UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
Bacharelado em Ciência da Computação
Trabalho de Formatura Supervisionado

Desenvolvimento de um jogo integrado à rede social Facebook

Thiago Tatsuo Nagaoka

Orientador: Professor Marco Dimas Gubitoso

São Paulo, novembro de 2013

Resumo

Atualmente, com o avanço tecnológico e o crescente número de usuários da internet, popularizou-se o uso das redes sociais. Aliado ao mercado de jogos eletrônicos, que já possui destaque dentro do panorama mundial, um novo segmento vêm se destacando e se consolidando: o jogo social.

Dentro deste contexto, este trabalho consiste no desenvolvimento de um jogo casual simples para a plataforma *web* voltado a este novo segmento, apresentando todas as etapas importantes da composição de um jogo social.

Utilizando a *engine* gráfica Unity3D, foi desenvolvido um jogo de estratégia do estilo *tower defense*. A linguagem de programação escolhida foi C#. O caráter social do jogo se deu na integração da aplicação com a rede social Facebook.

Utilizando as ferramentas disponibilizadas pela plataforma Facebook Developers, foi possível fazer com que o jogo fosse carregado de dentro da própria página do Facebook e utilizasse informações das relações entre usuários para tornar o jogo interativo e competitivo.

Prefácio

Este trabalho foi desenvolvido para a disciplina MAC0499 - Trabalho de Formatura Supervisionado, durante o ano letivo de 2013. Esta disciplina é obrigatória para os graduandos do curso de Bacharelado em Ciência da Computação do Instituto de Matemática e Estatística da Universidade de São Paulo.

O trabalho de conclusão de curso é dividido em quatro partes:

- monografia (este presente documento);
- pôster;
- apresentação;
- arquivos e código-fonte da aplicação.

Estes arquivos podem ser encontrados na página desenvolvida para este trabalho, no endereço:

http://www.linux.ime.usp.br/~tatsuo/mac449

Durante o desenvolvimento do trabalho, foi mantido um blog onde todo o andamento do projeto foi registrado para controle, consulta e supervisão do responsável da disciplina e professor orientador. O blog pode ser acessado em:

http://social.stoa.usp.br/thitatsuo/blog

A aplicação final deste trabalho já esta implantada e pode ser testada em:

https://apps.facebook.com/tdfortcc

É necessário ter uma conta no Facebook para ter acesso ao jogo.

Sumário

Índice de Figuras

1. Introdução

Com a evolução tecnológica, a informação digital passou a ser muito mais acessível a todos de diversas formas. Maior capacidade de processamento dos dispositivos, diferentes maneiras de conectá-los, maior velocidade de transferência e barateamento dos serviços de internet fez com que fosse possível se manter conectado por tempo ilimitado.

Com o crescente número de usuários da internet, popularizou-se também as redes sociais. Rede social é uma estrutura social composta por pessoas ou organizações, conectadas por um ou vários tipos de relações, que partilham valores e objetivos comuns. As pessoas passaram a utilizar as redes sociais para se comunicar, dividirem experiências e estabelecer novos contatos, amplificando sua interação social do dia-a-dia.

Juntamente com o crescimento das redes sociais, vem se popularizando os jogos sociais. Principalmente os jogos casuais, simples e fáceis de aprender, passaram a ser apreciados por muitos usuários como forma de diversão, para se distrair ou interagir com seus amigos.

Diante deste contexto, muitas empresas passaram a investir nesta área, o que deslanchou o mercado deste nicho.

2. Motivação

O atual crescimento do mercado de jogos voltado para as redes sociais motivou o desenvolvimento deste trabalho. A popularização dos jogos sociais mobilizou o surgimento de muitas empresas *startups,* diversificando os tipos de aplicações voltadas para este ramo do mercado, tornando-o muito atrativo para os usuários.

Então, visando adquirir conhecimento e experiência no desenvolvimento de um jogo voltado para esta área do mercado, este projeto baseou-se na composição de uma aplicação completa, desde o planejamento inicial à implantação, aplicando todo conhecimento adquirido durante o curso de Bacharelado em Ciência da Computação.

3. Objetivos

Este projeto tem como objetivo o desenvolvimento e implementação um jogo para plataforma *web* onde o usuário poderá interagir de com outros usuários através da integração com a rede social Facebook. Aplicando todo conhecimento adquirido durante o curso de Bacharelado em Ciência da Computação, o projeto passa por todas as fases de desenvolvimento de um *software*, desde o planejamento até a implementação e implantação.

O jogo é baseado em um simulador de construções e foi desenvolvido utilizando a game *engine* Unity3D, e utilizando a linguagem de programação C#, com auxílio de outras ferramentas auxiliares.

4 Conceitos Gerais

4.1 Redes Sociais

O termo rede social, antes do surgimento de tecnologias como a internet, era utilizado para definir a interação de pessoas em comunidade ou grupos, com a finalidade de unir indivíduos com mesmos interesses e afinidades.

Com a evolução das tecnologias de comunicação, este termo passou a ser usado principalmente para definir as redes de relacionamento, ou redes sociais digitais então surgidas. Uma rede social digital atualmente pode ser considerada como uma estrutura social composta por pessoas ou organizações, conectadas por um ou vários tipos de relações, que partilham valores e objetivos comuns em um espaço virtual comum para compartilhar informações e dados das mais diversas formas, sejam vídeos, fotos ou textos, e que utiliza a rede de conexões da internet para transmissão e recepção dos dados.

As pessoas passaram a utilizar as redes sociais para comunicar-se, dividirem experiências e estabelecer novos contatos, amplificando sua interação social do dia-a-dia, tornando-as muito populares. As redes sociais podem operar em diferentes níveis, como, por exemplo, redes de relacionamentos (Facebook, Orkut, MySpace, Twitter, Badoo), redes profissionais (LinkedIn), redes comunitárias (redes sociais em bairros ou cidades), redes políticas, dentre outras.

O uso das redes sociais para diferentes finalidades vêm crescendo muito nos últimos anos. O crescimento desses sites de relacionamento não tem sido apenas no número de usuários, mas também no tempo de navegação.

O Brasil é um dos países com maiores investimentos em redes sociais como Facebook e Orkut. Segundo pesquisa do Instituto Nielsen [5], os brasileiros são os que mais acessam redes sociais em *smartphones* no mundo, superando os Estados Unidos e os países do Bric (Rússia, Índia e China). São responsáveis por 65% do tráfego da internet no país e, no período de 1 ano, houve um crescimento de 82% no tempo gasto em sites como Orkut, Twitter e Facebook.

Devido a grande demanda de utilização, a indústria do entretenimento viu neste nicho de mercado uma grande oportunidade mercadológica. Assim, uma grande quantidade de investimentos foi voltado para jogos em rede de relacionamentos, cunhados de "jogos sociais".

4.2 Jogos Sociais

O termo jogo social já existia antes mesmo do surgimento das redes sociais e do crescimento da internet. Era utilizado para definir jogos que requerem interação entre duas pessoas ou mais. Muitos jogos foram criados e difundidos durante a história da humanidade, alguns datando de 3200 A.C. Foram criados com diferentes objetivos e finalidades, desde diversão, até disputas por posses e títulos, e estratégias de guerra. Muitos desses jogos, como o xadrez e o *go*, são populares até os dias de hoje, e são adorados e praticados, como passatempo, esporte ou estudo.

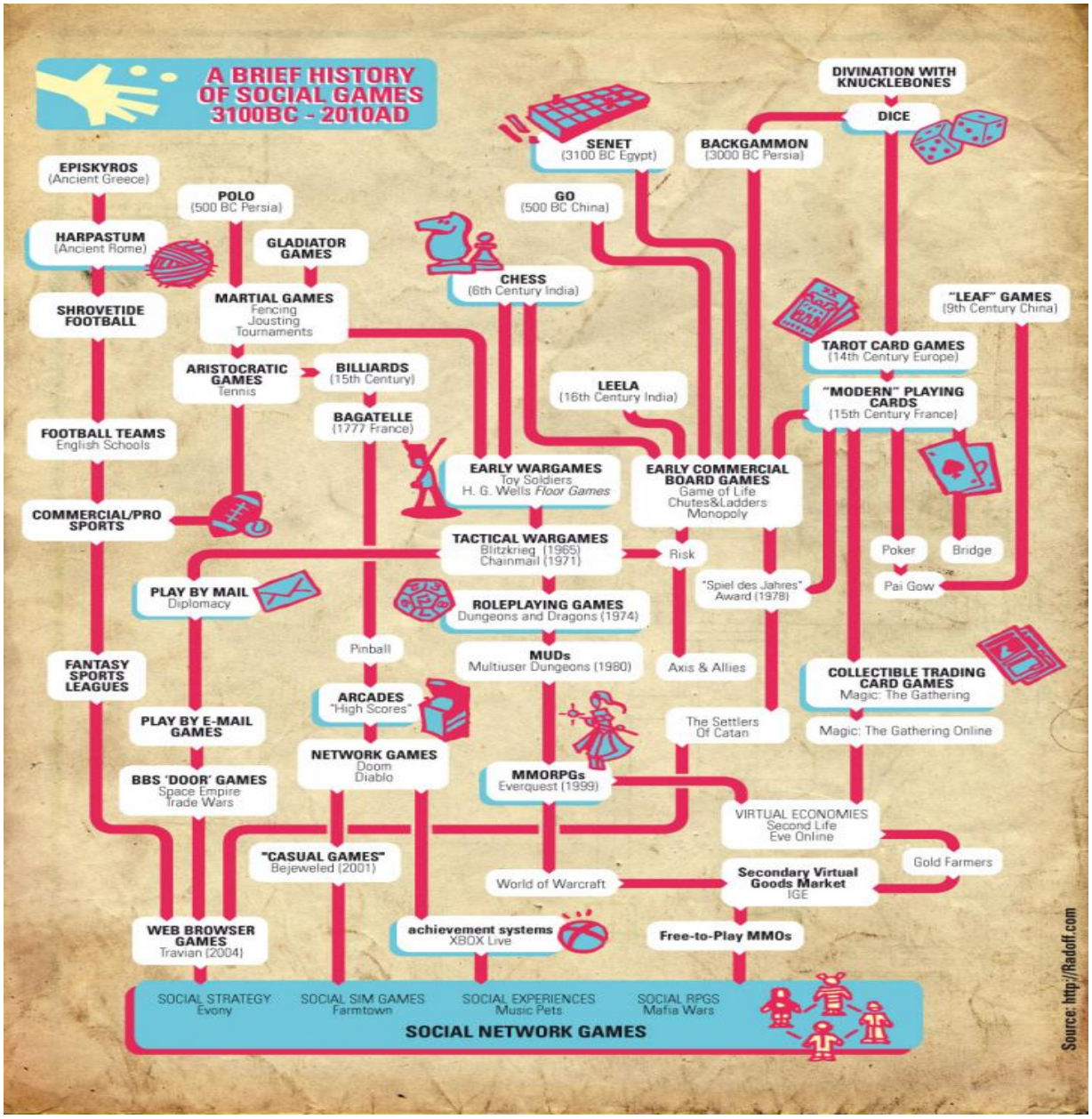

Figura 1: História dos jogos sociais
Fonte: página *Jon Radoff's Internet Wonderland*[1]

Porém, com a rápida evolução tecnológica e popularização das redes sociais, surgiu um novo tipo de jogo social, este o qual possui uma premissa diferente dos jogos sociais existentes desde então: não era mais necessária a interação física entre os participantes, sendo que os jogos podem ser conduzidos remotamente entre os jogadores, de maneira assíncrona, e em larga escala. Desta forma, pode-se, através da

[1] Disponível em: <http://radoff.com/blog/2010/05/24/history-social-games/> Acesso em setembro de 2013

internet, jogar com adversários do mundo todo, com várias pessoas conectadas ao mesmo tempo, e em uma escala maior do que a possível nos jogos sociais presenciais. Mais que isto, não necessariamente precisa ser em tempo real, pois o jogo pode conter uma “memória” onde os dados da partida ficam disponíveis toda vez que o jogador se conectar. Essa nova categoria pode ser classificada como “jogos sociais digitais” e em sua maioria baseiam-se na conectividade das pessoas através das redes sociais para uma maior interação entre seus utilizadores.

4.3 Jogos e o “Mercado Social”

Segundo a empresa eMarkerter, estima-se que no ano de 2013 as redes sociais devem crescer 16%, 11,6% em 2014 e, assim, atrairão 1,854 bilhões de pessoas [4]. Com o crescimento massivo da utilização das redes sociais, o mercado de entretenimento aproveitou-se desta popularização para investir nos jogos sociais. O mercado de jogos sociais no Brasil deve chegar a US$ 238 milhões até 2014, de acordo com o site Venture Beat, especializado em tecnologia [6].

A principal estratégia vem sendo investir e concentrar esforços em quatro fatores que culminaram no sucesso dos jogos sociais: conectividade, interatividade, competitividade e casualidade.

