enviados pela emissora apenas jogos, em horários definidos, como se fossem a programação do canal.

Os dois modelos apresentados acima não se utilizam de duas das principais características da TV Digital: som e imagem de alta qualidade. Sendo assim, outra forma de explorar os jogos na TV é correlacionando-os com os programas que estão sendo exibidos. Neste contexto, o maior desafio esta na concepção de como seriam tais jogos, pois nenhuma das plataformas de jogos tradicionais (PC, consoles, celular) possui esta característica, de forma que os *GameDesigners* que são responsáveis por idealizar os jogos têm que se esforçar para elaborar jogos voltados a essa nova plataforma.

Outra possibilidade é a utilização da TV como terminal para jogos nas nuvens (*GameClouding*). Neste cenário o motor (*engine*) do jogo não mais executa no cliente, mas sim na nuvem, onde é realizada a maior parte dos processos mais intensivos do ponto de vista

computacional. O cliente do jogo resume-se ao envio de eventos para a nuvem e o recebimento de um *stream* de vídeo. Os eventos recebidos dos clientes, juntamente com o estado do mundo, permitem a renderização da aparência do ambiente virtual também na nuvem, e as imagens resultantes são enviadas para os clientes via *streaming*. Desta forma, o jogo possui uma dependência muito menor do dispositivo cliente específico do usuário.

Capítulo

3

Trabalhos Correlatos

No desenvolvimento de aplicações, um framework é um conjunto cooperativo de classes que compõem um projeto reutilizável para um domínio específico de softwares. O framework fornece orientação arquitetural através da divisão do projeto em classes abstratas e definindo suas responsabilidades e colaborações. Um desenvolvedor customiza o framework para uma aplicação específica utilizando instancias das classes do framework (Gamma et. al., 1994).

Isto contrasta com a forma tradicional de desenvolvimento de aplicações, onde cada aplicação é desenvolvida a partir do zero. A maior diferença entre o desenvolvimento tradicional e o desenvolvimento baseado em frameworks é a necessidade de mapear a estrutura do problema a ser resolvido na estrutura do framework, forçando a aplicação a reutilizar o design do framework. O lado positivo disto é que não precisamos projetar o aplicativo desde o principio, reaproveitando arquitetura, código, e acelerando o processo de desenvolvimento (Flores, 2008).

Segundo (Pessoa, 2002) um framework voltado para o âmbito de jogos deve satisfazer alguns requisitos, onde se destacam:

- Arquitetura modular, para que possa ser utilizado em cada novo jogo;
- Bom nível de abstração, ao prover objetos com funcionalidades já implementadas;
- Definição de objetos básicos, como o objeto do jogo e o mapa (cenário) do jogo;
- Detecção e gerenciamento de eventos gerados por teclado, mouse, joysticks e outros;
- Algoritmos para desenho dos objetos do jogo, podendo ser 2D e 3D;
- Funcionalidades úteis a jogos 3D, como iluminação e transformações geométricas;
- Execução dos sons do jogo em resposta a eventos ocorridos;
- Implementação de algoritmos de IA para os objetos inteligentes dos jogos;

- Implementação de algoritmos para troca remota de dados, gerenciamento de sessões e sincronização das informações compartilhadas do jogo;
- Implementação de algoritmos para modelagem física de objetos.

Com os requisitos de frameworks para jogos definidos, e características de frameworks apresentadas, são mostrados a seguir trabalhos que apresentam estas características voltadas para a TV Digital, em especial o Ginga.

3.1 Ferramentas para Desenvolvimento para TV Digital

O middleware Ginga atrai cada vez mais programadores para a área de TV Digital, sendo assim cada vez mais frequente o uso de ferramentas de auxilio ao programador. Na literatura encontramos propostas, tais como o iTVProject (Oliveira, 2008), o FrameIDTV (Henrique et. al., 2010), o framework TUGA (Ferreira e Souza, 2009) e o GingaGame (Barboza e Gonzales, 2009). Nesta seção detalhamos essas propostas.

3.1.1 iTVProject

iTVProject (Oliveira, 2008) é uma ferramenta de autoria web, cujo foco é o rápido desenvolvimento de aplicações interativas para TV Digital. Ferramentas de autoria devem possuir uma interface amigável e simples para possibilitar o desenvolvimento de aplicações através de uma interface gráfica, o que dispensa o conhecimento técnico em programação.

A proposta do iTVProject é a criação de um ambiente Web para o desenvolvimento de aplicativos interativos para televisão digital, executável em ambos midllewares das normas européias e brasileiras - DVBMHP e Ginga-J, respectivamente. Para o desenvolvimento do projeto, foram utilizadas tecnologias para a web, como JEE1, AJAX2, JavaScript e JSON3. Assim, o ambiente e projetos criados a partir iTVProject são acessíveis de qualquer computador com um navegador e uma conexão para Internet.

A ferramenta iTVProject utiliza o framework TUIG (Oliveira 2008). Os recursos fornecidos pelo TUIG foram escolhidos a partir da análise de estruturas para o desenvolvimento de aplicações para ambientes de desktop, e sua conseqüente adaptação às limitações do ambiente da TV Digital. No framework, é possível gerar interfaces gráficas

dinamicamente através de um arquivo XML que define os componentes que estarão presentes na tela. Este arquivo pode definir os atributos dos componentes, tais como gráficos, cor e tamanho e eventos a serem lançados a partir do gatilho de um determinado componente. Através do processamento do arquivo XML, as classes responsáveis por gerar as interfaces gráficas são geradas automaticamente. Porém, interfaces mais complexas e programas maiores geram também arquivos XML muito mais complexos, tornando difícil a manutenção do código, problema similar ocorre com a linguagem NCL. Outro problema apresentado pelo framework é a dificuldade de adaptação de profissionais de outras áreas, como designers e produtores de conteúdos, no uso da criação dos programas através da linguagem XML.

O Projeto iTV entra então para facilitar a criação destes arquivos XML. O iTV atua apenas como um *front-end* para o usuário. O produto final do iTV é um arquivo XML compatível com o framework TUIG. Assim, o iTV Project gera apenas o XMLs, mas o framework TUIG ainda é responsável por interpretá-los e gerar as classes Java correspondentes às aplicações interativas.

3.1.2 FrameIDTV

O FrameIDTV (Henrique et. al., 2010) se propõe a dar suporte aos principais requisitos das aplicações para TV Digital Interativa. Para atingir esse objetivo, o framework é baseado em orientação a objetos (OO), de forma que o design do framework forneça um nível adequado de abstração ao programador final.

Para a criação do Framework, foi utilizada a metodologia encontrada em Johnson (1997), que estabelece a necessidade de compreender o domínio da aplicação através da observação de exemplos concretos. Os exemplos utilizados pelo FrameIDTV foram aplicações de: T- Government para troca de informações entre o telespectador e o servidor de informações, uma aplicação para envio de emails a partir da TV, uma EPG e um portal onde o usuário pode navegar entre as diversas aplicações acima citadas.

A arquitetura proposta pelo FrameIDTV usa, na sua organização, a estrutura de pacotes que agrupa classes relacionadas a um escopo comum. A Fig. 6 ilustra a distribuição dos pacotes em camadas de acordo com o padrão MVC.

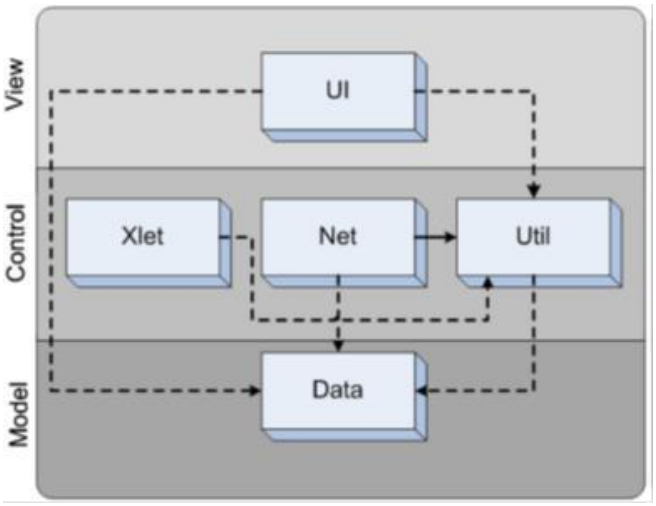


Figura 6 Arquitetura do FrameIDTV. (Henrique et. al., 2010)

Pacote UI (user interface): está apresentado na camada de visão. Este pacote contém as classes responsáveis pela montagem dos elementos que compõem a interface gráfica (GUI) do aplicativo. Além disso, um grupo de componentes gráficos faz parte deste pacote, como ícones e caixas de texto, dentre outros.

Pacote Xlet: neste pacote, as classes que compõem as aplicações Xlet são agrupados. Xlets são semelhantes aos applets Java, porém são implementados especificamente para STBs da TV Digital.

Pacote Util: contém várias classes que ajudam as aplicações, melhorando a legibilidade do código. Este pacote é composto por classes, como conversores de dados, gerenciador de imagens, elementos controle das interfaces de entrada de dados, controle de exceção, etc.

Pacote Net: este pacote contém as interfaces e classes responsáveis para lidar com a comunicação entre aplicações cliente e servidor. Entre as funcionalidades implementadas neste pacote estão o acesso remoto através do canal de retorno, gerenciamento de status de conexões TCP / IP, e os métodos para o estabelecimento de conexões seguras.

Pacote de Dados: é, de acordo com o paradigma MVC, a camada de modelo. Esta camada compreende a infraestrutura de armazenamento e manipulação de dados dos aplicativos.

O FrameIDTV apresenta diversos resultados, como aplicações de T-commerce, T-Banking, envio de mensagens SMS e aplicações esportivas, porém não apresenta nenhuma

implementação relacionada a jogos, apesar de afirmar que "jogos 2D simples são satisfeitos pelo FrameIDTV" (Henrique et. al., 2010).

3.2 Frameworks para Jogos na TV Digital

Na literatura (Ferreira e Souza, 2009) (Barboza e Gonzales, 2009) também podemos encontrar frameworks específicos para o desenvolvimento de jogos para a TV Digital, em particular, voltados ao SBTVD.

3.2.1 TUGA

O middleware TUGA (Ferreira e Souza, 2009) propõe dar suporte ao desenvolvimento de jogos em TVDI. Esse middleware, além da estrutura de execução dos jogos, disponibiliza também uma API de desenvolvimento de aplicações para a TV digital. O middleware oferece suporte aos seguintes sistemas: (i) Sistema Gráfico; (ii) Sistema Sonoro; e (iii) Sistema de Entrada.

O sistema gráfico é o responsável por permitir a integração com o vídeo, ou seja, por permitir que as imagens e as primitivas gráficas sejam apresentadas no monitor de TV/PC, dando um *feedback* visual ao jogador.

O sistema sonoro é o responsável por permitir o efeito de imersão, com o suporte a uma boa ambientação sonora durante a atividade interativa imersiva.

O sistema de entrada é o responsável por permitir a interação do jogador com o dispositivo, dando assim inicio ao processo de interação.

Além das funcionalidades iniciais apresentadas, o middleware possui uma camada auxiliar, o *framework* GBF. Tv, que fornece os seguintes recursos essenciais para auxiliar em alto nível a criação de jogos: Gerenciador de *Sprites*; Gerenciador de Fontes; Gerenciador de Personagens; Gerenciador de Sons; Gerenciador de Interface Gráfica com Usuário; Gerenciador de Tempo. A arquitetura de todos componentes do TUGA pode ser vista na Fig. 7.

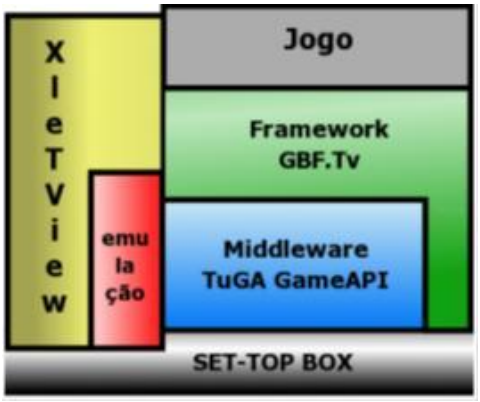


Figura 7- Arquitetura do Middleware TuGA. Fonte: (Ferreira e Souza, 2009)

3.2.2 GingaGame

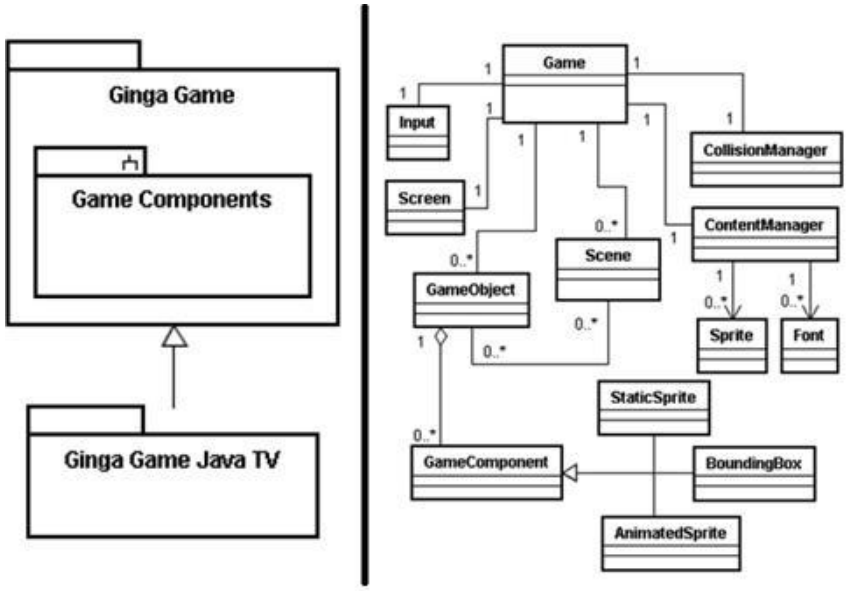
GingaGame (Barboza e Gonzales, 2009) é um framework de desenvolvimento de jogos para a TV Digital. Seu objetivo é fornecer uma estrutura que torna mais fácil o desenvolvimento de jogos para a TV Digital e torna essa tarefa mais semelhante ao desenvolvimento de jogos para computadores pessoais.

Este framework é subdividido em três diferentes pacotes Java (vide Fig. 8), de forma que as classes que precisam de recursos específicos da plataforma estão em um pacote separado das classes que não têm este tipo de dependência. Assim, a migração do GingaGame para outra plataforma pode ser feita apenas fazendo as modificações necessárias em pacotes específicos, enquanto os outros permanecem inalterados.

O pacote GingaGame fornece algumas interfaces abstratas que devem ser implementados em uma plataforma específica. Neste pacote são definidos os conceitos básicos do framework, como objetos e componentes do jogo, cenas e o modelo de aplicativo que gerencia esses objetos.

O GingaGame. GameComponent possui um conjunto de componentes prontos para uso. Esses 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 controle de colisão usando retângulos).

O último pacote, GingaGameJavaTV, inclui todas as classes específicas da plataforma. Um exemplo de recurso específico da plataforma é o gerenciador de janelas. JavaTV usa a classe HScene (do pacote HAVi) para acessar a janela do aplicativo. Essas classes são separadas das demais classes do framework para tornar fácil a mudança de plataforma de execução do jogo quando necessário. Dessa forma, somente o código específico da plataforma deve ser modificado, mas as interfaces permanecem os mesmos.



(a) (b)

Figura 8 - Diagrama UML do GingaGame. Em (a) Diagrama de Pacotes e em (b) Diagrama de Classes. Fonte: (Barboza e Gonzales, 2009).

3.3 Análise Comparativa

Tendo em vista os trabalhos relacionados acima citados, foi elaborada a tabela de comparação entre os trabalhos (Tab. 1), cujos parâmetros de comparação são as funcionalidades de um framework de jogos conforme apresentado em (Pessoa, 2002) e os seguintes parâmetros: disponibilização de uma API, Algoritmos de Suporte a Animação, Adequação ao Ginga, Desenvolvimento de jogos como resultado, Internacionalização, Utilização de metodologias de desenvolvimento de frameworks, Técnicas de prevenção de *Flicker*, Suporte a animação via *Sprites*, Suporte a utilização de tiles, Suporte a eventos de sincronização provindos da emissora.

A API é o último artefato gerado por um framework. A API deve apresentar informações suficientes de forma que o framework possa ser utilizado por programadores independentemente de treinamento ou suporte dados pelos programadores do framework.

Os algoritmos de suporte a animação devem ser considerados quando os jogos demandam taxas constantes de FPS (*frames por segundo*) e UPS (*updates por segundo*), de forma a garantir a correta execução do jogo, e evitar problemas na apresentação do mesmo, pois taxas de FPS e UPS oscilantes podem causar desconforto ao jogador.

Adequação ao Ginga: a norma Brasileira de TV Digital define todas chamadas possíveis ao Ginga, de forma que as aplicações não podem utilizar outros artifícios que não estejam normatizados, caso contrário não haveria como garantir o funcionamento de tal aplicação. O mesmo vale para os frameworks, sendo assim é necessário que estes também sigam as normas para garantir compatibilidade com o middleware.

A verificação do framework como ferramenta para desenvolvimento de jogos pode ser feita através do desenvolvimento de jogos com a mesma, sendo assim a apresentação de um jogo como resultado auxilia nesta verificação.

O parâmetro de internacionalização se refere à possibilidade de utilização do framework pelos países que vêm adotando o Ginga. O ponto mais importante deste parâmetro é o fato de que os países que adotaram o Ginga como padrão, fora o Brasil, adotaram apenas o subsistema declarativo, o GingaNCL.

A utilização de metodologias de desenvolvimento de frameworks permite uma melhor avaliação das etapas utilizadas na construção do mesmo, permitindo melhor entendimento do uso do framework e facilitando o entendimento de possíveis problemas encontrados.

Flicker é um problema comum no uso de gráficos na programação. Operações de elementos gráficos que exigem múltiplas operações de pintura podem causar as imagens desenhadas parecerem piscar ou ter uma aparência inaceitável. Para resolver esse problema, podem-se usar diversas técnicas de tratamento de flicker, dentre elas o buffer duplo (double buffering).

O uso da técnica de *sprites* permite a criação de animações com recursos limitados, pois tal técnica utiliza pouco espaço em disco e necessita de baixo processamento.

Um tile é uma imagem que ao ser desenhada diversas vezes, uma ao lado da outra, passa a impressão de uma imagem maior. Esta técnica permite o reaproveitamento de uma pequena imagem para construção de grandes cenários. Um exemplo é visto na Fig. 9.

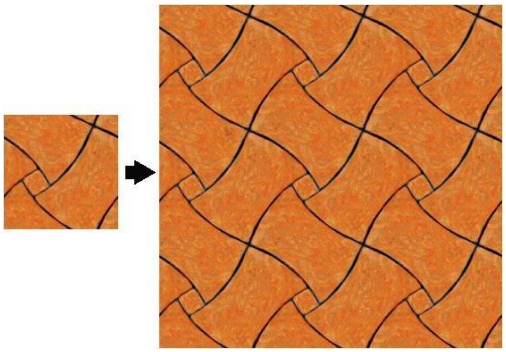


Figura 9 - Exemplo de utilização de um Tile.

O SBTVD permite que a emissora de TV envie eventos de sincronização da emissora para a TV do telespectador. Estes eventos, cobertos na norma (ABNT), permitem não apenas a sincronia de medias entre as aplicações, com isto é possível que a programação interaja com o jogo, enviando eventos e valores enquanto o telespectador joga em sua TV "contra" a programação;

Tabela 1 - Comparativo entre os trabalhos relacionados

Parâmetro	iTVProject	FrameIDTV	TUGA	GingaGame
API	-	-	-	-
Arquitetura Modular	X	X	X	X
Algoritmos de desenho	X	X	X	X
IA	-	-	-	-
Anti-Flicker	-	-	X	-
Cenário de Jogo	-	-	X	-
Dados Remotos	-	X	-	-
Dispositivos de Interação	X	X	X	X
Execução de Áudio	-	-	X	-
Funcionalidades 3D	-	-	-	-
Internacionalização	-	-	-	-
Jogos como resultado	-	-	X	-
Normatização	X	X	-	X
Objetos elementares	-	-	X	X
Sincronização com Emissora	-	-	-	-
Sprites	-	-	X	X
Suporte a Animação	-	-	X	-
Tiles	-	-	X	-
Uso de Metodologias	-	X	-	-

Analisando a Tabela 01, observamos que nenhum dos frameworks analisados leva em consideração a interação direta entre a emissora e o telespectador, de forma que não abordam uma forma de capturar os eventos enviados pela mesma.

Apesar de três dos frameworks seguirem a norma brasileira ou serem projetados de forma a fácil adaptação a mesma, pois ocorreram mudanças no padrão desde suas publicações, nenhum leva em consideração o subsistema GingaNCL para o desenvolvimento das aplicações, de forma a se tornarem incompatíveis com os países que vêm adotando o Ginga como padrão.

Três dos frameworks analisados não levam em consideração o canal de interatividade, por onde as emissoras podem coletar dados dos telespectadores, e por onde existe a possibilidade de interação direta entre os próprios usuários, sem intermédio da emissora.

