UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

VISEDU: APLICATIVO DE REALIDADE AUMENTADA USANDO OBJETOS INTERATIVOS

JÚLIO CÉSAR DOS SANTOS

BLUMENAU
2015

2015/2-14

JÚLIO CÉSAR DOS SANTOS

VISEDU: APLICATIVO DE REALIDADE AUMENTADA USANDO OBJETOS INTERATIVOS

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

Prof. Dalton Solano dos Reis, Mestre - Orientador

BLUMENAU
2015

2015/2-14

VISEDU: APLICATIVO DE REALIDADE AUMENTADA USANDO OBJETOS INTERATIVOS

Por

JÚLIO CÉSAR DOS SANTOS

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, Mestre – FURB
Membro:	Prof. Maurício Capobianco Lopes, Doutor – FURB

Blumenau, 08 de dezembro de 2015

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

AGRADECIMENTOS

Aos meus pais Osni e Rosaine pelo incentivo e confiança.

A minha namorada Nayara pelo auxilio e paciência para o desenvolvimento deste trabalho.

Ao meu primo Marco pelo interesse no desenvolvimento deste trabalho.

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

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

A persistência é o caminho do êxito.

Charles Chaplin

RESUMO

Este trabalho apresenta o desenvolvimento de um aplicativo para a plataforma Android que utiliza Realidade Aumentada inserida em um ambiente simples e interativo de trânsito, permitindo um entretenimento diversificado para o usuário. O aplicativo foi desenvolvido com a utilização da plataforma de criação de jogos e aplicativos Unity 3D em conjunto com a biblioteca de Realidade Aumentada Vuforia. Durante o desenvolvimento do trabalho foram implementadas diferentes possibilidades de interatividade do usuário com os objetos virtuais gerados a partir da Realidade Aumentada e, com base nos resultados obtidos, foi selecionada a mais adequada para ser utilizada pelo aplicativo. Além disso, são apresentados conceitos, técnicas e componentes utilizados para a implementação da Realidade Aumentada e da interatividade com os objetos virtuais gerados. Como resultado, foi desenvolvido um aplicativo que permite ao usuário imergir em um cenário de Realidade Aumentada através da visualização da imagem de um mapa pela câmera de um dispositivo Android, e interagir com objetos virtuais gerados no cenário.

Palavras-chave: Android. Interatividade. Realidade aumentada. Unity 3D. Vuforia.

ABSTRACT

This work presents the development of an Android Application that uses Augmented Reality in a simple and interactive environment of vehicle traffic, allowing diversified forms of entertainment for the user. The application was developed using the game creation platform Unity 3D and the library of Augmented Reality Vuforia. Throughout the development of this work, different possibilities of user interactivity with virtual objects generated using Augmented Reality were implemented, and based on the obtained results, the most adequate were selected to be used by the application. Besides, there are presented concepts, techniques and components used in the implementation of the Augmented Reality and the interactivity with the generated virtual objects. As result, it was developed an immersive application in an Augmented Reality scenario through the visualization of a map image by an Android device camera, that interacts with virtual objects generated in the scenario.

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

LISTA DE FIGURAS

Figura 1 - Objetos virtuais gerados: (A) Casa, (B) Animal. 20
Figura 2 - Editor gráfico do Unity 22
Figura 3 - Janela `Hierarchy` com e sem *parenting* 23
Figura 4- Janela `Inspector` 24
Figura 5 - `Box Collider` do cubo 25
Figura 6 - Jogo ARhrrrr! 28
Figura 7 - Jogo ARDefender 29
Figura 8 – Aplicativo eyeBuy 30
Figura 9 - Diagrama de casos de uso do aplicativo 32
Figura 10 - Classes desenvolvidas para o aplicativo 34
Figura 11 - Diagrama de atividades das funções executadas pelo usuário 35
Figura 12 - Diagrama de atividades do sistema 36
Figura 13 - Importando Vuforia no Unity 37
Figura 14 - Adicionando um alvo de imagem 38
Figura 15 - Configurações básicas para a `ARCamera` 39
Figura 16 - Configuração da `Image Target` 39
Figura 17 – Elementos do aplicativo 40
Figura 18 - Atributos para o controle das barras dinâmicas 40
Figura 19 - Opções definidas no marcador 42
Figura 20 - Componentes do carro 43
Figura 21 - Tela inicial do aplicativo 47
Figura 22 - Preferencias do Unity 48
Figura 23 - Configurações na Build Settings do Unity 49
Figura 24 - Visualização do mundo 3D gerado a partir do marcador (mapa) 50
Figura 25 - Colisão do carro com uma casa 51
Figura 26 - Incremento da barra de gasolina 51
Figura 27 - Incremento da barra de mecânica 52
Figura 28 - Seleção de opções após a barra de mecânica se esgotar 52
Figura 29 - Tela em modo retrato 53
Figura 30 – Visualização do cenário real sem e com a `Scene ControlePorCarroVirtual` 54

Figura 31 – Visualização do cenário real sem e com a `Scene ControlePorCarroReal` 55
Figura 32 - Visualização de frente dos carrinhos utilizados 55
Figura 33 - Visualização do cenário real sem e com a `Scene ControlePorJostick` 56
Figura 34 - Demonstração do aplicativo deste trabalho 66
Figura 35 - Marcador utilizado para a exposição 67

LISTA DE QUADROS

Quadro 1 - Caso de uso UC01 – Visualizar marcador para impressão 32
Quadro 2 - Caso de uso UC02 – Visualizar instruções básicas sobre o trânsito 33
Quadro 3 – Caso de uso UC03 – Visualizar página do projeto 33
Quadro 4 - Caso de uso UC04 – Selecionar opção para iniciar 33
Quadro 5 - Caso de uso UC05 – Movimentar *joystick* 33
Quadro 6 - Criação de componentes de interface dinâmicos 41
Quadro 7 - Métodos para o controle das barras de gasolina e mecânica 42
Quadro 8 - Controle do carro 44
Quadro 9 - Tratamento de colisões para decrementar a barra de mecânica 45
Quadro 10 - Tratamento de colisão para incrementar barra de gasolina e mecânica 46
Quadro 11 - Método para carregar a tela inicial do aplicativo 47
Quadro 12 - Métodos da tela inicial 48
Quadro 13 - Comparativo dos trabalhos correlatos e este trabalho 59

LISTA DE TABELAS

Tabela 1- Características dos modelos em 3D do aplicativo 57
Tabela 2 - Consumo de memória das `Scenes` 58
Tabela 3 - Taxa de FPS das `Scenes` 59

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

FPS – *Frames Per Second*

GB – *Gigabyte*

GHz – *Gigaheartz*

IDE – *Integrated Development Environment*

JDK – *Java Development Kit*

JPG – *Joint Photographic Group*

MB – *Megabyte*

PNG – *Portable Network Graphic*

RA – Realidade Aumentada

RGB – *Red Green Blue*

SDK – *Software Development Kit*

SUMÁRIO

1 INTRODUÇÃO 16
1.1 OBJETIVOS 16
1.2 ESTRUTURA 17
2 FUNDAMENTAÇÃO TEÓRICA 18
2.1 REALIDADE AUMENTADA 18
2.1.1 Interatividade 18
2.1.2 Inserção de informações virtuais no ambiente real 19
2.2 UNITY 3D 20
2.2.1 Elementos básicos 21
2.2.2 Editor gráfico 22
2.2.3 Controle de física e colisão (interatividade) 24
2.2.4 Vuforia 25
2.3 TRABALHOS CORRELATOS 27
2.3.1 ARhrrrr! 27
2.3.2 ArDefender 28
2.3.3 eyeBuy 29
3 DESENVOLVIMENTO 31
3.1 REQUISITOS 31
3.2 ESPECIFICAÇÃO 32
3.2.1 Diagrama de Casos de uso do aplicativo 32
3.2.2 Diagrama de classes do aplicativo 34
3.2.3 Diagramas de atividades 35
3.3 IMPLEMENTAÇÃO 36
3.3.1 Técnicas e ferramentas utilizadas 37
3.3.1.1 Inicio do projeto 37
3.3.1.2 Aplicativo 39
3.3.1.3 Tela inicial 47
3.3.1.4 Plataforma disponibilizada para o aplicativo 48
3.3.2 Operacionalidade da implementação 49
3.4 RESULTADOS E DISCUSSÕES 53
3.4.1 Cenários 53

3.4.2 Testes.....58
3.5 COMPARATIVO DOS TRABALHOS CORRELATOS 59
4 CONCLUSÕES.....61
4.1 EXTENSÕES 62
REFERÊNCIAS 63
APÊNDICE A – APRESENTAÇÃO DO APLICATIVO NO XI SEMINÁRIO INTEGRADO DAS LICENCIATURAS 66

1 INTRODUÇÃO

Quando os jogos eletrônicos surgiram, há décadas atrás, as únicas opções eram pouco interativas, não atraíam e não possuíam uma boa tecnologia. O uso da Realidade Aumentada nas indústrias de jogos mudou essa situação, tornando-os muito mais interativos. O recurso trouxe inovação, permitindo misturar a realidade com o mundo virtual (KARASINSKI, 2009). Para esta possibilidade são necessários dispositivos capazes de transmitir imagens, como por exemplo, *smartphones* com câmeras e um software especifico capaz de reconhecer objetos reais e transportá-los para um contexto virtual (HAUSTSCH, 2009).

A Realidade Aumentada inovou o conceito de jogos interativos, deixando-os mais interessantes, pois o que agrada o jogador é o fato dele se sentir dentro do jogo, fazendo parte da história. Um aspecto do sucesso dos jogos está em seus gráficos, porém não é o bastante para agradar o público de jogadores. As inovações tecnológicas e as formas de interação com os jogos fazem com que o jogador possa usufruir de novas maneiras de visualizar um jogo. Agora que Realidade Aumentada está cada vez mais presente nos jogos, podem-se esperar inovações em maiores proporções (KARASINSKI, 2009).

Atualmente a Realidade Aumentada não tem um limite de aplicações, sendo utilizada no entretenimento como na criação de jogos interativos, na área educativa em materiais educacionais, campanhas publicitárias, melhorias na área da saúde, na indústria automobilística, além de outras alternativas que ainda serão criadas (HAUSTSCH, 2009). Com o aumento da criação dos jogos e aplicativos interativos, as empresas de celulares passaram a melhorar a interatividade dos aparelhos, como no caso do iPhone, que mostra-se cada vez melhor nestes aspectos (KARASISNKI, 2009).

Diante deste contexto, foi realizado o desenvolvimento de um aplicativo com o objetivo de simular um cenário simples de trânsito para demonstrar ao usuário, uma maneira diferente de entretenimento através do uso da Realidade Aumentada. Para este aplicativo, foi disponibilizado um mundo virtual 3D com objetos interativos. Além disto, nota-se que o aplicativo possui características semelhantes a um jogo, como a interatividade e a simulação. Também foi disponibilizado através do aplicativo, um conteúdo com instruções básicas sobre o trânsito.

1.1 OBJETIVOS

O objetivo deste trabalho é criar um aplicativo de Realidade Aumentada para a plataforma móvel Android. Os objetivos específicos do trabalho são:

a) definir um mundo virtual 3D usando um mapa como marcador;

b) simular formas de interagir com os objetos deste mundo;
c) ter um retorno visual da movimentação e colisão dos objetos no mundo virtual.

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 descreve conceitos e áreas da Realidade Aumentada. A seção 2.2 apresenta o Unity 3D, seus elementos básicos, seu editor gráfico, o Vuforia e a integração dessas duas tecnologias. Por fim, a seção 2.3 traz um breve estudo de três trabalhos correlatos.

2.1 REALIDADE AUMENTADA

A Realidade Aumentada (RA) é caracterizada como uma variação da Realidade Virtual. Na Realidade Virtual o usuário é transportado para um ambiente virtual sem a possibilidade de visualização do mundo real. Já na RA o usuário tem uma maior percepção da realidade, tornando o ambiente real e o virtual em um só, fazendo com que os objetos reais sejam sobrepostos ou compostos por virtuais. Logo a RA complementa a realidade, ao invés de substituí-la por completo, fazendo com que objetos reais e virtuais coexistam no mesmo espaço (AZUMA, 1997, p.2).

Na RA, o usuário tem a sua percepção elevada e também a sua interatividade com o ambiente real. Para isto, objetos virtuais exibem informações as quais o usuário não consegue visualizar com os seus próprios sentidos. Estas informações auxiliam a execução de tarefas do dia-a-dia em áreas como manutenção e reparação de equipamentos, visualizações para tratamentos cirúrgicos e também para o entretenimento de pessoas (AZUMA, 1997, p.3).

Segundo Azuma (1997, p. 2-3), para a existência da RA deve-se ter a combinação do ambiente real com o ambiente virtual, a interatividade em tempo real do usuário com a realidade e a percepção dos objetos sobrepostos em 3D como reais. Dentre os principais componentes da RA pode-se citar a interatividade e a inserção de informações virtuais no ambiente real.

2.1.1 Interatividade

A interatividade pode ser classificado como “[...] qualquer coisa cujo funcionamento permite ao seu usuário algum nível de participação ou troca de ações.” (SILVA, 2001, não paginada). Nos ambientes dos jogos, esta característica permite ao jogador participar ou interferir de alguma maneira, fazendo com que suas ações influenciem diretamente no contexto do jogo.

No inicio da criação dos jogos eletrônicos a única forma de interação do jogador com a aplicação era mandando uma pequena bola de um canto a outro no famoso jogo da época "Pong". Com a evolução desses jogos, dispositivos revolucionários foram proporcionando ao

jogador novas maneiras de visualizá-los, fazendo com que ele se sinta cada vez mais parte do jogo (KARASINSKI, 2009).

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

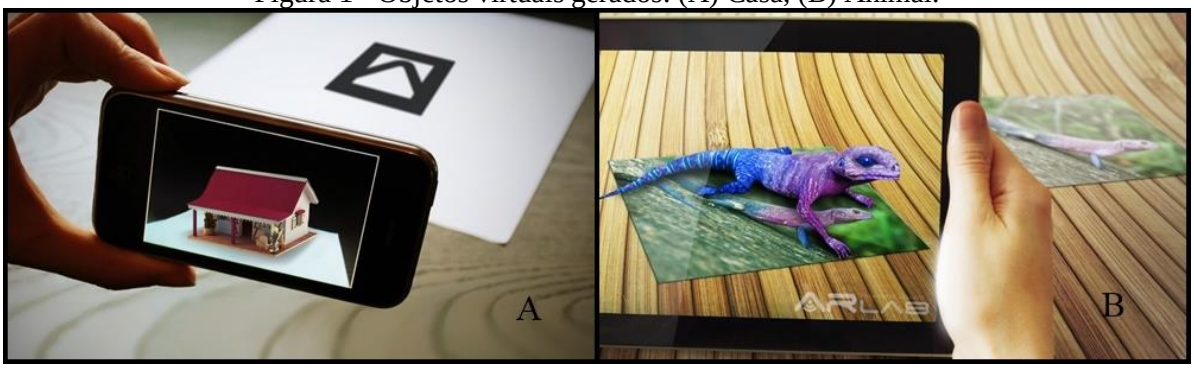
2.1.2 Inserção de informações virtuais no ambiente real

A possibilidade de inserção de informações sobre o que se está olhando é conhecida como ponto de vista, sendo constituída por uma posição, uma orientação e um campo de visão. As mudanças ocorridas no ponto de vista são nomeadas de rastreamento ou *tracking*. Na RA, rastrear um ponto de vista serve como suporte para unir informações virtuais em um ambiente real. Para aplicar informações de um ambiente virtual em um ambiente real, é necessário rastrear o ponto de vista real e transferir estas informações para o virtual. Desta forma, é gerada uma ilusão do virtual estar junto ao ambiente real (FERNANDES, 2012, p. 24-25).

Conforme explicado acima, para a inserção de objetos digitais sobre reais, é necessária a realização de um rastreamento (ou *tracking*) do ambiente real, este podendo ser através da câmera de um dispositivo. A inserção dos objetos digitais sobre reais ou até mesmo a sua atualização de movimento em tempo de execução é realizada por meio do posicionamento, da orientação e do campo de visão de um objeto que pode ser utilizado para estas funções: o marcador (FERNANDES, 2012, p. 24-25).

Fernandes (2012, p. 25-26) apresenta dois tipos de marcadores. O primeiro marcador é do tipo fiducial, definido como símbolos impressos ou desenhos, ambos em preto e branco e geralmente em formas geométricas. Este tipo de marcador transmite informações de dimensão e profundidade, sendo possível identificar o ponto de vista da câmera em relação ao marcador. Esta tecnologia pode ser utilizada tanto para rastreamento como para armazenamento de dados (QRCodes). Na Figura 1 (A) é apresentado um marcador fiducial que foi identificado pela visualização do ambiente real através da câmera de um dispositivo, assim gerando uma casa virtual. O segundo marcador apresentado é do tipo natural, definido como um identificador de objetos físicos e de captação imagens, como fotos e mapas. Conforme Figura 1 (B), é apresentada uma foto (marcador natural) que foi identificada pela visualização do ambiente real através da câmera de um dispositivo, assim gerando um animal virtual. Ressalta-se que os marcadores também são nomeados de alvos.

Figura 1 - Objetos virtuais gerados: (A) Casa, (B) Animal.



Fonte: Dolz (2012).

2.2 UNITY 3D

A empresa Unity Technologies com a perspectiva de popularizar o desenvolvimento de jogos e nivelar está área para desenvolvedores, criou o produto Unity (UNITY, 2015j). Este produto "[...] é uma plataforma de desenvolvimento flexível e eficiente, usado para criar jogos e experiências interativas 3D e 2D em multiplataforma" (UNITY, 2015a).

A plataforma Unity possui um editor flexível e ampliável com vários recursos já criados para a praticidade no desenvolvimento, como animações, áudios, física em 2D e 3D, recursos gráficos, *scripts*, entre outros. Além destes recursos prontos (tendo a possibilidade de adaptá-los ao cenário do jogo), é possível a criação de novos modelos desses recursos. Estas possibilidades tornam o Unity uma plataforma de interesse da indústria de jogos pelas suas características, como a qualidade de suas otimizações, velocidade e eficácia de seus fluxos de trabalho (UNITY, 2015o).

Para o aprendizado na plataforma, é disponibilizada uma ampla documentação no site da Unity, uma grande comunidade de desenvolvedores, treinamentos *on-line* e uma loja *on-line* onde é possível encontrar diversos recursos para *download* (gratuitos e pagos) para a utilização em desenvolvimentos de jogos. São disponibilizados nesta loja recursos como áudios, modelos gráficos, texturas, *scripts*, entre outros recursos, possibilitando aos desenvolvedores, não se preocuparem em criar seus próprios recursos.

Segundo informações estatísticas da empresa Unity (2015i), seu produto possui mais de 45% de participação no mercado de ferramentas de jogos e 47% dos desenvolvedores de jogos o utilizam. O Unity é a ferramenta com maior popularidade em relação a outras desta categoria, tendo como a grande parte de jogos famosos em 3D para dispositivos móveis, desenvolvidos em seu ambiente (UNITY, 2015i)..

2.2.1 Elementos básicos

Um jogo no Unity pode conter vários níveis. Os níveis contêm objetos distintos entre si para a formação de seu cenário. Estes níveis são nomeados de `Scenes`. Cada `Scene` contém uma parte de jogo responsável por alguma funcionalidade, como um menu inicial, os níveis do jogo, seleção de personagens, entre outras opções (UNITY, 2015c).

Para a criação de `Scenes`, são necessários objetos para a sua formação. Neste cenário, destaca-se o elemento mais importante do Unity, nomeado de `Game Object`. Ele pode ser definido como um recipiente vazio que servirá como base para a combinação de funcionalidades e comportamentos, a fim de se criar um objeto final. Estas funcionalidades e comportamentos são chamados de componentes, podendo ser áudios, física, *script*, eventos, dentre outros tipos (UNITY, 2015e). Um importante componente do Unity é o `Event System`. Este componente é um gestor e facilitador em termos de entrada de usuário (toque na tela, arraste de mouse entre outros tipos de entrada). Sua funcionalidade consiste no envio de eventos para objetos com base na entrada do usuário (UNITY, 2015d).

Todo objeto em um cenário de jogo no Unity é um `Game Object`. Câmeras, luzes, Objetos em 3D como árvores, carros, cubos, são definidos como `Game Objects` e possuem componentes distintos para a sua criação (UNITY, 2015e).

