UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

VISEDU: JOGO DE REALIDADADE AUMENTADA DE LETRAS COM CONTEÚDO DINÂMICO

VIVIAN DE LIMA PANZENHAGEN

BLUMENAU
2016

VIVIAN DE LIMA PANZEHAGEN

VISEDU: JOGO DE REALIDADE AUMENTADA DE LETRAS COM CONTEÚDO DINÂMICO

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Dalton Solano Dos Reis , Mestre - Orientador

BLUMENAU
2016

VISEDU: JOGO DE REALIDADE AUMENTADA DE LETRAS COM CONTEÚDO DINÂMICO

Por

VIVIAN DE LIMA PANZENHAGEN

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Dalton Solano dos Reis, Mestre – Orientador, FURB
Membro:	Prof. Alexander Roberto Valdameri, M.Sc. – FURB
Membro:	Prof. Matheus Carvalho Viana, Dr. – FURB

Blumenau, 4 de julho de 2016

Dedico este trabalho aos meus pais, ao meu noivo e ao meu orientador que me apoiaram desde o início até a conclusão do mesmo.

AGRADECIMENTOS

A meu pai Dilmar Panzenhagen e minha mãe Marli Aparecida de Lima por todo incentivo e confiança.

Ao meu noivo Bruno André Kestring pelo auxilio e paciência para o desenvolvimento deste trabalho.

Aos meus amigos que contribuíram direta ou indiretamente para o desenvolvimento deste trabalho.

Ao meu orientador Dalton, por seu apoio, interesse e tempo investidos neste trabalho.

RESUMO

Este trabalho apresenta a extensão do trabalho de Lira (2015) através do desenvolvimento de um aplicativo que utiliza Realidade Aumentada com conteúdo dinâmico para o reconhecimento de palavras. Para o desenvolvimento do aplicativo foi utilizado a plataforma de criação de jogos e aplicativos Unity 3D em conjunto com a biblioteca Vuforia. Durante o desenvolvimento deste trabalho foi implementado uma forma de utilizar imagens de forma customizada pelo usuário. Além disso, são apresentados conceitos, técnicas e componentes utilizados para a implementação da Realidade Aumentada. Como resultado, foi desenvolvido um aplicativo que permite ao usuário imergir em um cenário de Realidade Aumentada através do reconhecimento de texto utilizando a câmera de um dispositivo Android.

Palavras-chave: Android. Realidade aumentada. Reconhecimento de texto. Unity 3D. Vuforia.

ABSTRACT

This paper presents the development of an application that uses Augmented Reality with dynamic content for the recognition of words that involves the extension of a library for creating games. For application development has been used to create games platform and Unity 3D applications in conjunction with the Vuforia library. During the development of this work was implemented a way to use images in a customized manner by the user. Furthermore, they are presented concepts, techniques and components used for the implementation of Augmented Reality. As a result, it developed an application that allows the user to immerse yourself in a Augmented Reality scenario through text recognition using the camera of an Android device.

Key-words: Android. Augmented reality. Text recognition. Unity 3D. Vuforia.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

IDE – *Integrated Development Environment*

JDK – *Java Development Kit*

JPG – *Joint Photographic Group*

LIBRAS – Língua Brasileira de Sinais

MB – *Megabyte*

PNG – *Portable Network Graphic*

RA – Realidade Aumentada

RF- Requisito Funcional

RNF – Requisito Não Funcional

RV – Realidade Virtual

SDK – *Software Development Kit*

SUMÁRIO

1 INTRODUÇÃO

Os jogos atraem pessoas há muito tempo, sendo usados para lazer e o desenvolvimento cognitivo, não somente na infância, mas como em outros momentos. De uma forma geral, os jogos podem ser ferramentas eficientes durante o período de aprendizagem, pois eles divertem enquanto motivam aumentando a capacidade de absorver o que é proposto no cenário do jogo (FABRE, 2004).

Acredita-se que a RA possa contribuir no processo de aprendizagem, pois disponibiliza uma forma diferente de representação de conteúdo. Segundo Araújo (2009), esta tecnologia permite, a partir da projeção de objetos, uma maior interação entre o usuário e o conteúdo exposto possibilitando um melhor entendimento sem um treinamento prévio da tecnologia utilizada.

As bases da RA surgiram na década de 1960 com o pesquisador Ivan Sutherland, que escreveu um artigo sobre a possível evolução da Realidade Virtual (RV) e seus reflexos no mundo real. Ele desenvolveu um capacete de visão ótica direta rastreada para visualização de objetos 3D no ambiente real (SUTHERLAND, 2011). Porém, somente na década de 1980 surgiu o primeiro projeto de Realidade Aumentada, que foi desenvolvido pela Força Aérea Americana. Consistindo de um simulador de *cockpit* de avião, com visão ótica direta, misturando elementos virtuais com o ambiente real do usuário (TORI; KIRNER; SISCOUTTO, 2006).

Atualmente, a utilização do conceito de Realidade Aumentada para o ensino pode propiciar um melhor aproveitamento dos conteúdos apresentados nas escolas (ELER, 2015). Isso acontece por meio de aplicativos que auxiliam na alfabetização nas séries iniciais ou até mesmo de assuntos que se tornam difíceis de compreender por meio da ilustração de um livro, tal como a ilustração do fluxo de elétrons em um circuito.

Diante desse contexto, este trabalho foi realizado o desenvolvimento de um aplicativo de Realidade Aumentada com conteúdo dinâmico para dispositivos móveis, através de um jogo para o reconhecimento de palavras por meio de um marcador.

1.1 OBJETIVOS

O objetivo deste trabalho é estender a biblioteca desenvolvida por Lira (2015) por meio de um aplicativo de Realidade Aumentada utilizando marcadores, em um jogo de letras para dispositivos móveis com conteúdo dinâmico. Os objetivos específicos são:

a) utilizar marcadores contendo letras para o reconhecimento da palavra;
b) possibilitar a utilização de imagens definidas pelo usuário;

c) possibilitar inserir a imagem definida pelo usuário em diferentes níveis de dificuldade.

1.2 ESTRUTURA

A estrutura deste trabalho é apresentada em quatro capítulos. O primeiro capítulo apresenta a introdução do trabalho e seus objetivos gerais e específicos. O segundo capítulo contém a fundamentação teórica necessária para o entendimento deste trabalho. O terceiro capítulo apresenta as etapas de desenvolvimento deste trabalho, apresentando a especificação através de diagrama de caso de uso, diagrama de classes e diagramas de atividades. Ainda no terceiro capítulo são apresentadas as principais técnicas de implementação, trechos de códigos, imagens e explicações textuais. Neste capítulo também são apresentadas a operacionalidade do aplicativo desenvolvido e os resultados e discussões a respeito do desenvolvimento e uma breve comparação entre este trabalho e os trabalhos correlatos apresentados no capítulo 2. Por fim, o quarto capítulo apresenta a conclusão do trabalho e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Inicialmente a Seção 2.1 apresenta o conceito de Realidade Aumentada e suas características. Na Seção 2.2 são feitos comentários do trabalho de Lira (2015), no qual esse trabalho se baseia. Na Seção 2.3 é abordado o conceito de iteratividade. A Seção 2.4 apresenta o Unity e seus elementos básicos e na Seção 2.5 é apresentado o *Software Development Kit* (SDK) Vuforia. Por fim, a Seção 2.6 apresenta três trabalhos correlatos e um comparativo entre eles, respectivamente.

2.1 REALIDADE AUMENTADA

A Realidade Aumentada pode ser definida como o enriquecimento do ambiente real com objetos virtuais, usando algum dispositivo tecnológico e funcionando em tempo real (TORI; KIRNER; CISCOUTO, 2006). Possui a vantagem de proporcionar uma experiência utilizando tato, gestos, voz, facilitando a interação do usuário sem qualquer tipo de treinamento. Por estes motivos, pode ser utilizada em vários segmentos, com diversos propósitos. Dentre as várias aplicações de Realidade Aumentada, podem-se citar medicina, treinamento militar, estratégias comerciais e em jogos.

Segundo Azuma (1997, p.2), a Realidade Aumentada é um ambiente que envolve a Realidade Virtual com elementos do mundo real, criando um ambiente combinado em tempo real. Ou seja, ela combina elementos virtuais com o ambiente real sendo interativa e tem processamento em tempo real concebido em três dimensões, largura, altura e profundidade.

Para Tori, Kirner e Ciscouto (2006, p 24) a Realidade Aumentada envolve três aspectos importantes: renderização de alta qualidade do mundo combinado; calibração precisa, envolvendo o alinhamento dos objetos virtuais em posição e orientação dentro do mundo real; interação em tempo real entre objetos reais e virtuais (TORI; KIRNER; CISCOUTO, 2006).

2.2 FURBRA-LIBRARY - BIBLIOTECA PARA DESENVOLVIMENTO DE JOGOS MULTI JOGADORES DE REALIDADE AUMENTADA PARA ANDROID

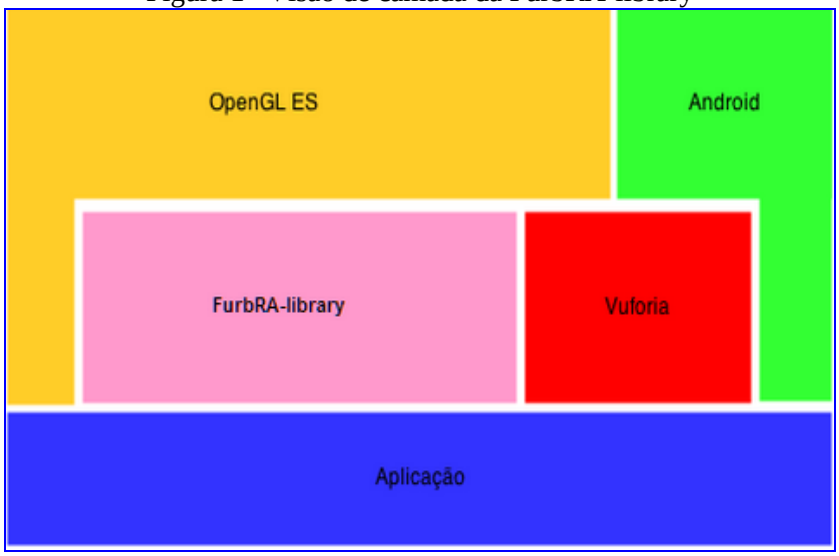
A biblioteca FurbRA-library tem como objetivo auxiliar no desenvolvimento de jogos de Realidade Aumentada (RA) para Android, baseada em marcadores. É oferecido suporte a dois tipos de marcadores, do tipo frame e os de palavras. Lira (2015) desenvolveu a detecção de marcadores através da comunicação com a biblioteca Vuforia. Os marcadores do tipo frame são marcadores de identificação única codificados com um padrão binário ao longo das bordas. Foi utilizado também a *Application Programming Interface* (API) de detecção de

textos fornecida pela biblioteca Vuforia, que obriga a configuração das palavras, previamente à execução do aplicativo, por meio de um arquivo. A API permite que palavras não existentes nesse arquivo sejam detectadas por meio de uma lista adicional, podendo conter até 10 mil palavras, fornecida à biblioteca por meio de um arquivo de texto.

A biblioteca FurbRA-library, disponibiliza ainda um conjunto de funções e recursos de forma a facilitar o gerenciamento do áudio e da comunicação local entre os aparelhos através da especificação de rede WiFi. Concede também um conjunto de funções e recursos de que facilitam a utilização da biblioteca OpenGL ES necessária no desenvolvimento gráfico dos jogos de RA.

A FurbRA-library é uma camada intermediária que agrupa quatro áreas necessárias a uma biblioteca de RA para jogos multijogadores: renderização gráfica, RA, comunicação e multimídia. A biblioteca se localiza entre a aplicação, a biblioteca OpenGL ES (utilizada como biblioteca gráfica), a biblioteca Vuforia (utilizada para a criação da RA) e o *framework* Android (utilizado para comunicação e multimídia), como pode ser observado na Figura 1 (LIRA, 2015).

Figura 1– Visão de camada da FurbRA-library



Fonte: Lira (2015, p.32).

Para demonstrar a operacionalidade da biblioteca FurbRA-library, Lira (2015) desenvolveu um jogo de RA, denominado Animais. Este jogo tem propósito educativo de auxiliar na alfabetização de crianças. Nele, crianças em etapa de alfabetização utilizarão peças com letras para montar o nome de animais que são apresentados na tela do dispositivo.

Para iniciar o jogo, o jogador deve escolher a opção “Com amiguinhos ou Sem amiguinhos”. A opção “Sem amiguinhos” inicia um jogo para participação de apenas um

jogador (*singleplayer*), no qual o mesmo deve utilizar as peças, para montar o nome de animais que são mostrados em tela.

A opção “Com amiguinhos” inicia um jogo competitivo (*multiplayer*) baseado em turnos. Quando esta opção é pressionada, o sistema apresenta uma listagem das instâncias do aplicativo que estão rodando na rede local. Após selecionar qualquer um dos usuários é realizado um convite de jogo ao mesmo. Enquanto não houver resposta por parte dos outros usuários, é demonstrada uma tela de carregamento. Caso haja o aceite, o sistema apresenta uma tela de carregamento com as instruções do jogo.Em estrutura de turnos, ou seja, cada grupo de jogadores que revezam a atividade, é apresentada a figura de um animal em um dos dispositivos enquanto nos demais é apresentada uma mensagem de espera.

Para auxiliar a identificação do animal, caso o jogador toque na tela do dispositivo, o aplicativo emite o som do animal. No dispositivo onde a figura do animal for apresentada o usuário deverá utilizar as peças com as letras de maneira que essas formem o nome do animal exibido. O jogador deve mostrar a palavra formada para a câmera do dispositivo de modo que a aplicação possa reconhecer se a mesma foi escrita corretamente. O jogador terá dois minutos para executar essa operação de forma correta, após esse tempo o usuário perderá o turno. Quando o usuário acertar a formação da palavra o mesmo será notificado e o turno é passado para o outro usuário. O jogo possui oito animais cadastrados, sendo que um animal nunca é mostrado duas vezes nem para mais de um usuário. O jogo termina quando todos os oito animais forem apresentados aos usuários. Vence aquele que tiver acertado o maior número de animais.

A aplicação também possui as mensagens em tela do jogo mostradas na forma de Língua Brasileira de Sinais (LIBRAS) de maneira a, além de estimular a alfabetização, estimular na criança do aprendizado de LIBRAS. Pode-se observar na Figura 2, que no início do jogo é exibida uma imagem (no caso, de um macaco) e o jogador deve posicionar os marcadores para formar a palavra que representa a imagem. A aplicação foi desenvolvida com suporte mínimo à plataforma Android 4.1 (Jelly Bean) e utilizando a IDE Android Studio 1.2.1.1

Figura 2- Aplicação animais

Fonte: Lira (2015, p.81).

2.3 INTERATIVIDADE

Segundo Silva a interatividade pode ser classificado como “qualquer coisa cujo funcionamento permite ao seu usuário algum nível de participação ou troca de ações.” (SILVA, 2001, não paginada). A interatividade é um elemento essencial do jogo eletrônico, é o que fundamentalmente diferencia o jogo de outras mídias similares, como a televisão e o cinema.

No meio de toda a história dos jogos eletrônicos, a interatividade sempre tentava aparecer e conquistar os jogadores, mas nunca conseguia de forma satisfatória. Ou porque não atraía, ou porque a tecnologia não era tão boa assim.

No Nintendo 8 Bits surgia em 1989 a Power Glove, uma luva que detectava movimentos, conforme Figura 3. Entretanto não teve muita aceitação por mau funcionamento e desconforto (TECMUNDO,2009).

Figura 3- Luva de detecção de movimentos

Fonte: Tecmundo (2009).

Em jogos de RA, a interatividade é um fator importante para o sucesso da aplicação. Nos jogos desta categoria, o fato do jogador poder executar interações entre um ambiente real com elementos de uma cena virtual é a maneira de conquistar o público para estes tipos de jogos.

2.4 UNITY

Unity é uma plataforma de desenvolvimento flexível e eficiente, usada para criar jogos e experiências interativas 3D e 2D, como simulações de treinamento e visualizações para médicos e arquitetos, através de dispositivos móveis, Internet, desktop, console e outras plataformas (UNITY, 2015a). Uma vantagem da Unity é a disponibilização de ferramentas de aprendizado para o desenvolvedor. Estão disponíveis vários tutoriais, além de toda a documentação necessária para desenvolver utilizando as classes da Unity em seus *scripts*. Também é possível realizar o *download* de elementos gráficos, a partir da loja oficial da Unity a *Asset Store*. A loja possui elementos disponíveis, onde é possível encontrar modelos simples ou até mesmo projetos completos, onde o desenvolvedor pode conhecer e aprender mais. Existem vários elementos gratuitos ou que podem ser utilizados no desenvolvimento do projeto.

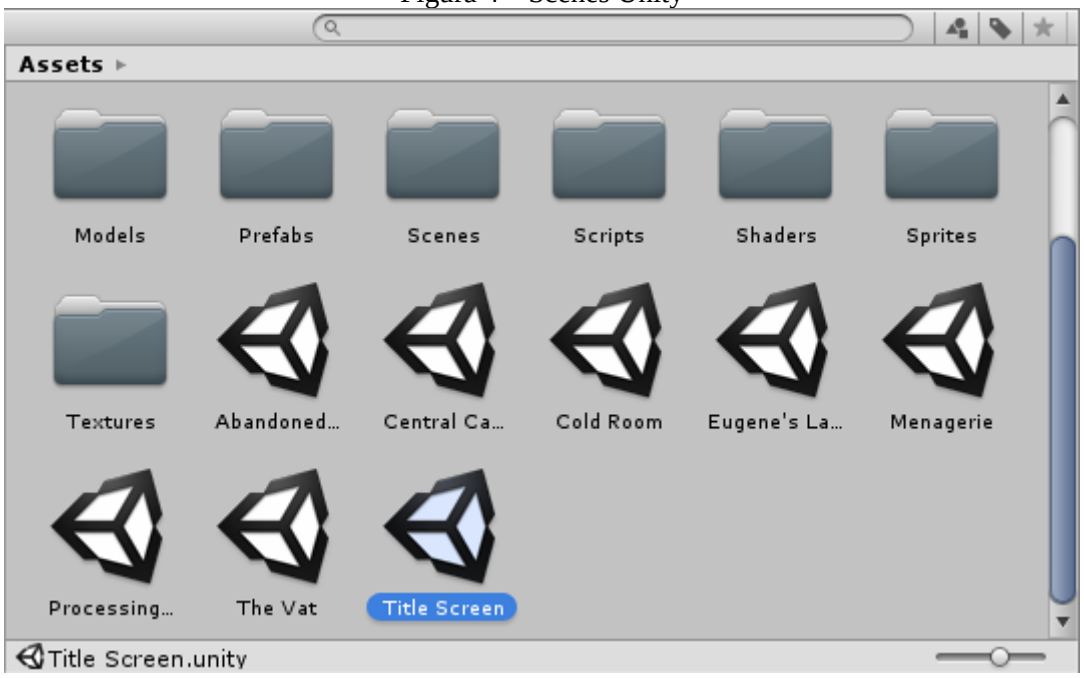
A Unity conta com a *Unity Asset Server*, uma solução de controle de versão para os *assets* e *scripts* utilizando PostgreSQL como um *backend*. Além disso, ela também foi construída com suporte para a *engine* de física Nvidia PhysX (a partir da Unity 3.0) com suporte adicional para a simulação de tecido em tempo real, *raycasts* e camadas de colisão (DEVMEDIA, 2015).

Para o desenvolvimento de aplicações de RA no ambiente Unity é necessário o *plugin Unity Extension* (QUALCOMM, 2015), que realiza a junção do Unity e o Vuforia. Com este *plugin* é possível desenvolver o jogo e a interação com a RA, como animações para serem atreladas a marcadores, além de interações dos objetos virtuais dos marcadores com o ambiente do jogo e o usuário.

