

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-  
GRANDENSE - IFSUL, CAMPUS PASSO FUNDO  
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET**

**RÔMULO REIS DE OLIVEIRA**

**DESENVOLVIMENTO DE JOGOS ELETRÔNICOS ONLINE EM  
TEMPO REAL, PARA MÚLTIPLOS JOGADORES E  
MÚLTIPLOPLATAFORMAS**

**Élder F. F. Bernardi**

**PASSO FUNDO, 2012**

**RÔMULO REIS DE OLIVEIRA**

**DESENVOLVIMENTO DE JOGOS ELETRÔNICOS ONLINE EM  
TEMPO REAL, PARA MÚLTIPLOS JOGADORES E  
MULTIPLATAFORMAS**

Monografia apresentada ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-Rio-Grandense, *Campus* Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Élder F. F. Bernardi

**PASSO FUNDO, 2012**



**RÔMULO REIS DE OLIVEIRA**

**DESENVOLVIMENTO DE JOGOS ELETRÔNICOS ONLINE EM TEMPO REAL,  
PARA MÚLTIPLOS JOGADORES E MULTIPLATAFORMAS**

Trabalho de Conclusão de Curso aprovado em \_\_\_\_/\_\_\_\_/\_\_\_\_ como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Banca Examinadora:

---

Prof. Me. Élder Francisco Fontana Bernardi

---

Prof. Me. Evandro Miguel Kuszera

---

Prof. Me. Lisandro Lemos Machado

---

Prof. Me. Evandro Miguel Kuszera  
(Coordenação do Curso)

**PASSO FUNDO, 2012**

## **AGRADECIMENTOS**

Primeiramente devo agradecer ao meu orientador, Élder Francisco Fontana Bernardi, pela paciência, apoio e por acreditar em mim. Aos meus pais, Romário Gayer de Oliveira e Maria Regina Reis de Oliveira, por toda a educação e o afeto que me forneceram e pelos estímulos nos momentos difíceis. Ao meu Companheiro, Diego Henrique da Silva, o qual entendeu minha ausência em vários momentos. À minha amiga Aline Zanin, que não cansava de me oferecer ajuda. À minha colega Suélen Camargo, que criou todos os elementos gráficos do protótipo e aos demais amigos que conseguiram compreender a importância deste trabalho e que não viraram as costas para mim, mesmo quando tive de negar ajuda por estar sobrecarregado.

Também gostaria de agradecer aos professores que fizeram parte da minha formação, principalmente ao (in memoriam) professor Juliano Menegaz, o qual me ensinou que devemos fazer o que amamos e devemos aproveitar ao máximo nossa vida, sem nos deixar abalar por coisas pequenas.

## RESUMO

A adoção em massa de novas plataformas de jogos, como *smartphones* e *tablets*, contribuiu para a mudança no perfil dos jogadores, os quais querem jogar seus jogos favoritos em diversos lugares, de diferentes formas e em diferentes plataformas. Além disso, impulsionou a indústria de jogos, por causa da facilidade de compra e também trouxe novos desafios para o desenvolvimento de jogos. Um desses desafios é escolher de que forma o jogo será desenvolvido, pois, quando optado pelo desenvolvimento de maneira genérica, o jogo é produzido apenas uma vez para ser executado em várias plataformas, mas provavelmente não será apto para utilizar os recursos ímpares de cada plataforma. Quando é optado pela produção de jogos para cada plataforma específica, o jogo pode utilizar os recursos ímpares de cada plataforma, como o GPS do *smartphone*, entretanto, o mesmo jogo será desenvolvido para cada plataforma. Outro desafio é o desenvolvimento de jogos eletrônicos online em tempo real, para múltiplos jogadores e multiplataformas, pois esse tipo de jogo tem vários pontos críticos que impactam no desempenho do mesmo. Motivado por estes desafios, essa pesquisa investigativa sobre desenvolvimento de jogos propõe o desenvolvimento de um protótipo, entretanto, a arquitetura do primeiro protótipo desenvolvido não foi eficiente para o tipo de jogo que foi proposto, sendo assim, foi necessário pesquisar uma forma de resolver os problemas ocorridos nesta arquitetura, então, uma possível solução foi encontrada e implementada em um segundo protótipo. Após concluir o desenvolvimento do segundo protótipo, o desempenho de ambas as arquiteturas e tecnologias foram medidas e comparadas, sendo que, a arquitetura e tecnologia utilizada para o desenvolvimento do segundo protótipo obtiveram melhores resultados.

Palavras-chave: desenvolvimento de jogos; desenvolvimento de jogos multiplataformas; desenvolvimento de jogos MMO.

## ABSTRACT

The mass adoption of new games platforms, as smartphones and tablets, has contributed to the changing in the players' profile, who wants to play their favorites games in several places, by different ways and on different platforms. Moreover, it has boosted the game industry, because of the ease of purchasing and it has also brought new challenges to the game development. One of these challenges is to choose the way the game will be developed, because, when chosen the generic development way, the game is produced only once for running on several platforms, but it probably will not be able to use the odd resources from each platform. When it is chosen the games production for each specific platform, the game is able to use the odd resources of each platform, as the GPS smartphones, however, the same game will be developed for each platform. Other challenge is the cross platform real-time massively multiplayer online electronic games, because this kind of game has a lot of critical points which impacts on its performance. Motivated by these challenges, this investigative research about game development propose the development of a prototype, however, the first prototype developed architecture was not efficient for the game type that it has been proposed, thus, it was necessary to research a way to solve the problems occurred in this architecture, so, a possible solution was found and implemented on a second prototype. After completing the second prototype development, the performance of both architectures was measured and compared, thus, the architecture and technology used for developing the second prototype has obtained better results.

Key words: games development; cross-platform games development; MMO games development.

## SUMÁRIO

1	INTRODUÇÃO .....	7
1.1	MOTIVAÇÃO .....	8
1.2	OBJETIVOS.....	9
1.2.1	Objetivo Geral .....	9
1.2.2	Objetivos específicos .....	9
1.3	ORGANIZAÇÃO DO DOCUMENTO .....	9
2	JOGOS ELETRÔNICOS.....	10
2.1	CLASSIFICAÇÕES DE JOGOS ELETRÔNICOS.....	10
2.1.1	Dimensionalidade .....	10
2.1.2	Perspectiva de visão do jogador .....	10
2.1.3	Número de Jogadores Simultâneos.....	11
2.1.4	Tipos de Jogabilidade e Mecânica do Jogo .....	12
2.1.5	Gêneros de Jogos .....	13
2.1.6	Classificações dos Tipos de Jogos Quanto ao Público Alvo .....	14
2.1.7	Classificação dos tipos de jogadores .....	15
3	DESENVOLVIMENTO DE JOGOS ELETRÔNICOS .....	16
3.1	CICLO DE DESENVOLVIMENTO DE JOGOS ELETRÔNICOS.....	16
3.1.1	Concepção.....	16
3.1.2	Pré-Produção.....	16
3.1.3	Produção .....	17
3.1.4	Pós-Produção .....	18
3.1.5	Pós-Lançamento .....	18
4	DOCUMENTO DE <i>GAME DESIGN</i> .....	19
4.1	PRIMEIRO DOCUMENTO – “ <i>THE ONE-SHEET</i> ” .....	19
4.2	SEGUNDO DOCUMENTO – “ <i>THE TEN-PAGE</i> ” .....	19
4.3	DOCUMENTO DE <i>GAME DESIGN</i> .....	23
5	PLATAFORMAS.....	24
5.1	PORTATÉIS .....	24
5.2	COMPUTADOR PESSOAL (PC).....	25
5.3	CONSOLES .....	26



6	DESENVOLVIMENTO DE JOGOS ONLINE MULTIPLATAFORMA PARA MÚLTIPLOS USUÁRIOS .....	27
6.1	TECNOLOGIAS E CONCEITOS .....	27
6.1.1	<i>Massively Multiplayer Online Games</i> .....	27
6.1.2	Jogos Multiplataformas .....	30
6.1.3	<i>Engine</i> .....	32
6.1.4	Jogos eletrônicos para navegadores .....	37
7	PROTÓTIPO .....	39
7.1	Desenvolvimento dos Protótipos .....	39
7.1.1	Desenvolvimento do primeiro protótipo .....	40
7.1.2	Desenvolvimento do segundo protótipo .....	42
7.1.3	Primeira abordagem para tratar múltiplos jogadores simultâneos no protótipo .....	45
7.1.4	Segunda abordagem para tratar múltiplos jogadores simultâneos no protótipo .....	46
7.1.5	Avaliação das abordagens para tratar múltiplos jogadores simultâneos nos protótipos 51	
7.1.6	Ambientes de Testes .....	51
7.1.7	Testes .....	52
8	CONSIDERAÇÕES FINAIS .....	55
	REFERÊNCIAS .....	56
	APÊNDICES .....	58
	APÊNDICE I .....	58
	APÊNDICE II .....	60
	APÊNDICE III .....	66

## 1 INTRODUÇÃO

“O mercado mundial de jogos eletrônicos movimentava mais de 56 bilhões de dólares anuais” (SOLLITTO, 2012), e esse valor tende a aumentar, visto que esse mercado sofreu um grande impulso com as novas oportunidades de negócios geradas pelo surgimento e adoção em massa de novas plataformas como celulares, *tablets* e *smartphones*, os quais apresentam *hardwares* e preços cada vez mais atrativos, além dos novos modelos de negócios proporcionados pela facilidade de acesso à Internet.

A grande maioria desses novos dispositivos permite o acesso à internet via rede *wireless* ou 3G, o que facilita consideravelmente a compra de jogos eletrônicos, além de permitir que as pessoas estejam conectadas à Internet a maior parte do tempo. Entretanto, o acesso à rede 3G ainda é limitada. Hoje a cobertura 3G atinge 45% da população, mas é previsto que em 2017 a rede 3G será acessível a 85% da população mundial. (TRAFFIC, 2012)

Tendo como base os pronunciamentos das grandes empresas de jogos e consoles durante o evento IE3<sup>1</sup> em 2012, foi possível notar que os novos jogos têm tendência a serem *online*, sociais, multiplataformas ou apresentarem alguma forma de integração entre distintas plataformas de *hardware*.

Com todas essas tendências no cenário mercadológico, é necessário analisar de maneira mais detalhada a forma com que os jogos eletrônicos são projetados, pois quando a produtora restringe o meio de acesso a esse jogo, a mesma perde a oportunidade de alcançar um número maior de pessoas, além disso, os jogadores querem ter acesso ao jogo de qualquer plataforma.

Uma maneira adotada para minimizar esse problema é a utilização de *engines* de desenvolvimento que suportam exportar um jogo eletrônico para várias plataformas. Entretanto, deste modo, o jogo pode sofrer perdas em alguns aspectos, visto que há variação no *hardware* entre as plataformas, além disso, dificulta ou inviabiliza a utilização das características ímpares de cada plataforma, como, por exemplo, o GPS do *smartphone*. Por esses motivos, algumas empresas optam por desenvolver um projeto para várias plataformas em paralelo, o que acaba gerando grandes gastos.

---

<sup>1</sup> Maior exposição do mundo de produtos relacionados à jogos eletrônicos e plataformas.  
[www.e3expo.com](http://www.e3expo.com)

Outra maneira adotada é a utilização do conceito de Arquitetura Orientada a Serviços (SOA do inglês, *Service-Oriented Architecture*) ao projetar o jogo. Entretanto, isso torna o jogo obrigatoriamente *online*, visto que funcionará através de serviços *web*, contudo, como citado anteriormente, hoje a cobertura da internet alcança um significativo número de pessoas no mundo, sendo este o principal motivo desta solução estar sendo largamente adotada no mercado atual.

Após essa breve apresentação das abordagens adotadas no cenário atual do desenvolvimento de jogos eletrônicos, pode-se perceber que há muitos aspectos a serem pesquisados e melhorados no desenvolvimento de jogos computacionais. Com base nisso, o presente trabalho apresenta uma pesquisa sobre o desenvolvimento de jogos eletrônicos multiplataformas de acordo com o cenário atual, levando em consideração aspectos como plataformas, acessibilidade, tecnologias, metodologias e processos no desenvolvimento de jogos.

Com base nos resultados obtidos através desta pesquisa, foi proposto o desenvolvimento de um estudo de caso, o qual consiste de um jogo que adota os processos e tecnologias estudadas no decorrer da pesquisa. Esse estudo de caso é um jogo de tiro online em tempo real para múltiplos jogadores e multiplataformas, onde vários jogadores compartilham o mesmo mundo virtual do jogo.

## 1.1 MOTIVAÇÃO

O mercado de jogos eletrônicos se mantém longe da crise, além disso, várias novas tecnologias estão surgindo para auxiliar o desenvolvimento de jogos para multiplataformas e para suprir as novas necessidades dos consumidores, que inclui o desejo de jogar seu jogo favorito de qualquer dispositivo em qualquer lugar.

Embora já haja várias maneiras de abordar o desenvolvimento de jogos multiplataformas, ainda não há um consenso comum das melhores maneiras de desenvolver esse tipo de jogo a fim de obter o melhor custo-benefício, pois cada abordagem tem suas vantagens e desvantagens, além dos problemas comuns enfrentados. Também não há consenso de como conseguir utilizar o máximo de características e recursos de cada plataforma.

## **1.2 OBJETIVOS**

### **1.2.1 Objetivo Geral**

Pesquisar sobre as tecnologias, metodologias e processos adotados no desenvolvimento de jogos eletrônicos multiplataformas no cenário atual e desenvolver um protótipo com base nos resultados desta pesquisa.

### **1.2.2 Objetivos específicos**

- Estudar aspectos gerais dos jogos eletrônicos;
- Pesquisar sobre metodologias e tecnologias para desenvolvimento de jogos multiplataformas;
- Propor um protótipo;
- Identificar quais metodologias e tecnologias melhor atendem os requisitos do protótipo;
- Desenvolver um protótipo de um jogo computacional online em tempo real, para múltiplos jogadores e independente de plataforma;

## **1.3 ORGANIZAÇÃO DO DOCUMENTO**

Para contextualizar o leitor, o capítulo dois, apresenta alguns conceitos sobre jogos eletrônicos, Já o capítulo seguinte, descreve os processos para desenvolver um jogo. Ou seja, todo o ciclo de vida desse tipo de software, no qual são geradas várias documentações, que forma o documento de game design, o qual é abordado no capítulo 4. Já o capítulo 5 é dedicado às plataformas de jogos disponíveis no mercado e quais são os mais populares.

O capítulo 6 apresenta os conceitos e as tecnologias utilizadas para montar o protótipo, mas é no capítulo 7 que é descrito como foi o processo de desenvolvimento do protótipo, quais foram os problemas enfrentados e os resultados, além disso, é efetuada a comparação entre duas abordagens e tecnologias que foram utilizadas para o desenvolvimento dos protótipos. O último capítulo é dedicado às considerações finais do estudo, destacando as principais contribuições da pesquisa, e descrevendo possíveis melhorias.

## **2 JOGOS ELETRÔNICOS**

Neste capítulo são abordados conceitos e explicações sobre o desenvolvimento de jogos eletrônicos, além de apresentar o cenário atual do mercado de jogos eletrônicos.

Um jogo é “um sistema formal, fechado, que representa um subconjunto de realidade” (CRAWFORD, 1982, P.7), já SALEN (2004) define jogo como um “sistema onde o jogador se engaja em um conflito artificial, definido por regras, que resultam em um resultado quantitativo”, logo, jogo eletrônico pode ser definido como um software que sustente a definição de jogo. Para melhor compreensão deste trabalho, a palavra jogo deve ser interpretada como jogo eletrônico.

### **2.1 CLASSIFICAÇÕES DE JOGOS ELETRÔNICOS**

Neste capítulo são apresentadas algumas classificações que um jogo pode ter, essas moldadas a fim de criar uma identidade entre jogos semelhantes e para facilitar o estudo das características de cada jogo com base nos seguintes critérios de classificação: dimensionalidade, perspectiva de visão do jogador, número de jogadores simultâneos, tipos e gêneros de jogos (LOPES, 2006).

#### **2.1.1 Dimensionalidade**

Segundo LOPES (2006), “Esse critério considera o número de dimensões em que o jogo se expressa”, o parâmetro para tal classificação é o número de coordenadas usadas para representar a posição de cada objeto no cenário do jogo, sendo classificados como jogos 2D, quando a representação do jogo ocorre em duas dimensões e jogos 3D, quando ocorre representação tridimensional (LOPES, 2006).

#### **2.1.2 Perspectiva de visão do jogador**

A classificação de um jogo quanto à perspectiva de visão do jogador está fortemente ligada à posição da câmera no jogo ou o ponto de vista do jogador. Este tópico é muito importante dentro do projeto de um jogo, pois a escolha errada pode frustrar o jogador fazendo com que o mesmo desista do jogo. Grande parte dos jogos adota uma das três principais perspectivas de visão do jogador a seguir:

## Primeira Pessoa

Um jogo computacional é considerado perspectiva em primeira pessoa quando o jogador vê as ações através dos olhos do personagem controlado pelo jogador durante toda ou maior parte da partida, como ocorre nos jogos da série Doom<sup>2</sup> e em alguns jogos da série Quake<sup>3</sup>, ambos produzidos pela ID Software<sup>4</sup>.

## Terceira Pessoa

Um jogo é considerado em terceira pessoa quando as ações do personagem não são vistas através de sua perspectiva, mas ao redor do personagem controlado pelo jogador, como ocorre na série Banjo & Kazooie<sup>5</sup> produzido pela Rare<sup>6</sup>.

## Top-Down

Jogos *top-down* têm o ponto de vista do jogador em cima do personagem, como se o jogador estivesse vendo o jogo de um helicóptero, como ocorre no jogo Age of Empires II<sup>7</sup>.

### 2.1.3 Número de Jogadores Simultâneos

Jogos eletrônicos também podem ser classificados pelo número máximo de jogadores simultâneos que o mesmo suporta, conforme descrito a seguir.

#### Jogador único

Um jogo é caracterizado como de jogador único (*single-player*) quando apenas um jogador pode jogar, ou seja, não são permitidos jogadores simultâneos, mesmo que o jogo permita que dois jogadores se revezem ele ainda é considerado *single-player* por causa da não simultaneidade, como ocorre jogo de plataforma lançado pela Sega<sup>8</sup> em 1991 para o console Mega Drive, Sonic the Hedgehog.