A maior parte no desenvolvimento de um jogo está no manuseio de arquivos como texturas, modelos, efeitos sonoros e *scripts* de comportamentos dos `Game Objects`. Estes arquivos já prontos são chamados de `Assets` e desenvolvidos em ferramentas externas, especializada em seu tipo. Os `Assets` após serem criados ou baixados devem ser importados para o Unity através do arrasto do mouse para a janela `Project` (PASSOS et al, 2009, p. 10).

Os `Prefabs` são um tipo de `Asset` que possuem objetos armazenados por completo (com todos seus componentes e propriedades). Modificações feitas neles refletem nos demais `Prefabs` do mesmo tipo já incluídos nas `Scenes`. Eles podem ser adicionados em qualquer `Scene` e são utilizados quando é necessário ter um mesmo objeto várias vezes em um jogo sem precisar realizar novas configurações (UNITY, 2015k).

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

propriedades de componentes ao longo da execução de uma aplicação e tratam entradas do usuário, conforme desejado (UNITY, 2015b).

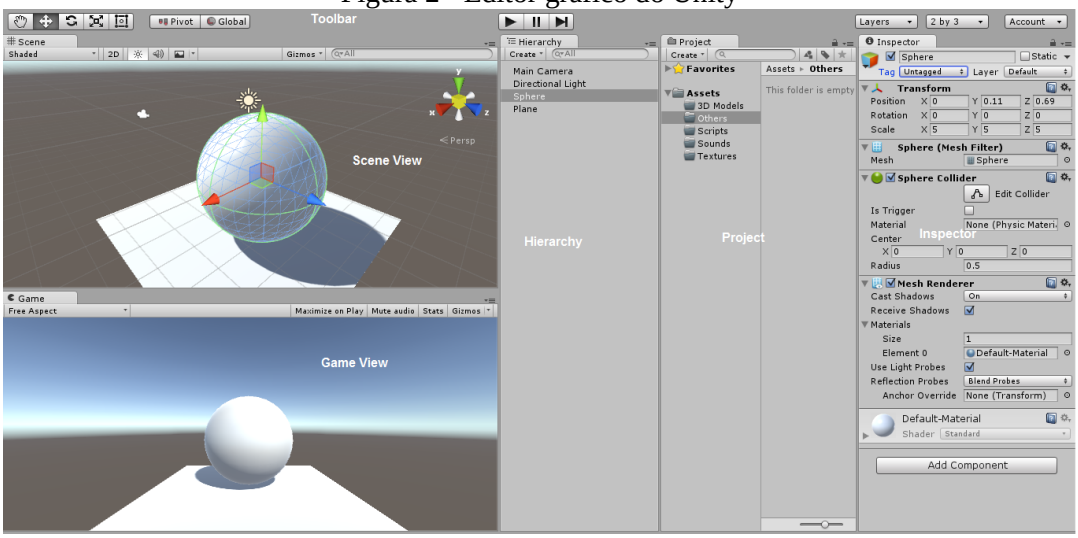
Os *scripts* podem ser criados diretamente no Unity através da janela `Project` e são programáveis nas linguagens C# e JavaScript. O Unity utiliza por padrão a IDE MonoDevelop para a edição dos *scripts* (UNITY, 2015b).

Os *scripts* funcionam internamente através da herança da classe `MonoBehaviour` que por sua vez, define dois métodos que devem ser implementados. O método `Update` que é chamado a cada execução de um *frame* e o método `Start` que é chamado antes da `Scene` ser carregada (UNITY, 2015b). Classes que herdam de `MonoBehaviour` também podem implementar o método `OnGUI` que é chamado a cada execução de um *frame*. Os controles de `GUIs` são estruturas de interface simples e controladas via código (UNITY, 2015g).

2.2.2 Editor gráfico

O Unity possui um editor gráfico para a construção de jogos e sua visualização em tempo real. O editor conta com várias janelas para realizar a construção e organização do jogo. Na Figura 2 é apresentado o editor gráfico do Unity e suas janelas.

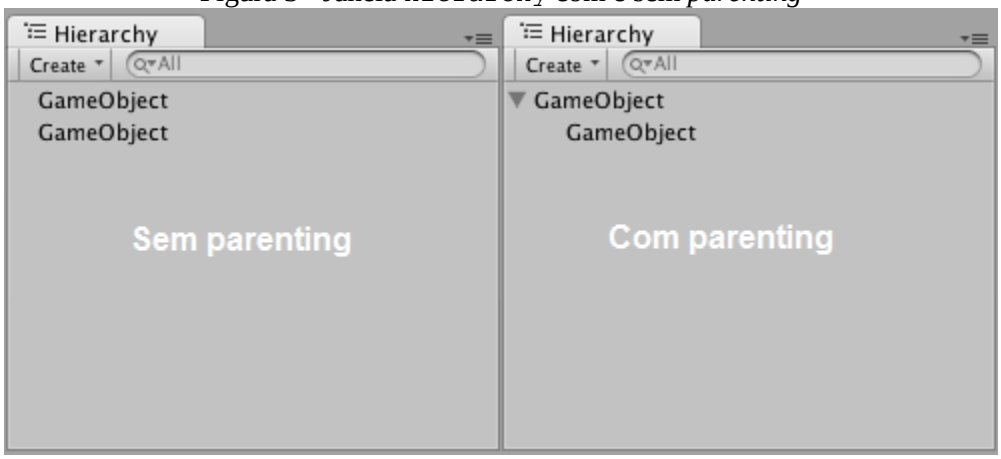
Figura 2 - Editor gráfico do Unity



A `Game View` é uma visualização em tempo real do jogo desenvolvido. Para utilizar esta janela, o editor do Unity conta com uma `Toolbar` com um botão para iniciar a simulação da aplicação e outro para pausá-la, podendo também realizar alterações nos `Game Objects` em tempo de execução. Porém, após finalizar a simulação, os valores originais são mantidos. A visualização da simulação é feita através de uma ou mais câmeras criadas no jogo. A janela também conta com outras opções, como alteração da proporção da tela e desempenho da aplicação (UNITY, 2015f).

A janela `Hierarchy` apresenta todos os `Game Objects` contidos na `Scene` atual. O Unity utiliza o conceito de *parenting*, fazendo com que qualquer `Game Object` possa ser filho de outro, sendo assim, um objeto pai poderá ter um ou mais objetos filhos. Isto pode ser realizado arrastando o objeto filho desejado para o pai na janela `Hierarchy`. Os objetos filhos herdam os movimentos e rotações de seus pais (UNITY, 2015h). Na Figura 3 é apresentada a janela `Hierarchy` com e sem *parenting*.

Figura 3 - Janela `Hierarchy` com e sem *parenting*

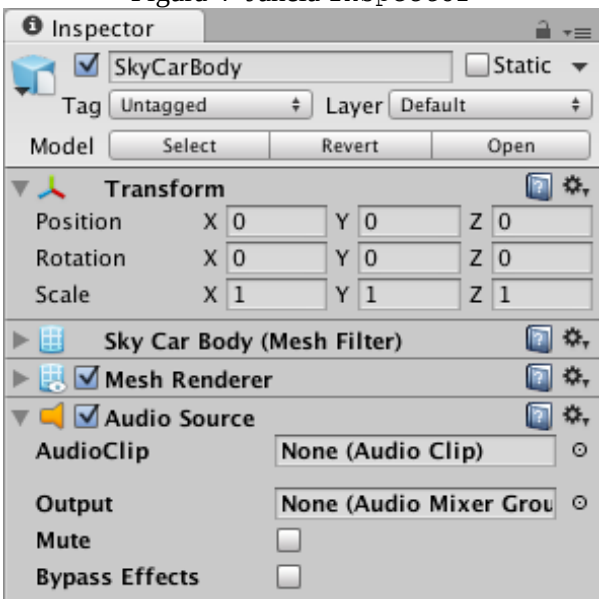


Fonte: adaptado de Unity (2015h).

A janela `Project` é responsável pelo acesso e gerenciamento de todos os `Assets` do projeto (UNITY, 2015l). Esta janela organiza todos os recursos do projeto, facilitando seu manuseio entre as `Scenes`. Uma vez criado ou importado um `Asset`, ele estará contido nesta janela e poderá ser utilizado em todas as `Scenes` do projeto.

O `Inspector` é uma janela no Unity na qual são exibidas informações e componentes do `Game Object` atualmente selecionado. Nesta janela é possível definir e alterar os valores das propriedades dos componentes do `Game Object` selecionado. Quando algum componente possui um *script* vinculado, as propriedades públicas do *script* podem ser modificadas através desta janela. Nota-se que as propriedades dos componentes podem ser alteradas em tempo de execução (UNITY, 2015i). Na Figura 4 é apresentada a janela `Inspector`.

Figura 4- Janela `Inspector`



Fonte: adaptado de Unity (2015i).

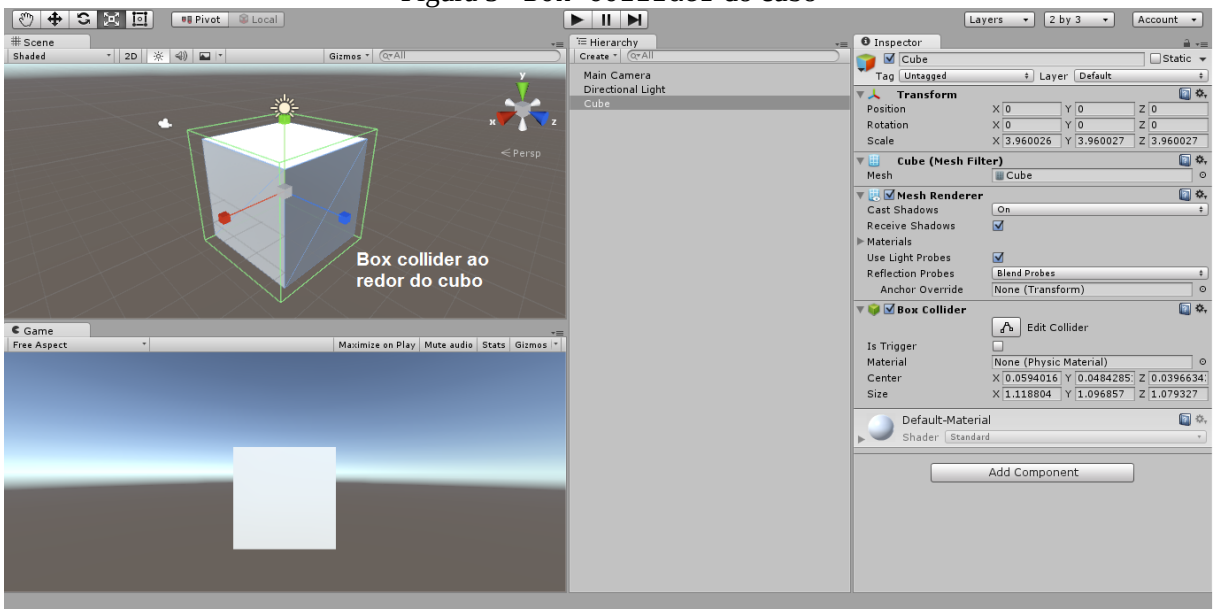
A `Scene View` é uma janela interativa responsável pela manipulação de `Game Objects` contidos na `Scene`. A manipulação desses objetos é realizada através de botões disponibilizados na `Toolbar` que alteram a escala, a rotação e a translação do objeto, sem precisar atribuir valores a estas propriedades diretamente pela janela `Inspector` (UNITY, 2015n).

2.2.3 Controle de física e colisão (interatividade)

Para que um `Game Object` possa agir sob o controle de física, o Unity disponibiliza o componente `Rigidbody`. Este componente é adicionado aos `Game Objects` que devem implementar o controle de física. Com este componente é possível implementar articulações, colisões, gravidade, movimentações, restrições na física entre outras possibilidades (UNITY, 2015m).

Para que ocorra colisões entre objetos é necessário utilizar um componente para o controle da colisão ao lado do `Rigidbody` nos `Game Objects`. O Unity disponibiliza componentes para tratamento de colisões como o `Box Collider`, o `Sphere Collider` e o `Mesh Collider` (UNITY, 2015m). Cada um destes componentes possui suas características distintas, como o `Box Collider` que possui o formato de uma caixa, na qual a colisão é detectada (ver Figura 5). Estes componentes são invisíveis em tempo de execução e adicionados ao redor de objeto, podendo ser configurados os seus tamanhos.

Figura 5 - `Box Collider` do cubo



Quando `Game Objects` com `Rigidbody` se chocam e possuem algum componente de colisão, o motor de física age sobre seus corpos. Caso algum deles não possua um componente de colisão, um objeto atravessa o outro (UNITY, 2015m).

As classes que herdam de `MonoBehaviour` possuem métodos para o tratamento de colisão. O método `OnCollisionEnter` é chamado quando um objeto toca outro objeto. Enquanto um objeto estiver colidindo com outro objeto, o método `OnCollisionStay` é chamado uma vez por *frame*. Quando o objeto sair da colisão, o método `OnCollisionExit` é chamado. Caso não seja necessário envolver física nas colisões, porém seja preciso gerar os eventos, no componente de colisão deve ser marcada a opção `IsTrigger`. Com isto, devem ser utilizados os métodos `OnTriggerEnter`, `OnTriggerStay` e `OnTriggerExit` que funcionam da mesma maneira como os descritos acima. Estes métodos possuem como parâmetro o objeto que foi colidido (CAROTENUTO, 2014).

2.2.4 Vuforia

Embora o Unity seja uma plataforma de desenvolvimento de jogos e aplicativos, ele não possui recursos próprios para o desenvolvimento de aplicações com RA. Para suprir esta necessidade, destaca-se o *Software Development Kit* (SDK) desenvolvido pela empresa Qualcomm: Vuforia. Esta tecnologia tem como objetivo a praticidade no desenvolvimento de aplicações com RA (QUALCOMM, 2015b), disponibilizando vários recursos, como reconhecimento e rastreamento de objetos (QUALCOMM, 2015i).

Vuforia é utilizada por diversas empresas para propor entretenimentos diversificados, possuindo desenvolvedores que estão ampliando os recursos da biblioteca. Esta tecnologia

pode fazer o reconhecimento de diversas coisas em um ambiente real para utilização como marcadores (fiduciais e naturais) para proporcionar a RA. Dentre as principais estão: palavras, cilindros, caixas, reconhecimento de imagens diversificadas (como imagens de revistas, embalagens e propagandas) e objetos físicos, como brinquedos, sapatos entre outros produtos (QUALCOMM, 2015i).

O Unity e o Vuforia podem ser integrados a partir do *plugin* Unity Extension (QUALCOMM, 2015e). Isto permite o desenvolvimento de aplicações de RA no ambiente Unity através da disponibilização de `Prefabs`, *scripts* e outros componentes para o controle da RA na aplicação.

O `Prefab Image Target` é utilizado para o Vuforia detectar e rastrear imagens. Estas imagens são cadastradas através do gerenciador de alvos do Vuforia. Podem ser cadastradas imagens no formato JPG ou PNG em tons de cinza ou no sistema de cores RGB com tamanho máximo de 2 MB. Após o cadastro da imagem, o gerenciador de alvos, processa e avalia características especificas da imagem para determinar um nível de detecção (com base nas características), sendo que quanto maior o nível, melhor é a chance de detecção (QUALCOMM, 2015h). Segundo QUALCOMM (2015a) uma imagem cadastrada na base de alvos deve ser rica em detalhes, deve ter regiões claras e escuras, ser bem iluminada e não possuir padrões repetitivos para permitir uma melhor detecção pelo SDK do Vuforia.

`Frame Marker` é um tipo de marcador fiducial exclusivo do Vuforia. Este marcador possui o formato de um quadrado com um padrão binário ao longo de sua borda o qual é utilizado para identificar o marcador em tempo de execução. Além disto, é possível incluir dentro de suas fronteiras, uma imagem (porém não utilizada para identifica-lo). O Vuforia disponibiliza 512 tipos diferentes de `Frame Markers` para a utilização em um aplicativo, estando disponíveis na pasta da instalação do SDK do Vuforia (QUALCOMM, 2015d).

O `Prefab ARCamera` realiza a *renderização* da imagem da câmera e manipula os objetos para serem rastreados (QUALCOMM, 2015f). Este `Prefab` possui diversas propriedades para configuração no Unity. A propriedade `Load Data Set` carrega uma base de alvos exportada do gerenciador de alvos do Vuforia quando o aplicativo é iniciado. A propriedade `Active` é utilizada em conjunto com a `Load Data Set` para ativar a base após a mesma ser carregada. A propriedade `World Center Mode` é utilizada para definir um objeto que servirá como origem do espaço 3D da cena. Quando `World Center Mode` for definido como `SPECIFIC_TARGET` é habilitada a propriedade `World Center` para a seleção de um objeto para ser a origem do espaço 3D da cena (QUALCOMM, 2015g).

Vuforia possui uma opção para um tipo de rastreamento especial de objetos, nomeada de `Extend Tracking`. Esta opção permite o monitoramento e rastreamento mesmo quando o alvo não está mais em vista. Para isto, Vuforia mapeia o alvo e assume que o alvo e o ambiente em que ele está inserido são em sua maior parte, estáticos. Dentre as opções de marcadores que o Vuforia suporta, somente os `Frame Markers` e os `Word Markers` não suportam o `Extend Tracking` (QUALCOMM, 2015c).

2.3 TRABALHOS CORRELATOS

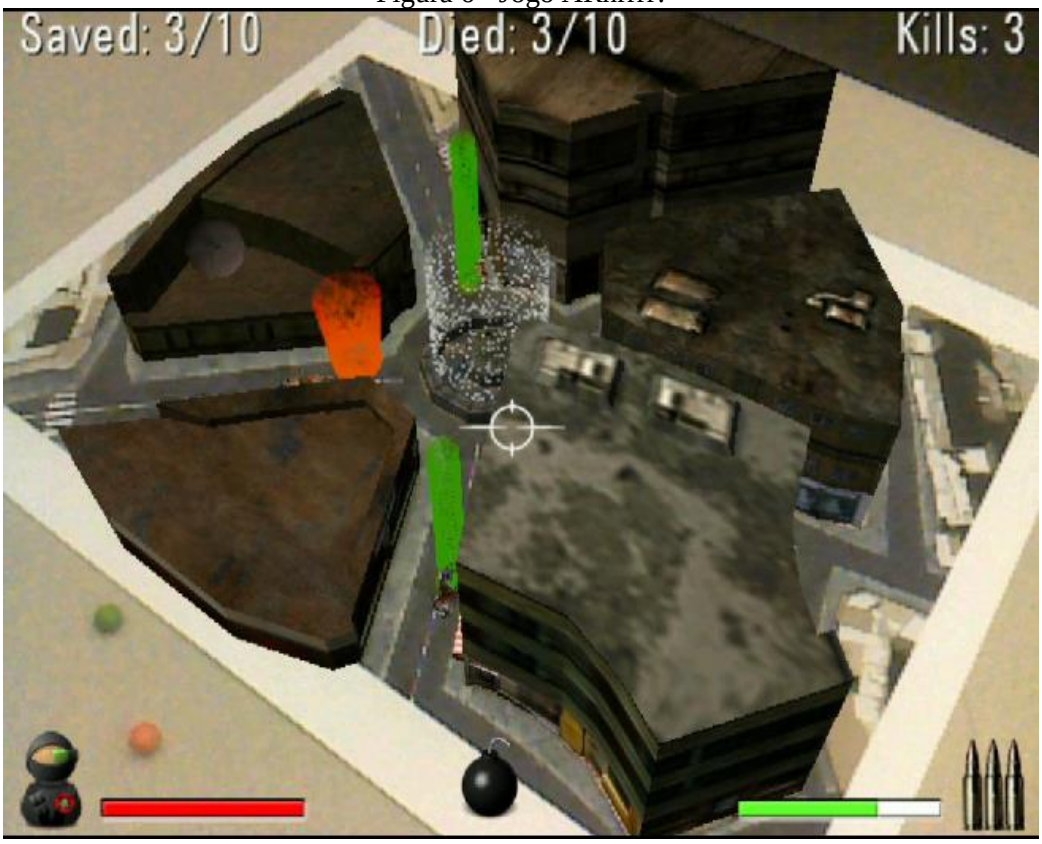
A seguir são apresentados três trabalhos com características semelhantes a este. O item 2.3.1 detalha o jogo acadêmico ARhrrrr!, o item 2.3.2 apresenta o jogo ARDefender, um dos primeiros jogos do gênero disponibilizado para iOS (APPSDOIPHONE, 2012) e o item 2.3.3 apresenta um trabalho acadêmico que por meio da RA facilita a visualização de produtos por pessoas com acuidade visual.