2.4.1 Elementos básicos

Um aplicativo no Unity pode conter vários níveis. Os níveis contêm objetos distintos entre si para a formação de seu cenário. Estes níveis são nomeados de `Scenes`. As `Scenes` contém os objetos do jogo. Eles podem ser usados para criar um menu principal, níveis individuais ou qualquer outra coisa. A Figura 4 mostra algumas `Scenes`.

Figura 4 – Scenes Unity



Fonte: Unity (2016a).

Um importante componente do Unity é o `Event System`. Este componente é um gestor e facilitador em termos de entrada de usuário (toque na tela, arraste de mouse entre outros tipos de entrada). Sua funcionalidade consiste no envio de eventos para objetos com base na entrada do usuário (UNITY, 2016b).

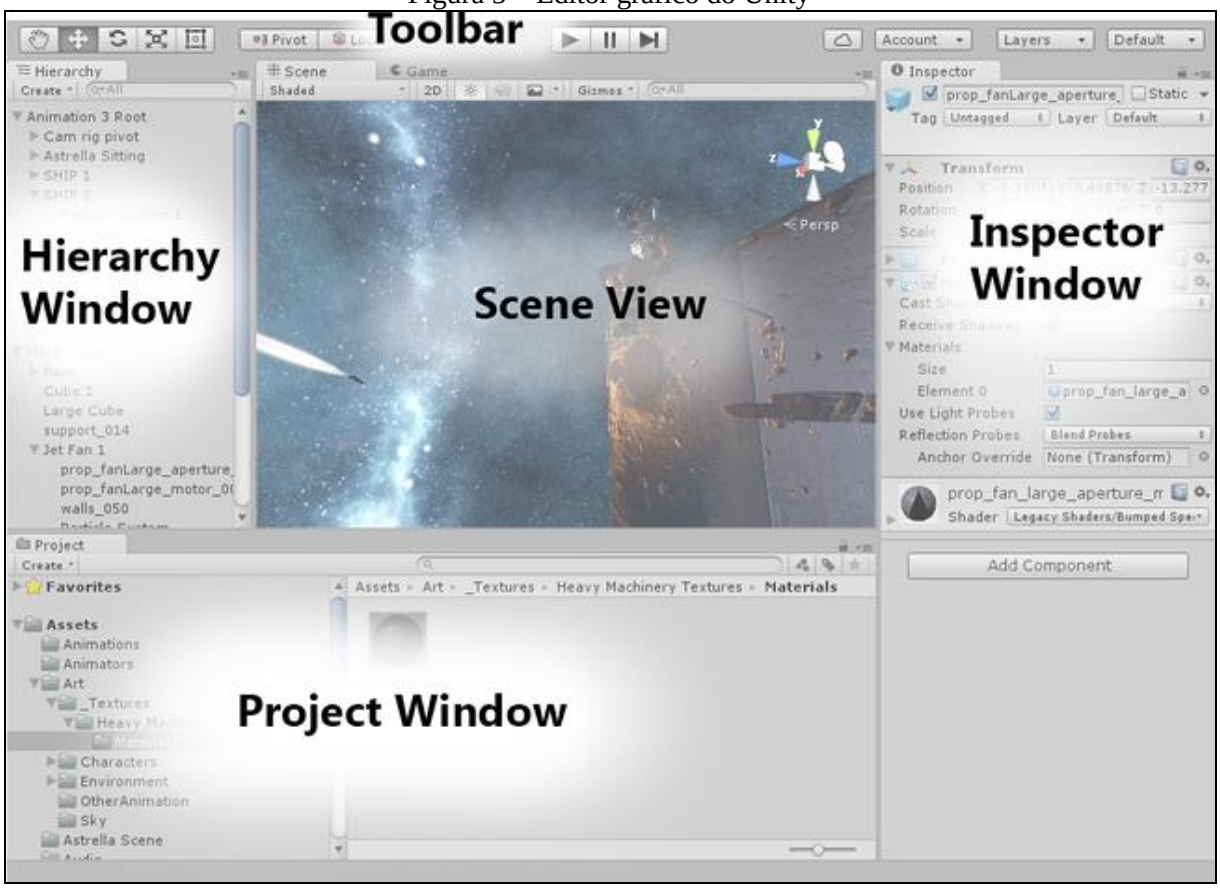
O Unity é baseado em uma arquitetura em que um objeto de jogo é especificado a partir da composição de várias funcionalidades que podem ser adicionadas ou removidas. Cada funcionalidade é implementada por um componente denominado `Game Object` e funciona como um repositório de funcionalidades, ou mais especificamente, componentes.

Os componentes ligados a um `Game Object` definem o seu comportamento de uma maneira flexível. Porém, com o passar do desenvolvimento de um projeto, somente os comportamentos definidos por componentes não são o suficiente para o controle total da aplicação. Para suprir esta necessidade, é possível criar componentes próprios através de *scripts* e adicioná-los aos `Game Objects`. Os *scripts* adicionam eventos ao jogo, modificam propriedades de componentes ao longo da execução de uma aplicação e tratam entradas do usuário, conforme desejado (UNITY, 2016b).

2.4.2 Editor gráfico

O Unity possui um editor gráfico para a construção de jogos e sua visualização em tempo real. O editor conta com várias janelas para realizar a construção e organização do jogo. Na Figura 5 é apresentado o editor gráfico do Unity e suas janelas.

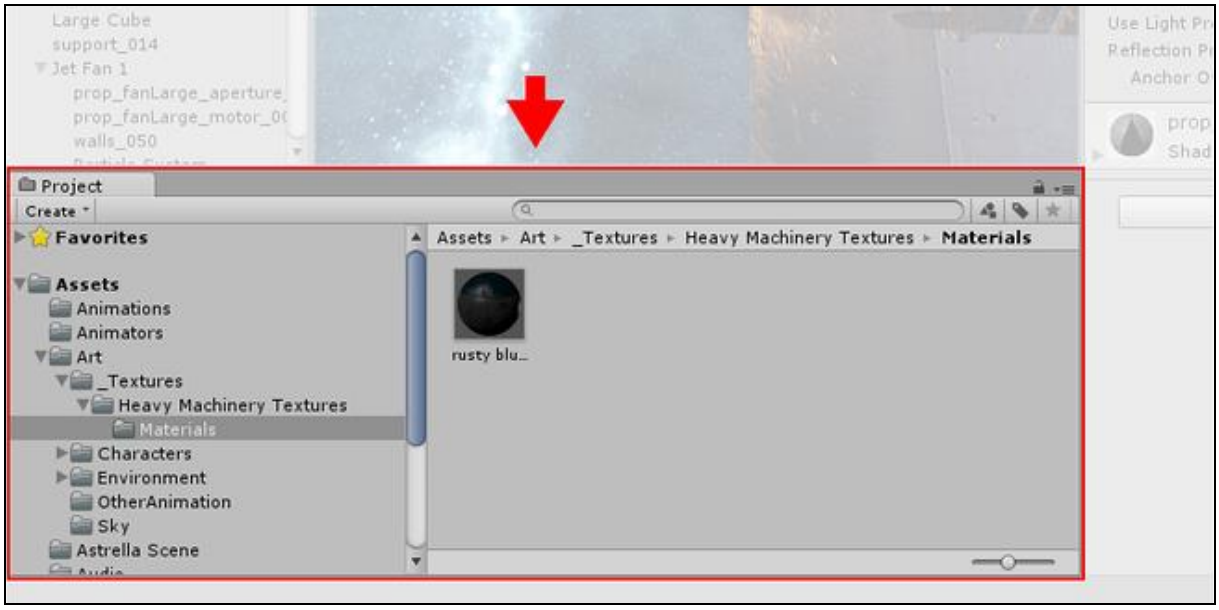
Figura 5 – Editor gráfico do Unity



Fonte: Unity(2016)

A janela *Project* é a interface de manipulação e organização de vários arquivos (`Assets`) que compõem um projeto, conforme Figura 6. Tais como *scripts*, modelos, texturas, efeitos de áudio, imagens, dentre outros. A estrutura exibida na janela *Project* é correspondente a sub-pasta `Assets` dentro da pasta do projeto no sistema de arquivos do computador (UNITY, 2016a).

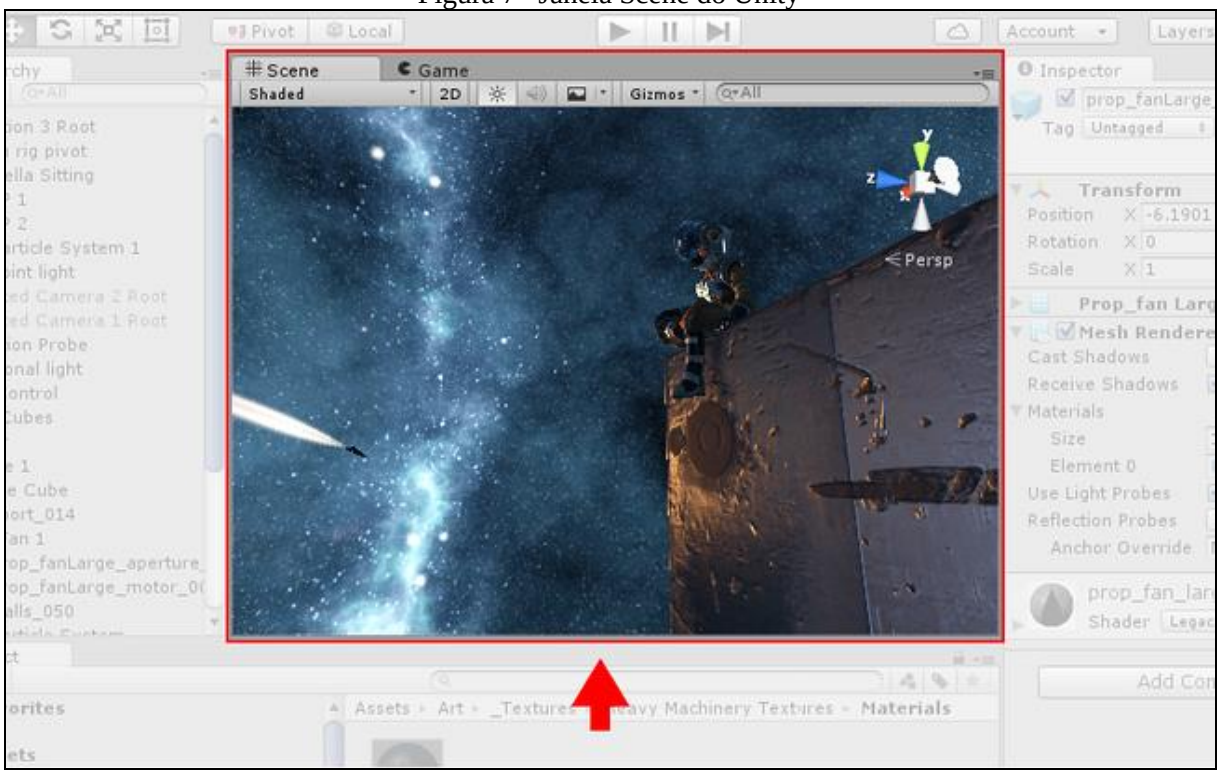
Figura 6– Janela Project do Unity



Fonte: Unity(2016a)

A janela *Scene* é a principal forma de manipulação de elementos visuais no editor de cenas do *Unity*, possibilitando a orientação e posicionamento desses elementos com um *feedback* imediato das alterações realizadas, conforme exibe a Figura 7.

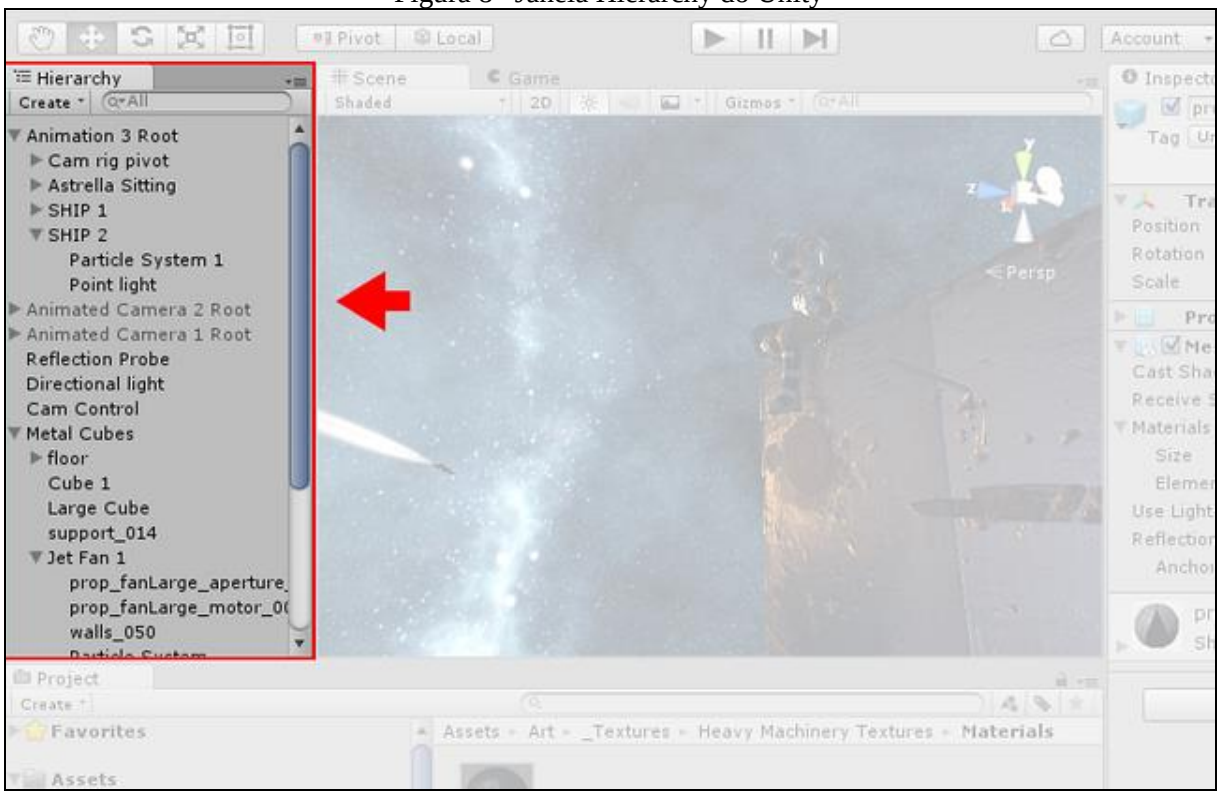
Figura 7– Janela Scene do Unity



Fonte: Unity(2016)

Como pode ser observado na Figura 8, a janela *Hierarchy* é uma representação de texto hierárquica de cada objeto na cena que está sendo editada. Cada item na cena é exibido na hierarquia e revela a estrutura da forma como os objetos estão ligados um a outro.

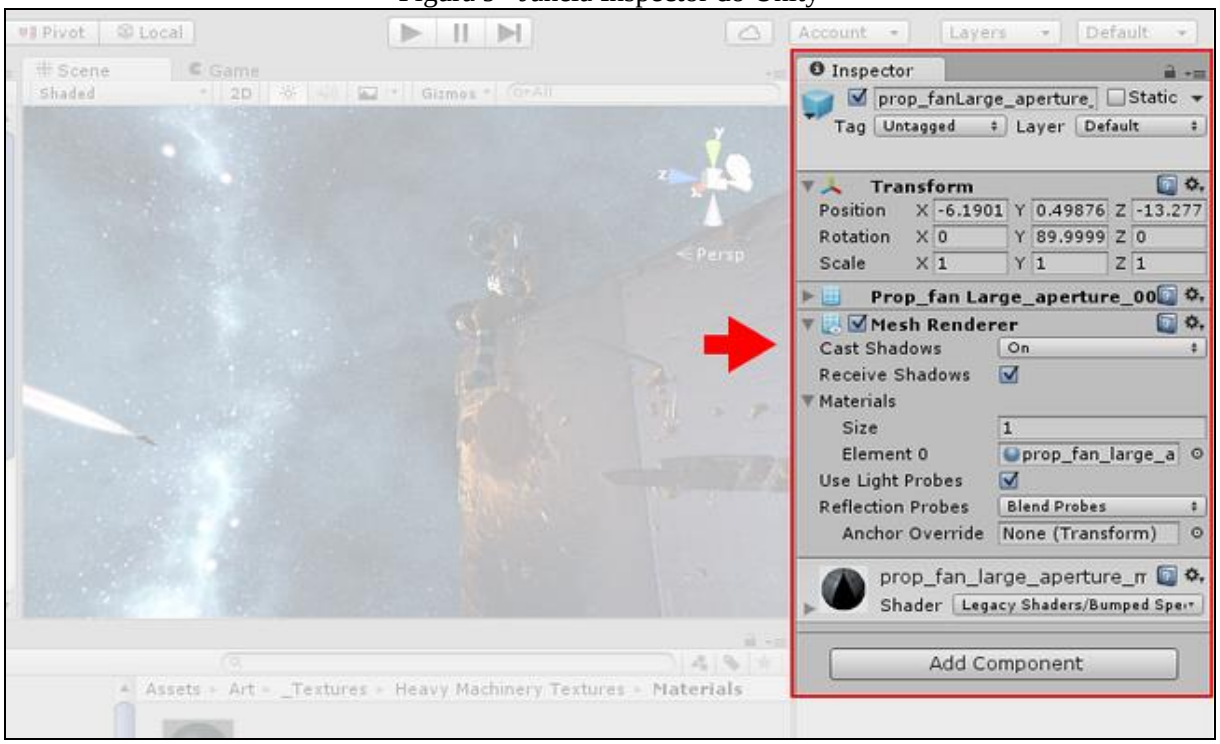
Figura 8– Janela Hierarchy do Unity



Fonte: Unity(2016)

Na janela *Inspector*, tem-se acesso aos vários parâmetros de um objeto presente no cenário, bem como aos seus componentes e atributos. Cada tipo de objeto tem diferentes conjuntos de propriedades e o *layout* e conteúdo da janela pode mudar, conforme Figura 9.

Figura 9– Janela Inspector do Unity



Fonte: Unity(2016)

A barra de ferramentas *Toolbar*, fornece acesso aos recursos de trabalho mais essenciais. À esquerda que contém as ferramentas básicas para manipular a vista de cena e os objetos dentro dela. No centro estão os controles de *play*, *pause* e passar para próximo passo. Os botões à direita dará acesso a sua *Unity Cloud Services* e sua Conta *Unity*, seguido de um menu de visibilidade da camada e o menu *layout* do editor (que fornece alguns *layouts* alternativos para as janelas do editor). A barra de ferramentas não é uma janela, e é a única parte da interface *Unity* que não pode-se reorganizar, conforme Figura 10.

Figura 10– Janela Inspector do Unity

Fonte: Unity(2016a)

2.5 VUFORIA

Vuforia é uma plataforma desenvolvida pela empresa Qualcomm Technologies para o desenvolvimento de aplicativos com Realidade Aumentada (RA) baseada no reconhecimento de imagens, imagens definidas pelo usuário, cilindros, texto, caixas e de objetos (QUALCOMM, 2015). Para a utilização do Vuforia há uma licença denominada Starter, onde todos os recursos estão disponíveis e sem custo. Já a licença Cloud permite que o desenvolvedor hospede e gerencie a aplicação em nuvem, sendo utilizado principalmente em aplicações que precisam de escalabilidade utilizando mais de um milhão de *targets* e flexibilidade de integração com outros sistemas. A seguir serão descritos alguns tipos de reconhecimento realizado pelo Vuforia.

