CENTRO UNIVERSITÁRIO CATARINENSE FACVEST
CURSO DE CIÊNCIA DA COMPUTAÇÃO
TRABALHO DE CONCLUSÃO DE CURSO

DESENVOLVIMENTO DE JOGOS DIGITAIS UTILIZANDO O GAME ENGINE IMPACT

JOÃO DEOCLIDES FAUSTINO DA MOTA

LAGES
2012

CENTRO UNIVERSITÁRIO CATARINENSE FACVEST
CURSO DE CIÊNCIA DA COMPUTAÇÃO
TRABALHO DE CONCLUSÃO DE CURSO

DESENVOLVIMENTO DE JOGOS DIGITAIS UTILIZANDO O GAME ENGINE IMPACT

JOÃO DEOCLIDES FAUSTINO DA MOTA

Projeto apresentado à Banca Examinadora do Trabalho de Conclusão de Curso do Curso de Ciências da Computação para análise e aprovação.

LAGES
2012

EQUIPE TÉCNICA

Acadêmico

João Deoclides Faustino da Mota

Professor Orientador

Profº. Márcio José Sembay, Msc.

Coordenador de TCC

Profº. Márcio José Sembay, Msc.

Coordenador do Curso

Profº. Márcio José Sembay, Msc.

SUMÁRIO

RESUMO

Este trabalho tem por finalidade estudar e divulgar as principais funcionalidades do Game Engine Impact para o desenvolvimento de jogos eletrônicos interativos. Como resultado do estudo, ao final do trabalho, é apresentado um protótipo de jogo digital, desenvolvido com características básicas de produtos similares presentes no mercado, procurando exaltar o potencial do desenvolvimento e produção de jogos digitais.

Palavras Chaves: Game Engine, Impact, Jogos Eletrônicos, Protótipo.

ABSTRACT

This work aims to study and disseminate key features of the Impact Game Engine to develop interactive electronic games. As a result of the study, at the end of the work, we present a prototype digital game, developed with basic features of similar products on the market, seeking to exalt the potential development and production of digital games.

Key Words: Game Engine, Impact, Electronic Games, Prototype.

AGRADECIMENTOS

A toda minha família que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida;
A todos os professores do curso, que foram tão importantes na minha vida acadêmica e no desenvolvimento deste trabalho;
Aos amigos e colegas, pelo incentivo e pelo apoio constante.

LISTA DE ABREVIATURAS

HTML – *HyperText Markup Language*

RIA –*Rich Internet Application*

XHTML – *Extensible HyperText Markup Language*

CSS –*Cascading Style Sheets*

API – Application Programming Interface

DOM – *Document Object Model*

JS – JavaScript

2D – Duas Dimensões

PHP – Hypertext Preprocessor

GDD – Game Design Document

IA – Inteligência Artificial

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS

1. INTRODUÇÃO

Enquanto a maioria dos softwares precisam apenas seguir uma série de requisitos e atender bem os propósitos para os quais foram elaborados, uma característica imprescindível para um jogo é que ele deve ser divertido e agradável de utilizar, uma vez que seu principal objetivo é proporcionar entretenimento para as pessoas. Os jogos computadorizados precisam criar a sensação de imersividade nos usuários, tal característica obtida pela combinação de aspectos artísticos e tecnológicos. (Battaiola, 2001)

A maior dúvida de quem quer desenvolver jogos é começar. Esta pergunta sempre aparece nos fóruns de discussão, já que ninguém nasce sabendo. Só que o problema é que o usuário nem procura informações na internet sobre isso, e já coloca uma pergunta básica num tópico: o que eu devo fazer? Como começar a criar games?

Neste contexto este trabalho tem por objetivo mostrar alguns dos passos necessários para a criação de um jogo, além de realizar uma ampla explicação sobre como o game engine Impact pode ajudar a todos a realizarem um passo para a realização de um sonho.

Foi justamente pensando nesse sonho que desenvolvi e demonstrarei um projeto de um jogo destacando a importância do mesmo para os futuros desenvolvedores de software.

1.1 Justificativa

Criar jogos digitais é um trabalho duro que requer muitas habilidades técnicas, muito planejamento, e um compromisso para completar o projeto. Pensando nisso, vislumbrou-se a criação deste trabalho, sendo não somente um projeto de jogo digital, mas também uma forma de ajudar outros a se aventurarem nesse mundo de desenvolvimento de jogos digitais.

1.2 Importância

Esse trabalho é importante, pois pretende demonstrar que a tecnologia surge a cada dia novos modelos de software, o que nos possibilita fabricar jogos de qualidade e padrão inigualáveis. Qualidade nos jogos é algo que todos querem. Um fator decisivo para criação de novos jogos é acima de tudo satisfazer seus clientes.

1.3 Objetivo Geral

O objetivo geral deste trabalho é demonstrar e apresentar as funcionalidades e competências do Game Engine Impact para desenvolvimento de jogos eletrônicos na forma de um protótipo de um jogo baseado no clássico Metal Slug, o qual foi dado o nome de Zombie Invader; de maneira a apresentar um ponto de partida para interessados em desenvolver jogos com esta ferramenta.

1.4 Objetivos Específicos

Como objetivo secundário da presente reflexão vê se a urgência e necessidade de criar condições para fomentar o interesse dos possíveis clientes; e também

a) Compreender a utilização do Impact no desenvolvimento de Jogos Eletrônicos;

b) Esquematizar e conceituar o projeto de um protótipo de Jogo Eletrônico, com o foco em mensurar como o Impact atende as necessidades de um projeto do tipo ao qual ele é destinado;

c) Desenvolver o protótipo utilizando o Impact;

d) Oferecer possibilidade para que futuros alunos conheçam o funcionamento do jogo utilizando do game engine;

Neste momento surgem os seguintes questionamentos: como despertar e motivar o potencial cliente na utilização deste novo modelo de jogo, assim este trabalho constitui uma

oportunidade de aprofundamento teórico e metodológico como ferramenta indispensável para dar oportunidade a muitas pessoas que hoje procuram incansavelmente novos modelos de jogos, tornando-os mais agradáveis possíveis.

1.5 Metodologia

Iniciemos agora uma importante reflexão. Sabemos que a evolução dos jogos está historicamente vinculada aos avanços tecnológicos. Entretanto, embora os avanços tecnológicos sejam fundamentais, estes representam apenas uma alternativa capaz de atender as exigências do mercado. Há que se ressaltar então que o grande desafio em termos de realidade é o acesso pleno ao universo de informações disponíveis e a oportunidade de vivenciar e aprender nesta nova realidade.

Diante disso, iniciei a elaboração do meu trabalho seguindo algumas etapas que julgo essencial para a estruturação e qualificação do mesmo.

Inicialmente foi desenvolvido uma pesquisa procurando entender sobre as novas funções que o HTML 5 pode oferecer para o desenvolvimento web, além de uma breve pesquisa sobre o desenvolvimento de jogos digitais.

No segundo momento foi realizado um estudo sobre todas as características, além das funcionalidades que o Impact pode oferecer para o desenvolvimento de jogos digitais.

Em seguida foi elaborado um protótipo de um jogo no qual seria utilizado o Game Engine Impact.

Sem dúvida este trabalho pode e deve vir a contribuir para a elaboração de jogos, sabemos que para obter êxito será uma caminhada árdua que terei que percorrer.

Em um mundo movido pela tecnologia, considerando que o jogo faz parte da cultura dos jovens, adolescentes e porque não dizer de todos que tem intensidade, fascinação, liberdade e capacidade de excitar. É através do jogo que podemos adquirir iniciativa, autoconfiança e equilíbrio. É sabido que o jogo é uma atividade que ajuda no desenvolvimento mental.

1.5.1 Cronograma

O seguinte cronograma foi utilizado para a realização do presente trabalho.

Atividades Realizadas	AGO	SET	OUT	NOV	DEZ
Pesquisa					
Elaboração da revisão bibliográfica					
Implementação					
Testes					
Entrega do TCC					
Defesa do TCC					

Tabela 1: Cronograma do desenvolvimento do trabalho

1.6 Estrutura do Trabalho

Nesse cenário, o trabalho visou identificar a estrutura do trabalho como um todo, propondo alternativas, bem como contribuindo para desenvolver propostas cada vez mais eficientes, com novas possibilidades como descrevemos a seguir:

a) **O HTML5**: explana uma breve introdução ao HTML5, demonstrando algumas de suas novas características, além de alguns de seus novos elementos;

b) **IMPACT:** demonstra algumas funcionalidades e capacidades que o Impact pode realizar no desenvolvimento de jogos;

c) **GAME DESIGN:** relata sobre o primeiro passo para a criação de um jogo, que se baseia na criação de um documento onde ficarão todas as idéias e funcionalidades do jogo;

d) **CONCEPÇÃO DO JOGO:** relata o processo de desenvolvimento em cada um dos seus aspectos e funcionalidades, além de apresentar um passo a passo das etapas as quais foram seguidas para a sua prática;

Para atingir os objetivos do trabalho, foi feito o levantamento bibliográfico dos dados necessários para dar inicio ao trabalho, depois foi feita uma revisão bibliográfica para fundamentar a teoria, dando um embasamento técnico ao trabalho, justificando os tópicos abordados para o desenvolvimento do TCC, deu-se inicio a modelagem do jogo que foi desenvolvido para demonstrar e apresentar as funcionalidades e competências do game engine Impact para desenvolvimento de jogos eletrônicos na forma de um protótipo de um jogo

baseado no clássico Metal Slug, o qual foi dado o nome de Zombie Invader; de maneira a apresentar um ponto de partida para interessados em desenvolver jogos com esta ferramenta.

