UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

VISEDU: INTERFACE DE USUÁRIO TANGÍVEL UTILIZANDO REALIDADE AUMENTADA E UNITY

ANTÔNIO MARCO DA SILVA

BLUMENAU
2016

ANTÔNIO MARCO DA SILVA

VISEDU: INTERFACE DE USUÁRIO TANGÍVEL UTILIZANDO REALIDADE AUMENTADA E UNITY

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, M. Sc. - Orientador

BLUMENAU
2016

VISEDU: INTERFACE DE USUÁRIO TANGÍVEL

UTILIZANDO REALIDADE AUMENTADA E UNITY

Por

ANTÔNIO MARCO DA SILVA

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, M. Sc. – Orientador, FURB
Membro:	Prof. Aurélio Faustino Hoppe, M. Sc. – FURB
Membro:	Prof. Maurício Capobianco Lopes, Doutor – FURB

Blumenau, 07 de julho de 2016

Dedico este trabalho ao meu pai e minha mãe, aos meus amigos e ao meu orientador que me apoiaram desde o início até o fim deste trabalho.

AGRADECIMENTOS

Ao meu pai Marco Aurélio e minha mãe Marcia Jacobsen pelo incentivo.

Ao meu amigo Gibran Peruzzolo ao me acompanhar no desenvolvimento deste trabalho.

Ao meu orientador Dalton Reis, por seu apoio, interesse e colaboração com este trabalho.

Para criar um novo padrão, você tem que estar preparado para o desafio e realmente apreciá-lo.

Shigeru Miyamoto

RESUMO

Este trabalho apresenta o desenvolvimento de um aplicativo para a plataforma Android, que utiliza os conceitos da Interface de usuário tangível em conjunto com a Realidade Aumentada em um ambiente para manipular objetos tridimensionais virtuais, permitindo um ambiente intuitivo para alterar as informações destes objetos virtuais. O aplicativo foi desenvolvido utilizando a plataforma de criação de jogos e aplicativos Unity, em conjunto com a biblioteca de Realidade aumentada Vuforia. Durante o desenvolvimento do trabalho foram executadas diferentes formas de interatividade de usuário utilizando as técnicas de Interface Tangível e, assim, interagindo e manipulando objetos virtuais em um ambiente de Realidade Aumentada. Além disso, são apresentados conceitos, técnicas e componentes utilizados para a implementação da Realidade Aumentada e da Interface de usuário Tangível para interagir com os objetos virtuais gerados. Como resultado, foi desenvolvido um aplicativo que permite manipular com objetos virtuais em um cenário de Realidade Aumentada através de uma câmera de um dispositivo Android, e interagir com os objetos virtuais gerados utilizando objetos reais. Com os testes realizados, foi possível perceber uma infamiliaridade com a usabilidade da interface de usuário tangível, porém todos os usuários conseguiram realizar as tarefas sugeridas e mostraram um grande interesse na aplicação desenvolvida.

Palavras-chave: Interface de usuário tangível. Realidade aumentada. Unity. Vuforia.

ABSTRACT

This work presents the development of an application for the Android Platform that uses the concepts of the Tangible user interface together with an Augmented Reality in an environment that allows manipulation of virtual tree-dimensional objects, allowing an intuitive environment to modify the information of these virtual objects. The Application was developed using the game creation platform Unity, together with the Augmented Reality library Vuforia. Throughout the development of this work, different forms of interactivity with the user, using the Tangible user techniques, therefore interacting and manipulating virtual objects in an Augmented Reality environment. Besides that, there are presented concepts, techniques and components used in the implementation of the Augmented Reality and Tangible user Interface to interact with the generated virtual objects. As a result, it was developed an application that allows manipulating virtual objects in an Augmented Reality scenario through a camera from an Android device and interacting with virtual objects generated using real objects. With performed tests, it was possible to notice an unfamiliarity with the usability of tangible user interface, but all users were able to perform the suggested tasks and showed a great interest in the developed application.

Key-words: Tangible user interface. Augmented reality. Unity. Vuforia.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

RA – Realidade Aumentada

SDK – *Software Development Kit*

SUMÁRIO

1 INTRODUÇÃO

No contexto da Realidade Virtual encontra-se a Realidade Aumentada, sendo configurada como uma modalidade de interface computacional avançada, que busca alcançar a interação humano-computador de uma forma mais natural, misturando em tempo real objetos virtuais tridimensionais gerados por computador com elementos do ambiente físico (RODRIGUES; SATO; BOTEGA, 2012).

A Realidade Aumentada visa simplificar a vida do usuário trazendo informação virtual não só para seus arredores imediatos, mas também para qualquer visão indireta do ambiente do mundo real, tal como transmissão ao vivo em vídeo (CARMIGNIANI; FURHT, 2011 p. 3, tradução nossa).

Entre os vários usos para a Realidade Aumentada, se encontra a Interface de Usuário Tangível. Com esta interface é possível manipular objetos do mundo real e obter resposta no mundo virtual, assim sendo, torna-se possível misturar o virtual e o real tendo um resultado mais natural.

A Interface de Usuário Tangível inovou o conceito de interação entre o usuário e o computador, permitindo manipular seu ambiente físico e interagir com o mundo digital, como Jacko (2012, p. 466, tradução nossa) explica, “A idéia chave da interface tangível é dar forma física para a informação digital. Essa Forma física serve tanto para suas ligações digitais interligadas e associações.”

No contexto de Realidade Aumentada encontra-se o Vuforia, um kit de desenvolvimento de software que torna possível a criação de aplicações de Realidade Aumentada para dispositivos móveis. Entre suas plataformas de desenvolvimento, Vuforia fornece uma extensão para o motor de jogos Unity, que por sua vez facilita o desenvolvimento de aplicações e jogos tridimensionais e bidimensionais.

Diante deste contexto, este trabalho desenvolveu uma Interface de Usuário Tangível utilizando a extensão Vuforia para o motor de jogos Unity e assim utilizando esta Interface de Usuário Tangível para manipular objetos virtuais, podendo associá-los a um marcador e permitir alterar seus atributos.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver uma Interface de Usuário Tangível para manipular objetos tridimensionais virtuais utilizando Realidade Aumentada.

Os objetivos específicos do trabalho são:

a) criar uma Interface de Usuário Tangível que utilize imagens codificadas

(marcadores);

b) possibilitar que objetos tridimensionais virtuais interajam entre si utilizando os marcadores;
c) realizar testes com usuários.

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 é apresentada a operacionalidade do aplicativo desenvolvido e por fim são apresentados os resultados e discussões a respeito do desenvolvimento e uma breve comparação entre este trabalho e os trabalhos correlatos apresentados no capítulo dois. Por fim, o quarto capítulo apresenta a conclusão do trabalho e sugestões para trabalhos futuros.

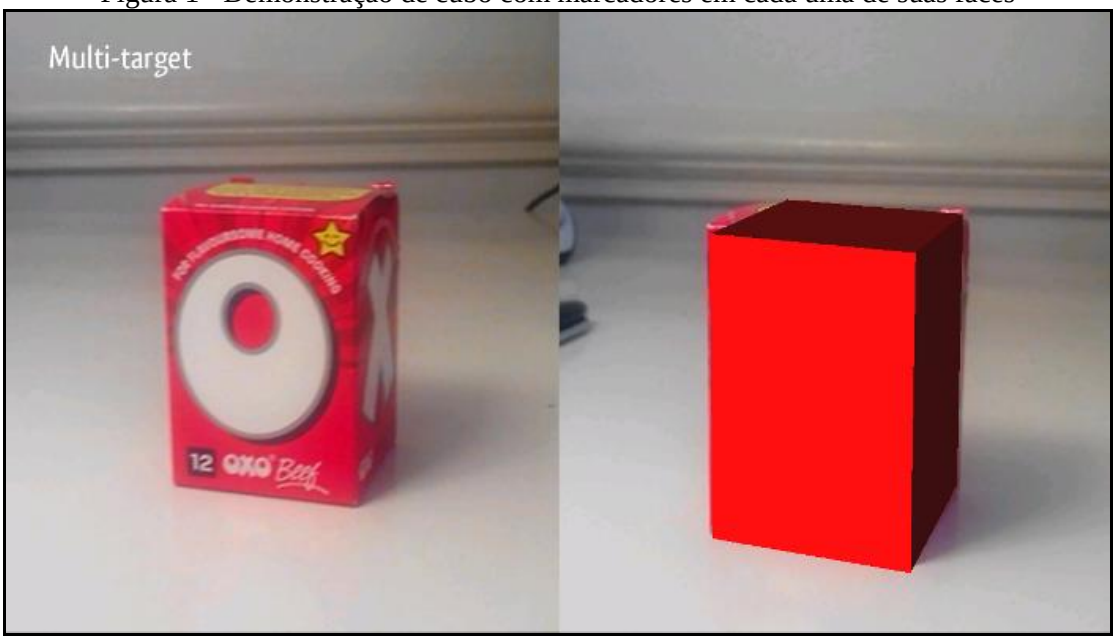
2 FUNDAMENTAÇÃO TEÓRICA

A seção 2.1 apresenta os conceitos básicos da interface tangível. Na seção 2.2 serão apresentados os conceitos da Realidade Aumentada e suas aplicações. Na seção 2.3 será apresentado o motor de desenvolvimento de jogos Unity e suas principais funcionalidades. E finalmente, na seção 2.4 são apresentados os trabalhos correlatos.

2.1 INTERFACES DE USUÁRIO TANGÍVEL

A Interface de Usuário Tangível, conhecida também como interface “palpável”, tem como um dos objetivos, a possibilidade de manipular as informações de um ambiente virtual ou ambiente aumentado utilizando objetos físicos (JACKO, 2012, p. 466). Para quaisquer tipos de interface são necessários dois componentes, sendo eles: Entrada (*input*) e Saída (*output*) de informação. A Entrada é responsável pela entrada e manipulação de informações, podendo ser um cubo com marcadores em cada um de seus lados. A Saída é a representação externa que será percebida pelo ser humano, por exemplo, em um monitor de um computador (JACKO, 2012 p. 468). A Figura 1 apresenta dois objetos em formato de caixa. Na caixa da esquerda são utilizadas imagens distintas das suas faces para representar as Entradas. Na caixa da direita muda-se a Saída utilizando as imagens da Entrada como marcadores, para alterar a percepção do usuário através de uma representação virtual.

Figura 1 - Demonstração de cubo com marcadores em cada uma de suas faces

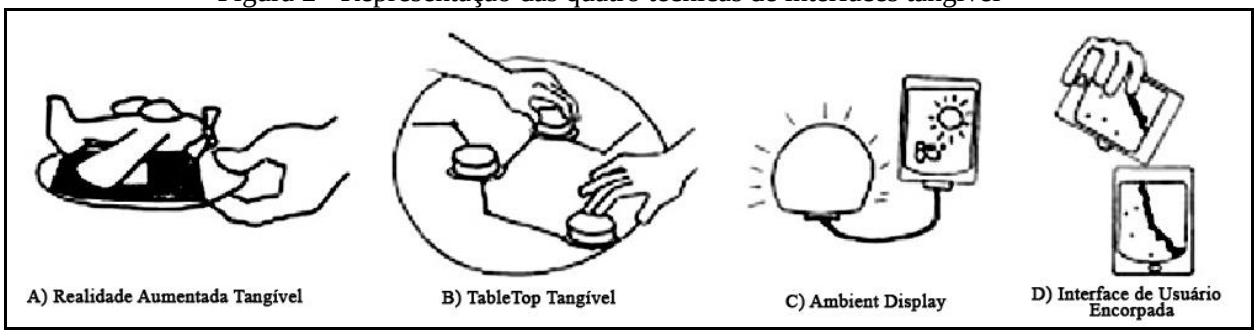


Fonte: Rajawali (2015).

Shaer e Hornecker (2010, p. 13-16) apresentam quatro mecânicas de interação com o usuário. O primeiro é a Realidade Aumentada Tangível que combina a entrada palpável com a

saída em realidade, com objetos virtuais encaixados em objetos físicos os quais o usuário pode manipular utilizando marcadores (Figura 2a).

Figura 2 - Representação das quatro técnicas de interfaces tangível



Fonte: Adaptação de Shaer e Hornecker (2010, p. 16).

A segunda mecânica é a Interação TableTop Tangível, que utiliza as técnicas e tecnologias de interação utilizando como entrada objetos que podem ser lidos a partir de *chips* ou de câmeras e a saída pode ser exibida em projetores ou monitores como pode ser visto na representação da Figura 2b (SHAER; HORNECKER, 2010, p. 13-16).

A terceira mecânica de Interface de Usuário Tangível é conhecida como Ambient Display onde a saída de informação pode ser mostrada em uma mídia Ambiente, como paredes e chão. Na representação da Figura 2c, a informação passada do dispositivo é mostrada por um dispositivo externo, como uma lâmpada (SHAER; HORNECKER, 2010, p. 13-16).

A quarta mecânica é a Interface de Usuário Encorpada em que a interação manual com o objeto que o torna uma interface. Na representação da Figura 2d, o dispositivo móvel é integrado com o conteúdo digital (SHAER; HORNECKER, 2010, p. 13-16).

2.2 REALIDADE AUMENTADA

O objetivo da Realidade Aumentada (RA) é criar a sensação que objetos virtuais estão presentes no mundo real combinando elementos da Realidade Virtual com o mundo real (CAWOOD; FIALA, 2007, p.1). A Realidade Aumentada é uma variação da Realidade Virtual. Enquanto a Realidade Virtual transporta o usuário para um ambiente sintético e sem contato com o mundo real, a Realidade Aumentada amplia o senso de realidade sobrepondo objetos virtuais sobre o mundo real (CARMIGNIANI; FURHT, 2011 p. 3).

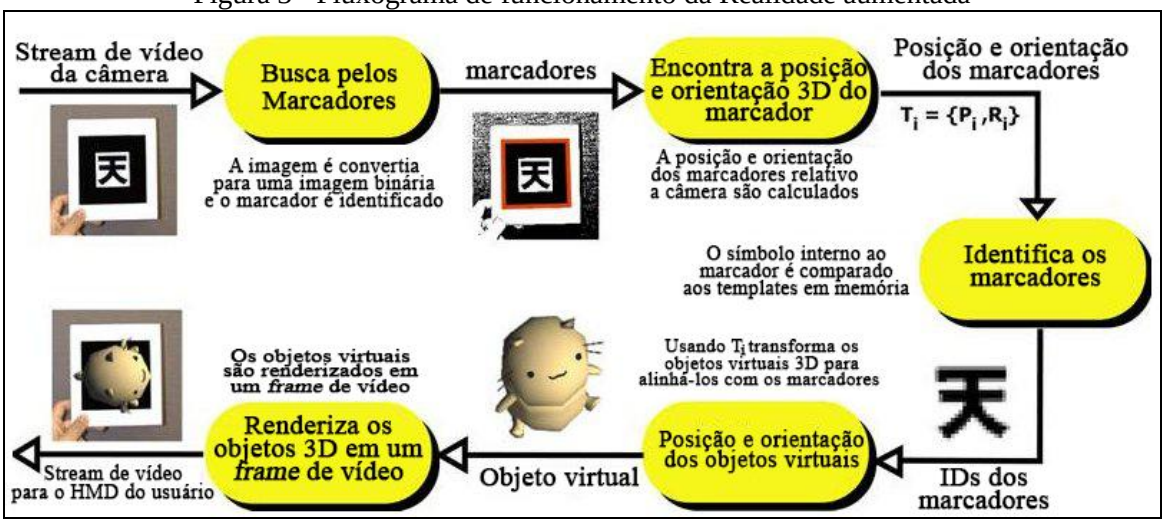
Segundo Cawood e Fiala (2007, p. 1, tradução nossa), “quando objetos virtuais são adicionados em um cenário, eles são conhecidos como *visual AR*. Por definição, Elementos AR não são visíveis por olho nu, então visual AR depende de um tipo de *Display*”. Hoje em dia, está cada vez mais fácil de implementar, graças ao crescente número de aparelhos móveis

com alto processamento, conforme Carmigniani e Furht, (2011, p. 5, tradução nossa), “Hoje em dia, com os novos avanços da tecnologia, uma quantidade crescente de sistemas de RA e aplicação são produzidas [...]”. Para um sistema ser considerado uma Realidade Aumentada, é necessário que possa combinar elementos reais e virtuais em um ambiente real, funcionar de forma interativa e em tempo real e registrar e alinhar objetos reais e virtuais uns com os outros (ZANDOMENEGUI, 2014, p. 268).

2.2.1 Rastreamento e Marcadores

Entres os dispositivos para aplicação de Realidade Aumentada, um dos mais populares e de baixo custo são as Webcams, isso graças ao poder de processamento de microcomputadores e às técnicas de rastreamento (KIRNER; SISCOUTTO, 2007, p.13). De acordo com ZANDOMENEGUI (2014, p. 271), “Para que as imagens do mundo real se sobreponham corretamente, é necessário que uma câmera funcione como um sensor, rastreando ininterruptamente um ou mais sinais, sejam estes naturais ou construídos” como demonstrado na Figura 3.

Figura 3 - Fluxograma de funcionamento da Realidade aumentada



Fonte: adaptação de Artoolkit (2012b).

Primeiramente a câmera captura o vídeo e envia para o computador, então o computador analisa cada *frame* de vídeo procurando pelo marcador e, uma vez encontrado, calcula-se a posição da câmera em relação ao mesmo. Após esta etapa, é identificado o marcador para definir qual objeto virtual deve ser desenhado no monitor (ARTOOLKIT, 2012a). Há outros tipos de indicadores para posicionar os objetos virtuais no mundo real. Porém, como explicado por Cawood e Fiola (2007, p. 4-5), “A forma mais simples de marcador é um modelo único que é visível para a câmera de Realidade Aumentada [...]”,

dentre estes marcadores está o *Quick Response Code* (*QRCode*), por possuir um código único que pode ser reconhecido pela câmera (CARMIGNIANI; FURHT, 2011, p. 340).

2.2.2 Visualização

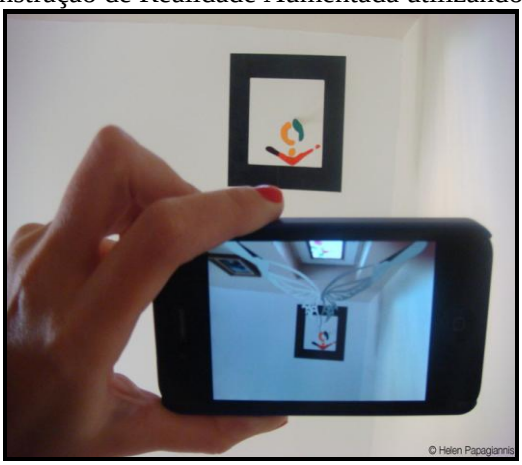
A visão computacional que gera os elementos virtuais tridimensionais utiliza a mesma perspectiva que é obtido pelo dispositivo de captura, após a geração dos objetos virtuais, o resultado é gerado para um monitor para a visualização do usuário.

Há duas técnicas comuns para a visualização da Realidade Aumentada, sendo elas o *Magic Mirror* e o *Magic Lens*. O *Magic Mirror*, ou Espelho Mágico, envolve colocar um monitor atrás da área capturada pela câmera, assim dando a impressão de espelho para o usuário, com a câmera e o retorno visual apontada para o Usuário. Como na Figura 4, o aplicativo AR Magic Mirror (APPLE, 2011), em que a câmera e o monitor estão virados para o usuário. Já a técnica *Magic Lens*, do português Lentes Mágicas, o monitor fica à frente do usuário enquanto a câmera pode estar posicionada atrás do monitor ou em outra posição, como mostrado na Figura 5, possibilitando ao usuário ver uma representação do mundo com realidade aumentada (CAWOOD; FIALA, 2007, p. 3-4). Diferente da técnica de Espelho Mágico, a técnica de Lentes Mágicas geralmente depende do uso de marcadores para a geração de seu ambiente de Realidade Aumentada, com o monitor direcionado ao usuário, esta técnica se tornou a mais utilizada em aplicações de Realidade Aumentada por mostrar maior imersão ao usuário no ambiente.

Figura 4 - Aplicativo AR Magic Mirror

Fonte: Apple (2011).

Figura 5 - Demonstração de Realidade Aumentada utilizando um *Smartphone*



Fonte: Papagiannis (2010).

2.3 UNITY E VUFORIA

Com o emergente mercado de jogos e visando nivelar o campo de jogos para desenvolvedores, a Unity Technologies desenvolveu uma plataforma de desenvolvimento de conteúdo interativo 3D e 2D (UNITY, 2015b). Sendo flexível e eficiente, "O mecanismo de jogos Unity é o mais popular entre os desenvolvedores do que qualquer outro software de desenvolvimento de jogos" (UNITY, 2015b).

A plataforma Unity possui um editor ampliável contendo vários recursos prontos para desenvolvedores, isso incluindo editores de animações, *scripts*, áudio, gráfico, física 2D e 3D, sendo possível customizá-los completamente. Graças a estas ferramentas disponibilizadas para os desenvolvedores e o mecanismo ampliável do editor, Unity se tornou uma plataforma de desenvolvimento de grande interesse (UNITY, 2015a).

Para a aprendizagem do uso da ferramenta, são disponibilizados manuais, guias e uma ampla documentação, e é possível aprender com especialistas em uma sessão on-line ao vivo disponibilizado pela empresa. Possuindo uma vasta comunidade ativa, a plataforma Unity se destaca por sua facilidade de aprendizagem. Possuindo uma loja virtual para o *download* de conteúdo gratuito e pago, é possível obter de forma simples objetos virtuais, áudios, extensões, entre outros conteúdos para facilitar ao desenvolvedor criar seus jogos (UNITY, 2015c).

De modo a ampliar as funcionalidades de seu editor utilizando extensões, a empresa Qualcomm desenvolveu a SDK Vuforia para desenvolvimento de aplicações em Realidade Aumentada. Vuforia consegue identificar diferentes tipos de marcadores. É possível detectar imagens com o *Image Target*, um marcador fiducial especial chamado *frame markers*,

marcadores de forma cilíndrica utilizando o *Cylinder Targets*, e é possível detectar objetos 3D reais utilizando o *Objetct Targets* (QUALCOMM, 2015).

