Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

LEONARDO DE SOUZA

CRIAÇÃO DE UM JOGO MOBILE E DESKTOP UTILIZANDO A FERRAMENTA UNITY

Assis/SP
2018

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

LEONARDO DE SOUZA

CRIAÇÃO DE JOGO MOBILE E DESKTOP UTILIZANDO A FERRAMENTA UNIY

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Municipal de Ensino Superior de Assis – IMESA e da Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando(a): Leonardo de Souza
Orientador(a): Célio Desiró

Assis/SP
2018

FICHA CATALOGRÁFICA

S729c SOUZA, Leonardo de
Criação de jogo mobile e desktop utilizando a ferramenta Unity / Leonardo de Souza. – Assis, 2018.

38p.

Trabalho de conclusão do curso (Análise e Desenvolvimento de Sistemas). – Fundação Educacional do Município de Assis-FEMA

Orientador: Esp. Célio Desiró

1.Jogos 2.Unity 3.Mobile-Jogos

CDD 005.11

CRIAÇÃO DE UM JOGO MOBILE E DESKTOP UTILIZANDO A FERRAMENTA UNITY

LEONARDO DE SOUZA

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

Orientador: Prof. Célio Desiró

Examinador: Prof. Dr. Osmar Aparecido Machado

Assis/SP
2018

DEDICATÓRIA

Dedico este trabalho para minha família e a todos que me apoiaram, inclusive meu orientador que me orientou na criação e desenvolvimento do projeto, e pelo grande conhecimento que recebi durante todo o curso.

AGRADECIMENTOS

Agradeço a Deus por estar presente e ajudar nas horas difíceis.

Ao meu orientador Célio Desiró, pela paciência e atenção que foi essencial para a finalização do projeto.

A meus familiares que sempre me apoiaram e me estimularam a não desistir de forma alguma. Em especial minha mãe que me ajudou a ir para faculdade independente da situação.

Aos meus amigos que me acompanharam nessa longa etapa da Faculdade, que sempre estiveram ao meu lado nos momentos bons e ruins e sempre incentivando para eu continuar e não desistir.

Enfim, aos Mestres e Doutores do curso de Análise e Desenvolvimento de Sistemas que sempre se preocuparam em passar aos alunos seus conhecimentos, e que sempre demonstraram atenção e força de vontade para passar o conhecimento em sala de aula, conhecimento esse que abrirão portas no mercado de trabalho, sendo um forte ponto de positivismo em nosso currículo.

RESUMO

Este trabalho descreve o estudo da *game engine Unity*, seu funcionamento e importância na indústria de *games*.

Unity é uma ferramenta muito usada atualmente, principalmente por oferecer suporte a diversas plataformas de distribuição de software produzido com ela e fornecer suporte para a criação desde pequenos jogos por uma pessoa ou pequenos grupos até a criação de jogos de grande porte feitos por estúdios renomados e consolidados na indústria.

O objetivo final do trabalho é a criação de um jogo utilizando a ferramenta *Unity*, para dispositivos móveis e *desktops* com intenção de adquirir experiência na área de desenvolvimento de jogos.

Como resultado obteve-se a criação de um jogo em 2D do gênero plataforma, utilizando a ferramenta *Unity*.

Concluiu-se que é possível criar um jogo com certa facilidade para o mercado de *games* e o aprimoramento do projeto pode propiciar a criação de um estúdio de jogos próprio ou ingressar em uma grande empresa na área de criação de jogos.

Palavras-chave: 1.*Unity***;** 2.*Games***;** 3.Desenvolvimento de jogos**.**

ABSTRACT

This work describes the study of the Unity game engine, its operation and importance in the games industry.

Unity is a widely used tool, mainly for supporting various software distribution platforms produced with it and providing support for small game creation by one person or small groups to the creation of large games made by renowned studios and industry.

The ultimate goal of the work is to create a game using the Unity tool for mobile devices and desktops with the intention of gaining experience in the area of game development.

As a result we obtained the creation of a 2D game of the platform genre, using the Unity tool.

It was concluded that it is possible to create a game with a certain ease for the gaming market and the improvement of the project may lead to the creation of a gaming studio of its own or to join a large company in the area of game creation.

Keywords: 1.Unity; 2.Games; 3.Game Development.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Jogos *indies*, como também são chamados os jogos independentes, cresceram muito nos últimos anos. Se antigamente eles eram conhecidos como “produtos de segunda” e jogos de baixo orçamento, hoje são referência na indústria de games e também na forma como a relação comercial funciona.

1.1 OBJETIVOS

O objetivo deste projeto é desenvolver um jogo 2D, do gênero plataforma, no qual possa ser aplicado o conhecimento básico no desenvolvimento de jogos.

Além disso, pretende-se que a documentação do trabalho possa inspirar outras pessoas na criação de novos projetos e que o jogo produzido possa ser estudado e aprimorado, uma vez que o código fonte será *open-source* e estará disponível.

1.2 JUSTIFICATIVAS

A produção de jogos digitais no Brasil vem aumentando gradativamente nos últimos anos, gerando notoriedade gerando um mercado promissor.

A geração que cresceu jogando, assim como este autor, agora está produzindo os seus próprios jogos para a geração que está começando, uma vez que as ferramentas estão facilitando o desenvolvimento, tanto com relação à produção da arte quanto da programação, abrindo ainda mais a oportunidade de se inserir no mercado de jogos.

Portanto, a intenção do jogo é motivar outras pessoas a produzir novos projetos, aprimorando sempre.

1.3 MOTIVAÇÕES

Sempre gostei de *games* e seu funcionamento e espero ingressar nessa indústria após o término do curso. Assim, o desenvolvimento de um trabalho sobre esse tema é um incentivo e um desafio para alcançar esse objetivo.

1.4 PÚBLICO ALVO

O jogo destina-se a jogadores com faixa etária bem variável, desde adolescentes até adultos, que não são jogadores assíduos de *games* e que estão à procura de um jogo dinâmico e relativamente rápido, que seja capaz de proporcionar momentos de diversão em uma partida “solo”.

1.5 ESTRUTURA DO TRABALHO

O primeiro capítulo apresenta introdução, o objetivo, as motivações, o público alvo e a justificativa para o desenvolvimento deste trabalho.

O segundo capítulo introduz ao leitor sobre o mercado de games no Brasil.

O terceiro capítulo apresenta a fundamentação teórica deste trabalho.

O quarto capítulo apresenta as ferramentas utilizadas.

O quinto capítulo apresenta o desenvolvimento do projeto.

O sexto capítulo apresenta a arte do projeto.

O sétimo capítulo apresenta as considerações finais e projetos futuros.