## Múltiplos Jogadores

---

<sup>2</sup> [www.idsoftware.com/games/doom/](http://www.idsoftware.com/games/doom/)

<sup>3</sup> [www.idsoftware.com/games/quake/](http://www.idsoftware.com/games/quake/)

<sup>4</sup> [www.idsoftware.com](http://www.idsoftware.com)

<sup>5</sup> [www.rare.net/games/banjokazooie-nuts-bolts](http://www.rare.net/games/banjokazooie-nuts-bolts)

<sup>6</sup> [www.rare.net](http://www.rare.net)

<sup>7</sup> <http://www.microsoft.com/games/age2/>

<sup>8</sup> [www.sega.com](http://www.sega.com)

Um jogo é considerado para múltiplos jogadores (*multiplayer*) quando permite mais de um jogador simultâneo como ocorre no jogo Dance Central 2<sup>9</sup> produzido pela Harmonix Music Systems<sup>10</sup>.

### ***Massively Multiplayer Online (MMO)***

Jogos MMO são jogos onde um grande número de jogadores pode interagir entre si e entre um mundo persistente, como ocorre no jogo World of Warcraft<sup>11</sup> produzido pela Blizzard Entertainment<sup>12</sup>.

### **2.1.4 Tipos de Jogabilidade e Mecânica do Jogo**

Há vários tipos genéricos de jogos no que diz respeito à interação entre o próprio jogo e o jogador, como descrito a seguir.

#### **Tempo real**

Jogos computacionais em tempo real são aqueles em que as ações e os acontecimentos do jogo não param para que o jogador faça uma inserção de dados ou comandos, como ocorre na série Age of Empires produzido pela Ensemble Studios<sup>13</sup>.

#### **Baseado em turnos**

O tempo de jogo é dividido em turnos e a cada turno o jogo é pausado para aguardar entrada de dados/comandos do jogador, no instante em que o jogador entra com os dados é encerrado o turno, podendo ou não ocorrer alguma ação em jogo. Um exemplo de jogo baseado em turnos é o Final Fantasy Tactics lançado para o PlayStation Portable pela Square Enix<sup>14</sup>.

#### **Baseado em texto**

---

<sup>9</sup> [www.dancecentral.com](http://www.dancecentral.com)

<sup>10</sup> [www.harmonixmusic.com](http://www.harmonixmusic.com)

<sup>11</sup> [us.blizzard.com/pt-br/games/mists](http://us.blizzard.com/pt-br/games/mists)

<sup>12</sup> [www.blizzard.com](http://www.blizzard.com)

<sup>13</sup> [www.ensemblestudios.com](http://www.ensemblestudios.com)

<sup>14</sup> [www.square-enix.com](http://www.square-enix.com)

São jogos computacionais onde as entradas e saídas de dados ocorrem através de texto, podendo utilizar gráfico como recurso secundário, mas sempre tendo a principal interatividade com o jogador via texto.

### 2.1.5 Gêneros de Jogos

Durante o decorrer dos anos, os jogos foram categorizados em diferentes gêneros e subgêneros. Segundo Rogers (2010) o gênero de um jogo descreve o estilo de *gameplay* ou jogabilidade. Com base em Rogers (2010) e Thomas (2007) serão descritos os principais gêneros e subgêneros de jogos eletrônicos.

**Ação:** jogos de ação necessitam de habilidade do jogador com os controles e comandos para jogar, normalmente está acompanhado de um outro gênero.

**Aventura:** jogos deste gênero focam em solução de problemas, coleta de itens, exploração de cenários, sem exigir do jogador habilidades com o controle.

**Plataforma:** este gênero de jogo tem como desafio passar por passagens complexas chamadas de plataformas, pulando, nadando, atirando ou lutando, ou seja, desafiando as habilidades do jogador exigindo precisão quanto aos controles.

**Tiro:** neste gênero de jogo o jogador tem como objetivo atirar projéteis em alvos.

**Role-Playing Game (RPG):** neste gênero o jogador deve interpretar um personagem, além de evoluir o mesmo durante o jogo pelo acúmulo de experiência adquirida através de batalhas, exploração e solução de problemas.

**Massively Multiplayer Online Role-Playing Game (MMORPG):** adota o gênero RPG, porém permite que os jogadores interajam entre si e entre um mundo persistente.

**Sobrevivência e horror:** neste gênero o jogador deve sobreviver a um cenário de horror com recursos limitados, o jogo tenta passar sentimento de medo ao jogador através de um cenário de suspense.

**Quebra-Cabeça (Puzzle):** jogos que focam em desafios de problemas lógicos, segundo o SALEN (2004), esse gênero de jogo apresenta apenas uma solução.

**Estratégia:** jogos deste gênero focam em elaborar estratégias/táticas baseando-se nas regras do jogo para ganhar do oponente ou conquistar um objetivo.

**Esportes:** jogos de esportes têm pretensão de simular algum tipo de esporte.

**Simulação:** são jogos computacionais, onde a principal proposta é simular os processos do mundo real, ou então ,quando combinado com um gênero de jogo, tem o propósito de deixar



explícito o grau de realismo abordado no game, como ocorre em Gran Turismo 5<sup>15</sup>, o qual podemos descrever como sendo um jogo de simulação de corridas (*Racing Simulation*). Muitas vezes esses jogos não têm o objetivo final explícito como ocorre nas renomadas séries *The Sims*<sup>16</sup> e *SimCity*<sup>17</sup>.

**Corrida:** tem como foco a distância percorrida em determinado período de tempo.

**Luta:** tem como mecânica, combate “um a um” entre dois personagens.

### 2.1.6 Classificações dos Tipos de Jogos Quanto ao Público Alvo

É necessário focar em um público alvo específico antes de começar a desenvolver um jogo, pois o mesmo deve ser pensando em conquistar o maior número possível de pessoas que façam parte desse público. Sendo assim, os jogos apresentam um conjunto de características para satisfazer seu público alvo, esse conjunto de características permite a classificação geral dos jogos em jogos casuais, jogos tradicionais, jogos sociais e jogos educacionais.

Nos jogos casuais o jogador completa uma partida do jogo em poucos minutos, podendo esse jogo ser jogado apenas para passar o tempo por qualquer pessoa, visto que jogos casuais possuem uma curva de aprendizado mínima, ou seja, não há necessidade de ter nem uma habilidade para jogar, além disso, tem um preço menor que jogos *hardcore*.

Já os jogos tradicionais, mais conhecidos por jogos *hardcore*, exigem vários minutos ou horas para que o jogador tenha uma experiência completa no jogo, ou seja, que consiga ter acesso a todos os recursos do jogo. Além disso, algumas vezes exige dedicação do jogador em pesquisas e estudos, pois são jogos mais complexos.

Os jogos sociais têm como foco redes sociais, onde é possível haver interatividade entre seus contatos, isso permite cooperação e competição. Já os jogos educacionais têm como foco transmitir conhecimento ao jogador. Entretanto, jogos sociais ou educacionais podem ser também enquadrados como jogos casuais ou tradicionais.

---

<sup>15</sup> <http://www.polyphony.co.jp/english>

<sup>16</sup> [http://thesims.com/en\\_us/home](http://thesims.com/en_us/home)

<sup>17</sup> [http://www.simcity.com/en\\_US](http://www.simcity.com/en_US)

### 2.1.7 Classificação dos tipos de jogadores

Primeiramente os jogadores foram divididos por Bartle (1996) em quatro categorias, empreendedores, exploradores, socializadores e matadores. Já Klug (2006) identifica e classifica os jogadores de forma mais específica, conforme descrito a seguir.

O primeiro tipo de jogador é o jogador competidor, o qual quer ser sempre melhor que os outros jogadores, o jogador explorador, que quer explorar todo o jogo para descobrir primeiro o que os outros jogadores ainda não descobriram o jogador colecionador, que joga para obter todos os itens do jogo e o jogador empreendedor, que joga para ser o melhor jogador de todo o jogo, para estar presente nos rankings e ganhar todos os campeonatos.

Além destes, há também o jogador brincalhão, o qual joga para se divertir e gosta de aspectos sociais inseridos no jogo, o diretor, o qual quer sentir a emoção de estar no comando, ele quer orquestrar os eventos, o fluxo do jogo, o jogador narrador joga com o objetivo de criar ou viver em um mundo alternativo, o artesão, que joga para construir e resolver enigmas e o exibicionista, que joga com o objetivo de ser o astro do jogo.

Tanto para Bartle (1996) quanto para Klug (2006), a maioria dos jogadores é uma combinação de duas ou mais categorias de jogadores, por isso os jogos tendem a adotar táticas para abordar a maior quantidade possível de categorias de jogadores alterando o foco do jogo durante o *gameplay*.

### 3 DESENVOLVIMENTO DE JOGOS ELETRÔNICOS

Neste capítulo serão abordados os assuntos referentes ao processo de desenvolvimento de jogos eletrônicos, desde sua concepção até o seu pós lançamento, além de toda a documentação gerada durante esse processo, que segundo Lopes (2006) assemelha-se a processos de desenvolvimento artístico, que envolvem concepção, produção e acabamento.

#### 3.1 CICLO DE DESENVOLVIMENTO DE JOGOS ELETRÔNICOS

De acordo com SALEN (2004) “*Game design* é o processo pelo qual o game designer cria um jogo [...]”, ou seja, é o processo de produção de um jogo eletrônico, o qual Sloper (2002) divide em cinco estágios, concepção, pré-produção, produção, pós-produção e pós-lançamento

##### 3.1.1 Concepção

De acordo com Berthém (2007), a concepção de um jogo ocorre de maneira individual ou durante uma reunião da equipe para *brainstorm*, onde são criadas e discutidas idéias para jogos, sendo que cada ideia deve ser analisada nos aspectos de originalidade, inovação, público-alvo, plataforma e possibilidades de mercado.

Quem coordena a concepção do projeto é o projetista de jogos (*Game designer*) que é responsável por ter e amadurecer ideias para criar os conceitos do jogo, além de projetar e acompanhar o desenvolvimento de cada item do documento de *game design*, o qual será escrito pelo próprio.

A definição e amadurecimento das ideias pré-selecionadas pelo projetista de jogos geram os conceitos que serão a base do jogo. Após isso é necessário realizar um estudo e verificar se o projeto será viável economicamente. Só após a comprovação da viabilidade econômica inicia-se o desenvolvimento do Documento Conceitual (*Conceptual Document*).

O Documento Conceitual contém a apresentação das idéias do jogo de maneira integrada, também é pré-definido o gênero do projeto, semelhanças e diferenças entre este e outros jogos, público alvo, fluxo geral e *look&feel* do mesmo.

##### 3.1.2 Pré-Produção

Neste estágio é desenvolvido o Documento de *Game Design* (*game design document*), que contém a descrição detalhada de todos os elementos necessários para o desenvolvimento

do projeto, tais como, *design* de jogabilidade, mecânica, apresentação, interface, interação, detalhamento técnico e arte. Visto a importância e extensão deste documento, foi dedicado o capítulo 4 para apresentá-lo detalhadamente.

Ainda nesta etapa, o escritor deve escrever o roteiro da história do jogo, história dos personagens, descrição da época e do ambiente em que o jogo se passa, além da narração e dos diálogos. Para Borges & Barreira & Souza (2009) é uma das partes fundamentais, pois é o roteiro que poderá convencer os investidores das potencialidades do jogo.

Segundo Berthém (2007), a partir do Documento de *Game Design*, o planejador de software (*softwareplanner*), que de acordo com Rogers (2010), é responsável por gerenciar o desenvolvimento do projeto, divide o projeto elaborado pelo projetista de jogos em um conjunto de requisitos técnicos, além de estimar tempo e esforço para a implementação de tais requisitos.

Ainda segundo Berthém (2007), o arquiteto de software (*lead architect*) trabalha em conjunto com o planejador de software sendo responsável pela arquitetura geral do projeto, sendo sua função definir um conjunto de especificações de módulos de acordo com os requisitos técnicos identificados pelo planejador de software.

Durante o trabalho do planejador de software e do arquiteto de software é gerada uma grande carga de trabalho que deverá ser balanceada pelo gerente de projeto (*Project manager*) a fim de planejar e organizar de maneira eficiente a distribuição das atividades entre a equipe, sendo que o mesmo deve cobrar e adaptar as atividades durante o desenvolvimento do projeto para que cumpram as metas no cronograma elaborado por ele.

### 3.1.3 Produção

Durante a produção é desenvolvido o Documento de Arte e o Documento de Som, que podem ser ou não anexados ao Documento de *Game Design*, além disso, é necessária a participação de vários profissionais nesta etapa, entre esses profissionais encontram-se:

- Programadores: são responsáveis por codificar o jogo utilizando uma linguagem de programação, eles implementam técnicas de computação gráfica, inteligência artificial, interação, sistemas de física, efeitos sonoros, sistemas de controles, sistemas de câmeras.
- Testadores: sua função é testar o jogo, procurando falhas, inconsistências e verificando o balanceamento do jogo.

- Escritores: responsáveis por escrever os textos do jogo, desde a história do jogo, diálogos, narrações, manuais.
- Designers/Artistas: ficam responsáveis por desenvolver um ou mais desses itens, criação de layout do jogo, personagens, cenários, objetos, texturas, ilustrações, animações, *storyboard*, efeitos visuais, *interface* de usuário, etc.
- Músicos/Sonoplastas: Responsáveis por compor e produzir a trilha sonora e os sons dos efeitos do jogo.

Cada profissional fica responsável por uma função específica, mas sempre em contato com os outros membros da equipe, para que haja coerência e equilíbrio entre a produção, também é necessário deixar claro que o Documento de *Game Design* ainda continua a ser desenvolvido e modificado.

#### **3.1.4 Pós-Produção**

Segundo Sloper (2002), deve ser elaborado um plano de testes para começar a etapa de testes dos últimos protótipos, onde após cada período de teste as falhas são reportadas e corrigidas, até que não haja um número significativo de falhas que impacte no funcionamento do jogo. Em paralelo à fase de testes ocorrem também os ajustes quanto aos aspectos artísticos e sonoros.

Outro documento que deve ser elaborado nesta etapa é o Post Mortem, que é um documento de *feedback*, onde a equipe deve registrar o que foi aprendido durante o desenvolvimento do projeto, assim como os pontos positivos e negativos do mesmo, esse documento tem como objetivo auxiliar a empresa nos futuros projetos.

#### **3.1.5 Pós-Lançamento**

O estágio de pós-lançamento é desconsiderado por alguns autores, como Keer (2006), pois este estágio ocorre após a liberação da versão final do jogo para a empresa responsável pelo lançamento e distribuição, neste estágio os clientes são observados para levantamento de dados estatísticos, esses dados têm função de guiar os próximos passos da empresa, dependendo dos resultados da estatística, a empresa sabe se é viável criar uma expansão ou seqüências do jogo.

## 4 DOCUMENTO DE *GAME DESIGN*

Como citado anteriormente, o documento de *game design* (GDD, do inglês Game Design Document) contém todas as especificações do jogo, Rogers (2010) defende que esse documento deve ser elaborado em três etapas, tendo início na pré-produção, sendo que cada etapa vai gerar um documento que será incrementado e amadurecido nas próximas etapas até formar a versão final de produção do documento de *game design*.

### 4.1 PRIMEIRO DOCUMENTO – “*THE ONE-SHEET*”

Rogers (2010) estruturou e nomeou cada documento gerado nessas quatro etapas, sendo o primeiro denominado de “*The one – sheet*”, pois o conteúdo deste deve preencher uma folha de papel, este documento deve apresentar uma visão geral do jogo abordando os seguintes tópicos:

- Título do jogo;
- Sistema pretendido do jogo;
- Idade do público alvo;
- Pretensão da avaliação do jogo quanto à idade;
- Resumo da história focando o *gameplay*;
- Modos de jogo;
- Pontos extras e diferenciais;
- Jogos similares e concorrência;

O primeiro documento do jogo proposto neste trabalho está no Apêndice I, esse documento faz a primeira apresentação do jogo, dando uma idéia geral do que será proposto.

### 4.2 SEGUNDO DOCUMENTO – “*THE TEN-PAGE*”

O documento de design “*The Ten-Page*” tem como proposta, apresentar ao leitor uma idéia do produto final, mas sem se deter a muitos detalhes de tópico, além disso, deve-se levar em consideração o leitor deste documento, pois o mesmo pode ser elaborado tendo em vista a equipe de produção, a equipe de marketing ou os financiadores do projeto, por este motivo pode haver mais de uma versão deste documento, entretanto é válido citar que pode ser elaborado só um documento que seja acessível a todos os tipos de leitores.

Durante a elaboração deste documento para a equipe de produção, são providenciados diagramas claros do *gameplay*, comparações do *gameplay* com outros jogos similares neste aspecto e também outros detalhes sobre o jogo, sempre que possível usando terminologias específicas e frases curtas e diretas para esclarecer as intenções do jogo.

Quando este documento é elaborado, para os financiadores do projeto ou para a equipe de marketing, é interessante incluir imagens excitantes do jogo e também descrições usando exemplos e títulos de jogos modernos que fazem sucesso no mercado, além de textos simples e empolgantes.

O segundo documento do protótipo se encontra no Apêndice II e tem como foco a equipe de produção, visto que o mesmo tem propósito acadêmico.

Rogers (2010) estruturou o conteúdo deste documento por tópicos, onde cada tópico corresponde aproximadamente uma página de conteúdo, entretanto isso pode variar de projeto para projeto, ou seja, cada tópico será responsável por um tópico que irá descrever características específicas do jogo, mas não necessariamente vai ocupar uma página inteira. Também é importante ressaltar que a ordem dos tópicos pode variar visando a melhor formatação do documento.

#### **a) Primeira Página – Folha de Rosto**

A primeira página deve fornecer informações básicas sobre o que se trata o resto do documento, apresentando o título do jogo, de preferência já com a logo marca, sistema do jogo, faixa etária do público alvo, avaliação da idade recomendada pelo órgão responsável, previsão de lançamento do produto.

#### **b) Segunda Página – Esboço do Jogo**

A segunda página deve apresentar o resumo de toda a história e fluxo do jogo e o fluxo da mesma, descrevendo o tipo do jogo, gênero, os desafios que serão encontrados durante o jogo e quais habilidades o jogador terá para superá-los, os locais onde ocorrerá a trama, como ocorrerá a evolução do personagem e quais os objetivos do mesmo.

#### **c) Terceira Página – O Personagem**

A terceira página é dedicada ao personagem principal, onde deve constar sua biografia, descrevendo ao longo da biografia as características físicas e psicológicas e também algumas armas, ataques e habilidades que o personagem pode utilizar. Após isso se pode

inserir a imagem de controle ou teclado descrevendo qual a função ou ação cada botão apresentará.