Para detectar imagens, a SDK Vuforia faz uma comparação através da câmera com o banco de dados na nuvem ou com o banco interno da aplicação, conhecida como *Image Target*. Após a imagem ser identificada, a aplicação executa a função associada à mesma. Assim como *Image Target*, a SDK Vuforia possui a habilidade de reconhecer marcadores especiais conhecidos como *frame markers*. Estes marcadores fiduciais são mais leves e mais rápidos para serem reconhecidos, pois, possuem uma identificação única codificada com um padrão binário ao longo de suas bordas, como pode ser observado na Figura 6 (QUALCOMM, 2015).

Figura 6 - Marcador do tipo *frame*

Fonte: Qualcomm (2015).

Para a detecção de objetos cilíndricos, a SDK Vuforia utiliza o *Cylinder Targets*. Reconhecendo parte da imagem, é possível identificar qual imagem está associada ao objeto cilíndrico. Para a detecção de objetos tridimensionais reais, é utilizado o *Object Target*, em que um objeto de forma quadrada representa um marcador, podendo ser uma *Image Target* ou um *frame marker*, após o reconhecimento do marcador é calculado a sua posição no espaço virtual e a posição de seus lados não visíveis (QUALCOMM, 2015).

2.4 TRABALHOS CORRELATOS

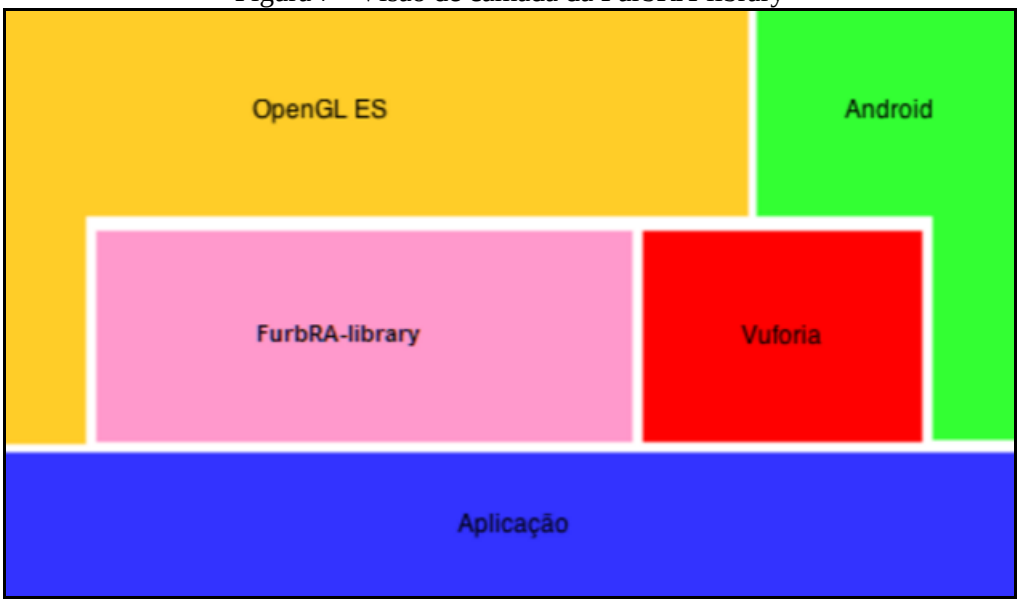
A seguir são apresentados três trabalhos correlatos. A seção 2.4.1 demonstra a biblioteca desenvolvida por Lira (LIRA, 2015). A seção 2.4.2 demonstra o trabalho acadêmico desenvolvido por Rodrigues, Sato e Botega (RODRIGUE; SATO; BOTEGA, 2012). A seção 2.4.3 demonstra o artigo desenvolvido por Lee et al. (2004). Por fim, a seção 2.4.4 demostra uma comparação das características mais importantes entre os trabalhos correlatos.

2.4.1 Biblioteca para desenvolvimento de jogos multijogadores de Realidade Aumentada para Android

Este trabalho, conhecido também como FurbRA-library, desenvolvido por Lira (2015) é uma biblioteca para plataforma Android. O objetivo do trabalho é fornecer funções que possam permitir o desenvolvimento de jogos multijogadores utilizando as propriedades básicas de Realidade Aumentada (LIRA, 2015, p. 97).

Esta biblioteca é uma camada intermediária que agrupa quatro áreas necessárias em uma biblioteca de Realidade Aumentada para jogos multijogadores, sendo elas: renderização gráfica, Realidade Aumentada, comunicação entre dispositivos e multimídia. A biblioteca FurbRA-library utiliza a biblioteca OpenGL ES para a renderização gráfica, a SDK Vuforia para a criação da Realidade Aumentada, o *framework* Android para a comunicação e multimídia. Esta arquitetura pode ser vista na Figura 7 (LIRA, 2015, p. 32-33).

Figura 7 - Visão de camada da FurbRA-library



Fonte: Lira (2015).

Utilizando a biblioteca OpenGL ES para desenvolvimento de conteúdo gráfico e a biblioteca Vuforia, foi possível criar uma aplicação demonstrativa denominada de Animais (LIRA, 2015, p. 77). Esta aplicação demonstrativa tem como propósito auxiliar a alfabetização de crianças e mostrar as letras em forma de LIBRAS. A Figura 8 apresenta a aplicação Animais. Nesta aplicação o usuário deve posicionar os marcadores com letras em uma determinada ordem para acertar o animal mostrado no dispositivo. É possível também ouvir qual é o animal clicando na tela do dispositivo móvel. Com esta aplicação demonstrativa também é possível a conexão via *bluetooth* de dois dispositivos Android para poderem competir.

Figura 8 - Aplicação Animais

Fonte: Lira (2015).

2.4.2 Integrando Interface Tangível com Técnicas de Realidade Aumentada para Ampliar a Experiência Interativa do Usuário

Desenvolvido por Rodrigues, Sato e Botega (2012) tem como objetivo apresentar a união entre as técnicas de Realidade Aumentada com os elementos de interface de usuário tangível *tabletop* para demonstrar uma experiência mais interativa e realista. Também é possivel modificar o tamanho e comportamento dos marcadores exibidos pelo *tabletop* utilizando a funcionalidade *touchscreen.*

Com o uso de um dispositivo móvel para exibir os modelos tridimensionais virtuais como mostra a Figura 9, foi possível demonstrar uma imersividade maior entre o usuário e o computador. Foram utilizados objetos reais com marcadores de Realidade Aumentada que era lido pelo *tabletop*, assim podendo interagir com a interface de forma pré-definida.

Figura 9 - Representação de um cenário utilizando uma interface tangível

Fonte: Rodrigues, Sato e Botega (2004).

Para desenvolvimento desta interatividade, foi utilizada a Interface de Programação de Aplicativos (API) AndAR para Realidade Aumentada e integrando-a com técnicas de Interface de Usuário Tangível. Usou-se eventos de toque na tela do dispositivo móvel para

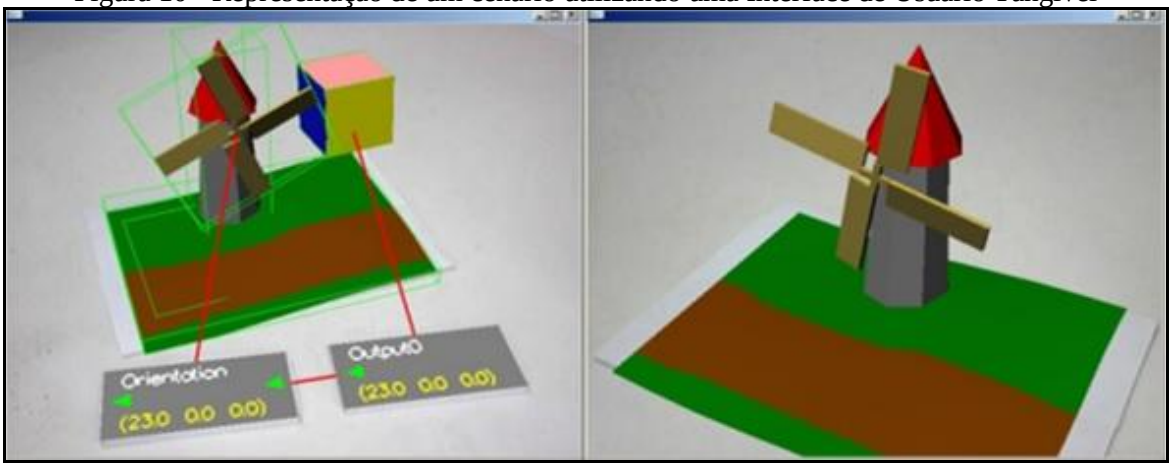
interação com os objetos 3D virtuais e eventos de toque com a interface *tabletop* para interação com os marcadores (RODRIGUES; SATO; BOTEGA, 2012).

2.4.3 Immersive Authoring of Tangible Augmented Reality Applications

Em 2004 foi proposto o desenvolvimento de uma Interface de Usuário Tangível (LEE et al., 2004, p. 1). Foi utilizada uma técnica conhecida como *Immersive Authoring*, que permite desenvolver, associar especificações e testar durante o processo de execução, assim criando um sistema mais rápido e de aprendizagem mais fácil e intuitiva. A aplicação mostrada neste artigo utilizou a biblioteca ARToolKit para facilitar o seu desenvolvimento graças ao seu cálculo de posição e orientação da câmera em relação aos marcadores fiduciais (LEE et al., 2004, p. 1).

O objetivo desta aplicação é abordar de uma forma diferenciada as aplicações de Realidade Aumentada Tangível por Autoria. Utilizando um Sistema de Autoria e os conceitos de Realidade Aumentada, usuários sem experiências na área de programação poderão desenvolver aplicação de Realidade Aumentada, como explicado por Lee et al. (2004, p. 8). A Figura 10 utiliza objetos virtuais associados a marcadores fiduciais para construção de um cenário virtual. Na imagem representada à esquerda, o marcador (que está conectado à hélice do moinho virtual) está alterando a orientação do objeto virtual de acordo com as informações de saída do segundo marcador, que está conectado ao cubo virtual. A imagem representada à direita é possível visualizar o cenário virtual criado sem o uso dos marcadores, o cubo virtual não é visível no cenário final por este objeto ser um objeto lógico, funcionando como um motor adicionando o valor gerado à hélice virtual, com isso, este motor não é visível e é obtido um cenário em que é possível visualizar um moinho virtual com sua hélice em movimentação.

Figura 10 - Representação de um cenário utilizando uma Interface de Usuário Tangível



Fonte: Lee et al. (2004).

Com o uso desta aplicação, um público juvenil que nunca teve contato com modelagem tridimensional ou Realidade Aumentada, conseguiu criar um cenário com facilidade (LEE et al., 2004, p. 8).

2.4.4 Comparativo entre os trabalhos correlatos

O Quadro 1 apresenta um comparativo entre as características mais relevantes dos trabalhos correlatos apresentados nessa seção.

Quadro 1 - Comparativo entre os trabalhos correlatos

Características/Trabalhos Relacionados	Lira (2015)	Rodrigues, Sato e Botega (2012)	Lee et al. (2004)
possui Interface de Usuário Tangível		X	X
criação de RA baseada em marcadores	X	X	X
permite renderização de objetos 3D virtuais	X	X	X
detecção de múltiplos marcadores	X	X	X
utiliza o SDK Vuforia	X		
interação entre objetos virtuais			X

Através das informações demonstradas no Quadro 1, percebe que o trabalho de Lira (2015), diferente dos outros trabalhos, não possui Interface de Usuário Tangível, entretanto, assim como o este trabalho desenvolvido, utiliza a SDK Vuforia. No trabalho de Rodrigues, Sato e Botega (2012) e no trabalho de Lee et al. (2004), nota-se o uso da Interface de Usuário Tangível. Percebe-se também que o este trabalho possui uma maior similaridade com trabalho de Lee et al. (2004), exceto pelo uso da SDK Vuforia. Em todos os trabalhos nota-se que são criados objetos tridimensionais virtuais utilizando marcadores que podem ser detectados simultaneamente.

3 DESENVOLVIMENTO

Neste capítulo são apresentadas as etapas de desenvolvimento do aplicativo. 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 os seguintes requisitos:

a) utilizar uma imagem como marcador para geração da biblioteca de objetos virtuais (Requisito Funcional - RF);
b) utilizar um cubo com seis imagens como marcador para selecionar e movimentar os objetos gerados pela biblioteca (RF);
c) utilizar uma imagem como marcador para inserir o objeto virtual selecionado com o cubo no ambiente virtual (RF);
d) utilizar botões virtuais no marcador de biblioteca para selecionar o objeto virtual (RF);
e) utilizar uma imagem como marcador para selecionar um atributo do objeto virtual (RF);
f) utilizar uma imagem como marcador para modificar o valor de um atributo do objeto virtual (RF);
g) ser disponibilizado para a plataforma Android (Requisito Não Funcional - RNF);
h) utilizar a SDK Vuforia para a implementação dos recursos de RA (RNF);
i) utilizar a linguagem C# para definir os *scripts* do aplicativo (RNF);
j) utilizar o motor de jogos Unity para a implementação do espaço 3D (RNF).

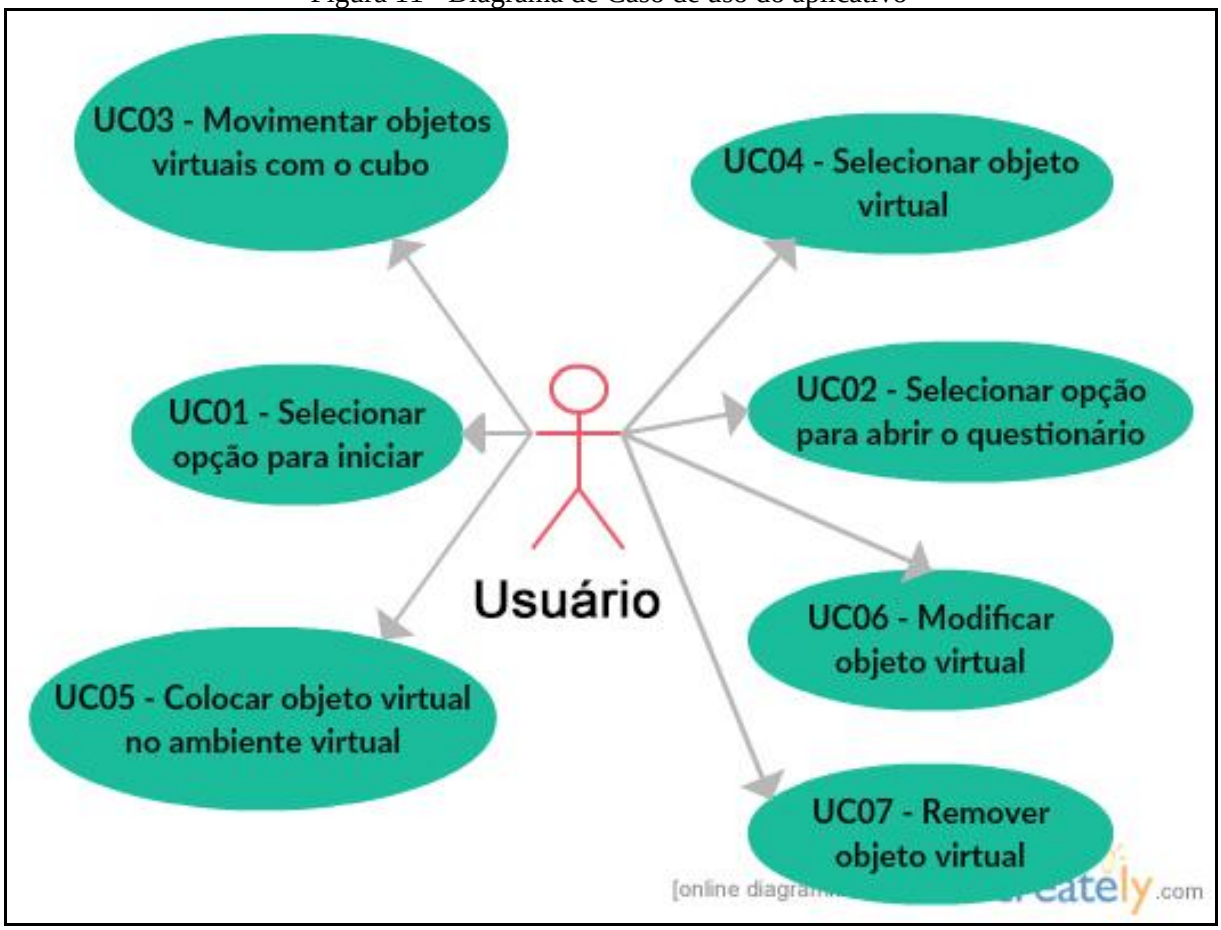
3.2 ESPECIFICAÇÃO

A especificação do aplicativo foi desenvolvida seguindo a análise orientada a objetos, utilizando a *Unified Modeling Language* (UML) em conjunto com a ferramenta Creately.com para a elaboração dos casos de uso do diagrama de classes e dos diagramas de atividade.

3.2.1 Diagrama de Caso de uso do aplicativo

Nesta seção são apresentados os casos de uso do aplicativo, ilustrados na Figura 11. Identificou-se o ator `Usuário` que utiliza o aplicativo. Do Quadro 15 até o Quadro 21 são descritos os casos de uso do ator `Usuário`.

Figura 11 - Diagrama de Caso de uso do aplicativo



No diagrama de caso de uso apresentado, o caso de uso UC01 - Selecionar opção para iniciar, o usuário pode tocar em um botão presente na tela inicial para iniciar a aplicação de Realidade Aumentada. No caso de uso UC02 - Selecionar opção para abrir o questionário, o Usuário pode tocar em um botão presente na tela inicial para abrir a página do questionário. No caso de uso UC03 - Movimentar objetos virtuais com o cubo, o Usuário pode utilizar um cubo real para movimentar objetos virtuais. No caso de uso UC04 - Selecionar objeto virtual, o Usuário pode selecionar objetos virtuais e selecionar um de seus aributos. No caso de uso UC05 - Colocar objeto virtual no ambiente virtual, o Usuário pode posicionar um objeto virtual no ambiente virtual utilizando marcadores, assim o objeto virtual poderá interagir com outros marcadores. No caso de uso UC06 - Modificar objeto virtual, o Usuário pode modificar a escala, posição ou rotação do objeto virtual utilizando marcadores. No caso de uso UC07 - Remover objeto virtual, o Usuário pode remover o objeto virtual selecionado do ambiente virtual. O detalhamento dos casos de uso do protótipo está no Apêndice A.

3.2.2 Diagrama de classes do aplicativo

O diagrama de classes da Figura 12, 13, 14 e 15 apresenta as classes desenvolvidas para a criação do aplicativo.