2.3.1 ARhrrrr!

ARhrrrr! é um jogo acadêmico de RA desenvolvido pelas instituições Georgia Tech Augmented Environments Lab e Savannah College of Art and Design, projetado especificamente para aparelhos com o processador móvel Nvidia Tegra. A RA utilizada no jogo o torna envolvente e interativo possibilitando ao jogador se sentir em um ambiente totalmente diferenciado (FORREST, 2009).

No jogo ARhrrrr! é utilizado um mapa especialmente desenvolvido para possibilitar misturar realidade virtual com a real. Através da câmera do aparelho apontada para o mapa, é projetada uma cidade em 3D repleta de zumbis e personagens comuns (sobreviventes ao ataque) que devem ser protegidos do ataque dos zumbis. Para isto, na tela do aparelho é criada uma mira para o jogador poder atirar nos zumbis, podendo assim, salvar os sobreviventes. Nota-se que além dos zumbis atacarem os sobreviventes, existe a possibilidade dos zumbis atacarem o jogador atirando seus órgãos em direção ao aparelho (GEORGIA TECH, 2015). Na Figura 6 são visualizados prédios virtuais gerados a partir da identificação do mapa físico.

Figura 6 - Jogo ARhrrrr!



Fonte: Georgia Tech (2015).

Um dos atrativos do jogo é a possibilidade de utilizar balas Skittles como marcadores para a geração de eventos interativos com o jogo. Para isto, o jogador deve posicionar a bala em cima de um local desejado do mapa do jogo e atirar com a mira em cima da bala. Isto gera uma animação interativa que, dependendo da cor da bala, pode gerar desde portais para salvar os sobreviventes até servir como bomba para eliminação dos zumbis (KARASINSKI, 2009).

2.3.2 ArDefender

ARDefender é um jogo de RA que "[...] usa a câmera do telefone para misturar o mundo do jogo e o mundo real [...]" (GOOGLE, 2013, tradução nossa). Foi desenvolvido pela Int13 e disponibilizado para as plataformas Android e iOS. Este jogo é classificado com o gênero de *tower defense*. Para jogá-lo, é necessário imprimir através do site da desenvolvedora, um marcador que é utilizado pelo aplicativo do jogo para gerar uma torre virtual, sobreposta ao marcador.

O objetivo do jogo ARDefender é a defesa de uma torre gerada a partir da RA de ataques de criaturas virtuais geradas pelo próprio jogo e visualizadas na tela do aparelho. Para a defesa da torre, na tela do jogo são disponibilizados recursos, como armamentos para a

eliminação das criaturas virtuais (APPSDOIPHONE, 2012). Conforme visualizado na Figura 7, a torre virtual foi gerada no lado inferior central da tela do *smartphone*.

Figura 7 - Jogo ARDefender

Fonte: Google Play (2013).

Este jogo foi um dos primeiros criados para a plataforma iOS que empregou a utilização da RA (APPSDOIPHONE, 2012). Os desenvolvedores do ARDefender tiveram como fator motivacional no desenvolvimento do jogo, a aplicação da RA para criar um jogo inovador para dispositivos móveis (APPLE, 2015).

2.3.3 eyeBuy

O eyeBuy é um trabalho acadêmico desenvolvido para o auxilio de pessoas com problemas de acuidade visual. Ele provê autonomia no processo de seleção e aquisição de informações de produtos encontrados em ambientes como supermercados. A RA foi aplicada como um meio de facilitar o desenvolvimento de interfaces acessíveis para a aplicação (FORTE; SILVA; MARENGONI, 2012).

Segundo os desenvolvedores (FORTE; SILVA; MARENGONI, 2012), o aplicativo foi desenvolvido especificamente para o sistema operacional Android, se caracterizando como uma aplicação para *tablets*. O objetivo deste aplicativo é a utilização da câmera do *tablet* para identificar produtos previamente cadastrados no banco de dados da aplicação e por meio da RA, apresentar uma tabela nutricional (virtual) ao lado do produto em tamanho maior (Figura

8). Isto tem como beneficio a facilidade da visualização da tabela nutricional de produtos, devido a mesma possuir um pequeno tamanho fisicamente.

No desenvolvimento do aplicativo, foi utilizada a plataforma Unity 3D para a modelagem de objetos virtuais (a tabela nutricional apresentada ao lado do produto quando o mesmo for identificado) e o SDK Vuforia para a identificação e realização do posicionamento dos objetos virtuais aos produtos. Nota-se que essas tecnologias também são utilizadas para o desenvolvimento deste trabalho.

Figura 8 – Aplicativo eyeBuy

Fonte: Forte, Silva e Marengoni (2012).

3 DESENVOLVIMENTO

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

3.1 REQUISITOS

O aplicativo deverá atender aos seguintes requisitos:

a) possuir uma tela inicial para seleção de opções (Requisito Funcional - RF);
b) possuir um ambiente de simulação usando um cenário simples de trânsito (RF);
c) utilizar a imagem de um mapa como marcador para a geração de um mundo virtual 3D (RF);
d) disponibilizar uma função para visualizar a imagem da câmera (RF);
e) disponibilizar uma função para poder visualizar e imprimir o marcador (RF);
f) disponibilizar uma função para visualizar instruções básicas sobre o trânsito (RF);
g) movimentar um carro disponibilizado no mundo virtual 3D através da interatividade com um *joystick* virtual (RF);
h) simular a utilização de um carro real de brinquedo para interagir com o mundo virtual 3D (RF);
i) simular a utilização de um marcador de `Frame` para interagir com o mundo virtual 3D (RF);
j) possuir um retorno visual da movimentação do carro no mundo virtual 3D através da atualização de uma barra de gasolina (RF);
k) possuir um retorno visual das colisões do carro no mundo virtual 3D através da atualização de uma barra de mecânica e de gasolina (RF);
l) gerar botões para voltar à tela inicial ou recomeçar a partida caso a barra de mecânica ou a barra de gasolina se esgote (RF);
m) utilizar modelos em 3D já desenvolvidos (Requisito Não Funcional – RNF);
n) utilizar imagens prontas e posteriormente formatá-las no programa Gimp 2 para criar ícones e texturas (RNF);
o) ser disponibilizado para a plataforma Android (RNF);
p) utilizar o SDK Vuforia para a implementação dos recursos de RA (RNF);
q) utilizar a linguagem C# para definir os *scripts* do aplicativo (RNF);

r) utilizar o motor de jogos Unity 3D para a implementação do espaço 3D (RNF).

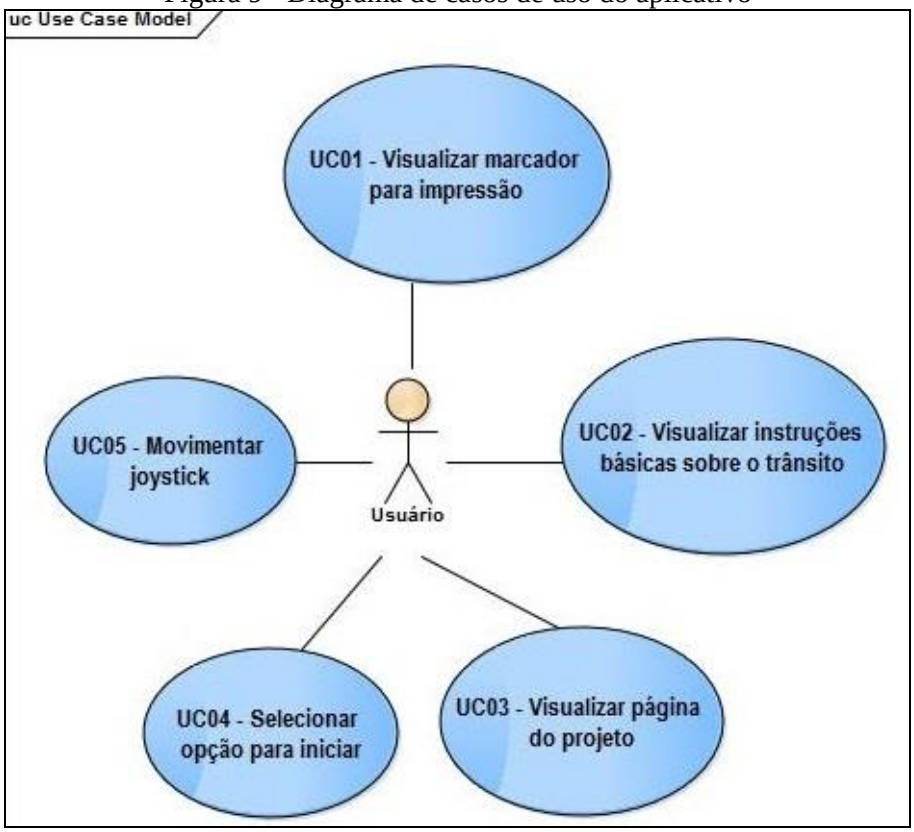
3.2 ESPECIFICAÇÃO

A especificação do aplicativo foi desenvolvida seguindo a análise orientada a objetos, utilizando a *Unified Modeling Language* (UML) em conjunto com a ferramenta Enterprise Architect na versão 11.0.1106 para a elaboração dos casos de uso, do diagrama de classes e dos diagramas de atividades.

3.2.1 Diagrama de Casos de uso do aplicativo

Nesta seção são apresentados os casos de uso do aplicativo, ilustrados na Figura 9. Identificou-se o ator `Usuário` que utiliza o aplicativo. Do Quadro 1 até o Quadro 5 são descritos os casos de uso do ator `Usuário`.

Figura 9 - Diagrama de casos de uso do aplicativo



Quadro 1 - Caso de uso `UC01 – Visualizar marcador para impressão`

Descrição	Este caso de uso tem por objetivo disponibilizar o marcador para visualização do mundo 3D para impressão.
Pré-Condição	Estar com o aplicativo aberto, possuir um navegador no dispositivo, conexão com a internet e um local para impressão.
Cenário Principal	1. O `Usuário` irá tocar no botão `Imprimir Marcador`; 2. O aplicativo irá abrir o navegador padrão do dispositivo do `Usuário` em uma página contendo o marcador.
Pós-Condição	O `Usuário` poderá imprimir o marcador.

Quadro 2 - Caso de uso UC02 – Visualizar instruções básicas sobre o trânsito

Descrição	Este caso de uso tem por objetivo disponibilizar conteúdo com informações educativas sobre trânsito.
Pré-Condição	Estar com o aplicativo aberto, possuir um navegador no dispositivo e conexão com a internet.
Cenário Principal	1. O Usuário irá tocar no botão Educação no trânsito; 2. O aplicativo irá abrir o navegador padrão do dispositivo do Usuário em uma página com instruções básicas para orientar sobre o trânsito.

Quadro 3 – Caso de uso UC03 – Visualizar página do projeto

Descrição	Este caso de uso tem por objetivo disponibilizar uma página contendo os arquivos do projeto.
Pré-Condição	Estar com o aplicativo aberto, possuir um navegador no dispositivo e conexão com a internet.
Cenário Principal	1. O Usuário irá tocar no *link* http://www.inf.furb.br/gcg/tecedu/transito; 2. O aplicativo irá abrir o navegador padrão do dispositivo do Usuário no *link* correspondente.
Pós-Condição	O Usuário poderá baixar os arquivos do projeto (aplicativo, marcador e as instruções básicas sobre o trânsito).

Quadro 4 - Caso de uso UC04 – Selecionar opção para iniciar

Descrição	Este caso de uso tem por objetivo carregar um cenário para visualização da imagem da câmera e a geração do mundo virtual 3D.
Pré-Condição	Estar com o aplicativo aberto.
Cenário Principal	1. O Usuário irá tocar no botão Iniciar; 2. É apresentada a imagem da câmera na tela do dispositivo; 3. O Usuário deve visualizar o marcador através da imagem da câmera do dispositivo; 4. O aplicativo reconhece o marcador.
Exceção 1	Caso o marcador não seja visualizado pela câmera do dispositivo, o mundo virtual 3D não é gerado.
Pós-Condição	É gerado um mundo virtual 3D sobreposto ao marcador

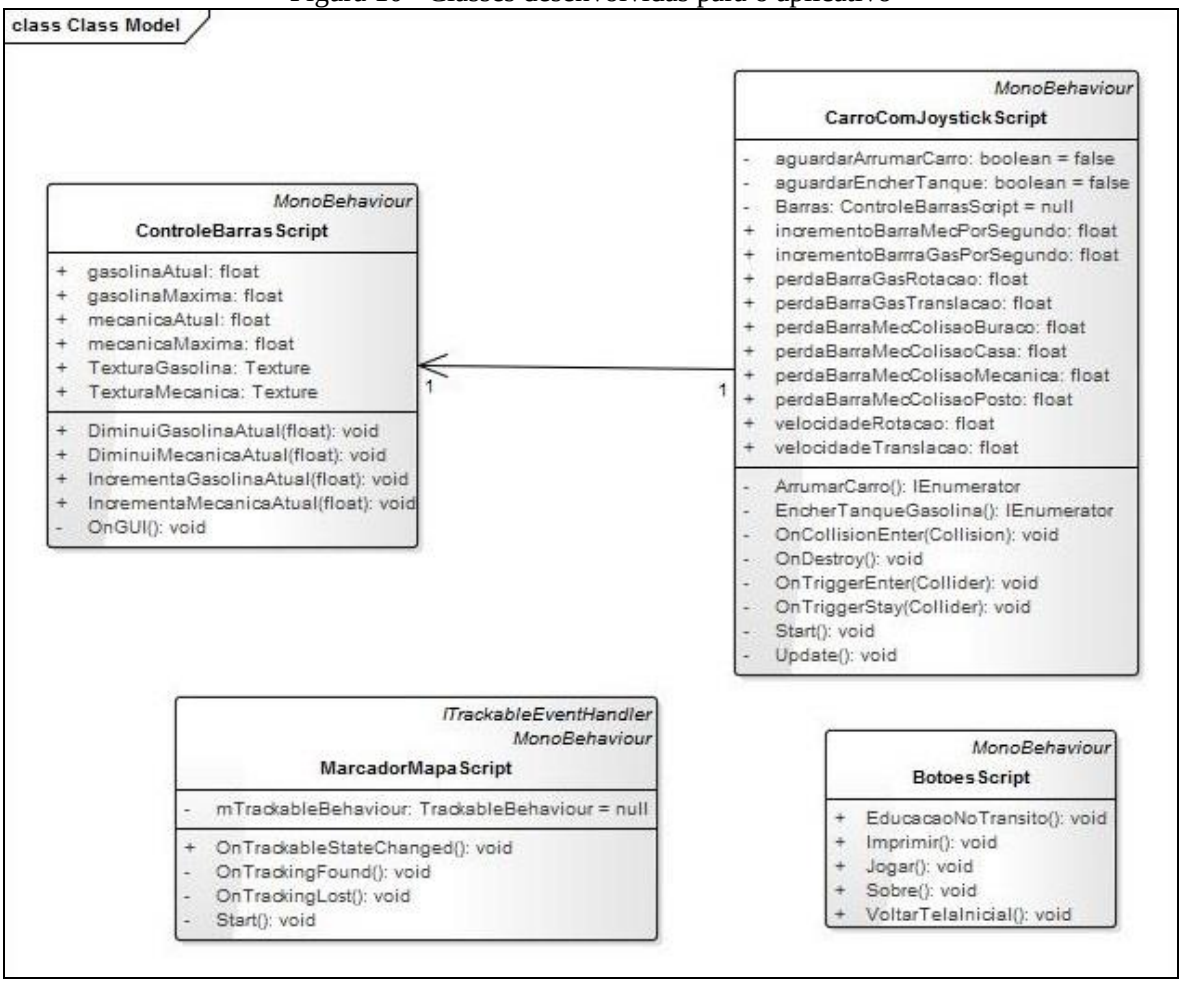
Quadro 5 - Caso de uso UC05 – Movimentar *joystick*

Descrição	Este caso de uso tem por objetivo movimentar um *joystick* virtual.
Pré-Condição	Estar com o aplicativo aberto e executar o UC04.
Cenário Principal	1. O Usuário movimentará o *joystick* virtual.
Fluxo Alternativo 1	A qualquer momento o Usuário poderá retornar para a tela inicial, a fim de selecionar novamente os casos de uso UC01, UC02, UC03 e UC04 através do botão Voltar.
Pós-Condição	Através da movimentação aplicada no *joystick*, um carro gerado no mundo virtual é movimentado e uma barra representando sua gasolina diminui. Caso o carro colida com algum objeto do mundo virtual, a barra de gasolina pode aumentar, uma barra representando sua mecânica pode diminuir ou aumentar. Estas modificações nas barras dependem do objeto em que o carro colidir.

3.2.2 Diagrama de classes do aplicativo

O diagrama de classes da Figura 10 apresenta as classes desenvolvidas para a criação do aplicativo.

Figura 10 - Classes desenvolvidas para o aplicativo



A classes `BotoesScript` disponibiliza os métodos executados ao tocar nos botões do aplicativo. Cada um dos botões do aplicativo possui um evento de toque associado a um método desta classe.

A classe `MarcadorMapaScript` é uma cópia da classe `DefaultTrackableEventHandler` (associada aos marcadores do Vuforia) a qual possui métodos para a habilitar e desabilitar os modelos em 3D associados a um marcador, quando o mesmo for reconhecido, perdido ou estiver em `Extend Tracking`. Foram incrementados os métodos desta classe para habilitar e desabilitar o *joystick* virtual do aplicativo quando o marcador for reconhecido.

A classe `ControleBarrasScript` é responsável pela criação e manutenção das barras de mecânica e gasolina que são visualizadas ao iniciar a imagem da câmera. As barras de mecânica e gasolina são criadas através do método `OnGUI` conforme as variáveis atribuídas no

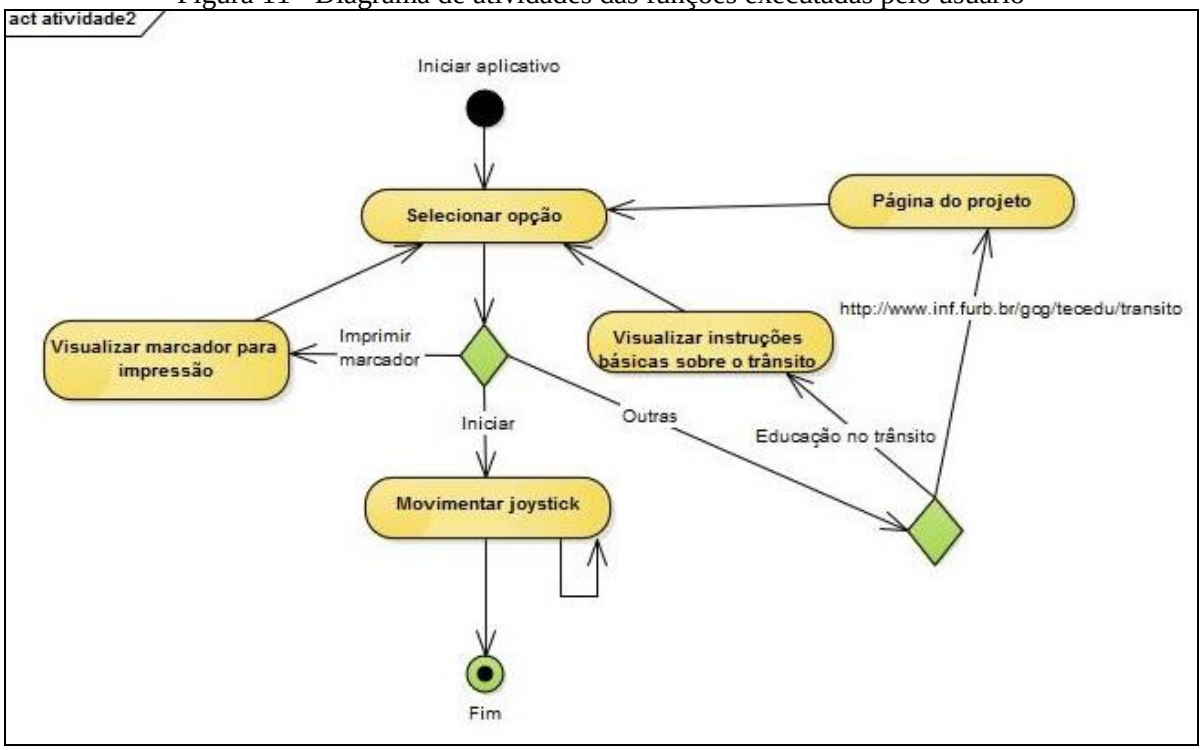
Unity para esta classe. Esta classe disponibiliza os métodos `DiminuirGasolinaAtual`, `DiminuirMecanicaAtual`, `IncrementaGasolinaAtual` e `IncrementaMecanicaAtual` para o controle do tamanho das barras.