4.3.1 Casualidade

O termo jogo eletrônico casual é utilizado para caracterizar jogos digitais que são considerados simples, rápidos e fáceis de se aprender, diferentemente dos jogos tradicionais que são mais complexos e exigem tempo e dedicação do jogador. Desta forma podem ser uma opção de diversão para um simples passatempo de alguns minutos.

A empresa de análise comercial Packs Associates pronunciou que o mercado casual vai crescer até uma fatura anual de786 milhões de euros no ano de 2013 [7].

Por se tratarem de jogos simples e divertidos, atinge um público diferenciado dos jogadores tradicionais de jogos eletrônicos. Segundo o site geek.com.br [8], especializada em tecnologia, a fabricante de jogos sociais Popcap Games concluiu através de uma pesquisa encomendada com a Information Solutions Group (ISG) [9] que, apesar do grande público dos jogos tradicionais ser composto de homens jovens, 55% dos usuários dos jogos sociais digitais são mulheres, e destes 22% estão na faixa dos 50 a 59 anos, derrubando estereótipos.

Esta pesquisa (realizada nos Estados Unidos e Reino Unido) mostrou também que as mulheres são as jogadoras mais ávidas: 38% delas afirma jogar diversas vezes por dia, contra 29% dos homens.

4.3.2 Interatividade

Sobre a questão social dos jogos, constatou-se através da pesquisa da Information Solutions Group [9], que quase metade (43%) dos usuários joga com seus pais, filhos ou outros parentes, 62% dos entrevistados concorda que os jogos sociais ajudam a retomar o contato com antigos amigos, colegas etc., e 63% acredita que eles são fonte de novas amizades. Quando questionados sobre outros atrativos dos jogos sociais além da interação, foram citadas “competição amigável” (59%), “interatividade” (49%) e “oportunidade de ganhar prêmios” (35%).

O sucesso dos jogos sociais se deve também ao fato da vasta malha de contatos proporcionada pelas redes sociais. Mais de 50% dos usuários começou a jogar por uso ou recomendação dos amigos.

4.3.3 Competitividade

Muitos jogos permitem a comparação de desempenhos, o que motiva os usuários a jogarem cada vez mais em busca de melhores resultados para superar seus amigos. Assim é gerada uma “competitividade amigável”, que atrai cada vez mais usuários que encontram nas partidas uma nova maneira de interagir socialmente.

A empresa Color Cube Games divulgou uma pesquisa informando que até o primeiro trimestre de 2010 haviam no Brasil cerca de 39 milhões de usuários de *web*, do quais 7 milhões utilizavam o tempo conectado para jogar [10]. Destes 7 milhões, 90% utilizavam algum método comparativo em seus resultados. Desta maneira, estes jogadores acabaram fazendo uma projeção para os outros usuários de modo que instigavam a curiosidade sobre a crescente popularidade dos jogos.

4.3.4 Conectividade

A evolução tecnológica permite nos dias atuais que informações e recursos possam ser acessados e utilizados em qualquer lugar e em qualquer momento. Juntamente com isso, nos últimos anos, a explosão dos *smartphones* com maiores capacidades de processamento, e capazes de acessar a internet por tempo descontínuo nos impulsionou para a crescente era da computação móvel. A

computação móvel é a área da tecnologia que amplia o domínio da computação distribuída, pois faz uso da comunicação sem fio para eliminar a limitação da mobilidade. Através de um dispositivo portátil é possível se comunicar com a parte fixa da rede e com outros computadores móveis.

Juntamente com isso, as redes sociais migraram também para os dispositivos móveis, possibilitando seu acesso através de aplicativos personalizados em *tablets* e *smartphones*. Não demorou muito para as empresas migrarem também seus jogos no formato de aplicativo móvel, aproveitando a possibilidade dos usuários então poderem se manter conectados 100% do tempo.

5. Conceitos e Tecnologias

5.1 *Engine* Unity3D

Unity3D é um motor de jogos (*game engine*) multi-plataforma, criado pela Unity Technologies. Auxilia na criação de jogos e aplicativos para diferentes propósitos e plataformas. Aceita como linguagens de programação o javascript, C# e boo script. O Unity3D possui duas versões principais: a versão comercial Unity Pro, e a versão gratuita, que contém apenas as funcionalidades básicas.

Segundo a própria desenvolvedora, o Unity3D já alcança a marca de dois milhões de desenvolvedores registrados, tornando-se a *game engine* mais utilizada para desenvolvimento de jogos na atualidade e mantendo a maior comunidade ativa de desenvolvedores do mundo. Seu foco principal é o desenvolvimento para plataformas *mobile* e *web games*, mas seu uso para criação de jogos para *console* (videogames) e PC vem sido muito difundido também.

O Unity3D é apropriado para todos os tipos de desenvolvedores, desde grandes e pequenas empresas, profissionais independentes e estudantes. Muitas empresas multinacionais utilizam a Unity3D no desenvolvimento de seus projetos. Dentre elas estão as empresas Cartoon Network, Coca-Cola, Disney, Electronic Arts, LEGO, Microsoft, NASA, Nexon, Nickelodeon, Square, Ubisoft, US Army e Warner Bros.

5.1.1 Plataformas

Com um projeto feito com a Unity3D, os desenvolvedores tem a capacidade de compilar e gerar um executável para diversas plataformas. São elas:

- Dispositivos *mobile* (celulares *smartphones* e *tablets*), com sistemas iOS e Android;

- Computadores *desktop*, com sistemas Operacionais Windows, Mac OS e Linux;
- *Web Player*, utilizando Adobe Flash ou *plugin* Unity Web Player*,* compatível com os navegadores Internet Explorer, Firefox, Safari, Opera, Google Chrome e Camino;
- Videogames consoles Playstation 3, Xbox 360 e Wii.

No ano de 2012, a Unity Technologies anunciou o lançamento de ferramentas de desenvolvimento com suporte para Black Berry 10, Playstation 4, Playstation Vita, Playstation Mobile, Tizen, Wii U, Windows Phone 8, Windows Store e Xbox One.

5.1.2 Bibliotecas, Recursos e Componentes

A Unity3D é uma ótima *engine* para criação de jogos em 3D, possuindo biblioteca de física, detector de colisão e ferramentas para animação. É também muito utilizada para o desenvolvimento de jogos 2D, com o auxílio de *plugins* e *frameworks* específicos.

Desenvolvedores podem também criar bibliotecas, componentes e recursos, que auxiliam em projetos específicos. Esses recursos então podem ser disponibilizados na *Asset Store* (Loja de Recursos) de maneira gratuita ou comercial. Existem na loja modelos de personagens, texturas e materiais, ferramentas de pintura de paisagem, ferramentas de efeitos sonoros, ferramentas de desenvolvimento 2D, entre muitos outros recursos. Dessa forma, com os recursos específicos incorporados, cada usuário tem o poder de adaptar o Unity3D para atender as suas necessidades de desenvolvimento.

A *engine* gráfica usa Direct3D (Windows, XBOX 360), OpenGL (Mac, Windows, Linux, PS3), OpenGL ES (Android, iOS) e APIs proprietários (Wii).

A *engine* suporta ainda arquivos e formatos de objetos 3D de vários programas de modelagem populares no mercado, tais como 3ds Max, Maya, Softimage, Blender, modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop e Adobe Fireworks. Estes recursos podem ser adicionados ao projeto, e manejados pela interface gráfica do Unity3D.

5.1.3 Terminologia Básica

Será descrito nos próximos itens os termos mais utilizados no desenvolvimento de aplicativos e games utilizando o Unity3D, que são importantes para o desenvolvimento e organização dos projetos.

5.1.3.1 *Projects*

Conjunto dos elementos que envolvem todo o projeto do jogo, como modelos, *scripts, prefabs*, *assets*, sons e texturas. Deve ser bem organizado, utilizando uma estrutura de pastas (*folders*), agrupando os elementos pertinentemente de maneira estrutural (elementos de mesmo tipo) ou lógica (elementos que são relacionados entre si). Todas as pastas criadas no *project* corresponderá a uma pasta real no sistema de arquivos do computador, localizado dentro da pasta *assets* do projeto.

Em geral, um único jogo é representado por um *project,* que contem todos seus elementos necessários para seu funcionamento.

5.1.3.2 *Assets*

Dentro do Unity3D, os *assets* podem assumir muitas definições. Eles podem ser elementos ou aspectos do jogo que serão referenciados por algum componente ou outro *asset*.

Podem ser externos (e precisam ser importados para dentro do projeto), como modelos 3D, texturas, efeitos sonoros e scripts. Ou podem ser internos como materiais, *shaders* e *prefabs*.

5.1.3.3 *Scenes*

São as divisões do jogo. Um jogo em geral é dividido em níveis ou *level,* e dentro do projeto cada um deles representará uma *scene.* Não só os níveis do jogo são *scenes*, mas também toda subdivisão como telas de início e fim, abertura, menu principal, créditos são modeladas como *scene*.

Cada scene contém todos os *game objects* e *scritps* que irão compor a cena, e serão capazes de dar lugar para uma nova *scene* carregando-a quando algum evento ocorrer, criando a linearidade do jogo.

5.1.3.4 *Game Objects*

São os elementos do jogo que compõe cada *scene*. Os objetos podem ser instanciados, movidos, rotacionados, escalados, pertencer a uma hierarquia de objetos e podem ser definidos por componentes (o que define características como funcionalidades e comportamentos).

5.1.3.5 *Packages*

Conjunto de arquivos de um projeto, agrupados de forma que é possível ser importado e exportado de um *project*, dependendo da necessidade.

Pode ser tratado como um conjunto de *asset* que pode conter vários arquivos como modelos, texturas, scripts, que podem estar agrupados e comprimidos, e ao importar para o Unity3D, são incorporados ao projeto.

Quando se deseja exportar algum elemento, o Unity3D é capaz de reconhecer todos os arquivos que contém dependência com este elemento e os comprimem junto criando o pacote.

5.1.3.6 *Prefabs*

Grupo de *assets* com suas devidas características como texturas, comportamento, modelos, e que em geral representam um elemento do jogo. São utilizadas para que se possa instanciar múltiplas cópias destes elementos com as mesmas características.

Todos os elementos criados a partir de uma *prefab* herdam as características determinadas na *prefab*. Quando desejar alterar ou inserir uma nova característica a todas as instâncias de um elemento, todas as alterações executadas na *prefab* serão atualizadas nos objetos instanciados a partir dela.

É possível fazer mudanças pontuais em cada objeto também, aplicando estas mudanças diretamente ao objeto instanciado, sem alterar as definições das *prefab*.

5.1.3.7 *Components*

Grupo de parâmetros e funcionalidades que definem o que um *game object* faz. É possível atribuir funcionalidades como com física de colisão, emissão de luz, modos de renderização, texturas, efeitos, áudio, entre vários.

5.1.3.8 *Scripts*

São códigos que determinam comportamentos a objetos ao qual são associados, para integrar à lógica do jogo. O Unity3D aceita *scripts* na linguagem C#, javascript e booscript.

5.2 *Facebook Plataform*

Com o crescimento desenfreado das redes sociais, desenvolvedores estão investindo cada vez mais nas aplicações integradas a elas. Jogos, aplicações e uma infinidade de sistemas integrados surgem diariamente, e são responsáveis por boa parte do sucesso da plataforma.

O Facebook, uma das redes sociais mais populares atualmente, desenvolveu o *Facebook Platform*, para facilitar a construção de aplicativos sociais *no* Facebook e na *web*. Consiste em uma coleção abrangente de avançados API's (*Application program interface*) e SDK's (*Software Development Kit*).

5.2.1 *Facebook API*

API é um acrônimo para "Interface de Programação de Aplicativos". Consiste em um conjunto de padrões e rotinas pré-estabelecidas para a utilização das funcionalidades um software, encapsulando sua implementação, e oferecendo seus serviços de maneira simples e transparente.

A API principal da *Facebook Platform* é a *API Graph* que permite a leitura e gravação de dados a partir da rede do Facebook.

A *API Graph* do Facebook considera como um objeto um grupo de dados sobre um usuário: seu perfil, lista de amigos, postagens, fotos, preferências, entre outros. Mantém também dados dos relacionamentos e conexões entre eles. Aos objetos, é atribuído um ID exclusivo e eles são facilmente acessíveis, usando, por exemplo, um endereço de um recurso *Web* (URL) com as devidas permissões.

Com a *API Graph*, é possível recuperar objetos, excluir e publicar, atualizar, filtrar resultados e até descobrir dinamicamente conexões/relacionamentos de um objeto.

Por padrão, os aplicativos têm acesso aos dados públicos do usuário, e para acessar dados privados, devem ser solicitadas as permissões do usuário, chamadas de permissões estendidas.

5.2.2 *Unity Facebook SDK*

SDK é o acrônimo para "Kit de Desenvolvimento de *Software*". O *Unity Facebook SDK* é disponibilizado pelo Facebook Developers, segmento que auxilia e suporta os desenvolvedores de aplicativos e páginas *web* voltados para o Facebook. É constituído por uma série de ferramentas desenvolvidas para que programadores externos tenham uma melhor integração com a *Facebook API*.