#### **d) Quarta Página – O *Gameplay***

A quarta página é dedicada ao *Gameplay*, primeiramente são apresentadas várias características do jogo incluindo os aspectos abordados na seção 2.1, também é descrito como serão divididas as fases do jogo, podendo ser por níveis, “*rounds*”, partidas, capítulos, etc. Outro ponto abordado são os cenários que o personagem será imerso quais obstáculos ele enfrentará e quais ações ele pode ter para vencer esses obstáculos, isso pode incluir dirigir, pilotar, atirar, pular, enfrentar entre outros verbos.

Ainda nesta página é descrito qualquer mini-game que faça parte do jogo principal, assim como quais são os recursos extras que o jogador pode ter, tais como, *download* conteúdo extra, quantidade de mini-jogos durante o jogo, possibilidade de jogar online, integração com rede social.

Também são especificados todos os requisitos tecnológicos específicos de cada plataforma adotada, no caso de computadores pessoais (PC) é descrito os requisitos mínimos e recomendáveis para executar o software, já no caso de consoles atuais, deve informar como funcionará o sistema de entrada do jogo, se ele vai necessitar de controles, teclados ou algum hardware que capture os movimentos do jogador.

Por fim é informada a maneira que o jogo será distribuído. Quando optado por distribuir via *download*, deve ser informado qual o tamanho do arquivo. Entretanto se for adotado uma mídia digital, deve ser informado o tipo de mídia que será utilizado.

#### **e) Quinta Página – O Mundo do Jogo**

Nesta página devem ser apresentados os cenários e ambientes que o personagem irá frequentar durante o jogo, assim como os desafios que serão encontrados pelo jogar em cada cenário, também é importante esclarecer como esses cenários e ambientes se ligam entre eles, qual o propósito de cada um.

#### **f) Sexta Página – Experiência de Jogo**

Nesta parte do documento devem ser descritas as emoções que o jogador deve sentir durante o jogo, quais sentimentos as cenas, os cenários e as músicas devem provocar no jogador, qual a primeira visão do *gameplay* e qual a primeira impressão o jogador deve ter do jogo.



### **g) Sétima Página – Mecânica do Jogo**

A mecânica do jogo são todas as regras do jogo e como todo o sistema vai funcionar com todas essas regras integradas, isso inclui as possibilidades de interação entre o jogador e os elementos do jogo, como mover plataformas, abrir portas, ser cortado por uma lamina, ser morto por uma chama de fogo, sistema de combate, sistema de casamento, sistema de troca de mensagens.

Como exemplo comum pode-se citar *Power-up* que é um tipo de elemento que tem função de auxiliar o jogador no *gameplay*, são considerados *Power-up* itens de cura, itens que aumentam algum atributo do jogador, etc. (Rogers, 2012)

Já os elementos coletáveis/coleccionáveis são itens que podem ser coletados, mas não apresentam impacto no *gameplay*. Normalmente esses itens são moedas ou estrelas, eles podem ou não ter uma função, normalmente são utilizados para destravar opções no jogo ou liberar acesso a determinados ambientes ou habilidades. (Rogers, 2012)

Caso o seu jogo utilize algum sistema econômico, deve-se descreva-lo e explicar como os jogadores irão conseguir esse “dinheiro” e como eles poderão utilizar esse “dinheiro”.

Cada mecânica do jogo deve ser apresentada através da especificação da função da mecânica, uma introdução, a descrição detalhada e algumas considerações finais, caso haja alguma.

### **h) Oitava Página – Inimigos**

A oitava página apresenta as características dos inimigos que fazem parte do jogo, assim como quais habilidades eles têm, como podem ser evitados ou derrotados, aspectos visuais e história de cada inimigo. É importante citar a força do inimigo, a frequência de participação do mesmo e se ele irá possuir algum tipo de inteligência artificial.

### **i) Nona Página – Cenas**

As cenas podem ser feitas utilizando várias tecnologias. Podem por exemplo, serem filmadas com atores reais ou serem simples animações, que descreva como e quando as cenas do jogo serão apresentadas ao jogador, se serão apresentadas no início ou fim de cada fase ou então após cada ação do jogador.

### **j) Décima Página – Materiais de Bônus**

Um recurso para que o jogador continue jogando após concluir os principais objetivos do jogo é incluir fases extras, ambientes secretos, entre outros segredos que só podem ser acessados após o jogador concluir todas as principais missões do jogo e é na última página que devem ser apresentadas todas as razões para que o jogador continue jogando.

### **4.3 DOCUMENTO DE *GAME DESIGN***

Após o desenvolvimento do documento "*The Ten-Page*" é iniciado a evolução do mesmo, que resultará no Documento de *Game Design*, este documento deve conter todos os tópicos do documento "*The Ten-Page*" detalhados, e também a maneira que as regras serão introduzidas ao jogador, além de um conjunto de todos os itens que fazem parte do jogo.

O documento de *game design* do jogo proposto neste trabalho se encontra no Apêndice III.

## 5 PLATAFORMAS

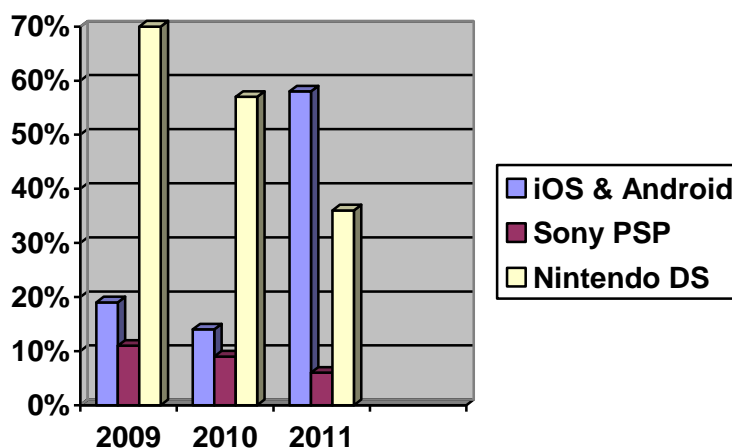
É apresentado neste capítulo as plataformas disponíveis no mercado, considerando plataforma como um conjunto de hardware e software para rodar um determinado aplicativo.

A escolha da plataforma de hardware adotada em um projeto é um fator importante, pois cada plataforma tem um perfil de usuário/consumidor, então é necessário definir o objetivo que se pretende atingir com projeto antes de selecionar a plataforma, além disso, a escolha da plataforma está ligada ao sucesso, fracasso e lucratividade do jogo.

### 5.1 PORTATÉIS

Segundo a Flurry Analytics<sup>18</sup>, com base nos dados obtidos do National Purchase Diary<sup>19</sup> (NPD Group), empresa de estudo de mercado de jogos eletrônicos nos Estados Unidos, as plataformas que tenham sistema operacional iOS ou Android estão crescendo de maneira rápida no mercado de games para plataformas portáteis, como pode ser notado Gráfico 1, entretanto, mesmo a tendência sendo que esse número não pare de aumentar, ainda é cedo para afirmar que os consoles portáteis específicos para games serão descontinuados, visto que na Ásia eles ainda tem uma grande fatia do mercado.

Gráfico 1 – Receita por venda de jogos para plataformas portáteis nos Estados Unidos da América



Fonte: Flurry Analytics, 2012.

<sup>18</sup> [www.flurry.com](http://www.flurry.com)

<sup>19</sup> [www.npd.com](http://www.npd.com)

Um dos fatores para o crescimento das plataformas com sistema operacional iOS e Android é o custo benefício, pois um console portátil específico para jogos custa em média \$200,00 e cada cartucho de jogo custa em média \$40,00, já uma plataforma com sistema operacional Android ou iOS, além de ter um número significativo de recursos a mais, os jogos podem ser adquiridos através da internet, muitas vezes gratuitamente ou por um preço baixo, normalmente \$0,99 cada jogo pago.

O preço baixo dos jogos para sistemas iOS e Android ocorre basicamente por três motivos, o primeiro motivo são as baixas taxas pagas para o Google e Apple, que ficam responsáveis pelas vendas e cobram uma pequena taxa pelo serviço, outro motivo é a não necessidade de mídia física, pois os games são obtidos via *download*, o terceiro fator é o grande número de clientes em potencial.

A Apple declarou que há 250 milhões de dispositivos com sistema iOS ativados, já a Google, 190 milhões de dispositivos com sistema Android ativados, sendo que a receita gerada por venda de games pelas duas plataformas nos Estados Unidos foi de 2.7 bilhões de dólares no ano de 2009, sendo estimada para 2010 e 2011 uma receita de 2.5 bilhões e 3.3 bilhões respectivamente.

A partir dessas informações é possível afirmar que o desenvolvimento para essas plataformas pode ser lucrativo, pois a produção de um game para essas plataformas é de baixo orçamento, além disso, a empresa responsável estará visível por aproximadamente 440 milhões de pessoas, baseando nos dados apresentados pela empresas Apple<sup>20</sup> e Google<sup>21</sup> e considerando que cada dispositivo tenha um usuário único.

## 5.2 COMPUTADOR PESSOAL (PC)

Computadores e internet estão a cada dia mais acessíveis para a maior parte dos habitantes de países desenvolvidos ou emergentes, isso reflete diretamente no faturamento das vendas de jogos que adotam esta plataforma.

Durante a *Game Developer Conference* em 2012, a PC Game Alliance declarou que as vendas de jogos computacionais para PC continuam a aumentar, em 2010 gerou uma receita de \$16.2 bilhões de dólares, 20% a mais que em 2009, mas o recorde de faturamento foi batido em 2011, onde o faturamento foi de \$18.6 bilhões de dólares, 15% a mais que em 2010.

---

<sup>20</sup> [www.apple.com](http://www.apple.com)

<sup>21</sup> [www.google.com](http://www.google.com)

Uma pesquisa, encomendada pela PC Game Alliance, conclui que a venda de jogos para PC vai continuar crescendo e prevê um faturamento de aproximadamente \$25.5 bilhões de dólares em 2015, causado pelo aumento do acesso à tecnologia e facilidade das soluções adotadas para distribuição e pagamento.

### 5.3 CONSOLES

Hoje os consoles que mais se destacam são *Nintendo Wii*<sup>22</sup>, *Sony PlayStation 3*<sup>23</sup>, *Microsoft Xbox 360*<sup>24</sup> e mesmo com a adoção em massa dos dispositivos portáteis, os jogadores ainda preferem os consoles como pode ser observado no Gráfico 1, onde constatamos que em 2009 os consoles dominavam 71% do mercado, já em 2010 a taxa subiu para 75%, de qualquer forma é importante notar que iOS e Android tiveram um aumento de 3% na fatia de mercado dos portáteis.

---

<sup>22</sup> Nintendo Wii, lançado em 2006 pela Nintendo Company, Limited.

<sup>23</sup> Sony PlayStation 3, lançado em 2006 pela Sony Computer Entertainment.

<sup>24</sup> Microsoft Xbox 360, lançado em 2005 pela Microsoft.

## 6 DESENVOLVIMENTO DE JOGOS ONLINE MULTIPLATAFORMA PARA MÚLTIPLOS USUÁRIOS

Desenvolver um protótipo ao final deste trabalho, baseado nas pesquisas efetuadas, era um dos objetivos do mesmo, sendo assim, neste capítulo são apresentados algumas tecnologias e conceitos necessários para desenvolvimento do protótipo proposto no Apêndice III.

### 6.1 TECNOLOGIAS E CONCEITOS

Para que fosse possível propor uma idéia de protótipo que fosse condizente com o contexto atual, foram ressaltados alguns pontos relevantes no decorrer deste trabalho, principalmente a respeito da tendência mercadológica e tecnológica da indústria de jogos.

Durante a pesquisa, foi possível perceber que jogos MMO e jogos multiplataformas são aspectos que estão em alta na indústria de jogos, entretanto, são poucos os títulos disponíveis que abordam esses dois aspectos, isso ocorre, pois, o desenvolvimento de jogos MMO multiplataformas ainda é difícil, visto que há poucas *engines* eficientes e várias limitações nas existentes, principalmente no que diz respeito ao MMO.

A partir dos pontos citados no parágrafo anterior e também no decorrer de todo o trabalho, foi definido que o protótipo seria um jogo MMO multiplataformas. Para isso ser possível, foi necessário pesquisar quais tecnologias poderiam ser adotadas para desenvolver o mesmo, também foi necessário definir alguns conceitos que fazem parte desse tipo de jogo. Como citado anteriormente, o resultado desta pesquisa encontra-se descrito nesta seção.

#### 6.1.1 *Massively Multiplayer Online Games*

Como descrito anteriormente, *Massively Multiplayer Online Games* são jogos onde vários jogadores podem simultaneamente interagir com outros jogadores através de seus personagens em um mundo persistente.

Esse tipo de jogo tinha como principal limitador a velocidade da internet, tanto para o cliente, quanto para o servidor, mas atualmente esse não é o maior problema, hoje há vários elementos limitadores de escalabilidade desse tipo de jogo, entre eles os que mais se destacam são: arquitetura, linguagem de programação, banco de dados e sincronização.

### 6.1.1.1 Arquitetura de Jogos MMOG

O jogo proposto é multiplayer, em tempo real e, por isso, cada ação de um jogador reflete no mundo virtual que é compartilhado. Assim, todos os jogadores devem saber das ações efetuadas entre eles. Desta forma, a atualização entre os jogadores tem de ser constante. Por este motivo, fez-se necessário encontrar uma arquitetura que oferecesse suporte a tais requisitos.

Segundo Cronin, Filstrup e Kurc (2001) e Cronin, Jamin, Filstrup e Kurc (2004), há três arquiteturas que podem ser adotadas para jogos MMO, sendo que a arquitetura cliente-servidor (*Client-Server architecture*) é a mais utilizada, pois é facilmente implementada e permite o servidor fazer todo o estado e controle do jogo, ou seja, os usuários só se comunicam através do servidor, o qual também fica responsável pelo estado do jogo, o que dificulta trapaças (*Cheats*).

Já a arquitetura ponto a ponto (*Peer-to-peer architecture*) (P2P) tem uma implementação mais complexa, pois o jogo no cliente fica responsável pelo estado e controle do mesmo, o que facilita trapaças, entretanto, essa arquitetura apresenta uma baixa latência e elimina os gargalos no servidor.

A arquitetura de servidores espelhados (*Mirrored-Server architecture*) requer um complexo protocolo de consistência, tendo a implementação mais complexa entre as três arquiteturas, mas ela tem baixa latência e permite o controle de estado e fluxo do jogo nos servidores.

### 6.1.1.2 Escalabilidade de MMOG

A internet é um dos principais meios de compartilhamento de dados e a possibilidade de oferecer jogos *multiplayer* através da mesma tem como um dos principais desafios a escalabilidade e o desempenho. Ou seja, manter um sistema sempre disponível, com um bom tempo de resposta e robusto para uma demanda que pode ser de milhões de usuários simultâneos.

Atualmente, partes dos programadores adotam técnicas de programação paralela do modelo baseado em encadeamento de execução (*threads*), que permite dividir as aplicações em processos concorrentes ou cooperativos, tornando a execução dos mesmos mais eficientes, sempre que necessário manipular um grande número de requisições de entrada e saída (I/O), como ocorre em servidores interligados (arquitetura distribuída) que tratam de múltiplas conexões simultâneas de clientes distintos. (Tilkov, 2010)

Visto que as linguagens de programação server-side, como Java e PHP, consomem aproximadamente 2MB de memória por conexão efetuada ao servidor, isso gera um limite de conexões baseado na memória RAM da máquina, já que a mesma tem uma capacidade limitada de memória RAM que consegue gerir com desempenho, sendo inviável adotar o escalonamento vertical obrigando o escalonamento horizontal. (Abernethy, 2011)

Entretanto o escalonamento horizontal torna a arquitetura do sistema mais complexa, pois, uma arquitetura horizontalmente escalável necessita de servidores para balanceamento, uma rede muito bem estruturada, entre outros fatores que aumentam a probabilidade de problemas técnicos, além do alto tráfego, o que gera altos custos para manter essa arquitetura. Enquanto uma arquitetura verticalmente escalável necessita somente de melhorar o hardware, o que pode se tornar financeiramente inviável, mas em suma, ambas as arquiteturas são caras e limitadas. (Abernethy, 2011)

Sendo assim, a prática das técnicas de programação paralela, além de tornar o desenvolvimento da aplicação mais complexa, apresenta vários gargalos, impostos pelas características e limitações das linguagens de programação e do hardware.

Por essa e outras razões surgiu a necessidade de efetuar alguma otimização que permitisse um grande número de conexões simultâneas, escalável, com um menor consumo de memória RAM.

Uma abordagem para minimizar esse problema foi a utilização de programação orientada a eventos, onde as aplicações têm como centro de referência os eventos, que são indicações que algo ocorreu, sendo que há dois atores nesse modelo de programação, o produtor do evento e o consumidor do evento. (Junior, 2012)

Uma linguagem de programação orientada a eventos que se encaixa bem nesse contexto é o Java script, pois o mesmo segue o modelo de eventos assíncronos de entrada e saída, além de suporte a callbacks. Por exemplo, no navegador web, quando um dado é recebido ou uma requisição ajax é efetuada, é considerado um evento, o qual pode disparar um callback, ou seja, sempre que ocorre um evento no cliente o servidor pode ser notificado, tomando as devidas providencias e notificando clientes específicos ou todos os clientes. (Tilkov, 2010)

A partir da necessidade gerada por todo o contexto descrito anteriormente, surgiu o Node.js<sup>25</sup>, que “é uma plataforma que tem como objetivo fornecer uma maneira fácil de criar

---

<sup>25</sup> <http://nodejs.org/>



aplicações de rede rápidas e escaláveis”<sup>26</sup>. Para isso, Node.js aborda de uma outra maneira de como as conexões ao servidor são tratadas, ao invés de alocar memória para cada conexão, ele cria um processo por conexão efetuada, sem necessidade que o bloco de memória o acompanhe. (Abernethy, 2011)

Isso ocorre porque ao ser invocado um nó de servidor de processos (*node server process*), ele roda apenas em um *thread* que suporta um alto número de conexões, isso é possível, pois há um *loop* implícito que cobre o código, esse *loop* é denominado de *event loop* e tem como função esperar os eventos e repassá-los ao manipulador de eventos. (Tilkov, 2010)

O node.js também não faz uso de bloqueios. Mesmo ao realizar chamadas de E/S não há bloqueios diretos, ele trata as entradas e saídas de forma assíncrona, sendo assim dificilmente haverá gargalos, perda de desempenho ou problemas gerado por bloqueios, pois a aplicação não será bloqueada para esperar alguma operação de E/S. (Tilkov, 2010)

Além disso, é possível compartilhar *sockets* entre processos através da biblioteca multi-node (Zyp, 2010), sendo assim, pode-se ter múltiplos nós de servidores de processos trabalhando em paralelo, cada um em core do processador, mas ATENDENDO/ESCUTANDO a mesma porta, assim o próprio SO atua como balanceador de carga. (Junior, 2012)

Deste modo, um servidor Node.js pode suportar dezenas de milhares de conexões simultâneas, pois ele altera todo o contexto do servidor e o único gargalo passa a ser a capacidade de tráfego de um sistema e não mais o número de conexões. (Abernethy, 2011)

### 6.1.2 Jogos Multiplataformas

Tendo o termo plataforma sido definido pode-se abordar o conceito de multiplataformas, que é um atributo conferido ao software que é interoperado em várias plataformas distintas. Em outras palavras, multiplataformas é o termo utilizado para descrever uma linguagem ou software que funcione em mais de uma plataforma de hardware.

