CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DE BOTUCATU
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

THIAGO AUGUSTO JORGE

DESENVOLVIMENTO DE JOGO ELETRÔNICO PARA DISPOSITIVO MÓVEL UTILIZANDO A PLATAFORMA UNITY 3D

Botucatu-SP
Janeiro-2017

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
FACULDADE DE TECNOLOGIA DE BOTUCATU
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

THIAGO AUGUSTO JORGE

DESENVOLVIMENTO DE JOGO ELETRÔNICO PARA DISPOSITIVO MÓVEL UTILIZANDO A PLATAFORMA UNITY 3D

Orientador: Prof. Dr. Gustavo Kimura Montanha

Projeto de Conclusão de Curso apresentado à FATEC - Faculdade de Tecnologia de Botucatu, para obtenção do título de Tecnólogo no Curso Superior de Análise e Desenvolvimento de Sistemas.

Botucatu-SP
Janeiro-2017

Aos meus pais, Vitor e Solange por todo o apoio e dedicação.

AGRADECIMENTOS

Primeiramente agradeço aos meus pais, Vitor Roberto Jorge e Solange Aparecida Pacharone Jorge, que dispuseram de sua total dedicação e respeito a todas as minhas decisões tomadas até o momento.

A minha namorada Pamela Cristina Gallerani, por todo o incentivo e compreensão prestados a mim durante os momentos mais difíceis.

Ao meu professor orientador Gustavo Kimura Montanha, por confiar no meu trabalho e dedicar seu tempo e esforço durante a criação desse projeto. A todos os professores do curso de Análise e Desenvolvimento de Sistemas, que contribuíram para a construção do meu conhecimento nessa área.

A todos os meus colegas de sala, por todos os momentos de alegria e descontração durante as aulas.

Por fim, aos demais funcionários da FATEC.

A todos, muito obrigado!

RESUMO

O surgimento dos jogos se deu em uma época muito antiga bem antes do surgimento dos dispositivos eletrônicos e computacionais. Desde o princípio os jogos possuíam o intuito de entretenimento e estavam relacionados com o costume e culturas locais de diferentes civilizações. O surgimento dos primeiros dispositivos eletrônicos rapidamente rompeu a restrição de acesso a esse tipo de recurso e reuniu jogos cada vez mais complexos e diversificados. Atualmente os jogos se tornaram extremamente populares devido ao constante crescimento da tecnologia e da indústria eletrônica. Esse crescimento impulsionou também o mercado de dispositivos móveis que se adaptou rapidamente a indústria de jogos no mundo inteiro. A consolidação dessa indústria se deu através da adoção de métodos que proporcionaram um desenvolvimento ágil e flexível dos jogos. Para isso, surgiram diversas plataformas que trouxeram todos os recursos necessários para edição, programação e compilação dos jogos. Esse projeto tem como objetivo a demonstração do desenvolvimento de um jogo eletrônico para dispositivos móveis através da plataforma Unity.

PALAVRAS-CHAVE: Desenvolvimento. Dispositivos Móveis. Jogos Digitais.

LISTA DE FIGURAS

Página

LISTA DE ABREVIATURAS E SIGLAS

2D – DUAS DIMENSÕES

3D – TRÊS DIMENSÕES

APP - APLICATIVOS

BMP – BITMAP

C# – C SHARP

FBX - FILMBOX

GB – GIGABYTE

IOS – IPHONE OPERATIONAL SYSTEM

JPG – JOINT PHOTOGRAPHIC EXPERTS GROUP

MP3 – MPEG-1/2 AUDIO LAYER 3

PNG – PORTABLE NETWORK GRAPHICS

PSD – PHOTOSHOP DOCUMENT (DOCUMENTO DO PHOTOSHOP)

RAM – RANDOM ACCESS MEMORY (MEMÓRIA DE ACESSO ALEATÓRIO)

SCIELO – SCIENTIFIC ELECTRONIC LIBRARY ONLINE (REVISTA CIENTÍFICA ELETRÔNICA ONLINE)

WAV – WAVEFORM AUDIO FILE FORMAT

WEB – REFERÊNCIA A WORLD WIDE WEB (WWW)

SUMÁRIO

Página

1 INTRODUÇÃO

O conceito de jogo eletrônico conhecido atualmente surgiu no momento em que os primeiros equipamentos digitais foram criados. Devido à dificuldade de acesso a esses dispositivos, os primeiros jogos não tornaram-se populares e a consolidação da indústria de jogos aconteceu apenas alguns anos mais tarde.

Historicamente, Clua e Bittencourt (2005) descrevem que o primeiro jogo eletrônico foi criado por Steve Russel por volta de 1962, denominado *SpaceWar!*. Esse tipo de jogo ainda era "testado em laboratório" no qual apenas o seu criador tinha acesso ao dispositivo que executava essa aplicação.

Os mesmos autores salientam ainda que a popularização dos jogos eletrônicos se deu por volta de 1972, momento em que o jogo *Atari Pong* teve grande repercussão e popularidade inovando assim o conceito de entretenimento eletrônico. A partir de então, a tecnologia envolvida na produção dos jogos eletrônicos, bem como sua popularidade tem se expandido e desenvolvido no decorrer dos anos.

Em 1994 é lançado o celular da empresa *Cetelco* denominado *Hagenuk MT-2000*. Esse aparelho foi importante para o surgimento dos primeiros jogos eletrônicos voltados para dispositivos móveis. Embora na época a novidade não tenha feito o sucesso esperado, esse aparelho abriu portas para a chegada de jogos eletrônicos como o *Tetris* e possibilitou o pontapé inicial para a expansão desse mercado (GRUPO NZN, 2016).

Alguns anos depois, em 2004, estima-se que a indústria de entretenimento digital tenha movimentado cerca de 40 bilhões de dólares somente nesse ano, ultrapassando de forma significativa o faturamento da indústria cinematográfica (CLUA; BITTENCOURT, 2005).

Os jogos sofreram um salto gigantesco em relação a qualidade gráfica e jogabilidade, isso aconteceu entre os anos 2007 e 2009, momento em que os primeiros celulares com tela colorida surgiram no mercado e os primeiros jogos com cenário 3D ficaram disponíveis para esses dispositivos (GRUPO NZN, 2016).

A popularização dos *smartphones* e o crescente uso dos dispositivos móveis tem impulsionado significativamente o mercado de jogos no mundo inteiro. As empresas motivadas por esse crescimento têm investido e se dedicado cada vez mais ao desenvolvimento para esse ramo.

A partir dessa perspectiva surgiu uma imensa quantidade de jogos tratando dos mais diversos assuntos, temáticas e público-alvo. Observa-se o surgimento desde jogos mais simples para entretenimento de crianças, até jogos extremamente complexos, voltados para a diversão de jovens e adultos.

Com a complexidade inerente a esses jogos eletrônicos, as empresas passaram a necessitar de melhor adequação para tornar o processo de desenvolvimento o mais completo e produtivo possível. Para auxiliar tal cenário, surgem plataformas integradas de desenvolvimento de jogos eletrônicos, que contém quase todas ferramentas necessárias para a edição, programação e compilação dos mesmos.

Para que essa adaptação aconteça, há uma necessidade de padronização do desenvolvimento a fim de facilitar todas as etapas durante o processo de criação dos jogos. Essa padronização ocorre através da adoção de métodos que possibilitem o desenvolvimento ágil e integral das aplicações.

Para o desenvolvimento desse projeto será utilizada uma das plataformas integradas contidas no mercado denominada *Unity 3D*. Essa plataforma caracteriza-se por apresentar fácil utilização na edição, programação e compilação em até três linguagens de programação diferentes, *UnityScript* (*Javascript* adaptado para a plataforma), *C#* e *Boo* (Linguagem baseada em *Python*).

Além disso o *Unity 3D* possui capacidade de compilação para múltiplas plataformas, o que significa que os desenvolvedores utilizando essa ferramenta têm a possibilidade de desenvolver para *Windows Phone*, *Windows*, *iOS*, *Linux*, *Android*, *consoles PlayStation 3*, *PlayStation 4*, *Xbox 360*, *Xbox One*, dentre outras.

1.1 Objetivo

Esse projeto teve como objetivo desenvolver um jogo eletrônico para dispositivos móveis utilizando a plataforma *Unity 3D* e especificar parte do processo de criação, construção, edição e finalização.

1.2 Justificativa e Relevância do Tema

Os tempos mudaram, atualmente a indústria 3D não está mais somente relacionada ao cinema, mas também à indústria de jogos. A computação gráfica gerou uma grande mudança que culminou na migração de diversos jogos 2D para a nova realidade 3D (BLACKMAN, 2013).

As empresas desenvolvedoras estão cada vez mais se adaptando a essa nova realidade. Além disso, estão sempre em busca de criar os jogos mais realistas, com a melhor jogabilidade e da forma mais produtiva possível.

Atualmente os jogos eletrônicos estão participando ativamente da vida de grande parte das pessoas no mundo inteiro. A partir dessa perspectiva, as empresas desenvolvedoras decidiram adotar estratégias para explorar as oportunidades presentes nesse mercado, para isso, o mundo corporativo necessita se adequar as altas demandas desse tipo de software.

Para que esse novo mercado seja explorado, há uma necessidade de padronizar o desenvolvimento de jogos a fim de agilizar processos, cumprir demandas e se habituar as novas tecnologias. Para suprir essas e outras necessidades, esse projeto demonstrará os diversos recursos utilizados para o desenvolvimento de jogos presentes na plataforma *Unity 3D*.

2 REVISÃO DE LITERATURA

2.1 Espaço 3D

Para Clua e Bittencourt (2005, p. 1313): “Um jogo 3D é um *software* especial, pois contém elementos muito variados: módulos de Computação Gráfica, Inteligência Artificial, Redes de Computadores, Multimídia, entre outros.”

No espaço 3D se utiliza o que é chamado de plano cartesiano. No *Unity* o eixo X é representado como horizontal, o Y como vertical e o Z como a profundidade ou terceira dimensão. Através desse panorama os objetos podem ser rotacionados, movidos ou ampliados/reduzidos (BLACKMAN, 2013).

Segundo Clua e Bittencourt (2005), para a construção de um jogo 3D são necessários também elementos 2D:

> Os jogos 3D não são construídos somente com modelos tridimensionais, na produção de um jogo também é necessário compor imagens bidimensionais. Em geral, tais imagens serão usadas como texturas, mas também serão usadas para compor a interface gráfica *ingame* e *outgame*, tais como, botões, janelas, barras de energia e outros componentes gráficos (CLUA; BITTENCOURT, 2005, p. 1330).

Esses elementos são importantes, pois permitem a interação do usuário com o jogo através de *menus*, botões e elementos visuais que têm determinadas funções no aplicativo.