Não foram apresentados resultados de validação dos frameworks com seus usuários finais, e apenas em um framework houve o desenvolvimento de um game como forma de teste do framework resultante.

Durante a pesquisa apenas os códigos do framework TUGA puderam ser analisados, pois os demais não estavam disponíveis na Internet nem os autores os disponibilizaram até a presente data. Mesmo tendo os códigos do framework em mãos, a ausência de uma documentação de uso e incompatibilidade com a norma dificultaram a realização de testes.

Capítulo

4

ATHUS

A elaboração da arquitetura do framework, definição de suas funcionalidades e desenvolvimento são etapas fundamentais na construção do framework. Este capítulo aborda a metodologia utilizada para realização destas etapas e os resultados obtidos.

4.1. Desenvolvimento de Jogos para TV

A interatividade no mercado nacional ainda esta dando seus primeiros passos, e com isso os jogos para TV aqui ainda estão em fase de estudos. Isto pode ser visto analisando trabalhos como os de Barboza e Gonzales (2009), Souza e Santos (2009) e Ferreira e Souza (2009). O primeiro refere-se à proposta de uma arquitetura de jogos para o Ginga-J, para tal apresenta a construção simples de um quiz usando a arquitetura proposta no emulador do Ginga-J. O segundo também trabalha no mesmo sentido do trabalho anterior, porém encontra-se em um estágio mais avançado, e com uma implementação mais robusta, apresentando como exemplo uma adaptação de um jogo feito para PC em versão para TV. Já o ultimo trabalho apresenta outra perspectiva, ele trabalha sobre a parte declarativa do Ginga, o NCL, porém não apresenta grandes detalhes de arquitetura e implementação, apenas uma revisão sobre jogos e apresentação de resultados, que em relação aos outros são superiores em gráficos e jogabilidade.

Quanto ao âmbito internacional, existem muitos trabalhos e experiências que servem de base para o desenvolvimento de jogos nacionalmente. Artigos como o de Nolan (2002) e Heliö S. e Järvinen (2003) mostram classificações dos jogos para TV, desafios para o desenvolvimento nessa plataforma e também propõe soluções imediatas e futuras para tais problemas.

Nolan mostra os perfis das pessoas que jogam na televisão, classificando-os em Geração I, Early Clickers (jovens clicadores) e Daytime Dabblers. A geração I corresponde aos jovens que jogam um jogo sem se ater a jogá-lo novamente e nem em que canal o jogo esta sendo transmitido. Eles simplesmente navegam entre os jogos assim como navegam entre os canais, procurando por algum entretenimento rápido. Early Clickers correspondem às crianças entre três e dez anos de idade que possuem familiaridade com a tecnologia e não possuem acesso ao computador por imposição dos pais, tendo sua atenção transferida para a TV. O ultimo grupo é representado por pessoas que não tem familiaridade com tecnologias, mas que vêem a TV como algo que podem dominar. Essas pessoas jogam por ser uma forma fácil de entretenimento, pois são mais simples que os demais serviços para a TV.

Jogos na TV podem ser utilizados de diferentes formas. Eles podem ser utilizados como produtos únicos de uma emissora, ou seja, um canal especializado em jogos para TV digital. Esses canais transmitem apenas jogos, em horários definidos do dia, como se fossem programas de TV, podendo assim os jogadores praticar todos os dias, e até submeterem seus recordes a partir do canal de retorno. Um exemplo de canal de jogo é o canal PlayinTV, um dos maiores da Europa apresentando em sua lista de jogos para TV mais de 200 títulos, indo de jogos clássicos como xadrez, a criações próprias como o Frogs, cujo objetivo do jogo é levar um sapo de um lado a outro do rio pulando para fugir dos desafios. Como podemos ver na Fig. 10, tais jogos apresentam uma boa qualidade gráfica mesmo com as limitações de jogos para a TV, e a diversidade de jogos encontrados no canal mostra que existem muitas possibilidades de tipos de jogos que podem agradar a todos que jogam pela TV.

Figura 10 - Jogos: a) Chess (jogo de tabuleiro); b) Frogs (jogo arcade); c) Johnny Megatone (jogo de ação arcade) d)Tennis (jogo esportivo)

Jogos também podem ser utilizados junto à programação, na forma de conteúdo extra. Eles podem, por exemplo, em um programa de auditório colocar o usuário interagindo com o programa ao responder perguntas que podem ser feitas aos participantes do programa, dando ao telespectador uma imersão maior no programa. Os jogos também podem ser utilizados na TV para atos publicitários substituindo ou estendendo a propaganda com um jogo.

No Brasil, comercialmente as experiências com jogos na TV são escassas, se limitando a jogos pré-instalados em receptores de TVs pagas (vide Fig. 11) e alguns poucos jogos em TV aberta, como o jogo Garganta e Torcicolo (vide Fig. 12), que se tratava de um jogo onde dois participantes ligavam para o programa via telefone, e controlavam os personagens do jogo através das teclas do telefone.

Figura 11 - Jogo na TV por assinatura SKY

Figura 12 - Jogo Garganta e Torcicolo, cujo objetivo era destruir o maior número de ovelhas

Apesar de poucas, existem tentativas de uso de jogos na televisão brasileira, de forma que com o Ginga estas tentativas podem aumentar, e passarem a ser um nincho de negócio para as emissoras interessadas. Desta forma o desenvolvimento de um framework compatível com o Ginga pode ser capaz de acelerar o desenvolvimento de jogos para TV Digital ajudando na popularização de jogos e programas que se utilizam destes jogos na TV.

4.2. Metodologia de Desenvolvimento

Para a concepção do Athus foi utilizada a metodologia encontrada em (Henrique et al., 2010) que divide a criação de um framework em seis passos: análise, criação da arquitetura, refinamento, implementação, testes e documentação. Apesar de serem apresentados como passos sequênciais, foram utilizados ciclos de desenvolvimento, de forma que à medida que testes fossem realizados, fosse possível levantar novos requisitos e assim expandir e corrigir o framework.

Duas outras metodologias foram estudadas: Johnson (1997) e (Pree et al. ,1999), porém a descrita anteriormente foi a escolhida. Isto ocorreu pelo fato de não terem sido desenvolvidas aplicativos de jogos para TV pela equipe, requisito necessário para utilização da metodologia proposta por Johnson (1997), que propõe pegar a partir de diversas aplicações de um mesmo domínio as características comuns entre elas e a partir destas gerar o framework. A outra metodologia descartada foi a encontrada em (Pree et al. ,1999), cuja abordagem é feita a partir da construção de um modelo de objetos para uma aplicação específica, que é refinado através de sucessivas iterações. Para cada iteração, os pontos flexíveis do aplicativo são inicialmente identificados. Estes pontos flexíveis são documentados a medida que são concebidos e implementados. Depois disso, o framework é testado para determinar se os requisitos do domínio foram cumpridos. Caso não tenham sido, aperfeiçoamentos são feitos através de uma nova rodada de iterações.

A etapa de análise aborda o levantamento dos requisitos e conceitos para a criação do framework. Os principais requisitos levantados podem ser encontrados no início do quarto capítulo, que engloba as principais características do domínio. Porém, levando tais requisitos para o âmbito do Ginga, percebeu-se que não seria possível atender alguns deles, como suporte a objetos gráficos 3D, devido a incompatibilidade do middleware com objetos gráficos 3D e tal suporte não estar mencionado na norma do Ginga. Porém, também baseados

no Ginga, novos requisitos surgiram, como a possibilidade de diversos tipos de dispositivos de entrada de dados, não sendo possível apenas o uso do controle remoto, mas também celulares e outros dispositivos móveis para a entrada de dados para as aplicações. Também ao longo dos ciclos foi identificada a necessidade de dar suporte aos eventos lançados pela emissora de TV, que podem ser utilizados tanto para sincronização da aplicação com o vídeo como para avisar de ações enviadas pela emissora de TV.

A arquitetura foi moldada a cada ciclo do desenvolvimento atribuindo desde elementos básicos, como suporte a animação e objetos de cena, até os módulos mais específicos como o de suporte a troca remota de dados e tratamento de eventos lançados pela emissora, sendo a cada ciclo necessário o refinamento e detalhamento de cada mudança na arquitetura.

A fase de implementação foi a mais longa no processo de desenvolvimento do framework. Foi utilizada nessa fase a IDE Eclipse para a codificação do projeto, em conjunto com dois ambientes de testes distintos para o subsistema utilizado, o NCL. O primeiro foi o Set-Top-Box Virtual desenvolvido pelo laboratório Telemídia da Pontifícia Universidade Católica do Rio de Janeiro, que implementa o Ginga-NCL em uma máquina virtual de forma aos usuários poderem fazer testes de seus aplicativos em um middleware no próprio ambiente de programação. O segundo ambiente utilizado foi um Set-Top-Box comercial da empresa Proview com o middleware da RCA Software, que permite testar os aplicativos em um ambiente real de TV Digital, enquanto o STB virtual apenas simula o middleware, ignorando restrições como o controle remoto, atraso de interação, formato da TV, dentre outros aspectos que são vistos em nossas casas, ou seja, ambiente reais da TV Digital.

A etapa de testes e validação foi realizada em três etapas: a primeira baseada em testes unitários para cada nova funcionalidade desenvolvida. Logo após o desenvolvimento de um número suficiente de componentes foram desenvolvidos pequenos jogos para a verificação de integração entre os componentes e de suas funcionalidades. Ao fim da implementação de todos componentes, foi feita a validação do framework com testes realizados com programadores para avaliação da ferramenta. Estes testes são descritos no capítulo VI.

Ao final do desenvolvimento, testes e demais etapas de produção de um framework, é necessário fazer documentação do framework para que outros programadores possam utilizar o framework de forma independente dos autores. Desta forma os módulos são apresentados

nas seções seguintes. O detalhamento do mesmo e obtenção do código podem ser encontrados no link: http://h264.lavid.ufpb.br/~ricardo/AthusR1/doc/.

4.3 API

O Athus consiste de um framework de suporte ao desenvolvimento de jogos para TV Digital. Sendo assim, é necessário disponibilizar suporte as diversas características da plataforma (Ginga) com conceitos de desenvolvimento de jogos. O framework proposto utiliza o paradigma de orientação a objetos (Korson, 1990), disponibilizando as funcionalidades para o desenvolvedor conforme Tabela 2.

Tabela 2 - Funcionalidades providas pelo Athus

Funcionalidade	Descrição
Executar vídeos e áudios	Execução de áudio e vídeo locais à aplicação, e fluxos remotos
Troca de dados remotos	Suporte a conexão com aplicações remotas, como servidores e outros jogadores
Sincronização com Fluxo de vídeo	Permite que os jogos criados possam ser sincronizados com o fluxo de vídeo que é enviado junto à aplicação
Controle de I/O	Controle dos dispositivos de entrada e saída de dados, correspondendo ao controle da tela e do controle remoto
Construção de cenários	Suporte as técnicas de Tilemapping e Tilesets
Detecção de colisão	Algoritmos de detecção de colisões
Suporte a IA	Algoritmos para desenvolvimento de agentes inteligentes: NPCs (*Non-Playable Characters*)
Técnicas de animação	Suporte a animações utilizando tiles e loops constantes de animação

Por se tratar de um framework para o Sistema Brasileiro de Televisão Digital, o Athus segue a norma ABNT NBR 15606-2 "Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital Parte 2: GingaNCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações".

Em relação a sua distribuição, o Athus foi disponibilizado sob a licença GPL (GPL, 2007). O uso desta licença permite que qualquer programador que utilize o framework para seus projetos tenha a liberdade de modificá-lo e adicionar funcionalidades extras, sob a demanda de manter o código resultante sob a mesma licença do Athus e atribuir a autoria do Athus aos desenvolvedores originais.

4.4. Arquitetura

O ATHUS atua diretamente sobre o Ginga, de forma independe, como podemos observar na Fig. 13. O modelo de camadas utilizado pelo Athus garante que qualquer plataforma que implemente o Ginga, execute as aplicações desenvolvidas utilizando o framework Athus.

Figura 13 - Arquiteturas Athus e Ginga.

A arquitetura inicial apresentada na Fig.13 é detalhada na Fig. 14. A arquitetura detalhada do ATHUS apresenta-se dividida em componentes criados para dar suporte ao desenvolvedor de jogos para TV Digital Brasileira. A arquitetura utilizada é baseada no modelo MVC (Burbeck, 1992), que permite a separação dos módulos em três categorias: acesso a dados (*Model*), regras de negócio (*Controller*) e interface gráfica (*View*). A diferença está nas camadas *Controller* e *Model*, as quais foram renomeadas para *Core* e *Interaction*, respectivamente. Esta mudança ocorre, devido à camada *Core* não apenas controlar os demais componentes, ela é responsável também por gerar o comportamento dos elementos apresentados pela camada *View*. A camada *model* foi substituída pela camada *Interaction*, que abstrai as três formas de uma aplicação Ginga receber informações: via canal de interatividade, via eventos lançados pela emissora e via dispositivo de interação.

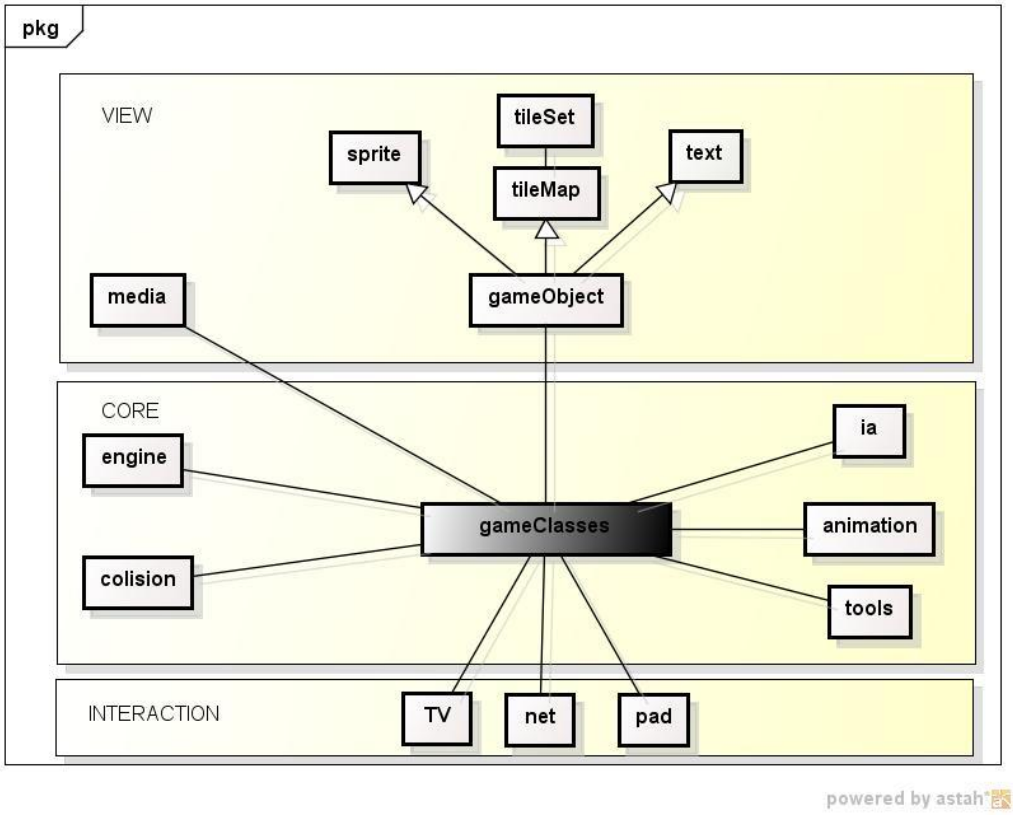


Figura 14 - Detalhamento da Arquitetura do ATHUS

Apesar da mudança, a arquitetura é beneficiada com a flexibilidade de manutenção oferecida ao desenvolvedor, pois possibilita que mudanças, por exemplo, feitas na interface não tenham impacto nas outras camadas, característica do modelo MVC.

Na arquitetura da Fig. 14, notamos a existência de um componente que conecta todos os demais, o *gameClass*. O *gameClass* não é constituído por apenas um componente, ele é constituído por toda lógica de programação do programador, que pode utilizar os módulos do ATHUS que desejar nesta lógica, já que os módulos do framework foram desenvolvidos de forma que são independentes entre si.

A Fig. 15 mostra a arquitetura expandida em um diagrama de classes, apresentando as funções contidas em cada módulo.

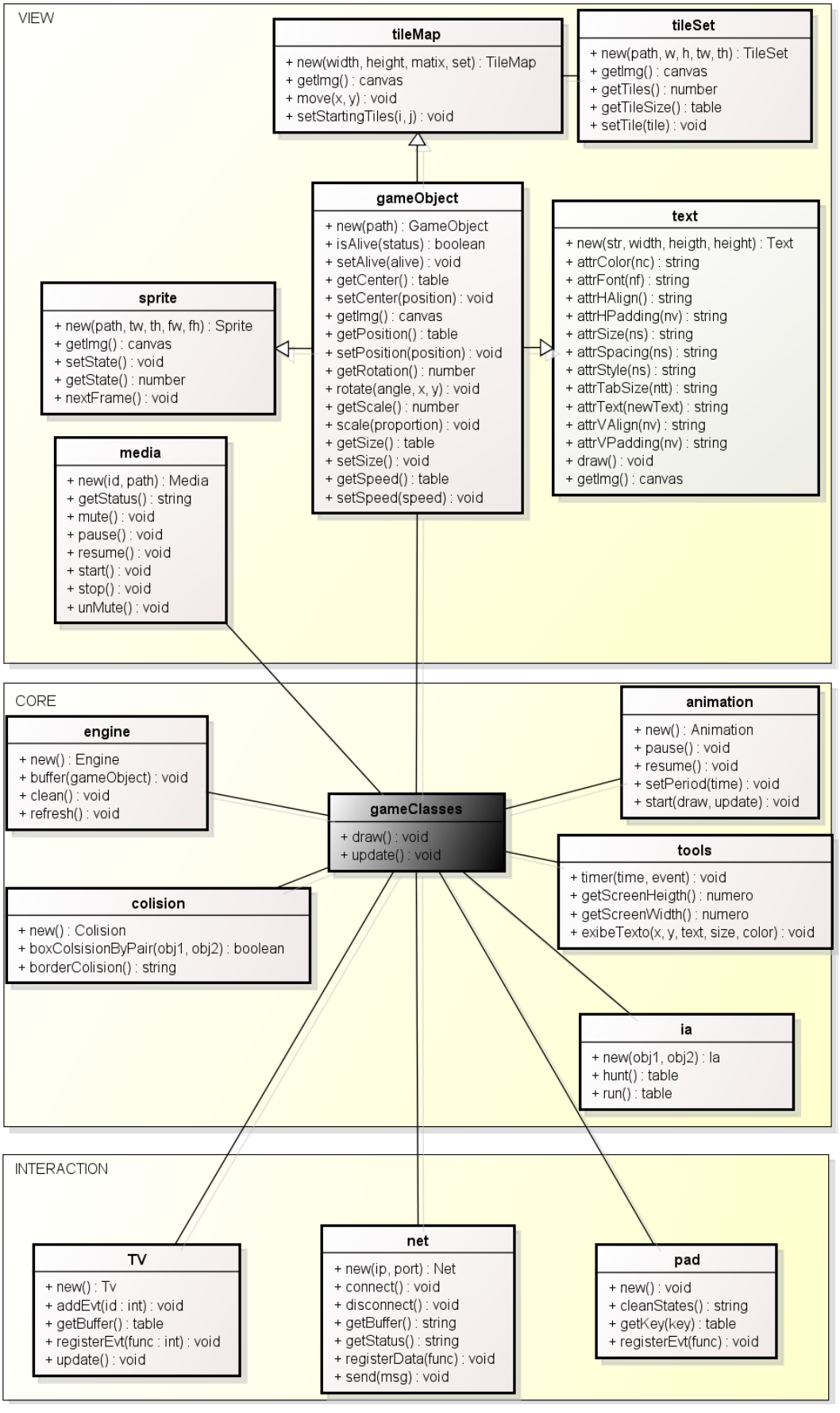


Figura 15 – Diagrama de Classes do framework Athus

Todos os módulos e suas funcionalidades são apresentados a seguir, descrevendo como funcionam, seus principais atributos, características e parâmetros.

4.4.1 Componente *Animation*

Componente responsável pela manutenção das taxas de atualização de tela(frames por segundo - FPS) e de jogo(updates por segundo - UPS). Para isto o componente recebe duas funções, uma que deve conter todas as chamadas de desenho do jogo, e outra com todas as chamadas de atualização do estado interno do jogo, desta forma é possível priorizar a manutenção das taxas de atualização do jogo, de forma a pular algumas chamadas da função de desenho. Isto é uma boa técnica devido ao fato de que alguns frames que sejam pulados, não irão interferir na jogabilidade, porém se houver atraso na atualização do jogo, pode haver problemas que prejudiquem a jogabilidade, como o travamento temporário do jogo.

Este loop (vide Fig. 16) de chamadas das funções de desenho e atualização podem ser iniciadas e paradas de acordo com a vontade do programador, de forma que o jogo pode ser pausado e reiniciado sem perda de informações.

A taxa de chamada deste loop, também pode ser ajustada, atribuindo ao componente o valor do período desejado. Este período corresponde ao tempo de espera entre uma chamada e outra, sendo assim, quanto menor o período maior o número de chamadas e maiores as taxas de FPS e UPS.

