



Católica  
do Tocantins

**FACULDADE CATÓLICA DO TOCANTINS**

**CURSO SISTEMAS DE INFORMAÇÃO**

Eryck Kaique Nascimento Limeiro

**UTILIZAÇÃO DO MOTOR DE JOGO UNITY PARA DESENVOLVIMENTO DE  
APLICAÇÃO COM REALIDADE VIRTUAL PARA DISPOSITIVOS MÓVEIS  
ANDROID DE PARTE DO IMÓVEL DO CAMPUS I DA FACULDADE  
CATÓLICA DO TOCANTINS**

Palmas – TO

2018



**Católica  
do Tocantins**

**FACULDADE CATÓLICA DO TOCANTINS**

**CURSO SISTEMAS DE INFORMAÇÃO**

**Eryck Kaique Nascimento Limeiro**

**UTILIZAÇÃO DO MOTOR DE JOGO UNITY PARA DESENVOLVIMENTO DE  
APLICAÇÃO COM REALIDADE VIRTUAL PARA DISPOSITIVOS MÓVEIS  
ANDROID DE PARTE DO IMÓVEL DO CAMPUS I DA FACULDADE  
CATÓLICA DO TOCANTINS**

Trabalho apresentado como requisito para aprovação na disciplina de Trabalho de Conclusão de Curso da Faculdade Católica do Tocantins (FACTO), sob a orientação do professor Me. Silvano Maneck Malfatti.

Palmas – TO

2018

**INFORMAÇÕES DO ACADÊMICO:**

Nome: Eryck Kaique Nascimento Limeiro

Matrícula: FC20120948

E-mail: eryckknl@hotmail.com

**INFORMAÇÕES ACADÊMICAS**

Professor Orientador: SILVANO Malfatti

Área de realização do TCC: Sistemas de Informação

Data: \_\_\_\_/\_\_\_\_/\_\_\_\_

Assinatura do Aluno

**ACEITE DO ORIENTADOR:**

Observações:

Data: \_\_\_\_/\_\_\_\_/\_\_\_\_

Assinatura do Orientador

**PARECER DO COORDENADOR DO TCC:**

Observações:

Data: \_\_\_\_/\_\_\_\_/\_\_\_\_

Coord. Do TCC

## DEDICATÓRIA

*Dedico este trabalho a minha esposa Bruna Neves Lima, a meus pais Maria do Espírito Santo e Waldeck Francisco, e a todos os apaixonados por jogos eletrônicos.*

## **AGRADECIMENTOS**

Esse projeto marca o fim de uma jornada, que abrirá novas portas para novos desafios. Durante essa longa jornada, muitos colegas e amigos foram feitos, e a eles tenho muito o que agradecer, pelo apoio e ajuda para seguir em frente vencendo os desafios.

Em especial agradeço a minha esposa por sempre me apoiar em minhas decisões e jornadas, a minha família, a meus velhos amigos Heliney Resende e Thiago Mamede, e aos meus caros colegas/amigos conquistados durante esses anos de curso, que me ajudaram muito nessa consolidação: Thiago Aquino Lacerda, Ranieu de Sousa da Silva, Eleilson Carneiro Lima, Sérgio Jach Júnior, Júlio Hortêncio M. Nepomuceno e a todos os outros que participaram comigo dessa conquista, deixo aqui meu muito obrigado.

## RESUMO

Este trabalho desenvolveu-se em três etapas. Iniciando com o estudo de motores de jogos (*Game Engine*) para desenvolvimento de jogos e aplicações gráficas 2D e 3D com suporte a plataforma móvel Android. Analisando a interface, usabilidade, capacidade de desenvolvimento e desempenho dos motores. Na etapa seguinte realizou-se o estudo de caso nesses motores de jogos, realizando o processo de desenvolvimento de uma aplicação para dispositivos móveis Android em cada motor. Culminando na seleção do motor de jogo para o desenvolvimento da etapa final, a utilização do motor de jogo UNITY para desenvolvimento de aplicação com realidade virtual de parte do Campus I da Faculdade Católica do Tocantins para dispositivos Android. Os motores de jogos analisados foram: GODOT, UNITY e Unreal Engine. Com a seleção do motor de jogos UNITY, desenvolveu-se esse trabalho que teve como objetivo a virtualização em 3D de parte do imóvel do Campus I da Faculdade Católica do Tocantins. Esse desenvolvimento terá como finalidade um aplicativo para auxílio à localização dentro do campus e apresentação do ambiente. Para esse desenvolvimento além do motor de jogo UNITY utilizou-se o modelador 3D BLENDER para criação dos modelos 3D que compõe o projeto.

**Palavras Chave:** Android, Games, Motor de Jogo Gráfico, Modelagem 3D.

## **ABSTRACT**

This work was developed in three stages. Beginning with the study of game engines (Game Engine) to develop games and graphic applications 2D and 3D with support Android platform. Analyzing the interface, usability, development capability and performance of the engines. In the next step the case study was carried out on these game engines, carrying out the process of developing an application for Android mobile devices in each engine. Culminating in the selection of the game engine for the development of the final stage, the use of the UNITY game engine for application development with virtual reality of part of the Campus I of the Catholic University of Tocantins for Android devices. The game engines analyzed were: GODOT, UNITY and Unreal Engine. With the selection of the UNITY gaming engine, this work was developed with the objective of 3D virtualization of part of the Campus I building of the Catholic University of Tocantins. This development will aim at an application to aid in on-campus location and presentation of the environment. For this development in addition to the game engine UNITY was used 3D modeler BLENDER to create the 3D models that make up the project

**Keywords:** Android, Games, Game Engine, Modeling 3D.

## INDICE DE FIGURAS

Figura 1 Tela Inicial GODOT .....	18
Figura 2 Tela Inicial de Projetos GODOT configurada em 3D .....	19
Figura 3 Tela Inicial UNITY .....	21
Figura 4 Tela Inicial de Desenvolvimento UNITY .....	21
Figura 5 Epic Games Laucher .....	24
Figura 6 Criação de Projeto Unreal Engine .....	24
Figura 7 Tela de Desenvolvimento Unreal Engine .....	25
Figura 8 BluePrint Unreal Engine .....	26
Figura 9 Get To Teddy .....	29
Figura 10 Towns Of The Dead .....	29
Figura 11 Adventure of Poco Eco - Lost Sounds.....	30
Figura 12 Hitman GO .....	30
Figura 13 Bad Piggies .....	31
Figura 14 Epic Citadel .....	31
Figura 15 Unreal Engine 4 Demo .....	32
Figura 16 Projeto Desenvolvido na GODOT .....	37
Figura 17 Jogo Desenvolvido na UNITY .....	41
Figura 18 Modelagem 3D do Projeto Católica.....	49
Figura 19 Modelagem de Superfícies duras em Polígonos.....	49
Figura 20 Textura Encaixável Das Parede Externas.....	50
Figura 21 3D e Mapa UV da Caixa de Incêndio .....	50
Figura 22 Diretório de Arquivos do Projeto .....	52
Figura 23 Ponto de partida para modelagem .....	53
Figura 24 Hierarquia de objetos do térreo .....	54
Figura 25 Objetos.Terreo base para replicação do Cenário.....	55
Figura 26 Modelos 3D usados no projeto.....	56
Figura 27 Estrutura de componentes da porta animada .....	57
Figura 28 Animação de abertura para porta de incêndio .....	58
Figura 29 Interação entre as animações no Animator .....	58
Figura 30 Estrutura de Canvas e Painel para interação em dispositivos moveis. .....	60
Figura 31 Rotas de Pathfinding .....	63
Figura 32 Menu de Pausa do modo Guia.....	65



Figura 33 Menu Principal.....	67
Figura 34 Build Settings e Player Settings .....	68
Figura 35 Projeto finalizado.....	69

## INDICE DE TABELAS

Tabela 1 Motores de Jogos.....	16
Tabela 2 Motores de Jogos Usabilidade e Interface .....	28
Tabela 3 Configuração da Máquina Utilizada.....	32
Tabela 4 Consumo de recursos com projeto aberto.....	33
Tabela 5 Consumo de recursos com processos que exigem desempenho .....	33
Tabela 6 Análise dos Motores de Jogos .....	44
Tabela 7 Medições do ambiente. ....	46

## LISTA DE ABREVIATURAS E SIGLAS

**2D** – Bidimensional

**3D** – Tridimensional

**VR** – Virtual Reality

**AR** – *Augmented Reality*

**PC** – *Personal Computer*

**DX9** – *Direct X 9*

**DX11** – *Direct X 11*

**APK** – *Android Application Pack*

**API** – *Application Programming Interface*

**SDK** – *Software Development Kit*

**NDK** – *Native Development Kit*

**M** – Metros

**UI** – *User Interface*

**IA** – Inteligência Artificial

**FBX** – *Filmbox*

## SUMÁRIO

<b>1. INTRODUÇÃO</b>	14
1.1 OBJETIVOS	15
1.1.1 Geral	15
1.1.2 Específicos	16
<b>2. METODOLOGIA</b>	16
2.1 ESTUDO DOS MOTORES DE JOGOS	17
2.1.1 Godot	17
2.1.2 Unity	19
2.1.3 Unreal Engine	22
2.2 ANÁLISE DAS INTERFACES E USABILIDADES	26
2.3 CAPACIDADE DE CRIAÇÃO DOS MOTORES DE JOGOS	28
2.4 DESEMPENHO	32
2.5 PRINCIPAIS CARACTERÍSTICAS DOS MOTORES DE JOGOS ESTUDADOS	34
2.6 DESENVOLVIMENTO NOS MOTORES DE JOGOS	34
2.6.1 Desenvolvimento em Godot	34
2.6.2 Desenvolvimento em Unity	37
2.6.3 Desenvolvimento em Unreal Engine	42
2.7 SELEÇÃO DO MOTOR DE JOGO ( <i>GAME ENGINE</i> )	42
2.8 MODELAGEM E DESENVOLVIMENTO DO PROJETO CATÓLICA	45
2.8.1 Modelagem Tridimensional	45
2.8.2 Texturas	46
2.8.3 Mapa UV	47
2.8.4 Criação e desenvolvimento do projeto em UNITY	47
<b>3. MODELAGEM DOS OBJETOS 3D DO CAMPUS</b>	48
3.1 MODELAGEM	48

3.2 TEXTURIZAÇÃO.....	50
<b>4. DESENVOLVIMENTO DO PROJETO EM UNITY.....</b>	<b>51</b>
4.1 CRIAÇÃO E CONFIGURAÇÃO DO PROJETO .....	51
4.2 CONSTRUÇÃO DO AMBIENTE VIRTUAL .....	52
4.2.1 <b>Correções e novas modelagens para construção do cenário.....</b>	<b>55</b>
4.3 ILUMINAÇÃO .....	56
4.4 ANIMAÇÃO .....	56
4.4 CRIAÇÃO DOS MODOS DE JOGO.....	59
4.4.1 <b>Modo Livre</b> .....	<b>60</b>
4.4.2 <b>Modo Guia</b> .....	<b>62</b>
4.4.3 <b>Menu Principal</b> .....	<b>65</b>
4.5 EXPORTAÇÃO DO PROJETO .....	67
4.6 FINALIZAÇÃO DO DESENVOLVIMENTO DO PROJETO.....	69
<b>5. CONCLUSÃO .....</b>	<b>69</b>
<b>6. REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>72</b>

## 1. INTRODUÇÃO

Ao iniciar os estudos em desenvolvimento de jogos ou aplicações gráficas animadas com interação do usuário, encontram-se os motores de jogos, conhecidos também pelo seu termo em inglês (*Game Engine*). Essas ferramentas são produtos criados por desenvolvedoras de jogos ou *softwares*, que por sua vez os disponibilizam em acesso para o uso público de forma gratuita ou paga, e em alguns casos apenas para uso da própria desenvolvedora como no caso da EA (*Eletronic Arts*) com a *engine* Frostbite que é usada para criação de seus jogos por seus estúdios.

Esses motores possuem a capacidade de criar aplicações gráficas das mais variadas formas, seja para a criação de simples jogos e animações em dimensão 2D, como também para grandes produções em 3D com altos níveis de detalhes gráficos, física e funções.

Um motor de jogo trata-se de um *software* que possui um grupo de bibliotecas que abstraem a programação de baixo nível, possibilitando que haja uma interface e usabilidade mais agradável para o trabalho de desenvolvimento. (LEWIS, 2002). O benefício de uso dessas ferramentas é o reaproveitamento de código, o que gera maior agilidade para o desenvolvimento. (BURDEA, 2003).

Atualmente, o mercado conta com diversas opções de motores de jogos disponíveis para os desenvolvedores, com suporte a 2D, 3D, efeitos de física, efeitos de iluminação, sons, inteligência artificial, loja de *assets*, suporte a multiplataforma, dentre outros recursos, que podem variar a partir do motor que for escolhido.

A escolha certa de um motor de jogo é uma decisão crucial para a criação, desenvolvimento e sucesso de um projeto, definido os requisitos e a plataforma desejada pode-se analisar qual motor atenderá ao projeto. Para isso, neste trabalho foram realizados estudos iniciais em motores de jogos para a seleção de um no qual o projeto foi desenvolvido.

Para seleção dos motores de jogos analisados, foram colocados como requisitos a gratuidade, suporte a plataforma móvel Android e suporte as dimensões 2D e 3D, atendendo a esses requisitos encontrou-se os seguintes

motores de jogos: GODOT, UNITY e Unreal Engine, que após análises de interface, usabilidade, conteúdo disponível para estudo, desempenho e desenvolvimento auxiliaram na seleção do motor de jogo UNITY para o desenvolvimento deste trabalho.

Para criação de ambientes ou objetos tridimensionais, faz-se necessário o uso de programas para modelagem 3D, pois um motor de jogo por si só não foi desenvolvido para criar os objetos que compõe os ambientes de suas aplicações, um motor de jogo apenas importa esses objetos para o projeto, onde dentro do projeto o motor de jogo monta, anima, programa e aplica efeitos de iluminação e física a esses objetos.

A modelagem tridimensional desse projeto utilizou o modelador, animador e renderizador BLENDER que é desenvolvido pela BLENDER Foundation e é disponibilizado de forma gratuita através do seu próprio site. Trata-se de um *software* de código aberto e que possui uma alta capacidade de criação para modelagem, animação e renderização em 3D e 2D, atendendo muito bem as necessidades e propósitos para a modelagem e criação desse projeto.

## 1.1 OBJETIVOS

O campus da Faculdade Católica do Tocantins é um ambiente que dispõe de muitos cômodos, e por vezes é confuso encontrar alguma sala, principalmente para os calouros que não conhecem a estrutura e a lógica de numeração do ambiente, fazendo com que se perca tempo vagando pela instituição em busca das salas, ou como ocorre muito nos inícios dos períodos letivos, alunos entrando ou estando em salas de aula erradas durante as aulas.

### 1.1.1 Geral

O desenvolvimento do presente estudo propõe a virtualização tridimensional da Faculdade Católica do Tocantins, Campus I, bloco I através da UNITY. Almejando a criação de um aplicativo/jogo voltado para dispositivos móveis com sistema operacional Android, com a função de localização de salas onde o usuário através do aplicativo seleciona a sala desejada, e o aplicativo o guia através do ambiente 3D para localização da sala, e um outro modo onde o usuário pode andar livremente dentro do ambiente 3D como em um jogo de perspectiva de primeira pessoa para conhecer o ambiente da instituição.

Diante disso utilizou-se o software de modelagem 3D BLENDER para modelagem do cenário e o motor UNITY para a construção, desenvolvimento e animação de todo ambiente virtual.

### 1.1.2 Específicos

- Modelagem tridimensional de parte do bloco I do campus I da Faculdade Católica do Tocantins.
- Construção do ambiente virtual do campus na UNITY.
- Desenvolver o modo de guia, no qual o usuário indica onde está e para onde deseja ir, e o aplicativo o guia pelo ambiente virtual.
- Desenvolver o modo livre, onde o usuário pode andar livremente pelo ambiente virtualizado do campus.
- Exportar o projeto para a plataforma móvel Android.

## 1. METODOLOGIA

A tabela a seguir apresenta oito motores de jogos dentre os que estão disponíveis para o público no mercado, e os critérios para a seleção dos motores de jogos analisados neste trabalho, e atendendo aos requisitos propostos chegamos as *engines* selecionadas, GODOT, UNITY e Unreal Engine, como pode ser analisado abaixo.




Motor de Jogo	2D	3D	Gratuidade	Suporte Para Android
	SIM	SIM	NÃO	SIM
	NÃO	SIM	NÃO	NÃO
	SIM	SIM	NÃO	SIM
	SIM	SIM	SIM	SIM
	SIM	NÃO	SIM	SIM
	SIM	NÃO	SIM	SIM
	SIM	SIM	SIM	SIM
	SIM	SIM	SIM	SIM

Tabela 1 Motores de Jogos  
Fonte: (AUTOR, 2017)



## 2.1 ESTUDO DOS MOTORES DE JOGOS

O estudo inicial dos motores de jogos trata-se da análise de usabilidade, interface, desempenho, capacidade de desenvolvimento, conteúdo e manutenção dos motores de jogos por parte das desenvolvedoras das três ferramentas estudadas, essas medições serão realizadas de acordo com as definições da qualidade de *software* estabelecida pela Organização Internacional de Normatização (ISO), através da norma ISO/IEC 25010:2011, que define que um *software* de qualidade tem que atender a requisitos como funcionalidade, performance, compatibilidade e usabilidade.

### 2.1.1 Godot

Desenvolvido por Juan Linietzky e Ariel Manzur, o motor de jogo GODOT é o único dos selecionados que não possui versões pagas, sendo o seu uso totalmente gratuito, seu código é aberto e está disponível na plataforma GitHub.

Como descrito em seu site, a GODOT é um motor de jogo com avançado pacote de recursos, multiplataforma, e dimensões 2D e 3D de código aberto. Durante esta pesquisa estava em sua versão 2.1.4 que pode ser instalada em Microsoft Windows de 32 e 64bits, Linux de 32 e 64bits, Linux Server e Apple Mac OS X de 32 e 64bits. Tem como requisito mínimo Microsoft Windows 7 ou superior e suporte a OpenGL 2.1+. A GODOT usa uma linguagem de programação própria chamada GDScript, e um editor de Scripts integrado a *engine*, sua linguagem de programação se afeiçoa a linguagem Python pela simplicidade de codificação.

Nesse estudo foi realizado o *download* da versão para Microsoft Windows de 64 bits, que possui tamanho de 28,3MB, e vem em pacote compactado. A instalação do motor é simples, basta descompactar o pacote e executar o arquivo, como essa ferramenta não é instalável, é recomendado que se crie uma pasta para a mesma e para os projetos que serão criados a partir dele.

A *Engine* possui suporte a diversos idiomas, inclusive o português do Brasil, mas, para esse estudo e universalização de aprendizado e uso, a ferramenta foi utilizada na versão de idioma inglês.