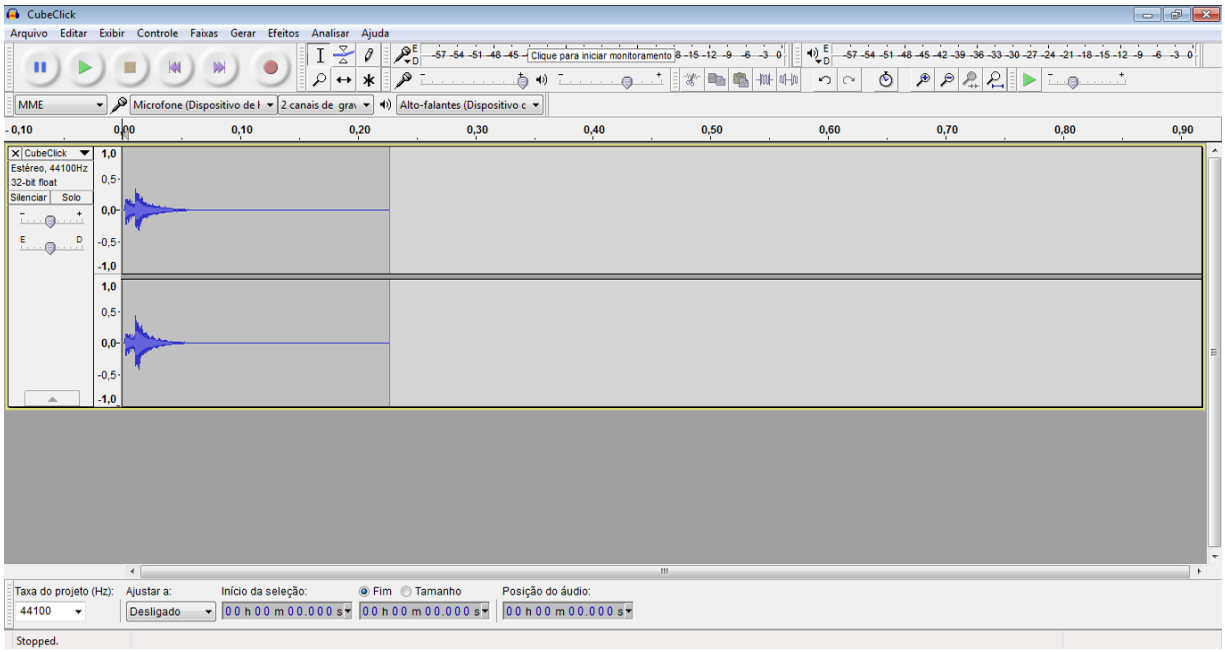
2.2 Áudio

O Áudio é um elemento fundamental na criação de um jogo, efeitos sonoros, músicas de fundo e elementos do ambiente são os principais tipos de áudio produzidos em um jogo.

Para a criação desses elementos pode-se utilizar ferramentas especializadas em produção sonora, nesse projeto foi utilizada a ferramenta *Audacity*, ilustrado na Figura 1.

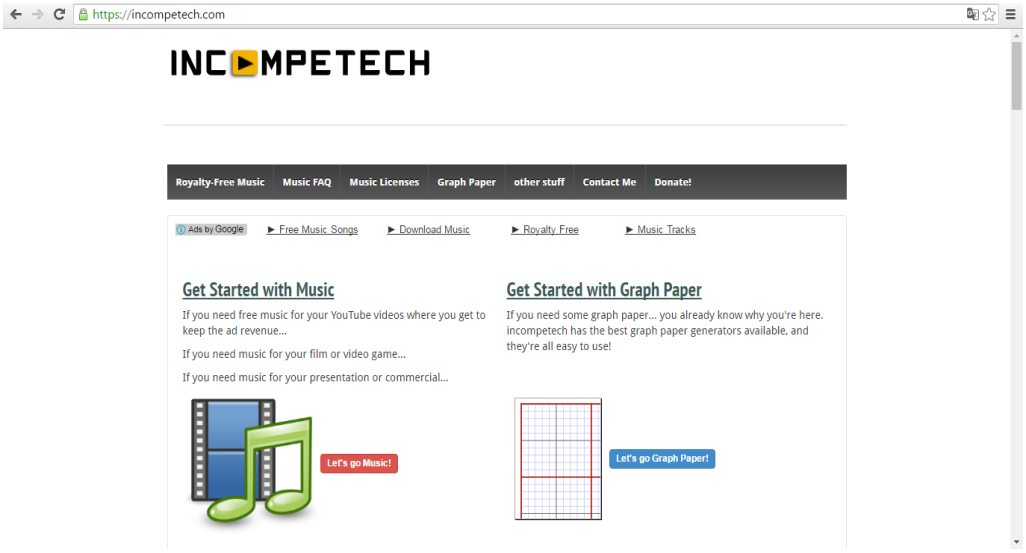
Segundo Clua e Bittencourt (2005, p. 1329): "O *Audacity* [...] é uma ferramenta livre que permite a criação de áudio, inclusive combinando diferentes canais de som, importando e exportando arquivos no formato *WAV*, *MP3* e *Ogg Vorbis*". Para Mazzoni (2016), próprio autor do Audacity: [...] é um *software* de código fonte livre utilizado para gravar e editar sons (MAZZONI, 2016).

Figura 1. *Interface* básica do *Audacity*, exibindo a linha de edição de um efeito sonoro.



Caso o desenvolvedor prefira não produzir o áudio do jogo com programas específicos, ele tem a opção de utilizar efeitos sonoros livres de *copyright* denominados áudios *royalty free*, ou seja, áudios com direitos de uso livre. Vários *sites*, conforme ilustrado na Figura 2, disponibilizam esse tipo de arquivo sendo encontrados facilmente por meio de pesquisas na *internet*. Esse processo pode facilitar a produção de áudio uma vez que tal atividade pode demandar muito tempo no desenvolvimento do jogo.

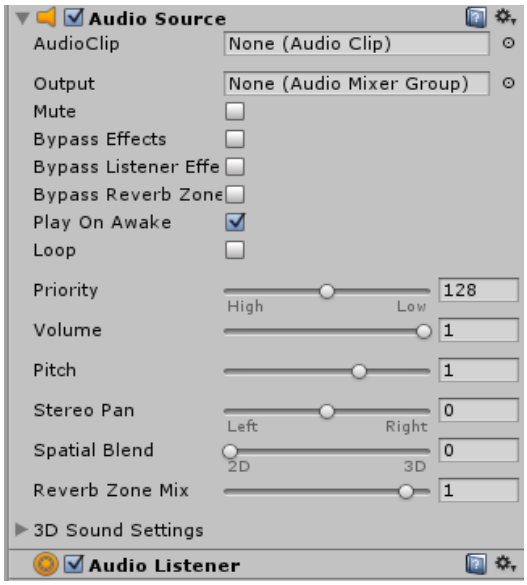
Figura 2. *Incompetech*, *website* que disponibiliza áudio com direito de uso livre.



No momento em que o áudio é importado para a plataforma ele se torna um *Áudio Clip* e necessita ser executado e interpretado na cena em que o jogo se passa. Na *Unity*, os arquivos de áudio são uma instância de um a*udio clip*, que é um componente tão flexível que pode ser usado para qualquer tipo de áudio (LAVIERI, 2015).

Os componentes responsáveis por essa execução e interpretação são respectivamente o *Áudio Source* e o *Áudio Listener* (Figura 3). *Unity Technologies* (2015) sobre esses componentes: "O *Audio Source* reproduz um *Audio Clip* na cena. O *clip* pode ser tocado por um *audio listener* [...]." Sem esses dois componentes, torna-se impossível reproduzir o áudio na cena.

Figura 3. Componentes *Audio Source* e *Audio Listener*.

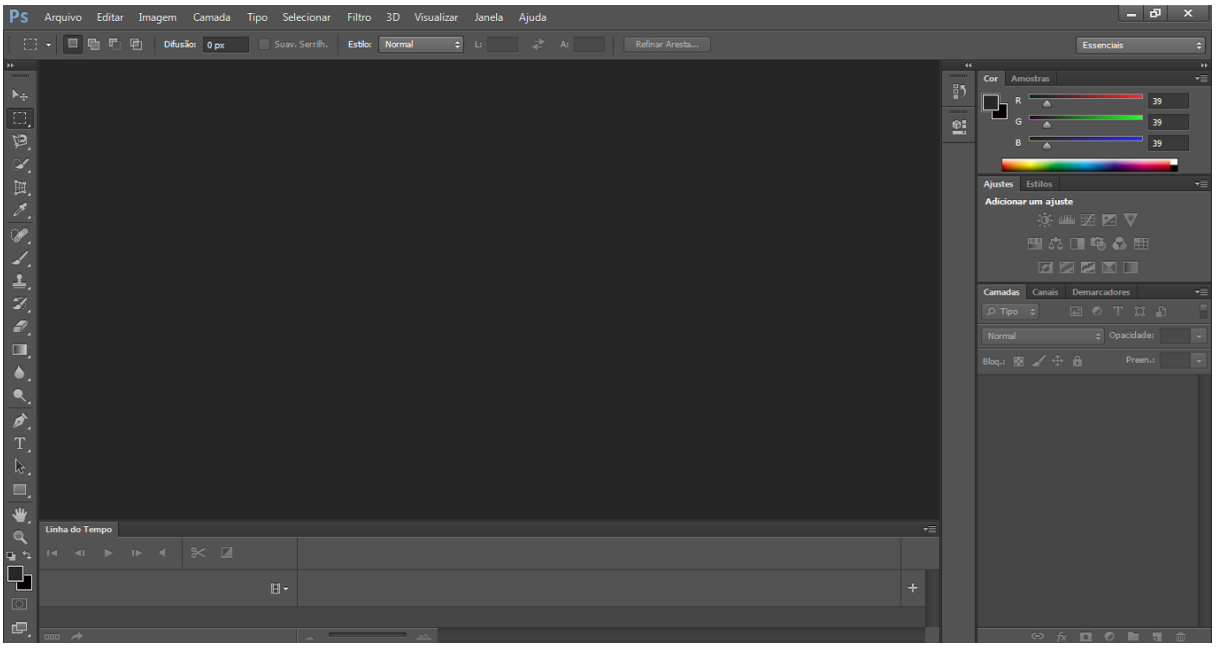


2.3 Adobe Photoshop e Adobe Illustrator

Muitas vezes, os elementos visuais de um jogo tais como texturas, *wallpapers* e imagens em geral, precisam de ferramentas específicas para ser construídos. Nesse projeto as duas ferramentas utilizadas foram o *Adobe Photoshop* e o *Adobe Illustrator*.

Andrade (2013, p. 9) sobre o *Photoshop*: "Este *software* é um aplicativo de edição de imagens digitais, retoque de fotografias e produção de gráficos para a *Web*. É um dos mais utilizados em computadores de plataforma *Windows* ou *Macintosh* [...]." A Figura 4 apresenta uma visão geral do *Adobe Photoshop CS6*.

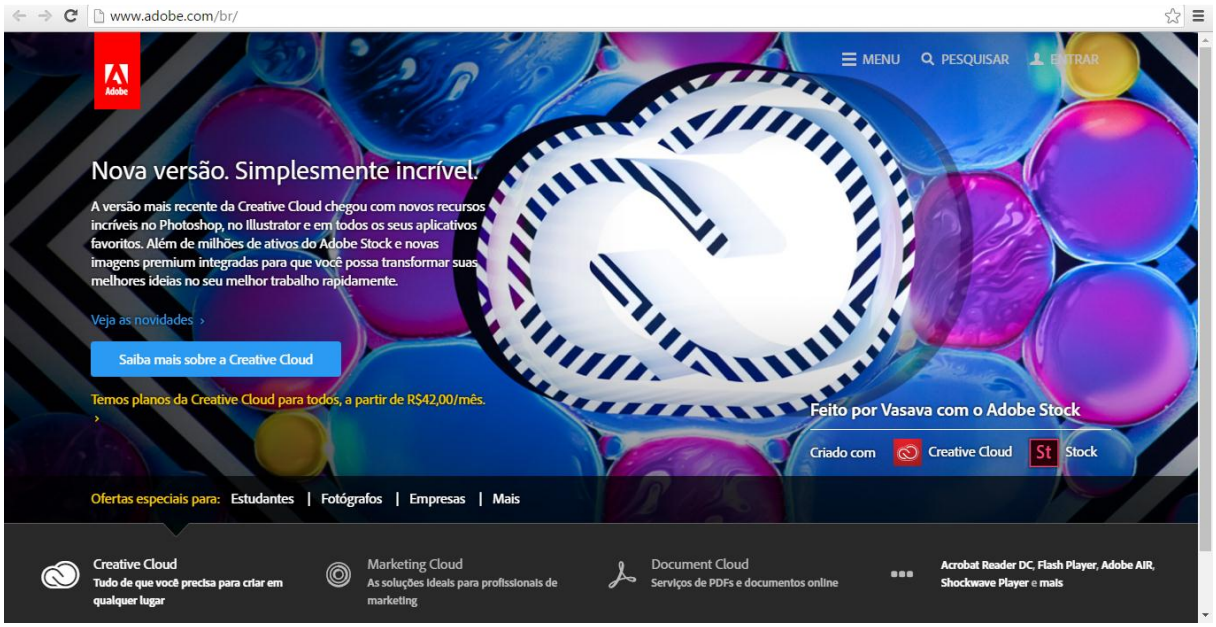
Figura 4. Visão geral do *Adobe Photoshop CS6*.