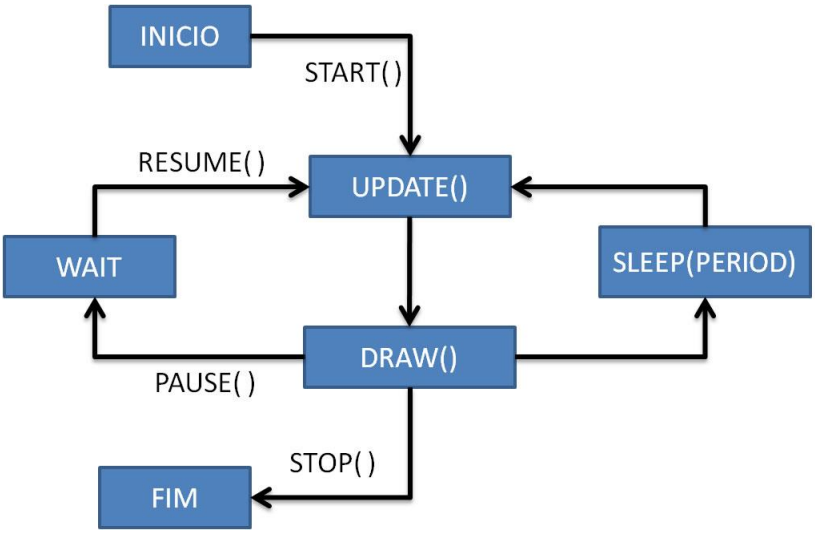


Figura 16 - Comportamento componente Animation. O Ciclo, depois iniciado, fica indeterminadamente fazendo chamadas às funções UPDATE e DRAW com um intervalo de tempo fixo entre a sequência de chamadas. O ciclo pode ser pausado ou parado pelo programa.

4.4.2 Componente *Engine*

Engine é o componente responsável por desenhar as imagens dos componentes do jogo em tela. Para isso é implementado a técnica de *double buffer* (vide Fig. 17). Esta técnica consiste em usar um buffer de memória para resolver os problemas de cintilação (flicker) associados com as operações de pintura múltiplas. Quando o buffer duplo está habilitado, todas as operações de pintura são processadas pela primeira vez para um buffer de memória em vez de a superfície de desenho na tela. Depois de todas as operações de pintura serem concluídas, o buffer de memória é copiado diretamente para a superfície de desenho associado a ele. Como apenas uma operação de gráficos é realizada na tela, o problema da imagem piscando associada com as operações de pintura é eliminado.

À medida que imagens são adicionadas ao buffer, elas começam a ocupar o mesmo espaço na imagem em memória, sendo assim os gráficos bufferizados primeiro irão ocupar os níveis mais profundos da imagem, enquanto os bufferizados por último ocuparão os níveis mais superficiais da tela. Um exemplo é: existem duas imagens A e B. Primeiramente A vai para o buffer, de forma que será desenhada sobre o conteúdo do buffer, que anteriormente estava vazio. Agora B é colocada no buffer. Com isto B será desenha sobrepondo os pontos que tiver em comum com o buffer, ou seja, a imagem A. Se outras imagens forem colocadas

no buffer o mesmo ira ocorrer, ate que o buffer seja desenhado em tela e depois limpo para a criação de uma nova tela.

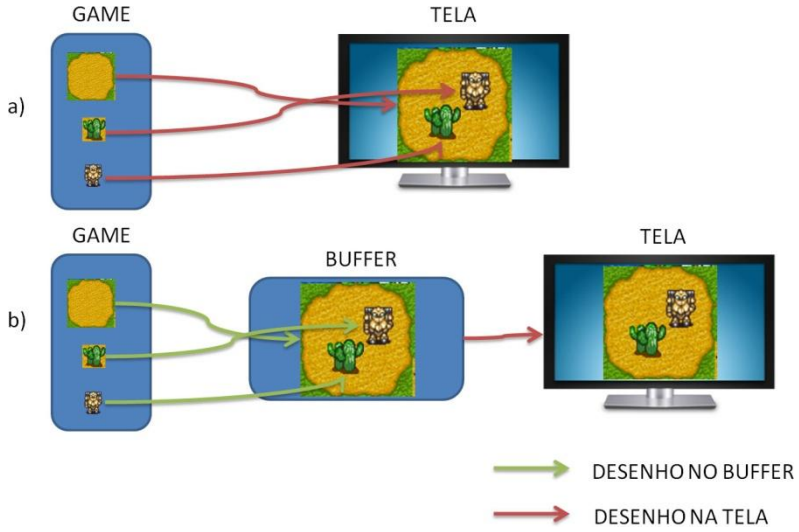


Figura 17 - Técnica do double buffer. a) Quando não há uso da técnica, todos os gráficos são desenhados diretamente na tela, de forma que todo desenho causa atualização da tela. b) Com o uso do double buffer, os gráficos desenham em um buffer de memória, e depois de tudo desenhado, ocorre apenas uma chamada de desenho na tela.

4.4.3 Componente *Colision*

Possui algoritmos que auxiliam o programador a identificar o acontecimento de colisões entre objetos no jogo.

Uma técnica simples implementada no componente é de detecção de colisão por comparação de retângulos. Nesta técnica o componente a ser testado é considerado sendo um retângulo, de forma a calcular se naquele momento um retângulo está sobrepondo o outro, ou seja, se eles possuem área em comum.

Isto é feito da seguinte forma:

a. São extraídas dos objetos as seguintes variáveis:
   - x1 = inicio horizontal do retângulo 1;
   - y1 = inicio vertical do retângulo 1;
   - x2 = inicio horizontal do retângulo 2;
   - y2 = inicio vertical do retângulo 2;
   - tx1 = tamanho horizontal do retângulo 1;

- ty1 = tamanho vertical do retângulo 1;
- tx2 = tamanho horizontal do retângulo 2;
- ty2 = tamanho vertical do retângulo 2;

b. Depois feitas as seguintes comparações:
   - x1 + tx1 < x2;
   - y1 + ty1 < y2;
   - x1 > tx2 + x2;
   - y1 > ty2 + y2;

c. Se alguma destas hipóteses for verdadeira, não haverá colisão, caso contrário ocorreu colisão.

Novos algoritmos de colisão podem ser adicionadas ao framework por qualquer usuário, de forma que ele irá personalizar o framework a suas necessidades. Assim, um desenvolvedor que necessitasse utilizar uma técnica de colisão analisando a circunferência dos objetos, precisaria apenas adicionar seu algoritmo ao componente Colision, podendo assim reutilizar esta técnica em seus trabalhos futuros.

4.4.4 Componente *Tools*

O componente tools é um componente de auxilio ao desenvolvimento. Ele possui funcionalidade que ajudam na construção da lógica do jogo e ferramentas para auxilio de debug do mesmo. Dentre estas funcionalidades temos: exibição de texto diretamente na tela, para testes em TVs que não possuam terminal de saída para logs; consulta dinâmica ao tamanho da tela de desenho do jogo; modificação dinâmica do tamanho da tela de desenho do jogo; temporizador para chamadas de funções após um tempo específico.

4.4.5 Componente *IA*

O componente IA é responsável por implementar algoritmos de inteligência artificial, de forma a auxiliar o programador em jogos que exijam a utilização de agentes inteligentes no jogo.

Dois algoritmos básicos de IA foram desenvolvidos para testes no framework: um de perseguição e outro de fuga. Um algoritmo de perseguição e fuga pode ser divido em duas

partes (BOURG e SEEMAN, 2004): uma, é a tomada de decisão para que se inicie o processo, e outra é a ação propriamente dita. Pode-se dizer que existe um terceiro processo a ser considerado, que é a evasão de possíveis obstáculos que possam estar no caminho encontrado pelo algoritmo. Nos algoritmos implementados, não são considerados tais obstáculos.

Os algoritmos usados no componente envolvem o decremento das distâncias entre as coordenadas do caçador e da presa, seguindo as seguintes regras:

- Se a posição no eixo X do caçador for menor que a posição no eixo X da presa, a posição X do caçador será incrementada.
- Se a posição no eixo Y do caçador for menor que a posição no eixo Y da presa, a posição Y do caçador será incrementada.
- Se a posição no eixo X do caçador for maior que a posição no eixo X da presa, a posição X do caçador será decrementada.
- Se a posição no eixo Y do caçador for maior que a posição no eixo Y da presa, a posição Y do caçador será decrementada.

4.4.6 Componente *Net*

Possui a função de gerenciar a troca de dados entre a aplicação local e uma aplicação remota. Para criar uma conexão com uma aplicação remota, é necessário conhecer IP e porta da aplicação remota. Com uma conexão estabelecida, é possível enviar e receber mensagens.

Mensagens de erros e notificações da conexão são guardados em uma fila, de forma que o programa pode ficar consultando esta fila para obter informações de possíveis problemas como desconexão, e também confirmações como o pedido de conexão.

Semelhante as notificações também há uma fila para mensagens vindas da aplicação remota. Caso nada novo tenha chegado, a fila estará vazia, caso contrario o programa retira a mensagem da fila para tratá-la.

Estas duas filas fazem parte do componente quando este está configurado em seu modo passivo, ou seja, o programa principal que esta utilizando o módulo deve fazer os pedidos ao componente que ira retornar a próxima informação na fila. Em seu modo ativo o

componente recebe como parâmetros duas funções, uma para as notificações e outra para as mensagens, que serão chamadas imediatamente caso chegue alguma informação do computador remoto, sem ser armazenada na fila.

4.4.7 Componente Pad

Componente com a funcionalidade de tratar os eventos provindos do dispositivo de interação, ou seja, o controle remoto. Sabendo a tecla desejada, o programador faz uma chamada ao componente pedindo o estado atual daquela tecla, que podem ser:

- free: indicando que nada ocorreu com a tecla;
- press: indicando que a tecla esta sendo pressionada no momento da consulta;
- release: indica que a tecla foi pressionada e solta;

As teclas suportadas pelo framework são:

- Teclas Coloridas:
    - RED, GREEN, BLUE, YELLOW;
- Teclas Numéricas:
    - 0,1,2,3,4,5,6,7,8,9;
- Setas:
    - CURSOR_DOWN, CURSOR_UP, CURSOR_LEFT, CURSOR_RIGHT;
- Teclas de Configuração:
    - MENU, INFO, ENTER;

Com estas informações, o programador poderá saber o que esta ocorrendo com cada tecla que influenciar na dinâmica do jogo. Assim como o componente Net, o Pad possui o comportamento ativo e passivo. As funcionalidades acima citadas fazem parte do comportamento passivo, pois no comportamento ativo uma função é registrada para receber a informação de qual tecla foi pressionada ou solta assim que o evento ocorrer.

4.4.8 Componente TV

Componente responsável pela comunicação entre a emissora e a aplicação na TV. A emissora de TV pode mandar em tempo de execução da aplicação informações que podem atualizar a

aplicação, podendo assim modificar de alguma forma o estado do jogo, sincronizando este com o programa que esta sendo exibido.

Para isto, o programador terá que produzir sua aplicação em conjunto com a emissora, para que ele saiba quais os eventos que serão enviados pela emissora e seu valor. Sabendo estes dados, o programador insere no componente o nome de tais eventos, para que o componente TV fique escutando quando este evento chegar, para adicionar o evento e seu valor a uma fila.

Semelhante aos componentes Net e Pad, o programador poderá registrar uma função (ativando o modo ativo) que receba imediatamente a informação de qual evento ocorreu e seu valor, ao contrario do modo passivo em que ele tem que fazer chamadas ao componente para consultar se houve mudanças e quais foram.

4.4.9 Componente *GameObject*

O objeto de jogo é o componente básico do game. Nele podem ser armazenadas todas as informações básicas de um objeto, como velocidade, posição na tela, dimensões da imagem relativa ao objeto, posição do centro do objeto e informações de operações que possam ter sido feitas sobre ele. Estas operações são operações básicas para a criação de efeitos e a apoios a animações, como operação de escala sobre a imagem, que trata de redimensionar a imagem original do objeto, aumentando-a ou reduzindo-a. Outra operação semelhante é a de rotacionar o objeto em torno de seu eixo, ou de algum ponto passado a função.

Este é o componente básico do jogo, pois todos os outro componente gráficos manipuláveis pelo programador herdam as características do gameObject, não se limitando apenas aos componentes desenvolvidos no framework, mas outros componentes desenvolvidos pelos programadores podem herdar estas características.

4.4.10 Componente Media

O componente media é responsável pela exibição e controle de mídias externas ao jogo, como fluxos de vídeo remotos e áudios e vídeos locais, que não podem ser executados diretamente via linguagem Lua. Com isto os programadores não terão que se preocupar com

o código NCL, pois o mesmo é gerado automaticamente pelo componente mídia, de forma que o programador tem a sua disposição a manipulação das mídias com chamadas simples, como iniciar, pausar e parar a mídia.

Este comportamento só é possível devido à linguagem NCL poder ser editada em tempo de execução, ou seja, ela permite que novas mídias e comportamentos sejam adicionados a seu documento mesmo enquanto esta está sendo executada.

4.4.11 Componente Sprite

O componente Sprite herda todas as características de um *GameObject*, implementando uma funcionalidade específica, a técnica de tratamento de sprites, que consiste no uso de sprites para simular a animação de um objeto. Sprites são os conjuntos de dados, ou no nosso caso imagens, que definem determinado objeto ou personagem num jogo. Para um avatar, por exemplo, podemos ter um sprite que contenha as posições verticais e horizontais dela no mundo e a direção para onde ela está virada. Cada uma dessas imagens pode representar um frame de uma animação maior, como o andar do avatar, e vários frames passados em uma taxa constante irá passar a impressão de uma animação. Na Fig. 18 é mostrado o sprite utilizado para simular a caminhada de um avatar, e na Fig. 19 a exemplificação da técnica.

Figura 18 - Exemplo de sprites utilizados.

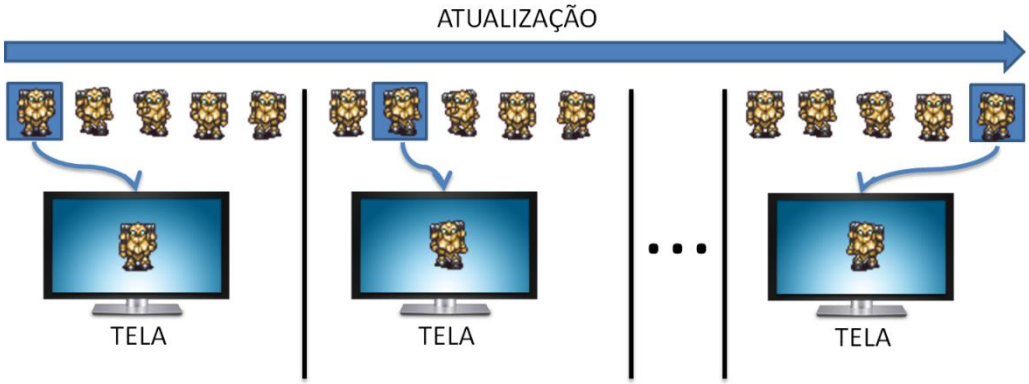


Figura 19 - Exemplo da utilização de Sprites.

4.4.12 Componente Text

Também herda as características de um *gameObject*. Este componente é responsável pela manipulação de textos no jogo. Por este componente é possível manipular várias características de um texto, como tamanho, fonte, cor, alinhamento vertical, alinhamento horizontal e espaçamento.

A utilização dessas funcionalidades permite a criação de textos organizados e gerados dinamicamente, podendo assim gerar maior conteúdo ao jogo sem necessidade de ter tais textos salvos como imagens.

A utilização deste componente em conjunto com o componente de sprites e gameObject permite a criação de menus que auxiliem o uso do jogo, como telas de pause, instruções de jogo e outros.

4.4.13 Componente TileMap

O componente TileMap é responsável pela criação de cenários do jogo a partir da técnica de uso de tiles. Para isto é definida no TileMap uma matriz de tamanho variável (parâmetros passados pelo programador) que irá guardar informações sobre qual tile representa qual posição na matriz e o tipo do tile. Essas informações servem para criar cenários de vários tamanhos logicamente, ou seja, o cenário esta guardado na memória, porém por não estar desenhado, não ocupa grande espaço em memória. Para transformar a matriz da memória em uma imagem a ser desenhada na tela, é visto primeiramente o tamanho da imagem a ser

desenhada, de forma que apenas uma janela da matriz será utilizada naquele momento para o desenho, para que não seja necessário desenhar toda a matriz de cenário (vide Fig. 20). Com a janela a ser desenhada definida, começa a etapa de desenho, onde a janela é percorrida, vendo qual tile deve ser desenhado na imagem relativa ao cenário, e desenhando-o na imagem. Este processo vai desde a primeira posição da janela até a ultima. Ao final, uma imagem com o cenário visível naquele momento foi gerado e pode ser desenhado na tela.

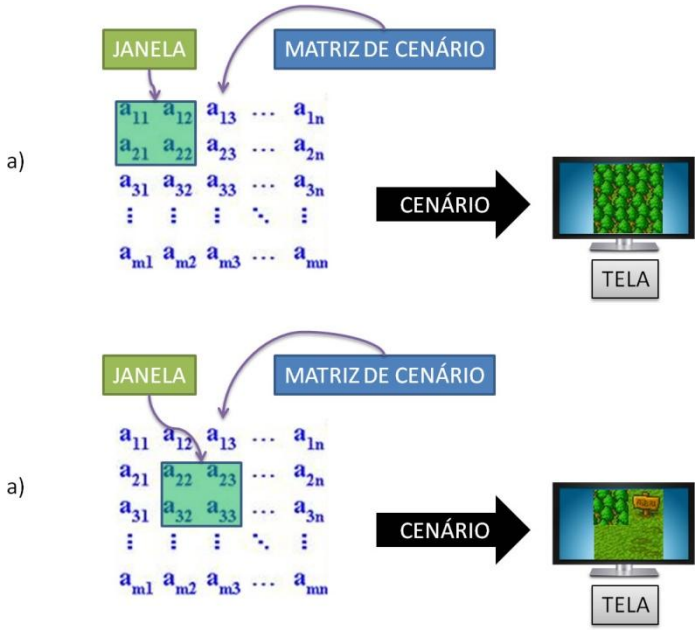


Figura 20 - Funcionamento do TileMap. A posição da janela de desenho determina a parte do cenário a ser desenhada, em 'a' o cenário mostrado é diferente do cenário de 'b' devido à mudança da posição da janela em relação à matriz.

4.4.14 Componente *TileSet*

Para auxiliar a manipulação dos tiles, foi criado o módulo *tileSet*. Um jogo geralmente não possui apenas um tile, mas sim diversos tiles que geralmente são encontrados juntos em uma única imagem (sets). Assim, o tileSet possui funcionalidades para separar este tiles e retorná-los separadamente (vide Fig.21).

Para isso, cada tile do conjunto de tiles recebe um número identificador, que é dado pela sua ordem na figura. Desta forma, para pegar um tile de um tileSet, basta passar o identificador do tile.

Figura 21 - Retirada de um tile do tileset, a partir da chamada de uma função.

4.5 Detalhes de Implementação

Para a implementação do Athus, foi utilizado o ambiente de desenvolvimento Eclipse. O Eclipse é uma IDE (*Integrated Development Environment*) desenvolvida em Java, com código aberto para a construção de programas de computador. O projeto Eclipse foi iniciado na IBM que desenvolveu a primeira versão do produto e doou-o como software livre para a comunidade. Por ter seu código aberto, muitos plugins para desenvolvimento de aplicações são produzidos para uso no Eclipse, e dentre estes plugins estão alguns utilizados para o desenvolvimento de aplicações para TV Digital, o NCL-Eclipse e o LuaEclipse.

O NCL-Eclipse tem o objetivo de agilizar o desenvolvimento de aplicações para TV digital Interativa em NCL, pois possui funcionalidades como a correção automático de erros sintáticos e semânticos de código, como também permite a pré-visualização da aplicação antes de ser executado em um STB. Porém Ncl-Eclipse não apresenta suporte a linguagem Lua, sendo assim necessário o uso de outro plugin, o LuaEclipse.

O LuaEclipse ajuda no desenvolvimento de aplicações para linguagem Lua, possibilitando ao desenvolvedor um ambiente com syntax highlight, code completion, verificação de erros de compilação, agrupamento de código e comentários e um executor de código para testes. O uso destas ferramentas facilita muito o desenvolvimento de aplicações, pois evita pequenos erros sintáticos e permite o teste continuo das aplicações, porém há uma falha neste ambiente, que é a falta de integração dos dois plugins: enquanto o Ncl-Eclipse tenta seguir a norma brasileira para aplicações de TV-Digital, possuindo assim compatibilidade com estes requisitos, o LuaEclipse não tem como foco aplicações para TV Digital, não dando suporte assim aos módulos que são exclusivos desta plataforma, fazendo com que o programador possa testar estes módulos apenas no ambiente real ou em

simuladores. Para amenizar este problema, um terceiro plugin é utilizado, o Remote System Explorer. O RSE permite o acesso a arquivos e sistemas remotos. Com isto é possível agilizar testes na máquina virtual, pois é possível modificar diretamente pela interface do eclipse as aplicações que já estão no STB virtual e executá-las.