Ao iniciar a GODOT, a primeira tela é a de criação de projetos e projetos recentes, outro aspecto que se apresenta é que ao entrar em execução a

ferramenta inicia uma tela de comando que sempre ficará aberto junto com a *engine*.

A criação de projetos na GODOT é simples e rápida, necessitando apenas configurar o nome e local que o projeto será salvo, feito isso a *engine* é rapidamente carregada. Ao iniciar o projeto é que a ferramenta dá a opção de qual dimensão será usada para o desenvolvimento, 2D ou 3D, o que pode ser configurado facilmente, apenas selecionado a dimensão desejada.

A interface do motor é intuitiva, organizada e parecida com a dos outros motores estudados, onde tem-se a tela desenvolvimento na parte central, tanto para aplicações gráficas quanto para edição dos scripts quando selecionados, nas laterais ficam as opções de diretórios, *assets* e suas configurações, animações e objetos adicionados ao projeto, na parte inferior ficam as saídas para testes, debug e configurações de animação, por fim na parte superior se tem as opções de execução do projeto, loja de *assets* da GODOT, configurações para o projeto e para a GODOT e as opções de dimensão e execução do projeto, que ao ser executado abre uma nova janela.

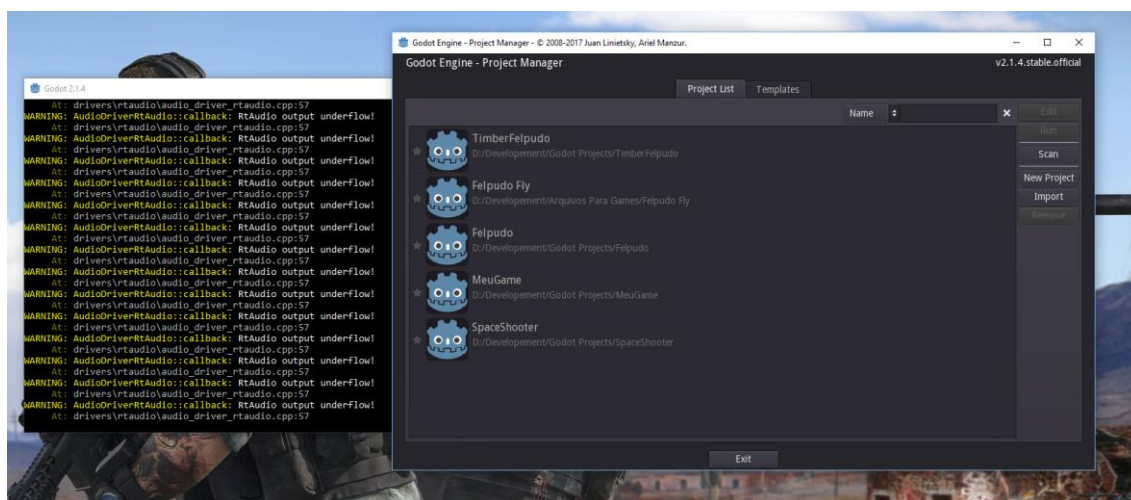
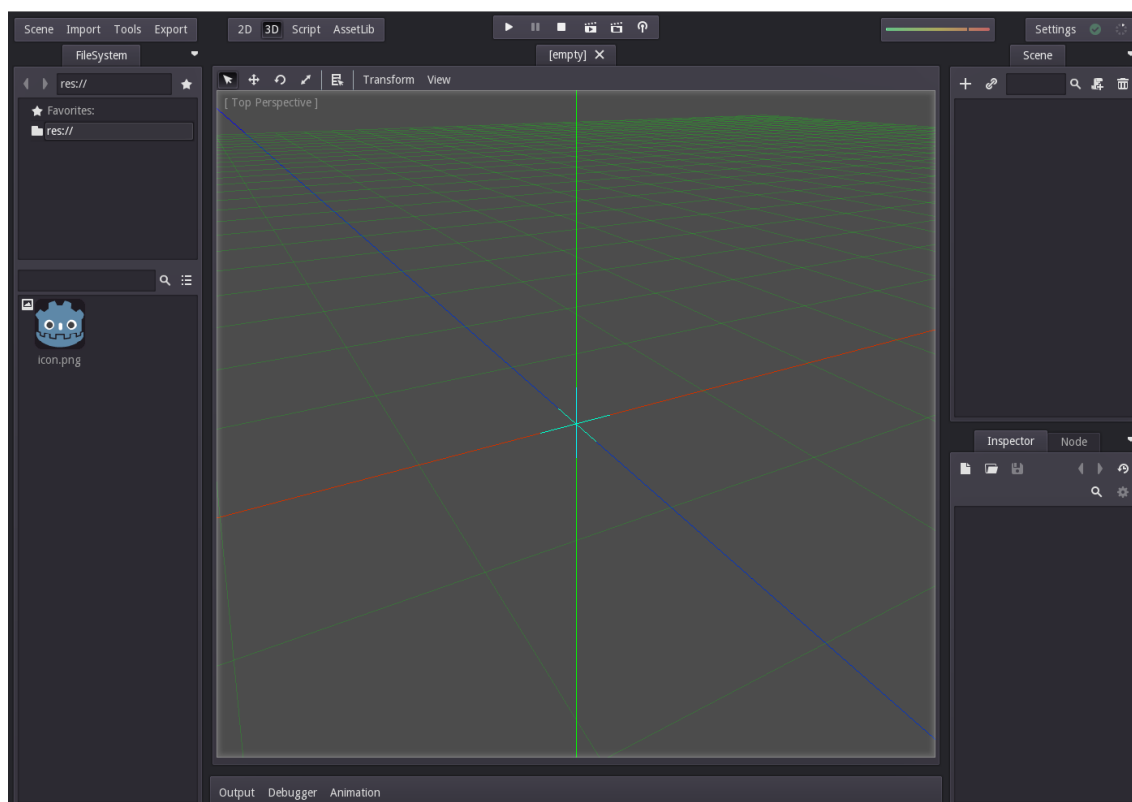


Figura 1 Tela Inicial GODOT  
Fonte: (AUTOR, 2017)



*Figura 2 Tela Inicial de Projetos GODOT configurada em 3D*  
*Fonte: (AUTOR, 2017)*

Além da plataforma desejada nesse estudo a GODOT também exporta seus projetos para as plataformas BlackBerry10, HTML5, Linux X11, Apple Mac OS X, Microsoft Windows Desktop e Microsoft Windows Universal, demonstrando seu poder de multiplataforma.

### 2.1.2 Unity

A UNITY é desenvolvida pela UNITY Technologies, que descreve sua ferramenta em seu site como a *engine* de criação de jogos líder mundial, sendo um motor de jogo poderoso e um editor completo, repleto de recursos e altamente flexível, um centro criativo onde artistas, designers e desenvolvedores trabalham juntos, oferecendo uma plataforma para criação de jogos e aplicativos excepcionais em 2D, 3D, VR e AR.

A UNITY pode ser baixada em seu próprio site após realização de um cadastro, seus requisitos são Microsoft Windows 7 ou superior, Apple Mac OS X 10.9 ou superior, CPU com suporte a instruções SSE2 e placa de vídeo com suporte a DX9 (shader model 3.0) ou DX11 com suporte a *feature level* 3.0. A *engine* possui três planos para *download*: Plano Pessoal, Plus e Pro, o plano

Pessoal é gratuito para iniciantes, estudantes e entusiastas e conta com todos os recursos principais da *engine*. O plano Plus adiciona recursos como relatórios de performance, a escolha de dois pacotes essenciais para alguma plataforma ou formato e outros recursos, esse plano tem um custo de R\$ 72,00 por mês, e tem como requisito da empresa assinante um rendimento anual ou recursos arrecadados de até US \$200.000, se a empresa assinante possuir um rendimento acima disso, deverá assinar o plano Pro, que possui custo mensal de R\$ 360,00, o plano Pro conta com todos os recursos disponíveis no plano Plus, com adição de serviços de nível pro, suporte Premium e planos de acesso aos códigos fontes disponíveis, ressalta-se que os valores foram consultados durante o período de pesquisa deste trabalho em novembro de 2017, podendo ser ajustado a qualquer momento por seus responsáveis

O plano utilizado para este estudo foi o Pessoal, baixado um assistente de instalação no site da *engine* que possui um tamanho de 719Kb, com esse assistente foram selecionados os pacotes de instalação da UNITY na versão 2017.1.0f3, Documentation, Standard Assets, Example Project e Android Build Support, o total desse pacote de instalação foi de 3.1GB para *download*, o tempo de instalação decorre de acordo com a velocidade da internet disponível pelo desenvolvedor.

Ao iniciar a ferramenta pela primeira vez, é necessário realizar autenticação com o mesmo usuário criado no site para o *download*. A tela inicial da UNITY tem opção para criação de projetos, opção para abrir um projeto já criado e opção de aprendizado.

Para criação de projetos é necessário configurar o nome do projeto, o local de armazenamento, recomenda-se criar uma pasta apenas para armazenar os projetos dessa ferramenta, mantendo uma melhor organização dos arquivos dos projetos, e a escolha da dimensão do projeto, definindo se o projeto será em 2D ou 3D, caso selecione uma das dimensões e depois do projeto criado queira alterar a dimensão selecionada, é possível, entretanto é recomendável escolher com cautela a dimensão desejada para sua criação, pois a UNITY já cria seu projeto de acordo com a dimensão selecionada.

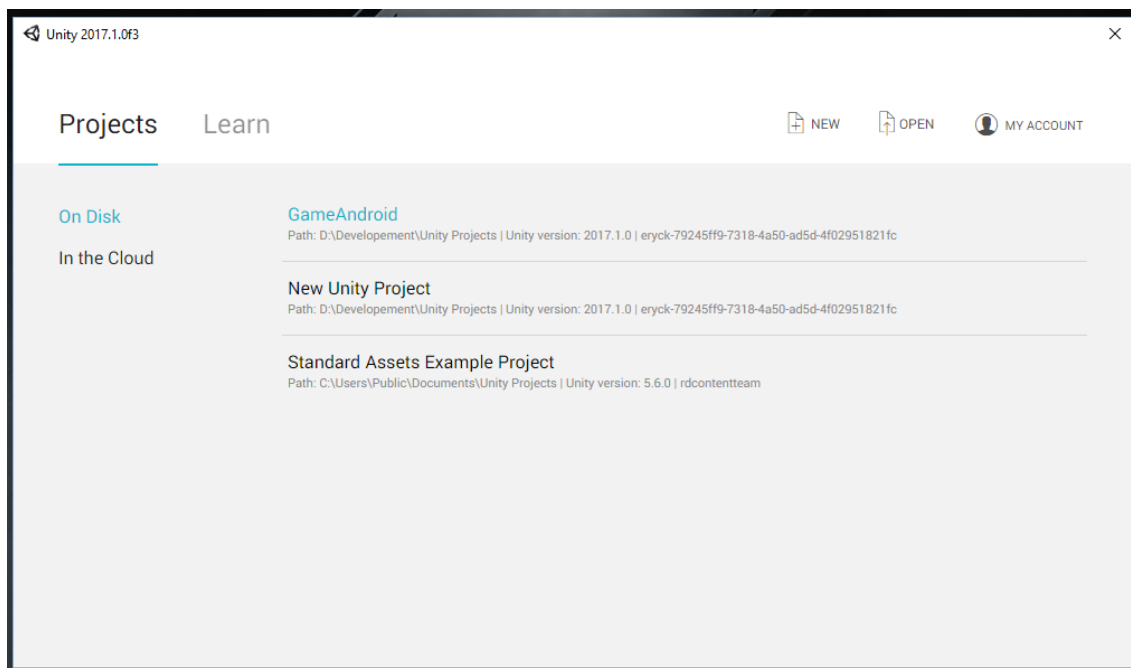


Figura 3 Tela Inicial UNITY  
Fonte: (AUTOR, 2017)

A criação de um projeto é feita de forma rápida, e em questão de segundos a ferramenta está aberta e pronta para o início do desenvolvimento.

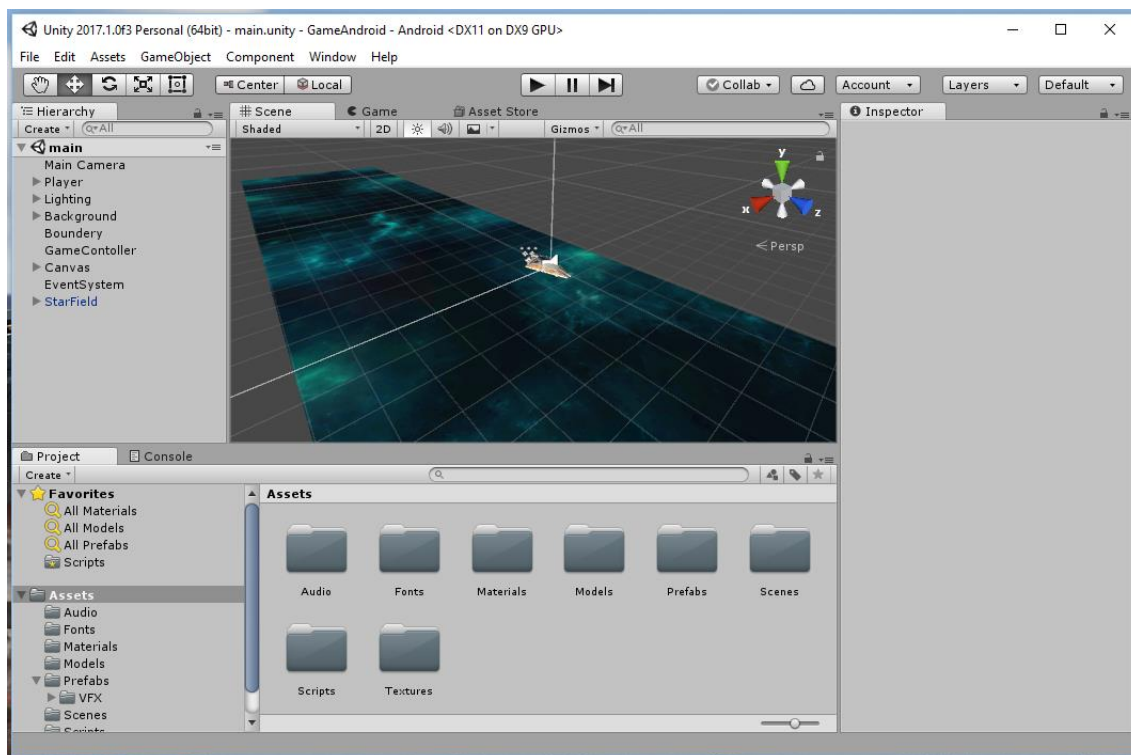


Figura 4 Tela Inicial de Desenvolvimento UNITY  
Fonte: (AUTOR, 2017)

A interface da UNITY também é intuitiva e organizada assim como a ferramenta discutida anteriormente, a GODOT, possuindo suporte a diversos

idiomas, inclusive o português brasileiro. A UNITY tem opções para alteração e configuração de seu layout, porém, a configuração padrão de layout é a observada na figura 4, onde se tem a imagem em destaque no centro da tela, com o diferencial que a UNITY possui uma tela para editar a cena com nome de “Scene” e uma tela para execução e visualização do projeto, com o nome de “Game”. Nas laterais encontra-se a hierarquia de objetos do projeto e o Inspector, onde se configuram as opções dos objetos adicionados entre outras configurações do projeto. Na parte inferior fica a hierarquia de diretórios do projeto e o console de debug, e na parte superior ficam as opções de configurações da UNITY, do projeto, loja de *assets* e execução do projeto. O C# é a linguagem de programação utilizada pela UNITY, a *engine* dá a opção de escolha do editor onde serão criados e editados seus scripts, ao instalar a UNITY em uma máquina com Windows, ela oferece a opção de instalação do Microsoft Visual Studio, contudo, a ferramenta conta com um editor chamado MonoDevelop-UNITY, que é instalado junto com a *engine*.

Em seu site, a UNITY Technologies afirma que sua *engine* é líder no setor de multiplataforma (UNITY PLATAFORMS, 2018), dado suporte às seguintes plataformas: IOS, Android, Tizen, Windows, Universal Windows Platform, Mac, Linux/ Steam OS, WebGL, Playstation 4, Playstation Vita, Xbox One, Wii U, Nintendo 3DS, Oculus Rift, Google Card Board Android & IOS, Steam VR PC & Mac, Playstation VR, Gear VR, Windows Mixed Reality, Daydream, Android TV, Samsung SMART TV, tvOS, Nintendo Switch, FireFox OS, Facebook Gameroom, Apple ARKit, Google ARCode e Vuforia. Destacando que ao criar uma vez o projeto, o mesmo pode ser implementado para as plataformas que desejar para conseguir uma maior audiência.

### 2.1.3 Unreal Engine

A Unreal Engine é desenvolvida pela Epic Games, que descreve em seu site que seu produto é o motor de criação mais poderoso, que ao longo de duas décadas se tornou um motor de confiança e é o mais seguro do mundo, com um conjunto completo de ferramentas de criação concebidas para atender as visões artísticas mais ambiciosas, sendo suficientemente flexível para garantir o sucesso de equipes de todos os tamanhos, com um motor estabelecido como

líder na indústria, oferecendo um desempenho poderoso e comprovado (EPIC GAMES, 2017).

A fabricante também destaca a comunidade da *engine*, descrevendo a mesma como comprometida com a qualidade, onde é possível encontrar profissionais do mais alto calibre a iniciantes que partilham o desejo de criação com excelência (EPIC GAMES, 2017).

A Epic Games disponibiliza a Unreal Engine com todos os seus recursos de forma gratuita, mas se o produto construído com *engine* ultrapassar um rendimento de U\$3.000,00 dólares no seu primeiro trimestre de vendas, é necessário pagar royalty de 5% do rendimento para Epic Games.

Para obter a Unreal Engine é necessário realizar cadastro no site da *engine*, após esse processo é escolhido qual plataforma será instalada, a Unreal Engine tem como requisitos: Microsoft Windows 7 ou superior de 64Bits, MacOS 10.13 ou superior, e tem como requisitos de hardware processador quadre-core de no mínimo 2.5Ghz, 8Gb de RAM, e placa de vídeo com suporte a DirectX 11 no Windows e Metal 1.2 no Mac. A instalação para Windows é feita através do EpicInstaller que possui tamanho para download de 30,5MB, através desse instala-se o Epic Games Launcher, que se trata de uma central com todos os produtos da Epic Games, como seus jogos, versões da Unreal Engine e a loja de *assets* da *engine*.

Através do Epic Games Launcher podemos escolher a versão da Unreal Engine que se deseja instalar, durante essa pesquisa a *engine* chegou a versão 4.18.0, mas nos estudos dessa ferramenta foi usado a versão 4.16.3, pois no desenvolvimento do estudo, foram importados alguns projetos que não eram compatíveis com as versões mais novas. O pacote de instalação da Unreal Engine 4.16.3 após a instalação totalizou 17.4GB.