2 MERCADO DE GAMES

Com 66,3 milhões de *players* e uma movimentação de US$ 1,3 bilhão em 2017, o Brasil é o principal mercado de jogos da América Latina e o décimo terceiro no ranking mundial, conforme levantamento realizado pela *Newzoo*. Nessa pesquisa, o perfil do jogador vem mudando ao longo dos anos, apresentando crescimento no número de jogadores, que hoje já são 41% do total. (DINO, 2018)

Com o avanço anual do setor, a promoção e a distribuição dos jogos tornam-se desafios tanto para empresas brasileiras quanto para as internacionais, que precisam se especializar cada vez mais em marketing e novos formatos de divulgação para garantir o sucesso e a adesão das pessoas aos games. Afinal, com a facilidade na produção - que dá oportunidade para pequenos e médios desenvolvedores criarem e lançarem seus jogos - houve o aumento na concorrência. (DINO, 2018)

Com o avanço das possibilidades, o setor também passa por mais uma mudança: a relação dos pais com os jogos eletrônicos. A profissionalização da área faz com que os games sejam vistos de maneira positiva pelos familiares. Conforme apontado pela Game Brasil 2017, realizada pela Sioux, Blend New Research e ESPM, 65% dos pais acham que, se usados de forma moderada, os games podem ajudar na construção de perfil e no desenvolvimento de raciocínio lógico.(DINO, 2018)

De acordo com pesquisa realizada pela empresa Homo Ludens, o mercado de jogos eletrônicos cresceu em todas as cinco regiões do país entre 2013 a 2018. Os dados preliminares fazem parte do 2º Censo da Indústria Brasileira de Jogos Digitais, que foi apresentado pelo Ministério da Cultura durante a feira Brazil's Independent Games Festival (BIG Festival) de 2018, em São Paulo. (Homo Ludens, 2018)

O estudo revela que nos últimos cinco anos o número de estúdios de

desenvolvimento de games no Brasil passou de 142 para 375 e que somente nos últimos dois anos foram produzidos 1.718 jogos no país, 43% deles desenvolvidos para dispositivos móveis, 24% para computadores, 10% para plataformas de realidade virtual e realidade virtual aumentada e apenas 5% para consoles de videogame. Também foi revelado que deste total, 874 jogos foram de conteúdo educativos e 785 voltados ao entretenimento. (Homo Ludens, 2018)

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são tratados conceitos básicos sobre *engines*, que servirão de fundamentação para os capítulos posteriores.

3.1 ENGINES

Da mesma maneira que softwares como o Microsoft Visual Studio e Eclipse são utilizados para a produção de sistemas, existem diversas *frameworks* voltadas para a produção de *games*.

Uma *engine* é uma biblioteca, um pacote de funcionalidades que são disponibilizadas para facilitar o desenvolvimento de um jogo e impedir que sua criação tenha que ser feita do zero (NILTON, 2011).

Uma *engine* deve ser capaz de habilitar o usuário a programar o funcionamento do jogo e possibilitar o manuseio e aplicação de recursos gráficos e sonoros, assim como simulações de física, e outros recursos que possam ser adicionados ao produto final. Alguns exemplos de *engines* que estão em constante crescimento são *Unity*, *CryEngine*, *Unreal Engine 4*, *Source Engine* etc.(NILTON, 2011)

3.1.1 Gêneros

Existem inúmeros gêneros de jogos e a cada dia mais gêneros e tipos de jogos são pensados e desenvolvidos, além de que existem jogos que englobam diversos gêneros para uma experiência mais completa e envolvente, os gêneros dos jogos não limitam o que um jogo pode oferecer, mas sim indicam ao consumidor o que esperar do mesmo.(GAMEHALL, 2018)

A seguir serão descritos alguns dos gêneros de videogames mais comuns, não serão apresentados gêneros de nicho e específicos, mas sim uma visão mais ampla dos jogos que estão sendo produzidos na atualidade.(GAMEHALL, 2018)

Os gêneros mais comuns são os seguintes:

- *Puzzles*(Quebra-cabeças): Jogos que envolvem a resolução de quebra-cabeças e desafios, geralmente envolvendo coordenação motora e destreza com os controles ou pensamento lógico e solução de problemas. Exemplos: Tetris.(GAMEHALL, 2018)

- *Shooters*(Tiro): Jogos que envolvem o jogador atirando contra oponentes, depende do tempo de reação e precisão do jogador para alcançar seus objetivos, existem dezenas de subgêneros. Exemplos: Doom, Counter Strike, Battefield.(GAMEHALL, 2018)

- Plataformas: Jogos em que o jogador atravessa um nível que é constituído por plataformas, geralmente o objetivo é atravessar um nível de seu começo ao fim passando por diversos obstáculos. Exemplos: Sonic, Super Mario World.(GAMEHALL, 2018)

- RPGs(*Role Playing Game*): Um dos gêneros mais abundantes e de maior sucesso, jogos onde o jogador assume o controle de um ou mais personagens e contam uma historia através do jogo. Combate, progressão de personagem e enredo envolvente são comuns a esse gênero que pode ser para um jogador ou para milhões online. Exemplos: Jogos da franquia The Elder Scrolls, Fallout, Legend of Zelda, World of Warcraft.(GAMEHALL, 2018)

- Simuladores: Jogos que buscam simular algo do mundo real, gênero extremamente abrangente, que varia desde jogos de futebol até simuladores de empilhadeiras. Exemplos: FIFA, PES, The Sims, Cities Skylines.(GAMEHALL, 2018)

- Corrida: Jogos de corrida, normalmente de carros e motos, porém o limite é a imaginação dos desenvolvedores. Exemplos: Need For Speed, Mario Kart.(GAMEHALL, 2018)

- Ação/Aventura: Jogos de ação e/ou aventura, que combinam combate, resolução de problemas e quebra-cabeças e destreza com os controles. Exemplos: Tomb Raider, Uncharted, Assassins Creed.(GAMEHALL, 2018)

- Luta: jogos onde a ênfase é o combate podem variar entre Arena Fighters, Spectacle Fighters, Beat‟em Up e muitos outros. Exemplos: Devil May Cry, Mortal Kombat, God of War.(GAMEHALL, 2018)

- Estratégia: Jogos de estratégia variam entre tempo real e por turnos. São jogos onde conhecimentos, estratégias e planejamento são essenciais para a vitória, podendo ser jogados em tempo real ou por turnos. Exemplos: Franquia Sid Meyer, Age of Empires, Starcraft II.(GAMEHALL, 2018)

Esses são apenas alguns gêneros de *games* que existem e cada um desses se divide em diversos outros, levando em consideração o estilo de arte, tipo de controles, objetivos, temática, etc.

3.1.2 Plataformas

Games estão disponíveis em diversas plataformas, como as demonstradas a seguir.

- Dispositivos móveis – Smartphones, tablets. Consoles – Aparelhos que se conectam a uma TV ou monitor, como Playstation, Xbox, Wii. E aparelhos que já vem com uma tela, handhelds são chamados os consoles portáteis como o Nintendo DS e o PSVITA.

- *Arcades* – Máquinas eletrônicas que contem apenas um jogo e possuem um SO e sistema de controles próprio, essas máquinas são comumente encontradas em shoppings pelo Brasil.

- Computadores pessoais – PCs comuns com sistemas operacionais Windows, Mac ou baseados em Linux.