Para a geração da API do framework, foi utilizado o LuaDoc. LuaDoc é um instrumento gerador de documentação para códigos Lua . Ele analisa as declarações e os comentários nos códigos de um projeto Lua e produz um conjunto de páginas XHTML descrevendo os comentários, as declarações e funções de cada arquivo. LuaDoc é um software livre e utiliza a mesma licença do Lua.

Para os testes das aplicações foram utilizados dois ambientes, um virtual e um real. O Virtual: O Set-top Box Virtual GingaNCL é uma máquina virtual construída para facilitar o processo de distribuição e implantação do GingaNCL, a versão do player NCL que conta com os mais novos recursos de apresentação de aplicações declarativas, melhor desempenho e maior proximidade de uma implementação embarcada em set-top-boxes reais. Uma máquina virtual é a implementação em software de um computador que executa programas tal qual uma máquina real. Ela pode ser vista como uma cópia isolada de uma máquina real. Cabe ao software de virtualização dividir o hardware real, gerenciado por um sistema operacional servidor, entre diversas instâncias virtuais desse hardware, gerenciadas por sistemas operacionais convidados. Os sistemas operacionais convidados não precisam ser o mesmo que o hospedeiro, e nem precisam ser os mesmos entre si. Isso quer dizer que em um sistema hospedeiro pode-se ter diferentes sistemas operacionais (e suas aplicações) executando ao mesmo tempo, concorrentemente, disputando o compartilhamento do hardware real. O uso do STB virtual tem pontos positivos e negativos. Os principais pontos positivos são:

- Instalação descomplicada: o programador não tem que se preocupar com problemas de configuração e instalação do middleware, pois tudo já vem configurado na máquina virtual.
- Portabilidade entre diferentes Sistemas Operacionais (SO): isto permite que os desenvolvedores possam programar e fazer testes compatíveis entre si, mesmo que usem máquinas com configurações diferentes, mesmo de SO, pois a maquina virtual roda acima destes, não tendo diferença qual o programador esteja utilizando.

- Ótimo ambiente de testes de aplicações NCL / NCLua: o STB virtual possui suporte a aplicações NCL puras e com o uso de Lua que sigam as normas da ABNT 15606-5, tornando possível desenvolver e testar aplicações que deverão executar corretamente em ambientes reais, pois estes ambientes também devem seguir as mesmas normas.

Os principais pontos negativos do uso de um ambiente virtual são:

- Não é a realidade comercial: muitos middleware que estão no mercado não possuem todas as funcionalidades escritas nas normas, e também algumas que implementam o fazem de maneira incorreta, fazendo com que muitas aplicações desenvolvidas no STB virtual não funcionem corretamente em alguns ambientes reais.

- Aspectos de interação e usabilidade: o principal problema de uso de um STB virtual é em relação à interação entre usuários e dispositivos, pois enquanto usa o STB virtual, o desenvolvedor e testador estão interagindo com um computador, que possui teclado e mouse para a interação, porém isto não condiz com a realidade de programas para TV digital, que se baseia no uso de televisores e controles remotos para interação do usuário, podendo assim o uso do ambiente virtual causar problemas de usabilidade quando o programa for ser utilizado em ambiente real.

Testes também foram realizados em uma plataforma comercial. O set-top-box XPS-1000 da Proview utiliza o middleware Ginga implementado pela RCA Software. Este middleware implementa muitas das funcionalidades previstas para o Ginga, sendo possível testar animações, trocas de dados remotas e a construção de alguns jogos. Porém, funcionalidades como o controle do dispositivo de interação, transformações de escala e rotação não puderam ser testadas corretamente devido à falta de compatibilidade com as normas Ginga. A execução de aplicações neste ambiente proporcionou identificar erros no componente Animation, quando havia limitação de memória, o jogo ficava lento e não recebia eventos de teclas do controle remoto. Isto ocorria pois a pilha de processos gerada pelo algoritmo se tornava enorme e não era possível liberar o programa para receber os demais eventos. Para correção deste problema, o algoritmo de animação foi modificado. Foi adicionado ao módulo uma variável que diz a quantidade máxima de frames de desenho do jogo que podem ser descartadas para dar prioridade a atualização do jogo. Caso esse número seja atingido, o componente limpa toda a pilha de chamadas tanto de desenho como de atualização, iniciando o tratamento das chamadas do zero, retomando assim a aplicação, porém com alguma perda e criação de *flicker* para o usuário.

Capítulo

5

Experimentos Desenvolvidos

Durante o desenvolvimento do ATHUS, foi necessário validar os componentes desenvolvidos a cada ciclo, de forma que alguns jogos resultaram de testes realizados com o uso do framework. Estes jogos utilizam funcionalidades de animação, construção de cenário, troca remota de dados e controle de dispositivos de interação. Para testar as funcionalidades do framework, foram desenvolvidos os seguintes jogos: AsteroidsTV, GingaCraft, RoboWalker, Poker4TV, GingaHero e Turma da Árvore. Nas subseções seguintes são descritas as experiências realizadas.

5.1. Asteroids4TV

Asteroids[1] é um jogo de fliperama lançado em 1979 pela Atari Inc, foi um dos jogos mais populares e influentes da Idade de Ouro dos Jogos Arcade. Neste jogo, o jogador controla uma nave espacial em um campo de asteróides. O objetivo do jogo é atirar e destruir os asteróides e não colidir com qualquer um deles ou seus destroços, aumentando assim a pontuação do jogador.

O jogo Asteroids4TV, foi baseado no Asteroids. Possui os mesmos objetivos e jogabilidade do jogo original, sendo baseado em algoritmos para desenhar a nave (triângulo), tiros (pontos) e os asteróides (funções matemáticas). Para a TV, os comandos do jogo foram mapeadas para as setas e botão de confirmação do controle remoto. Nas setas para esquerda e direita, o jogador controla o ângulo da nave, para frente e para trás controla a velocidade, e com o botão de confirmação atira.

[1][http://www.ataritimes.com/article.php?showarticle=174]

Este jogo serviu como experimento para o componente de animação, o que possibilitou taxas constantes de atualização das posições dos asteróides, tiros e nave. O resultado pode ser visto na Fig. 22.

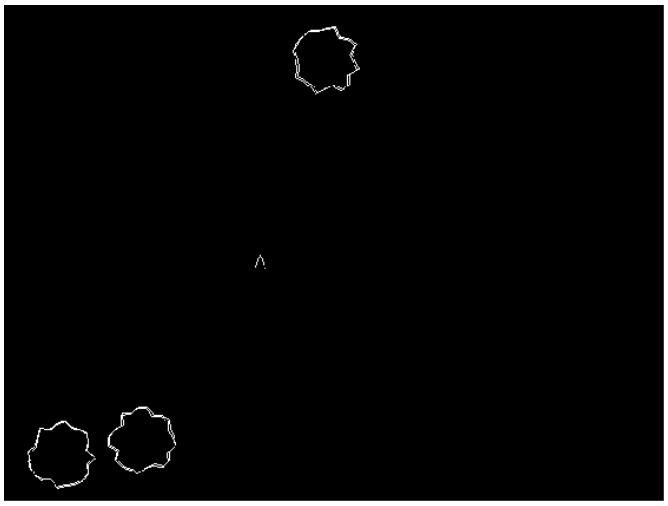


Figura 22- Jogo Asteroids4TV.

5.2. GingaCraft

Pong foi um dos primeiros jogos de arcade, consta de um jogo de tênis simples com gráficos bidimensionais sendo um dos primeiros jogos a ganhar grande popularidade. O objetivo do jogo é derrotar o seu adversário em um jogo simulando uma partida de tênis, ganhando o jogador que obtiver a maior pontuação. O jogo foi originalmente fabricado pela Atari Inc. [Atari], e lançado em 1972.

O GingaCraft (Segundo et. al., 2009) (vide Fig. 23) é um jogo baseado no Pong, se diferenciando em dois aspectos: primeiro por utilizar os eixos X e Y para controle da raquete; segundo por trazer um contexto futurístico ao jogo, substituindo as raquetes por naves espaciais. No seu desenvolvimento foram utilizados os componentes *Animation*, GameObject e Pad. O component *gameObject* foi utilizado para o uso de imagens externas, ou seja, imagens pré-redenrizadas que foram utilizadas no jogo e administração das velocidades e posições dos objetos (naves e bola). O componente animation foi utilizado para manter uma atualização constante do desenho e das atualizações, e por fim o uso do Pad. O Pad foi utilizado para abstrair o dispositivo de controle utilizado, pois este jogo teve versões que suportavam o uso de um teclado comum e de um mouse no lugar do controle remoto.

Em uma versão posterior, foi adicionado ao jogo o componente Net, que possibilitou a interação entre dois usuários, onde um controlava o game a partir do controle remoto e outro controlava a outra nave via canal de retorno.

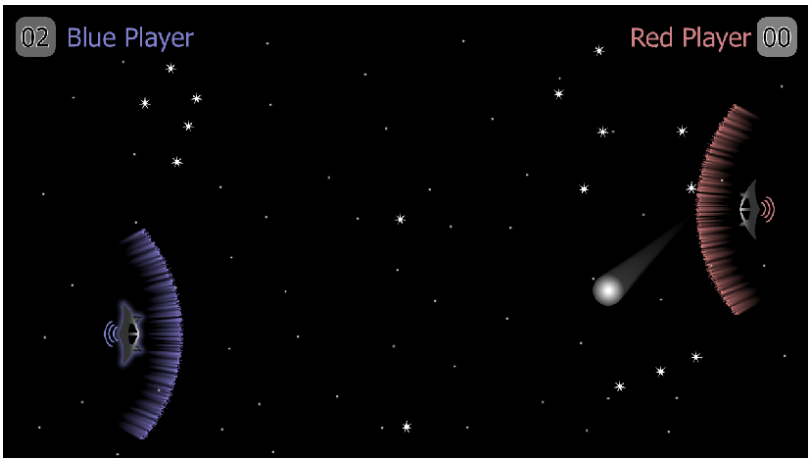


Figura 23- Jogo GingaCraft.

Uma grande contribuição do desenvolvimento deste jogo ao framework foi o surgimento de oscilações do jogo na tela, o flicker. Isto ocorreu devido ao aumento da taxa de desenho na tela de 10fps (no asteroids4TV) para 30fps (no GingaCraft) e ao uso de imagens maiores, pois o GingaCraft se utilizava da resolução HD (1920 x 1080 pixels), enquanto o asteroids utilizava uma resolução SD (640 x 480 pixels). Como não havia o uso da técnica de double buffer, a cada segundo eram feitas 180 atualizações de tela em alta resolução (6 componentes de jogo X 30 frames por segundo).

Com o surgimento deste problema, no ciclo seguinte o componente *engine* foi adicionado ao framework, de forma a não haver mais problemas como o apresentado nos testes.

5.3. Robowalker

O RoboWalker a princípio não pode ser considerado um jogo, pois não apresenta objetivos a serem concretizados pelo jogador. Na realidade, este aplicativo serviu para auxiliar no desenvolvimento e aperfeiçoamento de quatro componentes do ATHUS: *Pad*, *TileSet*,*TileMap* e *Sprite*.

Neste protótipo foi possível analisar um avanço no nível gráfico, devido ao uso das técnicas de tiles e sprites, possibilitando assim uma experiência mais robusta que a dos jogos anteriores. Na execução do Robo-Walker é possível ver a animação do personagem a partir dos Sprites, a construção do ambiente a partir de Tiles e a movimentação do cenário a partir da técnica de TileMapping. Com o uso destas técnicas o jogo se assemelha a jogos da geração 8 bits, o que não impressiona quando comparado a jogos de gerações mais recentes. Porém como as possibilidades e limites do desenvolvimento de jogos para a TV Digital ainda estão sendo vistos, podemos considerar um avanço sair de um jogo no estilo Pong para uma aplicação com animação de personagem e cenário. A Fig. 24 mostra uma imagem retirada da execução do jogo.

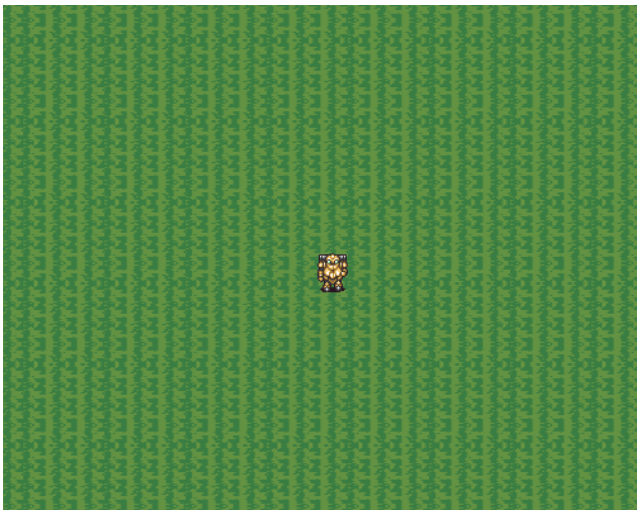


Figura 24 - Protótipo RoboWalker.

5.4. Poker4TV

O jogo Poker4TV é baseado no jogo de poker real, na modalidade Texas Hold'em. Ele foi desenvolvido para aperfeiçoar o módulo Net, de modo a possibilitar trocas de informações usando protocolos, WebService, e conexões diretas entre TVs.

O jogo possui uma aplicação servidora que esta localizada em um servidor web, aberto a conexões via sockets que podem ser feitas por qualquer dispositivo programável que possa ter uma versão do jogo e possua conexão em rede. Devido ao objetivo único de melhorar o módulo Net, nenhuma interface foi desenvolvida, ficando seus testes apenas em interface de linha de comando.

O servidor é responsável pelo controle de todo o jogo. Nele são gravadas cartas de cada jogador, analisando qual jogador possui a melhor sequência de cartas, controla também as fases do jogo e distribuição das cartas. Ao final é o servidor que diz quem venceu o jogo. Aos clientes cabe a responsabilidade de exibir as informações enviadas pelo servidor em arquivos xml, como a mão do jogador ao início do jogo, as cartas na mesa e o vencedor ao final.

5.5 GingaHero

O GingaHero é um jogo musical baseado em grandes sucessos comerciais como GuitarHero e RockBand. Ele foi desenvolvido para ser executado no Sistema Brasileiro de TV Digital (SBTVD) utilizando o middleware Ginga. Para seu desenvolvimento, foi necessário o uso de diversos componentes do Athus.

A arquitetura desenvolvida (vide Fig. 25) apresenta diversos componentes, que controlam animações, taxas de atualização dos jogos (frames por segundo e *updates* por segundo), *listeners* do controle remoto, tratamento de colisões, manipulação de áudio e vídeo, efeitos de animação através da técnica de *Sprites* e abstração dos comandos de mais baixo nível. Os módulos utilizados do Athus estão marcados em azul. Outros módulos específicos foram desenvolvidos para desenvolvimento do jogo, como um módulo música para controle das notas do jogo e um componente para controle de pontuação. O GingaHero foi o primeiro jogo a possuir um menu mostrando os comandos do jogo, desenvolvedores e inicio do jogo. A inicio pensou-se em adicionar ao Athus um componente para controle de menus, porém como foi possível desenvolver o menu apenas utilizando sprites, imagens e texto, este componente foi descontinuado.

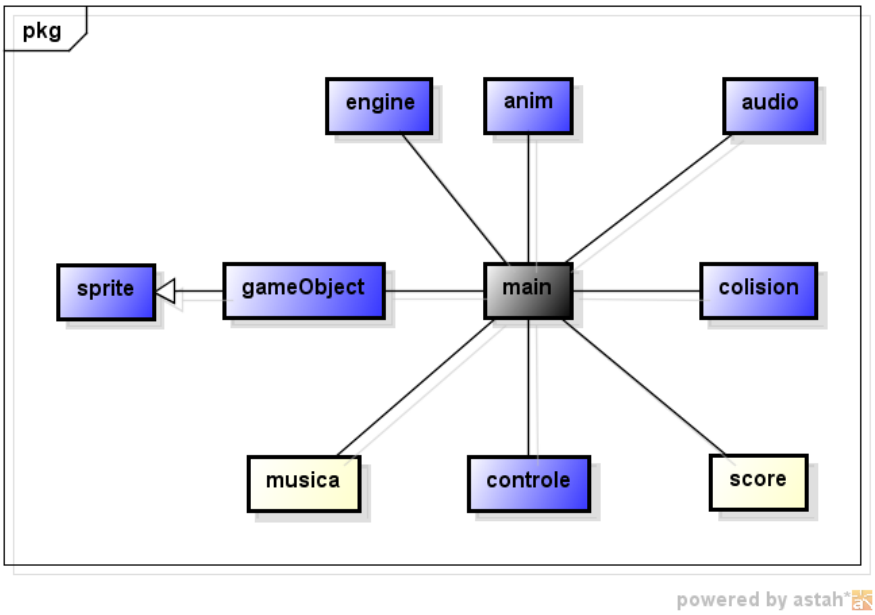


Figura 25- Classes GingaHero. Em azul componentes do Athus, em amarelo componentes específicos do jogo e em preto núcleo lógico do jogo.

A jogabilidade do GingaHero é muito semelhante ao dos jogos de gênero musical, onde o jogador deve acompanhar o ritmo pressionando o botão adequado para cada situação.

Ao jogar o GingaHero, um braço de uma guitarra é estendido e exibido na tela verticalmente, estando as hastes na posição horizontal. Com o avanço da música, as notas, representados por marcadores coloridos, indicam na tela, qual botão do controle deve ser pressionado, acompanhando assim o ritmo da canção de acordo com a sonoridade da guitarra. Quando a nota (marcador colorido) chega ao fim de seu curso, passando no local indicado onde a nota deve ser precisamente pressionada (indicado por círculos), o jogador deve pressionar o botão corresponde em seu controle (independentemente de qual seja o controle usado). Em cada nota pressionada corretamente, o som da nota da guitarra correspondente ao marcador colorido é executado e o jogador arrecada pontos que são totalizados no canto superior do jogo. Em cada nota errada, o som da nota correspondente a que devia ser pressionada, não é executado, tendo assim uma "falha" na música.

As notas musicais são indicadas por marcadores coloridos. Cada nota musical real é indicada por uma cor diferente correspondente as cores do controle remoto (vide Fig. 26).

Figura 26 - Controle do GingaHero :
Vermelho: 1ª nota. Verde: 2ª nota; Amarelo: 3ª nota; Azul: 4ª nota

As notas podem surgir únicas, ou compostas de duas a quatro notas em conjunto, fazendo assim um acorde. Também podem haver notas prolongadas que exigem maior habilidade dos jogadores. Nas Figs. 27 e 28 podemos ver as telas resultantes do jogo.

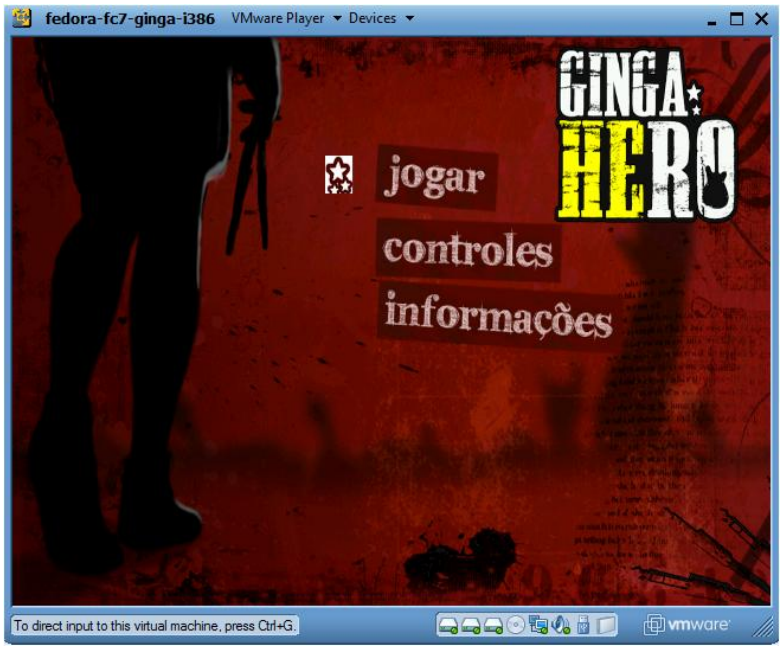


Figura 27- Menu do GingaHero

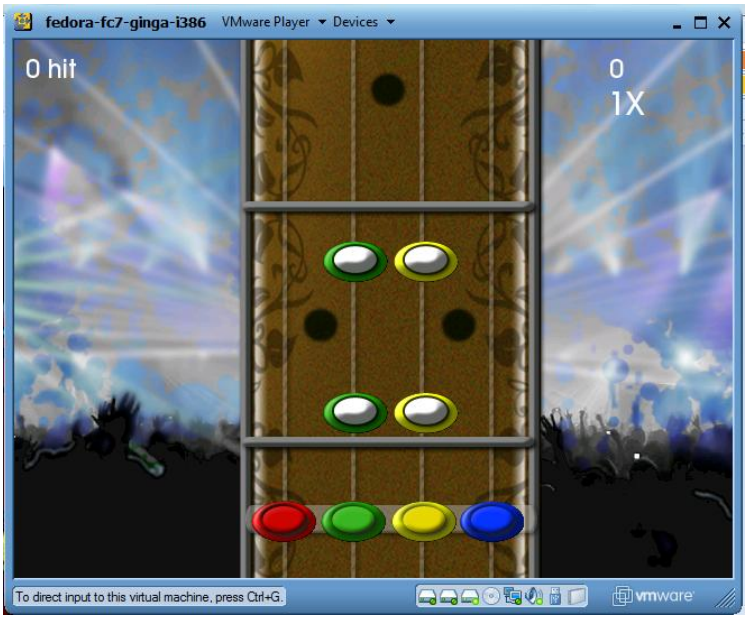


Figura 28- Interface GingaHero.