Figura 12 - Classes desenvolvidas para o aplicativo

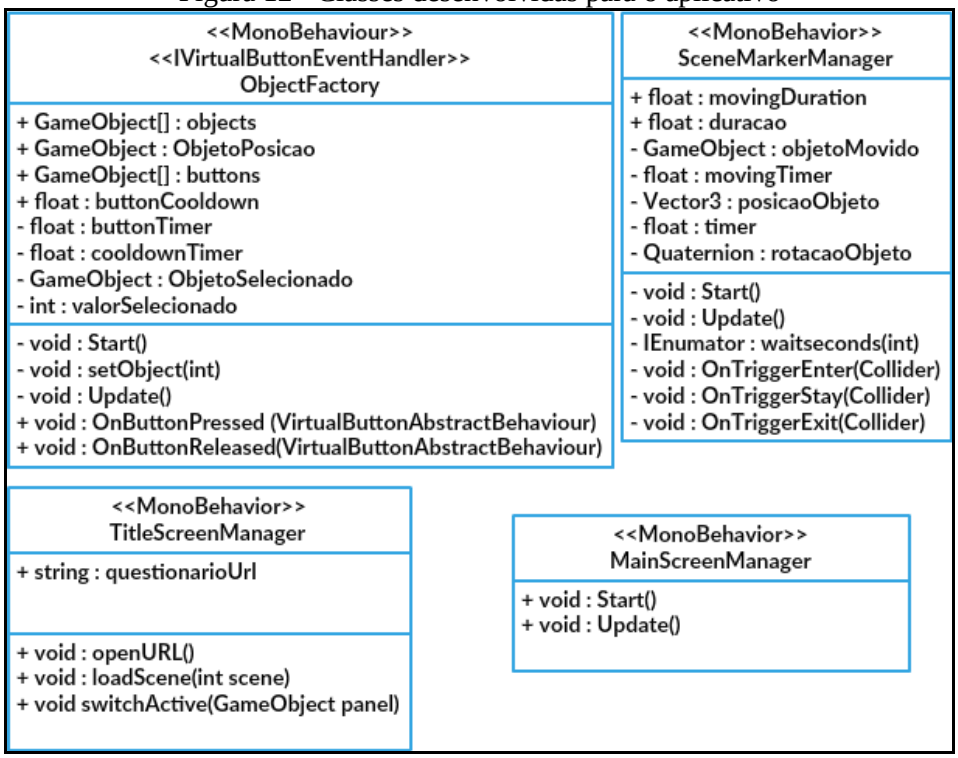


Figura 13 - Classes desenvolvidas para o aplicativo

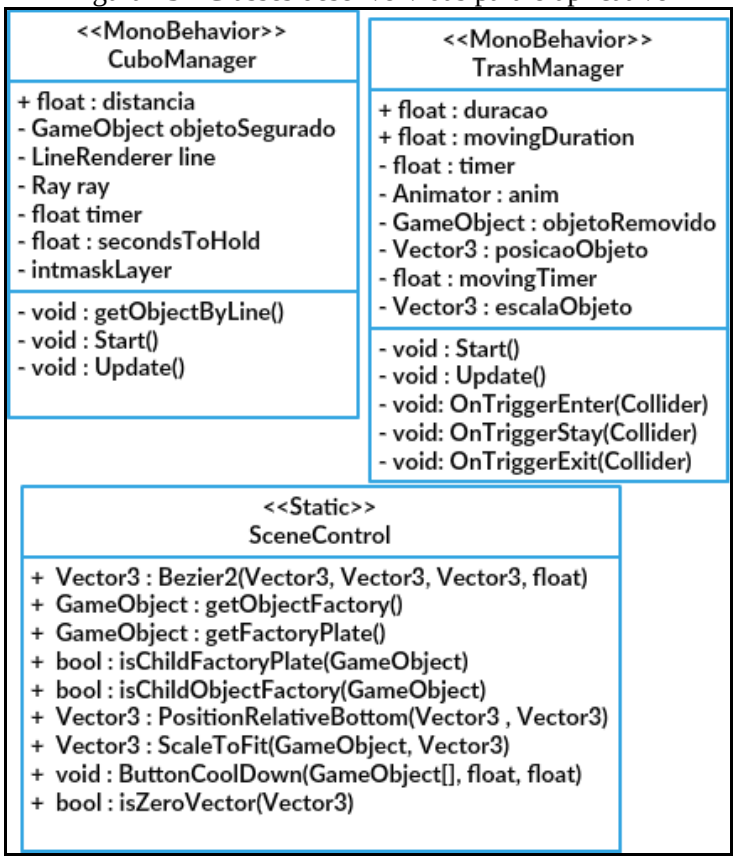


Figura 14 - Classes desenvolvidas para o aplicativo

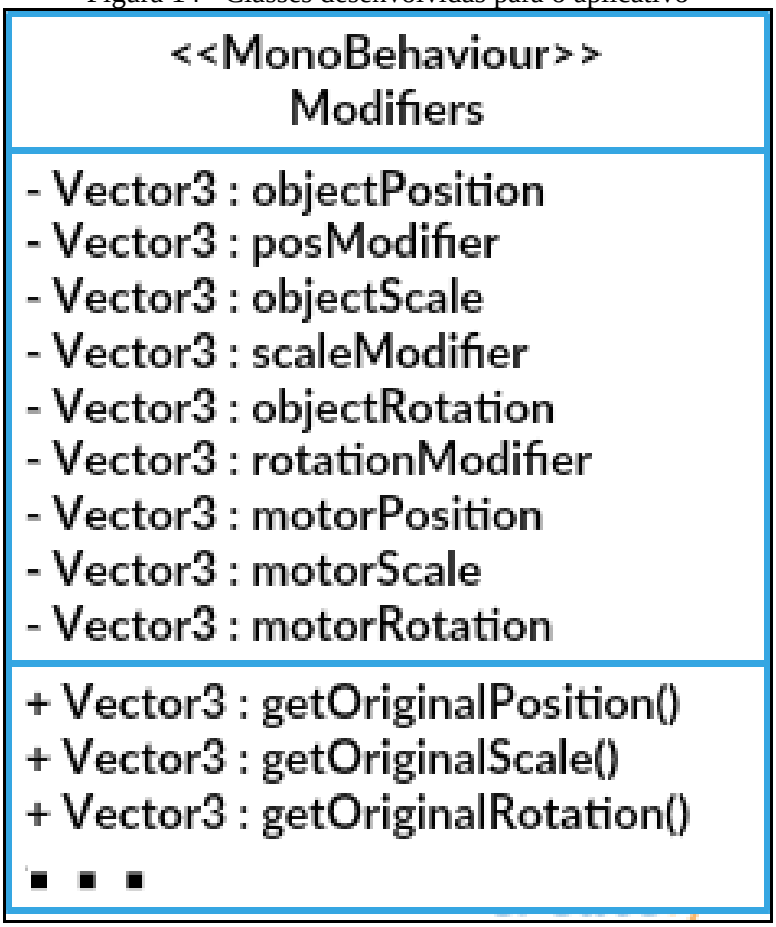
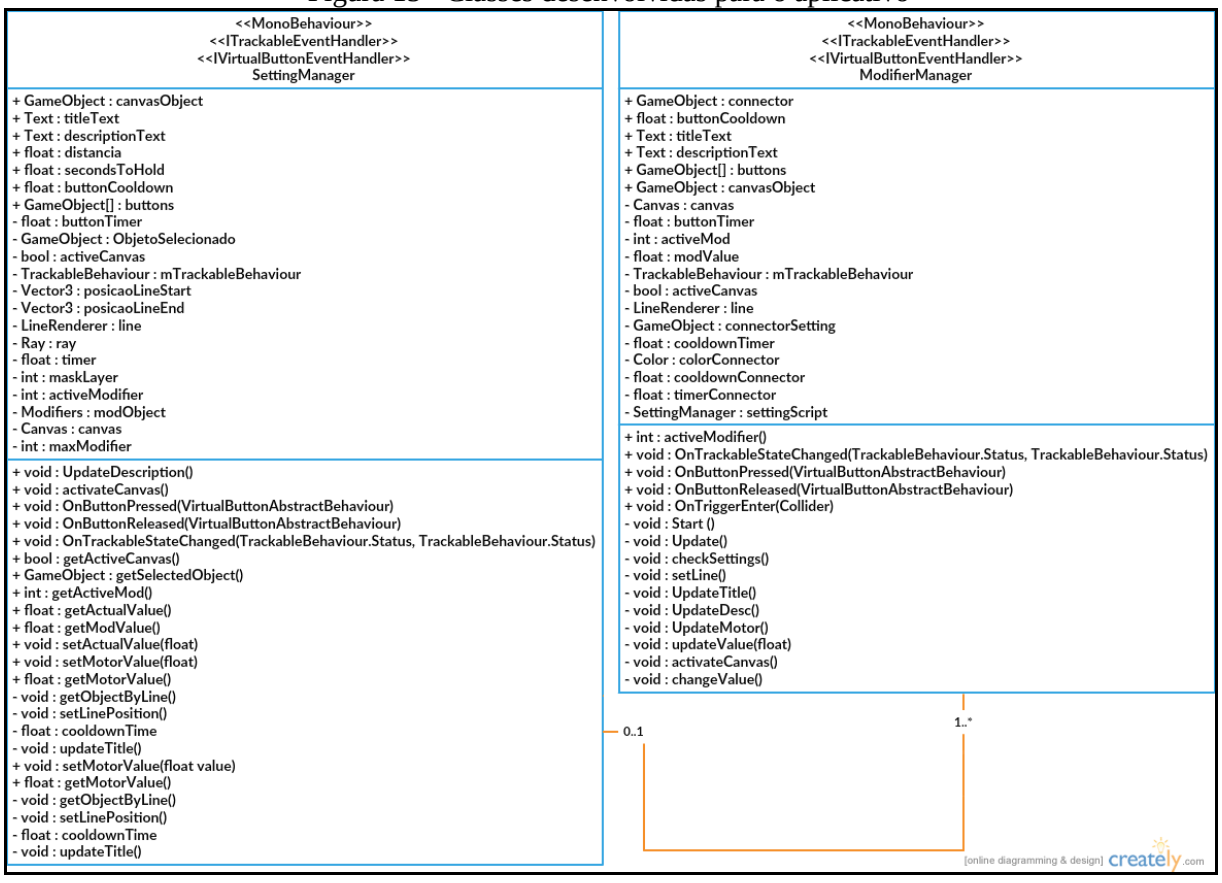


Figura 15 - Classes desenvolvidas para o aplicativo



A classe estática `SceneControl` disponibiliza funções utilitárias que são utilizadas por outras classes, como o método de verificar se um vetor está zerado. Assim como as outras classes, esta classe possui o objeto `Vector3` que é fornecida pela API do Unity que tem o propósito de fornecer a posição e a rotação de um ponto no espaço tridimensional, a API do Unity também fornece a classe `GameObject` que é a base de todos os objetos presentes em um cenário.

A classe `ObjectFactory` é responsável pela criação e manutenção dos objetos que poderão ser selecionados pelo usuário. Ao pressionar os botões, os objetos exibidos pelo objeto `ObjectFactory` serão destruídos e substituídos pelo próximo `GameObject` na `array` da variável `objects`. Os botões são ativados pelos métodos `OnButtonPressed` e `OnButtonReleased` que são implementados pela classe `IVirtualButtonEventHandler` da SDK Vuforia. O métodos `setObject` é responsável por alterar o objeto gerado pela biblioteca de objetos.

A classe `CuboManager` é responsável pelo funcionamento do cubo ou cursor real, desenhando a linha virtual para obter o objeto virtual. Esta classe possui a classe `Ray` que pertence a API do Unity e permite criar uma linha infinita apartir de um ponto de origem e de uma direção, esta classe é utilizada em conjunto do `Raycast` para adquirir `GameObject` que entram em colisão com esta linha. No método `getObjectByLine` é criado um `Raycast` que adquire o objeto virtual na distância informada pela variável `distancia`.

A classe `TrashManager` é responsável pelo funcionamento de remoção de objeto virtuais. Esta classe possui uma variável do tipo `Animator` que é fornecido pela API Unity para controlar o mecanismo de animação dos objetos virtuais. Se um objeto virtual entrar no `BoxCollider` do objeto e que não tem o objeto `ObjectFactory` como parente ou não possuir a `tag Guides`, este objeto será destruído após a duração definida pela variável `duracao` e após a animação.

A classe `SceneMarkerManager` é responsável por inserir o objeto virtual selecionado pelo cubo no cenário de Realidade aumentada. Após o objeto colidir com o `BoxCollider` do objeto por meio do método `OnTriggerEnter`, sendo que o objeto não tenha o `ObjectFactory` como parente e não possuir a `tag Guides`, este objeto será reposicionado acima do marcador e terá o `objectFactory` como parente.

A classe `Modifiers` é responsável pela alteração dos objetos virtuais que podem ser manipulados pelo usuário. Esta classe disponibiliza o método `setValues` que é responsável por fixar os valores atuais nas variáveis `objectPosition`, `objectRotation` e `ObjectScale`,

e o método `resetModPosition` que é responsável por zerar os valores modificados do objeto virtual associado a essa classe. Esta classe também disponibiliza as funções para fixar os valores modificados para a posição, rotação e escala do objeto virtual assim como `getters` e `setters` para cada um de seus atributos.

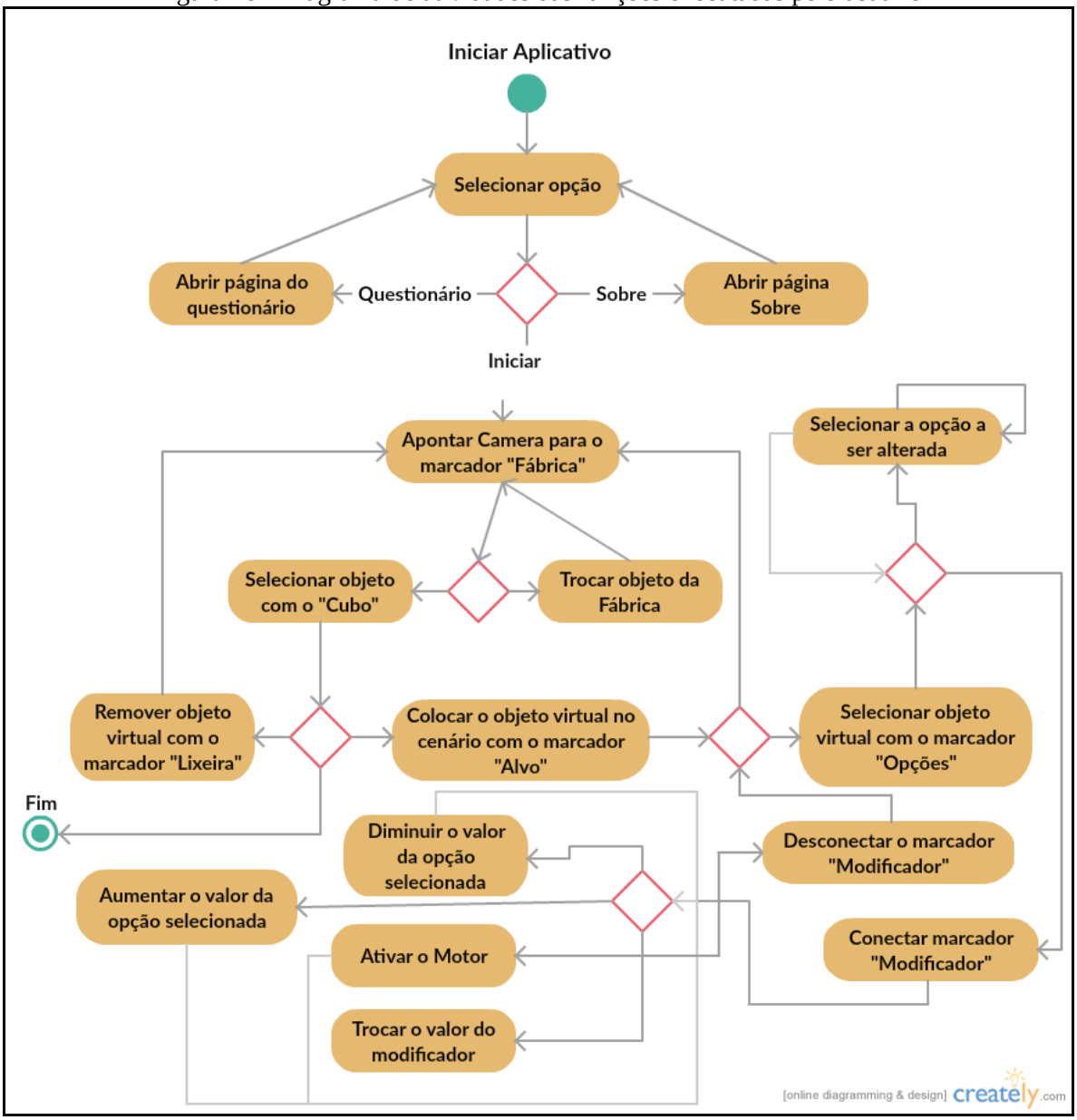
A classe `SettingManager` é responsável pela seleção de um objeto virtual no cenário de Realidade Aumentada assim como é responsável pelo controle dos botões do objeto `SettingMarker`. Esta classe fornece o método `getActiveCanvas` que retorna um `bool` caso o objeto `canvas` esteja visível. Esta classe também fornece o método `getActiveMod` um `integer` associado ao valor a ser modificado no objeto virtual selecionado. Já a classe `ModifierManager` é responsável por alterar os valores da classe `Modifier` associado a um objeto virtual. A classe possui uma instancia de `SettingManager` para obter o valor a ser modificado. E por fim, a classe `TitleSceneManager` é responsável pelas ações dos botões da tela inicial do aplicativo.

A classe `MainSceneManager` é responsável pela funcionalidade de retornar á tela inicial ao pressionar o botão voltar no dispositivo móvel.

3.2.3 Diagrama de atividades

A Figura 16 apresenta o diagrama de atividades das funções executadas pelo usuário no aplicativo. Suas atividades incluem selecionar objetos virtuais, selecionar e movimentar os objetos virtuais com o cubo e poder alterar as propriedades deste objeto virtual.

Figura 16 - Diagrama de atividades das funções executadas pelo usuário



A Figura 16 apresenta o ciclo de funções do sistema, o qual é responsável por controlar as funcionalidades da tela de título e das funcionalidades da interface tangível e tendo como retorno a exibição de objetos virtuais tridimensionais na tela do dispositivo móvel.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas para a implementação do aplicativo, junto com a descrição de suas operacionalidades da implementação, respectivamente seção 3.3.1 e seção 3.3.2.

3.3.1 Técnicas e ferramentas utilizadas

O aplicativo foi desenvolvido no *framework* Unity na versão 5.3.5f1 *Personal* em conjunto com a IDE Microsoft Visual Studio Community 2015, utilizando a linguagem C Sharp.

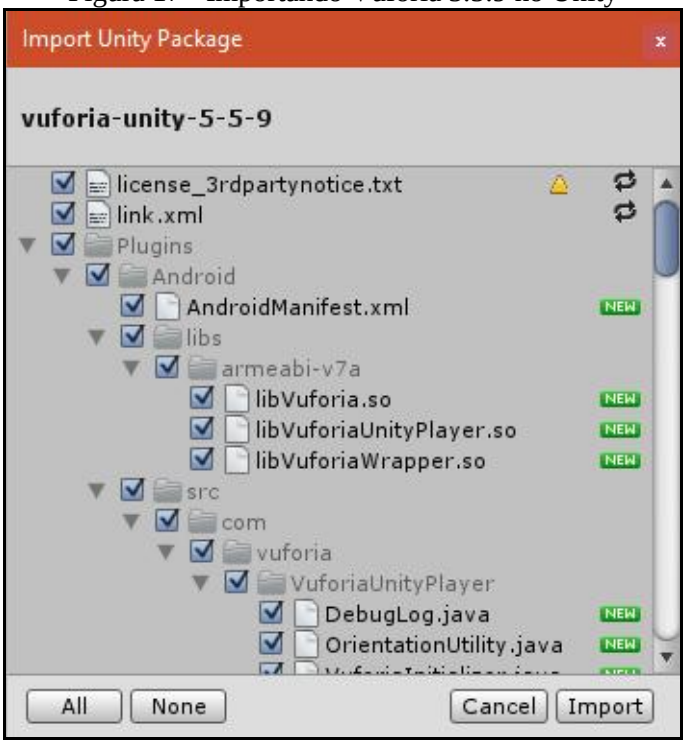
Para a criação da Realidade Aumentada, foi utilizado o SDK Vuforia na versão 5.5.9 integrado ao Unity. Para a criação dos marcadores foi utilizado o programa Photoshop CS6.

Para a execução e a realização dos testes de todo o desenvolvimento do trabalho, utilizou-se um *smartphone* Samsung Galaxy J7 com o sistema operacional Android 5.1.1, possuindo 1,5 GB de memória RAM e um processador 8 Core de 1.5 GHz.

3.3.1.1 Início do projeto

Para dar início ao projeto no Unity, é necessária a importação do SDK do Vuforia para o Unity, disponibilizado na página de desenvolvedores do Vuforia. A importação pode ser feita clicando na aba *Assets*, selecionar a opção *Import Package* e clicar em *Custom Package*. Após isso é aberto uma janela para selecionar um arquivo, após selecionar o pacote do SDK do Vuforia e clicar em Abrir, uma janela com todos os arquivos a serem importados é aberta (Figura 17).

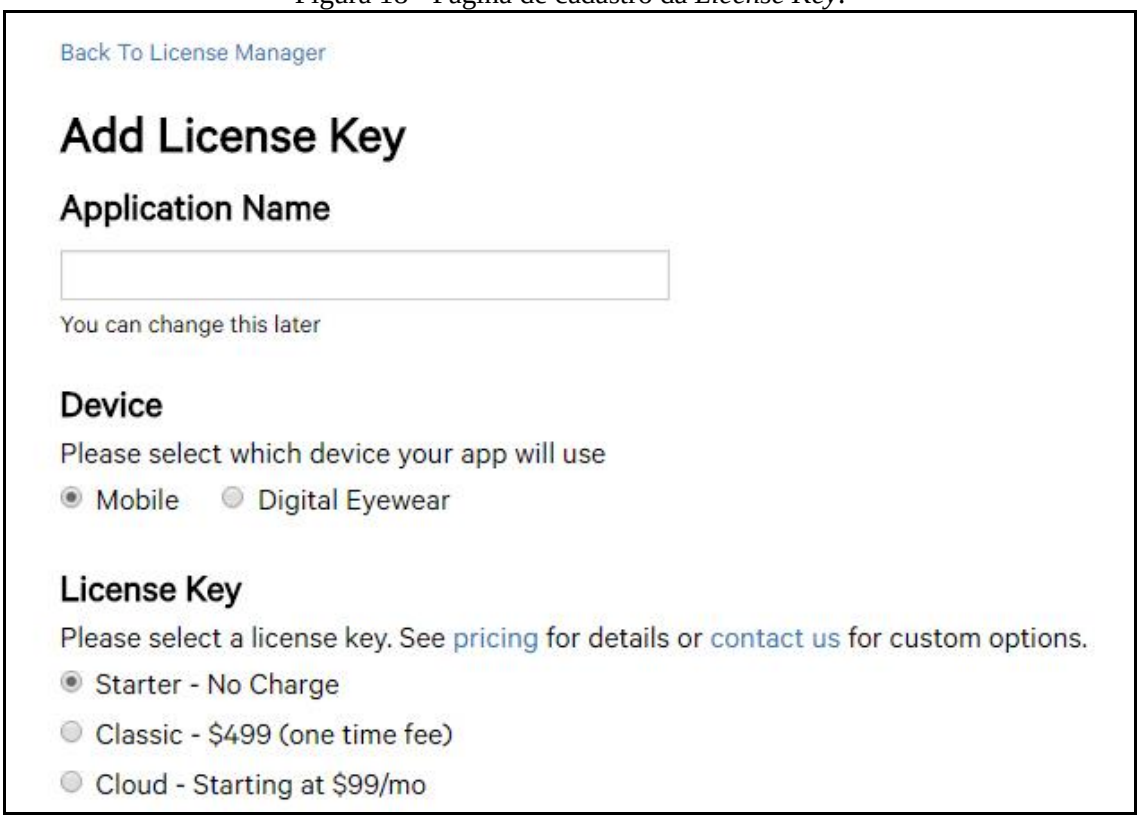
Figura 17 - Importando Vuforia 5.5.9 no Unity



Após a importação do SDK Vuforia para o Unity, é necessário fazer um cadastro na página de desenvolvedor do Vuforia para obter a chave de licença para o uso no Unity. Para

cadastrar uma chave de licença, é necessário ir à página *Develop* e clicar no botão *Add License Key*, após isso deve ser inserido um nome para um aplicativo, escolher o tipo de dispositivo e escolher um tipo de licença como mostrado na Figura 18.

Figura 18 - Página de cadastro da *License Key*.