A classe `CarroComJoystickScript` é responsável pela movimentação de um carro virtual em 3D e o tratamento de suas colisões com outros objetos. Esta classe possui uma instância da classe `ControleBarrasScript` para o controle das barra de gasolina e mecânica nas ações executadas no carro virtual 3D. As variáveis atribuídas a esta classe pelo Unity controlam a quantidade de perda ou incremento das barras de mecânica e gasolina e também da velocidade do carro.

3.2.3 Diagramas de atividades

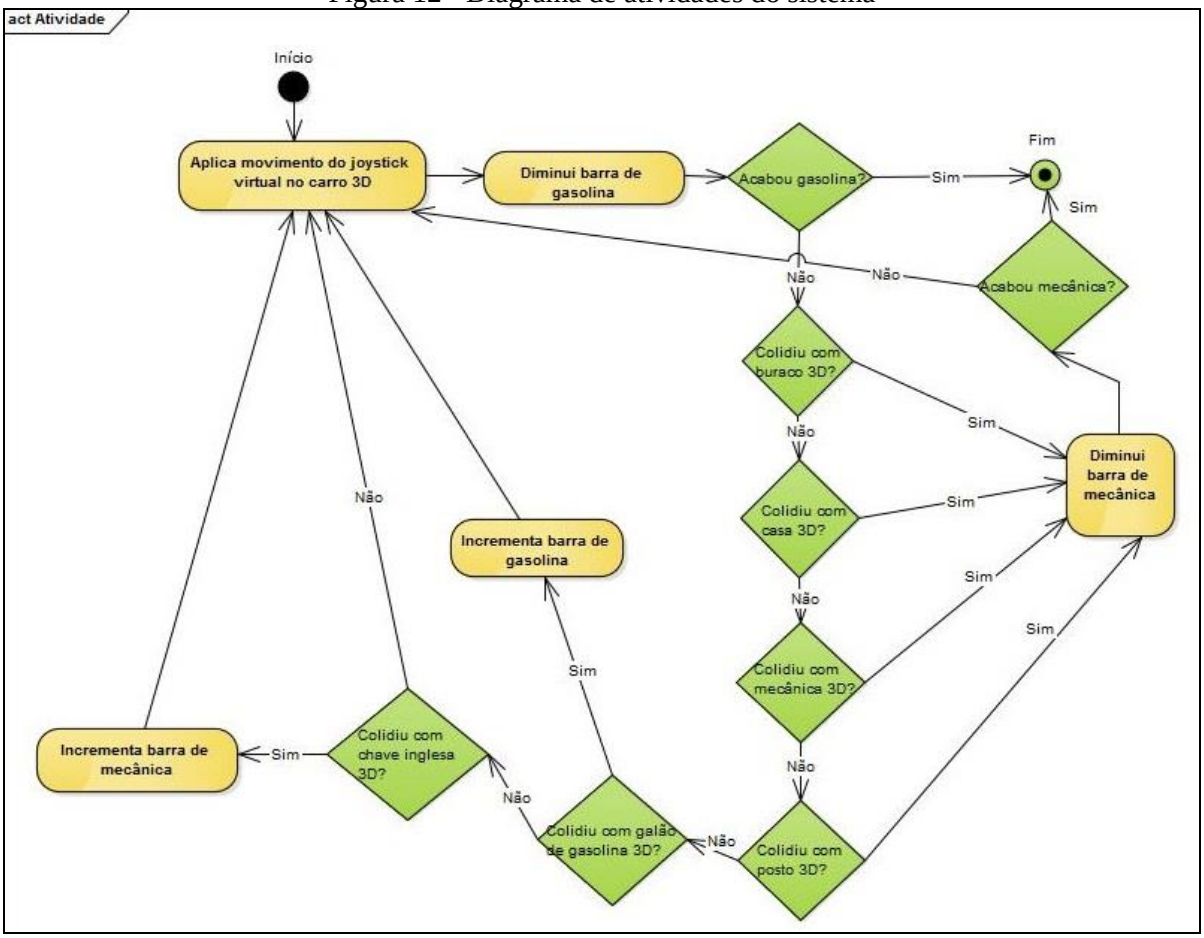
A Figura 11 apresenta o diagrama de atividades das funções executadas pelo usuário no aplicativo. Suas atividades incluem selecionar uma de três opções disponíveis na tela inicial e controlar um *joystick* virtual para interagir com o mundo virtual 3D do aplicativo.

Figura 11 - Diagrama de atividades das funções executadas pelo usuário



A Figura 12 apresenta o ciclo de funções do sistema, o qual é responsável por controlar a movimentação do carro virtual 3D, tratar suas colisões e dar um retorno visual destes acontecimentos no aplicativo através da barra de mecânica e de gasolina.

Figura 12 - Diagrama de atividades do sistema



O sistema inicia suas atividades aplicando o movimento recebido do araste do *joystick* virtual (realizado pelo usuário na tela do dispositivo) no carro virtual 3D. Em seguida, o sistema diminui a barra de gasolina e verifica se ela se esgotou. Caso positivo, suas atividades se encerram. Caso negativo, é verificado se o carro virtual colidiu com alguma casa, algum buraco, a mecânica, o posto de gasolina, o galão de gasolina ou a chave inglesa. Se não houve nenhuma colisão, o sistema volta para o inicio de suas atividades. Se houve colisão com a chave inglesa, o sistema aumenta a barra de mecânica e volta para o início de suas atividades. Se houve colisão com o galão de gasolina, o sistema aumenta a barra de gasolina e volta para o início de suas atividades. Se houve colisão com algum dos demais objetos o sistema diminui a barra de mecânica e verifica se ela se esgotou, sendo que se o resultado for positivo, as atividades se encerram. Caso contrário, o sistema volta para o início de suas atividades.

3.3 IMPLEMENTAÇÃO

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

3.3.1 Técnicas e ferramentas utilizadas

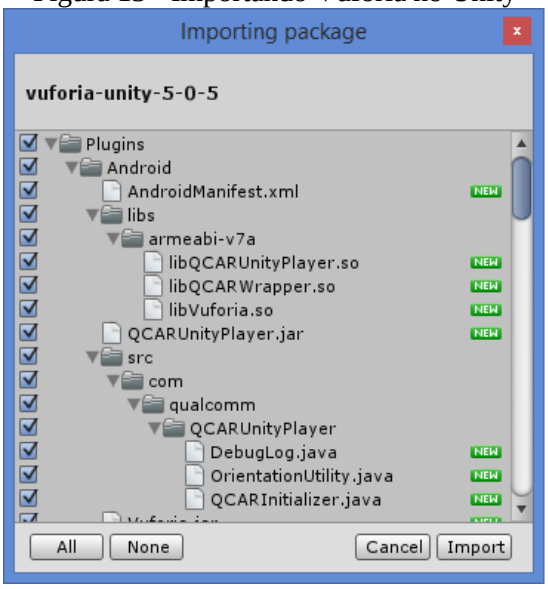
O aplicativo foi desenvolvido no *framework* Unity 3D na versão 5.1.1 f1 *Personal* em conjunto com a IDE MonoDevelop 4.0.1, utilizando a linguagem de programação C Sharp. Para a criação da RA foi utilizado o SDK Vuforia na versão 5.0.5 integrado ao Unity. Para a criação do mapa (marcador) foi utilizado o programa Power Point na versão 2010. Foi utilizado também o programa Gimp 2 para ajustar a transparência do fundo de imagens baixadas da internet para utilização como ícones na tela inicial (*Sprites*). O Gimp 2 também foi utilizado para a criação das imagens das barras de gasolina e mecânica do carro (texturas).

Para a execução e a realização de testes de todo o desenvolvimento do trabalho, utilizou-se um *tablet* Samsung Galaxy tab E com sistema operacional Android 4.4, 1,5 GB de memória RAM e processador *Quad-Core* de 1.3 GHz.

3.3.1.1 Inicio do projeto

Para dar inicio ao projeto no Unity, é necessário a importação do SDK do Vuforia para o Unity, disponibilizado na página de desenvolvedores do Vuforia. A importação pode ser realizada clicando duas vezes no SDK do Vuforia. Com isto, é aberta uma janela apresentado todos os arquivos para serem importados (Figura 13).

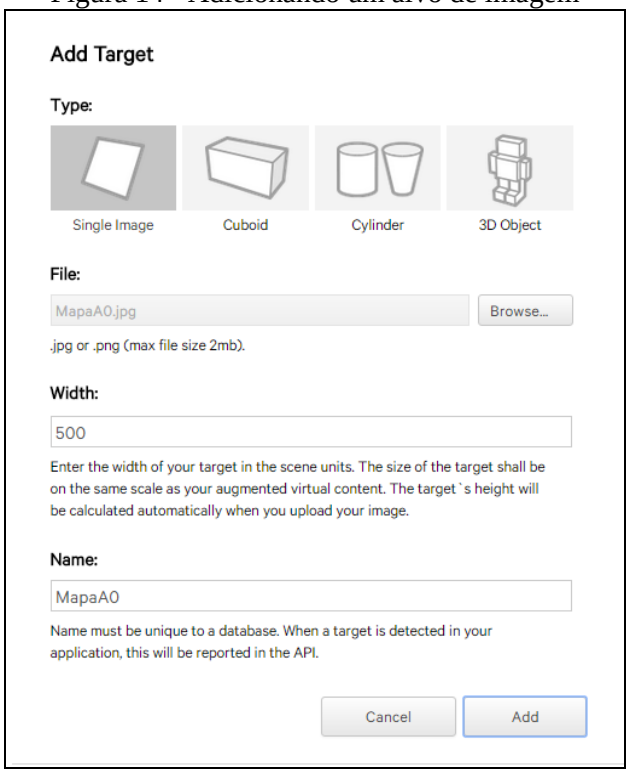
Figura 13 - Importando Vuforia no Unity



Após a importação do Vuforia no Unity, é preciso gerar uma chave de licença para seu uso através na página de desenvolvedores do Vuforia (está chave é adicionada no `Prefab ARCamera`). Ainda na página de desenvolvedores, é necessário o cadastro de uma base de alvos (imagens, cubos, cilindros e objetos em 3D) através do gerenciador de alvos, contendo a imagem do mapa físico. Para o cadastro da base, é necessário nomeá-la para a sua

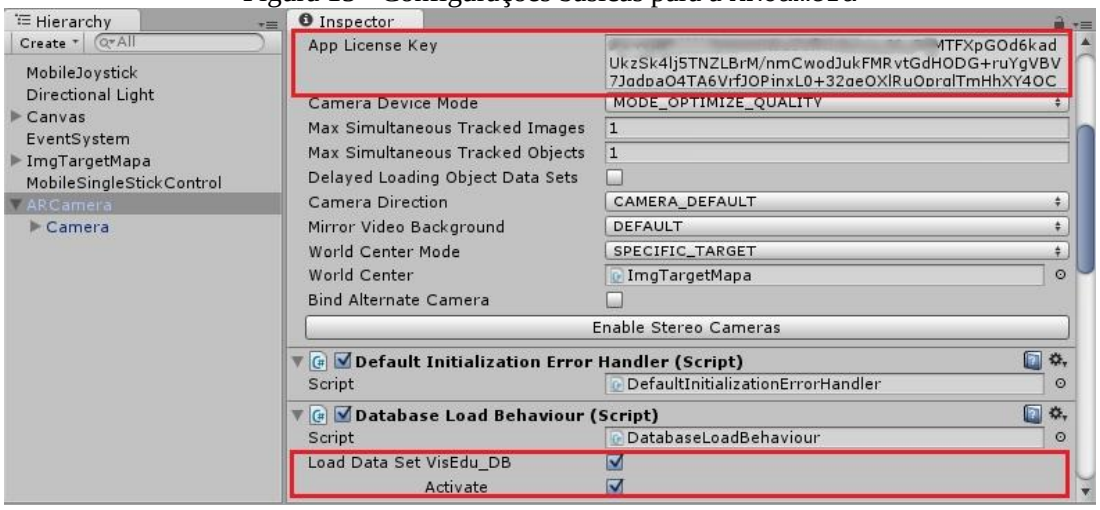
identificação (para o aplicativo criado, a base foi nomeada de VisEdu_DB) Após isto, é necessário adicionar um alvo do tipo *Single Image* para a base (Figura 14). Após o cadastro de um alvo, a página irá disponibilizar uma classificação para o alvo com base em características especificas da imagem. Quanto maior a classificação, melhor será a chance de detecção das características da imagem na aplicação. Após o cadastro, é necessário exportar a base e importá-la no Unity.

Figura 14 - Adicionando um alvo de imagem



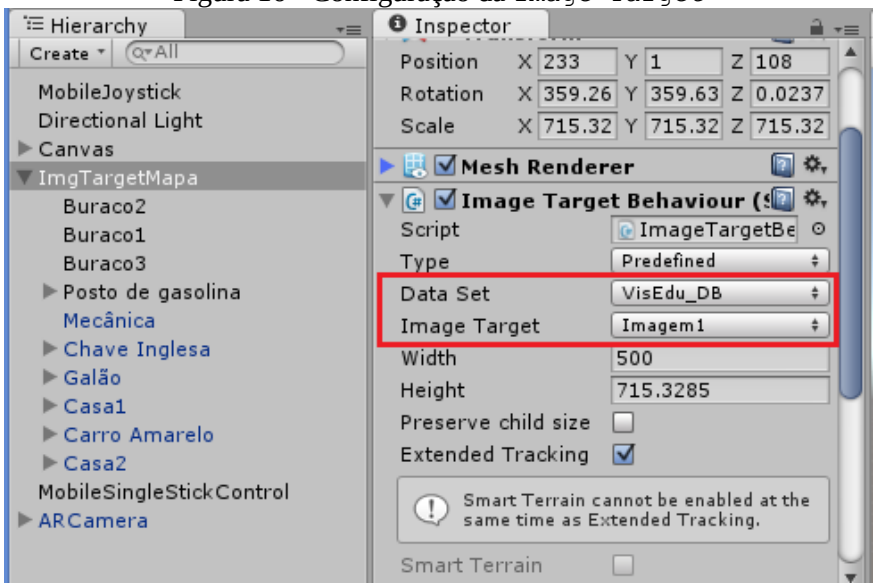
Na janela `Hierarchy` do Unity é preciso deletar a câmera (nomeada de `Main Camera`) que vem por padrão em uma `Scene` e adicionar o `Prefab ARCamera` (localizado dentro da pasta Vuforia na janela `Project`), pois o Vuforia necessita de um tipo especial de câmera para gerar a RA. Na janela `Inspector` da `ARCamera` é necessário incluir a chave de licença obtida, carregar e ativar da base exportada do gerenciador de alvos do Vuforia nas propriedades do *script* `Vuforia Behavior`. A Figura 15 apresenta nos quadros em vermelho, as configurações básicas necessárias para realizar na `ARCamera`.

Figura 15 - Configurações básicas para a `ARCamera`



No aplicativo foi utilizado o `Prefab Image Target` para a geração dos objetos em 3D. Com isto, deve ser adicionado o `Prefab Image Target` na janela `Hierarchy`. Na janela `Inspector` deste `Prefab`, deve ser configurado nas propriedades do *script* `Image Target Behaviour` a base de alvos do Vuforia e qual imagem da base deve ser reconhecida (quadro vermelho da Figura 16). Após adicionar o `Prefab` e configurá-lo, deve-se incluir os modelos em 3D como filhos (*parenting*) do `Image Target`. Com isto, quando o marcador for reconhecido através da câmera do dispositivo, os objetos em 3D serão gerados.

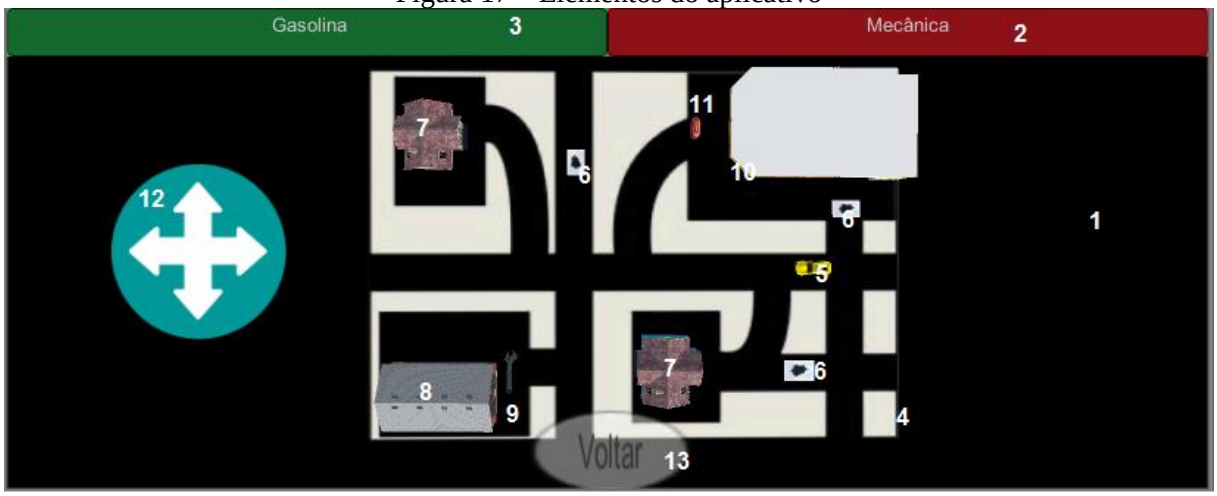
Figura 16 - Configuração da `Image Target`



3.3.1.2 Aplicativo

A `Scene` para a utilização da RA do aplicativo foi construída conforme os itens apresentados na Figura 17 abaixo.

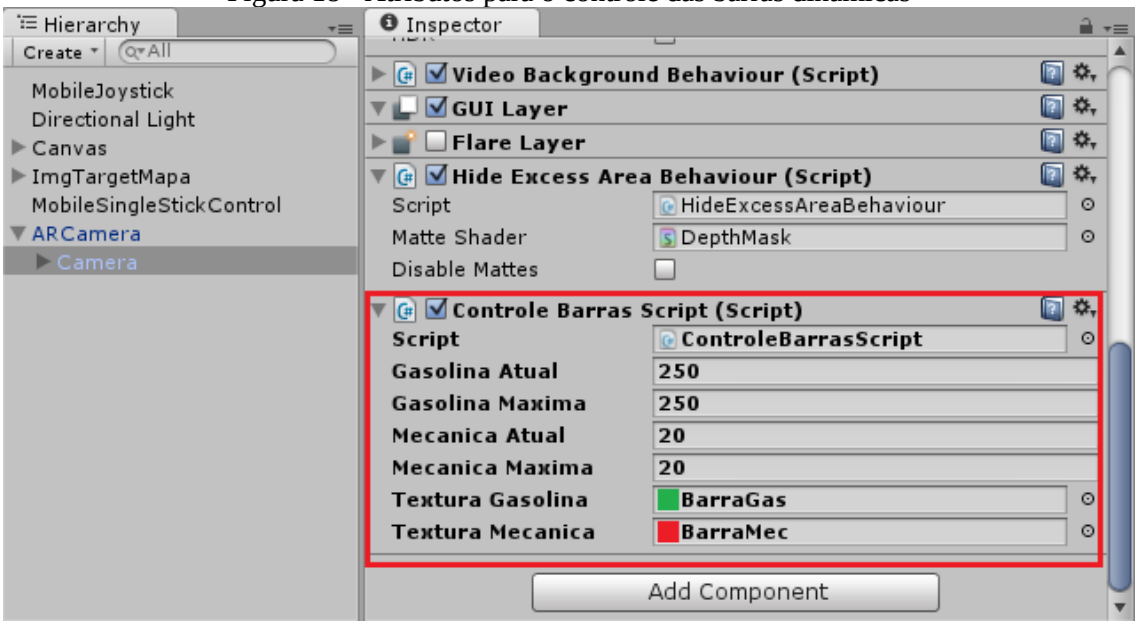
Figura 17 – Elementos do aplicativo



O `item 1` da Figura 17 representa a visualização obtida pela `ARCamera`. Para a utilização da `ARCamera`, foi cadastrada a imagem do mapa no gerenciador de alvos do Vuforia. Através do gerenciador, foi exportada uma base de alvos contendo a imagem do mapa e importada no Unity para ser utilizada pela `ARCamera`. Também foi configurada a chave de licença utilizada, definido o `World Center Mode` como `SPECIFIC_TARGET` e definido o `World Center` com a `Image Target` do mapa nas propriedades da `ARCamera`.