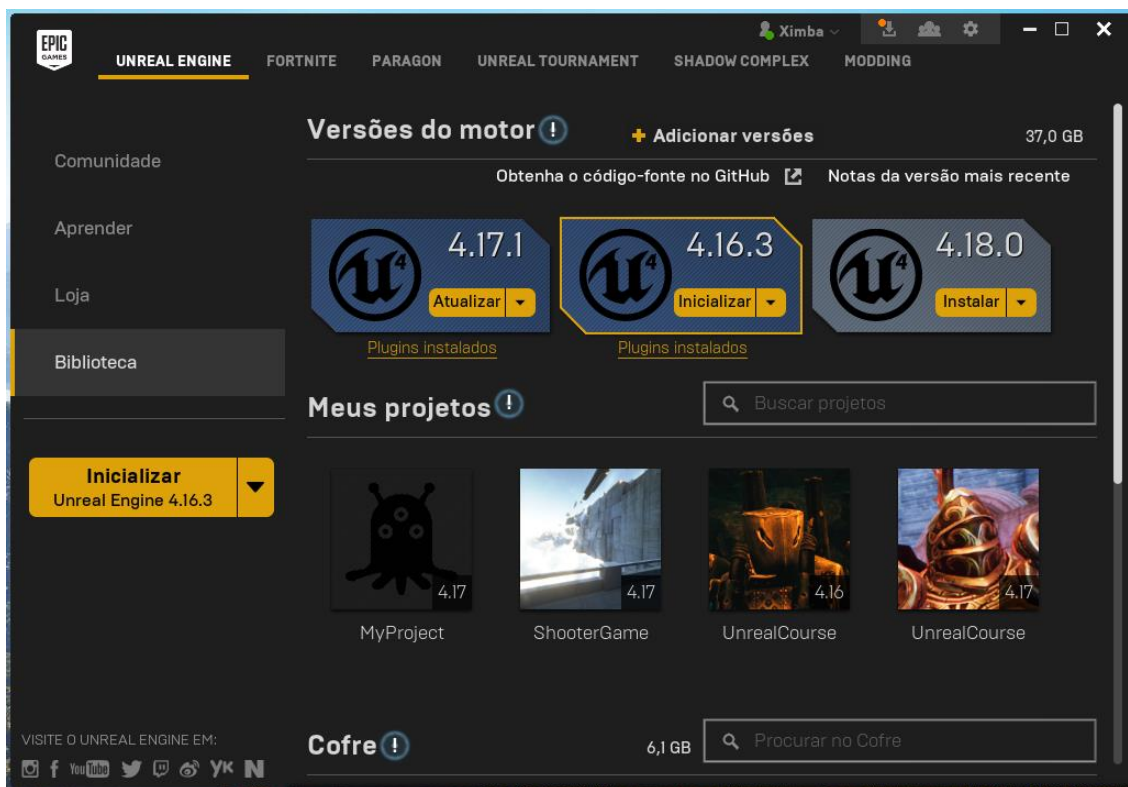


Figura 5 Epic Games Launcher  
Fonte: (AUTOR, 2017)

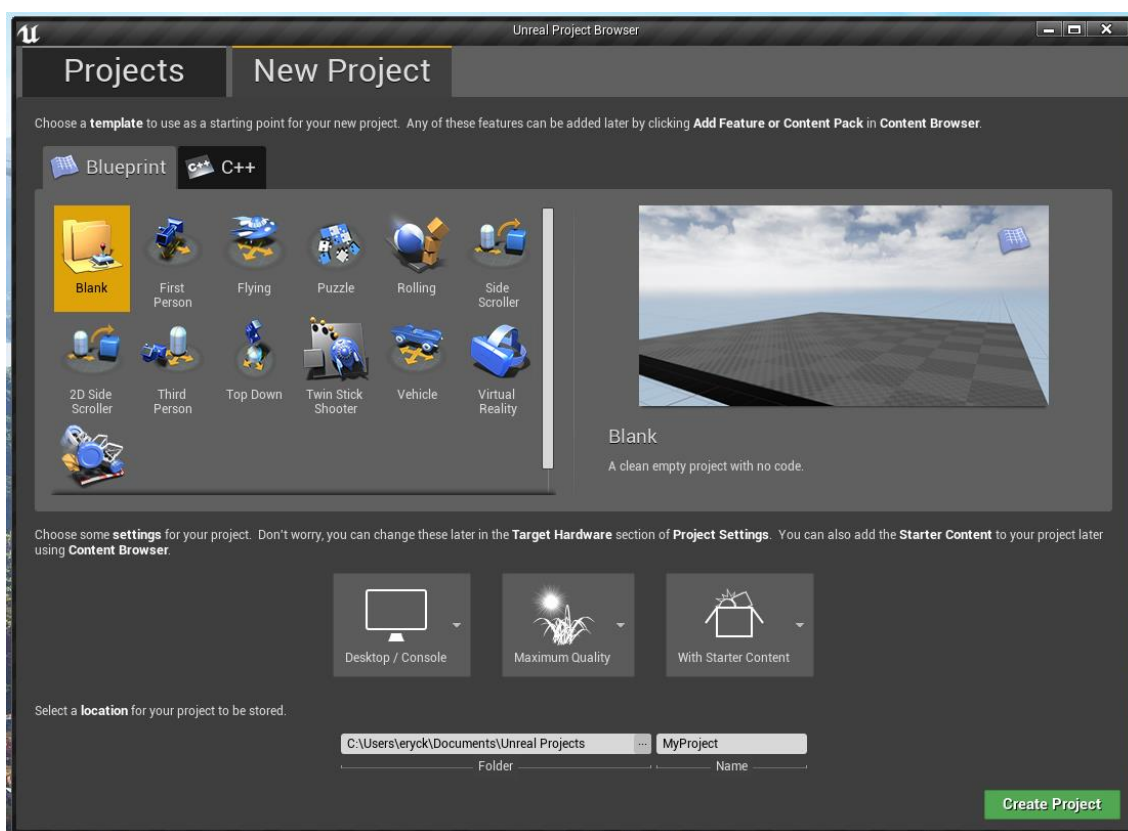


Figura 6 Criação de Projeto Unreal Engine  
Fonte: (AUTOR, 2017)



É através do Epic Games Launcher que se inicializa a *engine*, gerencia e importa projetos e *assets*. A Unreal Engine dentro das estúdios, é o motor que oferece mais recursos iniciais para criação de um projeto, o motor de jogo dá a opção de criar um novo projeto com a linguagem visual Blueprint ou em C++. Com a linguagem selecionada tem-se opções de templates de projetos em branco ou com algumas físicas, iluminações e elementos já criados, como um jogo de tiro em terceira pessoa, um jogo de corrida, um jogo de nave entre outras opções que podem ser vistas na Figura 6.

Na criação de um novo projeto também se escolhe para qual plataforma será desenvolvido selecionando entre Desktop/ Console ou Mobile/ Tablet, definindo a qualidade gráfica entre Maximum Quality ou Scalable 3D or 2D, e define com o que o projeto será iniciado, onde tem-se opções entre With Starter Content, que carrega o projeto com um ambiente pré-criado ou No Starter Content que inicia o projeto limpo sem nenhum ambiente. Configura-se também o local de armazenamento do projeto e por fim o nome que o projeto terá.

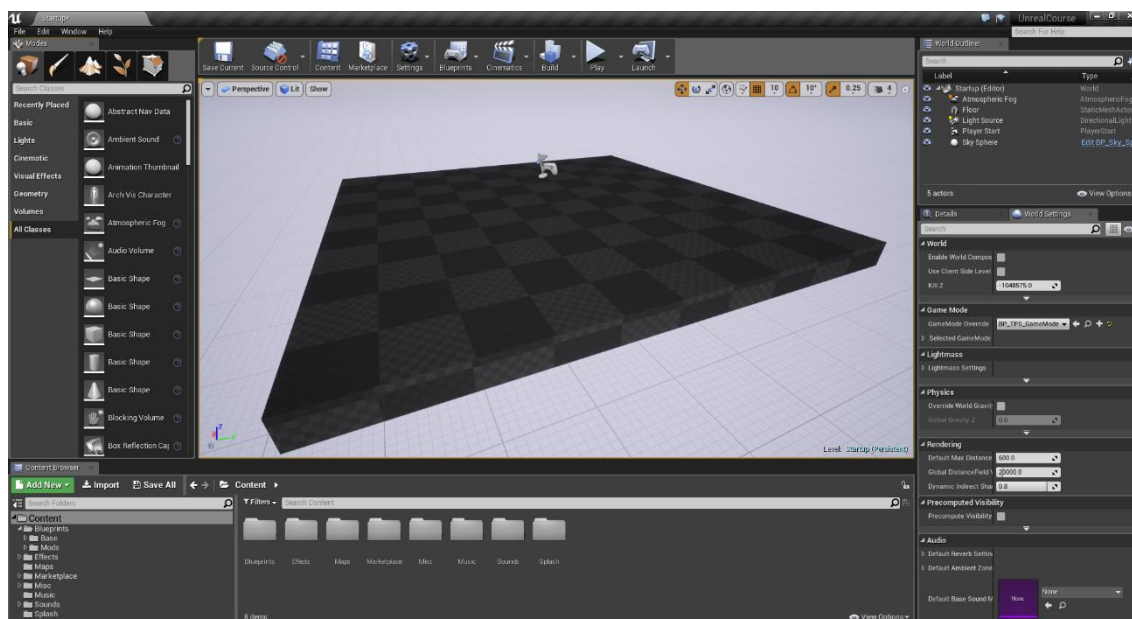


Figura 7 Tela de Desenvolvimento Unreal Engine  
Fonte: (AUTOR, 2017)

Semelhante aos motores de jogos já apresentados, a Unreal Engine segue o padrão com a cena sendo o centro do projeto, como demonstrado na Figura 7, mas seu layout é totalmente customizável, e a *engine* oferece opção para salvar essas personalizações. Nas laterais da Unreal Engine tem-se a opção Mod, a qual disponibiliza vários componentes que podem ser adicionados

ao projeto, como luzes, formas básicas e efeitos visuais, a outra lateral tem as configurações desses componentes ou qualquer outro componente adicionado ao projeto, na parte inferior ficam os diretórios de arquivos, e na parte superior tem-se as configurações da Unreal e projeto, execução de projeto, loja de *assets*. Ao executar o projeto o mesmo é executado no local da cena no *layout* padrão.

O maior diferencial da Unreal Engine, é sua linguagem visual Blueprint para programação dos elementos do projeto, que possui uma interface que se assemelha aos mapas mentais, que por sua vez na Unreal Engine conecta-se e configura-se os nós para criação de elementos de jogabilidade. O Blueprint é extremamente flexível e poderoso, pois oferece aos designers a possibilidade de usar praticamente toda a gama de conceitos e ferramentas em sua maioria disponibilizadas apenas para programadores (EPIC GAMES, 2017).

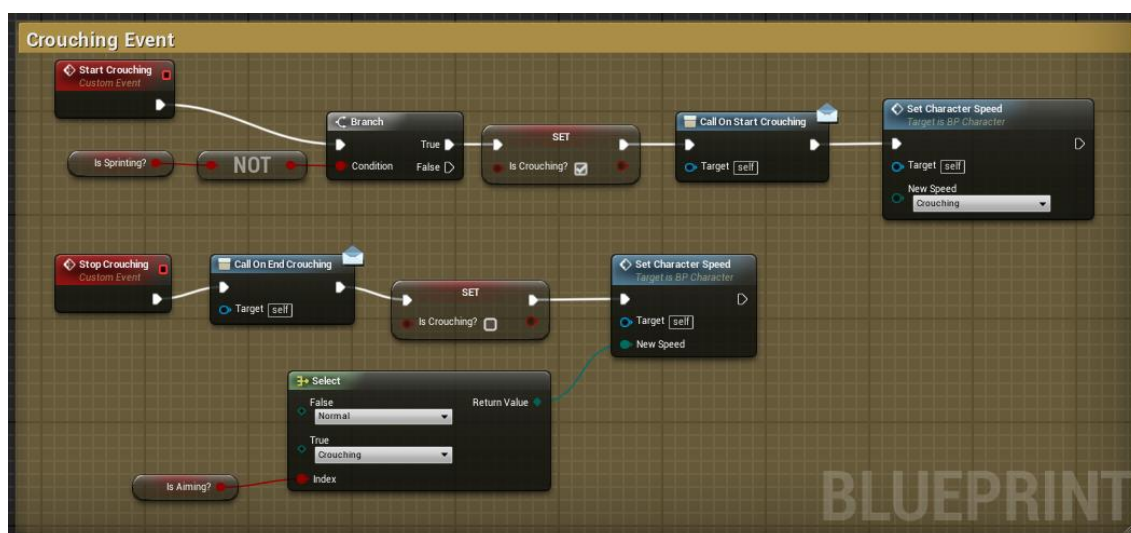


Figura 8 Blueprint Unreal Engine  
Fonte: (AUTOR, 2017)

Em seu site, a Epic Games destaca que a Unreal Engine oferece suporte profundo as plataformas que segundo ela realmente importam, são elas: Playstation 4, Xbox One, Nintendo Switch, Mac OS, Windows, Steam OS, Linux, HTML 5, IOS, Android, Playstation VR, Oculus Rift, Samsung Gear VR, VIVE PORTE e DayDream.

## 1.2 ANÁLISE DAS INTERFACES E USABILIDADES

A interface e a usabilidade de um *software* definem se o mesmo em seu desenvolvimento foi pensado em seus usuários, pois a interface é o meio pelo qual o usuário irá manipulá-lo, e uma boa usabilidade traz um aprendizado ditada

a manipulação do *software*, mesmo que o usuário nunca tenha utilizado a ferramenta antes. A interface é o meio por onde se manipula um *software* através de entrada e saída de dados, e uma boa usabilidade é caracterizada pela facilidade de manipulação e aprendizado do mesmo (REVISTABW, 2015), um *software* que atende a esse requisito é considerado uma ferramenta de qualidade.

Como pode ser observado nas Figuras 2, 4 e 7, as interfaces das ferramentas se assemelham, possuindo muitos pontos em comum, tais como a localização da cena na parte central, que é uma característica em muitos dos *softwares* de motores de jogos, a presença da hierarquia de diretórios de arquivos, a hierarquia de objetos da cena do projeto, e o botão de execução do projeto estão presentes nas telas iniciais dos três motores de jogos. Esse aspecto cria familiaridade as ferramentas de desenvolvimento de jogos, tornando-as mais intuitivas, para as pessoas que já trabalharam com algum motor de jogo.

A GODOT e a Unreal Engine possuem um layout com cores escuras, a UNITY vem configurada com um layout mais claro, mas também pode ser alterado para um layout escuro, sendo mais agradável para alguns.

Por serem ferramentas mais avançadas, a UNITY e Unreal Engine, dispõem de opções para modificação de suas interfaces de modo que o usuário pode criar padrões de interface de acordo com suas escolhas e salva-los. Para alterações rápidas quando preferir, essa característica é um excelente ponto de usabilidade dessas *engines*, pois o usuário pode criar padrões que melhor atenda às necessidades do seu projeto.

Em questão de usabilidade, as *engines* foram bem projetadas, disponibilizando as ferramentas que mais serão usadas para desenvolvimento em suas telas iniciais, o que facilita o uso, e aumenta a produtividade do desenvolvedor. Ferramentas que serão usadas uma ou poucas vezes durante o desenvolvimento do projeto ficam em submenus, que podem ser acessados através das opções dos projetos. Os motores dispõem de atalhos para as principais ferramentas e opções, como salvar e executar o projeto.

A navegação dentro de uma cena em construção de um projeto, pode ser um pouco confusa para um iniciante, a navegação é feita de forma diferente em

cada uma das *engines*, onde os botões do *mouse* executam ações diferentes ao navegar e os atalhos do teclado podem variar, sendo padrão apenas o W, A, S e D para movimentação, por sua vez a Unreal Engine aceita o uso de *joysticks* de vídeo game para realizar essas ações, o que para alguns pode melhorar a usabilidade da *engine*.

No geral as *engines* atendem as características de interface com boa usabilidade, as ferramentas são bem intuitivas para o usuário no quesito de uso geral, suas exceções são algumas configurações complexas que são usadas poucas vezes, onde necessitam de mais atenção do desenvolvedor para o uso, entretanto não é algo que desmereça sua eficácia, com base na experiência de uso, e de acordo com as definições de interface e usabilidade foi criada a tabela a seguir.

Motor de Jogo	Usabilidade	Interface Adaptável	Multilinguagem
<b>GODOT</b>	Boa	Não	Sim
<b>UNITY</b>	Excelente	Sim	Sim
<b>Unreal Engine</b>	Excelente	Sim	Sim

Tabela 2 Motores de Jogos Usabilidade e Interface  
Fonte: (AUTOR, 2017)

### 2.3 CAPACIDADE DE CRIAÇÃO DOS MOTORES DE JOGOS

As capacidades de criação dos motores podem ser classificadas como iniciante para a GODOT e avançado para as demais.

A GODOT possui poucos recursos de iluminação e ambiente, o que limita a criação de grandes cenários mais trabalhados e com muitos efeitos, alguns dos projetos publicados criados com a *engine* estão disponíveis para acesso no site da GODOT em SHOWCASE, pode-se notar que grande maioria dos projetos desenvolvidos na *engine* são de jogos casuais que segundo (NEVES, D. E., SANTOS, L. G. N. O., SANTANA, R. C., e ISHITANI, L., 2013) são jogos simples, de aprendizado rápido, e com recompensas rápidas.

Mas, alguns projetos merecem destaque pela qualidade, como o game Get Teddy, Figura 9, que apesar de simples e em 2D, foi bem trabalhado e desenvolvido por seus criadores, possuindo bons efeitos de animações e sons. Outro projeto para Android que pode ser encontrado no SHOWCASE da GODOT que demonstra sua capacidade de 3D para jogos *mobile*, o game Towns Of The

Dead, Figura 11, que demonstra mais efeitos de física, como tiros e movimentação que podem ser utilizado na *engine*.



Figura 9 Get To Teddy  
Fonte: (GOSOT SHOWCASE, 2017)

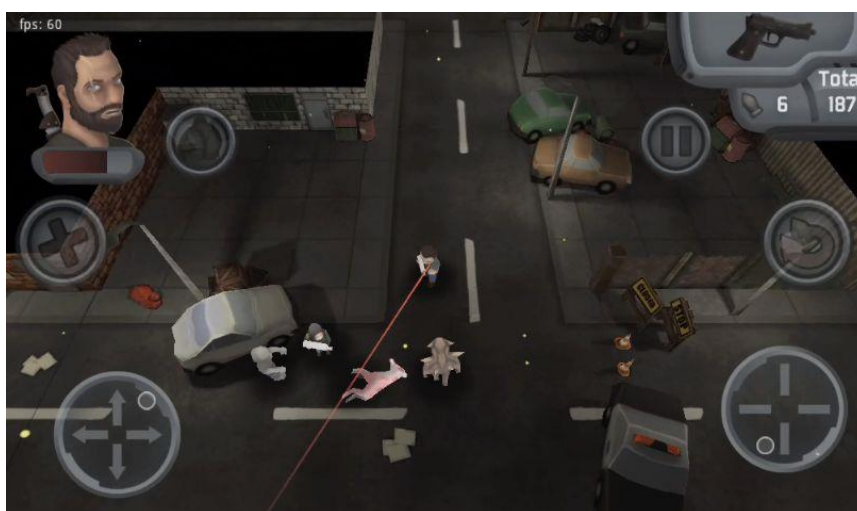


Figura 10 Towns Of The Dead  
Fonte: (GOSOT SHOWCASE, 2017)

A UNITY possui alto poder de criação de cenários, com excelentes efeitos de iluminação, animação, partículas, sons e físicas, a maioria dos grandes jogos desenvolvidos para Android são criados na UNITY como pode ser visto em sua página de Showcase, onde pode-se encontrar de jogos simples em 2D, a jogos



de alto padrão e de grandes franquias e desenvolvedoras como a Rovio e Square Enix.