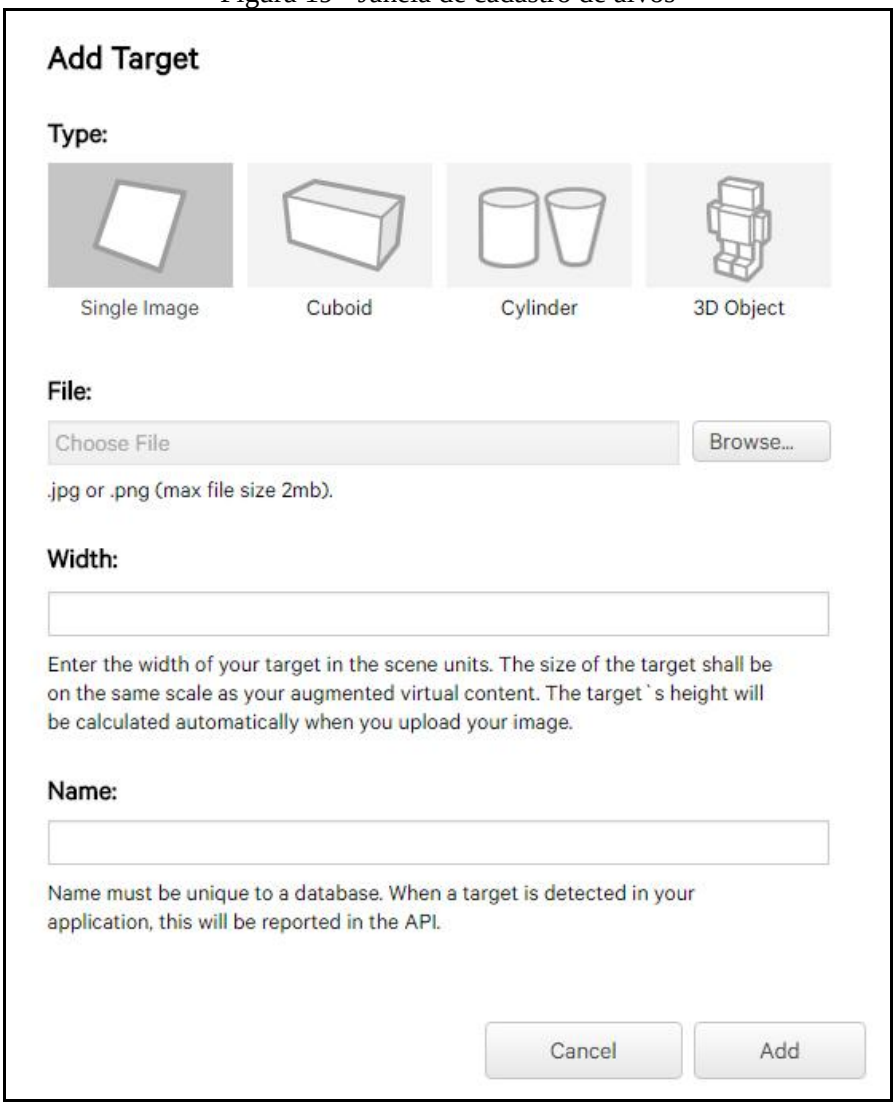


Depois de feito o cadastro da *license key* e adicionado ao `Prefab ARCamera`, cadastra-se uma base de alvo, para isso é necessário acessar a página *Target Manager*. Nesta página é preciso clicar no botão *Add Database*, uma janela irá abrir para que seja inserido o nome da base de alvos e para escolher o tipo da base. O tipo *Device* é utilizado para acessar os alvos diretamente pelo aplicativo e o tipo *Cloud* para acessar os alvos pela internet (para o aplicativo desenvolvido foi nomeado VisEdu e o tipo utilizado foi *Device*).

Depois do cadastro da base de alvos é necessário acessá-la pela lista de *Database*, após isso, cadastra-se um marcador que pode ser acessado pelo botão *Add Target* uma janela irá aparecer para o cadastro do alvo.

Nesta janela mostrada na Figura 19 é necessário escolher o tipo do alvo (na aplicação desenvolvida foi utilizado os tipos *Single Image* e *Cuboid*), o arquivo de imagem que será o marcador utilizado, a largura da imagem e o nome do marcador. Tendo sido realizado o cadastro do alvo, a página irá disponibilizar a classificação da imagem baseando nas características específicas da imagem selecionada. Quanto maior a classificação, melhor será a detecção do marcador na aplicação.

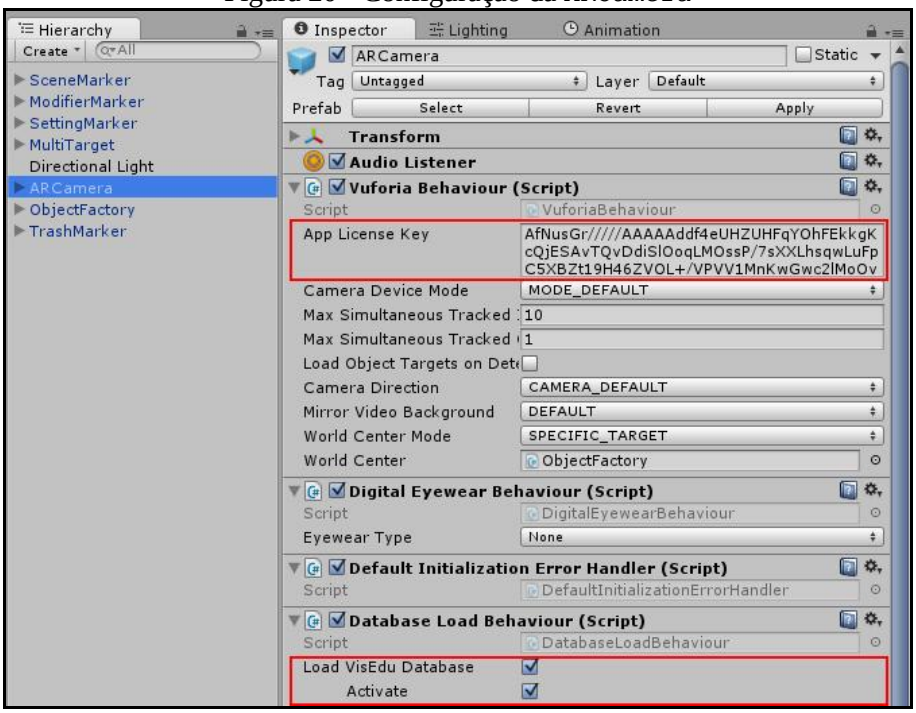
Figura 19 - Janela de cadastro de alvos



Após o cadastro dos marcadores que serão utilizados, deve se exportar para o Unity clicando no botão *Download Database (All)* e escolher a plataforma que utilizará esta base. Com o arquivo transferido, é possível importá-lo utilizando a opção *Custom Package* presente na opção *Import Package* na aba *Assets*.

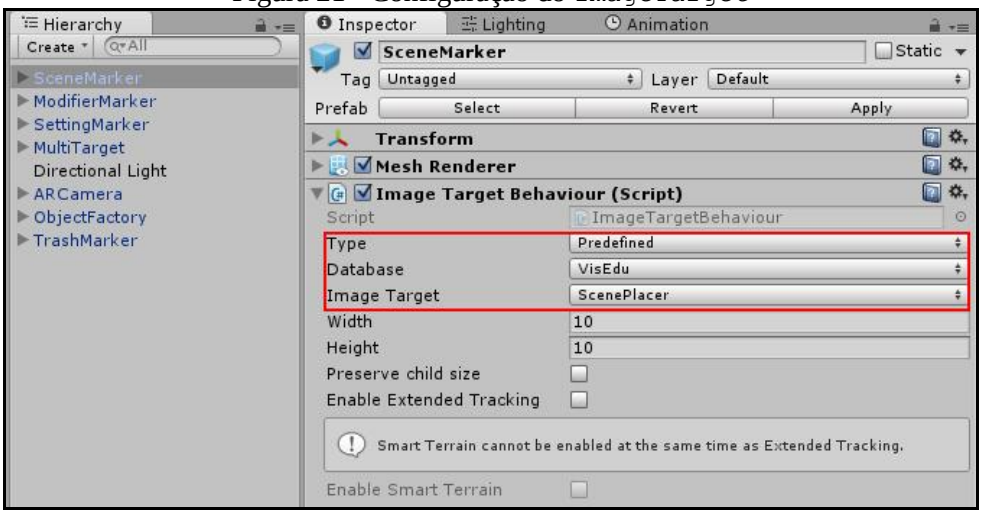
Ao concluir a importação do *Package* é necessário remover o objeto `Main Camera` presente na janela `Hierarchy` (adicionada por padrão ao criar um projeto Unity) e adicionar o `Prefab ARCamera` (localizado dentro da pasta `Prefab` que está localizado dentro da pasta Vuforia na janela `Project`). Este `Prefab` é responsável pela câmera especial utilizada para gerar a Realidade Aumentada. Após incluir este `Prefab` em sua `Scene`, na janela `Inspector` deve-se inserir a chave de licença obtida no atributo `App License Key` no *script* `Vuforia Behavior` presente, selecionar e ativar a base de alvos cadastrada no *script* `Database Load Behavior`. A Figura 20 demonstra nos quadros em vermelho as configurações necessárias para a utilização da `ARCamera`.

Figura 20 - Configuração da `ARCamera`



No aplicativo desenvolvido utilizou-se os `Prefabs ImageTarget` e `MultiTarget`, sendo utilizado para a geração de objetos tridimensionais e botões virtuais. Com isso, é necessário adicionar o `Prefab ImageTarget` ou `MultiTarget` na janela `Hierarchy`. Com o `Prefab` selecionado na janela `Hierarchy`, na janela `Inspector` deve ser configurado a base de alvos que será utilizado pelo `Prefab`. Para isso, no *script* `Image Target Behavior` deve-se selecionar `Predefined` no atributo `Type`, escolher a base de alvos cadastrado no atributo `Database` e o alvo cadastrado no atributo `Image Target`, que pode ser visto por um quadro em vermelho na Figura 21.

Figura 21 - Configuração do `ImageTarget`

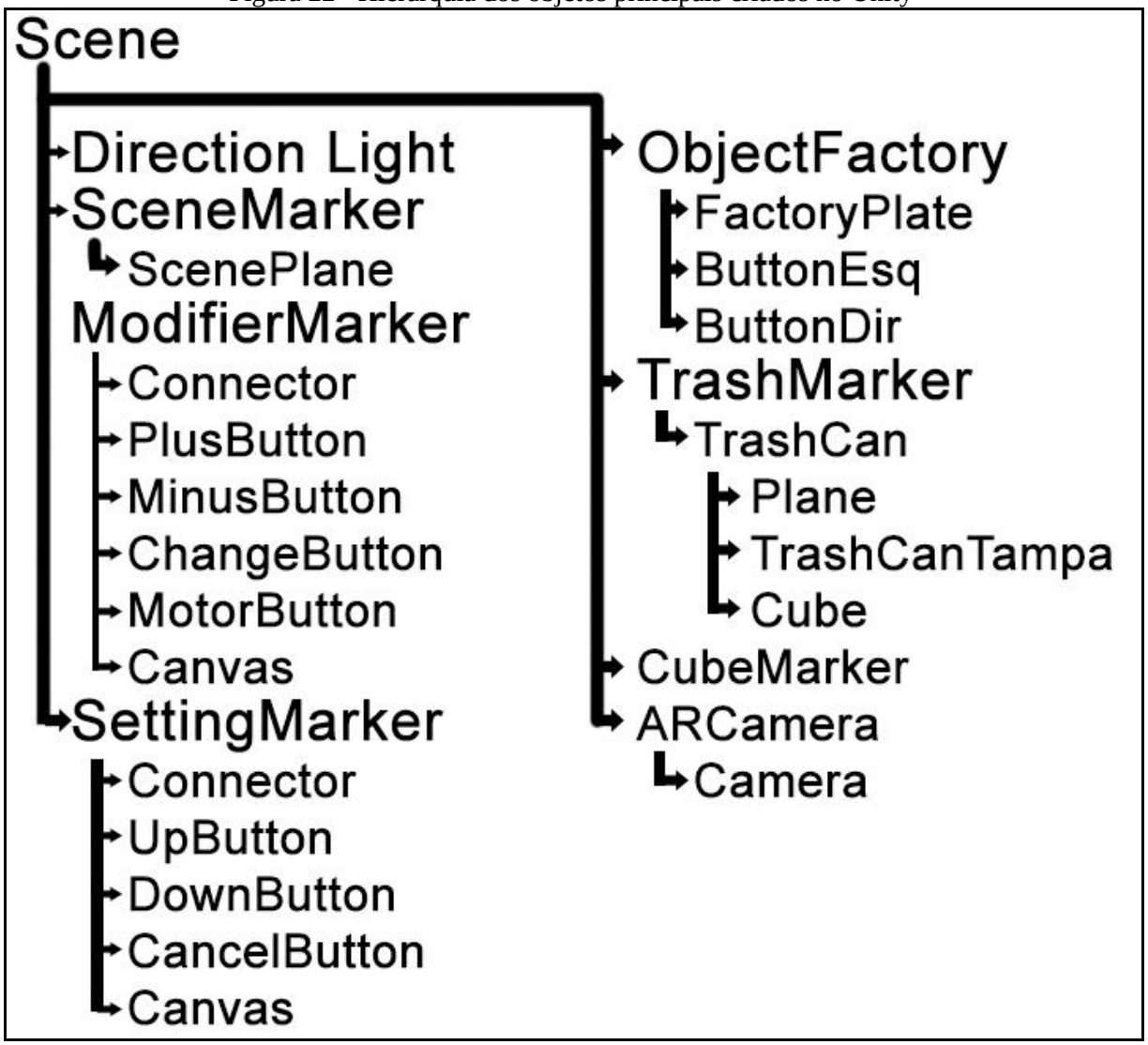


Após a configuração do `Prefab ImageTarget`, adicionam-se objetos tridimensionais para serem gerados ao detectar o marcador pela câmera do dispositivo, tem de se inserir os modelos tridimensionais como filhos na árvore de objetos da janela `Hierarchy`.

3.3.1.2 Aplicativo

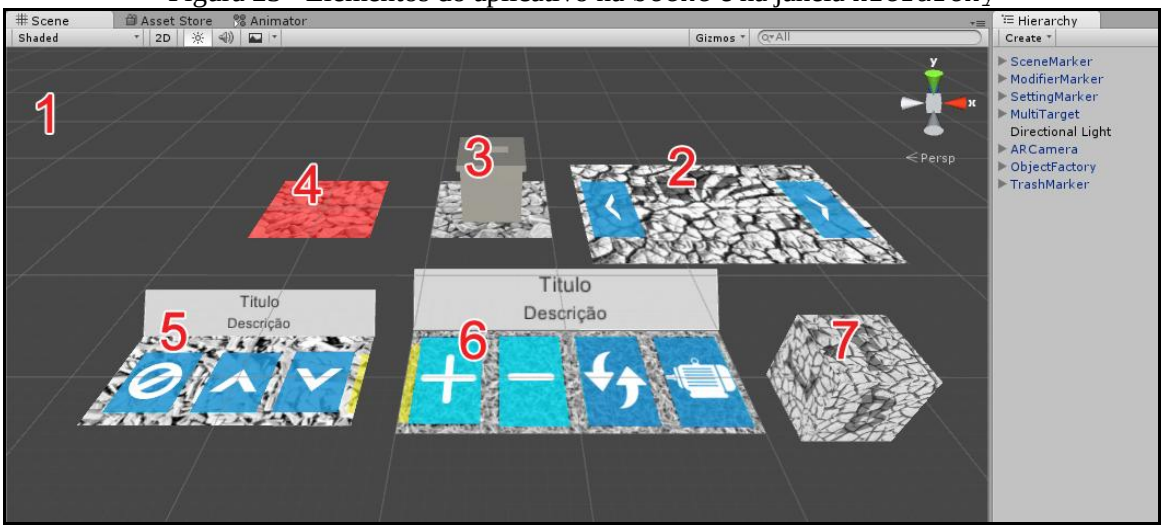
A aplicação possui objetos que são adicionados à cena. Estes objetos, chamados `GameObject`, possuem objetos que são filhos na cadeia hierárquica. A Figura 22 mostra a hierarquia dos objetos principais da aplicação.

Figura 22 - Hierarquia dos objetos principais criados no Unity



Os itens utilizados na RA do aplicativo estão presentes na `Scene` mostrado na Figura 23.

Figura 23 - Elementos do aplicativo na `Scene` e na janela `Hierarchy`

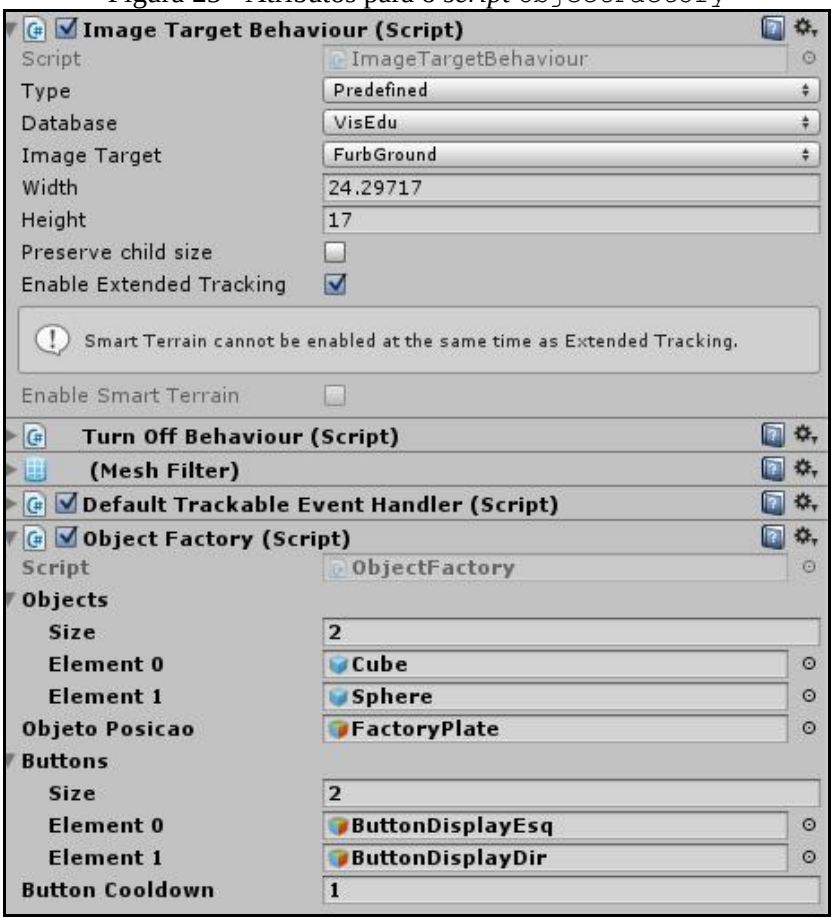


No `item 1` da Figura 23 mostra `Scene` com os objetos virtuais e seus marcadores utilizados na aplicação. Na `ARCamera` foi selecionada a opção `SPECIFIC_TARGET` no atributo `World Center Mode` e definido o objeto `ObjectFactory` (`item 2`) no atributo `World Center`. A `ARCamera` possui um `GameObject` do tipo câmera como filho. Neste `GameObject` foi adicionado o *script* `MainScreenManager` para carregar a tela inicial da aplicação ao pressionar o botão voltar no dispositivo.

O `item 2` da Figura 23 é um `Prefab` do tipo `ImageTarget` e foi dado o nome de `ObjectFactory` na janela `Hierarchy`. Ele é chamado de `Fábrica` neste trabalho. A imagem desenvolvida para este item está definida na Figura 24. No *script* `ImageTargetBehavior` foi ativado o atributo `Enable Extended Tracking`, o que permite rastrear o marcador mesmo fora da visão da câmera do dispositivo. Esta funcionalidade é feita a partir das informações obtidas do ambiente capturado pela câmera. Também foi adicionado o *script* `ObjectFactory` para o controle do comportamento dos botões virtuais e para a geração de objetos virtuais. Neste *script* são informados os atributos `Objects`, `Objeto Posicao`, `Buttons` e `Buttons Cooldown` (Figura 25).

Figura 24 - Imagem associada ao objeto `ObjectFactory`

Figura 25 - Atributos para o *script* `ObjectFactory`



O Quadro 2 apresenta o método `setObject` da classe `ObjectFactory` que é chamada quando um dos botões virtuais é pressionado utilizando o método `OnButtonPressed`. Este método é responsável por destruir o objeto virtual existente associado à `Fábrica` e inserir o

próximo objeto virtual na lista de objetos da variável `objects`. Caso o objeto criado não obtenha um componente do tipo `Collider`, então é adicionado um `BoxCollider`, como pode ser visto na linha 44 do Quadro 2. Este componente é adicionado para tratar a seleção deste objeto a ser colocado na cena com o `Cubo` (`item 7`) e para ajustar o tamanho do `BoxCollider` do `Prefab FactoryPlate`, que é um filho do `ObjectFactory`. Se o objeto criado não obter um componente do tipo `Modifiers`, então é adicionado este componente a ele, como pode ser visto na linha 49 do Quadro 2.

Quadro 2 - Criação do objeto virtual no `Prefab ObjectFactory`

```
...
37 private void setObject(int value)
38     {
39         if (ObjetoSelecionado != null)
40             Destroy(ObjetoSelecionado);
41         ObjetoSelecionado = Instantiate(objects[value], new
           Vector3(0, 0, 0),   Quaternion.identity) as GameObject;
42         if (ObjetoSelecionado.GetComponent<Collider>() == null)
43         {
44             BoxCollider bc =
         ObjetoSelecionado.AddComponent<BoxCollider>() as BoxCollider;
45             bc.size = new Vector3(1, 1, 1);
46             bc.center = new Vector3(0, 0, 0);
47         }
48         if (ObjetoSelecionado.GetComponent<Modifiers>() == null)
49             ObjetoSelecionado.AddComponent<Modifiers>();
50      ObjetoSelecionado.transform.SetParent(ObjetoPosicao.transform);
51         ObjetoSelecionado.transform.localScale =
           SceneControl.ScaleToFit(ObjetoSelecionado, new
           Vector3(0.5f,0.5f,0.5f));
52         ObjetoSelecionado.transform.position =
           SceneControl.PositionRelativeBottom(
               ObjetoSelecionado.GetComponent<Renderer>().bounds.size,
               ObjetoPosicao.transform.position);
53     }
...
```

Com os componentes adicionados foi preciso atribuir o `GameObject ObjetoPosicao`, que foi nomeado como `FactoryPlate` na janela `Hierarchy` como parente do objeto criado. Após isso foi alterado a escala local do objeto virtual criado para encaixar ao tamanho do `Collider` presente no `FactoryPlate`. Para isso é chamado o método `ScaleToFit` da classe estática `SceneControl` que tem como objetivo mudar a escala do `GameObject`, sendo esse, passado por parâmetro até que todas as dimensões mantenham sua proporção e fiquem dentro do parâmetro `Vector3` (Quadro 3). Com a escala modificada, o objeto criado é movido até entrar em contato com o fundo do `BoxCollider` do `FactoryPlate`. Isso é feito utilizando o método `PositionRelativeBottom` da classe estática `SceneControl`.