A `ARCamera` possui um `Game Object` do tipo `Camera` como sua filha. Neste `Game Object` foi adicionado o *script* `ControleBarrasScript` para a criação das telas dinâmicas do aplicativo (`item 2` e `item 3`). Neste *script* são informados os atributos `Gasolina Atual`, `Gasolina Maxima`, `Mecanica Atual`, `Mecanica Maxima`, `Textura Gasolina` e `Textura Mecanica` (Figura 18). Os valores e as texturas são utilizadas para o controle das barras de gasolina (`item 3`) e mecânica (`item 2`).

Figura 18 - Atributos para o controle das barras dinâmicas



O Quadro 6 apresenta o método `OnGUI` da classe `ControleBarrasScript` que é chamado a cada execução de *frame* e é responsável por criar a barra de gasolina e a barra de mecânica. Essas barras são criadas com altura e largura para atender a tela do dispositivo. Quando a variável `gasolinaAtual` ou `mecanicaAtual` forem menor que zero, são desenhados na tela dois botões. O primeiro botão carrega novamente a `Scene` do aplicativo através do método `LoadLevel`. O segundo botão carrega a tela inicial através do método `LoadLevel`.

Quadro 6 - Criação de componentes de interface dinâmicos

```
void OnGUI ()
{
      // Desenha barra de gasolina
      GUI.BeginGroup (new Rect (0, 0, Screen.width / 2.0f, Screen.height /
10.0f));
      GUI.DrawTexture (new Rect (0, 0, (Screen.width / 2.0f) *
gasolinaAtual / gasolinaMaxima, Screen.height / 10.0f), TexturaGasolina);
      GUI.Box (new Rect (0, 0, Screen.width / 2.0f, Screen.height /
10.0f), string.Format ("Gasolina"));
      GUI.EndGroup ();

      // Desenha barra de mecanica
      GUI.BeginGroup (new Rect (Screen.width / 2.0f, 0, Screen.width /
2.0f, Screen.height / 10.0f));
      GUI.DrawTexture (new Rect (0, 0, (Screen.width / 2.0f) *
mecanicaAtual / mecanicaMaxima, Screen.height / 10.0f), TexturaMecanica);
      GUI.Box (new Rect (0, 0, Screen.width / 2.0f, Screen.height /
10.0f), string.Format ("Mecânica"));
      GUI.EndGroup ();

      // Quando acabar a gasolina ou a mecanica (Carro destruido)
      if (this.gasolinaAtual <= 0 || this.mecanicaAtual <= 0) {
            Destroy (GameObject.FindGameObjectWithTag ("BotaoVoltar"));

            float width = Screen.width / 4;
            float heigh = Screen.height / 2;
            float posX = Screen.width / 4 - width / 2;
            float posY = Screen.height / 2 - heigh / 2;

            if (GUI.Button (new Rect (posX, posY, width, heigh),
"Recomeçar")) {
                  Application.LoadLevel ("ControlePorJoystick");
            }

            if (GUI.Button (new Rect (posX * 5, posY, width, heigh),
"Tela inicial")) {
                  Application.LoadLevel ("TelaInicial");
            }
      }
}
```

Para o controle da barra de gasolina, são disponibilizados os métodos `DiminuiGasolinaAtual` e `IncrementaGasolinaAtual` que devem receber como parâmetro a quantidade de gasolina que é decrementada ou incrementada da variável `gasolinaAtual`.

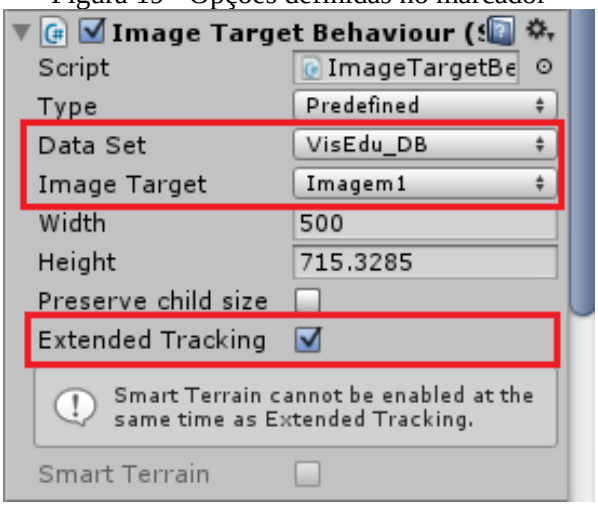
Para o controle da barra de mecânica são disponibilizados os métodos `DiminuiMecanicaAtual` e `IncrementaMecanicaAtual` que devem receber como parâmetro a quantidade de mecânica que é decrementada ou incrementada da variável `mecanicaAtual`, conforme Quadro 7.

Quadro 7 - Métodos para o controle das barras de gasolina e mecânica

```
public void DiminuiGasolinaAtual (float quantidade)
{
	if (this.gasolinaAtual > 0) {
		this.gasolinaAtual = this.gasolinaAtual - quantidade;
	}
}
public void IncrementaGasolinaAtual (float quantidade)
{
	if (gasolinaAtual < gasolinaMaxima) {
		this.gasolinaAtual = this.gasolinaAtual + quantidade;
	}
}
public void DiminuiMecanicaAtual (float quantidade)
{
	if (this.mecanicaAtual > 0) {
		this.mecanicaAtual = this.mecanicaAtual - quantidade;
	}
}
public void IncrementaMecanicaAtual (float quantidade)
{
	if (this.mecanicaAtual < this.mecanicaMaxima) {
		this.mecanicaAtual = this.mecanicaAtual + quantidade;
	}
}
```

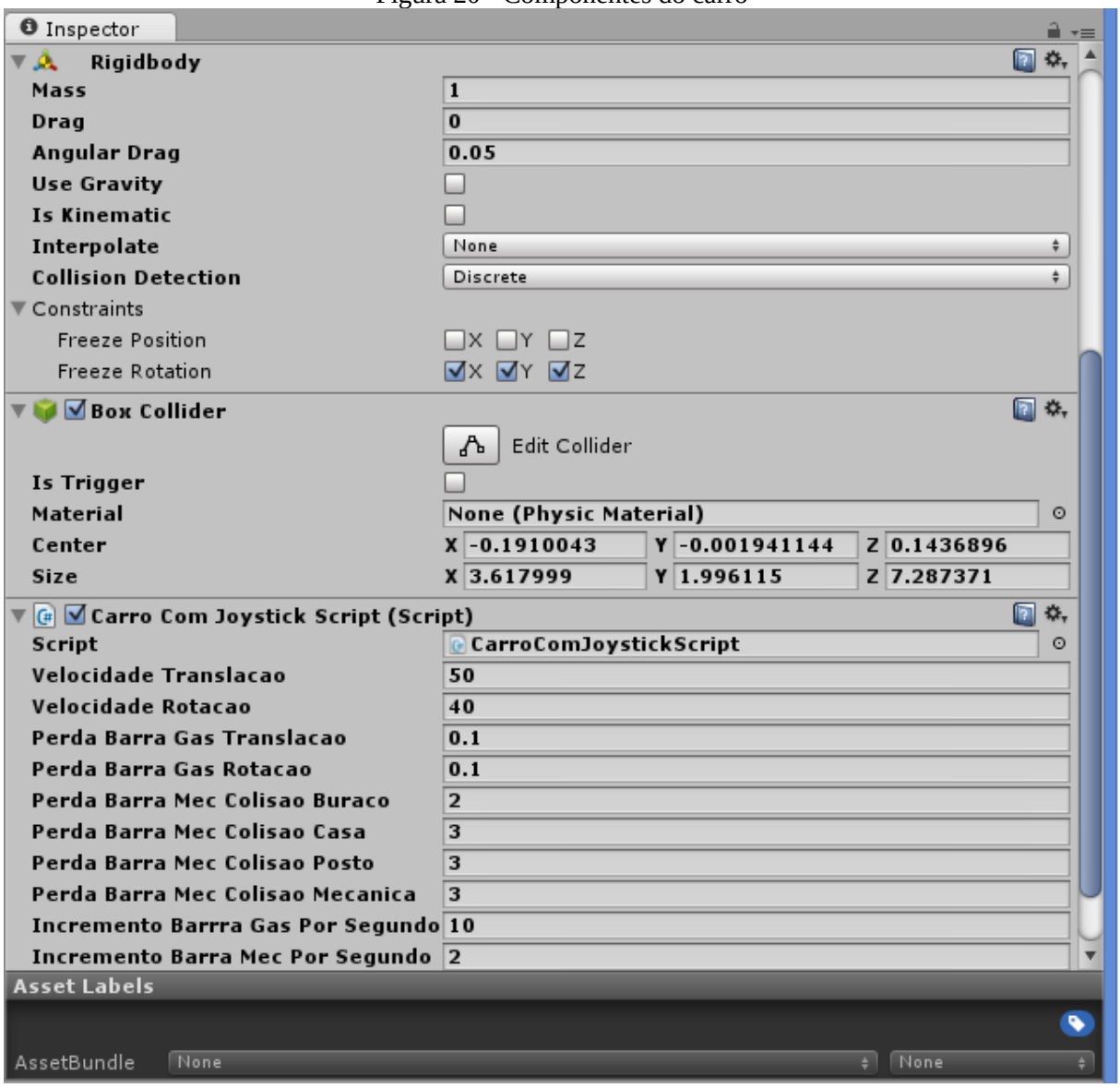
O `item 4` da Figura 17 apresenta o `Prefab` do tipo `Image Target` que é responsável pela geração do mundo virtual 3D (`itens 5` a `11` da Figura 17). Este `Prefab` é o pai dos `itens 5` a `11` da Figura 17. Ele foi configurado para reconhecer a imagem do mapa e marcado para realizar `Extend Tracking` nas opções do *script* `Image Target Behaviour` (Figura 19).

Figura 19 - Opções definidas no marcador



No item 5 é apresentado um modelo em 3D de um carro que é controlado através do *joystick* virtual (item 12) para simular uma movimentação semelhante a de um carro real. Neste modelo foi adicionado um Rigidbody, um *script* e um Box Collider, conforme Figura 20.

Figura 20 - Componentes do carro



No Rigidbody foi desmarcada a opção Use Gravity para o objeto não ser interferido por gravidade e marcadas todas as propriedades do Freeze Rotation, fazendo com que o carro não seja interferido por rotação ao colidir com outro objeto. O Box Collider foi ajustado ao tamanho do carro com a opção Edit Collider. No *script* CarroComJoystickScript são informados atributos para a velocidade do carro ao transladar e rotacionar, perda de barra de gasolina e mecânica, e incremento das barras de mecânica e gasolina (ressalta-se que os valores informados para estes atributos, irão depender dos valores

informados para criar a barra de mecânica e gasolina do *script* `ControleDeBarras`). No Quadro 8 é apresentado o método `Update` do *script* `CarroComJoysctickScript`.

Quadro 8 - Controle do carro

```
void Update ()
{
      float movimentoHorizontal = CrossPlatformInputManager.GetAxis
("Horizontal");

      float movimentoVertical = CrossPlatformInputManager.GetAxis
("Vertical");

      // Rotaçao
      if (movimentoHorizontal != 0 && (movimentoHorizontal > 0.30f ||
movimentoHorizontal < -0.30f)) {
            float rotacao = (movimentoHorizontal * velocidadeRotacao) *
Time.deltaTime;

            if (movimentoHorizontal < 0 && movimentoVertical < 0 ||
                movimentoHorizontal > 0 && movimentoVertical < 0) {
                  transform.Rotate (0, rotacao * -1, 0);
            } else {
                  transform.Rotate (0, rotacao, 0);
            }
            Barras.DiminuiGasolinaAtual (perdaBarraGasRotacao);
      }

      //Translaçao
      if (movimentoVertical != 0) {
            float translacao = (movimentoVertical * velocidadeTranslacao)
* Time.deltaTime;
            transform.Translate (0, 0, translacao);
           Barras.DiminuiGasolinaAtual (perdaBarraGasTranslacao);
      }

      //Mantem a posiçao y fixa devido a rotaçao
      if (movimentoHorizontal != 0 || movimentoVertical != 0) {
            transform.localPosition = new Vector3
(transform.localPosition.x, 0.0087f, transform.localPosition.z);
      }

      if (Barras.gasolinaAtual <= 0) {
            Destroy (gameObject);
      }
}
```

No método `Update` que é chamado a cada *frame* desenhado é aplicada a movimentação do carro, a perda da barra de gasolina e a destruição do carro caso a gasolina se esgote. A variável `movimentoHorizontal` recebe o movimento horizontal aplicado pelo usuário no *joystick* virtual e com isto é calculada a velocidade que o carro rotaciona através do método `transform.Rotate`. A variável `movimentoVertical` recebe o movimento vertical aplicado pelo usuário no *joystick* virtual e com isto é calculada a velocidade que o carro translada através do método `transform.Translate`. Quanto maior for o movimento aplicado no *joystick* virtual, maior é o movimento do carro. Quando houver movimento horizontal ou

vertical, é decrementada a gasolina através do método `DiminuiGasolinaAtual` da instância da classe `ControleBarrasScript (this.Barras)` que se encarrega de atualizar na tela a barra de gasolina. Por fim, é verificado se a barra de gasolina acabou, caso positivo, é chamado o método `Destroy` passando como parâmetro o próprio `Game Object` (carro) que destrói o carro da `Scene`.

No Quadro 9 é apresentado o tratamento para a perda da barra de mecânica. No método `OnCollisionEnter`, se o carro colidir com o posto de gasolina (`item 10`) ou a mecânica (`item 8`) ou uma casa (`itens 7`), é decrementada a barra de mecânica através da instância da classe `ControleBarrasScripts` (`this.barras`) que se encarrega de atualizar na tela a barra de mecânica. Se na colisão a barra de mecânica esgotou, o carro é destruído através do método `Destroy`. O método `OnTriggerEnter` decrementa a barra de mecânica quando o carro colidir com algum objeto do tipo buraco (`item 6` da Figura 17).

Quadro 9 - Tratamento de colisões para decrementar a barra de mecânica

```
void OnCollisionEnter (Collision ObjetoColidido)
{
      switch (ObjetoColidido.gameObject.tag) {
      case "Casa":
            this.Barras.DiminuiMecanicaAtual (perdaBarraMecColisaoCasa);
            break;
      case "Mecanica":
            this.Barras.DiminuiMecanicaAtual
(perdaBarraMecColisaoMecanica);
            break;
      case "PostoGasolina":
            this.Barras.DiminuiMecanicaAtual (perdaBarraMecColisaoPosto);
                  break;
      }
      if (Barras.mecanicaAtual <= 0) {
            Destroy (gameObject);
      }
}
void OnTriggerEnter (Collider ObjetoColidido)
{
      if (ObjetoColidido.gameObject.tag == "Buraco") {
            this.Barras.DiminuiMecanicaAtual (perdaBarraMecColisaoBuraco);
      }
      if (Barras.mecanicaAtual <= 0) {
            Destroy (gameObject);
      }
}
```

No Quadro 10 são apresentados os métodos para incrementar a barra de mecânica e de gasolina. No método `OnTriggerStay`, se o carro permanecer em cima do galão de gasolina (`item 11`), é chamado o método `EncherTanqueGasolina` para incrementar a quantidade da barra de gasolina definida por parâmetro, a cada segundo. Se o carro permanecer em cima chave inglesa (`item 9`), é chamado o método `ArrumarCarro` que incrementa a quantidade da

barra de mecânica definida por parâmetro, a cada segundo. Para que o carro possa ficar em cima do galão de gasolina ou da chave inglesa, estes objetos possuem uma Box Collider com a opção Is Trigger marcada para que o carro possa atravessá-los (não ser afetado por física) e gerar evento.

Quadro 10 - Tratamento de colisão para incrementar barra de gasolina e mecânica

```
void OnTriggerStay (Collider ObjetoColidido)
{
      switch (ObjetoColidido.gameObject.tag) {
      case "ChaveInglesa":
            if (!this.aguardarArrumarCarro) {
                  StartCoroutine (ArrumarCarro ());
            }
            break;
      case "GalaoDeGasolina":
            if (!this.aguardarEncherTanque) {
                  StartCoroutine (EncherTanqueGasolina ());
            }
            break;
      }
}
private IEnumerator EncherTanqueGasolina ()
{
      this.Barras.IncrementaGasolinaAtual (incrementoBarrraGasPorSegundo);
      this.aguardarEncherTanque = true;
      yield return new WaitForSeconds (1);
      this.aguardarEncherTanque = false;
}
private IEnumerator ArrumarCarro ()
{
      this.Barras.IncrementaMecanicaAtual (incrementoBarraMecPorSegundo);
      this.aguardarArrumarCarro = true;
      yield return new WaitForSeconds (1);
      this.aguardarArrumarCarro = false;
}
```

O item 6 da Figura 17 representa um buraco. Para este objeto não foi utilizado um modelo pronto. Foi criado um objeto 3D do tipo Plane e associado uma imagem de buraco. O objeto possui uma Box Collider sem estar marcado com Is Trigger. Quando o carro passar por cima do buraco, será decrementada a barra de mecânica.

O item 7 é um modelo em 3D de uma casa, o item 8 é um modelo em 3D de uma mecânica e o item 10 é um modelo em 3D de um posto de gasolina. Estes modelos somente fazem a barra de mecânica ser decrementada quando houver colisão do carro com um deles.

O *joystick* virtual (item 12) é um Prefab disponibilizado no pacote Vehicles do Unity e nomeado de MobileSingleStickControl. Este Prefab é composto do Game Object principal MobileSingleStickControl responsável pela criação do componente Event System para o gerenciamento do toque na tela e pela habilitação e controle do posicionamento em tela de seu Game Object filho, nomeado de MobileJoystick. No Game

`Object MobileJoystick` é possível definir uma imagem para o *joystick*, a quantia que ele poderá ser arrastado em tela e as direções para arrastá-lo (vertical e/ou horizontal). O `MobileJoystick` também é responsável pelo envio da sua movimentação (horizontal e/ou vertical) para a classe `CrossPlatformInputManager` quando o mesmo for arrastado em tela (o carro utiliza estas informações de movimentação para rotacionar e transladar).

Por fim, o `item 13` é um botão nomeado de `Voltar`. Ao tocar neste botão, é chamado o método `VoltarTelaInicial` apresentado no Quadro 11, que carrega a tela inicial através do método `LoadLevel`.

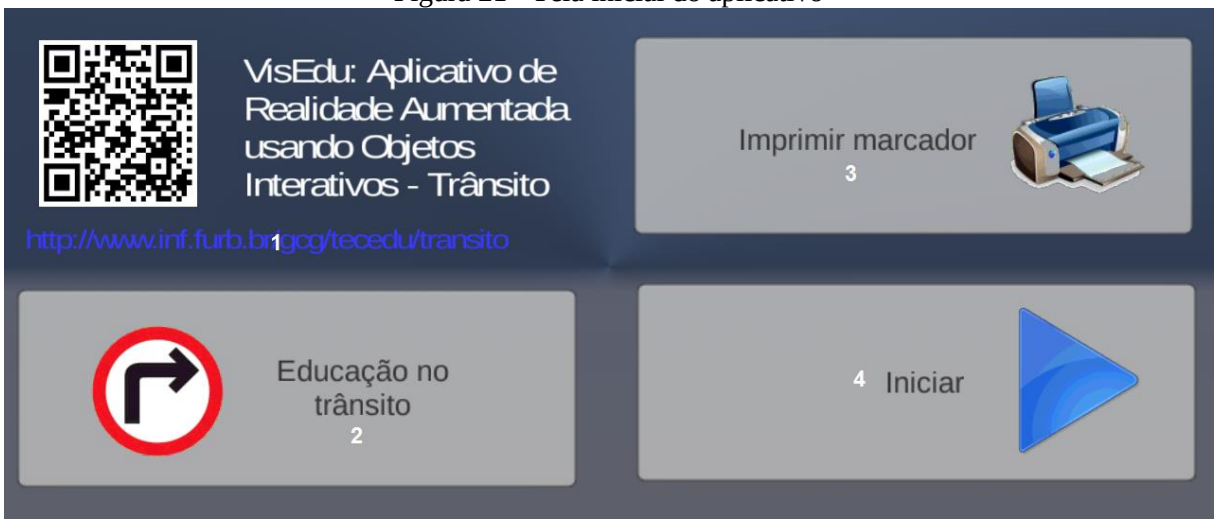
Quadro 11 - Método para carregar a tela inicial do aplicativo

```
public void VoltarTelaInicial ()
{
      Application.LoadLevel ("TelaInicial");
}
```

3.3.1.3 Tela inicial

Quando o aplicativo é aberto, a tela inicial apresenta o título da aplicação, junto com um QRCode e três botões de opções. Abaixo são apresentadas as funcionalidades das opções, conforme Figura 21.