A última etapa contou com a prática do jogo com suas fases, testes e as conclusões a que se foi possível.

Todos nós sabemos que até certo tempo atrás grande parte dos jogos para navegadores eram criados no Flash um software do Adobe que também é muito usado para a criação de animações, tocador de vídeos entre outras aplicações.

O Adobe afirma que 98% dos navegadores possuem sua tecnologia instalada e que a mesma é aberta, segura e com bom desempenho(Lynch, 2010). Isto sempre foi aceito pela comunidade da web até que a Apple disse que não incorporaria o Adobe Flash aos seus novos produtos iPhone e iPad por considerá-lo com muitos erros, sem segurança e consumidor voraz de bateria(Apple, 2010)

2. O HTML 5

Neste capitulo será apresentado uma breve síntese sobre o HTML, dando ênfase ao HTML 5 como meio de oferecer oportunidade ao leitor no desenvolvimento de suas atividades.

O HTML (Hyper Text Markup Language) é a primeira camada do desenvolvimento client-side, responsável por organizar e formatar páginas desde que foi introduzido à internet no inicio de 1990. A versão 4.0.1 da linguagem, aprovada no ano de 1999 não dispõe de meios necessários para incorporar com sucesso recursos que permitem mais expressividade e controle multimídia, sendo que muitos sites dependem de plug-ins para fornecer essa funcionalidade.

Atualmente, a inserção de recursos de enriquecimento de informações é feita por meio de plug-ins nos navegadores, as alternativas utilizadas são tecnologias como o Microsoft Silverlight, Adobe Flash, Sun JavaFX e Adobe Flex que fornecem uma solução cross-browser compatível para incorporação de aplicativos de internet rica (RIA).

Porém, não depender de plug-ins externos para reproduzir elementos de mídia é interessante para navegabilidade dos usuários e simplificação de codificação para programadores.

Nesse quesito surge o HTML 5, que introduz um conjunto de novos elementos que facilitam a estruturação e desenvolvimento front-end, capaz de fornecer uma informação qualitativa sobre os diferentes elementos da página, uma vez que a falta de mecanismos criou a necessidade do uso de outras estruturas para criação destes elementos.

Surgindo como a próxima geração do HTML e substituindo a versão 4.01, XHTML 1.0 e XHTML 1.1, o HTML 5 fornece novos recursos que são necessários para aplicações web modernas e padronização de muitas características da plataforma web que já são usadas por desenvolvedores há muitos anos, mas que nunca foram aprovados ou documentados por um comitê de padrões. (Berjon, Leithead, Navara, O'Connor, & Pfeiffer)

Uma das primeiras tarefas do HTML 5 foi documentar o não documentado, a fim de aumentar a interoperabilidade, deixando poucas adivinhações para autores da Web e implementadores de browsers.(Lawson & Sharp, 2011)

A nova versão da linguagem tem como principal objetivo dar aos desenvolvedores a facilidade na manipulação dos elementos de marcação, possibilitando a modificação das características dos objetos de forma não intrusiva e transparente para o usuário final.

Diferente das versões anteriores, o HTML 5 fornece ferramentas para o CSS e o Javascript fazerem seu trabalho da melhor maneira possível, com isso, a linguagem permite a manipulação das características dos elementos através de suas APIs, também conhecidas como DOM, de forma que as páginas ou aplicações continuem leves e funcionais.

É verdade que há muito tempo que existem implementações de jogos web utilizando javascript, ou até mesmo applets java, mas somente agora com o HTML 5 sendo cada vez mais bem suportado pelos navegadores, podemos contar com bons recursos de processamento gráfico e outras funções a mais.

Os dados estatísticos demonstram a representatividade e a importância do HTML 5 na elaboração de jogos, o que representa sem sombra de duvida mais espaço no panorama tecnológico e um desafio constante em empregá-lo de forma a trazer benefícios e uma alternativa concreta na utilização deste.

Com a geração atual de navegadores, e com a atualização do padrão HTML, já é possível criar jogos webgames sem depender de nenhum plug-in. Isto permite que os jogos possam ser executados em todos navegadores atuais, inclusive em smartphones e tablets. O Impact vem para facilitar a criação de jogos oferecendo seu framework que por uma de suas várias definições é um conjunto de scripts feitos por pessoas mais inteligentes que eu para eu programar de forma mais rápida, fácil e eficiente.

Portanto, seguindo o mesmo pensamento é fundamental que se considere as diferentes versões na aplicabilidade do jogo.

2.1 Novas características da linguagem

Numa perspectiva atual, as novas características de linguagem passam a ser um processo desafiador, articulado e motivador que auxilia na construção no processo.

Desde sua criação o HTML5 oferece novas *tags* de marcações e aprimoramento em algumas funções já existentes e um novo padrão universal para a criação de seções comuns e específicas nos documentos, como rodapé, menu, e etc. Além de um padrão de

nomenclatura de IDs, classes ou tags e métodos para auxiliar na captura automática das informações localizadas nos rodapés dos websites. As figuras abaixo representam respectivamente a estrutura desenvolvida com o HTML4 e a estrutura simplificada que o HTML5 oferece.

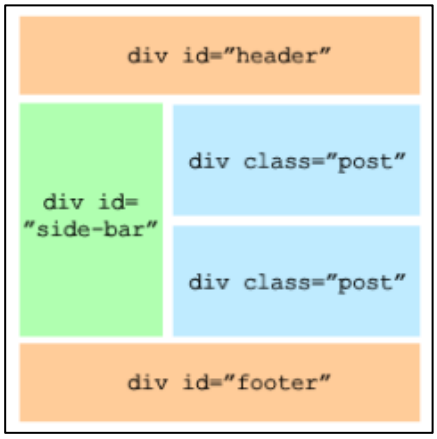

Figura 1: Estrutura HTML 4
Fonte: Smashingmagazine, 2012

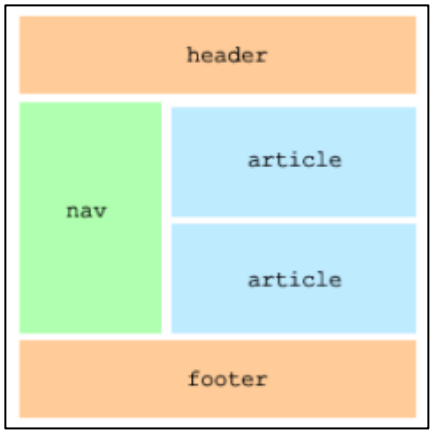

Figura 2: Estrutura HTML5
Fonte: Smashingmagazine, 2012

Esta nova versão da linguagem modifica a forma como os desenvolvedores deverão escrever seus códigos, visando à melhor organização das informações na página, fornecer mais semântica com menor quantidade de códigos, além de mais dinâmica através das atualizações frequentes, isso devido a forma em que a comunidade de desenvolvedores da linguagem estão se dividindo, fazendo com que módulos pequenos da linguagem sejam escritos por pequenos grupos independente e cada grupo pode lançar atualizações a qualquer momento. (Berjon, Leithead, Navara, O'Connor, & Pfeiffer)

Sempre que pensamos em evolução, mudança, transformação, será necessário analisar os elementos estruturais, os desafios e a importância para atender o objetivo final.

2.2 Elementos Estruturais

Os elementos estruturais mostram o quanto é necessário para contribuir no desenvolvimento dos jogos, na busca de um novo modelo de aprendizagem.

Nas versões anteriores do HTML, era possível marcar diversos elementos do layout, diferenciando um parágrafo de um título ou de subtítulo, mas não era possível

diferenciar qual parte da página compunha o rodapé ou o cabeçalho. Esta diferenciação era apenas percebida visualmente pelo layout pronto ou pela posição dos elementos na estrutura.

Em 2009 a Opera Software realizou uma pesquisa, vasculhando cerca de dois bilhões de páginas, analisando não somente as classes utilizadas, mas também os IDs mais utilizados. Os resultados dessa pesquisa são apresentados na tabela abaixo:

Popularidade	Valor	Frequência
1	Footer	288,061
2	Content	228,661
3	Header	223,726
4	Logo	121,352
5	Container	119,877
6	Main	106,327
7	table1	101,677
8	Menu	96,161
9	layer1	93,920
10	autonumber1	77,350
11	Search	74,887
12	Nav	72,057
13	Wrapper	66,730
14	Top	66,615
15	table2	57,934
16	layer2	56,823
17	Slideber	52,416
18	image1	48,922
19	Banner	44,592
20	Navigation	43,664

Popularidade	Valor	Frequência
1	Footer	179,528
2	Menu	146,673
3	style1	138,308
4	Msonormal	123,374
5	Text	122,911
6	Content	113,951
7	Title	91,957
8	style2	89,851
9	Header	89,274
10	Copyright	86,979
11	Button	81,503
12	Main	69,620
13	style3	69,349
14	Small	68,995
15	Nav	68,634
16	Clear	68,571
17	Search	59,802
18	style4	56,032
19	Logo	48,831
20	Body	48,052

Tabela 2: 20 IDs e classes mais utilizada

Como resultado desta pesquisa, a nova versão da linguagem HTML traz aos desenvolvedores uma série de elementos que auxiliam na definição dos primeiros setores do documento. Com estes elementos, é possível diferenciar áreas importantes do site, como cabeçalho, menu, rodapé, entre outros que são muito utilizados. Estas mudanças simplificam o trabalho de sistemas como dos buscadores que conseguem vasculhar o código de maneira mais eficaz e estocar rapidamente informações exatas sobre o documento.