É importante destacar que cada plataforma apresenta características específicas e recursos extras, então o jogo deve ser adaptado para cada plataforma a fim de prover uma melhor experiência de jogo e aproveitar os recursos/características de cada plataforma, o que será um desafio para o desenvolvimento do protótipo proposto.

---

<sup>26</sup> <http://nodejs.org/>

Michael Kalkowski, cofundador e diretor criativo da *Game Duell*, apresentou durante o evento *Casual Connect Europe 2012* um modelo genérico adotada pela Game Duell e que pode ser utilizado por empresas que desenvolvem jogos online multiplataformas. Essa estrutura é composta por cinco pilares<sup>27</sup>.

### **Primeiro pilar – Marcas (*Brands*)**

Esse pilar foi denominado brands/marcas, pois de um ponto de vista mercadológico, visa levar além do jogo, a marca dos seus jogos para várias plataformas.

### **Segundo pilar – *Marketing***

Neste pilar o marketing tem de ser multiplataformas, por exemplo, se o usuário efetua o login em uma rede social pelo celular e tenta acessar o aplicativo pelo, é informado ao usuário que há uma versão deste aplicativo para celular, ou ainda, é possível informar aos jogadores os outros jogos que a empresa lançou.

### **Terceiro pilar – Gráfico Social Compartilhado (*Shared Social Graph*)**

O terceiro pilar é caracterizado pela utilização de recursos do gráfico social compartilhado para o jogador poder verificar quais os amigos que jogam o mesmo jogo, os pontos ou posição no ranking de cada amigo, além de enriquecer a experiência de jogo, permitir que os jogadores possam convidar os amigos e auxiliar os jogadores a se socializarem, já que o jogo será algo em comum entre os jogadores.

### **Quarto pilar – Economia Virtual Compartilhada (*Shared Virtual Economy*)**

O quarto pilar é mais referente ao modelo de negócio, ocorre quando é permitido ao jogador adquirir, através de dinheiro real, bens virtuais ou vantagens no jogo e conseguir resgatar o que foi adquirido independente da plataforma que esteja jogando, ou seja, todos os bens virtuais ou vantagens adquiridas no jogo devem ser acessíveis de qualquer plataforma.

### **Quinto pilar – Gameplay Multiplataforma (*Cross-Platform Gameplay*)**

O quinto pilar ocorre quando o jogo tem conexão com ele mesmo independente de plataforma, ou seja, o jogador pode iniciar o jogo em uma plataforma e continuar o mesmo

---

<sup>27</sup> KALKOWSKI, Michael. **Online-transplataform: Success Factors and Pitfalls of Cross-Platform Gaming**. Palestra no Casual Connect Europe 2012. Hamburg, 2012.

jogo em outra plataforma, normalmente utilizado por jogos multiplayer, onde os jogadores podem interagir entre si independentemente da plataforma que estão utilizando para jogar.

### 6.1.3 *Engine*

Uma *engine* (*game engine*, no original) é uma biblioteca, um pacote de funcionalidades que são disponibilizadas para facilitar o desenvolvimento de um jogo e impedir que sua criação tenha que ser feita do zero. (Kleina, 2011)

Difícilmente um jogo é desenvolvido sem uma engine, muitas vezes as empresas desenvolvem a *engine* para posteriormente desenvolver um jogo em específico, isso ocorre porque a *engine* ajuda na padronização do projeto e também aumenta a velocidade de desenvolvimento.

As *engines* que suportam desenvolvimento de jogos para navegadores e que mais foram vistos na pesquisa efetuada foram: Impact<sup>28</sup>, LimeJS<sup>29</sup>, Isogenic Engine<sup>30</sup>, RPGJS<sup>31</sup>, CraftyJS<sup>32</sup>, entretanto as engines Unity 3D e Construct 2 se destacaram pelos recursos oferecidos, além de ambas apresentam um conjunto de ferramentas completas para desenvolver o jogo proposto, por esses motivos, essas engines foram pré selecionadas para serem pesquisadas, e a partir desta pesquisa, relatada nas sessões 6.1.3.1 e 6.1.3.2, foi selecionada uma delas para a implementação do protótipo.

#### 6.1.3.1 Construct 2

Segundo a Scirra (<http://www.scirra.com>, recuperado em 15, junho, 2012), produtora da *engine*, o Construct 2 é uma *engine* proprietária, desenvolvida para auxiliar no desenvolvimento de jogos eletrônicos com gráficos 2D para navegadores web, utilizando recursos de HTML5, sendo assim não há necessidade de instalação de plugins nos navegadores, o que garante uma melhor experiência ao usuário.

De acordo com a Scirra, a *engine* adota as bibliotecas gráficas Canvas 2D e WebGL, e ainda a biblioteca de física Box2DWeb, todas essas bibliotecas são *open source* e tem documentação disponível na internet, já a documentação da *engine* é encontrada no próprio

---

<sup>28</sup> <http://impactjs.com/>

<sup>29</sup> <http://www.limejs.com/>

<sup>30</sup> <http://www.isogenicengine.com/>

<sup>31</sup> <http://rpgjs.com/>

<sup>32</sup> <http://craftyjs.com/>.

site, sendo que a mesma possui uma comunidade de desenvolvimento ativa a qual alimenta o fórum do Construct 2, o que contribui para a curva de aprendizagem sobre a *engine* ser de rápida evolução, sendo que em poucas horas o usuário consegue fazer projetos personalizados.

A Scirra também informa que a *engine* funciona somente em sistemas operacionais Windows XP ou superiores e permite importação de arquivos gerados por outras *engines* semelhantes, Além disso, tem integrado um simples editor gráfico e de animações, mas que cumprem com eficiência os seus propósitos.

Construct 2 ainda possui integração nativa com a API do Facebook, além de suporte a todos os recursos proporcionados pelo HTML5, como WebStorage, além de *touch screen*, Ajax, XML, JSON, tamanho de telas dinâmico e possibilidade do usuário jogar os jogos off-line. Se os recursos não forem suficientes, é possível estender o Construct 2 usando um SDK Java script. Segue, portanto, a ficha técnica da ferramenta, os recursos da *engine* na Tabela 1 e os tipos de licenças, conforme informado pela Scirra:

Sistema Operacional: Windows.

Plataforma: Navegador Web.

Gráfico: Gráficos 2D.

Bibliotecas Gráficas: Canvas 2D, WebGL.

Bibliotecas de Física: Box2DWeb.

Documentação: Sim, disponível no site, sem custos.

Tutoriais: Sim, disponível no site, sem custos.

Fórum Ativo: Sim.

Curva de aprendizagem: Fácil e Rápido.

**Tabela 1 - Recursos da *engine* Construct 2**

<b>Tabela de Recursos</b>
Integração nativa com API do Facebook
A engine possui um simples editor de imagens integrado
Pode-se estender o Construct 2 usando um SDK Java script
Suporte à touch screen, inclusive multi touch
Suporte à Ajax
Suporte à WebStorage

Suporte à XML
Suporte à JSON
Suporte à AppMobi
Suporte à Dicionário (para sistemas multilíngües)
Suporte à tamanhos diversos de tela
Suporte à jogos off-line

Fonte: Scirra (<http://www.scirra.com>, recuperado em 15, junho, 2012)

### **Licença: Software Proprietário**

Há três tipos de licenças, todas são adquiridas através de um único pagamento que cobre todas as atualizações e novas versões lançadas da engine, além de não expirar.

#### **Free Editon (valor: \$00,00)**

Apresenta praticamente todos os recursos da versão completa, entretanto impõe as seguintes limitações:

- Não pode ser usado para fins comerciais;
- Número máximo de eventos por projeto: 100;
- Número máximo de layers por projeto: 4;
- Não permite personalizar a organização das pastas do projeto;
- Não apresenta o sistema de busca por eventos;
- Não tem barra de configurações;
- Não permite criar famílias (utilizar de recursos como herança);
- Não permite pré-visualização em LAN;

#### **Personal Licence (valor: \$119,00)**

Essa licença é somente para pessoas físicas ou organizações sem fins lucrativos e permite o uso de todos os recursos da engine, porém caso o lucro obtido através dos jogos desenvolvidos com construct 2 ultrapassarem o valor de cinco mil dólares, o usuário deverá comprar a licença "Business License".

#### **Business License (valor: \$399,00)**

Essa licença é voltada para pessoas jurídicas ou pessoas físicas que obtém um lucro, proveniente de jogos desenvolvidos utilizando Construct 2, superior ao valor de cinco mil dólares.

### 6.1.3.2 Unity 3D

Segundo a Unity Technologies (<http://unity3d.com/>, recuperado em 15, junho, 2012), proprietária do Unity 3D, a *engine* foi criada para auxiliar no desenvolvimento de jogos eletrônicos com gráficos 3D. Essa *engine* se destaca por permitir a exportação do jogo criado para várias plataformas, além de permitir a personalização de projeto para cada plataforma. Assim é possível utilizar os recursos ímpares proporcionados por cada plataforma. Por exemplo, no caso do Android, pode-se utilizar recursos tais como *multi touch*, câmera, GPS, sensores, etc.

A Unity Technologies informa em seu site que esta *engine* permite exportar para as seguintes plataformas: Flash, iOS, Android, PC/Mac, Nintendo Wii, PS3, Xbox 36 e Navegador Web, entretanto este último necessita que o usuário instale um *plugin* no navegador.

De acordo com a Unity Technologies, a *engine* funciona em sistema operacional Windows XP, Mac OSx ou superiores e utiliza a biblioteca de física NVIDIA® PhysX® next-gen Physics Engine, também permite a importação de arquivos de vários softwares gráficos, além de ter vários subsistemas integrados à *engine*, como editor de imagens, editor de animação e renderização.

Mesmo sendo fácil encontrar vários materiais didáticos sobre a *engine*, a curva de aprendizagem da *engine* é longa e evolução relativamente lenta, sendo que para fazer algo personalizado o Unity 3D exige conhecimento do usuário, o qual, segundo a Unity Technologies, pode desenvolver seus jogos utilizando as linguagens JavaScript, C# e um dialeto de Python denominado de Boo.

Segue, portanto, a ficha técnica da ferramenta, os recursos da *engine* na Tabela 2 e os tipos de licenças, conforme informado pela Unity Technologies:

Sistema Operacional: Windows, Mac OS X.

Plataforma: Navegador Web (necessita plugin próprio), Flash, iOS, Android, PC/Mac Nintendo Wii, PS3, Xbox 360.

Gráfico: Gráficos 3D.

Bibliotecas de Física: NVIDIA® PhysX® next-gen Physics Engine.

Documentação: sim, porém pouco, disponível no site, sem custos.

Tutoriais: Sim, porém pouco, disponível no site, sem custos.

Fórum Ativo: Sim.

Curva de aprendizagem: Lento e difícil.

**Tabela 2 - Recursos da engine Unity 3D**

<b>Tabela de Recursos</b>
A <i>engine</i> possui vários subsistemas integrados, como editor de imagens, editor de animação, renderizador, etc.
Suporte à <i>touch screen</i> , inclusive <i>multi touch</i> (recurso disponível via pagamento extra)
Suporta importar arquivos do Maya, 3ds Max, Cinema 4D, Cheetah3D, Blender e outros
Suporta programação em JavaScript, C#, and a dialect of Python named Boo

Fonte: Unity Technologies (<http://unity3d.com/>, recuperado em 15, junho, 2012)

### **Licença: Software Proprietário**

A licença custa \$1.500,00 para cada versão da engine.

#### **6.1.3.3 Escolha da Engine**

Há várias diferenças entre as *engines* Unity 3D e Construct 2, uma delas é que a licença da primeira custa mais, porém, a mesma é completa e madura, enquanto a segunda engine tem o preço da licença menor, mas é recente e ainda falta implementar vários recursos. Outra diferença é o gráfico, pois o Unity 3D tem como foco gráficos em três dimensões, já o Construct 2 tem como foco apenas 2 dimensões.

Além disso, o Unity 3D necessita que o usuário instale um *plugin* no navegador web para poder jogar e o Construct 2 exporta de maneira nativa, tendo apenas como requisito navegadores web que suportem HTML 5, isso gera uma melhor experiência ao usuário, pois o mesmo apenas acessa o jogo, não precisando configurar todo um ambiente para jogar, isso aproxima o jogo do conceito de multiplataforma, pois como citado anteriormente, na atualidade é comum uma pessoa acessar a internet de vários dispositivos, e seria muito trabalhoso configurar cada dispositivo para poder jogar. Então, pelos motivos apresentados nesta seção, a *engine* Construct 2 foi selecionada para a implementação do protótipo.

#### 6.1.4 Jogos eletrônicos para navegadores

Segundo Scheib (2012), entre as principais vantagens do uso de navegadores como plataforma para jogos está o fato de não haver necessidade de instalação de plugins, além de o jogo ser executável diretamente no navegador e apresentar opções para deixar o jogo disponível mesmo quando não há acesso à Internet.

Embora apenas 47% dos navegadores desktops suportam executar jogos, somente o navegador Google Chrome, o qual suporta execução de jogos, tem mais de 160 milhões de usuários ativos, o que é o dobro do número de vendas do Nintendo Wii, onde cada venda não significa que o usuário é ativo, ou seja, mesmo a porcentagem de navegadores com suporte à HTML5, é possível alcançar mais usuários usando navegador do que outras plataformas. (SCHEIB, 2012).

Há vários recursos, implementados nos navegadores por causa do HTML 5, que por mais que seja só uma linguagem de marcação, serviu como um termo genérico para todas as novas tecnologias que surgiram para os navegadores, as quais podem ser utilizados para o desenvolvimento de jogos eletrônicos, entre esses recursos, podemos citar:

- File System API, permite acesso ao sistema operacional via navegador.
- Geolocation<sup>33</sup> é uma API que permite o servidor saber a localização de quem está acessando uma página.
- WebGL, se trata de uma API para Gráficos, neste caso o WebGL é a versão do OpenGL ES 2 para o contexto de navegadores web.
- Canvas, se trata de outra API para gráfico, com foco em gráficos 2D.
- Web Audio API<sup>34</sup> é uma API de áudio de alto desempenho e baixa latência que oferece vários recursos avançados para trabalhar com áudio.
- WebSocket<sup>35</sup> é uma API *Javascript*, sendo uma alternativa para *web services*, permite a comunicação bi-direcional entre cliente e servidor apresentando baixa latência.
- Web Workers<sup>36</sup> é uma API *Javascript* que permite a executar tarefas programadas em Javascript em segundo plano.

---

<sup>33</sup> <http://dev.w3.org/geo/api/spec-source.html>

<sup>34</sup> <https://dvcs.w3.org/hg/audio/raw-file/tip/webaudio/specification.html>

<sup>35</sup> <http://dev.w3.org/html5/websockets>

<sup>36</sup> <http://www.whatwg.org/specs/web-apps/current-work/multipage/workers.html>



- Scalable Vector Graphics<sup>37</sup> (SVG) permite a criação de gráficos utilizando XML, é similar ao Canvas em questão de funcionalidade e propósito.

---

<sup>37</sup> <http://www.w3.org/TR/SVG>

## 7 PROTÓTIPO

Este capítulo descreve como foram desenvolvidos os protótipos e quais abordagens foram adotadas para o jogo ser MMO, visto que o Construct 2 não oferece tal suporte e como descrito no documento de game design, Apêndice III, o protótipo é um jogo MMO em tempo real, multiplataforma, de tiro, horror e sobrevivência, sendo que para ser multiplataforma, foram utilizados o navegador como plataforma e a *engine* Construct 2 para o desenvolvimento, já que o mesmo tem recursos que permite aproveitar algumas características ímpares de algumas plataformas, como é o caso do *touch screen*.

Neste protótipo o jogador tem como objetivo principal sobreviver ao ataque de inimigos, podendo interagir com os demais jogadores e um ponto crítico do jogo é o fato dos recursos serem limitados, então o jogador deve estar sempre a procura de munição, alimentos e água para conseguir prosseguir no jogo.

Toda a especificação do protótipo se encontra no Apêndice I, II e III, sendo que para compreender este capítulo é necessária a leitura prévia destes documentos, além disso, para melhor entendimento deste capítulo é necessário o acompanhamento do documento de *game design* do jogo proposto presente no Apêndice III.

### 7.1 Desenvolvimento dos Protótipos

Primeiramente é necessário informar que foram desenvolvidos dois protótipos, ambos possuem as mesmas funções, porém, cada um aborda de maneira distinta o suporte à múltiplos jogadores simultâneos.

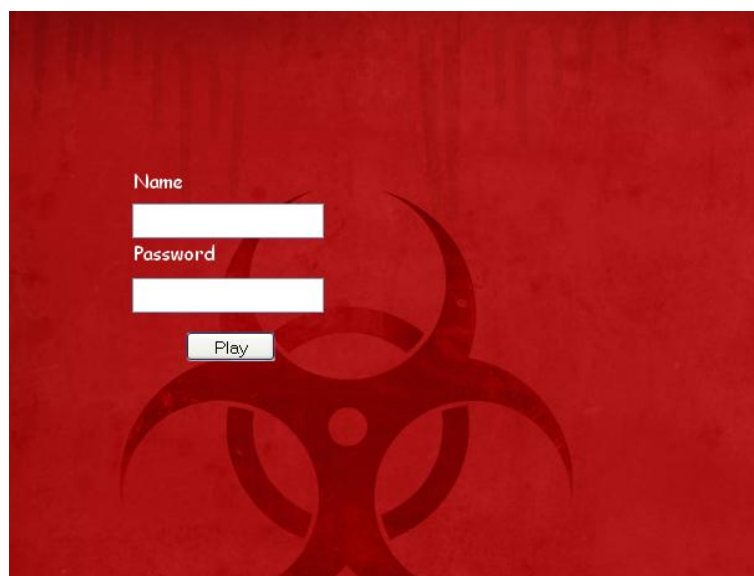
O ciclo de vida do protótipo iniciou pela concepção, onde foram organizadas algumas ideias e posteriormente foi escutada a opinião de algumas pessoas, então, após isso, o conceito do jogo foi definido e iniciou-se a etapa da pré-produção, onde foram elaborados os documentos *The one-sheet* (Apêndice I) e *The ten-page* (Apêndice II), os quais auxiliaram na formação do documento de *game design* (Apêndice III).