Figura 21 - Tela inicial do aplicativo



O Quadro 12 apresenta os métodos que são executados quando o usuário tocar em algum dos itens apresentados na Figura 21. Ao tocar no `item 1` é chamado o método `Sobre`, que abre o navegador padrão do dispositivo através do método `Application.OpenURL`, passando como parâmetro o caminho do site. Ao tocar no botão do `item 2` é executado o método `EducacaoNoTransito`, que abre um site disponibilizando um conteúdo com informações educativas sobre trânsito. Ao tocar no botão do `item 3` é executado o método

`Imprimir`, que abre um site para visualizar e imprimir o marcador utilizado no aplicativo. Ao tocar no botão do `item 4` é carregada a `Scene` para visualizar a RA do aplicativo.

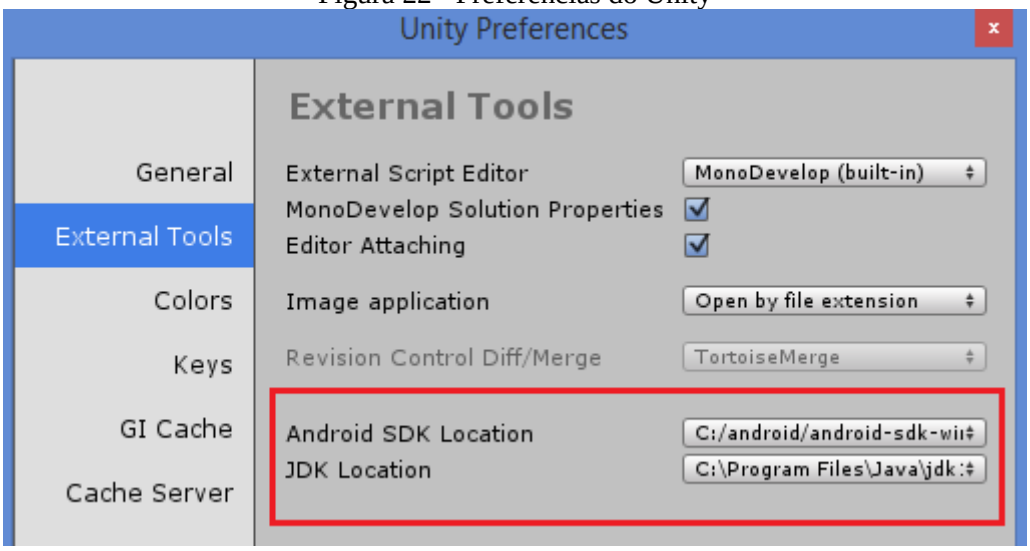
Quadro 12 - Métodos da tela inicial

```
public void Sobre ()
{
      Application.OpenURL
     ("http://www.inf.furb.br/gcg/tecedu/transito");
}

public void EducacaoNoTransito ()
{
      Application.OpenURL
     ("http://www.inf.furb.br/gcg/tecedu/transito/educa.pdf");
}

public void Imprimir ()
{
      Application.OpenURL
     ("http://www.inf.furb.br/gcg/tecedu/transito/mapa.pdf");
}

public void Iniciar ()
{
      Application.LoadLevel ("ControlePorJoystick");
}
```

3.3.1.4 Plataforma disponibilizada para o aplicativo

O aplicativo foi disponibilizado para a plataforma Android nas versões 4.4 e superiores. Para gerar o projeto para dispositivos com sistema operacional Android é necessário definir nas preferências do Unity o local onde se encontra o SDK do Android e o JDK. A Figura 22 apresenta estas configurações.

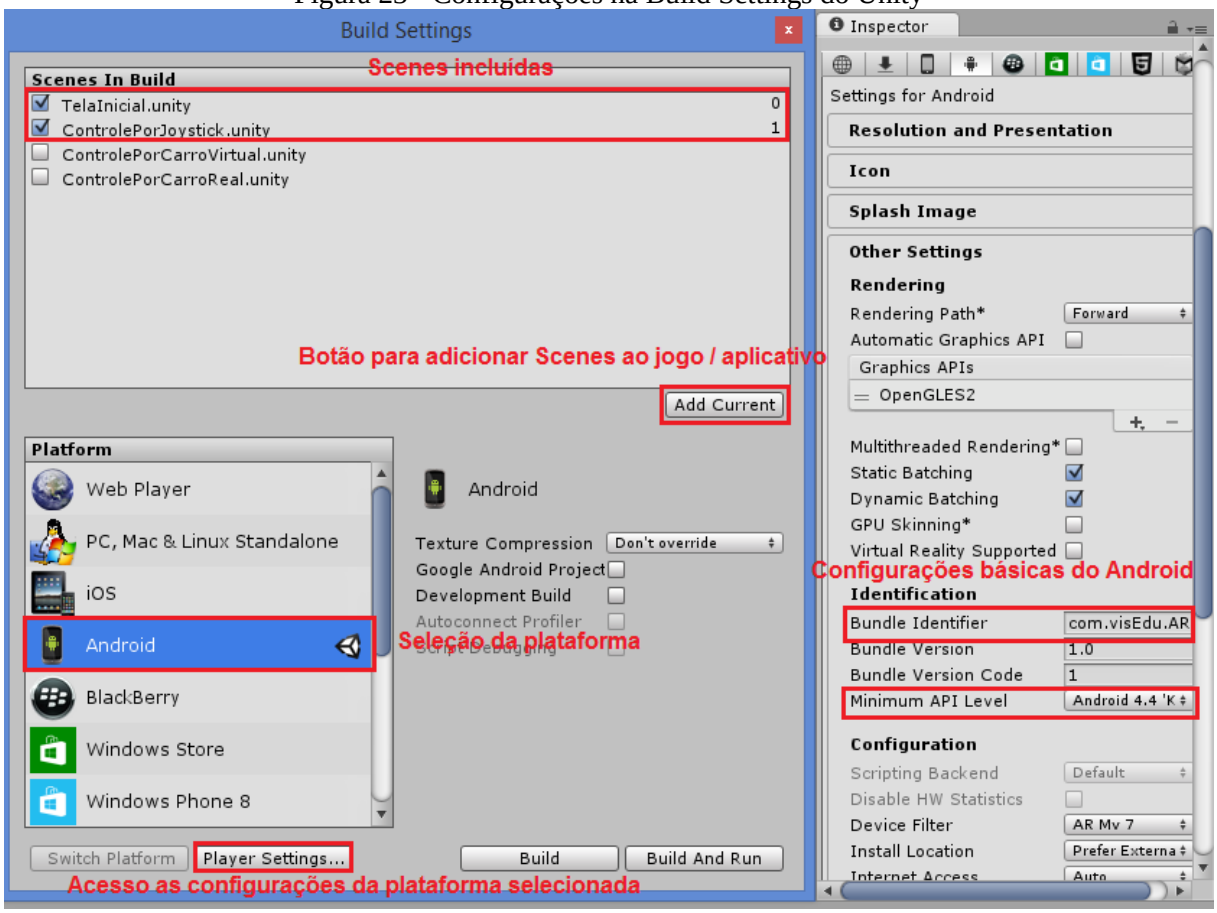
Figura 22 - Preferencias do Unity



Após as configurações, é necessário acessar as `Build Settings` do Unity, selecionar a plataforma Android e clicar em `Player Settings`. As configurações básicas do `Player`

`Settings` para a geração do aplicativo são: definir um `Bundle Identifier` (identificação do pacote) e `Minimum API Level` (versão mínima do Android para rodar a aplicação). Para que todas `Scenes` sejam incluídas no aplicativo é necessário adicioná-las uma por uma através do botão `Add Current` e definir uma ordem de execução. Na Figura 23 são apresentadas as opções descritas acima.

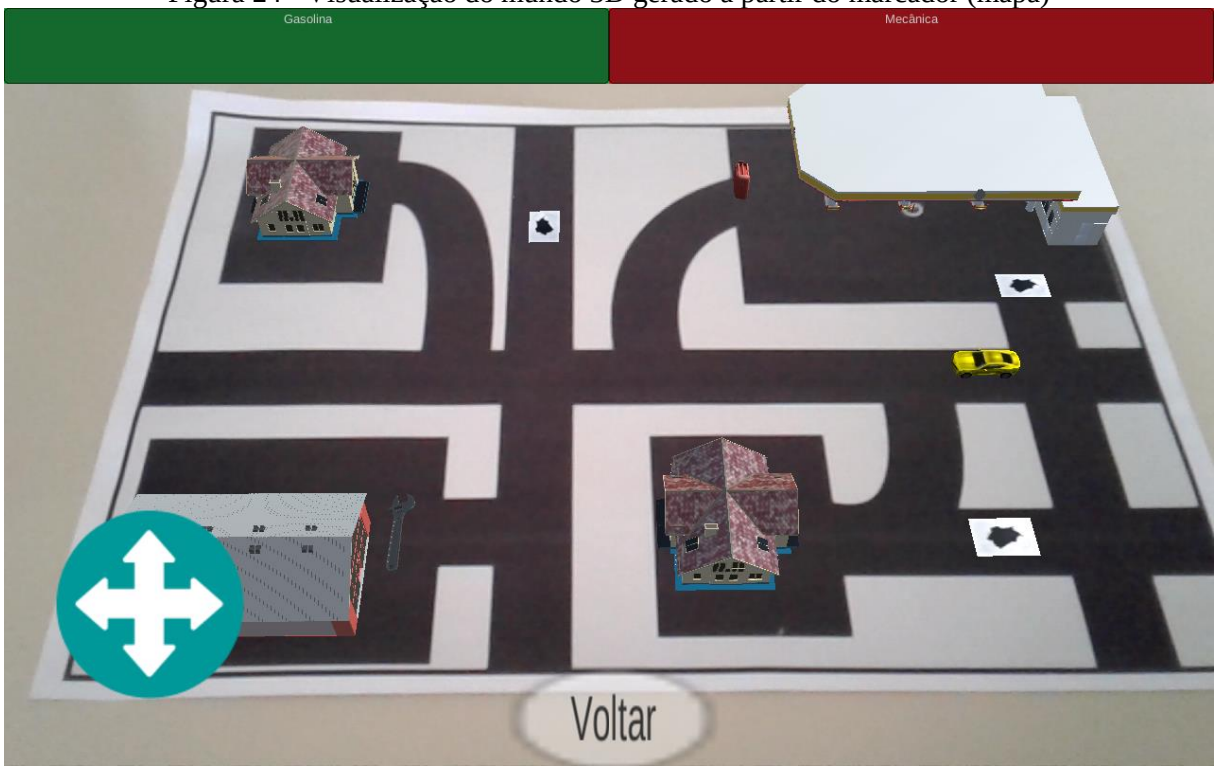
Figura 23 - Configurações na Build Settings do Unity



3.3.2 Operacionalidade da implementação

Ao iniciar o aplicativo, na tela inicial o usuário deve imprimir o marcador através da página que é aberta para visualização ao tocar no botão `Imprimir marcador`. Após a impressão do marcador, o usuário pode visualizar instruções básicas sobre o trânsito através do toque no botão `Educação no trânsito`. Ao tocar no botão `Iniciar`, é carregada uma tela no aplicativo que funciona através da câmera do dispositivo. O usuário deve visualizar o marcador do aplicativo, fazendo com que os objetos virtuais sejam gerados (Figura 24).

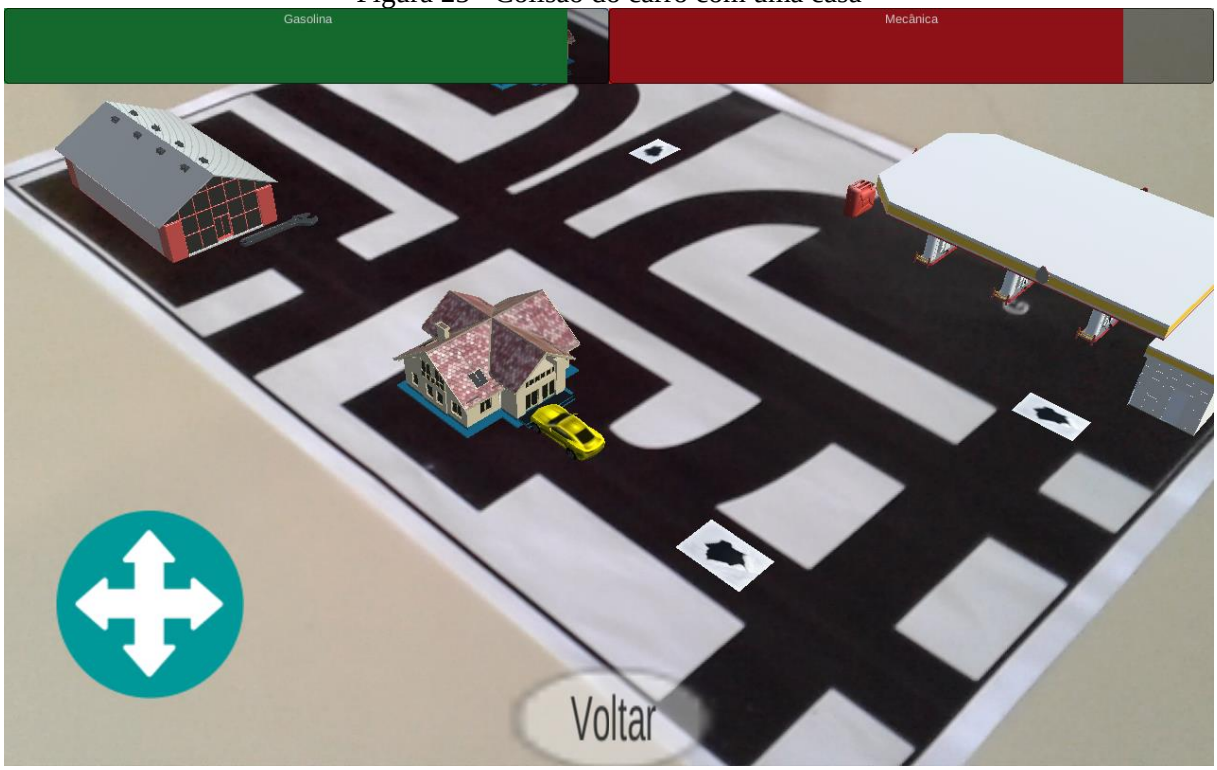
Figura 24 - Visualização do mundo 3D gerado a partir do marcador (mapa)



Conforme o movimento aplicado no *joystick* virtual, o carro se movimenta. A direção do movimento do carro depende da posição em que o usuário estiver em relação a posição do carro (isto ocorre devido ao usuário controlar a direção da câmera). Caso o usuário fique de frente para a traseira do carro, os movimentos aplicados no carro correspondem a direção do movimento do *joystick.* Caso o carro esteja com a sua dianteira de frente para o usuário, o carro se movimenta ao contrário do movimento aplicado no *joystick* (para uma melhor utilização do *joystick*, é recomendado que o usuário fique de frente para a traseira do carro). Quanto maior a direção do movimento que o usuário aplicar no *joystick*, maior é a velocidade de locomoção do carro. Ressalta-se que o carro pode ser movimentado em qualquer direção do ambiente real, não estando restrito as linhas (ruas) e ao marcador (mapa).

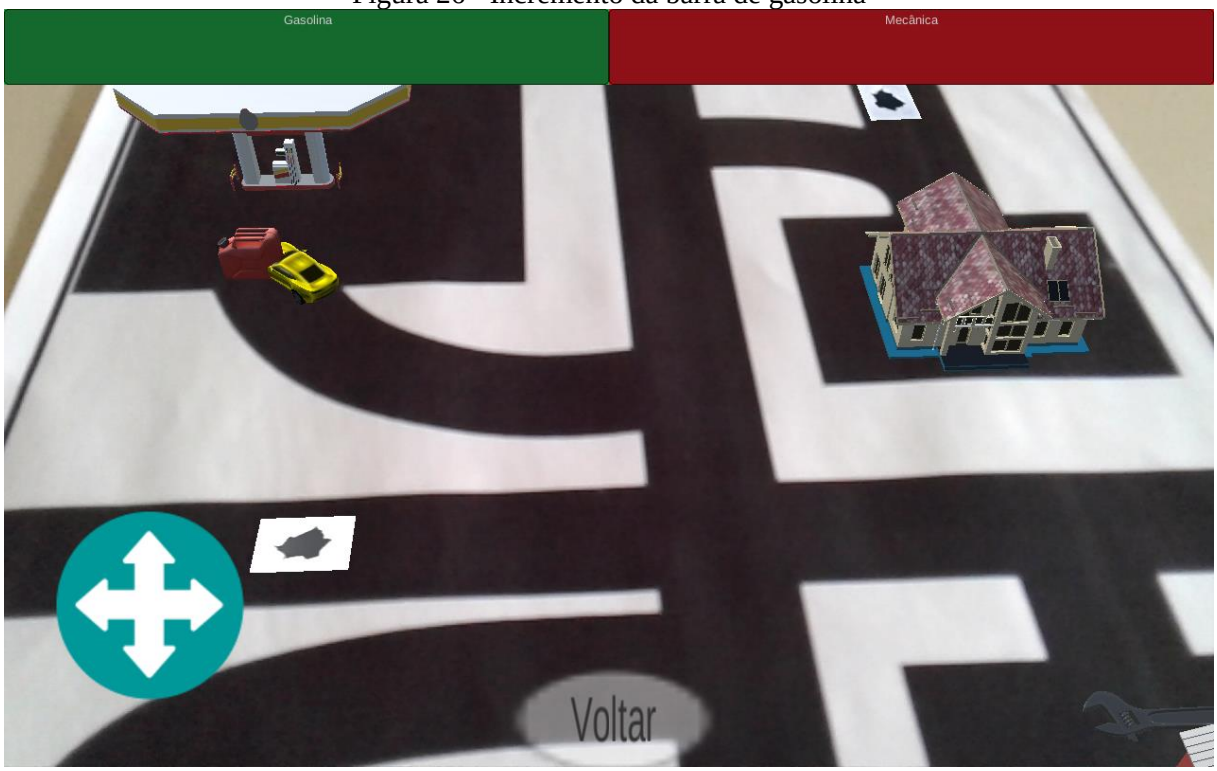
Enquanto o carro estiver se movimentando, a barra de gasolina é decrementada. Quando o carro colidir com o posto de gasolina, alguma casa, a mecânica ou algum buraco, a barra de mecânica será decrementada. Conforme a Figura 25, o carro colidiu com a casa e a barra de mecânica foi decrementada.