2.3 Elementos Multimídia

Nas versões anteriores do HTML, não havia nenhuma maneira baseada em padrões para incluir vídeos ou áudios nas páginas web, a forma de se fazer isso era através de

uma canalização de um plug-in de terceiros – QuickTime, RealPlayer ou Flash. Mas estes plug-ins causavam muitos transtornos aos usuários que tentavam assistir a um vídeo em uma plataforma o qual não tinha suporte.

Com uma das soluções trazidas pela versão cinco da linguagem, o HTML introduziu uma forma padrão para incorporar vídeos em uma página web, oferecendo a habilidade de embutir mídia em documentos HTML. Os códigos abaixo apresentam exemplos de utilização das novas tags para inserção de áudio e vídeo na página, atribuindo valores para os atributos principais.

```
<audiosrc=”mus.oga” controls=”true” autoplay=”true” />
<videosrc=”vid.ogv” controls=”true” autoplay=”true” />
```

Código 1: Exemplos multimídia

2.4 Elemento Canvas

Representa uma tela bitmap, que pode ser usada para desenhar gráficos, fazer composições de fotos, desenvolvimento de animações, jogos ou renderização de imagens em tempo real, através de auxílio de scripting, usualmente JavaScript.(Berjon, Leithead, Navara, O'Connor, & Pfeiffer). A inovação neste elemento reside no fato de não ser necessário o uso de plug-ins para utilização de recursos 2D e futuramente 3D, embora ainda nem todos os navegadores sejam capazes de usufruir deste recurso. Para posicionar elementos em <*canvas*> considera-se o eixo de coordenadas em duas dimensões, que começa no canto superior esquerdo da tela. A tela produzida terá dimensões indicadas com os atributos *width* e *height*, e o tamanho do espaço de coordenadas não necessariamente representa o tamanho do bitmap real que a agente irá utilizar internamente ou durante o processamento, como demonstrada na imagem abaixo. Em monitores de alta definição, por exemplo, o agente pode usar internamente um bitmap com dois pixels por unidade de dispositivo no espaço de coordenadas, de modo que a representação se mantém em alta qualidade.

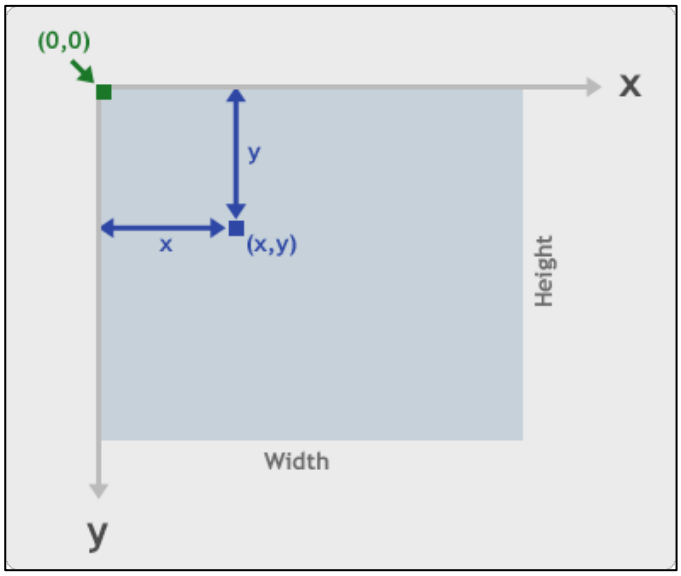

Figura 3: Exemplo coordenadas canvas
Fonte: Criar Web

Usufruindo da marcação<*canvas*> é possível desenhar através de codificação. (Sepherd, 2010) O seguinte código exemplifica a utilização de como desenhar uma imagem no elemento canvas, através do JavaScript.

```
var canvasElem =document.getElementById('areaDesenho');
var contexto = canvasElem.getContext('2d');
contexto.drawImage(imagem, x, y, largura, altura);
functiondrawImage() {
varcanvasElem=document.getElementById('areaDesenho');
if (canvasElem.getContext) {
//Obtém o contexto "2d" de desenho para o canvas
var context = canvasElem.getContext('2d');

//Desenha a imagem no canvas
varimg=newImage();
img.onload=function(){
context.drawImage(img,10, 50);
      }
img.src='canvas_image.png';
   }
}
```

Código 2: Exemplo canvas

Para desenhar imagens no contexto de desenho do canvas, usamos a função drawImage, que recebe como parâmetros, uma imagem, a posição x e y, a largura e a altura da imagem. A largura e altura são opcionais, e se não forem especificadas serão usadas a altura e largura padrão da imagem.

Existem inúmeras possibilidades de criação com o elemento <*canvas*>, incluindo animações, fator que possibilita a personalização de itens com movimento simples, trazendo a independência de formatos animados como o .GIF e potencialidade de interação com usuário.

2.5 HTML5 X Outras Tecnologia

Conforme visto acima, podem-se visualizar os elementos que foram e estão sendo executados neste projeto, entretanto passamos a descrever o HTML 5 X outras tecnologias para acompanhamento e sucesso do mesmo.

Cada projeto ou cliente apresenta necessidades diferentes, e cabe aos desenvolvedores analisarem as situações com intuito de definir a melhor ferramenta de desenvolvimento. O HTML 5 apresenta recursos já utilizáveis que podem se encaixar no perfil de determinados projetos.

Quando se comparam tecnologias tão abrangentes, é indispensável delimitar sobre qual funcionalidade ocorre à análise, pois não existem ferramentas e tecnologias cabíveis a todas as situações.

É justificável considerar que diversas aplicações podem vir a ser dispensáveis com o advento do HTML5. Algumas tecnologias e características frequentemente solicitadas no desenvolvimento web são apresentadas na tabela abaixo, em caráter comparativo entre HTML5, Microsoft Silverlight e Adobe Flex.

Tecnologia	**HTML5+JS**	**Adobe Flex 4**	**Microsoft Silverlight 4**
Runtime	nativo em navegadores compatíveis	Flash Player 10.1	Silverlight 4.0.5
Aplicações Multi-Plataforma	todas onde houver compatibilidade com o navegador em questão	Windows, MacOS, Linux, Solaris, Google Android	Windows, MacOS, Windows Phone 7, XBox 360, Microsoft Surface
Animação – Timeline	Canvas / Frame-by- Frame	Frame-by-Frame	Baseada em tempo, Frame-by-Frame
3D	Baseado em simulação de perspectiva X, Y e Z	Baseado em simulação de perspectiva X, Y e Z	Baseado em simulação de perspectiva X, Y e Z
Aplicações baseadas em Desktop 2.0	Drag-and-drop, API de aplicações offline	Adobe Air	Silverlight OOB
Modelo de programação	Linguagem de marcação, orientado a objetos via JavaScript	Orientado a Objetos	Orientado a Objetos
Linguagem de	Baseado em XML /	Baseado em XML, o	Baseado em XML, o XAML

interfaces	Javascript	MXML	
Linguagem base – Client	HTML e integráveis via API	ActionScript 3	C#, VB.NET, IronRuby, IronPython
Linguagens utilizadas com frequência – Server/Side	Java, PHP, C#	Java, PHP, ColdFusion, C#	C#, VB.NET, PHP
Arquivo executável binário	-	Arquivo SWF	Arquivo XAP
Característica da plataforma	Vinculação de dados, tratamento de eventos, Layout, Navegação, Controles, DataGrid, visualização de dados, estilização, suporte a impressão, Web workers, offline cache, webstorage	Vinculação de dados, tratamento de eventos, Layout, Navegação, Controles, DataGrid, visualização de dados, estilização, câmera e microfone, suporte Offline DRM, Peer- assisted P2P, Suporte a impressão, HTML hosting, Aceleração de Hardware	Vinculação de dados, tratamento de eventos, Layout, Navegação, Controles, DataGrid, visualização de dados, tematização, câmera e microfone, suporte offline DRM, multicast streaming, suporte a impressão, HTML hosting, Aceleração de Hardware, COM Interop, Multi- Threading
Ferramentas de desenvolvimento	Dreamweaver CS5 e editores HTML	Flash Builder 4	Microsoft Visual Studio 2010
Ferramentas de design	Dreamweaver CS5 e editores HTML	Adobe Flash CS5, Adobe Catalyst CS5	Microsoft Expression Blend 4, Microsoft Expression Design 4
Integração com outras ferramentas de design	Adobe Photoshop CS5, Adobe Illustrator CS5	Adobe Photoshop CS5, Adobe Illustrator CS5, Adobe Fireworks CS5	Adobe Photoshop CS5, Adobe Illustrator CS5
Linguagens baseadas em estilos	CSS3, CSS2.1	CSS2.1	XAML
Suporte a Imagens	Todos os formatos, incluindo SVG	Todos os formatos	PNG, JPG
Acesso remoto	HTTP, Socket, Web Services	HTTP, Socket,Web Services, Remoting	HTTP, Socket,Web Services, WCF
Manipulação de vídeo	tipos de arquivo (webm, ogg, mp4, wmv) Ferramenta: Javascript API	Tipos de arquivos (flv, f4v), Ferramenta: Adobe Media Encoder Suporte a DRM	Tipos de arquivos (wmv, f4v), Ferramenta: Expression Media Encoder, Suporte a DRM
Distribuição de vídeo	Nativa HTML5	Adobe Flash Media Server	Microsoft IIS Media Services

Tabela 3: Comparativo HTML5 x Flex x Silverlight
Fonte: Adaptado de (Fernandes, 2010)

3. IMPACT

3.1 Introdução ao Impact

Dando continuidade ao trabalho passo a discorrer sobre o Impact que é um JavaScript game engine criado por Dominic Szablewski. O Impact usa o elemento Canvas (capítulo 2.4) dos navegadores, a fim de criar jogos digitais de alto desempenho em duas dimensões para a Web e também para dispositivos móveis.