4 FERRAMENTAS UTILIZADAS

4.1 UNITY 3D

Este projeto utilizará a ferramenta *Unity3D*, que é um framework completo para a criação de jogos, que consiste em sua versão grátis ou paga e permite que o usuário criar seus jogos para diversas plataformas, como Android, Windows Phone, IOS, Web e PCs.(DIAS, 2017)

Outra vantagem da ferramenta é o suporte aos mais diversos tipos de arquivos de imagem, vídeo e som, assim como integração com sistemas de anúncios para a monetização dos jogos criados, permitindo aumentar os ganhos dos desenvolvedores assim como da própria empresa desenvolvedora da Unity. (DIAS, 2017)

Esta ferramenta possui 45% do mercado de game *engines* e tem mais de quatro milhões de desenvolvedores registrados para utilizar a ferramenta que oferece suporte para a criação de jogos 2D e 3D de todos os gêneros imagináveis. (DIAS, 2017)

Inicialmente a *Unity* foi usada por desenvolvedores independentes e pequenas empresas, porém nos últimos anos essa ferramenta foi utilizada para a criação de projetos de grande porte e com aclamação de críticos e fãs, jogos feitos com a *Unity* incluem:

Os jogos criados na *Unity* podem ser comercializados sem a necessidade de adquirir a versão paga do software, porém ao alcançar renda superior a $100.000,00 em ano fiscal é necessário adquirir a versão paga para continuar a disponibilizar o jogo. (DIAS, 2017)

A versão gratuita e paga não tem diferenças drásticas em desempenho e funcionalidade, o diferencial da versão paga é o suporte a projetos grandes e ferramentas avançadas de teste e analise, o preço é de $25,00/mês segundo o site oficial. (UNITY, 2018)

Apesar de que o desenvolvimento de jogos é o foco da ferramenta ela pode ser usada para a criação de diversos outros softwares como livros interativos, simulações, artes gráficas, treinamentos virtuais e outros. (DIAS, 2017)

4.1.1 Funcionamento da UNITY

Todo o processo de criação de um jogo é baseado no princípio de orientação a objetos, cada objeto representa um elemento do jogo e vários são instanciados no decorrer da execução. (SCOLATISCI, 2015)

Uma das características mais marcantes da *Unity* é sua interface fácil de usar e configurável, permitindo que tanto usuários iniciantes como experientes possam usufruir de suas funcionalidades. (SCOLATISCI, 2015)

Será utilizado um projeto pronto de um tutorial para exemplificar as diferences funcionalidades da interface e o funcionamento da *Unity*.

A figura a seguir mostra a interface da *Unity* em sua forma padrão, no decorrer deste trabalho os principais aspectos serão analisados e estudados.

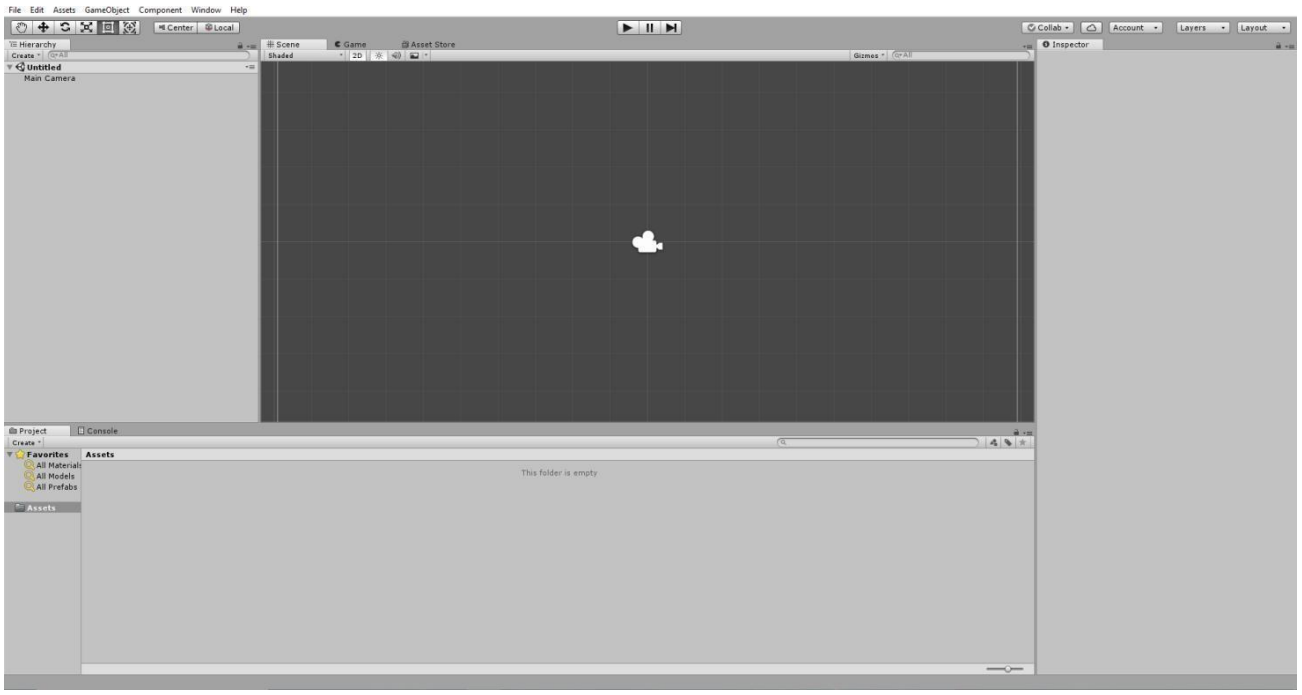

Figura 1: Tela do Projeto

O conceito principal da organização dos jogos produzidos com a *Unity* é o de cenas, cada cena pode conter inúmeros objetos e também é possível criar um jogo utilizando apenas uma cena. (SCOLATISCI, 2015)

Essas cenas contem os objetos criados pelo desenvolvedor e são o ambiente de

execução do programa, uma das vantagens de se usar mais de uma cena é que os objetos só são carregados na cena que está ativa, então quando ocorre a mudança de cenas os recursos utilizados pela cena anterior são liberados tornando a execução do jogo mais eficiente. (SCOLATISCI, 2015)

A figura abaixo mostra a tela *scene* onde o desenvolvedor pode modificar os objetos diretamente na cena.

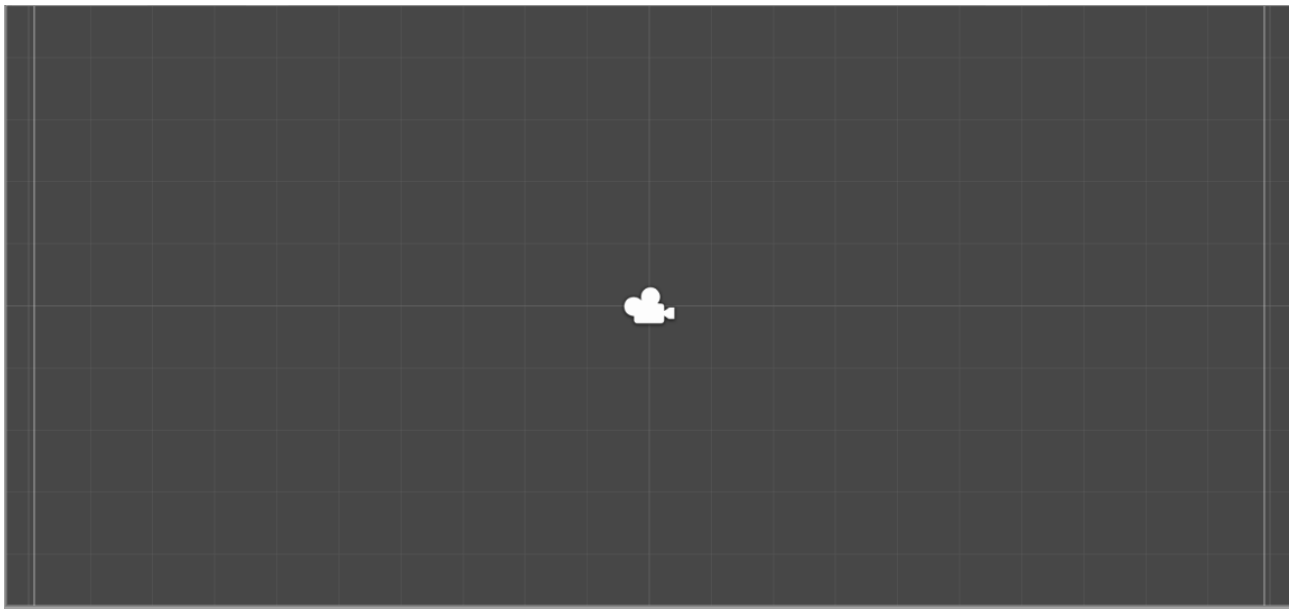

Figura 2: Tela de cena

Essa parte da interface da *Unity* mostra todos os objetos que compõem a cena, até mesmo objetos que são invisíveis e desconhecidos do jogador.