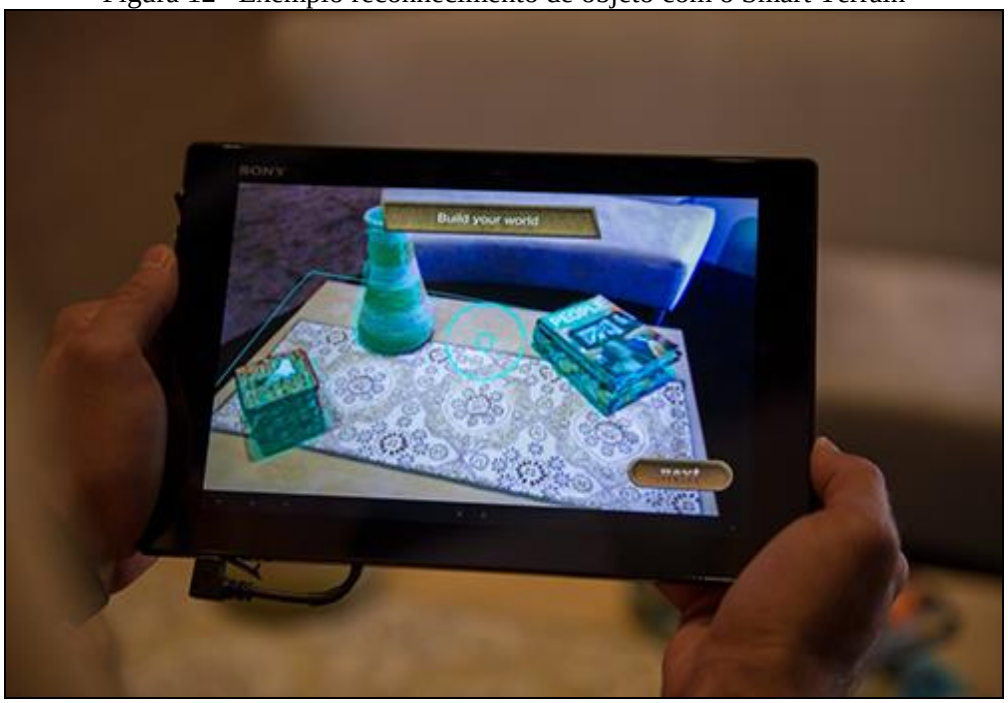
O reconhecimento de imagens acontece através de um marcador onde a grande vantagem consiste no fato de poder usar uma imagem ou fotografia que faça parte do conteúdo físico no qual se pretende combinar os modelos 3D virtuais, conforme Figura 11. Já o reconhecimento de imagens definidas pelo usuário permite que possa ser escolhida uma imagem em tempo de execução, permitindo a escolha de uma cena rica em detalhes, por exemplo, uma rua ou grupo de pessoas.

Figura 11– Target de reconhecimento de imagem

Fonte: Qualcomm (2015).

Diferentemente do reconhecimento de imagens, o reconhecimento de objetos funciona com a realização de um *scan* do objeto real independente do ponto de vista. São calculadas a altura, a largura, a profundidade desse objeto e, em seguida, permitir uma interação com os objetos (Figura 12). Já o reconhecimento de texto permite ao aplicativo a leitura de palavras, fazendo com que seja criada uma nova geração de experiências educacionais em RA (Figura 13).

Figura 12– Exemplo reconhecimento de objeto com o Smart Terrain



Fonte: Qualcomm (2015).

Figura 13– Exemplo de reconhecimento de texto

Fonte: Gizmodo (2013).

O reconhecimento de cilindros permite detectar e rastrear imagens em formato cônico e cilíndrico, suportando também a detecção de imagens sobre as faces planas do topo e do fundo do cilindro (Figura 14). Suportam o reconhecimento dos mesmos formatos de imagem usados pelo reconhecimento de imagens tendo tamanho inferior a 2 *MB*.

Figura 14– Exemplo reconhecimento de cilindro

Fonte: Qualcomm (2015).

2.6 TRABALHOS CORRELATOS

A seguir estão relacionados três trabalhos correlatos. O item 2.6.1 detalha o OsmoPlay, responsável por apoiar o desenvolvimento jogos que utilizam recursos para uma maior interação com o usuário. O jogo Forma Palavras descrito no item 2.6.2 ensina de forma dinâmica a sequência de letras para construir uma palavra. Por fim, o item 2.6.3 descreve o jogo Monta Palavras, que diferente do jogo anterior às palavras estão separadas por sílabas.

2.6.1 OsmoPlay

A empresa OsmoPlay desenvolveu um acessório exclusivo para dispositivos iPad da plataforma iOS que possibilita a interação do usuário com os jogos. O acessório, denominado OSMO, é um refletor que encaixa na câmera do dispositivo, com o objetivo de refletir a imagem posicionada na frente do dispositivo fazendo com que o aplicativo interprete a informação de acordo com a necessidade (PLAYOSMO, 2015). Em seguida será descrito alguns jogos da OsmoPlay.

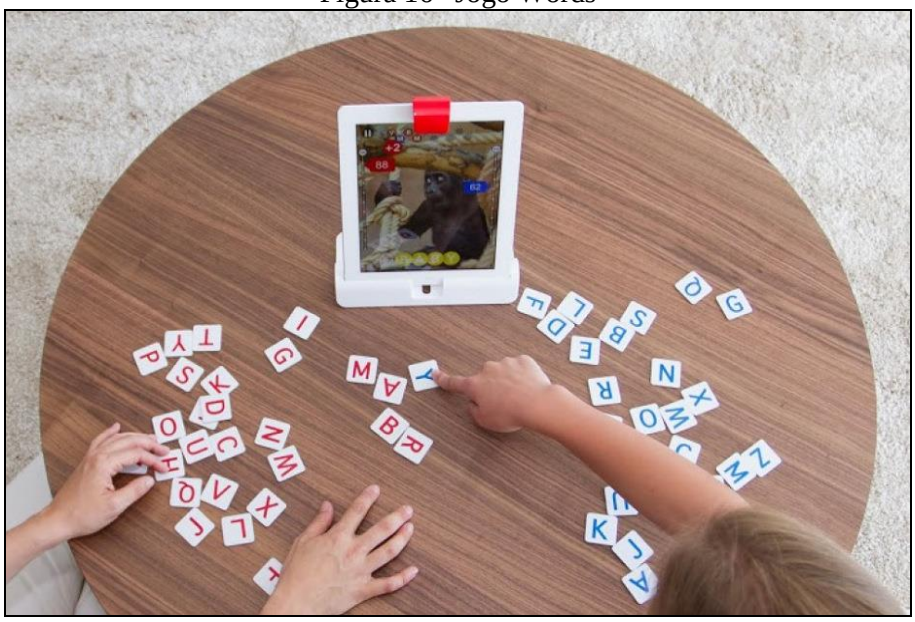
O primeiro jogo, Masterpiece, transforma o iPad em uma ferramenta de desenho. É possível tirar uma foto de qualquer coisa, um brinquedo, pessoa, objeto, ou até mesmo buscar na internet, e em seguida o jogo transforma a imagem em linhas simples para que possa ser fácil e divertido de desenhar (Figura 15).

Figura 15– Jogo Masterpiece

Fonte: Coolmomtech (2015).

O segundo jogo, Words, propõe que seja adivinhada a imagem exibida na tela posicionando letras até que a palavra esteja completa. É necessário posicionar as letras, possibilitando o arranjo aleatório em frente ao iPad. Ao finalizar a palavra, é adquirinda uma pontuação de acordo com a palavra acertada (Figura 16).

Figura 16– Jogo Words



Fonte: Techlive (2015).

O terceiro jogo, o Tangram possui as mesmas regras do jogo tradicional. O objetivo é posicionar as peças sem sobreposição, de maneira que forme uma figura em frente ao iPad (Figura 17). Esse jogo possibilita um *feedback* visual e sonoro em tempo de execução e contém vários níveis de dificuldade (PLAYOSMO, 2015, tradução nossa).

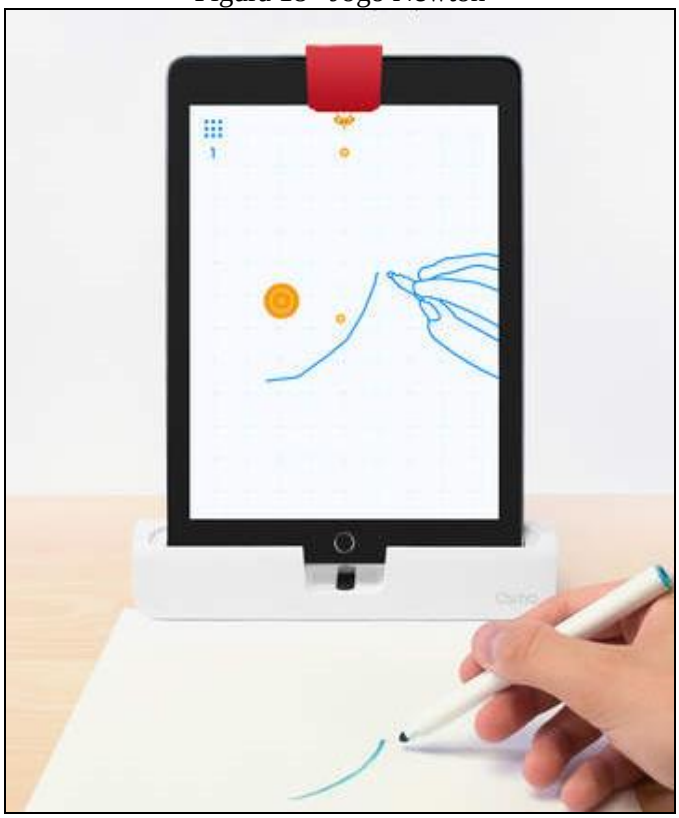
Figura 17– Jogo Tangram

Fonte: Techlive (2015).

Por fim, o jogo Newton que permite desenhar em uma folha de papel e interagir com as bolas que vão caindo até que o alvo seja atingido. Qualquer objeto colocado em frente à

câmera do iPad pode se tornar uma estrutura dentro do jogo, como as peças do jogo Tangram ou brinquedos (Figura 18).

Figura 18– Jogo Newton



Fonte: Itunes (2015).

2.6.2 Forma Palavras

Forma palavras é um jogo desenvolvido pela ESCOLA GAMES, com o objetivo de aprimorar a leitura e a escrita. O jogo consiste em organizar as lâmpadas até formar a palavra indicadas pelo desenho (GAMES, 2015).

Na tela inicial do jogo é possível selecionar duas opções de jogo. Na primeira opção, denominada Alfabeto, são exibidas as letras do alfabeto em ordem crescente. No momento que a letra é exibida, o som da mesma é emitido, seguida de um exemplo de um palavra que inicia com a letra que está sendo exibida. Como por exemplo, C de casa. A segunda opção do jogo denominada Palavras, inicialmente é exibida as regras de como jogar. Basicamente é necessário arrastar as letras da roleta para as portas e em seguida clicar no botão conferir. Esta opção é dividida em cinco fases, onde a dificuldade quase não é alterada, a cada palavra exibida é uma nova fase. As palavras variam e não seguem um padrão. Como por exemplo, a seguinte sequência de palavras: uva, iate, gorila (Figura 19), iglu e foguete (GAMES,2015).

O jogo está disponível gratuitamente, somente para aparelhos com a plataforma Android 2.3 ou superior.

Figura 19– Jogo Forma Palavras

Fonte: Games (2015).

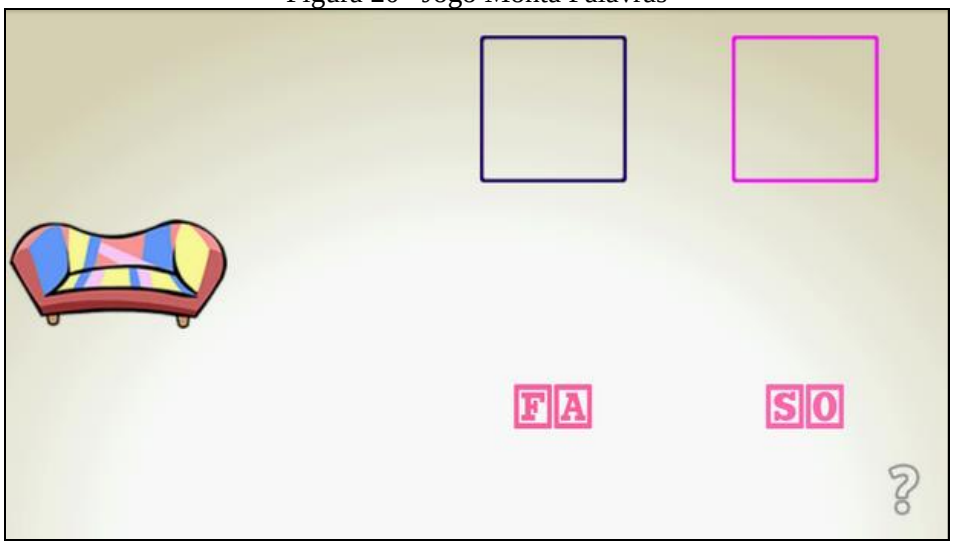
2.6.3 Monta Palavras

O jogo Monta Palavras foi desenvolvido pela Game for Kids. A empresa foi criada através de um projeto de pesquisa na Universidade Federal de Santa Catarina, UFSC – Araranguá, com a missão de integrar novas tecnologias com a educação (KIDS, 2015).

O jogo oferece a um desafio lógico onde deve construir as palavras com base na imagem que é exibida na tela. Em seguida é necessário identificar as sílabas e colocar na ordem correta, tendo assim o intuito de desenvolver a expressão e exploração individualizada, permitindo aprender brincando (KIDS, 2015).

Inicialmente é necessário selecionar um dos níveis diferentes de dificuldade, sendo eles Fácil com palavras dissílabas, Normal com palavras trissílabas e nível Difícil com palavras polissílabas. Na próxima tela é exibida a imagem referente a palavra e suas respectivas silabas que sempre são exibidas fora de ordem (Figura 20). Neste momento as sílabas devem se arrastadas para os quadros posicionados na parte superior da tela de forma correta e sequencial. Caso ainda haja dificuldade durante a montagem há um botão onde é possível visualizar a palavra que deve ser montada. Após montar a palavra uma nova palavra surge para ser montada e assim sucessivamente. O jogo está disponível gratuitamente, somente para aparelhos com a plataforma Android 2.3 ou superior.

Figura 20– Jogo Monta Palavras



Fonte: Kids (2015).

3 DESENVOLVIMENTO

Neste capítulo são apresentadas as etapas de desenvolvimento do aplicativo proposto. Na Seção 3.1 são apresentados os principais requisitos, e a Seção 3.2 apresenta a especificação. A Seção 3.3 apresenta de forma detalhada a implementação do aplicativo e, por fim, a Seção 3.4 apresenta os resultados obtidos.

3.1 REQUISITOS

O aplicativo deverá atender a os seguintes requisitos:

a) possuir uma tela inicial para seleção de opções do modo de busca de palavras (Requisito Funcional - RF);
b) disponibilizar uma opção de jogo que contenha imagens em cada nível de dificuldade (RF);
c) disponibilizar uma forma de adicionar imagens dinamicamente ao aplicativo (RF);
d) disponibilizar uma opção de jogo para poder visualizar e imprimir o marcador (RF);
e) disponibilizar uma opção de ajuda contendo informações que auxiliam o usuário antes de iniciar o jogo (RF);
f) disponibilizar diferentes níveis de dificuldade (RF);
g) disponibilizar uma opção de parar o jogo durante o reconhecimento de texto (RF);
h) ser disponibilizado para a plataforma Android (RNF);
i) utilizar o *Software Development Kit* (SDK) Vuforia para a implementação dos recursos de Realidade Aumenta (Requisito Não Funcional - RNF);
j) utilizar a linguagem C# para definir os *scripts* do aplicativo (RNF);
k) utilizar o motor de jogos Unity 3D para a implementação da cena gráfica 3D (RNF).

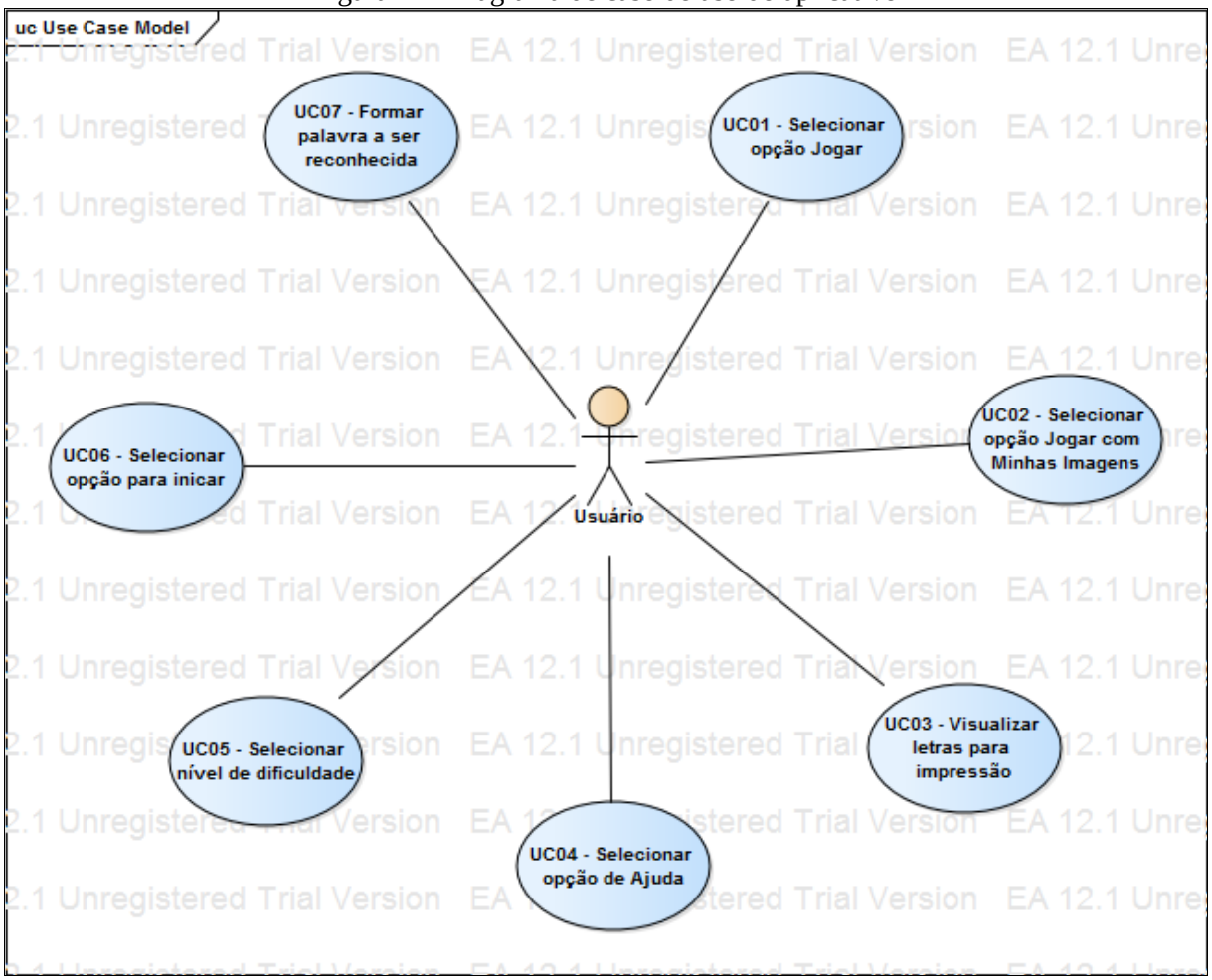
3.2 ESPECIFICAÇÃO

A especificação do aplicativo foi desenvolvida seguindo a análise orientada a objetos, utilizando a Unified Modeling Language (UML) em conjunto com a ferramenta Enterprise Architect na versão 12.1.1229 para a elaboração dos casos de uso, do diagrama de classes e dos diagramas de atividades.

3.2.1 Diagrama de Casos de uso do aplicativo

Nesta seção são apresentados os casos de uso do aplicativo, ilustrados na Figura 21. Identificou-se o ator `Usuário` que utiliza o aplicativo. Do quadro 1 até o quadro 7 são descritos os casos de uso do ator `Usuário`.

