UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA

GABRIEL RÉGIS BIURRUM

DESENVOLVIMENTO DE JOGOS ELETRÔNICOS COM HTML5

VITÓRIA DA CONQUISTA

2015

GABRIEL RÉGIS BIURRUM

DESENVOLVIMENTO DE JOGOS ELETRÔNICOS COM HTML5

Monografia apresentada como exigência parcial para obtenção do grau de Bacharelado em Ciência da Computação da Universidade Estadual do Sudoeste da Bahia.

Orientador: Prof. Dr. Fábio Moura Pereira

VITÓRIA DA CONQUISTA

2015

FOLHA DE APROVAÇÃO

GABRIEL RÉGIS BIURRUM

DESENVOLVIMENTO DE JOGOS ELETRÔNICOS COM HTML5

Monografia apresentada como exigência parcial para obtenção do grau de Bacharelado em Ciência da Computação da Universidade Estadual do Sudoeste da Bahia.

Aprovado em: ____/____/____.

BANCA EXIMINADORA

_____

Prof.ª. Dra. Cátia Mesquita Brasil Khouri
Universidade Estadual do Sudoeste da Bahia

_____

Prof. Dr. Fábio Moura Pereira (Orientador)
Universidade Estadual do Sudoeste da Bahia

_____

Prof. Dr. Roque Mendes Prado Trindade
Universidade Estadual do Sudoeste da Bahia

Vitória da Conquista, 03 de dezembro de 2015

AGRADECIMENTOS

Dedico este trabalho a minha família, que muito me apoiou e me incentivou a realizá-lo.

“Um certo tipo de perfeição só pode ser atingido através
de uma acumulação limitada de imperfeição.”

Haruki Murakami

RESUMO

O desenvolvimento de jogos com HTML5 se torna cada vez mais importante à medida que esta tecnologia confirma seu potencial e que aplicações de todos os tipos passam a migrar para o ambiente web. O objetivo deste trabalho é apresentar a criação de jogos eletrônicos com HTML5, abordando todos os processos de desenvolvimento e mostrando os principais recursos desta tecnologia. Antes disso, uma contextualização completa é realizada, partindo dos jogos em geral, seguindo até os jogos eletrônicos e seu desenvolvimento, e finalizando com a criação de jogos eletrônicos para Internet. Como exemplo da utilização do HTML5, um jogo completo e um *framework* básico foram desenvolvidos com esta tecnologia. A metodologia empregada no projeto consistiu da não utilização de códigos auxiliares nas etapas de desenvolvimento e de uma abordagem em todo o trabalho que dá ênfase na clara divisão entre os processos do desenvolvimento de jogos. A completude do HTML5, sua facilidade de uso, seu suporte, sua padronização, seu desempenho e sua constante evolução nos navegadores, fazem desta tecnologia uma das mais importantes para o desenvolvimento de jogos eletrônicos para Internet.

Palavras-chave: HTML5, jogo, desenvolvimento, Internet, web, *framework*.

ABSTRACT

Game development with HTML5 becomes increasingly important as this technology confirms its potential and applications of all types begin to migrate to the web environment. The objective of this paper is to present the creation of electronic games with HTML5, addressing all development processes and showing the main features of this technology. Before that, a complete context is performed, starting from the games in general, continuing to electronic games and their development, and ending with the creation of electronic games for the Internet. As an example of using HTML5, a complete game and a basic framework have been developed with this technology. The methodology used in the project consisted of not using ancillary codes in the development stages and an approach in all the work that emphasizes the clear division between the processes of game development. The completeness of HTML5, its ease of use , their support, their standardization, its performance and its constant evolution in browsers make this technology one of the most important for the development of electronic games to the Internet.

Keywords: HTML5, game, development, Internet, web, framework.

SUMÁRIO

1 INTRODUÇÃO

Os jogos sempre foram uma das principais opções de lazer do ser humano. Presentes na maioria das culturas, começaram de maneira mais primitiva, com a utilização apenas do corpo humano ou de recursos simples da natureza (VELOSO, 2009). Com o surgimento dos dispositivos eletrônicos, a variedade de jogos se torna gigantesca. As possibilidades de interação e imersão aumentam à medida que os componentes físicos se tornam mais modernos (AHISTORIA, 2012).

Com o avanço da Internet, as páginas web se tornaram dinâmicas, capazes de exibir qualquer tipo de conteúdo. Isso permitiu que jogos inteiros pudessem ser colocados ali, trazendo inúmeras vantagens aos jogadores (MEYER, 2011). Jogos web podem ser acessados de qualquer lugar e a qualquer hora; tudo que o jogador precisa é de um navegador conectado, que pode estar sendo executado sobre qualquer plataforma. Outra praticidade está no fato destes jogos não precisarem ser baixados para poderem ser executados, eliminando problemas com instalação e aumentando a segurança dos jogadores, que não precisarão se preocupar com *malwares* (JUNIOR, 2010).

O HTML5 mudou o conceito de jogos para Internet. Antes desta tecnologia, os jogos eletrônicos precisavam de *plugins* ou outros recursos extras, em geral não padronizados, para poderem funcionar (BONATTI, 2014). O HTML5 é executado nativamente nos navegadores, e os seus jogos utilizam as mesmas linguagens e ferramentas que qualquer outra aplicação web nativa utiliza. É uma tecnologia padrão, isso garante que ela dificilmente deixará de funcionar e que atualizações e suporte provavelmente sempre serão uma realidade (MEYER, 2011).

Desenvolver um jogo eletrônico não é uma tarefa fácil. Muitos processos de diversas áreas distintas precisam ser levados em consideração para que tudo funcione corretamente. É necessário idealizar e produzir seu roteiro, sua arte gráfica, sua sonoplastia, seus meios de interação, sua lógica, suas telas e sua codificação (RABIN, 2012). Ao fim tudo precisa ser minuciosamente testado, para que nenhum problema ocorra durante a experiência dos jogadores (ARRUDA, 2013).

O desenvolvimento de um jogo para Internet utiliza os mesmos processos de um jogo eletrônico comum. Apenas algumas questões específicas desta plataforma, como o carregamento de arquivos, devem ser levadas em consideração (BONATTI, 2014).

Percebidos o enorme potencial do HTML5 para jogos e o quanto o HTML5 ainda tende a crescer, este texto tem como objetivos: mostrar os principais recursos desta tecnologia

web; explicar, genericamente, como acontece o desenvolvimento de jogos através dela; desenvolver um jogo eletrônico completo utilizando o HTML5; desenvolver um *framework* que auxilie na criação de jogos com esta tecnologia; demonstrar como foi a criação destes dois artefatos; e apresentar suas principais funcionalidades. Antes de especificamente abordar o HTML5, todos os processos do desenvolvimento de jogos eletrônicos serão bem explanados, tornando este trabalho compreensível até mesmo para quem nunca se aventurou na área da criação de jogos.

Como metodologia para o desenvolvimento do jogo, o autor optou por não utilizar códigos auxiliares, dando ênfase aos recursos elementares do HTML5, o que aprimora o aprendizado desta tecnologia. Como ferramenta de desenvolvimento, o Netbeans foi escolhido. Durante todo o projeto existe um esforço em tornar nítida a separação de todos os processos do desenvolvimento de jogos eletrônicos. Neste trabalho, estes são: game design, arte gráfica, sonoplastia, interação, lógica, *level design*, dados, software e testes.

O desenvolvimento do jogo seguiu sequencialmente as seguintes etapas: listagem dos principais recursos encontrados nos jogos eletrônicos; busca de alternativas eficientes para implementar em HTML5 todos estes recursos listados; estudo completo de todas essas alternativas encontradas; desenvolvimento da primeira versão do framework, que inicialmente continha recursos para a criação, organização, manipulação e exibição de elementos gráficos, e recursos para a manipulação de eventos do teclado, do mouse, das telas sensíveis ao toque e da rede através do protocolo TCP; formulação da ideia de um jogo que utilizasse um pouco de todo o conteúdo que foi estudado; elaboração do roteiro do jogo; adição do sistema de colisões no framework; implementação do servidor Java; implementação das quatro cenas do jogo, utilizando gráficos provisórios e apenas o protocolo TCP na comunicação; adição no framework de recursos para a utilização de áudios e da comunicação através do protocolo UDP; inserção dos sons do jogo; alteração no jogo de algumas ocorrências do protocolo TCP para o protocolo UDP; criação do servidor PeerJS; e por fim, a inserção do gráfico final do jogo. Durante todas estas etapas, testes foram realizados e os erros encontrados foram solucionados.

À medida que os capítulos deste texto avançam, os assuntos apresentados se tornam mais específicos. O capítulo 2 inicia falando sobre jogos em uma abordagem genérica e segue para os jogos eletrônicos, trazendo informações sobre suas plataformas, seus gêneros e seu papel no mercado. No capítulo 3, discute-se o desenvolvimento de jogos eletrônicos, apresentando todos os processos de produção. O capítulo 4 fala sobre o desenvolvimento de jogos para Internet, apresentando para este propósito as principais opções de tecnologias

encontradas atualmente e suas diferenças mais relevantes. O capítulo 5 é o foco deste trabalho. Ele explica resumidamente como funcionam todos os processos do desenvolvimento de jogos eletrônicos quando utilizamos o HTML5, além de apresentar os principais componentes desta tecnologia e qual é a sua atual realidade no mercado. Por fim, no capítulo 6 é mostrado um jogo desenvolvido em HTML5, deixando evidente como todos os processos do desenvolvimento de jogos eletrônicos foram utilizados neste caso específico. Além disso, são apresentados os principais elementos e recursos deste jogo, e como está a sua atual compatibilidade nos navegadores modernos.

2 JOGOS

Um jogo é normalmente definido como uma atividade com um ou mais jogadores onde estes devem tentar alcançar um ou mais objetivos, dentro das regras estipuladas, e com o propósito principal de se entreterem (BARRA; CARNEIRO; LEME; OTA, 1996).

Os jogos foram e são uma das maiores fontes de entretenimento do ser humano. Sejam seus jogadores crianças ou idosos, a ampla diversidade de jogos existentes está aí para agradar a qualquer tipo de pessoa (DEMONWEB, 2008). Existem registros da prática dessas atividades desde 2600 a.C., e em todas as culturas humanas. Com o passar do tempo, a quantidade de jogos foi aumentando exponencialmente, sendo hoje praticamente impossível contabilizar todos eles (OBOMPASTOR, 2010).

O ser humano sempre se mostrou bastante criativo quanto à elaboração de jogos. Essa motivação inicialmente surgia pela simples necessidade de lazer ou competição. Com o passar do tempo neste ambiente surgiu também uma vertente comercial, onde os jogos em geral passaram a ser planejados e construídos com seus desenvolvedores focados primeiramente no lucro que poderiam conseguir através destes instrumentos (DEMONWEB, 2008).

2.1 Jogos Eletrônicos

Jogos eletrônicos são jogos que dependem de um dispositivo eletrônico para funcionarem, como máquinas de cassino, computadores e celulares. Esses jogos podem estar agregados fisicamente aos equipamentos, ou podem ser inseridos nestes através de fitas, cartuchos, discos, cartões de memória, ou qualquer outro recurso de armazenamento de dados (OLIVEIRA, 2014).

A maioria dos jogos eletrônicos reproduz áudio e vídeo, sendo este último praticamente essencial para que os jogadores possam saber os acontecimentos atuais do jogo, e com base nisso interagir com ele de maneira eficiente (RABIN, 2012).

Os primeiros jogos eletrônicos não possuíam cores, sua resolução de tela era baixíssima, possuíam poucos elementos visuais, não continham qualquer inteligência artificial e a interação dos usuários era bastante limitada. Atualmente encontramos jogos eletrônicos com bilhões de cores, com elementos gráficos em três dimensões que beiram ao realismo, telas em alta definição, áudio de excelente qualidade, personagens que simulam muitas características complexas do ser humano, sistemas de física bastante avançados, interação ocorrendo por voz ou movimentos corporais, e uma intensa imersão de seus jogadores. Todo

este processo de desenvolvimento aconteceu em um período de seis décadas, mas foi nas últimas duas que os principais avanços ocorreram (AHISTORIA, 2012). Não seria de se espantar caso, daqui a alguns anos, pudéssemos interagir com os jogos apenas usando o nosso pensamento.

2.1.1 Plataformas

Por plataforma entende-se o conjunto de hardware, *firmware* e software necessários para um determinado jogo funcionar. Normalmente, como hardware nos referimos ao equipamento eletrônico por inteiro, independente de suas peças, e como software, ao sistema operacional, ou outros programas básicos (quando este sistema não existe), e a possíveis navegadores, emuladores e máquinas virtuais que sejam fundamentais para a execução do jogo eletrônico (RABIN, 2012).

As quatro plataformas mais famosas para a execução de jogos são os consoles, os fliperamas, os computadores pessoais (desktops, notebooks, tablets, smartphones etc) e estes últimos acrescidos de um navegador (ARRUDA, 2013). Outras plataformas, inicialmente não idealizadas para executarem jogos eletrônicos, estão adquirindo também este papel, e a tendência é que isso continue a aumentar. Muitas TVs atuais, por exemplo, dão suporte à execução de alguns jogos simples sem a necessidade de qualquer equipamento extra (RABIN, 2013).

Consoles

Consoles são plataformas bem definidas, desenvolvidas exclusivamente para a execução de jogos (AIDOBONSAI, 2010). Estes equipamentos, de maneira geral, não permitem que você faça modificações em seu hardware e software; eles permanecem iguais, funcionalmente falando, desde quando foram fabricados até o fim de suas vidas úteis (AHISTORIA, 2012). O componente de interação mais comum dos consoles é o *gamepad*, mas muitos outros acessórios podem ser externamente conectados a estes equipamentos para aprimorar a experiência dos jogadores (OLIVEIRA, 2014).

As principais vantagens de um console estão na dificuldade de se ter problemas com o desempenho dos seus jogos, e também de não ter que se preocupar com manutenções constantes neste equipamento. Como possuem jogos eletrônicos específicos, estes certamente foram desenvolvidos para obter uma excelente performance nesses dispositivos. E como as

pessoas poderão, no máximo, instalar alguns aplicativos básicos nos consoles, o sistema destes permanecerá praticamente intacto, dificultando qualquer manifestação de erros (AHISTORIA, 2012).

Os consoles modernos mais populares são: Xbox One, Playstation 4, Wii U, PS Vita e Nintendo 3DS (AHISTORIA, 2014).

Fliperamas

Fliperamas são plataformas ainda mais definidas que os consoles, e também desenvolvidas para o uso exclusivo de jogos. Basicamente existem duas grandes diferenças entre estas duas plataformas. Os fliperamas, na maioria das vezes, são equipamentos construídos para executarem um único jogo, diferente do console, em que você pode alternar entre centenas ou milhares de jogos, através de cartuchos e discos, por exemplo. Já a segunda diferença está no tamanho físico dos fliperamas, já que alguns destes equipamentos podem ocupar até mesmo salas inteiras. Este espaço avantajado na maioria dos casos está relacionado apenas à estética do equipamento, não se aplicando ao seu hardware (FLIPER MANIA, 2012). Exemplos clássicos de fliperamas são as máquinas de cassino, e os simuladores de corridas de carro, onde neste último o jogador interage com o jogo eletrônico principalmente através de um volante.

Computadores Pessoais

Computadores pessoais são computadores projetados para pertenceram a uma, ou no máximo, algumas pessoas, funcionando como um bem de caráter privado. Por esta razão, todos os tipos de dados, inclusive informações confidenciais, podem ser armazenados nestes equipamentos. Em situações comuns, os computadores pessoais só permitem o acesso de um usuário por vez (AHISTORIA, 2012). Estes equipamentos possuem arquiteturas que permitem a instalação de sistemas operacionais de propósitos gerais, como o Windows ou o Android, que possibilitam ao usuário gerenciar seus arquivos, acessar a Internet, desenvolver documentos e projetos, visualizar imagens e vídeos, ouvir músicas, se entreter com jogos, além de muitas outras aplicações, que podem ser instaladas sempre que o usuário desejar (HAMANN, 2011).

Desktops, notebooks, netbooks, ultrabooks, tablets e smartphones são exemplos clássicos de computadores pessoais.

Navegadores

Com o advento da Internet, programas computacionais chamados de navegadores, ou *browsers*, foram desenvolvidos para permitir o acesso às páginas web. Funcionam sobre um sistema operacional e destacam-se entre eles o Internet Explorer, o Mozilla Firefox, o Google Chrome, o Opera e o Safari (SITES GOOGLE, 2013).

Basicamente, todo conteúdo presente nas páginas exibidas pelos navegadores é resultado de códigos nas linguagens HTML, CSS e JavaScript, ou da utilização de *plugins*. Estas tecnologias podem exibir aos usuários textos, imagens, áudios, vídeos e recursos interativos, como jogos (SITES GOOGLE, 2013). Este conteúdo completamente dinâmico e interativo há poucos anos só era possível através da instalação de *plugins* nos navegadores. Com o surgimento do HTML5, jogos em páginas web, dentre outros conteúdos semelhantes, podem ser nativamente executados nos *browsers*, sem a necessidade de qualquer recurso adicional ser instalado (BONATTI, 2014).

Nos últimos anos, jogos web se tornaram muito populares, superando em quantidade todas as outras plataformas. Isso aconteceu basicamente pela facilidade de acesso e segurança fornecida por estes jogos eletrônicos. Não existe a necessidade de fazer o download e a instalação dos jogos, e, além disso, como existem versões dos principais navegadores para os sistemas operacionais mais comuns, estes jogos eletrônicos acabam de certa maneira sendo multiplataformas, podendo ser executados, por exemplo, tanto no Windows, como no Linux, sem qualquer configuração extra. Em relação à segurança, os navegadores não permitem que seus jogos fujam dos seus próprios escopos de acesso, ficando impossibilitados, por exemplo, de ler os arquivos pessoais dos usuários (GARSIEL; IRISH, 2011).

2.1.2 Gêneros

Classificar os jogos eletrônicos por gêneros é uma tarefa, no mínimo, complicada. Tempos atrás, quando os jogos eletrônicos eram mais simples, era fácil encontrar uma palavra que descrevesse a sua ideia principal. Porém, hoje em dia, com os jogos misturando uma série de elementos, muitas vezes completamente opostos entre si, é no mínimo equivocado dizer que estes simplesmente se tratam de jogos de ação, ou jogos de aventura, por exemplo (TECMUNDO, 2010).

No mercado de jogos, essa classificação é obrigatória e importante para organizar e filtrar os jogos eletrônicos. Infelizmente, a maneira de identificação atual os separa pela

utilização de apenas um termo, como RPG, tiro em primeira pessoa, plataforma etc., o que é bastante vago e não diz basicamente nada sobre a essência dos jogos (TECMUNDO, 2010). Dizer que Grand Theft Auto IV é simplesmente um jogo de ação, faz qualquer pessoa leiga imaginar uma série de possibilidades - será que se trata de um jogo de ficção científica? Será que se trata de um jogo sangrento?

Primeiramente, é necessário definir os tipos de classificações, ou seja, sobre quais aspectos os jogos serão comparados. Alguns exemplos clássicos são: em que posição em relação ao personagem principal a câmera ficará situada? Qual a sua temática mais importante? Existe o quesito tempo como obstáculo crucial? Pode ser jogado por várias pessoas simultaneamente? Faz simulação de algum jogo não eletrônico? É necessário um alto reflexo? São muitas perguntas que precisam de respostas, e certamente responder a apenas uma delas não ajuda muito a definir um jogo eletrônico (TECMUNDO, 2010).

Os termos mais utilizados para a classificação dos jogos eletrônicos são listados e brevemente explicados no Apêndice A. Uma boa definição de gênero para um determinado jogo geralmente está na junção de alguns desses itens.

2.1.3 Mercado

O mercado de jogos eletrônicos, apesar de existir a apenas quatro décadas, se desenvolveu mais rapidamente que qualquer outra indústria. Atualmente a receita anual obtida através dos jogos eletrônicos já supera a obtida pelo cinema, pela música e pelos livros, ficando atrás apenas do mercado bélico e de automóveis. Esta quantia anual já ultrapassa US$ 68 bilhões (MARQUEZ, 2012).

No quesito demanda, o Brasil está muito bem posicionado; o nosso país é o quarto maior consumidor de jogos eletrônicos do mundo, ficando atrás apenas dos Estados Unidos, da Rússia e da Alemanha (NUNES, 2013). Os jogos eletrônicos já fazem parte da cultura dos brasileiros, é muito difícil encontrar no país alguém que nunca tenha experimentado jogar por alguns minutos. Esse cenário motiva as grandes empresas internacionais a investirem no Brasil, seja barateando o preço de seus consoles e jogos, seja construindo produtoras de consoles no país, ou mesmo tornando disponíveis versões dos jogos eletrônicos em português (MARQUEZ, 2012).

Já em termos de produção de jogos eletrônicos, o Brasil ainda está muito atrasado em relação aos grandes produtores. Atualmente, os países que mais lucram com a criação de jogos eletrônicos são os Estados Unidos, o Japão e o Canadá. Apesar de ainda ser

consideravelmente pequeno, este negócio está crescendo bastante no Brasil; já existem por aqui mais de 220 empresas especializadas nessa área. A maioria dessas são de pequeno e médio porte e possuem até nove funcionários. Algumas outras empresas maiores já conseguem alcançar uma receita anual que supera R$ 15 milhões (NUNES, 2013).

Três grandes fatores ainda prejudicam um maior crescimento na produção de jogos eletrônicos no Brasil. O primeiro deles é a ausência de mão de obra especializada, o que está mudando com a constante abertura de cursos de graduação na área de desenvolvimento de jogos. O segundo é o alto custo de produção de um jogo, que em alguns casos pode ultrapassar milhões de reais. E o terceiro é a grande quantidade de impostos que as empresas devem pagar ao governo para exportar seus jogos eletrônicos (MARQUEZ, 2012).