O *Unity Facebook SDK* possui diversas classes e funções que podem ser integradas ao projeto no Unity3D, simplificando e otimizando o uso das principais funcionalidades do *Facebook API* e acesso aos dados. Utilizando esta ferramenta, é possível adicionar funcionalidades aos aplicativos, tais como:

- Fazer *login* com o usuário do Facebook;
- Habilitar o botão *Like* e outros *Social Plugins* no aplicativo;
- Integração com a API primária do Facebook, o *Graph API*, que resgata dados e objetos;
- Adicionar filtros às buscas de objetos;
- Permite envio de mensagens, que provê que usuários compartilhem informações;
- Facilita a comunicação entre aplicativos.

5.3 *NGUI Framework*

Framework de software compreende em um conjunto de classes implementadas em uma linguagem de programação específica, usadas para auxiliar o desenvolvimento de *software*, e é usado para resolver um problema de um domínio específico.

O *NGUI Framework* é um arcabouço de abstração desenvolvido para atuar nos projetos Unity3D, e que reúne diversas classes e funções provendo uma funcionalidade genérica para a interface gráfica de interação com o usuário.

Desenvolvida pela empresa Tasharen Entertainment [17], foi implementada em C# e possui as seguintes características que se adequaram perfeitamente no projeto:

- *Open source*, possibilitando modificações que para propósitos específicos;
- É adaptável, sendo que os *scripts* podem ser utilizados em classes e objetos externos ao *Framework*;
- Interage com os elementos do jogo, classes e objetos de maneira eficaz e otimizada;
- Suporte para a criação de *atlas* (arquivo compactado com todos os elementos visuais utilizados, para otimizar o uso da memória e chamadas de renderização gráfica);
- Código simples e de fácil entendimento;
- Boa documentação e tutoriais de uso disponíveis.

Assim, *NGUI Framework* provê soluções simples para a manipulação de elementos de interação com o usuário, como a criação de botões interativos, mostradores de mensagens, alertas, *checkbox*, *inputs*, entre outros.

5.4 Linguagens de Programação

5.4.1 C#

C# é uma linguagem de programação orientada a objetos e fortemente tipada desenvolvida pela Microsoft, como parte da plataforma *.NET*. Possui muitas similaridades com a linguagem C++, na qual foi baseada juntamente com influências de Java e C, adaptando os melhores recursos de cada linguagem e acrescentando novas capacidades próprias.

Possui funcionalidades diversas que auxiliam bastante no desenvolvimento de aplicações como o deste presente projeto, tais como:

- Orientação a objetos, tudo que é instanciado é um objeto (*System.Object* é a classe base de todo o sistema de tipos de C#);

- Fortemente tipada, ajudando a evitar erros por manipulação imprópria de tipos e atribuições incorretas;
- Controle de versões, onde cada *assembly* gerado, seja como *EXE* ou *DLL*, tem informação sobre a versão do código;
- Tratamento de exceções;
- Altamente escalável, permitindo que uma mesma aplicação possa ser executada em diversos dispositivos de *hardware;*
- Suporte a multimídia (áudio, imagens, animação e vídeo);
- Processamento de banco de dados.

5.4.2 PHP

PHP (um acrônimo recursivo para *"PHP: Hypertext Preprocessor"*) é uma linguagem de programação interpretada livre, usada originalmente apenas para o desenvolvimento de aplicações presentes e atuantes no lado do servidor, capazes de gerar conteúdo dinâmico em desenvolvimento *web*.

O código é interpretado no lado do servidor pelo módulo PHP, que pode também gera a página *web* a ser visualizada no lado do cliente.

Tem como principais características:

- Estruturado e orientação a objetos;
- Portabilidade (independência de plataforma);
- Tipagem dinâmica;
- Sintaxe similar a C/C++ e o Perl;
- *Open-source*;
- *Server-side* (o cliente manda o pedido e o servidor responde em página HTML).

Neste projeto, foi utilizado como ferramenta de comunicação entre o jogo e o banco de dados que persiste as informações de cada usuário. O PHP no servidor possui bibliotecas de conexão com o banco de dados através do sistema gerenciador PostgreSQL, com o qual foi possível executar consultas e atualizações nas relações. A

Unity3D possui métodos e classes que auxiliam no acesso de páginas da internet, tanto HTML como PHP, integrando assim estas tecnologias.

6 O Jogo

6.1 Descrição do Jogo

O jogo desenvolvido segue o estilo *Tower Defense*. É um jogo de estratégia, onde o objetivo é tentar impedir que inimigos cruzem um mapa a partir de um ponto inicial pré-determinado (chamado *spawn point*) para um ponto objetivo fixo (chamado *objective point*). Para isso, o jogador deve construir armadilhas para atrasar a travessia, ou construir torres, que são elementos estáticos responsáveis por atacar os inimigos para eliminá-los antes de alcançar o ponto objetivo. Os inimigos e torres têm habilidades diferentes, custos e melhorias. Sempre que um inimigo é eliminado o jogador recebe pontos (para inseri-lo na classificação geral) e dinheiro (recursos), para comprar e melhorar torres, aprimorando suas defesas.

Cabe então ao jogador criar uma estratégia para compras, posicionamento e melhoria de suas torres, para passar por vários níveis do jogo e manter a defesa de seu mapa a melhor possível.

6.2 Jogador

Para poder jogar, o usuário deve ter uma conta cadastrada na rede social Facebook. A partir de sua conta, o jogo é disponibilizado na própria página do Facebook, através de sua aba de aplicativos, sendo executado no navegador de acesso. O jogo funciona mediante a instalação do *plugin Unity Web Player* no navegador, o que é feito automaticamente na primeira vez que o jogo é executado. Cada jogador possui uma identificação (chamada de ID) que o identifica univocamente. Esse ID é a mesma utilizada pelo Facebook.

Cada jogador tem atributos que definem os estados do seu jogo. Através deles, é possível definir o nível do jogador, o quanto evoluiu no jogo, e comparar seu desempenho com outros usuários. Esses dados são persistidos em um banco de dados em um servidor para que seja possível salvar o estado de um jogo, e retomá-lo quando desejado. Os principais atributos são:

- Pontos: Toda vez que o jogador eliminar um inimigo ele será contemplado com pontos, que serão acumulados. Assim, será gerada uma classificação geral de todos os jogadores, onde poderão comparar seus desempenhos.

- Vida: Os jogadores têm uma quantidade inicial de vida. Toda vez que um inimigo conseguir alcançar o ponto objetivo, uma vida será subtraída. Quando o estoque de vida se esgotar, então para cada inimigo que conseguir alcançar o fim do caminho será retirada uma quantidade grande dos pontos, podendo até deixar o jogador com uma pontuação negativa, fazendo-o ter uma classificação geral baixa do jogo.

- Recursos: Os recursos são importantes para construir os elementos de defesa. A construção de torres e seus melhoramentos têm um custo, que é inflacionado conforme o jogador for evoluindo no jogo. As vidas também podem ser compradas com os recursos. O jogador tem uma quantidade inicial de recursos, e para cada inimigo que conseguiu eliminar será contemplado com recursos de acordo com o nível do inimigo.

- Mapa: É um atributo que guarda a configuração do mapa de um jogador. Trata-se de uma matriz que representa o mapa, guardando a posição das torres construídas com suas configurações, além de marcar também os espaços construíveis disponíveis e os não construíveis.

- Nível: Atributo que guarda qual o atual nível do jogo. Uma quantidade de inimigos aparecerá em grupos em um dado nível. Sempre que o jogador eliminar todos os inimigos de um nível sem que o seu atributo “vida” chegue ao zero, ele avançará para um nível seguinte. A quantidade de inimigos será determinada pelo valor do atributo “nível”, através de uma expressão matemática pré-determinada.

6.3 Inimigos

Os inimigos são os elementos que acrescentam dificuldade ao jogo. Aparecem em grupos (chamadas *waves*) em cada nível. O jogador terá o controle de quando iniciar uma *wave* de um nível, podendo assim ter tempo para planejar suas táticas de defesa, e quando estiver pronto, iniciar a partida do nível corrente. Uma vez iniciada a partida, não poderá ser interrompida até que o último inimigo seja eliminado ou alcance o ponto objetivo.

Os inimigos têm atributos, que definem sua força e a dificuldade de cada nível. Quanto maior o nível, mais poderosas são as *waves*, aumentando a quantidade de inimigos e sua força. Os atributos são:

- Vida: Um inimigo é eliminado quando o valor do seu atributo “vida” for menor ou igual a zero. A cada ataque de uma torre, a vida do inimigo é subtraída pelo valor do ataque da mesma.

- Velocidade: O atributo “velocidade” determinará o tempo que um inimigo leva para percorrer todo o caminho desde o ponto de início até o ponto objetivo no mapa. Quanto maior a velocidade, menor será o tempo de travessia, aumentando a dificuldade do nível.

6.4 Torres

As torres são os elementos estáticos que o jogador pode construir para conter o avanço dos inimigos. Poderão ser construídas em lugares específicos do mapa, tendo um custo de recursos. Depois de construída, passará a atacar todo inimigo que passar por sua área de ação. Tem como atributos:

- Área de ação: Toda torre possui uma área de ação, que determina quando e qual inimigo poderá atacar. Uma circunferência com centro coincidente ao centro da torre representa a área de ação, sendo que o valor do atributo determina o raio. Todo inimigo que entrar dentro desta área estará sujeito ao ataque desta torre.

- Força: A torre que identificar um inimigo dentro da sua área de ação passará a atacá-lo. A cada ataque, o inimigo sofrerá um dano, que será subtraído do valor de sua vida. A quantidade de dano aplicado será determinada pelo atributo força.

- Velocidade de ataque: Enquanto um inimigo estiver dentro do raio de ação de uma torre, ele será continuamente atacado pela mesma. Porém a torre tem um atraso entre um ataque e outro, que pode ser ilustrado como um tempo de recarga de sua arma. Este tempo de atraso é representado pela “velocidade de ataque” e quanto maior o valor, menor será o tempo de atraso.

Uma vez construída, uma torre pode receber melhorias dependendo da estratégia do jogador. Pagando um valor determinado, é possível aumentar sua força, área de ação ou a velocidade de ataque independentemente. A cada nova melhoria feita, o preço da próxima melhoria cresce exponencialmente, afim de não tornar o jogo muito fácil executando melhorias repetidamente.

É possível também vender uma torre já construída, recebendo um valor inferior do que foi gasto para sua construção e melhorias (50% do valor total gasto), deixando então livre o espaço por ela ocupado.

6.5 Caminho

Os inimigos seguem por um caminho que passa através do mapa, iniciando de um ponto inicial para o ponto objetivo. O caminho de travessia é determinado através de um algoritmo de busca de caminho A* Pathfinder, configurando-se a área em que é possível caminhar e mapeando os obstáculos.

6.6 Integração com Facebook

Por se tratar de um jogo social, está integrado com a rede social Facebook. Esta integração permite que o jogo desenvolvido para a plataforma *web* possa ser executado diretamente do navegador, na própria página do Facebook.

A aplicação é capaz de utilizar métodos previstos pela API do Facebook para tornar o jogo interativo para o usuário. É possível que o jogador envie requisições e convites para seus contatos. Existe também a possibilidade do usuário optar que o aplicativo poste uma notificação em seu mural pessoal com sua pontuação atual para desafiar seus contatos a obterem melhor desempenho, explorando o caráter de competitividade e interatividade do jogo social.

7. Descrição das atividades realizadas

7.1 Planejamento

Com objetivos definidos sobre o projeto, inicialmente foi necessário um planejamento para garantir as prioridades de implementação, detalhes das etapas, direção e foco no desenvolvimento do aplicativo, dentro dos prazos estabelecidos.

Baseado em conceitos de métodos ágeis, o plano de ações e tarefas seguiram o modelo iterativo e incremental. O sistema foi desenvolvido através de repetidos ciclos (iterações) e em pequenas porções (incrementos), aproveitando assim o que foi aprendido durante o desenvolvimento e das melhorias do sistema em iterações anteriores.

Cada iteração é baseada no levantamento dos requisitos, iniciando por uma simples implementação e aumentando iterativamente até que o aplicativo esteja completo.

Em cada incremento é realizado todo o ciclo do desenvolvimento de *software*, do planejamento aos testes do sistema já em funcionamento, como ilustrado na figura 2. A fase de comunicação se trata dos levantamentos dos requisitos junto ao cliente, sobre as expectativas de funcionamento do sistema para esta etapa e os resultados. No caso deste projeto o próprio desenvolvedor faz o papel de cliente, e, portanto a fase de comunicação passa a ser o levantamento de requisitos, estudo e análise do sistema para o atual incremento.

Cada etapa produz um sistema totalmente funcional, apesar de ainda não cobrir todos os requisitos. Uma iteração acaba quando um incremento já está implantado e testado, e assim uma nova iteração pode ser iniciada para implementação e adição de uma nova funcionalidade.