Para mostrar como o projeto final está sendo feito a *Unity* também tem a tela *Game* que mostra como a cena será mostrada para o jogador.

Durante o teste do jogo é possível adicionar ou remover objetos e alterar os valores de seus atributos e observar os resultados em tempo real.(SCOLATISCI, 2015)

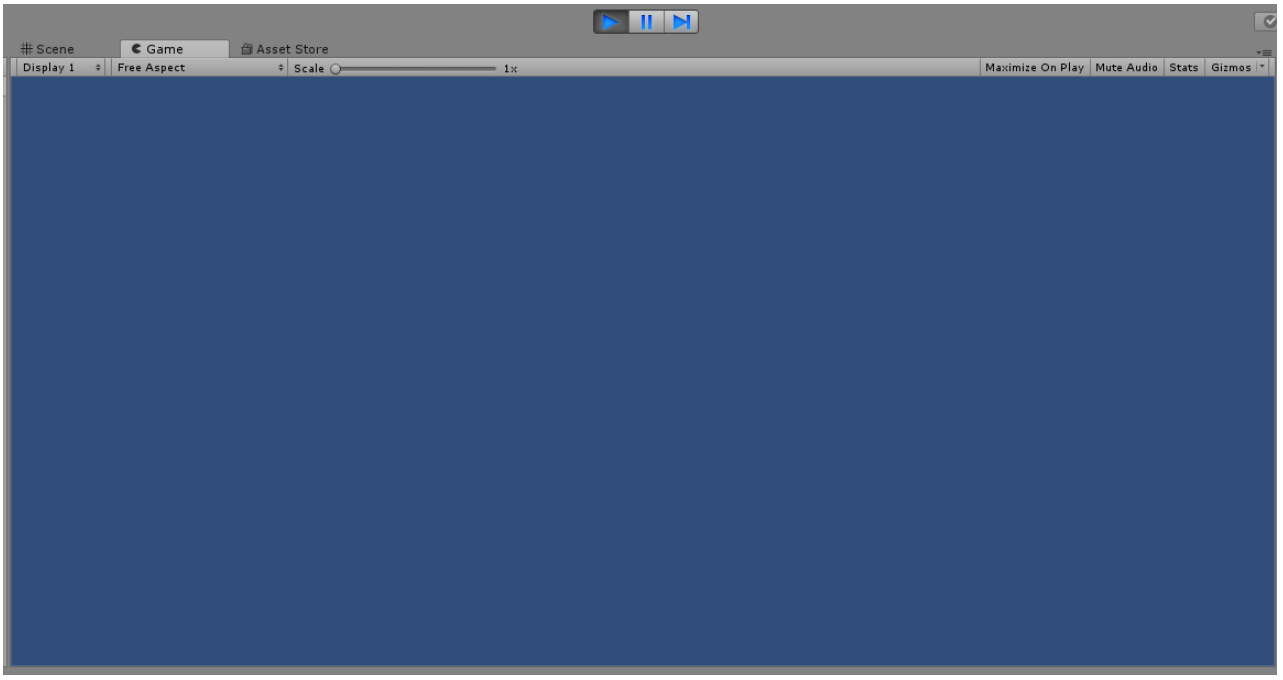

Figura 3: Tela de Testes

Cada cena tem uma hierarquia de objetos, essa hierarquia contem os objetos que estão presentes no inicio da execução.(SCOLATISCI, 2015)

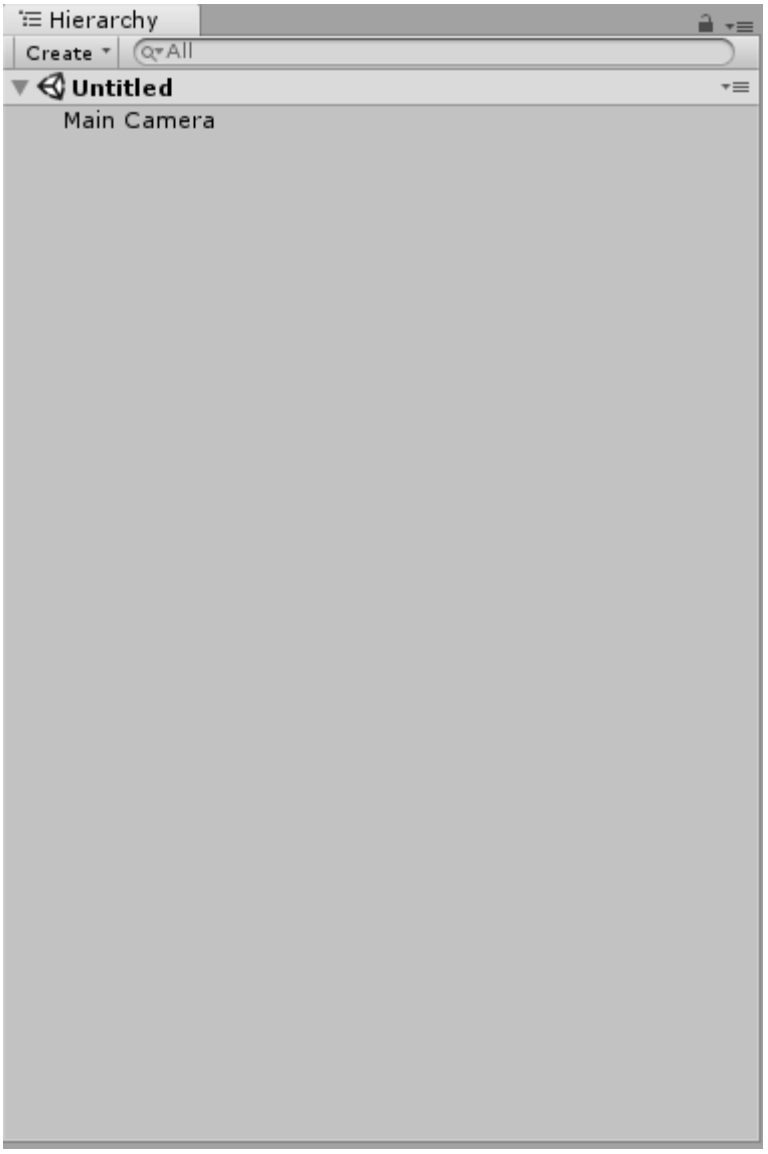

Figura 4: Hierarquia

Durante o teste do jogo, mostrado na figura acima, os objetos que são instanciados durante a execução do código são mostrados na tela de *hierarchy* e são vitais para o desenvolvedor entender como cada objeto interage com os outros na cena.