A etapa de produção foi iniciada logo após o documento de game design estar com todos os requisitos mínimos definidos, nessa etapa, foi criado um projeto na *engine* Construct 2, onde foi implementado uma tela de login e uma tela de mapa, com suas respectivas folhas de eventos, estas responsáveis por manipular os eventos do jogo. Nas próximas seções são descritos o desenvolvimento de cada protótipo utilizando o Construct 2.

### 7.1.1 Desenvolvimento do primeiro protótipo

Primeiramente foi criado um novo projeto no Construct 2, onde foi desenvolvida uma tela para o jogador efetuar o *login* para resgatar os dados do personagem e acessar o mundo virtual, essa tela pode ser observada na Figura 1.

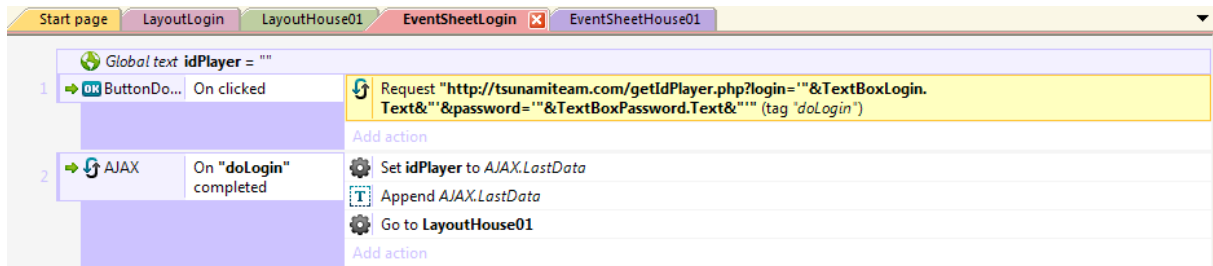
Figura 1 - Tela de login do primeiro protótipo



Fonte: Autor

No Construct 2 a principal programação é feita em páginas de eventos, onde são tratados todos os eventos que podem ocorrer em determinada tela. Neste caso, a página de eventos da tela de *login* efetua uma requisição ajax ao servidor PHP quando o botão “*Play*” da tela for pressionado, essa requisição envia ao servidor o *login* e a senha do jogador e obtém como resposta o número identificador do personagem (*idPlayer*), o qual é armazenado em uma variável global, como pode ser observado na Figura 2, após isso o jogador é redirecionado para uma outra tela.

Figura 2 - Página de eventos da tela de login do primeiro protótipo

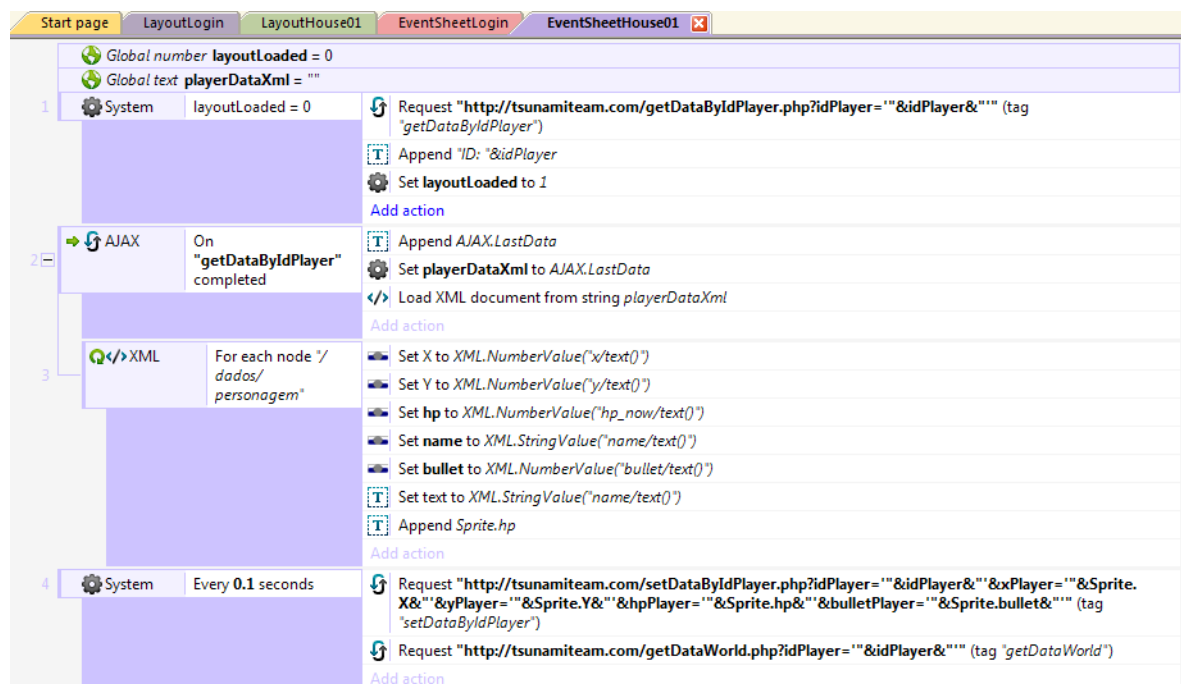


Fonte: Autor

A segunda tela é um mapa do mundo virtual, onde, assim que a tela estiver pronta, é efetuada uma requisição ajax para obter os dados iniciais do jogador, passando por parâmetro o identificador único do jogador obtido na tela de login, sendo assim, os atributos do jogador serão preenchidos ao receber a resposta da requisição com dados do jogador em formato XML, como pode ser observado na Figura 3.

Ainda na Figura 3, pode ser observado que a cada 0.1 segundo (esse valor foi alterado posteriormente) são efetuadas duas requisições ao servidor PHP, onde a primeira tem como função atualizar os dados do jogador no banco de dados e a segunda tem como função requisitar os dados atualizados do mundo virtual.

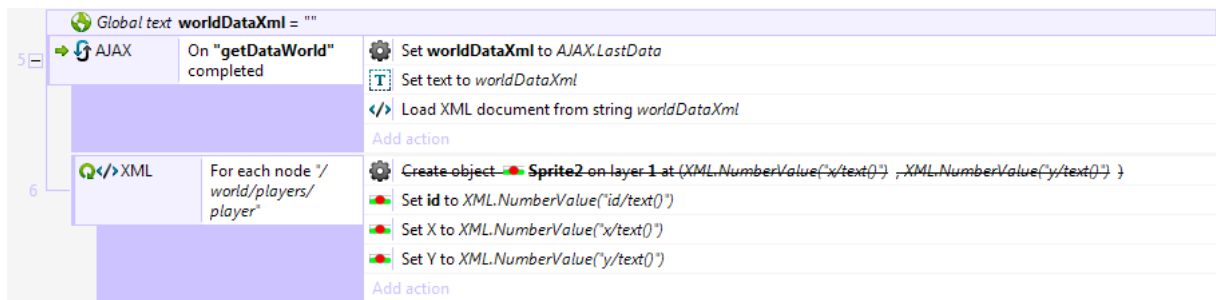
Figura 3 - Página de eventos da segunda tela do primeiro protótipo



Fonte: Autor

A segunda requisição tem como resposta os dados do mundo virtual em formato XML. É importante ressaltar que esses dados são consultados em um banco de dados quando o servidor PHP recebe a segunda requisição, após essa consulta os dados são organizados no formato XML. Quando o cliente recebe os dados do mundo virtual em XML, é efetuada a atualização de todos os elementos do jogo, incluindo as coordenadas X, Y dos outros jogadores, como pode ser verificado na Figura 4.

**Figura 4 - Página de eventos da segunda tela do primeiro protótipo**



**Fonte: Autor**

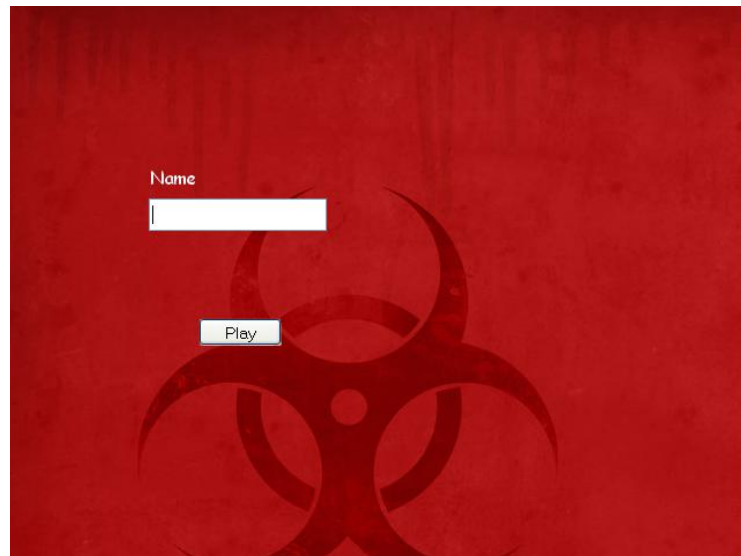
No primeiro protótipo foi implementado o sistema de login e o sistema de atualização dos jogadores. Os demais requisitos descritos no documento de game design não foram implementados, pois a abordagem deste protótipo não foi eficiente para os requisitos do protótipo, como será detalhado na seção 7.1.3.

### 7.1.2 Desenvolvimento do segundo protótipo

O segundo protótipo foi desenvolvido com uma outra abordagem, detalhada na seção 7.1.4, visto a ineficiência da abordagem adotada no protótipo anterior. A composição do segundo protótipo é a mesma, uma tela de login e um mapa do mundo virtual, porém na tela de login há somente um campo para inserir um nome para o jogador, como demonstrado na Figura 5, sendo que, após o nome ser inserido e o botão “Play” for pressionado, o jogador será redirecionado para um mapa do mundo virtual, conforme observado na Figura 6.

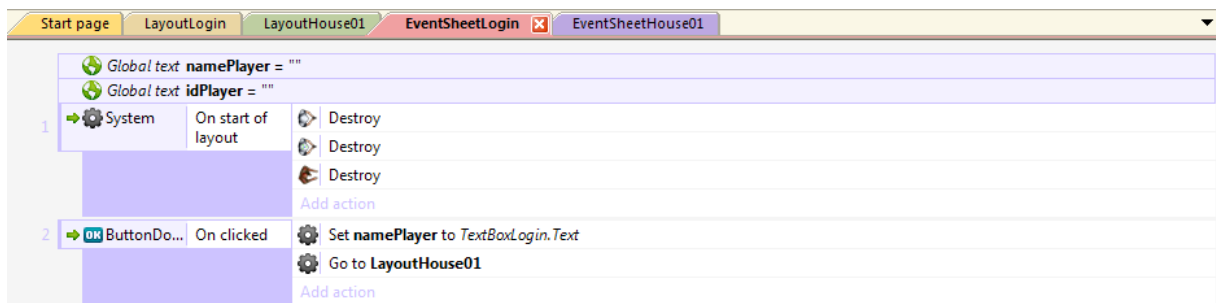
Nessa abordagem não foi adotado banco de dados, sendo assim o acesso as variáveis é controlado pelo Node.js.

**Figura 5 - Tela de login do segundo protótipo**



**Fonte: Autor**

**Figura 6 - Página de eventos da tela de login do segundo protótipo**

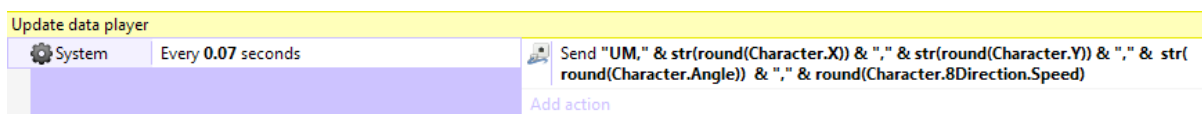


**Fonte: Autor**

Como citado anteriormente, a segunda tela é um mapa do mundo virtual, onde pode interagir com os demais jogadores. Assim que essa tela é carregada, é criada uma conexão com o servidor Node.js, o qual vai notificar todos os demais clientes conectados que há um novo jogador no mapa. Os clientes, ao receberem essa notificação, criam uma instancia desse jogador.

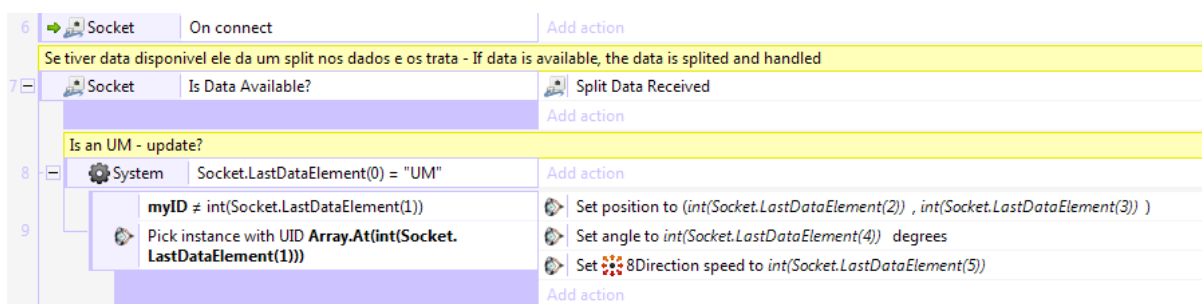
Nesta nova abordagem, o cliente informa ao servidor Node.js as coordenadas X, Y, o ângulo e a velocidade do jogador a cada 0.07 segundo, como pode ser observado na Figura 7. O servidor ao receber esses dados, atualizada eles na memória RAM e notifica dos demais clientes sobre essa atualização. Já os demais clientes ao receberem essa notificação, atualizam os dados do personagem, como demonstrado na Figura 8.

Figura 7 - Página de eventos do segundo protótipo, atualização dos dados do jogador



Fonte: Autor

Figura 8 - Página de eventos do segundo protótipo, atualização dos demais jogadores



Fonte: Autor

Neste ponto, o segundo protótipo já consegue permitir que vários jogadores compartilhem o mesmo mapa de maneira eficiente. Porém, no segundo protótipo, os demais requisitos, detalhados no documento de game design (Apêndice III), foram levados em consideração, sendo assim, foi implementado o sistema de *Chat*, que permite a interatividade entre os jogadores.

Também foi implementado o sistema de fome, sede e envenenamento, sendo que quando os pontos de vida de um jogador alcançam um valor menor que 1, o mesmo notifica o servidor Node.js e após isso o jogador é redirecionado para a tela de login, já o servidor, ao receber essa notificação, informa os demais clientes que a instância desse jogador deve ser destruída.

Outra funcionalidade implementada foi a adição dos recursos, assim o jogador pode encontrar e consumir recursos como água, comida e munição, além disso, foi implementado a função de disparo, assim o jogador pode efetuar disparos para reduzir os pontos de vida dos inimigos, sendo que, quando um disparo é efetuado, o cliente notifica o servidor, o qual notifica os demais clientes, que por sua vez, criam uma instância desse disparo.

Foi desenvolvida uma outra versão do cliente do jogo, essa versão é voltada para a pessoa responsável por controlar o mundo virtual, assim, foram acrescentadas algumas funcionalidades a mais, como, a função de enviar mensagens de texto para todos os jogadores

online e a função de gerar inimigos, como zumbis. Mais detalhes sobre esse cliente será apresentado na seção 7.1.4.

### 7.1.3 Primeira abordagem para tratar múltiplos jogadores simultâneos no protótipo

Visto que o Construct 2 não oferece suporte nativo à jogos MMO, primeiramente foi adotada a arquitetura cliente-servidor, pois, como citado anteriormente, é a mais simples para o desenvolvimento, sendo assim o controle e o estado do jogo fica dependente do servidor.

A arquitetura do primeiro protótipo é composta pelos clientes, que podem ser qualquer dispositivo com um navegador compatível com os requisitos do jogo e acesso à internet. Pelo servido PHP, que fica responsável por receber as solicitações dos clientes e responde-las, além de fazer a ligação entre cliente e banco de dados MySQL.

Para o cliente conectar no servidor, era necessário efetuar o login no jogo, enviando seu usuário e a senha, ao efetuar o *login*, os dados do personagem, armazenados no banco de dados eram acessados constantemente, como descrito a seguir.

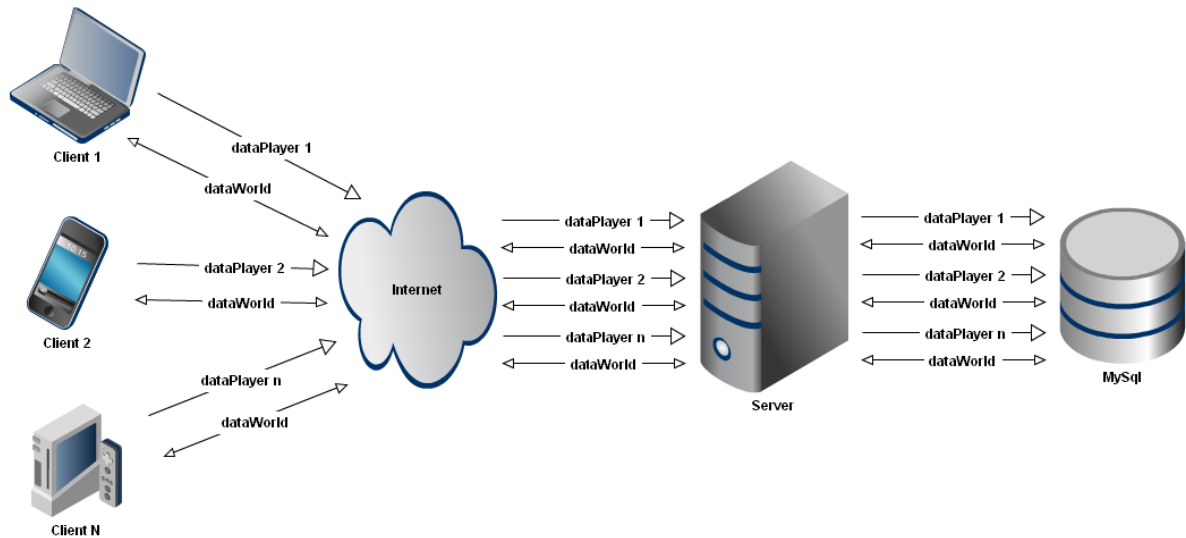
Nesta arquitetura o cliente envia duas requisições (*requests*) a cada 0.16 segundo, como pode ser observado na Figura 9, onde os clientes (*Client 1*, *Client 2*, *Client 3*) estão efetuando duas requisições, *dataPlayer X* e *dataWorld*, onde X é a referência ao cliente emissor da requisição.