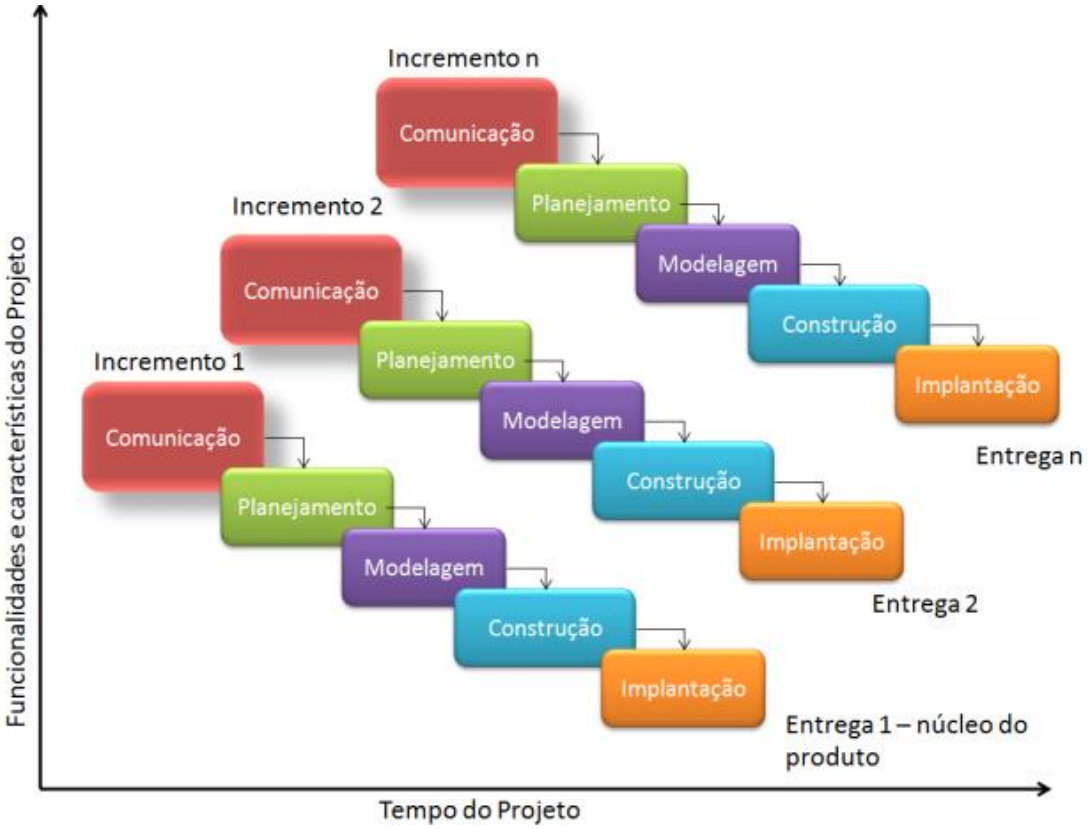

Figura 2: Etapas do modelo iterativo incremental
Fonte: PRESSMAN (2010)

Utilizando estes métodos, temos a vantagem de rapidamente obtemos resultados iniciais, e proporcionar um *feedback* logo no início do projeto. Isso torna possível detectar com antecedência os problemas e resolver as necessidades de mudanças com menos esforço do que seria após a conclusão total do projeto.

7.2 Levantamento de Requisitos

Os requisitos estão ligados às funcionalidades propostas inicialmente pelo sistema. Descrevem como o aplicativo deve se comportar e reagir às interações com usuários e outros *softwares*.

Para o presente projeto foram determinados os seguintes requisitos:

- Jogo para plataforma *web*, podendo ser executado nos principais navegadores (Internet Explorer, Firefox e Google Chrome);
- Interface com gráficos 3D, mas limitando a jogabilidade a dois eixos;

- Visualização do mapa com todas as configurações;
- Pelo menos um tipo de inimigo, tendo seus atributos balanceados de acordo com o nível jogado;
- Inimigos devem possuir inteligência artificial, capaz de planejar a rota mais eficiente até o ponto objetivo através dos caminhos livres do mapa;
- Disponibilidade de pelo menos dois tipos de torres, podendo ser melhoradas inúmeras vezes em três atributos (área de ação, força e velocidade de ataque);
- Torres devem possuir inteligência artificial, capaz de identificar quais os inimigos estão em sua área de ação e determinar qual atacar;
- Sistema de compra, venda e melhoramentos de torres, e compra de vidas;
- Persistência dos dados da configuração do mapa em banco de dados no servidor, para poder ser carregado a qualquer momento que o jogador se conectar;
- Salvar e recuperar os dados em banco de dados, permitindo reiniciá-los quando desejado;
- Integração com a rede social Facebook, sendo possível obter dados da conta do usuário (como ID, nome e lista de amigos) com as devidas permissões.

7.3 Arquitetura do Jogo

Por se tratar de um jogo casual, ou seja, simples e de fácil entendimento para qualquer usuário, a estrutura do jogo também é simplificada. A estrutura pode ser representada por uma máquina de estados ou autômato, onde cada estado representa uma cena do jogo. Assim temos as transições mostradas na figura a seguir.

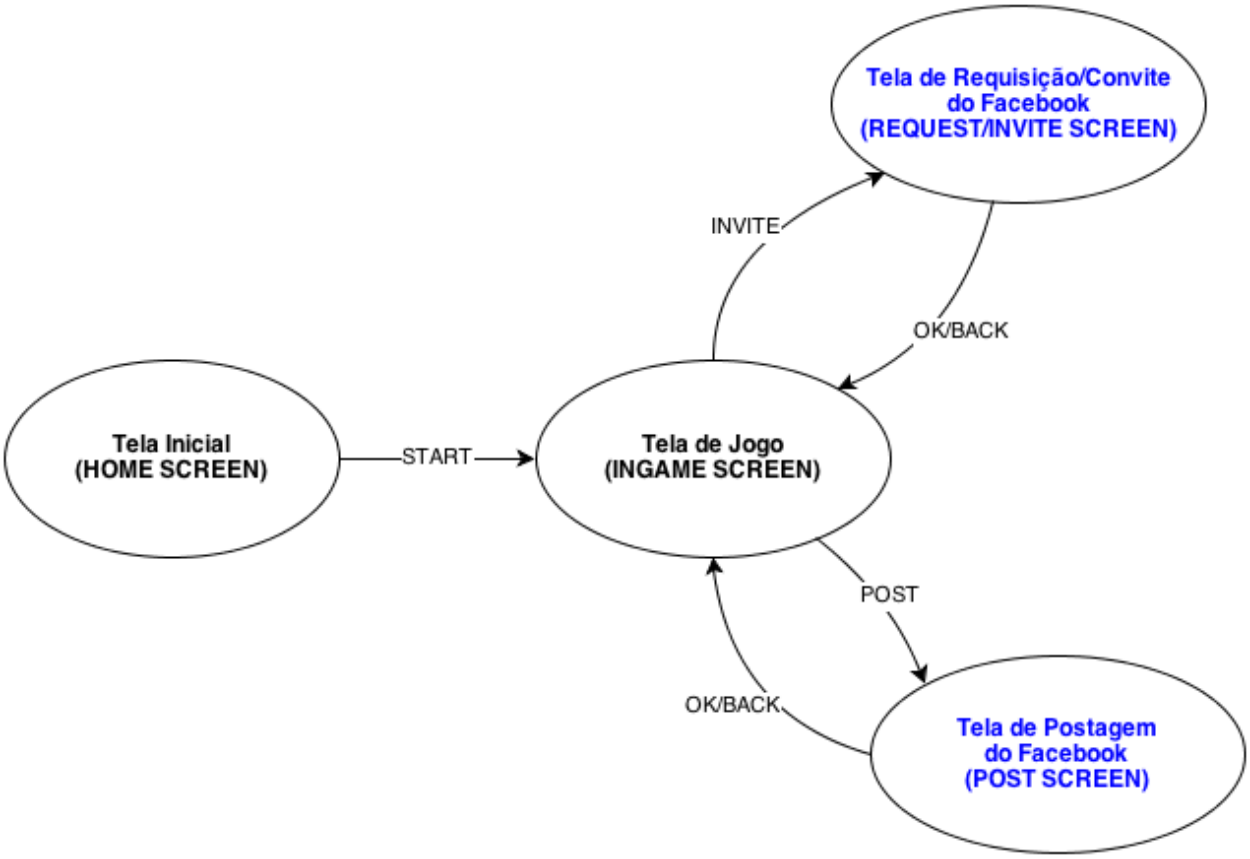

Figura 3: Arquitetura do jogo

7.4 Incrementos e Iterações

Após um estudo sobre as funcionalidades do aplicativo descritas na seção 7.2, houve um planejamento sobre quais as iterações seriam cruciais, e que incrementos deveriam ter. A seguir, é descrito cada uma das etapas determinadas. Para cada etapa foi feita uma previsão do tempo que seria gasto, baseado em uma dedicação de 2 horas por dia, 5 dias por semana.

7.4.1 Iteração 1: Desenvolvimento da base do sistema

- Tempo de duração de implementação prevista: 5 dias;
- Data limite prevista: 27 de julho de 2013;
- Data de conclusão: 02 de agosto de 2013.

Ao fim desta iteração, o sistema apresentou as seguintes funcionalidades já integradas:

1. Apresentação do terreno base do jogo, com configurações visuais de interface gráfica pré-determinadas;
2. Controle de câmera, com movimentação sobre o terreno (movimentação para cima, baixo, esquerda, direita e zoom) e limitando o espaço de atuação do jogador.

7.4.2 Iteração 2: Desenvolvimento do controle de menus e Interface Gráfica

- Tempo de duração de implementação prevista: 5 dias;
- Data limite prevista: 03 de agosto de 2013;
- Data de conclusão: 08 de agosto de 2013.

Ao fim desta iteração, o sistema apresentou as seguintes funcionalidades já integradas, atuando conjuntamente com as funcionalidades inseridas na iteração anterior:

1. Inserção dos menus fixos do jogo, atualizando os dados de maneira simplificada de acordo com a interação com o usuário;
2. Mostrar nos menus os dados importantes do jogo, atualizando sempre que necessário.
3. Alternar menus de acordo com o solicitado pelo usuário.

7.4.3 Iteração 3: Desenvolvimento e inserção do módulo de construção de torres

- Tempo de duração de implementação prevista: 10 dias;
- Data limite prevista: 17 de agosto de 2013;
- Data de conclusão: 22 de agosto de 2013.

Ao fim desta iteração, o sistema apresentou as seguintes funcionalidades já integradas, atuando conjuntamente com as funcionalidades inseridas nas iterações anteriores:

1. Criação de modelos bases para as torres;
2. Desenvolvimento básico da inteligência artificial das torres (detectar inimigos, mirar e atirar);
3. Instanciar torres no mapa de acordo com os comandos do usuário;
4. Integrar o módulo de construção de compras com os menus já funcionais, atualizando atributos como recursos e valores de compra.

7.4.4 Iteração 4: Desenvolvimento e controle de *waves* de inimigos

- Tempo de duração de implementação prevista: 10 dias;
- Data limite prevista: 31 de agosto de 2013;
- Data de conclusão: 02 de setembro de 2013.

Ao fim desta iteração, o sistema apresentou as seguintes funcionalidades já integradas, atuando conjuntamente com as funcionalidades inseridas nas iterações anteriores:

1. Inserir inimigos no jogo de acordo com as configurações selecionadas pelo usuário.
2. Desenvolvimento básico da inteligência artificial dos inimigos (encontrar o melhor caminho até o ponto objetivo e percorrer);
3. Integrar as informações das *waves* de inimigos com os menus já funcionais, atualizando atributos como recursos ganhos e vidas perdidas.

7.4.5 Iteração 5: Persistência dos dados em banco de dados no servidor

- Tempo de duração de implementação prevista: 5 dias;
- Data limite prevista: 7 de setembro de 2013;
- Data de conclusão: 12 de setembro de 2013.

Ao fim desta iteração, o sistema apresentou as seguintes funcionalidades já integradas, atuando conjuntamente com as funcionalidades inseridas nas iterações anteriores:

1. Salvar o estado atual do jogo em banco de dados em um servidor;
2. Recuperar estado salvo do jogo do banco de dados e carregá-los no jogo.

7.4.6 Iteração 6: Integração com Rede Social Facebook

- Tempo de duração de implementação prevista: 10 dias;
- Data limite prevista: 21 de setembro de 2013;
- Data de conclusão: 4 de outubro de 2013.

Ao fim desta iteração, o sistema apresentou as seguintes funcionalidades já integradas, atuando conjuntamente com as funcionalidades inseridas nas iterações anteriores:

1. Integrar o sistema com o Facebook, para permitir *login*;
2. Recuperar do Facebook dados importantes (como ID do usuário e lista de amigos) e transmiti-las para o sistema;

7.4.7 Iteração 7: Ajustes finais

- Tempo de duração de implementação prevista: 10 dias;
- Data limite prevista: 5 de outubro de 2013;
- Data de conclusão: 11 de outubro de 2013.

Nesta iteração todos os elementos do jogo receberam ajustes a fim de tornar o jogo mais apresentável e manter uma jogabilidade apropriada. O jogo foi testado e foram corrigidas configurações como valores de compra de construções, força, e velocidade de torres e inimigos e dificuldade dos níveis. Foi feito um balanceamento

dos elementos principais, para que a experiência de jogabilidade fosse a melhor possível.