Figura 25 - Colisão do carro com uma casa



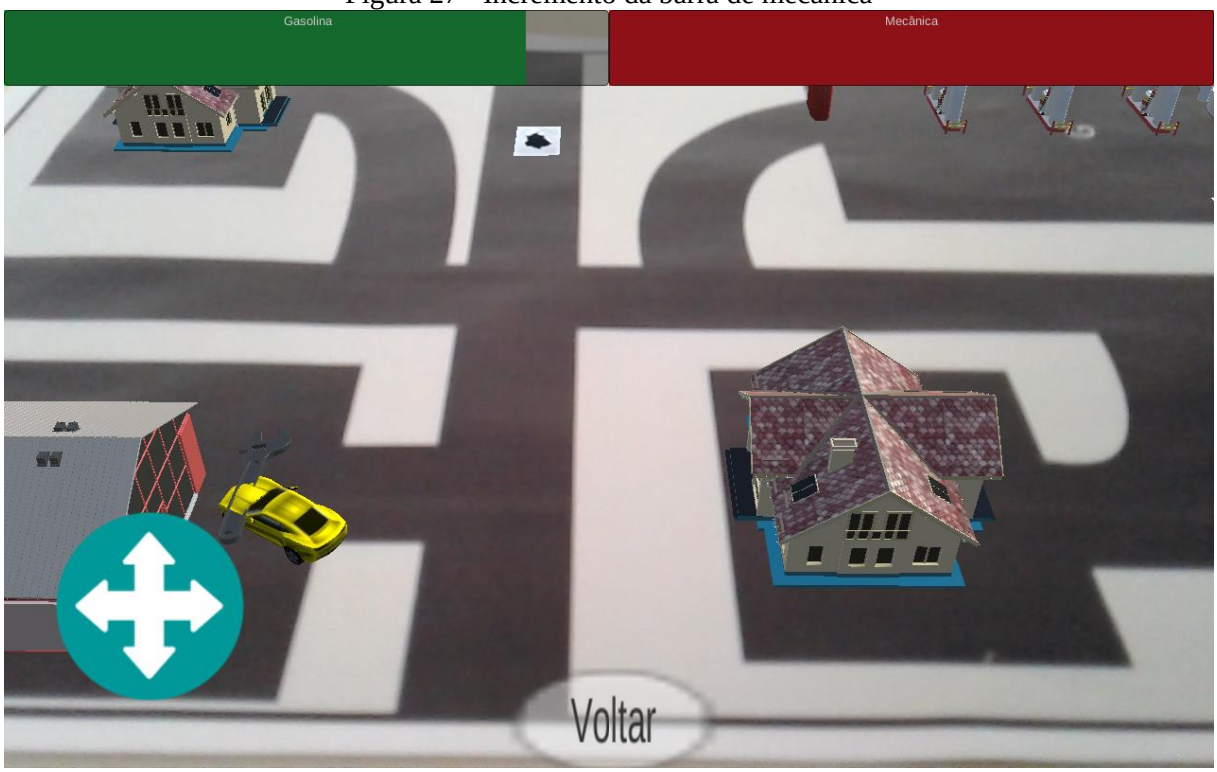
Enquanto o carro estiver em cima do galão de gasolina, a barra de gasolina é incrementada. Na Figura 26 é visualizado o carro em cima do galão de gasolina para o incremento da barra.

Figura 26 - Incremento da barra de gasolina



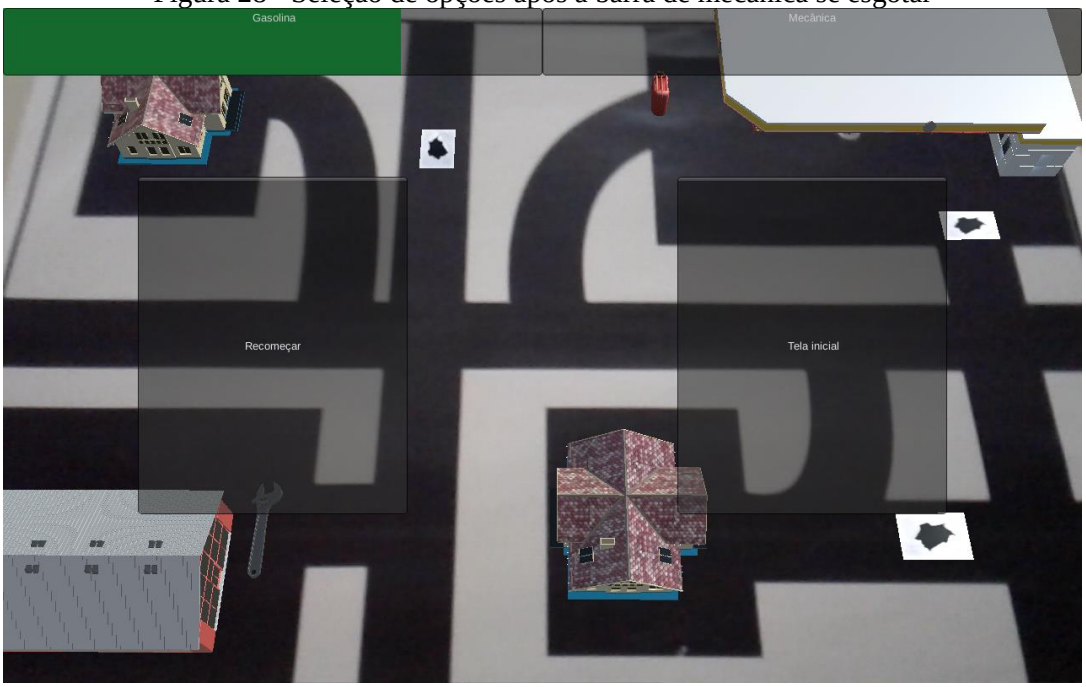
Enquanto o carro estiver em cima da chave inglesa, a barra de mecânica é incrementada. Na Figura 27 é visualizado o carro em cima da chave inglesa para o incremento da barra de mecânica.

Figura 27 - Incremento da barra de mecânica



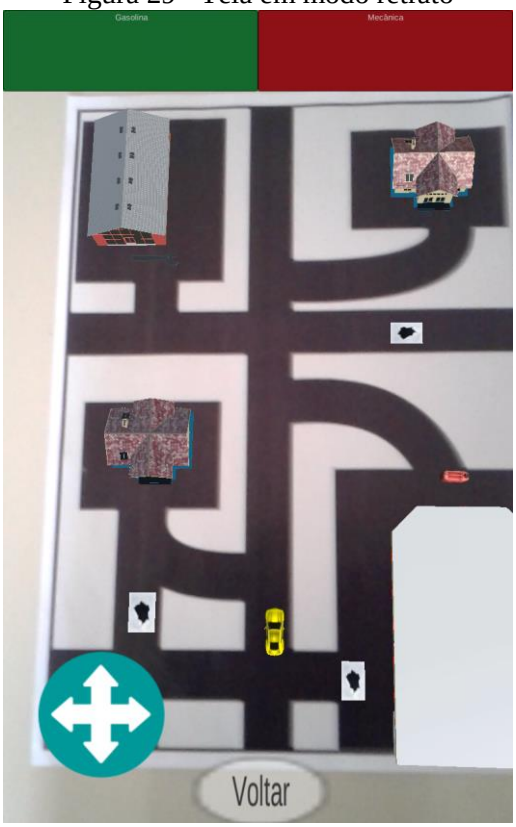
Caso a barra de gasolina ou mecânica se esgote, o carro é destruído e são exibidos dois botões com opções para voltar para tela inicial ou recomeçar. Na Figura 28 são apresentadas as opções disponíveis após a barra de mecânica se esgotar.

Figura 28 - Seleção de opções após a barra de mecânica se esgotar



As figuras apresentadas anteriormente nesta seção foram tiradas com o aplicativo em modo de visualização paisagem. Conforme Figura 29, também é possível utilizar o aplicativo em modo de visualização retrato.

Figura 29 - Tela em modo retrato



3.4 RESULTADOS E DISCUSSÕES

O presente trabalho teve como objetivo inicial a criação de três cenários para o desenvolvimento do ambiente de simulação (cenário simples de trânsito). Estes três cenários foram desenvolvidos de forma semelhante, porém, tendo como principal diferença, a forma com que o usuário irá interagir com os objetos do mundo virtual 3D. Nota-se que a interação do usuário resulta na movimentação de um carro virtual que interage com os demais objetos do mundo virtual 3D. No item 3.4.1 abaixo são apresentados os cenários desenvolvidos para a aplicação e no item 3.4.2 são apresentados os testes realizados com os cenários. No Apêndice A descreve-se a exibição do aplicativo gerado por este trabalho no evento XI Seminário Integrado das Licenciaturas.

3.4.1 Cenários

O primeiro cenário desenvolvido para o ambiente de simulação (cenário simples de trânsito) foi projetado para o usuário controlar um carro virtual 3D a partir da detecção de um

Frame Marker. O Frame Marker seria movimentado pelo usuário no mundo virtual 3D gerado a partir da detecção de uma Image Target, configurada com a imagem de um mapa. No Unity foi desenvolvida a Scene ControlePorCarroVirtual para atender ao cenário.

A Scene ControlePorCarroVirtual foi desenvolvida para dois usuários poderem utilizar a aplicação, porém em dispositivos separados. Com isto, foi desenvolvida a Scene SelecaoMarcador permitindo a escolha de um carro virtual na cor amarela através da detecção de um Frame Marker ou o mesmo modelo de carro, na cor vermelha através da detecção de outro Frame Marker. Na aplicação, os usuários poderiam movimentar seus Frame Markers, colidir seus carros com os modelos gerados a partir do mundo virtual 3D e inclusive com o carro do usuário oposto. Para as colisões, uma barra representando a mecânica do carro geraria um retorno visual. Para o movimento do Frame Marker, uma barra representando a gasolina do carro geraria um retorno visual. Do lado esquerdo da Figura 30 é apresentada a visualização do cenário real sem utilizar a Scene ControlePorCarroVirtual, e do lado direito é apresentada a visualização do cenário real pela Scene ControlePorCarroVirtual.

Figura 30 – Visualização do cenário real sem e com a Scene ControlePorCarroVirtual

Nota-se na Figura 30 que foram colados palitos nos Frame Markers para realizar o seu movimento, sem precisar tocar na imagem destes marcadores para realizar o movimento. É necessário estar visualizando todo o Frame Marker para o seu reconhecimento, e posteriormente a geração de objetos virtuais.

O segundo cenário foi desenvolvido para o usuário poder utilizar um carro de brinquedo no mundo virtual 3D (gerado por uma Image Target) através de um Frame Marker colado em cima do carro. No Unity foi desenvolvida a Scene ControlePorCarroReal para atender ao cenário.

A Scene ControlePorCarroReal foi desenvolvida semelhante a Scene ControlePorCarroVirtual. Na Scene ControlePorCarroReal também é necessário que os usuários selecionem um Frame Marker, porém não sendo gerados os carros virtuais 3D,

apenas uma esfera na cor amarela ou vermelha dependendo do `Frame Marker` selecionado, para identificar que o marcador foi reconhecido. Os marcadores dos carros de brinquedo poderiam colidir com os objetos virtuais do mundo 3D e inclusive com o carro oposto, devido ao `Frame Marker` colado. Do lado esquerdo da Figura 31 é apresentada a visualização do cenário real sem utilizar a `Scene ControlePorCarroReal`, e do lado direito é apresentada a visualização do cenário real pela `Scene ControlePorCarroReal`. Na Figura 32 é exibida a visualização da frente dos carrinhos utilizados.

Figura 31 – Visualização do cenário real sem e com a `Scene ControlePorCarroReal`

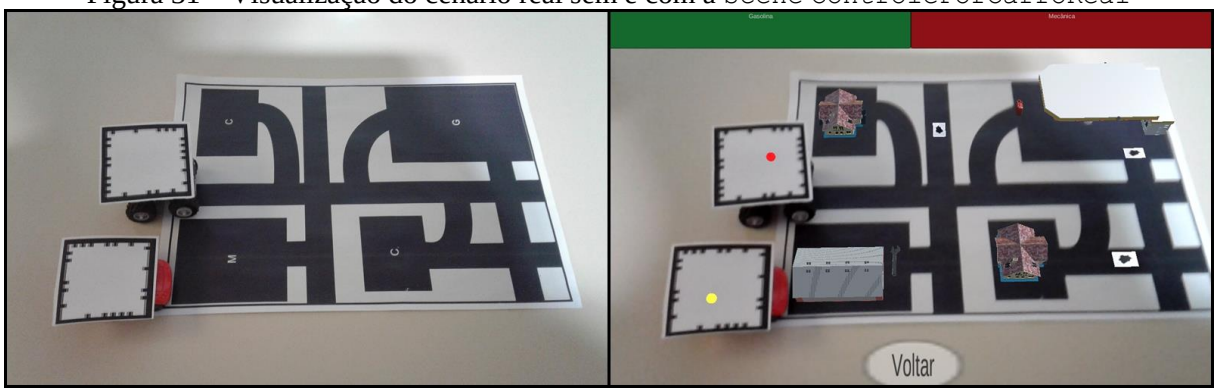
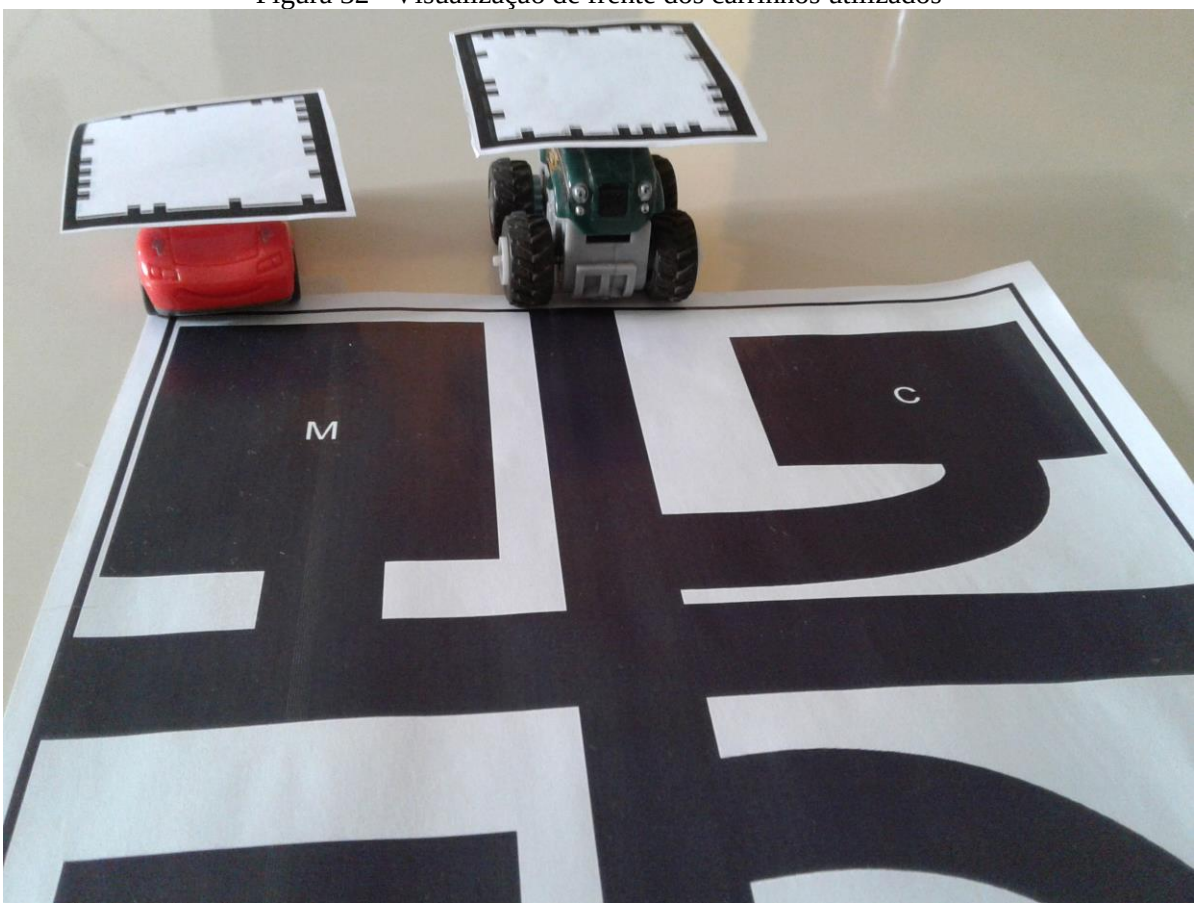


Figura 32 - Visualização de frente dos carrinhos utilizados



O último cenário foi desenvolvido para o controle de um carro virtual 3D a partir da interação com um *joystick*, assim sendo nomeado de `ControlePorJoystitck`. O mundo

virtual 3D é semelhante ao do primeiro e segundo cenário, porém o carro faz parte dos objetos da `Image Target`, não sendo gerado por um `Frame Marker`. Este cenário se encontra detalhado na seção 3.3. Ressalta-se que este cenário foi projetado apenas para um usuário, devido ao movimento do carro ser controlado internamente através do movimento do *joystick*. Do lado esquerdo da Figura 33 é apresentada a visualização do cenário real sem utilizar a `Scene ControlePorJoystick`, e do lado direito é apresentada a visualização do cenário real pela `Scene ControlePorJoystick`.

Figura 33 - Visualização do cenário real sem e com a `Scene ControlePorJostick`

Nos testes realizados com o cenário `ControlePorCarroVirtual`, foi utilizado um `Frame Marker` impresso em tamanho 6 x 6 centímetros e a imagem do mapa impressa no tamanho A4 (21,0 x 29,7 centímetros) e no tamanho A0 (84,1 x 118,9 centímetros). Foi utilizado um `Frame Marker` com tamanho 6 x 6 centímetros devido a dificuldade da câmera manter em foco um `Frame Marker` com tamanho menor. Não foi possível implementar controle de física, devido a possibilidade de movimentar o `Frame Marker` para dentro de um objeto virtual 3D, fazendo o carro perder seu posicionamento original em cima do marcador. Também constatou-se dificuldades para movimentar o `Frame Marker` em cima da imagem do mapa em tamanho A4 devido ao pouco espaço para movimentação no mapa. Utilizando o mapa A0, não foi possível movimentar o `Frame Marker` em cima do mesmo, devido a dificuldade de visualização da imagem do mapa inteiro na tela do dispositivo e movimentação do marcador.

O cenário `ControlePorCarroReal` é semelhante ao `ControlePorCarroVirtual`, com isto, constatou-se quase os mesmos problemas, porém movimentando um carro real (de tamanho que possa colar um `Frame Marker` 6 x 6 centímetros) com um marcador colado em cima dele. O problema distinto neste cenário é que o marcador ficaria elevado em cima do carro de brinquedo, com isto, o seu componente de colisão foi configurado com uma maior altura para conseguir colidir com os objetos virtuais.

O cenário `ControlePorJoystick` apresentou-se estável nos testes realizados com a imagem do mapa em tamanho A4 e A0, possibilitando a movimentação do carro, com a interatividade do *joystick*. Neste cenário, a posição do movimento no *joystick* aplicada pelo usuário e posteriormente utilizada para realizar a translação e rotação do carro virtual teve problemas devido ao controle da câmera ser realizada pelo usuário. Para que a direção do movimento aplicado no *joystick* corresponda a mesma direção do movimento do carro, o usuário deverá estar sempre posicionado de frente para a traseira do carro, caso contrário, o carro irá realizar o movimento contrário ao aplicado no *joystick*.

Apenas o cenário `ControlePorJoystick` foi mantido no aplicativo (ver seção 3.3). Os cenários `ControlePorCarroVirtual` e `ControlePorCarroReal` foram removidos do aplicativo, porém mantidas no projeto no Unity.

A Tabela 1 demonstra as características dos modelos em 3D utilizados nas `Scenes` do aplicativo: buraco, esfera, casa, carro, chave inglesa, galão de gasolina, mecânica e posto de gasolina. Para obter as informações, foi utilizado o software de modelagem Blender, o qual permite a importação de modelos em 3D já criados. As informações listadas são a quantidade de vértices e a quantidade de faces no qual foram utilizadas para criar os modelos 3D.

Tabela 1- Características dos modelos em 3D do aplicativo

Modelo	Quantidade de vértices	Quantidade de faces
Buraco	121	200
Casa	40.028	20.228
Carro	979	916
Chave inglesa	25.637	36.554
Galão de gasolina	908	914
Mecânica	26.072	16.690
Posto de gasolina	21.222	33.964
Esfera	515	768

Observando a Tabela 1 percebe-se que objetos de forma simples (que não precisariam de tantos detalhes) possuem um número elevado de vértices e faces. Desta forma, deve-se fazer um estudo mais detalhado para escolher quais tipos de objetos gráficos utilizar na cena levando em consideração o seu contexto dentro da aplicação. A elevada quantidade de vértices e faces em modelos virtuais podem causar um grande custo computacional, dificultando a *renderização* e interação em tempo real.

3.4.2 Testes

Os testes foram realizados em cima das três `Scenes` desenvolvidas: `ControlePorCarroVirtual`, `ControlePorCarroReal` e `ControlePorJoystick`. Para as `Scenes ControlePorCarroVirtual` e `ControlePorCarroReal` foram utilizados dois `Frame Markers` com tamanho 6x6 centímetros e uma imagem no tamanho A4 (21,0 x 29,7 centímetros). Para a `Scene ControlePorJoystick` foi utilizado apenas uma imagem no tamanho A4 (21,0 x 29,7 centímetros). Para a execução dos mesmos, foi utilizado o dispositivo Samsung Galaxy tab E com sistema operacional Android 4.4, 1,5 GB de memória RAM e processador *Quad-Core* de 1.3 GHz.