Como uma alternativa para se manterem e evoluírem neste negócio mesmo com um baixo investimento financeiro, os desenvolvedores brasileiros encontraram uma excelente oportunidade nos jogos para dispositivos móveis e navegadores, além daqueles com temática educacional e de marketing. Estes jogos eletrônicos normalmente podem ser produzidos por uma pequena equipe, em poucos meses, e sem muitos gastos. A possibilidade de comercializar seus jogos através da Internet também tem incentivado a abertura e permanência de muitas empresas no Brasil (NUNES, 2013).

3 DESENVOLVIMENTO DE JOGOS ELETRÔNICOS

O desenvolvimento de jogos eletrônicos consiste em uma tarefa dividida em diversos processos flexíveis e possivelmente independentes entre si, mas que se integram em alguma fase do projeto (RABIN, 2012). Os jogos eletrônicos são programas computacionais, dessa maneira, quase todas as metodologias e técnicas presentes no desenvolvimento de sistemas podem ser aplicadas também nos jogos. Entretanto, alguns conceitos e processos se tornam mais evidentes e importantes quando criamos jogos eletrônicos (ARRUDA, 2013).

Diferentemente de como acontece com um software convencional, em um jogo eletrônico, a memória, o processador e a placa de vídeo são constantemente requisitados, pois a todo instante a lógica do jogo deve ser calculada, os objetos gráficos filtrados e renderizados, os sons reproduzidos, os dados de rede manipulados (em jogos eletrônicos online), além de diversos outros processamentos (CHANDLER, 2012). Tudo isso precisa fluir perfeitamente, com uma performance que torne a experiência dos jogadores agradável.

A seguir, serão apresentadas as principais etapas do processo de desenvolvimento de jogos eletrônicos. Outras podem surgir, dependendo da especificidade do jogo eletrônico.

3.1 Game Design

O primeiro passo para se desenvolver um jogo eletrônico é pensar em sua ideia principal. Ideias inovadoras têm se mostrado um sucesso no mundo dos jogos eletrônicos, uma indústria que possui uma infinidade de produtos semelhantes (SCHUYTEMA, 2008).

Com a ideia em mente, a próxima etapa é escrever o Game Design do jogo, um documento que deve resumir as suas características mais relevantes. Este artefato é importante para analisar a viabilidade de se desenvolver este jogo eletrônico, e para fornecer um material de consulta básico a toda a equipe de desenvolvimento. Além disso, pode ser utilizado como uma apresentação do projeto, objetivando conseguir bons investimentos, semelhante a um plano de negócios de uma empresa que deseja iniciar suas atividades (SCHUYTEMA, 2008).

O Game Design deve fazer uma breve descrição do jogo, apresentando seu gênero, enredo, fluxo principal, personagens, telas e conceitos de arte gráfica empregados, sistema de física, conceitos de inteligência artificial utilizados, recursos de interação dos jogadores, sons, e qualquer outro fator que seja relevante. Este documento também deve informar algumas características técnicas do jogo eletrônico, tal como as suas plataformas de execução. E por fim é importante acrescentar informações relacionadas ao campo de atuação do jogo, como

por exemplo, o seu público alvo (CHANDLER, 2012).

3.2 Arte Gráfica

Arte gráfica é o processo no desenvolvimento de um jogo eletrônico que consiste em confeccionar todos os seus artefatos visuais, tais como imagens e modelos tridimensionais, além da aplicação de técnicas gráficas, como filtragem, animação e iluminação (ARRUDA, 2013). Em resumo, essa área corresponde à criação de todo conteúdo visual a que os jogadores terão acesso.

Sua importância é enorme, pois geralmente é a nossa visão que nos faz em um primeiro contato, ou ter uma empatia por um jogo eletrônico, ou criar certa aversão a ele. Essa área precisa ser minunciosamente trabalhada, e a criatividade em conjunto com excelentes técnicas artísticas são de extrema valia (RABIN, 2012).

Os jogos eletrônicos podem conter elementos visuais bidimensionais (2D) ou tridimensionais (3D). Elementos em duas dimensões são imagens, eles possuem largura e altura, mas não possuem profundidade. São mais simples de serem confeccionados e mais rápidos de serem processados em comparação a objetos em três dimensões, e portanto geralmente são escolhidos quando a plataforma de execução tem um hardware limitado, ou quando a equipe de desenvolvimento é pequena. Essas imagens possuem dois formatos básicos, mapa de bits e vetorial. Imagens em mapa de bits são representadas por pixels, e as imagens vetoriais são representadas por equações matemáticas. O primeiro formato é processado com mais rapidez, mas ocupa mais espaço de memória e perde qualidade quando as imagens são ampliadas, diferente do segundo formato (ARRUDA, 2013).

Os objetos gráficos bidimensionais podem ser animados através da troca constante de seus conteúdos. A animação em duas dimensões é uma sequência de imagens (ou *frames*), que devem ser alternadas em grande velocidade (CHANDLER, 2012). Por exemplo, para animar um personagem que realiza um salto, geralmente quatro *frames* são suficientes para proporcionar uma boa sensação aos jogadores. Na maioria dos casos, uma grande quantidade de *frames* é armazenada em um mesmo arquivo, para que não haja a necessidade do jogo eletrônico acessar a memória secundária diversas vezes para uma mesma animação, o que poderia tornar este processo lento. Esse arquivo contendo várias imagens para animações é chamado de *spritesheet* (RABIN, 2012).

Os elementos gráficos tridimensionais são compostos por vários polígonos, por texturas que serão aplicadas sobre estas figuras geométricas e, possivelmente, por

agrupamentos ou ossos invisíveis, que servem para facilitar a animação desses objetos. Os triângulos são os polígonos mais utilizados na arte gráfica 3D, e esses podem ser definidos a partir das informações de posição dos seus vértices, e das arestas que conectam estes pontos. Já as texturas são gráficos bidimensionais que irão "cobrir" a geometria dos elementos tridimensionais, lhes dando um tom mais realístico. Quanto mais polígonos e texturas um objeto possuir, mais realista ele será e também mais demorado será seu processamento (ARRUDA, 2013).

3.3 Sonoplastia

Sonoplastia é o processo no desenvolvimento de jogos eletrônicos que trata da confecção de todos os seus sons. Tudo que é audível em um jogo eletrônico é de responsabilidade desta área (ARRUDA, 2013).

Apesar de no mundo dos jogos eletrônicos a arte gráfica normalmente ser mais importante que a sonoplastia, esta última está presente em quase todos os jogos. As músicas e efeitos sonoros tornam os jogos eletrônicos mais imersivos e em alguns casos aprimoram sua jogabilidade (RABIN, 2012). Às vezes não conseguimos perceber ao mesmo tempo todos os detalhes importantes em uma cena, e utilizar outro sentido além da visão para isso certamente ampliará a percepção dos jogadores (ARRUDA, 2013).

Os áudios contidos nos jogos eletrônicos podem ser basicamente encontrados na forma de efeitos sonoros, músicas de ambiente, e diálogos. Os efeitos sonoros são sons de curta duração, que são disparados nos jogos quando ações específicas ocorrem, tais como quando o personagem salta ou abre uma porta. As músicas de ambiente são áudios com média ou longa duração, e que normalmente são tocados repetidamente ao longo de uma cena. Os diálogos, assim como os efeitos sonoros, são disparados a partir de eventos específicos dentro do jogo eletrônico. Estes são formados exclusivamente por falas humanas em algum idioma, e possuem durações variadas (RABIN, 2012).

3.4 Interação

Interação é o processo no desenvolvimento de jogos eletrônicos que trata dos seus dados de entrada e de saída, ou seja, da comunicação existente entre os jogos eletrônicos e os jogadores, ou entre os jogos eletrônicos e outros dispositivos que não sejam aqueles em que estes jogos estejam sendo executados (CHANDLER, 2012). Prover uma excelente interação

torna menor a barreira existente entre os jogadores e os seus personagens, tudo fica mais fluido e natural, tornando os jogos eletrônicos muito mais imersivos (RABIN, 2012).

A entrada tem como objetivos especificar os equipamentos que poderão fornecer informações ao jogo, e informar quais eventos específicos acontecerão em decorrência da utilização desses dispositivos (ARRUDA, 2013). Por exemplo, em um jogo eletrônico que faz uso de um *gamepad*, o que acontece quando o jogador pressiona o botão direcional superior e seu personagem está no mapa principal? Já a saída tem por objetivos especificar os equipamentos que externarão os dados, e selecionar as melhores maneiras de transmitir essas informações (CHANDLER, 2012).

Existe uma grande quantidade de mecanismos de entrada que podem ser utilizados nos jogos eletrônicos. Os mais comuns são os *gamepads*, os *joysticks*, os teclados, os mouses, as telas sensíveis ao toque, os dispositivos de rede e os acelerômetros. Já os mecanismos de saída possuem menor variedade, e podem ser classificados em quatro tipos: reprodutores de vídeo, reprodutores de áudio, reprodutores de movimento e componentes de rede (RABIN, 2012).

A possibilidade de comunicação através das redes de computadores, principalmente através da Internet, ampliou consideravelmente a interação nos jogos eletrônicos. Os jogadores podem interagir, em um mesmo jogo eletrônico, com até milhares de outros jogadores, podendo estes estarem em qualquer lugar do planeta (RABIN, 2012). Nesses jogos eletrônicos é muito importante que os profissionais projetem adequadamente os pacotes que serão trafegados pela rede, aprimorando a velocidade e segurança da comunicação, itens essenciais em jogos online (ARRUDA, 2013).

3.5 Lógica

A lógica é o processo no desenvolvimento de jogos eletrônicos que trata de todos os seus eventos e ações, assim como as condições necessárias para que estes possam ocorrer. É a área que mais requer inteligência e raciocínio por parte de seus profissionais, e essa funciona como o “cérebro” dos jogos eletrônicos, tomando todas as decisões a partir dos seus dados de entrada para então produzir seus dados de saída (RABIN, 2012). Os responsáveis pela lógica de um jogo eletrônico devem construir algoritmos que descrevam todo o funcionamento deste jogo eletrônico (ARRUDA, 2013).

3.5.1 Inteligência Artificial

Inteligência artificial (IA), no contexto de jogos eletrônicos, é uma subárea da lógica que trata do "cérebro" dos personagens e de outros elementos ativos que não são controlados pelos jogadores (VIEIRA, 2012). Quanto mais abrangente e menos limitada essa for projetada, maiores serão os desafios encontrados pelos jogadores, e menor será a sensação de "solidão" quando estes estiverem jogando sozinhos (NORVIG; THALMANN; LATHAM, 2014).

A inteligência artificial pode ser utilizada para simular, na medida do possível, os comportamentos humanos dentro de algum escopo. Isso é importante para substituir um jogador real por um virtual. A maioria dos jogos eletrônicos que permite disputas entre duas ou mais pessoas, possibilita a virtualização de jogadores, tais como grande parte dos jogos de corrida, esporte, estratégia e luta (VIEIRA, 2012). Além da simulação de jogadores, a IA pode ser utilizada nos jogos eletrônicos para inserir comportamentos em outros elementos, tais como armadilhas e inimigos.

3.5.2 Física

Física, no contexto de jogos eletrônicos, é uma subárea da lógica que trata da inserção de elementos da ciência física nos jogos eletrônicos. Como os jogos, na grande maioria das vezes, apresentam conteúdo e jogabilidade ficcional, é comum encontrar nestes uma física um pouco distorcida, com variações em suas equações, e sem muito compromisso com a realidade. Entretanto em outros, como aqueles dos gêneros de corrida e esporte, a física é levada a sério, a fim de simular com bastante precisão o que acontece no mundo real (RABIN, 2012).

Basicamente, encontramos nos jogos eletrônicos características de dois grandes segmentos da física: mecânica clássica e ondulatória. Dessas áreas é comum a utilização dos conhecimentos da cinemática, dinâmica, trabalho e mecânica, sistema de partículas, colisões, movimento rotacional, gravitação, fluidos, som e luz. Os dois últimos são tão importantes que possuem seus próprios processos dentro do desenvolvimento de um jogo (ARRUDA, 2013).

3.6 Level Design

Level design é o processo no desenvolvimento de jogos eletrônicos que trata da confecção dos seus mapas, cenários e telas como um todo, e não dos elementos pelos quais esses são

compostos. (ARRUDA, 2013) Os seus profissionais (*level designers*) a partir dos conceitos empregados no Game Design e da lógica modelada para o jogo, organizam e distribuem os artefatos gráficos em seus respectivos ambientes. Esses desenvolvedores arquitetam os espaços onde os jogadores irão interagir, restando aos programadores apenas codificar estes modelos (RABIN, 2013). Essa área possui forte influência na jogabilidade, pois ela está bastante atrelada com a criação dos desafios encontrados nos jogos eletrônicos, principalmente em jogos de ação, aventura e plataforma (CHANDLER, 2012).

3.7 Software

Quando todos os artefatos de um jogo eletrônico já tiverem sido confeccionados ao longo dos seus diversos processos de desenvolvimento, chega o momento de integrar tudo o que foi produzido e transformar o projeto em algo funcional, ou seja, em um software. Um jogo eletrônico, tecnicamente falando, é um programa computacional, e como tal, ao fim de sua construção precisa gerar componentes (geralmente arquivos) que tornem sua execução possível em determinada plataforma (RABIN, 2013).

3.7.1 Codificação

Codificação é o processo no desenvolvimento de jogos eletrônicos que consiste em transformar em código todos os seus algoritmos e parte de seus outros artefatos. Normalmente, apenas os elementos resultantes dos processos de interação, lógica e *level design* são codificados. Entretanto, várias técnicas e procedimentos oriundos da arte gráfica e sonoplastia também devem ser convertidos em código para poderem funcionar (ARRUDA, 2013).

O processo de codificação geralmente acontece pela utilização de linguagens de programação. No âmbito dos jogos eletrônicos, as mais comuns são: Assembly, C, C++, C#, Java, JavaScript, ActionScript, Lisp, Prolog e Lua. Dependendo do jogo, é comum o uso em conjunto de duas ou três destas (RABIN, 2013). Em alguns casos a codificação também pode ocorrer utilizando-se elementos gráficos ao invés de textuais, como diagramas ou painéis de eventos. Isso é bastante comum em motores de jogos destinados a desenvolvedores iniciantes (ARRUDA, 2013).

3.7.2 Motores de Jogos

Os motores de jogos são programas computacionais especializados no desenvolvimento de jogos eletrônicos que acoplam funções de vários de seus processos de confecção em um único ambiente. Todos artefatos se interagem com facilidade, poupando os desenvolvedores de muito trabalho desnecessário. Alguns motores são simples, recomendados para quem está iniciando nesse mundo, porém existem outros com um poder de produção impressionante, podendo ser utilizados para grandes projetos, e até mesmo por grandes empresas (JÚNIOR, 2015). Como exemplos desses sistemas têm-se o Construct e o Unity3D, o primeiro é mais indicado para jogos bidimensionais e o segundo para tridimensionais.

Todo bom motor de jogos deve conter alguns elementos que são essenciais para que ele seja produtivo. O editor de cenas é um recurso de bastante importância para os *level designers* e designers de interface gráfica. Com ele é possível posicionar, e manipular os artefatos gráficos do jogo eletrônico, assim como ajustar suas propriedades básicas, sem a necessidade de um software extra, facilitando a integração com os outros processos. Os motores de jogos também devem permitir uma fácil importação e organização de todos os artefatos do jogo eletrônico, tais como imagens e sons. Um ambiente de codificação também é fundamental nestes motores, pois é ali que o jogo em si ganhará “vida”. Os jogos eletrônicos também devem ser depurados, testados e convertidos para a linguagem primitiva de alguma plataforma de jogos com facilidade (PIXSTUDIOS, 2014).

3.8 Testes

Após um jogo eletrônico, ou parte dele, estar pronto para ser executado, seu comportamento deve ser minunciosamente analisado, à procura de falhas técnicas e conceituais. Os profissionais que realizam esses testes, os testadores, agem como jogadores, explorando todos os elementos presentes no jogo eletrônico. Qualquer problema encontrado deve ser relatado à equipe de desenvolvimento, para que sua correção aconteça o mais depressa possível (NETO, 2013).

Após os jogos eletrônicos passarem por rigorosos testes, e seus problemas eminentes serem corrigidos, eles estarão prontos para serem distribuídos em sua versão final (RABIN, 2013).

4 DESENVOLVIMENTO DE JOGOS PARA INTERNET

Os jogos web, ou para Internet, são aqueles que possuem navegadores como plataformas de execução. Apresentam os mesmos processos de desenvolvimento dos jogos eletrônicos convencionais, entretanto existem algumas diferenças consideravelmente relevantes entre estes dois tipos de jogos eletrônicos (KARASINSKI, 2010).

Os artefatos dos jogos tradicionais, tais como códigos, imagens e áudios, ficam armazenados no hardware do jogador, já nos jogos web, por estes possuírem uma arquitetura cliente-servidor, inicialmente, todos os seus elementos estão presentes apenas no hardware do servidor web. À medida que o jogo eletrônico vai necessitando de seus dados, estes vão sendo enviados pela rede do servidor até o navegador cliente. Não existe a necessidade de todos os artefatos serem carregados para que o jogo possa iniciar, e na verdade essa é uma característica positiva deste tipo de jogo eletrônico, pois os jogadores só "baixam" o que de fato irão utilizar. Entretanto, para jogos maiores, como aqueles com elementos tridimensionais, constantes e longos carregamentos podem frustrar os jogadores (QUETANTO, 2013). As telas exibidas enquanto todos os artefatos necessários estão sendo carregados são chamadas de *preloaders* (CREATEJS, 2015).

Outra característica importante dos jogos eletrônicos web é a capacidade de serem executados em diversas plataformas distintas. O sistema operacional e o hardware que façam parte da plataforma e estejam em camadas inferiores ao navegador podem ser abstraídos. Se os navegadores, e as máquinas virtuais, que podem estar acima desses, forem capazes de executar um jogo, então não importa se o jogo eletrônico está a ser executado em um smartphone com Android ou em um desktop com Linux (QUETANTO, 2013). Entretanto, por estes jogos funcionarem sobre plataformas com tantas camadas, e por normalmente serem interpretados sem uma pré-compilação, a performance destes acaba sendo um pouco reduzida.

Os jogos web exigem que os jogadores estejam conectados à Internet para poderem jogar, já que dados precisam ser trafegados entre o servidor e o cliente. Entretanto, em alguns casos, se o jogo eletrônico já estiver sido completamente carregado, e os arquivos ainda permanecerem localmente armazenados, este poderá funcionar off-line (KARASINSKI, 2010).

Estes jogos eletrônicos podem ser facilmente integrados com as redes sociais e com outros sistemas web, já que ambos fazem parte de um mesmo ambiente (CAVALCANTI, 2012). Por exemplo, um jogo poderia utilizar as informações pessoais contidas no Facebook do jogador, com a finalidade de aprimorar seu entretenimento.

Os jogos web pecam em relação à segurança dos seus códigos, sendo fácil fazer cópias destes ou extrair informações que deveriam ser secretas. Como os códigos possuem no máximo uma simples compilação, estes são encontrados em alto nível, podendo ser facilmente entendidos (QUETANTO, 2013).

Nos tópicos a seguir serão apresentadas as principais tecnologias para o desenvolvimento de jogos web.

4.1 Flash

O Flash é uma tecnologia que foi desenvolvida pela Macromedia e que atualmente é atualizada e mantida pela Adobe Systems. Ela foi idealizada para se criar animações interativas embutidas nos navegadores. Mas sua capacidade de produção se estendeu a conteúdos mais complexos, como os jogos eletrônicos (KARASINSKI, 2010). A maior parte dos jogos web atuais foram desenvolvidos através desta tecnologia (PRADA, 2010).

Assim como a maioria das tecnologias web de mesmo propósito, para se executar jogos eletrônicos em Flash é necessário que um *plugin* específico seja instalado nos navegadores. No caso desta tecnologia, a máquina virtual Adobe Flash Player é utilizada (PRADA, 2010).

Para o desenvolvimento destes jogos eletrônicos é necessário utilizar a linguagem de programação ActionScript, de propriedade também da Adobe. Esta é uma linguagem interpretada e bastante poderosa, possuindo classes em sua API padrão para as mais variadas funcionalidades. Normalmente não é necessário utilizar bibliotecas ou outros códigos externos para a criação de jogos eletrônicos, já que a ActionScript é bem completa em termos de multimídia (KARASINSKI, 2010). O software Adobe Flash, que pode ser considerado um motor de jogos, é comumente utilizado para integrar todos os processos de desenvolvimento (CAVALCANTI, 2012).

4.2 Unity

O Unity é um motor de jogos proprietário desenvolvido pela Unity Technologies, que produz com maestria jogos eletrônicos bidimensionais, e principalmente tridimensionais, para uma infinidade de plataformas distintas, dentre essas os navegadores (GASPAROTTO, 2014). É a principal tecnologia para o desenvolvimento de jogos eletrônicos web 3D (PRADA, 2010). Para esses jogos poderem funcionar é necessário que um *plugin* denominado Unity Web

Player seja instalado nos *browsers* (CAVALCANTI, 2012).

Esse motor de jogos é bastante completo e sofisticado. Ele possui um excelente editor de cenas e uma poderosa IDE para a programação dos jogos eletrônicos. O Unity trabalha com três linguagens de programação: Boo, C# e JavaScript. Além disso, ele trás por padrão uma série de artefatos prontos que podem agilizar o processo de desenvolvimento dos jogos (PRADA, 2010).

Entretanto, os jogos web desenvolvidos no Unity possuem um carregamento demorado nos navegadores dos jogadores que possuem uma velocidade de Internet não muito boa, já que estes jogos eletrônicos são mais pesados em termos de armazenamento, uma consequência dos seus inúmeros recursos avançados (PRADA, 2010).

4.3 Silverlight

O Silverlight é uma tecnologia de propriedade da Microsoft, desenvolvida para concorrer diretamente com aplicações Flash. Inicialmente, era mais utilizada para exibição de vídeos, mas com o tempo foi ganhando espaço no mundo dos jogos web. Para que estes jogos eletrônicos funcionem é necessário que um *plugin* específico, também denominado Silverlight, esteja instalado nos navegadores dos usuários (CAVALCANTI, 2012).