(SCOLATISCI, 2015)

Para permitir o acesso rápido e fácil a todos os arquivos que compõe o seu projeto na *Unity* existe uma tela de projeto que mostra todos os diretórios e arquivos contidos neles, podendo simplesmente arrastar os arquivos com o mouse para a cena atual e eles serão criados automaticamente. (SCOLATISCI, 2015)

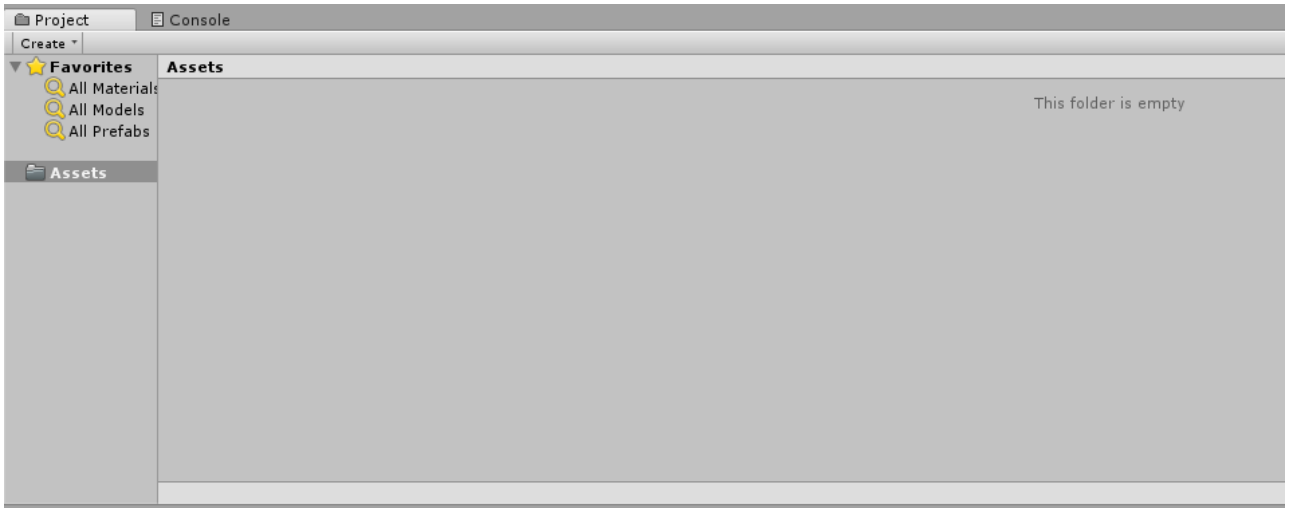

Figura 5: Tela de projeto

Para cada objeto criado é possível configurar seus atributos e elementos através da tela *Inspector* que permite adicionar e modificar com simplicidade elementos e atributos a qualquer objeto selecionado. (SCOLATISCI, 2015)

Variáveis publicas nos códigos também podem ser alteradas através dessa tela.

Esses códigos escritos em C# ou JavaScript são a principal parte dos jogos, pois eles ditam o comportamento dos objetos e suas interações com o jogador e outros objetos durante a execução. (SCOLATISCI, 2015)

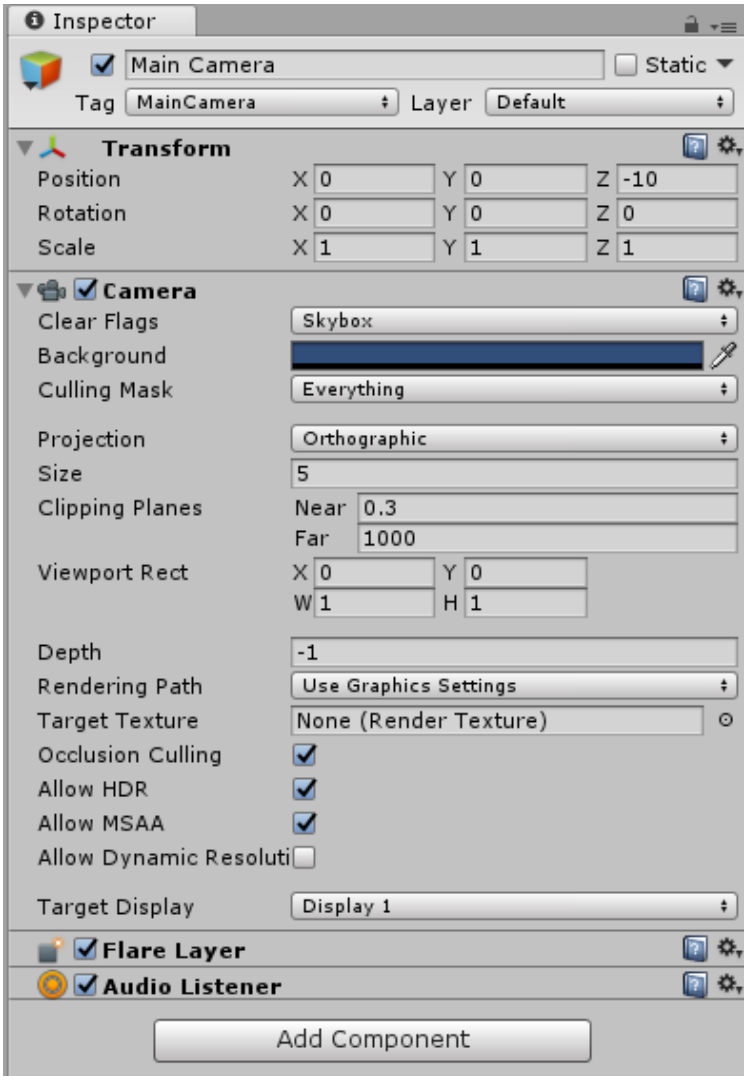

Figura 6: Inspetor de Objetos

A *Unity* oferece integrado a sua ferramenta o editor de textos MonoDevelop, que é utilizado para escrever os scripts e oferece funcionalidades como completar código e bibliotecas próprias da *Unity* como padrão. (SCOLATISCI, 2015)

Essas várias bibliotecas contem funções e métodos que simulam física, leitura de entradas e interação entre objetos do jogo, além de inúmeras outras funcionalidades próprias para o desenvolvimento de jogos. (SCOLATISCI, 2015)

Uma das vantagens da *Unity* é sua portabilidade, podendo desenvolver jogos para PC, Mac, Linux, Android, IOS, Web browsers e consoles, todas essas opções são definidas na tela chamada de Build Settings. (SCOLATISCI, 2015)