O *Photoshop* permite a você trabalhar no *desktop* ou em dispositivos móveis para criar e aprimorar fotografias, ilustrações *3D*, entre outros (ADOBE SYSTEMS INCORPORATED, 2016).

O *Adobe Illustrator* é um *software* de *design* e ilustração criado pela *Adobe Systems Incorporated* (BOTELLO, 2013). Segundo o próprio autor do *software*, o *Illustrator*, aplicativo padrão do setor de gráficos vetoriais, permite a criação de logotipos, ícones, esboços, ilustrações pra impressão, conteúdo interativo e etc. (ADOBE SYSTEMS INCORPORATED, 2016). A Figura 5 apresenta o *website* da *Adobe*.

Figura 5. *Site* oficial da empresa *Adobe*.



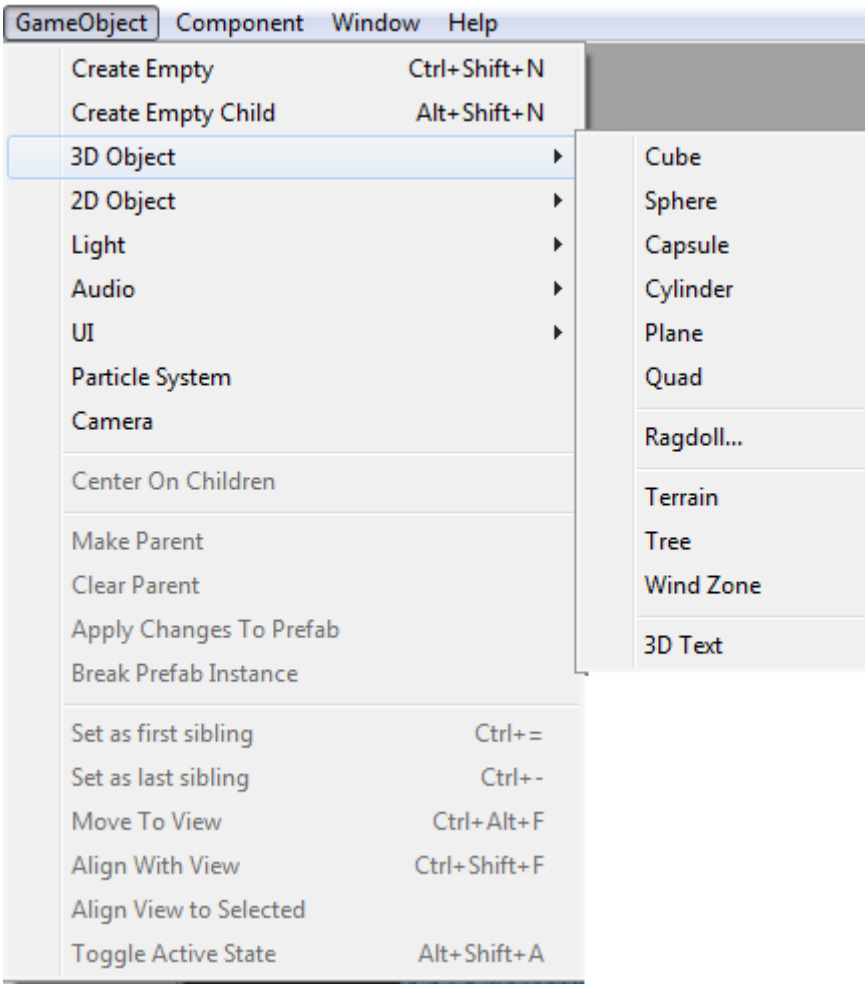
2.4 Unity

Unity 3D é um motor de jogos que permite o desenvolvimento de jogos 3D para diversas plataformas. “O motor de jogos *Unity3D* possui uma *interface* bastante simples e amigável que objetiva facilitar o desenvolvimento de jogos de diversos gêneros e outros sistemas de visualização” (PASSOS et al., 2009, p. 4).

Unity 3D é uma nova parte da tecnologia que pretende facilitar a vida para os desenvolvedores de jogos, além disso, é uma ferramenta de criação de jogos que permite que as pessoas criativas construam jogos (CREIGHTON, 2010).

Além disso, a *Unity* possui alguns elementos tridimensionais básicos para a utilização no cenário, conforme ilustrado na Figura 6, cubos e cilindros são alguns dos exemplos. Porém, na maioria da vezes, um jogo não é composto apenas por elementos básicos, por isso é necessário a criação de elementos mais complexos através de ferramentas de modelagem 3D.

Figura 6. Modelos 3D básicos, disponíveis na *Unity*.

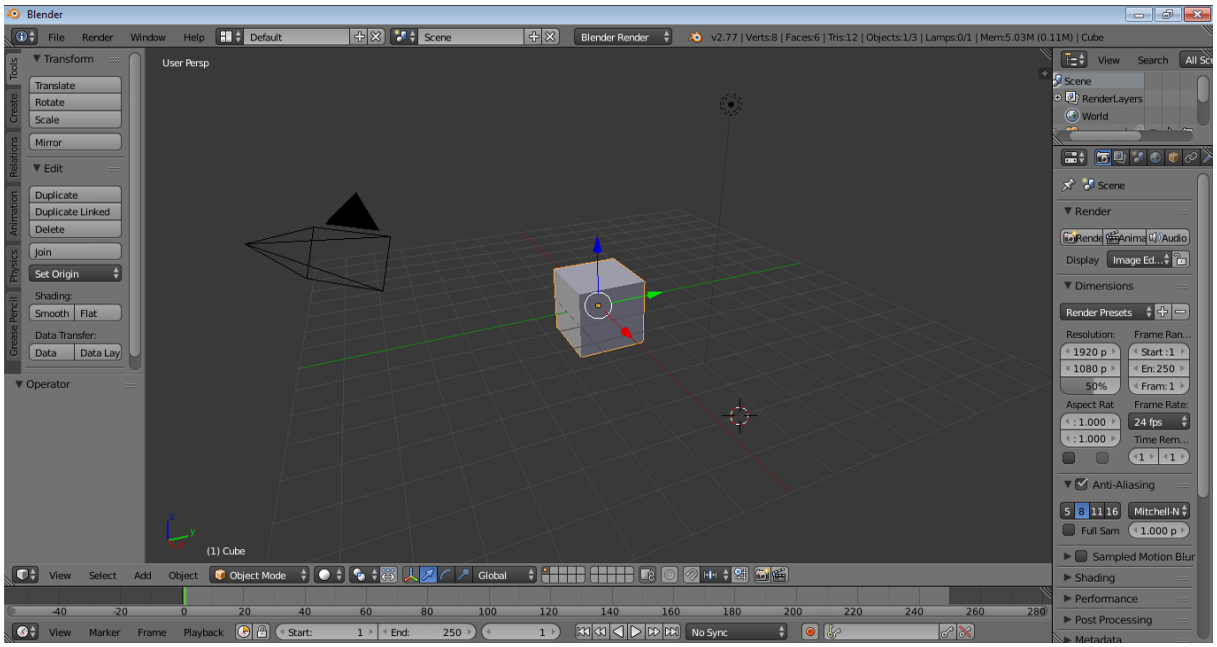


Muitas vezes uma empresa grande de desenvolvimento de jogos possui uma equipe de modelagem: "A equipe de modelagem 3D será responsável por criar os objetos geométricos das fases" (CLUA; BITTENCOURT, 2005, p. 1331). Na prática, em uma empresa de pequeno porte, o *designer* ou até mesmo o programador podem atuar como modeladores.

A modelagem 3D é necessária para criar os elementos tridimensionais complexos de um jogo, personagens, elementos do ambiente, objetos multifacetados, são criados através das ferramentas de modelagem.

Existem uma série de ferramentas que podem ser utilizadas para a criação de objetos 3D para o uso na *Unity*, uma das mais comuns é o *Blender* (Figura 7). Há também disponível uma série livros e *tutoriais online* que ensinam como usar o *Blender* para criar diferentes *Assets* (LAVIERI, 2015).

Figura 7. Área de Trabalho da ferramenta Blender.



Em relação ao desenvolvimento de jogos para aplicativos móveis, é importante destacar que todos os modelos 3D devem ser criados com o mínimo de polígonos possíveis, para evitar sobrecarga do dispositivo e permitir que a maioria dos aparelhos móveis executem esses jogos sem nenhum problema.

Conforme Clua e Bittencourt (2005), o processo de redução de polígonos também é importante:

> É comum durante o processo de modelagem criar objetos com mais polígonos do que se pode suportar no jogo. Assim sendo, é importante que um pacote de modelagem forneça recursos para reduzir o número de polígonos de objetos, minimizando a sua perda de qualidade (CLUA; BITTENCOURT, 2005, p. 1332).

O jogo que será apresentado nesse projeto, por ser um jogo simples, utilizou somente os modelos básicos fornecidos pela plataforma *Unity 3D* (Figura 8).

Figura 8. Visão geral da plataforma.

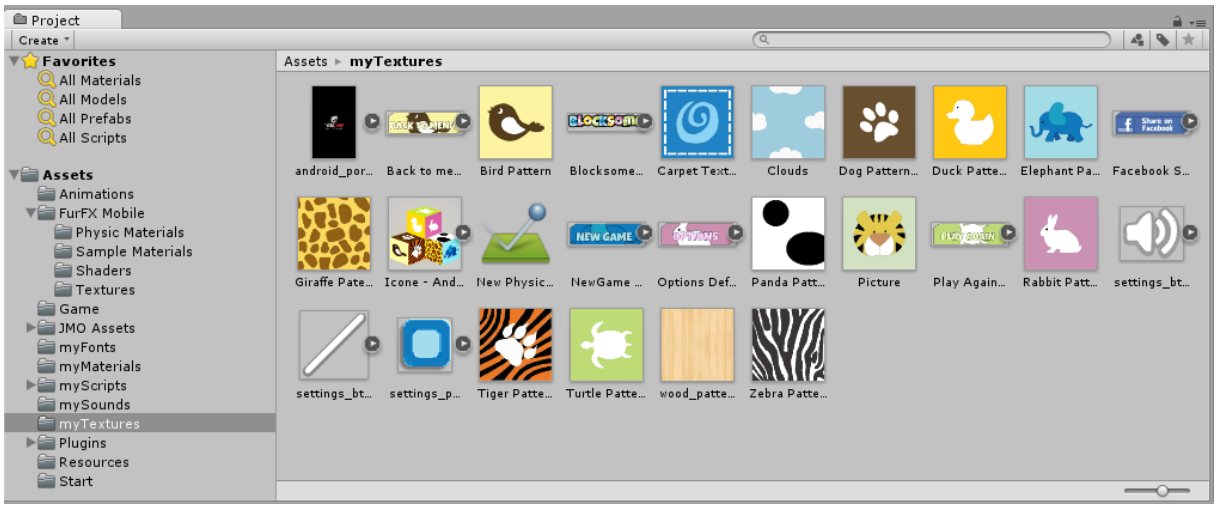
Passos et al. (2009, p. 4, grifo do autor) sobre a *Unity*: "Sua área de trabalho é composta de várias janelas chamadas **views**, cada uma com um propósito específico." Para Lavieri (2015): As *views* da *Unity* permitem ver componentes específicos do projeto. Além disso, a *Unity* permite ao desenvolvedor manipular e posicionar suas janelas de forma a torná-las mais adaptáveis ao usuário.

2.4.1 Project View

A *Project view* é um espaço para manipulação e organização dos arquivos (*Assets*) que formam um projeto (PASSOS et al., 2009). É uma janela que exibe todos os elementos internos do projeto tais como arquivos, materiais, texturas, entre outros (Figura 9).

Lavieri (2015) sobre a *Project view*: Exibe a estrutura de arquivos dos componentes do jogo, incluindo objetos, arte, *scripts* e muito mais. A estrutura dos arquivos que compõe o seu projeto é exibida nela, portanto, se quiser fazer alguma alteração, faça-as através dessa *view.*

Figura 9. *Project View* exibindo as texturas dentro da pasta selecionada (*myTextures*).



2.4.2 Hierarchy View

A *Hierarchy view* funciona como uma lista que exibe todos os elementos da cena que se está editando (PASSOS et al., 2009). Lavieri (2015) complementa, essa *view* exibe uma lista hierárquica de todos os objetos na cena.

Todos esses elementos estão inseridos em algum lugar da cena, mas nem sempre são elementos visuais, eles podem ser um objeto com algum *script* embutido (*script* que controla os painéis do jogo, por exemplo), um objeto que representa uma câmera no cenário, um objeto equivalente a um *player* de áudio, etc. A Figura 10 apresenta a *Hierarchy View*.

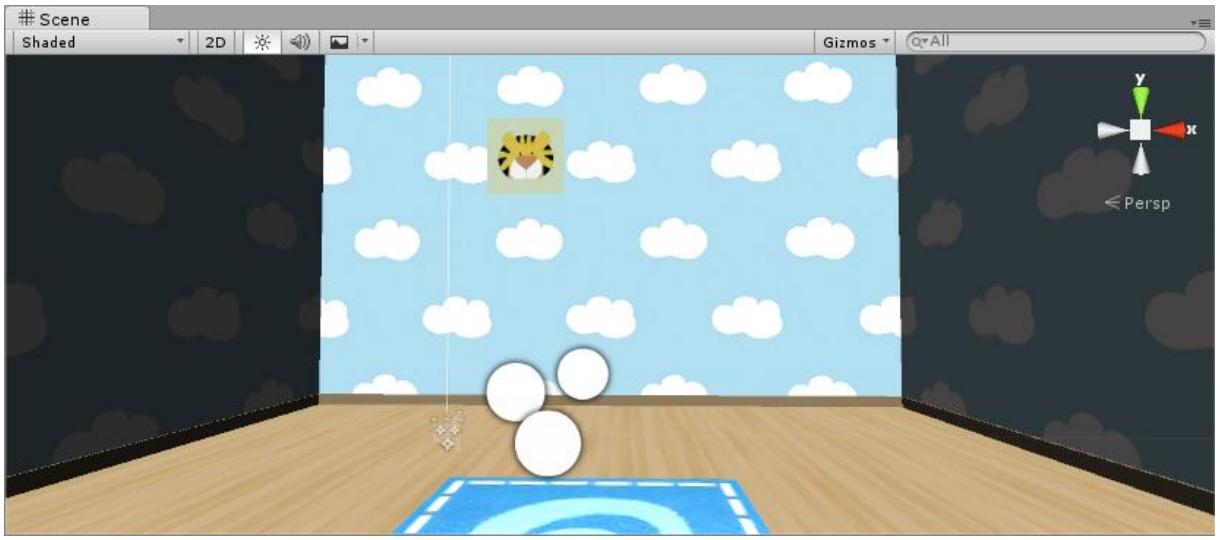
Figura 10. *Hierarchy View* exibindo os elementos presentes na cena.

2.4.3 Scene View

Passos et al. (2009, p. 6) sobre essa *view*: “A janela *Scene* é a forma principal de manipulação dos elementos visuais no editor de cenas da *Unity*, possibilitando a orientação e posicionamento desses elementos com um *feedback* imediato do efeito das alterações efetuadas.” Essa janela possibilita a manipulação e posicionamento dos elementos do jogo na cena.

As cenas na *Unity* são equivalentes aos níveis de um jogo, deverá existir uma cena para cada nível do jogo. A *Scene view* é o local onde se coloca os objetos visuais, conforme ilustrado na Figura 11, como os personagens, construções, terrenos entre outros (LAVIERI, 2015).

Figura 11. *Scene View* e os elementos posicionados em cena.



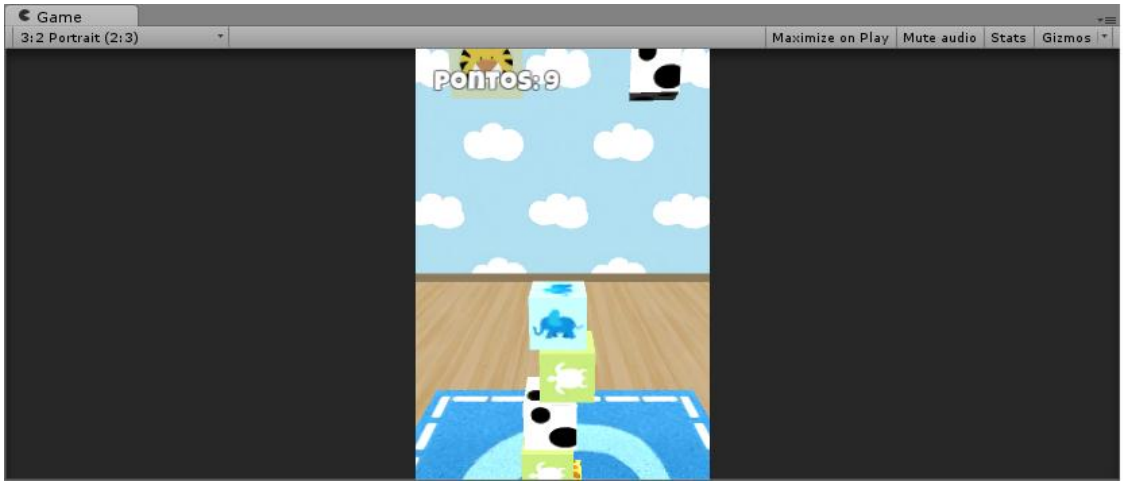
2.4.4 Game View

Conforme Passos et al. (2009, p. 7): "A janela *Game* é responsável pela visualização da aplicação em desenvolvimento da forma que ela será exibida quando finalizada. Nessa janela, pode-se rapidamente ter uma prévia de como os elementos estão se comportando dentro da aplicação" (Figura 12).

Uma das grandes características da *Unity* é que o jogo pode ser executado dentro da própria *engine*. Isso significa que não há necessidade de compilar e executar o jogo fora da plataforma, ele será executado diretamente dentro da *Unity* através dessa *view* (LAVIERI, 2015).

Dependendo da plataforma alvo escolhida (*Android*, *Windows Phone*, *WebPlayer*, etc.), o desenvolvedor tem a opção de escolher a resolução adequada para a execução do jogo em tempo real. No exemplo abaixo, como a plataforma utilizada foi a *Android*, a *Unity* disponibilizou, entre outras, a resolução 3:2 *Portrait* (2:3), que é equivalente a resolução em modo retrato de alguns dispositivos *Android*.

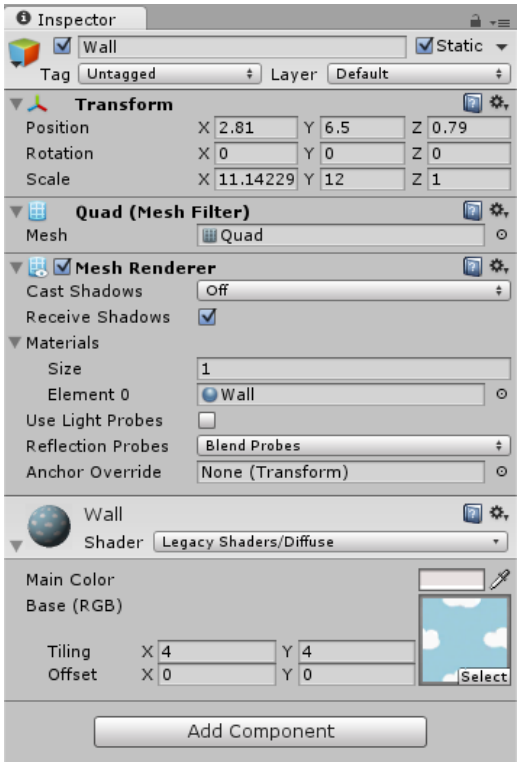
Figura 12. *Game View* executando o jogo.



2.4.5 Inspector View

A janela *Inspector* exibe vários parâmetros de um objeto presente no cenário e também os atributos de seus componentes (PASSOS et al., 2009). Quando os objetos são selecionados em outra *view*, os detalhes deles são exibidos na *Inspector view* (LAVIERI, 2015). Na prática, essa janela exibe quais componentes estão presentes em um determinado objeto do jogo que esteja selecionado (Figura 13).

Figura 13. *Inspector View* e os componentes presentes no objeto *Wall*.



2.4.6 Game Objects

São os objetos do jogo colocados em cena e também a classe base para manipulação de objetos e componentes. “É importante notar que num jogo haverá muitos *game objects*” (FEIJÓ; CLUA; SILVA, 2010, p. 128). O *Game Object* é o *container* base para os objetos, *scripts* e outros componentes no *Unity* (BLACKMAN, 2013).

Os *Game Objects* podem representar qualquer elemento no cenário, sendo desde uma simples câmera ou até um personagem apenas pelos diferentes componentes que possui (PASSOS et al., 2009). Logo, cada objeto do jogo se comportará de uma certa maneira e possuirá determinadas características de acordo com seus componentes.

Passos et al. (2009, p. 9) sobre uma particularidade desses objetos: “Uma observação importante sobre os *Game Objects* é que todos eles já possuem pelo menos o componente *Transform*, responsável pelo seu posicionamento, orientação e escala no sistema referencial da cena.” Todos esses objetos do jogo devem ser posicionados, através do *Transform*, uns próximos aos outros a fim de montar uma cena, fase ou cenário.

2.4.7 Materiais, Shaders e Texturas

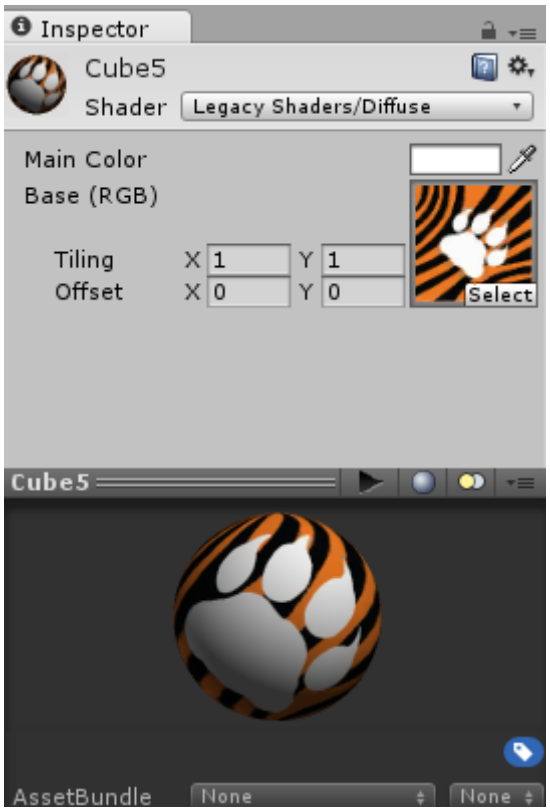
A renderização na *Unity* é feita através dos Materiais, *Shaders* e Texturas. Existe uma relação muito próxima entre esses três elementos (UNITY TECHNOLOGIES, 2015). Esses elementos, ilustrados na Figura 14, se referem a parte visual do jogo, é através deles que torna-se possível a inserção de imagens (texturas) dentro de um objeto 3D simples ou de um objeto modelado.

Material é uma definição de como uma superfície deve ser renderizada, inclui referências a texturas utilizadas, matrizes de cores e muito mais. As opções disponíveis para um material dependem do *shader* que o material está utilizando (UNITY TECHNOLOGIES, 2015). Um Material funciona como um *container* para as propriedades visuais que cada objeto da cena possui (PASSOS et al., 2009).

Os *shaders* são pequenos *scripts* que contém cálculos matemáticos e algorítmos para calcular a cor de cada *pixel* renderizado, baseado na iluminação e na configuração do material (UNITY TECHNOLOGIES, 2015). Existem pedaços específicos de código chamados *shaders* que dirão para sua placa gráfica o que fazer com cada material (BLACKMAN, 2013). Um mesmo material tem comportamentos diferentes de acordo com o *shader* aplicado, alguns podem ser mais reflexivos à luz, outros mais opacos, etc.

Texturas são imagens *bitmap*. Um material pode conter várias texturas, então o *shader* do material pode usar essas texturas no momento de calcular a cor da superfície de um objeto. Além da cor básica (*Albedo*) da superfície de um objeto, as texturas podem representar muitos outros aspectos da superfície de um material, tal como sua refletividade ou aspereza (UNITY TECHNOLOGIES, 2015).

Figura 14. *Material* exibido no *Inspector View* utilizando o *Shader Diffuse* e a imagem da pata de tigre como textura.



2.4.8 Canvas

O *Canvas* é a área em que todos os elementos de *interface* com o usuário devem estar (UNITY TECHNOLOGIES, 2015). O conceito mais importante do sistema de *interface* com o usuário é o objeto *Canvas*, todos os elementos de *interface* estarão contidos no *Canvas*, ele deve ser considerado como um *container* desses elementos (LAVIERI, 2015).

Dentro do *Canvas* estarão localizados todos os painéis, opções, textos, imagens, botões e objetos que são visíveis para o usuário. Esses elementos podem ser interativos, como uma opção do *menu*, ou não interativos, como um logo.

Criar um elemento de *interface* com o usuário, como, por exemplo, uma imagem, usando o *menu GameObjec*t > *UI* > *Image*, irá criar automaticamente o *Canvas* se não houver nenhum na cena. Então, o elemento de *interface* será inserido dentro do *Canvas* (UNITY TECHNOLOGIES, 2015).

2.4.9 Terrenos

Segundo Clua e Bittencourt (2005, p.1333 - 1334), "[...] elaborar um terreno resume-se a criar um mapa de altura adequado para o cenário". Muitas vezes, no processo de desenvolvimento de jogos, é necessária a criação de um ou mais terrenos, eles servirão de base para a sustentação do cenário em que se passará o jogo.

A grande maioria das *game engines* contém módulos geradores de terrenos, através deles é possível criar o formato do terreno, colinas, vales, lagos e muito mais (BLACKMAN, 2013). A *Unity* não é uma exceção e possui todo o necessário para a criação do mapa geográfico do jogo.

2.4.10 Importação de Assets

Na *Unity* os *Assets* são objetos utilizados no jogo, eles podem ser *scripts*, *shaders*, *prefabs* materiais, animações e muitos outros (LAVIERI, 2015). A plataforma disponibiliza uma série de *Assets* básicos para serem utilizados, mas a grande maioria é produzido a partir de outras ferramentas e posteriormente importado à *Unity*.

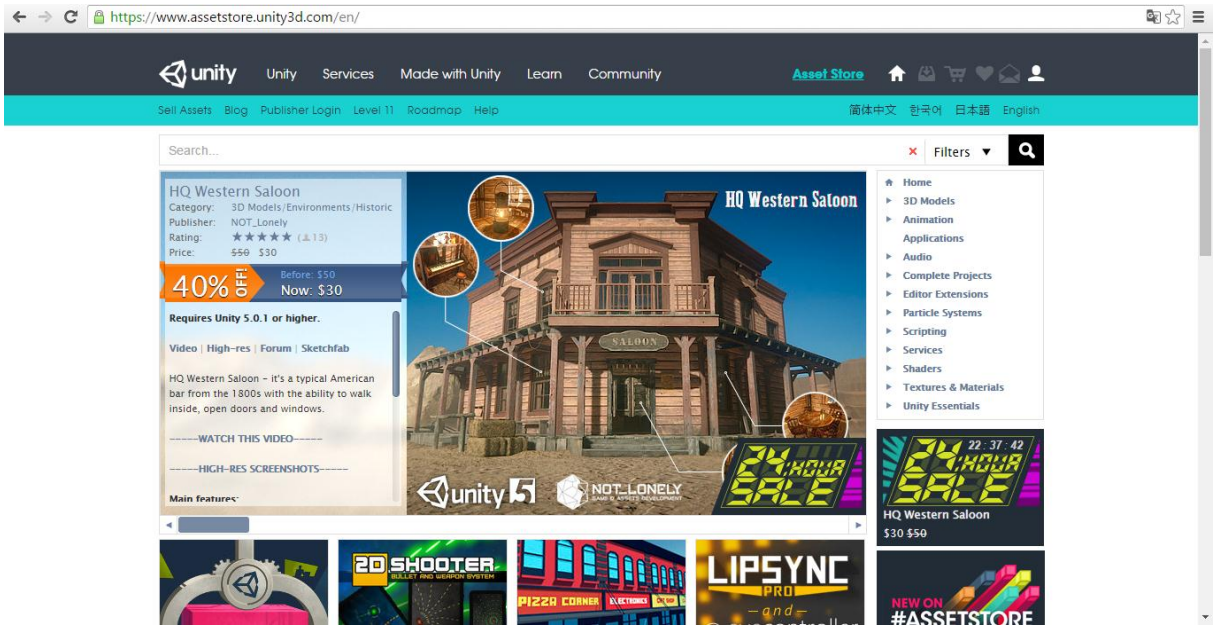
Muitos dos elementos utilizados nos jogos, tais como texturas, modelos 3D, efeitos de som são criados por ferramentas externas. Após sua criação ou edição, esses elementos precisam ser importados para dentro do editor de cenas do motor de jogos (PASSOS et al., 2009).

Na *Unity* esse processo pode ser feito de uma forma simples, bastando arrastar os elementos para dentro de uma pasta da janela *Project*. Após isso a importação é feita automaticamente, ficando disponível imediatamente para o uso na aplicação (PASSOS et al., 2009).

Passos et al. (2009, p. 10) sobre os formatos suportados pela plataforma: "A *Unity* aceita formatos de distribuição populares para modelos 3D (*.FBX*), áudio (*wav*, *mp3*, etc) e texturas (*jpg*, *png*, *bmp* ou mesmo *.PSD* diretamente)."

A *Unity* ainda possui uma loja de comercialização de *Assets*, chamada *Asset Store*, que permite aos criadores de conteúdo venderem ou negociarem seus *Assets* (Figura 15).

Figura 15. *Site* oficial da *Asset Store*.



2.4.11 Física do Jogo

A física do jogo é responsável por imitar os aspectos da realidade no jogo. "A *Unity3D* utiliza internamente o popular motor de física da *PhysX* da *Nvidia* para efetuar simulação física de corpos rígidos[...]" (PASSOS et al., 2009, p. 18). Isso permite que os elementos do cenário interajam entre si sofrendo ação da física, como por exemplo, gravidade, força de um objeto sobre o outro, peso, torque e muito mais.

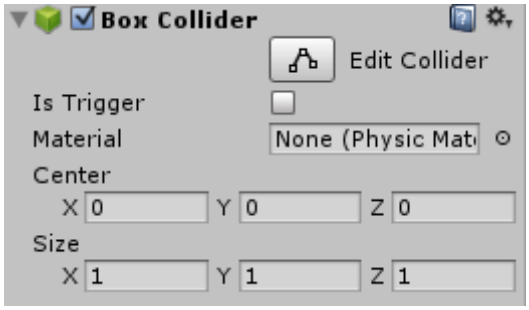
A primeira coisa a se entender em um *game engine* é que a física não é a mesma da realidade, a física aplicada ao jogo é uma série de simulações que podem ser computadas rapidamente para gerar resultados verossímeis (BLACKMAN, 2013).

2.4.11.1 Colisores

Os colisores permitem que os objetos que contém o componente *Rigidbody* (componente responsável pela física dos objetos do jogo), colidam entre si, impedindo que um objeto ultrapasse o outro e permitindo que um objeto aplique força sobre o outro no

momento da colisão. Blackman (2013) complementa: Para que a interação entre os objetos aconteça, será necessário o uso de colisores (Figura 16).

Figura 16. Componente colisor de um dos cubos do jogo.



Passos et al. (2009, p. 18-19) sobre os *colliders*: "Além de sua função principal na simulação física, estes componentes também podem ser utilizados como *triggers*, ou seja, elementos que ativam o processamento de um trecho de código caso ocorra uma colisão com estes".

2.4.12 Scripting

A criação e edição de *scripts* se dão através de três linguagens: "[...] à escolha do programador: *javascript*, *C#* ou *Boo* (um dialeto de *Python*). Não existe penalidade por se escolher uma linguagem ou outra, sendo inclusive possível se usar mais de uma delas em um mesmo jogo" (PASSOS et al., 2009, p. 22).

O *Javascript* é a linguagem da *Web*, a maioria dos modernos *websites* utiliza *Javascript*... (FLANAGAN, 2011). A documentação oficial disponível pela *Unity* utiliza, em sua maioria, o *Javascript* como exemplo (PASSOS et al., 2009). Essa linguagem também foi predominantemente utilizada nesse projeto (Figura 17).

Figura 17. Trecho de código em *Javascript*.

FirstCameraMove.js

```
#pragma strict
private var changeCubeScr : changeCube;

function Awake () {
    changeCubeScr = gameObject.Find("ChangeCube").GetComponent("changeCube");
}

function CallCubes () {
    gameObject.GetComponent(Animator).enabled = false;
    changeCubeScr.Cubes[changeCubeScr.i].GetComponent(MoveCamera).enabled = true;
    changeCubeScr.Cubes[changeCubeScr.i].GetComponent(MoveCube).enabled = true;
    changeCubeScr.Cubes[changeCubeScr.y].GetComponent(MoveCamera).enabled = true;
    changeCubeScr.Cubes[changeCubeScr.y].GetComponent(MoveCube).enabled = true;
    changeCubeScr.Cubes[changeCubeScr.z].GetComponent(MoveCamera).enabled = true;
    changeCubeScr.Cubes[changeCubeScr.o].GetComponent(MoveCamera).enabled = true;
    changeCubeScr.Cubes[changeCubeScr.e].GetComponent(MoveCamera).enabled = true;
}
```

Passos et al. (2009, p. 22) sobre os *scripts***:** "De forma consistente à arquitetura desenvolvida, *scripts* na *Unity3D* são acoplados como componentes de *game objects*. Dessa forma, é importante projetar os *scripts* de maneira modular, ganhando com isso a flexibilidade do reuso".

2.4.13 PlayerPrefs

Guarda e acessa as preferências do jogador entre as sessões de jogo (UNITY TECHNOLOGIES, 2015). Um dos recursos mais interessantes da *Unity* para salvar informações depois que o jogo é encerrado se chama *PlayerPrefs*, dependendo do tipo de informação a ser gravada, caso seja uma informação simples, esse recurso elimina a utilização de um banco de dados. Nesse projeto ele foi utilizado para salvar a melhor pontuação do usuário.

Figura 18. Classe *PlayerPrefs* utilizada para salvar a melhor pontuação do jogo.

```
if (cubeCount > PlayerPrefs.GetInt("bestScore")) {
    PlayerPrefs.SetInt("bestScore", cubeCount);
}
```

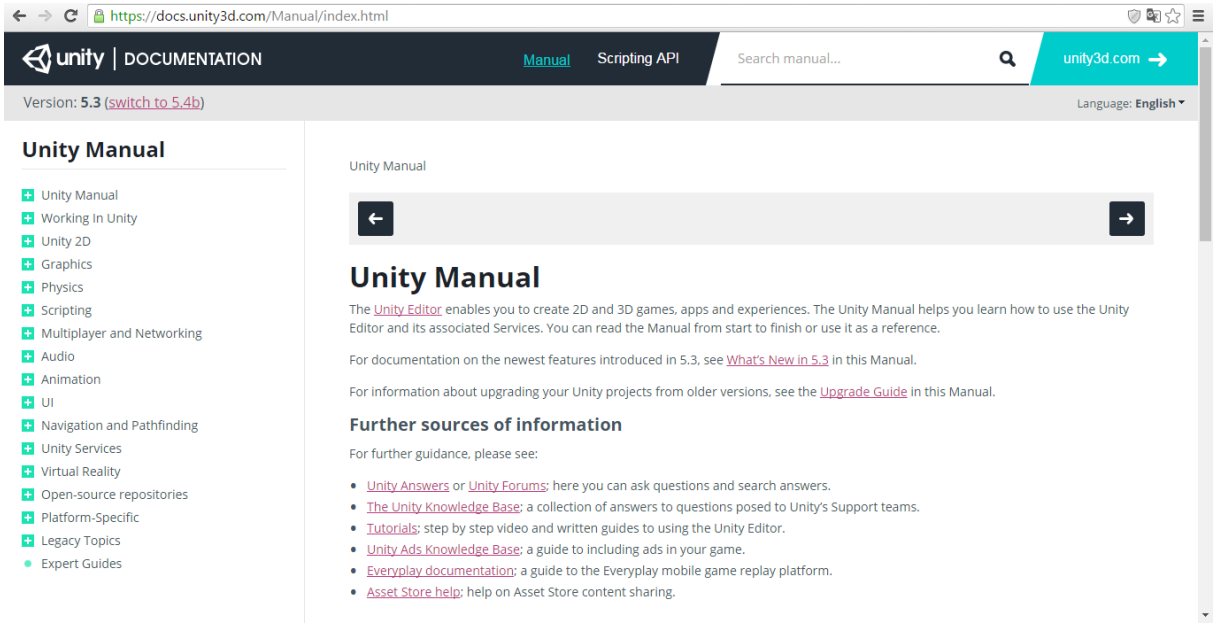
Existem diferentes métodos que podem ser utilizados para salvar o progresso em um jogo, os desenvolvedores utilizam classes especiais para salvar os dados. O *PlayerPrefs*,

ilustrado na Figura 18, pode ser utilizado para armazenamentos básicos, guardando os dados em sua forma nativa, como *integers*, *floats* ou *strings* (BLACKMAN, 2013).

2.4.14 Manual e Suporte

Uma das coisas mais interessantes na *Unity* é a quantidade de conteúdos disponíveis em seu manual *online*, conforme ilustrado pela Figura 19. A *Unity Technologies*, empresa responsável pela documentação oficial, disponibiliza uma série de textos e trechos de programação que proporcionam um grande suporte ao desenvolvedor.

Figura 19: Página oficial da documentação da *Unity*.



O manual foi desenvolvido pra ajudar você a aprender como usar o *Unity* do básico ao avançado (UNITY TECHNOLOGIES, 2015). Além do manual a *Unity Technologies* dispõe de um *site* de respostas e um fórum que permitem a solução de possíveis dúvidas que o desenvolvedor venha a ter.

3 MATERIAL E MÉTODOS

3.1 Material

Todo o material utilizado para o desenvolvimento do jogo foi fornecido pela empresa onde foi realizado o estágio supervisionado.

Especificações do computador utilizado no projeto:

- Processador *Intel Core i5* 750
- 2 *GB* de Memória *RAM*
- Disco Rígido de 500 *GB*
- Sistema Operacional *Windows* 7 *Ultimate*
- Placa de Vídeo *NVIDIA GeForce GT* 430

Versões dos *softwares*:

- *Unity3D 4.6/5.0*
- *Audacity*
- *Adobe Photoshop CS6*
- *Adobe Illustrator CC 17.0.0*

Parte da pesquisa e documentação foi realizada coletando-se as informações de livros, eventos e artigos *online*. As demais bibliografias foram coletadas a partir de livros sobre o conteúdo específico abordado e pela documentação oficial da *Unity*.

3.2 Métodos

O levantamento de requisitos foi fornecido pela empresa onde foi realizado o estágio. A proposta foi criar um jogo voltado para o público infantil que fosse simples, mas possibilitasse o entretenimento desse público.

Num estudo realizado pela empresa foi constatado que o público infantil apresentava uma maior tendência em visualizar propagandas do que os outros públicos, a partir desta premissa, foi proposta então a estratégia de monetização do jogo.

A ideia foi a de desenvolver um jogo eletrônico para empilhar bloquinhos que apresentasse um visual bastante apelativo para o público alvo.

Toda a parte visual das texturas utilizadas no projeto foi criada e editada através do *Adobe Illustrator* e *Adobe Photoshop*. A cena em que o jogo transcorre foi inteiramente desenvolvida por meio da plataforma *Unity 3D*, bem como a programação, movimentação, *score*, painéis e botões.

Finalmente, a prática foi aliada à teoria através desse projeto, coletando-se os dados disponíveis e mais relevantes sobre os temas abordados e relacionando-os com o software desenvolvido.

3.3 Desenvolvimento e Validação

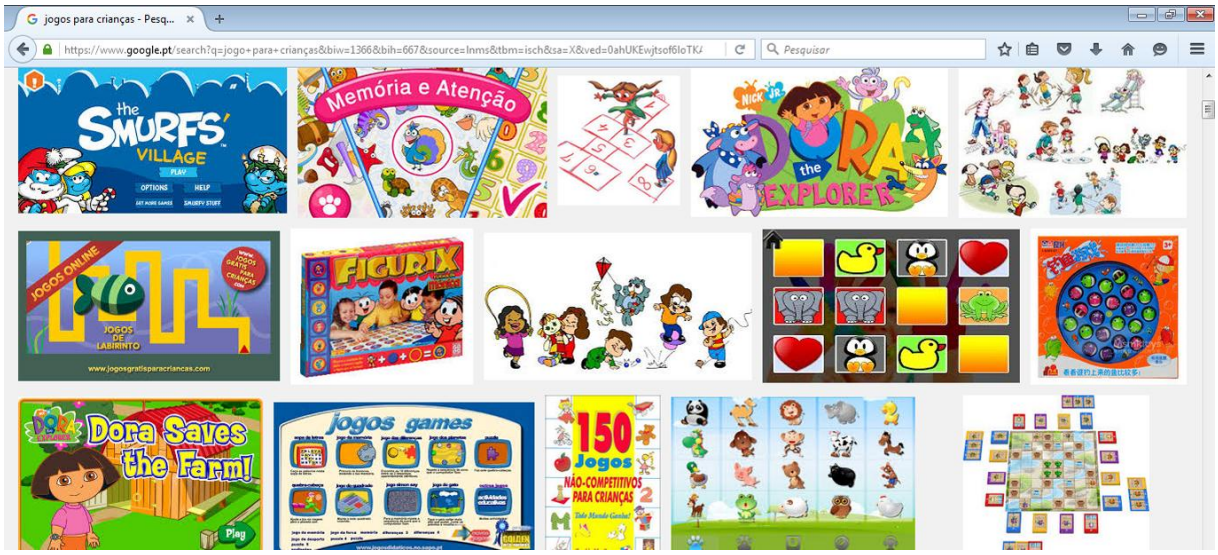
O jogo foi desenvolvido com o intuito de ser relevante para as crianças, satisfazendo suas necessidades de diversão, cultura e aprendizado. Toda a etapa de desenvolvimento foi feita pensando nessa premissa básica, por isso a jogabilidade é simples e os gráficos são voltados para esse público alvo.

A etapa do desenvolvimento foi dividida em algumas partes seguindo-se a rotina passada pela empresa, é importante salientar que, na prática, nem todas as etapas foram sendo cumpridas sucessivamente, muitas vezes algumas etapas foram realizadas antes de outras ou até mesmo consecutivamente:

- Pesquisa de materiais relevantes sobre o tema abordado (Figura 20):

Esse tipo de pesquisa foi feita para permitir um melhor entendimento sobre a área de desenvolvimento do jogo e para servir de inspiração ao desenvolvedor no momento da criação da parte visual do jogo. Para essa pesquisa foi utilizado o mecanismo de busca de imagens do *Google*, por conter um grande acervo de conteúdos visuais que podem servir de base ao desenvolvedor.

Figura 20. Pesquisa sobre o tema jogos para crianças, realizada através do *Google* Imagens.

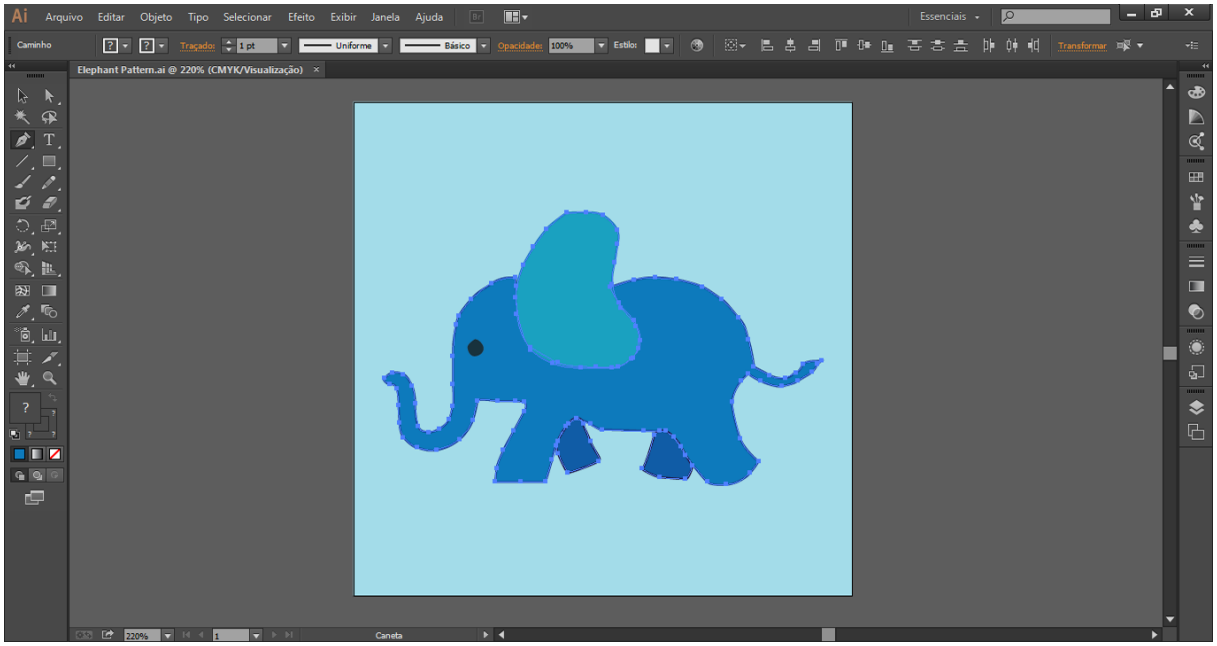


- Definição do tipo de arte a ser utilizada no jogo:

Com algum tempo de pesquisa conseguiu-se obter uma boa noção sobre possíveis temas a serem abordados. Depois de algumas discussões acerca do assunto surgiu a ideia de criar animais com temática infantil para serem utilizados como texturas que seriam aplicadas aos bloquinhos do jogo, essa proposta foi prontamente aceita pela equipe e deu início ao desenvolvimento gráfico do jogo.

- Criação e edição dos elementos visuais do jogo:

Figura 21. Textura de um dos cubos do jogo feita através do *Adobe Illustrator*.



As texturas foram criadas, através da ferramenta caneta do *Illustrator*, inserindo-se pontos âncora até formar a imagem desejada e, por fim, aplicando-se as cores ao objeto. A Figura 21 demonstra uma das imagens utilizadas no jogo, a mesma foi criada por partes, primeiro foi criado o corpo do elefante, depois a orelha e, finalmente, as patas da direita. Esse processo foi feito para permitir que diferentes tons de azul pudessem ser aplicados a cada uma das partes criadas.

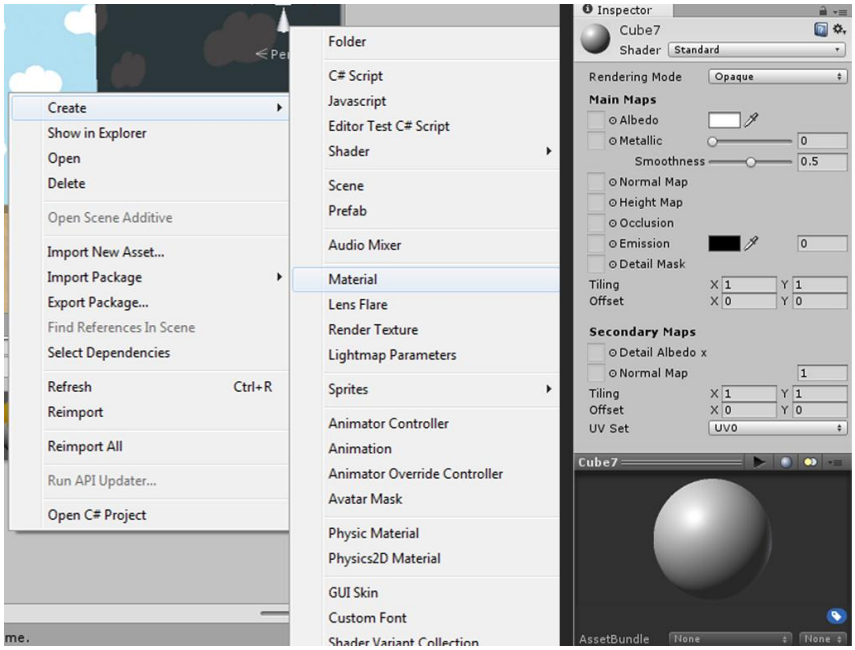
Figura 22. Textura de um dos botões do jogo finalizada utilizando o *Adobe Photoshop*.

O *Adobe Photoshop* foi necessário no momento da criação de algumas das texturas, no exemplo acima ele foi utilizado para a finalização do botão, o cubo atrás do botão foi criado através da ferramenta Retângulo (criando-se três retângulos) juntamente da opção Distorcer, depois disso foi inserido o texto com a ferramenta Texto Horizontal e, finalmente, o desenho da tartaruga foi importado a partir do *Illustrator*. As *layers* do *Photoshop* foram organizadas permitindo assim a disposição dos itens do botão uns acima dos outros (Figura 22).

- Importação das texturas para a plataforma e adição das texturas aos materiais:

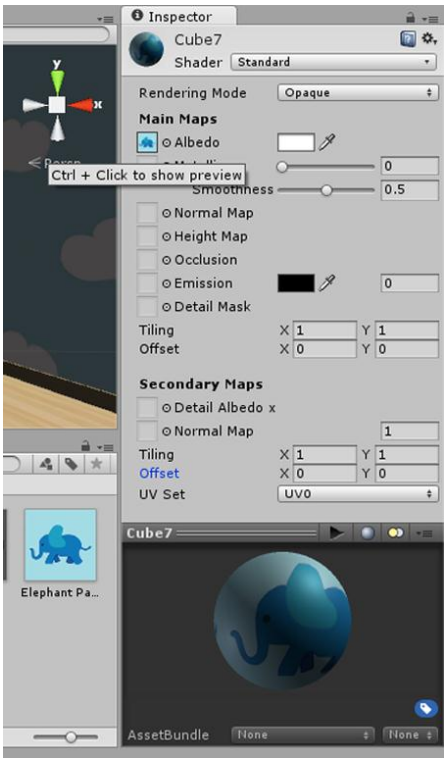
Todas as texturas criadas foram importadas para a *Unity* e aplicadas a um determinado material, criado dentro da própria plataforma, para então serem aplicadas aos objetos 3D, nesse caso foi utilizado o objeto *Cube*. Para criar um material, basta selecionar uma pasta na janela *Project* apertar com o botão esquerdo do mouse dentro da pasta, escolher a opção *Create* e depois *Material* (Figura 23).

Figura 23. Criando material e informações do material criado.



Uma textura nunca pode ser adicionada diretamente a um objeto do jogo, ela sempre necessita ser aplicada a um material para que este seja inserido como um componente ao objeto 3D. Para aplicar a textura ao material basta arrastá-lo da janela *Project* até o espaço reservado pra textura no *Inspector* do material selecionado (Figura 24).

Figura 24. Textura aplicada ao material criado.



- Criação da primeira cena do jogo:

A primeira cena buscou reunir todos os bloquinhos anteriormente criados juntamente de uma animação de movimentação da câmera. Essa cena exibe todos os tipos de bloquinhos existentes no jogo (Figura 25).

Figura 25. Cena inicial do jogo.

- Criação da cena principal e *scripting*:

Nessa etapa foi criada a cena em que o jogo realmente acontece, a ideia inicial era montar um quarto parecido com um quarto de criança bem simplificado. Além disso, diversos *scripts* foram criados durante o desenvolvimento dessa cena, desde *scripts* que controlam os efeitos sonoros até *scripts* que controlam o sorteio dos cubos no início do jogo. Um dos *scripts* que controla os cubos do jogo é exibido na Figura 26.

Figura 26. Trecho de um dos *Scripts* que comandam a física dos cubos no jogo.

```
39      if (mainScr.cameraEnd == false) {
40          var ray : Ray = Camera.main.ScreenPointToRay(Input.mousePosition);  //Raycast na posiçao do mouse
41          if (Physics.Raycast (ray,hit)) {
42              if (hit.transform.name == "Cube") {
43                  if (a == 0) {
44                      a++; //Somente 1 cubo selecionado
45                  }
46              }
47              if (hit.transform.position.x == transform.position.x && hit.transform.position.y == transform.position.y && a == 1 &&mainScr.a == false) {
48                  selected = true;
49                  mainScr.reset = false;
50                  mainScr.spawnTrue = false;
51                  soundScr.PlayAudio(1);
52              }
53          }
54      }
```

4 RESULTADOS E DISCUSSÃO

Para a discussão dos resultados serão analisados as telas e funcionalidades do jogo bem como os dados da página principal do jogo disponível na plataforma *online Google Play*.

Figura 27. Tela principal do jogo.

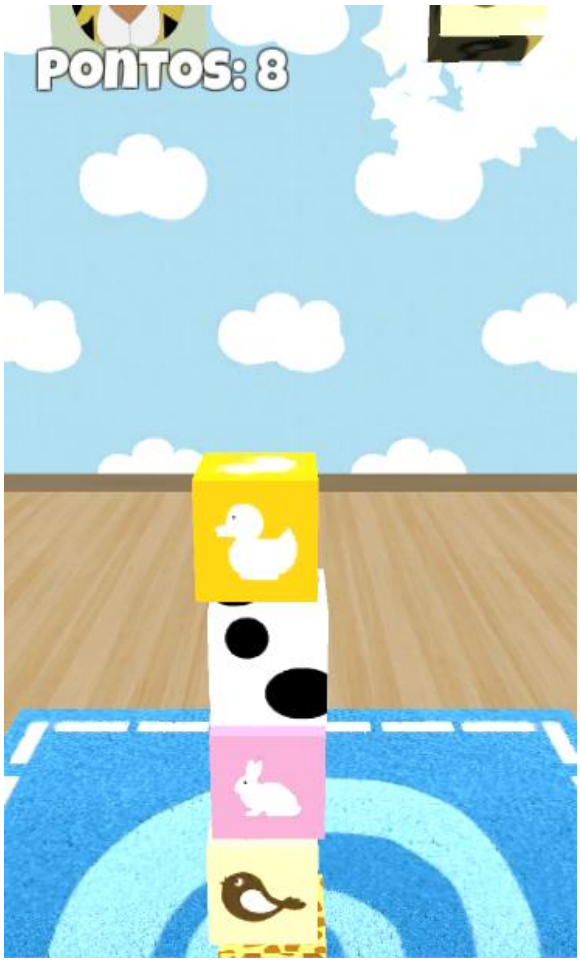
A Figura 27 demonstra a tela inicial do jogo, nela estão disponíveis os botões de iniciar um novo jogo e as opções gerais. Além disso, há uma animação e movimentação de câmera para melhor interatividade e visual do jogo.

O *menu* inicial é bem simples e possui algumas opções auxiliares do jogo tais como créditos, informações gerais e uma opção para habilitar ou desabilitar o som. Após consultar o *menu* é possível voltar a tela inicial através do botão voltar (Figura 28).

Figura 28. *Menu* inicial

A execução do jogo acontece a partir do momento em que se escolhe a opção novo jogo na tela principal. Nesse momento é possível empilhar os bloquinhos que estão na cena, de forma que novos bloquinhos vão surgindo à medida em que os anteriores são empilhados (Figura 29). O jogo termina no momento em que a pilha de blocos se desequilibra e cai no chão, gerando a pontuação final do jogador.

Figura 29. Jogo em execução.

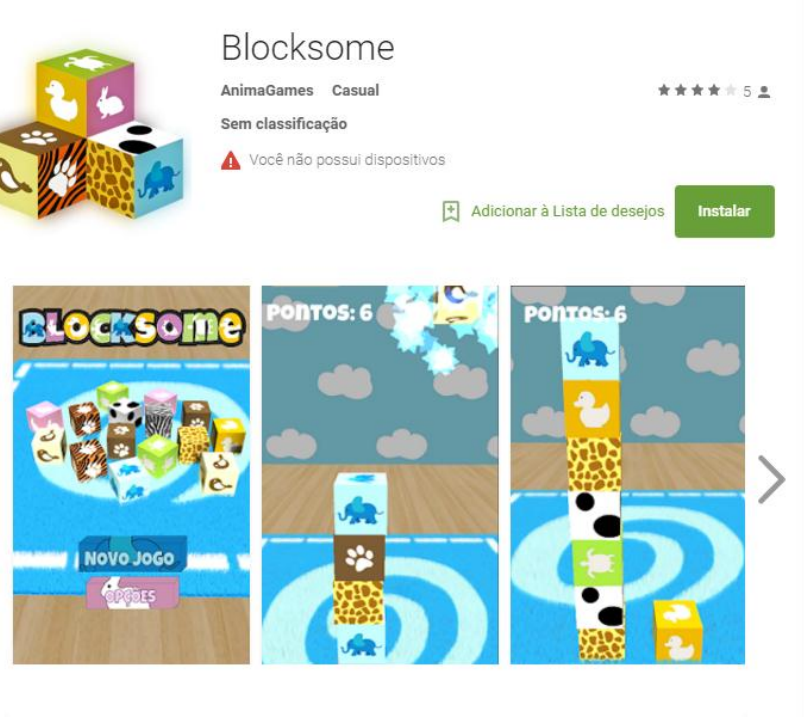


No momento em que a pilha de blocos é derrubada, o jogo exibe a tela final com a pontuação atual e com a maior pontuação alcançada até o momento. Nessa tela é possível reiniciar o jogo diretamente clicando no botão jogar novamente ou retornar ao *menu* inicial clicando no botão voltar ao *menu* (Figura 30).

Figura 30. Tela final e pontuação do jogo.

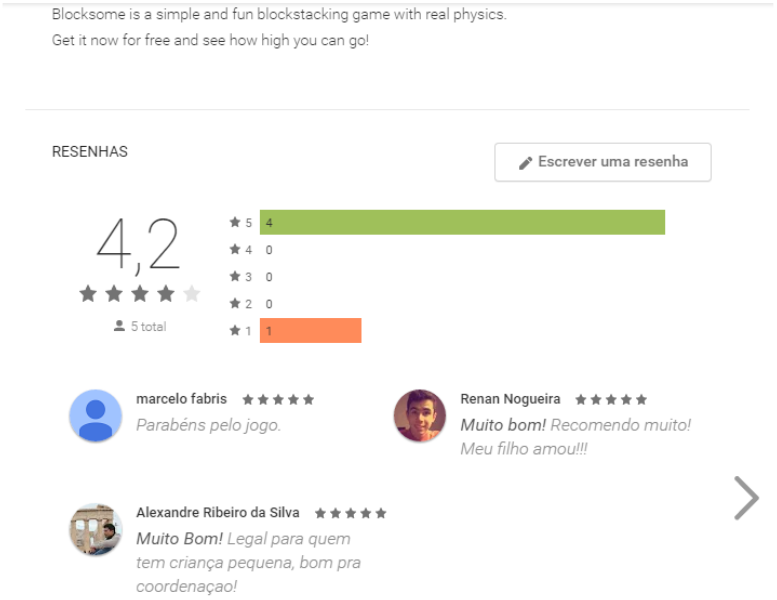
Os detalhes da publicação do jogo são analisados através da página do jogo disponível na plataforma *Google Play*. A Figura 31 apresenta os detalhes iniciais do jogo, nela encontram-se o nome (*Blocksome*) e categoria do jogo (Casual), nome da conta *Google* utilizada para a publicação do jogo (*AnimaGames*) e um botão que permite o *download* e posterior instalação do *app*.

Figura 31. Visão geral do *app*, parte 1.



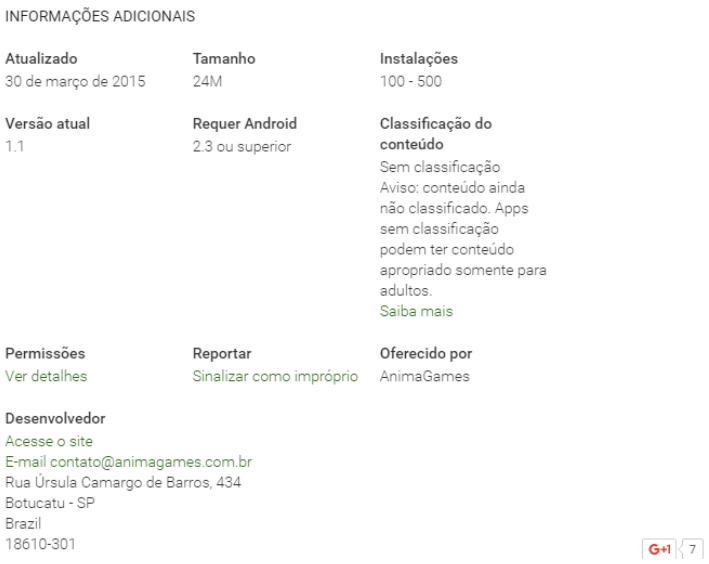
Além das informações básicas, na segunda parte da página está disponível um breve comentário da empresa sobre o jogo e as resenhas feitas pelos usuários. Esta seção permite o envio de uma nota, que varia de uma a cinco estrelas, e também possibilita o envio de comentários (Figura 32).

Figura 32. Visão geral do *app*, parte 2.



Na última parte da página, ilustrada pela Figura 33, estão presentes os últimos detalhes sobre o jogo, o número médio de instalações, a versão do jogo, versão mínima do *Android* necessária para a execução correta do jogo, o tamanho do *app*, as informações do desenvolvedor, etc.

Figura 33. Visão geral do *app*, parte 3.



A entrada da empresa com esse jogo no mercado infantil demonstrou uma pequena retenção de público, o que não quer dizer que tenha sido algo malsucedido ou irrelevante, provavelmente pelo fato da temática bem apelativa voltada para as crianças, o jogo acabou não dando tanta repercussão em relação a outros públicos, que possivelmente tem um maior acesso a esse tipo de aplicativo.

A experiência de criar jogos com uma temática inovadora e não antes acessada pela empresa, proporcionou a mesma entender melhor como esse tipo de mercado funciona. A criação desse jogo foi fundamental para compreender esse processo.

5 CONCLUSÃO

O jogo foi implementado e publicado com sucesso e está disponível para *tablets e smartphones* que utilizam o sistema operacional *Android 2.3* ou superior. Além disso, o jogo também foi revisado e aprovado pela empresa e está atendendo ao público infantil.

A ferramenta *Unity 3D* se mostrou eficiente do início ao fim desse projeto, permitindo assim um desenvolvimento ágil e completo do jogo. É altamente recomendado o uso dessa plataforma independentemente do tipo, classificação e tamanho do jogo a ser desenvolvido.

Por fim, essa pesquisa demonstrou os materiais necessários, etapas de produção, planejamento e resultados através dos exemplos publicados. Espera-se que ela possa servir de auxílio a todos aqueles que desejam ampliar seus conhecimentos na área do desenvolvimento de jogos e explorar o constante crescimento do mercado de *games* no Brasil e no mundo.

REFERÊNCIAS

ADOBE SYSTEMS INCORPORATED. **Adobe Illustrator.** 2016. Disponível em: <http://www.adobe.com/br/products/illustrator.html>. Acesso em: 11 maio 2016.

ADOBE SYSTEMS INCORPORATED. **Adobe Photoshop.** 2016. Disponível em: <http://www.adobe.com/br/products/photoshop.html>. Acesso em: 10 maio 2016.

ANDRADE, M. S. Apresentação do programa e de seus conceitos básicos. In: ANDRADE, M. S.**Adobe Photoshop CS6.** São Paulo: Senac São Paulo, 2013. Cap. 1. p. 7-9.

BLACKMAN, S. Introduction to Game Development: New to Real Time vs. Pre-render. In: BLACKMAN, S. **Beginning 3D Game Development:** All-In-One, Multi-Platform Game Development. 2. ed. New York: Technology In Action, 2013. Cap. 1, p. 20.

BLACKMAN, S. Unity UI Basics - Getting Started: Menus. In: BLACKMAN, S. **Beginning 3D Game Development:** All-In-One, Multi-Platform Game Development. 2. ed. New York: Technology In Action, 2013. Cap. 3, p. 34,41.

BLACKMAN, S. Action Objects: Colliders. In: BLACKMAN, S. **Beginning 3D Game Development:** All-In-One, Multi-Platform Game Development. 2. ed. New York: Technology In Action, 2013. Cap. 8, p. 263.

BLACKMAN, S. Physics and Special Effects: Physics. In: BLACKMAN, S. **Beginning 3D Game Development:** All-In-One, Multi-Platform Game Development. 2. ed. New York: Technology In Action, 2013. Cap. 11, p. 395.

BLACKMAN, S. Menus and Levels: Level Save and Load. In: BLACKMAN, S. **Beginning 3D Game Development:** All-In-One, Multi-Platform Game Development. 2. ed. New York: Technology In Action, 2013. Cap. 19, p. 748.

BOTELLO, C. Getting to Know Illustrator. In: BOTELLO, C. **Adobe Illustrator CS6 Revealed.** Nova York: Delmar, Cengage Learning, 2013. Cap. 1, p. 2.

CLUA, E. W. G.; BITTENCOURT, J. R. Desenvolvimento de jogos 3D: concepção, design e programação. In: JORNADAS DE ATUALIZAÇÃO EM INFORMÁTICA (JAI), 24., CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 24., 2005. **Anais ...** São Leopoldo, 2005. p. 1313-1357.

CREIGHTON, R. H. That's One Fancy Hammer! In: CREIGHTON, R. H. **Unity 3D Game Development by Example:** A seat-of-your-pants manual for building fun, groovy little games quickly. Birmingham: Packt Publishing Ltd, 2010. Cap. 1. p. 7-28.

FEIJÓ, B.; CLUA, E.; SILVA, F. S. C. Programação Gráfica. In: FEIJÓ, B.; CLUA, E.; SILVA, F. S. C. **Introdução à ciência da computação com jogos:** Aprendendo a programar com entretenimento. Rio de Janeiro: Elsevier, 2010. Cap. 8. p. 128.

FLANAGAN, D. Introduction to Javascript. In: FLANAGAN, D. **JavaScript:** the definitive guide. 6. ed. Sebastopol: O'reilly Media, Inc, 2011. Cap. 1. p. 1.

GOOGLE. **Google Acadêmico.** Disponível em: <http://scholar.google.com.br>. Acesso em: 29 ago. 2016.

GOOGLE. **Google Imagens.** Disponível em: <https://www.google.com.br/imghp?hl=pt-PT>. Acesso em: 02 set. 2016.

GOOGLE. **Google Play:** Blocksome**.** Disponível em: <https://play.google.com/store/apps/details?id=com.animagames.blocksome>. Acesso em: 02 set. 2016.

GRUPO NZN. **Da cobrinha ao realismo: como os jogos de celular evoluíram com o tempo.** 2016. Disponível em: <http://www.tecmundo.com.br/video-game-e-jogos/102175-cobrinha-realismo-jogos-celular-evoluiram-tempo.htm>. Acesso em: 03 maio 2016.

LAVIERI, Dr. E. Adding a Graphical User Interface: Unity 5's UI System. In: LAVIERI, Dr. E. **Getting Started with Unity 5:** Leverage the power of Unity 5 to create amazing 3D games. Birmingham: Packt Publishing, 2015. Cap. 6. p. 114.

LAVIERI, Dr. E. Getting Jiggy with the Unity Interface: Views. In: LAVIERI, Dr. E. **Getting Started with Unity 5:** Leverage the power of Unity 5 to create amazing 3D games. Birmingham: Packt Publishing, 2015. Cap. 1. p. 10-14.

LAVIERI, Dr. E. Polishing and Optimizing the Game: Sight and sound. In: LAVIERI, Dr. E. **Getting Started with Unity 5:** Leverage the power of Unity 5 to create amazing 3D games. Birmingham: Packt Publishing, 2015. Cap. 7. p. 131.

LAVIERI, Dr. E. Working with Assets: Using Blender to create assets for your game. In: LAVIERI, Dr. E. **Getting Started with Unity 5:** Leverage the power of Unity 5 to create amazing 3D games. Birmingham: Packt Publishing, 2015. Cap. 3. p. 44-45,57.

MAZZONI, D. **Audacity.** Disponível em: <http://www.audacityteam.org/>. Acesso em: 10 maio 2016.

PASSOS, E. B. et al. Tutorial: Desenvolvimento de jogos com unity 3d. In: BRAZILIAN SYMPOSIUM ON GAMES AND DIGITAL ENTERTAINMENT, 8., 2009. **Anais …** Rio de Janeiro, 2009. p. 1-30.

UNITY TECHNOLOGIES. **Asset Store.** 2016. Disponível em: <https://www.assetstore.unity3d.com/>. Acesso em: 02 set. 2016.

UNITY TECHNOLOGIES. **Unity Documentation:** Audio Source. 2015. Disponível em: < http://docs.unity3d.com/Manual/class-AudioSource.html>. Acesso em: 08 fev. 2016.

UNITY TECHNOLOGIES. **Unity Documentation:** Canvas. 2015. Disponível em: <http://docs.unity3d.com/Manual/UICanvas.html>. Acesso em: 09 fev. 2016.

UNITY TECHNOLOGIES. **Unity Documentation:** Materials, Shaders & Textures. 2015. Disponível em: <http://docs.unity3d.com/Manual/Shaders.html>. Acesso em: 27 jan. 2016.

UNITY TECHNOLOGIES. **Unity Documentation:** PlayerPrefs. 2015. Disponível em: <http://docs.unity3d.com/ScriptReference/PlayerPrefs.html>. Acesso em: 15 jan. 2016.

UNITY TECHNOLOGIES. **Unity Documentation:** Unity Manual. 2015. Disponível em: <http://docs.unity3d.com/Manual/index.html>. Acesso em: 28 dez. 2015.

Botucatu, ____ de _____ de 2017.

_____
Thiago Augusto Jorge

De Acordo:

_____
Prof. Dr. Gustavo Kimura Montanha
Orientador (a)

_____
Prof. Esp. Rogério Ferreira Sgoti
Coordenador do Curso de Análise e Desenvolvimento de Sistemas