7.4.8 Iteração 8: Inserção das telas do jogo

- Tempo de duração de implementação prevista: 5 dias;
- Data limite prevista: 12 de outubro de 2013;
- Data de conclusão: 18 de outubro de 2013.

Nesta iteração foram inseridas ao sistema telas básicas da arquitetura do jogo, como a tela de início, *pause* e opções.

7.4.9 Iteração 9: Inserção de elementos gráficos

- Tempo de duração de implementação prevista: 10 dias;
- Data limite prevista: 26 de outubro de 2013;
- Data de conclusão: 4 de novembro de 2013.

Esta iteração foi tratada no início como opcional, pois seria desenvolvida somente se houvesse tempo disponível. A apresentação visual do jogo é importante, mas não foi o objetivo principal deste projeto. Dado que a criação gráfica dos elementos 3D demanda muito tempo e conhecimento razoável de ferramentas de modelagem específicas, os objetos receberam elementos gráficos bem definidos dependendo do andamento das iterações principais do projeto. Mesmo com o atraso sofrido nas iterações anteriores, foi possível concluir esta etapa e apresentar as seguintes funcionalidades já integradas, atuando conjuntamente com as funcionalidades inseridas nas iterações anteriores:

1. Apresentação das torres e inimigos como objetos 3D desenvolvidos e desenhados em ferramentas de modelagem gráfica (Blender);
2. Inserção de elementos gráficos ao cenário do jogo, tornando o visual do jogo mais agradável.

7.5 Acessos e Conexões

O jogo deste projeto foi desenvolvido para a plataforma *web* utilizando a *game engine* Unity3D. Esta *engine* é capaz de exportar todo um projeto para um arquivo executável compatível com navegadores (*browers*) mais populares do mercado, tal como Google Chrome, Firefox e Internet Explorer. Este arquivo executável pode ser de dois tipos: formato *Flash* (necessário instalação do *plugin Adobe Flash*), ou no formato unity3d (necessário instalação de um *plugin* proprietário e do *web player* disponibilizado pelo próprio desenvolvedor da *engine* Unity3D).

Para que o aplicativo pudesse ser integrado com o Facebook foi necessário criar uma conta *Facebook Developers.* A partir desta conta, o aplicativo foi registrado, ganhando assim um código de identificação única para poder ser integrado à rede social. Esta identificação então é inicializada no Unity Facebook SDK, que passa a interagir com a API do Facebook diretamente do projeto Unity.

Concluídos estes passos, o jogo passa a ser disponibilizado para ser executado a partir do Facebook.

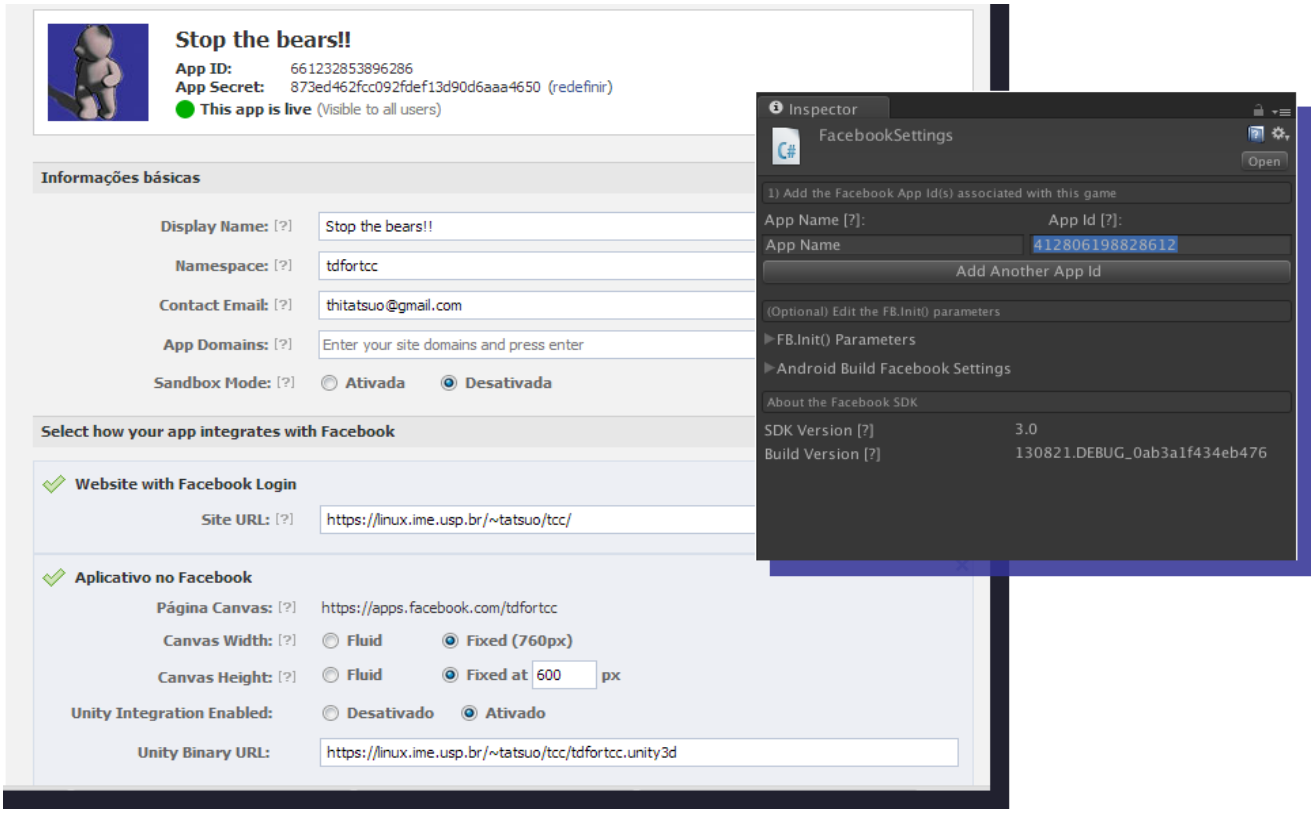

Figura 4: Cadastro do aplicativo no Facebook e integração com Unity3D

7.5.1 Acesso

Quando o jogo é solicitado pela página do Facebook, o executável hospedado no servidor da rede Linux é carregado e executado dentro do *canvas* (área na página destinada para renderização dinâmica de gráficos). Assim o jogo passa a rodar de dentro do navegador, carregado na página do próprio Facebook. O jogo então possui métodos para acesso às páginas PHP externas também hospedadas na rede Linux, que servem de meio de conexão com um banco de dados que persiste informações dos jogos de cada usuário.

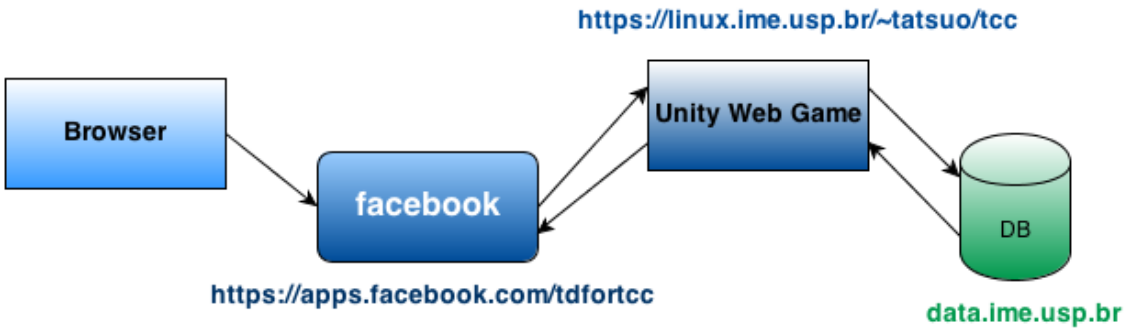

Figura 5: Estrutura de acesso do jogo

7.5.2 Autorizações

Na primeira vez que o jogo é acessado, um módulo existente no Unity Facebook SDK entra em execução, acessando o oAuth2.0.

O oAuth2.0 é um protocolo aberto utilizado pelo Facebook que possibilita a autorização segura entre aplicações *web*, *desktop* e móveis. A aplicação pede permissão de acesso a dados para o usuário, que tem a opção de conceder ou não, sem que para isso tenha que informar a senha. Essa permissão independe da senha do usuário, ou seja, mesmo que seja alterada a permissão continuará válida. Além disso, a permissão dada à aplicação pode ser revogada a qualquer momento.

Para que o jogo possa ser iniciado, o usuário precisa permitir acesso a informações como o primeiro nome, último nome e lista de amigos.

7.5.3 Interatividade

Quando o usuário autoriza que a aplicação acesse os seus dados, a Unity Facebook SDK passa a utilizar métodos da API principal do Facebook para tornar o jogo mais interativo entre os usuários.

Para este projeto foram utilizados dois meios principais para explorar a interatividade e competitividade do jogo social.

7.5.3.1 *INVITE/REQUEST*

A partir do jogo, ao acionar um botão de convite, o seguinte método da Facebook API é chamado:

```
FB.AppRequest(
   message: "Come to see this amazing game!",
   title: "Come join me with Unity!",
   callback: FeedCallback
 );
```

Este método aciona uma chamada para resgatar a lista de contatos do usuário e mostrar esta lista em uma janela. O usuário então pode selecionar alguns dos seus contatos para enviar um convite ou requisição para conhecer o jogo.

Figura 6: Janela de seleção de contatos

7.5.3.2 *POST*

Uma outra funcionalidade que o jogo possui é provida pelo método Feed:

```
FB.Feed(
     link: "https://apps.facebook.com/tdfortcc/",
     linkName: "Stop the bears!!",
     linkCaption: "I'm playing this amazing Game!",
     linkDescription: "My current score is " + score + ". I
                                challenge everyone!",
     picture:
"https://linux.ime.usp.br/~tatsuo/tcc/bearIcon128.jpg",
     callback: FeedCallback
);
```

Quando acionado, a aplicação encarrega-se de efetuar uma postagem automática no mural de notificações do usuário, que fica disponível para ser visualizado

por todos os seus contatos, com uma mensagem com sua pontuação e desafiando-os, explorando assim o lado competitivo do jogo.

Figura 7: Janela de postagem no mural de notificações

7.6 Estrutura

O projeto foi estruturado seguindo o paradigma de orientação a objeto. Possui classes que interagem entre si por meio de composições, agregações e herança. A seguir, será explicado qual o papel e funcionalidade das principais classes.

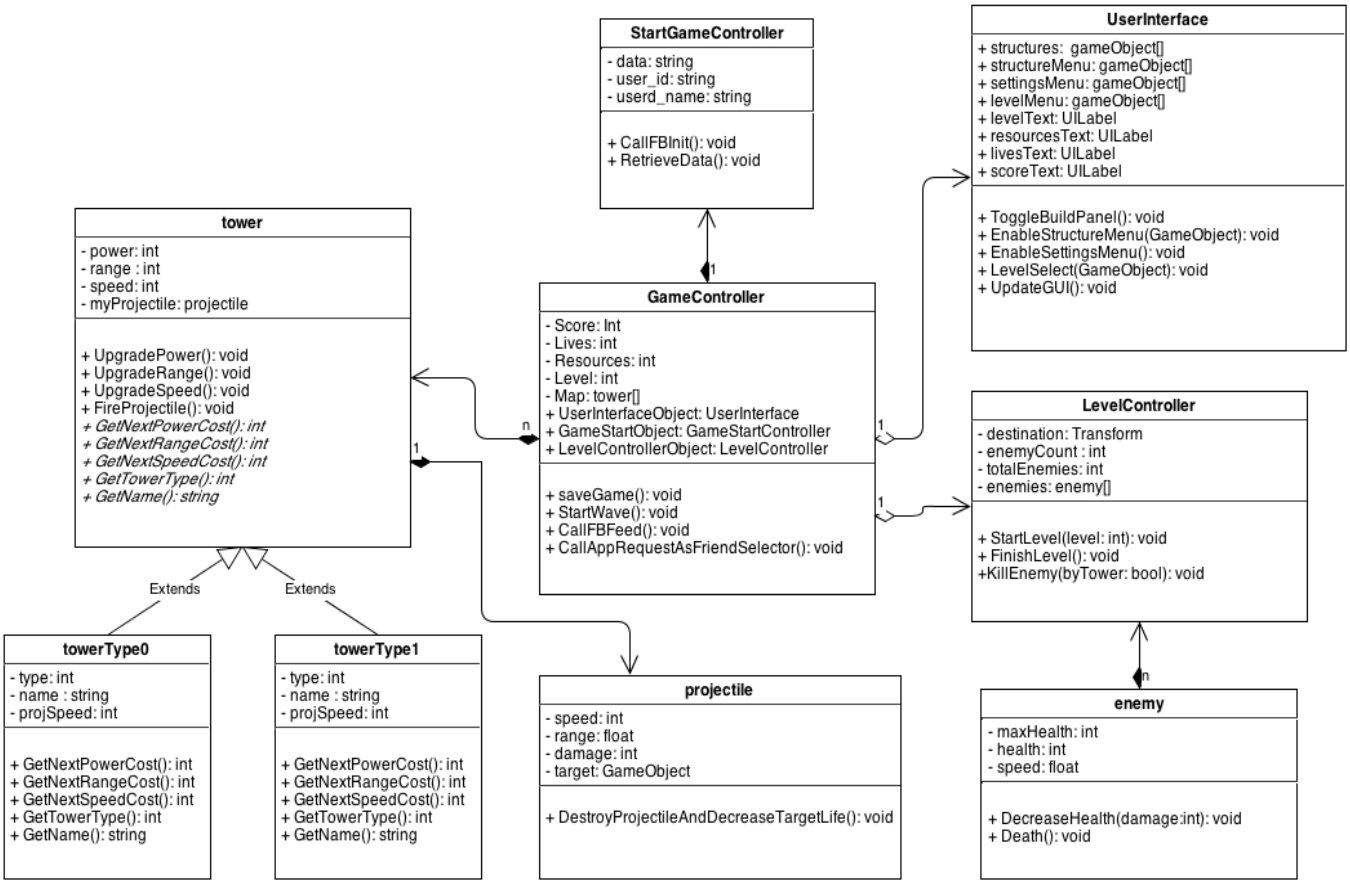

Figura 8: Diagrama de Classes simplificado do projeto

7.6.1 *Start Game Controller*

Classe instanciada na tela inicial do jogo, é responsável por inicializar o objeto principal do Unity Facebook SDK. Mantém a lógica e os métodos de verificação das autorizações concedidas pelo usuário, e carregar previamente as informações básicas.