No Silverlight a codificação dos seus jogos pode acontecer pela utilização de quaisquer linguagens de programação da plataforma .NET, tais como C#, VB.NET, Python e Ruby. A principal vantagem é que todas essas linguagens podem ser combinadas em um mesmo projeto, possibilitando a utilização do melhor de cada uma delas. Para a confecção das telas uma linguagem de marcação denominada XAML pode ser utilizada, separando a visão da lógica dos jogos eletrônicos, e facilitando o serviço dos *level designers* e profissionais da interface gráfica. Como IDE os desenvolvedores geralmente utilizam o Visual Studio, um ambiente completo para a construção de aplicações da Microsoft (SATO, 2012).

4.4 HTML5

Até poucos anos atrás, jogar através dos navegadores exigia a instalação de algum software proprietário adicional, o que para muitos jogadores é uma grande inconveniência. Estes *plugins* podem comprometer a segurança dos usuários e reduzir a performance dos seus sistemas. Além disso, é mais um elemento que os jogadores precisarão manter atualizado (NÖRNBERG, 2011).

O próximo capítulo abordará em maiores detalhes os principais recursos do HTML5 para o desenvolvimento de jogos eletrônicos.

5 DESENVOLVIMENTO DE JOGOS COM HTML5

É importante salientar que o termo HTML5 como utilizado neste texto, da mesma maneira como faz a maioria dos autores, se refere à tecnologia como um todo, ou seja, a junção das linguagens HTML, CSS e JavaScript, e não apenas à versão mais atual do HTML. Antes da versão 5 do HTML, estas linguagens eram mais independentes, era mais comum falar delas em separado. Atualmente elas se completam e se integram de uma maneira mais padronizada, cada uma com sua função, mas unidas organizadamente para um mesmo propósito final: desenvolver páginas web (PAGANI, 2013).

Desenvolver jogos utilizando HTML5, inicialmente, pode parece uma tarefa árdua. Apesar de apresentar uma enorme quantidade de recursos multimídia, essa tecnologia não é específica para os jogos eletrônicos, muitas coisas elementares da criação de jogos deverão ser implementadas “na mão” (BONATTI, 2014). Mas isso não é exclusividade do HTML5, já que todas as outras tecnologias apresentadas no capítulo anterior também foram criadas para outros propósitos.

Para evitar boa parte do esforço, na grande maioria dos casos os desenvolvedores utilizam motores de jogos, bibliotecas ou *frameworks* voltados para a programação de jogos eletrônicos (NÖRNBERG, 2011). Entretanto, é fundamental saber como as coisas essencialmente acontecem, se os desenvolvedores não quiserem ficar limitados aos recursos dessas ferramentas auxiliares. Estas muitas vezes não lidam com situações bastante específicas, ou não possuem a melhor das otimizações para certos cenários (PAGANI, 2013).

Este capítulo fará uma abordagem geral de praticamente todos os recursos do HTML5, que combinados possibilitam o desenvolvimento de qualquer tipo de jogo para Internet. Serão apresentadas as primitivas mais básicas da tecnologia, sem a utilização de quaisquer ferramentas auxiliares.

5.1 HTML

HTML (HyperText Markup Language, que significa Linguagem de Marcação de Hipertexto) é uma linguagem de marcação desenvolvida inicialmente para interligar instituições de pesquisa próximas, e compartilhar documentos com facilidade entre elas. Em pouco tempo se tornou a linguagem padrão para descrever páginas web do mundo inteiro. Com a especificação do HTML5, os recursos multimídia foram ampliados, possibilitando o desenvolvimento de qualquer tipo de jogo para Internet (BONATTI, 2014).

Um documento HTML é basicamente composto por *tags* (etiquetas), que servem para descrever, dentre outras coisas, quais são os elementos gráficos presentes na página, e onde estes estão localizados em relação uns aos outros. Grande parte destas *tags* podem ter outras *tags* filhas em sua composição (W3SCHOOLS, 2015). Por exemplo, um documento pode conter uma *tag* "div" e outra *tag* "input" como filha desta primeira.

Para a maioria dos jogos em HTML5, a *tag* mais importante é o "*canvas*". As demais, "div", "input", "button", "p", "img", etc, costumam ser utilizadas apenas na interface gráfica do usuário (GUI), como nas telas que precedem o funcionamento do jogo eletrônico em si (MEYER, 2011).

5.2 CSS

CSS (Cascading Style Sheets, que significa Folhas de Estilo em Cascata) é uma linguagem de folhas de estilo específica para a formatação de elementos codificados em uma linguagem de marcação, tal como HTML. Foi desenvolvida com o intuito de separar os códigos que definem os elementos de uma página, dos códigos que descrevem a maneira como estes elementos serão apresentados na mesma, tornando o processo mais organizado e mais fácil de ser atualizado (BONATTI, 2014). No desenvolvimento de jogos, o CSS tem sua maior utilidade na formatação de elementos GUI, tais como botões ou caixas de texto (PAGANI, 2013).

Os documentos CSS são compostos por blocos. Cada um desses blocos é composto de seleções e de instruções. As seleções visam filtrar um ou mais elementos atualmente presentes na página com base em suas características atuais. A *tag*, o id, as classes e os valores dos atributos são alguns dos itens de um elemento que podem ser utilizados em uma seleção. As instruções serão executadas para todos elementos que satisfizeram os critérios das seleções. Cada instrução é composta por um par: propriedade-valor. A propriedade se trata da característica do elemento que será modificada, e o valor corresponde ao novo comportamento desta característica (W3SCHOOLS, 2015).

5.3 JavaScript

JavaScript é uma linguagem de programação baseada na ECMAScript, e que possui como objetivo principal fornecer mais dinamismo às páginas web (BONATTI, 2014). Com ela a interação entre os usuários e uma página podem aumentar exorbitantemente. É possível lidar

com a maioria dos dispositivos de entrada existentes, manipular elementos html, formatar estes elementos através de propriedades herdadas do CSS, alterar as configurações da janela do navegador, ou praticamente qualquer outra coisa que esteja relacionada ao escopo da página. Sem ela temos páginas estáticas e sem vida, com ela abrimos um leque de opções, sendo possível, por exemplo, construir verdadeiras aplicações e poderosos jogos (PAGANI, 2013).

No desenvolvimento de jogos com HTML5, JavaScript é o recurso mais importante deste processo, sem esta linguagem é impossível criar qualquer tipo de jogo eletrônico. Ela é responsável por carregar, controlar, manipular e unir todos os artefatos do jogo, tais como imagens e áudios; por manusear todos os dispositivos de interação; por criar toda a lógica do jogo eletrônico, incluindo sua inteligência artificial e sua física; por construir e organizar a maioria dos elementos na tela; e por qualquer outro processamento presente no jogo (MEYER, 2011).

JavaScript é muito semelhante a linguagens como C++, Java e Python. Ambas possuem sintaxes muito próximas e apresentam um comportamento imperativo, estruturado e baseado em objetos (W3SCHOOLS, 2015).

5.4 Navegadores

Atualmente os navegadores mais utilizados pelos usuários são o Google Chrome, o Internet Explorer, o Mozilla Firefox, o Safari e o Opera. Ambos apresentam versões para várias plataformas distintas, com menos destaque nesse sentido para o Internet Explorer e o Safari, que basicamente possuem suporte apenas nas plataformas pertencentes às empresas desses *browsers* (SITES GOOGLE, 2013).

Estes navegadores são bastante semelhantes. Em suas versões mais recentes ambos possuem, dentre outras coisas, abas, botões de controle, uma barra de endereços, sites favoritos, histórico, configurações gerais, e uma área maior onde o conteúdo da página é exibido. Após o usuário submeter o endereço de uma página, se este for localizado, o servidor web associado a ele enviará um documento como resposta ao usuário, além de todos os arquivos de que esta página necessita. Assim que estes vão sendo recebidos, tudo que for pertinente irá sendo renderizado pelo *browser* (SITES GOOGLE, 2013).

Os renderizadores estão entre os principais componentes dos navegadores. Boa parte da performance das páginas está atrelada ao desempenho deles. São eles que desenham tudo que vemos na tela de um *browser*, todos os elementos. WebKit é o renderizador utilizado

atualmente pelo Safari e pelo Opera, e que por um longo tempo foi empregado pelo Google Chrome. Este último recentemente migrou para o Blink. O Mozilla Firefox faz uso do Gecko, e o Internet Explorer do Trident (GARSIEL; IRISH, 2011).

5.4.1 Compatibilidade de Recursos

As tecnologias que compõem o HTML5 estão em constante evolução, exigindo que os navegadores estejam sempre se atualizando para conseguirem acompanhar estes novos recursos. Primeiro estas funcionalidades são definidas conceitualmente e padronizadas internacionalmente. A partir destas definições cabe aos desenvolvedores dos navegadores implementarem, o que diz nestas especificações, o mais semelhante quanto possível (GARSIEL; IRISH, 2011).

Os navegadores adicionam novos recursos conforme os padrões internacionais pedem, mas a maneira com eles fazem isso pode variar um pouco de um para outro. Talvez certos detalhes nas especificações sejam ambíguos ou nem mesmo estejam definidos, o que pode gerar diferentes interpretações. Isso faz com que um mesmo código nem sempre tenha o mesmo comportamento em todos os *browsers*. Geralmente as divergências são pequenas, mas é importante os desenvolvedores ficarem atentos a isso, tentando sempre adaptar seus códigos às particularidades de cada navegador (CANIUSE, 2015).

5.5 Arte Gráfica

O HTML5 permite a renderização de qualquer tipo de gráfico, seja ele em sua composição bidimensional ou tridimensional. Existem diversas maneiras de tratar este conteúdo visual utilizando esta tecnologia (ROUSSET, 2013). Os principais métodos serão abordados a seguir.

5.5.1 Canvas

O Canvas é o recurso visual mais utilizado quando se pretende desenvolver jogos em HTML5. Em HTML ele é representado pela *tag* "*canvas*", e em sua forma inicial é graficamente um retângulo invisível. A manipulação deste elemento só é possível através de códigos em JavaScript (ROUSSET, 2013).

Com ele é possível reproduzir tanto gráficos bidimensionais quanto tridimensionais. O

Canvas é um mapa de bits, ou seja, ele é composto de pixels, não sendo portanto muito utilizado para o desenho de imagens vetoriais (BONATTI, 2014).

Gráficos bidimensionais são renderizados no contexto 2D de um Canvas. Para obter este contexto basta aplicar o método “getContext(“2d”)” em um objeto Canvas. Neste objeto que foi retornado é possível desenhar imagens, textos e primitivas gráficas, tais como retângulos, círculos, retas e arcos. Também é possível alterar várias das propriedades destes desenhos, como preenchimento, borda, sombra, opacidade, escala, rotação e posição. Além disso, onde as cores podem ser aplicadas também é possível aplicar gradientes e imagens (W3SCHOOLS, 2015).

Para animar um Canvas é necessário alterar o desenho nele contido. Tudo que é renderizado neste elemento vai sobrepondo o que já foi desenhado anteriormente. Portanto na maioria dos casos, antes de alterar o conteúdo da região do Canvas que será animada, é necessário limpar esta área com um retângulo transparente, utilizando para isso o método “clearRect”. Então para uma animação contínua basicamente este ciclo deve acontecer: a área é apagada, desenhos são feitos nela, a área é apagada novamente, desenhos são feitos nela novamente e assim por diante (W3SCHOOLS, 2015).

WebGL

WebGL é um recurso do HTML5 que porta a API gráfica mais popular, OpenGL, para os navegadores. Com ele é possível construir qualquer tipo de jogo web tridimensional e obter excelente performance em sua execução, pois o WebGL utiliza todo o potencial da placa de vídeo dos jogadores (PRADA, 2010).

O WebGL funciona sobre o elemento Canvas, semelhante a como ocorre com gráficos bidimensionais, e para utilizá-lo também é necessário primeiramente obter um objeto de contexto. Para tal, o método “getContext(“webgl”)” deve ser utilizado em um Canvas. As funcionalidades deste contexto lidam com entidades tridimensionais, tais como vértices e texturas, e suas propriedades e métodos são bastante semelhantes àqueles encontrados no OpenGL. Quase todo código que é aplicado sobre este contexto, é interpretado pelo navegador e executado diretamente na placa de vídeo (TAVARES, 2012).

5.5.2 Elementos HTML

Elementos HTML são entidades gráficas padrão da linguagem HTML representadas por *tags*

de marcação. Podem ser alvos de códigos CSS, possuem vários comportamentos, e podem ser manipulados através de códigos em JavaScript (PAGANI, 2013). Nos jogos com HTML5 são fortemente utilizados para construir a interface do usuário, não sendo muito apropriados para uso em outras ocasiões. Alguns elementos HTML em particular são de extrema importância para os jogos eletrônicos. Eles são: “div”, “button”, “input”, “p”, “label”, “img” e “svg” (BONATTI, 2014).

A utilização destes elementos pode aumentar a acessibilidade dos jogos. Visualmente falando, todos eles são gráficos vetoriais, podendo ser escalonados sem perda de qualidade. Sonoramente falando, os textos contidos nestes elementos podem ser facilmente lidos por sistemas de leitura para deficientes visuais instalados nos navegadores (ROUSSET, 2013).

5.6 Sonoplastia

Antes do HTML5, reproduzir áudios em um navegador só era possível através de *plugins*. Atualmente, com a utilização do elemento HTML representado pela *tag* “audio”, os sons podem ser nativamente executados nos *browsers* (SMUS, 2013). Este elemento, assim como todos os elementos gráficos mencionados anteriormente, também pode ser criado e manipulado em tempo de execução através de códigos em JavaScript.

Reproduzir áudios com HTML5 é bastante simples. Primeiramente um elemento “audio” deve ser criado e o endereço de um determinado arquivo sonoro deve ser fornecido. Após parte deste arquivo ser automaticamente carregado, basta utilizar o método “play” para que a reprodução inicie. O formato de arquivo de áudio “MP3” atualmente encontra suporte nos principais navegadores. Já os outros dois formatos padrão - “WAV” e “OGG” - não são aceitos por alguns *browsers* modernos. O método “canPlayType” pode ser utilizado para verificar o suporte de determinado tipo de arquivo no navegador em que este procedimento está sendo executado (W3SCHOOLS, 2015).

Quando o jogo eletrônico exige um sistema de sonoplastia mais avançado, que deve ir além de carregar, reproduzir e parar um áudio, uma nova tecnologia do HTML5 deve ser utilizada: a WebAudio. Com ela é possível manipular completamente dados de áudio, podendo criar quaisquer tipos de efeito neles. Tudo acontece em torno de um objeto do tipo AudioContext, que representa um gráfico de processamento de áudio construído a partir de módulos de áudio ligados entre si, cada um representado por um AudioNode (SMUS, 2013).

5.7 Interação

O HTML5 oferece suporte a quase todos os mecanismos de entrada e saída existentes no mundo dos jogos eletrônicos. Destes, os principais são:

- Teclado – os seus principais eventos são: “keydown”, “keypress” e “keyup”. Estes normalmente possuem o objeto “window” como ouvinte, e sempre fornecem um objeto do tipo “KeyboardEvent” às suas funções de *callback* (W3SCHOOLS, 2015). A função de *callback* de um determinado evento é aquela função definida pelo desenvolvedor que será executada sempre este evento for disparado.
- Mouse e Telas Sensíveis ao Toque – os seus principais eventos são: “click”, “dblclick”, “mousedown”, “mouseup”, “mousemove”, “mouseenter”, “mouseover”, “mouseleave”, “mouseout”, “mousewheel”, “touchstart”, “touchend” e “touchmove”. Estes possuem elementos gráficos como ouvintes e fornecem objetos dos tipos “MouseEvent” e “TouchEvent” às suas funções de *callback* (W3SCHOOLS, 2015).
- Acelerômetro – os seus principais eventos são: “deviceorientation” e “devicemotion”. Estes possuem o objeto “window” como ouvinte, e suas funções de *callback* recebem objetos que informam a orientação e a aceleração do dispositivo (FILHO, 2012).
- Rede – para a comunicação através do protocolo TCP, um objeto do tipo WebSocket deve ser utilizado. Neste são adicionados ouvintes para os eventos: “open”, “error” e “message” (UBL; KITAMURA, 2010). Já para a comunicação através do protocolo UDP a tecnologia WebRTC deve ser empregada (DUTTON, 2014).

5.8 Lógica

A lógica em um jogo com HTML5 é completamente desenvolvida a partir de códigos em JavaScript. Esta linguagem é utilizada para descrever todo o comportamento do jogo eletrônico (MEYER, 2011).

Não existem no JavaScript mecanismos para lidar especificamente com a inteligência artificial. E, além disso, bibliotecas externas para aplicar IA nos jogos com HTML5 não são nada comuns (VIEIRA, 2012). Normalmente tudo é realizado utilizando apenas os recursos padrão desta linguagem, mas é possível escrever códigos relacionados à inteligência artificial em outras linguagens de programação mais apropriadas, e desenvolver em JavaScript uma maneira de interpretá-los.

No HTML5 não existem recursos específicos para lidar com a física nos jogos

eletrônicos, tal como a detecção de colisões entre elementos. Os algoritmos devem ser implementados do zero, a menos que os desenvolvedores façam uso de bibliotecas auxiliares, tais como a Box2D-JS e a Physijs, bastante populares dentre as existentes. Com estas é possível simular o segmento da mecânica clássica presente na física em duas dimensões e em três dimensões, respectivamente (MOON, 2012).

5.8.1 Multiprocessamento

Atualmente a maioria dos computadores é encontrada portando processadores com mais de um núcleo de processamento, capazes de realizar diversas tarefas simultaneamente (BONATTI, 2014). Com o propósito de aproveitar esta tecnologia nos navegadores, a API WebWorkers foi inserida no HTML5. Ela permite a execução de mais de um script ao mesmo tempo, tirando proveito total dos processadores dos usuários (BIDELMAN, 2010).

Utilizar esta API para criar novas tarefas paralelas é bastante simples. Primeiramente um objeto do tipo “WebWorker” deve ser criado no script principal da página, fornecendo ao seu construtor o endereço do arquivo JavaScript que contém o código da nova tarefa. Feito isso, o método “postMessage” pode ser utilizado neste objeto para iniciar sua execução. Este mesmo método também é utilizado para a troca de qualquer informação entre o script principal e o script da tarefa. Para que todas essas ações possam acontecer é necessário entretanto que ambos os scripts adicionem ouvintes para o evento “message” (BIDELMAN, 2010).

5.9 Dados

O HTML5 e os navegadores reconhecem diversos tipos de dados e arquivos. São aceitas como linguagens o HTML, o CSS, o JavaScript, o XML e os derivados destes. Como dados multimídia os principais formatos de imagem, áudio e vídeo são permitidos. Outros dados comumente aceitos são os textos simples, os arquivos PDF e as compactações JSON (CREATEJS, 2015). Além disso, quaisquer outros tipos de arquivos e dados podem ser lidos e manipulados, contanto que os desenvolvedores criem mecanismos para interpretá-los corretamente.

5.9.1 Preloader

Antes que um arquivo possa ser utilizado em um jogo ele precisa primeiramente ser carregado do servidor web para o dispositivo do jogador. Esse carregamento pode acontecer a partir de dois métodos distintos: pela definição da localização do arquivo através de elementos HTML ou pela requisição do arquivo através de um objeto XMLHttpRequest (BIDELMAN, 2011).

5.9.2 Armazenamento Local

Antes do HTML5 os navegadores só podiam armazenar localmente, na máquina dos jogadores, pequenas quantidades de dados. Atualmente este limite foi consideravelmente acrescido, dando novas possibilidades aos jogos eletrônicos web (BELEM, 2012).

Os objetos globais localStorage e sessionStorage podem ser utilizados para armazenar informações nos navegadores. O primeiro mantém estes dados permanentemente, mesmo que os *browsers* sejam finalizados. Já o segundo apenas mantém estas informações durante o período em que os navegadores estão em execução. Em ambos os casos a manipulação destes dados acontece através das propriedades destes objetos (BELEM, 2012).

5.10 Software

A codificação de jogos eletrônicos em HTML5 consiste na utilização de três importantes linguagens: HTML, CSS e JavaScript. Para esta última existe uma enorme quantidade de bibliotecas auxiliares que facilitam a programação de todos os processos de um jogo (BONATTI, 2014).

Alguns *frameworks* também podem ser utilizados para ditar o fluxo principal de um jogo eletrônico, tal como o Cappuccino, que oferece uma arquitetura MVC (Modelo-Visão-Controle) às aplicações (BONATTI, 2014). *Frameworks* específicos para jogos podem oferecer diversas estruturas essenciais deste tipo de sistema prontas, como o laço principal, o manuseio de cenas, e o fluxo de eventos de interação, cabendo aos programadores apenas preencherem essas “molduras” com o conteúdo específico de seu jogo eletrônico (MEYER, 2011). O *framework* desenvolvido pelo autor deste texto para a confecção do projeto do estudo de caso utiliza este princípio.

Ainda existem poucos motores de jogos que geram jogos eletrônicos em HTML5, mas a tendência é que apareçam muitos deles nos próximos anos. Alguns motores já consolidados

há algum tempo, como o GameMaker e o Unity3D, estão inserindo o HTML5 como uma de suas plataformas alvo (PIXSTUDIOS, 2014). O Construct é o principal motor de jogos dedicado a jogos eletrônicos em HTML5. Com ele é possível criar praticamente qualquer tipo de jogo em duas dimensões. É bastante fácil de ser manuseado, e possui um bom editor de cenas (HTML5GAMEENGINE, 2015). Entretanto sua codificação através de linhas de eventos torna confusa a implementação de algoritmos mais complexos.