Com a UNITY, a capacidade de criação de uma aplicação não fica mais limitada ao motor de jogo, e sim a capacidade de criação do desenvolvedor, pois o motor fornece um alto padrão de ferramentas para construção do projeto, como pode ser visto nas figuras 11, 12 e 13.



Figura 11 Adventure of Poco Eco - Lost Sounds  
Fonte: (UNITY SHOWCASE, 2017)



Figura 12 Hitman GO

Fonte: (UNITY SHOWCASE, 2017).



Figura 13 Bad Piggies

Fonte: (UNITY SHOWCASE, 2017).

A capacidade de criação da Unreal Engine, assim como a UNITY é muito alta, trazendo um grande poder gráfico para os dispositivos móveis. Essa *engine* disponibiliza efeitos de iluminação, animação, partículas, sons e físicas para o desenvolvimento do projeto, onde mais uma vez, quem dita a capacidade de criação do projeto é a desenvolvedora, pois a *engine* oferece todo suporte para o desenvolvimento de alta qualidade. Em sua página de Showcase podem ser vistos alguns projetos de jogos *mobile*.

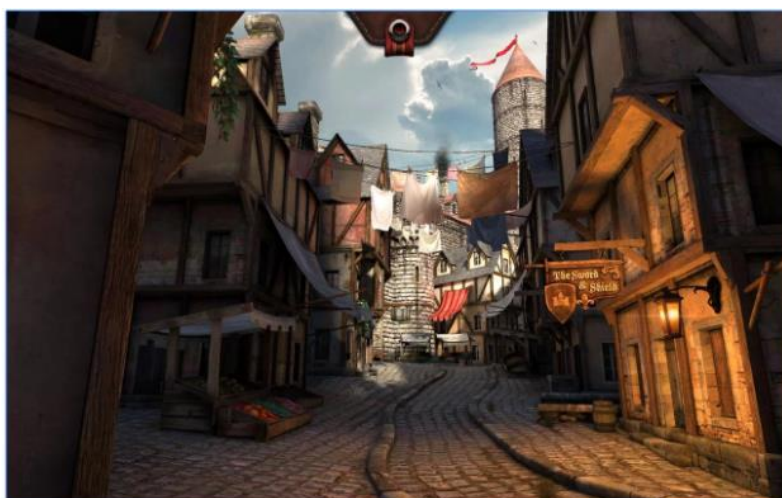


Figura 14 Epic Citadel

Fonte: (EPIC GAMES SHOWCASE, 2017).



Figura 15 Unreal Engine 4 Demo  
Fonte: (EPIC GAMES SHOWCASE, 2017)

## 2.4 DESEMPENHO

Segundo o dicionário Dicio desempenho se define:

*“Modo com alguém ou alguma coisa se comporta tendo em conta sua eficiência, seu rendimento”.*

O computador utilizado para os estudos dos motores de jogos e posteriormente para o desenvolvimento do projeto possui a configuração descrita na tabela 3.

Processador	Memória RAM	Placa Gráfica	Sistema Operacional	Armazenamento
AMD FX 8320e Octal-core 3.2Ghz	16 GB	AMD Radeon RX 480 de 8GB GDDR 5	Windows 10 PRO	120 GB SSD 1.7 TB HDD

Tabela 3 Configuração da Máquina Utilizada  
Fonte: (AUTOR, 2018)

A configuração da máquina atende aos requisitos dos três motores de jogos, requisitos esses que foram apresentados na seção 2.1 desse projeto dentro da apresentação de cada um dos motores gráficos.



Para essa análise, foi coletado dados de consumo da máquina usada para os testes com as *engines* através do gerenciador de tarefas do Windows 10, e registrado o consumo de processamento, consumo de memória e placa gráfica.

A primeira tabela demonstra o consumo de recursos das *engines* com um projeto aberto.

Motor de Jogo	CPU Mínimo	RAM Mínimo	GPU Mínimo
<b>GODOT</b>	0.7%	122.5 MB	1.0%
<b>UNITY</b>	0.4%	316.4 MB	0.3%
<b>UNREAL ENGINE</b>	1.4%	885.5 MB	0.6%

*Tabela 4 Consumo de recursos com projeto aberto  
Fonte: (AUTOR, 2017).*

A segunda tabela, demonstra o consumo máximo que as *engines* atingiram ao realizar atividades que necessitam de mais desempenho, como execução de projetos, exportação de projetos e importação de objetos e componentes para o projeto.

Motor de Jogo	CPU Máximo	RAM Máximo	GPU Máximo
<b>GODOT</b>	16%	190.3 MB	2.8%
<b>UNITY</b>	92.7%	894.7 MB	5.4%
<b>UNREAL ENGINE</b>	99%	1550 MB	28.5%

*Tabela 5 Consumo de recursos com processos que exigem desempenho  
Fonte: (AUTOR, 2017)*

Através dessa medição pode-se constatar que a GODOT é a *engine* que menos consome recursos durante o processo de desenvolvimento, seguida pela UNITY que apresenta um consumo regular de recursos, apresentando uma alta taxa de uso de processamento apenas em tarefas mais complexas como exportação de um projeto. Com a Unreal Engine o consumo pode ser considerado de médio a alto apenas com um projeto aberto, ao realizar atividades como exportação ou importação de grandes projetos, a *engine* chega a consumir quase todo recurso de processamento da máquina.

## 2.5 PRINCIPAIS CARACTERÍSTICAS DOS MOTORES DE JOGOS ESTUDADOS

As principais características da GODOT são sua leveza, simplicidade e intuitividade, possuindo um pequeno pacote de instalação, e exigindo poucos recursos da máquina que a executa, pode ser usada com tranquilidade para criação de projetos de baixa complexidade e média complexidade. A sua linguagem GDScript possui fácil implementação, precisando de poucas linhas de código para criação dos scripts de ações. A *engine* já está com uma nova versão disponível, a GODOT 3.0, que traz o incremento de efeitos de iluminação, físicas e aumento da capacidade de criação do motor.

A UNITY possui como principais características sua alta capacidade de criação e multiplataforma, alinhados a uma ferramenta leve com diversos recursos disponíveis, possui excelentes recursos de iluminação, partículas, programação, inteligência artificial, controles e animação.

Uma das características mais marcantes da Unreal Engine, sem sombra de dúvidas é a sua linguagem visual Blueprint, outra forte característica da *engine* são suas opções para criação de projetos, que oferece vários pré templates com físicas e iluminação aplicados, agilizando o desenvolvimento do projeto.

## 2.6 DESENVOLVIMENTO NOS MOTORES DE JOGOS

Será apresentado neste trabalho o processo de desenvolvimento de uma aplicação para Android em cada um dos motores gráficos selecionados, O Processo de Desenvolvimento de *softwares* é o conjunto de tarefas estabelecidos para criação, teste e conservação de um *software* (Jair C Leite, 2000).

E durante o processo de desenvolvimento serão realizadas as análises propostas para a seleção do motor de jogo que será usado no desenvolvimento da aplicação gráfica em 3D para Android de parte do Campus I da Faculdade Católica do Tocantins.

### 2.6.1 Desenvolvimento em Godot

O processo de desenvolvimento foi iniciado com o motor de jogos GODOT, onde criou-se um projeto de um jogo clone do jogo Flappy Bird, jogo

esse que tem o objetivo de fazer com que um pássaro passe entre canos, e a cada obstáculo vencido soma-se um ponto, a entrada de interação do usuário é apenas o toque, no qual faz com que o pássaro suba um pouco a cada toque.

Para criação desse projeto na GODOT, foi configurado a pasta de armazenamento e nome do projeto, com o projeto criado, configurasse a dimensão, que para esse desenvolvimento foi usado em 2D, outras configurações foram realizadas como a configuração da resolução do projeto, orientação de tela que foi definida como horizontal, ativado a emulação do touchscreen, e criação da ação de toque na tela como entrada de interação do usuário.

Com essas configurações realizadas, foram importados os *assets* do projeto, como imagens e sons, e criado as pastas do diretório para uma melhor organização do projeto, o diretório possui as pastas de *assets* para armazenamento das imagens, *scenes* para armazenar as cenas, *scripts* para armazenamento dos códigos e *sound* para armazenamento dos sons.

As criações dos componentes do projeto são realizadas através dos Nodes das cenas, cada cena possui um nodo principal que recebe os objetos e componentes da cena. O jogo em questão possui duas cenas, a cena principal do jogo foi nomeada como “game”, e seu node como “GameNode” e a segunda cena do game recebeu o nome de “Cano” e seu Node recebeu o mesmo nome “Cano”.

A hierarquia dentro dos Nodes são o que ditam que componentes aparecem primeiro na tela, no caso do Node “GameNode” que é a cena principal do jogo, essa composição ficou da seguinte forma: a animação de fundo, o corpo rígido do pássaro, os campos de colisão para que o pássaro não saia da tela, a referência ao Node “Canos”, um timer que conta 2 segundos para reiniciar o jogo após uma derrota, um terceiro Node, chamado de “Node2D” que controla os componentes para exibição da pontuação, e os sons de, pontuação e batida do pássaro nos canos. O Node “Cano” conta com dois *Sprites* de cano definidos como cano de cima e cano de baixo, uma área 2D para armazenar os campos de colisão para os *sprites* dos canos, e outra área 2D para definir o campo de colisão para gerar a pontuação quando o pássaro passar.

Ao todo foram criados quatro scripts para controle e comunicação das ações do jogo, o script “cano” que define funções como velocidade com que os canos passam, configuração das funções dos campos de colisão do pássaro com os canos, pontuação ao passar entre os canos. O script “felpudo” define as funções do pássaro, como impulso para o pulo do pássaro ao receber o toque, e quanto o pássaro descerá quando não houver toques na tela. O script “gerador” realiza a criação aleatória dos canos na tela, e configura a velocidade, com que os canos se movimentam. O script “game” é o script principal do jogo, nesse script são configuradas funções como definição se o *game* está em execução ou parado, tempo para reinício do jogo, contagem de pontuação, estados através de constantes para definir se o jogo continua executando e se para, esse script pode ser analisado abaixo:

```
extends Node2D
#Node que comanda todo o game

#Referenciando o Pássaro
onready var felpudo = get_node("Felpudo")

#Referenciando o Timer da GameNode
onready var timereplay = get_node("TimerToReplay")

#Referência ao label de pontuação
onready var label = get_node("Node2D/Control/Label")

#Variável para contagem de Pontos
var pontos = 0
#Variável para controle de estado
var estado = 1

#Constante para estado do jogo
const JOGANDO = 1
const PERDENDO = 2

func _ready():
    pass

# Função para matar o game
func kill():
    #Cria um impulso que empurra o pássaro
    #para trás quando bater no cano
    felpudo.apply_impulse(Vector2(0,0), Vector2(-1000,0))
    #parar a animação do fundo
    #referenciando animação de fundo
    get_node("BackAnim").stop()
    # alterando o estado do game
    estado = PERDENDO
    #Iniciar o Timer para reinício do game
    timereplay.start()
    get_node("SomHit").play()
    print("Kill!")
```

```
#função para pontuar
func pontuar():
    pontos += 1
    #str converto em texto
    label.set_text(str(pontos))
    get_node("SomScore").play()

# Quando acaba tempo configurado
# após perder, reinicia o jogo
func _on_TimerToReplay_timeout():
    #Recarrega a o jogo
    get_tree().reload_current_scene()
```

A GODOT possui um editor de scripts incorporado a *engine*, a sua linguagem GDScript que é simples e de fácil aprendizado, o tempo de construção desse projeto foi muito baixo, por se tratar de um jogo simples e em 2D. A exportação do projeto é feita de forma fácil, os passos para exportação de projetos podem ser encontrados no site da *engine*, com a configuração realizada a exportação foi feita de forma rápida, e sem muito esforço da máquina usada.

O projeto desse jogo foi realizado através do curso de Criação de Jogos para Android e IOS, ministrado pelo professor Daniel Ciolfi na plataforma Udemy, abaixo tem-se uma captura de tela do resultado final do projeto desenvolvido durante o estudo.

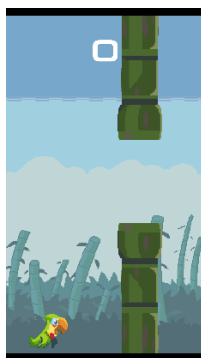


Figura 16 Projeto Desenvolvido na GODOT  
Fonte: (AUTOR, 2017)

## 2.6.2 Desenvolvimento em Unity

Para o processo de desenvolvimento na UNITY, foi usado um projeto de ensino que pode ser encontrado no site da *engine* em inglês na aba aprenda/tutoriais, o projeto Space Shooter tutorial, e que também pode ser encontrado em português e com alguns recursos bem mais explicados na plataforma Udemy, no curso Criando Jogo para Android com UNITY, ministrado

pelo professor Hugo Vasconcelos, o projeto trata-se de um jogo de nave espacial, com efeitos de iluminação, partículas, física, programação e animação.

Para criação do projeto foi configurado o nome, local do projeto e selecionada a dimensão 3D. Apesar de ser criado em 3D o jogo terá uma perspectiva 2D, essa configuração aplica um efeito de profundidade, e permite que os objetos do jogo sejam visualizados em 3D, e que os efeitos de iluminação aplicados criem impressões de sombreamento e direção da luz.

Com o projeto criado, são configuradas as pastas de diretórios de arquivos do projeto, para a importação dos arquivos que serão usados para a criação do jogo. A pasta raiz do projeto é a pasta “Assets” que recebe as pastas: “Audio”, “Fonts”, “Materials”, “Models”, “Scenes”, “Scripts”, “Textures” e “Prefabs” que possui a pasta “VFX” que contém as pastas “Engines”, “Explosions” e “Starfiel”. São importados para o projeto arquivos de áudio, texturas, efeitos, fontes, modelos, e matérias de texturização.

Realizada a configuração dos diretórios é criado na cena principal do jogo, adicionada à iluminação do ambiente e adicionado à câmera que dará a visualização do jogo ao jogador. Após essas configurações é criado o objeto “player” que será a nave principal do jogo controlada pelo jogador. Na nave é configurado um campo de colisão auxiliado por um modelo com uma estrutura 3D mais básica do modelo da nave, essa estrutura faz com que o campo de colisão fique mais preciso e menos complexo para o processamento durante a execução, melhorando o desempenho que usar uma estrutura completa da nave, que possui muitos cantos e profundidades.

Com a nave e com seu campo de colisão criado, é ajustado a posição da câmera em relação à nave e ao fundo para dar uma perspectiva de 2D, a câmera é elevada no eixo Y e posicionada virada para baixo em direção ao eixo X. Realizado esse posicionamento da câmera, é possível criar os efeitos de iluminação sobre a nave, que geram sombreamentos e tons de cores que acompanham o fundo que será aplicado posteriormente.

Após a configuração base da nave realizada, é criado o Script para controle de movimentação da nave, a UNITY usa a linguagem C#, e da liberdade ao desenvolvedor de usar o editor que preferir. Para esse desenvolvimento foi usado o editor da Microsoft, Visual Studio 2017, que se integra muito bem com

a *engine*. O script criado para controle da nave recebeu o nome de “playerController” nesse script foi programado todas as ações da nave, como movimentação, disparos, sons, referência ao corpo rígido e criação de limites na tela para que a nave não ultrapasse os cantos da tela.

Para a configuração dos disparos da nave foi criado o script “weapeonController” que será usado tanto para os disparos da nave do jogador quanto para as naves inimigas, nesse script foi definido ações como movimentação, velocidade de movimentação, sons e tempo entre disparos. Fora do script, os disparos também receberam o campo de colisão e foram adicionados ao “Player” na cena principal.

A animação de fundo é criada como objeto da cena principal “Backgroud”, esse objeto recebe uma imagem que através do script de mesmo nome, é realizada as configurações do efeito de movimentação em paralaxe, que faz com que essa imagem transcorra e volte para o início da tela, no script é configurado a velocidade de movimentação, loop, tamanho, posição inicial e final da imagem de fundo do ambiente, proporcionando um efeito belíssimo de movimentação.

O jogo possui dois tipos de inimigos, os asteroides e as naves inimigas. A criação dos asteroides segue inicialmente o mesmo conceito de criação da nave “Player”, com exceção da estrutura 3D que foi aplicada na nave para configuração do seu campo de colisão. Nos asteroides foi criado um campo de colisão simples em capsula, que na UNITY chama-se “Capsule Colider”, sem usar uma estrutura como feito na nave, esse campo que foi aplicado ao asteroide é regulado para o tamanho do asteroide, fazendo com que o asteroide fique dentro do campo de colisão aplicado. As ações dos asteroides foram configuradas no script principal do jogo, o “GameController”, no script foram realizadas ações para geração randômica dos asteroides e seus posicionamentos, direção de movimentação e velocidade.

Criado os asteroides, configurou-se os efeitos de explosões importados inicialmente e aplicados para a nave e asteroides. Efeitos de som foram configurados para as explosões e disparos da nave.

Nesse ponto do projeto tem-se a base principal construída, com a nave movimentando e atirando, asteroides gerados randomicamente e em posições

aleatórias, efeitos de explosões e sons aplicados e fundo realizando a movimentação em paralaxe.

Começa-se a realizar reaproveitamento de código e funções na criação das naves inimigas, onde aplicasse os mesmos efeitos criados para a nave “Player”, como sons e disparos, o campo de colisão da nave inimiga é criado da mesma forma que os campos usados nos asteroides, outro fator reutilizado dos asteroides é a movimentação, e geração em posições aleatórias com tempos randômicos. As naves inimigas receberam dois incrementos não presentes nos outros elementos, configurou-se o disparo automático a cada dois segundos, e uma movimentação lateral randômica, para aumentar a dificuldade do jogo.

O placar do jogo é contabilizado através de um texto que é incrementado no script principal “GameController”, gerando pontuação maior ao destruir uma nave inimiga e uma pontuação menor ao destruir um asteroide, e posicionado no topo da tela.

O desenvolvimento desse projeto ao ser finalizado conta com nove objetos essenciais dentro da cena principal, e onze scripts de criação e configuração desses objetos. O script de configuração do fundo pode ser analisado a baixo:

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

public class BackgroundScroll : MonoBehaviour {
    //Variável para definir a posição das imagens
    private Vector3 starPosition;
    //Tamanho do objeto
    private float tileSize;
    //variável para aplicar velocidade ao fundo
    public float scrollSpeed;

    // Use this for initialization
    void Start () {
        //Start Position recebe a posição inicial
        starPosition = transform.position;
        //Seleciona o tamanho do eixo Y
        tileSize = transform.localScale.y;
    }

    // Update is called once per frame
    void Update () {
        //captura e realiza o movimento
```



```
//Realiza o loop no movimento do backgroud  
//Suaviza o movimento e transição das imagens  
float newPosition = Mathf.Repeat(Time.time * scrollSpeed, tileSize);  
transform.position = starPosition + Vector3.forward * newPosition;  
    }  
}
```

A experiência de desenvolvimento com a UNITY foi satisfatória. O projeto possui uma complexidade de desenvolvimento considerada alta, pelo fato de ser desenvolvida em 3D, possuir efeitos de iluminação, animação de inimigos, física para as colisões e disparos, contagem de pontuação, e efeitos de partículas. Apesar de ser um jogo de jogabilidade simples, a experiência com esse desenvolvimento trouxe muitos conceitos e técnicas para criação de uma aplicação gráfica. A UNITY durante o desenvolvimento manteve um excelente desempenho, não exigindo muito da máquina usada e sequer longos tempos para importação e criação do projeto, sempre respondendo bem aos comandos e atividades realizadas, o resultado final do projeto pode ser conferido a baixo na Figura 17.

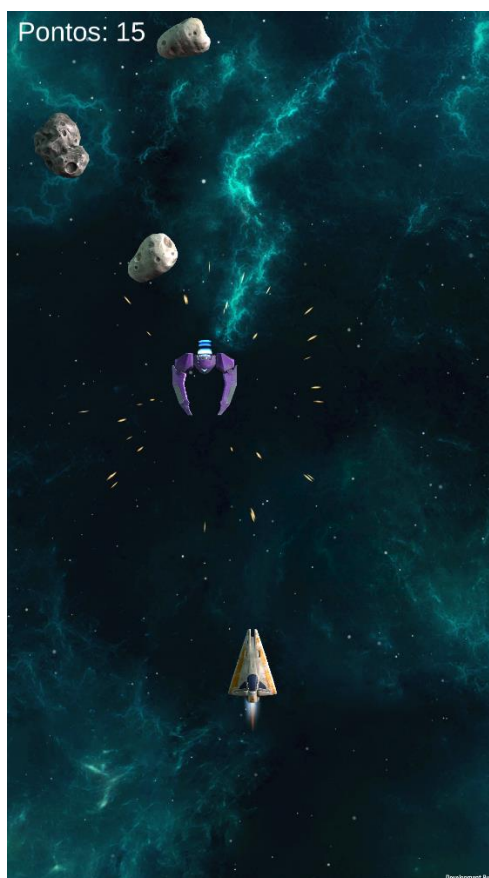


Figura 17 Jogo Desenvolvido na UNITY  
Fonte: (AUTOR, 2017)

### 2.6.3 Desenvolvimento em Unreal Engine

A Unreal Engine é um motor de jogo que produz grandes jogos para console e PC, apesar do suporte as plataformas *mobile*, acha-se até então pouco conteúdo para criação de aplicações específicas para essa plataforma com o motor, devido seu alto poder gráfico, o que torna suas aplicações mais complexas e exigindo mais desempenho, devido esse fato foi iniciado um projeto de desenvolvimento para a plataforma PC, para adquirir experiência de desenvolvimento com o motor.

O desenvolvimento para estudo nessa ferramenta seria através de um curso para criação de um jogo de tiro em primeira pessoa para a plataforma PC. Mas o tempo de instalação da *engine* e importação de projetos para início do desenvolvimento foi além do esperado devido grandes e demorados *downloads*, e tempos para importação dos objetos base do projeto, inviabilizando o desenvolvimento completo nessa *engine*, onde apenas criou-se o projeto, realizado importações de *assets*, físicas e realizado a configuração de alguns movimentos do personagem através dos scripts visuais BluePrint.

O projeto não passou de um mundo aberto com uma plataforma e com um personagem configurado do zero, realizando ações como andar, correr, pular e agachar. Realizou-se também a configuração da iluminação do cenário, e outras configurações dentro do projeto, como ícones e nome do jogo. Devido à demora para o início do desenvolvimento com essa *engine*, o projeto foi parado para dar sequência ao desenvolvimento desse estudo.

## 2.7 SELEÇÃO DO MOTOR DE JOGO (GAME ENGINE)

Este trabalho apresenta três motores gráficos, suas características, desempenho em máquina, a proposta que cada um pode atender e o processo de desenvolvimento completo em duas das três *engines* estudadas.

Tem-se neste estudo um motor de jogo novo no mercado, e dois motores já conceituados, onde cada uma dessas *engines* atendem as necessidades de criação de quem as buscam.

A GODOT oferece a seu usuário leveza e fluidez, em um pacote totalmente gratuito. Quem busca a construção de um jogo ou aplicação gráfica de baixo orçamento, a GODOT oferece boas possibilidades para esse

desenvolvimento, como visto com o jogo Get Teddy Figura 9, mas a *engine* peca pela pouca quantidade de efeitos e físicas, o que pode limitar um projeto que será construído na mesma. Outro ponto negativo da GODOT são os materiais didáticos disponíveis, como se trata de um motor relativamente novo, encontra-se pouco conteúdo disponível para estudo, onde grande maioria está em inglês.

A *engine* GODOT oferece um bom nível de multiplataforma, contendo 7 plataformas para exportação, sendo algumas as principais do mercado como o Android e o PC. A depender do projeto que será desenvolvido nesse motor, o processo de desenvolvimento pode ser muito rápido e descomplicado, da criação a exportação do projeto, o que é um ponto muito positivo da *engine*, em sua nova versão a GODOT 3 a *engine* oferece muitos novos recursos, elevando sua capacidade de criação de jogos.

A UNITY surpreende por seu desempenho apresentando, poder e leveza a seus usuários, é uma *engine* que pode ser usada para desenvolvimento em máquinas não tão robustas, criando projetos de baixa e média complexidade gráfica. Com uma máquina mais robusta disponível para o desenvolvedor, a *engine* oferece todos os recursos para criação de grandes projetos de maneira fluida.

Possui interface modificável de acordo com a preferência do desenvolvedor, podendo ser encontrado muitos conteúdos e cursos específicos para aprendizagem de desenvolvimento e criação na ferramenta, tanto em inglês quanto em português, possui um suporte para multiplataforma incrível, onde um projeto pode ser adaptado e exportado para mais de 25 plataformas.

O desenvolvimento na ferramenta se mostrou ágil, com reaproveitamento de código e funções aplicados aos objetos do jogo. Os efeitos de iluminação e partículas usados no projeto demonstraram o poder gráfico que a *engine* possui. A exportação do projeto para a plataforma Android foi realizado de forma rápida, mas exigindo um pouco mais da máquina usada. O ponto forte desse *engine* é a qualidade oferecida para desenvolvimento e produto final, onde pode-se criar excelentes aplicações sem a necessidade de investimentos altos.

O Unreal Engine não esconde sua capacidade de desenvolvimento, o motor oferece um poder gráfico surpreendente para criação de cenários, com

muitos efeitos de animação, iluminação, partículas, sons e física, mas todo esse poder tem um certo custo para o desenvolvedor, pois sem uma máquina robusta e uma boa internet, a experiência de desenvolvimento pode ser frustrante devido os tempos de download e processamento para importação de projetos. Assim como na UNITY, com a Unreal Engine é possível encontrar muitos conteúdos didáticos e cursos específicos para o motor, em inglês e português, existem muitos sites e comunidades focados no motor, que com uma busca rápida podem ser encontrados, mas grande maioria destina-se ao desenvolvimento de aplicação para plataformas mais robustas, como o PC.

Apesar de uma capacidade multiplataforma inferior a UNITY, a Unreal Engine exporta seus projetos para uma considerável quantidade de plataformas, consideradas pela fabricante como as principais do mercado (EPIC GAMES, 2017).

O desenvolvimento com a Unreal não chegou a ser concluído, onde o projeto chegou a ser iniciado, mas devido o tempo que levou para instalação, realização das configurações e importação de projetos e processamentos, ocasionou atraso no cronograma de desenvolvimento deste trabalho, tendo o projeto encerrado em sua fase inicial para dar continuidade no desenvolvimento desse estudo.

A tabela a seguir demonstra o resultado obtido com cada uma das *engines*, auxiliando na escolha do motor de jogo.

<b>Objetivos</b>	<b>GODOT</b>	<b>UNITY</b>	<b>Unreal Engine</b>
<b>Interface</b>	<b>Boa</b>	<b>Excelente</b>	<b>Excelente</b>
<b>Usabilidade</b>	<b>Boa</b>	<b>Excelente</b>	<b>Excelente</b>
<b>Capacidade Gráfica</b>	<b>Média</b>	<b>Muito Alta</b>	<b>Muito Alta</b>
<b>Desempenho</b>	<b>Excelente</b>	<b>Bom</b>	<b>Ruim</b>
<b>Desenvolvimento</b>	<b>Bom</b>	<b>Bom</b>	<b>Bom</b>

*Tabela 6 Análise dos Motores de Jogos*  
*Fonte: (AUTOR, 2017)*

A *engine* UNITY atendeu a todos os requisitos, entregando um desenvolvimento eficaz e robusto, o produto final do desenvolvimento no motor

foi um jogo completo de nave espacial, com belos efeitos 3D, movimentação, luzes e partículas, demonstrando a capacidade de desenvolvimento e reaproveitamento de código que a *engine* oferece.

Com base na análise apresentada nesse trabalho viu-se que a UNITY apresenta a capacidade, robustez e ferramentas necessárias para o desenvolvimento da virtualização tridimensional de parte do campus I da Faculdade Católica do Tocantins, pois a mesma dispõe de ambiente 3D amplo, efeitos de movimentação, criação de cenas animadas, efeitos de iluminação, programação e inteligência artificial, além de uma excelente interface e desempenho em máquina. Sendo assim a escolhida para o desenvolvimento desse projeto.

## 2.8 MODELAGEM E DESENVOLVIMENTO DO PROJETO CATÓLICA

Nesta seção apresenta-se as metodologias utilizadas no desenvolvimento do projeto católica.

### 2.8.1 Modelagem Tridimensional

A modelagem 3D é a técnica de abstração de um objeto ou elemento tridimensional por meio de *softwares*, simulando esse objeto em um ambiente virtual (LIRA 2007). A modelagem tridimensional pode-se dividir em duas vertentes, a de superfícies duras e a orgânica, na qual as superfícies duras são todos e quaisquer elementos criados pelo homem como construções e automóveis, e a orgânica são elementos naturais como corpos, animais e árvores (LEVINSKI 2017).

Para modelagem utilizou-se a geometria poligonal, que consiste numa série de vértices com posições nos eixos X, Y e Z, que formam vários polígonos caracterizando as superfícies dos objetos modelados (ELIS V. R., BARROSO C. M. R., KIANG C. H, 2004) e (BARROS, 2012).

A modelagem tridimensional dos objetos que compõem a estrutura do imóvel da Faculdade Católica do Tocantins foi realizada através do modelador 3D BLENDER com uso da técnica de modelagem poligonal, para a reprodução fidedigna na modelagem foram realizadas mais de 180 fotos e algumas filmagens de vários pontos do imóvel, assim como uso de uma trena para

medição de pontos estratégicos para reprodução aproximada dos tamanhos dos ambientes.

Objeto	Altura	Largura	Comprimento
<b>Cerâmica Piso</b>	x	0.39M	0.39M
<b>Cerâmica Parede</b>	x	0.39M	0.39M
<b>Parede Geral Interna</b>	2.2M	x	3.3M
<b>Janela Parede Interna Geral</b>	0.74M	x	1.64M
<b>Parede Externa Térreo</b>	2.7M	x	x
<b>Parede Externa Piso 1 e 2</b>	3.27M	x	x
<b>Parede externa Piso 3</b>	1.25M	x	x
<b>Pilar Térreo</b>	1.27M	0.65M	0.65M
<b>Degraus Escadaria</b>	0.16M	2M	0.36M
<b>Janela Externa Piso 3</b>	1.86M	x	1.4M
<b>Parede Corredor banheiro Alta</b>	2.75M	x	8.2M
<b>Parede Corredor banheiro Baixa</b>	1.3M	x	4M
<b>Portal</b>	2.2M	1.4M	x
<b>Portas</b>	2.1M	0.98M	x
<b>Janela Acima Das Portas</b>	0.74M	x	0.94M

*Tabela 7 Medições do ambiente.  
Fonte: (AUTOR, 2018)*

O Ambiente a ser modelado, contém quatro pisos, que vão do térreo ao terceiro piso, três escadarias, um elevador, quarenta e uma salas e oito banheiros. O projeto não tem como objetivo a modelagem dos interiores das salas e banheiros, apenas a modelagem dos corredores e áreas de circulação que levam a estas salas.

### 2.8.2 Texturas

Com a realização da modelagem dos objetos feita é necessário realizar a texturização. Uma textura segundo a documentação do editor de imagem GIMP é uma figura muitas vezes de baixa resolução utilizada para compor a estrutura de objetos tri ou bidimensionais.

Para essa atividade foi utilizado uma câmera fotográfica semiprofissional NIKON COOLPIX P520 para captura de imagens nítidas para criação das texturas, como fotos das tintas das paredes, dos pisos e tintas das portas, para criação e edição das texturas no computador utilizou-se o *software* de edição de imagens GIMP 2 que é gratuito e de código aberto seguindo o propósito do projeto de gratuidade.

Com o GIMP 2 foi utilizada a função de mapeamento que torna encaixável a imagem, transformando-a em uma textura encaixável que segundo a documentação do GIMP é capacidade de uma textura de se unir as demais sem

a geração de emendas, como uma imagem única que preenche a região texturizada.

### 2.8.3 Mapa de UV

Um Mapa de UV segundo a documentação do *software* BLENDER é o processo que pega a malha tridimensional do objeto em 3D nos eixos X, Y e Z e a abre em uma imagem bidimensional plana que para aplicação das texturas ou texturização dessa imagem 2D.

### 2.8.4 Criação e desenvolvimento do projeto em UNITY

Para essa etapa do projeto foi levado em conta à experiência adquirida anteriormente no estudo do motor de jogo, e para aquisição de mais experiência e domínio com o motor UNITY realizou-se mais alguns cursos através da plataforma Udemmy antes de dar início ao processo de modelagem, esses cursos auxiliaram e orientaram a forma de como conduzir o desenvolvimento do projeto.

Os cursos realizados foram os de Desenvolvimento de jogos 3D com Unity 2017 ministrado por Gunnar Correa, Iluminação para Unity 5 – Do Básico ao Avançado ministrado por Vinicius Jorge Munhoz e Unity Descomplicado: Nível 1 ministrado por Glauco Pires, adquirindo mais experiência e domínio para uso do motor de jogo.

No desenvolvendo do projeto Católica foi utilizado a Unity na versão 2017.2.1f1. Recomenda-se que após a inicialização de um projeto não atualizar a engine, pois o uso de versões diferentes pode ocasionar erros ao projeto. A atualização só se faz necessária caso a versão usada esteja apresentando algum erro no projeto e a versão nova traga a correção desse erro. Durante todo o processo de desenvolvendo não foi necessário realizar atualizações.

Para criação dos menus e interface de usuário para os dispositivos moveis, necessitou do componente Canvas, que é componentes de UI “*User Interface*” interface de usuário. Um Canvas segundo descrito pela documentação da UNITY é o local onde adiciona-se todos os componentes de interface de usuário, como botões, menus e painéis.

A realização da navegação automatizada utilizou-se da ferramenta de IA da UNITY Navigation and Pathfinding, Navegação e Encontre o Caminho, esta ferramenta como descrita em sua documentação, é um sistema de navegação

que possibilita gerar personagens que se movem de maneira automatizada e inteligente pelo cenário, através de malhas criadas automaticamente com a ferramenta utilizando a geometria do cenário.

## **2. MODELAGEM DOS OBJETOS 3D DO CAMPUS**

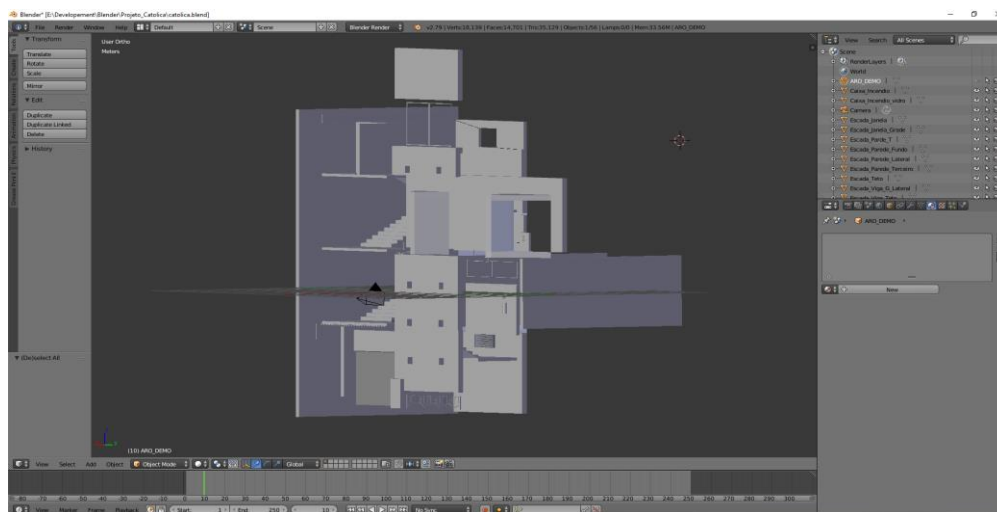
Para dar início ao processo de modelagem foram realizadas medições de partes estratégicas da estrutura do campus, como paredes que mais se repetem, tamanho de portas, portais, janelas, pilares e lajotas do piso e paredes. O processo seguinte foi a realização de captura de imagens em diversos locais a fim de auxiliar a modelagem fidedigna da estrutura de cada um dos andares do bloco, para reprodução dessas estruturas no modelador 3D BLENDER. Buscando mais auxílio para a fidelidade na modelagem, realizou-se algumas pequenas filmagens de locais que continham muitos elementos, como as entradas das escadarias, elevador e partes dos corredores que continham muitas salas.

### **3.1 MODELAGEM**

O processo de modelagem no BLENDER usou como base a modelagem de superfícies duras em polígonos, técnica muito utilizada para criação das mais diversas formas em modelagem 3D, pois com essa técnica obteve-se mais resultado e desempenho para modelagem dos objetos.

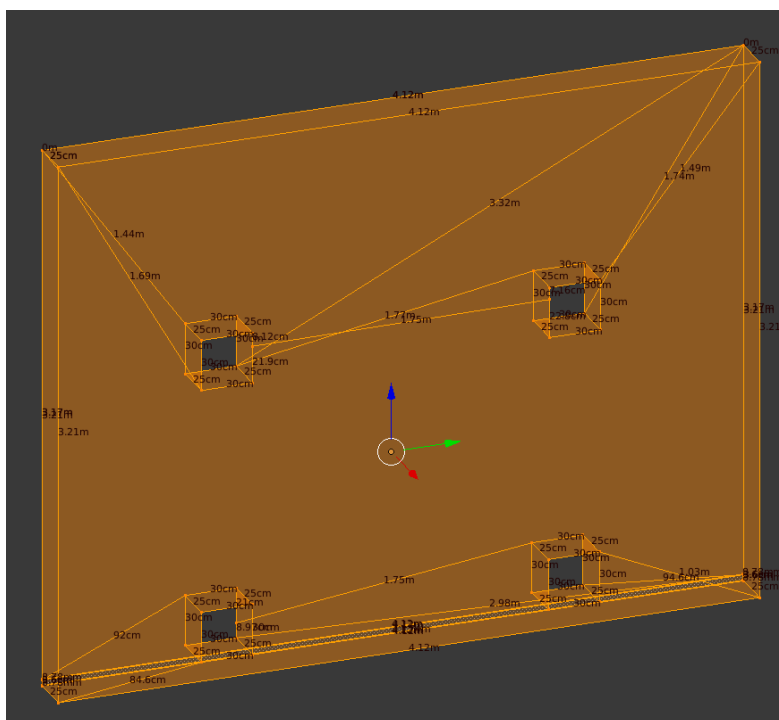
Definida a técnica de modelagem criou-se o primeiro projeto no BLENDER de nome catolica.blend, no qual modelou-se o que seriam os componentes básicos para reprodução do ambiente, componentes esses que possibilitam a reutilização para a montagem e construção do ambiente na UNITY. Ao todo o projeto criado no BLENDER católica.blend contém 55 modelos tridimensionais, como portas, portais, janelas, paredes, caixa de incêndio, extintor, grades, vigas, pilares e vidros.





*Figura 18 Modelagem 3D do Projeto Católica*  
 Fonte: (AUTOR, 2018)

As criações desses componentes partem de um vértice que é estendido digitando o tamanho em metros desejado, gerando os próximos vértices ligando-o aos demais. O BLENDER possibilita a utilização da métrica em metros realizando a configuração na aba Scene para a medição métrica e após isso ativando a régua na aba Transform, com essa configuração os objetos 3D ficam mais próximos em tamanho e formato dos objetos reais aumentando a fidelidade da modelagem.



*Figura 19 Modelagem de Superfícies duras em Polígonos*  
 Fonte: (AUTOR, 2018)

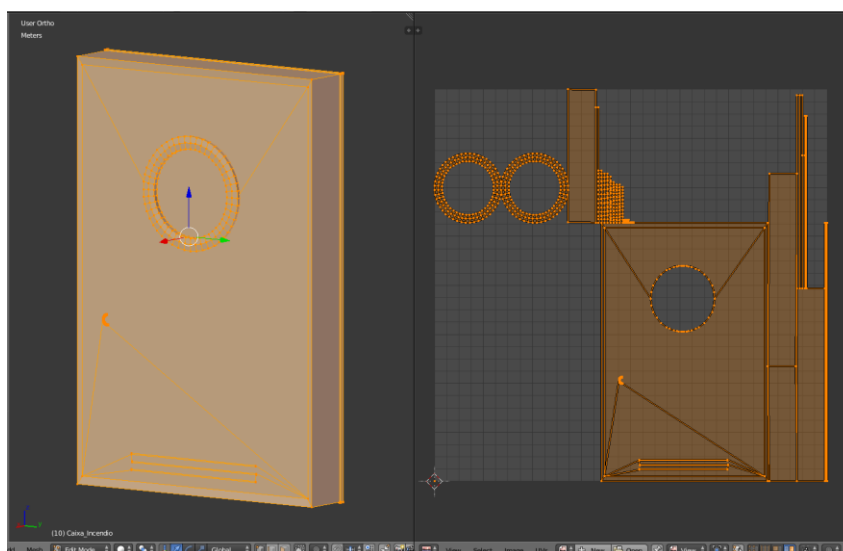
### 3.2 TEXTURIZAÇÃO

A criação das texturas do projeto, utilizou-se das fotos capturadas no campus com a câmera fotográfica semiprofissional NIKON COOLPIX P520. Essas fotos foram cortadas em seus melhores pontos, definidas com resolução de 1024 pixels por 1024 pixels para uma textura de qualidade alta, e com o uso do editor de imagens GIMP aplicou-se o filtro na textura que mapeia a imagem, e a torna encaixável, assim criando as texturas utilizadas no projeto, abaixo pode-se visualizar a textura criada para as paredes externas do campus.



*Figura 20 Textura Encaixável Das Parede Externas  
Fonte: (AUTOR, 2018).*

Para que fosse possível realizar a texturização correta dos objetos na UNITY, é preciso a criação dos mapas de UV de cada objetos modelado, como pode ser visto na Figura 21 abaixo, onde o processo foi realizado no objeto 3D da caixa de incêndio. Após esse mapeamento da malha a modelagem estava pronta para ser exportada.



*Figura 21 3D e Mapa UV da Caixa de Incêndio  
Fonte: (AUTOR, 2018).*

### 3.3 EXPORTAÇÃO DOS MODELOS PARA UNITY

O primeiro pacote de objetos 3D modelados “católica.blend” levou pouco mais de um mês para ser modelado. Com todos os modelos criados e configurados o projeto foi exportado do BLENDER para o formato FBX que é suportado na UNITY, o processo de exportação é simples e pode ser realizado rapidamente, mesmo se tratando de um projeto com muitos componentes.

Para a exportação da modelagem em FBX é necessário realizar algumas rápidas configurações no momento da exportação, são elas: alterar o eixo que indica a orientação vertical, no BLENDER esse eixo é representado pelo Z e na UNITY é representado pelo Y e remover componentes da exportação que não vão ser utilizados na UNITY, o BLENDER além de ser um modelador 3D é também um animador e renderizador, e dispõe de componentes como iluminação e câmeras, e esses componentes são incorporados ao projeto para exportação, com isso ao exportar um projeto em FBX para UNITY esses componentes devem ser removidos, pois os mesmos serão criados e configurados através da UNITY.

Realizadas essas configurações o primeiro pacote modelado foi exportado em FBX com nome de “pacote\_catolica.fbx” para ser importado no motor de jogos.

### **3. DESENVOLVIMENTO DO PROJETO EM UNITY**

Com a finalização da modelagem foi dado início ao processo de desenvolvimento no motor de jogo.

#### **4.1 CRIAÇÃO E CONFIGURAÇÃO DO PROJETO**

Para criação do projeto colocou-se o nome do projeto de: “Projeto\_Catolica”, configurou-se em 3D e não foi adicionado ao projeto nenhum pacote de *Asset* inicial.

Após a criação realizou-se a configuração dos diretórios de projeto, para fácil localização dos componentes do projeto, as pastas de diretórios inicialmente criados para organização do projeto foram: “Cena”, “Material”, “Modelos”, “Scripts” e “Sound”.

Para auxílio no desenvolvimento, a UNITY dispõe de pacotes que podem ser importados para o projeto gratuitamente, esses pacotes contam com scripts,

funções, personagens, iluminação, terrenos, efeitos, controles entre outros recursos iniciais necessários em muitos projetos. Tendo em vista a necessidade do projeto, foram importados os pacotes “Standar Assets”, “Terrain Toolkit 2017” e importado da loja se assetes da UNITY o pacote gratuito “Virtual Joystick Pack”.

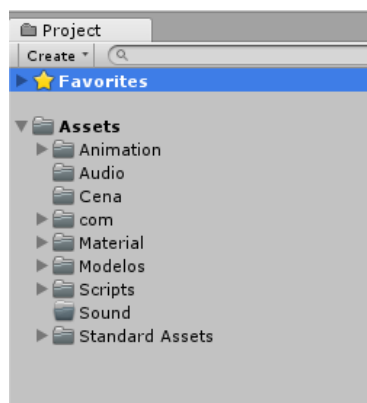


Figura 22 Diretório de Arquivos do Projeto  
Fonte: (AUTOR, 2018)

## 4.2 CONSTRUÇÃO DO AMBIENTE VIRTUAL

A construção do ambiente começou com a importação do pacote de modelagem “pacote\_catolica.fbx” para a pasta de Modelos no diretório da Unity. O processo para importação é simples, bastando arrastar o arquivo de modelagem para a pasta Modelos, a UNITY processa esse arquivo gerando um arquivo do tipo *prefab* “pré fabricado” que contém todos modelos como um pacote e cria uma pasta de materiais contento os materiais que são elementos para aplicação das texturas aos modelos, criando um material para cada modelo 3D. Esses materiais gerados foram movidos para a pasta “Material” que havia sido criada anteriormente, para manter a organização do projeto.

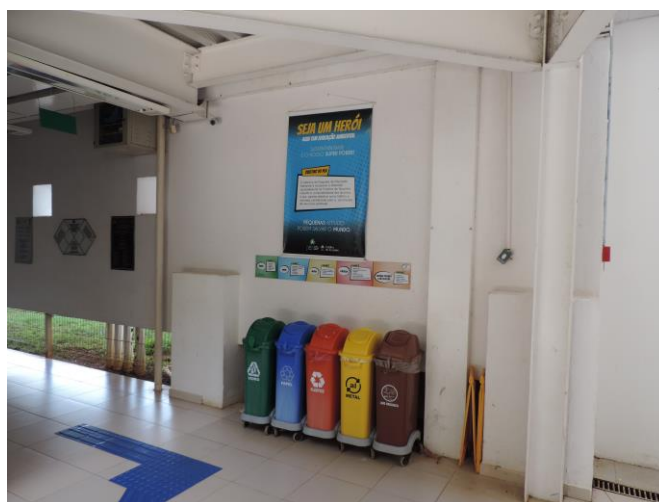
O início da construção da cena começou com a configuração de diretórios da cena, criando um objeto vazio nomeado de FACTO na cena, o qual receberia os modelos 3D que compõem o projeto.

Dentro do objeto FACTO criou-se outro objeto vazio de nome Assets, dentro desse objeto colou-se toda base modelada para replicação e construção do ambiente virtual.

Realizou-se a adição de mais um objeto vazio ao objeto FACTO de nome “Objetos.Terreo”, onde seriam adicionados todos os modelos que compõem o terreno da área virtualizada.

A cena inicial do projeto foi salva na pasta do diretório de arquivos Cena com nome Católica.

Feitas essas configurações deu-se início ao processo de montagem, começando com a montagem a partir da entrada no corredor do térreo do bloco I e seguindo até a porta de vidro na saída para o estacionamento dos professores.



*Figura 23 Ponto de partida para modelagem  
Fonte: (AUTOR, 2018)*

Cada modelo adicionado à cena, será configurado sua localização no espaço virtual, e realizando sua texturização através das ferramentas nativas da UNITY, utilizando as texturas criadas anteriormente. Uma vez que esse modelo era configurado, ele estava pronto para ser replicado. Esse processo foi realizado em cada um dos modelos à medida que eram adicionados a cena do projeto.

O processo de replicação dos modelos adicionados à cena agiliza a construção do cenário, uma vez que o modelo está pronto, basta replica-lo e arrastar o modelo replicado para a posição desejada no espaço virtual, aproveitando o posicionamento o nos eixos que não serão modificados, obtendo precisão no posicionamento dos modelos ao cenário.

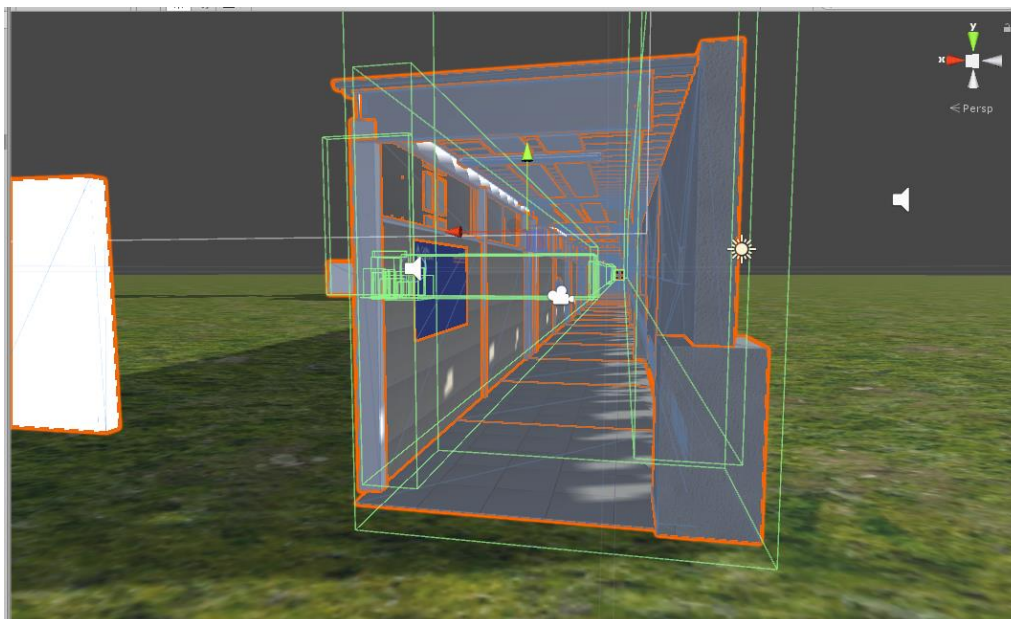
Para maior organização dos modelos do cenário, dentro do objeto vazio “Objetos.Terreo” foram criados 18 outros objetos vazios para receber cada tipo de modelo e suas replicações, como paredes, portas, teto entre outros, trazendo organização e boas práticas para o desenvolvimento do projeto, como pode ser visto na imagem a seguir:



Figura 24 Hierarquia de objetos do térreo  
Fonte: (AUTOR, 2018)

Para teste no cenário durante a construção, adicionou-se ao projeto o componente do Standard Assets o “FPSController”, o qual contém uma câmera com controles e visão de um jogo em primeira pessoa. Através desse componente realizou-se a inspeção da construção do cenário em execução, onde podem ser encontradas falhas que só ocorrem quando o projeto está em execução, como luzes atravessando paredes, locais sem colisão para controle do personagem e modelos fora de posicionamento.

Com a execução do projeto realizou-se as correções das falhas de construção no cenário, finalizando a construção do térreo, o objeto de cena “Objetos.Terreo” se transforma na base para replicação e criação dos demais andares, com sua construção finalizada, esse objeto foi replicado para criação dos demais andares, os quais receberam os nomes de: “Objetos.Primeiro”, “Objetos.Segundo” e “Objetos.Terceiro”, referenciando os objetos de cada andar.



*Figura 25 Objetos.Terreo base para replicação do Cenário  
Fonte: (AUTOR, 2018)*

Com a replicação e criação dos andares, aplicou-se a caracterização e particularidades de cada andar, apesar de serem parecidos, cada andar da instituição possui suas particularidades como varanda no segundo andar, grades e pilares na parede externa do térreo, janelas na parede externa do terceiro andar, e o posicionamento de algumas portas.

Devido à complexidade e a alta quantidade de elementos existentes nas escadarias, realizou-se o mesmo procedimento de construção do térreo, construindo através do objeto “Escadaria1” a base, realizando os posicionamentos das paredes, degraus, pisos, vigas e demais componentes da escadaria. Com a base construída o objeto foi replicado criando a “Escadaria2”, e “Escadaria3”, que foram posicionadas em seus devidos locais.

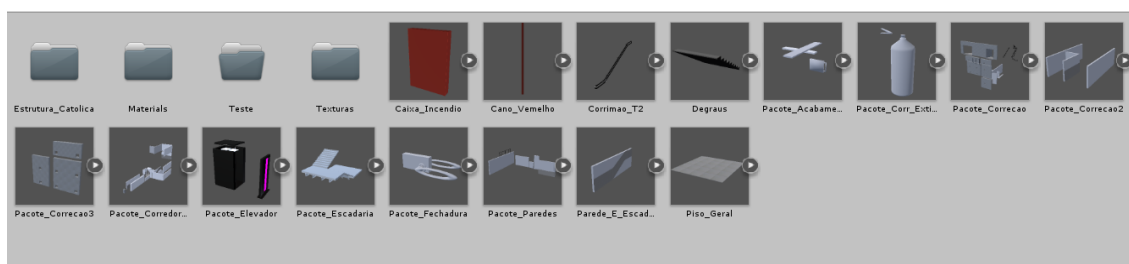
#### **4.2.1 Correções e novas modelagens para construção do cenário**

Durante o processo de montagem do cenário foram detectadas várias falhas no pacote de modelos “pacote\_catolica.fbx”, e a falta de alguns componentes que não haviam sido modelados e algumas particularidades em alguns modelos necessitando de modificação.

Alguns modelos continham falhas em sua malha, apresentando distorções da modelagem na UNITY, outros modelos estavam com suas faces voltas para o lado errado, e dependendo da visão do jogador essa face fica invisível, devido esses problemas. Durante o processo de construção do cenário foram realizadas

novas modelagens no BLENDER, para correção dos modelos defeituosos e para a construção de novos modelos.

As correções e criações de novos modelos resultaram na criação de 15 novos pacotes de modelos, e para evitar novos erros, dentro do diretório de arquivos da UNITY na pasta de Modelos foi criado uma pasta de nome “Teste”, que recebia os novos modelos antes de ser adicionado aos modelos já existentes, para não gerar arquivos desnecessários ao projeto, realizando os testes da modelagem na UNITY e detectando falhas para correção.



*Figura 26 Modelos 3D usados no projeto*  
*Fonte: (AUTOR, 2018)*

### 4.3 ILUMINAÇÃO

O cenário conta a iluminação padrão da Unity, usando a Direction Ligth que ilumina todo o cenário, no objeto Direction Ligth realizou-se apenas a rotação do angulo para simular a iluminação da luz solar na parte da tarde.

A luz gerada pelo Direction Ligth tem capacidade de iluminar todo o cenário, testou-se a iluminação do tipo Point Ligth aplicado nas luminárias do imóvel, deixando a iluminação global escura para simular a noite. O resultado dessa iluminação trouxe um enriquecimento ao ambiente, mas consumia muito recurso de *hardware*, o que é limitado nos *smartphones*, e devido esse alto consumo de *hardware* foram removidas, ficando apenas a iluminação global como configurada anteriormente com aspecto de tarde.

### 4.4 ANIMAÇÃO

O processo de animação do ambiente alinha-se ao processo de programação, as animações são criadas na UNITY através de suas ferramentas Animation e Animator, sendo que a primeira cria a animação através de uma linha de tempo e a segunda organiza o funcionamento e atividade das



animações, e além das duas ferramentas é necessária programação para realizar as execuções de suas funções da forma desejada.

No projeto existem apenas dois objetos animados, que são as portas de incêndio das escadarias, e as portas do elevador. Apenas estes objetos possuem animação pois são os únicos que necessitam de movimentação ao realizar a interação com o usuário.

Para animação da porta de incêndio necessitou-se adicionar um pivô ao canto da modelagem para possibilitar a realização da rotação no eixo desse pivô, simulando a abertura da porta, sem esse pivô, a rotação seria realizada no centro da porta o que não condiz com a realidade.

O pivô de rotação para a porta de incêndio foi criado como um objeto vazio dentro do objeto “PortaIncendio”, e todo os modelos da porta como maçanetas, caixa de fechadura, porta e placas, foram movidos para dentro do objeto pivô, para que recebessem a rotação que seria aplicado ao pivô na animação da porta.

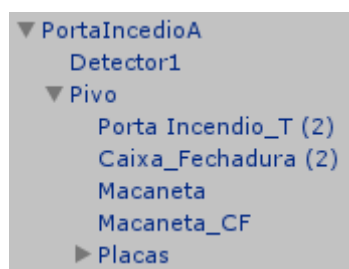


Figura 27 Estrutura de componentes da porta animada  
Fonte: (AUTOR, 2018)

Feito essa configuração iniciou-se o Animation da UNITY que possui uma linha de tempo onde se realiza as animações. No tempo zero a porta está em seu local e rotação original, é iniciada a gravação da animação, a agulha da linha do tempo é movida para 1 segundo e 42 centésimos, a rotação é aplicada à porta até o ângulo desejado de abertura, e finalizasse-se a gravação, o processo é repetido para realizar o fechamento da porta, mas dessa vez partindo do ângulo de abertura até o ângulo em que a porta se fecha. Finalizado esse processo, é possível testar o resultado no Animation apertando *Play*, e verificando se a animação ficou como desejada.

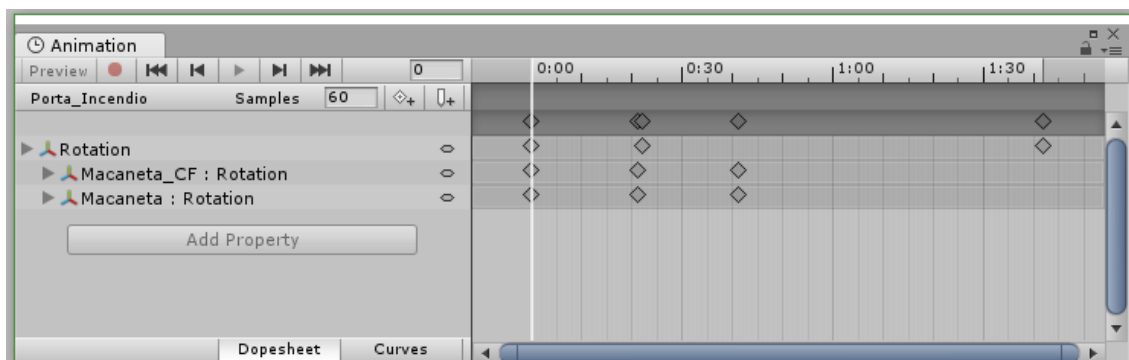


Figura 28 Animação de abertura para porta de incêndio  
Fonte: (AUTOR, 2018)

Utilizando outra ferramenta da UNITY o Animator foi realizado a ligação das duas animações criadas para a porta de incêndio, “Porta\_Incendio” e “FechaPorta\_Incendio”, através do Animator realiza-se a ligação e interação entre as animações, no Animation configurou-se um estado inicial que mantém a porta fechada, e os estado que chamam a abertura e fechamento da porta interligando os eventos.

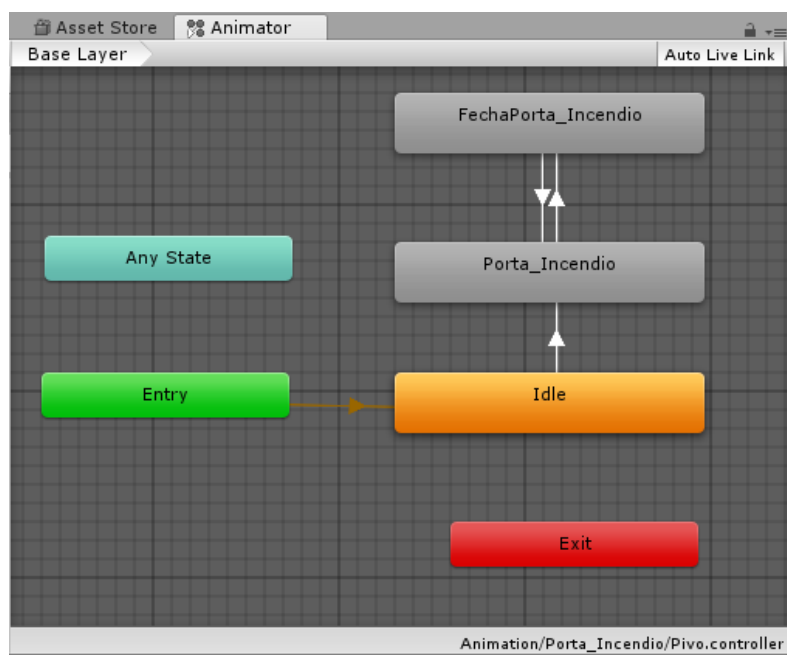


Figura 29 Interação entre as animações no Animator  
Fonte: (AUTOR, 2018)

No processo seguinte adicionou-se um campo de colisão no objeto “PortalIncendio” para detecção da chegada do usuário. A esse campo de colisão foi adicionado o *script* “Detecta” para ativar as funções de detecção e início das animações de abertura e fechamento da porta, de acordo com a entrada e saída do usuário ao campo de colisão, o *script* pode ser conferido a seguir:

```

public class Detecta : MonoBehaviour {

    //Referência ao Player
    public string _playerTag = "Player";

    //Detecta colisão através do Trigger
    void OnTriggerEnter(Collider _cole)
    {
        //Detecta apenas o Player
        if (_cole.CompareTag(_playerTag))
        {
            Debug.Log(_cole.gameObject.name);
            //Ativa a animação para abrir a porta
            PortalIncendio._abrirPortalIncendio = true;
        }
    }
    //Detecta colisão através do Trigger
    void OnTriggerExit(Collider _cole)
    {
        //Detecta apenas o Player
        if (_cole.CompareTag(_playerTag))
        {
            Debug.Log(_cole.gameObject.name);
            //Ativa a animação para fechar a porta
            PortalIncendio._fecharPortalIncendio = true;
        }
    }
}

```

O processo de animação das portas elevador decorreu da mesma maneira, com a diferença que a porta do elevador por ser uma porta de correr, não foi necessário adicionar o pivô de rotação, o segundo diferencial nessa animação foi realizar a animação síncrona das duas partes da porta para abertura e fechamento, as demais atividades da configuração foram usando os mesmo processos e *script* usados na porta de incêndio, para detecção do usuário e acionamento das animações.

#### 4.4 CRIAÇÃO DOS MODOS DE JOGO

Construído e animado o ambiente virtual, obteve a base do projeto para a criação dos modos de jogos propostos no objetivo, que foram: o modo em que o aplicativo guia o usuário dentro do ambiente virtual para a sala desejada chamado de Modo Guia, e o modo em que o usuário pode navegar livremente dentro do ambiente virtual chamado de Modo Livre.

#### 4.4.1 Modo Livre

Para configuração desse modo, a cena do jogo a qual se fez todo o projeto até o momento teve o nome alterado de Católica para Free. E utilizando-se do Standard Assets “FPSController” que havia sido adicionado anteriormente ao projeto, a base do modo livre estava construída, pois tinha-se o ambiente virtual pronto e um controle de personagem funcional já adicionado ao projeto, mas esse controle de personagem não estava pronto para dispositivos *mobile*.

Para realizar o controle do personagem através das plataformas *mobile*, criou-se um Canvas na cena Free, dentro do objeto Canvas criou-se um painel que simula a tela e visão do usuário para realizar a adição e organização de componentes a tela, como botões e menus.

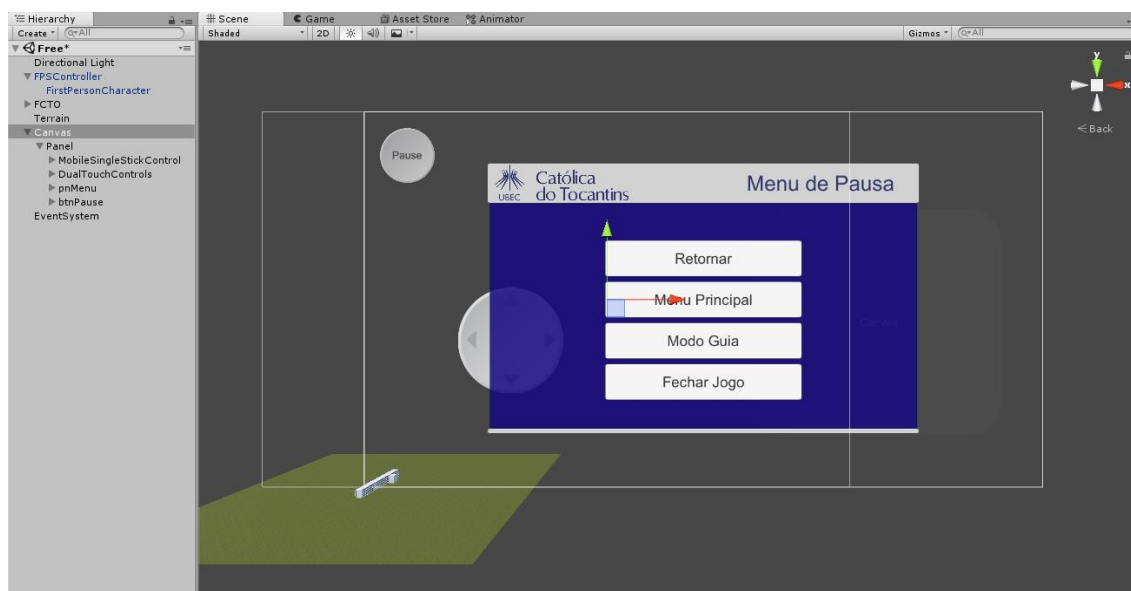


Figura 30 Estrutura de Canvas e Painel para interação em dispositivos moveis.  
Fonte: (AUNTOR, 2018)

Dois pacotes do Standard Assets da UNITY foram importados para o Painel, o “MobileSingleStickControl” e o “DualTouchControls”, através da mesclagem de funções desses componentes, utilizando-se o componente “MobileJoystick” do pacote “MobileSingleStickControl” para o controle de movimentação do personagem, e o componente “TurnAndLookTouchpad” do pacote “DualTouchControls” para movimentação da visão do personagem, os dois pacotes continham mais componentes que não seriam necessários e foram removidos.

Os controles mobiles adicionados interagem sem necessidade alguma de configuração com o pacote “FPSController”, onde as configurações realizadas

nos controles foram apenas de tamanho e posicionamento na tela, assim obtendo-se o controle mobile do personagem construído.

Adicionou-se ao painel um botão de pause e outro painel menor onde criou-se o menu de pausa que pode ser visto na imagem acima. O botão pause foi configurado para acionar o menu de pausa e pausar completamente a aplicação enquanto o menu estiver aberto, para isso foi adicionado ao Canvas o *script* MenuPause que possui os métodos e funções para as ações do botão pause e dos botões do menu de pausa, a seguir o *script* criado para operacionalizar o botão Pause e o Menu.

```
public class MenuPause : MonoBehaviour {

    //Cria um estado para jogo
    private bool _estadopause = false;
    //Local para associar o objeto de cavas do meu pause
    public GameObject _canvasPause;
    //Local para associar o objeto do botão pause
    public GameObject _btnPause;

    private void Update()
    {
        //Pausa o game no menu de pausa
        if (_estadopause)
        {
            Time.timeScale = 0;
        }
        else
        {
            Time.timeScale = 1;
        }
    }

    public void ModoPause()
    {
        //Ativa o pause
        _estadopause = true;
        //Ativa o Menu
        _canvasPause.SetActive(_estadopause);
        //Oculta e desativa o botão de pause
        _btnPause.SetActive(false);
    }

    public void Retornar()
    {
        //Desativa o pause
```

```

        _estadopause = false;
        //Desativa e Oculta o Menu de Pausa
        _canvasPause.SetActive(_estadopause);
        //Ativa o Botão pause
        _btnPause.SetActive(true);
    }
    //Metodo que chama a cena do Modo Guia
    public void ModoNavegation()
    {
        //Carrega a cena de Guide e fecha as outras cenas
        SceneManager.LoadScene("Guide", LoadSceneMode.Single);
    }

    //Metodo que chama a cena do Menu Inicial
    public void MenuPrincipal()
    {
        //Carrega a cena de Menu e fecha as outras cenas
        SceneManager.LoadScene("Menu", LoadSceneMode.Single);
    }

    //Metodo que finaliza o Game
    public void SairJogo()
    {
        //Fecha a aplicação
        Application.Quit();
    }
}

```

Com a realização dessas configurações o modo livre ficou pronto e totalmente funcional.

#### 4.4.2 Modo Guia

O modo guia tem o objetivo de realizar navegações programadas dentro do ambiente virtual, onde o usuário indicaria onde está no ambiente virtualizado e para qual sala deseja ir, com isso o aplicativo gera a rota mais próxima para o objetivo do usuário e produz uma animação movendo a câmera de visão do usuário guiando-o através do ambiente virtual.

O desenvolvimento do modo guia fez-se a partir do Modo Livre, para isso realizou-se uma cópia do modo livre que foi renomada para Guide, nesse modo o usuário não realiza o controle do personagem, não havendo a necessidade dos controles *mobile* e do personagem que foram adicionados anteriormente, esses componentes foram removidos da cena, preservando o Canvas, o Pannel e o menu de pausa e os demais componentes base do projeto.

Utilizando a função de encontro o caminho *Pathfinding* da UNITY, realizou-se o mapeamento do ambiente virtualizado através da ferramenta “Navigation”, esse mapeamento consiste em selecionar todos os locais/objetos por onde o personagem possa transitar.

Realizou-se esse mapeamento em etapas, graças a organização estrutural dos objetos da cena esse mapeamento pode ser realizado de forma organizada e rápida, inicialmente selecionou todo piso do térreo, e na ferramenta “Navigation” na aba “Objetc” marcou-se as obções “Navigation Static”, “Generate OddMeshLinks” e em “Navigation Area” selecionou-se “Walkable”, essas opções definem que os objetos selecionados poderão gerar rotas para a navegação do objeto de navegação.

Após realizar essas configurações ainda na ferramenta “Navigation” na aba “Bake”, configurou-se o “Agent Radius” para 0.4, o “Step Height” para 0.4, essas funções definem o tamanho do objeto que será usado para a navegação e clicou-se no botão “Bake” para gerar as rotas, o resultado pode ser observado a seguir, sendo as trilhas azuis os locais de geração automática de rotas por onde o objeto de navegação pode percorrer.

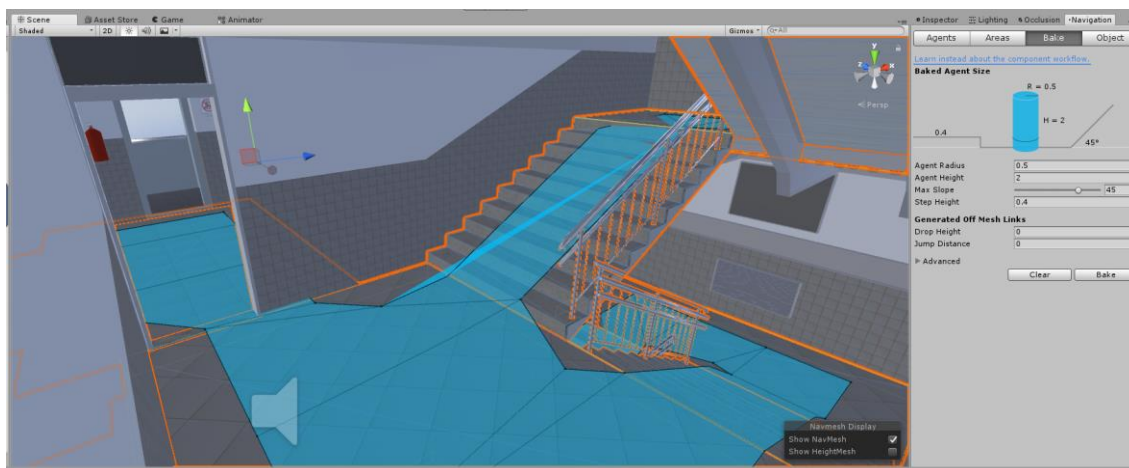


Figura 31 Rotas de Pathfinding  
Fonte: (AUTOR, 2018)

As rotas geradas automaticamente continham algumas falhas, as quais faziam a navegação atravessar paredes e outros locais que não deveria como pode ser visto a cima a rota azul atravessando a parede, para correção dessas falhas criou-se um cubo configurando-o na ferramenta “Navigation” como os demais componentes configurados para gerar rota, com a exceção na configuração “Navigation Area” a qual foi definida como “Not Walkable”, feito

essa configuração, replicou-se o cubo para as áreas com falhas, posicionando o cubo dentro dos objetos parede, ocultando esses componentes da visão do usuário, após esse processo foi realizado um novo Bake, obtendo-se a correção das rotas de navegação.

Adicionou-se a cena do projeto uma câmera, a qual será a visão do usuário nesse modo do aplicativo, a esta câmera adicionou-se o componente de navegação “Nav Mesh Agent”, tornando-a o objeto de navegação, e posicionando-a no início do corredor do térreo.

Para realização de testes de *Pathfinding* com objeto de navegação, criou-se o *script* “PontoNave”, baseado em um *script* presente na documentação da UNITY, associando esse *script* a câmera da cena. Esse código adiciona ao objeto de navegação que no caso é a câmera, um local para inserir um objeto que se torna o alvo da câmera, definido o objeto alvo a inteligência artificial da UNITY traça a rota mais rápida e leva a câmera até o objeto alvo, o código pode ser conferido a seguir:

```
public class PontoNave : MonoBehaviour {

    //Variável que recebe o objeto alvo
    //para navegação
    public Transform _local;

    // Use this for initialization
    void Start () {
        //Associa objeto de navegação ao componente Nav Mesh Agent
        NavMeshAgent agent = GetComponent<NavMeshAgent>();
        //Move Obejto de navegação através das rotas
        //até o local desejado
        agent.destination = _local.position;
    }