A primeira requisição (*dataPlayer X*) tem como função atualizar os dados do jogador no banco de dados, esses dados são compostos pelos atributos do jogador, como as coordenadas x, y, ângulo, velocidade de movimentação, etc.

Já a segunda requisição (*dataWorld*) tem como função solicitar ao servidor todos os dados do mundo virtual, esses dados são compostos pelos dados dos outros jogadores que estão *online*, o processo da requisição *dataWorld* é descrita no próximo parágrafo, mas para prosseguir é necessário compreender a estrutura da arquitetura adotada, a qual pode ser observada na Figura 9.



**Figura 9 - Arquitetura do primeiro protótipo**



**Fonte: Autor**

O servidor que recebe as requisições ajax (*dataWorld*), contém um algoritmo escrito na linguagem PHP que gera e retorna um XML após uma consulta ao banco de dados, neste caso MySQL, todo esse processo exige um período significativo de tempo, como a aplicação é real-time, esse tempo de processo não poderia ser significativo, pois torna o jogo lento.

Além disso, utilizando essa arquitetura, o usuário percebe atrasos e anomalias (*lags*), essas causadas também pela falta de sincronização e controle das requisições, onde o primeiro que chegasse ao servidor seria o primeiro a ser respondido. Ou seja não é possível efetuar uma comunicação eficiente entre os clientes com essa abordagem.

A falta de sincronização poderia ser facilmente resolvida com algoritmos específicos para resolução desse tipo de problema, entretanto esses iriam aumentar o tempo de resposta entre o servidor e cliente, o que torna inviável implementar um sistema de sincronização.

Sendo assim, percebeu-se que tal arquitetura não é uma opção eficaz para o jogo proposto, pois não consegue atender os requisitos mínimos do mesmo, como descrito anteriormente. Portanto foi implementado um segundo protótipo com uma abordagem diferente a fim de resolver tais problemas.

#### **7.1.4 Segunda abordagem para tratar múltiplos jogadores simultâneos no protótipo**

Para o segundo protótipo foi adotada a arquitetura cliente-servidor, pois, como descrito anteriormente, é a mais simples para o desenvolvimento, sendo assim o controle e o

estado do jogo ficaria praticamente dependente do servidor, entretanto foi adotada uma abordagem orientada a eventos, sendo assim, é possível o cliente ter várias responsabilidades.

A arquitetura do segundo protótipo, demonstrada na Figura 10, é diferente da primeira em alguns aspectos, os quais serão abordados no decorrer desta seção. O único componente comum entre as arquiteturas do primeiro e do segundo protótipo são os clientes, que podem ser qualquer dispositivo com um navegador compatível com os requisitos do jogo e acesso a internet.

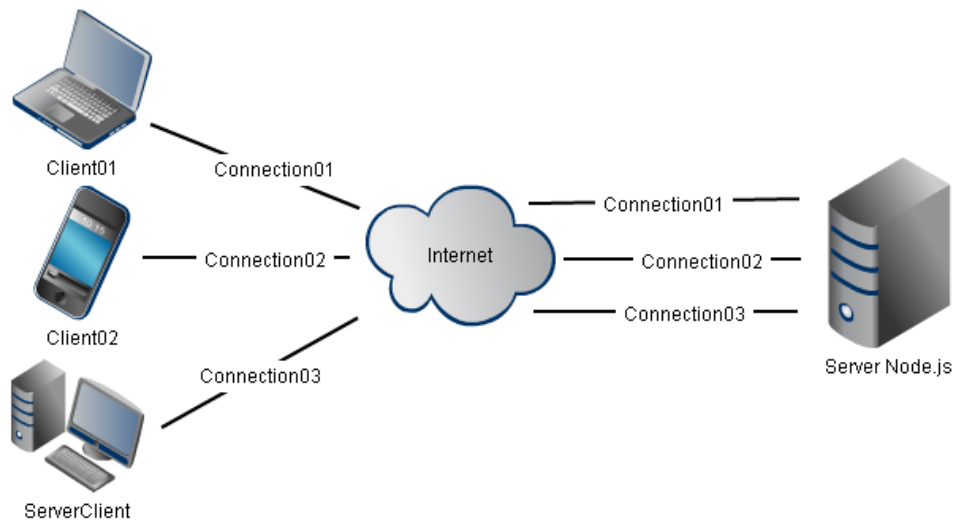
Já o servidor será Node.js, já descrito na seção 6.1.1.2, foi selecionado para minimizar os problemas relatados do primeiro protótipo referentes à linguagem de programação, a qual além de não ser facilmente escalável também consumia um número considerável de memória RAM.

Um dos problemas do primeiro protótipo era o tempo que o servidor demorava em responder uma requisição, isso ocorria, pois a linguagem processava apenas uma requisição por vez e ainda tinha de acessar o banco de dados. Esse problema foi resolvido não só pela adoção do Node.js, o qual tem uma conexão “permanente” com o cliente e permite o processo de várias requisições ao mesmo tempo, mas também pela maneira com que os dados são armazenados e manipulados.

Para o segundo protótipo, o banco de dados tradicional foi extinto e os dados são armazenados e manipulados diretamente na memória RAM do servidor, sendo assim, é possível acessar os dados da maneira mais rápida possível, ou seja, os dados não são persistidos.

Além disso, essa arquitetura conta com um cliente especial denominado *ServerClient*, como observado na Figura 10, o qual será responsável por gerenciar os dados do jogo e fazer a interface entre o jogo e o administrador do jogo (GM, do inglês *Game Manager*), assim o administrador do jogo poderá manipular os dados do jogo e interagir com os demais jogadores.

**Figura 10 - Arquitetura do segundo protótipo**

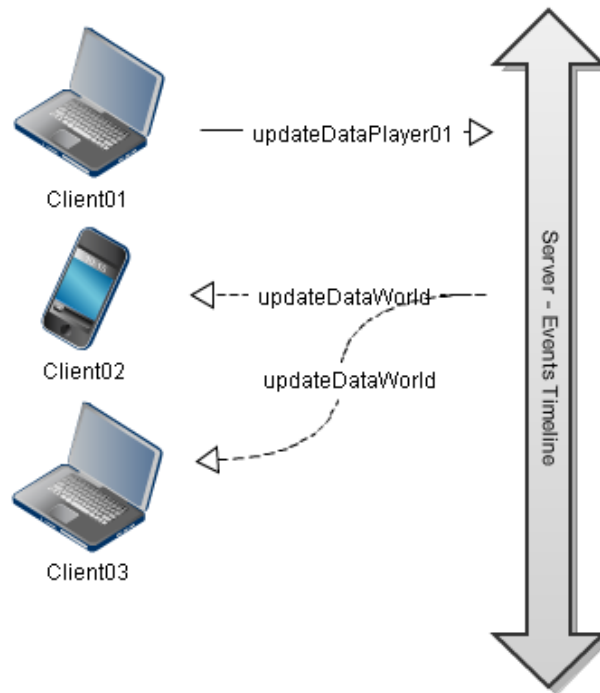


**Fonte: Autor**

Nessa nova abordagem o cliente pode efetuar quatro requisições ao servidor, entretanto, somente a primeira (*updateDataPlayerX*, onde *X* faz referência ao cliente), demonstrada na Figura 11, ocorre continuamente a cada 0.16 segundo. Nessa requisição o cliente (*Client*) envia ao servidor os dados do personagem, sendo esses dados compostos pelas coordenadas *X*, *Y* do jogador, o ângulo do *sprite* do jogador e a velocidade com que o mesmo estava a se deslocar.

Ao receber os dados da requisição *updateDataPlayerX*, o servidor os organiza e atualiza os dados do mundo virtual, então notifica os demais clientes, ao mesmo tempo, com os novos dados do mundo virtual (*UpdateDataWorld*).

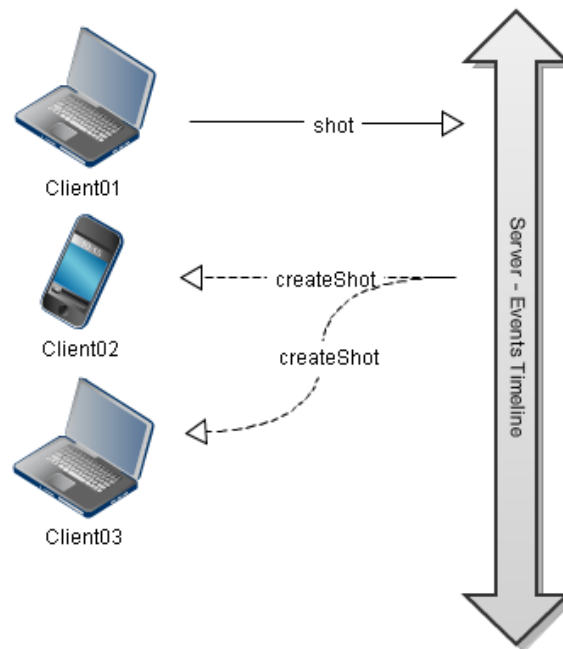
A primeira requisição também pode incluir o nome do mapa do jogador, mas esse dado só é repassado ao servidor quando o jogador efetuar *login* no sistema ou trocar de mapa, essa decisão foi tomada por questões de desempenho.

**Figura 11 - Demonstração da requisição updateDataPlayer**

**Fonte: Autor**

A segunda requisição (*Shot*) que pode ser usada pelo cliente, demonstrada na Figura 12, ocorre somente quando um jogador efetua um disparo, então são enviados ao servidor os dados do disparo, que são compostos pelas coordenadas X, Y do *sprite* do projétil e do ângulo e velocidade do mesmo. Ao receber esses dados o servidor notifica todos os jogadores, os quais estão no mesmo mapa em que o disparo foi efetuado, de uma só vez, conforme demonstrado na Figura 12.

**Figura 12 - Demonstração da requisição shot**



**Fonte: Autor**

A terceira requisição ocorre quando um jogador envia uma mensagem de texto aos demais jogadores, nesse caso o servidor recebe esses dados e efetua a validação dos mesmos, notificando os destinatários, conforme descrito na seção dedicada ao sistema de *CHAT* do Documento de *Game Design* (Apêndice III).

A quarta requisição ocorre quando um jogador morre, ou seja, quando os pontos de vida do jogador alcançam o valor mínimo, então o cliente deste jogador notifica o servidor, que notifica todos os demais jogadores, de forma semelhante a requisição *shot*. Deve-se ressaltar que o cliente responsável pelos inimigos é o *ClientServer*, e o mesmo ficará responsável em notificar o servidor quando algum inimigo morrer.

Essa arquitetura resolveu todos os problemas relatados do primeiro protótipo, isso pelo fato de não haver banco de dados tradicional e sim os dados estarem disponíveis na memória RAM do server, o que faz com que os dados sejam rapidamente acessados. Também pela maneira que a tecnologia Node.js, tratando cada cliente como uma conexão e manipulando várias requisições de forma “paralela” e pela programação orientada a eventos, que não sobrecarrega o servidor.

### 7.1.5 Avaliação das abordagens para tratar múltiplos jogadores simultâneos nos protótipos

Para afirmar que a estrutura e as tecnologias adotadas para desenvolver o segundo protótipos eram mais eficientes, foi necessário obter algumas métricas de ambas as soluções, assim foi possível comparar a solução do primeiro e do segundo protótipo. Para obter os valores utilizados na comparação, foi efetuada uma série de testes em um determinado contexto, os quais os resultados permitiram também avaliar a escalabilidade de cada arquitetura.

Havia dois momentos críticos no primeiro protótipo, o primeiro ocorria ao fazer a atualização dos dados do jogador e o segundo ao requisitar os dados do mundo virtual. Por isso os dados selecionados para avaliar as soluções foram: os tempos de requisição do cliente e resposta do servidor, além da taxa de erros por negação de serviço, pois esses fatores foram apontados como os principais causadores de problemas do primeiro protótipo e foram apontados como solucionados na segunda abordagem.

### 7.1.6 Ambientes de Testes

Para efetuar os testes, os protótipos tiveram de ser colocados em um ambiente de testes. Para o primeiro protótipo, foi necessário um servidor PHP e um servidor Mysql, sendo assim, foi implementado um ambiente em um servidor compartilhado<sup>38</sup> da empresa Dream Host.

O ambiente é composto por um servidor Apache que suporta a linguagem PHP 5 e um servidor MySQL 5.0.51<sup>a</sup>, os servidores estão em máquinas distintas, pois cada máquina é configurada para ter o melhor desempenho para cada tipo específico de serviço.

Neste caso o servidor PHP armazenava dois arquivos, sendo o arquivo *setDataByIdPlayer.php*, responsável por fazer a atualização dos dados do jogador no mundo virtual, para isso é executada uma consulta no banco de dados passando os dados atualizados do jogador. Essa requisição não teria uma resposta, mas para fins de teste, foi implementado uma resposta composta por um conjunto de seis caracteres.

Já o arquivo *getDataWorld.php*, responsável por gerar uma resposta na estrutura de XML com os dados obtidos através de uma consulta no banco de dados e enviar essa resposta em XML para o cliente.

---

<sup>38</sup> <http://dreamhost.com/web-hosting>

Para o segundo protótipo foi utilizado o ambiente de desenvolvimento Cloud 9 IDE<sup>39</sup>, o qual permite instanciar e executar aplicações Node.js remotamente. Nesse caso era executado um aplicativo Node.js tinha como função manipular os dados do mundo virtual que eram armazenados na memória RAM, além de receber e responder uma requisição específica.

Essa requisição tem como função solicitar ao servidor que os dados do jogador, nesse caso o emissor da requisição, sejam atualizados, e que após isso o servidor informe os demais clientes os dados atualizados do mundo virtual, função essa atribuído ao arquivo *setDataByIdPlayer.php* do ambiente de testes do primeiro protótipo.

O segundo protótipo, por ter uma abordagem orientada a eventos, não necessita que o cliente requisiute os dados do mundo virtual, visto que sempre que algum dado for alterado, o cliente será automaticamente notificado pelo servidor. Outro fato que deve ser ressaltado, é que a primeira requisição não teria uma resposta, mas para fins de teste, foi implementado uma resposta, composta por uma string de seis caracteres, para tal requisição.

### 7.1.7 Testes

Para fazer a simulação e obter os dados dos testes foi utilizado o software Jmeter<sup>40</sup>, que é uma aplicação Java desenvolvida para fazer testes de sobrecarga e mensurar desempenho de aplicações web, atualmente já tem suporte à mais tipos de aplicações.

Para que esses dados fossem mais próximos do ambiente de produção, foi montado dois planos de testes para cada protótipo, entretanto, é essencial ressaltar que o primeiro protótipo necessitava de duas requisições para ter seu funcionamento básico, já o segundo protótipo necessita somente de uma requisição.

O primeiro plano de teste (Plano de Teste A) simula quinze usuários fazendo duas requisições distintas ao servidor a cada 0.16 segundo cinquenta vezes consecutivas, isso para o primeiro protótipo. Já para o segundo protótipo, é simulado quinze usuários fazendo apenas uma requisição ao servidor a cada 0.16 segundo cinquenta vezes consecutivas.

O segundo plano de teste (Plano de Teste B) simula trinta usuários fazendo as mesmas duas requisições do Plano de Teste A ao servidor a cada 0.16 segundo por cinquenta vezes consecutivas, isso para o primeiro protótipo, já para o segundo protótipo, é simulado trinta

---

<sup>39</sup> <https://c9.io/>

<sup>40</sup> <http://jmeter.apache.org/>

usuários fazendo apenas uma requisição ao servidor a cada 0.16 segundo por cinquenta vezes consecutivas.

Cada plano de teste foi executado seis vezes para cada protótipo, sendo o maior valor e o menor valor dos dados obtidos é eliminado por questões de estatística. Os resultados obtidos seguem na Tabela 1 e na Tabela 2.

**Tabela 3 - Resultados Obtidos no Plano de Teste A**

<i>Protótipo</i>	<i>Request</i>	<i>Média de Tempo Mínimo de Resposta (ms)</i>	<i>Média de Tempo Máximo de Resposta (ms)</i>	<i>Média de Tempo de Resposta (ms)</i>	<i>Taxa de Erros por Negação de Serviço</i>
Primeiro Protótipo	setDataPlayer	211,6	4005,25	383,83	0%
Primeiro Protótipo	getDataWorld	212,3	3905,5	312,75	0%
Segundo Protótipo	setDataPlayer	129,5	495,75	210,25	0%

Fonte: Autor

**Tabela 4 - Resultados Obtidos no Plano de Teste B**

<i>Protótipo</i>	<i>Request</i>	<i>Média de Tempo Mínimo de Resposta (ms)</i>	<i>Média de Tempo Máximo de Resposta (ms)</i>	<i>Média de Tempo de Resposta (ms)</i>	<i>Taxa de Erros por Negação de Serviço</i>
Primeiro Protótipo	setDataPlayer	169	4565,66	453	41,8%
Primeiro Protótipo	getDataWorld	171,66	4895,33	373,66	20,5%
Segundo Protótipo	setDataPlayer	128,75	540,75	209,5	0%

Fonte: Autor



Como o ciclo básico de requisição e resposta do primeiro protótipo era composto por duas requisições, pode-se concluir que esse fato impactava de maneira significativa o desempenho do jogo.

Se tratando de um jogo em tempo real, a média de tempo máximo de resposta deveria ser inferior à 500 ms, sendo que valores acima de 700ms são considerados inaceitáveis para tal tipo de jogo, sendo assim, fica visível que uma das causas dos problemas do primeiro protótipo realmente era o tempo entre a requisição ao servidor e a resposta.

Também pode-se concluir que o primeiro protótipo não tem condições de suportar trinta usuários neste ambiente, visto o alto índice da taxa de erros obtidos, conforme exposto na Tabela 2, também é visível que em  $\frac{3}{4}$  dos planos de testes, os resultados das médias de tempo de resposta obtidos do primeiro protótipo, foi aproximadamente o dobro do dos resultados obtidos do segundo protótipo, o que confirma que a segunda solução proposta é mais eficiente.

O tempo ideal entre a requisição e a resposta é no máximo 250ms, sendo assim, pode-se afirmar que a solução adotada para o segundo protótipo é eficiente, além disso, com a abordagem orientada a eventos, o cliente não necessita mais efetuar duas requisições, mas apenas uma, o que impacta positivamente no desempenho do jogo.

## 8 CONSIDERAÇÕES FINAIS

Com o desenvolvimento deste trabalho, ficou claro que o desenvolvimento de jogos MMO de tempo real e multiplataformas não é trivial, é um processo que tem vários pontos críticos que impactam no desempenho do jogo. O difícil desenvolvimento provavelmente seja a principal razão de não haver muitos títulos desse tipo de jogo.