5.11 Testes

Os jogos eletrônicos desenvolvidos em HTML5 possuem como plataforma de execução os *browsers*. Por essa razão, para alcançarem uma gama maior de jogadores, é fundamental que estes jogos sejam testados no maior número possível de navegadores (NETO, 2013). A maioria dos *browsers* modernos traz um painel de ferramentas que pode ser ativado para prover uma série de utilidades aos testadores dos jogos eletrônicos (MEYER, 2011).

Em decorrência das linguagens que compõem o HTML5 serem interpretadas, sem qualquer pré-compilação, é comum erros sintáticos surgirem durante a execução dos jogos eletrônicos. Uma maneira de evitar este problema é fazer uso de IDEs que automaticamente detectam e alertam a ocorrência destes erros. Além disso, por essas linguagens serem dinâmicas, erros semânticos jamais serão automaticamente detectados antes da execução dos jogos (BONATTI, 2014).

5.12 Mercado

O mercado de jogos em HTML5 ainda é muito recente. Pouco se tem conhecimento sobre pessoas ou empresas que adquirem receitas através deste segmento. Entretanto é notória a grande demanda por jogos eletrônicos que são executados diretamente nos navegadores. Estes são práticos, seguros e podem ser jogados em uma grande quantidade de plataformas distintas (MEYER, 2011).

6 ESTUDO DE CASO: BOLITA

Este capítulo exemplificará a criação de um jogo eletrônico em HTML5. Inicialmente será apresentada uma síntese do Game Design do jogo. Em seguida, os objetos e classes mais importantes do seu *framework* são explicados. Após isso, é demonstrada a aplicação no jogo eletrônico dos principais processos do desenvolvimento de jogos eletrônicos. E, por fim, são abordadas questões relacionadas à compatibilidade de execução deste jogo.

Um *framework* simples foi desenvolvido para auxiliar em etapas repetitivas do desenvolvimento. Seus detalhes de implementação são claros e não escondem as estruturas básicas da programação em HTML5. Nenhuma API, *framework* ou motor de jogos de terceiros é utilizado, o que deixa o código ainda mais limpo e didático, útil para quem deseja conhecer ou aprender esta tecnologia web.

6.1 Visão Geral

Bolita é um jogo casual para web, que pode ser jogado tanto ao lado de jogadores artificiais (NPCs), quanto ao lado de outros jogadores reais, utilizando para isso a Internet.

Neste jogo eletrônico, cada jogador controla seu próprio e único personagem, que se trata de um objeto circular em constante movimento. Não é possível pará-lo, apenas alterar a sua direção e velocidade. Os jogadores devem ficar atentos, pois quando seus personagens colidem com outros personagens ou com as extremidades do cenário, eles perdem uma porção de sua vida, que está diretamente associada ao tamanho do seu círculo. Além disso, os personagens podem liberar pedaços de si em diferentes tamanhos, com vida igual à vida que o personagem em questão perdeu pela divisão. Estes fragmentos também possuem movimento constante, e perdem vida ao colidirem com as extremidades da cena. Entretanto, assim como os personagens, suas direções também podem ser alteradas. Estas partes dos personagens são mais lentas, porém funcionam como bombas. Ao colidirem com personagens rivais, são destruídas, mas causam um dano considerável neles. Além disso, diferentes itens surgem na tela conforme o andamento do jogo. Estes, ao colidirem com os personagens ou com seus fragmentos, podem lhes oferecer benefícios, como também desvantagens. Os jogadores são divididos em equipes, podendo uma equipe ter de um a três integrantes. A última equipe sobrevivente será a vencedora da partida.

Para se darem bem no jogo, os jogadores devem ter um raciocínio bastante ágil, pois eles precisarão ficar atentos aos seus personagens e a todas as suas partes o tempo inteiro, já que

estes não param e podem sofrer colisões indesejadas. Além disso, se quiserem vencer a partida, deverão encontrar uma boa maneira de destruir os seus adversários.

Este jogo eletrônico possui um sistema de salas que permite aos jogadores encontrarem seus amigos, ou conhecerem pessoas do mundo todo antes de iniciarem a partida propriamente dita. Nestas salas é possível conversar com outros jogadores ali presentes, além de definir diversas opções da partida e dos personagens.

A jogabilidade de Bolita, somada à sua característica *multiplayer* online, traz um diferencial muito importante para este jogo.

6.2 Objeto "Jogo"

"Jogo" é o objeto principal do jogo eletrônico e do *framework*. Todas as referências de objetos e funções do jogo eletrônico possuem "Jogo" como raiz. Ele contém o quadro principal do jogo; os objetos de conexão com o servidor principal e com o servidor do PeerJS; informações relacionadas aos estados atuais destas conexões; listas com as teclas pressionadas e com os arquivos carregados; e outras informações diversas que incluem o tempo da última animação e dados sobre resolução e FPS.

O jogo eletrônico iniciará sua execução quando o método "Jogo.iniciar" for chamado. Esse é automaticamente invocado quando a página é carregada. Em sua execução, a cena inicial é criada e passada ao quadro principal que também acabara de ser criado; ouvintes para eventos da tela e do teclado são adicionados; a tela do jogo eletrônico é ajustada de acordo com a resolução do dispositivo; e por fim é solicitado um *frame* do navegador através de "requestNextAnimation*Frame*".

Após o navegador fornecer este *frame*, o método "Jogo.animar" será executado. Em sua execução, o FPS será calculado, o temporizador da conexão com o servidor principal será continuado, o quadro principal será atualizado, e uma nova solicitação de *frame* será feita ao navegador. Isso causará um ciclo em torno de "Jogo.animar", gerando o famoso *gameloop*, um elemento fundamental em jogos eletrônicos.

As funções de *callback* para os eventos do teclado e de rede são definidas dentro do objeto "Jogo". Quando estas funções são executadas, elas invocarão funções de mesmo nome no quadro principal, propagando assim estes eventos. Além dos métodos mencionados nesse capítulo, o objeto "Jogo" possui basicamente funções relacionadas à tela do jogo eletrônico.

6.3 Classe “Quadro”

Um objeto da classe “Quadro” é um ElementoHTML do tipo “div”, que funciona como um contêiner de cenas. Ao ser criado, ele insere em seu interior um elemento Canvas de mesmas dimensões e adiciona ouvintes para os principais eventos do mouse ou das telas sensíveis ao toque, a depender do dispositivo em questão.

Quando uma cena é adicionada a um quadro, este faz uma leitura de todos os elementos HTML dela e os adiciona em seu interior. Se um elemento HTML é inserido, removido ou alterado em uma cena que já está dentro de um quadro, esta ação na cena é repassada ao quadro, que faz diretamente a devida operação com este elemento.

Diferente dos elementos HTML, que já são automaticamente renderizados pelo navegador, os elementos Canvas necessitam de constante renderização por código JavaScript. Este processo é realizado dentro da função “atualizar” do quadro, que é chamada constantemente dentro de “Jogo.animar”. Esta função do quadro primeiramente executa a lógica de sua cena e em seguida desenha todos os elementos Canvas dela através do método “desenharElementoCanvas”. Esta última função utilizará diferentes métodos primitivos de desenho, a depender do tipo do elemento a ser desenhado. Se um ElementoBitmap será renderizado, “drawImage” será utilizado para exibir sua animação atual. Se um ElementoCirculo será desenhado, “beginPath”, “arc”, “fill” e “stroke” serão utilizados. Agora, se um ElementoTexto será renderizado, então usam-se “fillText” e “strokeText”.

Quando um quadro possui dimensões menores que as dimensões da cena dentro dele, nem todo conteúdo visual pode ser exibido de vez. A definição de qual região da cena é mostrada é realizada através dos atributos “viewportX” e “viewportY” do quadro.

Um quadro possui funções que são executadas em decorrência de eventos com apontadores, teclado e rede. Os ouvintes e as funções de *callback* dos eventos com apontadores são definidos no próprio quadro; já os ouvintes e as funções de *callback* dos demais eventos são definidos no objeto “Jogo”, que propagará estas ações até este quadro. Nestas funções do quadro, todos os eventos serão propagados até sua cena.

6.4 Classe “Cena”

Um objeto da classe “Cena” representa a estrutura de uma tela. A cena em si não é um componente visual; ela mantém referência para todos os elementos gráficos que são exibidos quando ela se encontra dentro de um quadro. A cena diz ao quadro o que deve ser exibido, e

este é quem de fato mostra os elementos. Ela possui uma série de métodos para inserir, remover e alterar a ordem dos seus componentes visuais.

A cena também é responsável por armazenar os eventos de todos os seus elementos. Para cada evento relacionado a apontadores, ela mantém uma lista de quais são os seus elementos que estão ouvindo este evento. A cena em si também pode ouvir eventos de apontadores. Estes serão disparados quando acontecerem em qualquer região do quadro.

O jogo deste projeto utiliza uma classe diferente para cada uma de suas cenas. Essas classes herdam a classe “Cena”, e seis métodos desta são normalmente sobrescritos por elas. São eles: “logica”, “adicionadaAoQuadro”, “inicializarInterface”, “adicionarOuvintes”, “removidaDoQuadro” e “removerOuvintes”. “logica” representa a lógica daquela tela e será executado constantemente. “adicionadaAoQuadro” é invocado sempre que a cena é adicionada a algum quadro. “inicializarInterface” é executado apenas na primeira vez que a cena é adicionada a algum quadro, já que a interface gráfica de uma tela só precisa ser inicializada uma vez; é nesta função que o processo de *level design* desta tela acontece, ou seja, é nela que se encontra todo o código inicial dos elementos gráficos. “adicionarOuvintes” é invocado sempre que “adicionadaAoQuadro” é chamado; nesta função, normalmente aparecerão todas as adições de ouvintes para os eventos envolvendo os elementos iniciais da cena. “removidaDoQuadro” é executado sempre que a cena deixa o quadro devido à entrada de outra cena em seu lugar. “removerOuvintes” é invocado sempre que “removidaDoQuadro” é chamado. Nesta função, normalmente aparecerão todas as remoções de ouvintes para os eventos envolvendo os elementos atuais da cena.

6.4.1 Classe “CenaPreloader”

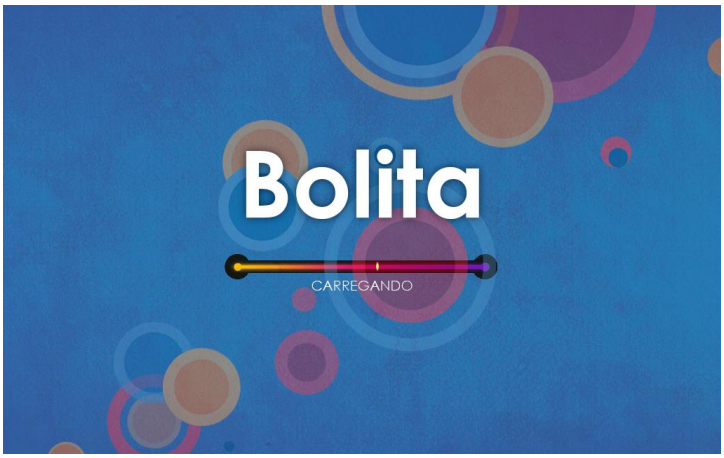

Figura 1: Tela gerada pela classe “CenaPreloader”

A Figura 1 mostra a tela inicial do jogo, responsável por carregar todos os arquivos que serão utilizados nas telas seguintes e apresentar aos jogadores um feedback do quanto já foi e do quanto ainda resta a ser carregado.

Para se criar uma instância de “CenaPreloader”, devem ser passados como argumentos uma lista com os arquivos a serem carregados e a classe da cena que será exibida quando o carregamento finalizar. Assim que esta cena é adicionada ao quadro, ela cria um objeto “createjs.LoadQueue”, que será responsável por carregar todos os dados. Ouvintes são adicionados a este objeto e seu método “loadManifest” é chamado para iniciar de fato o carregamento dos arquivos.

A função “carregamentoProgresso” é chamada sempre que existir um progresso no carregamento. Neste caso, o jogador será informado sobre este processo. “carregamentoErro” é executada quando um erro no carregamento é encontrado; se o erro estiver associado a um determinado arquivo, este arquivo tentará ser carregado novamente. “carregamentoCompleto” é chamada quando todos os arquivos tiverem sido carregados; estes se tornarão acessíveis para todo o jogo eletrônico, e a cena seguinte, no caso a CenaSalas, será definida como a cena atual do quadro.

A arte gráfica desta cena é composta apenas por elementos HTML com os quais os jogadores não podem realizar interações.

6.4.2 Classe “CenaSalas”

Figura 2: Tela gerada pela classe “CenaSalas”

Conforme mostra a Figura 2, esta tela possui como principal função exibir aos jogadores as salas criadas pelos outros jogadores que estão atualmente abertas. Eles poderão entrar em uma

destas salas, criar a sua própria sala, ou optar por jogar off-line apenas com NPCs.

Quando uma instância de CenaSalas é adicionada a um quadro, é verificado se o jogador está desconectado. Caso ele esteja, uma tentativa de conexão será iniciada. Esta cena possui funções de *callback* para diversos eventos de rede. “canalTcpMensagemRecebida” é executada sempre que o servidor principal envia respostas para algumas interações do jogador nesta tela, e quando alterações significativas são realizadas nas salas do jogo. Sempre que a comunicação com os servidores forem encerradas, novas tentativas de conexão logo serão executadas.

Os métodos de CenaSalas em sua maioria são respostas aos eventos de interação do jogador com os elementos de GUI da tela, tais como botões, caixas de texto e “divs”. Dentro destes, em diversos momentos, caso o jogador esteja conectado, mensagens informando estas ações serão enviadas ao servidor principal através do método “Jogo.canalTcp.send”.

A arte gráfica dessa cena é composta apenas por elementos HTML. Janelas, que são “divs” personalizadas, podem ser exibidas no decorrer da execução, informando algum acontecimento ou requisitando alguns dados do jogador.

O fluxo completo desta cena é encontrado no Game Design do jogo.

6.4.3 Classe “CenaSala”

Figura 3: Tela gerada pela classe “CenaSala”

Esta tela (Figura 3) representa a sala na qual o jogador entrou, independente da sala ter sido criada por ele ou não. Aqui, dentre outras pequenas coisas, são mostrados todos os jogadores presentes, suas informações pessoais e de jogo (que em geral podem ser modificadas), um chat de comunicação e diversas opções para configurar a partida a gosto dos jogadores.

Esta cena possui funções de *callback* para diversos eventos de rede. “canalTcpMensagemRecebida” é executada sempre que o servidor principal envia ao jogador respostas às suas interações na sala ou a confirmação a respeito de interações nesta sala realizadas pelos outros jogadores presentes.

Quando o administrador da sala permite o início da partida, algumas configurações de extrema importância são iniciadas. Os jogadores sincronizadamente passam a abrir conexões UDP com todos os outros jogadores ali presentes. A tentativa de conexão é realizada através do método “Jogo.peerConectar”. Quando um jogador recebe uma proposta de conexão, a função “peerConexaoRecebida” é executada. Se esta conexão foi efetivada, ambos os jogadores invocam a função “canalUdpIniciou”. Quando um jogador percebe que já possui canais de comunicação abertos com todos os demais jogadores, ele notifica este fato ao servidor. O servidor, ao receber esta notificação de todos os jogadores, informa ao administrador que a configuração foi concluída. Este, por sua vez, é levado até a cena da partida, onde calcula as informações iniciais dos personagens. Estes dados são enviados aos demais jogadores por intermédio do servidor. Somente após receber estas informações, estes jogadores não-administradores serão levados até a cena da partida.

Jogadores desconectados em CenaSala são automaticamente direcionados de volta a CenaSalas correspondente. A única exceção é caso esta desconexão seja do jogador em relação ao servidor WebSocket do PeerJS, e todos os canais UDP com os demais jogadores já tenham sido abertos. Neste caso, nada acontecerá e a partida seguirá normalmente.

Os métodos de CenaSala em sua maioria são respostas aos eventos de interação do jogador com os elementos de GUI da tela, tais como botões, caixas de texto e “divs”. Dentro destes métodos, em diversos momentos, caso o jogador esteja conectado, mensagens informando estas ações serão enviadas ao servidor principal através do método “Jogo.canalTcp.send”.

A arte gráfica dessa cena é composta apenas por elementos HTML. Janelas podem ser exibidas no decorrer da execução informando algum acontecimento ou requisitando algum dado do jogador.

O fluxo completo desta cena é encontrado no Game Design do jogo.

6.4.4 Classe “CenaPartida”

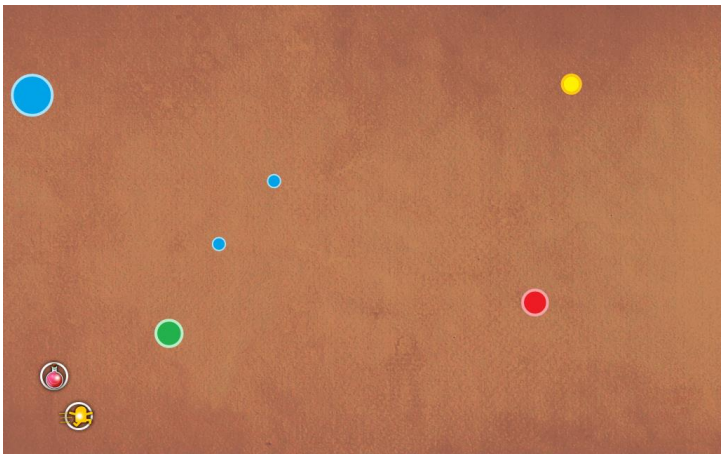

Figura 4: Tela gerada pela classe “CenaPartida”

A tela principal do jogo, apresentada na Figura 4, é onde de fato a jogabilidade acontece e onde os jogadores devem utilizar suas habilidades para encarar os constantes desafios e tentar alcançar a vitória.

Na função de inicialização da interface desta cena, se o jogador é o administrador da sala ele calculará, utilizando um fator aleatório, as posições e as direções iniciais dos personagens. Estas informações são enviadas aos demais jogadores, caso o jogo esteja online, e estes na mesma função de inicialização, utilizarão estes dados para posicionar e direcionar os personagens da partida. Ao fim de “inicializarInterface” o contador será criado.

Quando este contador chega ao seu fim, a partida entra em execução e a cena adiciona ouvintes para os seguintes eventos de apontadores: “mousedown”, “mousemov” e “mouseup”. Um ouvinte em “window” para o evento “blur” também é adicionado, pois o jogador pode retirar o jogo eletrônico de foco durante sua execução. Estando sem foco, o navegador não reconhecerá caso o jogador libere um apontador. Portanto, na função de *callback* deste evento, todos os apontadores pressionados são definidos como liberados.

A função “logica” nesta cena praticamente só tem alguma utilidade quando o jogador que a executa é o administrador da sala. Neste caso, o contador para iniciar a partida é atualizado se ainda estiver em jogo; a inteligência artificial é aplicada; as ações decorrentes dos apontadores pressionados são executadas; a movimentação dos personagens e dos fragmentos é continuada; as possíveis colisões são verificadas e resolvidas; os itens são criados no tempo correto; todas as mudanças são enviadas pela rede para os outros jogadores. Nesta função “logica”, algumas animações não-convencionais também são executadas, neste

caso, para todos os jogadores.

As funções de *callback* para os eventos com apontadores quando executadas pelo administrador já executam toda a lógica relativa a estas interações. Porém, quando executadas por jogadores comuns, primeiramente uma mensagem é enviada ao administrador informando sobre tal ação, e só após uma confirmação deste administrador é que de fato a lógica envolvendo aquela interação irá ocorrer. Esta arquitetura de centralização do administrador é importante para que não ocorram inconsistências no jogo eletrônico de um jogador para outro.

Esta cena possui funções de *callback* para diversos eventos de rede. “canalTcpMensagemRecebida” é executada sempre que uma mensagem segura transmitida por um jogador chega por intermédio do servidor. O administrador em geral receberá mensagens relativas às interações com apontadores realizadas pelos outros jogadores. Já os jogadores comuns geralmente receberão mensagens relacionadas a alguma alteração nas estruturas do jogo durante o processamento de sua lógica no administrador. Quando a partida termina e uma equipe vence, o jogador permanecerá na cena da partida até que pressione um determinado botão que o levará de volta até a cena da sala. Se o jogador ainda estiver na partida, e outro jogador que já tenha retornado para a cena da sala realizar alguma alteração ali, as mensagens recebidas do servidor pelo primeiro jogador, que deveriam ser interpretadas em “CenaSala”, não chegarão até li porque a cena atual do jogador é “CenaPartida”. Baseado nisso, quando uma mensagem que deveria ser entregue até “CenaSala” chega em “CenaPartida” esta mensagem é automaticamente redirecionada até “CenaSala”, para que as medidas corretas sejam tomadas.

Uma mensagem é enviada para um canal de comunicação UDP através do método “canalUdp.send”, onde canalUdp é o canal em questão. Quando uma mensagem de segurança não-garantida enviada por um jogador é recebida diretamente por outro jogador, a função de *callback* “canalUdpMensagemRecebida” é executada. Os tipos de mensagens recebidas aqui, tanto pelo administrador quanto pelos jogadores comuns, são semelhantes àquelas recebidas em “canalTcpMensagemRecebida”.

Se um jogador é desconectado na cena da partida, ele será levado até a cena das salas, e os jogadores restantes de sua sala receberão uma mensagem do servidor informando sobre esta desconexão. Estes jogadores removerão da sala o jogador desconectado e este será tido como derrotado na partida. Se este jogador era o administrador da sala, o novo administrador receberá dos demais jogadores uma mensagem informando quais apontadores eles mantém pressionados. Este procedimento é importante para manter o jogo em execução e sem inconsistências após a troca de administradores.

Dos métodos de “CenaPartida” não mencionados neste capítulo, a grande maioria funciona como procedimentos sem qualquer retorno, utilizados apenas para diminuir a reescrita de código.