O primeiro teste realizado foi referente ao consumo de memória utilizado pelos cenários. Para o teste, foi aberto o aplicativo na `Scene` correspondente e detectados os marcadores informados acima, correspondentes com a `Scene`. Para a obtenção dos valores, foi verificado o consumo no gerenciador de aplicações do Android. A Tabela 2 apresenta os valores obtidos.

Tabela 2 - Consumo de memória das `Scenes`

`Scene`	Consumo de memória em MB
`ControlePorCarroVirtual`	82
`ControlePorJoystick`	72
`ControlePorCarroReal`	81

Com base nos resultados disponibilizados na Tabela 2, é possível observar que as `Scenes ControlePorCarroVirtual` e `ControlePorCarroReal` tiveram um consumo de memória semelhante, decorrente dos seus cenários não possuírem diferenças significativas. Comparando este resultado com o obtido a partir da `Scene ControlePorJoystick`, é constatado um menor consumo de memória por parte desta `Scene`. Acredita-se que devido a `Scene ControlePorJoystick` não precisar detectar os marcadores distintos das outras `Scenes`, houve esta variação. Porém, para se chegar a um parecer concreto seria necessário realizar um aprofundamento e investigações quanto aos fatores envolvidos nesses resultados.

O segundo teste realizado foi referente aos *Frames Per Second* (FPS), que são os quadros desenhados por segundo. Para a obtenção dos valores, foi desenvolvida uma versão de cada uma das `Scenes` de teste que possibilita a medição do FPS. A execução destes testes foi realizada em um local fixo para visualização com a câmera. Foram realizados testes visualizando apenas o cenário (sem marcadores) e visualizando todos os objetos virtuais gerados nos marcadores. A Tabela 3 apresenta os valores obtidos.

Tabela 3 - Taxa de FPS das `Scenes`

`Scene`	Quantidade de quadros desenhados por segundo sem visualizar marcadores	Quantidade de quadros desenhados por segundo visualizando marcadores (geração de objetos virtuais)
`ControlePorCarroVirtual`	Entre 34 a 36	Entre 14 a 15
`ControlePorCarroReal`	Entre 34 a 36	Entre 14 a 15
`ControlePorJoystick`	Entre 34 a 36	Entre 14 a 15

Com base nos resultados disponibilizados na Tabela 3, é verificado que não houve variação na taxa de FPS entre as `Scenes` (visualizando e não visualizando marcadores). As `Scenes` compartilham os mesmos objetos em 3D, a não ser pela *renderização* de dois carros na `ControlePorCarroVirtual` em relação a apenas um na `ControlePorJoystick`, e duas esferas na `ControlePorCarroReal`. Esta diferença não impactou nos resultados obtidos. Após a *renderização* dos objetos virtuais visualizados através dos marcadores, foi constatada uma queda significativa na taxa de FPS, isto decorrente da utilização de alguns modelos em 3D com muitos detalhes (ver Tabela 1), os quais ocasionaram uma maior quantidade de processamento gráfico para execução.

3.5 COMPARATIVO DOS TRABALHOS CORRELATOS

Esta seção é responsável por realizar a comparação dos trabalhos correlatos descritos no capítulo anterior e este trabalho. O Quadro 13 apresenta a relação de existência de características nos trabalhos apresentados. O presente trabalho está identificado como Santos (2015).

Quadro 13 - Comparativo dos trabalhos correlatos e este trabalho

Características/Trabalhos relacionados	ARhrrrr! (2009)	ARDefender (2012)	eyeBuy (2012)	Santos (2015)
Permite a interação entre objetos	X	X		X
Permite a utilização de objetos naturais como marcadores	X		X	X
Permite a utilização de objetos fiduciais como marcadores		X		
É um trabalho acadêmico	X		X	X
É um jogo de RA	X	X		
É uma aplicação de RA			X	X
Tecnologias	Não identificadas	Não identificadas	Unity e Vuforia	Unity e Vuforia

Através das informações disponíveis no Quadro 13, nota-se que os trabalhos possuem, em sua maioria, características semelhantes entre si. A utilização de objetos naturais como marcadores só não é presente no jogo ARDefender o qual utiliza os tradicionais marcadores fiduciais. Os jogos ARhrrrr!, ARDefender e este trabalho permitem a interatividade entre os objetos virtuais, sendo um atrativo para a aplicação, ao contrário do eyeBuy que apenas gera um objeto virtual 3D para visualização.

Nota-se que o presente trabalho e o aplicativo eyeBuy são aplicativos desenvolvidos academicamente e criados utilizando as mesmas tecnologias (Unity e Vuforia). Ao contrário destes trabalhos, o ARhrrrr! e ARDefender são jogos, e somente o ARhrrrr! foi desenvolvido academicamente. As tecnologias utilizadas para o desenvolvimento dos jogos não foram identificadas.

4 CONCLUSÕES

O presente trabalho apresentou a criação de um aplicativo de Realidade Aumentada com objetos interativos para a plataforma Android. O aplicativo seguiu as propriedades definidas por Azuma (1997) para a existência da RA, as quais são descritas na seção 2.2.

Os objetivos deste trabalho foram definir um mundo virtual 3D usando um mapa como marcador, utilizar um *joystick* virtual para interagir com os objetos deste mundo e disponibilizar um retorno visual da colisão e movimentação dos objetos. Para alcançar os objetivos do trabalho, foi utilizada a plataforma de desenvolvimento de jogos e aplicativos Unity 3D em conjunto com a biblioteca Vuforia, a qual disponibilizou os recursos necessários para a implementação da RA do aplicativo. A combinação do Unity com Vuforia pode ser considerada adequada, pois forneceu os recursos para a implementação do trabalho.

Ao utilizar em conjunto Unity com Vuforia, foi possível utilizar a imagem de um mapa para a geração de objetos 3D através da câmera de um dispositivo. O Unity possibilitou a utilização de um *joystick* virtual para a interatividade do usuário com um carro virtual e os demais objetos gerados, tendo como retorno visual, duas barras dinâmicas que tinham sua largura decrementada ou incrementada conforme movimento do carro e colisão com os demais objetos virtuais.

Os objetivos propostos foram atingidos. Com isto, nota-se que inicialmente foram definidos três cenários para o ambiente de simulação (cenário simples de trânsito) os quais definiam a forma que o usuário iria interagir com o cenário. Os três cenários foram desenvolvidos, porém apenas um dos cenários foi mantido para o aplicativo. No primeiro cenário, o usuário utilizaria um marcador para movimentar e interagir um carro no mundo virtual 3D. No segundo cenário, o usuário utilizaria um marcador colado em cima de um carro de brinquedo real para movimentação e interação no mundo virtual 3D. Estes cenários não foram mantidos devido a dificuldades na movimentação dos marcadores no mundo virtual e visualização no aplicativo. O último cenário o qual foi mantido no aplicativo, utilizou um *joystick* virtual para a movimentação do carro no mundo virtual 3D e consequentemente interagir com os demais objetos.

Por fim, o trabalho se mostrou de grande valia, pois, gerou um aplicativo que poderá ser estendido para outros cenários que utilizem RA como foco do aplicativo. O trabalho também poderá ser incrementado para a utilização no ensino e na aprendizagem sobre a educação no trânsito através da RA e da interatividade proporcionada. Por ter sido desenvolvido com a utilização do Unity e do Vuforia, destaca-se a facilidade em gerar o

aplicativo para outras plataformas, por exemplo, o iOS. Com base no estudo realizado pelo trabalho, outros da mesma área poderão ser criados, com base nas técnicas apresentadas.

4.1 EXTENSÕES

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

a) utilizar carros de brinquedo (com `Object Scannig Vuforia`) para realizar a interatividade com os objetos virtuais do mapa;
b) realizar um comparativo das principais diferenças entre utilizar `Frame Markers` ao utilizar `Image Targets`;
c) avaliar os tipos de objetos gráficos a serem utilizados na cena;
d) implementar novos mundos 3D e mapas;
e) incrementar os recursos do aplicativo a fim de torná-lo um jogo lúdico;
f) adaptar o carro para utilizar o componente `Wheel Collider`;
g) utilizar Realidade Misturada para permitir que o espaço virtual venha alterar o espaço real, e não só o real altere o virtual;
h) utilizar o carro disponibilizado no pacote `Vehicles` do Unity;
i) testar outras formas de controle do carro que sejam independentes da posição do usuário;
j) possibilitar que vários usuários possam utilizar simultaneamente o aplicativo através da rede WiFi;
k) aprimorar o conteúdo disponibilizado sobre o trânsito;
l) testar a utilização de uma WebCam para aplicar os cenários `ControlePorCarroVirtual` e `ControlePorCarroReal`.

REFERÊNCIAS

APPLE. **ARDefender**. [S.l.], 2015. Disponível em: <https://itunes.apple.com/br/app/ardefender/id393879882?mt=8>. Acesso em: 06 maio 2015.

APPSDOIPHONE. **ARDefender**. [S.l.], 2012. Disponível em: <http://www.appsdoiphone.com/2012/02/ardefender.html>. Acesso em: 21 mar. 2015.

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

CAROTENUTO, Attilio. **Unity 3D: Collisions Basics.** [S.l.], 2014. Disponível em: <https://www.binpress.com/tutorial/unity-3d-collisions-basics/114>. Acesso em: 14 out. 2015.

DOLZ, Jose. **Markerless Augmented Reality**. [S.l.], 2012. Disponível em: <http://www.arlab.com/blog/markerless-augmented-reality>. Acesso em: 04 abr. 2015.

FERNANDES, G. A. **Realidade Aumentada Aplicada a Atividades de Inspeção e Manutenção em Engenharia Civil**. 2012. 106 f. Tese (Doutorado em Engenharia Civil) - Curso de Pós-Graduação em Engenharia Civil, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

FORREST, Brady. **ARhrrrr! : Augmented Reality Zombie and Helicopter Game**. [S.l.], 2009. Disponível em: <http://radar.oreilly.com/2009/06/arhrrrr-augmented-reality-zomb.html>. Acesso em: 16 mar. 2015.

FORTE, C. E.; SILVA, S. L.; MARENGONI, M. Uso de Realidade Aumentada em Interface Acessível para Consumidores com Redução de Acuidade Visual. In WORKSHOP DE REALIDADE VIRTUAL E AUMENTADA, 9., 2012, Paraná. **Anais eletrônicos**... Paraná: 2012. Disponível em <http://www.lbd.dcc.ufmg.br/colecoes/wrva/2012/0035.pdf>. Acesso em: 29 mar. 2015.

GEORGIA TECH. **ARhrrrr!**. [S.l.], 2015. Disponível em: <http://ael.gatech.edu/lab/research/games/arhrrrr>. Acesso em: 16 mar. 2015.

GOOGLE. **ARDefender**. [S.l.], 2013. Disponível em: <https://play.google.com/store/apps/details?id=net.int13.ardefender>. Acesso em: 21 mar. 2015.

HAUSTSCH, Oliver. **Como funciona a Realidade Aumentada**. [S.l.], 2009. Disponível em: <http://www.tecmundo.com.br/realidade-aumentada/2124-como-funciona-a-realidade-aumentada.htm>. Acesso em: 30 mar. 2015.

KARASINSKI, Eduardo. **Novas tecnologias: jogos interativos**. [S.l.], 2009. Disponível em: <http://www.tecmundo.com.br/webcam/2394-novas-tecnologias-jogos-interativos.htm>. Acesso em: 16 mar. 2015.

PASSOS, Erick Baptista et al. Tutorial: Desenvolvimento de jogos com Unity 3D. In: VIII BRAZILIAN SYMPOSIUM ON GAMES AND DIGITAL ENTERTAINMENT, 8., 2009, Rio de Janeiro. **Anais eletrônicos...** Rio de Janeiro: PUC-Rio, 2009. Disponível em: <http://sbgames.org/papers/sbgames09/computing/tutorialComputing2.pdf>. Acesso em: 12 out. 2015.

QUALCOOM. **Attributes of an Ideal Image Target.** [S.l.], 2015a. Disponível em: <https://developer.vuforia.com/library/articles/Best_Practices/Attributes-of-an-Ideal-Image-Target>. Acesso em: 20 out. 2015.

_____. **Augmented Reality (Vuforia).** [S.l.], 2015b. Disponível em: <https://www.qualcomm.com/products/vuforia/features>. Acesso em: 23 mar. 2015.

_____. **Extended Tracking.** [S.l.], 2015c. Disponível em: <https://developer.vuforia.com/library/articles/Training/Extended-Tracking>. Acesso em: 20 out. 2015.

_____. **Frame Markers.** [S.l.], 2015d. Disponível em: <https://developer.vuforia.com/library/articles/Training/Frame-Markers-Guide>. Acesso em: 30 out. 2015.

_____. **How To Install the Vuforia Unity Extension.** [S.l.], 2015e. Disponível em: <https://developer.vuforia.com/library/articles/Solution/Installing-the-Unity-Extension>. Acesso em: 28 mar. 2015.

_____. **How To Setup a Simple Unity Project.** [S.l.], 2015f. Disponível em: <https://developer.vuforia.com/library//articles/Solution/Compiling-a-Simple-Unity-Project>. Acesso em: 20 out. 2015.

_____. **How To Use the Vuforia Object Recognition Unity Sample.** [S.l.], 2015g. Disponível em: <https://developer.vuforia.com/library//articles/Solution/How-To-Use-the-Vuforia-Object-Recognition-Sample>. Acesso em: 20 out. 2015.

_____. **Image Targets.** [S.l.], 2015h. Disponível em: <https://developer.vuforia.com/library/articles/Training/Image-Target-Guide>. Acesso em: 20 out. 2015.

_____. **What your app can see.** [S.l.], 2015i. Disponível em: <https://www.qualcomm.com/products/vuforia/features>. Acesso em: 23 mar. 2015.

SILVA, Marco. **O que é interatividade**. [S.l.], 2001. Disponível em: <http://www.senac.br/informativo/bts/242/boltec242d.htm>. Acessado em: 05 abr. 2015.

UNITY. **A melhor plataforma de desenvolvimento para criar jogos**. [S.l.], 2015a. Disponível em: <https://unity3d.com/pt/unity>. Acesso em: 18 mar. 2015.

_____. **Creating and Using Scripts.** [S.l.], 2015b. Disponível em: <http://docs.unity3d.com/Manual/CreatingAndUsingScripts.html>. Acesso em: 12 out. 2015.

_____. **Creating Scenes.** [S.l.], 2015c. Disponível em: <http://docs.unity3d.com/Manual/CreatingScenes.html>. Acesso em: 12 out. 2015.

_____. **Event System.** [S.l.], 2015d. Disponível em:

< http://docs.unity3d.com/Manual/EventSystem.html>. Acesso em: 23 out. 2015.

_____. **GameObjects.** [S.l.], 2015e. Disponível em: <http://docs.unity3d.com/Manual/GameObjects.html>. Acesso em: 12 out. 2015.

_____. **Game View.** [S.l.], 2015f. Disponível em: <http://docs.unity3d.com/Manual/GameView.html>. Acesso em: 13 out. 2015.

_____. **GUI Basics (Legacy).** [S.l.], 2015g. Disponível em: <http://docs.unity3d.com/Manual/gui-Basics.html>. Acesso em: 17 out. 2015.

_____. **Hierarchy.** [S.l.], 2015h. Disponível em: <http://docs.unity3d.com/Manual/Hierarchy.html>. Acesso em: 13 out. 2015.

_____. **Inspector.** [S.l.], 2015i. Disponível em: <http://docs.unity3d.com/Manual/Inspector.html>. Acesso em: 13 out. 2015.

_____. **O software líder global da indústria de jogos.** [S.l.], 2015j. Disponível em: <https://unity3d.com/pt/public-relations>. Acesso em: 18 mar. 2015.

_____. **Prefabs.** [S.l.], 2015k. Disponível em: <http://docs.unity3d.com/Manual/Prefabs.html>. Acesso em: 12 out. 2015.

_____. **Project Browser.** [S.l.], 2015l. Disponível em: <http://docs.unity3d.com/Manual/ProjectView.html>. Acesso em: 13 out. 2015.

_____. **Rigidbody.** [S.l.], 2015m. Disponível em: <http://docs.unity3d.com/Manual/class-Rigidbody.html>. Acesso em: 14 out. 2015.

_____. **Scene View.** [S.l.], 2015n. Disponível em: <http://docs.unity3d.com/Manual/SceneView.html>. Acesso em: 13 out. 2015.

_____. **Um editor repleto de recursos e altamente flexível.** [S.l.], 2015o. Disponível em: <http://unity3d.com/pt/unity/editor>. Acesso em: 18 mar. 2015.

APÊNDICE A – APRESENTAÇÃO DO APLICATIVO NO XI SEMINÁRIO INTEGRADO DAS LICENCIATURAS

No sábado, dia 07/11/2015 houve o evento XI Seminário Integrado das Licenciaturas na Fundação Universidade Regional de Blumenau que contou com a exposição de trabalhos desenvolvidos por estudantes. Neste evento, foi apresentado o aplicativo desenvolvido neste trabalho para o público que teve interesse no assunto (Realidade Aumentada). Na Figura 34 é visualizada a demonstração do aplicativo para uma das pessoas presentes no evento que se interessou pelo assunto. Na demonstração do aplicativo deste trabalho, ressaltou-se a sua possível utilização para o ensino sobre educação no trânsito, pois o cenário desenvolvido é para o trânsito, com a utilização de Realidade Aumentada e objetos interativos que proporcionaram um maior atrativo ao aplicativo. Para a apresentação, foi realizado um melhoramento no mapa (marcador) criado por este trabalho. Para a nova versão do mapa, foram criadas faixas de trânsito para a estrada. Estas alterações foram desenvolvidas pelo aluno João Paulo Serôdio Gonçalves do projeto Prodocência (financiamento CAPES). A Figura 35 apresenta o mapa utilizado.

Figura 34 - Demonstração do aplicativo deste trabalho

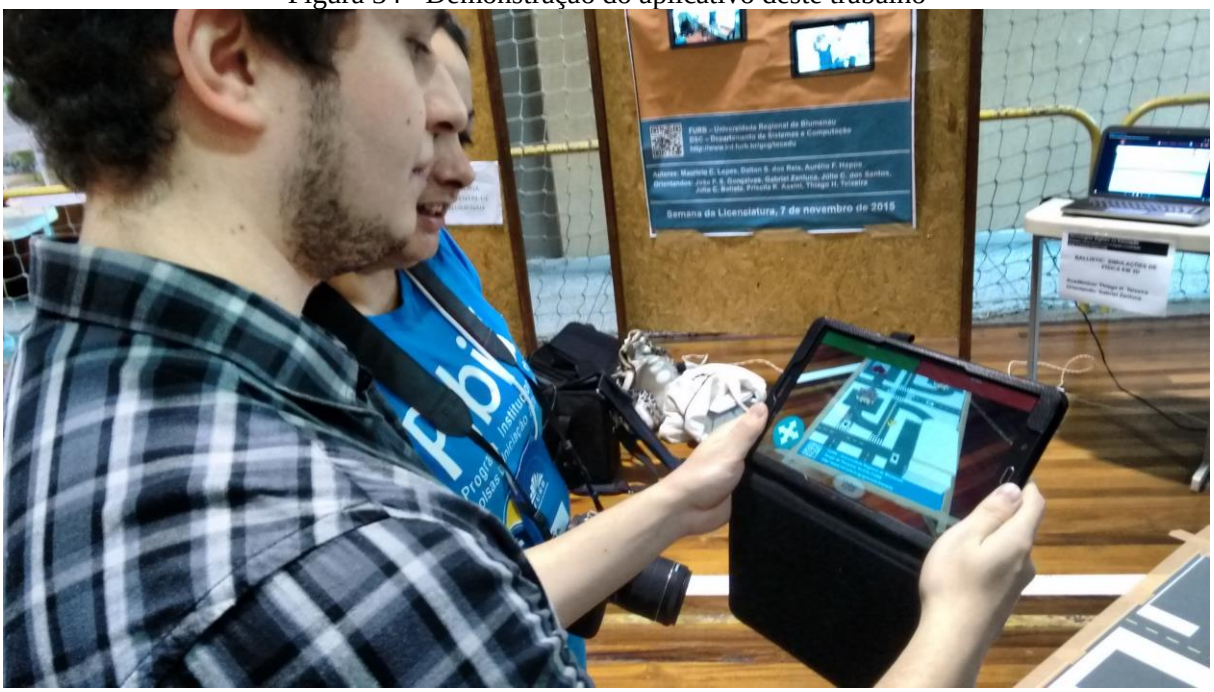


Figura 35 - Marcador utilizado para a exposição