O GingaHero foi premiado como melhor aplicação do I CONCURSO LATINO-AMERICANO DE CONTEÚDO PARA TV DIGITAL INTERATIVA (Concurso, 2010), coordenado pelo Laboratório TeleMídia PUC-Rio (TeleMídia, 2011) na categoria Entretenimento. Fato muito animador, pois reconhece o potencial dos jogos como aplicações de entretenimento na TV e reconhece a qualidade do jogo produzido a partir do Athus.

5.6 Turma da Árvore

O jogo da Turma da Árvore foi o primeiro jogo desenvolvido que possui relação com o conteúdo transmitido junto à aplicação. O programa relacionado possui o mesmo nome da aplicação, Turma da Árvore. Trata-se de um programa de bonecos cujas personagens principais são três crianças, um quadro negro e um cachorro. Essas personagens se encontram em uma casa de madeira, situada em cima de uma árvore, no meio de uma praça. O encontro em uma casa suspensa faz com que as personagens vejam o mundo de outra perspectiva. Eles observam as situações do cotidiano que ocorrem em torno da casa e comentam. (Tavares et.al., 2007).

Baseado no programa de TV Turma da Árvore, o jogo utiliza um personagem do programa (Anaximandro) como personagem do jogo e sua temática é proposta baseada em assuntos utilizados no programa infantil, como a temática da reciclagem.

O jogo objetiva fazer a reciclagem do lixo que aparece na tela, e a cada lixo coletado, o ambiente é atualizado, retirando o lixo do cenário e fazendo a árvore também presente no cenário crescer.

O jogo funciona da seguinte forma:

- Ao iniciar a aplicação a tela inicial do jogo surge (vide Fig. 29), pedindo para o telespectador apertar 'ok' para iniciar a partida;

Figura 29- Tela inicial do jogo

- Após iniciada a partida, o jogador tem como objetivo colocar cada lixo na lixeira adequada: plástico na lixeira vermelha, papel na azul, vidro na verde e metal na amarela.
- Para isso o jogador pode selecionar entre os diferentes lixos através das setas esquerda e direita (vide Fig. 30). Após selecionado o lixo, o jogador deverá selecionar o lixeiro correto, com as setas ou os botões coloridos do controle que correspondem a cada lixeira. Caso acerte a combinação o lixo irá desaparecer e a árvore irá crescer. Caso erre, receberá um não do Anaki e tentará de novo até acertar, reiniciando o ciclo.

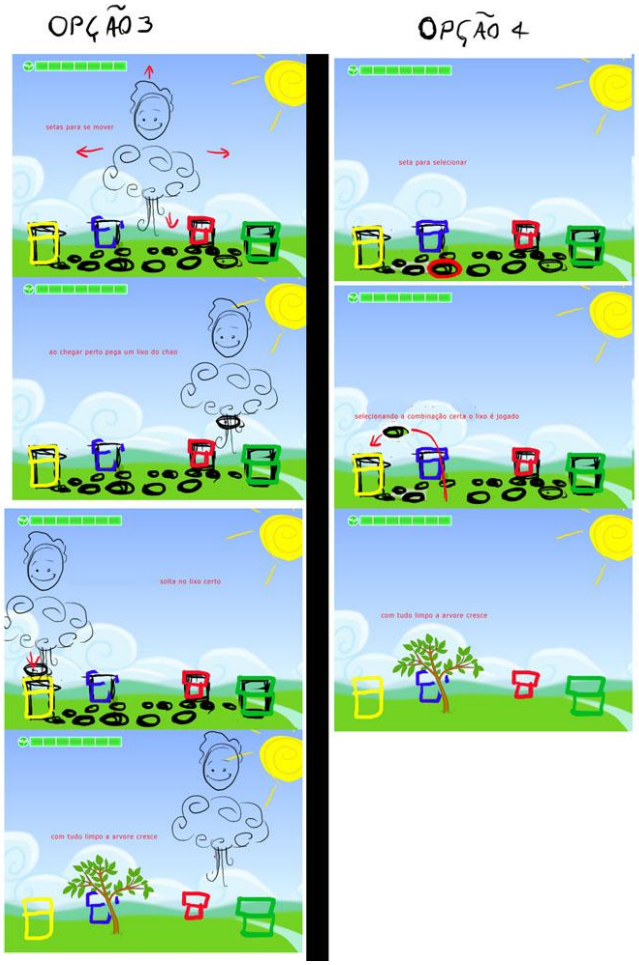


Figura 30- Storyboard do jogo Turma da Arvore.

- Após limpar todos os lixos, o jogador fará a árvore crescer até o fim e receberá um parabéns do Anaximandro. Então após 10s o jogo encerrará automaticamente.

A Fig. 31 mostra o resultado final do desenvolvimento.

Figura 31 - Jogo da Turma da Árvore.

Os seguintes componentes foram utilizados para a construção do jogo:

- GameObject: cada uma das cestas de lixo correspondem a uma gameObejct, assim como a imagem de fundo utilizada como cenário;
- Sprite: cada lixo na tela possui dois estados, selecionado e solto, possuindo uma imagem para cada estado, assim é utilizado um sprite não para fazer a animação, mas para mostrar a seleção do jogador. Já a árvore e o personagem são sprites e são utilizados para animação, a árvore cresce ao passar do jogo e o personagem fica em constante movimento em sua nuvem;
- Pad: o estado ativo do pad foi utilizado neste jogo. Ao pressionar uma tecla, a função de atualização do jogo é imediatamente chamada de forma a atualizar o mundo para ser redesenhado na próxima iteração do loop;
- Engine: utilizado para evitar problemas de exibição na tela;
- Colision: é utilizado durante as animações para averiguar se os objetos já chegaram a suas posições, como quando um lixo selecionado é solto na lixeira;

Não foi necessário nenhum módulo extra para o desenvolvimento do jogo, sendo utilizados apenas os módulos do Athus.

Capítulo

6

Validação

Neste capítulo é apresentada a metodologia utilizada na validação do framework Athus. No intuito de validar o framework proposto em termos de sua utilização prática propomos uma metodologia para avaliar o uso do ATHUS por desenvolvedores.

6.1. Metodologia Proposta

Durante a pesquisa, não foram encontradas metodologias que abordassem como validar frameworks através de testes com usuários. Desta forma, foram coletadas algumas experiências metodológicas acerca de frameworks e processos envolvendo testes com usuários de forma semelhante ao desejado, destacando-se os trabalhos: [(Cagnin,2005), (Camargo 2006), (Braga, 2002) e (Silva e Freiberger, 2008).

A metodologia proposta é dividida em três fases, como mostra a Fig. 32, e cada fase possui suas respectivas etapas de execução. A Fase I consiste na elaboração, a Fase II na execução e, por fim, a Fase III é a etapa de análise dos resultados.

Figura 32 - Fases da metodologia

6.1.1 Elaboração

Na fase de elaboração são definidos os aspectos específicos de cada teste, tais como os formulários a serem utilizados na pesquisa e os parâmetros específicos do domínio do framework. A Fig. 33 mostra um detalhamento desta fase.

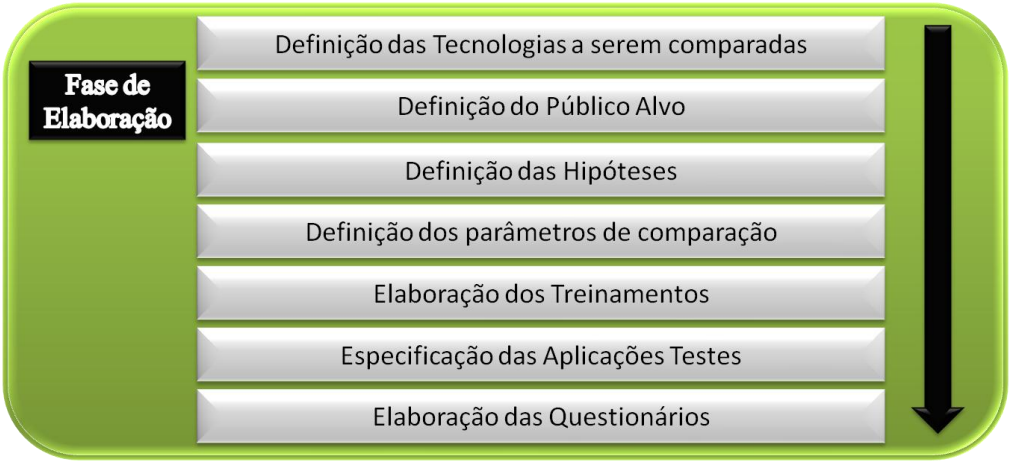


Figura 33 - Etapas da fase de elaboração

Durante a definição das tecnologias a serem comparadas são definidas quais tecnologias serão utilizadas nos testes, ou seja, quais outras técnicas serão comparadas ao framework. Devem ser selecionadas tecnologias que proporcionem funcionalidades semelhantes ao framework, para que seja possível implementar aplicações semelhantes com ambas tecnologias, de modo a compará-las em aspectos quantitativos e qualitativos.

Definir o público alvo para realização dos testes consiste em detalhar a quem se destinam as tecnologias e qual deve ser o perfil dos participantes que irão fazer parte do experimento a ser realizado, de forma que a avaliação deste grupo seja relevante para validar as hipóteses a serem levantadas.

O próximo passo é definir as hipóteses acerca do uso do framework que devem nortear os testes. Elas servirão de base para etapas posteriores, como a definição dos parâmetros a serem utilizados nos testes e as repostas a serem encontradas na análise dos resultados.

> "Hipótese é o que se pretende demonstrar e não o que já se tem demonstrado evidente, desde o ponto de partida, pois, neste caso, já não há mais nada a demonstrar, e não se chegará a nenhuma conquista e o conhecimento não avança." (Severino, 2002).

Exemplos de hipóteses que podem ser consideradas no contexto de frameworks são:

- O uso do framework facilitou o desenvolvimento das aplicações testes pelos programadores.
- O uso do framework reduziu o tempo de desenvolvimento das aplicações testes pelos programadores.
- O uso do framework reduziu o número de linhas escritas no desenvolvimento das aplicações testes pelos programadores.
- Os programadores preferiram utilizar o framework a outras tecnologias.

Outros questionamentos que podem ser investigados como hipóteses alternativas são:

- O uso do framework dificultou o desenvolvimento das aplicações testes pelos programadores.
- O uso do framework aumentou o tempo de desenvolvimento das aplicações testes pelos programadores.
- O uso do framework aumentou o número de linhas escritas no desenvolvimento das aplicações testes pelos programadores.
- Os programadores preferiram utilizar as tecnologias comparadas ao framework.

Os parâmetros de comparação devem ser definidos com antecedência. Parâmetros qualitativos serão utilizados para medir a satisfação dos usuários, no caso programadores, em relação ao uso do framework. Desta forma é possível saber se o framework satisfaz os requisitos do programador ou se mudanças devem ser feitas para melhorar o seu uso.

Exemplos de parâmetros que podem ser explorados são:

- Facilidade de uso;
- Facilidade de aprendizado;
- Satisfação com a documentação do framework;

Parâmetros quantitativos também devem ser definidos de forma que a eficácia do framework em relação às outras tecnologias possa ser mensurada. Alguns exemplos são:

- Quantidade de linhas de código escritas;
- Quantidade de Unidades Criadas (classes, aspectos, pacotes, páginas Web, etc.);
- Tempo de desenvolvimento das aplicações teste;

Treinamentos serão necessários para que os participantes se familiarizem com as tecnologias que serão utilizadas. É a partir dele que os participantes criarão os alicerces necessários para o desenvolvimento das aplicações que serão propostas. Inicialmente, devem-se especificar os tópicos e subtópicos a serem abordados em cada treinamento. Logo em seguida, são preparados os cursos a serem ministrados aos participantes.

As aplicações a serem desenvolvidas pelos participantes dos testes devem ser previamente especificadas, de forma a gerar documentação suficiente para que os programadores possam durante os testes compreender sem dificuldades o programa a ser implementado. Exemplos de documentos que podem ser gerados são: documentos de especificação de requisitos e documentos de arquitetura de software. Documentos específicos de uma área de conhecimento também podem ser utilizados, como *o Game Development Document* (GDD) para o domínio dos jogos.

Outra etapa é a elaboração de questionários. Os questionários servem para coletar dados dos participantes do teste.

Propomos a utilização de um questionário para coleta de informações pessoais que consiste em coletar informações diversas sobre o participante, como nome, escolaridade, idade e outros aspectos pessoais. Informações técnicas também são abordadas para descobrir o grau de conhecimento dos participantes, de forma a averiguar a experiência do programador no domínio do framework, sobre linguagens de programação utilizadas no contexto e experiências dos programadores. O questionário qualitativo abordará os participantes de forma a coletar informações acerca da satisfação e facilidade de uso provida pelo framework. Por fim é necessária a criação do termo de consentimento, onde os participantes acordarão com todos os termos legais descritos para o uso das informações coletadas através dos testes.

6.1.2 Execução

A fase de execução corresponde à fase de aplicação dos testes, utilizando os artefatos gerados na fase de elaboração. Suas etapas são vistas na Fig.34.

Figura 14 - Etapas da fase de execução

Os participantes podem ser recrutados através de uma chamada pública destinada ~~ao~~ interessados na área abordada. Na chamada é interessante manter segredo dos reais objetivos da pesquisa, de forma a evitar qualquer ação que possa comprometer os resultados dos testes, ou seja, os participantes irão apenas receber o treinamento nas tecnologias apresentadas na chamada pública.

O framework e as tecnologias a serem utilizadas possuirão pontos em comum, desde o domínio que abrange a construção de jogos, até a linguagem de programação utilizada. Sendo assim, na fase de treinamento devem ser apresentados e ensinados aos programadores os fundamentos básicos em comum das aplicações, servindo também para que os pesquisadores já observem quais sujeitos possuem experiência na área e nas ferramentas utilizadas. Após o treinamento inicial, deve ser aplicado um questionário com o objetivo de identificar cada um dos testadores coletando dados pessoais como nome, idade, sexo e outros. Neste questionário

dados técnicos são também coletados de forma a identificar a experiência dos programadores em áreas comuns aos programas a serem testados, e o grau de instrução deles, de forma a avaliar quem seriam os melhores sujeitos para avaliar os softwares.

Existe a possibilidade de diversos indivíduos se apresentarem para os testes. Dentre estes, alguns podem não estar dentro do público alvo definido na fase de elaboração, sendo assim, os questionários pessoais de cada individuo devem ser analisados para definir se ele pertence ao público alvo ou não. O termo de consentimento é aplicado aos participantes que permanecerem, pois este é uma obrigação para pesquisa e para exercícios profissionais envolvendo seres humanos e representa o respeito à autonomia. Assim, antes que os participantes sejam submetidos a algum processo de avaliação, deve-se aplicar um termo de consentimento para que eles autorizem todo e qualquer resultado dos testes.

A divisão dos participantes em grupos ocorre para que cada ferramenta possa ser testada, aplicada e avaliada paralelamente. Com isto é possível avaliar se a ordem de treinamento influência de alguma maneira os participantes.

Na divisão dos grupos, devem-se considerar os resultados da filtragem dos participantes, de forma a gerar grupos que possuam membros semelhantes em cada grupo, ou seja, ambos os grupos devem possuir programadores experientes e programadores menos experientes, mantendo um nivelamento entre os grupos.

Na primeira etapa do treinamento específico, cada grupo recebe um treinamento em uma ferramenta específica, ou seja, no caso de dois grupos, enquanto o grupo A recebe treinamento na ferramenta I, o grupo B recebe treinamento na ferramenta II. Para realização do desenvolvimento da primeira aplicação após este treinamento, os grupos recebem a especificação de uma mesma aplicação a ser desenvolvida, de forma assistida, com a tecnologia apresentada no treinamento feito na etapa anterior. É utilizada a mesma aplicação pelos dois grupos para que possa ser feita futuramente uma análise quantitativa entre as aplicações.

Após a etapa de desenvolvimento cada grupo recebe um questionário para avaliar a experiência de desenvolvimento do software da etapa anterior, relacionando o desenvolvimento à tecnologia utilizada (questionário qualitativo).

Após a realização da primeira etapa, ocorre a segunda etapa do treinamento específico e os grupos chaveiam as tecnologias utilizadas anteriormente. Quem recebeu treinamento na

tecnologia I agora recebe na II e vice-versa. Logo em seguida, os grupos recebem a especificação de uma nova aplicação a ser desenvolvida, de forma assistida, com a tecnologia apresentada no novo treinamento. Novamente, os dois grupos desenvolvem a mesma aplicação para que possa ser feita uma análise quantitativa entre as aplicações desenvolvidas.

Cada grupo recebe um questionário para avaliar a experiência de desenvolvimento do software da etapa anterior, relacionando o desenvolvimento à tecnologia utilizada.

Por fim, um novo questionário é entregue de forma a fazer com que o participante compare as tecnologias utilizadas durante as duas etapas de desenvolvimento realizadas. Através desse questionário questões de ordem comparativa devem ser abordadas.

6.1.3 Análise

Esta fase caracteriza-se por fazer a coleta e análise de todos os dados fornecidos pelos participantes, tanto nos questionários quanto nos resultados obtidos por eles, a fim de se verificar se as hipóteses especificadas na primeira fase estão de acordo com o esperado. As etapas desta fase são vistas na Fig. 35.

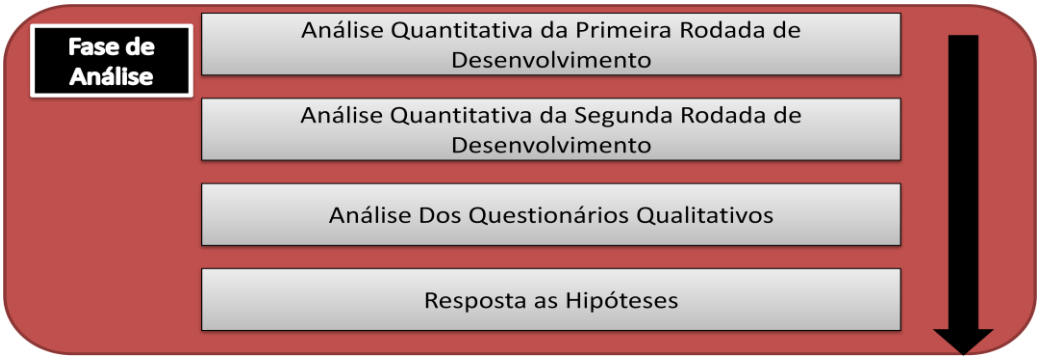


Figura 35 - Etapas da fase de execução

Durante a análise quantitativa da primeira rodada de desenvolvimento são avaliados os códigos produzidos pelos participantes na primeira etapa de desenvolvimento. É necessário levar em consideração os parâmetros quantitativos levantados na etapa de elaboração para fazer esta análise.

Durante a análise quantitativa da segunda rodada de desenvolvimento são avaliados os códigos produzidos pelos participantes na segunda etapa de desenvolvimento. É

necessário levar em consideração os parâmetros quantitativos levantados na etapa de elaboração para fazer esta análise.

É importante lembrar que nesta etapa os programadores adquiriram experiência no domínio da aplicação após a realização da primeira fase de programação, podendo este fato ter efeito nesta etapa.

A partir dos testes efetuados pelos participantes, é feita uma análise qualitativa das opiniões que foram fornecidas por eles nos questionários qualitativos. Com isso, leva-se em conta a existência ou aparecimento de certa qualidade, que é esperada ou não. Ou seja, analisar os resultados de um experimento em que devem ser observados aspectos qualitativos significa perceber a existência e as alterações nesses aspectos.

Os dados coletados nos questionários qualitativos e dos dados quantitativos adquiridos resultam em uma avaliação para definir quais das hipóteses levantas na fase de elaboração foram confirmadas.

6.2. Validação do ATHUS com a Metodologia Proposta

Com a metodologia definida e detalhada na seção anterior, realizamos os testes de validação do Athus.

6.2.1 Elaboração

Duas tecnologias de desenvolvimento de jogos para o SBTVD foram utilizadas para os testes:

- A linguagem Lua utilizando os Módulos NCLua específicos para o Ginga, apresentados na norma;
- A linguagem Lua utilizando o framework Athus;

O público alvo do framework são programadores de aplicativos para TV Digital e interessados em iniciar a programar nesta plataforma, desenvolvendo aplicações avançadas com foco em jogos para o Ginga.