Figura 21– Diagrama de caso de uso do aplicativo



Quadro 1– Caso de uso uc01 – Selecionar opção Jogar

Descrição	Este caso de uso tem por objetivo escolher a opção de jogar com as imagens disponibilizadas pelo jogo.
Pré-Condição	Estar com o aplicativo aberto.
Cenário Principal	1. O `usuário` irá tocar no botão `Jogar`.
Pós-Condição	O `usuário` poderá escolher o nível de dificuldade.

Quadro 2– Caso de uso uc02 – Selecionar opção Jogar com Minhas Imagens

Descrição	Este caso de uso tem por objetivo escolher a opção de jogar com as imagens do seu dispositivo.
Pré-Condição	Estar com o aplicativo aberto.
Cenário Principal	1. O `usuário` irá tocar no botão `Jogar com Minhas Imagens`.
Pós-Condição	O `usuário` poderá escolher o nível de dificuldade.

Quadro 3 – Caso de uso uc03 – Visualizar letras para impressão

Descrição	Este caso de uso tem por objetivo disponibilizar as letras para impressão para formar as palavras que serão reconhecidas.
Pré-Condição	Estar com o aplicativo aberto.
Cenário Principal	1. O `usuário` irá tocar no botão `imprimir letras;` 2. O aplicativo irá abrir o navegador padrão do dispositivo do `usuário` no *link* www.inf.furb.br/gcg/tecedu/palavras que contém as letras para a impressão.
Pós-Condição	O usuário poderá imprimir as letras utilizadas para o reconhecimento de texto.

Quadro 4– Caso de uso uc04 – Selecionar a opção Ajuda

Descrição	Este caso de uso tem por objetivo disponibilizar conteúdo com informações sobre como escolher o nível de dificuldade e reconhecimento de texto.
Pré-Condição	Estar com o aplicativo aberto e executar `uc01 ou uc02`.
Cenário Principal	1. O `usuário` irá tocar no botão `Ajuda;` 2. O aplicativo irá exibir as informações de como escolher o nível de dificuldade. 3. O aplicativo irá habilitar os botões `Próximo` e `Sair Ajuda`. 4. O `usuário` irá tocar no botão `Próximo;` 5. O aplicativo irá exibir as instruções para o reconhecimento de texto. 6. O `usuário` irá tocar no botão `Sair Ajuda;` 7. O aplicativo volta a exibir a tela de níveis de dificuldade.
Pós-Condição	O `usuário` poderá visualizar instruções de como funciona o reconhecimento de texto.

Quadro 5– Caso de uso uc05 – Selecionar nível de dificuldade

Descrição	Este caso de uso tem por objetivo selecionar o nível de dificuldade do aplicativo.
Pré-Condição	Estar com o aplicativo aberto e executar o `uc02`.
Cenário Principal	1. O `usuário` irá tocar em um dos botões dos níveis disponíveis. Sendo eles `Fácil, Médio ou Difícil`. 2. O aplicativo irá habilitar o botão de play para iniciar o jogo.
Pós-Condição	O aplicativo irá carregar somente as imagens pertencentes ao nível de dificuldade selecionado

Quadro 6– Caso de uso uc06 – Selecionar opção para iniciar

Descrição	Este caso de uso tem por objetivo iniciar o jogo
Pré-Condição	Estar com o aplicativo aberto e executar o `uc06`
Cenário Principal	1. O `usuário` irá tocar no botão `play;` 2. O aplicativo irá abrir o a tela de reconhecimento de texto;
Pós-Condição	O `usuário` irá acessar com seu usuário e senha, em seguida poderá cadastrar novas imagens.

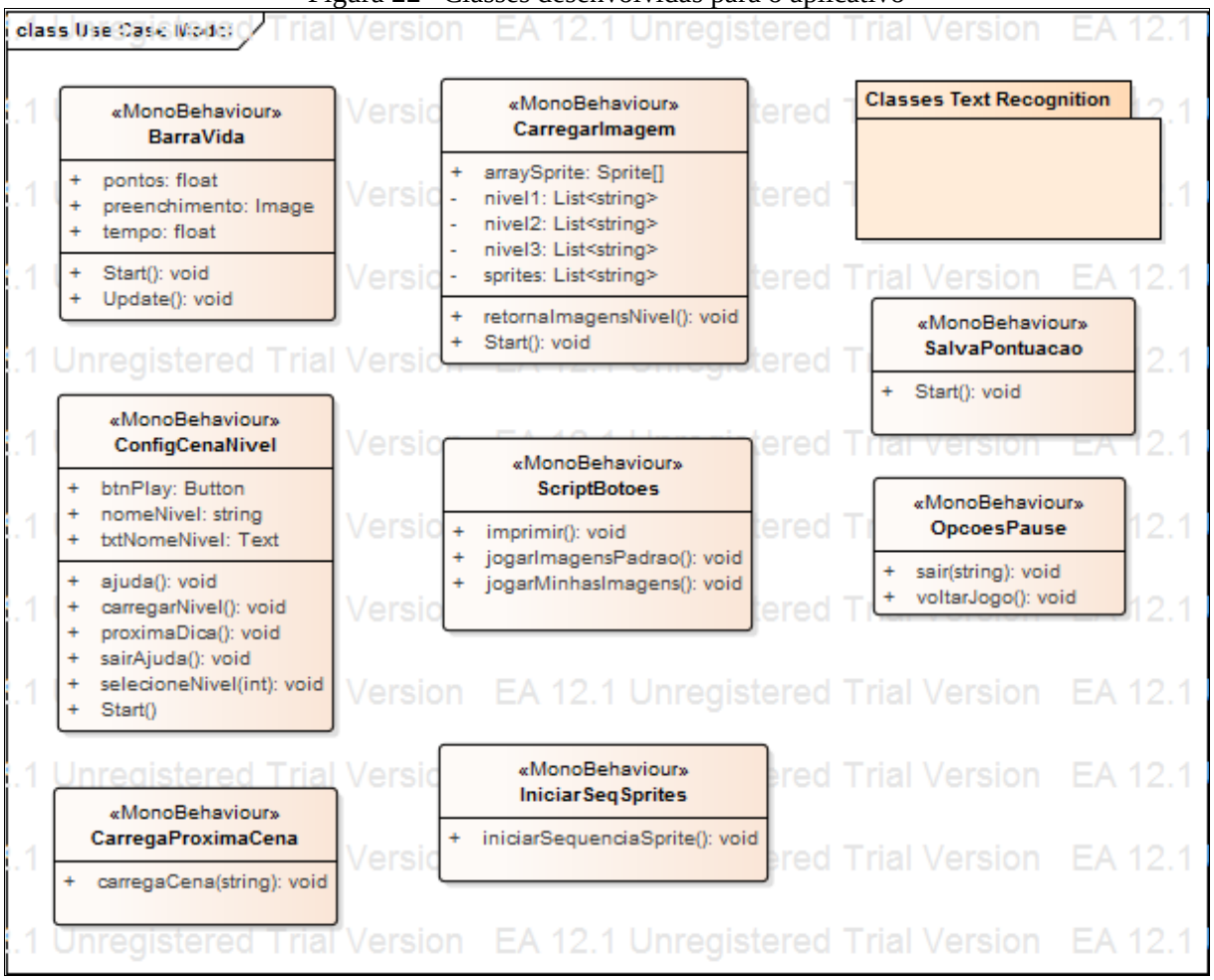
Quadro 7– Caso de uso uc07 – Formar palavra a ser reconhecida

Descrição	Este caso de uso tem por objetivo formar a palavra a ser reconhecida.
Pré-Condição	1. Estar com o aplicativo aberto e executar o `uc06`.
Cenário Principal	1. O aplicativo irá exibir uma imagem; 2. O `usuário` deve formar uma palavra com as letras que corresponde a imagem; 3. O aplicativo irá diminuir a barra de vida enquanto a palavra não é reconhecida; 4. O `usuário` irá apontar o dispositivo para palavra formada na região específica na tela;
Fluxo Alternativo 1	A qualquer momento o `usuário` poderá retornar a tela inicial, a fim de selecionar novamente o casos de uso `uc06`.
Exceção 1	Caso o a barra de vida terminar e o `usuário` ainda não informou a palavra correta, o aplicativo irá abrir a tela contendo a nota do `usuário`.
Pós-Condição	O aplicativo irá reconhecer a palavra e será exibida a tela contendo a nota do `usuário`.

3.2.2 Diagrama de classes do aplicativo

O diagrama de classes da Figura 22 apresenta as classes desenvolvidas para a criação do aplicativo.

Figura 22– Classes desenvolvidas para o aplicativo



A classe `ScriptBotoes` disponibiliza os métodos executados ao tocar nos botões do aplicativo na tela principal. Cada um dos botões da tela principal possui um evento de toque associado a um método desta classe.

A classe `ConfigCenaNivel` implementa métodos a relacionados a cena de níveis e ações específicas, tais como dar um retorno ao usuário de acordo com cada nível escolhido e apresenta opções para visualizar imagens que auxiliam no entendimento do jogo.

A classe `CarregaProximaCena` disponibiliza um método que permite a mudança de `Cena` durante a execução do aplicativo. Sempre que uma `Cena` precisa ser chamada no aplicativo e não necessita de um comportamento específico este método é chamado.

A classe `CarregarImagem` é responsável por exibir a imagem da palavra a ser reconhecida. A imagem pode ser carregada diretamente do dispositivo móvel ou da pasta `Resources`.

A pasta `Resources` permite armazenar objetos de acesso, incluindo `assets`. Todos os `assets` que estão dentro desta pasta são carregados sob demanda a partir de um *script*. Um

`asset` é qualquer arquivo que pode ser útil no aplicativo, como por exemplo, animações, arquivos de áudio ou imagens.

A classe `BarraVida` é responsável pelo preenchimento de uma imagem correspondente ao tempo restante para que a palavra seja reconhecida.

A classe `OpcoesPause` é responsável por tratar o comportamento do duplo clique na cena de reconhecimento de texto, fazendo com que a aplicação fique em estado parado.

A classe `SalvaPontuacao` é responsável por salvar a pontuação recebida após o reconhecimento da palavra.

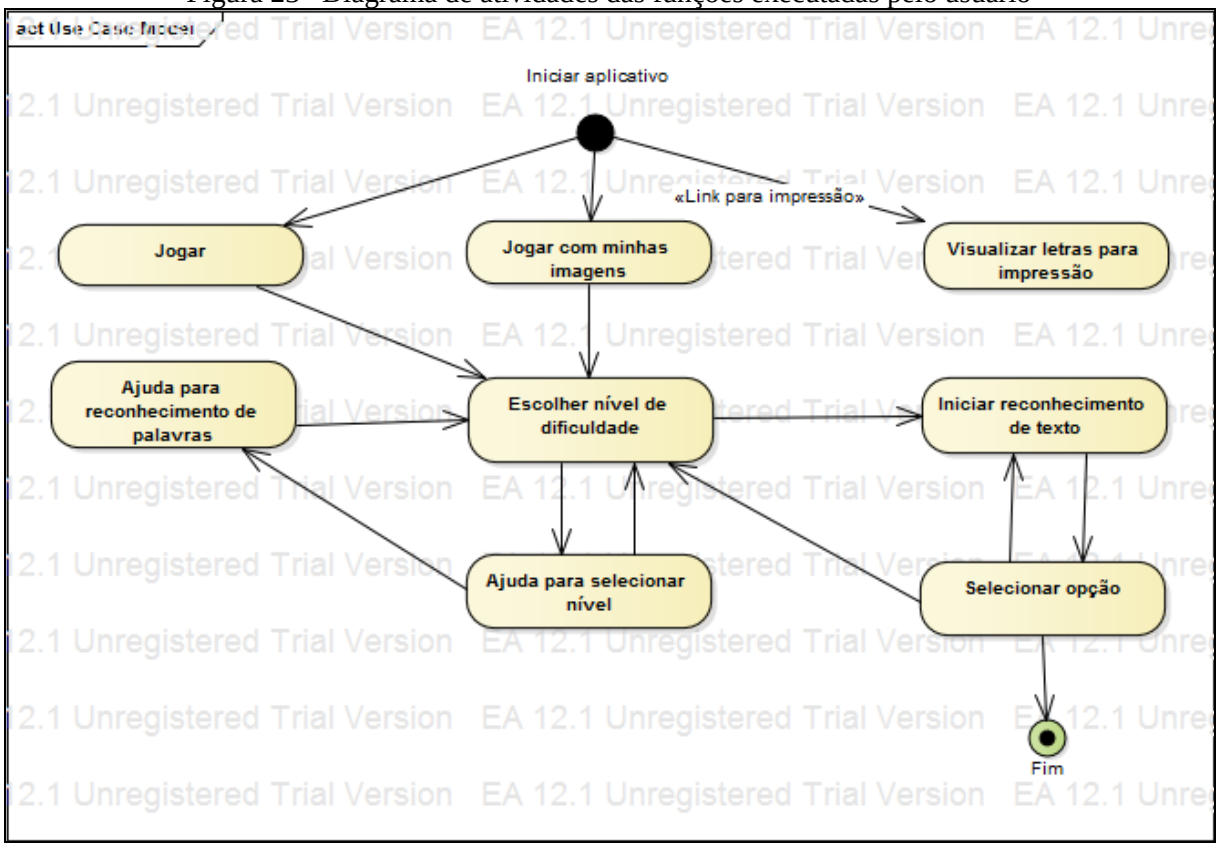
A classe `IniciarSeqSprites` contém um método que é responsável por atribuir o valor da variável `SequenciaSprites`. Esta variável armazena a sequência da imagem a ser apresentada no aplicativo.

O pacote `Classes Text Recognition` contém as classes padrão do Vuforia utilizadas no exemplo de reconhecimento de texto.

3.2.3 Diagrama de atividades

A Figura 23 apresenta o diagrama de atividades das funções executadas pelo usuário do aplicativo. Suas atividades incluem, selecionar o modo de busca das imagens, escolher o nível de dificuldade, habilitar opções de ajuda, iniciar o reconhecimento de texto e escolher se deseja jogar novamente ou alterar o nível de dificuldade.

Figura 23– Diagrama de atividades das funções executadas pelo usuário



3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

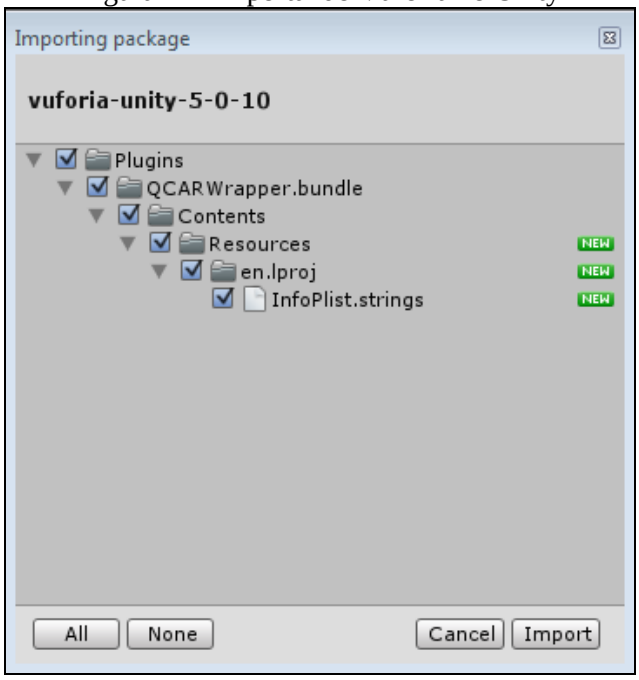
3.3.1 Técnicas e ferramentas utilizadas

O aplicativo foi desenvolvido no *framework* Unity 3D na versão 5.2.2 f1 Personal em conjunto com a IDE MonoDevelop 4.0.1, utilizando a linguagem de programação C Sharp. Para a criação da RA foi utilizado o SDK Vuforia na versão 5.0.10 integrado ao Unity. Para a execução e a realização de testes de todo o desenvolvimento do trabalho, utilizou-se um Celular LG G3 com sistema operacional Android 4.4.2, 16 GB de memória RAM e processador Quad-Core de 2.5 GHz.

3.3.1.1 Inicio do projeto

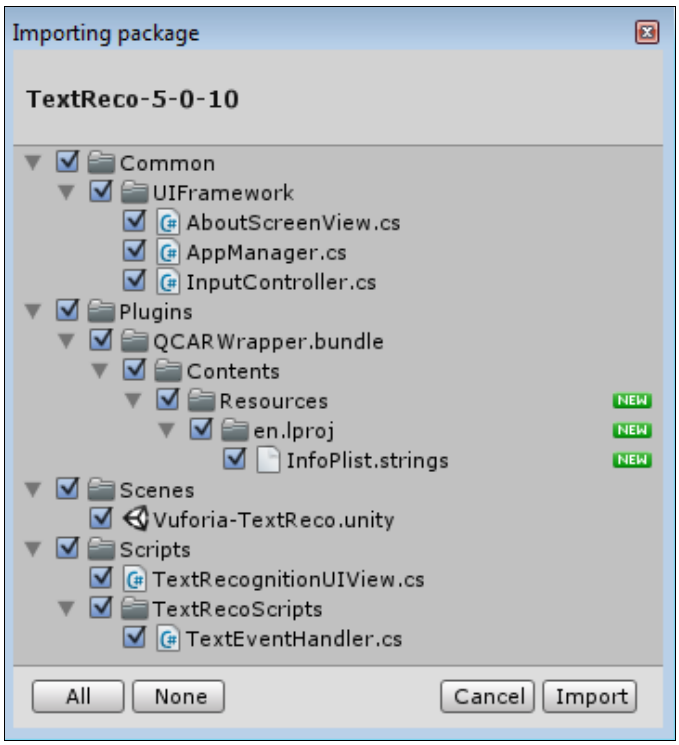
Para dar início ao projeto no Unity, é necessário a importação do SDK do Vuforia para o Unity, disponibilizado na página de desenvolvedores do Vuforia. A importação pode ser realizada clicando duas vezes no SDK do Vuforia. Com isto, é aberta uma janela apresentado todos os arquivos para serem importados (Figura 24).

Figura 24– Importando Vuforia no Unity



Foi realizado a importação do SDK de exemplo de reconhecimento de texto da mesma forma que o SDK Vuforia. Na Figura 25 são exibidos todos os arquivos a serem importados.

Figura 25– Importando TextReco no Unity



Para o desenvolvimento desse aplicativo foi necessário utilizar um arquivo com palavras adicionais, que contém as palavras que o aplicativo vai reconhecer. Este arquivo com a extensão `lst` contem palavras personalizadas nomeado de `AdditionalWords`. Este arquivo pode ser alterado em editores de texto padrão, o que facilita sua personalização.

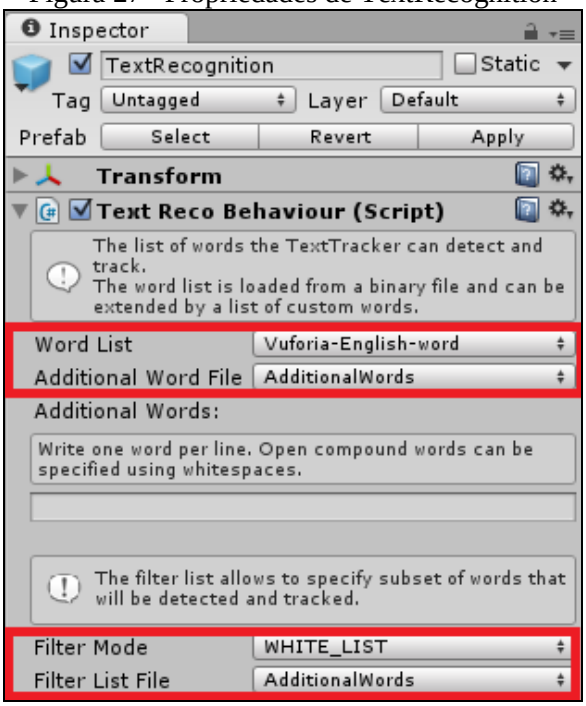
O `AdditionalWords` é utilizado pois o SDK Vuforia contém um arquivo contendo um conjunto de palavras que podem ser utilizadas no reconhecimento de texto, porém as palavras estão na língua inglesa.