Quadro 3 - Método para escalar o GameObject

```
...
34    public static Vector3 ScaleToFit(GameObject obj, Vector3 size)
35    {
36        Vector3 objSize = obj.GetComponent<Renderer>().bounds.size;
37        Vector3 ret = new Vector3(objSize.x, objSize.y, objSize.z);
38        float vy = objSize.y;
39        if (ret.y > size.y) {
40            vy = size.y / objSize.y;
41            ret.x *= vy;
42            ret.y *= vy;
43            ret.z *= vy;
44        }
45        if(ret.x > size.x) {
46            vy = size.x / objSize.x;
47            ret.x *= vy;
48            ret.y *= vy;
49            ret.z *= vy;
50        }
51        if (ret.z > size.z) {
52            vy = size.z / objSize.z;
53            ret.x *= vy;
54            ret.y *= vy;
55            ret.z *= vy;
56        }
57        return ret;
58    }
...
```

O item 3 da Figura 23 é um Prefab do tipo ImageTarget e foi nomeada TrashMarker na janela Hierarchy. Este item é chamado de Lixo neste trabalho. A imagem desenvolvida para este item está definida na Figura 26.

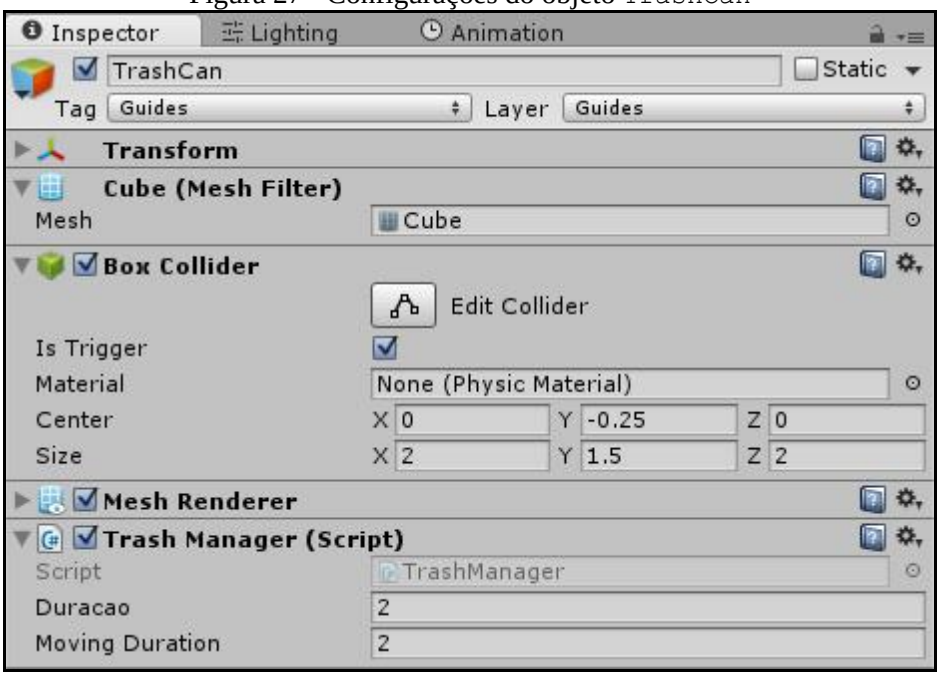
Figura 26 - Imagem associada ao objeto TrashManager

Este GameObject tem como objetivo remover objetos selecionados pelo GameObject CubeMarker (item 7). Este objeto é representado por uma lixeira virtual tridimensional, o objeto TrashCan que é filho do TrashMarker e possui um componente BoxCollider com o

atributo `isTrigger` selecionado. Também foi adicionado o *script* `TrashManager`, como pode ser visto na Figura 27.

O *script* `TrashManager` possui os atributos para controlar o tempo necessário para o objeto ser removido e a duração da animação que movimenta o mesmo até a lixeira virtual que está associado aos atributos `Duracao` e `MovingDuration` respectivamente.

Figura 27 - Configurações do objeto `TrashCan`



O Quadro 4 apresenta os métodos `OnTriggerEnter`, `OnTriggerStay` e `OnTriggerExit` que são ativados assim que um objeto virtual que possua um componente do tipo `Collider` entre em colisão com o `BoxCollider` do `GameObject TrashCan`, se mantenha dentro ou saia do componente `BoxCollider`. Se um `GameObject` entrar na área do `BoxCollider` o método ignora todos os `GameObjects` que possuem a `Tag Guides` e a `Tag GuidesConnector`. Já se o `GameObject` não for filho do `ObjectFactory` e do `FactoryPlate`, a variável `timer` é zerada e é iniciada a animação de abertura da tampa da lixeira virtual.

Quando um objeto se mantém dentro do `BoxCollider` do `TrashCan`, o método irá ignorar os mesmos `GameObject` que o método `OnTriggerEnter` ignorou e irá iniciar um contador de tempo que, após o tempo associado ao atributo `Duracao`, associará o parente do objeto a ser removido com um `null` e irá destruí-lo.

Se durante este processo o objeto virtual que iria ser removido sair da área do `BoxCollider` do `TrashCan` irá ativar a animação para fechar a tampa da lixeira virtual.

Quadro 4 - Controle da lixeira

```
...
33    void OnTriggerEnter(Collider colliderInfo) {
34        if (colliderInfo.gameObject.tag.Equals("Guides") ||
          colliderInfo.gameObject.tag.Equals("GuidesConnector"))
35            return;
36        if (colliderInfo.gameObject.transform.parent !=
          SceneControl.getFactoryPlate()
          && colliderInfo.gameObject.transform.parent !=
          SceneControl.getObjectFactory()) {
37            timer = 0;
38            anim.SetBool("opening", true);
39        }
40    }
41    void OnTriggerStay(Collider colliderInfo) {
42        if (colliderInfo.gameObject.tag.Equals("Guides") ||
          colliderInfo.gameObject.tag.Equals("GuidesConnector"))
43            return;
44        if (colliderInfo.gameObject.transform.parent !=
          SceneControl.getFactoryPlate()
          && colliderInfo.gameObject.transform.parent !=
          SceneControl.getObjectFactory()) {
45            if (timer >= duracao || objetoRemovido != null) {
46                objetoRemovido = colliderInfo.gameObject;
47                objetoRemovido.transform.SetParent(null);
48                posicaoObjeto = objetoRemovido.transform.position;
49                escalaObjeto = objetoRemovido.transform.localScale;
50            } else {
51                timer += Time.fixedDeltaTime;
52            }
53        }
54    }
55    void OnTriggerExit(Collider colliderInfo) {
56        if (colliderInfo.gameObject.transform.parent !=
          SceneControl.getFactoryPlate()
          && colliderInfo.gameObject.transform.parent !=
57        SceneControl.getObjectFactory()) {
58          anim.SetBool("opening", false);
59        }
60    }
...
```

O `item 4` da Figura 23 é um `Prefab` do tipo `ImageTarget` e foi nomeada `SceneMarker` na janela `Hierarchy`. Este item é chamado de `Alvo` neste trabalho e a imagem desenvolvida para este item está definida na Figura 28.

Figura 28 - Imagem associada ao `Prefab SceneMarker`

Este `GameObject` tem como objetivo adicionar um objeto virtual selecionado pelo `GameObject CuboMarker` (`item 7`) e associar este objeto no ambiente virtual. Isso é feito associando o `GameObject ObjectFactory` como parente do objeto selecionado. Foi adicionado um `GameObject` filho do tipo `Plane` como filho e nomeado `ScenePlane`, este `GameObject` é definido por padrão com a cor vermelha. Foi adicionado a este `GameObject` o *script* `SceneMarkerManager` e adicionado um componente do tipo `BoxCollider` com o atributo `isTrigger` selecionado. Como também um `GameObject` do tipo `Plane` que foi nomeado de `ScenePlane`, como pode ser visto na Figura 29.

Figura 29 - Configuração do objeto `ScenePlane`

O Quadro 5 apresenta os métodos `OnTriggerEnter`, `OnTriggerStay` e `OnTriggerExit` presentes no *script* `SceneMarkerManager`. Estes métodos são ativados da mesma forma que os métodos de mesmo nome no *script* `TrashManager`.

Quadro 5 - Tratamento de colisão do marcador `Alvo`

```
...
50    void OnTriggerEnter(Collider colliderInfo) {
51        timer = 0;
52    }
53    void OnTriggerStay(Collider colliderInfo) {
54        if (!SceneControl.isChildFactoryPlate(colliderInfo.gameObject)
          && !SceneControl.isChildObjectFactory(colliderInfo.gameObject)
          && colliderInfo.gameObject.tag!="GuidesConnector") {
55            GetComponent<Renderer>().material.color = Color.yellow;
56            if (timer >= duracao || objetoMovido != null) {
57                objetoMovido = colliderInfo.gameObject;
58                objetoMovido.transform.SetParent(null);
59                posicaoObjeto = objetoMovido.transform.position;
60                rotacaoObjeto = objetoMovido.transform.rotation;
61            } else {
62                timer += Time.fixedDeltaTime;
63            }
64        }
65    }
66    void OnTriggerExit(Collider colliderInfo) {
67        GetComponent<Renderer>().material.color = Color.red;
68        timer = 0;
69    }
...
```

No método `OnTriggerEnter` apresentado no Quadro 5, a variável `timer` é associado ao valor zero, no método `OnTriggerStay` são ignorados os `GameObject` que são filhos do `GameObject FactoryPlate`, filhos do `GameObject ObjectFactory` e `GameObject` que possuem a `tag GuidesConnector` e a `tag Guides`. Após o tempo definido pela variável `duracao`, a variável `objetoMovido` do tipo `GameObject` é referido ao `GameObject` que está dentro do componente `BoxCollider`. Após isso é associado o valor `null` como parente do `GameObject` e em seguida são obtidos os valores de posição e rotação do objeto selecionado. Estes valores são utilizados no método `Update`, que é apresentado no Quadro 6. No método `OnTriggerExit` a cor do `ScenePlane` é alterada para vermelho.

O método `Update` do *script* `SceneMarkerManager` é demonstrado abaixo no Quadro 6, que chamado a cada *frame* desenhado, é verificado se a variável `ObjetoMovido` não é `null`. Se for o caso, é verificado se o tempo de duração que é guardado na variável `movingTimer` é menor que o tempo associado na variável `movingDuration` que é associado ao atributo `Moving Duration` na janela `Hierarchy` apresentada na Figura 29. Durante este intervalo soma-se a diferença de tempo do último *frame* ao atual, e este valor é dividido por `movingDuration`. Com isso o objeto é movimentado em forma de curva até a posição do

ScenePlane sendo relativo ao fundo do objeto movimentado. Após isso é criado um atraso de um segundo. Com a variável movingTimer sendo maior que movingDuration, o objeto movido tem como o parente associado ao objeto ObjectFactory. Também é verificado se o objeto movido possui o componente Modifiers, caso contrário, o componente é adicionado, e após isso é chamado o método setValues que irá guardar a nova posição deste objeto. Por fim a variável objetoMovido é associado com null, a cor do ScenePlane é associado com vermelho e a variável movingTimer reduzido para zero. Com isso, o objeto movido estará incluído no ambiente virtual sendo associado com o objeto ObjectFactory.

Quadro 6 - Método Update do *script* SceneMarkerManager

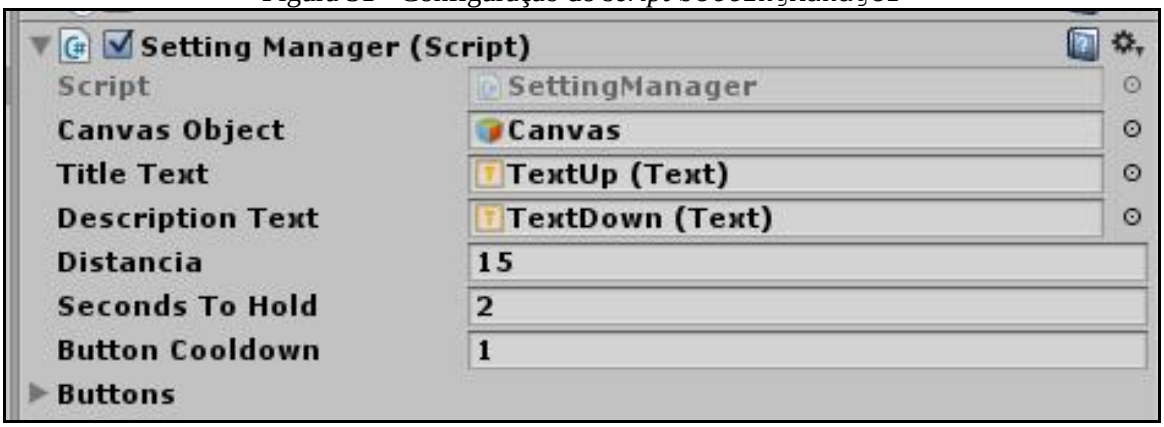
```
...
20 void Update () {
21     if (objetoMovido != null) {
22         if (movingTimer < movingDuration) {
23             movingTimer += Time.deltaTime / movingDuration;
24             objetoMovido.transform.position =
25             SceneControl.Bezier2(transform.position,
               transform.position + (transform.up * 10),
               posicaoObjeto, movingTimer);
26             objetoMovido.transform.position =
               Vector3.Lerp(posicaoObjeto,
               SceneControl.PositionRelativeBottom(
               objetoMovido.GetComponent<Renderer>().bounds.size,
               transform.position), movingTimer);
27             objetoMovido.transform.rotation =
28             Quaternion.Lerp(rotacaoObjeto, transform.rotation,
               movingTimer);
29             GetComponent<Renderer>().material.color = Color.green;
30             StartCoroutine(waitSeconds(1));
31         } else {
32             objetoMovido.transform.SetParent(
               SceneControl.getObjectFactory().transform);
33             if (objetoMovido.GetComponent<Modifiers>() == null) {
34                 objetoMovido.AddComponent<Modifiers>();
35                 objetoMovido.GetComponent<Modifiers>().setValues();
36             } else {
37                 objetoMovido.GetComponent<Modifiers>().setValues();
38             }
39             objetoMovido = null;
40             GetComponent<Renderer>().material.color = Color.red;
41             movingTimer = 0;
42         }
43     }
44 }
...
```

O item 5 da Figura 23 é um Prefab do tipo ImageTarget e foi nomeada de SettingMarker na janela Hierarchy. A imagem desenvolvida para este item está definida na Figura 30. Neste GameObject foi adicionado o componente Line Renderer que cria uma linha virtual que tem como funcionalidade conectar ao GameObject selecionado que está inserido no ambiente virtual. Este componente foi desativado para ser ativado via *script*.

Foram adicionados botões virtuais para controlar o comportamento do objeto quando um objeto for selecionado. Estes botões servem para cancelar a conexão entre o `Prefab SettingMarker` e o objeto virtual selecionado, bem como selecionar o próximo atributo. Por fim, selecionar o atributo anterior. Também foi adicionado o *script* `SettingManager` para o controle do comportamento dos botões virtuais e selecionar o atributo que será modificado pelo `item 6`. Neste *script* são informados os atributos `CanvasObject`, `TitleText`, `DescriptionText`, `Distancia`, `SecondsToHold`, `ButtonCooldown` e `Buttons` (Figura 31).

Figura 30 - Imagem associada ao `Prefab SettingMarker`

Figura 31 - Configuração do *script* `SettingManager`



O `GameObject` que está no ambiente virtual é selecionado com o uso do método `getObjectByLine` que é chamado a cada *frame* pelo método `Update`, como pode ser visto no Quadro 7 abaixo. Este método obtém o objeto que está na distância definida pela variável `distancia`, e ignora qualquer objeto que possui o atributo `Layer` associado á `Guides`.

Foi definido um contador de tempo, tendo o tempo definido pelo atributo `SecondsToHold` na Figura 31. Para definir quanto tempo a linha virtual deve estar em colisão com o objeto virtual a ser selecionado. Caso não haja qualquer objeto a ser selecionado, a linha terá a cor vermelha, caso exista um objeto em alcance a ser selecionado e ainda esteja

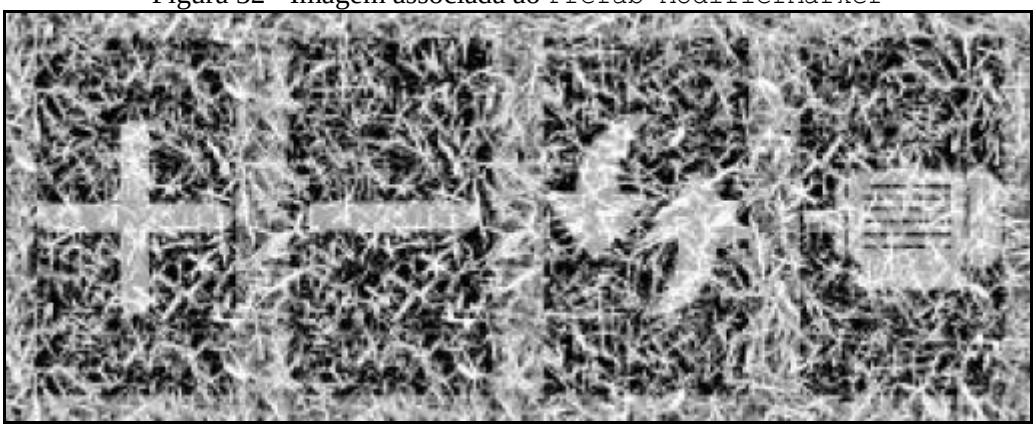
sendo feita a contagem de tempo, a linha se torna amarela. Quando o objeto é selecionado a linha se torna verde.

Quadro 7 - Método que obtém o objeto no final da linha virtual

```
...
235 private void getObjectByLine() {
236     ray = new Ray(posicaoLineStart, transform.forward);
237     RaycastHit hitInfo;
238     if (objetoSelecionado == null) {
239         if (Physics.Raycast(ray, out hitInfo, distancia + 2,
            maskLayer)) {
240             if (SceneControl.isChildObjectFactory(
                hitInfo.collider.gameObject)) {
241                 if (hitInfo.distance > distancia) {
242                     timer = 0;
243                     line.SetColors(Color.red, Color.red);
244                 } else {
245                     timer += Time.deltaTime;
246                     line.SetColors(Color.yellow, Color.yellow);
247                 }
248                 if (timer >= secondsToHold) {
249                     line.SetColors(Color.green, Color.green);
250                     objetoSelecionado = hitInfo.collider.gameObject;
251                     modObject =
                        objetoSelecionado.GetComponent<Modifiers>() as
                        Modifiers;
252                     timer = 0;
253                     updateTitle();
254                 }
255             }
256         } else {
257             line.SetColors(Color.red, Color.red);
258         }
259     } else {
260         line.SetColors(Color.green, Color.green);
261     }
262    }
...
```

O `item 6` da Figura 23 é um `Prefab` do tipo `ImageTarget` e foi nomeada de `ModifierMarker` na janela `Hierarchy`. A imagem desenvolvida para este item está definida na Figura 32.

Figura 32 - Imagem associada ao `Prefab ModifierMarker`



Neste GameObject foi adicionado o componente Line Renderer para conectar ao objeto SettingMarker. Este componente também foi desativado para ser ativado via *script*. Para fazer esta conexão entre estes GameObject, foi adicionado um prefab do tipo Plane contendo um BoxCollider em cada um destes marcadores. Com isso, quando estes dois objetos colidirem, isso fará que eles se conectem por uma linha virtual de cor verde.

Este objeto também possui botões virtuais que servem para adicionar e remover o valor base ao objeto selecionado, para alterar o valor base, e um botão que serve como um motor, que irá adicionar o valor base a cada segundo. Para o controle de todas estas funcionalidades, foi criado o *script* ModifierManager que foi adicionada ao objeto.

Quando um botão que aumenta ou diminui o valor é pressionado, é chamado o método updateValue, que verifica se a variável settingScript é diferente de null. Caso for diferente é chamado o método getModValue presente no settingScript e é adicionado o valor da variável local value. Após isso o valor é associado no *script* settingScript pelo método setActualMod.

Este objeto possui um Canvas que irá mostrar o valor base que irá ser adicionado ou removido e o valor modificado atual do objeto virtual selecionado pelo objeto SettingMarker, para isso foi adicionado uma variável chamada de settingScript que é uma referência ao *script* SettingManager. Os métodos UpdateTitle e UpdateDesc são responsáveis por atualizar os valores (Quadro 8), e o Canvas é somente ativado quando os dois objetos estiverem conectados. Os métodos que são responsáveis por atualizar o texto verificam se o Canvas está habilitado e se a variável settingScript é diferente de null.

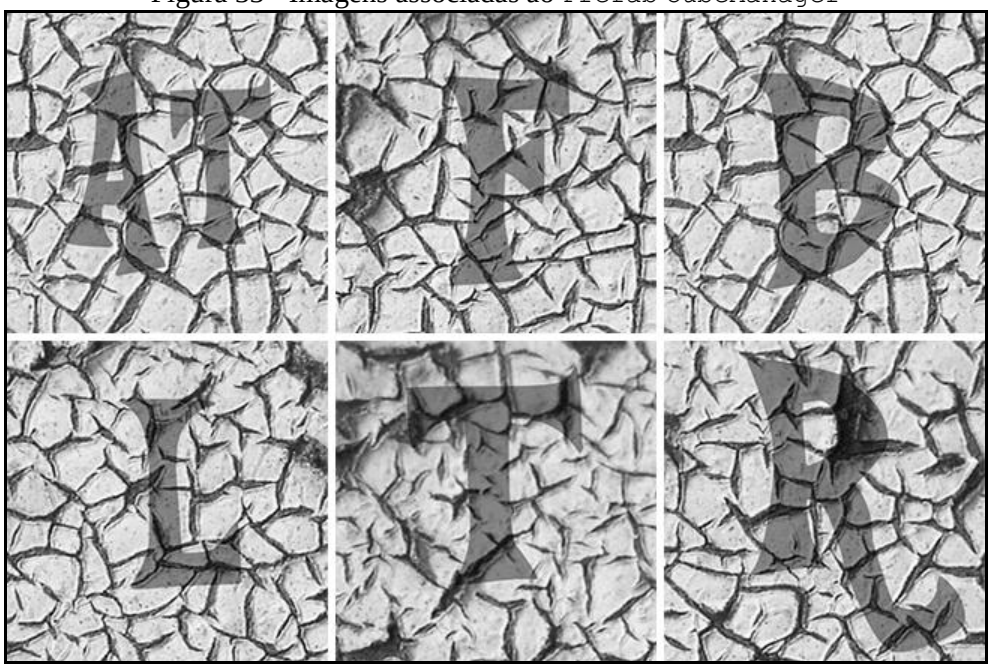
Diferente do método UpdateTitle, o método UpdateDesc verifica se o modificador selecionado, que é retornado pelo método getActiveMod é o mesmo guardado na variável modValue. Caso seja diferente, o valor da variável activeMod é atualizado assim como o texto da variável descriptionText.