O JavaScript é uma linguagem de programação interpretada e orientada a objetos baseada em protótipos(ou seja nela não existem classes) e em first-class functions. (Xavier, 2007).

Como framework de desenvolvimento de jogos, o Impact é focado em ter uma estrutura simples de criação e uma série de ferramentas robustas de desenvolvimento anteriormente exclusivas para outras plataformas de desenvolvimento.

Através da analise deste conceito, podemos nos perguntar por que usar o Impact, e posteriormente tomamos a decisão como podemos perceber no decorrer do texto.

3.2 Por que usar o Impact?

"I've tried four other JavaScript game engines, and this is the first one I've used that makes sense (…) Impact is the first truly professional-grade JavaScript and HTML5 game engine to hit the market."(Magazine, 2011)

O Impact é muito útil, pois transforma o desenvolvimento de jogos em HTML5 em algo muito simples com o seu framework. Ele é muito semelhante ao XNA, que é um framework para desenvolvimento de jogos em sistemas da Microsoft como o Windows, Zune e o Xbox. O Impact conta com uma gama completa módulos/classes muito útil para facilitar a criação de jogos. Outro grande atrativo para o uso do Impact sem dúvidas é seu editor de level chamado Weltmeister que apesar de não conter tudo que um designer de level necessita, ele pode criar um bom jogo. A seguir passamos a discorrer o ciclo de vida de um jogo no impact.

3.3 Ciclo de vida de um jogo no Impact

Quando se inicializa um jogo o Impact cria um intervalo que chama o método *ig.system.run()* 60 vezes por segundo. Este método realiza algumas tarefas e chama o método do jogo *.run()* que por padrão chama somente os métodos *.update()* e *.draw()* em si.

Os métodos .update() e .draw() tem funções muito semelhantes. Na execução das rotinas, o .draw() é chamado primeiro. Ele se encarrega de limpar a tela e plotar os gráficos do jogo necessários para aquele momento. Logo em seguida, o .update() é chamado e se responsabiliza por atualizar o estado dos objetos presentes no jogo de acordo com suas propriedades físicas, levando em conta o mapa de colisões.

Com o passar do tempo surge um novo projeto no Impact, com o intuito de demonstrar as etapas de cada jogo.

3.4 Novo projeto no Impact

Pode-se citar como exemplo, a seguir a estrutura novo projeto no Impact:

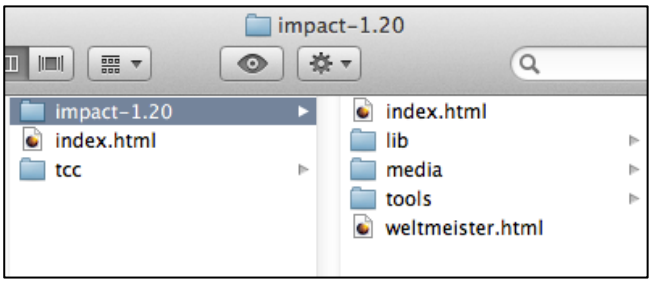

Figura 4: Estrutura novo projeto no Impact
Fonte: Elaborada pelo autor

A cada jogo que deseja ser criado será necessário copiar a pasta padrão do Impact em uma nova pasta em seu servidor e começar do zero (Figura1). Abaixo segue um pequeno resumo do que contém na pasta padrão do Impact:

Index.html: Este é o arquivo .html principal que irá executar o jogo.

lib: Esta é a parte central do Impact, onde está todo o seu código padrão e onde é armazenado todos os códigos JS específicos do jogo. Também é onde se encontra o código do editor de level Weltmeister.

media: Esta é a parte aonde todos os arquivos de imagens e sons serão carregados pelo Impact.

tools: Este diretório contém scripts em PHP para simplificar os arquivos do jogo e torná-los mais difícil para as pessoas terem acesso ao código fonte do jogo.

weltmeister.html: Este é o arquivo .html que carrega o editor de level.

A seguir passo a discorrer sobre o código de fonte do impact no qual é organizado em módulos como podemos perceber a seguir. A qualidade do jogo pode ser medida através da organização e esta pode estar ligada aos módulos como veremos a seguir.

3.5 Módulos

O código fonte do Impact é organizado em módulos. Como o Javascript próprio não possui uma função include() que pode carregar outros códigos JavaScript em um objeto, o Impact tem seu próprio sistema. Como mostrado abaixo:

```
ig.module(
    'game.my-file'
)
.requires(
    'impact.game'
    'impact.image'
    'game.other-file'
)
.defines(function(){
//code for this module
});
```

Código 3: Exemplo módulo no Impact

O primeiro bloco define o módulo com nome 'game.my-file', que diretamente corresponde ao nome do arquivo. Os módulos e suas dependências tipicamente residem na pasta lib/ do seu projeto do Impact, os subdiretórios são incluído no caminho usando a sintaxe de ponto. Sendo assim, o my-file.js está em lib/game/my-file.js

O segundo bloco define qualquer arquivo adicional que será carregado no tempo de execução. Como o javascript não possui um método estabelecido para carregar outros códigos em um objeto, o Impact possui seu próprio sistema. Os módulos listados no método .requires() serão carregados dentro de lib/impact/game.js, lib/impact/image.js, e lib/game/other-file.js respectivamente. Estes arquivos necessários serão carregados antes do módulo body e antes do último bloco abaixo do exemplo de módulo executado.

O último passo que o módulo passa é executar a função passadas para o método .defines(). Este processo linear ajuda a controlar quando o código é carregado e executado. É importante seguir a nomenclatura e estrutura de localização do Impact, pois ele tentará automaticamente carregar estes recursos na fase de pré-carregamento.

Podemos dizer que no impact aplicam as classes. O fato é que as classes estão inseridas dentro da aplicação conforme veremos a seguir.

3.6 Classes

No JavaScript não há nenhuma noção real de uma tradicional estrutura de classes como é visto em outras linguagens orientadas a objetos. No JavaScript, tudo é um objeto. O que permite o JavaScript ser incrivelmente flexível, entretanto isto dificulta estruturar seu código de uma forma reusável. Para resolver este problema, o Impact tem uma pseudo-classe objeto, que é a base parar criar todas as classes em um jogo.

Nas tradicionais linguagens orientadas a classe, a palavra-chave extends permite copiar toda a funcionalidade de outra classe existente. O que nos permite infundir funções adicionais para toda a nossa classe do jogo sem precisar duplicar códigos em outros locais.

A seguir segue um exemplo de como criar uma classe personagem fora da parte central do Impact:

```
//Criar uma nova classe “Personagem”
var Person =ig.Classe.extend({
name= ‘’,
init: function( name ) {
this.name= name;
}
});

// Instanciar um objeto para a primeira classe
var e =newPerson('João Mota');
e.name; // => João Mota
```

Código 4: Exemplo classe

No Impact também é possível extender outras classes customizadas. Novamente para extender outra classe é só usar novamente a funcionalidade .extend(). Abaixo segue um exemplo de como extender a classe personagem para criar uma nova classe zombie:

```
//Criar uma nova classe estendendo a classe personagem
var Zombie =Person.extend({
init: function( name ) {
this.parent('Zombie: '+ name );
}
});

// Instanciar um objeto para a segunda classe
var p =newZombie('Márcio');
p.name; // => Zombie: Márcio
```

Código 5: Exemplo classe extendida

Todas as classes criadas com .extend() também podem possuir um método .extend() que pode ser usado em outras subclasses. Quando se é trabalhado dentro de classes extendidas, pode-se usar .this e .parent para escopo.

Dessa forma, passamos a identificar as classes principais com suas funcionalidades facilitando a criação de novos jogos.

3.7 Classes principais

O Impact é composto de várias classes principais que visam diversas funcionalidades que facilitam a criação de jogos, abaixo segue uma lista delas com um breve resumo de sua determinada função:

ig Core: O objeto *ig* fornece a definição do módulo e sua capacidade de carregamento, bem como algumas funções de utilidade.

Animation: O objeto ig.Animation cuida de animar uma entidade ou um tile de um Background Map.

AnimationSheet: Ele é um fino wrapper que pertence ao objeto *ig.Image.* Ele especifica a largura e altura para cada quadro de animação. É usado pela classe *ig.Animation.*

BackgroundMap: Um *ig.BackgroundMap*desenha os tiles de um Tileset, como indicado pelo array de dados 2D.

CollisionMap: Um *ig.Collision*usa um TileMap 2D permitindo a detecção de colisões.

Entity: Os objetos interativos no mundo dos jogos são normalmente uma subclasse desta classe de entidade. Ele fornece o desenho, animação e a física básica. O *ig.Entity* garante que suas subclasses possam ser adicionadas ao mundo do jogo, reagindo ao mapa de colisões e a outras entidades, e sendo usado pelo Weltmeister.