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

    }

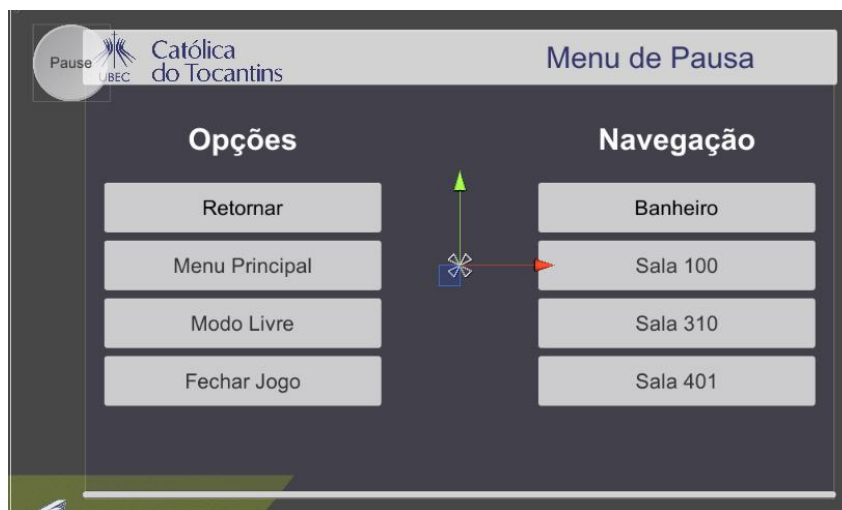
}
```

Os testes foram realizados obtendo sucesso na navegação automatizada, bastando adicionar um objeto do local desejado a câmera, para que a câmera navegasse automaticamente pelo cenário até esse local.

O menu de pausa da cena Guia recebeu uma modificação para acrescentar as funções desse modo, o *script* usado para configuração do menu



levou como base o *script* criado para o Modo Livre e nomeado como “MenuPauseNav”. Nesse novo menu foram adicionados mais quatro botões para realização da demonstração do modo guia, pois não havia mais tempo para desenvolvimento de todo guia para navegação no ambiente, assim adicionou-se ao menu os botões Banheiros, Sala 100, sala 310 e sala 401.



*Figura 32 Menu de Pausa do modo Guia*  
*Fonte: (AUTOR: 2018)*

Realizou-se várias buscas na documentação e em tutorias da UNITY e não se encontrou em tempo hábil uma forma de associar esses os objetos alvo aos botões, e fazer com que os botões adicionassem os objetos alvo a câmera, inviabilizando a escolha do local alvo para navegação, essa função ficou pendente devido o tempo curto para o desenvolvimento do projeto.

Outra função pendente devido ao pouco tempo para a construção do projeto é a de tele-transporte do objeto de navegação câmera, para o local onde o usuário indica que está.

Este modo é finalizado com uma função pré-definida ao objeto de navegação, que o leva até a varanda do segundo andar, apenas para demonstração da navegação automatizada.

#### 4.4.3 Menu Principal

Para a primeira cena ao abrir o jogo criou-se uma nova cena de nome Menu. A está cena adicionou-se um Canvas, e neste elemento foram inseridos um componente “Image” para configuração da imagem de fundo, um Painel para abrigar a logo da instituição e três botões, com função de chamar as outras cenas e finalização do jogo/aplicativo.

Assim como nas outras cenas o objeto Canvas recebeu um *script* para adicionar as funções dos botões, mas com um diferencial nestes botões, uma vez que foram clicados, esses botões tem os textos alterados para “carregando...” e impossibilitam que o usuário realize cliques em outro botão durante o carregamento da cena ou finalização do projeto, o *script* pode ser conferido a seguir:

```
public class MenuPrincipal : MonoBehaviour {
    public Button _btnModoLivre;
    public Button _btnModoNave;
    public Button _btnSairJogo;
    public Text _text1;
    public Text _text2;
    public Text _text3;
    //Método que chama a cena de modo livre
    public void ModoLivre()
    {
        //Impede que o usuário realize mais de um click
        _btnModoLivre.enabled = false;
        //Altera o texto do botão ao clicar
        _text1.text = "Carregando...";
        //Carrega a cena de ModoLivre e fecha as outras cenas
        SceneManager.LoadScene("Free", LoadSceneMode.Single);
    }
    //Método que chama a cena de Guia
    public void ModoNavegation()
    {
        //impede que o usuário realize mais de um click
        _btnModoNave.enabled = false;
        //Altera o texto do botão ao clicar
        _text2.text = "Carregando...";
        //Carrega a cena de ModoLivre e fecha as outras cenas
        SceneManager.LoadScene("Guide", LoadSceneMode.Single);
    }
}
```

```
//Método que finaliza o Game
public void SairJogo()
{
    //Impede que o usuário realize mais de um click
    _btnSairJogo.enabled = false;
    //Altera o texto do botão ao clicar
    _text3.text = "Fechando...";
    //Fecha a aplicação
    Application.Quit();
}
}
```

Realizadas essas configurações o menu principal estava concluído, como pode ser observado na Figura 33 abaixo.



*Figura 33 Menu Principal  
Fonte: (AUTOR, 2018)*

#### 4.5 EXPORTAÇÃO DO PROJETO

Com a plataforma objetivo do projeto definida como Android, para realizar a compilação do projeto e testes para essa plataforma é necessário configurar a UNITY, os requisitos solicitados pela UNITY para exportação nessa plataforma é: instalação do Android Studio que consequentemente necessita da versão JDK do JAVA.

Realizadas essas instalações a UNITY pode ser configurada para a exportação, esse processo é realizado através no menu de “Build Settings”, selecionando o Android e clicando em “Switch Platform”. Feito isso, a depender

do tamanho do projeto já desenvolvido leva-se alguns minutos para a UNITY processar a configuração.

O processo seguinte é, ainda na tela de “Build Settings” selecionar as cenas que serão exportadas, que nesse projeto são as cenas Menu, Free e Guide, a cena de referência 0 (zero) na Build Settings é a cena que inicia o jogo, no caso é a cena Menu.

Ainda na tela “Build Settings”, ativa-se a opção “Player Settings”, iniciada no “Inspector”, esta opção traz configurações mais avançadas para exportação, configurando-se o nome do projeto, nomeado de “Católica Virtual”, adiciona-se o ícone padrão, seleciona a versão da API do Android, que neste utilizou-se da API level 23 para Android 6.0 Marshmallow. A API do Android deve estar instalada no Android Studio para ser utilizada pela UNITY, além dessas configurações que foram utilizadas nesse desenvolvimento, existem muitas outras que devem ser configuradas de acordo com as propostas de projeto a serem criados.

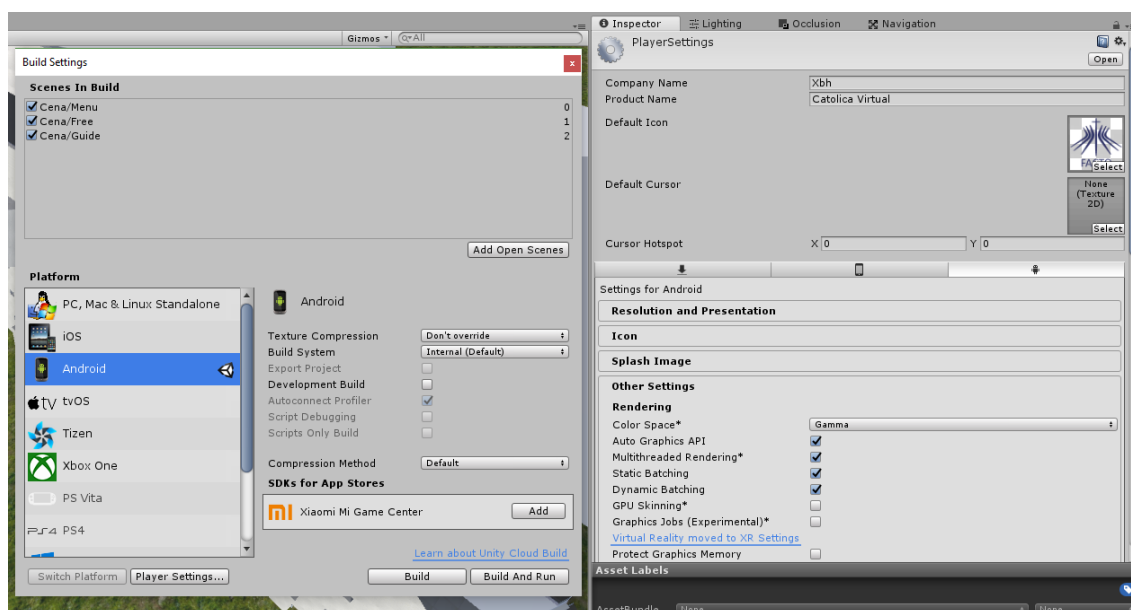


Figura 34 Build Settings e Player Settings  
Fonte: (AUTOR, 2018)

A última configuração necessária para a exportação/compilação do projeto para Android é realizada através do menu “Edit”, na opção “Preferences...”. Nessa janela na aba “External Tools” em “Android” adiciona-se os diretórios de instalação de SDK do Android Studio e o JDK do JAVA.

Realizadas essas configurações o projeto está pronto para compilação, o processo é realizado através da janela “Build Settings” clicando em “Build”, o processo, a depender do tamanho do projeto e do poder de processamento da máquina, pode ser rápido ou demorado. No caso deste trabalho o processo de compilação levou cerca de quatro minutos para ser realizado na máquina utilizada no estudo. A última versão do projeto compilada gerou uma APK de 34,7MB.

A exportação do projeto para outras plataformas, basta realizar as configurações de peculiaridades da plataforma desejada e realizar a exportação, pois a base do projeto está construída.

#### 4.6 FINALIZAÇÃO DO DESENVOLVIMENTO DO PROJETO

O desenvolvimento finalizou-se em um projeto UNITY com tamanho de 422 MB, estruturado e de fácil entendimento para continuidade do desenvolvimento. A versão compilada para Android compactou esse projeto em uma APK de 34,7MB em sua última versão, o que é um tamanho aceitável para um aplicativo móvel, devido a limitação de armazenamento de muitos desses dispositivos.



*Figura 35 Projeto finalizado*  
*Fonte: (AUTOR, 2018)*

#### 4. CONCLUSÃO

Os desafios foram grandes, num período de cinco meses foram realizadas as duas primeiras etapas deste estudo, as quais realizam os estudos e desenvolvimentos nos motores de jogos, que culminaram na escolha do motor

UNITY. Que posteriormente necessitaria de mais aprendizado para uso neste trabalho.

Com um cronograma curto, a terceira e última etapa do estudo levou mais cinco meses, realizando o aprimoramento no aprendizado para o uso do motor de jogo UNITY, aprendizado com o modelador 3D, a modelagem 3D do ambiente, a construção do cenário e o desenvolvimento do aplicativo Católica Virtual.

Com o objetivo de virtualização de parte do bloco I do Campus I da Faculdade Católica do Tocantins através do motor de jogo UNITY. Este projeto modelou as partes do imóvel através do modelador 3D BLENDER, e construiu e animou o cenário com o UNITY.

Realizada a construção do cenário virtualizado, os próximos objetivos a serem realizados seriam o desenvolvimento dos modos “Modo Livre” e “Modo Guia”. O “Modo Livre” teve sua construção finalizada com sucesso, possibilitando ao usuário na última versão do aplicativo a navegação pelo ambiente virtualizado.

O “Modo Guia” que teria a função de localização das salas, por ser o último modo do projeto a ser desenvolvido, não foi possível sua finalização, devido o curto tempo para o desenvolvimento do projeto. Contudo, o ambiente criado está mapeado para a função de localização. Uma demonstração do funcionamento desse modo foi criada, e está disponível na última versão exportada do projeto para Android.

Este estudo atendeu aos seus objetivos de modelagem, construção do ambiente virtual, desenvolvimento do “Modo Livre” e exportação do projeto para Android, deixando pendente a finalização do desenvolvimento do “Modo Guia” que está inicialmente configurado e permitindo que o trabalho possa ser concluído em um desenvolvimento futuro.

A proposta de desenvolvimento de aplicação com realidade virtual para Android com o motor de jogo UNITY, foi atendida. Ao final desse estudo tem-se como resultado um aplicativo/jogo criado para Android, com o ambiente virtualizado de parte do Campus I da Faculdade Católica do Tocantins.

Espera-se com este estudo demonstrar o processo de desenvolvimento de jogos, e virtualização de ambientes reais, e através desse trabalho inspirar

novos desenvolvedores de jogos. Pois o trabalho de criação é árduo, mas ver o resultado de tanto trabalho pronto é gratificante e inspirador.

## 6. REFERÊNCIAS BIBLIOGRÁFICAS

2017. ADVENTURE OF POCO ECO - LOST SOUNDS <http://pocoecogame.com>. Acessado em novembro/2017.
2017. APPGAMEKIT <https://www.appgamekit.com/>. Acessado em dezembro/2017.
2017. BAD PIGGIES <https://www.angrybirds.com/games/bad-piggies/>. Acessado em novembro/2017.
- BARROS, G. - MODELAGEM DIGITAL TRIDIMENSIONAL PARA O DESENVOLVIMENTO DE PROTOTIPAGEM RÁPIDA - 2012 <https://repositorio.ufpe.br/bitstream/123456789/11470/1/dissertaçãoGutenberg7-final-ebook.pdf> Acessado em maio/2018
2017. BLENDER <https://www.blender.org>. Acessado em dezembro/2017.
2018. BLENDER DOCUMENTAÇÃO MAPA UV [https://docs.blender.org/manual/en/dev/editors/uv\\_image/uv/overview.html](https://docs.blender.org/manual/en/dev/editors/uv_image/uv/overview.html) Acessado em maio/2018
- BURDEA, G. C. e COIFFET, P. Virtual Reality Technology. Wiley-Interscience, second edition edition, 2003.
2017. CORONA LABS <https://coronalabs.com>. Acessado em dezembro/2017.
2017. DICIONARIO DICIO, DEFINIÇÃO DE DESEMPENHO <https://www.dicio.com.br/desempenho/>. Acessado em novembro/2017.
2018. EA (ELECTRONIC ARTS) ENGINE FROSTBITE <https://www.ea.com/frostbite> Acessado em maio/2018
- ELIS V. R., BARROSO C. M. R., KIANG C. H - Aplicação de ensaios de resistividade na caracterização do Sistema Aquífero Barreiras / Marituba em Maceió – AL - 2004 [http://www.scielo.br/scielo.php?pid=S0102-261X2004000200001&script=sci\\_arttext](http://www.scielo.br/scielo.php?pid=S0102-261X2004000200001&script=sci_arttext). Acessado em maio/2018
- <https://play.google.com/store/apps/details?id=com.epicgames.EpicCitadel>. Acessado em novembro/2017.
2017. EPIC CITADEL <https://play.google.com/store/apps/details?id=com.epicgames.EpicCitadel>. Acessado em novembro/2017.
2017. EXPORTAÇÃO DE PROJETOS DA GODOT PARA ANDROID [http://docs.GODOTengine.org/en/stable/learning/workflow/export/exporting\\_for\\_android.html](http://docs.GODOTengine.org/en/stable/learning/workflow/export/exporting_for_android.html) Acessado em novembro/2017.
- FONSECA, G. L. - MODELAGEM TRIDIMENSIONAL DO CAMPUS PAMPULHA DA UFMG – UMA PROPOSTA EXPLORATÓRIA UTILIZANDO A FERRAMENTA GOOGLE SKETCHUP - 2007 <http://www.csr.ufmg.br/geoprocessamento/publicacoes/GizelleLira.pdf>. Acessado em maio/2018
2017. GAMEBRYO <http://www.gamebryo.com>. Acessado em dezembro/2017.



2017. GET TEDDY – Puzzle Game <http://www.guaranapps.com>. Acessado em novembro 2017.
2018. GIMP DOCUMENTAÇÃO TEXTURAS [https://docs.gimp.org/2.8/pt\\_BR/gimp-concepts-patterns.html](https://docs.gimp.org/2.8/pt_BR/gimp-concepts-patterns.html) Acessado em maio/2018
2017. GODOT <https://GODOTengine.org>. Acessado em novembro/2017.
2017. GODOT SHOWCASE <https://GODOTengine.org/showcase/>. Acessado em novembro 2017.
2017. HITMAN GO <https://www.square-enix-montreal.com/#/hitmango/>. Acessado em novembro/2017.
- JAIR C. L. - ENGENHARIA DE SOFTWARE, 2000 <https://www.dimap.ufrn.br/~jair/ES/c2.html>. Acessado em dezembro/2017.
2017. JOGO FLAPPY BIRD <https://play.google.com/store/search?q=flappy%20bird>. Acessado em novembro/2017.
- LEWIS, M. e JACOBSON, J. Game engines in scienti\_c research - introduction. Communications of the ACM, 45(1):27{31, 2002. URL <citeseer.ist.psu.edu/lewis02game.html>.
2017. LÖVE ENGINE 2D <https://love2d.org>. Acessado em dezembro/2017.
- NEVES, D. E., SANTOS, L. G. N. O., SANTANA, R. C., e ISHITANI, L. Avaliação de jogos sérios casuais usando o método GameFlow, 2013.
2017. PROFESSOR DANIEL CIOLF PLATAFORMA UDEMY <https://www.udemy.com/user/daniel-henrique-ciolfi/>. Acessado em novembro/2017.
2017. PROFESSOR HUGO VASCONCELOS PLATAFORMA UDEMY <https://www.udemy.com/user/hugo-vasconcelos/>. Acessado em novembro/2017.
- REVISTABW. Interface Homem-Máquina: Conceitos Iniciais.Revista Brasileira de Web: Tecnologia. Disponível em <http://www.revistabw.com.br/revistabw/ihtm-conceitos-iniciais/>. Criado em: 06/07/2015. Última atualização: 15/10/2015. Visitado em novembro 2017.
2017. SPACE SHOT TUTORIAL UNITY <https://UNITY3d.com/pt/learn/tutorials/s/space-shooter-tutorial>. Acessado em novembro/2017.
2017. TOWNS OF TTHE DEAD [http://www.naphelia.com/games/towns\\_of\\_the\\_dead.html](http://www.naphelia.com/games/towns_of_the_dead.html). Acessado em novembro/2017.
2017. UNITY <https://UNITY3d.com/pt/>. Acessado em novembro/2017.
2018. UNITY DOCUMENTAÇÃO UI <https://docs.unity3d.com/2018.1/Documentation/Manual/UISystem.html> Acessado em maio/2018.
2018. UNITY NAVIGATION AND PATHFINDING <https://docs.unity3d.com/Manual/Navigation.html> Acessado em maio/2018

2018. UNITY PLATAFORMS <https://unity3d.com/pt/unity> . Acessado em junho/2018.

2017. UNITY SHOWCASE <https://UNITY3d.com/pt/showcase/gallery/games?platform=android&genre=&gameType=t-all#gamesstart>. Acessado em novembro/2017.

2017. UNREAL ENGINE <https://www.unrealengine.com/>. Acessado em novembro/2017.

2017. UNREAL ENGINE 4 DEMO <https://play.google.com/store/apps/details?id=com.SAGames.VREnvironment>. Acessado em novembro/2017.

2017. UNREAL ENGINE SHOWCASE <https://www.unrealengine.com/en-US/showcase/category/mobile>. Acessado em novembro/2017.

2017. YOYO GAMES – GAME MAKER <https://www.yoyogames.com>. Acessado em dezembro/2017.