É necessário duas informações no cabeçalho do arquivo, sendo elas o tipo de codificação e a palavra `QCAR` conforme a Figura 26. As linhas seguintes do arquivo contém as palavras que o aplicativo pode reconhecer.

Figura 26– Arquivo AdditionalWords

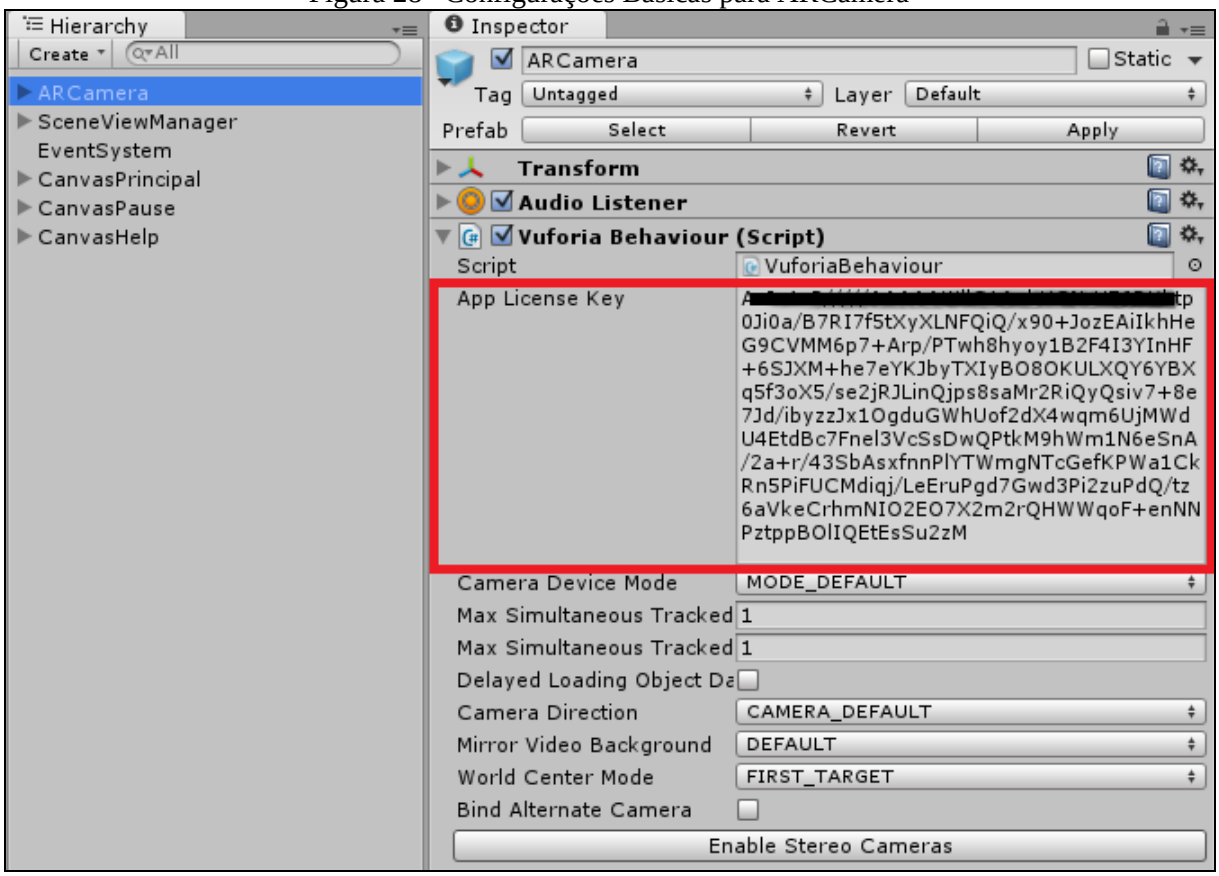
Após a criação do arquivo contendo as palavras adicionais é necessário associar o mesmo ao projeto. O arquivo deve estar na pasta contendo o projeto, seguido da estrutura de pastas "`\Assets\StreamingAssets\QCAR`" e na propriedade `Additional Word File` do Prefab `TextRecognition` é informado o arquivo `AdditionalWords`. Foi alterado o tipo de `Filter Mode` para `WHITE_LIST` e informado `AdditionalWords` no `Filter List Mode`. Isso significa que durante o reconhecimento de texto somente serão reconhecidas palavras que estão contidas neste filtro, conforme a Figura 27.

Figura 27– Propriedades de TextRecognition



Após a importação do Vuforia no Unity, é preciso gerar uma chave de licença para seu uso através na página de desenvolvedores do Vuforia (está chave é adicionada no *Prefab* ARCamera). Na janela *Hierarchy* do Unity é preciso deletar a câmera (nomeada de Main Camera) que vem por padrão em uma `Scene` e adicionar o *Prefab* `ARCamera` (localizado dentro da pasta Vuforia na janela *Project*), pois o Vuforia necessita de um tipo especial de câmera para gerar a RA. Na janela *Inspector* da `ARCamera` é necessário incluir a chave de licença obtida. A Figura 28 apresenta no quadro em vermelho, a configuração básica necessária para realizar na `ARCamera`.

Figura 28– Configurações Básicas para ARCamera



3.3.1.2 Tela inicial

Quando o aplicativo é aberto, a tela inicial apresenta o título da aplicação e três botões de opções. Abaixo são apresentadas as funcionalidades das opções, conforme Figura 29.

Figura 29– Tela inicial do aplicativo

Nesta tela é possível escolher o modo de jogo, ou seja, utilizar as imagens que o aplicativo já contém ou buscar dinamicamente.

O Quadro 8 apresenta os métodos que são executados quando o usuário tocar em algum dos botões da tela inicial.

Quadro 8– Métodos da tela inicial

```
7     public void jogarImagensPadrao(){
8         PlayerPrefs.SetString("jogarMinhasImagens","nao");
9         Application.LoadLevel ("CenaNivel");
10    }
11
12    public void jogarMinhasImagens(){
13        PlayerPrefs.SetString("jogarMinhasImagens","sim");
14        Application.LoadLevel ("CenaNivel");
15    }
16
17    public void imprimir(){
18        Application.OpenURL ("http://www.inf.furb.br/gcg/tecedu/palavras/letras.pdf");
19    }
```

Ao tocar no botão `Jogar` é chamado o método `jogarImagensPadrao`, onde é utilizado o comando `PlayerPrefs` para salvar um valor do tipo `string`. A variável

`JogarMinhasImagens` recebe o valor `nao`, caso tenha tocado neste botão e as imagens utilizadas pelo aplicativo são as disponibilizadas por padrão.

Ao tocar no botão `Jogar com Minhas Imagens`, o valor atribuído a variável `JogarMinhasImagens` é `sim` e as imagem utilizadas para o jogo serão as definidas pelo usuário. Ao selecionar qualquer opção de jogo o aplicativo direciona para a tela de escolha de níveis.

As imagens são definidas pelo usuário quando existem no diretório dispositivo móvel e devem estar dentro da pasta `/storage/emulated/0/download/VISEDU`. O nome da imagem deve ser o mesmo da palavra que deve ser reconhecida. As imagens devem conter em seu nome uma inicial representando o nível de dificuldade que a palavra pertence. Se a palavra pertence ao nível fácil, deve iniciar com `F_`, se a palavra pertence ao nível médio deve iniciar com `M_`, se a palavra pertence ao nível difícil deve iniciar com `D_`, conforme Figura 30.

Ao tocar no botão `Imprimir Letras` é executado o método imprimir, que abre um site para visualizar e imprimir as letras utilizadas no aplicativo, que estão disponíveis no Apêndice A.

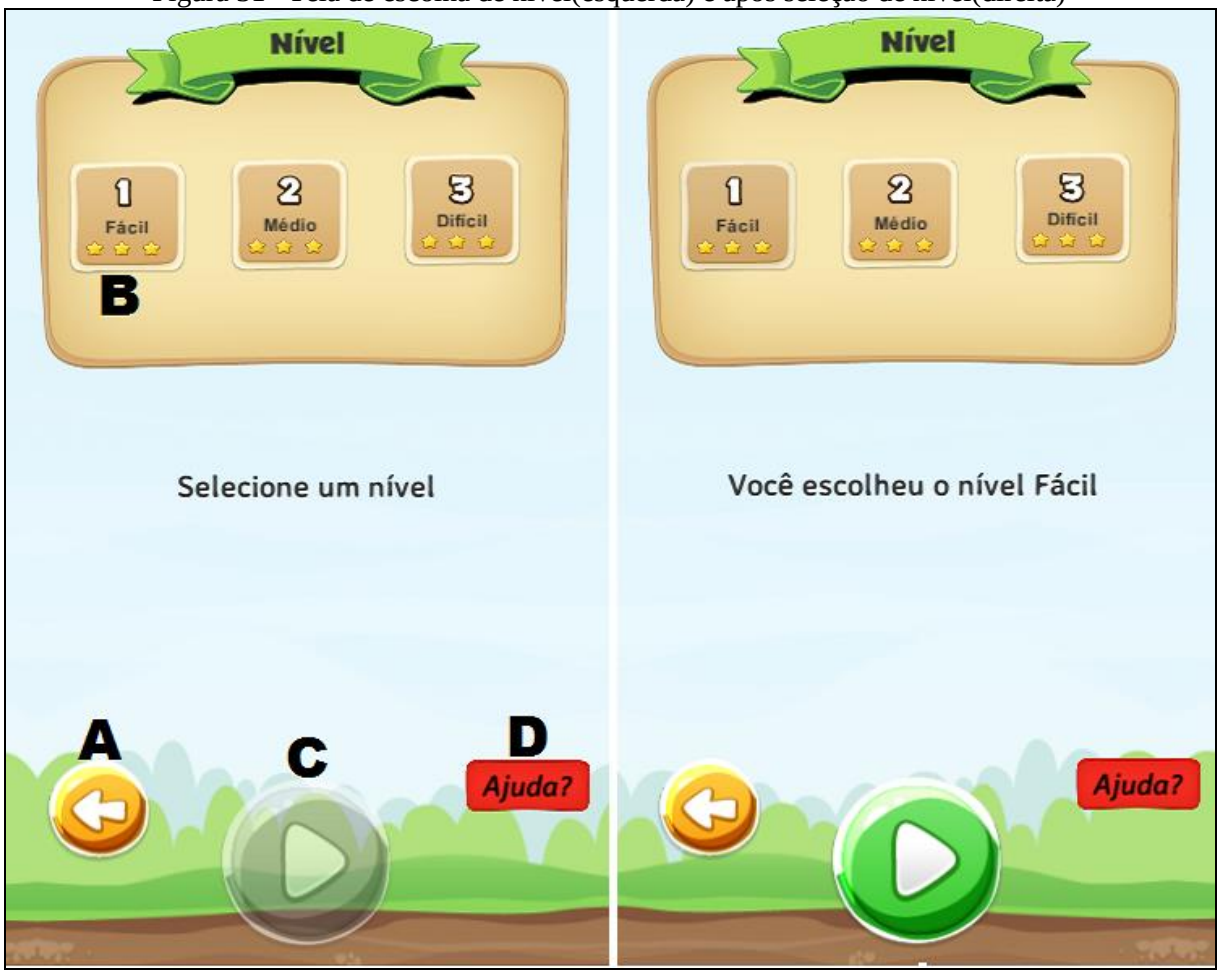
Figura 30– Tela do aplicativo móvel

3.3.1.3 Tela de escolha de nível

Após selecionar o modo de jogo, é necessário escolher o nível de dificuldade. A aplicação contém os níveis fácil, médio e difícil, conforme Figura 31. Ao escolher a opção de jogar com as imagens padrão foi utilizado o tamanho da palavra para definir em qual nível a imagem vai estar. Por exemplo a palavra gato e sua respectiva imagem estão no nível fácil. A palavra peixe, no nível médio e cachorro no nível difícil.

Caso o modo de jogo escolhido seja de jogar com as imagens definidas pelo usuário o mesmo deve definir quais imagens pertencem a cada nível.

Figura 31– Tela de escolha de nível(esquerda) e após seleção de nível(direita)

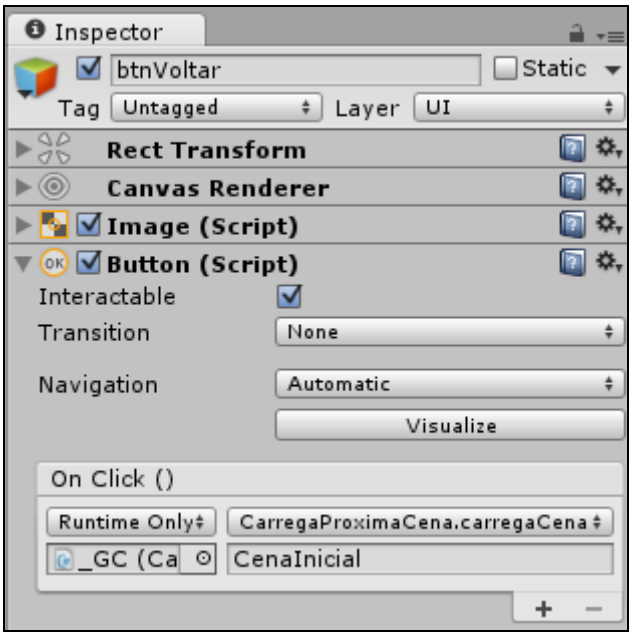


Na tela de seleção de nível há uma mensagem solicitando a escolha de um nível de dificuldade. Existe a opção de retornar a tela principal através do botão voltar, item A na Figura 31. Somente após tocar em um dos botões de níveis, item B, o botão de jogar será habilitado, item C. Ao selecionar um nível é exibida uma mensagem informando qual nível foi escolhido, conforme Figura 31(direita).

Ao tocar no botão `Ajuda?`, item D, são exibidas telas explicando o funcionamento do jogo.

O botão `btnVoltar` contém o *script* `CarregaProximaCena` como pode ser observado na Figura 32.

Figura 32– Propriedades do btnVoltar



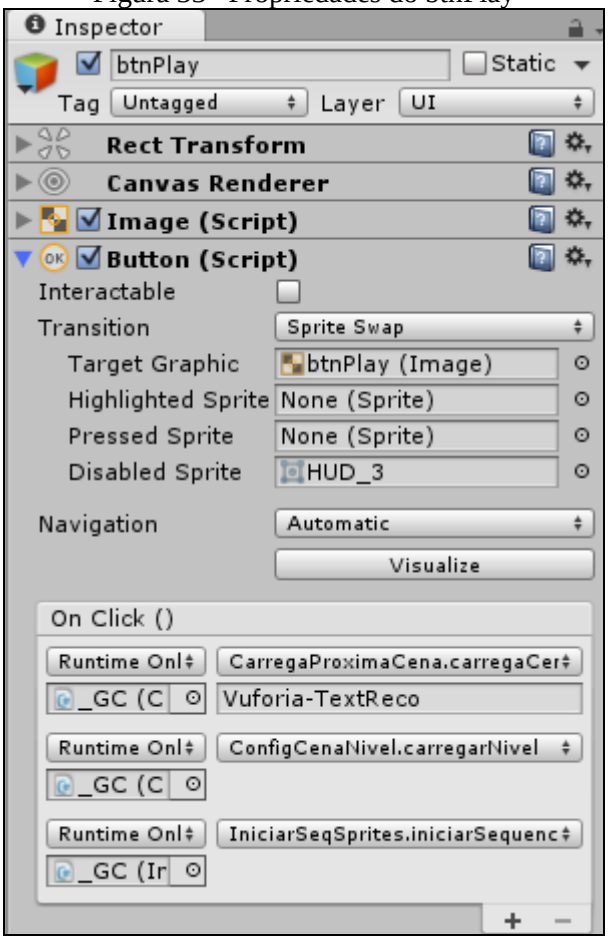
O *script* contém o método `carregaCena` que tem como objetivo carregar uma *scene* que deve ser passada por parâmetro, conforme Quadro 9. No caso do `btnVoltar` é carregado a `CenaInicial`, que retorna para a tela inicial do jogo.

Quadro 9– Método carregaCena da classe CarregaProximaCena

```
6     public void carregaCena(string nomeCena){
7         Application.LoadLevel(nomeCena);
8     }
```

O botão `btnPlay` também utiliza o *script* `carregaCena`, porém ao tocar no botão é carregada a *scene* de reconhecimento de texto chamada `Vuforia-TextReco`. O botão contém ainda os *scripts* `ConfigCenaNivel` e `IniciarSeqSprites`, conforme Figura 33.

Figura 33– Propriedades do btnPlay



O script `ConfigCenaNivel` contém os métodos utilizados na *scene* de escolha de níveis. O método utilizado pelo `btnPlay` é o `carregarNivel` que após ter sido escolhido o nível e dificuldade habilita o componente de texto onde é exibido ao usuário uma mensagem informando que o jogo está sendo carregado, como pode ser visto no Quadro 10.

Quadro 10– Método carregarNivel

```
55      public void carregarNivel(){
56          var texto = GameObject.Find ("txtCarregando").GetComponent<Text> ();
57          texto.enabled = true;
58      }
```

O botão contém ainda o *script* `IniciarSeqSprites`, que através do método `iniciarSequenciaSprite` altera o valor da variável `SpriteEscolhida` para vazio, conforme Quadro 11. Essa variável é responsável por armazenar o nome das *sprites* que já foram escolhidas.

Quadro 11– Método iniciarSequenciaSprite

```
6       public void iniciarSequenciaSprite(){
7           PlayerPrefs.SetString ("SpriteEscolhida","");
8       }
```

Ao tocar no botão `Ajuda?`, na tela de escolha de nível são exibidos balões de texto informando ao usuário o significado de cada item na tela para a escolha do nível. São

habilitados os botões `Próximo` que após tocado exibe informações necessárias para o reconhecimento de texto e o botão `Sair Ajuda` que desabilita as imagens de ajuda, conforme Figura 34.

Figura 34– Opção de ajuda(esquerda) e próxima ajuda(direita)

Após escolher o nível de dificuldade e tocar no botão de jogar é exibido ao usuário uma mensagem informando que deve aguardar enquanto o jogo é carregado, conforme Figura 35.

Figura 35– Mensagem informando que o nível está sendo carregado

3.3.1.4 Tela de reconhecimento de texto

Após selecionar o nível de dificuldade é exibido a tela de reconhecimento de texto. A *scene* para a utilização da RA do aplicativo e contém os itens apresentados na Figura 36.

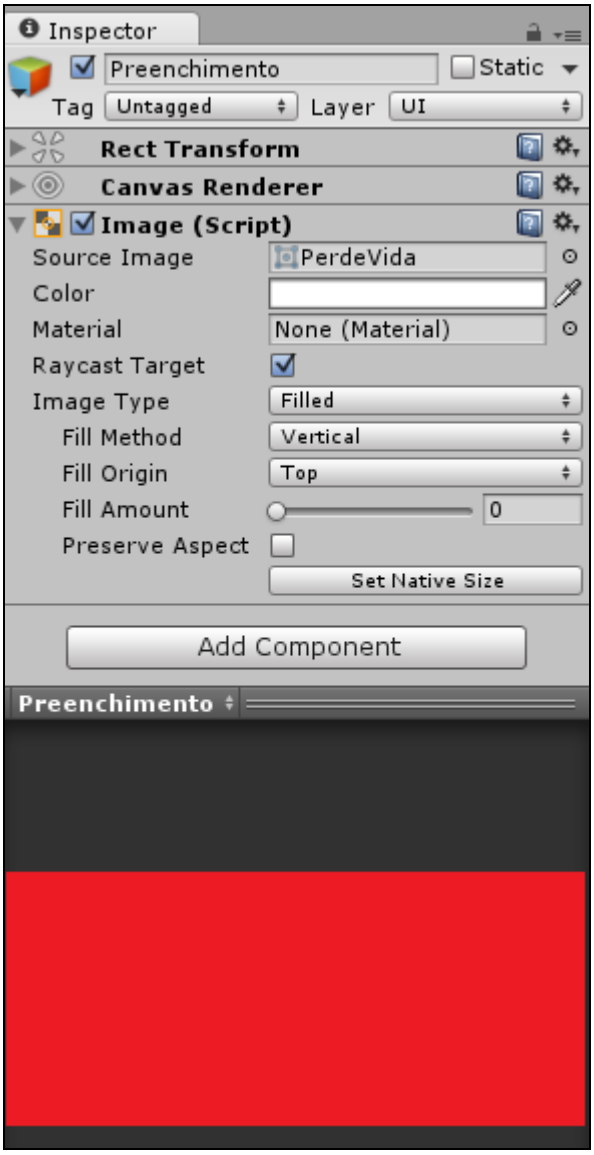
Figura 36– Scene de Reconhecimento de texto

A área demarcada pelo quadro amarelo contém a barra dinâmica que exibe de uma forma gráfica o tempo restante para o reconhecimento da palavra. A área demarcada pelo quadro azul exibe a imagem utilizada como referência para a palavra que deve ser reconhecida. A área demarcada pelo quadro vermelho é a região de interesse onde a palavra deve ser posicionada para ser identificada. Para o controle da barra dinâmica foi utilizado dois componentes `Image` com os nomes `Preenchimento` e `Contorno`.