Font: Um objeto *ig.Font* carrega uma imagem de fonte especialmente formatada e permite o desenho de texto com ela.

Game: O *ig.Game* é o principal eixo para o jogo. Ele abriga todas as entidades ativas, os BackgroundMap e um CollisionMap.

Image: O *ig.Image* envolve tudo sobre recursos de imagens (.png, .gif, .jpeg). Cuida do carregamento de imagens e dimensionamento. Pode-se desenhar toda uma imagem usando o *.draw()* ou apenas um tile dela usando *.drawTile()*.

Input: O *ig.Input* cuida da entrada de dados do teclado e mouse.

Loader: O *ig.Loader* um pré-carregador padrão de todas as imagens e sons que o jogo necessita.

Map: O *ig.Map* é a classe base para o *ig.BackgroundMap* e *ig.CollisionMap*. Ele apenas fornece acesso básico para os tiles em um mapa de dados.

Music: O *ig.Music* oferece a habilidade de tocar uma lista de músicas em plano de fundo em ordem ou randomicamente.

Sound: Uma instância do *ig.Sound* representa um arquivo de som que será usado como musica de fundo ou um som do jogo.

SoundManager: O SoundManager cuida do carregamento dos sons e proporcioná-los para as instâncias *ig.Music* e *ig.Sound*. Uma instância do SoundManager é automaticamente criado usando o *ig.soundManager* pela função *ig.main()*.

System: O *ig.System* cuida da inicialização e paralização do ciclo de execução e chama o método *.run()* em um determinado objeto do jogo. Também cuida da inicialização para o *ig.Input* e proporciona alguns métodos utilitários.

Timer: O *ig.Timer* tem dois métodos distindos de operação. Pode ser usado para pegar a diferença usando *.delta()* entre o tempo atual e o tempo destino (setado pelo construtor ou pelo *.set()*) ou apenas pegar o tempo atual desde a última chama pelo *.tick()*.

Isso significa que se deve garantir na criação de jogos as classes principais, de forma a garantir as várias funcionalidades. Dessa forma, é ampliando a concepção de weltmeister uma das melhores ferramentas presentes no Impact.

3.8 Weltmeister

Uma das melhores ferramentas presentes no Impact é o seu editor de levels (mapas) que é chamado de Weltmeister, ele está presente dentro da pasta libs/weltmeister do seu projeto do impact. Ele pode ser aberto entrando na raíz do seu projeto, abrindo o arquivo weltmeister.html.

Por padrão quando se abre o weltmeister mesmo que não exista um mapa, ele já vem com uma camada (layer) que é a camada de entidades. Todas as entidades como jogador, inimigo e outros são adicionados usando esta camada. As camadas criadas no weltmeister podem ser escondidas simplesmente clicando em um quadrado que está perto do devido nome da camada, facilitando em partes o desenho de um jogo, mesmo com uma camada escondida ela aparecerá normalmente no jogo.

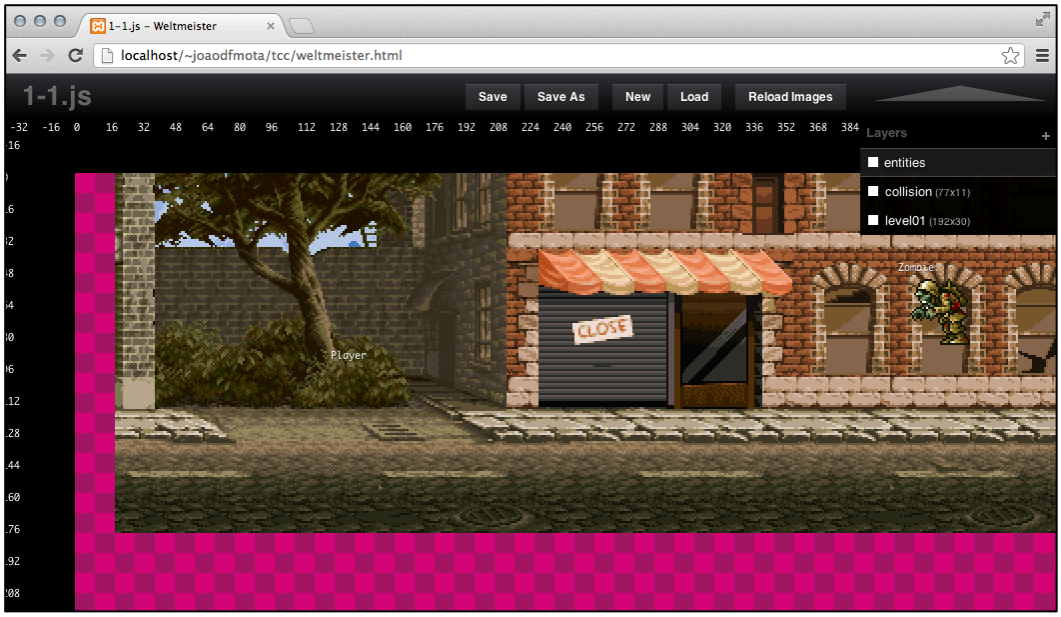

Figura 5: Weltmeister
Fonte: Elaborada pelo autor

No weltmeister pode facilmente criar uma camada de colisões, ou seja marcar o local aonde algum personagem não pode passar, outra função muito interessante no weltmeister é o fato de poder manipular as entidades, como por exemplo criar uma trigger como entidade e ligá-la a outra para que ela faça o efeito somente na entidade ligada com ela. Dessa forma passamos a discorrer exemplos de jogos como ferramenta para o desenvolvimento no Impact.

3.9 Exemplos de jogos

Neste tópico são apresentados alguns jogos desenvolvidos pela ferramenta como forma de estabelecer um referencial para o potencial de desenvolvimento para o Impact.

3.9.1 Z-Type

Vencedor do Communty Choice Award na competição Mozilla Labs Game On 2010. O Z-Type (Figura 3) é um jogo baseado no R-Type sendo um jogo de digitação, uma maneira muito interessante para se aprender a digitar.

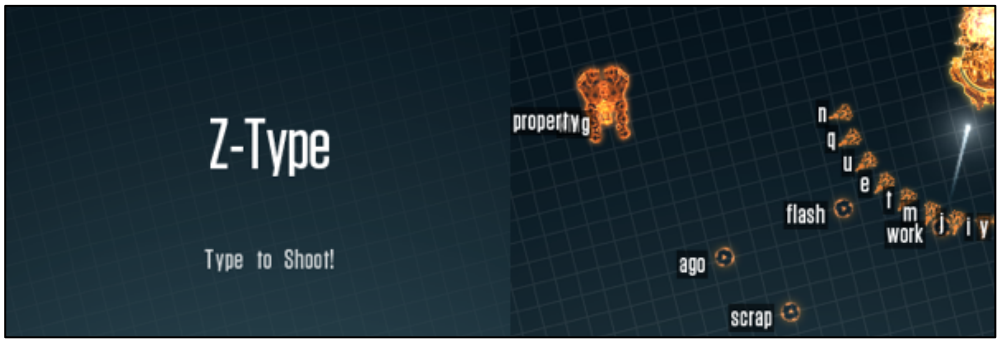

Figura 6: Z-Type
Fonte: Impact, 2012

3.9.2 Creatures And Castles

Um jogo (Figura 4) de ação e estratégia terrivelmente viciante no qual você ajuda um jovem caçador de tesouros a roubar grandes quantidades do tesouro de um castelo medieval.

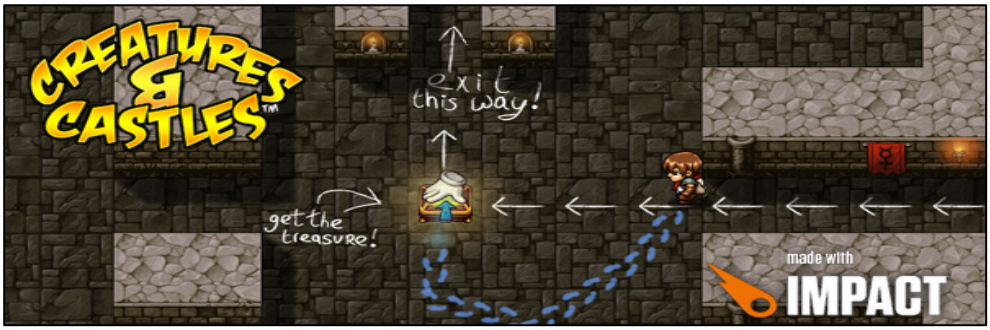

Figura 7: Creatures & Castles
Fonte: Impact, 2012

4. GAME DESIGN

4.1 Introdução

Game Design é a criação e planejamento dos elementos, regras e dinâmicas de um jogo. Dentro de uma equipe de desenvolvimento de um jogo, o Game Designer tem esta função importantíssima. Ele será o cérebro de todo o projeto. Criará a idéia do jogo, suas interações, seu enredo, suas regras e todos os elementos que deverão existir dentro deste jogo.

> É importante entender que o esforço para a criação de um bom jogo começa muito antes da fase de codificação. Embora muitos programadores de jogos não-profissionais subestimem a etapa de planejamento, ela com certeza é a fase mais importante de um projeto de desenvolvimento de jogos, uma vez que nesta etapa são definidos os principais pontos a serem seguidos nas fases seguintes.(Farias, 2010)

4.2 O que é o design de jogos?

Quando se fala sobre design de jogo, não se deve referir ao estilo visual do jogo, mas sim a mecânica do jogo propriamente dita. O design de jogos em sua concepção é uma forma de arte, e provavelmente uma das partes mais difíceis na criação de qualquer jogo. Isto é nele não se trata basicamente em transformar uma idéia para algo físico para outros jogarem, mas também garantir que o jogo fique divertido. O Primeiro passo para projetar um jogo normalmente começa com um documento de concepção de jogo ou game design document (GDD).