Responsável também por executar a primeira conexão com o banco de dados (utilizando páginas PHP contendo códigos de acesso ao banco de dado como meio de comunicação), recuperando os dados salvos da última partida do jogador.

Passa então todos estes dados carregados para a classe *GameController*, responsável por controlar as variáveis principais do jogo e jogador.

7.6.2 *Game Controller*

Esta classe controla os atributos principais do jogador e seu jogo. Gerencia e atualiza os atributos de pontos, vida, nível e recursos do jogador. Mantém também atualizada a estrutura de dados que mapeia as torres construídas e seus melhoramentos.

Responsável por controlar e executar as chamadas para os métodos da Facebook API, e manipulação dos dados do jogador no banco de dados. Executa o método *saveGame,* que inicia conexão com o banco de dados para atualizar as informações do jogo do usuário corrente.

Controla também as chamadas para a classe que gerencia as hordas de inimigos e lança gatilhos de eventos para a classe que gerencia os elementos gráficos.

Pode ser considerada como o elemento central do jogo, pois é composto ou agregado dos demais objetos essenciais.

7.6.3 *User Graphic Interface Controller*

Responsável por controlar os elementos gráficos de interação com o jogador, utiliza extensivamente as funcionalidade do *NGUI Framework.*

Instancia e gerencia os botões, mostradores e menus do jogo, além de controlar quando devem ficar visíveis e quando devem ficar ativos. O jogo possui 5 mostradores principais, onde exibe informações sobre quantidade de pontos, recursos, vida, nível atual e usuário conectado. Existem três menus, um de informações das torres, um *menu* de compra de torres e um de interação com o Facebook (contendo os botões de *Invite* e *Post*).

Recebe notificações das outras classes quando um evento é disparado, atualizando os mostradores e estados dos botões e menus.

7.6.4 *Level Controller*

Classe que controla as hordas de inimigos (*waves*). É acionado pelo evento lançado quando o usuário aperta o botão “go”, que inicia um nível do jogo. Esta classe então fica responsável por instanciar os inimigos, e manter o estado do jogo ativo no nível corrente até que todos os inimigos sejam eliminados ou alcancem o ponto objetivo.

Mantêm informações do ponto de origem (*start point*) e ponto objetivo (*objective point*), e determina a quantidade de inimigos e o intervalo de aparição entre dois deles. Para determinar estes valores foram utilizadas funções lineares, onde n é o nível:

- Quantidade de inimigos dado por

  $$q(n) = piso(2.667x + 3.334)$$

- Intervalo de aparição entre dois inimigos em segundos dado por

  $$d(n) = -0.01515x + 2.5151$$

Desta forma, estas funções garantem que quanto maior o nível em que o jogador estiver jogando, maior será a dificuldade, pois os inimigos irão aparecer em maior quantidade e em intervalos de tempos menores.

Para cada inimigo que as torres conseguirem destruir, uma recompensa em recursos é somada ao montante do jogador. O valor desta bonificação é vinte vezes o nível da *wave*.

7.7 Lógica dos Inimigos

A classe que controla os níveis do jogo é responsável por instanciar os objetos inimigos para cada horda. Através do menu, o jogador pode selecionar o nível da fase, e ao apertar o botão de início os inimigos surgem.

Os valores dos atributos dos inimigos são determinados através de funções lineares baseadas no nível escolhido.

Figura 9: Inimigo

Os atributos dos inimigos são:

- Vida: número inteiro que marca quanto de vida tem o inimigo. A cada dano recebido pelas torres este valor vai decrescendo até chegar a zero, quando o inimigo é destruído. O seu valor inicial é determinado pela seguinte função, onde n é o nível da *wave*:

$$v(n) = \text{piso}(18.889x + 11.111)$$

- Velocidade: número real que determina a velocidade com que os inimigos transitam pelo cenário. É determinado pela seguinte função linear, onde n é o nível da *wave*:

$$s(n) = 0.2413x + 0.758$$

O inimigo possui um *script* vinculado a ele que é responsável por determinar seu comportamento. No momento em que é instanciado, o *script* executa chamadas de métodos que fazem o objeto inimigo transitar pelo mapa até o ponto objetivo com a velocidade determinada. O *script* também possui métodos que verificam se o objeto foi alvo de alguma torre, e reduz a vida caso um projétil seja disparado em sua direção.

Para decidir que caminho tomar até o ponto objetivo, foi utilizado uma biblioteca do Unity3D*,* o *pathfinder.* Foram determinadas no mapa quais as áreas em que os inimigos podem transitar. Esta área é mostrada na figura a seguir, representada pelos retângulos marcados.

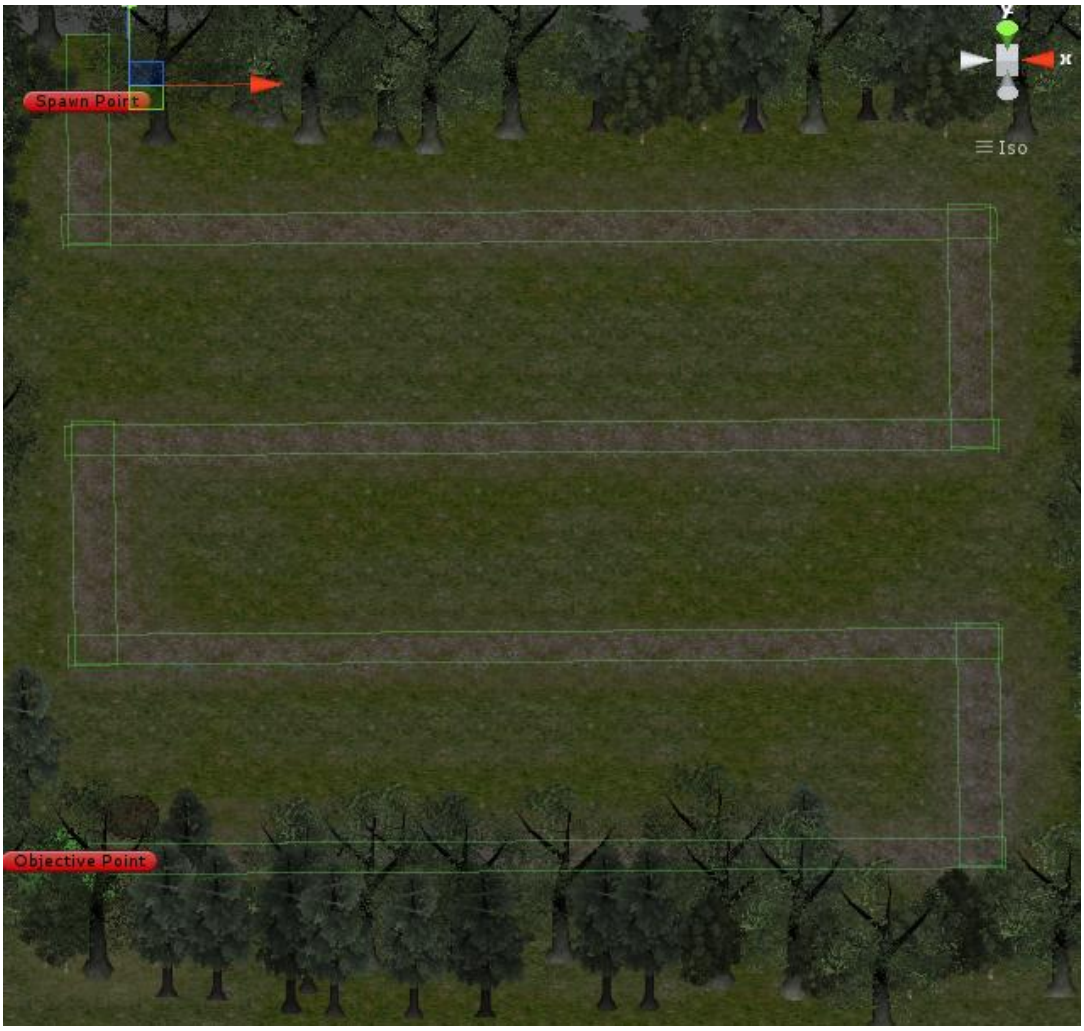

Figura 10: Caminho dos inimigos

Com esta área mapeada, a biblioteca *pathfinder* utiliza internamente o algoritmo de busca A* para determinar o melhor caminho dentro destes limites a partir de pontos de origem e destino pré-determinados (*spawn point* e *objective point* respectivamente).

O modelo gráfico do inimigo é um urso, e foi retirada de um exemplo disponibilizado como modelo no *site* do Unity3D. Para as animações utilizou-se uma classe auxiliar existente nos recursos disponibilizados pela Unity3D. Esta classe, nomeada de *animator* auxilia na animação gráfica dos objetos.

A animação do urso é mapeada como uma máquina de estados. Dependendo de sua ação, que envolve ficar parado, correr ou andar, girar no eixo, ou virar para a esquerda ou direita, seus modelos gráficos são carregado. Assim, através do algoritmo de busca de caminho as direções que o objeto inimigo tem que seguir são determinadas, e a animação gráfica é renderizada de acordo com as transições da máquina de estados. A figura seguinte ilustra a máquina de estados utilizada pela classe *animator*.

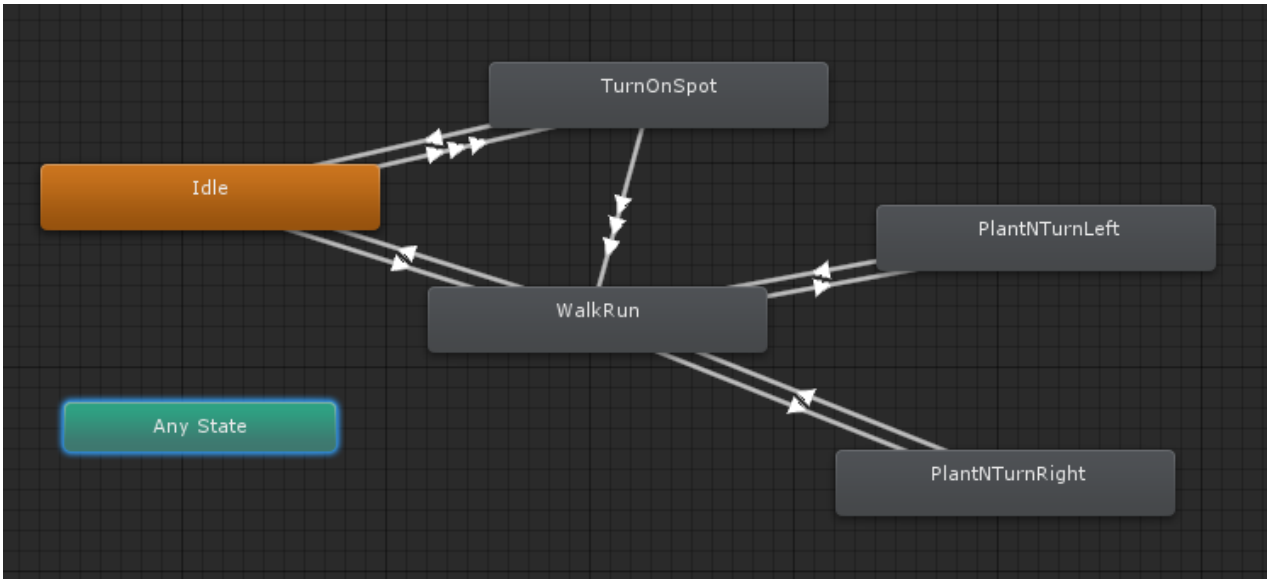

Figura 11: Máquina de estados do comportamento gráfico do inimigo

7.8 Lógica das Torres

As torres constituem os objetos sob o qual o jogador tem controle. Elas têm como objetivo atirar em inimigos que adentrarem em sua área de atuação para tentar eliminá-los antes que alcancem o ponto objetivo.