O componente `Preenchimento` contém uma imagem associada a ele, chamada `PerdeVida`. Essa imagem possui as mesmas dimensões da `Image Contorno`, a diferença é que a sua cor é vermelha. Para que se tenha a visualização de uma barra de vida, onde com decorrer do tempo a vida diminui, foi necessário utilizar algumas propriedades para dar o efeito de animação. A primeira propriedade alterada foi a `Image Type` para `Filled`. Essa alteração significa quanto da imagem será exibida e foi escolhida a opção para deixar de forma preenchida. As propriedade `Fill Metod` e `Fill Origin` definem o método que a imagem deve ser preenchida e qual borda é a origem, contendo os valores `vertical` e `top`.

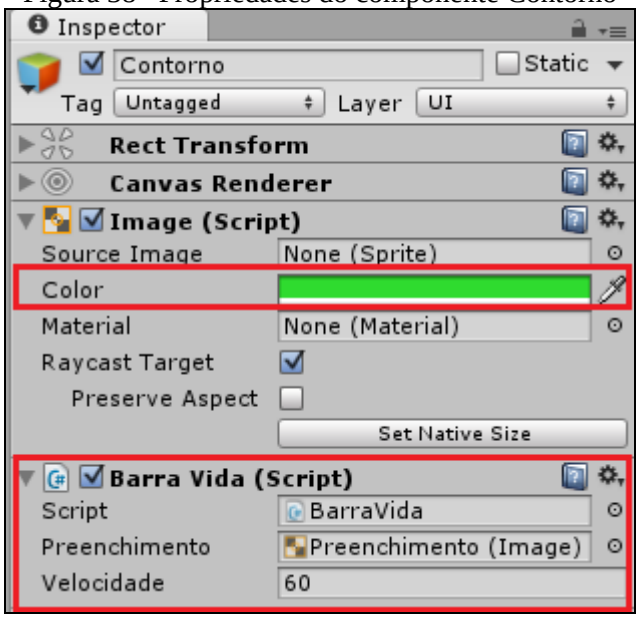
A propriedade que define o quanto da imagem `PerdeVida` é exibida chama-se `Fill Amount`, sendo o valor 0 que nada mostra e 1 sendo a imagem completa (Figura 37).

Figura 37– Propriedades do componente Preenchimento



O `Contorno` tem a propriedade `Color` alterada para verde e foi adicionado o *script* `BarraVida` (Figura 38). Nesse *script* são informados os atributos `preenchimento` e `tempo`. A variável `preenchimento` é do tipo `Image` e recebe o componente `Preenchimento`. A variável `velocidade` define a velocidade em que a imagem deve ser exibida.

Figura 38– Propriedades do componente Contorno



O Quadro 12 apresenta o método `Update` da classe `BarraVida` que é chamado a cada *frame* desenhado responsável por alterar a barra dinâmica.

Quadro 12–Método Update da classe BarraVida

```
15      void Update (){
16          preenchimento.fillAmount += 1.0f/velocidade * Time.deltaTime;
17          pontos =(1 - preenchimento.fillAmount) * 100;
18          PlayerPrefs.SetInt ("SalvaPonto",(int)pontos);
19
20          if (preenchimento.fillAmount == 1) {
21              Application.LoadLevel("notaFinal");
22          }
23      }
```

A pontuação para cada palavra reconhecida pode ser no máximo 100 e mínimo 0. O valor de `Fill Amount` varia de 0 até 1, logo quanto menor seu valor significa que mais rápido a palavra foi reconhecida. Por esse motivo é utilizada a fórmula na linha 17, onde do valor 1 é removido o valor atual do `preenchimento.fillAmount` e multiplicado por 100.

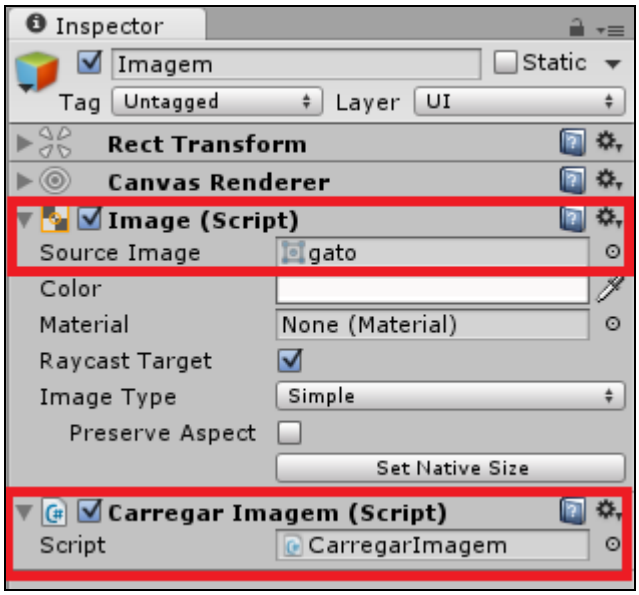
É utilizado o comando `PlayerPrefs` para salvar um valor no tipo inteiro. Através deste comando é possível recuperar o valor de uma variável em qualquer `Scene`. A variável `SalvaPonto` é responsável por armazenar o valor da pontuação que será exibida na `Scene notaFinal`.

Se o `preenchimento.fillAmount` conter o valor 1 significa que o tempo para o reconhecimento finalizou e a `Scene notaFinal` é chamada.

A imagem exibida como referência para a palavra ser reconhecida utiliza o componente `Imagem` do tipo `Image`. O componente contém a propriedade `Source Image` onde deve ser informado um arquivo tipo `Sprite`, que são objetos gráficos 2D utilizados para

personagens, adereços e outros elementos de jogos 2D. A `Imagem` utiliza o *script* `CarregarImagem`, que é responsável por alterar a imagem de acordo com o nível de dificuldade previamente selecionado (Figura 39).

Figura 39– Propriedades do componente Imagem



As imagens utilizadas podem ser buscadas de duas maneiras distintas. Uma delas é o conjunto de imagens que são disponibilizadas por padrão, localizadas dentro da estrutura do projeto. Outra maneira são imagens definidas pelo usuário. As imagens são carregadas através do método `Start()` da classe `CarregarImagem` (Quadro 13).

Quadro 13–Método Start da classe CarregarImagem

```
27      public void Start(){
28
29          if (PlayerPrefs.GetString ("jogarMinhasImagens") == "sim") {
30              buscarImagens ();
31          } else if (PlayerPrefs.GetInt ("NivelEscolhido") == 1) {
32              arraySprite = Resources.LoadAll<Sprite> ("Sprites/facil");
33          } else if (PlayerPrefs.GetInt ("NivelEscolhido") == 2) {
34              arraySprite = Resources.LoadAll<Sprite> ("Sprites/medio");
35          } else if (PlayerPrefs.GetInt ("NivelEscolhido") == 3) {
36              arraySprite = Resources.LoadAll<Sprite> ("Sprites/dificil");
37          }
38
39          adicionaSpriteEscolhida ();
40          buscarImagensAletorias ();
41      }
```

Inicialmente é realizado a verificação do modo de jogo selecionado. Se na tela inicial a escolha foi de jogar com as imagens contidas no dispositivo móvel é realizado a busca das imagens através do método `buscarImagens`. Caso tenha sido escolhido jogar com as imagens

padrão, apenas são retornadas as imagens contidas nas pastas fácil, médio ou difícil dependendo do nível selecionado.

Em seguida nas linhas 39 e 40, do quadro 13, são executados métodos responsáveis por verificar se a imagem já foi escolhida e buscar a mesma aleatoriamente. Isso permite que a imagem exibida a cada jogada não seja repetida e a cada novo jogo a ordem em que as imagens são exibidas seja aleatória, tornando o jogo menos repetitivo.

O método `buscarImagens` busca no diretório se existem imagens no formato PNG ou JPG. Em seguida armazena o nome da imagem sem sua extensão na variável `nomeImagem`, conforme linha 133 do Quadro 14. Com o nome da imagem é possível identificar a qual nível de dificuldade a mesma pertence. De acordo com cada nível é adicionado o caminho completo da imagem, incluindo nome e extensão, em uma lista.

Quadro 14- Método buscarImagens (primeiro trecho)

```
118     public void buscarImagens(){
119         //string myDir = "C:\\Users\\Vivian\\Documents";
120         string myDir = "/storage/emulated/0/Download/VISEDU";
121         DirectoryInfo dir = new DirectoryInfo (myDir);
122         nivel1 = new List<string>();
123         nivel2 = new List<string>();
124         nivel3 = new List<string>();
125 
126         List<System.IO.FileInfo> info = new List<System.IO.FileInfo> ();
127         info.AddRange (dir.GetFiles ("*.png", System.IO.SearchOption.TopDirectoryOnly));
128         info.AddRange (dir.GetFiles ("*.jpg", System.IO.SearchOption.TopDirectoryOnly));
129         if (info != null) {
130             string[] paths = new string[info.Count];
131             for (int i = 0; i < info.Count; i++) {
132                 paths [i] = info [i].FullName.ToString ();
133                 string nomeImagem = Path.GetFileNameWithoutExtension (paths [i]);
134 
135                 if (nomeImagem.Contains ("F_")) {
136                     nivel1.Add(paths[i]);
137                 } else if (nomeImagem.Contains ("M_")) {
138                     nivel2.Add(paths[i]);
139                 } else if (nomeImagem.Contains ("D_")) {
140                     nivel3.Add(paths[i]);
141                 }
142             }
143         }
```

Após ter o caminho de cada imagem em uma lista de acordo com o nível pertencente é necessário identificar qual será carregada. Isso ocorre de acordo com o nível selecionado pelo usuário. Essa verificação tem início na linha 144, conforme Quadro 15.

Quadro 15- Método buscarImagens (segundo trecho)

```
144         if (PlayerPrefs.GetInt ("NivelEscolhido") == 1) {
145             sprites = nivel1;
146         } else if (PlayerPrefs.GetInt ("NivelEscolhido") == 2) {
147             sprites = nivel2;
148         } else if (PlayerPrefs.GetInt ("NivelEscolhido") == 3) {
149             sprites = nivel3;
150         }
151 
152         arraySprite = new Sprite[sprites.Count];
153 
154         for (int i = 0; i < sprites.Count; i++) {
155             byte[] data = File.ReadAllBytes (sprites.ElementAt(i));
156             Texture2D texture = new Texture2D (270, 186, TextureFormat.ARGB32, false);
157             texture.LoadImage (data);
158             texture.name = Path.GetFileNameWithoutExtension (sprites.ElementAt(i));
159             Sprite spt = Sprite.Create (texture,new Rect(0,0,texture.width, texture.height),Vector2.zero);
160             spt.name = texture.name.Substring(2);
161             Debug.Log ("Sprite Texture = " + spt.name);
162             arraySprite[i] = spt;
163             }
164     }
```

Como pode ser visto no Quadro 15, linha 154, é realizado um laço de repetição sobre a lista de imagens, usado para criar todas as *Sprites*. Isto é realizado da seguinte forma. Na linha 155 é criado um *array* de *bytes* da imagem contida no *array* `sprites`. Em seguida, é criada uma textura e carregado os dados do arquivo através do método `LoadImage()`. Na linha 159 é criado uma *Sprite* por meio do método `Sprite.Create()`, com os valores da textura criada previamente. Para obter nome da *Sprite* é necessário utilizar o método `Substring()` para ignorar os dois primeiros caracteres que são os identificadores de nível, conforme linha 160.

A variável `arraySprite` recebe a *Sprite* criada que vai ser utilizada para exibir a imagem no aplicativo.

Após buscar as imagens que devem ser exibidas no jogo é necessário executar o método `adicionaSpriteEscolhida`. Esse método é responsável por verificar as *sprites* que já foram escolhidas e salvar a quantidade, conforme Quadro 16.

Quadro 16–Método adicionaSpriteEscolhida

```
89      private void adicionaSpriteEscolhida(){
90 
91          nomeSprite = new string[arraySprite.Length];
92          spritesEscolhidas = PlayerPrefs.GetString ("SpriteEscolhida");
93          string[] sprites = spritesEscolhidas.Split (';');
94          for(int i = 0; i < sprites.Length;i++){
95              if(sprites[i].Equals("")){
96                  sizeNomeSprite = 0;
97                  return;
98              }
99              nomeSprite[i] = sprites[i];
100         }
101         sizeNomeSprite = sprites.Length;
102     }
```

Inicialmente na linha 91 a variável `nomeSprite` recebe uma *string* com o tamanho de *sprites* carregadas.

Na linha 92 é carregado o valor da variável `SpriteEscolhida` que contém as *sprites* que já foram escolhidas. Em seguida é aplicado o comando `Split()` para separar as *sprites* através do separador ‘;’ e armazenado na variável `sprites`. Para a quantidade de `sprites` é verificado se o *array* `sprites` está vazio. Em seguida a variável `sizeNomeSprite` armazena a quantidade de *sprites*.

Após verificar as *sprites* que já foram escolhidas é chamado o método `buscarImagenAleatorias` que busca imagens aleatórias para ser exibida no jogo, conforme Quadro 17.

Quadro 17–Método buscarImagenAleatorias

```
44      public void buscarImagensAletorias(){
45          var img = GameObject.Find ("Imagem").GetComponent<UiImage> ();
46          int valor = Random.Range (0, arraySprite.Length);
47          Sprite sp = arraySprite[valor];
48          if (sizeNomeSprite == 0) {
49              img.sprite = sp;
50              sizeNomeSprite = 1;
51              nomeSprite[0] = sp.name;
52          } else {
53              if(arraySprite.Length == sizeNomeSprite){
54                  nomeSprite = new string[arraySprite.Length];
55                  img.sprite = sp;
56                  sizeNomeSprite = 1;
57                  nomeSprite[0] = sp.name;
58                  spritesEscolhidas = "";
59              }else{
60                  while(contemSprite(sp)){
61                      valor = (valor + 1) % arraySprite.Length;
62                      sp = arraySprite[valor];
63                  }
64                  img.sprite = sp;
65                  nomeSprite[sizeNomeSprite] = sp.name;
66                  sizeNomeSprite++;
67              }
68          }
69          if (spritesEscolhidas.Equals ("")) {
70              spritesEscolhidas = sp.name;
71          } else {
72              spritesEscolhidas = spritesEscolhidas + ";" + sp.name;
73          }
74          PlayerPrefs.SetString ("SpriteEscolhida", spritesEscolhidas);
75      }
```

Como pode ser visto no quadro 17, linha 45, é localizado o componente `Image` que vai exibir a imagem. Em seguida é gerado um valor aleatório que vai de 0 até a quantidade de *sprites*. Na linha 47 é criada a variável `sp` do tipo *Sprite* sendo atribuído uma *sprite* em uma

posição randômica. Se `sizeNomesprite`, que é a variável que contém a quantidade de *sprites* que já foram escolhidas for 0, significa que a *sprite* localizada pode ser carregada, conforme linha 55. Em seguida é incrementado o valor de `sizeNomesprite` e a variável `nomeSprite` que é um *array* de *string* recebe o nome da *sprite* selecionada.

Se a variável `sizeNomesprite` é diferente de 0 é verificado se a quantidade de *sprites* total é igual a quantidade de *sprites* já escolhidas. Caso isso for verdade significa que já foram exibidas todas as imagens disponíveis. Senão, enquanto o método `contemSprite()` for verdadeiro busca uma próxima *sprite*.

Na linha 69 é verificado se a variável `spritesEscolhidas` está vazia, que vai ocorrer na primeira execução. Se for verdadeiro vai adicionar o nome da *sprite* escolhida. Se essa variável já conter valores é adicionado o ‘;’ concatenado com o nome da *sprite* escolhida.

Para finalizar a variável `SpriteEscolhida` recebe a *sprite* que foi escolhida. Conforme mencionado anteriormente o método `contemSprite` retorna verdadeiro se a *sprite* selecionada está contida no *array* de *sprites* já escolhidas, conforme Quadro 18.

Quadro 18– Método contemSprite

```
108     private bool contemSprite(Sprite sp){
109         for (int i = 0; i < nomeSprite.Length; i ++) {
110             if (sp.name.Equals (nomeSprite [i])) {
111                 return true;
112             }
113         }
114         return false;
115     }
```

Após exibir a imagem na tela é realizado o processo de reconhecimento de texto. Para realizar a identificação da palavra formada é necessário que a mesma seja posicionada na região de interesse. Sempre que for detectada uma palavra é chamado o método `OnWordDetected` da classe `TextEventHandler` (Quadro 19).

Quadro 19–Método OnWordDetected da classe TextEventHandler

```
160     public void OnWordDetected(WordResult wordResult){
161
162         var word = wordResult.Word;
163         var img = GameObject.Find("Imagem").GetComponent<UiImage>();
164
165         if (ContainsWord(word))
166             Debug.LogError("Word was already detected before!");
167
168         if (wordResult.Word.StringValue == img.sprite.name.ToUpper()){
169             Application.LoadLevel("notaFinal");
170         }
171
172         AddWord(wordResult);
173
174     }
```

No parâmetro do método é utilizada a variável `wordResult` do tipo `WordResult`. Esse tipo fornece o estado atual de uma palavra, contendo todas as suas informações que são alteradas a cada *frame*.

Na linha 168 é realizado a comparação da palavra detectada com o nome da *sprite* utilizada pela imagem. Caso seja verdadeiro significa que a palavra foi reconhecida e a aplicação é direcionada para a *scene* `notaFinal`.

Durante o reconhecimento de texto é possível parar o jogo. Ao tocar duas vezes na tela é habilitado o *canvas* `ImagemPause`, que contém os botões `Iniciar` e `Voltar`, conforme pode ser visto na Figura 40.

Figura 40– Tela de reconhecimento de texto com a opção pause

Ao dar dois toques na tela é executado o método `UpdateInput` da classe `InputController`, Quadro 20.