As seguintes hipóteses foram consideradas na fase:

- O uso do Athus facilitou o desenvolvimento das aplicações testes pelos programadores, em comparação aos módulos NCLua nativos do GINGA;

- O uso do Athus reduziu o tempo de desenvolvimento das aplicações testes pelos programadores, em comparação aos módulos NCLua nativos do GINGA;
- O uso do Athus reduziu o número de linhas de código escritas no desenvolvimento das aplicações testes pelos programadores, em comparação aos módulos NCLua nativos do GINGA;
- O programadores preferiram o utilizar o Athus, em comparação aos módulos NCLua nativos do GINGA;

E também as hipóteses alternativas:

- O uso do Athus dificultou o desenvolvimento das aplicações testes pelos programadores, em comparação aos módulos NCLua nativos do GINGA;
- O uso do Athus aumentou o tempo de desenvolvimento das aplicações testes pelos programadores, em comparação aos módulos NCLua nativos do GINGA;
- O uso do Athus aumentou o número de linhas de código escritas no desenvolvimento das aplicações testes pelos programadores, em comparação aos módulos NCLua nativos do GINGA;
- O programadores preferiram o utilizar os módulos NCLua nativos do GINGA, em comparação ao Athus;

Como parâmetros qualitativos foram utilizados:

- Facilidade de aprendizado: avaliar a facilidade dos participantes em aprender a utilizar as tecnologias avaliadas;
- Facilidade de uso: avaliar a facilidade dos participantes em utilizar as tecnologias avaliadas;
- Documentação: verificar se a documentação das tecnologias é suficiente para seu uso, ou se melhoras devem ser feitas;

Já os parâmetros quantitativos foram:

- Número de linhas de códigos escritas nas aplicações geradas;
- Tempo de desenvolvimentos das aplicações geradas;
- Funcionalidades desenvolvidas na aplicação;

O treinamento foi efetuado através de aulas expositivas e práticas, com duração total de vinte quatro horas, levando em consideração as aulas mais o desenvolvimento de aplicações pelos participantes. O treinamento foi realizado entre os dias três e cinco de agosto de 2011 com aulas durante um turno inteiro (quatro horas), intercalando pela manhã aulas teóricas e

pela tarde aulas práticas auxiliadas. O pré-requisito para o treinamento para os participantes foi possuir noção de algoritmos e estruturas de programação, para que apenas pessoas com capacidade de programação pudessem participar do curso. Já alguns conhecimentos eram desejáveis, pois constituiriam participantes mais experientes na área abordada. Estes conhecimentos foram: sobre TV Digital, linguagem de programação NCL, linguagem de programação Lua, aspectos de Orientação a Objeto.

Três treinamentos foram elaborados para serem realizados com os participantes: o primeiro consistiu em treinar os participantes na linguagem de programação lua mais a introdução a técnicas de desenvolvimento de jogos em duas dimensões. O segundo treinamento abordou os módulos de programação NCLua, seguido do treinamento prático onde os participantes utilizaram os conhecimentos adquiridos para construir um jogo. O terceiro treinamento abordou o Athus, e semelhante ao segundo foi seguido da prática dos participantes.

Para elaboração das práticas, dois documentos de especificação de jogos foram criados para serem utilizados como referências pelos participantes na etapa de desenvolvimento. Um documento detalhou o jogo Snake e o outro o jogo Arkanoid. Ambos podem ser vistos nos anexos A e B, respectivamente.

A coleta de informações pessoais foi feita através de formulário, criado utilizando a ferramenta GoogleForms do Google. A utilização do formulário online facilitou o preenchimento por parte dos participantes e a coleta dos dados por parte dos pesquisadores. O formulário criado pode ser encontrado no Anexo D.

Utilizando também formulários online, foi criado um formulário (Anexo E) para análise qualitativa das tecnologias. As perguntas abordadas foram:

1. Como você avalia o seu aprendizado da tecnologia utilizada?
2. Como você avalia a prática de uso da tecnologia utilizada no desenvolvimento do jogo?
3. O que você achou da documentação (API, manual, links) disponibilizada durante o curso?
4. Quais os pontos positivos no uso da tecnologia?
5. Quais os pontos negativos no uso da tecnologia?
6. Você já conhecia a lógica do jogo desenvolvido?
7. Comente sua experiência no desenvolvimento.

O termo de consentimento utilizado está disponível no Anexo C.

6.2.2 Execução

A fase de execução foi iniciada com o recrutamento dos participantes, através de uma chamada pública em diversas listas de discussão da área de computação de estado da Paraíba.

Os interessados se inscreveram no evento informando dados pessoais, email para contato, instituição de origem, motivo do interesse e outras informações através do formulário pessoal. Ao fim desta fase, que durou cinco dias, vinte pessoas se inscreveram para participar da experiência, e, consequentemente, dos cursos realizados.

Como planejado na fase de elaboração, o primeiro dia do evento foi voltado apenas às tecnologias em comum, no caso treinamento na linguagem de programação Lua e em técnicas de programação de jogos.

O questionário pessoal não necessitou ser aplicado após o treinamento geral, pois todas as informações necessárias já haviam sido coletadas no ato da inscrição de cada participante através do formulário de inscrição.

Devido a alguns participantes não residirem próximos ao local de realização do curso, foi tomada a decisão de fazer a filtragem dos participantes logo após as inscrições, para que algum participante não se deslocasse para assistir apenas um dia de curso.

Dos inscritos, apenas um não era da área de computação, sendo da área de comunicação e sem instrução para desenvolvimento de aplicações. Apenas cinco possuíam conhecimento em Lua, linguagem base do experimento. Em relação ao conhecimento metade dos inscritos informaram ter experiência no desenvolvimento de jogos, e todos que informaram não possuir experiência, relataram ter muito interesse em aprender mais sobre a área.

O número de participantes foi limitado pelo espaço físico do local do evento e pela quantidade de pesquisadores (três) que poderiam auxiliar os desenvolvedores durante o desenvolvimento dos jogos. Desta forma, foram selecionados dez participantes, onde as prioridades de seleção foram:

- Domínio da linguagem Lua;
- Experiência no desenvolvimento de jogos;
- Grau de experiência em programação;
- Grau acadêmico;

- Curso;

Os participantes foram informados imediatamente do resultado.

O termo de consentimento foi então aplicado ao inicio do segundo dia, quando foi explicado que os resultados do curso seriam analisados e utilizados na pesquisa. Nenhum participante se negou a assinar o termo. Então os participantes foram divididos em dois grupos (I e II), dividindo entre os grupos os participantes que já conheciam a linguagem de programação Lua, e os demais equilibrando os grupos de acordo com o nível de programação e experiência no desenvolvimento de jogos.

A primeira etapa do treinamento específico funcionou da seguinte forma:

- O grupo I recebeu treinamento no framework Athus;
- Enquanto o grupo II recebeu treinamento nos módulos NCLua;

O treinamento durou um turno, resultando no desenvolvimento de um pequeno jogo em cada grupo. No turno seguinte ao treinamento, foi apresentado aos dois grupos o GDD do jogo Snake. Cada membro de cada grupo teve então que desenvolver o jogo utilizando a tecnologia estudada no treinamento da etapa anterior, desta forma o grupo I desenvolveu o Snake utilizando o Athus e o grupo II utilizando NCLua.

Ao final da implementação, os códigos de cada participante foram coletados para análise posterior. Os participantes foram convidados a acessar questionário qualitativo e respondê-lo relacionando as perguntas ao desenvolvimento anterior. Um dos participantes relatou estar com pressa para sair e acabou por não preencher o formulário antes de deixar o laboratório.

No inicio do segundo dia, três dos dez participantes não apareceram para continuar o curso, dando assim procedência ao curso com sete participantes.

A segunda etapa do treinamento específico funcionou da seguinte forma:

- O grupo II recebeu treinamento no framework Athus;
- Enquanto o grupo I recebeu treinamento nos módulos NCLua;

O treinamento durou um turno, resultando no desenvolvimento de um pequeno jogo em cada grupo. No turno seguinte ao treinamento, foi apresentado aos dois grupos o GDD do

jogo Arkanoid (BlockBreacker). Cada membro de cada grupo teve então que desenvolver o jogo utilizando a tecnologia estudada no treinamento da etapa anterior. Desta forma o grupo I desenvolveu o Arkanoid utilizando NCLua e o grupo II utilizando o Athus.

Ao final da implementação, os códigos de cada participante foram coletados para análise posterior. Os participantes foram convidados a acessar o formulário que possuía o questionário qualitativo e respondê-lo relacionando as perguntas ao desenvolvimento anterior.

6.2.3 Análise e Resultados Obtidos

Com todos os testes realizados, códigos coletados e questionários respondidos foi possível iniciar a fase de análise dos resultados. O primeiro passo para tal, como previsto na metodologia, é a análise dos códigos gerados pelos desenvolvedores, na primeira e na segunda fase.

A Tabela 3 apresenta os resultados médios do primeiro dia de desenvolvimento.

Tabela 3 – Resultados do primeiro dia de desenvolvimento

Parâmetro	Athus	GingaNCL
Tempo de desenvolvimento	03h 40min	03h 41min
Número de linhas escritas	92 linhas	88 linhas
Porcentagem de funcionalidades desenvolvidas	59% das funcionalidades desenvolvidas	18% das funcionalidades desenvolvidas

Fazendo um comparativo entre as tecnologias, podemos perceber que no primeiro dia o uso do Athus permitiu que o grupo I programasse um número maior de funcionalidades em um tempo semelhante. O número de linhas de códigos escritas foi semelhante, porém, como um número maior de funcionalidades foram desenvolvidas com o Athus, podemos afirmar que em média uma linha de código do Athus gerou um número maior de funcionalidades. O mesmo pode ser dito em relação ao tempo gasto.

Em relação ao segundo dia, os resultados obtidos são mostrados na tabela 4.

Tabela 4 – Resultados do segundo dia de desenvolvimento

Parâmetro	Athus	GingaNCL
Tempo de desenvolvimento	03h 20min	02h 39min
Número de linhas escritas	78 linhas	99 linhas
Porcentagem de funcionalidades desenvolvidas	78% das funcionalidades desenvolvidas	55% das funcionalidades desenvolvidas

Observando os dados da segunda rodada de desenvolvimento foi notar com maior evidencia a diferença no número de linhas de código escritas utilizando NCLua puro e o Athus. O uso do Athus reduziu em média 20% do número de linhas de código escritas ao mesmo tempo em que possibilitou uma porcentagem maior das funcionalidades desenvolvidas. Já o fator tempo não pode ser devidamente avaliado nesta etapa pois os membro do grupo I deixaram o laboratório com certa antecedência devido a dificuldades no desenvolvimento (fato que pode ter impactado na porcentagem de funcionalidades desenvolvidas).

Analisando os resultados dos dois dias de desenvolvimento, vemos que o Athus em ambos reduziu o número de linhas de código escritas pelos desenvolvedores, como também permitiu o desenvolvimento de um número maior de funcionalidades em ambos os dias, e levando em consideração apenas o primeiro dia é possível avaliar que o Athus permitiu um desenvolvimento maior de funcionalidades em menor tempo.

O próximo passo a ser realizado foi a análise dos pontos qualitativos avaliados pelos participantes. Com os dados foi possível avaliar a satisfação dos participantes em relação ao uso do framework Athus e do uso de NCLua puro.

A primeira informação retirada dos questionários foi que apenas um participante já havia programado um dos jogos utilizados (o Snake), e todos os outros já conheciam os jogos por já os terem jogado.

A Fig. 36 mostra gráficos com as avaliações das tecnologias em relação à facilidade de aprendizado. Segundo o gráfico de satisfação o Athus possui no primeiro e segundo dia, respectivamente, notas: 8,75 e 8,25 enquanto o NCLua possui 6,8 e 7,7. Essas notas mostram que em ambos os dias o Athus foi melhor qualificado.

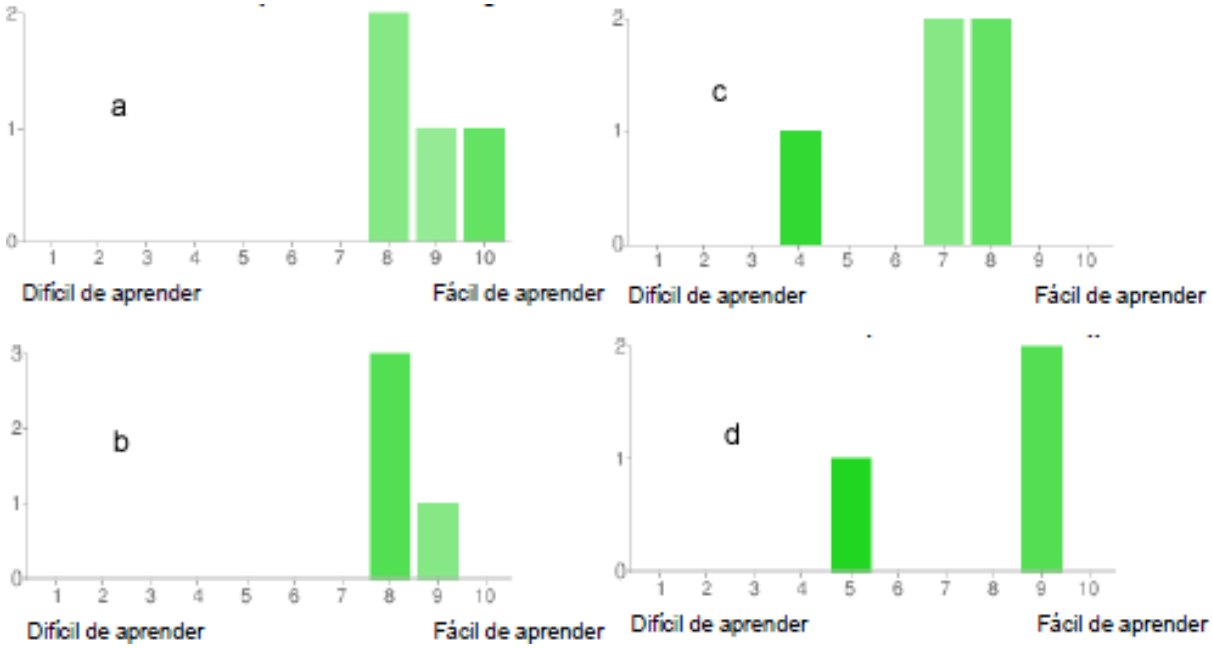


Figura 36 – Facilidade de aprendizado: a) Athus dia I, b)Athus dia II, c)NCLua dia I e d) NCLua dia II;

A Fig. 37 mostra gráficos semelhantes, só que relacionados à facilidade de uso das tecnologias. De acordo com o gráfico as notas do Athus foram: 8,0 e 8,5 enquanto NCLua puro obteve: 6,2 e 7,7. Novamente podemos ver que o Athus em ambos os dias obteve notas melhores.

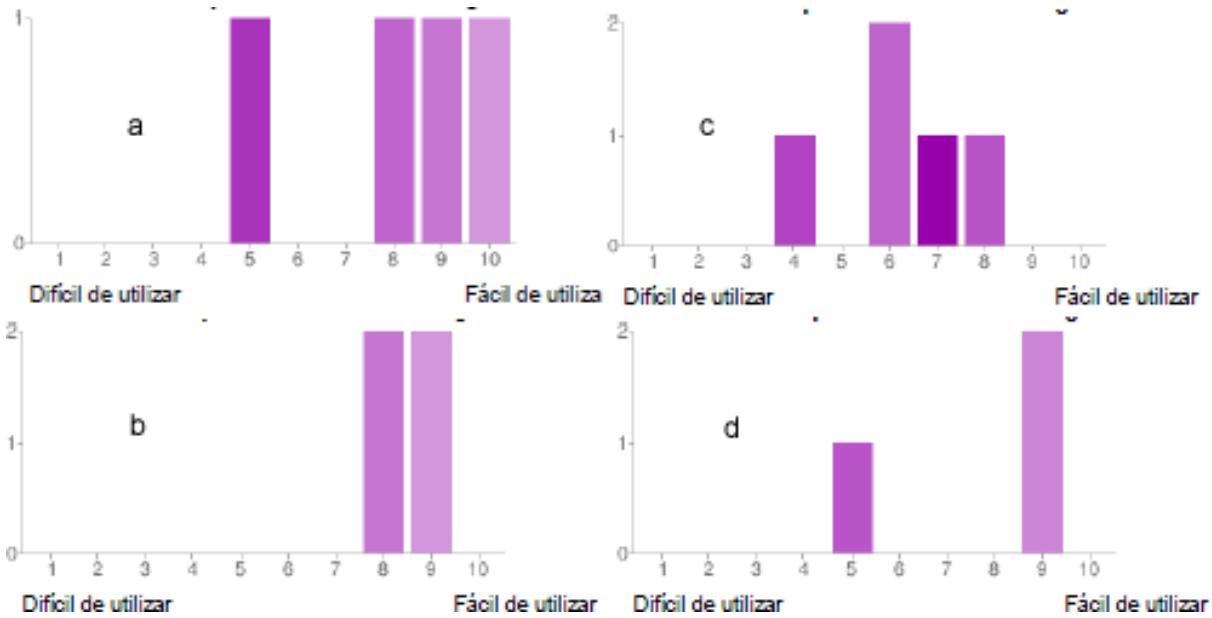


Figura 37 - Facilidade de uso: a) Athus dia I, b)Athus dia II, c)NCLua dia I e d) NCLua dia II;

Em relação à documentação das tecnologias avaliadas, as duas apresentaram em mais de 80% dos formulários como tendo a documentação “adequada para conhecimento básico”, e os outros 20% como “completa” e “completa e de fácil leitura”.

A última parte do questionário permitia que os participantes falassem de forma aberta sobre os pontos positivos e negativos das tecnologias e suas experiências durante o desenvolvimento.

Os principais problemas apontados sobre o Athus foram sobre inconsistências e falta de padrão na nomenclatura das funções e classes do framework. Outro problema apontado foi ausência de alguns métodos que ainda não vieram a ser implementados devido a limitações dos ambientes de execução das tecnologias.

Pontos positivos também foram destacados, como a facilidade de uso do framework, abstração do uso de técnicas para desenvolvimento de jogos como uso de colisão, sprites e outros.

Sobre a experiência no desenvolvimento, três participantes do grupo II compararam a experiência de uso do Athus com a experiência anterior de usar NCLua, relatando que foi mais fácil utilizar o Athus e que seu uso facilitou o desenvolvimento.

Os pontos negativos do uso do NCLua foram a dificuldade de utilizar a linguagem NCL, necessária para executar áudios nos jogos, a não presença do gerenciamento de colisões, problemas para criar loops de animação e a renderização dos gráficos dos jogos.

Pontos positivos do uso de NCLua foram sua orientação a eventos, e principalmente o uso da linguagem lua, que também está presente no uso do Athus.

Sobre a experiência no desenvolvimento usando o NCLua, o maior número de comentários foi sobre problemas no ambiente de desenvolvimento, problema também encontrado no uso do Athus.

De acordo com as análises feitas com os códigos e formulários, é possível dizer que todas as hipóteses levantadas foram confirmadas, e são elas:

- O uso do Athus facilitou o desenvolvimento das aplicações testes pelos programadores, em comparação aos módulos NCLua nativos do GINGA;
- O uso do Athus reduziu o tempo de desenvolvimento das aplicações testes pelos programadores, em comparação aos módulos NCLua nativos do GINGA;

- O uso do Athus reduziu o número de linhas de código escritas no desenvolvimento das aplicações testes pelos programadores, em comparação aos módulos NCLua nativos do GINGA;
- O programadores preferiram o utilizar o Athus, em comparação aos módulos NCLua nativos do GINGA;

Capítulo

7

Considerações Finais

> "Se os jogos são capazes ou não de fazer alguém chorar não importa, pois o vídeo game continua sendo uma mídia poderosa, afinal de que outra forma uma pessoa pode se tornar herói com o apertar de um botão?" (Discovery, 2007).

Os jogos tomam cada vez mais espaço na sociedade, tanto no aspecto econômico, concorrendo com as maiores indústrias de entretenimento, quanto no aspecto social se tornando parte do cotidiano de milhões de pessoas. Os jogos podem envolver uma pessoa de forma que ela sinta e participe de uma história como se ele realmente participasse desta. Desta forma a pesquisa realizada possui grande importância, pois aproxima o mundo dos jogos com uma das formas de entretenimento mais populares no Brasil, podendo levar cada pessoa em sua casa a ter um pouco da experiência de ser um herói.

7.1. Resultados Obtidos

O principal objetivo da pesquisa foi o desenvolvimento de um framework para desenvolvimento de jogos para Televisão Digital através do middleware Ginga, de forma a possibilitar ao desenvolvedor customizar o framework para desenvolver aplicações específicas utilizando instancias das classes do framework. Este objetivo foi alcançado graças à realização de várias etapas, que vão desde investigar modelos de jogos para a TV Digital até a validação do Framework com programadores de forma a avaliar se o framework teria impactos positivos no desenvolvimento de jogos para o Ginga.

Dentre os modelos pesquisados, é possível ver que a TV não será uma concorrente para plataformas que já existem hoje, pois as restrições da plataforma Ginga não permitem

hoje a criação de jogos ao nível de jogos para consoles como PS3 e XBOX360, mas a TV permitirá jogos com características únicas quando estes foram relacionados a grade de programação das emissoras, algo que nenhuma outra plataforma de jogos pode oferecer.