Um menu no canto inferior contém as opções de torres disponíveis. Para este projeto foram implementadas dois tipo de torres: um que atira projéteis de flecha, e outro que atira balas de canhão. Cada um destes dois tipos tem custos de compra e melhoramentos distintos, e performances diferenciadas. No jogo, a torre de canhão é mais cara para construir e melhorar, porém possui ataque maior e menor velocidade.

Como explicado no item 6.4, cada torre têm três atributos principais: força, área de ação e velocidade de ataque. Quando uma torre é instanciada, cada um destes atributos recebem um valor inteiro igual a 1, correspondente ao nível deste atributo

para esta torre. Com estes níveis são definidos como cada torre interage individualmente com os demais elementos do jogo em eventos determinados. Os níveis de cada atributo podem sofrer melhoria em troca de recursos adquiridos durante o jogo, assim o desempenho da torre pode ser aperfeiçoada. Os níveis podem crescer entre os números inteiros de 1 (valor inicial) a 9 (valor final).

7.8.1 Força

Através do nível do atributo força é determinado qual o valor do dano que cada tiro causa em um inimigo alvo. O dano é um valor inteiro, calculado a partir de uma função linear dada por:

$$f(n) = 5.111x - 1.111 \quad \text{(para a torre de flecha)}$$

$$f(n) = 5.777n + 2.222 \quad \text{(para a torre de canhão)}$$

onde n é o nível do atributo força. Assim, quando uma torre estiver com um inimigo como alvo, ao disparar um projétil, é calculado o dano dado pelo valor de f(n) em função do nível do atributo força, e este valor é subtraído da vida do inimigo.

7.8.2 Área de ação

Com o valor deste atributo é determinado qual a área em que uma torre consegue identificar alvos. Esta área é dada por uma circunferência com centro coincidente com o ponto central da torre. Assim, o raio desta circunferência é calculado através de uma função linear dada por:

$$a(n) = 3.333x + 16.667 \quad \text{(para a torre de flecha)}$$

$$a(n) = 2.222n + 12.778 \quad \text{(para a torre de canhão)}$$

onde n é o nível do atributo.

Para a detecção dos inimigos, uma esfera invisível com raio igual ao valor de a(n) em função do nível é anexado à torre. A esta esfera então é atribuído um colisor,

classe pertencente à biblioteca de física da *engine* Unity3D. Este colisor detecta quando um objeto colide com ela, acionando um gatilho (*trigger*) e lançando um evento especial. Este evento verifica se o elemento que colidiu é um inimigo, e se for, passa a considerá-lo como um alvo e começa a disparar projéteis em sua direção até que seja eliminado ou até que saia da área de colisão. A saída do inimigo da área de colisão também dispara um gatilho, fazendo com que o então objeto não seja mais um alvo da torre. A figura seguinte ilustra uma torre, com sua esfera detector de colisões.

Figura 12: Modelo da torre e colisor

7.8.3 Velocidade de ataque

Quando uma torre tem um inimigo como alvo, passa a atirar projéteis em sua direção. A taxa de tiro dado por unidade de tempo é determinada pela velocidade de ataque. A partir do nível deste atributo é calculado, em segundos, o intervalo entre dois disparos, dado por:

$$v(n) = -0.0889x + 1.588 \quad \text{(para a torre de flecha)}$$

$$v(n) = -0.111n + 2.111 \quad \text{(para a torre de canhão)}$$

onde n é o nível do atributo. Então, ao melhorar continuamente este atributo (acrescentando uma unidade ao nível), a função v(n), por ser estritamente decrescente, devolverá um valor cada vez melhor, diminuindo o intervalo entre dois disparos consecutivos, aumentando a taxa de disparos.

7.8.4 Melhorias

Os atributos iniciam com nível 1. Conforme o jogador vai adquirindo recursos, é possível melhorar cada atributo individualmente, subindo o seu nível gradativamente (até o limite de nível 9). O custo de cada melhoria é dado por uma função do segundo grau, que toma como parâmetro o nível atual. A torre do tipo canhão, por ser mais forte, possui uma curvatura mais acentuada, mantendo custos mais elevados a cada melhoria. As funções são dadas por:

$$f(x) = 100x^2 \quad \text{(para a torre de flecha)}$$

$$f(x) = 100x^2 + 100x \quad \text{(para a torre de canhão)}$$

Ainda há para o jogador a opção de vender uma torre construída. Em retorno, o jogador recebe de volta 50% dos recursos gastos, incluindo o valor de construção e melhorias efetuadas.

7.9 Matriz Mapa

O cenário gráfico é mapeado como uma matriz de números inteiros. Esta matriz indica quais são as posições em que as torres podem ser construídas. Tem dimensões totais de 15 linhas e 15 colunas.

Figura 13: Matriz representando áreas construíveis

As células da matriz que coincidem com o caminho por onde os inimigos passam recebem valor -1, indicando que nesta posição não é possível construir uma torre. As demais células recebem o inteiro 0, representando posições abertas que podem receber uma instância do objeto torre.

Ao construir uma torre, a célula correspondente à posição da estrutura recebe um inteiro que codifica a torre. A cada melhoria de um atributo, este valor é atualizando

para manter o mapeamento das torres consistente para persistência no banco de dados e posteriormente recuperar um jogo salvo.

A codificação de uma torre para um inteiro é dado da seguinte forma:

- A torre é representada por um inteiro de 4 dígitos;
- O primeiro dígito indica o tipo de torre, 0 para torre do tipo flecha e 1 para torre do tipo canhão;
- O segundo dígito indica o nível do atributo força (inteiro de 1 a 9);
- O terceiro dígito indica o nível do atributo área de ataque (inteiro de 1 a 9);
- O quarto dígito indica o nível do atributo velocidade de ataque (inteiro de 1 a 9).

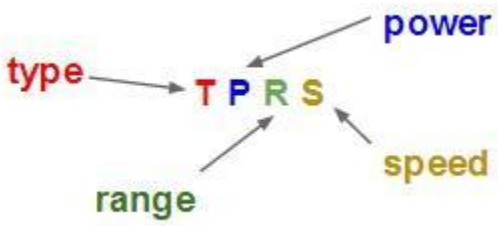

Figura 14: Representação de uma torre em um número inteiro

7.10 Persistência dos Dados

Todo jogador que possuir um *login* válido na rede social Facebook terá acesso ao jogo, bastando aceitar as permissões de acesso aos dados básicos, como nome e lista de amigos.

O jogo então passa a salvar os dados do jogador em um banco de dados relacional hospedado no servidor data.ime.usp.br, pertencente ao Instituto de Matemática e Estatística da Universidade de São Paulo. Para a disciplina MAC0439 - Laboratório de Banco de Dados foram disponibilizadas bases de dados relacionais para execução dos exercícios de aula. Com consentimento da professora Kelly Rosa Braghetto, foi possível utilizar a mesma base de dados para persistir os dados deste projeto para fins de teste.

O jogo utiliza métodos e a classe *WWW* existente na biblioteca do Unity3D para acessar páginas HTML e PHP da internet. Códigos PHP foram desenvolvidos para a

conexão com o banco de dados e para efetuar as requisições necessárias. Estas páginas contendo estes códigos foram hospedadas na rede Linux, utilizando o espaço disponibilizado para usuário, no endereço https://linux.ime.usp.br/~tatsuo. Essencialmente estas páginas aceitam métodos *POST* para receber valores e então executar requisições (*queries*) do tipo *SELECT*, INSERT e *UPDATE* no banco de dados. Utilizou-se o gerenciador de banco de dados PostgreSQL para recuperação dos dados.

Ao iniciar o jogo, a página get.php é acessada, onde informações do jogador previamente salvas são resgatadas através de requisições do tipo *GET,* para carregar o jogo como os atributos atuais. Se for a primeira vez que o jogador estiver jogando, então a página insert.php é acessada, onde uma requisição do tipo *INSERT* é executada, inserindo no banco de dados uma nova entrada de jogador com as configurações iniciais. Diversos momentos do jogo, como quando se inicia um nível de inimigos ou quando se constrói uma torre, o jogo é salvo acessando a página update.php, que efetua requisições do tipo *UPDATE* para atualizar os dados do jogador.

Foi necessário uma única relação para armazenar todos os dados necessário para salvar o jogo de um usuário. Como chave primária utilizou-se a identificação do usuário do Facebook, que é única. A relação possui os seguintes atributos:

ID	*Scores*	*Lives*	*Resources*	*Level*	*Map1*	*Map2*

- *ID*: número de identificação do usuário no Facebook (chave primária);
- *Scores*: pontuação atual do jogador;
- *Lives*: quantidade de vidas que o jogador possui;
- *Resources*: quantidade de recursos que o jogador possui;
- *Level*: nível máximo do jogo em que o jogador conseguiu chegar;
- *Map1*: mapeamento do mapa de estruturas do jogar;
- *Map2*: segunda parte do mapeamento do mapa de estruturas.

O mapa que registra todas as estruturas construídas com suas respectivas posições e melhorias é representado por uma matriz de inteiros como mostrado no

tópico anterior. Para persistência desta estrutura no banco de dados relacional foi necessário convertê-lo para o tipo *string*. Para isto, utilizou-se o modelo CSV (*comma separated values*). Para isto bastou transformar a matriz de inteiros em uma *string* (vetor de caracteres) com os valores seguidos de forma orientada a linhas, separada por vírgulas.

Foi necessário após isto dividir o mapa em duas *strings*, de modo que cada metade é guardada na relação em colunas separadas. Isto foi necessário pois o mapa codificado no tipo *string* gera um dado contínuo muito grande, incapaz de ser processado de uma única vez pelo método *post* do PHP.

7.11 Testes

Durante o desenvolvimento do projeto, a cada iteração concluída, uma nova versão era atualizada no servidor da rede Linux. Durante este tempo o jogo esteve em modo *sandbox* ou modo de teste, de maneira que não estava aberto aos usuários regulares do Facebook. Neste modo, apenas o desenvolvedor da aplicação e testadores selecionados têm acesso à aplicação.

Foram selecionados 6 usuários como testadores, que durante o desenvolvimento testavam cada versão nova, apontando falhas e sugestões de melhorias.

Ao fim do projeto o jogo passou a ficar aberto para que qualquer usuário pudesse jogar. Obtiveram-se muitos retornos positivos com sugestões construtivas para melhoria.

8. Análise dos Resultados Obtidos

Como resultado final obteve-se um protótipo jogável e implantado na página do Facebook. Dentro do proposto, foi possível cobrir os principais itens propostos, com um resultado melhor do que esperado.

Figura 15: Jogo sendo executado no navegador na página do *Facebook*

A princípio, o visual gráfico do jogo foi considerado como um complemento secundário que seria adicionado somente na viabilidade de tempo. Dado que boa parte de um projeto de um jogo é dedicado à arte gráfica por exigir mais tempo de desenvolvimento, já havia conformidade em manter a interface gráfica o mais simples possível e ater-se à implementação da lógica do jogo. Com o auxílio de ótimas ferramentas de arte, tais como o Blender e o Adobe Photoshop, e com os eficientes editores gráficos que a própria *engine* Unity3D disponibiliza, foi possível desenvolver ótimos elementos visuais para o jogo que agregaram uma melhor qualidade.

Por se tratar de um jogo com gráficos em três dimensões, a renderização dos elementos gráficos exigem um grau de processamento considerável da máquina em que estiver sendo executada. Por isto, acessar a aplicação em conjunto com muitos outros aplicativos executados em paralelo fez com que a performance do jogo diminuísse, como esperado.

Um fato que prejudicou alguns testes foi sobre a disponibilidade do *plugin* proprietário para suporte de arquivos executáveis no formato *unity3d* para navegadores no sistema operacional Linux. A desenvolvedora da *engine* Unity3D atualmente dá suporte para aplicações *web* nos principais navegadores que estiverem sendo

executados em sistemas *Windows* e *Mac*. Desta forma, o jogo não é compatível com navegadores nos sistemas Linux, o que prejudicou disseminação do jogo. Um *plugin* direcionado aos sistemas Linux já foi anunciado pela desenvolvedora, sendo que o jogo passará então a ser compatível.

9. Conclusão e Considerações finais

De uma maneira geral, o proposto se mostrou um projeto muito grande para ser desenvolvido individualmente no tempo disponibilizado. Foi necessário dedicar muito mais tempo que o exigido pela carga horário da disciplina para que um resultado satisfatório no desenvolvimento do projeto pudesse ser alcançado.

Durante o desenvolvimento, muitos contratempos surgiram e tomaram um tempo considerável para serem contornados, de modo que o planejamento inicial e cronograma mudaram muitas vezes.