Quadro 20–Método UpdateInput da classe InputController

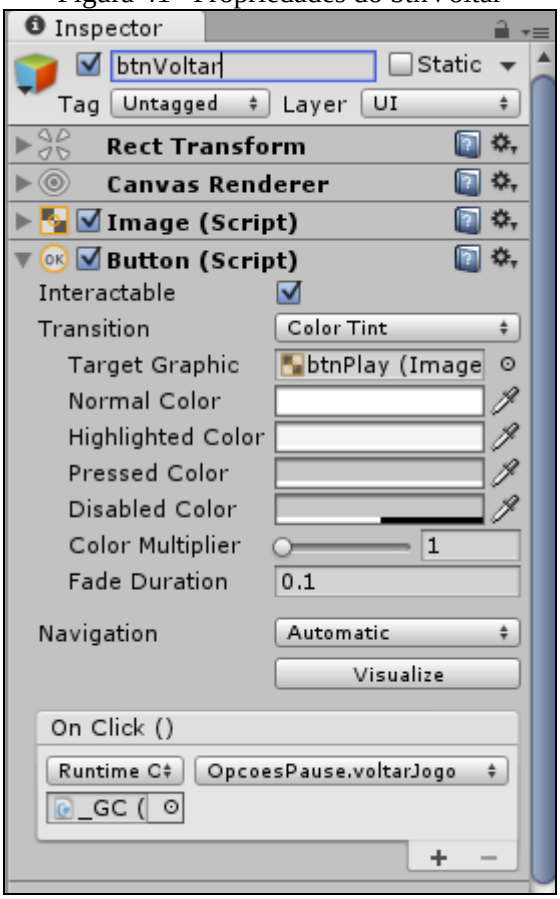
```
79      Debug.Log ("Double Tap Registered");
80      tapEventDispatched = true;
81      Time.timeScale = 0;
82      Canvas canvasPause = GameObject.Find("CanvasPause").GetComponent<Canvas>();
83      canvasPause.enabled = true;
```

Como pode ser visto no Quadro 20, linha 81 é utilizado o comando `Time.timeScale`. Esse comando faz com que as ações fiquem em estado de parada pois o `timescale` é a escala de tempo da *scene*, quando está com valor 1 (um) fica na velocidade normal, quando tem o

valor 0 (zero), fica em estado de parada. Na linha 83 é ativado o *canvas* que contém as opções quando o jogo está parado.

O botão Continuar contém o script `OpcoesPause` onde é executado o método `voltarJogo`, conforme Figura 41.

Figura 41– Propriedades do btnVoltar



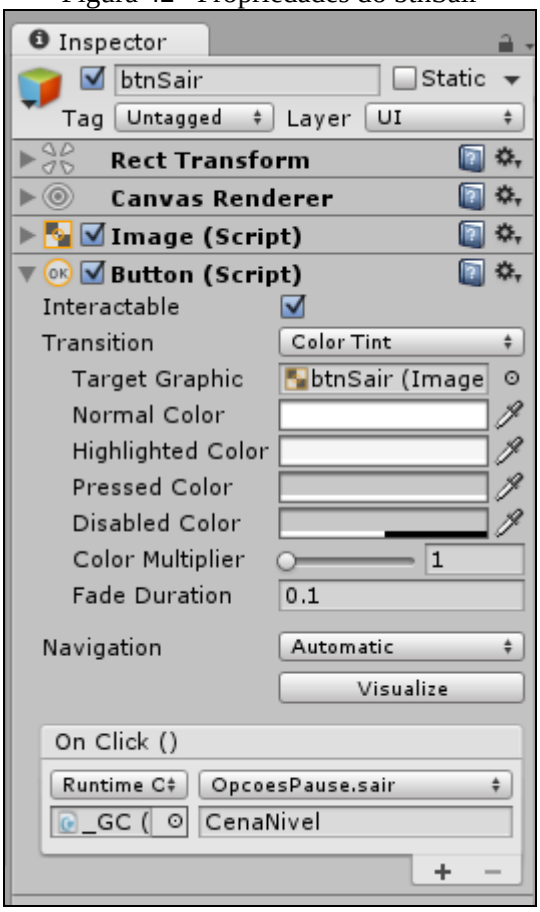
O método `voltarJogo` do script `OpcoesPause` altera a velocidade da escala de tempo do jogo para o estado normal, conforme linha 8 do Quadro 21. Em seguida é desabilitado o *canvas* `canvasPause`.

Quadro 21–Método voltarJogo da classe opcoesPause

```
7    public void voltarJogo () {
8        Time.timeScale = 1;
9        Canvas canvasPause = GameObject.Find("CanvasPause").GetComponent<Canvas>();
10       canvasPause.enabled = false;
11   }
```

O botão `Voltar` contém o *script* `opcoesPause` e executa o método `sair`, conforme Figura 42.

Figura 42– Propriedades do btnSair



O método sair altera a velocidade da escala de tempo do jogo para o estado normal, conforme linha 14 do Quadro 22. Em seguida é carregada a *scene* `CenaNivel`, saindo da *scene* de reconhecimento de texto.

Quadro 22- Método sair da classe opcoesPause

```
13      public void sair(string nomeCena){
14          Time.timeScale = 1;
15          Application.LoadLevel (nomeCena);
16      }
```

3.3.1.5 Tela nota final

Caso a palavra tenha sido reconhecida ou o tempo para o reconhecimento tenha finalizado a tela nota final é exibida. A tela contém um botão para retornar a escolha de nível e outro botão para continuar, conforme Figura 43.

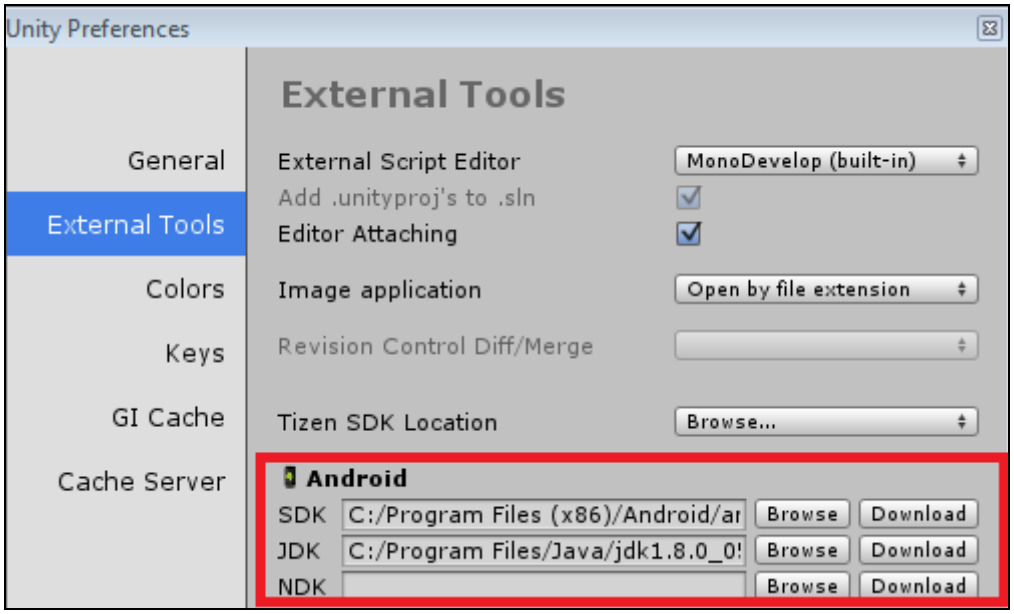
Figura 43– Tela de nota final

Ao tocar no botão de `voltar`, item A da Figura 43, a *scene* `CenaNivel` é exibida. Ao tocar no botão `continuar`, item B, é exibida a *scene* `Vuforia-TextReco`. Ambos os botões utilizam o método `carregaCena` do *script* `CarregaProximaCena` (ver seção 3.3.1.3).

3.3.1.6 Plataforma disponibilizada para o aplicativo

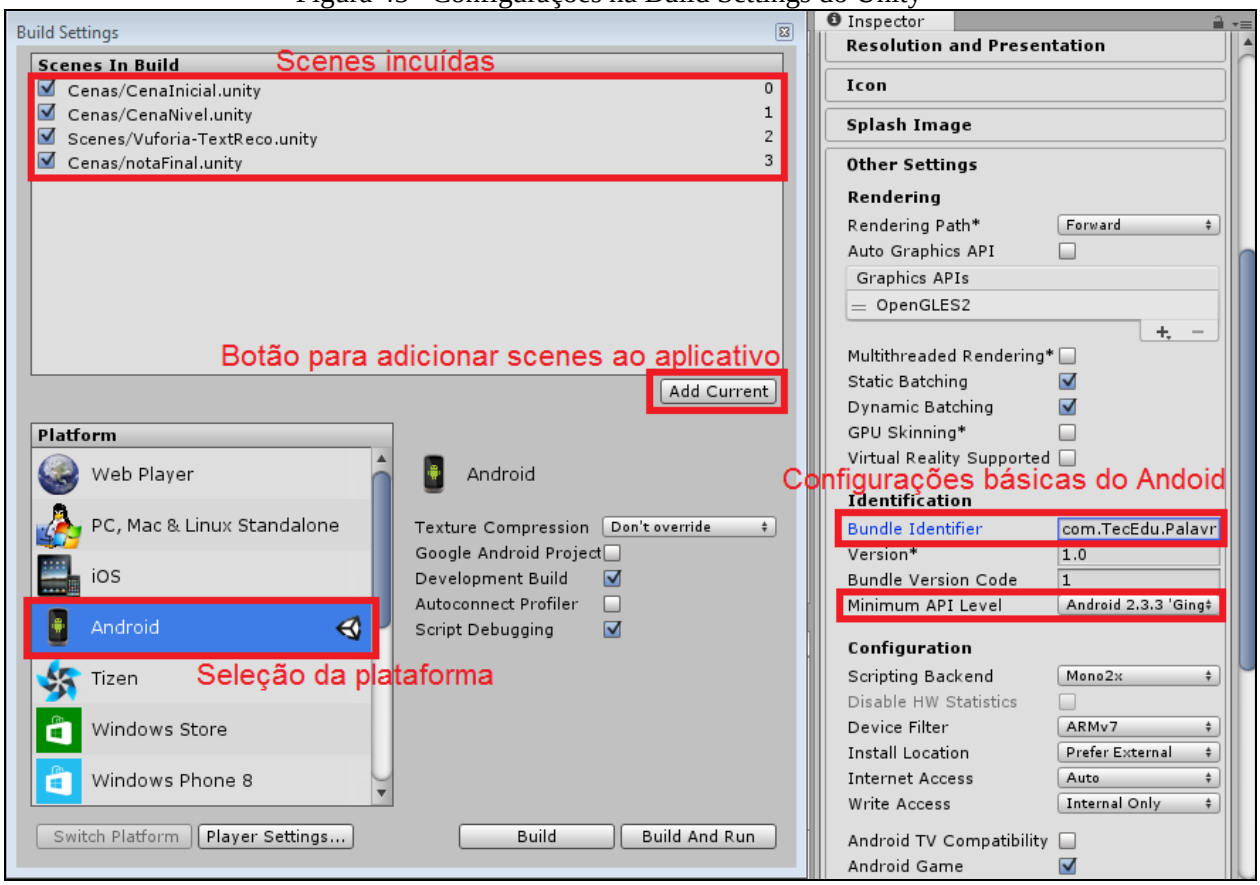
O aplicativo foi disponibilizado para a plataforma Android nas versões 2.3.3 e superiores. Para gerar o projeto para dispositivos com o sistema operacional Android é necessário definir preferencias do Unity o local onde se encontra o SDK do Android e o JDK, conforme Figura 44.

Figura 44– Preferencias do Unity



Após as configurações, é necessário acessar as `Build Settings` do Unity, selecionar a plataforma Android e clicar em `Player Settings`. As configurações básicas do `Player Settings` para a geração do aplicativo são: definir um `Bundle Identifier` (identificação do pacote) e `Minimum API Level` (versão mínima do Android para executar a aplicação). Para que todas *scenes* sejam incluídas no aplicativo é necessário adicioná-las uma por uma através do botão `Add Current` e definir uma ordem de execução. Na Figura 45 são apresentadas as opções descritas acima.

Figura 45– Configurações na Build Settings do Unity



3.3.2 Operacionalidade da implementação

Nesta seção serão apresentadas as funcionalidades das telas utilizadas pelo usuário. Esta seção apresenta a tela inicial do aplicativo, tela de seleção de nível, tela de reconhecimento de texto e tela nota final. Para as demonstrações a seguir, foi utilizado o dispositivo LG G3 com Android 4.4.2 instalado.

3.3.2.1 Tela inicial do aplicativo

A tela inicial do aplicativo é apresentada a opção `Imprimir letras`, para que o usuário possa imprimir as letras utilizadas para formar as palavras. Após a impressão das letras, o usuário deve escolher a forma que as imagens utilizadas no jogo devem ser buscadas. Caso o usuário toque no botão `Jogar`, as imagens carregadas são as disponibilizadas por padrão. Se o usuário tocar no botão `Jogar com minhas imagens`, as imagens são as escolhidas pelo usuário contidas no dispositivo. As imagens devem estar na pasta no diretório `/storage/emulated/0/download/VISEDU`. Cada imagem deve conter o nome da palavra que deve ser reconhecida ao exibir a imagem e a inicial do nível que a imagem deve pertencer. Se a imagem for do nível fácil, o nome da imagem deve iniciar com `F_`. Se a

imagem for do nível médio, o nome da imagem deve iniciar com `M_` e se a imagem for do nível difícil, o nome da imagem deve iniciar com `D_`. A Figura 46 mostra a tela inicial.

Figura 46– Tela inicial do aplicativo

3.3.2.2 Tela de escolha de nível

A tela de escolha de nível permite ao usuário escolher o nível de dificuldade para jogar. Os níveis disponíveis são fácil, médio ou difícil. Após tocar em um dos botões de nível é exibida uma mensagem informando o nível selecionado e somente então o botão de jogar é habilitado, conforme Figura 47. Existe a possibilidade de retornar para a tela inicial, tocando no botão de voltar.

Figura 47– Tela de escolha de nível

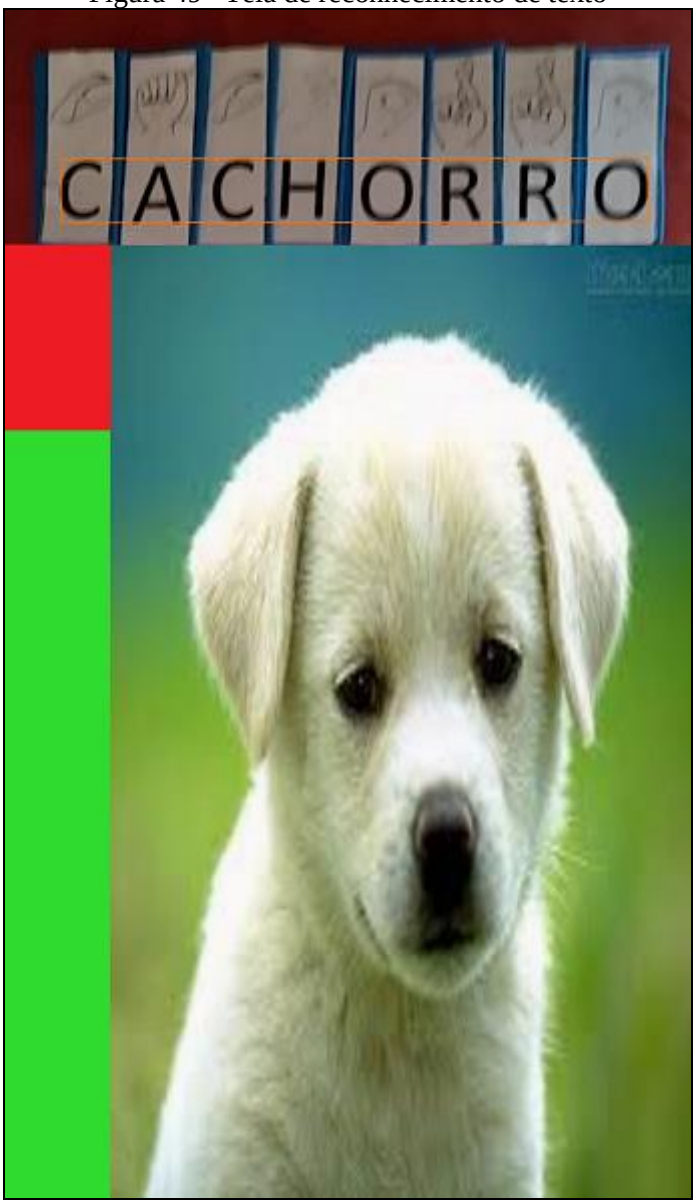
Essa tela contém a opção de ajuda. Ao tocar no botão `Ajuda?` são exibidas informações que auxiliam na escolha do nível de dificuldade e os botões `Próximo` e `Sair Ajuda` são habilitados. Ao tocar no botão `Próximo` são exibidas informações que auxiliam no reconhecimento de texto, tais como o local onde a palavra deve ser posicionada e o tempo restante para o reconhecimento representado pela barra de vida. Para retornar a seleção de nível é necessário tocar no botão `Sair Ajuda`. A Figura 48 (esquerda) exibe as opções de ajuda.

Figura 48– Opção de ajuda(esquerda) e próxima ajuda(direita)

3.3.2.3 Tela de reconhecimento de texto

Após escolher o nível de dificuldade e tocar no botão `Jogar` é exibida a tela de reconhecimento de texto. Uma imagem é exibida na tela como referência para a palavra que deve ser formada. A palavra deve ser formada em uma superfície plana e iluminada, com as letras próximas umas das outras. Após a formação da palavra o dispositivo deve ser posicionado de forma que a palavra esteja visível na área de interesse, ou seja a área onde é exibido a câmera. A palavra deve ser reconhecida antes que o tempo finalize. Esse tempo é representado por uma barra de vida, como pode ser visto na Figura 49.

Figura 49– Tela de reconhecimento de texto



Durante o reconhecimento tem se a opção de parar o jogo ao dar dois toques em qualquer lugar da tela, habilitando as opções `continuar` e `voltar`, conforme Figura 50. Ao tocar no botão `Continuar` apenas é fechada a opção de parar e o jogo continua. Caso o botão `voltar` seja tocado é exibido a tela de escolha de níveis.

Figura 50– Tela de reconhecimento de texto com a opção de parada habilitada

3.3.2.4 Tela de nota final

Após a palavra ser reconhecida ou o tempo de reconhecer a palavra terminar é exibida a tela nota final. Essa tela contém a pontuação feita pelo usuário e as opções de continuar jogando ou retornar para a tela de escolha de nível, conforme Figura 51. Ao tocar no botão `continuar` é exibida a tela de reconhecimento de texto contendo uma imagem do mesmo nível selecionado anteriormente.

Figura 51– Tela nota final

3.4 RESULTADOS E DISCUSSÕES

Nesta seção é apresentada um experimento realizado com o aplicativo e um teste realizado referente a quantidade de palavras utilizadas para o reconhecimento. Na seção 3.4.1. é descrita a metodologia do experimento realizado. O experimento detalhado na seção 3.4.2, foi realizado com doze usuários no laboratório LIFE da FURB. Os usuários são da turma de Sistemas Multimídia dos cursos de Ciências da Computação e Pedagogia. O experimento teve como objetivo avaliar a aplicabilidade do aplicativo no aprendizado de crianças em idade de alfabetização.

3.4.1 Metodologia

O experimento aconteceu no dia primeiro do mês de junho por meio de testes realizados com usuários individualmente, acompanhados com a autora desde trabalho. Para os testes foi disponibilizado um dispositivo móvel e um conjunto de letras impressas em papel cartão. No experimento foi disponibilizado um questionário de avaliação do aplicativo, que está disponível no Apêndice B.

3.4.2 Experimento

O cenário utilizado no teste consistia em selecionar o modo de jogo e jogar com as imagens do nível fácil. Antes da realização do teste foi apresentada uma visão geral do funcionamento do aplicativo. A primeira etapa consistia em selecionar o modo de jogo onde contém as imagens padrão do aplicativo, tocando no botão `Jogar`. Em seguida foi solicitado a que o nível fácil fosse selecionado. Após tocar no botão para iniciar o jogo, o usuário visualizou a imagem e iniciou a montagem da palavra. Percebeu-se que alguns usuários ao formar as palavras deixavam um espaço entre as letras, impossibilitando o reconhecimento da palavra. Porém, após aproximar as letras o reconhecimento foi realizado e a tela de nota final foi exibida. Em seguida, o usuário realizou o mesmo processo para as três imagens disponibilizadas para o nível fácil. Após finalizar o uso do aplicativo, todos os usuários responderam um questionário contendo perguntas sobre o perfil do usuário, instruções de uso, e usabilidade do aplicativo. Na Tabela 1 são exibidos os perfis dos usuários envolvidos no experimento.