Os GDDs podem ser escritos de diversas formas e tamanhos, podem ser rabiscados em um caderno de anotações, mas geralmente ele vem em forma de documento de texto de várias páginas que contém o conceito geral de um jogo e sua jogabilidade a ponto de responder a algumas perguntas básicas de como o jogo funciona realmente. No mínimo ele deve dar ao leitor uma idéia clara de como o jogo funciona.

O GDD é fundamental para as pessoas que estão iniciando na criação de jogos, porque, sem determinada experiência será muito fácil criar algo complexo e demorado sem ao

menos conseguir finalizá-lo. O GDD ajuda a manter um foco nas idéias principais, não significando que ele será somente o que está escrito, mas com ele pode-se pensar tanto na interação do jogo antes de iniciar com a codificação será muito mais fácil ter um caminho curto na criação do jogo.

De acordo com o livro Rules of Play: Game Design Fundamentals (Salen & Zimmerman, 2004) devem-se seguir alguns pontos-chave que serão úteis na criação de um GDD:

Comece o documento com um high-concept: Nele se deve criar uma descrição inicial do jogo. O high-concept é propriamente um termo da indústria cinematográfica, que geralmente vem com um cenário "e se". Como por exemplo: "E se o Mário fosse um verme com uma arma?".

Tente adicionar ilustrações, esboços e arte conceitual ao seu documento: Isto ajuda a analisar o que o jogo irá conter. Por natura pessoas irão se cansar de ler um documento de concepção de jogo sem indicação do estilo visual do jogo.

Ter uma visão clara da mecânica do jogo, como as coisas funcionam, e como eles irão interagir uns com os outros: Use o máximo de detalhes possíveis em torno das ações, como os combates funcionam, estatísticas, recompensas, etc.

Ilustre claramente como os objetos do jogo são construídos: Devem-se incluir coisas que um personagem pode ter no jogo como a vida, as armas, e muito mais. Isso será muito útil quando se usa isto como ponto de referência para o jogo.

Percebe-se que cada vez mais, torna-se essencial o conhecimento do design do game para que se possam criar os jogos. No próximo capítulo apresentamos game design e os princípios que norteiam o jogo.

5. CONCEPÇÃO DO JOGO

5.1. GAME DESIGN DOCUMENT

Partindo do principio que o Zombie Invader é uma variação de uma série de jogos chamada Metal Slug, que é um jogo no estilo gun-and-run que foi inicialmente desenvolvido para arcades pela empresa SNK.

O Zombie Invader tem por objetivo ser um jogo com um bom senso de humor e uma boa jogabilidade, visando ser perfeito para ser jogado quando você tem um tempo de inativade. Nele o personagem principal necessita ajudar o exército em uma grande luta contra uma invasão zumbi ao planeta.

Figura 8: Tela jogo GDD
Fonte: Elaborado pelo autor

5.1.1 Jogabilidade

O jogador tem que constantemente a atirar em um fluxo contínuo de inimigos, a fim de chegar ao final de um nível. No final de cada nível, o jogador pode enfrentar um chefe que muitas vezes é consideravelmente maior do que os inimigos normais e muito mais resistentes.

5.1.2 Combate

Como o jogo pertence ao estilo gun-and-run o combate proposto no jogo é basicamente a longa distância contra os inimigos

5.1.3 Completando um nível

Para se completar um nível e seguir para o próximo o jogador deve matar a maior quantidade possível de inimigos e ao final de cada nível enfrentar um inimigo chefe, logo após aparece uma tela mostrando o final do nível e a pontuação obtida no nível.

5.1.4 Morte

Quando o jogador morre, ele é transportado novamente para o início do nível, cada morte do jogador é computada e ao final do jogo uma tela é mostrada indicando as estatísticas do jogador.

5.1.5 Personagem Principal

Inicialmente apenas é oferecida a escolha de um personagem no qual o jogador irá usá-lo especialmente em todo o jogo.

Figura 9: Personagem principal
Fonte: Elaborado pelo autor

5.1.6 Inimigos

Inicialmente o jogo contém apenas um tipo de inimigo. Uma vez que um inimigo é derrotado, ele é removido do mapa e o jogador irá ganhar um ponto pela morte do inimigo.

Figura 10: Inimigos
Fonte: Elaborado pelo autor

5.1.7 Chefes

Neste jogo o jogador ao final de cada nível irá encontrar um tipo especial de inimigo que é chamado de chefe, sendo necessário matá-lo para que possa passar para o próximo nível. Os chefes são sempre mais poderosos do que os inimigos normais, então é necessário estar pronto para lutar até a morte.

5.1.8 Armas

Inicialmente o jogo irá contar com dois tipos de armas, uma pistola que conterá munição infinita e dez granadas que o jogador poderá utilizar em todo o jogo.

5.1.9 Terminando o jogo

Para se terminar o jogo é necessário que o jogador termine todos os níveis do jogo sem atingir um número específico de mortes, ao final do jogo aparecem às estatísticas do jogador.

Um dos grandes adjetivos desse jogo diz respeito exatamente ao prazer que este proporciona vindo daí a capacidade que têm de fazer com que as pessoas “saiam” da realidade de suas vidas e entrem num novo ambiente que é regrado e limitado, sendo que ali estão por vontade própria e porque gostam e buscam seu máximo, mesmo sabendo que o resultado de tal busca é inesperado e incerto.

5.2 Histórico da Implementação

Os primeiros passos consistiram em estudar as formas como se desenvolvem jogos com o Impact. A partir deste estudo, foi primeiramente desenvolvido um mapa (nível) do jogo usando o Weltmeister, no mesmo esboço do mapa foi também criado o mapa de colisões que determinam aonde o jogador pode ou não atravessar.

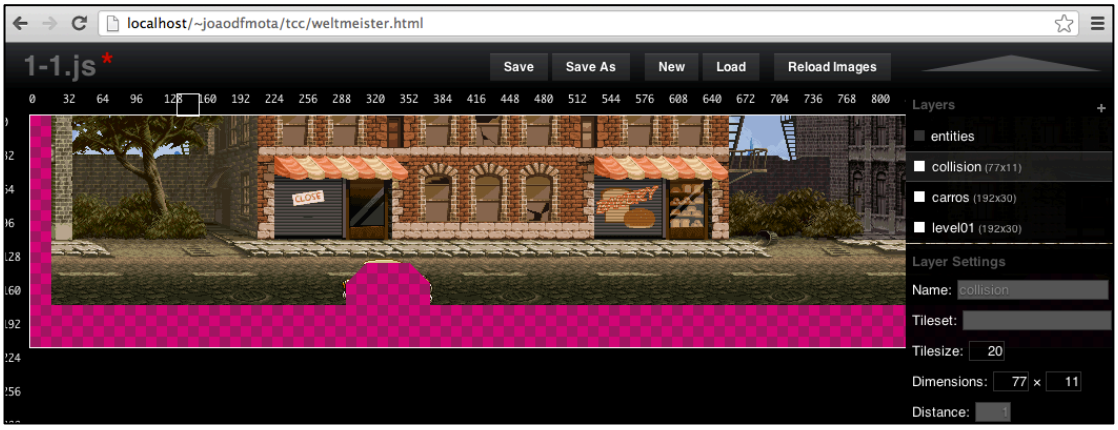

Figura 11: Implementação no Weltmeister
Fonte: Elaborado pelo autor

5.3 Primeira Fase da Implementação

Esta fase consistia em criar um protótipo inicial, com poucos recursos, somente a estrutura básica do jogo, onde foi criada toda a estrutura de como o personagem interagiria com o mapa e com seus inimigos.

5.3.1 Classe Principal

Logo após a criação do mapa foi necessário realizar algumas alterações na classe principal do Impact (*main.js)*, para que a mesma fosse possível de carregar o mapa criado anteriormente, simplesmente alterando dois blocos de código.

O primeiro bloco a ser alterado foi o *.requires(...)* que corresponde ao módulo que define arquivos a serem carregados pelo Impact, simplesmente adicionando uma linha de código:

```
'game.levels.1-1',
```

Código 6: Carregar novo level

Logo após foi necessário fazer uma alteração no método *init()*, novamente adicionando uma linha de código para que o jogo carregasse o devido mapa.

```
this.loadLevel(Level11);
```

Código 7: Carregar mapa

Ainda no método *init()*, foi adicionado algumas linhas para que o Impact usasse sua função de capturar eventos do teclado como por exemplo a linha de código abaixo que define que quando o usuário apertasse a tecla C do teclado o Impact iria ligar a tecla para o evento shoot.

```
ig.input.bind( ig.KEY.C, 'shoot' );
```

Código 8: Capturar evento teclado

5.3.2 Classe Personagem

Logo após algumas alterações na classe principal do jogo, foi necessário criar uma nova classe para o jogo, essa classe em princípio é a entidade do jogador. As entidades são tudo que pertencem ao nível, mas não ao mapa, como por exemplo, o personagem, os inimigos, os eventos, entre outros.

Para criar a classe foi necessário criar um novo arquivo chamado player.js dentro da pasta lib/game/entities, sendo assim possível esta classe estender a classe principal das entidades sendo possível herdar alguns comportamentos básicos.