Quadro 8 - Método que atualiza o texto de valores

```
...
 98 private void UpdateTitle() {
 99     if (!canvas.enabled || settingScript == null)
101         return;
102     if(settingScript!=null)
103         titleText.text = "Valor: " + modValue +
            (settingScript.getMotorValue()!=0 ? " (Motor)" : "");
104     else
105         titleText.text = "Valor Atual: " + modValue;
106 }
107
108 private void UpdateDesc() {
109     if (!canvas.enabled || settingScript == null )
110         return;
111     if (settingScript.getActiveMod() == activeMod)
112          return;
113     activeMod = settingScript.getActiveMod();
114     descriptionText.text = "Valor: " +
      settingScript.getActualValue();
115 }
...
112 private void updateValue(float value) {
113     if (settingScript != null)
114         settingScript.setActualValue(settingScript.getModValue() +
            value);
115 }
...
```

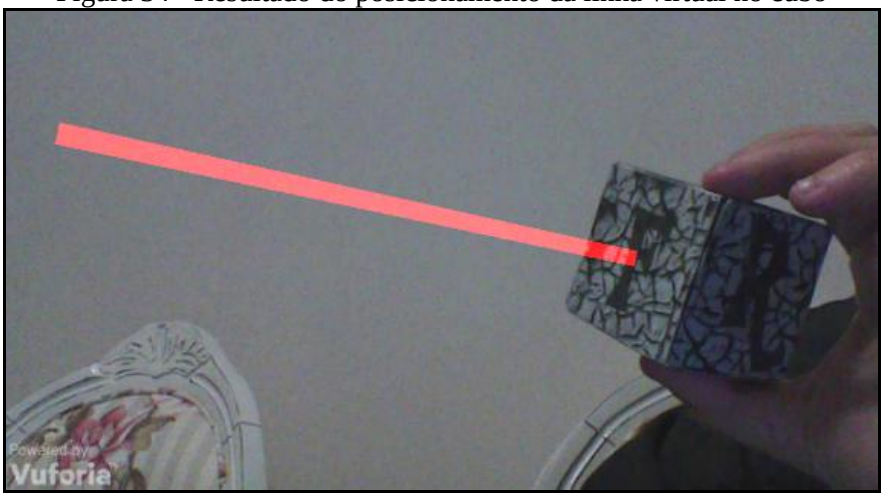
Por fim, o `item 7` da Figura 23 é um `Prefab` do tipo `MultiTarget` e foi nomeada de `CubeMarker` na janela `Hierarchy`. Este tipo de `prefab` possui um formato de cubo e possui um marcador para cada uma de suas faces, as seis imagens utilizadas são mostradas na Figura 33.

Figura 33 - Imagens associadas ao `Prefab` `CubeManager`



Este objeto possui um componente do tipo `Line Renderer` que gera uma linha virtual para selecionar os objetos virtuais, as posições da linha virtual são definidas no *script* `CubeManager`, a posição inicial desta linha virtual é igual a posição do cubo, a posição final é sempre posicionada a frente do cubo, tendo sua distância definida pelo valor da variável `distancia`. Com a posição final da linha associada, a linha sempre se posicionará a frente da face com a letra F, como mostrada na Figura 34.

Figura 34 - Resultado do posicionamento da linha virtual no cubo



A posição inicial e final da linha virtual é configurada no método `Start` (Quadro 9) que é ativado quando o objeto é criado, isso ocorre quando ele é reconhecido pela câmera do dispositivo pela primeira vez. O método `getObjectByLine` é responsável por selecionar o objeto virtual. Este método possui uma funcionalidade semelhante ao método `getObjectByLine` do *script* `SettingManager` (Quadro 7) para selecionar objetos virtuais, ignorando objetos com `Layer` igual ao valor `Guides`.

Neste método, enquanto não há nenhum objeto a ser agarrado ao alcance da linha virtual, a linha é pintada de vermelho. Quando há um objeto ao alcance, porém, ainda está na contagem de tempo a linha é pintada de amarelo. E quando o objeto finalmente é agarrado a linha é pintada de verde. Neste momento, caso o objeto for filho do objeto `FactoryPlate` é criado uma cópia do objeto a ser selecionado, assim o objeto é mantido na biblioteca de objetos (`item 2`) e associado como filho do `CubeManager`.

Dessa forma, se o objeto estiver no ambiente virtual, ou seja, após ter sido colocado no ambiente pelo objeto `SceneMarker`, é somente associado como filho do `CubeManager`. Após associar o objeto selecionado como filho do `CubeMarker`, é chamado o método `resetModPosition`, assim zerando as posições modificadas do objeto selecionado para o objeto continuar na posição no final da linha virtual. Após o objeto ser associado ao ambiente virtual ou ser removido pela lixeira virtual (`item 3`), a linha volta a ser vermelha.

Quadro 9 - Métodos do *script* `CubeManager`

```
...
15 void Start () {
16  line = gameObject.GetComponent<LineRenderer>();
17  line.SetColors(Color.red, Color.red);
18  line.SetPosition(1, new Vector3(0,distancia,0));
19  line.SetWidth(0.5f, 0.5f);
20  ray = new Ray(transform.position, Vector3.up);
12 }
...
25 private void getObjectByLine() {
26   ray = new Ray(transform.position, transform.up);
27   RaycastHit hitInfo;
28   if (objetoSegurado == null) {
29     if (Physics.Raycast(ray, out hitInfo, distancia + 2,
30     maskLayer)) {
31       if (hitInfo.distance > distancia) {
33         timer = 0;
32         gameObject.GetComponent<Renderer>().material.color =
           Color.red;
33         line.SetColors(Color.red, Color.red);
34       } else {
35         timer += Time.deltaTime;
36         line.SetColors(Color.yellow, Color.yellow);
37         gameObject.GetComponent<Renderer>().material.color =
           Color.yellow;
38       }
39       if (timer >= secondsToHold) {
40        line.SetColors(Color.green, Color.green);
41        gameObject.GetComponent<Renderer>().material.color =
          Color.green;
42        objetoSegurado = hitInfo.collider.gameObject;
43        if (SceneControl.isChildFactoryPlate(objetoSegurado)) {
44          objetoSegurado = Instantiate(objetoSegurado,
            objetoSegurado.transform.position,
            Quaternion.identity) as GameObject;
45          objetoSegurado.transform.SetParent(transform);
46          Vector3 scale = objetoSegurado.transform.localScale;
47          objetoSegurado.transform.localScale = scale*25;
48        } else
49          objetoSegurado.transform.SetParent(transform);
50          objetoSegurado.GetComponent<Modifiers>().setValues();
51          objetoSegurado.GetComponent<Modifiers>().resetModPosition();
52          timer = 0;
53       }
54     } else {
55       gameObject.GetComponent<Renderer>().material.color = Color.red;
56       line.SetColors(Color.red, Color.red);
57     }
58   } else if (objetoSegurado.transform.parent != transform)
59     objetoSegurado = null;
60 }
...
```

3.3.1.3 Tela inicial

Ao iniciar o aplicativo a tela inicial é apresentada com o titulo da aplicação junto com três botões, como apresentado na Figura 35. A tela permite ao usuário acessar a aplicação de

Realidade Aumentada, assim como acessar o questionário utilizado para testar a usabilidade das aplicações e, por fim, acessar as informações da aplicação e do desenvolvedor.

Figura 35 - Tela inicial do aplicativo

No Quadro 10 são apresentados os métodos do *script* `TitleScreenManager`, que por sua vez são chamados quando os botões da tela inicial (Figura 35) são tocados. Ao tocar no `item 1` da Figura 35 chama-se o método `loadScene`. Este método carrega a cena especificado pelo atributo `scene`, responsável por abrir a cena `MainScene`.

Quadro 10 - Métodos do *script* `TitleScreenManager`

```
...
10 public void openURL(string questionarioUrl) {
11     Application.OpenURL(questionarioUrl);
12 }
13 public void loadScene(int scene) {
14     SceneManager.LoadSceneAsync(scene);
15 }
16 public void switchActive(GameObject panel) {
17    panel.SetActive(!panel.activeSelf);
19 }
...
```

Os valores associados às cenas da aplicação podem ser vistos na janela *Build Settings* (Figura 36), que pode ser acessado pela aba *File*.

Figura 36 - Janela *Build Settings*

Ao tocar o `item 2` da Figura 35 chama-se o método `openURL` que possui o parâmetro `questionarioUrl`. Este atributo está associado à página do questionário *on-line* desenvolvida para esta aplicação utilizando a ferramenta Google Forms.

O `item 3` chama o método `switchActive` que habilita um `GameObject` desabilitado e vice-versa, com isso é habilitado um `GameObject` do tipo `Canvas` que foi desabilitado por padrão. Este `Canvas` mostra as informações do autor deste trabalho, assim como o nome da instituição e do curso. Nesta tela (Figura 37), chamada de `Sobre`, há um botão para desabilitar o `Canvas` utilizando o método `switchActive` e voltar a tela principal (Figura 35).

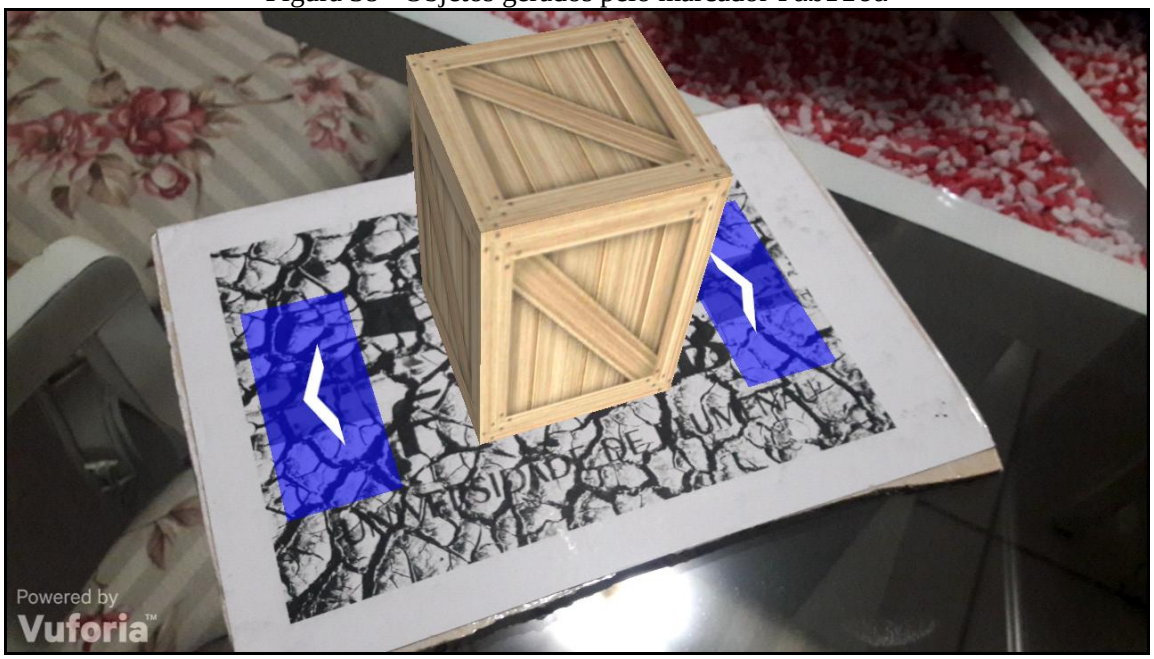
Figura 37 - Tela com as informações do desenvolvedor da aplicação

3.3.2 Operacionalidade da implementação

Para o funcionamento da aplicação é necessária a impressão dos marcadores e a construção do marcador cubo. O tamanho dos marcadores deve seguir as dimensões demonstradas pelos objetos no editor Unity. Este valor pode ser visto pelos atributos `Height` e `Width` na Figura 25.

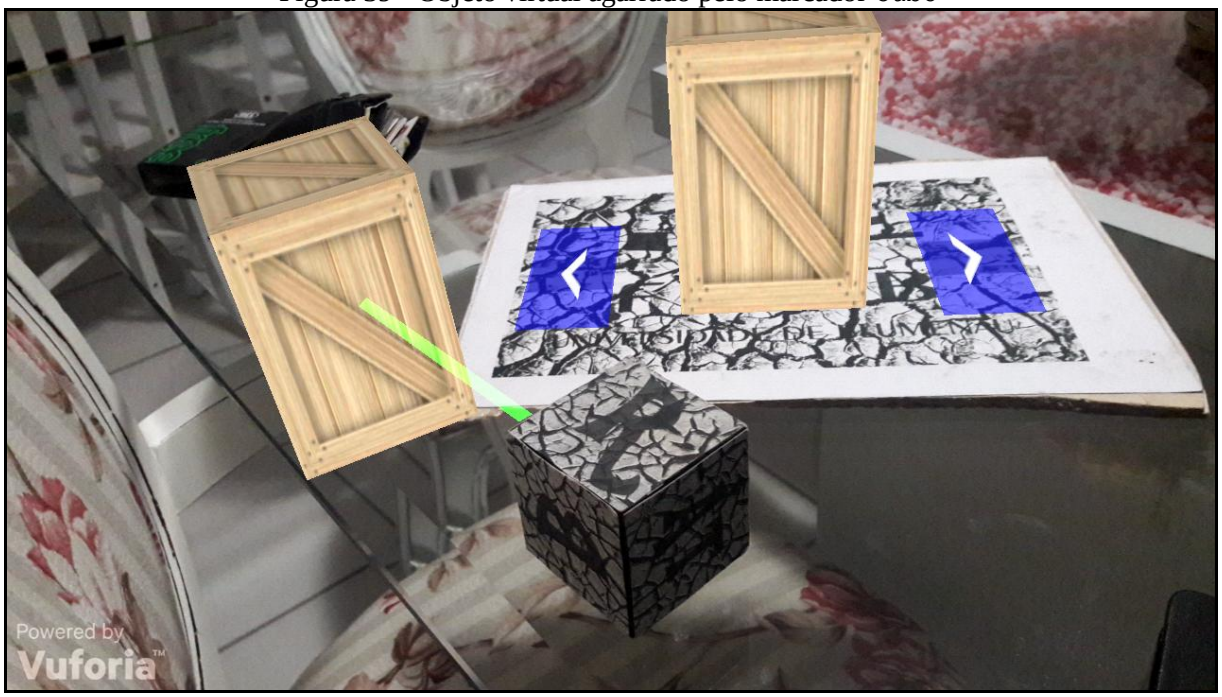
Com os marcadores em mão, o usuário deve tocar o botão `Iniciar` que carrega uma tela com a visão da câmera do dispositivo. O usuário deve apontar a câmera do dispositivo para o marcador `Fábrica` (Figura 24). Após gerar os objetos virtuais, o usuário pode selecionar o objeto utilizando os botões virtuais presentes no marcador (Figura 38).

Figura 38 - Objetos gerados pelo marcador `Fábrica`



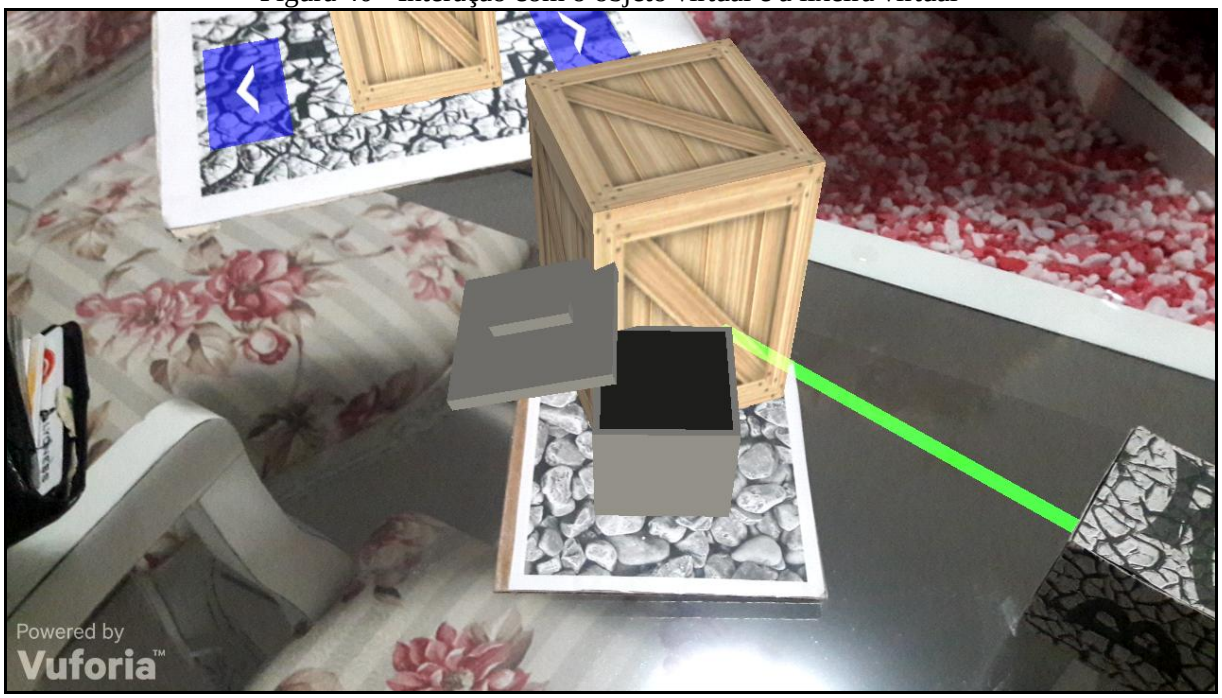
Com o objeto virtual selecionado, o usuário deve utilizar o marcador `Cubo` (Figura 34) e apontar a linha virtual ao objeto a ser agarrado pelo `Cubo`. Após a linha mudar de amarelo para verde o objeto é selecionado pelo `Cubo` (Figura 39).

Figura 39 - Objeto virtual agarrado pelo marcador `Cubo`



Com o objeto sendo agarrado pelo `Cubo`, o usuário pode remover o objeto selecionado com o marcador `Lixo` (Figura 26). Para isso, o usuário deve levar o objeto virtual agarrado ao lixo virtual. A tampa do lixo virtual irá levantar e após dois segundos o objeto será removido (Figura 40).

Figura 40 - Interação com o objeto virtual e a lixeira virtual

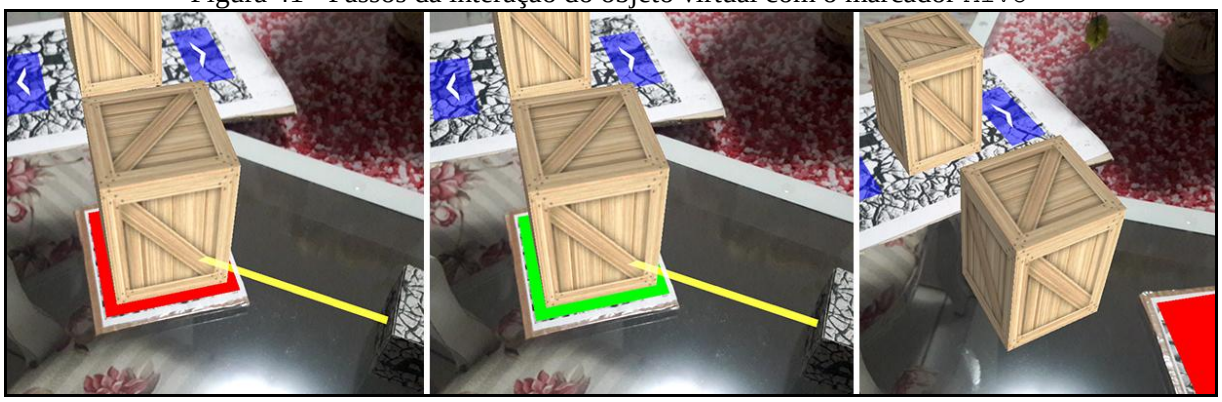


Para colocar o objeto virtual agarrado no cenário virtual é necessário utilizar o marcador `Alvo` (Figura 28), assim como remover o objeto utilizando o marcador `Lixo`. O usuário deve aproximar o objeto virtual agarrado ao marcador até a cor do marcador mudar

para amarelo. Após dois segundos o objeto será movido até a posição do marcador `Alvo`. O objeto virtual estará vinculado ao marcador `Fábrica`.

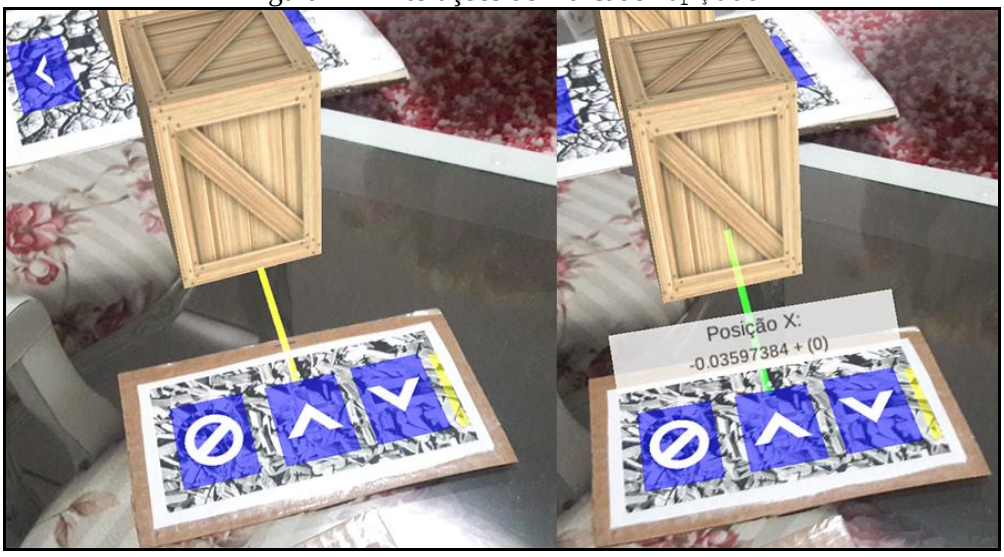
É recomendado que o marcador `Fábrica` esteja visível ou que não seja movido enquanto houver objetos virtuais no ambiente virtual (Figura 41).