A arte gráfica desta cena é composta basicamente de elementos “Canvas”. Apenas quando a partida termina, uma janela (“ElementoHtml”) é exibida informando aos jogadores a equipe vencedora. As classes dos elementos Canvas desta cena são: “Apelido” e “Contador,” que herdam “ElementoTexto”; “ElementoCirculoDinamico”, que estende “ElementoCirculo” e é estendida por “Personagem” e “Fragmento”; e por fim, “Superficie” e “Item”, que herdam “ElementoBitmap”.

O fluxo completo desta cena é encontrado no Game Design do jogo.

6.5 Arte Gráfica

Além do que será abordado nas subseções a seguir, em relação à arte gráfica deste jogo eletrônico, é importante salientar algumas outras coisas. O jogo entra em tela cheia e tem suas escalas ajustadas assim que é pressionado por um apontador pela primeira vez. Este modo permite ao jogador uma experiência muito mais imersiva. Além disso, dois arquivos CSS são utilizados para personalizar alguns elementos HTML em vários de seus estados.

6.5.1 Classe “ElementoGrafico”

Para toda a representação visual do jogo uma classe abstrata foi desenvolvida: “ElementoGrafico”. Ela generaliza qualquer componente gráfico do jogo eletrônico, deixando tudo mais padronizado. As imagens animadas dos itens, os círculos dos personagens, as janelas de alerta, os botões, as caixas de texto e todos os outros elementos herdam a classe “ElementoGrafico”.

Esta classe traz atributos e métodos fundamentais para a manipulação dos elementos gráficos de um jogo eletrônico, permitindo que eles sejam transladados, rotacionados, escalonados, e tenham sua opacidade e visibilidade alterados. Por exemplo, “personagem.setX(5)”, pode ser utilizado para posicionar o personagem a 5 pixels de distância da origem do eixo X.

“ElementoGrafico” é herdada pelas classes “ElementoHtml” e “ElementoCanvas”. “ElementoHtml” representa todos os componentes do HTML5 que podem ser construídos pela utilização de *tags*, tais como “button” e “input”. Por exemplo, “new

ElementoHtml(“button”)” é utilizado quando desejamos criar um botão. “ElementoCanvas” representa todos os elementos que necessitam de um *canvas* para serem exibidos, ou seja, tudo que já não é automaticamente renderizado pelo navegador. Ela é uma classe abstrata estendida por “ElementoBitmap” e “ElementoVetorialFechado”.

“ElementoBitmap” é utilizada para representar elementos que utilizam imagens em mapa de bits e que podem ser animadas *frame*-a-*frame*. Ela contém uma lista que deve conter todas as animações possíveis de um elemento deste tipo. “ElementoVetorialFechado” é uma classe abstrata que representa todos os elementos vetoriais que possuam uma geometria fechada, ou seja, que possam ter preenchimento em seu interior. Ela é herdada por “ElementoTexto” e “ElementoCirculo”. “ElementoTexto” representa um texto vetorial, e “ElementoCirculo” uma imagem vetorial com geometria circular.

6.5.2 Animação

Basicamente toda a animação presente neste jogo eletrônico faz uso da classe “Animacao”. Em alguns raros casos, a animação é desenvolvida em JavaScript na lógica da cena.

A classe “Animacao” representa uma animação em mapa de bits *frame*-a-*frame*. Uma instância desta classe deve ser associada a um único “ElementoBitmap”. Um objeto “Animacao” possui basicamente os seguintes componentes: uma imagem contendo as *sprites* (*spritesheet*), a definição destas *sprites*, e a ordem como estas *sprites* serão exibidas (*frames*). Várias opções podem ser fornecidas durante sua criação, e estas geralmente automatizam algumas informações, poupando, por exemplo, o desenvolvedor de definir cada *frame*, quando estes seguem uma ordem convencional. A Figura 5 mostra a *spritesheet* utilizada pelos itens da partida.

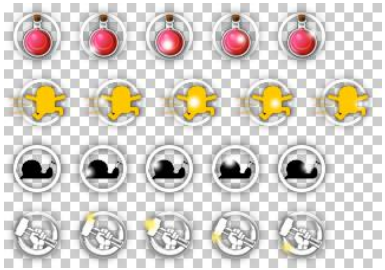

Figura 5: *Spritesheet* utilizada na animação dos itens da partida

O método “processar*Frame*Atual” de “Animacao” é sempre chamado durante a renderização do “ElementoBitmap” que está associado a essa animação. Ele verificará se já está no momento de trocar o *frame* atual. Se sim, o próximo *frame* será definido como o atual;

se não, nada mudará.

As demais funções de “Animacao” são comumente utilizadas para alterar o rumo da animação, permitindo pará-la, por exemplo. Elas são: “set*Frame*Atual”, “iniciar”, “reproduzir”, “pausar” e “parar”.

6.6 Sonoplastia

Bolita utiliza WebAudio na manipulação dos seus sons por duas razões. Primeiro porque os arquivos sonoros serão completamente carregados antes de serem utilizados, evitando que a reprodução de um áudio possa ser interrompida no meio do caminho. E segundo porque uma quantidade ilimitada de sons poderá estar em execução ao mesmo tempo.

O método “Jogo.reproduzirAudio” é utilizado sempre que um áudio precisa ser reproduzido. Ele recebe como parâmetros o nome do arquivo e as propriedades de reprodução. Os efeitos sonoros não utilizam qualquer propriedade, apenas são executados. Já as músicas de ambiente são tocadas ciclicamente e com um volume reduzido, através das propriedades “loop: -1” e “volume: 0.5”. Estas músicas são encerradas assim que suas respectivas cenas são removidas do quadro principal.

6.7 Interação

O *framework* deste projeto lida com quatro formas de interação bastante presentes nos jogos eletrônicos: teclado, mouse, telas sensíveis ao toque e rede.

6.7.1 Teclado

Este jogo não utiliza eventos do teclado, tendo como principal razão para isso sua execução correta em dispositivos que só possuem teclados virtuais, como a maioria dos modelos com telas sensíveis ao toque. Mesmo assim, o *framework* foi implementado para suportar este tipo de interação.

Os ouvintes para os eventos “keydown” e “keyup” são adicionados no objeto global “window” para que possam ser acionados em qualquer situação. Num jogo eletrônico, uma interação utilizando o teclado, na maioria das vezes, não está associada a um elemento em específico, mas sim à cena no geral. Por exemplo, “personagem.addEventListener(“keydown”, funcao)” não faz muito sentido, já que uma tecla

não pode ser pressionada especificamente em um personagem. Para elementos GUI, um evento de teclado específico ainda se torna possível, visto que estes podem ganhar e perder foco, intuindo que se um componente está em foco, qualquer evento do teclado está relacionado a ele.

Quando um destes dois eventos do teclado ocorre, sua respectiva função será chamada no objeto “Jogo”. Esta função atualizará uma lista que informa quais teclas estão atualmente pressionadas e quais não estão, e após isso propagará este evento até o quadro principal. Este quadro continuará com a propagação e levará este evento até sua cena. Esta cena então executará toda a lógica necessária em decorrência desta interação que foi realizada no teclado.

6.7.2 Mouse e Telas Sensíveis ao Toque

Os eventos acionados por estes dispositivos são os mais importantes deste jogo eletrônico. Eles estarão presentes durante todo o jogo, principalmente na execução da partida. Estes eventos são relativos a qualquer interação envolvendo o mouse e seus botões, e aos dedos pressionados numa tela sensível ao toque.

Apesar do mouse e das telas sensíveis ao toque possuírem métodos de interação semelhantes, eles não produzem os mesmos eventos. Para lidar com esta forma de interagir sem ter de sempre se preocupar se o jogador está utilizando um mouse ou uma tela sensível ao toque, o *framework* deste projeto oferece eventos genéricos para a interação com apontadores. O desenvolvedor lida com os eventos de uma maneira unificada, e o *framework* internamente e de maneira simples verifica qual o evento apropriado para aquele tipo de dispositivo.

Aplicar eventos de apontadores em elementos gráficos certamente faz muito mais sentido do que fazer isso com eventos do teclado, já que inevitavelmente a manipulação destes apontadores está associada a algum ponto da tela. Levando isso em consideração, os ouvintes para estes eventos abstratos são adicionados nos elementos da cena ou na própria cena. Quando o quadro é criado, ele adiciona ouvintes em si mesmo, em seu *canvas* e no objeto global “window” para os eventos elementares do mouse e das telas sensíveis ao toque (O “window” é utilizado em “mouseup” e “touchend”, pois estes eventos precisam ser disparados mesmo se o apontador for liberado fora do quadro). Quando o jogador interagir com o mouse ou com a tela sensível ao toque, uma função específica para aquele evento é disparada no quadro. Nestas funções, uma verificação na cena e em seus elementos é realizada à procura de ouvintes para os eventos genéricos que podem ser acionados por estes eventos elementares. Se ouvintes forem encontrados na cena, as funções de *callback* definidas

para estes eventos genéricos são invocadas. Se ouvintes forem encontrados nos elementos da cena, uma verificação de colisão será realizada entre a posição do apontador e o elemento em questão. Caso a colisão tenha ocorrido, as funções de *callback* definidas para estes eventos genéricos são invocadas.

Os eventos genéricos permitidos por este *framework* são: “mousedown”, “mouseup”, “click”, “mousemove”, “mouseover”, “mouseout” e “dblclick”. As funções de *callback* destes eventos recebem um objeto como argumento, que indica as coordenadas onde a ação aconteceu e qual foi o apontador envolvido. É nestas funções que a lógica do jogo eletrônico referente a estas interações deve ser implementada.

6.7.3 Rede

A interação através da rede tem um papel crucial neste jogo eletrônico. Se o jogador opta por jogar off-line, a rede se torna inútil; porém, se ele decide jogar online são os recursos de rede que tornarão isso possível.

Este jogo eletrônico utiliza TCP para mensagens que precisam chegar ao destino com segurança e UDP para mensagens que caso não cheguem corretamente ao destino não tragam problemas sérios ao jogo. Uma versão segura do UDP também é empregada em alguns casos. Esta possui segurança e velocidade razoáveis, um meio termo para os protocolos TCP e UDP puros.

A comunicação TCP só é possível com a utilização de um servidor. Os navegadores não permitem a criação de servidores “WebSocket”, apenas dos “WebSocket” propriamente ditos, que só podem iniciar uma conexão, e não receber. Portanto, outra plataforma deve ser escolhida para a criação de um servidor. Neste jogo, a JVM foi escolhida devido a sua característica multiplataforma, e os códigos deste servidor foram implementados utilizando Java.

A comunicação UDP, e sua variante, acontecem através da utilização da biblioteca “PeerJS”. Ela abstrai muitas diferenças encontradas nos navegadores quando este tipo de comunicação é utilizado. O “PeerJS” em seu interior possui uma conexão “WebSocket” com um servidor, que é responsável por administrar as conexões entre os navegadores. É importante salientar que este servidor apenas serve de suporte para um navegador se conectar a outro, e para um navegador saber quando sua conexão com este outro navegador foi perdida. As mensagens enviadas não passam por este servidor, são enviadas diretamente de um navegador ao outro.

Como o jogo eletrônico estando online necessita da comunicação TCP e da comunicação UDP ativas para funcionar adequadamente, quando uma dessas não funciona corretamente ou está desconectada, o jogador é tido como desconectado e não pode interagir com os outros jogadores.

O método “Jogo.iniciarConexao” é utilizado para tornar o jogo eletrônico online. Esta função tentará iniciar a conexão com o servidor em Java e com o servidor utilizado pelo “PeerJS”. Ouvintes são adicionados nos dois objetos de conexão, permitindo que estes disparem funções de *callback* para determinados eventos. Quando as duas conexões são efetivadas, um evento extra criado pelo *framework* deste projeto é disparado: “conexaoIniciou”. Ele é útil para o desenvolvedor saber quando o jogo está completamente conectado. O método “Jogo.peerConectar” é utilizado para um navegador iniciar uma conexão UDP com outro navegador. Ouvintes são adicionados nesta conexão para eventos disparados caso a conexão funcione e caso ela não dê certo. Se a conexão é efetivada, um canal UDP é criado entre os dois navegadores, e novos ouvintes serão adicionados nesta conexão para que a troca de dados possa acontecer. Todos os eventos de rede, incluindo “conexaoIniciou” são propagados para o quadro atual, e deste quadro para a cena contida nele.

Servidor Principal

O servidor em Java mencionado anteriormente é o servidor principal do jogo eletrônico quando este está online. Ele é fundamental para a estrutura de salas pré-partida, pois é nele que todas as informações das salas ficam armazenadas. Antes de a partida iniciar, ele é utilizado constantemente, já que as informações enviadas neste período precisam chegar com segurança ao destino, e já que ainda não existem conexões diretas entre os navegadores.

Mesmo durante a partida, o servidor principal é utilizado diversas vezes, quando é necessário enviar mensagens com segurança total de um jogador a outro, a exemplo da mensagem que é enviada quando um item é criado na tela. Durante a partida o servidor funciona basicamente como um retransmissor de mensagens entre os jogadores. Como a estrutura das salas geralmente não muda no decorrer das partidas, o servidor não precisará atualizar suas informações.

Quando um jogador se conecta ao servidor, ambas as partes criam um temporizador de conexão. Estes temporizadores enviam mensagens “PING/PONG” em curtos intervalos de tempo, com o intuito de verificar se ainda existe um canal de comunicação ali. Caso o jogador

ou o servidor fique muito tempo sem receber mensagens um do outro, a conexão entre eles é encerrada.

6.8 Lógica

Toda a lógica do jogo eletrônico é executada em dois importantes ambientes. O primeiro diz respeito à função “logica” contida em todas as cenas do jogo. A função “logica” da cena atual é executada repetidamente até que uma nova cena entre em seu lugar. Aqui acontece todo o processamento que não está diretamente associado com a interação do jogador ou com outros eventos, como por exemplo, a aplicação de inteligência artificial nos NPCs. Em outras palavras, todo processamento que necessite ser executado constantemente em ciclo precisa estar dentro da função “logica”. O outro ambiente onde a lógica do jogo eletrônico é executada diz respeito às funções de *callback*, que são invocadas após eventos serem disparados. Por exemplo, o processamento decorrente do pressionamento de uma tecla em uma cena deve ocorrer dentro da função “keyDown” desta cena.

Nas cenas Preloader, Salas e Sala toda a lógica é executada nas funções de *callback* dos seus eventos. Isso acontece porque nestas cenas nenhum processamento é constantemente repetido, todos precisam de um gatilho para serem executados. Já na cena Partida, tanto processamentos repetitivos, quanto de eventos são bem utilizados.

6.8.1 Física

Dois elementos da física são constantemente aplicados na partida do jogo. São eles: movimentação e colisão.

O movimento dos elementos gráficos acontece a partir da alteração de suas coordenadas X e Y na cena, tendo como base suas velocidades e direções atuais.

Neste jogo eletrônico não existe a necessidade de colisões pixel a pixel serem verificadas entre os elementos. São empregadas apenas colisões geométricas, que são processadas de maneira bem mais veloz. As formas geométricas encontradas neste jogo eletrônico são: ponto, linha e círculo. Cálculos matemáticos são realizados de maneira direta para verificar a presença ou ausência de colisão, excluindo a necessidade de estruturas de repetição serem utilizadas.

Todos os métodos utilizados para verificar colisões estão localizados no objeto “Colisao”. Por exemplo, “Colisao.elementoGraficoComElementoGrafico(personagem, fragmento)” poderia ser executado para verificar se existe colisão entre um determinado personagem e um determinado fragmento.

6.9 Level Design

Por Bolita ser um jogo eletrônico de poucas cenas e sem muitas informações na tela, não foi necessária a utilização de qualquer software para a codificação do *level design*. Toda a organização dos elementos na tela foi realizada por programação em JavaScript. O *framework* deste projeto oferece recursos que agrupam as propriedades dos elementos gráficos em uma única estrutura, tornando os códigos desta etapa bastante compactos. No entanto, inicialmente, alguns esboços confeccionados através do software de modelagem de telas - Balsamiq Mockups foram necessários para uma melhor percepção das cenas.

6.10 Dados

O jogo contém um único arquivo HTML, denominado de “index.html”. Este representa a página web onde o jogo eletrônico será exibido, e é responsável por carregar todos os arquivos CSS e JavaScript do projeto.

Todos os arquivos CSS e JavaScript do jogo eletrônico estão localizados no diretório raiz, com exceção das bibliotecas auxiliares em JavaScript, que estão armazenadas na pasta “js”. Além desta pasta existem outras três: “imagens”, “audios” e “fontes”. Elas contêm, respectivamente, as imagens do jogo em PNG, os áudios dele em MP3, e as fontes dos seus textos em TTF. A Figura 6 mostra a árvore de diretórios do jogo.

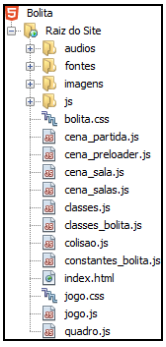

Figura 6: Árvore de diretórios do jogo eletrônico

Os arquivos visuais e de som são carregados na cena Preloader, que é a cena inicial do jogo eletrônico. A biblioteca PreloaderJS é utilizada para facilitar este procedimento, carregando vários arquivos ao mesmo tempo e fornecendo eventos que são disparados de acordo com o estado do carregamento, o que remove a necessidade de ter um código para cada arquivo individual. Todos os arquivos carregados aqui estarão disponíveis para as próximas cenas do jogo através do objeto global "Jogo.arquivosCarregados".

6.11 Software

Este jogo eletrônico faz uso de algumas pequenas bibliotecas auxiliares. É importante ressaltar que estas não alteram a proposta do *framework* principal de deixar evidente todas as estruturas elementares do HTML5. "Prototype" é utilizada para simular classes de maneira clara em javascript; "requestNextAnimation*Frame*" para prover uma abstração ao método que solicita um *frame* do navegador; "PreloaderJS" para o carregamento de arquivos; e PeerJS para a comunicação UDP.

Além destas bibliotecas, como já foi mencionado, o jogo eletrônico faz uso de um *framework* que foi desenvolvido pelo autor para uso neste projeto. Ele está anexado a este jogo, mas pode ser facilmente adaptado para a construção de qualquer outro jogo eletrônico em HTML5.

A modelagem do projeto foi realizada utilizando diagramas UML, confeccionados no software Astah. O mais importante destes diagramas foi o diagrama de classes, onde o jogo, e principalmente o *framework* puderam ser devidamente estruturados. Como ambiente de desenvolvimento, ou IDE, o Oracle NetBeans foi escolhido por seus recursos, que facilitam em muito o trabalho dos programadores.

6.12 Testes

Foram realizados longos testes em todo o jogo e todo o *framework* em busca de erros, problemas de desempenho e inadequações na jogabilidade. Apenas testadores humanos foram empregados, e estes utilizaram os seguintes navegadores para a execução dos códigos: Google Chrome, Mozilla Firefox, Opera, Google Chrome para Android e Mozilla Firefox para Android.

Após o jogo ter sido finalizado, 15 pessoas de diferentes idades e sexos foram convidadas a testá-lo, jogando-o por cerca de 10 minutos. Após este período de testes, os

usuários responderam uma avaliação que permitiu a eles dar notas de 0 a 10 às diversas características do jogo. Este formulário é encontrado no Apêndice C, e a média das notas fornecidas pelos jogadores para as 10 características apresentadas são exibidas na Tabela 1.

Característica Avaliada	**Média das Notas**
Arte gráfica	10.0
Sonoplastia	10.0
Interação antes da partida	9.1
Interação durante a partida	8.9
Jogabilidade	8.5
Criatividade	8.7
Carregamento dos arquivos	8.0
NPCs	9.7
Recomendação para amigos	8.3
Satisfação geral	8.2

Tabela 1: médias das notas dadas pelos jogadores para 10 características do jogo

6.13 Compatibilidade de Recursos

Bolita e o *framework* deste projeto fazem uso da WebRTC, que é uma tecnologia bastante recente do HTML5. Desta maneira, se um navegador tem suporte a este recurso, certamente ele terá suporte a todos os outros, e executará os códigos sem problemas. Entretanto, se um *browser* não oferece suporte à WebRTC ele muito provavelmente encontrará uma série de erros durante a execução.

Dos navegadores utilizados atualmente estes são comprovadamente compatíveis com Bolita e seu *framework*: Google Chrome, Mozilla Firefox, Opera, Android Browser a partir da versão 44, Google Chrome para Android, Mozilla Firefox para Android e Opera Mobile.

7 CONSIDERAÇÕES FINAIS

Após conhecer a fundo o HTML5 e todos os seus recursos úteis no desenvolvimento de jogos, foi possível entender porque esta tecnologia é tão importante. Mesmo ainda sendo consideravelmente nova, ela atendeu a todas as expectativas, fornecendo poderosas alternativas para todas as etapas do desenvolvimento de um jogo eletrônico.

Sua completude, sua facilidade de uso, seu suporte, sua padronização, seu desempenho e sua constante evolução nos navegadores fazem do HTML5 uma das principais tecnologias atuais para o desenvolvimento de jogos para Internet.

A única dificuldade encontrada pelo autor deste texto foi a falta de uma ferramenta de auxílio para o desenvolvimento do jogo. Isso é muito custoso e demanda bastante tempo, mas acaba sendo uma realidade independente da linguagem de programação que está sendo utilizada. Como o objetivo deste trabalho é apresentar a essência do HTML5 este método de desenvolvimento se fez necessário.

Para uma boa produtividade na criação de jogos com HTML5 é bastante válida a utilização de motores de jogos, bibliotecas, APIs ou *frameworks* que agilizem os processos de desenvolvimento. O *framework* confeccionado pelo autor deste texto pode ser utilizado para este propósito. Bolita foi construído em conjunto com ele, então muitas das funcionalidades deste *framework* não puderam de fato serem aproveitadas. Porém, com ele agora finalizado, novos jogos podem ser facilmente criados com sua utilização.