A plataforma Ginga oferece dois subsistemas de programação distintos para os desenvolvedores: o Ginga-J e o GingaNCL, porém atualmente apenas o GingaNCL possui uma plataforma estável para desenvolvimento e testes de aplicações. Este fato, a abrangência do GingaNCL e da difundida adoção, leveza e rapidez da linguagem lua, já verificada no desenvolvimento de jogos, levou ao desenvolvimento do Athus utilizando apenas GingaNCL.

Sendo então desenvolvido sobre o GingaNCL, o Athus teve como base a linguagem da programação lua, uma linguagem bem difundida e muito utilizada no desenvolvimento jogos. A linguagem lua permitiu a criação de uma arquitetura modularizada do Athus, pois esta foi baseada no paradigma de orientação a objeto. A arquitetura permitiu fácil reutilização do código e técnicas implementadas no Athus como também permitiu aos desenvolvedores uma base inicial para o desenvolvimento dos jogos, sem a necessidade de estudos avançados sobre a linguagem NCL e os módulos NCLua, pois ambos são abstraídos pelo uso do Athus.

Durante o desenvolvimento do Athus, diversos jogos foram produzidos de forma a testar as funcionalidades providas pelo Framework. Um destes jogos recebe destaque devido ao seu reconhecimento internacional pelos desenvolvedores de aplicativos para o Ginga e por ter ganho o I Concurso Latino-Americano de Conteúdo para TV Digital Interativa, realizado em Outubro de 2010 durante o Simpósio Brasileiro de Sistemas Multimídia e Web.

O reconhecimento do GingaHero possibilitou a divulgação do framework Athus, sendo seu uso solicitado pelo *PLADEMA Research Institute* (Pladema, 2011) para servir de base para o projeto *Efectos visuales y gestión óptima del tráfico de datos en TV Dgital* em Maio de 2011.

Os código do GingaHero e do Athus foram hospedados no SourceForge (SourceForge, 2011) de forma que possam ser acessados por qualquer desenvolvedor interessado neles. Através do site foi possível extrair informações sobre os downloads dos softwares. A Fig. 38 mostra um gráfico com o número de downloads do GingaHero no último ano e a Fig. 39 mostra a localidade de origem destes downloads.

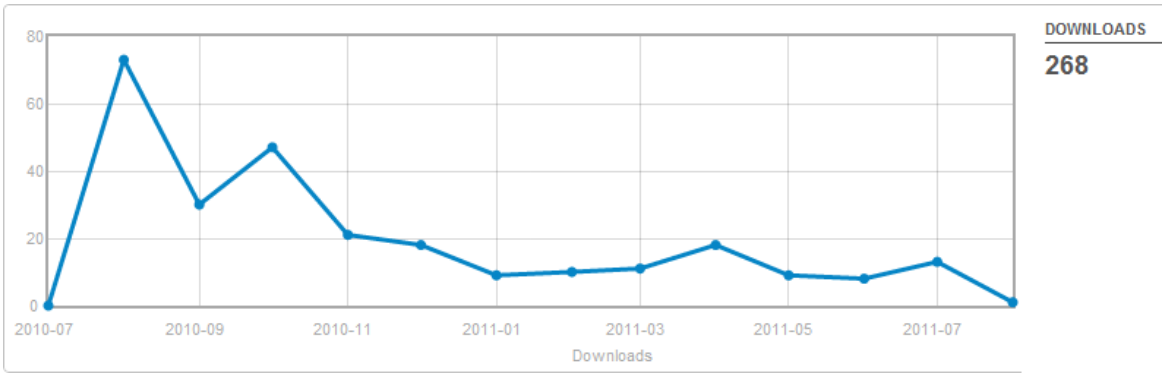


Figura 38 – Número de Downloads GingaHero

	Country	Downloads
1.	Brazil	214
2.	China	9
3.	Argentina	7
4.	Colombia	5
5.	France	5
6.	United States	5
7.	Portugal	4
8.	Satellite Provider	4
9.	Taiwan	4
10.	Chile	3
11.	Ecuador	3
12.	Saudi Arabia	1
13.	Italy	1
14.	Peru	1
15.	Uruguay	1
16.	Spain	1
		268

Figura 39 - Locais dos Downloads do GingaHero

Outra local de download do GingaHero é através do site de compartilhamento de aplicações para a linguagem NCL criado pela PUC-Rio. Não foi possível conseguir o número de downloads do GingaHero, mas é possível visualizar na página inicial do clube o

GingaHero como aplicação destaque do site após um ano de seu desenvolvimento, como destacamos na Fig. 40.

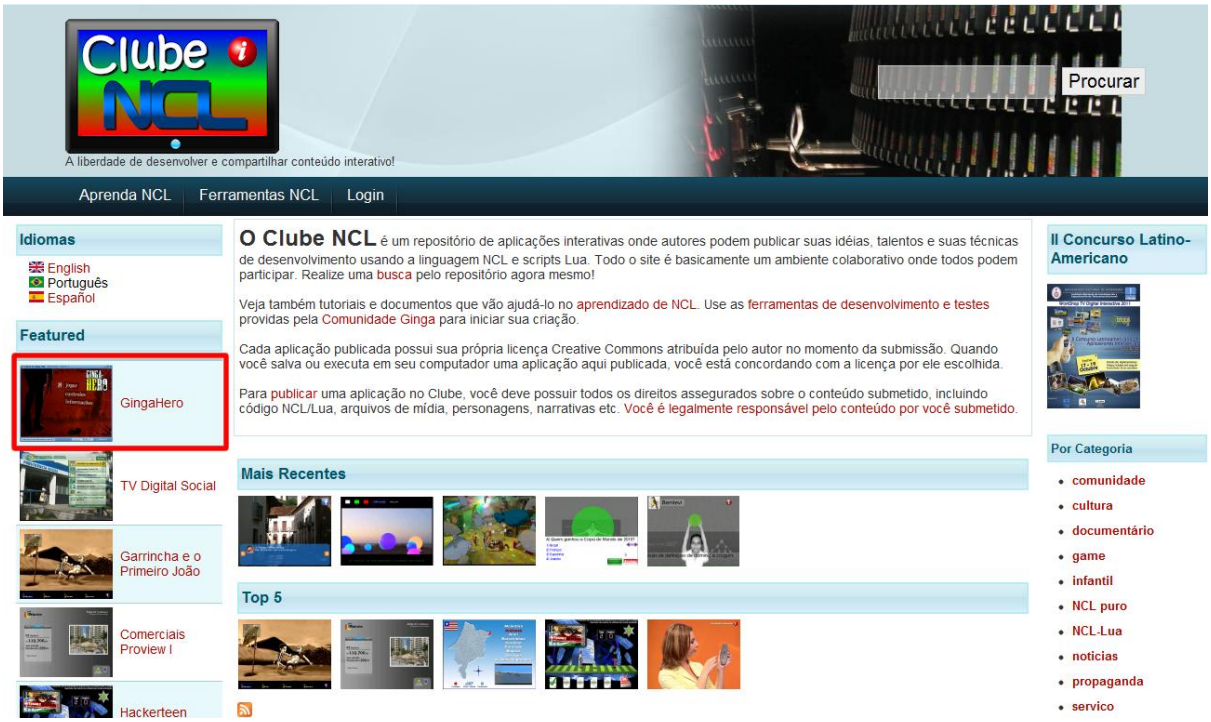


Figura 40- Aplicação GingaHero no ClubeNcl.

Os números de downloads do Athus são menos expressivos, alcançando em todas as suas versões disponibilizadas no SourceForge 130 downloads. A Fig. 41 mostra as principais origens destes downloads.

	Country	Downloads
1.	Brazil	96
2.	France	10
3.	Argentina	7
4.	United States	6
5.	Colombia	2
6.	Portugal	2
7.	China	1
		124

Figura 41 - Downloads Athus

Além do Concurso, os resultados da pesquisa foram publicados em diversos eventos acadêmicos. Estes trabalhos são listados abaixo:

- ATHUS: A Generic Framework for Game Development on Ginga Middleware. In: IX Simpósio Brasileiro de Jogos e Entretenimento Digital, 2010, Florianópolis, Santa Catarina. SBGames 2010 - Computing, 2010;
- ATHUS: um framework para desenvolvimento de jogos para TV Digital Brasileira. In: XVI Simpósio Brasileiro de Sistemas Multimídia e Web, 2010, Belo Horizonte - MG. WebMedia 2010 - WTD, 2010;
- Proposta de animação de jogos 2D para TV Digital. In: IX Simpósio Brasileiro de Jogos e Entretenimento Digital, 2010, Florianópolis, Santa Catarina. SBGames 2010 - Computing, 2010;
- DTV Games: new possibilities and challenges for sbtvd. In: IX Simpósio Brasileiro de Jogos e Entretenimento Digital, 2010, Florianópolis, Santa Catarina. SBGames 2010 - Computing, 2010;
- Jogos eletrônicos na TV Digital brasileira: gêneros e desafios para esta nova plataforma. 2010. Florianópolis, Santa Catarina. SBGames 2010.
- Desenvolvimento de Jogos para a plataforma Ginga utilizando NCLua. 2011. Salvador, Bahia. SBGames 2011.

Em relação aos trabalhos relacionados pesquisados, podemos retomar a Tabela 1 de comparação entre os mesmos, podemos modificá-la, adicionando agora uma coluna a mais, de modo a incluirmos na comparação o framework Athus, como mostrado na Tabela 5.

Na tabela é possível ver que o Athus cobre muitos parâmetros que nenhum dos trabalhos relacionados haviam abordado, como a sincronização de eventos junto a emissora, o uso do framework pelos diversos países que adotem o Ginga e a disponibilização do código e API para desenvolvedores interessados no seu uso.

Tabela 5 - Comparativo entre os trabalhos relacionados e o Athus

Parâmetro	iTVProject	FrameIDTV	TUGA	GingaGame	Athus
API	-	-	-	-	X
Arquitetura Modular	X	X	X	X	X
Algoritmos de desenho	X	X	X	X	X
IA	-	-	-	-	X
Anti-Flicker	-	-	X	-	X
Cenário de Jogo	-	-	X	-	X
Dados Remotos	-	X	-	-	X
Dispositivos de Interação	X	X	X	X	X
Execução de Áudio	-	-	X	-	X
Funcionalidades 3D	-	-	-	-	-
Internacionalização	-	-	-	-	X
Jogos como resultado	-	-	X	-	X
Normatização	X	X	-	X	X
Objetos elementares	-	-	X	X	X
Sincronização com Emissora	-	-	-	-	X
Sprites	-	-	X	X	X
Suporte a Animação	-	-	X	-	X
Tiles	-	-	X	-	X
Uso de Metodologias	-	X	-	-	X

Por último, destaca-se os resultados obtidos com os testes com programadores. Estes testes permitiram afirmar que o uso do Athus facilitou, acelerou e diminuiu o esforço para o desenvolvimento de jogos para a plataforma Ginga.

7.2 Trabalhos Futuros

Apesar dos vários resultados obtidos, muito pode ser feito a partir do estado atual do trabalho.

Inicialmente, baseado no resultado dos testes com os programadores, um refatoramento do código deve ser realizado a fim de tornar padrão a nomenclatura de classes e métodos do Athus. Outro ponto levantado pelos programadores é a dificuldade de encontrar os erros que ocorrem no código, problema comum ao Athus e o NCLua puro, desta forma um sistema de debug deve ser implementado para facilitar a correção de erros durante a implementação.

Algumas funcionalidades não foram totalmente desenvolvidas devido às plataformas de teste não estarem completas, sendo assim estas funcionalidades devem ser implementadas à medida que os ambientes de teste permitam a adição de novas funcionalidades. Exemplos disto são a parte de auto-geração de código NCL pelo framework e a construção de cenários por Tiles, que na versão mais recente do ambiente de teste até a presente data apresentaram problemas com as implementações criadas. Além de implementar funcionalidades já previstas, novas podem ser criadas a medida que mais jogos forem feitos na plataforma.

Outra pesquisa que deve ser feita acerca do Athus é sua utilização e adaptação para outros gêneros de aplicações para o Ginga, pois alguns dos componentes desenvolvidos podem ser utilizados no desenvolvimento de aplicações de outros gêneros, como exemplo disto podemos citar toda a camada de interação da arquitetura, que lida com características que todas aplicações de TV podem utilizar: dispositivo de interação, canal de interatividade e sincronização com a programação e muitos dos componentes de visão.

Um último estudo a ser realizado no Athus é investigar sobre a viabilidade de seu uso para o desenvolvimento de aplicações IPTV, já que o GingaNCL do Sistema Brasileiro de TV Digital virou Recomendação ITU-T para serviços IPTV (ITU H.761). No ensejo de utilizar fluxos via rede, também deverá ser explorada a possibilidade do Athus para implementar um cliente de GameClouding na plataforma Ginga.

Referências

ABNT (Norma Brasileira ABNT NBR 15606-2:2007). Disponível em <http://www.abnt.org.br/imagens/Normalizacao_TV_Digital/ABNTNBR15606-2_2007Vc_2008.pdf>.

ABRAGAMES (Associação Brasileira das Desenvolvedoras de Jogos Eletrônicos). A indústria brasileira de jogos eletrônicos - Um mapeamento do crescimento do setor nos últimos 4 anos, 2008. Disponível em < www.abragames.org/docs/Abragames-Pesquisa2008.pdf >

Armbrust, M. et all (2010). "A View of Cloud Computing", Berkeley Reliable Adaptive Distributed Systems Laboratory (RAD lab), University of California, USA.

Azevedo, T.. Entrevista: Bertrand Chaverot fala sobre a Ubisoft Brasil. 2008. Disponível em <http://jogos.uol.com.br/reportagens/ultnot/2008/06/24/ult2240u129.jhtm>. Acessado em: Janeiro 2010.

Barboza, D. C. e Gonzales E. W.. SBGames 2009 "Ginga Game : A Framework for Game Development for the Interactive Digital Television" .

BEACON. Disponível em < http://www.fatece.edu.br/projetos/projet1.asp >. Acessado em: Maio, 2011.

Big Brother. Disponível em < http://glo.bo/lFmJKz >. Acessado em: Maio, 2011.

BOURG, David; SEEMAN, Glenn. AI for Game Developers. O'Reilly, 2004.

Braga, R. T. V.. Um Processo para Cosntrução e Instanciação de Frameworks baseados em Linguagens de Padrões para um Domínio Específico. Digital Library of Theses and Dissertations of USP. São Carlos, 2002.

BSA (Business Software Alliance). 09 PIRACY STUDY. 2009. Disponível em <. http://portal.bsa.org/globalpiracy2009/studies/09_Piracy_Study_Report_A4_final_111010.pdf >. Acessado em: Março de 2011.

Burbeck S. Application programming in smalltalk-80: How to use model-view- controller (MVC), Technical Report, University of Illinois in Urbana-Cham- paign, 1992.

Cagnin, M. I. Tese de Doutorado: PARFAIT: uma contribuição para a reengenharia de software baseada em linguagens de padrões e frameworks. Digital Library of Theses and Dissertations of USP. São Carlos, 2005.

Camargo, V. V.. Tese de Doutorado: Frameworks transversais: difinições, classificações, arquitetura e utilização em um processo de desenvolvimento de software. Digital Library of Theses and Dissertations of USP. São Carlos, 2006.

Celes, W., Figueiredo, L. H. e Ierusalimschy, R., 2004. "A Linguagem Lua e suas Aplicações em Jogos". Disponível em:<http://www.tecgraf.pucrio.br/~lhf/ftp/doc/wjogos04.pdf>.

ClubeNCL. Disponível em: < http://clube.ncl.org.br/ >. Acessado em Agosto de 2011.

CONCURSO LATINO-AMERICANO DE CONTEÚDO PARA TV DIGITAL INTERATIVA, Laboratório TeleMídia PUC-Rio, 2010. Disponível em: < http://clube.ncl.org.br/node/82 >

Daniel Sánchez-Crespo Dalmau. Core Techniques and Algorithms in Game Programming. 2003.

Davison A.. Killer Game Programming in Java. 2005.

Discovery Channel. A Era do Vídeo Game. 2007.

DTV.org (Site oficial da Tv Digital Brasileira)Disponível em: <http://www.dtv.org.br/>Acessado em Maio de 2011.

EEDTV/VirtuaLabTV. Dísponivel em < http://virtualabtv.lavid.ufpb.br/ >. Acessado em: Maio de 2011.

ESA (Entertainment Software Association). Essential Facts About The Computer And Video Game Industry. Disponível em:< www.theesa.com/facts/pdfs/ESA_Essential_Facts_2010.PDF > Acessado em: Dezembro, Março de 2011.

ESBR (Entertainment Software Rating Board). Disponível em <http://www.esrb.org>. Acessado em: Novembro 2009.

Fernandes, J., Lemos, G., Elias, G.. Introdução à Televisão Digital Interativa: Arquitetura, Protocolos, Padrões e Práticas. JAI-SBC, 2004.

Ferreira, D. A. and Souza, C. T., 2009. TuGA: Um Middleware para o Suporte ao Desenvolvimento de Jogos em TV Digital Interativa. Centro Federal de Educação Tecnológica do Ceará. Disponível em: <http://code.google.com/p/tugasdk/downloads/detail?name=TuGA_Middleware.Jogos.TVDigital_v1.6.pdf >

FIAT. Jogo T-Racer. Disponível em <http://www.t-racer.com/>. Acessado em: Janeiro 2010.

Filho, G.L.D.S., Leite, L.E.C., and Batista, C.E.C.F. Ginga-J: The Procedural Middleware for the Brazilian Digital TV System. In Proceedings of J. Braz. Comp. Soc.. 2007, 47-56.

Flores N., Aguiar F.. "Patterns for understanding frameworks" October 2008, PLoP '08: Proceedings of the 15th Conference on Pattern Languages of Programs.

Gamma E., Helm R., Johnson R., and Vlissides J..Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley. October 1994.

Games For Change. Disponível em <http://www.gamesforchange.org/>. Acessado em: Dezembro 2009.

Games For Health. Disponível em <http://www.gamesforhealth.org>. Acessado em: Dezembro 2009.

Gamasutra.Disponível em < http://www.gamasutra.com >. Acessado em: Maio de 2011.

Gawlinski, M. Interactive Television Production. Oxford: Focal Press, 2003. 288p.

Ginga App Store. Dísponivel em < http://appstore.lavid.ufpb.br/ >. Acessado em: Maio de 2011.

GPL (*General Public License)*. Dísponivel em <http://www.gnu.org/copyleft/gpl.txt>.

Heliö S. e Järvinen A.. FiTV: GAMES AND GAMING FOR INTERACTIVE DIGITAL TELEVISION. November 2003. University of Tampere Hypermedia Laboratory.

Henrique S. L. Pequeno, George A. M. Gomes, Rossana M. C. Andrade, Jos\&\#233; N. de Souza, and Miguel F. de Castro. 2010. FrameIDTV: A framework for developing interactive applications on digital television environments. *J. Netw. Comput. Appl.* 33, 4 (July 2010), 503-511.

Iano, Y.; , "An improvement on the reception of Pal-M television signal using an additional simple delay filter," *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on* , vol.v, no., pp.V/49-V/50 vol.5, 19-22 Apr 1994.

IERUSALIMSCHY, R. . Programming in Lua, Second Edition. 2. ed. Rio de Janeiro: Lua.org, 2006. 326 p.

IGDA (International Game Development Association). Disponível em <http://www.igda.org/>. Acessado em: Janeiro 2010.

IGN Retro. Disponível em < http://retro.ign.com/>. Acessado em: Maio de 2011.

ITU H.761. Disponível em < http://www.itu.int/itu-t/aap/AAPRecDetails.aspx?AAPSeqNo=1894 >. Acessado em Agosto de 2011.

Jens F. Jensen. 2005.Interactive Television: New Genres, New Format, New Content. Proceedings of the second Australasian conference on Interactive entertainment.

Johnston, H. , Whitehead A.. 2009. Distinguishing Games, Serious Games, and Training Simulators on the Basis of Intent. Proceedings of the 2009 Conference on Future Play on @ GDC Canada.

Johnson RE. Frameworks=(components+patterns). Commun ACM 1997;40(10): 39–42. ISSN 0001-0782.

Kanode, C.M.,Haddad, H.M.. Software Engineering Challenges in Game Development."Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on".2009.

Korson,T and McGregor, J. "Understanding Object-Oriented: A Unifying Paradigm". Communications of the ACM 33,9 (Sept. 1990).

Mapa ISDB-Tb. Disponível em < http://bit.ly/irD0QV/ >. Acessado em: Abril 2011.

Menezes C.. TV Digital revoluciona maneira de assistir televisão, 2009. Disponível em < http://g1.globo.com/jornaldaglobo/0,,MUL1224980-16021,00-

TV+DIGITAL+REVOLUCIONA+MANEIRA+DE+ASSISTIR+TELEVISAO.html>. Acessado em: Abril 2011.

MPAA (Motion Picture Association of America). theatrical market statistics. Disponível em:<http://www.mpaa.org//Resources/653b11ee-ee84-4b56-8ef1-3c17de30df1e.pdf> Acessado em: Dezembro, Maio de 2011.

Montez, C., Becker, V. TV Digital Interativa: Conceitos, Desafios e Perspectivas para o Brasil. 2ed. Florianópolis: Ed. da UFSC, 2005. 200p.