Figura 41 - Passos da interação do objeto virtual com o marcador `Alvo`



Com o objeto posicionado no ambiente virtual o usuário pode agarrá-lo com o `Cubo` novamente e reposicioná-lo ou remover, o usuário pode também utilizar o marcador `Opções` (Figura 30) para selecionar um atributo a ser modificado. Para conectar o marcador `Opções` ao objeto é necessário apontar a linha virtual no objeto e assim como o `Cubo`, a linha irá mudar para verde quando conectado ao objeto virtual (Figura 42).

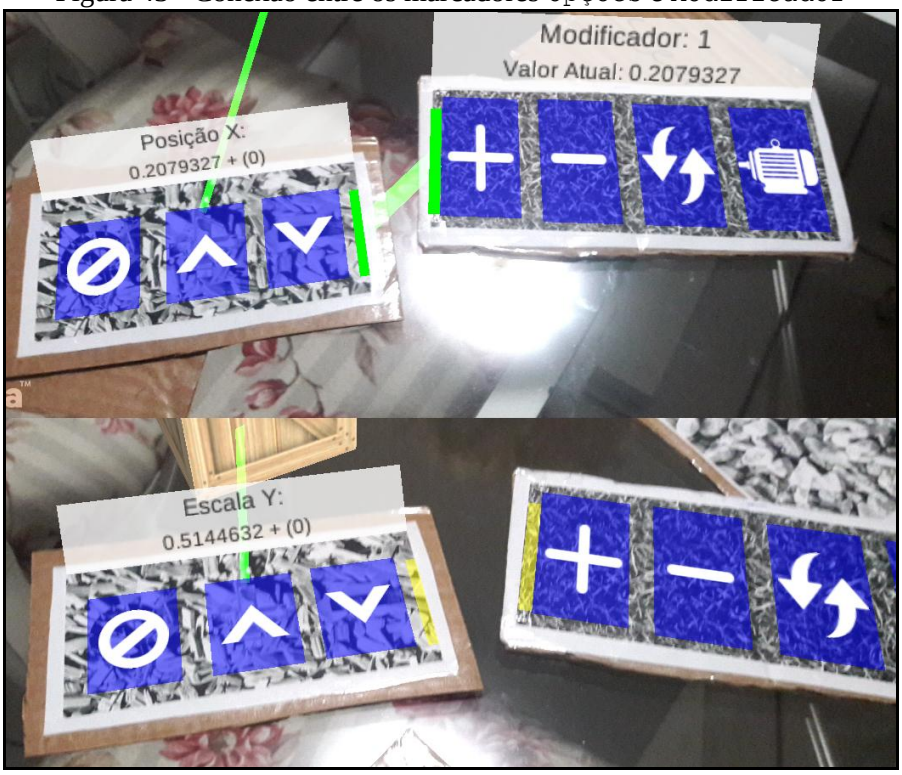
Figura 42 - Interações do marcador `Opções`



Com o objeto conectado ao marcador `Opções`, o usuário pode selecionar um atributo disponível, após isso, para modificar o valor selecionado o usuário deve conectar o marcador `Modificador` (Figura 32) ao marcador `Opções`.

Para conectar os marcadores devem-se ligar as duas marcas amarelas presentes nos marcadores, após a conexão as marcas ficam verdes e terá uma linha verde conectando os dois marcadores (Figura 43).

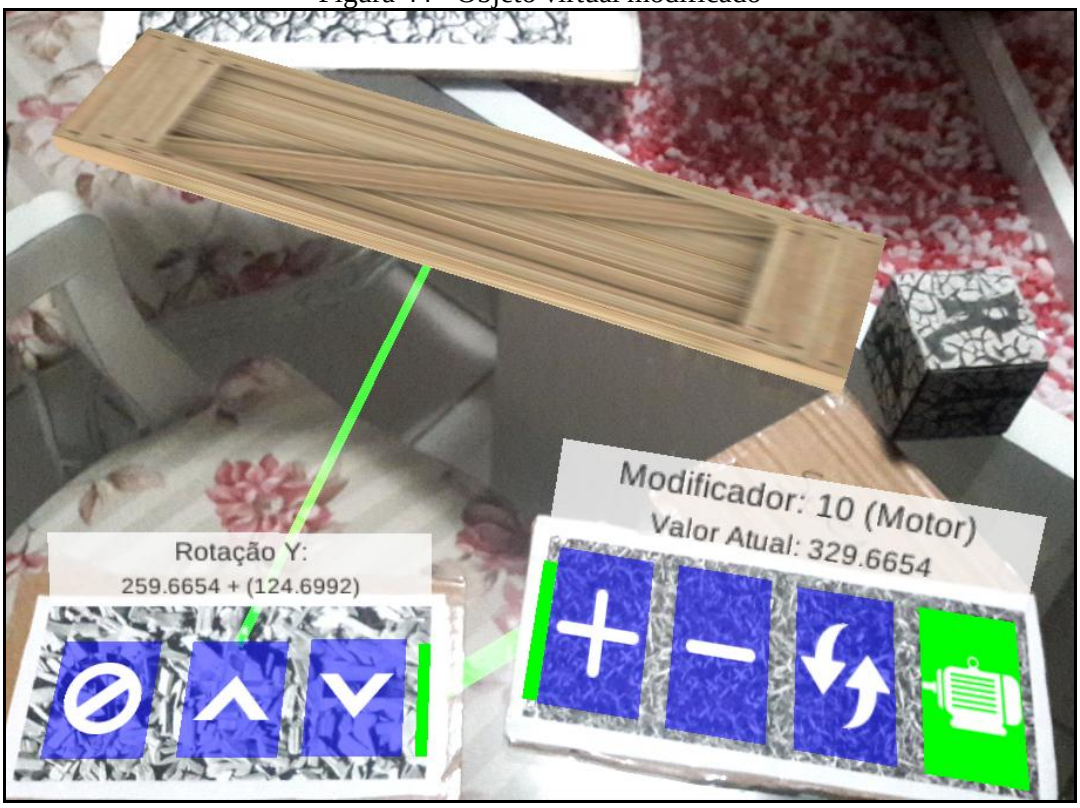
Figura 43 - Conexão entre os marcadores `Opções` e `Modificador`



Com o marcador `Opções` e o marcador `Modificador` conectados, é possível aumentar ou diminuir os valores do atributo selecionado. Usando o botão com o símbolo de adição o valor será aumentado de acordo com o valor modificador e utilizando o botão com o símbolo de subtração o valor será diminuído.

O botão com duas flechas é chamado de botão `Mudar`. Este botão altera o valor do campo modificador. O botão `Motor` que tem como finalidade adicionar o valor do campo modificador automaticamente, ressalta-se que mesmo alterando o atributo no marcador `Opções`, as alterações feitas no atributo anterior (incluindo o motor) permanecerão. Na Figura 44 é possível ver o resultado da alteração dos atributos.

Figura 44 - Objeto virtual modificado



Após a modificação dos atributos, o usuário pode desconectar os marcadores aproximando as marcas verdes da mesma forma que foi feito para conectá-los. Para desconectar o marcador `Opções` do objeto virtual, é necessário pressionar o botão `Cancelar`. Após isso, o usuário pode agarrar o objeto virtual para reposicioná-lo ou removê-lo do ambiente virtual com a lixeira virtual. Caso o usuário deseje voltar para a `tela inicial`, é necessário utilizar o botão voltar do dispositivo móvel.

3.4 RESULTADOS E DISCUSSÕES

Nesta seção é apresentado o experimento realizado com o aplicativo. Na seção 3.4.1.1 é apresentado a metodologia utilizada para o experimento. O experimento, detalhado na seção 3.4.1.2, foi realizado com cinco pessoas da matéria de Multimídia do curso de Ciências da Computação. O experimento teve como objetivo avaliar a usabilidade do aplicativo. Na seção 3.4.2 é apresentado a comparação dos trabalhos correlatos e o trabalho desenvolvido. No Apêndice B descreve-se a exibição do aplicativo gerado por este trabalho na Semana acadêmica da Computação 2016.

3.4.1.1 Metodologia

O experimento ocorreu durante o mês de maio de 2016 por meio de testes realizados com os usuários individualmente, acompanhados com o autor deste trabalho. Para os testes

foram disponibilizados os marcadores para a utilização do aplicativo e um *smartphone* Samsung Galaxy J7. No experimento, foi também fornecido um questionário de perfil, os objetivos e um questionário de avaliação de usabilidade. Este questionário está disponível no Apêndice C.

3.4.1.2 Experimento e avaliação

Este experimento foi realizado na sala do LIFE da Universidade Regional de Blumenau. Antes de realizar o experimento, os cinco usuários responderam um questionário de perfil, cujo resultado pode ser visto no Quadro 11.

Quadro 11 - Perfis de usuários envolvidos no experimento

Sexo	80% masculino 20% feminino
Idade	60% entre 21 a 25 anos 20% entre 26 a 30 anos 20% entre 16 a 20 anos
Nível de Escolaridade	100% ensino superior incompleto
Você utiliza o computador com qual frequência	100% frequentemente
Você utiliza dispositivos móveis com qual frequência	80% frequentemente 20% às vezes
Qual o seu grau de familiaridade com Realidade aumentada	80% já utilizei 20% conheço, mas nunca utilizei

Com o resultado do Quadro 11, é possível notar que boa parte dos usuários já possuem um certo grau de familiaridade com aplicações de Realidade Aumentada e utilizam com frequência dispositivos móveis.

Para a realização do experimento, os usuários seguiram os passos descritos no questionário demonstrado no Apêndice C e a cada passo o usuário informou se conseguiu concluir ou não a atividade proposta. Os resultados são demonstrados no Quadro 12.

Quadro 12 - Repostas do questionário de sim/não

Troque o objeto virtual no marcador da "Fábrica de objetos". A tarefa foi realizada?	100% sim
Utilize o cubo para segurar o objeto virtual	100% sim
Com o cubo, coloque o objeto virtual no cenário usando o marcador "Alvo". A tarefa foi realizada?	100% sim
Conecte o marcador "Opção" ao objeto virtual. A tarefa foi realizada?	100% sim
Selecione "Rotação Y" no marcador "Opção". A tarefa foi realizada?	100% sim
Conecte o marcador "Modificador" ao marcador "Opção". A tarefa foi realizada?	100% sim
Adicione 10 ao valor do modificador. A tarefa foi realizada?	100% sim
Ative o botão "Motor". A tarefa foi realizada?	100% sim
Desconecte o marcador "Modificador" e o marcador "Opção". A tarefa foi realizada?	100% sim
Desconecte o marcador "Opção" do objeto virtual. A tarefa foi realizada?	100% sim
Selecione o objeto virtual com o cubo e remova movendo até o marcador "Lixeira". A tarefa foi realizada?	100% sim

Conforme pode ser visto no Quadro 12, todas as tarefas foram realizadas, durante a execução das atividades, nenhum usuário demonstrou dificuldade em entender as tarefas sugeridas.

Após os usuários realizarem as tarefas, cada usuário respondeu a um questionário de usabilidade do aplicativo. Os resultados obtidos estão detalhados no Quadro 13.

Quadro 13 - Repostas do questionário de usabilidade

Das atividades solicitadas, quantas atividades você conseguiu executar sem auxílio?	60% todas 40% a maior parte
De modo geral, você achou o protótipo intuitivo e fácil de usar?	80% sim 20% não
Qual é a sua avaliação da aplicação?	60% bom 40% muito bom
Qual foi a sua maior dificuldade utilizando a aplicação?	Pegar objetos soltos no ar; Cubo orientação, fazer maior width; Linkar os objetos; Botões; Pegar os objetos.

Conforme pode ser visto no Quadro 13, o aplicativo obteve resultados geralmente positivos em relação à intuitividade. Porém alguns usuários demonstraram dificuldade com algumas funcionalidades. Um dos usuários recomendou aumentar a linha virtual do marcador `Cubo`. Esta recomendação foi desenvolvida, aumentando a linha virtual em um terço do seu tamanho original. Outra dificuldade foi a sensibilidade dos botões virtuais. Para isso foi aumentado a sensibilidade de todos os botões virtuais do aplicativo. A etapa que todos os

usuários demonstraram dificuldades em realizar é para se localizar na seleção de atributos utilizando o marcador `Opções`. Porém, todos os usuários mostraram interesse no conceito da aplicação.

3.4.2 Comparação entre os trabalhos correlatos e o trabalho desenvolvido

Nesta seção é apresentada uma comparação dos trabalhos correlatos descritos na seção 2.4 com o trabalho desenvolvido. O Quadro 14 apresenta um comparativo entre as características mais relevantes dos trabalhos correlatos.

Quadro 14 - Comparativo entre os trabalhos correlatos

Características/Trabalhos Relacionados	Aplicativo desenvolvido	Lira (2015)	Rodrigues, Sato e Botega (2012)	Lee et al. (2004)
possui Interface de Usuário Tangível	Sim		Sim	Sim
criação de RA baseada em marcadores	Sim	Sim	Sim	Sim
permite geração de objetos 3D virtuais	Sim	Sim	Sim	Sim
detecção de múltiplos marcadores	Sim	Sim	Sim	Sim
utiliza o SDK Vuforia	Sim	sim		
interação entre objetos virtuais	Sim			Sim

Como pode ser visto no Quadro 14, o trabalho desenvolvido, o trabalho de Rodrigues, Sato e Botega (2012) e o trabalho de Lee et al. (2004) possuem Interface de Usuário Tangível. Entretanto, somente o trabalho desenvolvido e o trabalho de Lira (2015) utilizam a SDK Vuforia. O trabalho de Lee et al. permite, assim como o trabalho desenvolvido, a interação entre vários marcadores. Percebe-se também que o trabalho desenvolvido possui uma maior similaridade com trabalho de Lee et al. (2004), exceto pelo uso da SDK Vuforia. Em todos os trabalhos nota-se que são criados objetos tridimensionais virtuais utilizando marcadores que podem ser detectados simultaneamente.

4 CONCLUSÕES

Este trabalho apresentou a criação de uma interface de usuário tangível com o uso da Realidade Aumentada, utilizando objetos reais para interagir com objetos virtuais usando o conceito de Realidade Aumentada tangível apresentada por Shaer e Hornecker (2010), que é descrito na seção 2.1.

Os objetivos deste trabalho foram criar um ambiente de Realidade Aumentada em que o usuário consiga manipular objetos virtuais utilizando objetos reais obtendo retorno visual destas alterações e também a interação entre objetos virtuais em um ambiente virtual aumentado. Para isso, foi utilizado a plataforma de desenvolvimento de jogos e aplicativos Unity em conjunto da biblioteca Vuforia, tendo este fornecido os recursos necessários para a implementação da Realidade Aumentada do aplicativo desenvolvido. Essa combinação entre o Unity e Vuforia se tornou adequada para a implementação, pois, forneceu os recursos necessários para o desenvolvimento do trabalho.

A utilização do Unity em conjunto com Vuforia, possibilitou o reconhecimento de vários marcadores e a geração de objetos virtuais tridimensionais utilizando a câmera de um dispositivo móvel. O Unity contribuiu com o uso de suas funcionalidades para posicionamento de objetos virtuais e a utilização de parentalidade para movimentar objetos virtuais filhos caso o objeto virtual pai seja movido, e assim permitindo que objetos sejam colocados em um cenário virtual que permitia modificar os seus atributos.

Os objetivos propostos foram atingidos com resultados satisfatórios, com isso permitindo ao usuário um entendimento do conceito da Interface de usuário tangível. Nos testes, foram percebidos a infamiliaridade dos usuários com a interação da realidade virtual utilizando objetos reais.

Por fim, o trabalho se mostrou de grande importância para o desenvolvimento científico devido a pouca exploração do conceito da interface de usuário tangível, pois, gerou uma aplicação que poderá ser estendida para outros trabalhos que utilizam Realidade Aumentada ou interface tangível como foco da aplicação. Pode ser destacado que, por este trabalho ter sido desenvolvido com o Unity e Vuforia, demonstra a facilidade de gerar o aplicativo para outras plataformas, como por exemplo, o iOS ou PC. Tendo como base o estudo realizado, outros trabalhos poderão ser desenvolvidos a partir dos conceitos apresentados neste trabalho. Com o trabalho desenvolvido, é possivel o desenvolvimento de aplicações para modelagem de objetos tridimensionais ou modelagem de plantas de imóveis.

4.1 EXTENSÕES

Como sugestão de extensões para melhoria e continuidade do trabalho, propõe-se:

a) tornar a interface mais intuitiva, adicionando uma lista no marcador de opções;
b) criar marcadores de condições lógicas como, por exemplo, "or", "and" e "not";
c) criar mais atributos para serem utilizados na alteração dos objetos virtuais, como animações, cor e textura;
d) criar mais tipos de objetos virtuais como, por exemplo, sons e luzes;
e) criar uma funcionalidade em que o objeto virtual fique "grudado" a um novo marcador;
f) permitir que vários usuários pudessem visualizar o ambiente virtual aumentado através de uma rede Wifi ou Bluetooth;
g) criar uma funcionalidade que permite associar os objetos virtuais no ambiente virtual aumentado com filho de outros objetos virtuais;
h) adicionar compatibilidade com a SDK Cardboard ou outros óculos de realidade virtual Aumentada.

REFERÊNCIAS

APPLE. **AR Magic Mirror**. [S.l.], 2011. Disponível em: < https://itunes.apple.com/us/app/ar-magic-mirror/id427091769?mt=8>. Acesso em: 27 ago. 2015.

ARTOOLKIT. **Artoolkit home page**. [S.l.], 2012a. Disponível em: <http://www.hitl.washington.edu/artoolkit/>. Acesso em: 27 ago. 2015.

_____. **Artoolkit Documentation (How does Artoolkit work?)**. 2012b. Disponível em: <http://www.hitl.washington.edu/artoolkit/documentation/userarwork.htm>. Acesso em: 27 ago. 2015.

CARMIGNIANI, Julie. FURHT, Borko. **Handbook of Augmented Reality**. New York, U.S.A: Springer-Verlag New York, 2011. Disponível em: <https://books.google.com.br/books?id=fG8JUdrScsYC>. Acesso em: 24 ago. 2015.

CAWOOD, Stephen; FIALA, Mark. **Augmented Reality**: A Practical Guide. [S.l.]: Pragmatic Bookshelf, 2007.

JACKO, Julie A., **Human Computer Interaction Handbook:** Fundamentals, Evolving Technologies, and Emerging Applications, Third Edition. U.S.A: CRC Press, 2012. <https://books.google.com.br/books?id=dVrRBQAAQBAJ>. Acesso em: 26 ago. 2015.

KIRNER, Claudio; SISCOUTTO, Robson. **Realidade Virtual e Aumentada:** Conceitos, Projeto e Aplicações. Porto Alegre: Editora SBC, 2007.

LEE, Gun A et al. **Immersive Authoring of Tangible Augmented Reality Applications.** Washington, DC, U.S.A: IEEE Computer Society Washington, 2004. Disponível em: <http://www.hitlabnz.org/administrator/components/com_jresearch/files/publications/2004-ISMAR-Iatar.pdf>. Acesso em: 05 set. 2015.

LIRA, Diego A. **Biblioteca para desenvolvimento de jogos multijogadores de realidade aumentada para Android.** Blumenau, 2015. Disponível em: <https://app.box.com/s/iaj3h8s4h2u9zwohh6xy1jw5rosrstsc>. Acesso em: 26 ago. 2015.

PAPAGIANNIS, Helen. **AR Hanging Mobile**. [S.l], 2010. Disponível em: <http://augmentedstories.com/2010/10/13/mobile/>. Acesso em: 27 ago. 2015.

QUALCOMM. **VUFORIA AUGMENTED REALITY**. [S.1.], 2015. Disponível em: <https://www.qualcomm.com/products/vuforia>. Acesso em: 28 ago. 2015.

RAJAWALI. **RajawaliVuforia**. [S.l.], 2015. Disponível em: <https://github.com/Rajawali/RajawaliVuforia>. Acesso em: 22 out. 2015.

RODRIGUES, Fábio; SATO, Fernando; BOTEGA, Leonardo. Integrando Interface Tangível com Técnicas de Realidade Aumentada para Ampliar a Experiência Interativa do Usuário. In: SIBGRAPI, 25, 2012, Ouro Preto, 2012. Marília. Disponível em: <http://www.decom.ufop.br/sibgrapi2012/eproceedings/wuw/102818_1.pdf>. Acesso em: 25 ago. 2015.

SHAER, Orit; HORNECKER, Eva, **Tangible User Interfaces**. Hanover, Massachusetts, U.S.A: Now Publishers Inc, 2010. <http://books.google.com.br/books?id=vh_tCfvK4M4C>. Acesso em: 25 ago. 2015.

UNITY. **Um editor repleto de recursos e altamente flexível.** [S.l.], 2015a. Disponível em: <http://unity3d.com/pt/unity/editor>. Acesso em: 28 ago. 2015.

_____. **O software líder global da indústria de jogos**. [S.l.], 2015b. Disponível em: <https://unity3d.com/pt/public-relations>. Acesso em: 28 ago. 2015.

_____. **Aprenda com Unity**. [S.1], 2015c. Disponível em: <https://unity3d.com/pt/learn>. Acesso em: 28 ago. 2015.

ZANDOMENEGUI, Ana L. A. O et al. **Conceitos e Práticas em Ambiente Virtual de Aprendizagem Inclusivo**. São Paulo: Pimenta Cultura, 2014. Disponível em: <https://books.google.com.br/books?id=1ettAwAAQBAJ>. Acesso em: 27 ago. 2015.