O seguinte código mostra a estrutura básica de como se devem criar entidades no Impact:

```
ig.module(
'game.entities.player'
)
.requires(
'impact.entity'
)
.defines(function(){
EntityPlayer=ig.Entity.extend({

    });
});
```

Código 9: Estrutura entidade

Na classe personagem foi definido algumas características básicas como, por exemplo, o arquivo de Sprite que será usado por essa classe, algumas opções físicas para o personagem e como será sua animação.

Logo após a criação dessas características a entidade jogador já vai estar disponível no Weltmeister para ser adicionada ao nível.

Figura 12: Entidade jogador no weltmeister
Fonte: Elaborado pelo autor

5.3.3 Classe Zombie

A criação de um monstro (Zombie) é similar á criação da classe do personagem principal. Nela também foram definidas algumas características básicas para que a entidade esteja disponível no Weltmeister.

5.3.4 Detecção de Colisão

Logo após ter criado o personagem principal e um inimigo, foi necessário pesquisar sobre sistemas de detecção de colisão no Impact. O Impact já vem com um método pronto para realizar a detecção, o sistema de colisão presente nele é baseado em Bounding Box, um sistema simples que realiza análise utilizando retângulos imaginários em torno de um Sprite. Por exemplo, se a imagem do Sprite for do tamanho 25 x 25 o sistema cria um retângulo do mesmo tamanho sobre a imagem. Adicionando uma propriedade. type em uma classe permite criar grupos de entidades, dessa forma pode-se facilmente fazer que entidades amigas só colidam com as inimigas.

É importante notar que um problema deste método de colisão é que ele não leva em conta qualquer espaço transparente em torno da Sprite, como mostrado na imagem abaixo.

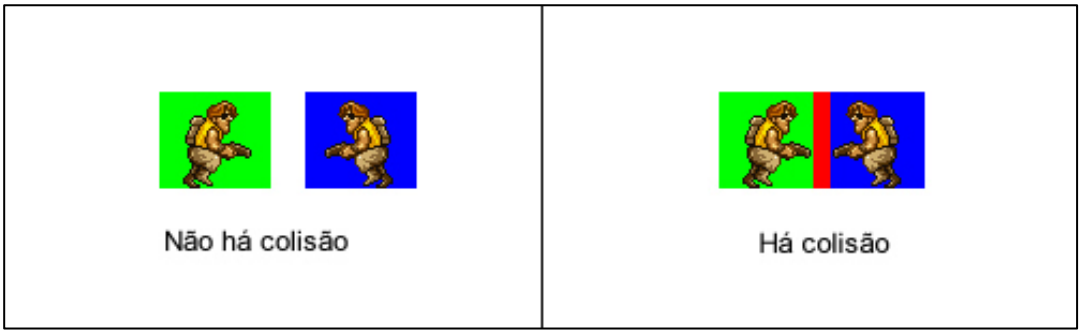

Figura 13: Exemplo colisão
Fonte: Elaborado pelo autor

Para resolver este problema o Impact oferece o uso do offset que realiza uma diminuição do tamanho desse retângulo de colisão, por exemplo, o código abaixo define que adicionando um valor de offset em 10 para x e 5 para y, o retângulo que realiza a detecção da colisão será menor em 10 pixels no eixo x e 5 no eixo y.

```
offset: {x:10, y:5},
```

Código 10: Uso do offset em colisões

5.3.5 Armas

Até este ponto o jogador estava indefeso, ou seja, quando ocorresse a colisão entre ele e um monstro ele morreria, não havia nenhum modo para que ele matasse o inimigo. Para resolver isto foi necessário criar uma classe interna dentro do arquivo player.js

```
EntityBullet=ig.Entity.extend({
size: {x:3, y:3},
animSheet: newig.AnimationSheet('media/tiro.png', 3, 3),
maxVel: {x:400, y:0},
type: ig.Entity.TYPE.NONE,
checkAgainst: ig.Entity.TYPE.B,
collides: ig.Entity.COLLIDES.PASSIVE,
init: function( x, y, settings){
this.parent( x+ (settings.flip ? -4 : 8) , y+18, settings);
this.vel.x=this.accel.x= (settings.flip ?-this.maxVel.x : this.maxVel.x);
this.addAnim('idle', 0.2, [0]);
      },
handleMovementTrace: function(res){
this.parent(res);
if(res.collision.x||res.collision.y){
this.kill();
        }
      },
check: function(other){
other.receiveDamage(3, this);
this.kill();
      }
  });
```

Código 11: Código entidade bala

O código acima basicamente define que a bala irá para a direção onde o personagem está virado, e que quando ela atinge um monstro ela irá aplicar um dano de 3 pontos ao inimigo.

Para que a arma seja disparada foi usado o seguinte código dentro do método update:

```
// shoot
if(ig.input.pressed('shoot') ) {
        ig.game.spawnEntity(EntityBullet, this.pos.x, this.pos.y, {flip:this.flip} );
}
```

Código 12: Disparar bala

Ele define que cada vez que a tecla de atirar for apertada o Impact deve gerar uma nova bala, na determinada posição onde o jogador está.

5.4 Segunda Fase da Implementação

Basicamente neste ponto já estava pronto uma primeira versão de teste do jogo, onde somente existia um inimigo e apenas um nível de jogo. Logo após está fase foi pensado em como o jogo podia ser melhorado, os principais pontos levantados foram: fazer a câmera seguir o jogador, adicionar mais um nível do jogo, como alguns sons e telas de inicio do jogo, além de algumas opções de I.A. nos inimigos.

5.4.1 Câmera

Até este momento enquanto o jogador andava com seu personagem pelo mapa a câmera do jogo não seguia o personagem, para resolver isto foi adicionado o seguinte código na função *update* da classe *main.js:*

```
//camera seguirjogador
var player =this.getEntitiesByType( EntityPlayer )[0];
if ( player ) {
        x =player.pos.x- (ig.system.width / 3);
        y =player.pos.y- (ig.system.height );
this.screen.x= (x >0&& x <this.mapWidth) ?x :this.screen.x;
this.screen.y= (y >0&& y <200) ?y :this.screen.y;
}
```

Código 13: Câmera

Este trecho de código usa uma importante API do Impact, ela permite-nos encontrar determinadas instâncias no jogo. Já que no jogo somente há um personagem principal, podemos fazer com que a câmera olhe diretamente para ele. Depois de o código verificar se o jogador existe, pegando-se a posição do jogador no mapa e também o tamanho da tela, ao definir um valor para *screen.x* e *screen.y* o processador irá ajustar automaticamente a câmera para essa posição.

5.4.2 Sons

Para adicionar sons ao jogo foi necessário usar a Classe ig.Sound. O Impact possui suporte a dois formatos de sons: Ogg Vorbis e MP3. Abaixo segue um exemplo de como configurar uma instância ig.Sound:

```
var sound =newig.Sound( 'media/sounds/atirar.ogg' );
var sound =newig.Sound( 'media/sounds/atirar.mp3' );
var sound =newig.Sound( 'media/sounds/atirar.*' );
```

Código 14: Sons

O último exemplo é muito importante, pois ele permite ao Impact usar o ig.SoundManager carregar o arquivo corretamente para nós, baseado em qual deles é o formato usado melhor em cada navegador. Os arquivos de sons são como imagens para o Impact, por isso eles devem estar dentro da pasta *media/* no jogo foi criado um subdiretório *sounds* para que o jogo ficasse mais organizado. Agora era necessário adicionar cada som para a classe do jogador. Foi adicionado o seguinte código bem ao topo da classe *player.js*:

```
shootSFX: newig.Sound( 'media/sounds/atirar.*' ),
```

Código 15: Som

Ainda na mesma classe deve-se adicionar o seguinte código em negrito na determinada linha de código em que o jogador atira:

```
if(ig.input.pressed('shoot')){
ig.game.spawnEntity(EntityBullet, this.pos.x, this.pos.y, {flip:this.flip})
this.currentAnim=this.anims.shot.rewind();
this.shootSFX.play();
}
```

Código 16: Tocar som

Como visto no decorrer da faculdade e na elaboração deste trabalho, devemos estar preparados para as diversidades encontradas, sejam nos seres humanos, sejam nos computadores, os quais passam a concorrer com o ser humano na inteligência e a partir das mudanças, das discussões, e com o olhar voltado ao novo, considerando o momento e a perspectiva do jogo e do jogador.

5.4.3 Inteligência Artificial

IA ao contexto dos jogos implica que os oponentes controlados pelo computador apresentem certo grau de cognitividade (percepção, esperteza) ao enfrentar o jogador humano. Existem várias maneiras de se programar, mas o mais importante é que eles pareçam apresentar certo grau de inteligência.

Os jogos procuram se tornar cada vez mais realista, o que atrai novos jogadores. Nesse processo de busca por mais realismo pode-se citar a melhoria dos gráficos do jogo, o que implica em melhoria dos componentes de hardware, melhoria dos enredos e jogabilidade, dentre outros. Os jogadores sempre buscam um jogo não muito fácil, mas nem tão difícil ao ponto de serem impossíveis de se completar o objetivo, eles buscam um jogo onde predomine o equilíbrio de dificuldade. Para se chegar ao equilíbrio é necessário utilizar a IA, pois é ela que proporciona inteligência aos inimigos e também a outras entidades do jogo.