Ao término da implementação do primeiro protótipo, ficou claro que a linguagem PHP e o banco de dados MySQL não são eficientes para esse tipo de jogo, entretanto, foi comprovado que a tecnologia Node.js e o paradigma de programação orientada a eventos é eficiente para a implementação do protótipo e provavelmente para os demais jogos de navegadores MMO *real time* multiplataformas.

Também deve ser registrado que a abordagem para desenvolvimento do Documento de *Game Design*, recomendada por Rogers (2010), foi válida e eficiente, abordando todos os aspectos do jogo.

Um desafio para o futuro é melhorar o desempenho do jogo, uma abordagem para isso, em nível de servidor, seria utilizar um banco de dados não tradicional (NoSQL), além de adotar uma arquitetura de servidor espelhado por questões de escalabilidade e usar dois servidores distintos, um responsável pelo jogo e outro responsável somente pelo sistema de *chat*.

Uma melhoria em nível de cliente seria a utilização do recurso de “*family*” e funções, porém esses recursos só são oferecidos na versão paga do Construct 2, por esse fato que esses recursos não foram utilizados para o desenvolvimento dos protótipos.

Outro recurso a ser implementado é a possibilidade de o jogador configurar as teclas utilizadas no jogo, além de uma melhoria na compatibilidade com os diferentes dispositivos; Outra melhoria é a utilização do recurso “dicionário”, oferecido pela *engine*, que permite que o jogo tenha suporte a várias idiomas.

Após a implementação de todas essas funções, deveria ser feito um novo plano de teste para obter dados dos impactos causados pelas modificações efetuadas no cliente e no servidor.

## REFERÊNCIAS

- ABERNETHY, Michael. **JustwhatisNode.js?**. Abril/2011. Disponível em: <<http://www.ibm.com/developerworks/library/os-nodejs/index.html>> Acesso em: 10 out. 2012.
- BARTLE, Richard. **Hearts, Clubs, Diamonds, Spades: Players Who Suit MUDs**. 1996. Disponível em: <<http://www.mud.co.uk/richard/hcds.htm>> Acesso em: 20 jun. 2012.
- BERTHÊM, A. C. et al. **Desenvolvimento de jogos eletrônicos**. 2. ed. São Paulo: Novatec, 2007.
- BORGES, Deise Miranda BARREIRA, Rafael Gonçalves, SOUZA, Jackson Gomes de Souza. **Comportamento de personagens em jogos de computador**. Palmas: Centro Universitário Luterano de Palmas, 2009.
- CRAWFORD, C. **The Art of Computer Design**. 1982. Disponível em: <<http://www.vancouver.wsu.edu/fac/peabody/game-book/Coverpage.html>> Acesso em: 20 ago. 2007.
- CRONIN , Eric; FILSTRUP, Burton; KURC , Anthony. Electrical Engineering and Computer Science Department. Maio/2001. Disponível em: <<http://warriors.eecs.umich.edu/games/papers/quakefinal.pdf>> Acesso em: 31 out. 2012.
- CRONIN, Eric; FILSTRUP, Burton; JAMIN, Sugih; KURC, Anthony R. **An Efficient Synchronization Mechanism for Mirrored Game Architectures**. 2004. Disponível em: <<http://hdl.handle.net/2027.42/47312>> Acesso em: 15 out. 2012.
- Apache JMeter™. 2000. Disponível em: < <http://jmeter.apache.org/>> Acesso em: 23 nov. 2012.
- JUNIOR, Francisco de Assis Ribeiro. **Programação Orientada a Eventos no lado do servidor utilizando Node.js**. 2012. Disponível em: <[http://www.infobrasil.inf.br/userfiles/16-S3-3-97136-Programa%C3%A7%C3%A3o%20Orientada\\_\\_\\_\\_.pdf](http://www.infobrasil.inf.br/userfiles/16-S3-3-97136-Programa%C3%A7%C3%A3o%20Orientada____.pdf)> Acesso em: 15 out. 2012.
- KERR, A. **The Business and Culture of Digital Games**. Sage Publications, London. 2006.
- Kleina, Nilton. O que é engine ou motor gráfico?. 2011 Disponível em: <<http://www.tecmundo.com.br/video-game/9263-o-que-e-engine-ou-motor-grafico-.htm#ixzz2DI5HI7Ze>> Acesso em: 15 out. 2012.
- KLUG, G.C. & SCHELL, J. (2006). **Why People Play Games: An Industry Perspective**. In Varderer, P. & Bryant, J. (Eds.), *Playing Video Games: Motives, Responses, and Consequences* (pp.91-100). Nahwah, NJ: Lawrence Erlbaum Associates.
- LOPES, Gilliard. **Jogos eletrônicos: conceitos gerais**. 2006. Disponível em: <[http://www-usr.inf.ufsm.br/~pozzer/disciplinas/cga\\_8\\_classificacao\\_jogos.pdf](http://www-usr.inf.ufsm.br/~pozzer/disciplinas/cga_8_classificacao_jogos.pdf)> Acesso em: 20 jun. 2012.

**Negócios divertidos: A rentável indústria dos jogos eletrônicos.** Disponível em: <[http://www.sebraepr.com.br/portal/page/portal/PORTAL\\_INTERNET/BEMPR\\_INDEX/BEMPR\\_ARTIGO?\\_dad=portal&\\_boletim=7&\\_filtro=239&\\_artigo=4095](http://www.sebraepr.com.br/portal/page/portal/PORTAL_INTERNET/BEMPR_INDEX/BEMPR_ARTIGO?_dad=portal&_boletim=7&_filtro=239&_artigo=4095)>. Acesso em 12 jun. 2012.

ROGERS, Scott. **Level Up!: The Guide to Great Video Game Design.** Chichester: John Wiley & Sons, 2010.

SALEN, Katie, ZIMMERMAN, Eric, 2004. **Rules of Play - Game design fundamentals.** Cambridge: The MIT Press.

SCHEIB, Vicent. **Super Happy Modern HTML5 Browser Games.** Palestra no Casual Connect Europe 2012. Hamburg, 2012.

SLOPER, T. **Following Up After the Game is Released: It's not Over when it's Over.** Game Design Perspectives. 2002.

SOLLITTO, André. **A maior diversão da terra.** Disponível em: <<http://revistaepoca.globo.com/ideias/noticia/2012/02/maior-diversao-da-terra.html>> Acesso em: 20 jun. 2012.

Tilkov, S, S. “**Node.js: Using Javascript to Build Gugh Performance Network Programs**”. Internet Computing IEEE, 2010.

**TRAFFIC and Market Report: On the pulse of the networked society.** 2012 (Ericson). Disponível em: <[http://www.ericsson.com/res/docs/2012/traffic\\_and\\_market\\_report\\_june\\_2012.pdf](http://www.ericsson.com/res/docs/2012/traffic_and_market_report_june_2012.pdf)> Acesso em: 15 jun. 2012.

THOMAS, David; ORLAND, Kyle; Steinberg, Scott. **The Videogame Style Guide and Reference Manual.** Power Play Publishing, 2007.

VANNUCCHI, Hélia; PRADO, Gilberto. **Discutindo o conceito de gameplay.** In: <Revista Texto Digital>. ISSN 1807-9288 - ano 5 n.2 2009. Disponível em: <<http://www.textodigital.ufsc.br/num09/heliagilbertto.htm>> Acesso em: dia mês 2012.

ZYP, Kris. 2010. **Multi-Node.** Disponível em <<http://github.com/kriszyp/multi-node>>. Acesso em: 31 out. 2012.

## APÊNDICES

### APÊNDICE I

#### **The Last Infection**

Assustador jogo MMO multiplataforma de tiro e sobrevivência.

**Plataforma:** Web Browser.

**Público Alvo:** Maiores de 16 anos que gostem de jogos de tiro com temática zumbi.

**Idade Indicada:** Maiores de 16 anos.

#### **Resumo da história do jogo:**

A companhia Z-index fazia pesquisas na linha de melhoramento genético, entretanto, ocorreu um acidente em um de seus laboratórios, um vírus causou infecção em uma sala do centro de pesquisa, entretanto o esforço para impedir disseminação do vírus foi em vão e o vírus está a se espalhar por todos os andares, você era um simples funcionário e tem como missão sair do centro de pesquisa e procurar ajuda, entretanto, ao sair do laboratório você descobrirá algo terrível.

As pessoas começaram a ser infectadas com o vírus que vazou dos laboratórios da Z-index, o mundo está em caos, há escassez de recursos básicos como comida e água potável, há vários saqueadores e assassinos escondidos pela cidade, esperando uma oportunidade de atacar.

#### **Esboço do jogo:**

O jogo será de tiro, sobrevivência e horror, com dimensionalidade 2D e em tempo real, onde o principal objetivo do jogo é sobreviver, para isso o jogador deve ficar atento à quantidade de armas e munição, além da sede e fome do personagem. O personagem terá uma mochila onde poderá guardar alguns itens, porém o espaço é limitado e o personagem só consegue carregar uma quantidade pré definida de quilogramas.

Os zumbis estão espalhados por todos os lugares e são capazes de ver e escutar, então qualquer barulho feito pelo usuário poderá chamar a atenção deles. No decorrer do jogo o personagem encontrará indícios que há uma vacina e um lugar seguro para ir, sendo a principal missão chegar até este local.

**Extras:**

- Mais de 3 tipos de armas;
- Mais de 4 tipos de inimigos;
- Atualizações semanais com melhorias e novos recursos no jogo.

**Jogos similares e concorrência:**

- World of the living dead - Jogo MMORPG modo texto;
- Dayz - Jogo MMO em primeira pessoa.

APÊNDICE II



# The Last Infection

Design by Rômulo Reis de Oliveira

On your Browser

ESRB Rating: Mature +17

**Blood and Gore - Fantasy Violence - Use of Drugs**

## História do Jogo:

A companhia Z-index fazia pesquisas na linha de melhoramento genético, certo dia, ocorreu um acidente em um laboratório, um vírus saiu do controle, o esforço para impedir disseminação do vírus foi em vão e o vírus está se espalhando por todos os andares.

Você é um simples funcionário e tem como missão sair do centro de pesquisa e procurar ajuda, apenas portando uma arma para se defender, entretanto, ao sair do laboratório você descobrirá algo terrível. As pessoas começaram a ser infectadas com o vírus que vazou dos laboratórios da Z-index.

Agora o mundo está em caos, há escassez de recursos básicos como comida e água potável, além disso, há vários saqueadores e assassinos escondidos pela cidade, esperando uma oportunidade de atacar.

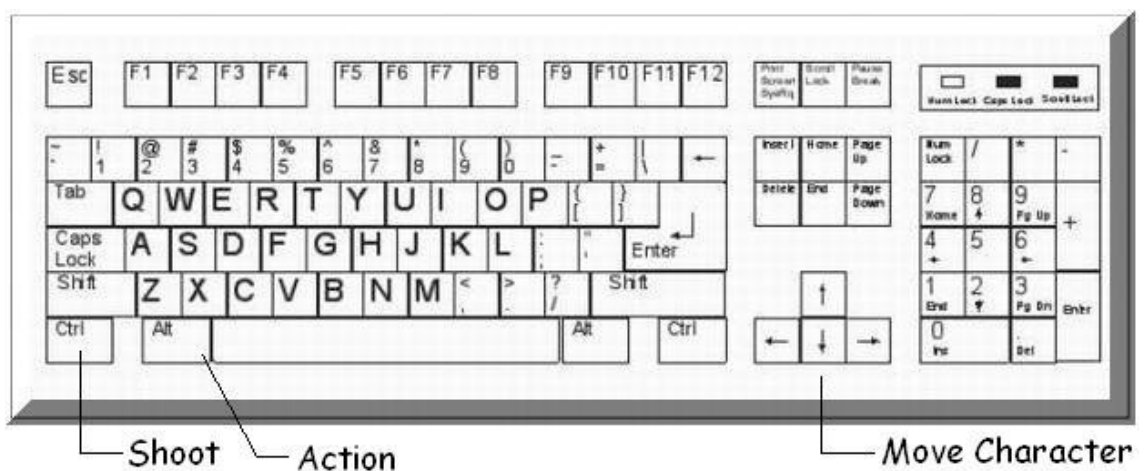
## Personagem:

O personagem era uma pessoa normal, sendo somente mais um funcionário da Z-index, que estava no local errado na hora errada, pois ele estava no local e na hora em que ocorreu o acidente com o vírus.

Por ser uma pessoa normal ele não tem grande resistência a principio, mas ao ser inserido neste ambiente de caos o personagem acaba evoluindo através de missões, no decorrer do jogo, então o jogador pode aumentar os atributos do personagem, além disso, o personagem também pode traçar seu perfil social, se tornando um saqueador ou um herói.

A interface com o jogador será a principio através do teclado, conforme ilustra a Figura 1, entretanto, haverá uma interface para telas touch screen nas versões futuras do jogo.

Figura 1 – Interface com usuário através do teclado





**Gameplay:**

O jogador deve sobreviver a ondas de zumbis e prosseguir até sair do laboratório, o qual é um labirinto cheio de quebra cabeças; Para se defender dos inimigos o jogador contará com diversos tipos de armas, mas, recursos como munição, água e comida são escassos.

Para piorar a situação, os zumbis tem ótimos ouvidos, sempre que o jogador emitir algum tipo de som, os zumbis mais próximos se encaminharão para o encontro do mesmo.

O jogador consegue carregar um número limitado de itens, baseado no peso máximo que ele consegue suportar e o número de itens em seu inventário, para aumentar a capacidade é necessário usar uma mochila, que possibilita que o jogador armazene mais itens em seu inventário, mas torna a velocidade de movimentação mais lenta, cabe ao jogador decidir qual a melhor escolha.

Sempre que um zumbi ferir um jogador, o jogador será infectado, isso fará com que ele perca pontos de vida a cada período de tempo, além disso, independente de estar infectado ou não, os pontos de fome e sede também são reduzidos a cada período de tempo.

A fome e a sede do personagem funcionam de maneira semelhante ao envenenamento. Ou seja, quando o valor da fome ou sede alcançar o valor mínimo, o personagem vai perder pontos de vida a cada período de tempo.

**Mundo do Jogo:**

O jogador começa em um laboratório, mas ao sair do laboratório tem um mundo inteiro para explorar, isso inclui campos com fazendas, grandes cidades com vários tipos de construções, as quais podem ser exploradas internamente para encontrar recursos escondidos.

O mundo também tem seu próprio clima, onde dias ensolarados são os preferidos dos zumbis, pois assim como o jogador, o campo de visão é aumentado, entretanto os zumbis ficam tontos quando o clima é de chuva e extremamente lentos quando há nevascas.

O jogador também sente os efeitos do clima, em dias quentes é muito mais fácil ficar sem água, já quando chove, é difícil escutar a aproximação dos zumbis, mas talvez o pior seja quando ocorrem as nevascas, que podem congelar o jogador.

**Experiência de Jogo:**

Todos os elementos gráficos do jogo, juntamente com os efeitos sonoros, fazem o jogo ter um ar de suspense e terror, e a emoção do usuário fica mais intensa a cada mapa explorado, pois ele não sabe o que o aguarda no próximo mapa.

O Jogo permite que haja interação com os outros jogadores via texto, assim o jogador pode moldar o perfil do personagem criado e passar a impressão desejada para os outros jogadores, que podem a vir se tornarem rivais ou aliados no jogo, sendo que quando o personagem é aliado de alguém, ele ajuda os outros personagens a progredirem no jogo, afinal, quanto mais jogadores colaborando, mais fácil é de se movimentar pelos mapas lotados de inimigos.

Sempre que o jogador quiser, pode destruir os outros jogadores e ficar com os itens que o mesmo carregava, pois quando um jogador quando é derrotado por outros jogadores os itens são transferidos para o seu algoz.

O maior desafio do jogo é sobreviver pelo maior período de tempo possível, sendo que para isso é necessário explorar os mapas atrás de recursos básicos que estão escondidos nos mais diversos lugares e muitas vezes o jogador tem de optar entre destruir um companheiro para pegar seus recursos ou arriscar perder o jogo por desidratação, fome ou falta de munição.

### **Mecânica do Jogo:**

O jogo é em 2D, sendo que a câmera tem uma visão de cima para baixo, os mapas não podem ser rotacionados ou aproximados e o personagem compartilha um único mundo virtual com outros jogadores, sendo que os mesmo podem infligir danos nos inimigos através de armas, onde cada tipo de arma tem um valor de dano diferente.

Os inimigos causam dano ao encostar no personagem, alguns tipos de inimigos causam danos maiores, além disso eles podem causar envenenamento ao atingirem o personagem, o qual começa a perder pontos de vida a cada período de tempo. Para se recuperar de um envenenamento o jogador deve usar um item de medicação, já para recuperar os pontos de vida, ele deve usar itens de cura.




Quando os pontos de Fome ou Sede do personagem atingem o valor mínimo, o personagem começa a perder pontos de vida a cada período de tempo, até que o jogador use itens do tipo bebida para aumentar os pontos de sede, ou itens do tipo comida, para recuperar os pontos de fome, entretanto, se os pontos de vida do jogador atingirem o valor mínimo, o mesmo perde o jogo, voltando ao último ponto salvo.

A velocidade do jogador é afetada pelo peso dos itens no inventário, quanto mais peso o personagem carrega, mais lento o mesmo vai ficar e ao atingir o peso máximo suportado pelo jogador, o jogador ficará impossibilitado de se movimentar.

### Inimigos:

Os inimigos serão gerados em pontos aleatórios de mapas a cada período de tempo, os mapas e o período de tempo serão pré determinados conforme o mapa e tipo do inimigo, além disso, cada inimigo tem características e habilidades únicas como é descrito na Tabela 1.

Tabela 1 – Lista de Inimigos

Zumbi	HP: 3	Maps: B2Lab2Under0, B2Lab3Under03, B2Lab4Under03, B2Lab5Under03.
	Power: 2	
	Speed: 3	
	Time: 70000	
Zumbi Dilacerado	HP: 3	Maps: B2Lab3Under03, B2Lab4Under03, B1Lab3Under02.
	Power: 1	
	Speed: 1	
	Time: 180000	
Cão Zumbi	HP: 4	Maps: B2Lab1Under01, B2Lab2Under01, B1Lab3Under02, B1Hall3Under02.
	Power: 3	
	Speed: 3	
	Time: 250000	

Outro ponto importante a ser citado é que os inimigos podem enxergar e ouvir, ou seja, sempre que o personagem entrar na zona de visão do inimigo, o inimigo vai ter como alvo esse jogador. Sempre que o jogador emite algum tipo de som, como por exemplo, o som

emitido ao efetuar um disparo, o inimigo que está perto do jogador também vai ter o jogador como alvo.