Como trabalhos futuros destacamos: realizar testes mais rígidos que validem por completo o *framework* desenvolvido; aperfeiçoar os protocolos de rede, reduzindo o tamanho dos pacotes e tornando a sincronização dos elementos na tela mais suave; implementar a possibilidade dos jogadores ampliarem a velocidade dos seus personagens através de simultâneos cliques na tela; unir o servidor Java ao servidor PeerJS; melhorar o desempenho do jogo eletrônico em dispositivos mais modestos; e ampliar a característica multiplataforma do jogo.

REFERÊNCIAS

ARRUDA, E. P. **Fundamentos Para o Desenvolvimento de Jogos Digitais - Série Tekne**. 1ª. ed. Porto Alegre - RS: Grupo A, 2013. 112p.

BONATTI, D. **Desenvolvimento de Jogos em HTML5**. 1ª. ed. São Paulo - SP: Brasport, 2014. 256p.

CHANDLER, H. M. **Manual de Produção de Jogos Digitais**. 2ª. ed. Porto Alegre - RS: Bookman, 2012. 508p.

JÚNIOR, J. F. C. **Ferramenta de Desenvolvimento Engine**. 1ª. ed. São Paulo - SP: Érica, 2015. 112p.

LAUREANO, M. **Máquinas Virtuais e Emuladores - Conceitos, Técnicas e Aplicações**. 1ª. ed. São Paulo - SP: Novatec, 2006. 184p.

MEYER, J. **O Guia Essencial do HTML5 - Usando Jogos para Aprender HTML5 e JavaScript**. 1ª. ed. Rio de Janeiro - RJ: Ciência Moderna, 2011. 408p.

RABIN, S. **Introdução ao Desenvolvimento de Jogos eletrônicos - Vol.3**. 1ª. ed. São Paulo - SP: Cengage Learning, 2013. 190p.

RABIN, S. **Introdução ao Desenvolvimento de Jogos eletrônicos - Vol.2**. 1ª. ed. São Paulo - SP: Cengage Learning, 2013. 520p.

RABIN, S. **Introdução ao Desenvolvimento de Jogos eletrônicos - Vol.4**. 1ª. ed. São Paulo - SP: Cengage Learning, 2013. 180p.

RABIN, S. **Introdução ao Desenvolvimento de Jogos eletrônicos - Vol.1**. 1ª. ed. São Paulo - SP: Cengage Learning, 2012. 192p.

SCHUYTEMA, P. **Design de Jogos eletrônicos - Uma Abordagem Prática**. 1ª. ed. São Paulo - SP: Thomson Learning, 2008. 472p.

AHISTORIA. **História do Videogame e Jogos Eletrônicos - A História**. Disponível em: <http://www.ahistoria.com.br/videogame-e-jogos-eletronicos/> Acesso em: 16 nov. 2015

AHISTORIA**. História dos Jogos de Computador - A História**. Disponível em: <http://www.ahistoria.com.br/jogos-de-computador/> Acesso em: 16 nov. 2015

AHISTORIA. **História dos Jogos e Videogames Portáteis - A História**. Disponível em: <http://www.ahistoria.com.br/historia-dos-jogos-e-videogames-portateis/> Acesso em: 16 nov. 2015

AIDOBONSAI. **A História dos video jogos eletrônicos. | Aido Bonsai**. Disponível em: < http://aidobonsai.com/2010/07/05/a-historia-dos-video-jogos eletrônicos-2/> Acesso em: 16 nov. 2015

BATTAIOLA, A. L. **Aspectos Fundamentais da Criação de Jogos em Shockwave 3D**. Disponível em: <http://docplayer.com.br/4600315-Aspectos-fundamentais-da-criacao-de-jogos-em-shockwave-3d.html> Acesso em: 16 nov. 2015

BARRA, V. M.; CARNEIRO, S. M. M.; LEME, S. E. G.; OTA, S. N. **Jogo: estratégia eficiente para a educação ambiental.** Curitiba: Universidade Livre do Meio Ambiente, 1996. Disponível em: <http://www.utfpr.edu.br/curitiba/estrutura-universitaria/diretorias/dirppg/grupos/tema/28jogo_estrat_educacaoambiental.pdf> Acesso em: 16 nov. 2015

BELEM, T. **Armazenando informações com localStorage e sessionStorage - Thiago Belem / Blog**. Disponível em: <http://blog.thiagobelem.net/armazenando-informacoes-no-computador-do-visitante-com-localstorage-e-sessionstorage/> Acesso em: 16 nov. 2015

BIDELMAN, E. **Capturing Audio & Video in HTML5 - HTML5 Rocks**. Disponível em: <http://www.html5rocks.com/pt/tutorials/getusermedia/intro/> Acesso em: 16 nov. 2015

BIDELMAN, E. **Conceitos básicos sobre Web Workers - HTML5 Rocks**. Disponível em: <http://www.html5rocks.com/pt/tutorials/workers/basics/> Acesso em: 16 nov. 2015

BIDELMAN, E**. Novos truques em XMLHttpRequest2 - HTML5 Rocks**. Disponível em: <http://www.html5rocks.com/pt/tutorials/file/xhr2/> Acesso em: 16 nov. 2015

CANIUSE. **Can I use... Support tables for HTML5, CSS3, etc**. Disponível em: <http://caniuse.com/> Acesso em: 16 nov. 2015

CAVALCANTI, D. **Existem vantagens em criar jogos para a web? | Virtualidade Latente**. Disponível em: <https://virtualidadelatente.wordpress.com/2012/12/03/existem-vantagens-em-criar-jogos-para-a-web/> Acesso em: 16 nov. 2015

CREATEJS. **PreloadJS v0.6.1 API Documentation : PreloadJS**. Disponível em: <http://www.createjs.com/docs/preloadjs/modules/PreloadJS.html> Acesso em: 16 nov. 2015

DEMONWEB. **Uma Breve História dos Jogos | DemonWeb**. Disponível em: <https://demonweb.wordpress.com/2008/06/18/uma-breve-historia-dos-jogos/> Acesso em: 16 nov. 2015

DUTTON, S. **Getting Started with WebRTC - HTML5 Rocks**. Disponível em: <http://www.html5rocks.com/en/tutorials/webrtc/basics/> Acesso em: 16 nov. 2015

FILHO, A. **Acelerômetro do Mallandro, Glu glu ié ié! – Loop Infinito**. Disponível em: <http://loopinfinito.com.br/2012/10/10/acelerometro-do-mallandro-glu-glu-ie-ie/> Acesso em: 16 nov. 2015

FILHO, A. **Veja mais com a Fullscreen API – Loop Infinito**. Disponível em: <http://loopinfinito.com.br/2012/11/27/veja-mais-com-a-fullscreen-api/> Acesso em: 16 nov. 2015

FLIPER MANIA. **Tudo sobre jogos de Fliperama: História dos Fliperamas**. Disponível em: <http://fliper-mania.blogspot.com.br/p/historia-dos-fliperamas.html> Acesso em: 16 nov.

2015

GASPAROTTO, H. M. **Unity 3D: Introdução ao desenvolvimento de jogos eletrônicos**. Disponível em: < http://www.devmedia.com.br/unity-3d-introducao-ao-desenvolvimento-de-jogos eletrônicos/30653> Acesso em: 16 nov. 2015

GARSIEL, T.; IRISH, P. **Como os navegadores funcionam: nos bastidores dos navegadores web modernos - HTML5 Rocks**. Disponível em: <http://www.html5rocks.com/pt/tutorials/internals/how*browsers*work/> Acesso em: 16 nov. 2015

HAMANN, R. **A evolução dos computadores - TecMundo**. Disponível em: <http://www.tecmundo.com.br/infografico/9421-a-evolucao-dos-computadores.htm> Acesso em: 16 nov. 2015

HTML5GAMEENGINE. **HTML5 Jogo eletrônico Engines - Find Which is Right For You**. Disponível em: <http://html5gameengine.com/> Acesso em: 16 nov. 2015

JORDÃO, F. **A história dos sistemas operacionais [infográfico] - TecMundo**. Disponível em: <http://www.tecmundo.com.br/sistema-operacional/2031-a-historia-dos-sistemas-operacionais-ilustracao-.htm> Acesso em: 16 nov. 2015

JUNIOR, A. **História dos Jogos Online**. Disponível em: <http://www.artigonal.com/jogos-artigos/historia-dos-jogos-online-2580716.html> Acesso em: 16 nov. 2015

KARASINSKI, E. **HTML5 x Flash: uma guerra iminente? - TecMundo**. Disponível em: <http://www.tecmundo.com.br/navegador/3608-html5-x-flash-uma-guerra-iminente-.htm> Acesso em: 16 nov. 2015

MARQUEZ, M. **Mercado de jogos eletrônicos já movimenta quase R$ 1 bilhão por ano no Brasil - Tecnologia e Ciência - R7**. Disponível em: <http://noticias.r7.com/tecnologia-e-ciencia/noticias/mercado-de-jogos-eletronicos-ja-movimenta-quase-r-1-bilhao-por-ano-no-brasil-20121103.html?question=0> Acesso em: 16 nov. 2015

MEDEIROS, H. **Java WebSockets: Introdução**. Disponível em: <http://www.devmedia.com.br/java-websockets-introducao/30443> Acesso em: 16 nov. 2015

MOON, S. **Make a simple html5 jogo eletrônico with box2d in javascript - tutorial**. Disponível em: <http://www.binarytides.com/make-html5-jogo eletrônico-box2d-javascript-tutorial/> Acesso em: 16 nov. 2015

NETO, A. C. D. **Artigo Engenharia de Software - Introdução a Teste de Software**. Disponível em: <http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035> Acesso em: 16 nov. 2015

NÖRNBERG, L. **HTML 5 - futuro dos jogos para web? | Abrindo o Jogo**. Disponível em: <http://abrindoojogo.com.br/html-5-futuro-dos-jogos-para-web> Acesso em: 16 nov. 2015

NORVIG, P.; THALMANN, N.; LATHAM, W**. Inteligencia Artificial - Desenvolvimento de Jogos**. Disponível em: <http://desenvolvimentodejogos.wikidot.com/inteligencia-

artificial> Acesso em: 16 nov. 2015

NUNES, DOUGLAS. **Mercado de jogos eletrônicos explode no Brasil - Empresas - iG**. Disponível em: <http://economia.ig.com.br/empresas/2013-12-26/mercado-de-jogos-eletronicos-explode-no-brasil.html> Acesso em: 16 nov. 2015

OBOMPASTOR. **obompastor**. Disponível em: < http://www.obompastor.com.br/cbp/wp-content/uploads/2010/03/Jogos.pdf> Acesso em: 16 nov. 2015

OLIVEIRA, L. S. **Gaveta de Bagunças - Atom**. Disponível em: < http://gavetadbagunca.blogspot.com.br/2014/07/a-origem-dos-jogos-eletronicos-primeira.html> Acesso em: 16 nov. 2015

PAGANI, T. **Desenvolvimento de Jogos com HTML5**. Disponível em: <http://pt.slideshare.net/talitapagani/html5-gamesusc> Acesso em: 16 nov. 2015

PEERJS. **PeerJS - Simple peer-to-peer with WebRTC**. Disponível em: <http://peerjs.com/> Acesso em: 16 nov. 2015

PIXSTUDIOS. **O que são as Jogo eletrônico Engines ou Motores de Jogos?**. Disponível em: <http://www.pixstudios.com.br/blog/novidades-de-computacao-grafica-e-jogos eletrônicos/o-que-sao-engine-de-jogos eletrônicos-ou-motor-de-jogo> Acesso em: 16 nov. 2015

PRADA, R. **WebGL, Flash ou Unity: quem dominará os jogos online em 3D? - TecMundo**. Disponível em: <http://www.tecmundo.com.br/video-jogo eletrônico-e-jogos/6832-webgl-flash-ou-unity-quem-dominara-os-jogos-online-em-3d-.htm> Acesso em: 16 nov. 2015

PROTOTYPEJS. **Prototype JavaScript *Framework* | Defining classes and inheritance**. Disponível em: <http://prototypejs.org/learn/class-inheritance.html> Acesso em: 16 nov. 2015

QUETANTO. **Vantagens e desvantagens de jogos on-line**. Disponível em: <http://quetanto.com/internet/web/3/vantagens-e-desvantagens-de-jogos-on-line.php> Acesso em: 16 nov. 2015

ROUSSET, D. **HTML5 - O guia completo para a criação de jogos HTML5 com Canvas e SVG**. Disponível em: <https://msdn.microsoft.com/pt-br/library/jj937876.aspx> Acesso em: 16 nov. 2015

SATO, A. T. **Artigo .net magazine 70 - Jogos eletrônicos em Silverlight**. Disponível em: < http://www.devmedia.com.br/artigo-net-magazine-70-jogos eletrônicos-em-silverlight/15802> Acesso em: 16 nov. 2015

SITES GOOGLE. **historia dos navegadores de internet**. Disponível em: <https://sites.google.com/site/historiasobreossitesdebusca/historia-dos-navegadores-de-internet> Acesso em: 16 nov. 2015

SMUS, B. **Como desenvolver para navegadores da web multitoque - HTML5 Rocks**. Disponível em: <http://www.html5rocks.com/pt/mobile/touch/> Acesso em: 16 nov. 2015

SMUS, B**. Developing Jogo eletrônico Audio with the Web Audio API - HTML5 Rocks**. Disponível em: <http://www.html5rocks.com/en/tutorials/webaudio/jogos eletrônicos/> Acesso em: 16 nov. 2015

SMUS, B. **Getting Started with Web Audio API - HTML5 Rocks**. Disponível em: <http://www.html5rocks.com/en/tutorials/webaudio/intro/> Acesso em: 16 nov. 2015

TAVARES, G. **WebGL Fundamentals - HTML5 Rocks**. Disponível em: <http://www.html5rocks.com/en/tutorials/webgl/webgl_fundamentals/> Acesso em: 16 nov. 2015

TECMUNDO. **Qual é o principal problema na classificação de jogos eletrônicos por gênero? - TecMundo Gamess**. Disponível em: < http://jogos eletrônicos.tecmundo.com.br/especiais/qual-e-o-principal-problema-na-classificacao-de-jogos eletrônicos-por-genero_165536.htm> Acesso em: 16 nov. 2015

TOSCHI, G. **Ação, RPG, Plataforma? Saiba mais sobre os gêneros de jogos e quais jogos eletrônicos se classificam nos mais conhecidos - Parte 1 - Nintendo Blast**. Disponível em: <http://www.nintendoblast.com.br/2012/04/gamedev-conheca-os-tipos-de-jogos.html> Acesso em: 16 nov. 2015

UBL, M.; KITAMURA, E. **Apresentando WebSocket: trazendo soquetes para a web - HTML5 Rocks**. Disponível em: <http://www.html5rocks.com/pt/tutorials/websockets/basics/> Acesso em: 16 nov. 2015

VELOSO, R. R. **Reflexões sobre o jogo: conceitos, definições e possibilidades**. Disponível em: < http://www.efdeportes.com/efd132/reflexoes-sobre-o-jogo.htm> Acesso em: 16 nov. 2015

VIEIRA, E. **Inteligência Artificial nos Jogos eletrônicos – Mais Aplicações | Abrindo o Jogo**. Disponível em: <http://abrindoojogo.com.br/inteligencia-artificial-nos-jogos eletrônicos-%E2%80%93-mais-aplicacoes> Acesso em: 16 nov. 2015

W3SCHOOLS. **W3Schools Online Web Tutorials**. Disponível em: <http://www.w3schools.com/> Acesso em: 16 nov. 2015

WIKIPEDIA. **Jogo – Wikipédia, a enciclopédia livre**. Disponível em: < https://pt.wikipedia.org/wiki/Jogo> Acesso em: 16 nov. 2015

WIMI5. **The Advantages of HTML5 Jogos eletrônicos in Advergaming**. Disponível em: < http://wimi5.com/advantages-html5-jogos eletrônicos-advergaming-campaign/> Acesso em: 16 nov. 2015

APÊNDICE A – Termos Comuns Utilizados Na Definição Dos Gêneros Dos Jogos

Ação: são jogos onde o personagem é constantemente desafiado, seja através de combates ou de armadilhas, e o jogador deve tomar rápidas decisões se quiser prosseguir no jogo eletrônico.

Advergame: são jogos que possuem o intuito de fazer propaganda sobre alguma marca, produto ou serviço.

Aventura: são jogos que possuem o enredo como principal fator, e não a ação.

Baseado em Obra: são jogos cujo enredo é baseado em livros, filmes, séries, desenhos animados ou animes.

Beat 'em ups: são jogos de luta em que os personagens enfrentam constantemente uma série de adversários, ao mesmo tempo em que vão seguindo linearmente pelo mapa.

Casual: são jogos simples e fáceis de aprender, e que normalmente não levam mais que poucos minutos para serem concluídos. A grande maioria deles apresenta um sistema de ranking.

Corrida: são jogos em que os jogadores normalmente controlam algum tipo de meio de transporte, e devem utilizá-lo para vencer seus adversários em corridas.

Educacional: são jogos que possuem como foco ensinar, testar ou aprimorar certas áreas do conhecimento em seus jogadores.

Eroge: são jogos com temática erótica, designados aos adultos.

Esporte: são jogos baseados em esportes reais, tais como futebol, tênis e natação.

Estratégia: são jogos em que para vencer, os jogadores devem utilizar seu raciocínio acima de qualquer outra de suas perícias.

Fantasia: são jogos em que o enredo se passa em cenários fantasiosos, místicos, e completamente fora da nossa realidade. Magia e elementos da idade média são muito comuns nestes jogos eletrônicos.

Feminino: são jogos voltados ao público feminino, geralmente contendo elementos de estética e moda.

Ficção Científica: são jogos cujos elementos principais são robôs, alienígenas, viagens espaciais, viagens temporais, ou outros componentes relacionados com a ficção científica.

FPA (First-Person Adventure): são jogos de aventura em que o jogador enxerga o mundo através dos olhos do personagem principal.

FPS (First-Person Shooter): são jogos de ação, e muito provavelmente de tiro, em que o jogador enxerga o mundo através dos olhos do personagem principal.

Infantil: são jogos destinados a crianças, e portanto não possuem elementos pesados, como violência e morte.

Luta: são jogos cujo objetivo principal é vencer um ou mais adversários em combates.

Minigames: são jogos cujo foco está na presença de outros pequenos jogos eletrônicos dentro de suas estruturas. Estes últimos podem ser completamente diferentes entre si, mas sempre estarão unidos por alguma característica em comum.

Mistério: são jogos em que grande parte do enredo é inicialmente desconhecido, e os jogadores possuem como principal objetivo desvendar estes mistérios.

Mitológico: são jogos cujo enredo possui fortes elementos de uma ou mais mitologias, como a mitologia grega, por exemplo.

MMOG (Massive Multiplayer Online Jogo eletrônico): são jogos que permitem a interação simultânea de vários jogadores através de uma rede.

MMORPG (Massive Multiplayer Online Role-Playing Jogo eletrônico): são jogos do gênero RPG que permitem a interação simultânea de vários jogadores através de uma rede.

MMOSG (Massively Multiplayer Online Social Jogo eletrônico): são jogos que permitem a interação simultânea de vários jogadores através de uma rede, e possuem como principal objetivo promover a socialização entre as pessoas ali presentes.

Mundo Aberto: são jogos onde o personagem tem liberdade para ir a vários lugares diferentes, tornando suas possibilidades de interação infinitamente maiores.

Musical: são jogos onde o elemento principal é a música.

Plataforma: são jogos onde os personagens saltam constantemente entre plataformas e outros elementos de cenário, ao mesmo tempo em que combatem seus adversários e coletam itens de bônus.

Puzzle: são jogos simples onde os jogadores utilizam do seu raciocínio para vencerem algum pequeno desafio, que normalmente se agrava com o tempo.

RPG (Role-Playing Jogo eletrônico): são jogos de mundo aberto em que o personagem principal segue em várias missões, interagindo com personagens não-jogadores, enfrentando adversários, adquirindo itens e aprimorando suas habilidades.

RTS (Real Time Strategy): são jogos de estratégia em que as decisões e ações devem ser realizadas em tempo real, pois não existe turnos destinados a cada jogador, tudo acontece simultaneamente.

Sandbox: são jogos que possuem várias maneiras diferentes de serem completados, várias alternativas, e não apenas um único caminho como na maioria dos jogos eletrônicos.

Saúde: são jogos utilizados na área da saúde, geralmente criados para aprimorar o tratamento

de algum problema físico ou psicológico de um paciente.

Shoot'em up: são jogos onde os jogadores normalmente controlam naves espaciais, ou outros elementos associados ao espaço, e devem atirar a todo instante para destruir ameaças que surgem das extremidades da tela.

Simulação: são jogos que normalmente simulam a construção e administração de negócios, cidades ou governos, ou que simulam a vida de pessoas ou animais de estimação.

Stealth: são jogos onde a prioridade do jogador é fazer com que seu personagem aja furtivamente para atingir seus objetivos.

Suspense: são jogos com alto teor de suspense, onde o jogador é constantemente surpreendido.

Survivor Horror: são jogos em que os personagens principais devem sobreviver e combater constantes ameças sobrenaturais, tais como zumbis e outras criaturas. Geralmente o ambiente e os protagonistas simulam a realidade.

TBS (Turn-Based Strategy): são jogos de estratégia onde os acontecimentos são divididos em turnos, normalmente um para cada participante.

Tiro: são jogos em que os personagens principais combatem os seus adversários normalmente utilizando armas de fogo.

TPS (Third-Person Shooter): são jogos em que o personagem é visto através de uma perspectiva em terceira pessoa.

Violento: são jogos onde sangue, mutilação, tortura e elementos semelhantes são comuns.

APÊNDICE B – Game Design de Bolita

1 Geral

Bolita é um jogo casual para web, que pode ser jogado tanto ao lado de jogadores artificiais (NPCs), quanto ao lado de outros jogadores reais, utilizando para isso a Internet.