APÊNDICE A – Detalhamento dos casos de uso

A seguir estão descritos detalhadamente os casos de uso, com a descrição, cenário, pré e pós condições. O caso de uso `UC01 - Selecionar opção para iniciar` descreve o processo de iniciar a aplicação. Detalhes deste caso de uso estão descritos no Quadro 15.

Quadro 15 - Caso de uso `UC01 - Selecionar opção para iniciar`

Requisitos atendidos	Nenhum.
Descrição	Este caso de uso tem por objetivo carregar o cenário do aplicativo que é responsável pela interface de usuário tangível.
Pré-Condição	Estar com o aplicativo aberto
Cenário Principal	1. O `Usuario` irá tocar no botão `Iniciar`; 2. É apresentada uma imagem de carregamento.
Pós-Condição	A câmera será habilitada para a utilização da realidade aumentada

O caso de uso `UC02 - Selecionar opção para abrir o questionário` descreve o processo de abrir a página do questionário. Detalhes deste caso de uso estão descritos no Quadro 16.

Quadro 16 - Caso de uso `UC02 - Selecionar opção para abrir o questionário`

Requisitos atendidos	Nenhum.
Descrição	Este caso de uso tem por objetivo carregar o questionário utilizado para a coleta de informação da aplicação.
Pré-Condição	Estar com o aplicativo aberto
Cenário Principal	1. O `Usuario` irá tocar no botão `questionário`; 2. É aberta uma página na internet com o questionário online.
Pós-Condição	O `Usuário` poderá preencher o questionário.

O caso de uso `UC03 - Movimentar objetos virtuais com o Cubo` descreve o processo de selecionar e movimentar um objeto virtual utilizando o marcador `Cubo`. Detalhes deste caso de uso estão descritos no Quadro 17.

Quadro 17 - Caso de uso UC03 – Movimentar objetos virtuais com o Cubo

Requisitos atendidos	RF02
Descrição	Este caso de uso tem por objetivo permitir a movimentação de objetos virtuais utilizando um objeto real em formato de cubo
Pré-Condição	Estar com o aplicativo aberto
Cenário Principal	1. O Usuário irá selecionar o objeto a ser movido; 2. O Usuário irá movimentar o objeto virtual utilizando o cubo real.
Pós-Condição	O Usuário poderá movimentar o objeto virtual no ambiente virtual.

O caso de uso UC04 – Selecionar objeto virtual descreve o processo de selecionar um objeto virtual e um atributo com o marcador Opções. Detalhes deste caso de uso estão descritos no Quadro 18.

Quadro 18 - Caso de uso UC04 – Selecionar objeto virtual

Requisitos atendidos	RF01, RF04
Descrição	Este caso de uso tem por objetivo permitir a seleção de objetos virtuais com uso de objetos reais e selecionar um atributo a ser modificado
Pré-Condição	Estar com um objeto virtual no ambiente virtual
Cenário Principal	1. O Usuário irá selecionar o objeto utilizando o marcador de opções; 2. O Usuário deverá usar os botões virtuais para selecionar o atributo do objeto virtual.
Pós-Condição	O objeto virtual terá um atributo selecionado para ser modificado.

O caso de uso UC05 – Colocar objeto virtual no ambiente virtual descreve o processo de selecionar um objeto virtual e adicioná-lo ao ambiente virtual. Detalhes deste caso de uso estão descritos no Quadro 19.

Quadro 19 - Caso de uso UC05 – Colocar objeto virtual no ambiente virtual

Requisitos atendidos	RF03
Descrição	Este caso de uso tem por objetivo permitir incluir um objeto virtual no ambiente virtual aumentado.
Pré-Condição	Estar com o objeto virtual selecionado com o cubo.
Cenário Principal	1. O Usuário, tendo o objeto virtual selecionado, irá posicionar o objeto virtual acima do marcador alvo; 2. O Usuário irá aguardar um determinado tempo até o objeto ser posicionado no ambiente virtual.
Pós-Condição	O objeto virtual estará posicionado no ambiente virtual para ser selecionado pelo marcador opções.

O caso de uso UC06 – Modificar objeto virtual descreve o processo de modificar os atributos do objeto virtual. Detalhes deste caso de uso estão descritos no Quadro 20.

Quadro 20 - Caso de uso UC06 – Modificar objeto virtual

Requisitos atendidos	RF05, RF06
Descrição	Este caso de uso tem por objetivo permitir a modificação do objeto virtual.
Pré-Condição	Estar com o objeto virtual selecionado com o marcador de opções.
Cenário Principal	1. O Usuário irá selecionar o objeto virtual com o marcador de opções; 2. O Usuário irá conectar o marcador opções com o marcador modificador; 3. O Usuário irá selecionar o componente a ser modificado pelo marcador de opções e alterar o valor pelo marcador modificador.
Pós-Condição	O objeto virtual será modificado com os valores selecionados pelo Usuário.

E por fim, o caso de uso UC07 – Remover objeto virtual descreve o processo de remover um objeto virtual. Detalhes deste caso de uso estão descritos no Quadro 21.

Quadro 21 - Caso de uso UC07 – Remover objeto virtual

Requisitos atendidos	Nenhum.
Descrição	Este caso de uso tem por objetivo permitir a remoção de objetos virtuais selecionados com o cubo real.
Pré-Condição	Estar com o objeto selecionado com o cubo.
Cenário Principal	1. O Usuário, tendo o objeto virtual selecionado com o cubo, irá movê-lo ao marcador lixeira; 2. O Usuário irá aproximar o objeto virtual até a lixeira virtual e aguardar;
Pós-Condição	O objeto será removido do ambiente virtual.

APÊNDICE B – APRESENTAÇÃO DO APLICATIVO NA SEMANA ACADÊMICA DE COMPUTAÇÃO 2016

Na quarta, dia 20/05/2016 houve o evento Semana acadêmica de computação na Fundação Universidade Regional de Blumenau. Neste evento, foi apresentado o aplicativo desenvolvido para o publico na apresentação do projeto da Caixa de areia interativa usando realidade virtual aumentada. Na demonstração deste aplicativo, foi apresentado o conceito da Interface de Usuário Tangível. Para a apresentação, foi utilizada a caixa de areia para simular um ambiente virtual com profundidade. A Figura 45 demonstra a apresentação do projeto da Caixa de areia utilizando realidade virtual aumentada.

Figura 45 - Apresentação do aplicativo deste trabalho

APÊNDICE C – Roteiro e questionário de avaliação de usabilidade da aplicação

Neste apêndice apresenta o questionário e o roteiro utilizado para os usuários testarem o aplicativo. O questionário de perfil é mostrado na Figura 46. As Figura 47 e Figura 48 mostram o questionário de atividades, contendo as tarefas que auxilia o usuário a usar aplicação. Por fim, na Figura 49 é mostrado o questionário de usabilidade.

Figura 46 - Questionário de usuário

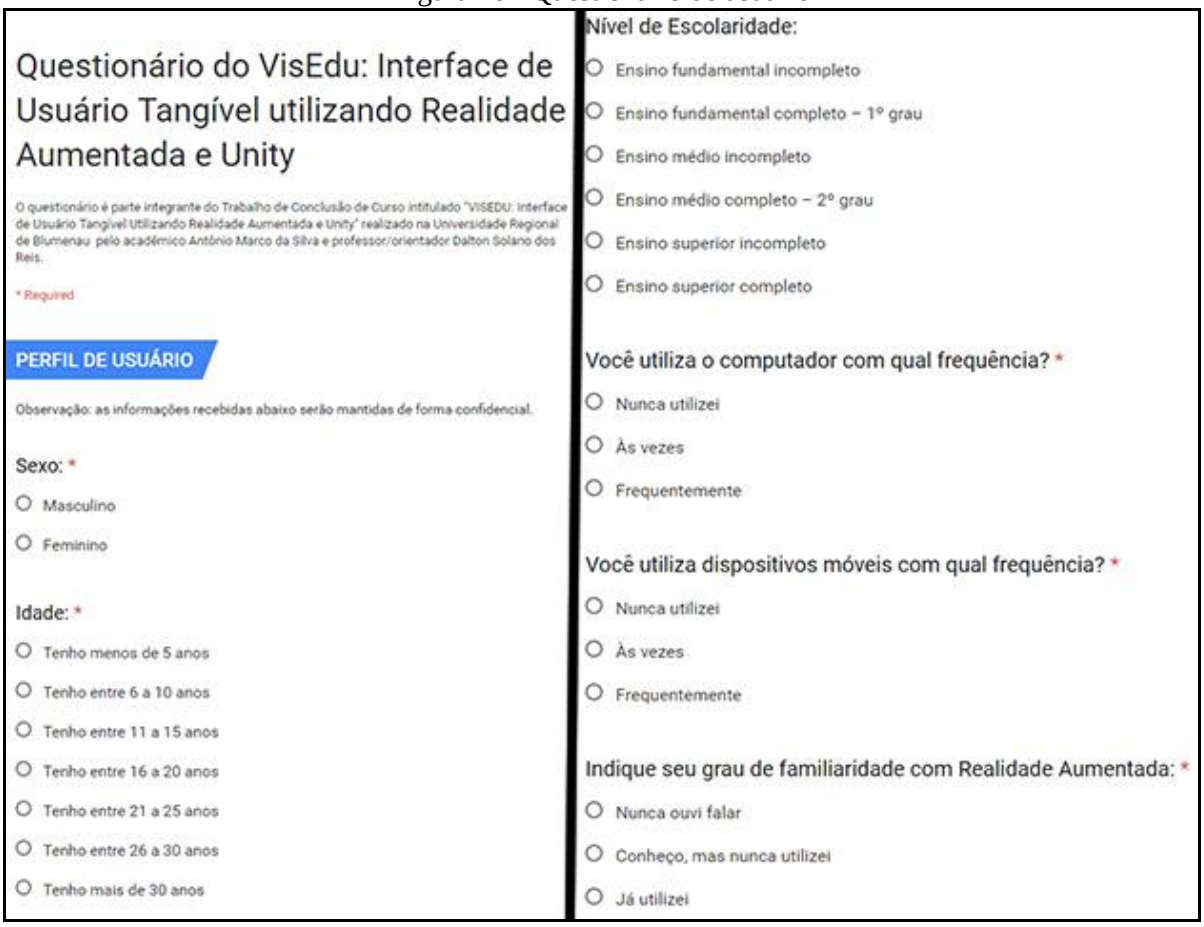


Figura 47 - Parte do questionário de atividades

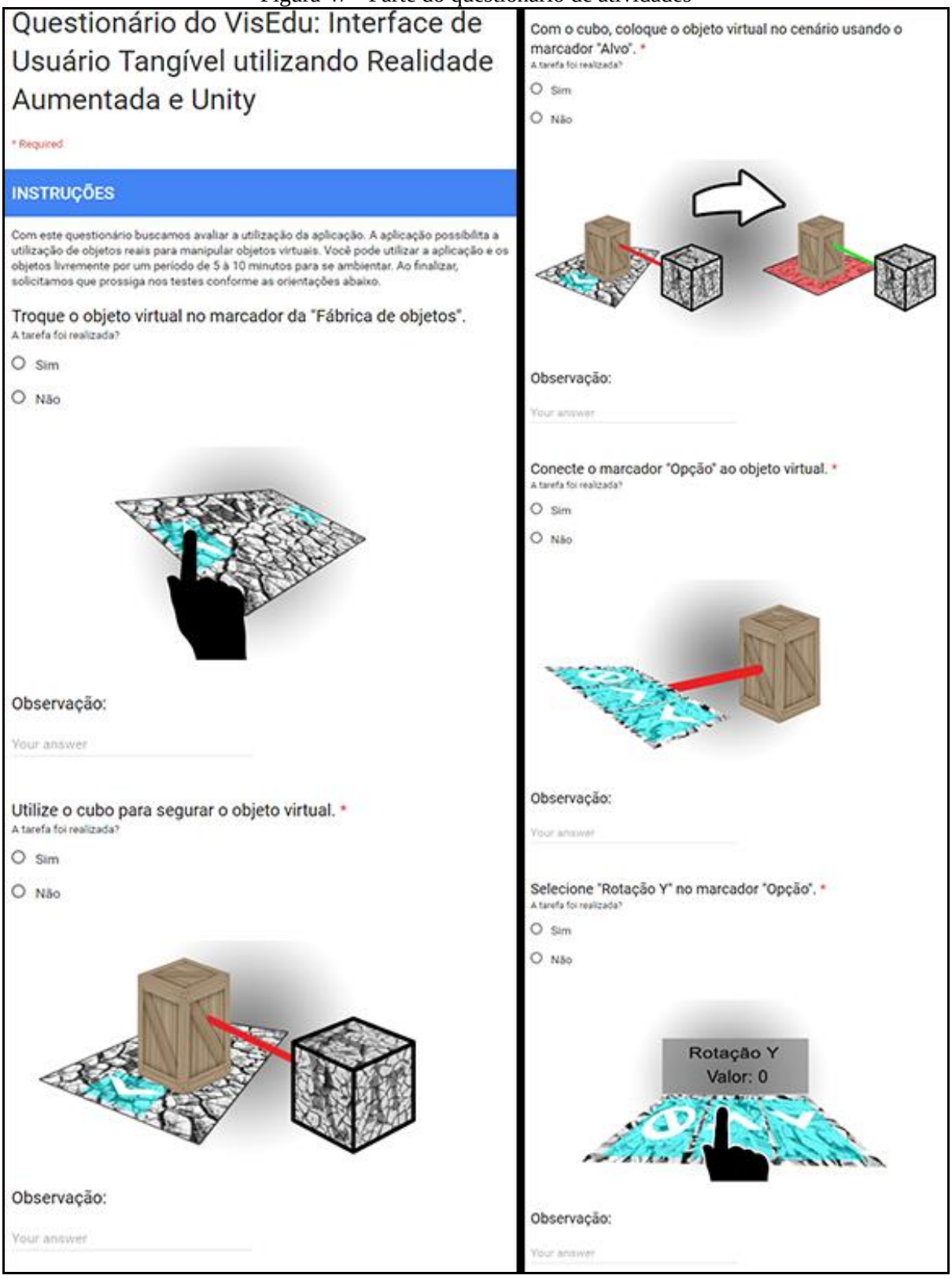


Figura 48 - Parte do questionário de atividades

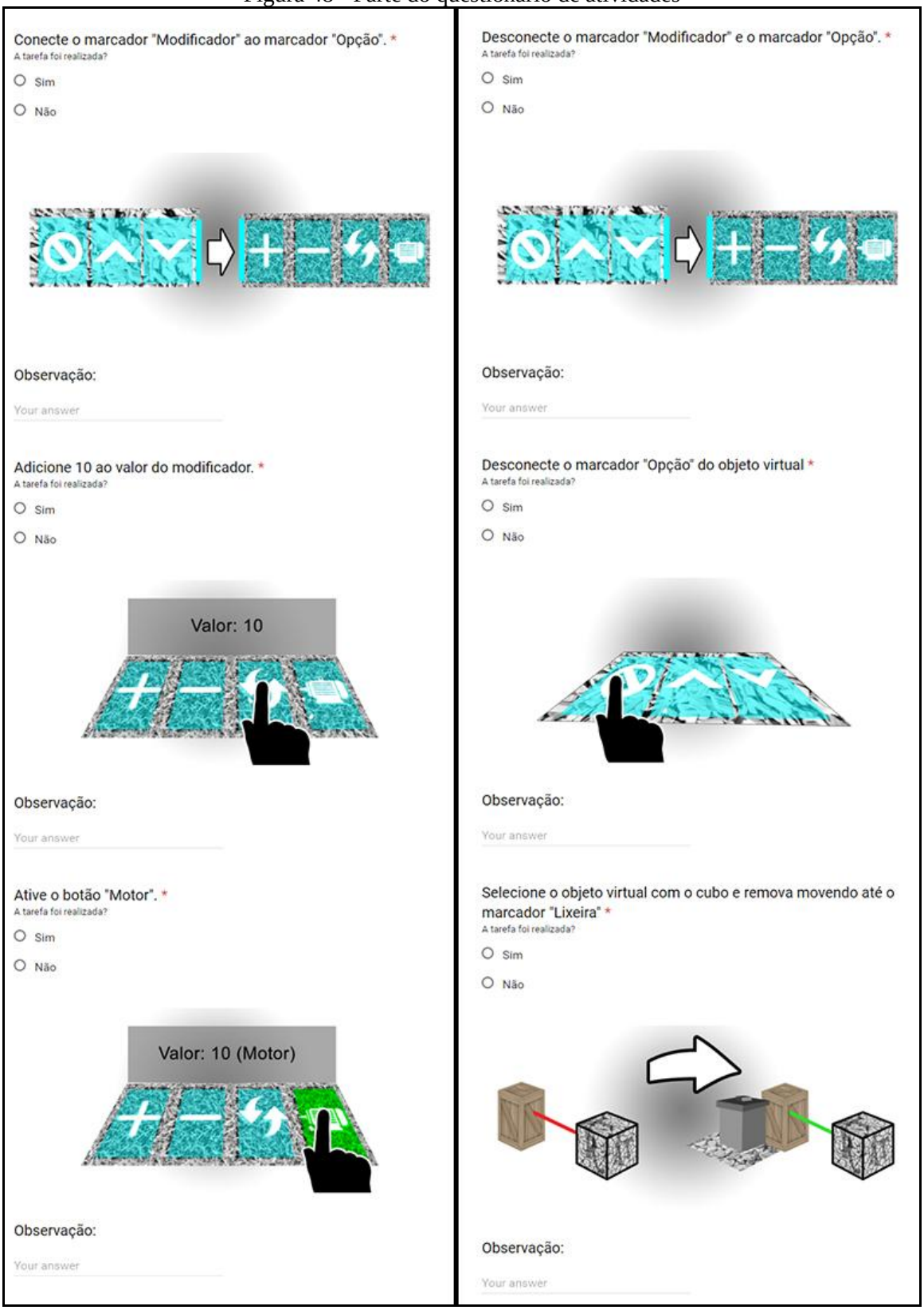
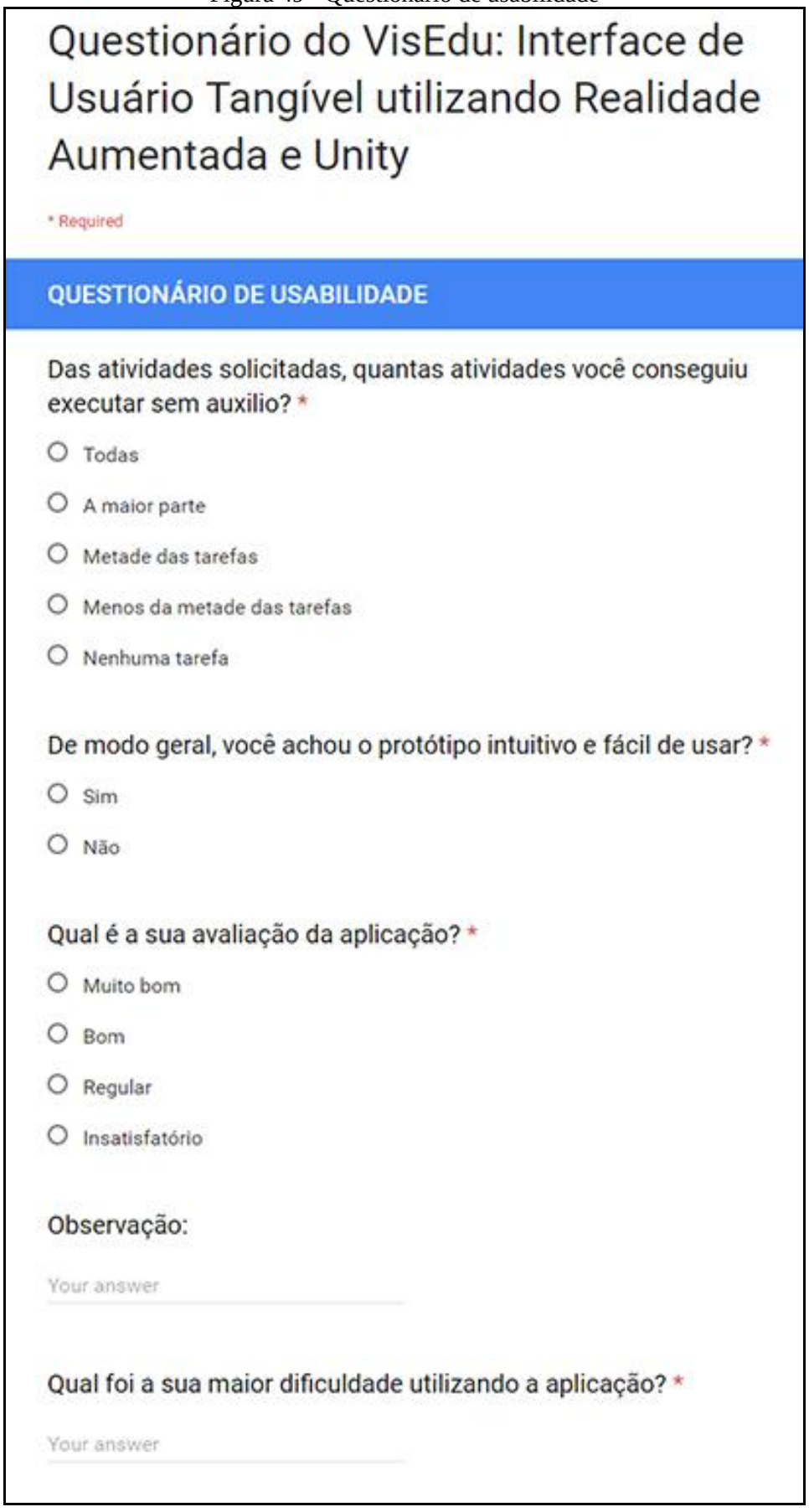


Figura 49 - Questionário de usabilidade



Questionário do VisEdu: Interface de Usuário Tangível utilizando Realidade Aumentada e Unity

* Required

QUESTIONÁRIO DE USABILIDADE

Das atividades solicitadas, quantas atividades você conseguiu executar sem auxilio? *

- 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

Observação:

Your answer

Qual foi a sua maior dificuldade utilizando a aplicação? *

Your answer