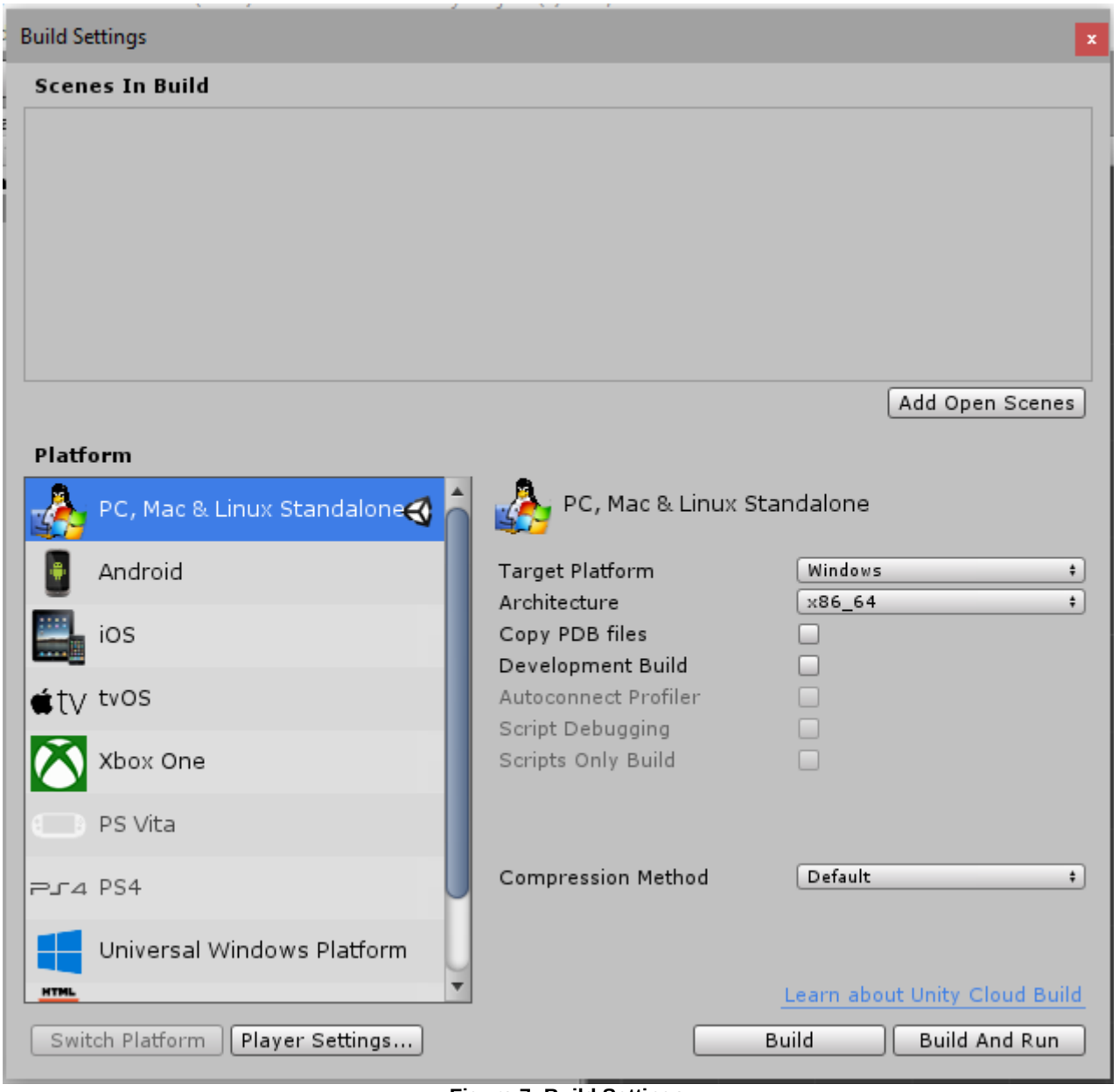

Figura 7: Build Settings

4.1.2 Suporte a Unity

Após essas configurações estiverem configuradas pelo desenvolvedor a *Unity* cria automaticamente um arquivo compatível com a plataforma escolhida, pronto para a distribuição. (SCOLATISCI, 2015)

A *Unity* oferece a todos os usuários uma loja virtual na qual qualquer usuário pode disponibilizar ou vender suas criações como objetos de um jogo ou até mesmo códigos fonte. (SCOLATISCI, 2015)

Nessa loja virtual também são encontradas os tutoriais da própria *Unity* e pacotes com recursos para novos desenvolvedores gratuitamente.

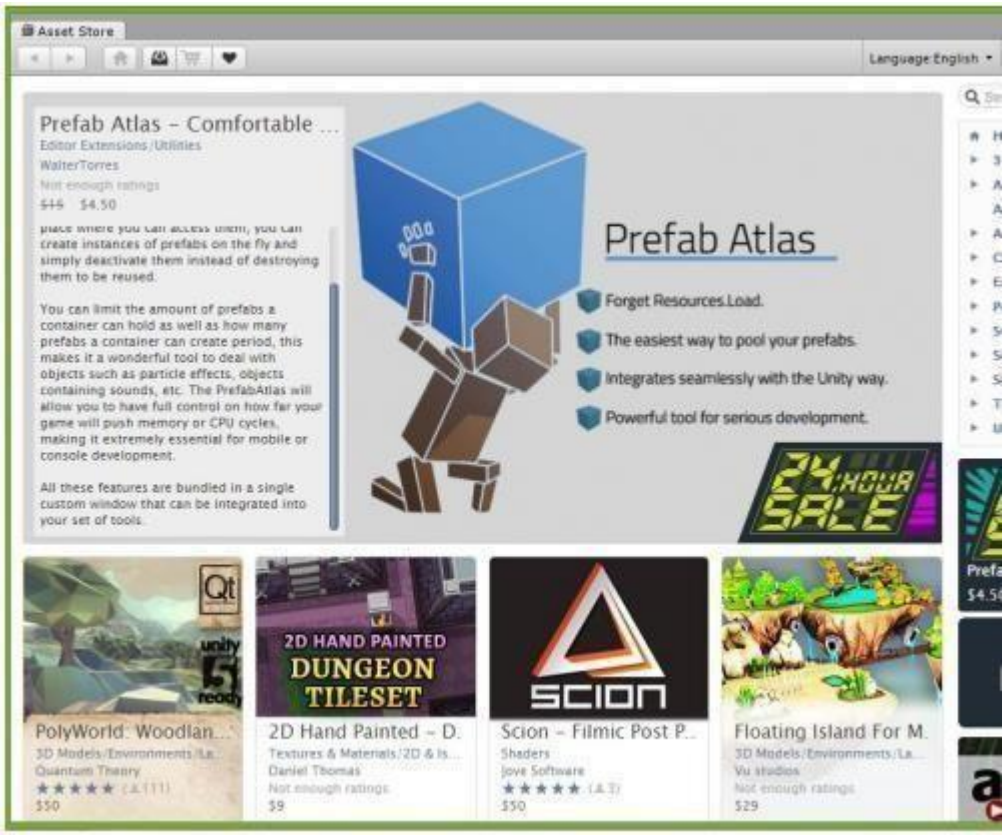

Figura 8: Loja Virtual

5 DESENVOLVIMENTO DO PROJETO

Este documento tem o intuito de demonstrar aspectos técnicos, artísticos e narrativos do jogo Wild Jewel Game. Este documento apresenta o enredo do jogo, a mecânica de jogo, seu objetivo, aspectos de jogabilidade e ferramentas de desenvolvimento. Com estes pontos é possível dar sequência ao processo de produção e desenvolvimento do jogo.

5.1 RESUMO DA HISTÓRIA

Um aventureiro está atrás de um tesouro à muito tempo perdido, que ao longo do caminho vai deixando joias criando um rastro que faz o aventureiro correr ainda mais pela floresta atrás do tesouro.
Mas durante sua corrida ele se depara com criaturas da floresta, que então devera enfrenta-las para sobreviver.

5.2 VISÃO TÉCNICA DO JOGO

O aventureiro tem como objetivo pegar as joias do cenário até o fim da cena para acumular pontos e eliminando as criaturas que estiverem na cena.

5.3 GÊNERO

O jogo tem o gênero plataforma, com um cenário bastante variado com plataformas altas e baixas, e uma diversidade de criaturas para enfrentar.
O jogo é baseado em jogos como Super Mario Bros, Sonic e Castlevania;

5.4 PUBLICO ALVO

O jogo destina-se a jogadores com faixa etária bem variável, que estão à procura de um jogo dinâmico e rápido, que seja capaz de proporcionar uma diversão rápida.

5.5 FLUXO DO JOGO

O fluxo do jogo é baseado no percurso que os heróis vão percorrer para conseguir conquistar os pontos para a vitória.

5.6 TEMPO DE JOGO:

- Primeira Fase: de 2 a 5 minutos
- Segunda Fase: de 2 a 5 minutos
- Terceira Fase: de 2 a 5 minutos

5.7 CONTROLES

Os comandos de controle para o jogo são:

- Setas esquerda e direita - Movimenta o personagem;
- Tecla de espaço - Pula o personagem;

5.8 MECÂNICA BÁSICA

O jogo tem como mecânica básica fazer com que o personagem ande acima da plataforma, devendo se esquivar das criaturas ao longo do percurso, coletando as joias acumulando pontos.

5.9 CONFIGURAÇÃO MÍNIMA DE HARDWARE

- Windows 7 ou superior;
- Processador 1GHz;
- Memória RAM Mínima 2GB;
- Placa de Vídeo 128mb;
- Mouse e Teclado;
- Som;

6 ARTE DO PROJETO

Neste capítulo serão mostradas as artes desenvolvidas para o jogo.

6.1 PERSONAGEM

Figura 9: Personagem principal

6.2 TEXTURAS DO CENÁRIO

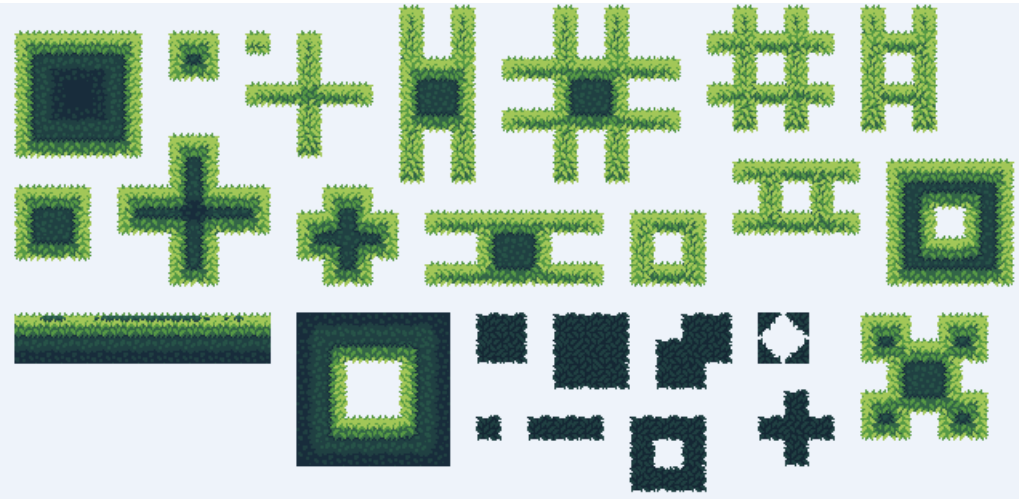

Figura 10: Texturas

6.3 VISÃO GERAL DO CENÁRIO

Figura 11: Visão do cenário

6.4 CÓDIGOS FONTE DO PROJETO

```
1  using UnityEngine;
2  using System.Collections;
3
4  public class PlayerController : MonoBehaviour {
5
6      public float speed;
7      public float jumpForce;
8
9      private Rigidbody2D rb;
10     private bool facingRight = true;
11     private bool jump = false;
12     private Animator anim;
13     private bool noChao = false;
14     private Transform groundCheck;
15
16     // Use this for initialization
17     void Start () {
18
19         rb = gameObject.GetComponent<Rigidbody2D>();
20         anim = gameObject.GetComponent<Animator>();
21         groundCheck =  gameObject.transform.Find("GroundCheck");
22
23     }
24
25     // Update is called once per frame
26     void Update () {
27
28         noChao = Physics2D.Linecast(transform.position, groundCheck.position, 1 << LayerMask.NameToLayer("Ground"));
29
30         if(Input.GetButtonDown("Jump") && noChao)
31         {
32             jump = true;
33             anim.SetTrigger("Pulou");
34         }
```

Figura 12: Código do personagem

```
    void FixedUpdate()
    {
        float h = Input.GetAxisRaw("Horizontal");
        anim.SetFloat("Velocidade", Mathf.Abs(h));

        rb.velocity = new Vector2(h * speed, rb.velocity.y);

        if(h > 0 && !facingRight)
        {
            Flip();
        }
        else if(h < 0 && facingRight)
        {
            Flip();
        }

        if (jump)
        {
            rb.AddForce(new Vector2(0, jumpForce));
            jump = false;
        }

    }

    void Flip()
    {
        facingRight = !facingRight;

        Vector3 theScale = transform.localScale;
        theScale.x *= -1;
        transform.localScale = theScale;
    }

}
```

Figura 13: Continuação do Código personagem

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

public class CristalScript : MonoBehaviour {

    Animator anim;
    CircleCollider2D col;

    AudioSource audioS;
    void Start () {

        anim = gameObject.GetComponent<Animator>();
        col = gameObject.GetComponent<CircleCollider2D>();
        audioS = gameObject.GetComponent<AudioSource>();

    }


    void Update () {

    }

    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.gameObject.CompareTag("Player"))
        {
            audioS.Play();
            GameManager.gm.SetCristal(1);
            col.enabled = false;
            anim.SetTrigger("Coletando");
            Destroy(gameObject, 0.150f);
        }
    }
}
```

Figura 14: Código do cristal

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

public class GameManager : MonoBehaviour {

    public static GameManager gm;

    public int vidas = 2;
    public int cristais = 0;


    void Awake () {
        |
        if (gm == null)
        {
            gm = this;
            DontDestroyOnLoad(gameObject);
        }

        else
            Destroy(gameObject);

    }

    void Start()
    {
        AtualizaHud();
    }


    void Update () {

    }

    public void SetVidas(int vida)
    {
        vidas += vida;
        if(vidas >= 0)
        AtualizaHud();
    }

    public int GetVidas()
    {
        return vidas;
    }

```

Figura 15: Código do *GameManager*

```
    public void SetCristal(int cristal)
    {
        cristais += cristal;
        if(cristais >= 30)
        {
            cristais = 0;
            vidas += 1;
        }

        AtualizaHud();
    }

    public void AtualizaHud()
    {
        GameObject.Find ("VidasText").GetComponent<Text> ().text = vidas.ToString ();
        GameObject.Find("CristalText").GetComponent<Text>().text = cristais.ToString();
    }


}

```

Figura 16: Continuação do código *gamemanager*

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

public class Inimigo : MonoBehaviour {

    public float velocidade;
    public bool direcao;
    public float duracaoDirecao;
    public float speed;
    public float jumpForce = 100;

    private float tempoNaDirecao;
    private Animator animator;

    AudioSource audioS;

    void Start () {

        animator = gameObject.GetComponent<Animator>();
        audioS = gameObject.GetComponent<AudioSource>();

    }


    void Update () {

        if (direcao) {
            transform.eulerAngles = new Vector2(0, 0);
        } else {
            transform.eulerAngles = new Vector2(0, 180);
        }
        transform.Translate(Vector2.right * velocidade * Time.deltaTime);

        tempoNaDirecao += Time.deltaTime;
        if (tempoNaDirecao >= duracaoDirecao) {
            tempoNaDirecao = 0;
            direcao = !direcao;
        }

    }

```

Figura 17: Código do Inimigo

```
    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.gameObject.CompareTag("Player"))
        {
            audioS.Play();
            BoxCollider2D[] boxes = gameObject.GetComponents<BoxCollider2D>();
            foreach(BoxCollider2D box in boxes)
            {
                box.enabled = false;
            }

            other.GetComponent<Rigidbody2D>().velocity = new Vector2(0, 0);
            other.gameObject.GetComponent<Rigidbody2D>().AddForce(new Vector2(0, jumpForce));

            speed = 0;
            transform.Rotate(new Vector3(0, 0, -180));
            Destroy(gameObject, 3);
        }
    }

    void OnCollisionEnter2D(Collision2D other)
    {
        if (other.gameObject.CompareTag("Player"))
        {
            other.gameObject.GetComponent<PlayerLife>().PerdeVida();
        }
    }


}
```

Figura 18: Continuação do código do Inimigo

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class Menu : MonoBehaviour {
7
8
9     void Start () {
10
11
12     }
13
14
15     void Update () {
16
17     }
18
19     public void CarregaFase(int cena)
20     {
21
22         SceneManager.LoadScene(cena);
23     }
24
25     public void Sair()
26     {
27         Application.Quit();
28     }
29 }
```

Figura 19: Código do Menu

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class MusicPlayer : MonoBehaviour {
5
6     private static MusicPlayer mp;
7
8
9     void Awake () {
10
11         if(mp == null)
12         {
13             mp = this;
14             DontDestroyOnLoad(gameObject);
15         }
16
17         else
18         {
19             Destroy(gameObject);
20         }
21
22     }
23
24
25     void Update () {
26
27     }
28 }
```

Figura 20: Código do adicionador de músicas

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

public class PassaFase : MonoBehaviour {


    void Start () {

    }


    void Update () {

    }

    void OnTriggerEnter2D(Collider2D other)
    {
        if (other.gameObject.CompareTag("Player"))
        {
            SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
        }
    }
}
```

Figura 21: Código para passar de fase

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

public class PlayerLife : MonoBehaviour {

    Animator anim;
    bool vivo = true;

    public AudioClip deathSound;
    AudioSource audioS;

    void Start () {

        audioS = gameObject.GetComponent<AudioSource>();
        anim = gameObject.GetComponent<Animator>();
        GameManager.gm.AtualizaHud();

    }


    void Update () {

    }

    public void PerdeVida()
    {
        if (vivo)
        {
            audioS.clip = deathSound;
            audioS.Play();
            vivo = false;
            anim.SetTrigger("Morreu");
            GameManager.gm.SetVidas(-1);
            gameObject.GetComponent<PlayerController>().enabled = false;
        }

    }

```

Figura 22: Código da vida do personagem

```
42      public void Reset()
43      {
44          if(GameManager.gm.GetVidas() >= 0)
45          {
46              SceneManager.LoadScene(SceneManager.GetActiveScene().name);
47          }
48
49          else
50          {
51              GameManager.gm.vidas = 1;
52              GameManager.gm.cristais = 0;
53              SceneManager.LoadScene(3);
54          }
55      }
56  }
```

Figura 23: Continuação do código da vida do personagem

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class Fundo : MonoBehaviour {
7
8
9      void Start () {
10
11     }
12
13
14     void Update () {
15
16     }
17
18     void OnCollisionEnter2D(Collision2D other)
19     {
20         if (other.gameObject.CompareTag("Player"))
21         {
22             other.gameObject.GetComponent<PlayerLife>().PerdeVida();
23         }
24     }
25 }
```

Figura 24: Código de armadilha

7 CONSIDERAÇÕES FINAIS

No decorrer do projeto foi possível entender uma parte do processo da criação de um game e os aspectos artísticos e técnicos que são necessários para um produto final satisfatório.

Durante o desenvolvimento desse projeto o tema e gênero do jogo foram modificados inúmeras vezes, pois diferentemente de um sistema comercial não é claro que características o produto final deve ter, esse aspecto subjetivo do projeto causou certa confusão durante os trabalhos realizados.

Esse trabalho de conclusão de curso trouxe muitos conhecimentos sobre assuntos que não estão na grade curricular do curso e sobre uma indústria que vem crescendo a cada ano, trazendo conhecimentos que serão usados para ingressar na indústria de games.

O produto final, o jogo titulado “Wild Jewel Game”, apesar de possuir relativa simplicidade, teve importante papel como o produto do aprendizado de todas essas tecnologias e conceitos aprendidos durante a realização deste projeto.

7.1 PROJETOS FUTUROS

Aprimoramentos serão feitos ao jogo produzido para esse projeto e ele será distribuído para duas plataformas para avaliar a recepção do público.

Com os conhecimentos adquiridos no trabalho pretendo criar mais e mais jogos para construir meu conhecimento nessa área e ter mais chances de ingressar na indústria de games com a experiência adquirida.

REFERÊNCIAS

DINO. Mercado de games nacional é um mundo inexplorado de oportunidades. 5 mar. 2018. Disponível em: <https://www.terra.com.br/noticias/dino/mercado-de-games-nacional-e-um-mundo-inexplorado-de-oportunidades,5cd0affd43ee184903db06153df8dc31aomryyai.html>.Acesso em: 8 ago. 2018.

DIAS. Raphael. Unity guia completo sobre a engine. 30 jan. 2018. Disponível em: <https://producaodejogos.com/unity/>.Acesso em: 9 ago. 2018.

GAMEHALL. Mercado de games cresce em todo o Brasil, aponta 2° Censo de games. 3 jul. 2018. Disponível em: <https://jogos.uol.com.br/ultimas-noticias/2018/07/03/mercado-de-games-cresce-em-todo-brasil-aponta-2-censo-de-games.htm>.Acesso em: 8 ago. 2018.

KLEINA.O que é *engine* ou motor gráfico?. 22 mar. 2011. Disponível em: <https://www.tecmundo.com.br/video-game-e-jogos/9263-o-que-e-engine-ou-motor-grafico-.htm>.Acesso em: 8 ago. 2018.

SCOLATISCI. Claudio. Unity 2D Game Development Cookbook. 1ª Edição. Birmnham: Packt Publishing Ltd, 2015.

TOSCHI. Gabriel. Saiba sobre os gêneros de jogos nos quais games se classificam nos mais conhecidos jogos. 22 abr. 2012. Disponível em: <https://www.nintendoblast.com.br/2012/04/gamedev-conheca-os-tipos-de-jogos.html>.Acesso em: 8 ago. 2018.

UNITY. *Engines* de jogo – como funcionam?. 2018. Disponível em: <https://unity3d.com/pt/what-is-a-game-engine>.Acesso em: 8 ago. 2018.

UNITY. Sobre a Unity. 2018. Disponível em: <https://unity3d.com/pt/public-relations>.Acesso em: 8 ago. 2018.

UNITY. Manual Unity. 2018. Disponível em: <http://docs.unity3d.com/Manual/index.html>.Acesso em: 8 ago. 2018.