Neste jogo eletrônico, cada jogador controla seu próprio e único personagem, que se trata de um objeto circular em constante movimento. Não é possível pará-lo, apenas alterar a sua direção e velocidade. Os jogadores devem ficar atentos, pois quando seus personagens colidem com outros personagens ou com as extremidades do cenário, eles perdem uma porção de sua vida, que está diretamente associada ao tamanho do seu círculo. Além disso, os personagens podem liberar pedaços de si em diferentes tamanhos, com vida igual à vida que o personagem em questão perdeu pela divisão. Estes fragmentos também possuem movimento constante, e perdem vida ao colidirem com as extremidades da cena. Entretanto, assim como os personagens, suas direções também podem ser alteradas. Estas partes dos personagens são mais lentas, porém funcionam como bombas. Ao colidirem com personagens rivais, são destruídas, mas causam um dano considerável neles. Além disso, diferentes itens surgem na tela conforme o andamento do jogo. Estes, ao colidirem com os personagens ou com seus fragmentos, podem lhes oferecer benefícios, como também desvantagens. Os jogadores são divididos em equipes, podendo uma equipe ter de um a três integrantes. A última equipe sobrevivente será a vencedora da partida.

Para se darem bem no jogo, os jogadores devem ter um raciocínio bastante ágil, pois eles precisarão ficar atentos aos seus personagens e a todas as suas partes o tempo inteiro, já que estes não param e podem sofrer colisões indesejadas. Além disso, se quiserem vencer a partida, deverão encontrar uma boa maneira de destruir os seus adversários.

Este jogo eletrônico possui um sistema de salas que permite aos jogadores encontrarem seus amigos, ou conhecerem pessoas do mundo todo antes de iniciarem a partida propriamente dita. Nestas salas é possível conversar com outros jogadores ali presentes, além de definir diversas opções da partida e dos personagens.

A jogabilidade de Bolita, somada à sua característica *multiplayer* online, traz um diferencial muito importante para este jogo.

1.1 Arte gráfica

Bolita possui uma arte gráfica simples, e ao mesmo tempo bastante sofisticada. Seus traços possuem um estilo infantil, com fontes expressivas, formas arredondadas e muitas cores; aspectos comumente encontrados em jogos casuais de sucesso.

Este jogo eletrônico pode ser executado em modo de tela cheia na maioria dos dispositivos, o que amplia consideravelmente a imersão dos jogadores.

Ao abrir uma janela interna dentro de uma tela, toda a interface desta tela é escurecida dando foco à janela. Algumas podem ser encerradas quando o jogador pressiona o apontador fora delas, porém outras apenas por alguma interação dentro dessas janelas, ou quando o jogo finaliza algum processamento específico.

1.2 Sonoplastia

O áudio do jogo acompanha seu visual casual, com músicas instrumentais alegres e efeitos sonoros graciosos.

As músicas são reproduzidas assim que as telas são iniciadas, e repetem sua execução infinitamente. Cada cena possui uma melodia diferente.

A GUI reproduz sons específicos para as seguintes ações:

- Quando o jogo perde sua conexão com o servidor.
- Quando um botão padrão é pressionado ou selecionado.
- Quando uma janela informando um problema é aberta.

1.3 Interação

Basicamente, toda a interação dos jogadores é realizada através de apontadores: mouses em dispositivos desktop e dedos em dispositivos com telas sensíveis ao toque. Apenas quando textos precisam ser inseridos, como numa mensagem a ser enviada em uma sala, é que um teclado também deverá ser utilizado em equipamentos desktop.

1.4 Rede

Os jogadores podem interagir através da Internet com outros jogadores do mundo inteiro. A partir de dois tipos de conexões, uma com intermédio de um servidor e outra direta entre eles,

os dados trafegam com alta velocidade e segurança, permitindo uma jogabilidade em tempo real.

2 Telas

2.1 Preloader

Esta é a tela inicial do jogo (Figura 1), responsável por carregar todos os arquivos que serão utilizados nas telas seguintes e apresentar aos jogadores um feedback do quanto já foi e do quanto ainda resta a ser carregado.

2.1.1 Elementos

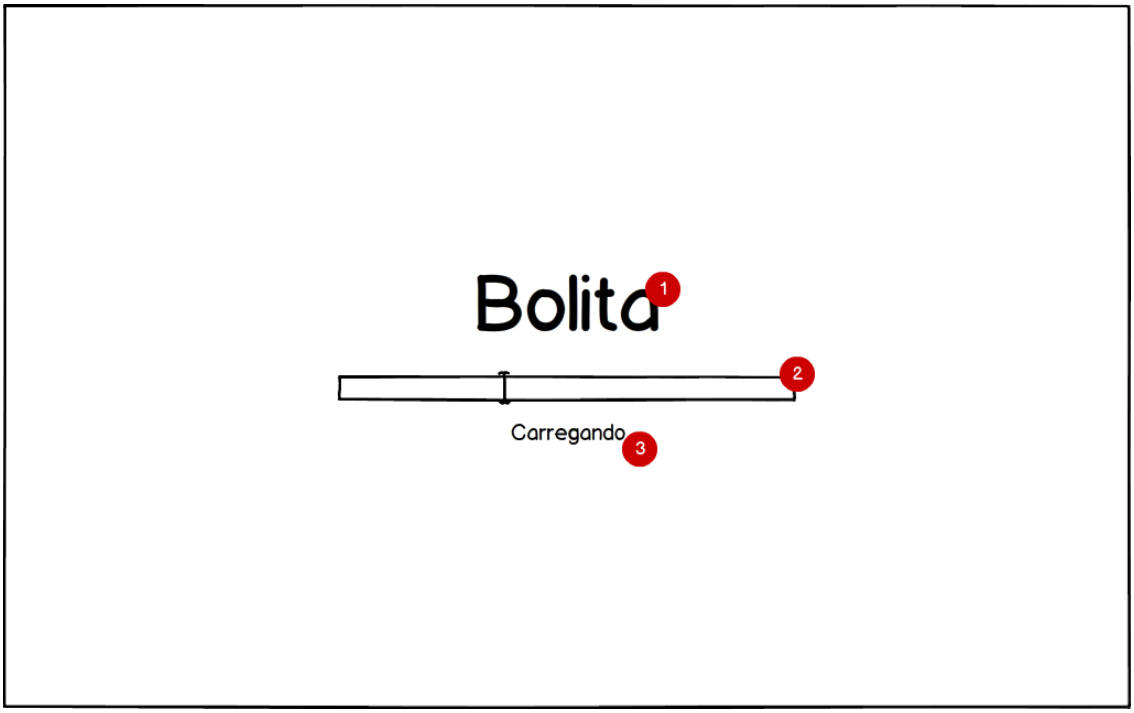

Figura 1: Tela “Preloader”

(1) Texto com o nome do jogo.

(2) Componente com uma agulha que se move para a direita conforme os dados vão sendo recebidos. Indica aos jogadores o quanto já foi e o quanto ainda resta ser carregado.

(3) Texto que indica aos jogadores que os dados estão sendo carregados.

2.1.2 Lógica / Fluxo

Após todos os dados das telas seguintes terem sido carregados, automaticamente o jogo seguirá para a tela Salas.

2.2 Salas

Esta tela (Figura 2) possui como principal função exibir aos jogadores as salas criadas pelos outros jogadores e que estão atualmente abertas. Eles poderão entrar em uma destas salas, criar a sua própria sala, ou optar por jogar off-line apenas com NPCs.

2.2.1 Elementos

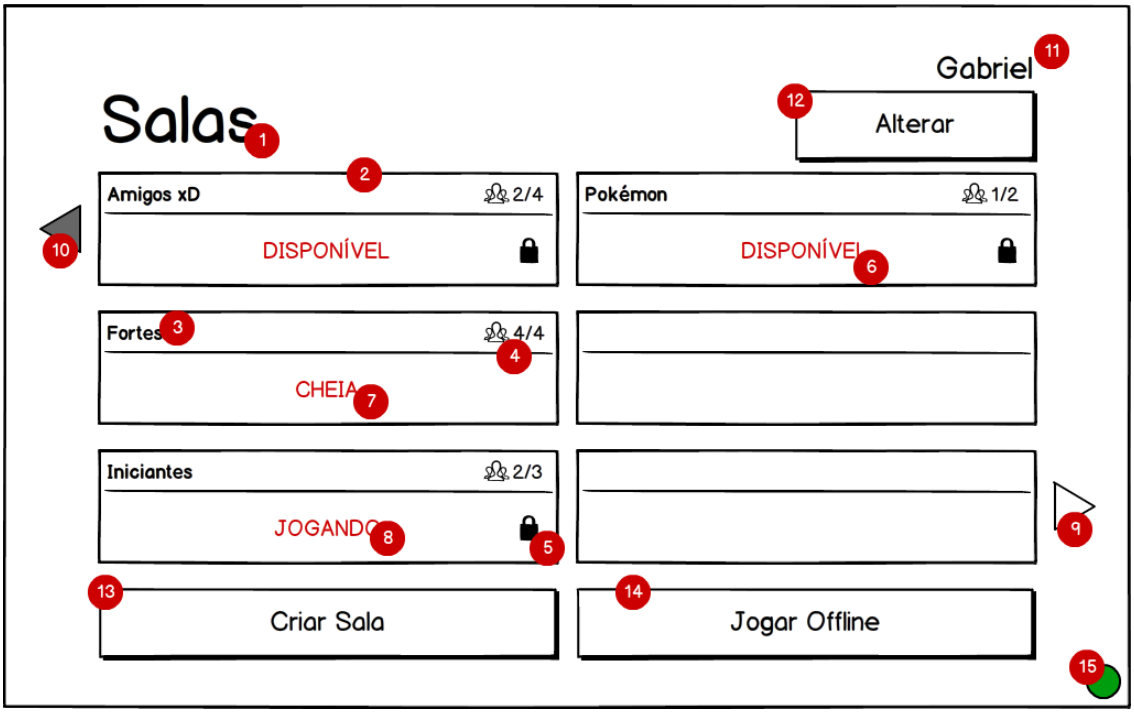

Figura 2: Tela “Salas”

(1) Texto que indica aos jogadores que eles estão na tela Salas.

(2) Componente que representa uma sala atualmente aberta, apresentando suas principais informações.

Se o jogador pressionar este componente duas vezes consecutivas em dispositivos desktop, ou uma única vez em dispositivos com telas sensíveis ao toque, e a sala estiver

disponível e destrancada, uma janela com uma mensagem de alerta informará ao jogador que ele está entrando naquela sala. Caso a sala esteja disponível, porém bloqueada, uma janela solicitando a senha dela será aberta.

(3) Texto que informa o nome daquela sala.

(4) Componente contendo um texto que informa a quantidade de jogadores presentes naquela sala, assim como a quantidade máxima de jogadores permitida ali.

(5) Ícone que indica que aquela sala possui senha.

(6) Texto que informa que o jogador pode entrar naquela sala.

(7) Texto que informa que aquela sala já atingiu sua quantidade máxima de jogadores.

(8) Texto que informa que a partida naquela sala já começou.

(9) Botão que quando pressionado mostra as salas que estão à direita das salas exibidas no momento. Este botão será desabilitado caso não existam salas à direita.

(10) Botão que quando pressionado mostra as salas que estão à esquerda das salas exibidas no momento. Este botão será desabilitado caso não existam salas à esquerda.

(11) Texto que informa o apelido do jogador.

(12) Botão que quando pressionado abre uma pequena janela onde o jogador poderá inserir um novo apelido para ele.

(13) Botão que quando pressionado abre uma pequena janela onde o jogador poderá fornecer as informações da sala que deseja criar. Este botão será desabilitado caso o jogo não esteja conectado ao servidor.

(14) Botão que quando pressionado abre uma pequena janela onde o jogador poderá indicar a quantidade de jogadores da partida.

(15) Ícone que indica o estado da conexão do jogo com o servidor. Se estiver verde o jogo está conectado. Se estiver vermelho ele está desconectado.

2.2.1.1 Janelas

Inserir apelido

Esta janela (Figura 3) é aberta quando a tela é iniciada pela primeira vez. Nesta ocasião ela não pode ser encerrada pressionando o apontador fora dela, porém em qualquer outra situação isso é possível de ser feito.

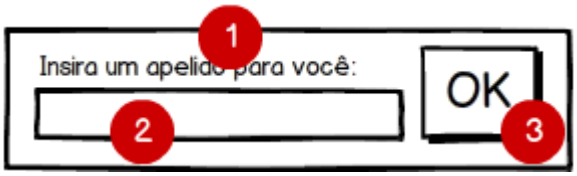

Figura 3: Janela “Inserir apelido”

(1) Texto que pede ao jogador para inserir um apelido.

(2) Caixa de texto onde o jogador deve inserir o seu apelido. Esta não pode estar vazia e nem conter o apelido atual.

(3) Botão que ao ser pressionado confirma a alteração do apelido do jogador e fecha a janela.

Criar sala

Esta janela (Figura 4) é encerrada quando o jogador pressiona o apontador fora dela, ou quando o jogo perde sua conexão com o servidor.

Figura 4: Janela “Criar sala”

(1) Texto informando que o jogador deve inserir o nome da sala no campo de texto logo abaixo.

(2) Caixa de texto onde o jogador deve inserir o nome da sala. Esta não pode estar vazia.

(3) Texto informando que o jogador deve inserir a senha da sala no campo de texto logo abaixo.

(4) Caixa de texto onde o jogador deve inserir a senha da sala. Esta é opcional.

(5) Texto informando que o jogador deve escolher a quantidade máxima de jogadores da sala dentre as opções logo abaixo.

(6) Botões onde o jogador deve selecionar apenas um deles para definir a quantidade máxima de jogadores da sala.

(7) Botão que ao ser pressionado encerra esta janela e abre uma outra que informa ao jogador que a sala está sendo criada.

Criando a sala

Esta janela (Figura 5) apenas será encerrada caso o jogo perca sua conexão com o servidor. Agora se tudo ocorrer bem o jogador será automaticamente levado até a tela que representa a sala que está sendo criada.

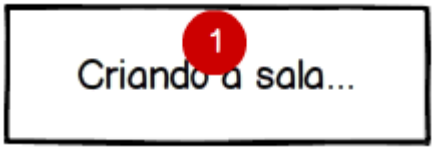

Figura 5: Janela “Criando a sala”

(1) Texto informando que o jogador deve aguardar até que a sala seja criada.

Inserir quantidade de jogadores

Esta janela (Figura 6) é encerrada quando o jogador pressiona o apontador fora dela.

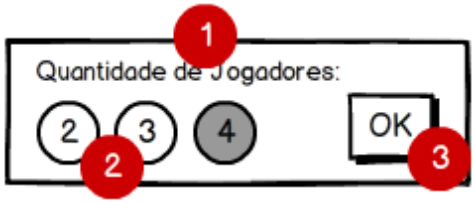

Figura 6: Janela “Inserir quantidade de jogadores”

(1) Texto informando que o jogador deve escolher a quantidade de jogadores da sala dentre as opções logo abaixo.

(2) Botões onde o jogador deve selecionar apenas um deles para definir a quantidade de jogadores da sala.

(3) Botão que ao ser pressionado leva o jogador para uma sala off-line onde ele poderá configurar a partida que pretende jogar com os NPCs.

Inserir a senha da sala

Esta janela (Figura 7) é encerrada quando o jogador pressiona o apontador fora dela ou quando o jogo perde sua conexão com o servidor.

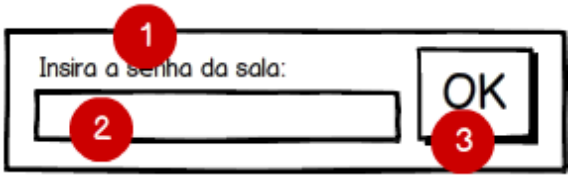

Figura 7: Janela “Inserir a senha da sala”

(1) Texto que pede ao jogador para inserir a senha da sala.

(2) Caixa de texto onde o jogador deve inserir a senha da sala. Esta não pode estar vazia.

(3) Botão que ao ser pressionado encerra esta janela e abre uma outra que informa ao jogador que ele está entrando na sala.

Entrando na sala

Esta janela (Figura 8) apenas será encerrada caso o jogo perca sua conexão com o servidor, ou exista algum fator que não permita ao jogador entrar na sala. Neste último caso uma janela correspondente ao erro será aberta. Agora, se tudo ocorrer bem, o jogador será automaticamente levado até a tela que representa esta sala.

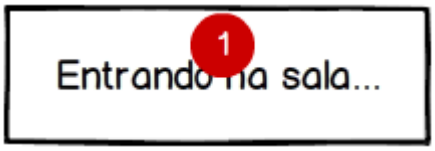

Figura 8: Janela “Entrando na sala”

(1) Texto informando que o jogador deve aguardar até que ele entre na sala.

Sala inexistente

Esta janela (Figura 9) é encerrada quando o jogador pressiona o apontador fora dela.

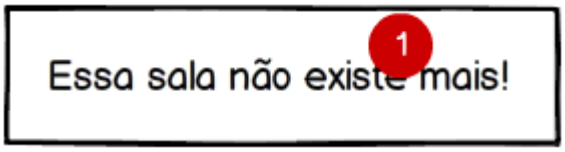

Figura 9: Janela “Sala inexistente”

(1) Texto informando ao jogador que a sala que ele tentou entrar não existe mais.

Senha da sala incorreta

Esta janela (Figura 10) é encerrada quando o jogador pressiona o apontador fora dela.

Figura 10: Janela “Senha da sala incorreta”

(1) Texto informando ao jogador que a senha da sala que ele inseriu está incorreta.

Sala cheia

Esta janela (Figura 11) é encerrada quando o jogador pressiona o apontador fora dela.

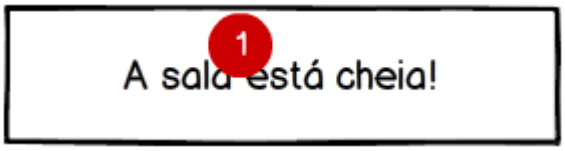

Figura 11: Janela “Sala cheia”

(1) Texto informando ao jogador que a sala em que ele tentou entrar agora está cheia.

Partida já começou

Esta janela (Figura 12) é encerrada quando o jogador pressiona o apontador fora dela.

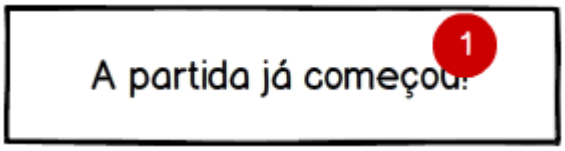

Figura 12: Janela “Partida já começou”

(1) Texto informando ao jogador que a partida da sala em que ele tentou entrar já iniciou.

2.2.2 Sonoplastia

Um efeito sonoro específico é reproduzido na seguinte situação:
- Quando o jogo consegue se conectar ao servidor.

2.3 Sala

Esta tela (Figura 13) representa a sala na qual o jogador entrou, independente desta ter sido criada por ele, ou não. Aqui, dentre outras pequenas coisas, são mostrados todos os jogadores presentes, suas informações pessoais e de jogo (que em geral podem ser modificadas), um chat de comunicação, e diversas opções para configurar a partida a gosto dos jogadores.

2.3.1 Elementos

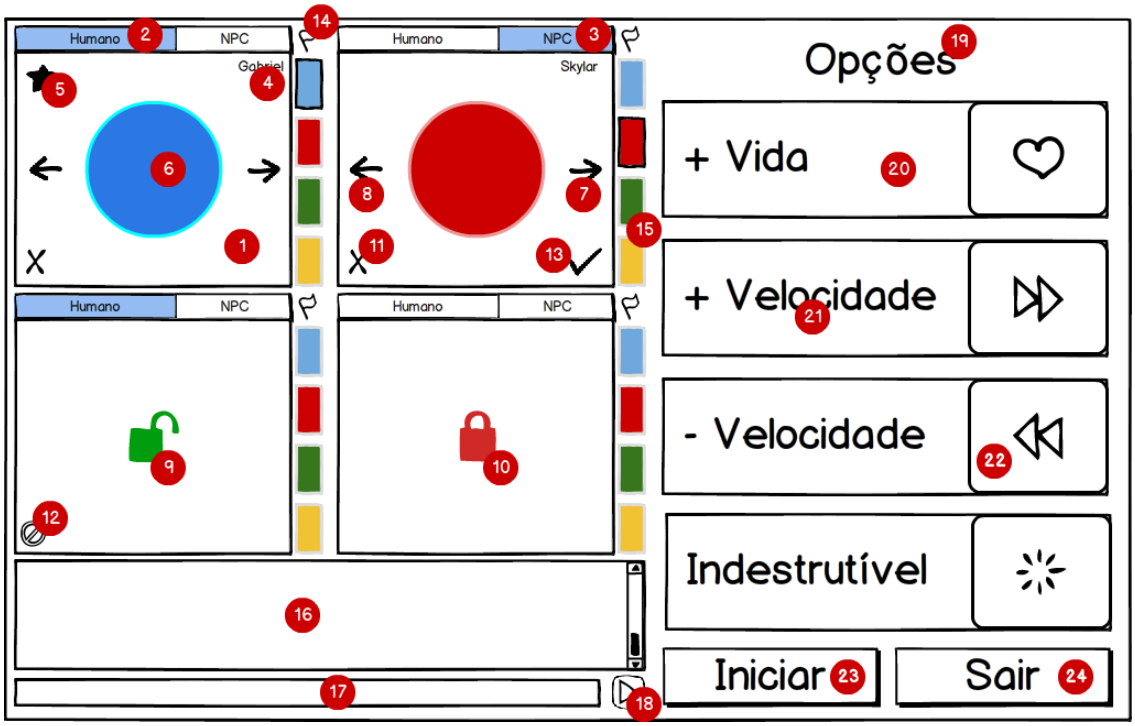

Figura 13: Tela “Sala”

(1) Componente que representa um slot da sala. Este slot pode estar liberado, bloqueado, ocupado por um jogador humano, ou ocupado por um NPC.

(2) Botão que quando está selecionado indica que naquele slot contém um jogador humano, ou que um jogador humano pode entrar ali. Este botão possivelmente só estará

habilitado para o administrador da sala. Após ser pressionado em um slot que atualmente é ocupado por um NPC, ou que está bloqueado, este slot será liberado e estará livre para um jogador humano ocupá-lo.