Narayanasamy, V., Wong, K. W., Fung, C. C., and Rai, S. 2006. Distinguishing games and simulation games from simulators. *Computers in Entertainment.* 4, 2 (Apr. 2006), 9.

Oliveira R.M., Filho B.P. , FerF.R. , iTV project: an authoring tool for mhp and ginga-j based on a web environment, Proceeding of the 1st international conference on Designing interactive user experiences for TV and video, October 22-24, 2008, Silicon Valley, California, USA.

OpenGinga. Disponível em < http://ginga.lavid.ufpb.br/>. Acessado em: Maio de 2011.

Parker, R.. The Economics of Digital TV's future, In D. Gerbarg, The economics, technology and content of digital TV, 1999, USA: Kluwer Academic Publishers.

Peng, C. Digital Television Applications. Dissertation for the degree of Doctor of Science in Technology. University of Technology, Espoo, Finland. 2002.

PESSOA, C. A.. wGEM: um Framework de Desenvolvimento de Jogos para Dispositivos Móveis. Mestrado em Ciências da Computação da Universidade Federal de Pernambuco, em 2002.

Peter K. Kaiser. Critical Flicker Frequency. Disponível em <http://www.yorku.ca/eye/cff.htm>. Acessado em 2009.

Playin'TV. Disponível em <http://www.playinTV.com/TV>.

Plademа Institute. Disponivel em <http://www.plademа.net/tvdigital/>. Acessado em Agosto de 2011.

Pree W, Sikora H. Design patterns for object-oriented software development. In: ICSE ’97: proceedings of the 19th international conference on Software engineering. New York: ACM; 1997. p. 663–4.

Proview, disponível em <https://www.kabum.com.br/cgi-local/kabum3/produtos/descricao.cgi?id=01:01:04:118:30>. Acessado em Março de 2011.

SEGUNDO, R. M. C. ; MORAIS, A. M. ; Santana, Eduardo Freire ; BARBOSA FILHO, J. R. B. ; SILVA, J. C. F. ; Manuella Dias Carvalho da Silva ; TAVARES, T. A. . A Utilização de Novos Dispositivos de Interação em Programas Interativos para TV Digital: um Estudo de Caso baseado na tecnologia Sun SPOT. In: 1º Simpósio Internacional de Televisão Digital, 2009, Bauru-SP. 1º Simpósio Internacional de Televisão Digital, 2009.

Severino, A. J. Metodologia do trabalho científico. 22. ed. rev. e ampl. São Paulo: Cortez, 2002. 333

Schell J.. The Art of Game Design. 2008.

SILVA, R.P. e FREIBERGER E. C.Metrics to Evaluate the Use of Object Oriented Frameworks. Oxford University Press on behalf of The British Computer Society. 2008.

Soares, L. F. G. TV Interativa se Faz com Ginga. Revista da SET-Sociedade Brasileira de Engenharia de Televisão, 2009. p. 30 – 35.

Soares L.F..Construindo Programas Audiovisuais Interativos Utilizando a NCL 3.0 e a Ferramenta COmposer. 2007. Disponível em <http://www.ncl.org.br/documentos/TutorialNCL3.0-2ed.pdf>

SOARES, L. F. G. ; BARBOSA, Simone Diniz Junqueira . Programando em NCL. 1. ed. Rio de Janeiro: Elsevier - Campus, 2009. 483 p.

SourceForge. Disponível em <http://sourceforge.net/>. Acessado em Agosto de 2011.

Souza, C. e Santos C. .Sbgames 2009 Development of Games for SET-TOP-BOXES with Brazilian's Middleware GINGA-NCL.

TAVARES, T. A. ; SAIBEL, Celso Alberto ; ASSIS, Thiago Rocha de ; BRAGA, Clarissa Bittencourt de Pinho e ; MARIELLO, Germano ; COSTA, Clarissa S. . A TV Digital Interativa como Ferramenta de Apoio à Educação Infantil. Revista Brasileira de Informática na Educação, v. 15, p. 31-44, 2007.

TeleMídia PUC-Rio. Disponível em< http://www.telemidia.puc-rio.br/>, acessado em Maio de 2011.

TQTVD. Disponível em<http://www.tqtvd.com.br/br/produtos_stickercenter.html>, acessado em Março de 2011.

UOL. Disponível em < http://www.lavid.ufpb.br/projetos/visualizar/tvda-ambiente-de-desenvolvimento-teste-de-aplicacoes-para-tv-digital >. Acessado em: Maio, 2011.

Ubisoft. Jogo Mirro's Edge 2D. Disponível em <http://www.mirrorsedge2d.com/>. Acessado em: Janeiro 2010.

UOL. Para EA, games já são mais importantes que cinema. Disponível em <http://jogos.uol.com.br/pc/ultnot/2008/04/14/ult3277u15576.jhtm>. Acessado em: Janeiro, 2010.

UOL. Ibope da TV paga, DVD e game já supera Band e Rede TV!. Disponível em < http://noticias.uol.com.br/ooops/ultimas-noticias/2010/05/11/ibope-da-tv-paga-dvd-e-game-ja-supera-band-e-rede-tv.jhtm>. Acessado em: Fevereiro, 2011.

Vit Vrba, Lubomir Cvrk, and Martin Sykora. 2006. Framework for digital TV applications. InProceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies(ICNICONSMCL '06). IEEE Computer Society, Washington, DC, USA, 184-. Weiser, Mark. The Computer of the 21st Century. Scientific American 265, n. 3, setembro 1991. pp. 94-10.

Você Decide. Disponível em < http://bit.ly/mB420L >. Acessado em: Maio, 2011.

Wink Dink and You. Disponível em < http://www.tvparty.com/requested2.html >. Acessado em: Maio, 2011.

Anexo A

Game Development Document do Jogo Snake

LABORATÓRIO DE VÍDEOS E APLICAÇÕES DIGITAIS

GDD - Snake

Version 0.1

Ricardo Mendes Costa Segundo

12 de Julho de 2011

Sumário

Game Overview

Jogo baseado no clássico Snake do Atari:

"Snake é um jogo que ficou conhecido por diversas versões, em vídeo-games e computadores. No fim dos anos 90 foi incluso em alguns celulares. O jogador controla uma longa e fina criatura que se arrasta pela tela, coletando comida (ou algum outro item), não podendo colidir com seu próprio corpo ou as "paredes" que cercam a área de jogo. Cada vez que a serpente come um pedaço de comida, seu rabo cresce, aumentando a dificuldade do jogo. O usuário controla a direção da cabeça da serpente (para cima, para baixo, esquerda e direita) e seu corpo segue."

Game Play Mechanics

O jogo possui quatro elementos básicos: a comida, a cabeça da serpente, o corpo da serpente, e o mundo.

1. A COMIDA:

A comida é simbolizada por um quadrado. Apenas uma comida aparece na tela por vez, podendo ter um contador regressivo para que ela mude de posição caso ela não seja atingida pelo jogador. Quando atingida, ela aparece em outro local de forma randômica.

2. *HEAD*

A *head* esta em movimento constante, seguindo a direção ditada pelo jogador ao apertar uma das setas do controle remoto, mudando assim sua trajetória. A *head* pode colidir tanto com a comida (pontuação aumenta) quanto com o resto do corpo (jogo encerra).

3. O CORPO

Para cada comida acertada pelo jogador, o corpo aumenta em um bloco. Todos os blocos devem seguir o mesmo caminho percorrido pela *head*, desta forma a posição de um bloco do corpo n será igual ao bloco n-1, o n-1 igual ao n-2, e assim por diante, até chegar a *head* que se desloca em 20 pixels na direção atual.

4. O MUNDO

O mundo constitui os cantos laterais, superior e inferior da tela, delimitando até onde a esfera pode ir. Caso a *head* colida com algum destes, o jogo é encerrado.

Controls

Os controles do jogo se limitam as quatro setas no controle remoto: direita move a serpente para a direita, esquerda para esquerda, cima para cima e baixo para baixo.

Interface

A interface é apresentada na figura 1.

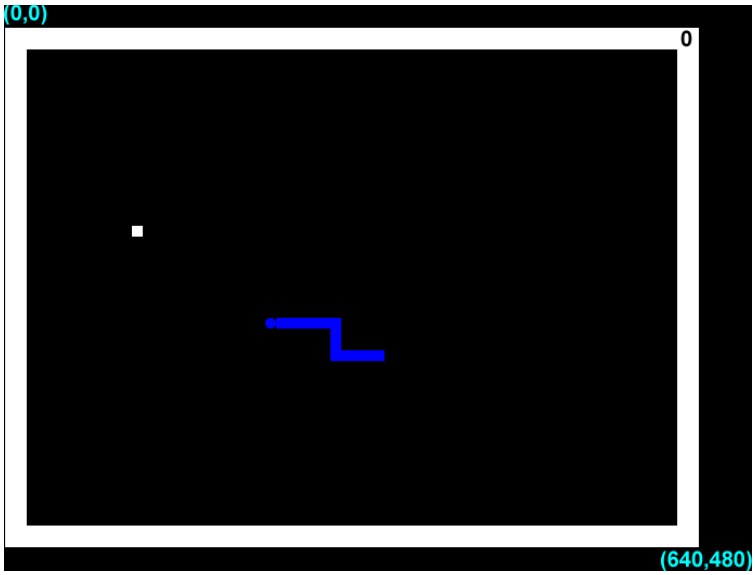


Figura 2- Interface com coordenadas

A figura apresenta as coordenadas extremas do jogo. As barras são sempre apresentadas nos extremos. Já os outros componentes estão sempre no espaço interior ao limitado pelas barras.

Abaixo (figura 2) a tela sem as coordenadas.

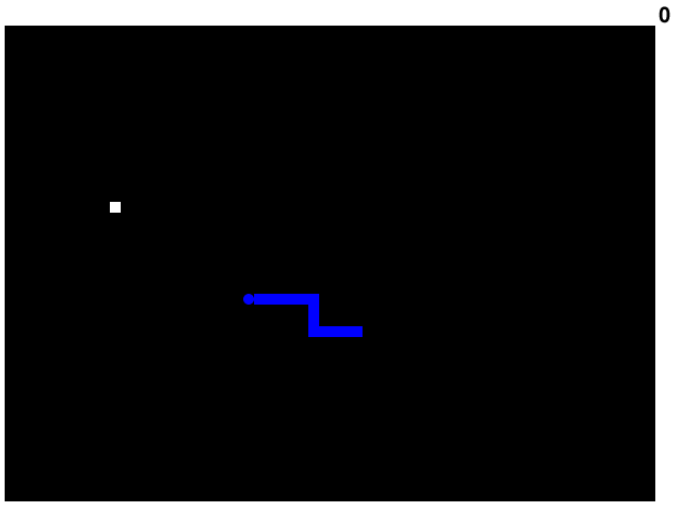


Figura 3 - Interface sem coordenadas

Scoring

A pontuação do jogo é mostrada em formato numérico no canto superior direito da tela. A pontuação aumenta em 25 a cada comida alcançada.

Sound

O jogo reproduz sons apenas no momento em que a serpente come um objeto.

Anexo B

Game Development Document do Jogo Arkanoid

LABORATÓRIO DE VÍDEOS E APLICAÇÕES DIGITAIS

GDD - BlockBreacker

Version 0.1

Ricardo Mendes Costa Segundo

12 de Julho de 2011

Conteúdo

Game Overview

Jogo baseado no clássico Arkanoid do Atari:

"O Arkanoid foi desenvolvido em 1986 pela Taito. O jogo foi baseado em jogos para o Atari dos anos 70. O jogador controla uma espécie de plataforma que impede que uma bola caia da área de jogo após rebater por diversos blocos. A bola, ao atingir tais blocos, faz com que eles desapareçam. Quando todos os blocos são destruídos, o jogador avança para outra fase, geralmente com alguma diferença bastante visível em relação à sua antecedente."

Game Play Mechanics

O jogo possui quatro elementos básicos: a esfera, os tijolos, a barra controlada pelo jogador e o mundo.

1. A ESFERA:

A esfera é um objeto do jogo que o jogador utiliza para destruir os tijolos. No inicio do jogo ela permanece estática até que o jogador pressione o botão "ENTER", dando a ela uma velocidade inicial de 100 pixels por segundo em direção aos tijolos.

Ao colidir com um tijolo, a esfera tem sua velocidade vertical invertida. A esfera também pode colidir com as paredes laterais, invertendo sua velocidade horizontal, onde, com a colisão na parte superior da tela resulta-se na inversão da velocidade vertical, e com a colisão na parte inferior resulta-se no final do jogo.

Por fim, ao colidir com a barra, a esfera tem sua velocidade vertical invertida, e à sua velocidade horizontal é adicionada a velocidade horizontal da barra no momento da colisão.

2. OS TIJOLOS

O jogo é constituído por 55 tijolos, dispostos em 5 linhas com 11 colunas cada, de forma estática. Ao sofrer a colisão de uma esfera, o tijolo é excluído do jogo. Quando todos os tijolos forem excluídos, o jogo é finalizado.

3. A BARRA

A barra representa o jogador dentro do jogo, sendo assim controlada pelo mesmo. Quando o jogador pressiona a seta esquerda(LEFT_BUTTON) do controle, a barra ganha um decréscimo de 20 unidades em sua velocidade horizontal e caso pressione a seta direita ganha um acréscimo de 20.

4. O MUNDO

O mundo constitui os cantos laterais, superior e inferior da tela, delimitando até onde a esfera pode ir.

Controls

O jogo possui apenas dois comandos: iniciar o jogo e controlar a barra.

Para iniciar o jogo basta ao jogador pressionar o Botão ENTER do controle remoto.

Para mover a barra, ele utiliza a seta esquerda e direita para movê-la na direção desejada.

Interface

A interface é apresentada na figura 1.

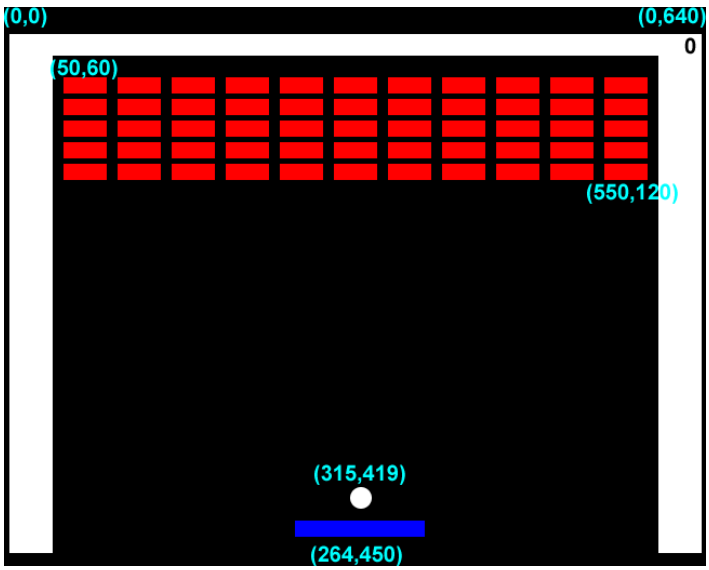


Figura 4- Interface com coordenadas

A figura apresenta as coordenadas iniciais dos elementos do jogo (barras laterais, tijolos, esfera e barra), onde os tijolos tem o espaçamento de 5 pixels entre si.

Abaixo (figura 2) a tela sem as coordenadas.

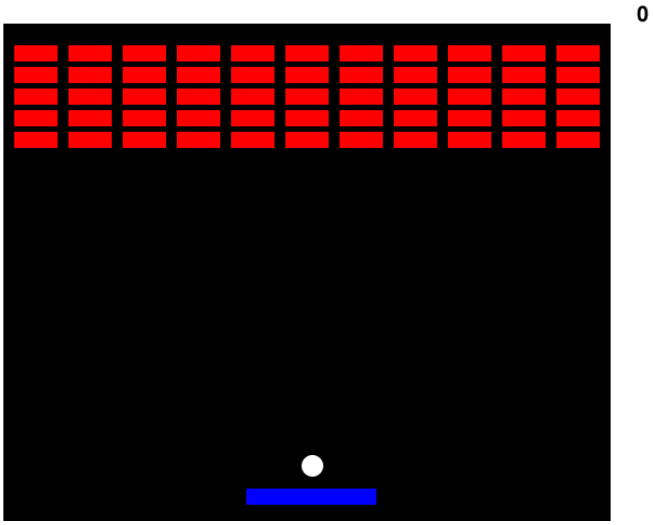


Figura 5 - Interface sem coordenadas

Scoring

A pontuação do jogo é mostrada em formato numérico no canto superior direito da tela. A pontuação aumenta em 25 a cada bloco destruído pelo jogador.

Sound

O jogo reproduz sons apenas no momento em que a esfera colide com a barra. Este som representa uma colisão, sendo acessível no arquivo de áudio “ball_bounce.mp3".

Anexo C

Termo de Consentimento

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Declaro, por meio deste termo, que concordei em participar na pesquisa de campo referente ao projeto/pesquisa intitulado(a) "**ATHUS: um framework para o desenvolvimento de jogos para TV Digital utilizando Ginga**" desenvolvida(o) por **Ricardo Mendes Costa Segundo**. Fui informado(a), ainda, de que a pesquisa é coordenada / orientada por **Tatiana Aires tavares**.

Afirmo que aceitei participar por minha própria vontade, sem receber qualquer incentivo financeiro ou ter qualquer ônus e com a finalidade exclusiva de colaborar para o sucesso da pesquisa. Fui informado(a) dos objetivos estritamente acadêmicos do estudo, que, em linhas gerais é: "o desenvolvimento de um framework para desenvolvimento de jogos para Televisão Digital através do middleware Ginga, de forma a possibilitar ao desenvolvedor customizar o framework para desenvolver aplicações específicas utilizando instancias das classes do framework".

Minha colaboração se fará de forma anônima, por meio de observação e coleta de dados através de formulários e análise de códigos a serem coletados a partir da assinatura desta autorização. O acesso e a análise dos dados coletados se farão apenas pelo(a) pesquisador(a) e/ou seu(s) orientador(es) / coordenador(es).

Fui ainda informado(a) de que posso me retirar desse(a) estudo a qualquer momento, sem prejuízo para meu acompanhamento ou sofrer quaisquer sanções ou constrangimentos.

Atesto recebimento de uma cópia assinada deste Termo de Consentimento Livre e Esclarecido.

Assinatura do(a) testemunha(a) Assinatura do pesquisador(a):

_____

Assinatura do(a) participante:

João Pessoa, ____ de _____ de _____

Anexo D

Desenvolvimento de Jogos para a plataforma Ginga utilizando NCLua

Questionário para coleta de dados dos interessados em participar do Mini-Curso: Desenvolvimento de Jogos para a plataforma Ginga utilizando NCLua.

* Required

Nome completo: *

Informe se nome completo.

Email: *

Informar o endereço eletrônico para contato.

Curso *

Informe o curso que atua ou atuou mais recentemente.

Grau acadêmico: *

Informe seu grau acadêmico

- Graduação;
- Graduação em andamento;
- Mestrado;
- Mestradoem andamento;
- Doutorado;
- Doutoradoem andamento;
- Other:

Grau de experiência em programação *

Informe seu conhecimento desenvolvimento de programas.

	1	2	3	4	5	6	7	8	9	10	
Nenhuma											Programador Senior

Linguagens de Programação *

Quais das Linguagens de Programação abaixo ja utilizou:

- C++;
- Java;
- JavaScript;
- NCL;

Lua;
Other:

Conhecimento sobre TV Digital *

Informe o que sabe sobre esta tecnologia/plataforma.

Conhecimento sobre Desenvolvimento de Jogos *

Informe sua experiência sobre desenvolvimento de jogos.

Necessidade de computador *

Responder se precisa que a organização disponibilize uma máquina para o participante. Caso possua notebook e o deseje utilizar, responda não.

Sim
Não

Submit

Anexo F

Questionário Qualitativo

Questionário para avaliar o grau de Satisfação das tecnologias utilizadas no curso.

* Required

Nome: *

Identificar o nome do participante.

Grupo: *

Identificar o grupo do participante.

- Grupo I
- Grupo II

Aplicação desenvolvida: *

Identificar a tecnologia utilizada no experimento.

- Blocks
- Snake

Você já conhecia a lógica do jogo desenvolvido? *

Domínio da Aplicação

- Não, nunca joguei e nem ouvi falar no jogo.
- Sim, como jogador.
- Sim, como desenvolvedor.

Tecnologia utilizada: *

Identificar a tecnologia utilizada no experimento.

- Framework Athus
- NCLua Puro

Como você avalia o seu aprendizado da tecnologia utilizada ? *

Facilidade de aprendizado:

1	2	3	4	5	6	7	8	9	10
Difícil de aprender									Fácil de aprender

Como você avalia a prática de uso da tecnologia utilizada no desenvolvimento do jogo? *

Facilidade de uso:

1	2	3	4	5	6	7	8	9	10
Difícil de utilizar									Fácil de utilizar

O que você achou da documentação (API, manual, links) disponibilizada durante o curso? *

Documentação:

Insuficiente
Adequada para conhecimento básico
Completa
Completa e de fácil leitura

Pontos positivos: *

Quais os pontos positivos encontrados no uso da tecnologia.

Pontos negativos: *

Quais os pontos negativos encontrados no uso da tecnologia.

Comente sua experiência no desenvolvimento. *

Submit