Dentre estes contratempos, o que mais tomou tempo foi sobre as rígidas exigências da política do Facebook quanto ao servidor hospedeiro dos arquivos que são carregados. É exigido um servidor com suporte ao protocolo HTTPS (HyperText Transfer Protocol Secure), para conferir segurança ao acesso dos dados. Os testes iniciais eram feitas em servidor local, sendo que para configurar e se adequar a todos os requisitos necessários exigidos para o servidor pelo Facebook demandou muito tempo.

Outro problema impactante foi quanto ao suporte de páginas PHP pelo servidor. Testes em servidor local não apresentaram grandes problemas no acesso a páginas PHP com métodos de requisições em banco de dados através do gerenciador PostgreSQL. Porém, ao migrar ao servidor real da rede Linux houve incompatibilidades de configurações quanto ao acesso a bibliotecas e pacotes que mantém o suporte a métodos PostgreSQL. Foi necessário contatar os administradores da rede para solicitar a disponibilidade os pacotes necessários. Durante o tempo entre a aprovação deste pedido e a disponibilização efetiva houve um período de incerteza se o projeto deveria sofrer mudanças radicais sobre esta funcionalidade. Felizmente os pacotes puderam ser instalados no servidor e o jogo passou a funcionar como esperado.

Observando o trabalho concluído como um todo, foi possível entender e aprender como um projeto deste porte exige um conhecimento sólido em diversas áreas, e como é importante conciliá-los de maneira adequada. É importante para um projeto deste tipo manter sempre o foco no planejamento proposto, e em cada etapa reavaliar as prioridades e os tempos de execução das iterações. Manter um projeto grande dentro do cronograma criado não é uma tarefa fácil, por isso é importante que

periodicamente o planejamento seja reavaliado para deixar o projeto o mais flexível possível sem se desviar da proposta inicial.

10. Futuro deste Trabalho

Futuramente este projeto pode ser melhorado e ampliado, com a inserção de novas funcionalidades ou elementos. Foi desenvolvido para ser facilmente estendido, de modo que atualizações e melhorias podem ser feitas sem muitos esforços, e sem necessidade de alteração na base principal do jogo. Dentre as próximas melhorias pode-se citar:

- Melhorar balanceamento do jogo;
- Adicionar efeitos sonoros;
- Adicionar novos inimigos e torres;
- Adicionar novas funcionalidades como reiniciar o mapa ou alterar a velocidade do jogo

11. Referências Bibliográficas

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Algoritmos - Teoria e Prática. 5ª Tiragem, Editora Campus, 2001.

[2] Russell, S. & Norvig, P. Articial Intelligence - A Modern Approach. PrenticeHall, 1995

[3] Elmasri, R. & Nsvathe, S.B. Sistemas de Banco de Dados. 6ª ed., Pearson, 2011.

[4] Crescimento das redes sociais tomam conta de 21% do planeta Terra. Disponível em: www.logicadigital.com.br/blog/crescimento-das-redes-sociais. Acesso em: maio de 2013.

[5] Brasil usa mais redes sociais em smartphones que países do BRICS e Estados Unidos. Disponível em: http://www.nielsen.com/br/pt/nielsen-pressroom/2013/brasil-usa-mais-redes-socias-em-smartphones-que-paises-dos-brics-e-estado-unidos.html . Acesso em junho: de 2013.

[6] Brazil social gaming market to hit $238 million by 2014. Disponível em: http://venturebeat.com/2011/11/17/brazil-social-gaming-market-to-hit-238-million-by-2014/. Acesso em: julho de 2013.

[7] L. Alves. Mercado de jogos casuais vai ultrapassar o mil milhões em 2013. Disponível em: http://www.eurogamer.pt/articles/mercado-de-jogos-casuais-vai-ultrapassar-o-mil-milhoes- em-2013 . Acesso em: julho de 2013.

[8] N. Dáuer. Mulheres são o principal público dos “jogos sociais”. Disponível em: http://www.geek.com.br/posts/12325-mulheres-sao-o-principal-publico-dos-jogos-sociais . Acesso em: julho de 2013.

[9] PopCap Social Gaming Research Results. Disponível em: http://www.infosolutionsgroup.com/2010_PopCap_Social_Gaming_Research_Results.pdf Acesso em: julho de 2013.

[10] O boom dos jogos Sociais. Disponível em: http://www.canalprogramadoresdejogos.com.br/2010/11/artigo-o-boom-dos-jogos-sociais.html. Acesso em: julho de 2013.

[11] G.R. Mateus e A.F. Loureiro. Introdução à Computação Móvel. Departamento de Ciência da Computação da Universidade Federal de Minas Gerais. Disponível em: homepages.dcc.ufmg.br/~loureiro/cm/docs/cm_livro_1e.pdf. Acesso em: maio de 2013.

[12] G.S. Tonin. Tendências em Computação Móvel. Departamento de Ciências da Computação do Instituto de Matemática e Estatística da Universidade de São Paulo. 2012. Disponível em: grenoble.ime.usp.br/~gold/cursos/2012/movel/ mono-1st/2305-1_Graziela.pdf. Acesso em: maio de 2013.

[13] Uma startup de 2 milhões de dólares na ponta dos seus dedos. Disponível em googlebrasilblog.blogspot.com.br/2011/04/uma-startup-de-2-milhoes-de-dolares-na.html.Acesso em: maio de 2013.

[14] Crescimento das redes sociais tomam conta de 21% do planeta Terra. Disponível em: www.logicadigital.com.br/blog/crescimento-das-redes-sociais. Acesso em: maio de 2013.

[15] Número de usuários de Internet móvel no Brasil vai chegar a 110 milhões em 2015. Disponível em: http://adrenaline.uol.com.br/internet/noticias/13218/numero-de-usuarios-de-internet-movel-no-brasil-vai-chegar-a-110-milhoes-em-2015. Acesso em: maio de 2013.

[16] PRESSMAN, R.S. Engenharia de Software, Sexta Edição. Editora MCGrawHill: Porto Alegre, 2010.

[17] Under the covers of OAuth 2.0 at Facebook. Disponível em: http://www.sociallipstick.com/?p=239. Acesso em: maio de 2013.

[18] NGUI: Next-Gen UI kit. Disponível em: http://www.tasharen.com/?page_id=140. Acesso em: novembro de 2013.

[19] Facebook Developers. Disponível em: http://portuguese.unity3d.com/ . Acesso em: novembro de 2013.

[20] Facebook SDK for Unity. Disponível em: https://developers.facebook.com/docs/unity/ . Acesso em: novembro de 2013.

[21] Unity - Game Engine. Disponível em: http://portuguese.unity3d.com/ . Acesso em: novembro de 2013.

12. Parte subjetiva

12.1 Desafios e frustrações

Um projeto deste porte com certeza me traria muitos desafios e já sabia que teria que enfrentá-los. Mas de fato foram muito mais recorrentes do que eu esperava.

No início do projeto a intenção era desenvolver este projeto em grupo. Faria em dupla com meu colega Felipe Yamaguti. As propostas iniciais foram discutidas durante a fase de proposta de trabalho. Como somos jogadores convictos, tínhamos ideias muito criativas para um jogo. Muitas delas fora da realidade do escopo do projeto, sendo extremamente inviáveis pela dificuldade ou pelo tamanho.

Por fim, o tema aqui descrito neste trabalho foi escolhido e apoiado junto com o professor orientador Marco Gubitoso. Satisfeitos com a escolha, imaginamos e documentamos todas as funcionalidades que gostaríamos de implementar. Infelizmente, no início do segundo semestre da disciplina, o Felipe Yamaguti recebeu uma proposta para realizar um intercâmbio no exterior. Passei então a manter o projeto sozinho. Neste ponto do desenvolvimento o projeto então parecia ser inviável novamente, mas como já havia sido iniciado não valeria pensar em um novo. Tive que diminuir o escopo do proposto para se adequar novamente, retirando algumas funcionalidades que já estavam definidas. E disto veio a primeira frustração: estava tão satisfeito com as funcionalidades definidas que decidir o que manter foi muito difícil. E estas escolhas acabaram me trazendo muito mais desafios. Decidi manter boa parte do projeto inicial de forma que foi necessário trabalhar muito mais do que o exigido para este trabalho.

Além dos imprevistos que surgiram por conta das exigências do Facebook quanto a segurança e do servidor não suportar inicialmente métodos de PostgresSQL em PHP como explicado na seção 9, mais dois pontos me trouxeram grandes desafios. Um projeto de jogo pode ser dividido basicamente em três partes: *game design*, arte e programação. O foco inicial era me ater à programação, mas não seria um jogo satisfatório sem as duas outras partes. Então tive que estudar o mínimo das outras duas áreas para poder desenvolver um projeto mais completo. E isto não foi uma tarefa fácil.

O conteúdo teórico para criação de jogos (*game design*) é muito vasto e envolve muitas outras áreas. O básico do que eu precisava era aprender a melhor forma de balancear as variáveis do jogo para torná-lo desafiador e atrativo, sem deixá-lo muito difícil. Esta é uma tarefa muito difícil que não consegui cumprir como esperado. O jogo no fim ficou muito fácil de jogar, e alterar minimamente uma variável desbalanceava totalmente o jogo todo.

Sobre a arte dos elementos gráficos, dediquei um bom tempo aprendendo a trabalhar com as ferramentas de modelagem Blender e de edição de imagem Adobe Photoshop. Como um fator impactante para o sucesso de um jogo vem dos efeitos gráficos, investi tempo e trabalho neste ponto, para tornar o jogo mais agradável visualmente. O Unity3D possui um suporte muito bom para confecção da parte gráfica com um editor gráfico próprio e ferramentas de animação que me auxiliaram bastante nesta etapa.

12.2 Disciplinas relevantes para o desenvolvimento do trabalho

Muitos conhecimentos adquiridos nas disciplinas cursadas durante a graduação tiveram grande importância. Dentre as que tiveram maior relevância foram:

- **MAC0110 - Introdução a Computação, MAC0122 - Princípios de Desenvolvimento de Algoritmos e MAC0323 - Estruturas de Dados:** ofereceram toda uma base sólida sobre lógica de programação, sintaxes de programação, algoritmos eficiente e estrutura de dados, sem o qual não seria possível se aprofundar nos tópicos estudados e implementados para este projeto.

- **MAC0425 - Inteligência Artificial:** ofereceu o conhecimento necessário para implementação da lógica do inimigo do jogo, com utilização de algoritmos de busca de melhor caminho e noções de solucionador de problemas gerais.

- **MAC0338 - Análise de Algoritmos:** com os estudo desenvolvidos nesta disciplina foi possível analisar o desempenho de meus próprios códigos e melhorá-los tornando-os mais eficientes por meio de estilos e adaptações de algoritmos conhecidos.

- **4310126 - Física I**: cálculo de trajetórias e velocidades foram facilitadas com o conteúdo ensinado nesta disciplina, que ensinou assuntos ligados a cinemática de corpos e colisões.

- **MAC0211 - Laboratório de Programação I e MAC0242 - Laboratório de Programação II:** Nestas disciplinas temos o contato com o desenvolvimento de um projeto mais completo, e aprendemos a importância de um bom planejamento, boa documentação e estruturação dos códigos. Foi de grande importância, pois foi o aprendido nestas disciplinas que foi mais aplicado a este projeto, como o contato com bibliotecas, pacotes e ferramentas externas, integração entre os *frameworks*, e primeiro contato com o paradigma orientado a objetos.

- **MAC0328 - Algoritmos em Grafos e MAC0414 - Linguagens Formais e Autômatos:** a modelagem dos estados do jogo em sua arquitetura, e das animações dos inimigos em seus movimentos basearam-se no conceito de máquina de estados ou autômatos, assuntos ligados à ementa destas disciplinas.

- **MAC0451 - Tópicos Especiais em Desenvolvimento para Web:** um dos assuntos tratados nesta disciplina era sobre a ferramenta web disponibilizada pelo Facebook, onde tive o primeiro contato e interesse sobre o assunto. Pude aprender o básico sobre os requisitos necessários para integrar aplicações à rede social Facebook.

- **MAC0426 - Sistemas de Bancos de Dados e MAC0439 - Laboratório de Bancos de Dados:** disciplinas importantes onde aprendi a modelar um banco de dado e manipular as informações de maneira segura e eficiente, para manter os estados dos jogos consistentes para serem carregados e atualizados sempre que fossem solicitados.

- **MAC0332 - Engenharia de Software:** através desta disciplina adquiri o conhecimento necessário para conduzir o projeto de forma eficiente e adequada, utilizando técnicas de planejamento e avaliação de desempenho, separando as etapas do desenvolvimento do *software*, utilizando metodologias e organização. Importante para manter o projeto no seu objetivo real, acompanhando, avaliando e guiando o andamento do projeto.

De maneira geral, cada aprendizado adquirido durante o curso foi importante, pois além do conteúdo técnico, me ensinou como abordar os problemas, pensar em soluções, estudar os conceitos, aplicar métodos e persistir nas tentativas de alcançar a melhor solução. Pude então conciliar toda esta experiência adquirida, e mesclar todo conhecimento de maneira a desenvolver este projeto da melhor forma possível, conseguindo ótimos resultados e ficando muito satisfeito.