Os inimigos também são afetados pelo clima do jogo, quando o ambiente é claro, os zumbis tem seu campo de visão aumentado, já em ambientes escuros o campo de visão é menor. Quando está chovendo, os inimigos têm seu campo auditivo diminuído por causa do som da chuva, já quando está nevando, eles se tornam mais lentos.

**Cenas:**

Não haverá cenas no jogo, porém haverá algumas narrativas e diálogos para ambientar o jogador e engaja-lo no jogo.

**Materiais Bônus:**

Além de freqüentes atualizações, onde serão inseridos novos mapas, inimigos, itens e correções de possíveis erros, também será possível adquirir itens e customizações no jogo através da compra de créditos para o jogo.

APÊNDICE III



# The Last Infection

Escrito por: Rômulo Reis de Oliveira

Versão: 0.9

Data de publicação: 03/10/2012

**História do Jogo:**

A companhia Z-index fazia pesquisas na linha de melhoramento genético, certo dia, ocorreu um acidente em um laboratório, um vírus saiu do controle, o esforço para impedir disseminação do vírus foi em vão e o vírus está se espalhando por todos os andares.

Você é um simples funcionário e tem como missão sair do centro de pesquisa e procurar ajuda, apenas portando uma arma para se defender, entretanto, ao sair do laboratório você descobrirá algo terrível. As pessoas começaram a ser infectadas com o vírus que vazou dos laboratórios da Z-index.

Agora o mundo está em caos, há escassez de recursos básicos como comida e água potável, além disso, há vários saqueadores e assassinos escondidos pela cidade, esperando uma oportunidade de atacar.

**Recursos do Jogo:**

Jogo MMO de tiro e sobrevivência em tempo real.

Pode ser jogado em qualquer dispositivo que tenha um navegador web com suporte a HTML 5.

Batalhas alucinantes contra ondas de zumbis.

Você constrói o perfil do personagem.

**Mundo do Jogo:**

O jogador começa em um laboratório, mas ao sair do laboratório tem um mundo inteiro para explorar, isso inclui campos com fazendas, grandes cidades com vários tipos de construções, as quais podem ser exploradas internamente para encontrar recursos escondidos.

O mundo também tem seu próprio clima, onde dias ensolarados são os preferidos dos zumbis, pois assim como o jogador, o campo de visão é aumentado, entretanto os zumbis ficam tontos quando o clima é de chuva e extremamente lentos quando há nevascas.

O jogador também sente os efeitos do clima, em dias quentes é muito mais fácil ficar sem água, já quando chove, é difícil escutar a aproximação dos zumbis, mas talvez o pior seja quando ocorrem as nevascas, que podem congelar o jogador.

**Experiência de Jogo:**

Todos os elementos gráficos do jogo, juntamente com os efeitos sonoros, fazem o jogo ter um ar de suspense e terror, e a emoção do usuário fica mais intensa a cada mapa explorado, pois ele não sabe o que o aguarda no próximo mapa.

O Jogo permite que haja interação com os outros jogadores via texto, assim o jogador pode moldar o perfil do personagem criado e passar a impressão desejada para os outros jogadores, que podem a vir se tornarem rivais ou aliados no jogo, sendo que quando o personagem é aliado de alguém, ele ajuda os outros personagens a progredirem no jogo, afinal, quanto mais jogadores colaborando, mais fácil é de se movimentar pelos mapas lotados de inimigos.

Sempre que o jogador quiser, pode destruir os outros jogadores e ficar com os itens que o mesmo carregava, pois quando um jogador quando é derrotado por outros jogadores os itens são transferidos para o seu algoz.

O maior desafio do jogo é sobreviver pelo maior período de tempo possível, sendo que para isso é necessário explorar os mapas atrás de recursos básicos que estão escondidos nos mais diversos lugares e muitas vezes o jogador tem de optar entre destruir um companheiro para pegar seus recursos ou arriscar perder o jogo por desidratação, fome ou falta de munição.

### **Mecânica do Jogo:**

O jogo é em 2D, sendo que a câmera tem uma visão de cima para baixo, os mapas não podem ser rotacionados ou aproximados e o personagem compartilha um único mundo virtual com outros jogadores, sendo que os mesmo podem infligir danos nos inimigos através de armas, onde cada tipo de arma tem um valor de dano diferente.

Os inimigos causam dano ao encostar no personagem, alguns tipos de inimigos causam danos maiores, além disso eles podem causar envenenamento ao atingirem o personagem, o qual começa a perder pontos de vida a cada período de tempo. Para se recuperar de um envenenamento o jogador deve usar um item de medicação, já para recuperar os pontos de vida, ele deve usar itens de cura.

Quando os pontos de Fome ou Sede do personagem atingem o valor mínimo, o personagem começa a perder pontos de vida a cada período de tempo, até que o jogador use itens do tipo bebida para aumentar os pontos de sede, ou itens do tipo comida, para recuperar os pontos de fome, entretanto, se os pontos de vida do jogador atingirem o valor mínimo, o mesmo perde o jogo, voltando ao último ponto salvo.

A velocidade do jogador é afetada pelo peso dos itens no inventário, quanto mais peso o personagem carrega, mais lento o mesmo vai ficar e ao atingir o peso máximo suportado pelo jogador, o jogador ficará impossibilitado de se movimentar.

### **Gameplay Geral:**

O jogador deve sobreviver a ondas de zumbis e prosseguir até sair do laboratório, o qual é um labirinto cheio de quebra cabeças; Para se defender dos inimigos o jogador contará com diversos tipos de armas, mas, recursos como munição, água e comida são escassos.

Para piorar a situação, os zumbis tem ótimos ouvidos, sempre que o jogador emitir algum tipo de som, os zumbis mais próximos se encaminharão para o encontro do mesmo.

O jogador consegue carregar um número limitado de itens, baseado no peso máximo que ele consegue suportar e o número de itens em seu inventário, para aumentar a capacidade é necessário usar uma mochila, que possibilita que o jogador armazene mais itens em seu inventário, mas torna a velocidade de movimentação mais lenta, cabe ao jogador decidir qual a melhor escolha.

Sempre que um zumbi ferir um jogador, o jogador será infectado, isso fará com que ele perca pontos de vida a cada período de tempo, além disso, independente de estar infectado ou não, os pontos de fome e sede também são reduzidos a cada período de tempo.

A fome e a sede do personagem funcionam de maneira semelhante ao envenenamento. Ou seja, quando o valor da fome ou sede alcançar o valor mínimo, o personagem vai perder pontos de vida a cada período de tempo.

### **Gameplay Detalhado:**

#### **Interação:**

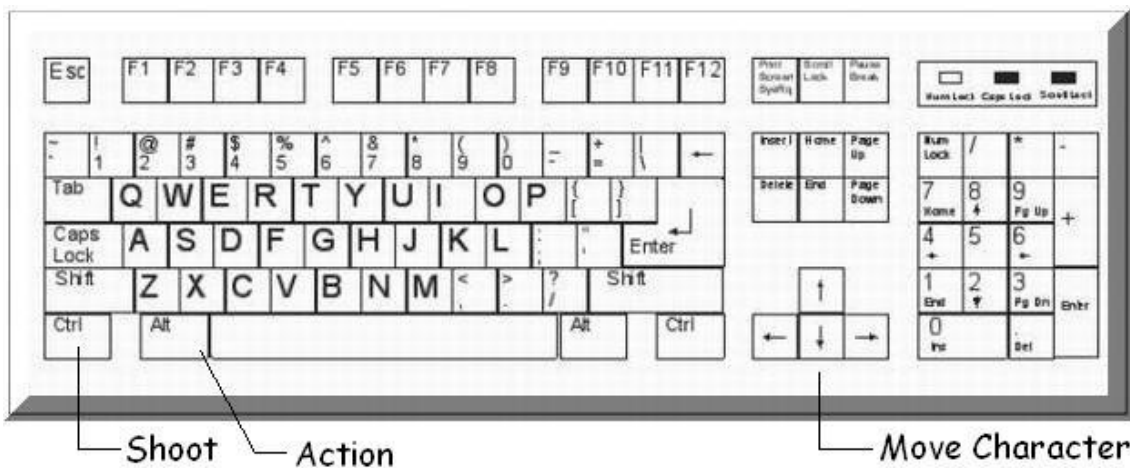
Para efetuar alguma acção, como pegar item, usar item, interagir com um objeto, utiliza-se o botão “ALT” do teclado.

#### **Batalha:**

Para efetuar disparos utiliza-se o botão “CTRL” do teclado, lembrando que cada disparo tem um delay, e o disparo só é efetuado se o jogador estiver equipado com uma arma e tiver munição suficiente.



Figura 1 – Interface com usuário através do teclado



### Fome e Sede:

Todos os personagens iniciam com 10 pontos de fome e sede, a cada 1 minuto esses pontos serão decrescidos. Ao atingir o valor mínimo, o jogador começa a perder 1 ponto de vida a cada 1 minuto.

### Envenenamento:

Quando o jogador estiver envenenado, ele perderá 2 pontos de vida por minuto.

### Pontos de salvação:

O estado do jogador só será salvo automaticamente quando ele chegar em determinados mapas especiais, ao morrer o jogador será transportado para esse mapa.

### Danos:

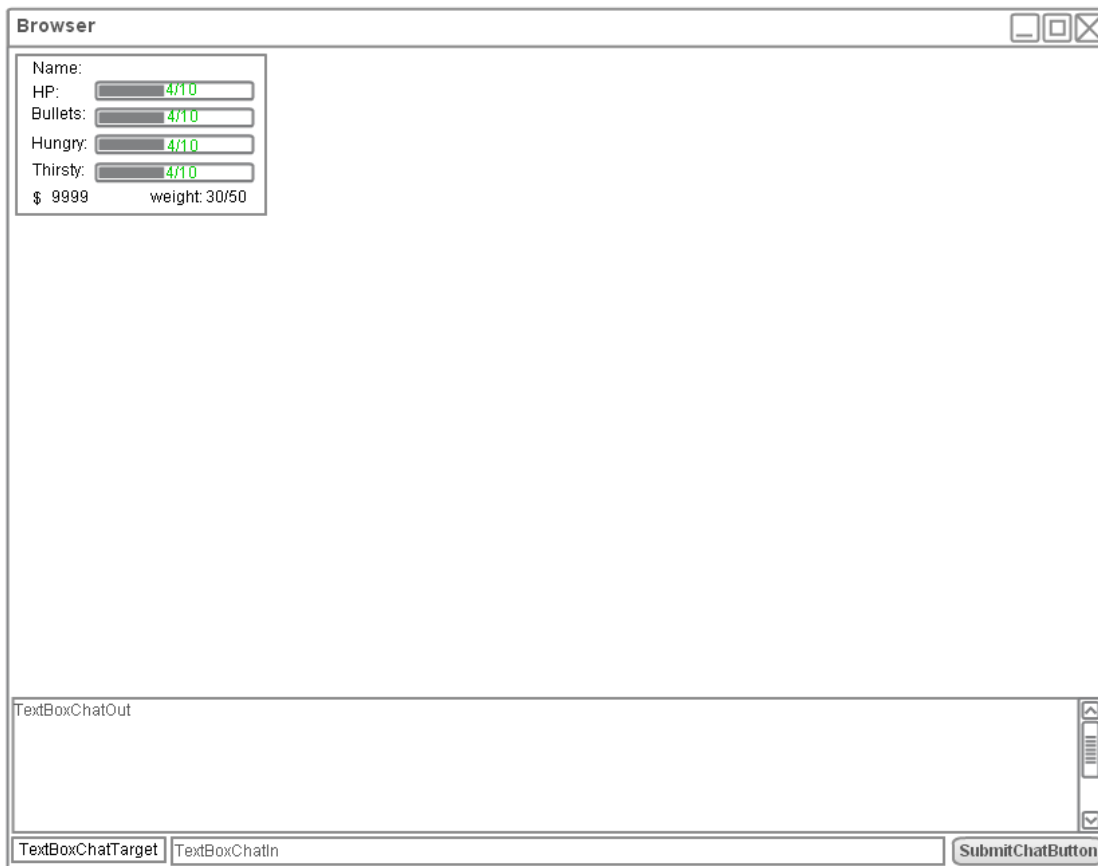
O personagem é ferido ao ser encostado por zumbis, ou ao colidir com objetos do tipo Bullet, sendo que o dano pode variar dependendo do poder do inimigo ou da munição.

Os Zumbis só são feridos ao colidirem com objetos do tipo Bullet.

### Chat

### Interface com o usuário:

O chat será composto por quatro itens gráficos, sendo eles, uma caixa de texto, onde serão apresentadas todas as mensagens enviadas pelos usuários, respeitando as regras de destinatário que serão descritas a seguir, uma caixa de texto, que contempla 60 caracteres, para o usuário inserir a mensagem a ser enviada e uma caixa de texto, que contempla 15 caracteres, para o usuário informar o destinatário da mensagem, caso deseje enviar a mensagem para um jogador específico, além disso haverá um botão que submete as informações ao servidor.



### Tipos de mensagens do chat:

- Pública: Esse tipo de mensagem é enviado para todos os usuários que estão no campo de visão do usuário que enviou a mensagem.
- Global: As mensagens globais são exibidas para todos os usuários on-line.
- Privada: Essa mensagem deve ter obrigatoriamente um destinatário, que será o único a receber a mensagem.

### Comportamento e funcionalidades do chat:

- Sempre que o usuário tentar enviar uma mensagem e o campo da mensagem estiver vazia, o envio será ignorado.
- Sempre que o usuário enviar uma mensagem sem especificar um usuário destino, a mensagem será considerada pública.

- Sempre que o destinatário for informado, a mensagem será considerada privada, caso o usuário de destino não esteja on-line, será enviado uma mensagem informativa ao usuário que enviou a mensagem.

## Comunicação

O servidor node.js estará escutando o cliente, aguardando as requisições, quando a mensagem for relacionada ao chat a mensagem do cliente para o servidor terá a seguinte estrutura:

*Identificador, Identificador do Tipo de mensagem, nome do usuário que enviou a mensagem, mensagem, nome do destinatário, nome do mapa*


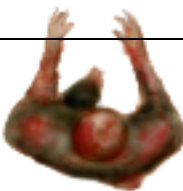
Onde o Identificador será a letra 'C', referente a palavra "CHAT", seguido pelo tipo da mensagem enviada, que deve respeitar a tabela a seguir:


Identificador do Tipo de Mensagem	Tipo de Mensagem
CPUBLIC	Mensagem Pública
CGLOBAL	Mensagem Global
CPRIVATE	Mensagem Privada

## Inimigos:

Os inimigos serão gerados em pontos aleatórios de mapas a cada período de tempo, os mapas e o período de tempo serão pré determinados conforme o mapa e tipo do inimigo, além disso, cada inimigo tem características e habilidades únicas como é descrito na Tabela 1.

Tabela 1 – Lista de Inimigos

Zumbi	HP: 3	Maps: B2Lab2Under0, B2Lab3Under03, B2Lab4Under03, B2Lab5Under03.
	Power: 2	
	Speed: 3	
	Time: 70000	
Zumbi Dilacerado	HP: 3	Maps: B2Lab3Under03, B2Lab4Under03,
	Power: 1	

	Speed: 1	B1Lab3Under02.
	Time: 180000	
Cão Zumbi	HP: 4	Maps: B2Lab1Under01, B2Lab2Under01, B1Lab3Under02, B1Hall3Under02.
	Power: 3	
	Speed: 3	
	Time: 250000	

Outro ponto importante a ser citado é que os inimigos podem enxergar e ouvir, ou seja, sempre que o personagem entrar na zona de visão do inimigo, o inimigo vai ter como alvo esse jogador. Sempre que o jogador emite algum tipo de som, como por exemplo, o som emitido ao efetuar um disparo, o inimigo que está perto do jogador também vai ter o jogador como alvo.

Os inimigos também são afetados pelo clima do jogo, quando o ambiente é claro, os zumbis tem seu campo de visão aumentado, já em ambientes escuros o campo de visão é menor. Quando está chovendo, os inimigos têm seu campo auditivo diminuído por causa do som da chuva, já quando está nevando, eles se tornam mais lentos.

### **Personagem:**

O personagem era uma pessoa normal, sendo somente mais um funcionário da Z-index, que estava no local errado na hora errada, pois ele estava no local e na hora em que ocorreu o acidente com o vírus.

Por ser uma pessoa normal ele não tem grande resistência a principio, mas ao ser inserido neste ambiente de caos o personagem acaba evoluindo através de missões, no decorrer do jogo, então o jogador pode aumentar os atributos do personagem, além disso, o personagem também pode traçar seu perfil social, se tornando um saqueador ou um herói.

O personagem terá os seguintes atributos visíveis:

- HP (Pontos de vida)

Ao zerar os pontos de vida o jogador morre.

- Bullets (Munição)

Necessário para efetuar os disparos, quando acaba a munição não é mais possível efetuar disparos. Para recuperar os pontos de munição basta usar o item “munição”

- Hungry (Fome), Thirsty (Sede)  
Todos os personagens iniciam com 10 pontos de fome e sede, a cada 1 minuto esses pontos serão decrescidos. Ao atingir o valor mínimo, o jogador começa a perder 1 ponto de vida a cada 1 minuto.
- Money (Dinheiro):  
Será possível usar valores para adquirir itens no jogo, esses valores serão obtidos pela venda de itens ou derrubados por inimigos.
- Weight (Peso):  
É o peso máximo que o jogador suporta, ao alcançar o valor máximo o jogador não poderá mais se movimentar, até que diminua o valor.

O personagem terá atributos não visíveis, como as coordenadas x, y e o ângulo do sprite do personagem e velocidade de movimentação do mesmo.

#### **Itens:**

AID KIT (Kit de primeiros socorros)

Recupera 5 pontos de vida, sem exceder o limite de pontos de vida do personagem;

Green Herb (Erva Verde)

Muda para inativo o estado de envenenado;

Bullet Kit (Kit de balas)

Adiciona 10 balas no inventário do usuário, até o limite do o jogador suporta;

Mineral Water (Água Mineral)

Recupera 10 pontos de Sede, até o limite do o jogador suporta;

Cookies (Bolachas)

Recupera 10 pontos de Fome, até o limite do o jogador suporta;

#### **Mapas:**

Só será visível o mapa que o jogador está;

Cada mapa terá um som de fundo, podendo ter alguns efeitos sonoros inclusive;

A área do elevador e das escadas serão mapas distintos.

Segue os nomes e a estrutura dos mapas:

