UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
Bacharelado em Sistemas de Informação
Magno Juliano Gonçalves Santos

DESENVOLVIMENTO DE JOGO EDUCACIONAL PARA DISPOSITIVOS MÓVEIS PARA AUXILIAR O PROCESSO DE ENSINO/APRENDIZAGEM DA ORTOGRAFIA UTILIZANDO A FERRAMENTA *UNITY*

Diamantina
2019

Magno Juliano Gonçalves Santos

DESENVOLVIMENTO DE JOGO EDUCACIONAL PARA DISPOSITIVOS MÓVEIS PARA AUXILIAR O PROCESSO DE ENSINO/APRENDIZAGEM DA ORTOGRAFIA UTILIZANDO A FERRAMENTA *UNITY*

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação do Departamento de Computação, como parte dos requisitos exigidos para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Luciana Pereira de Assis

Diamantina
2019

Magno Juliano Gonçalves Santos

Desenvolvimento de jogo educacional para dispositivos móveis para auxiliar o processo de ensino/aprendizagem da ortografia utilizando a ferramenta Unity

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação do Departamento de Computação, como parte dos requisitos exigidos para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Profa. Dra. Luciana Pereira de Assis

Data de aprovação 28/01/19.

Luciana Pereira de Assis

Profa. Dra. Luciana Pereira de Assis
Departamento de Computação - UFVJM

Prof. Dr. Alessandro Vivas Andrade
Departamento de Computação - UFVJM

Profa. Dra. Claudia Beatriz Berti
Departamento de Computação - UFVJM

Diamantina
2018

Dedico esse trabalho a Deus primeiramente, a minha família e a todos que contribuíram para a sua realização.

AGRADECIMENTOS

Agradeço primeiramente a Deus por mais um dia de vida, saúde e por me capacitar para que esse trabalho pudesse ser realizado, toda honra e glória a Ele. À minha orientadora Profa. Luciana Pereira de Assis pelo acompanhamento do trabalho e por todos os ensinamentos que me proporcionou. Agradeço meus pais Nivaldo Aparecido dos Santos, Magda Fernanda G. dos Reis Santos pelo apoio nos momentos que mais foi preciso e por tudo que fizeram por mim até hoje. Agradeço minha namorada e futura noiva Karina Vila Verde por ser compreensiva, por estar ao meu lado em todos os momentos e por ser a luz que guia minha vida. Agradeço, também, meu amigo Luíz Otávio Mendes Gregório por ser mais que um amigo - mas um irmão - que eu adquiri em Diamantina e por me ajudar a concluir essa etapa da minha formação. Sou grato também a minha equipe e amigos Caroline e Natan por termos juntos finalizado uns 80% das atividades e trabalhos deste curso. A todos que de forma direta e indireta contribuíram para a realização deste trabalho.

RESUMO

O presente trabalho tem por objetivo auxiliar o processo de ensino/aprendizagem da ortografia, a partir do uso de um aplicativo digital para dispositivos móveis. O aplicativo proposto estimula o resgate do lúdico do ambiente escolar, possibilitando aprendizado somado à diversão e entretenimento. Vale ressaltar que este trabalho representa uma extensão do aplicativo Laça-Palavras que apresenta o mesmo escopo de atuação da aplicação atual. Este surgiu da necessidade de implementar novas funcionalidades e adaptar às demandas que surgiram no desenvolvimento da primeira versão proposta. A aplicação foi projetada para atender a uma faixa de etária de crianças entre 8 a 10 anos. Busca-se com a realização deste trabalho apresentar o processo de desenvolvimento de jogos que embasou a implementação da solução e um manual contendo as principais ferramentas que constituem o motor de jogo *Unity*, tecnologia utilizada no trabalho, que possui um conjunto de bibliotecas para o desenvolvimento de jogos eletrônicos e aplicações gráficas. A metodologia empregada divide-se nas seguintes etapas: mapeamento das irregularidades ortográficas mais recorrentes na língua portuguesa a fim de delimitar a área de atuação do aplicativo, criação de uma base de dados de palavras para um mesmo campo semântico frequentes no vocabulário das crianças e que atendam ao quesito de mais de uma representação gráfica para o mesmo som, com base nesses dados, foi proposto o aplicativo abordando tais dificuldades ortográficas.

Palavras-chave: ortografia, dispositivos móveis, aplicativo. lúdico, aprendizagem.

ABSTRACT

The present work aims to help the teaching / learning process of the spelling, from the use of a digital application to mobile devices. The proposed application stimulates the rescue of the playfulness of the school environment, enabling learning in addition to entertainment. This application is designed to meet an age range of children ages 8 to 10. The aim of this work is to present the process of game development that based the implementation of the solution and a manual containing the main tools that make up the game engine Unity, technology used in the work, which has a set of libraries for the development of electronic games and graphic applications. The methodology used is divided into the following steps: mapping of the most recurrent orthographic irregularities in the Portuguese language in order to delimit the area of application of the application, creation of a database of words for a same semantic field frequent in the vocabulary of the children and that attend to the question of more than one graphic representation for the same sound and, based on these data, the application was proposed addressing such difficulties orthographic.

Keywords: spelling. mobile devices. application. ludic. learning.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A utilização da tecnologia em qualquer faixa etária se tornou uma realidade no mundo contemporâneo. No seu trabalho, Pereira (2016, p.1) explica que "[...] Os jogos e brincadeiras utilizadas de forma adequada como recurso pedagógico poderão contribuir para o processo de aprendizagem das crianças na escola, especialmente na educação infantil [...]". Isso se deve a característica lúdica que os jogos apresentam, sendo capazes de reter o interesse da criança favorecendo o processo de ensino/aprendizagem. O autor atesta essa afirmação ao explicar que:

> "A compreensão do lúdico na área escolar infantil não se realiza apenas como uma atividade recreativa, mas a partir do reconhecimento de seu potencial como recurso pedagógico para o processo de aprendizado. Portanto, é fundamental que os educadores da educação infantil entendam que, no brincar as crianças também aprendem e se desenvolvem (PEREIRA, 2016, p.1)."

A utilização de jogos eletrônicos para dispositivos móveis estimula o resgaste do lúdico no ambiente escolar, possibilitando aprendizado somado à diversão. Dessa forma, a influência cada vez mais intensa de aplicativos móveis alcançou o âmbito da educação, tornando-os uma ferramenta de auxílio na aprendizagem de conteúdos escolares, o que propicia o surgimento de uma nova mobilidade educacional denominada *m-learning* ou *mobile learning* correspondente a aprendizagem apoiada por dispositivos móveis. A modalidade *m-learning* consiste no uso educacional de dispositivos móveis e portáteis em atividades de ensino e aprendizagem (MÜLBERT; PEREIRA, 2011).

Segundo Onça (2014, p.11) "nas últimas três décadas, os jogos eletrônicos, popularmente conhecidos como *games*, ampliaram gradativamente seu espaço como uma das expressões culturais mais efervescentes da contemporaneidade". Um estudo realizado em 2014 pela Associação Brasileira de Desenvolvedores de Games (Abragames), em resposta à chamada pública do Banco Nacional de Desenvolvimento Econômico e Social (BNDES), apurou que cerca de 61 milhões de brasileiros se entretém com jogos online e eletrônicos, estatística que se deve à popularização de *smartphones* e *tablets* que permitem o acesso aos jogos a qualquer hora e em qualquer lugar.

O número de trabalhos presentes na literatura relacionados à aprendizagem móvel também tem crescido, conforme panorama apresentado por Mülbert e PEREIRA (2011) entre os anos de 2001 a 2010. Da mesma forma, Wu *et al.* (2012) apresenta uma síntese de 164 trabalhos entre os anos 2003 a 2010 com foco na aprendizagem móvel e na sua eficácia. As abordagens educacionais para aprendizagem móvel geralmente são apresentadas em formato de *indie games*, jogos eletrônicos implementados por produtores independentes. A *Unity Technologies* viabiliza o desenvolvimento de tais jogos através da ferramenta *Unity* que corresponde a um *game engine* (motor de jogo) utilizado para implementação de jogos, onde é possível unir animações, imagens, áudios e outros objetos para geração do aplicativo final. Uma das principais vantagens do *Unity*

é permitir a geração de aplicativos para diferentes dispositivos e sistemas: Android, IOS, PC, WebGL e até *consoles* PS4 e Xbox One (UNITY, 2018).

1.1 Jogo educacional para auxiliar no ensino da ortografia

Segundo Assis *et al.* (2017) o processo de aprendizagem de uma língua no ambiente formal passa pelo desenvolvimento de habilidades orais e escritas. Sendo papel da escola proporcionar meios para que os alunos possam entrar em contato com a variante padrão, reconhecendo-a e utilizando-a de forma proficiente. A aquisição dessas habilidades são de fundamental importância no processo de alfabetização, pois asseguram as capacidades de leitura e escrita do aprendiz. Soares (2016) discute sobre o efeito da regularidade ortográfica no que tange a leitura e a escrita que:

> "[...] a aprendizagem do sistema gráfico de notação alfabética, que torna a criança alfabética, deve completar-se, sobretudo no caso da escrita, com a aprendizagem das convenções que impõem determinada grafia em casos em que outras grafias são também possíveis: a aprendizagem da norma ortográfica, que vai tornando a criança ortográfica, para além de alfabética. Talvez isso explique a relevância que se atribui, na alfabetização, e no ensino da língua escrita em geral, à questão da aprendizagem da ortografia entendida predominantemente na direção grafofonêmica, isto é, na direção da escrita, já que, [...] o efeito da regularidade sobre a leitura do português brasileiro é pouco significativo, pois a norma ortográfica interfere pouco sobre o reconhecimento de palavras; ao contrário, interfere significativamente sobre a escrita, ou seja: a ortografia do português brasileiro é mais transparente para a leitura que para a escrita (SOARES, 2016, p.295-296)."

Zorzi (2009) complementa essa perspectiva ao afirmar que:

> "[...] aprender a escrever implica compreender uma série de propriedades ou aspectos da língua escrita que fazem parte do sistema ortográfico [...] compreender que uma mesma letra pode representar vários sons, assim como um mesmo som pode ser representado por diversas letras (ZORZI, 2009, p. 43)."

Dentre os conteúdos listados no processo de aprendizagem da língua portuguesa (LP), destaca-se a ortografia. Assis *et al.* (2017) explica em seu trabalho que é importante:

> "[...] ressaltar o caráter convencional da norma ortográfica, na qual se identificam duas grandes categorias. A primeira delas e aquela que se verifica a existência de palavras grafadas controladas por uma regra, em que a sua forma de dicionario possui uma motivação para a escrita, como é o caso de palavras derivadas (*casinha* é escrita com *s* porque vem do substantivo primitivo casa). De forma diferente, a segunda categoria, as palavras irregulares, não são controlada por uma regra: nesse caso, há letras concorrentes, não sendo possível identificar uma razão sistemática para a escolha de uma ou outra letra, como ocorre em casa (que poderia ser grafada com z, se essa fosse a convenção) (ASSIS *et al.*, 2017, p. 609 - 610)."

Assim sendo, o desenvolvimento de um jogo educacional que articule reflexões quanto a categoria de palavras irregulares não controladas por uma regra, se torna um campo a

ser explorado. Baseado na hipótese de ALVARENGA (1995), o processo de ensino/aprendizagem de dificuldades ortográficas que não são motivadas por uma regra, depende do uso da memória e que, portanto, cada palavra deve ser aprendida individualmente (ASSIS *et al.*, 2017). Diante dessa perspectiva o uso de jogos apresenta-se como uma ferramenta adequada para garantir a memorização e que as crianças consigam grafar corretamente as palavras de acordo com a convenção de dicionário.

1.2 Objetivos

O objetivo geral desse trabalho é desenvolver um jogo educacional que auxilie no processo de ensino/aprendizagem da ortografia, levando-se em consideração a categoria de palavras grafadas que não são controladas por uma regra elucidada no trabalho de Assis *et al.* (2017), além de apresentar uma descrição minuciosa da ferramenta de desenvolvimento *Unity* fornecendo uma explicação sobre as etapas que compõem o processo de implementação de um jogo considerando-se desde a concepção inicial até o procedimento final de desenvolvimento.

Este trabalho representa uma extensão do aplicativo Laça Palavras, tal aplicação consiste em apresentar ao usuário quatro grupos de palavras que se enquadram no caso da concorrência entre dígrafos (LEAL, 2016). Dentre os grupos elucidas por Leal (2016, p.43) "[...] contém as sílabas com "S" ou "Z", mas que possui som de [z] (exemplo a palavra "casa"), palavras que contém sílabas com "SS" ou "Ç", mas que possui som de [s] (exemplo a palavra "passo"), palavras que contém sílabas com "L" ou "U", mas que possui som de [u] (exemplo a palavra "alça") e palavras que contém sílabas com "G" ou "J", mas que possui som [g] (exemplo a palavra "majestade")."

Diante do exposto propõe-se o aplicativo *Grapphia* que apresenta o mesmo escopo do Laça-Palavras acrescido de alterações gráficas e de armazenamento de dados. Visando uma melhor experiência de usuário a *interface* gráfica do aplicativo foi aprimorada, também desenvolveu-se um módulo de apresentação de livros contendo as histórias ilustradas que proporcionam ao usuário um primeiro contato com as palavras contendo dígrafos concorrentes. No sentido de viabilizar o armazenamento de dados de forma mais eficaz, as palavras salvas localmente no banco de dados da aplicação são enviados via *e-mail*, para que posteriormente se possa realizar análises estatísticas com tais dados.

1.2.1 Objetivos específicos

No que tange aos objetivos específicos desse trabalho, pode-se enumerar:

1. Idealização de um manual de desenvolvimento utilizando a ferramenta *Unity*, de forma a contemplar um passo a passo desde sua concepção até a geração do aplicativo final.

2. Propiciar um *framework* para auxiliar na implementação de outros módulos do sistema.

3. Desenvolver uma primeira versão do aplicativo *Grapphia*, contendo o módulo que trabalha palavras com as letras s e z concorrentes, incluindo livro com história que contextualize as palavras e um jogo para auxiliar o ensino/aprendizagem da ortografia deste conjunto de palavras.
4. Implementar um módulo que simula um ditado, permitindo avaliar o desenvolvimento do usuário após a fase do jogo.
5. Implementar outros ajustes no sistema que foram identificados após a finalização do Laça-Palavras.

1.3 Justificativa e Metas

O interesse principal desse trabalho concentra-se no desenvolvimento de jogos eletrônicos educacionais utilizando a ferramenta *Unity*. O uso de jogos na educação tem sido encorajado e justificado por diversos autores, como Dohme (2003) que afirma que, além de serem fontes de diversão, os jogos podem ser utilizados para vários fins educativos e como instrumentos de desenvolvimento de crianças e jovens. Segue explicando que, para as crianças, o jogo constitui um fim, onde o objetivo principal é a diversão.

1.4 Metodologia

A metodologia empregada no desenvolvimento desse trabalho se segmentou nas seguintes etapas:

- Pesquisas bibliográficas em revistas, artigos científicos, trabalhos de conclusão de curso e livros relacionados ao tema.
- Mapeamento das irregularidades ortográficas por representações múltiplas.
- Desenvolvimento de jogos tutoriais presentes no *website* da *Unity Technologies* para assimilar o funcionamento da ferramenta *Unity.*
- Definição de ferramentas de controle de versão que propiciem o desenvolvimento compartilhado do sistema por intermédio de um repositório.
- Definição de ferramentas para controle da qualidade do *software* desenvolvido.
- Definição e adoção de um processo de desenvolvimento de jogos eletrônicos para implementação do sistema proposto.
- Explicação das ferramentas e componentes que do ambiente de desenvolvimento de jogos *Unity*.
- Desenvolvimento de um novo módulo do sistema que aborde as irregularidades ortográficas por representações múltiplas do grupo “s” e “z”, utilizando sua primeira versão como base.

1.5 Estrutura do Trabalho

O presente trabalho está estruturado da seguinte forma: o capítulo dois abordará um processo para desenvolvimento de jogos, o capítulo três ficará encarregado de fornecer uma explicação sobre o funcionamento e os componentes do *Unity*, o capítulo quatro descreverá o desenvolvimento do jogo educacional proposto, o capítulo cinco apresentará os trabalhos futuros e conclusões.

2 PROCESSO DE DESENVOLVIMENTO DE JOGOS

Nesse capítulo será apresentada uma metodologia para desenvolvimento de jogos, objetiva-se apresentar práticas gerais que são difundidas na produção de *games* de forma a torná-los bem estruturados e mais competitivos. As fases que constituem o processo de desenvolvimento de jogos estão divididas entre Pré-Produção, Produção e Pós-Produção (Figura 1).

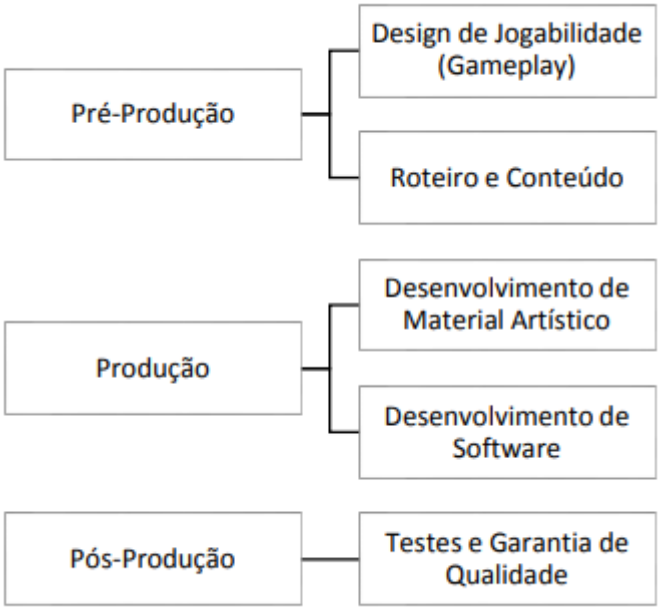

Figura 1 – Processo de desenvolvimento de jogos (MENDES, 2006).

O processo de desenvolvimento de jogos apresentado na Figura 1, inicia-se com o estabelecimento da concepção inicial. Posteriormente planeja-se o processo de pré-produção, que apresenta como artefatos gerados o *design* de jogabilidade e um roteiro/conteúdo utilizado como base a fim de possibilitar o desenvolvimento das demais etapas do processo. O processo de produção é responsável por implementar efetivamente a ideia proposta, a partir dos passos tomados nas etapas anteriores. Nesta etapa, desenvolve-se o material artístico a ser utilizado na aplicação (como *sprites*, cenários e elementos gráficos em geral) além da codificação/desenvolvimento do *software*. Por fim, a etapa de pós-produção objetiva o desenvolvimento de testes da aplicação e com base nas falhas e problemas identificados, propõe-se atualizações e melhorias de suas funcionalidades.

2.1 Concepção Inicial

Antes de introduzir o fluxo de atividades do processo de desenvolvimento de jogos descrito na Figura 1, deve-se estabelecer uma concepção inicial. JUNIOR, NASSU e JONACK

(2002) conceituam essa etapa como a ideia básica por trás do *game*, a sua premissa inicial, que pode levar ao seu real desenvolvimento. Dessa forma, esta etapa deve ser concluída antes de iniciar o planejamento do projeto.

2.2 Processo de pré-produção

Para Ferreira (2015), a fase do planejamento (ou pré-produção) do desenvolvimento consiste em ter uma proposta como um guia da arte e o plano de produção. Diante do explicitado na Figura 1 os artefatos gerados por essa fase são: *design* de jogabilidade (*gameplay*) e roteiro/conteúdo.

> Eu costumava pensar que finalizar um projeto era um processo mais difícil, mas mudei de idéia. Depois de terminar alguns títulos, você já sabe o que é necessário para finalizar um game. Agora acho que a parte mais difícil do projeto de desenvolvimento é a pré-produção. Gerar novas ideias realmente boas além de quais, utilizadas conjuntamente, poderão gerar um sucesso. Se os objetos da pré-produção não forem estabelecidos com precisão, mais adiante você acabará tendo de trabalhar muito para tornar o *game* divertido e bem-sucedido (NOVAK, 2010, p. 324).

Uma informação importante levantada no processo de pré-produção diz respeito a classificação dos jogos. O levantamento dessa informação é crucial para a definição do escopo de atuação do jogo produzido. Por não ser uma tarefa trivial, a subseção a seguir apresenta as categorias usuais que os jogos podem se enquadrar.

2.3 Classificação dos jogos

Os jogos possuem uma classificação definida por categorias de forma possibilitar o agrupamento deles por características semelhantes. De acordo com Lopes (2010) os critérios de classificação dão-se por dimensionalidade, por ponto de vista, por gênero e por número de jogadores.

- Classificação por dimensionalidade

  Leva em consideração o número de dimensões constituintes no jogo, dividindo-se em jogos 2D (que apresentam duas dimensões) e jogos 3D (representação por três dimensões).

Figura 2 – Exemplo de jogo 2D - A Lenda do Herói (Castro Brothers, 2016).

Figura 3 – Exemplo de jogo 3D - Tekken 5 (Namco, 2016).

- Classificação por ponto de vista

  A classificação por ponto de vista considera a perspectiva através da qual a apresentação gráfica do jogo é vista pelo jogador, com relação ao personagem por ele controlado (ROSA, 2014). Diante disso, existem dois tipos de perspectivas a em primeira pessoa, onde o personagem é controlado pelo jogador, e a em terceira pessoa, onde o jogador observa o jogo por um ponto de vista distinto de seu personagem.

Figura 4 – Exemplo de jogo em primeira pessoa - Counter-Strike 1.6 (Valve Corporation, 2000).

Figura 5 – Exemplo de jogo em terceira pessoa - Grand Theft Auto III (Rockstar North, War Drum Studios, Rockstar Vienna, 2001).

- Classificação por gênero

  A classificação do gênero concorda com o objetivo principal do jogo a ser criado. Ainda que com o passar dos anos, os jogos se fragmentaram em muitos e variados gêneros e subgêneros (ROGERS, 2012). Nesta classificação agrupam-se jogos com características e jogabilidades semelhantes. Existe uma grande quantidade de gêneros de jogos, dentre os principais pode-se elencar:

  1. Gênero Ação;
  2. Gênero Aventura;
  3. Gênero Estratégia;
  4. Gênero Esporte;
  5. Gênero Simulação;
  6. Gênero Tabuleiro e Quebra-Cabeças;

7. Gênero *Role-Playing Game* (RPG).

2.4 Desenvolvimento do protótipo

O protótipo pode ser um item de *software* operacional que capture na tela a essência do que torna o jogo especial, ou um exemplar de baixa fidelidade, de papel, mas que sirva para testá-lo, a fim de garantir a qualidade da mecânica do jogo e que seja divertido e atraente (NOVAK, 2010).

Vale ressaltar que nesta etapa deve-se considerar a criação de uma *interface*[1] que permita ao usuário obter uma visão panorâmica do conteúdo, navegar na massa de dados sem perder a orientação e, por fim, mover-se no espaço informacional de acordo com seus interesses (BONSIEPE; DUTRA, 1997). Novak (2010) destaca que entre as diretrizes para a produção de uma boa *interface* de jogos estão: consistência; teclas de atalhos; fornecer *feedback*; tarefas definidas; fácil cancelamento de ações; fornecer meios de controle ao jogador; simplicidade; recursos de personalização (de aspectos da *interface*); inclusão de um ponteiro sensível ao contexto (que mude a forma ao apontar para um objeto de interesse); implementar diferentes modos e utilizar convenções estabelecidas.

2.5 Produção

Corresponde a fase mais extensa do processo, quando efetivamente inicia-se o desenvolvimento do jogo. Para que não hajam imprevistos ao longo da produção, é necessário que as etapas anteriores tenham sido bem fundamentadas e desenvolvidas.

2.6 Fases de produção: Alfa, Beta, Ouro

2.6.1 Fase Alfa

Essa fase é encarregada de realizar ajustes e acabamentos finais do jogo. Talvez haja algumas lacunas e elementos artísticos não definitivos, porém o motor e a *interface* estejam completos. Testadores de jogabilidade são incorporados à equipe para localizar problemas (NOVAK, 2010).

2.6.2 Fase Beta

Nesta fase todos os materiais já estão incorporados ao jogo e o processo de produção terminou. A fase beta tem como objetivo estabilizar o projeto e eliminar o maior número possível de defeitos antes que o produto comece a ser vendido (NOVAK, 2010). Segundo Ferreira (2015) os fabricantes de jogos em sua maioria utilizam seus próprios testadores ou recrutam jogadores *online* para serem testadores betas dos jogos, atribuindo-os a responsabilidade encontrar o

[1] Elemento que proporciona uma ligação física ou lógica entre dois sistemas ou partes de um sistema que não poderiam ser conectados diretamente.

máximo de defeitos, erros e ajustes de desempenho antes que sejam efetivamente publicados e comercializados.

2.6.3 Fase Ouro

Concluindo a fase beta, inicia-se a fase ouro. Nesta fase o produto e as correções de erros foram avaliadas e após administração e averiguadores de defeitos concordarem que o jogo está pronto, ele vai para a fabricação, onde a mídia é gerada e embalada. Ao final desta fase, o jogo é comercializado. No caso de jogos distribuídos digitalmente, a fase de fabricação é eliminada (NOVAK, 2010).

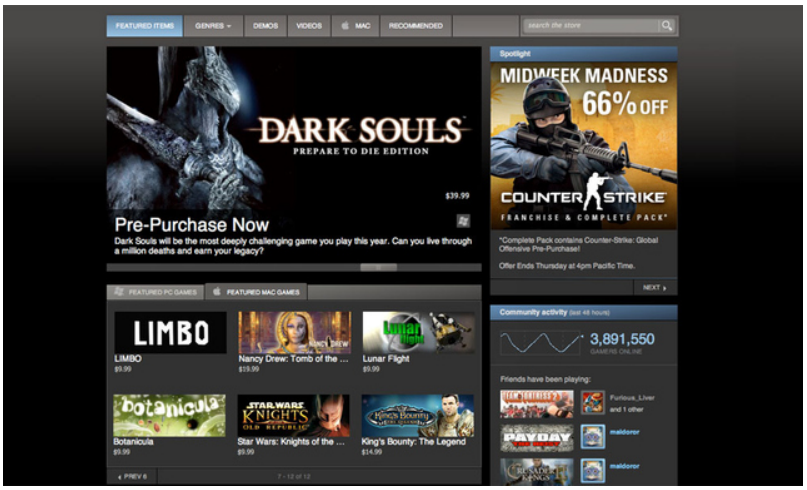

Figura 6 – Distribuição digital de jogos pela Steam (STEAM, 2018).

2.7 Pós-Produção

Na etapa de pós-produção, várias versões subsequentes podem ser lançadas para substituir e melhorar o jogo original, aumentando sua longevidade. Correções também podem ser aplicadas para solucionar falhas percebidas tardiamente ou mesmo acrescentar novas funcionalidades (NOVAK, 2010).

3 TECNOLOGIAS UTILIZADAS

Esta seção visa explanar sobre todas as tecnologias que foram utilizadas do desenvolvimento desde trabalho. Inicialmente será apresentado o motor de jogo utilizado para a construção da aplicação e aspectos de *interface* gráfica. Esta seção apresenta explicação minuciosa de todos os componentes e ferramentas que compõe o *Unity* e as configurações necessárias para a construção de versões executáveis da aplicação, priorizando a sua construção para a plataforma Android.

3.1 Unity

O *Unity* é um motor de jogo ou *game engine*. De acordo com Torres (2016) o motor de jogo corresponde a uma categoria de programa de computador que conta com um conjunto de bibliotecas para o desenvolvimento de jogos eletrônicos e aplicações gráficas. As funcionalidades fornecidas por um motor de jogo incluem: um motor gráfico para renderizar gráficos; um motor de física para promover a detecção de colisões; suporte para criação de animações, efeitos sonoros e inteligência artificial; além de suporte à linguagens de *script*, como *JavaScript*, *C Sharp (C#)* ou Boo (dialeto em *Phyton*) (TORRES, 2016).

> "Para estúdios e desenvolvedores independentes, o democrático ecossistema da Unity derruba barreiras de tempo e custo para criar jogos exclusivamente lindos. Eles estão usando o Unity para viver do que adoram fazer - desenvolver jogos que cativam e encantam os jogadores em qualquer plataforma (UNITY, 2018)."

3.2 *Interface*

O motor de jogos *Unity* trabalha com uma *interface* simples a fim de facilitar o desenvolvimento para utilizadores iniciantes. Sua área de trabalho é composta por janelas chamadas *views*, cada uma com um propósito específico (PASSOS *et al.*, 2009). Um esquema que identifica cada um dessas janelas no editor de cenas da *Unity* é apresentado na Figura 7.

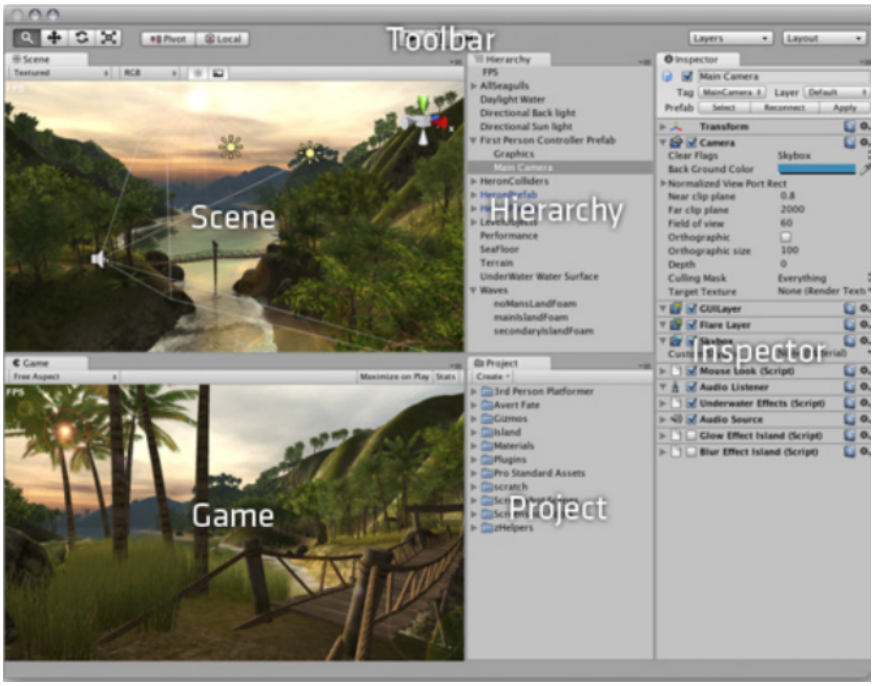

Figura 7 – Interface do editor de cenas do *Unity* (UNITY, 2018).

3.2.1 *Project*

A Figura 8 apresenta a janela corresponde ao *Project*, nele são organizados e manipulados os arquivos (*Assets*) que compõem um projeto. Contendo *scripts*, *materials*, modelos, efeitos sonoros, animações, cenas, *prefabs* ou pré-fabricados, pacote de fontes, organização de *sprites*, dentre outros. A estrutura exibida na janela *Project* é correspondente à sub-pasta *Assets* dentro da pasta do projeto no sistema de arquivos do computador (PASSOS *et al.*, 2009).

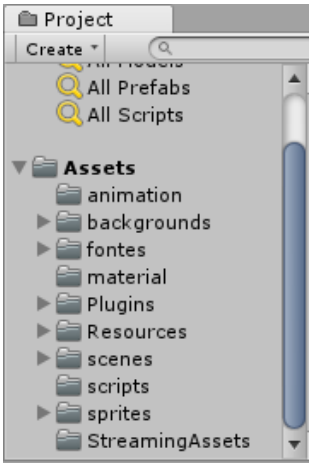

Figura 8 – *View Project.*

Fonte: Elaborada pelo autor

3.2.2 *Hierarchy*

Nesta janela são exibidos todos os objetos presentes na cena que está sendo editada. Em Programação Orientada à Objetos (POO) um objeto corresponde a uma instância (ou seja, um exemplar) de uma classe. Analogamente podemos entender uma classe como uma fábrica de carros e o objeto como um modelo dessa fábrica. No caso do motor de jogo *Unity* um objeto é representado por um *GameObject* (a definição dos *GameObjects* será apresentada com mais detalhes na seção 3.4). Além disso, na *Hierarchy* podemos organizar e visualizar a hierarquia de composição entre os vários objetos que constituem a cena (grafo de cena) (PASSOS *et al.*, 2009). Para incluir um novo objeto na cena deve-se utilizar o menu *Create*, ele permite criar vários tipos de objetos (Figura 9). Uma segunda forma de realizar o mesmo procedimento consiste em arrastar os elementos para a cena a partir do *Project*. A Figura 10 apresenta a janela *Hierarchy* contendo os objetos utilizados na cena atual da aplicação.

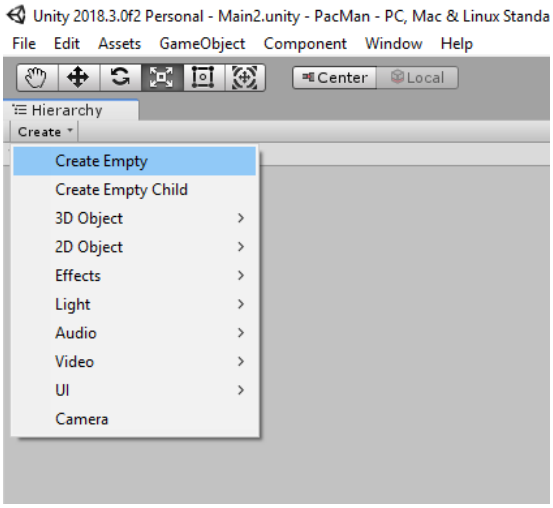

Figura 9 – *Opção Create da janela Hierarchy.*

Fonte: Elaborada pelo autor

Figura 10 – *View Hierarchy.*

Fonte: Elaborada pelo autor

3.2.3 *Inspector*

Na janela *Inspector*, tem-se acesso aos vários parâmetros de um objeto presente na cena, bem como os atributos que constituem seus componentes (PASSOS *et al.*, 2009). A Figura 11, exemplifica a janela *Inspector*. Note que o *GameObject* selecionado corresponde a *Main Camera* e este possui alguns componentes como: *Camera*, *Guid Layer*, *Flare Layer*, *Audio Listener* e cada componente desempenha uma função dentro do *GameObject*, como aspectos sonoros, organização de *layers* e definição da perspectiva da câmera que constitui a cena.

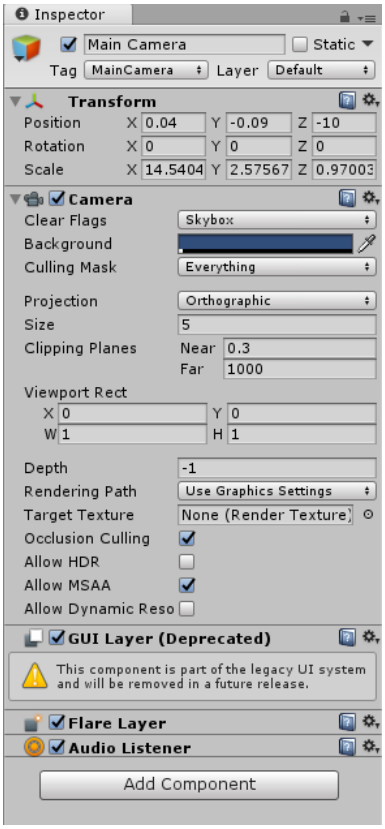

Figura 11 – *View Inspector.*

Fonte: Elaborada pelo autor

Também é possível incluir novos componentes ao *GameObject* (Figura 12) de acordo com uma série de componentes pré-definidos do *Unity*.

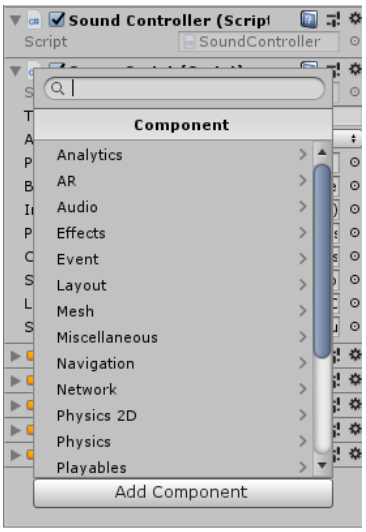

Figura 12 – *Adição de componentes na view inspector.*

Fonte: Elaborada pelo autor

3.2.4 *Scene*

A *Scene* é responsável pela manipulação dos elementos visuais no editor de cenas da *Unity*, possibilitando a orientação e posicionamento desses elementos com um retorno imediato do efeito proporcionado pelas alterações realizadas. É possível manipular graficamente cada um dos objetos através das opções de arrastar e soltar do *mouse* (PASSOS *et al.*, 2009). Armazena todos os *GameObjects* e demais itens presentes no editor. Na Figura 13 apresenta-se a tela inicial do aplicativo Grapphia.

Figura 13 – *View Scene.*

Fonte: Elaborada pelo autor

É possibilitado ao desenvolvedor, ferramentas básicas de manipulação dos elementos da cena, tais como: translação, rotação e escala (Figura 14).

Figura 14 – *Scene Options.*

Fonte: Elaborada pelo autor

3.2.5 *Game*

A janela *Game* é responsável pela visualização da aplicação em desenvolvimento da forma que ela será exibida quando finalizada. Pode-se rapidamente ter uma prévia de como

os elementos estão se comportando dentro da aplicação (PASSOS *et al.*, 2009). Dessa forma, o desenvolvedor pode avaliar e alterar as propriedades dos *GameObjects* em tempo de execução. A Figura 15 representa graficamente a janela *Game*, é possibilitado ao desenvolvedor opções para manipular a cena em tempo de execução, como *Maximize On Play*, *Mute Audio*, dentre outros.

Figura 15 – *View Game*

Fonte: Elaborada pelo autor

3.3 Componentes

De acordo com Passos *et al.* (2009) os componentes são responsáveis por implementar os diversos comportamentos que um *GameObject* pode ter, podendo variar desde um *script* a um geometria de colisão pautada em aspectos físicos e matemáticos.

> "Ao vincular componentes a um GameObject, você cria comportamentos, movimentos e define aparência. Luzes, malhas, efeitos especiais, áudios, câmeras e emissores de partículas são exemplos de componentes. Por sua vez, cada componente em um conjunto próprio de propriedades ajustáveis, alcance e intensidade para luz (UNITY, 2018)."

3.4 *GameObjects*

A *Unity* é um motor de jogo fundamentado em um modelo mais moderno para arquitetura de objetos, baseado em composição Passos *et al.* (2009 apud STOY, 2006, p.8). Nesse modelo, um objeto de jogo é especificado através da composição de vários componentes físicos, sejam eles *scripts*, componentes de efeito, *layout*,dentre outros. E estes são removidos ou agregados em cada objeto da cena pelo desenvolvedor da aplicação (vide a Figura 12). Cada atributo e/ou método implementado via *script* feito por intermédio de um componente (herdando de um componente básico) (PASSOS *et al.*, 2009). O componente básico é chamado *GameObject*, "*Game Object* é como uma panela vazia e os componentes são os ingredientes que irão criar sua receita de jogabilidade" (UNITY, 2018, p.1). Vale ressaltar a existência de *Game Objects* padrões, como a câmera virtual (Figura 16).

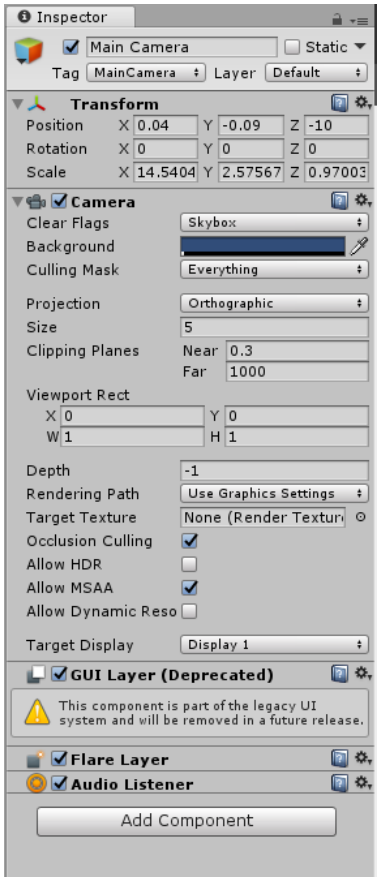

Figura 16 – *GameObject câmera virtual.*

Fonte: Elaborada pelo autor

3.4.1 Pré-Fabricados (*Prefabs)*

De forma geral os *prefabs* ou pré-fabricados representam um conjunto de objetos, que são modelados e salvos com o intuito de serem reutilizados. Segundo Passos *et al.* (2009) um *prefab* é simplesmente um modelo de composição de *GameObject* já definido, ou mais

precisamente, um *template* que define um elemento através da composição dos vários componentes. Um exemplo de *prefab* (Figura 17) pode ser a definição de um humanoide, que necessita basicamente em sua composição, de um *script* de movimentação e um componente de colisão. Observe que os componentes desse *GameObject* são previamente definidos e que ele está pronto para ser utilizado da forma que for adequada ao desenvolvedor, isso propicia um processo de desenvolvimento mais ágil e organizado.

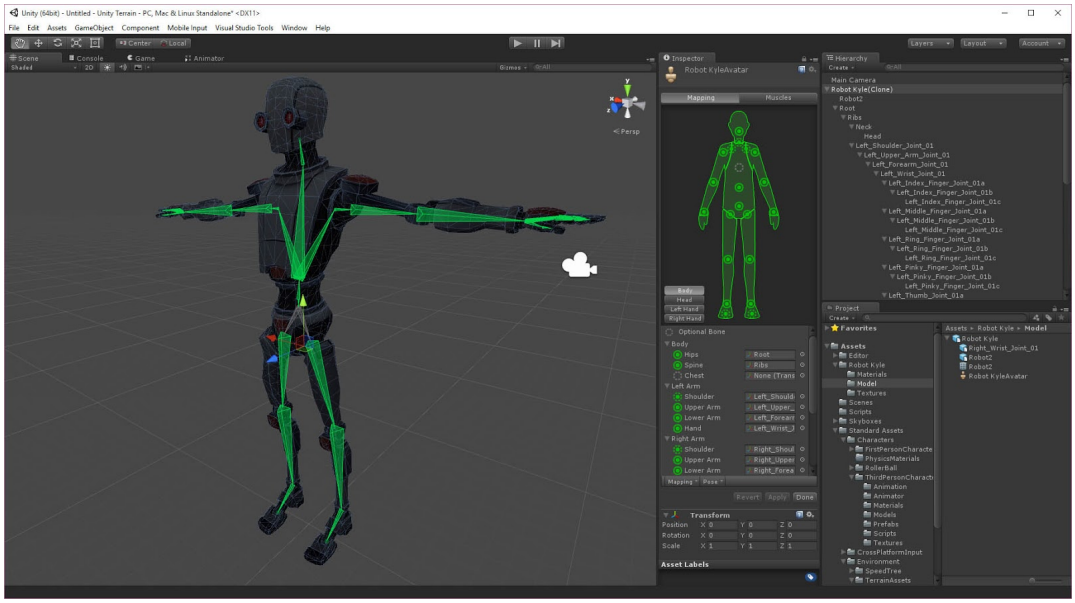

Figura 17 – *Exemplo de um prefab em dimensão 3D representando um humanoide.*

Fonte:
⟨http://ofcourseillplayit.blogspot.com.br/2015/08/unity3d-at-home-project-day-4-basic.html⟩

3.4.2 Materiais, Texturas e *Shaders*

> "Texturas são imagens; elas podem ser fotos ou imagens pintadas. Texturas são elementos 2D que definem determinado atributo de um material . Texturas podem ser usadas para definir os elementos táteis visual ou coloridos de um material, como relevo por exemplo. Quando um material é aplicado a um objeto, o *shader* é usado para mostrar como esse material vai reagir a luz e ao ângulo de visão do observador Ferreira (2015 apud WATKINS, 2012, p. 63)."

Torres (2016) explica que modelos gráficos são criados utilizando *materials*, *textures* e *shaders*. Os materiais são baseados em texturas e *shaders*. O *texture* determina o que será detalhado na superfície do material, enquanto os shaders definem como será desenhado. Neste trabalho os cenários foram concebidos a partir de materiais definidos através de imagens específicas. Perceba na Figura 18 que temos a imagem do cenário correspondente a cena de "Seleção de personagem", já na Figura 19 temos uma lista de *materials* que foram criados para popular as cenas que constituem a aplicação proposta.

Figura 18 – *Imagem da tela "Seleção de Personagem".*

Fonte: Elaborada pelo autor.

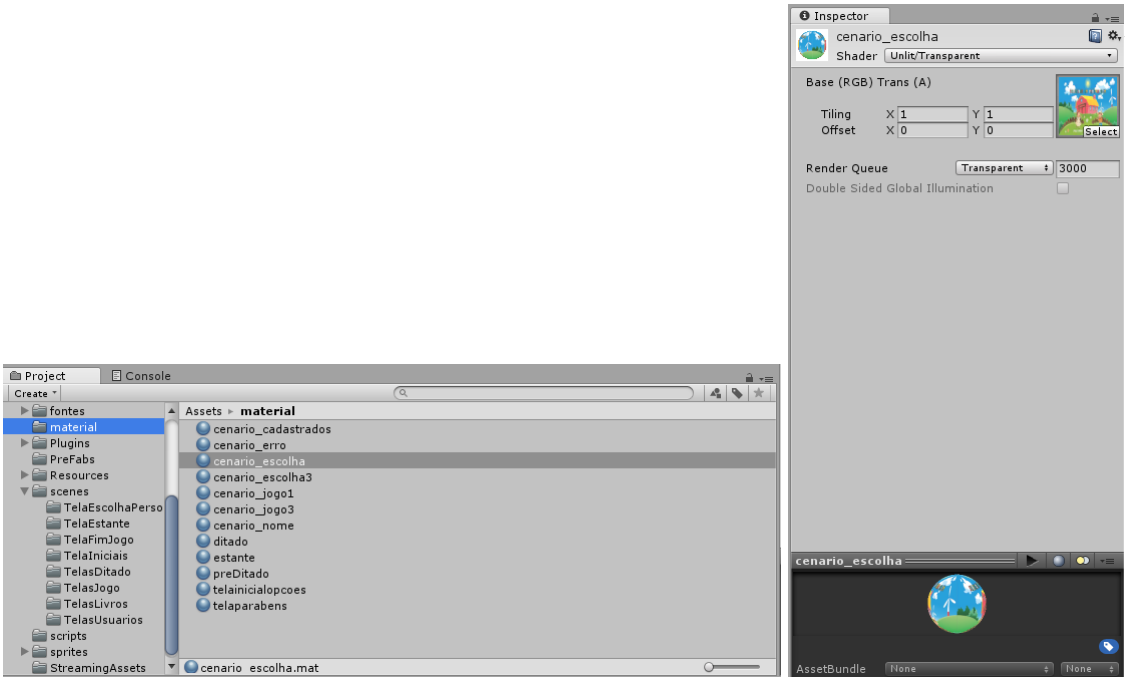

Figura 19 – *Material que contém o cenário de escolha de personagem.*

Fonte: Elaborada pelo autor.

3.4.3 *Plug-ins*

Entende-se como *plug-in* uma extensão capaz de fornecer uma funcionalidade a outros programas maiores, com uma finalidade específica. A *Unity* possui uma lista de *plug-ins*

gratuitos e/ou pagos em sua *Assets Store*[2]. Esse banco de ativos possui materiais prontos para jogos 2D ou 3D, como: modelos, códigos, *interfaces*, texturas, ou mesmo projetos já finalizados.

Para dinamizar o desenvolvimento deste trabalho foram utilizados alguns ativos ofertados na *Assets Store* de forma gratuita, sendo eles:

- *Book-and-Curl* [3]: Para possibilitar o efeito de transição de páginas dos livros que compõem a aplicação;
- GAF [4]: Para converter arquivos de formato *.swf* para *prefabs* a fim de compor a tela de jogo do mundo "Sol de Verão".

Perceba na Figura 20 que temos os *plug-ins* utilizados na aplicação. A utilização do *plug-in Book-and-Curl* propiciou o desenvolvimento do livro e a possibilidade de movimentação de suas páginas de forma livre por parte do utilizador da aplicação.

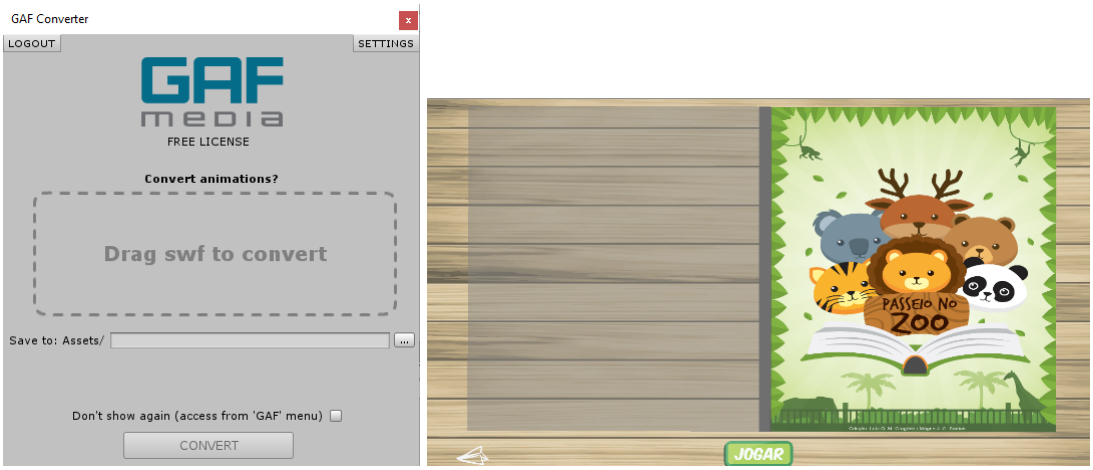

Figura 20 – *Plug-ins* utilizados no trabalho

Fonte: Elaborada pelo autor.

3.4.4 Animações

> "Pense nas animações como componentes de *GameObjects*. Sempre que um objeto deve ser animado no Unity, ele criará um arquivo separado editável (com um rótulo .anim) que armazenará as informações. O que isso significa é que a animação pode ser usada repetidas vezes em outros objetos (o que é um poderoso recurso por si só). Isso também significa que há um novo recurso armazenado na pasta de recursos Ferreira (2015 apud WATKINS, 2012, p.320)."

As animações são concebidas dentro do próprio *Unity*, apesar de existirem diversas ferramentas para animação no mercado. Para criar uma animação recomenda-se que inicialmente

[2] Corresponde a uma loja *online* do *Unity* que possui todo tipo de material útil para desenvolvimento de aplicações utilizando este motor de jogo. Desde cenários prontos, a imagens, *materials*, animações, dentre outros

[3] ⟨https://assetstore.unity.com/packages/tools/animation/book-page-curl-55588⟩

[4] ⟨https://assetstore.unity.com/packages/tools/animation/gaf-free-flash-to-unity-13593⟩

atribua um componente *Animator Controller* ao objeto que deseja animar e posteriormente crie a animação através de um conjunto de imagens, denominado *sprites*. A Figura 21 apresenta um exemplo de *sprite*, constate que cada imagem dentro de um *sprite* corresponde a um movimento que se pretende representar na animação.

Figura 21 – *Exemplo de uma sprite utilizada no trabalho.*

Fonte: Elaborada pelo autor.

3.4.5 Linguagens de desenvolvimento / *Scripts*

> "O Unity usa três linguagens de script: JavaScript(UnityScript), C# e Boo. Essas três linguagens de script são maneiras muito diversas de acessas a mesma funcionalidade básica do mecanismo de jogo; três ferramentas com o mesmo objetivo Ferreira (2015 apud WATKINS, 2012, p. 334)."

A *Unity* disponibiliza um sistema de *scripting* variado e flexível, possibilitando ao desenvolvedor utilizar três linguagens (C#, *Javascript* e Boo) mesmo que combinadas. Sendo as linguagens C# e *JavaScript* as mais utilizadas pelos desenvolvedores (WIKIDOT, 2018). Recomenda-se a linguagem C# para usuários mais experientes por possuir um acesso maior aos objetos e componentes e *JavaScript* aos iniciantes.

Os *Scripts*, são conceituados por diversos segmentos na computação. Em uma abordagem voltada para administração de redes COSTA (2007) define os *scripts* como sendo programas de computador com inúmeras finalidades, porém com características peculiares. Habitualmente os *scripts* são comandos (instruções), organizados em blocos denominados métodos/funções. Veja o *script* abaixo:

```
1 using System.Collections;
  using System.Collections.Generic;
3 using UnityEngine;
  using UnityEngine.SceneManagement;
5
  public class ditadoOrientacoes : MonoBehaviour {
```

```
7   public AudioClip sound;

9   // Use this for initialization
    void Start () {
11    StartCoroutine ("audioEnd");
    }
13
    IEnumerator audioEnd(){
15    yield return new WaitForSeconds (sound.length);
      SceneManager.LoadScene ("telaDitado");
17  }
}
```

Listing 3.1 – *Exemplo de codificação utilizando C# presente neste trabalho.*, label=Script

Este *script* possui um conjunto de bibliotecas que definem subprogramas a serem utilizados em situações específicas e dois blocos de código, que dizem respeitos as ações que devem ser executadas quando o *script* for inicializado e quando o áudio que estiver em execução for finalizado. Perceba que basicamente foram utilizadas bibliotecas do sistema e do motor de jogo, a explicação prática da definição de um *script* será apresentada no capítulo 4.

3.4.6 *User Interface (UI)*

O *User Interface* – ou interface do usuário – é tudo aquilo que é perceptível visualmente em alguma plataforma e leva o usuário a uma interação com um sistema. Pode ser um botão, um menu diferente ou até mesmo um som (RAFFCOM, 2018).

3.5 Configurações de construção de versões executáveis

"[...] Uma das vantagens da *Unity* é sua portabilidade, podendo desenvolver jogos para PC, Mac, Linux, Android, IOS, Web browsers e consoles [...]" (TORRES, 2016, p. 67).

3.5.1 Android

Para gerar a versão executável de sua aplicação para a plataforma *Android* é necessário que se obtenha os pacotes *Java SE Development Kit* (JDK)[6] e *Android SDK*[4]. JDK é um um ambiente de desenvolvimento utilizado para criar aplicativos e componentes usando a linguagem de programação *Java*, tal ambiente é composto basicamente por compiladores e bibliotecas (ORACLE, 2018). O *Android SDK* possibilita aos desenvolvedores conceberem aplicativos para a plataforma *Android* de forma nativa (as aplicações são projetadas para atuar

[6] O pacote Java SE Development Kit necessário para a produção de versões executáveis (.APK) deste trabalho, pode ser obtido através do link: ⟨http://www.oracle.com/technetwork/pt/java/javase/downloads/index.html⟩.

[4] O Android SDK necessário para a produção de versões executáveis (.APK) deste trabalho, pode ser obtido com todas as ferramentas do Android Studio: ⟨https://developer.android.com/studio/index.html⟩.

apenas em *Android*), o SDK inclui projetos de exemplo com código-fonte, ferramentas de desenvolvimento, bibliotecas, dentre outros.

Após instalar as ferramentas deve-se indicar no *Unity* o local que elas estão alocadas, para isso acesse *Edit* > *Preferences* > *External Tools* e escolha a opção *Browse* para informar o caminho de instalação. A Figura 24 apresenta as preferências que devem ser aplicadas para que seja possível construir uma versão executável da aplicação. Observe que devem ser incluídos nos campos de *SDK e JDK* o caminho para acessar os arquivos que contém estes pacotes.

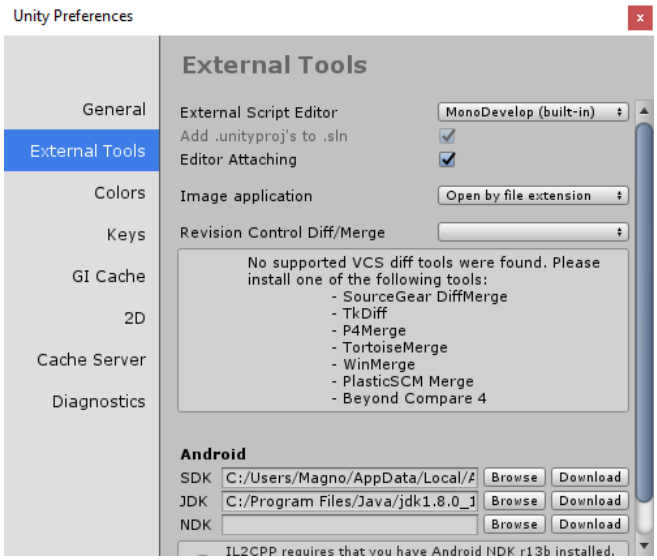

Figura 22 – *Definindo as preferências para construção da versão executável.*

Fonte: Elaborada pelo autor.

Posteriormente acesse as configurações de construção (*Build Settings*). Nesta tela deve selecionar as cenas que deseja incluir no *build*, a plataforma que a aplicação será construída e as configurações de jogador (*Player Settings*).Note na Figura 23 que é possível que o desenvolvedor escolha entre várias plataformas para construção de versões executáveis: *Android*, *IoS*, *Xbox One*, dentre outros.

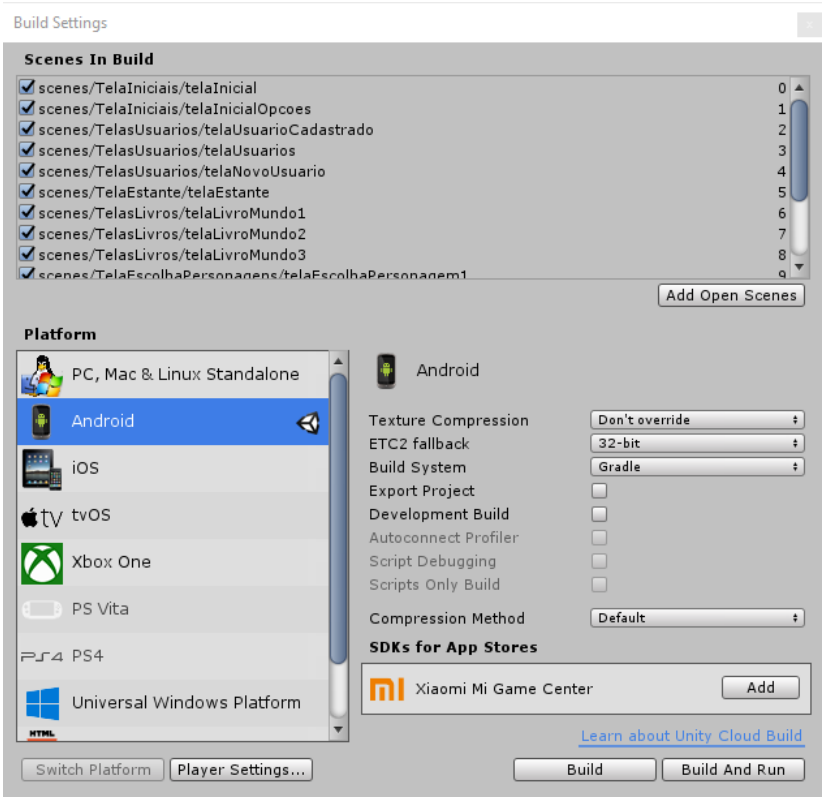

Figura 23 – *Tela de configurações de construção / Build Settings.*

Fonte: Elaborada pelo autor.

Nas configurações de jogador (*Player Settings*), deve-se definir o nome da aplicação além de seu ícone. Configure também a resolução e apresentação da aplicação, aspectos suportados e versão mínima do *Android*, imagem inicial de apresentação (Figura 24).

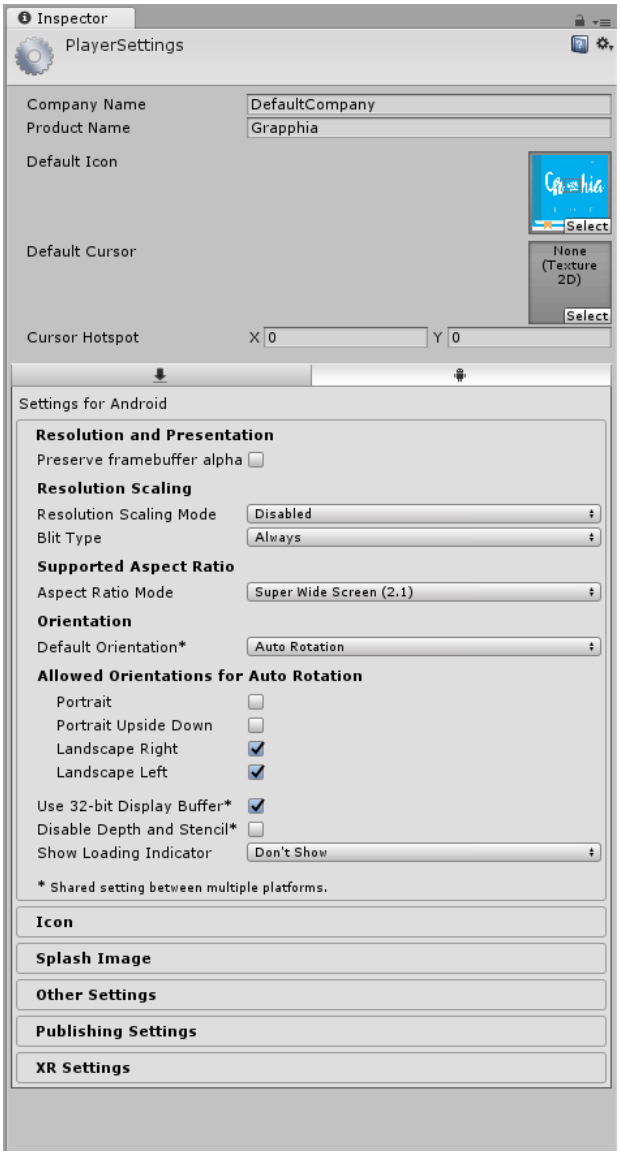

Figura 24 – *Tela de configurações de jogador / Player Settings.*

Fonte: Elaborada pelo autor.

Por fim, o projeto está pronto para ser "*buildado*", basta escolher dentre as opções de "*Build*" ou "*Build And Run*", sendo a primeira responsável por criar a versão executável no seu computador através do caminho transmitido e a segunda encarregada de criar a versão executável diretamente no dispositivo móvel com o auxílio de um cabo *Universal Serial Bus* (USB).

4 GRAPPHIA: APLICATIVO PARA DISPOSITIVOS MÓVEIS PARA AUXILIAR NO ENSINO DA ORTOGRAFIA

4.1 Metodologia de desenvolvimento

O aplicativo *Grapphia* foi proposto para auxiliar o ensino da grafia de palavras que possuem letras ou dígrafos concorrentes, que representam o mesmo som. O público-alvo a que ele se destina abrange crianças entre 8 e 10 anos e a tarefa a ser executada pelo utilizador do sistema é de completar palavras apresentadas na tela selecionando uma letra entre duas opções de respostas. A primeira versão deste aplicativo foi "Laça Palavras", proposto em 2014. A figura a seguir apresenta a primeira versão criada.

Figura 25 – *Telas da primeira versão do aplicativo.*

Fonte: Elaborada pelo autor.

A metodologia de desenvolvimento do aplicativo proposto baseou-se no processo de desenvolvimento de jogos descrito no capítulo 2, sendo organizado nas seguintes etapas:

1. Mapeamento das irregularidades ortográficas;

2. Desenvolvimento do aplicativo.

4.1.1 Mapeamento das irregularidades ortográficas

O domínio das habilidades relacionadas à aprendizagem da escrita da Língua Portuguesa (LP) compreende um dos objetivos de ensino na edução básica. Um dos aspectos relacionados a essa habilidade engloba o domínio da norma ortográfica, o que nos leva à identificação da importância da ortografia para o ensino da LP escrita.

A análise da produção de escrita das crianças que participaram do estudo de Zorzi (1998) permitiu a classificação dos erros ortográficos: representações múltiplas; apoio na oralidade; omissão de letras; junção-separação de palavras; confusão entre “am” e “ão”; generalização de regras; trocas surdas-sonoras; acréscimo de letras; letras parecidas e inversão de letras (ZORZI, 2009, p. 37). Constatou-se que determinados erros possuem maior incidência. Nesse caso, havia um índice alto de alunos produzindo erros por representações múltiplas, pois:

> “[...] a possibilidade de se utilizar uma mesma letra para escrever diferentes sons [...] apresenta-se como uma realidade de mais difícil compreensão por não haver uma forma estável de escrita. Temos nesse caso, um fato de natureza linguística importante que somente ao longo do tempo vai sendo compreendido e dominado [...] (ZORZI, 2009, p. 40-41).”

Partindo desse pressuposto, a figura 26 apresenta a marca *Grapphia* que foi registrada em 2018.

Figura 26 – *Aplicativo para auxiliar no processo de ensino/aprendizagem da ortografia.*

Fonte: Elaborada pelo autor.

4.1.2 Desenvolvimento do aplicativo

Após definir a dificuldade ortográfica a ser tratada, a metodologia adotada foi elaborar um levantamento de jogos digitais com temas ortográficos disponíveis, esse levantamento auxiliará no formato do aplicativo proposto neste trabalho.

Pretende-se que o aplicativo *Grapphia* seja composto por vários módulos, cada um contendo uma dificuldade ortográfica por representação múltipla específica. Esse trabalho concentra-se no desenvolvimento do módulo “A Fazenda”, tal módulo apresenta um conjunto de palavras grafadas com “s” ou “z” concorrentes. A figura 27 apresenta um diagrama de atividades do sistema, nela é possível acompanhar o fluxo de atividades a ser realizado a partir da interação do usuário com o sistema. A partir da tela inicial, apresentada na figura 28, o usuário passa pela fase de identificação no sistema, em que existem as opções de cadastrar um novo usuário ou selecionar um usuário já existente. A figura 29 retrata as telas de cadastro e seleção de usuário.

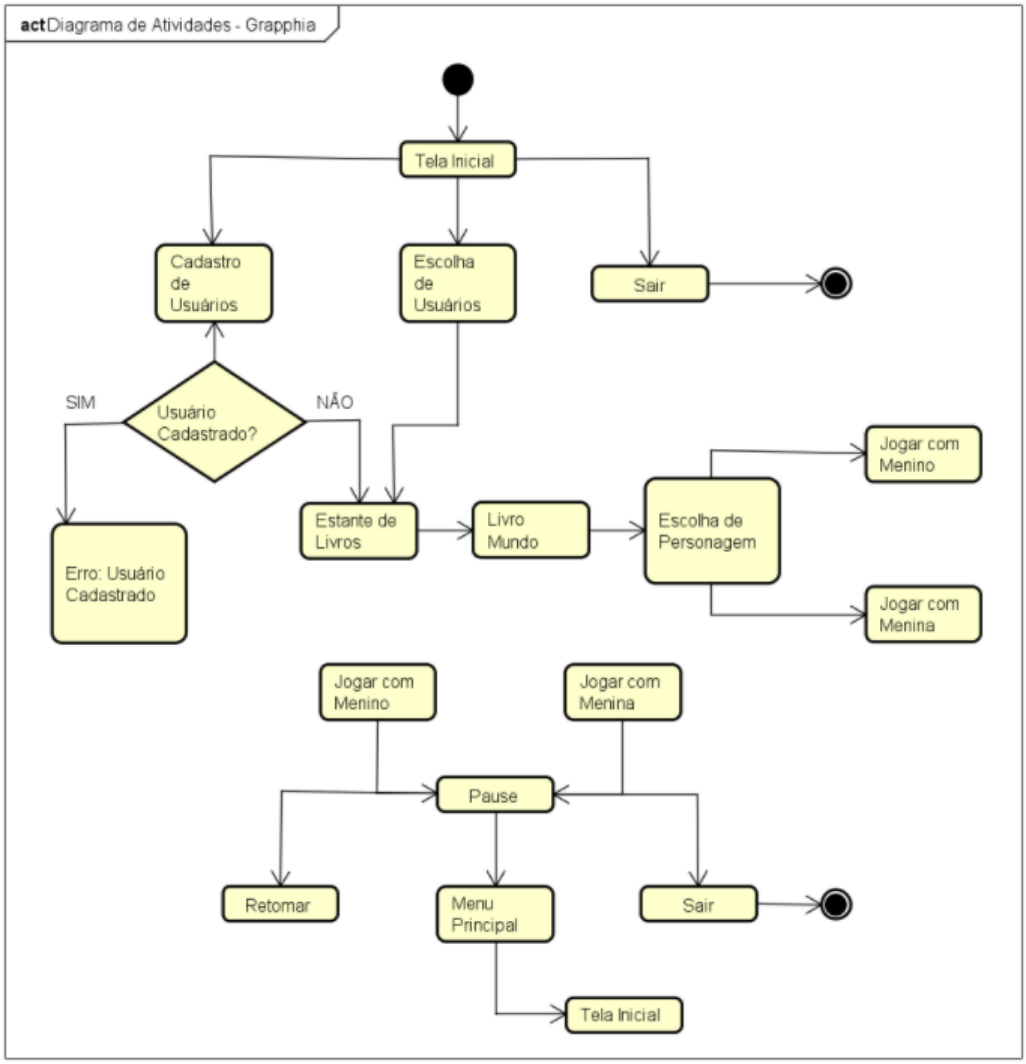

Figura 27 – *Diagrama de Atividades.*

Fonte: Elaborada pelo autor.

Figura 28 – *Tela Inicial - "Grapphia"*

Fonte: Elaborada pelo autor.

Figura 29 – *Telas de identificação do usuário.*

Fonte: Elaborada pelo autor.

A figura 30 representa a parte de seleção de módulo do sistema. Uma vez selecionado um livro, o aplicativo o abre e apresenta uma história ilustrada. O livro “A Fazenda” apresenta várias palavras grafadas com “s” e “z” concorrentes, dentre elas: paisagem, buzina, azul, tesoura, casamento, rosas, raposa, entre outras. O principal objetivo da história é proporcionar um primeiro contato das crianças com as palavras dentro de um contexto.

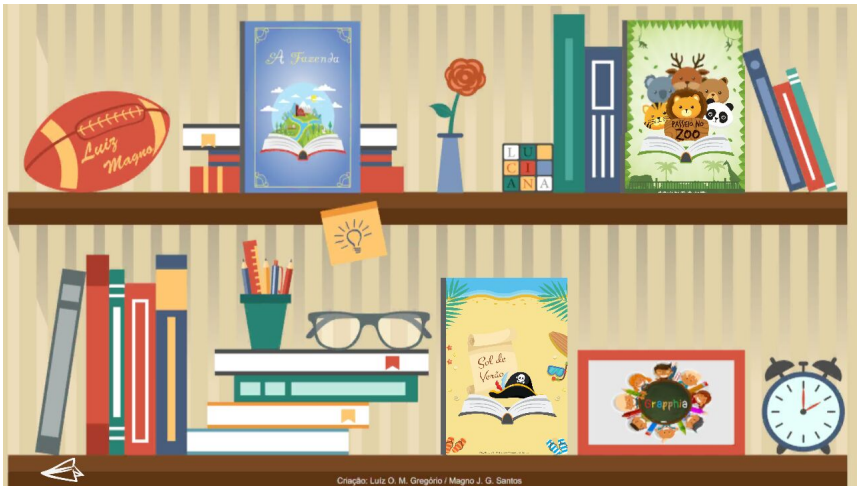

Figura 30 – *Tela de seleção de módulo.*

Fonte: Elaborada pelo autor.

A figura 31 apresenta a capa do livro e uma parte da história. Ao terminar a história o usuário poderá, então, entrar no jogo. Para isso, inicialmente deve selecionar um personagem (figura 32).

Figura 31 – *Telas do livro que contém a história, contextualizando as palavras que serão trabalhadas no módulo selecionado.*

Fonte: Elaborada pelo autor.

Figura 32 – *Tela de seleção de personagem.*

Fonte: Elaborada pelo autor.

Em seguida, dentro de um cenário que remete ao ambiente da fazenda, palavras irregulares são apresentadas, ocultando-se a letra concorrente. Duas letras são apresentadas como opção para completar a palavra: “s” e “z”. Ao selecionar a letra, o sistema indica se a criança acertou ou não (figura 33).

Figura 33 – *Tela do jogo.*

Fonte: Elaborada pelo autor.

Após passar pelo jogo, a última etapa consiste em um ditado que foi construído utilizando o próprio aplicativo. Ressalta-se que serão utilizadas as mesmas palavras do banco de dados do aplicativo ao qual os alunos terão sido expostos.

Figura 34 – *Tela de orientações que antecede o ditado.*

Fonte: Elaborada pelo autor.

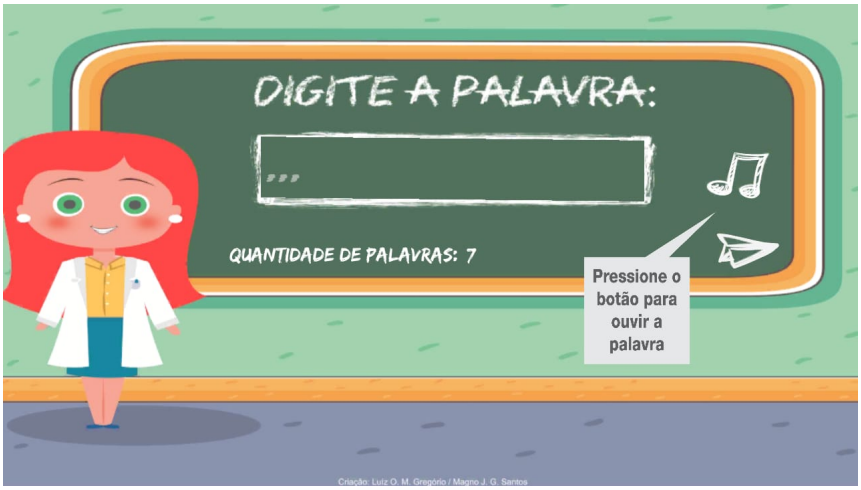

Figura 35 – *Tela de ditado.*

Fonte: Elaborada pelo autor.

Ao finalizar o ditado, os utilizadores do sistema serão direcionados a tela de "fim de jogo", onde será apresentada uma cena de parabenização (figura 36). Vale ressaltar que o usuário deve selecionar a opção de salvar para continuar jogando e explorar outros módulos do sistema.

Figura 36 – *Tela de fim de jogo.*

Fonte: Elaborada pelo autor.

Selecionar a opção "Salvar", possibilita que o sistema encaminhe um *e-mail* contendo as informações necessárias para o estabelecimento de análises de cunho social e estatístico. Dessa forma, seleciona-se o *identify* (id), nome, número de acertos no jogo, número de acertos no ditado e uma amostragem de erros e acertos por palavra jogada de cada usuário que utilizar a aplicação (figura 37). É importante frisar que, caso o dispositivo móvel não possua conexão à internet no momento de avaliação do sistema proposto, é salvo uma cópia do arquivo (.txt) armazenado na memória interna do aparelho, na pasta da aplicação. Para envio do *e-mail* foi utilizado o protocolo *Simple Mail Transfer Protocol* (SMTP). Segundo Formice (2009), o SMTP é o protocolo padrão para envio de *e-mails* usado tanto para o envio da mensagem original do

seu computador até o servidor SMTP do provedor, quanto para transferir uma mensagem para outros servidores até que ela chegue ao servidor de destino. A figura 37 ilustra o envio de *e-mail* com os dados para análise, note que o arquivo enviado apresenta uma sequência de 15 palavras e atribui *true* para quando o usuário acertar a palavra e *false* para quando ele errar. Dados como identificador do usuário, nome e quantitativo de acertos e erros também são apresentados no arquivo.

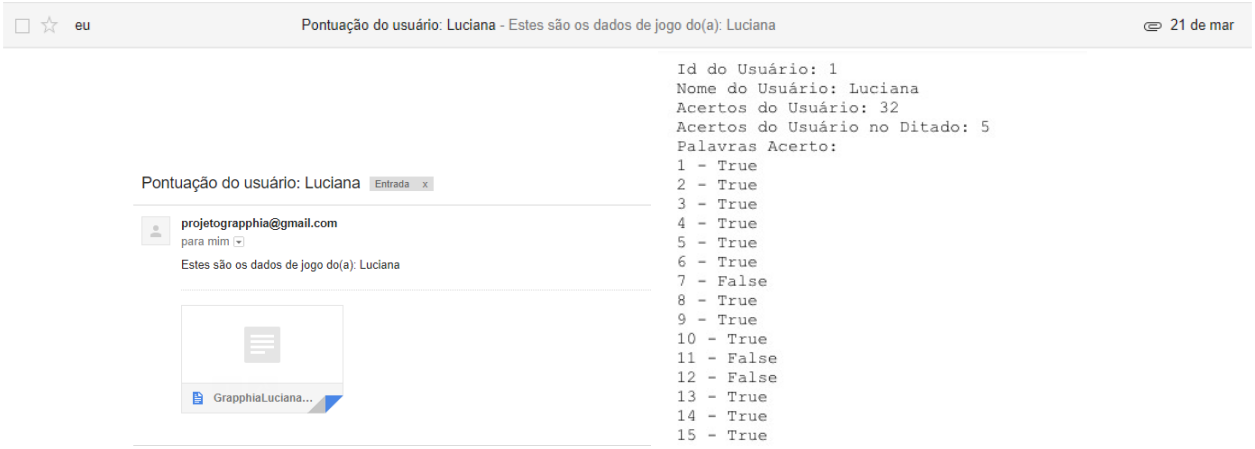

Figura 37 – *Envio de e-mail com as informações necessárias para análises.*

Fonte: Elaborada pelo autor.

A utilização do SMTP em C# (linguagem adotada para desenvolvimento da aplicação), é exemplificada na documentação da *Microsoft Developer Network*, através do seguinte link: ⟨https://msdn.microsoft.com/pt-br/library/system.net.mail.smtpclient(v=vs.110).aspx⟩. Na aplicação *Grapphia* utilizou-se:

- **Bibliotecas**

A seguir apresenta-se as bibliotecas utilizadas para que o SMTP seja funcional. Note que as bibliotecas *UnityEngine, UnityEngine.UI e UnityEngine.SceneManagement* são utilizadas para acessar componentes do *Unity*. As bibliotecas *System.Net, System.Net.Mail, System.Collections, System.IO, System.Net.Security, System.Security.Cryptography.X509Certificates* são bibliotecas providas pelo sistema, que possibilitam que o SMTP seja executado da forma que se pretende. Logo são informações indispensáveis para que o moto de jogo consiga integrar-se a uma plataforma de envio de *e-mail*.

```
using UnityEngine;
using UnityEngine.UI;
using System.Net;
using System.Net.Mail;
using System.Collections;
using UnityEngine.SceneManagement;
using System.IO;
using System.Net.Security;
```

```
using System.Security.Cryptography.X509Certificates;
```

Listing 4.1 – *Bibliotecas utilizadas para funcionamento do SMTP*

- **Funções**

As funções a seguir possibilitam que seja enviado o *e-mail* contendo as informações necessárias para realizar análises dos dados. Perceba que cada função atual sob uma responsabilidade definida, seguindo boas práticas de programação que dispõe sobre o princípio de responsabilidade única de métodos.

```
1  public void SalvaDadosJogado(){
       //Salvar txt para anexo
3      int idUsuario = dadosJogo.Instance.currentUser.Id;
       string nomeUsuario = dadosJogo.Instance.currentUser.Name;
5      int pontuacaoUsuarioAcerto = dadosJogo.Instance.currentUser.
           Score;
       int pontuacaoUsuarioAcertoDitado = dadosJogo.Instance.
           currentUser.scoreDitado;
7
       StreamWriter sw = new StreamWriter(Application.
           persistentDataPath + nomeUsuario + "-Pontuacao.txt");
9      sw.WriteLine("Id do Usuario: " + idUsuario);
       sw.WriteLine("Nome do Usuario: " + nomeUsuario);
11     sw.WriteLine("Acertos do Usuario: " + pontuacaoUsuarioAcerto
           );
       sw.WriteLine("Acertos do Usuario no Ditado: " +
           pontuacaoUsuarioAcertoDitado);
13     sw.WriteLine("Palavras Acerto: ");
       for(int i = 0; i < bancoPalavras.Instance.palavrasAcerto.
           Length; ++i) {
15       sw.WriteLine(bancoPalavras.Instance.palavrasAcerto[i].
             idPalavra + "-" + bancoPalavras.Instance.palavrasAcerto
             [i].acerto);
       }
17     sw.Close();
     }
19
   public void EnviarEmail()  {
21     //Codigo para enviar email utilizando o protocolo smtp
       MailMessage mail = new MailMessage();
23     string nomeUsuario = dadosJogo.Instance.currentUser.Name;

25     StartCoroutine("SalvaDadosJogado");

27     try {
         mail.From = new MailAddress("projetograpphia@gmail.com");
```

```
29          mail.To.Add ("projetograpphia@gmail.com");
            mail.Subject = "Pontua  o do usu rio: " + nomeUsuario;
31          mail.Body = "Estes s o os dados de jogo do(a): " +
                nomeUsuario ;
            mail.Attachments.Add (new Attachment (Application.
                persistentDataPath + nomeUsuario + " - Pontuacao.txt"));
                //anexo
33
            SmtpClient smtpServer = new SmtpClient ();
35          smtpServer.DeliveryMethod = SmtpDeliveryMethod.Network;
            smtpServer.Port = 587;
37          smtpServer.Host = "smtp.gmail.com";
            smtpServer.Credentials = new System.Net.NetworkCredential ("
                projetograpphia@gmail.com", "grapphia2017") as
                ICredentialsByHost;
39          smtpServer.EnableSsl = true;
            ServicePointManager.ServerCertificateValidationCallback =
41            delegate(object s, X509Certificate certificate, X509Chain
                  chain, SslPolicyErrors sslPolicyErrors) {
              return true;
43          };
            smtpServer.Send (mail);
45        } catch (SmtpException ex) {
            Debug.Log ("Exception: " + ex);
47        }

49        SceneManager.LoadScene ("telaEstante");
      }
```

Listing 4.2 – *Funções que possibilitam o armazenamento interno das informações no dispositivo móvel e o envio do e-mail*

4.1.3 Passo a passo: Construção da cena de Ditado

Com o intuito de tornar prático os conceitos apresentados neste trabalho, propõe-se a construção de uma das telas apresentadas no aplicativo *Grapphia*, correspondente ao ditado (figura 38).

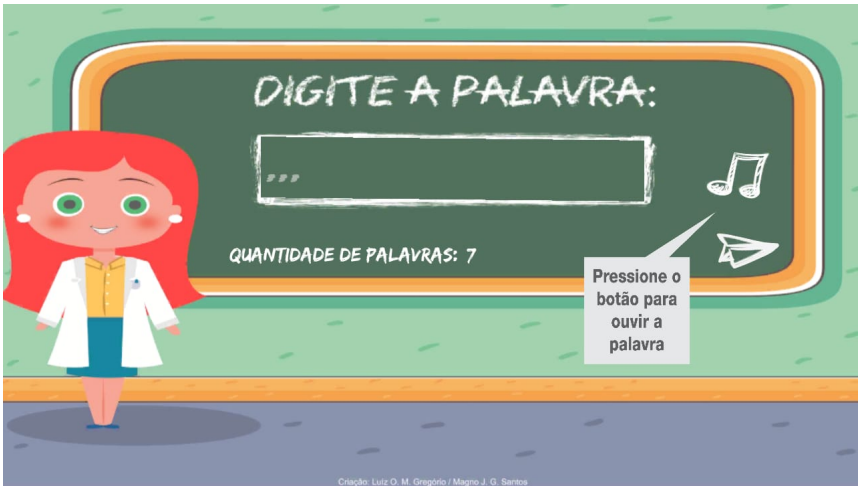

Figura 38 – *Tela de ditado.*

Fonte: Elaborada pelo autor.

Este capítulo destina-se a apresentar uma configuração básica de uma cena que compõe a solução implementada. Pretende-se criar uma cena desde o seu passo inicial até sua finalização, que servirá de base para a implementação das demais.

- **Passo 1**: Criação, nomeação e configuração da cena que iremos trabalhar.

No diretório temos a pasta *scenes*, responsável por alocar as cenas do projeto. Para criar uma cena deverá clicar com o botão direito do *mouse* na janela *Project*, selecionar *Create* e a opção *Scene*, conforme apresentado na figura 39.

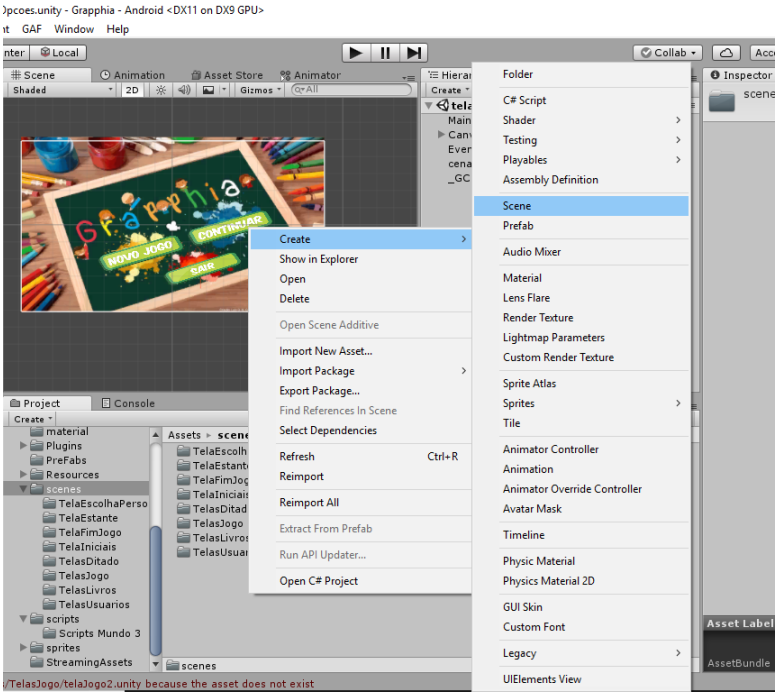

Figura 39 – *Procedimento para criação de cena.*

Fonte: Elaborada pelo autor.

Por padrão, as novas cenas criadas possuem dois *GameObjects*: *Main Camera* e *Directional Light*. A configuração inicial que será necessária estabelecer diz respeito a *Main Camera*, lembre-se de alterar o atributo *Projection* do componente Câmera, de *Perpective* para *Ortographic* (figura 40), para que os objetos presentes na cena sejam observados sem distorção de perspectiva. Em *Game* defina a resolução que sua aplicação irá suportar, neste trabalho adotou-se a resolução: 1280px x 720px.

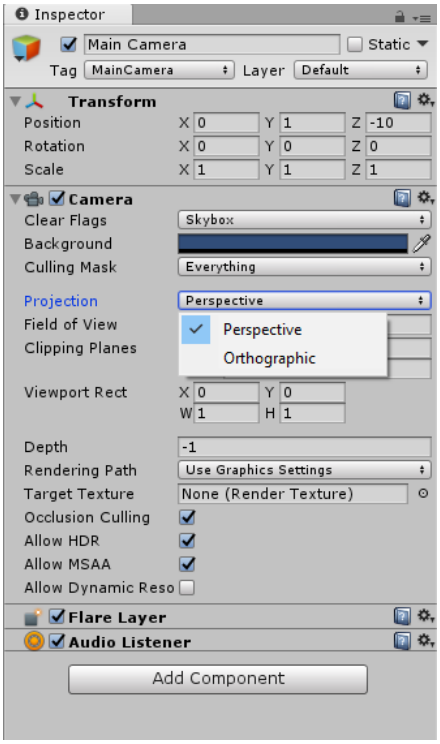

Figura 40 – *Procedimento para criação de cena.*

Fonte: Elaborada pelo autor.

- Passo 2: Criação do material contendo o cenário que será utilizado na cena.

Conforme explicitado no capítulo 3, um dos componentes de suma importância do *Unity* corresponde aos *Materials, Shaders e Textures*. Neste trabalho, utilizou-se materiais específicos para cada uma das cenas criadas.

No diretório temos a pasta *material*, onde são alocados os materiais de cada cena. Para criar um material deverá clicar com o botão direito do *mouse* na janela *Project*, selecionar *Create* e a opção *Material*, conforme apresentado na figura 41.

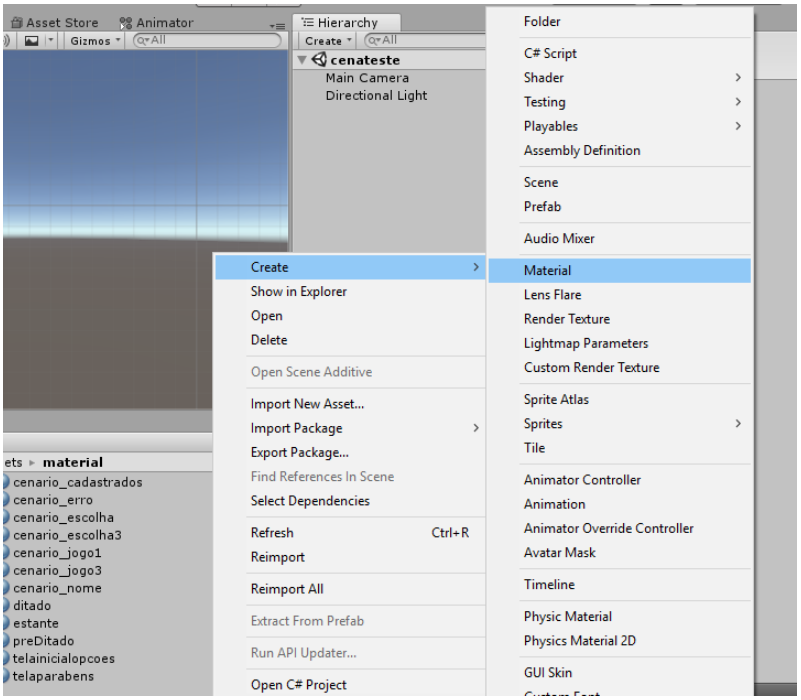

Figura 41 – *Procedimento para criação do material para geração do cenário da cena Ditado.*

Fonte: Elaborada pelo autor.

Ao criar um novo material será apresentado ao *Inspector* padrão de *materials* (figura 44), neste projeto utilizaremos o *shader*: *texture* (figura 43). Selecionado a textura que irá compor o cenário, estará pronto para desenvolvê-lo.

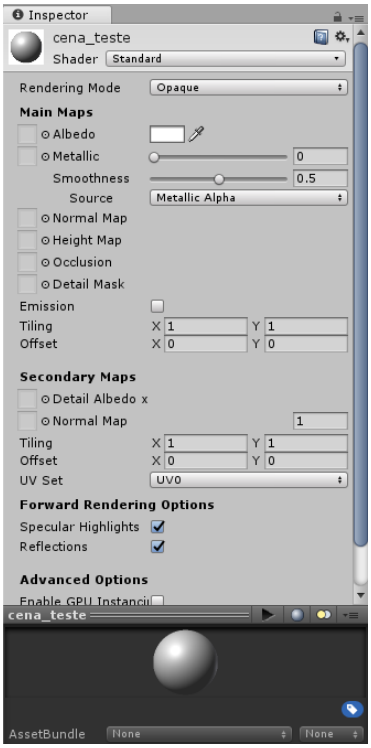

Figura 42 – *Inspector padrão do Material*

Fonte: Elaborada pelo autor.

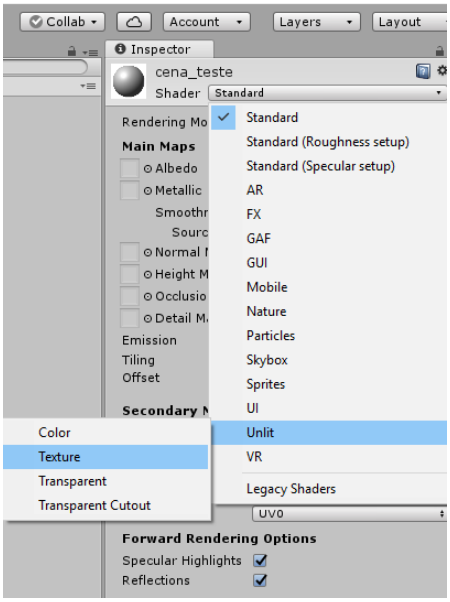

Figura 43 – *Texture para criação de material.*

Fonte: Elaborada pelo autor.

Feito isso, na cena criada adicione um *GameObject* para o cenário e os componentes *Mesh Filter* e *Mesh Renderer*. O *Mesh Filter* é responsável por pegar uma malha de seus ativos e passá-la para o *Mesh Renderer* para renderização na tela (UNITY, 2018). Vale ressaltar que este trabalho foi desenvolvido na plataforma 2D, recomenda-se a utilização do *Mesh Filter*: *Quad*.

O componente *Mesh Renderer* possui um atributo *Materials*, que deverá ser preenchido com o material criado.

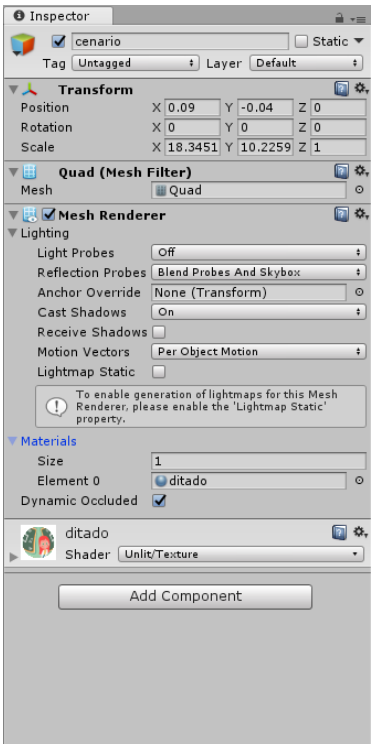

Figura 44 – *GameObject cenário com os componentes Mesh Filter e Mesh Renderer vinculados.*

Fonte: Elaborada pelo autor.

Para finalizar o cenário, deverá adicionar um componente de UI chamado *Canvas*, que representa o espaço destinado à adição de elementos de *interface* como: botões, caixas de texto, caixas de seleção, dentre outros. Uma configuração importante a ser realizada refere-se a resolução do *Canvas* que deverá ser a mesma da definida em *Game*. Para isso, deve-se alterar o atributo *UI Scale Mode* do componente *Canvas Scaler* (figura 46).

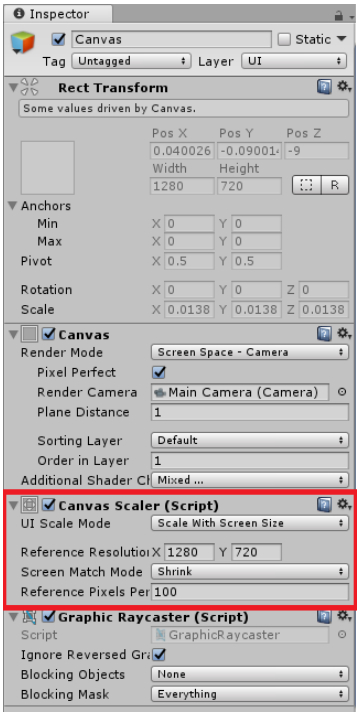

Figura 45 – *Alterar o Canvas Scaler para a resolução 1280px x 720px*

Fonte: Elaborada pelo autor.

O cenário da tela de Ditado apresenta uma animação que corresponde ao movimento da boca e olhos da professora ao narrar a palavra. Conforme já mencionado no capítulo 3, existem várias ferramentas para criação de animações disponíveis no mercado, porém o *Unity* viabiliza o desenvolvimento das mesmas dentro do seu próprio ambiente de desenvolvimento. Na pasta *animation*, deve-se ser criado dois componentes para a elaboração da animação o *Animator Controller* e o *Animation*. Para criá-los deve-se clicar com o botão direito do *mouse* sobre a janela *Project* selecionar a opção *Create* e *Animator Controller / Animation*, diante do exposto na figura 46.

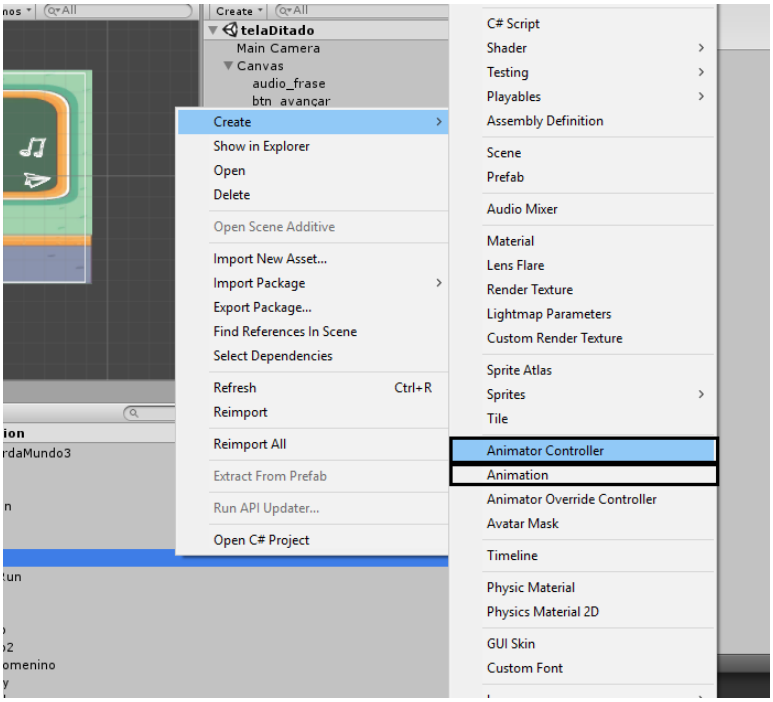

Figura 46 – *Criação do Animator Controller/Animation*

Fonte: Elaborada pelo autor.

As animações são criadas utilizando-se imagens denominadas *sprites* (figura 47). Estas correspondem a um conjunto de imagens que são utilizadas para “movimentar” os objetos na cena. Para utilizá-las no *Unity*, deve-se trabalhar com as seguintes configurações:

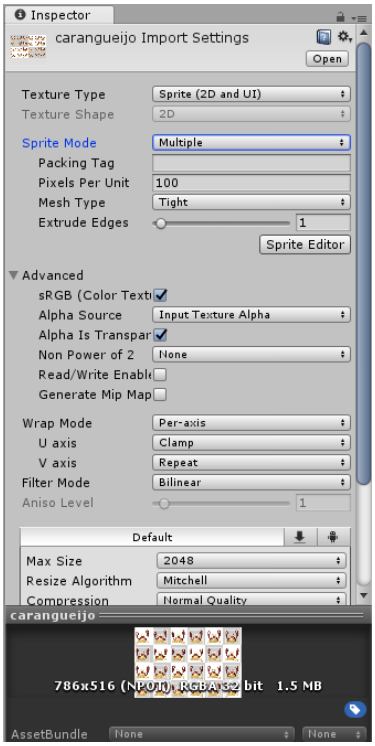

Figura 47 – *Configurações que devem ser utilizadas para trabalhar com sprites*

Fonte: Elaborada pelo autor.

Além disso o *Unity* apresenta a ferramenta *Sprite Editor*, onde é possível realizar o corte das *sprites* de forma homogênea (figura 48), através da ferramenta *Slice*.

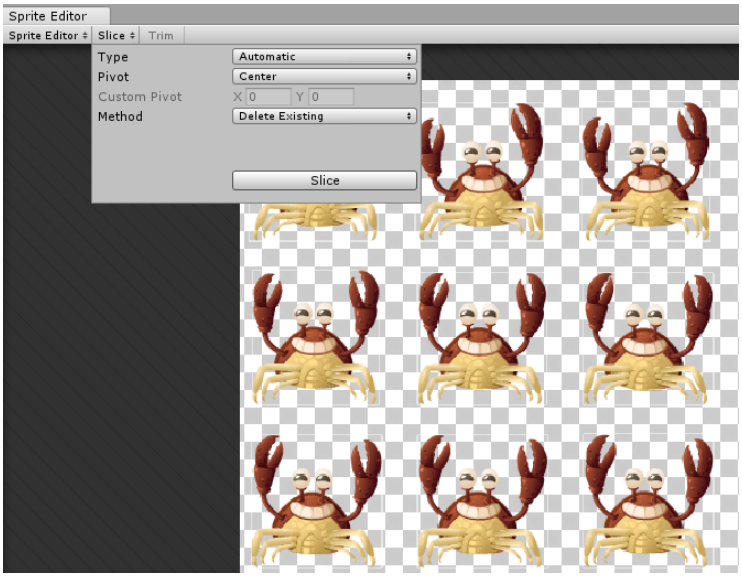

Figura 48 – *Sprite Editor com a opção Slice defina como Automatic*

Fonte: Elaborada pelo autor.

Após realizar o tratamento das imagens, deve-se criar um novo *GameObject* que irá receber a animação, lembre-se de vincular a ele um componente do tipo *Animator Controller*. Este componente permite que as animações sejam acrescentadas ao *GameObject*. Na janela

Animation crie uma nova animação (.anim) e arraste o conjunto de *sprites* que irão constituí-la (caso não esteja com a janela *Animation* habilitada, habilite-a clicando na aba *Window* > *Animation* ou através do atalho Ctrl + 6). Uma representação gráfica que apresenta essa janela pode ser visto na figura 49.

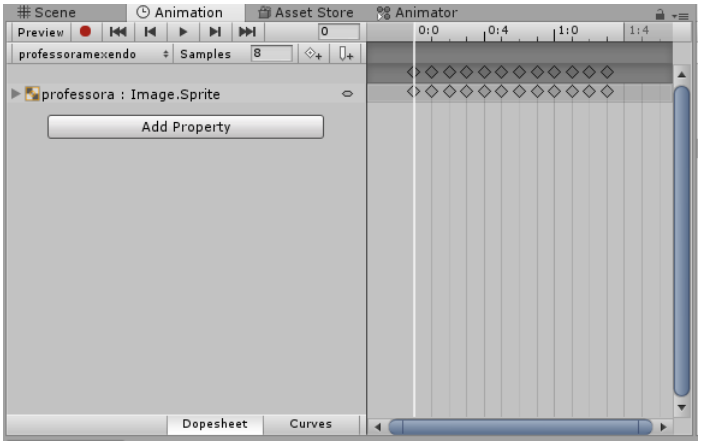

Figura 49 – *Criando uma animação, janela* ***Animation***

Fonte: Elaborada pelo autor.

É possível notar na janela de *Animator* a existência da transição do estado *"Entry"* para a animação criada, no caso deste exemplo a animação "professoramexendo" (figura 51).

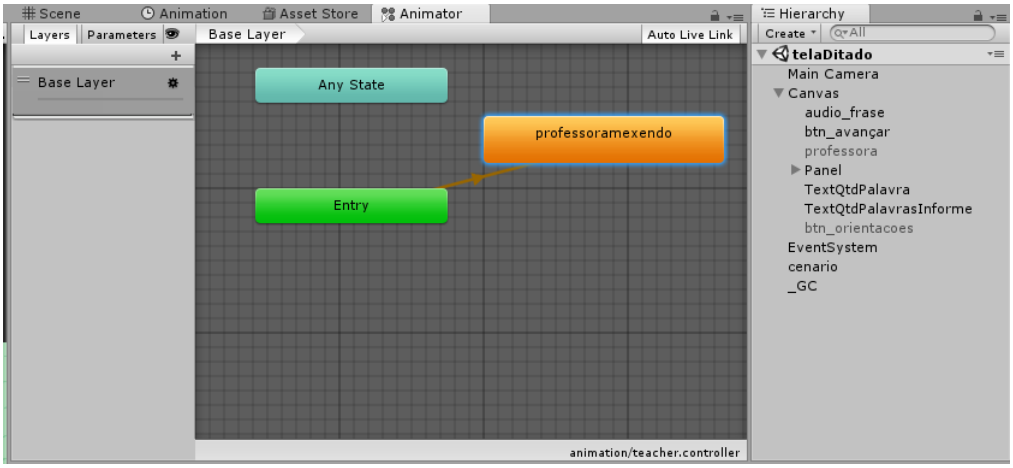

Figura 50 – *Animator representando a transição de estados de animações*

Fonte: Elaborada pelo autor.

Estes são os elementos que compõem a *interface* gráfica dessa cena. Diante disso, passamos para a etapa de criação de *scripts* para atribuição de eventos e comportamento. Para isso na pasta *scripts* do projeto, deve-se ser criado o *script* ditado.cs (lembrando que a extensão .cs refere-se a *scripts* que utilizam a linguagem C#). Para criar um novo *script* na janela *Project*, clique com o botão direito do *mouse*, selecione *Create* e posteriormente *C# Script*, conforme apresentado na figura 51.

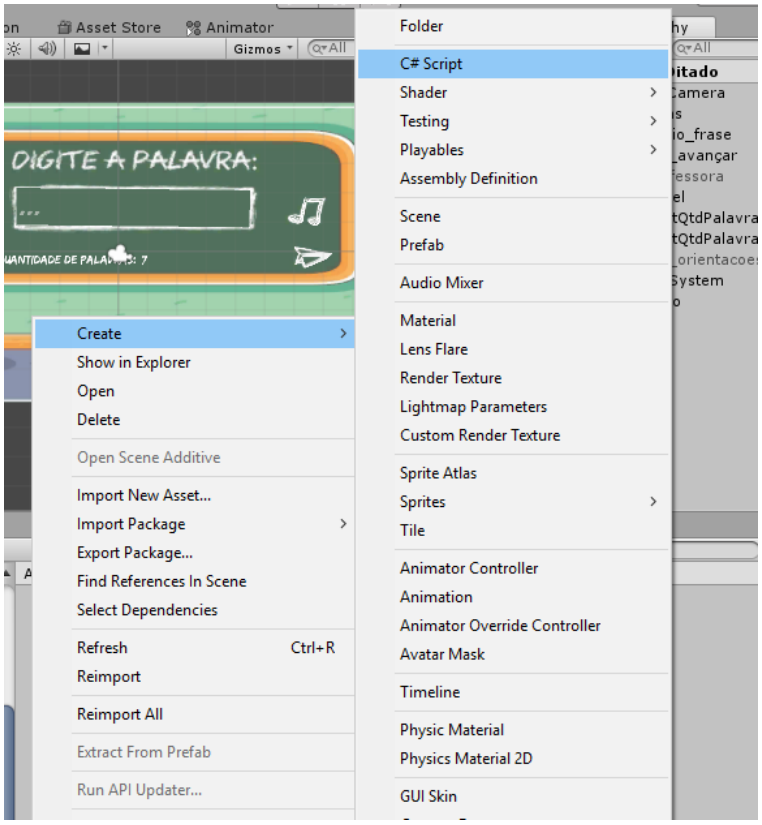

Figura 51 – *Criação de um script em C#*

Fonte: Elaborada pelo autor.

Por padrão os *scripts* em C# possuem a seguinte estrutura:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ditado1 : MonoBehaviour {

  // Use this for initialization
  void Start () {

  }

  // Update is called once per frame
  void Update () {

  }
}
```

Listing 4.3 – *Script padrão utilizando a linguagem C#*

Atribuiu-se o *script* ditado.cs para a cena de “Ditado” do sistema proposto. Em tal *script*, foram declaradas as seguintes variáveis:

```
    public GameObject animacaoProfessora;

private int idPalavra;

public InputField textoDigitado;

private int acertoPalavra, erroPalavra;

private int qtd_Palavras_Ditado = bancoPalavras.Instance.
    ListaIdPalavraAcerto.Count;

private int posicao;

public Text quantidadePalavrasApresentadas;

public GameObject orientacaoInicial;

private bool mensagemInicial_Aplicada = false;
```

Listing 4.4 – *Script ditado.cs utilizando a linguagem C#*

As variáveis de encapsulamento público encontram-se disponíveis para serem acessadas na *interface* do *Uniy*, por outro lado as de encapsulamento privado somente poderão ser acessadas e manipuladas dentro do *script* criado. Um *script* C# possui duas funções atribuídas por padrão, sendo elas: *Start()* e *Update()*. Ao inicializar a cena que possui o *script* ditado.cs vinculado, a princípio será chamado a função *Start()*, dessa forma tudo que estiver dentro dessa função será executado primeiro. A função *Update()* é chamada a cada *frame* sempre que houver uma atualização em tempo de execução da aplicação.

O *script* ditado.cs define a função *Start()*, como:

```
    void Start ()
{
    //Desativar o aparecimento constante da mensagem de
        orienta   o
    if (mensagemInicial_Aplicada == false) {
      orientacaoInicial.SetActive (true);
    } else {
      orientacaoInicial.SetActive (false);
    }

    posicao = Random.Range (0, qtd_Palavras_Ditado);
    idPalavra = bancoPalavras.Instance.ListaIdPalavraAcerto[
        posicao];
    bancoPalavras.Instance.ListaIdPalavraAcerto.RemoveAt (posicao)
        ;
    qtd_Palavras_Ditado--;
}
```

Listing 4.5 – *Script ditado.cs utilizando a linguagem C#*

Note que a função possui um controlador “mensagemInicial_Aplicada” para apresentar as orientações ao utilizador do sistema, além de um sorteio aleatório de palavras (dentre as jogadas pelo usuário nos módulos anteriores) através do chamamento do método *Random.Range()*. Dessa forma, evita-se que as palavras sejam apresentadas de forma repetida.

```
    public void PegaTexto (){
2     string palavraAnalisada = bancoPalavras.Instance.palavras [
          dadosJogo.Instance.currentUser.Nivel] [idPalavra].
          palavra_completa;
      string texto = textoDigitado.text;
4     string upperString = texto.ToUpper();

6     if (palavraAnalisada == upperString) {
        Debug.Log ("Acertou");
8       ++dadosJogo.Instance.currentUser.scoreDitado;
        textoDigitado.text = "";
10    } else {
        erroPalavra++;
12      textoDigitado.text = "";
      }
14
      int quantidadePalavrasAtual = bancoPalavras.Instance.
          numWordsDitado - 1;
16    quantidadePalavrasApresentadas.text = quantidadePalavrasAtual.
          ToString();
      quantidadePalavrasAtual--;
18
      bancoPalavras.Instance.numWordsDitado--;
20    StartCoroutine ("Start");

22    //Definir o numero de palavras no ditado no comandosBasicos.cs
      if (bancoPalavras.Instance.numWordsDitado == 0) {
24      //volta para 5 palavras, caso opte por jogar novamente no
            mesmo usu rio j  que essa vari vel  decrementada
        //durante o processo do ditado
26      bancoPalavras.Instance.numWordsDitado = 5;
        //toda vez que o jogo  executado a lista  novamente
            preenchida com todas as palavras do banco acrescida das
            palavras do jogo
28      bancoPalavras.Instance.ListaIdPalavraAcerto.Clear();
        SceneManager.LoadScene ("telaFimJogo");
30    }
  }
```

Listing 4.6 – *Script ditado.cs utilizando a linguagem C#*

O método *PegaTexto()* é responsável por comparar a palavra informada pelo usuário e a palavra presente no banco de dados da aplicação. Além disso, nele também se define a atualização da palavra e uma contagem da quantidade de palavras restantes a serem apresentadas. Tal método deve ser relacionado ao botão de avançar presente na cena anteriormente definida.

```
1    public void pressedAudioPalavra (){
       orientacaoInicial.SetActive (false);
3      mensagemInicial_Aplicada = true;

5      animacaoProfessora.SetActive (true);
       StartCoroutine ("audioEnd");
7      string arquivo = "audiosditado/" + bancoPalavras.Instance.
           palavras [dadosJogo.Instance.currentUser.Nivel] [idPalavra
           ].nome_audio_ditado;

9      AudioClip clip = (AudioClip)Resources.Load (arquivo);
       AudioSource audio;
11     audio = gameObject.AddComponent<AudioSource> ();
       audio.volume = 1;
13     audio.clip = clip;
       audio.Play ();
15   }

17   IEnumerator audioEnd(){
       string arquivo = "audiosditado/" + bancoPalavras.Instance.
           palavras [dadosJogo.Instance.currentUser.Nivel] [idPalavra
           ].nome_audio_ditado;
19
       AudioClip clip = (AudioClip)Resources.Load (arquivo);
21     yield return new WaitForSeconds (clip.length);
       animacaoProfessora.SetActive (false);
23   }
```

Listing 4.7 – *Script ditado.cs utilizando a linguagem C#*

Por outro lado o método *pressedAudioPalavra()* destina-se a atribuir o áudio da palavra narrada para que o utilizador tenha ciência de qual palavra deverá preencher. Note que o método *audioEnd()* do tipo *IEnumerator* é responsável por selecionar e atribuir o áudio da palavra narrada.

5 CONCLUSÕES E TRABALHOS FUTUROS

Conclui-se com a realização deste trabalho sobre a importância de jogos educacionais para dispositivos móveis no âmbito da educação, apresentando o aplicativo Grapphia. Este é composto por diferentes ferramentas lúdicas que tornam o processo de ensino de irregularidades ortográficas eficiente e motivante. Objetivou-se através da realização deste trabalho propiciar um descritivo das principais etapas que compõem o processo de desenvolvimento de jogos, além de gerar uma solução organizada e expansível seguindo a metodologia de desenvolvimento descrita nesse trabalho e boas práticas programação.

Também elencou-se as ferramentas que constituem o motor de jogo *Unity* tecnologia utilizada para o desenvolvimento da aplicação, fornecendo uma explicação detalhada de cada uma delas. Para os trabalhos futuros, espera-se a validação do sistema junto a alunos do ensino fundamental, para analisar o seu desempenho após a utilização do aplicativo. Além disso, a implementação de outros módulos do sistema a fim de sanar dificuldades ortográficas de mesma natureza.

REFERÊNCIAS

ALVARENGA, D. Análise de variações ortográficas. **Presença pedagógica**, v. 2, p. 24–35, 1995.

ASSIS, L.; BODOLAY, A.; GREGÓRIO, L.; SANTOS, M.; VIVAS, A.; PITANGUI, C.; BANDEIRA, D. Grapphia: Aplicativo para dispositivos móveis para auxiliar o ensino da ortografia. In: **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**. [S.l.: s.n.], 2017. v. 6, n. 1, p. 609.

BONSIEPE, G.; DUTRA, C. **Design: do material ao digital**. [S.l.]: FIESC/IEL, 1997.

COSTA, D. **Administração de Redes com scripts: Bash Script, Python e VBScript**. BRASPORT, 2007. ISBN 9788574523149. Disponível em: ⟨https://books.google.com.br/books?id=4WZHCnbhETIC⟩.

DOHME, V. D. **Atividades lúdicas na educação: o caminho de tijolos amarelos do aprendizado**. [S.l.]: Vozes, 2003.

FERREIRA, L. H. **Desenvolvimento de jogos eletrônicos utilizando a tecnologia Unity**. 2015. ⟨https://pt.slideshare.net/tecnoluisao/tcc-desenvolvimento-de-jogos-eletrnicos-utilizando-a-tecnologia-unity-luis-henrique-ferreira⟩. Acessado em 20/02/2018.

FORMICE, C. **Linux Configurações De Serviços De Rede Apostila Técnica**. [s.n.], 2009. Disponível em: ⟨https://books.google.com.br/books?id=7KZJBQAAQBAJ⟩.

JUNIOR, A. d. S. R.; NASSU, B. T.; JONACK, M. A. Um estudo sobre os processos de desenvolvimento de jogos eletrônicos(games). **Disponıvel em http://www. ademar. org/texts/-processodesenvgames. pdf. Ultimo acesso em**, v. 12, 2002.

LEAL, A. S. **Aplicação do método de aprendizagem por reforço Q-Learning na adaptatividade dinâmica de dificuldade de um jogo digital ortográfico**. 102 p. Monografia (Graduação) — Faculdade de Ciências Exatas e Tecnológicas, Diamantina, 2016.

LOPES, G. Jogos eletrônicos conceitos gerais. **Acedido a**, v. 30, 2010.

MENDES, C. **Jogos Eletrônicos: Diversão, Poder E Subjetivação**. Papirus, 2006. (Coleção Fazer/Lazer). ISBN 9788530808082. Disponível em: ⟨https://books.google.com.br/books?id=-ZFvA0rpbXMC⟩.

MÜLBERT, A. L.; PEREIRA, A. T. Um panorama da pesquisa sobre aprendizagem móvel (m-learning). **Associação Brasileira de Pesquisadores em Cibercultura**, 2011.

NOVAK, J. **Desenvolvimento De Games**. CENGAGE DO BRASIL, 2010. ISBN 9788522106325. Disponível em: ⟨https://books.google.com.br/books?id=QNKpcQAACAAJ⟩.

ONÇA, F. A. **A era dos games na sociedade da escolha**. Tese (Doutorado) — Universidade de São Paulo, 2014.

ORACLE. **Oracle Technology Network**. 2018. ⟨http://www.oracle.com/technetwork/index.html⟩. Acessado em 24/01/2018.

PASSOS, E. B.; JR, J. R. da S.; RIBEIRO, F. E. C.; MOURÃO, P. T. Tutorial: Desenvolvimento de jogos com unity 3d. In: **VIII Brazilian Symposium on Games and Digital Entertainment**. [S.l.: s.n.], 2009. p. 1–30.

PEREIRA, D. R. A contribuição dos jogos e brincadeiras no processo de ensino-aprendizagem de crianças de um cmei na cidade de teresina. **Revista Fundamentos**, v. 2, n. 2, 2016.

RAFFCOM. **O que é User Interface?** 2018. ⟨https://www.raffcom.com.br/blog/o-que-e-ui/⟩. Acessado em 12/04/2018.

ROGERS, S. **Level Up: UM GUIA PARA O DESIGN DE GRANDES JOGOS**. EDGARD BLUCHER, 2012. ISBN 9788521207009. Disponível em: ⟨https://books.google.com.br/books?id=8oZVmwEACAAJ⟩.

ROSA. Producao para games. 2014.

SOARES, M. Alfabetização: a questão dos métodos. **São Paulo: Contexto**, 2016.

STEAM. 2018. ⟨http://store.steampowered.com/?l=portuguese⟩. Acessado em 06/04/2018.

STOY, C. Game object component system. **Game Programming Gems**, v. 6, n. 393-403, p. 44, 2006.

TORRES, R. D. Desenvolvendo um jogo para ensinar física com unity 3d. 2016.

UNITY. 2018. ⟨https://unity3d.com/pt/company⟩. Acessado em 24/01/2018.

WATKINS, A. **Criando jogos com Unity e Maya: como desenvolver jogos 3D divertidos e de sucesso**. [S.l.]: Rio de Janeiro: Elsevier, 2012.

WIKIDOT. **Linguagens mais utilizdas no Unity3D**. 2018. ⟨http://unity3d.wikidot.com/linguagens⟩. Acessado em 11/04/2018.

WU, W.-H.; WU, Y.-C. J.; CHEN, C.-Y.; KAO, H.-Y.; LIN, C.-H.; HUANG, S.-H. Review of trends from mobile learning studies: A meta-analysis. **Computers & Education**, Elsevier, v. 59, n. 2, p. 817–827, 2012.

ZORZI, J. L. **Aprender a escrever: a apropriação do sistema ortográfico**. [S.l.: s.n.], 1998.

ZORZI, J. L. **Aprendizagem e distúrbios da linguagem escrita: questões clínicas e educacionais**. [S.l.]: Artmed Editora, 2009.