(3) Botão que quando está selecionado indica que naquele slot contém um NPC. Este botão só estará habilitado para o administrador da sala. Se pressionado, uma pequena janela será aberta solicitando que o jogador selecione o nível de dificuldade do NPC que ele deseja inserir naquele slot.

(4) Texto que mostra o apelido do jogador daquele slot.

(5) Ícone informando que o jogador daquele slot é o administrador da sala.

(6) Imagem demonstrando como será na partida o personagem do jogador daquele slot.

(7) Botão que quando pressionado altera a borda atual do personagem do jogador daquele slot para a borda logo à direita. Todos os jogadores humanos podem alterar a borda dos seus próprios personagens, porém apenas o administrador da sala poderá realizar esta operação para os NPCs ali presentes.

(8) Botão que quando pressionado altera a borda atual do personagem do jogador daquele slot para a borda logo à esquerda. Todos os jogadores humanos podem alterar a borda dos seus próprios personagens, porém apenas o administrador da sala poderá realizar esta operação para os NPCs ali presentes.

(9) Imagem indicando que aquele slot está liberado.

(10) Imagem indicando que aquele slot está bloqueado.

(11) Botão que ao ser pressionado expulsa da sala o jogador daquele slot, deixando este slot livre. Apenas o administrador tem acesso a este recurso.

(12) Botão que ao ser pressionado bloqueia aquele slot, que antes deste procedimento era livre. Apenas o administrador tem acesso a este recurso.

(13) Ícone indicando que o jogador daquele slot já forneceu permissão para que a partida seja iniciada.

(14) Ícone informando que as opções logo abaixo são relativas à escolha da equipe do jogador daquele slot.

(15) Botões representando as equipes disponíveis no jogo, onde o botão selecionado representa a equipe atual do jogador daquele slot. Todos os jogadores humanos podem alterar as suas próprias equipes, porém apenas o administrador da sala poderá realizar esta operação para os NPCs ali presentes.

(16) Espaço onde todas as mensagens enviadas no chat são exibidas. Este componente é desativado caso o jogo esteja off-line.

(17) Caixa de texto onde o jogador deve inserir a mensagem que deseja enviar aos demais jogadores da sala. Este componente é desativado caso o jogo esteja off-line.

(18) Botão que ao ser pressionado envia a mensagem inserida pelo jogador, caso exista alguma, para os demais jogadores da sala. Este componente é desativado caso o jogo esteja off-line.

(19) Texto informando que os componentes logo abaixo dele são relativos às opções da partida.

(20) Componente representando uma opção da partida com dois estados. Sua opacidade indica se esta opção está ativada ou não. Ao pressioná-lo o estado atual desta opção passará a ser o seu estado inverso. Apenas o administrador pode realizar esta operação.

(21) Texto informando o nome daquela opção.

(22) Ícone que representa graficamente aquela opção.

(23) Botão utilizado pelo jogador para autorizar, ou não, o início da partida. Ele funciona de duas maneiras diferentes, de acordo com o poder do jogador dentro da sala. Para os

jogadores comuns este botão possui dois estados distintos: autorizo e não autorizo. Quando o botão é pressionado o estado atual será o seu estado inverso. Para o jogador administrador, este botão permanece desativado até que todos os demais jogadores tenham permitido o início da partida. Neste caso, quando pressionado, se o jogo estiver online, uma janela será aberta informando aos jogadores que a configuração da partida está sendo realizada. Entretanto, se o jogo estiver off-line o jogador será redirecionado imediatamente até a tela onde a partida ocorrerá.

(24) Botão que quando pressionado faz o jogador sair desta sala e imediatamente retornar a tela Salas se o jogo estiver off-line. Ou que abre uma pequena janela informando ao jogador que ele está saindo da sala se o jogo estiver online.

2.3.1.1 Janelas

Inserir nível de dificuldade do NPC

Esta janela (Figura 14) é encerrada quando o jogador pressiona o apontador fora dela.

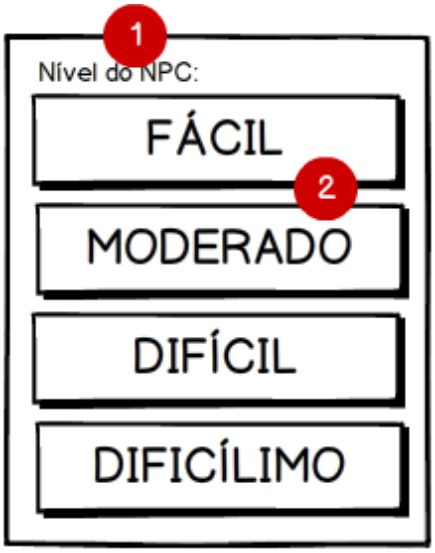

Figura 14: Janela “Inserir nível de dificuldade do NPC”

(1) Texto que indica ao jogador que ele deve selecionar um nível de dificuldade para o NPC que deseja inserir.

(2) Botões representando os níveis de dificuldade existentes para NPCs. Ao pressionar um

destes botões, um NPC com um nível de dificuldade relativo ao botão pressionado é inserido no slot que gerou esta janela. A janela é então encerrada.

Saindo da sala

Esta janela (Figura 15) não é encerrada quando o jogador pressiona o apontador fora dela. Após o servidor confirmar a saída do jogador, este deixará a sala e será redirecionado até a tela Salas.

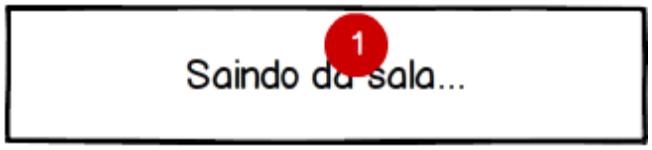

Figura 15: Janela “Saindo da sala”

(1) Texto que informa ao jogador que ele está saindo da sala.

Configurando a partida

Esta janela (Figura 16) só é encerrada caso um jogador seja desconectado do servidor durante a configuração da partida. Após todos os recursos necessários serem configurados o jogador será redirecionado para a tela onde a partida ocorrerá.

Figura 16: Janela “Configurando a partida”

(1) Texto que informa ao jogador que a partida está sendo configurada.

2.3.2 Sonoplastia

Um efeito sonoro específico é reproduzido nas seguintes situações:
- Quando outro jogador humano ou um NPC ocupa um slot.
- Quando outro jogador humano ou um NPC sai da sala por qualquer razão. Ou quando o

jogador é expulso da sala.
- Quando qualquer jogador autoriza o início da partida.

2.3.3 Lógica / Fluxo

Toda sala possui um administrador, que inicialmente é aquele jogador que a criou. Esta administração só passará para outro jogador caso o administrador atual saia da sala. Um jogador sem este poder só poderá alterar as suas próprias configurações, jamais outras configurações da sala.

Quando não se está jogando off-line, a maioria das operações realizadas dentro da sala requer uma confirmação do servidor para serem de fato concretizadas. Enquanto isso não acontece, um ícone representando um carregamento é exibido sobre o componente que iniciou a operação, e este componente é desativado.

Se o jogador perder sua conexão com o servidor, ele sairá da sala em que se encontra e será levado até a tela Salas.

Quando um jogador humano entra em uma sala, ele passa a ocupar o primeiro slot liberado. Quando ele sai dessa sala o slot que ele ocupava se torna livre.

2.4 Partida

Esta é a tela principal do jogo (Figura 17), onde de fato o jogo acontece e onde os jogadores devem utilizar suas habilidades para encarar os constantes desafios e tentar alcançar a vitória.

2.4.1 Elementos

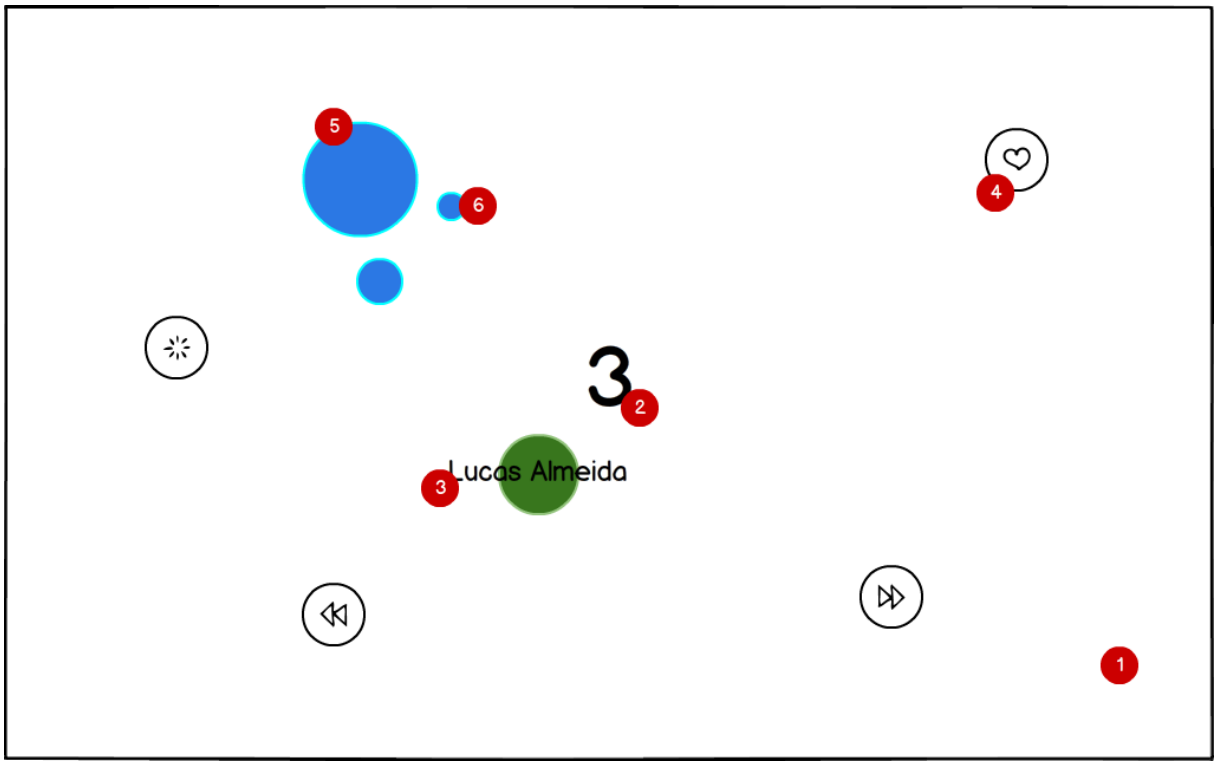

Figura 17: Tela “Partida”

(1) Imagem de fundo, de caráter totalmente visual, sem qualquer tipo de interação.

(2) Contador textual que indica quantos segundos ainda restam para os jogadores poderem de fato interagir no jogo.

(3) Texto informando o apelido do jogador, cujo personagem está sob ele. Este componente desaparece assim que o contador é finalizado.

(4) Gráfico animado representando um item de bônus ou penalidade. Quando um personagem ou um fragmento colide com um item, os efeitos relativos a este item são aplicados ao elemento que colidiu, e o item é destruído.

A cada 15~20 segundos um novo item surge em alguma região da cena onde não existem personagens e fragmentos. O tipo do item é definido aleatoriamente, levando em consideração apenas aqueles que foram habilitados pelo administrador da sala na cena desta sala.

Os tipos de itens existentes são:

- Aumento de vida: o elemento que utilizar este item ganha uma determinada quantidade de vida.
- Aumento de velocidade: o elemento que utilizar este item tem sua velocidade ampliada por um determinado período de tempo.
- Redução de velocidade: o elemento que utilizar este item tem sua velocidade reduzida por um determinado período de tempo.
- Indestrutibilidade: o elemento que utilizar este item se torna indestrutível por um determinado período de tempo.

(5) Gráfico circular que representa um personagem. Possui uma cor de preenchimento referente à equipe do respectivo jogador, e uma cor de borda escolhida por este durante a tela da sala.

Um personagem nunca entra em repouso, ele estará sempre em movimento seguindo em uma determinada direção. Esta direção só será alterada quando uma destas ações ocorrer:

- O jogador pressionar um apontador em qualquer região da cena. O personagem passará a se mover em direção a este ponto que foi pressionado.
- O personagem se colidir com outro personagem, com um fragmento ou com uma das extremidades da cena. Ele passará a se mover na direção oposta ao ponto de colisão.

Sua velocidade será sempre constante ao menos que uma destas ações ocorra:

- O jogador pressiona diversas vezes uma determinada região da tela. A velocidade do personagem será correspondente à velocidade com que o jogador pressiona o apontador. À medida que os pressionamentos vão se tornando menos frequentes o personagem tende a retornar à sua velocidade inicial.
- O personagem se colide com algum item de alteração de velocidade. Dependendo do item, sua velocidade será ampliada ou reduzida durante um determinado período de tempo.

Um personagem possui uma determinada quantidade de vidas que reflete diretamente na área do seu círculo: quanto mais vidas possuir, maior será a sua área. Ao atingir um tamanho mínimo o personagem é destruído e o seu jogador é derrotado. Este personagem ganha vida apenas ao colidir com um item de aumento de vida. Ele perde vida das seguintes maneiras:

- Ao colidir com outros personagens, com os fragmentos dos rivais ou com as extremidades da cena. No segundo caso, a perda de vida é mais ampla, e o valor perdido dependerá do tamanho do fragmento com que ele sofreu a colisão.

- Ao gerar fragmentos. Quando o jogador mantém pressionado um apontador na cena, o seu personagem entra em estado de divisão. Enquanto este apontador não for liberado o personagem vai doando sua vida proporcionalmente ao seu novo fragmento. Um fragmento só deixará de crescer quando o personagem atingir sua vida mínima. Ao liberar o apontador, o fragmento se desprenderá do personagem e seguirá em direção ao ponto onde este apontador foi liberado.

Um personagem que esteja momentaneamente indestrutível não sofre perda de vida em qualquer uma de suas colisões. Porém, ao se colidir com um fragmento de um rival, sua indestrutibilidade termina logo após esta colisão.

A inteligência artificial utilizada pelos NPCs e aplicada aos seus personagens funcionará da seguinte maneira:

- O personagem, estando destrutível, deve prioritariamente se afastar de fragmentos dos rivais, principalmente se estes forem grandes.
- O personagem, estando destrutível, deve evitar colisões com as extremidades da cena, com os personagens destrutíveis dos rivais que sejam maiores do que ele, e com os personagens indestrutíveis dos rivais.
- O personagem, estando indestrutível, deve evitar colisões com os menores fragmentos rivais da cena.
- O personagem, estando destrutível, deve buscar colisão com os personagens dos rivais destrutíveis que sejam menores do que ele e que estejam próximos.
- O personagem, estando destrutível, deve buscar colisão com os personagens dos rivais destrutíveis que sejam bem menores do que ele, mesmo que estejam distantes.
- O personagem, estando indestrutível, além de receber mais atenção do NPC, deverá buscar colisão com os fragmentos dos rivais, principalmente se estes forem grandes e destrutíveis.
- O personagem, estando indestrutível, deve buscar colisão com os personagens rivais destrutíveis, principalmente se estes estiverem próximos, e o personagem estiver mais veloz.
- O personagem deve ir de encontro aos itens que surgem na cena, exceto daquele que reduz sua velocidade. Com este ele deve evitar uma colisão. Se for mais vantajoso para algum elemento aliado colidir com um determinado item, o personagem não executará qualquer ação relacionada a este item.
- O personagem, estando destrutível, e tendo utilizado por engano um item de redução de velocidade, deixará o NPC mais atento a ele, principalmente para evitar colisões com fragmentos rivais.
- O personagem deve lançar fragmentos. A quantidade de fragmentos lançados e o tamanho

destes dependerão da capacidade do NPC de administrar muitos elementos ao mesmo tempo e da vida do seu personagem.

Quanto maior o nível de dificuldade de um NPC, maior será a frequência com que ele executa os procedimentos acima.

(6) Gráfico circular que representa um fragmento de um personagem. Possui uma cor de preenchimento referente à equipe do respectivo jogador, e uma borda fina com uma cor que foi escolhida por este durante a tela da sala.

Um fragmento inicia sua vida preso nas extremidades do seu personagem. Após o jogador liberar o apontador que gerou o fragmento, este se desprenderá do personagem, e passará a ter um movimento independente, inicialmente em direção ao ponto onde o apontador foi liberado. Um fragmento, assim como um personagem, nunca entra em repouso, ele estará sempre em movimento seguindo em uma determinada direção.Esta direção só será alterada quando uma destas ações ocorrer:

- O jogador arrastar por cima do fragmento um apontador pressionado e liberar este apontador em qualquer outra região da cena. Este fragmento passará a se mover em direção ao ponto onde o apontador foi liberado.
- O fragmento sofrer colisão com uma das extremidades da cena, ou com um personagem rival. Neste último caso o fragmento precisa estar em modo de indestrutibilidade para não ser destruído no processo. Em ambos os casos ele passará a se mover na direção oposta ao ponto de colisão.

A velocidade de um fragmento será sempre constante, a menos que ele colida com algum item de alteração de velocidade. Neste caso, dependendo do item, ela será ampliada ou reduzida por um determinado período de tempo.

Um fragmento possui uma determinada quantidade de vidas que reflete diretamente na área do seu círculo: quanto mais vidas possuir, maior será a sua área. Este fragmento ganha vida apenas ao colidir com um item de aumento de vida. Ele perde vida das seguintes maneiras:

- Após colidir com uma das extremidades da cena. Caso o fragmento atinja um tamanho mínimo ele será destruído.
-Após colidir com um personagem rival. O fragmento será automaticamente destruído e o personagem sofrerá uma perda de vida proporcional à vida que o fragmento possuía.

Um fragmento que esteja momentaneamente indestrutível não sofre perda de vida em qualquer uma de suas colisões. Porém ao se colidir com um personagem rival, sua

indestrutibilidade termina logo após esta colisão.

A inteligência artificial utilizada pelos NPCs e aplicada aos fragmentos dos seus personagens funcionará da seguinte maneira:

- O fragmento, estando destrutível, deve prioritariamente se afastar de personagens rivais indestrutíveis. Exceto se este fragmento for muito pequeno. Neste caso ele buscará colisão com estes personagens.
- O fragmento, estando destrutível, deve evitar colisões com as extremidades da cena.
- O fragmento deve buscar colisão com os personagens rivais destrutíveis. A decisão de qual personagem rival será o escolhido depende de alguns fatores: a distância do personagem até o fragmento, a vida do personagem, a vida do fragmento, a quantidade de fragmentos do personagem e a vida dos fragmentos do personagem.
- O fragmento deve ir de encontro aos itens que surgem na cena, se não for mais vantajoso para outro elemento aliado fazer isso.
- O fragmento que utilizou um item de aumento de velocidade ou de indestrutibilidade deve receber mais atenção do NPC, aproveitando desta vantagem temporária para conseguir colisões positivas.
- O fragmento que utilizou um item de redução de velocidade deve receber menos atenção do NPC, deixando este mais atento às outras situações da partida que estejam ocorrendo.

Quanto maior o nível de dificuldade de um NPC, maior será a frequência com que ele executa os procedimentos acima.

2.4.1.1 Janelas

Equipe venceu

Esta janela (Figura 18) não é encerrada quando o jogador pressiona o apontador fora dela. A sua cor de fundo será a mesma cor da equipe vencedora.

Figura 18: Janela “Equipe venceu”

(1) Texto que informa ao jogador a equipe vencedora da partida.

(2) Botão que quando pressionado leva automaticamente o jogador de volta a tela de sua sala.

2.4.2 Sonoplastia

Um efeito sonoro específico é reproduzido nas seguintes situações:
- A cada novo valor do contador que seja maior do que zero.
- Quando o valor do contador se torna zero.
- Quando um personagem colide com outro personagem, ou quando um personagem ou um fragmento colide com uma das extremidades da cena.
- Quando um personagem colide com um fragmento de um rival.
- Quando um personagem ou um fragmento colide com um item.
- Quando um jogador é derrotado.
- Quando uma equipe vence a partida.

2.4.3 Lógica / Fluxo

Quando um jogador perde sua conexão com o servidor ele é tido como derrotado na partida. Este jogador desconectado será automaticamente levado até a tela das salas.

Quando restar apenas uma equipe viva na partida, esta será a equipe vencedora. Uma janela será aberta informando ao jogador qual foi a equipe que venceu.

Quando o jogador é expulso da sala, mesmo estando ainda na tela da partida, ele será automaticamente levado até a tela das salas.

APÊNDICE C – Formulário para Avaliação de Bolita

Idade: _____ Sexo: _____

Após ter jogado Bolita por alguns minutos, dê uma nota de 0 a 10 para as características do jogo listadas abaixo:

1 – Arte gráfica: qualidade visual

0	1	2	3	4	5	6	7	8	9	10

2 – Sonoplastia: qualidade sonora

0	1	2	3	4	5	6	7	8	9	10

3 – Interação antes da partida: facilidade para manusear a interface gráfica

0	1	2	3	4	5	6	7	8	9	10

4 – Interação durante a partida: facilidade para manusear o personagem e os seus fragmentos

0	1	2	3	4	5	6	7	8	9	10

5 – Jogabilidade: diversão atingida ao jogar

0	1	2	3	4	5	6	7	8	9	10

6 – Criatividade: nível de criatividade do jogo

0	1	2	3	4	5	6	7	8	9	10

7 – Carregamento dos arquivos: o quão suave foi o carregamento dos arquivos

0	1	2	3	4	5	6	7	8	9	10

8 – NPCs: o quão adequado estão os níveis de dificuldade dos NPCs

0	1	2	3	4	5	6	7	8	9	10

9 – Recomendação para amigos: nível de frequência com que você recomendaria o jogo para seus amigos

0	1	2	3	4	5	6	7	8	9	10

10 – Satisfação geral: nível geral de satisfação com o jogo

0	1	2	3	4	5	6	7	8	9	10