As técnicas de IA para jogos podem ser divididas em duas principais categorias, que são denominadas “determinísticas” e “não-determinísticas”, conforme descrevo a seguir:

Determinísticas: tem comportamento previsível, consome poucos recursos da máquina e é mais fácil de ser implementada. Um exemplo de comportamento determinístico é o algoritmo básico de perseguição (usado na implementação). O problema desta abordagem é que se faz necessário prever todas as ações possíveis em um determinado momento, fazendo-se uso de muitas regras do tipo "se-então"; além disso, após pouco tempo de jogo, o jogador consegue prever facilmente o comportamento da máquina, pois, para um mesmo conjunto de entradas, as mesmas respostas são atribuídas;

Não Determinísticas: possui um grau de incerteza, que pode variar a cada implementação; um exemplo de comportamento não determinístico inclui o aprendizado que um personagem da máquina adquire ao jogar contra um humano após um determinado período. Tal aprendizado pode ser praticado por uma rede neural ou algoritmo genético, por exemplo. É possível fazer com que a máquina possa inclusive exibir comportamentos que vão além do que lhe foi programado, mas há um, porém, pois, em alguns casos, a máquina pode exibir um comportamento não desejado. Outra dificuldade se encontrou nos testes, pois fica mais difícil testar todas as combinações de comportamento que a máquina poderá exibir durante a execução do jogo.

A locomoção é uma das premissas básicas de um agente. Tal característica é fundamental para que este consiga atingir seus objetivos. Um algoritmo de perseguição e fuga pode ser divido em duas partes(Seemann & Bourg, 2004): uma, é a tomada de decisão para que se inicie o processo, e outra é a ação propriamente dita. Pode-se dizer também que existe um terceiro processo a ser considerado, que é a evasão de possíveis obstáculos que possam estar no caminho encontrado pelo algoritmo.

O seguinte código mostra a prática do algoritmo de perseguição no impact:

```
var player =ig.game.getEntitiesByType(EntityPlayer)[0];
varxdir=this.flip ? -1 :1;
if ( this.distanceTo(player) <200 ){
var target =0;
target=player.pos.x;
            }
if ( target <this.pos.x ){
this.flip=true;
this.currentAnim.flip.x=false;
this.vel.x=this.speed*xdir;
            } elseif( target>this.pos.x ) {
this.flip=false;
this.currentAnim.flip.x=true;
this.vel.x=this.speed*xdir;
            }

if (this.vel.x!=0){
this.currentAnim=this.anims.andando;
            }
```

Código 17: IA

Primeiramente o código seta duas novas variáveis uma variável player sendo a entidade 0 que é a entidade do jogador e a outra a xdir que é responsável em realizar um operador condicional. Logo em seguida o código propõe que se a distancia para o player for menor que 200 o destino do zombie vai ser a posição onde o jogador está, ou seja, ele vai andar em direção à posição do jogador. O restante do código realiza alterações de animações no zombie, além de adicionar a ação necessária para que o zombie possa ir em direção ao jogador com velocidades pré-determinadas na entidade do zombie.

5.5 Telas do Jogo

Em síntese apresentaremos algumas telas do jogo Zumbie Invader para que se possa ter uma pequena idéia do funcionamento do mesmo.

Figura 14: Tela início do jogo
Fonte: Elaborado pelo autor

Figura 15: Tela Nível 1
Fonte: Elaborado pelo autor

Figura 16: Tela inimigo nível 1
Fonte: Elaborado pelo autor

Figura 17: Tela fim do nível 1
Fonte: Elaborado pelo autor

Figura 18: Tela Game Over
Fonte: Elaborado pelo autor

6. CONCLUSÃO

O desenvolvimento de jogos eletrônicos é uma das atividades mais interessantes e mais complexas na área da computação, pois integra os mais diversos assuntos como Computação Gráfica, Inteligência Artificial, Sistemas Distribuídos, Engenharia de Software, Estrutura de Dados e Multimídia.

Ao fim do trabalho, o Impact provou que é uma ferramenta robusta para o desenvolvimento de jogos a qual pode ser utilizada por desenvolvedores iniciantes como forma de se inserirem nessa área de desenvolvimento. No entanto, sozinho, ele não é suficiente. Um exemplo disso é de que o jogo foi desenvolvido usando imagens e sons previamente distribuídos em sites sobre desenvolvimento de jogos. O Impact, por ser um framework para desenvolvimento da lógica e da mecânica do jogo, não pode solucionar esse tipo de problema.

Outro problema do Impact é de que, apesar de sua robustez, ele provê poucos componentes nativos para gerenciar algumas funções básicas de um jogo, como plotagem de gráficos sem auxílio de uma textura. É até possível realizar tal tarefa com uso de funcionalidades nativas, mas a complexidade dessa realização retira qualquer vantagem de sua utilização, tornando preferível usar a alternativa menos difícil, porem, mais sujeita a problemas de desempenho da aplicação.

No Impact é possível a utilização de componentes criados pela comunidade desenvolvedora do framework para gerenciar determinadas funções as quais o Impact, nativamente, não provê. No entanto, além de muitos deles possuírem a total ausência de um padrão de projeto, fazendo o desenvolvedor perder tempo adaptando essas funcionalidades para seu padrão, seu uso pode tornar questionável a autoria do desenvolvimento do jogo.

Acredito que o objetivo deste trabalho foi alcançado, embasado no feedback obtido por amigos que tiveram a oportunidade de jogar o Zombie Invader deram suas opiniões sobre todas as versões do jogo. É possível questionar de que alguns aspectos essenciais de um jogo digital estão faltando no Zombie Invader, como por exemplo, uma trilha sonora. Um trabalho futuro possível seria incluir essas características em uma nova versão.

Há muitos jogos divertidos feitos em anos anteriores os quais, em minha visão atual de jogo divertido, também são desprovidos de muitos aspectos essenciais de um jogo digital, como trilha sonora, curva de aprendizado, etc. Muitos deles foram refeitos para os

tempos atuais, através da inclusão dessas características. No entanto, nunca aconteceu de um desses jogos refeitos ter ultrapassado a diversão e a lembrança do jogo anterior. Isso é explicado pelo fato de que é impossível refazer o momento, o feeling, o ambiente em que aquele jogo foi criado naquela época. Esses fatores têm um papel crucial no desenvolvimento de qualquer trabalho; não sendo somente apenas para jogos digitais.

Espero que este trabalho seja um ponto de partida para futuros desenvolvedores iniciarem seu caminho na criação de um jogo.

Por fim, constata-se que os jogos digitais em geral exercem um papel condicionante e modificador no usuário. As atuações das novas tecnologias provavelmente influenciarão ainda muitas gerações no futuro, deixando nelas as suas marcas.

7. REFERÊNCIAS BIBLIOGRÁFICAS

Apple. (1 de Janeiro de 2010). *Thoughts on Flash.* Acesso em 17 de Outubro de 2012, disponível em Apple: http://www.apple.com/hotnews/thoughts-on-flash/

Battaiola, A. L. (2001). Desenvolvimento de jogos em computadores e celulares. *Revista de Informática Teórica e Aplicada* .

Berjon, R., Leithead, T., Navara, E. D., O'Connor, E., & Pfeiffer, S. (s.d.). *HTML5*. Acesso em 20 de Outubro de 2012, disponível em W3C: http://dev.w3.org/html5/spec/single-page.html

Farias, J. A. (2010). *XNA para desenvolvimento de jogos no Windows, Zune e Xbox 360.* Rio de Janeiro: Brasport.

Fernandes, R. (4 de Julho de 2010). *Características – Microsoft Silverlight 4 & Adobe Flex 4*. Acesso em 1 de Novembro de 2012, disponível em RIASoftware: http://www.riasoftware.com.br/blog/?p=1233

Lawson , B., & Sharp, R. (2011). *Introducing HTML5.*

Lynch, K. (2 de Fevereiro de 2010). *Open Access to Content and Applications*. Acesso em 6 de Novembro de 2012, disponível em Adobe Featured Blogs: http://blogs.adobe.com/conversations/2010/02/open_access_to_content_and_app.html

Magazine, G. D. (20 de Maio de 2011). Why use Impact.

Salen, K., & Zimmerman, E. (2004). *Rules of Play: Game Design Fundamentals.* Cambridge: MIT Press.

Seemann, G., & Bourg, D. (2004). *AI for Game Developers.* O'Reilly Media.

Sepherd, E. (2010). *Canvas Tutorial*. Acesso em 2 de Novembro de 2012, disponível em MDN: https://developer.mozilla.org/en-US/docs/Canvas_tutorial

Xavier, L. M. (07 de Maio de 2007). *Javascript orientado a objetos*. Acesso em 19 de Outubro de 2012, disponível em MDN: https://developer.mozilla.org/pt-PT/docs/Javascript_orientado_a_objetos

Referências Imagens:

Figura 1:
http://coding.smashingmagazine.com/2009/07/16/html5-and-the-future-of-the-web/html4.jpg
Acessado em 10/11/2012

Figura 2:
http://coding.smashingmagazine.com/2009/07/16/html5-and-the-future-of-the-web/html5.jpg
Acessado em 10/11/2012

Figura 3:
http://www.criarweb.com/artigos/images/html5/coordenadas-de-canvas.gif
Acessado em 12/11/2012

Figura 6:
http://impactjs.com/files/teaser/ztype.jpg
Acessado em 07/11/2012

Figura 7:
http://impactjs.com/files/teaser/creaturescastle.jpg
Acessado em 07/11/2012