Tabela 3- Perfis de usuários envolvidos no teste

Sexo	Masculino	8,3%
	Feminino	91,7%
Idade	Entre 16 e 20 anos	8,3%
	Entre 21 e 25 anos	91,7%
Nível de escolaridade	Ensino médio completo	8,3%
	Ensino superior incompleto	91,7%
Frequência que utiliza computador	Frequentemente	100%
Frequência que utiliza dispositivos móveis	Ás vezes	8,3%
	Frequentemente	91,7%
Grau de familiaridade com Realidade Aumentada	Conheço, mas nunca utilizei	50%
	Já utilizei	50%

Em seguida os usuários responderam a parte do questionário referente as instruções onde devia ser informado se conseguiu realizar cada uma das instruções. Os resultados são demonstrados na Tabela 2.

Tabela 4–Respostas do questionário de avaliação

Conseguiu selecionar a opção jogar	Conseguiu	100%
Conseguiu selecionar o nível de dificuldade	Conseguiu	100%
Conseguiu tocar no botão de jogar	Conseguiu	100%
Conseguiu formar a palavra de acordo com a imagem exibida	Conseguiu	100%
Conseguiu posicionar o dispositivo móvel de forma que a palavra seja exibida na área de reconhecimento	Conseguiu	100%
Na tela nota final escolher a opção para continuar	Conseguiu	100%
Repetir o processo para as três imagens do nível	Conseguiu	100%

Após os usuários responder a parte de instruções, cada usuário respondeu ao questionário de usabilidade do aplicativo. Os resultados obtidos estão detalhados na Tabela 3

Tabela 5–Respostas do questionário de usabilidade

Das atividades solicitadas, quantas conseguiu executar	A maior parte	16,7%
	Todas	83,3%
Achou o aplicativo intuitivo e fácil de usar	Sim	100%
Avaliação do aplicativo	Bom	25%
	Muito bom	75%
Acredita que jogos auxiliam no aprendizado	Sim	100%
Acredita que jogos juntamente com dispositivos móveis torna o aprendizado mais atrativo	Sim	100%
Sentiu atraído a aprender a língua de sinais	Não	8,3%
	Pouco atraído	16,7%
	Sim	75%

Conforme pode ser visto nas tabelas 4 e 5, o aplicativo obteve resultados relativamente positivos. Como o aplicativo foi criado voltado para crianças todos usuários e tinham idade acima de 16 anos acharam o aplicativo fácil de usar e intuitivo.

Metade dos usuários apenas conhecia Realidade Aumentada, porém não havia utilizado e mesmo assim conseguiram realizar quase todas as atividades solicitadas utilizando o aplicativo. Percebeu-se que a única dificuldade foi o posicionamento do dispositivo móvel para o reconhecimento da palavras, mas após o reconhecimento da primeira palavra foram se adaptando. Um usuário apenas achou que a área de reconhecimento de texto é pequena.

Nota-se que todos os usuários acreditam que os jogos auxiliam no aprendizado, inclusive utilizando dispositivos móveis tornando o aprendizado mais atrativo. Isso porque os usuários gostam do ambiente lúdico, das cores e cenário utilizados no aplicativo.

Nas letras utilizadas para o teste, acima de cada uma havia o sinal em LIBRAS (Língua Brasileira de Sinais), com o intuito de atrair para o aprendizado da mesma. Por meio deste experimento percebeu-se que quase todos os usuários se sentiram atraídos a aprender essa linguagem.

3.5 COMPARATIVO DOS TRABALHOS CORRELATOS

Esta seção é responsável por realizar a comparação dos trabalhos correlatos descritos no capítulo anterior e este trabalho. O Quadro 23 apresenta a relação de existência de características nos trabalhos apresentados. O presente trabalho está identificado como Lima (2016).

Quadro 23–Comparativo dos trabalhos correlatos e este trabalho

Características/ Trabalhos	OsmoPlay (2014)	Games (2015)	Kids (2015)	Lima (2016)
É um jogo que auxilia na alfabetização	X	X	X	X
Permite a utilização de marcadores	X			X
É um jogo de Realidade Aumentada	X			X
É um trabalho acadêmico				X
Utiliza biblioteca Vuforia para identificação de palavras				X
Plataforma Suportada	iOS	Android	Android	Android

Através das informações disponíveis no Quadro 23, percebe-se que todos os trabalhos auxiliam na alfabetização. Somente o OsmoPlay e este trabalho permitem a utilização de marcadores. Nota-se que o OsmoPlay e o presente trabalho são os únicos que utilizam Realidade Aumentada, porém apenas este trabalho utiliza a biblioteca Vuforia.

4 CONCLUSÕES

O presente trabalho apresentou a criação de um aplicativo de Realidade Aumentada com imagens de forma dinâmica para a plataforma Android. O objetivo deste trabalho foi realizar o reconhecimento de palavras por meio de marcadores contendo letras. Para isso foi utilizando como referência imagens definidas pelo usuário e possibilidade de definir o nível de dificuldade para as mesmas.

Para alcançar os objetivos do trabalho, foi utilizada a plataforma de desenvolvimento de jogos e aplicativos Unity 3D em conjunto com a biblioteca Vuforia, a qual disponibilizou os recursos necessários para a implementação da RA do aplicativo. A combinação do Unity com Vuforia pode ser considerada adequada, pois forneceu os recursos para a implementação do trabalho.

Ao utilizar em conjunto Unity com Vuforia, foi possível realizar o reconhecimento de texto através da câmera de um dispositivo de acordo com uma imagem exibida no aplicativo. O Unity possibilitou a criação de algumas telas que possibilitam configurações do modo de jogo e tendo como retorno visual uma barra dinâmica que é decrementada de acordo com o tempo a cada imagem exibida.

Os objetivos propostos foram atingidos. Contudo houveram algumas limitações, como a estrutura de arquivos necessária para a inserção de novas imagens e quantidade de palavras que podem ser reconhecidas.

Os resultados obtidos a partir dos testes realizados se mostraram satisfatórios, sendo que a maior parte dos usuários acharam fácil a usabilidade do aplicativo, ficaram satisfeitos com o ambiente lúdico, se sentiram motivados com os desafios propostos pelo jogo, além de acharem que a utilização de jogos juntamente com dispositivos móveis e Realidade Aumentada torna o ensino mais atrativo para as crianças, deixando-as mais motivadas para o aprendizado.

4.1 EXTENSÕES

Como sugestão de extensões para continuidade do presente trabalho tem-se:

a) permitir a utilização de qualquer dispositivo móvel para execução do aplicativo;
b) cadastro de usuário, sendo eles professor e aluno;
c) no perfil do usuário aluno, permitir acesso a evolução dentro do jogo;
d) utilização de um maior número de palavras no arquivo de palavras adicionais.

REFERÊNCIAS

ARAÚJO, M. D. **Uso De Realidade Aumentada Como Ferramenta Complementar Ao Ensino Das Principais Ligações Entre Átomos.** [S.l.], 2009. Disponível em: <http://sites.unisanta.br/wrva/st/62401.pdf >. Acesso em 18 set 2015.

AZUMA, Ronald T. A Survey of augmented reality. **Presence**: Teleoperators and Virtual Environments. [S.l.], v. 6, n. 4, p. 355-385. ago. 1997. Disponível em: <http://www.cs.unc.edu/~azuma/ARpresence.pdf>. Acesso em: 29 ago. 2015.

COOLMOMTECH. **The new Masterpiece app for Osmo helps turn kids into brilliant artists.** [S.l.], 2015. Disponível em: <http://coolmomtech.com/2015/03/masterpiece-app-for-osmo/>. Acesso em 19 set 2015.

DEVMEDIA. **Como criar aplicativos mobile híbridos e offline**. [S.l.], 2015. Disponível em: < http://www.devmedia.com.br/como-criar-aplicativos-mobile-hibridos-e-offline/32361>. Acesso em: 29 ago 2015.

_____. **Desenvolva jogos com a Unity 3D**. [S.l.], 2015. Disponível em: <http://www.devmedia.com.br/desenvolva-jogos-com-a-unity-3d/29125>. Acesso em: 29 ago 2015.

ELER, N. G. **Realidade aumentada auxilia na compreensão de conceitos de física.** [S.l.], 2015. Disponível em: <http://www.usp.br/aun/exibir.php?id=6962&edicao=1215>. Acesso em 18 set 2015.

FABRE, M. J. et al. **Jogos Educacionais**. [S.l.], 2004. Disponível em: < http://www.virtual.ufc.br/cursouca/modulo_3/Jogos_Educacionais.pdf>. Acesso em 18 set 2015.

GAMES, E. **Forma Palavras**. [S.l.], 2015. Disponível em: <https://play.google.com/store/apps/details?id=air.com.escolagames.FormaPalavras>. Acesso em: 29 ago 2015.

GIZMODO,2013. **Big Bird Will Teach Kids To Read With Qualcomm's Augmented Reality Tech.** [S.l.], 2015. Disponível em: < http://www.gizmodo.com.au/2013/01/big-bird-will-teach-your-kid-to-read-using-qualcomms-augmented-reality-tech/>. Acesso em 18 set 2015.

ITUNES. **Newton for Osmo.** [S.l.], 2015. Disponível em: <https://itunes.apple.com/us/app/newton-for-osmo/id876524974?mt=8>. Acesso em: 18 set 2015.

KIDS,G.F. **Monta Palavras**. [S.l.], 2015. Disponível em: <https://play.google.com/store/apps/details?id=br.com.jeronimo.montapalavras>. Acesso em: 29 ago 2015.

LIRA, D.A. **Biblioteca para Desenvolvimento de Jogos Múlti Jogadores de Realidade Aumentada para Android**. 2015. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

PLAYOSMO. **Award – Winning Educational Games System for iPad**. [S.l.], 2015. Disponível em: <https://www.playosmo.com/en/>. Acesso em: 29 ago 2015.

QUALCOMM. **Vuforia**. [S.l.], 2015. Disponível em: <http://www.qualcomm.com.br/products/vuforia/>. Acesso em: 29 ago 2015.

SUTHERLAND, I. E. **The Ultimate Display.** [S.l.], 2011. Disponível em: <http://www.de.ufpb.br/~labteve/publi/2011_svrps.pdf>. Acesso em 18 set 2015.

TECHHIVE, 2015**. Osmo review: Hands-on iPad games with real pieces give kids new ways to play.** [S.l.], 2015. Disponível em: < http://www.techhive.com/article/2452546/osmo-review-hands-on-ipad-games-with-real-pieces-give-kids-new-ways-to-play.html >. Acesso em: 19 set 2015.

TECMUNDO ,2009. **Jogos universais como futuro do entretenimento.** Disponível em: <http://www.tecmundo.com.br/realidade-aumentada/2795-jogos-universais-como-futuro-do-entretenimento.htm > Acesso em 14 set 2015.

http://www.tecmundo.com.br/webcam/2394-novas-tecnologias-jogos-interativos.htm

TORI, R., KIRNER, C., SISCOUTTO, R. 2006. **Livro do Pré-Simpósio. Fundamentos e Tecnologicas de Realidade Virtual e Aumentada. Editora SBC, VIII Symposium on Virtual Reality.**[S.l.], 2015. Disponível em: <http://www.ckirner.com/download/capitulos/Fundamentos_e_Tecnologia_de_Realidade_Virtual_e_Aumentada-v22-11-06.pdf>. Acesso em 14 set 2015.

SILVA, Marco. **O que é interatividade.** [S.l.], 2001. Disponível em: <http://www.senac.br/informativo/bts/242/boltec242d.htm>. Acessado em: 20 mai. 2016.

UNITY. **Unity Game Engine**. [S.l.], 2015a. Disponível em: <http://unity3d.com/pt/>. Acesso em: 29 ago 2015.

_____. **Learning the Interface.** [S.l.], 2016a. Disponível em:

< http://docs.unity3d.com/Manual/LearningtheInterface.html>. Acesso em: 26 mai. 2016.

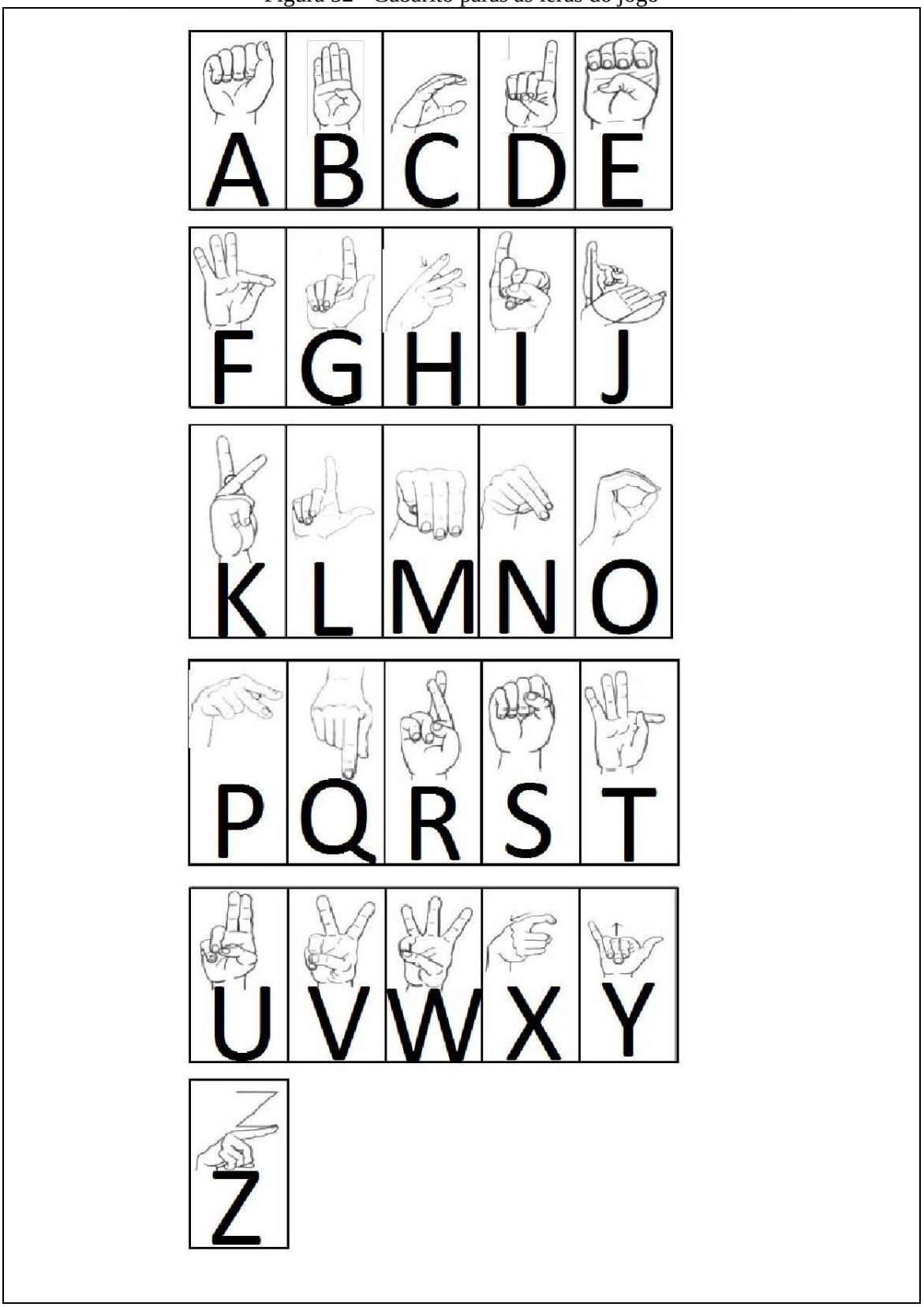
_____. **Event System.** [S.l.], 2016b. Disponível em:

< http://docs.unity3d.com/Manual/EventSystem.html>. Acesso em: 26 mai. 2016.

APÊNDICE A – Gabarito para as peças do jogo

A Figura 52 apresenta o gabarito para as letras utilizadas no jogo.

Figura 52 - Gabarito paras as leras do jogo



APÊNDICE B – Formulário de avaliação da experiência de usuário da aplicação

O Quadro 23 apresenta o formulário de avaliação de usabilidade da aplicação.

Quadro 24- Formulário de avaliação da experiência de usuário da aplicação

Questionário do VisEdu: Jogo de Realidade Aumentada de Letras com Conteúdo Dinâmico

O questionário é parte integrante do Trabalho de Conclusão de Curso intitulado "VISEDU: Jogo de Realidade Aumentada com Conteúdo Dinâmico" realizado na Universidade Regional de Blumenau pela acadêmica Vivian de Lima Panzenhagen e professor/orientador Dalton Solano dos Reis.

Perfil de usuário

Observação: as informações recebidas abaixo serão mantidas de forma confidencial.

Obrigatório*

Sexo: *
( ) Masculino
( ) Feminino

Idade: *
( ) Tenho menos de 5 anos
( ) Tenho entre 6 a 10 anos
( ) Tenho entre 11 a 15 anos
( ) Tenho entre 16 a 20 anos
( ) Tenho entre 21 a 25 anos
( ) Tenho entre 26 a 30 anos
( ) Tenho mais de 30 anos

Nível de escolaridade: *
( ) Ensino fundamental incompleto
( ) Ensino fundamental completo - 1º grau
( ) Ensino médio incompleto
( ) Ensino médio completo – 2º grau
( ) Ensino superior incompleto
( ) Ensino superior completo

Você utiliza o computador com qual frequência? *
( ) Nunca utilizei
( ) Ás vezes
( ) Frequentemente

Você utiliza dispositivos móveis com qual frequência? *
( ) Nunca utilizei
( ) Ás vezes
( ) Frequentemente

Indique seu grau de familiaridade com Realidade Aumentada: *
( ) Nunca utilizei
( ) Conheço, mas nunca utilizei
( ) Já utilizei

Instruções

Com este questionário buscamos avaliar a utilização da aplicação. A aplicação realiza o reconhecimento de palavras por meio de marcadores. Após selecionar o nível de dificuldade, deve ser posicionado os marcadores até formar a palavra de acordo com a imagem exibida. Ao finalizar, solicitamos que prossiga nos testes conforme as orientações abaixo.

Selecione a opção "Jogar" no menu inicial*
A tarefa foi realizada?
( ) Sim
( ) Não
Observação:_____

Selecione um dos níveis de dificuldade e clique no botão de "Play"*
A tarefa foi realizada?
( ) Sim
( ) Não
Observação:_____

Uma imagem será exibida. Forme a palavra correspondente a imagem exibida na cena do jogo em uma superfície plana e iluminada. *
A tarefa foi realizada?
( ) Sim
( ) Não
Observação:_____

Posicione o dispositivo móvel de forma que a palavra formada seja exibida na área de reconhecimento*
A tarefa foi realizada?
( ) Sim
( ) Não
Observação:_____

Após a palavra ser reconhecida será exibida a tela "Nota final". Selecione a opção "Play" para ir para a próxima imagem. *
A tarefa foi realizada?
( ) Sim
( ) Não
Observação:_____

Repita o processo para que a palavra seja reconhecida até que as três imagens do nível sejam reconhecidas. *
A tarefa foi realizada?
( ) Sim
( ) Não
Observação:_____

Questionário de usabilidade

Das atividades solicitadas, quantas você conseguiu executar?*
( ) Todas
( ) A maior parte
( ) Metade das tarefas
( ) Menos da metade das tarefas
( ) Nenhuma tarefa

De modo geral, você achou o protótipo intuitivo e fácil de usar?*
( ) Sim
( ) Não

Qual é a sua avaliação da aplicação?*
( ) Muito bom
( ) Bom
( ) Regular
( ) Insatisfatório

Qual foi a sua maior dificuldade utilizando a aplicação?
_____

Você acredita que jogos podem auxiliar no aprendizado?*
( ) Sim
( ) Ajuda pouco
( ) Não

Você acha que os jogos juntamente com dispositivos móveis tornam o aprendizado mais atrativo?*
( ) Sim
( ) Não

Você se sentiu atraído a aprender a linguagem de sinais?*
( ) Sim
( ) Não