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

LINCOLN TONELO DE ALMEIDA

CRIAÇÃO DE UM JOGO PLATAFORMA DE AVENTURA EM 2D UTILIZANDO A ENGINE UNITY

Assis
2018

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

LINCOLN TONELO DE ALMEIDA

CRIAÇÃO DE UM JOGO PLATAFORMA DE AVENTURA EM 2D UTILIZANDO A ENGINE UNITY

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: Lincoln Tonelo de Almeida

Orientador: Célio Desiró

Assis
2018

FICHA CATALOGRAFICA

A447c ALMEIDA, Lincoln Tonelo

Criação de um jogo plataforma de aventura em 2D utilizando a engine unity / Lincoln Tonelo de Almeida. – Assis, 2018

39p.

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

Orientador: Esp. Célio Desiró

1.Unity 2.Jogos 3.Programação.

CDD 005.11

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

CRIAÇÃO DE UM JOGO PLATAFORMA DE AVENTURA EM 2D UTILIZANDO A ENGINE UNITY

LINCOLN TONELO DE ALMEIDA

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
2018

RESUMO

Este trabalho, tem o intuito de mostrar as etapas de criação de jogos, e a como é a utilização da ferramenta Unity, e sua importância no mercado de desenvolvimento de jogos digitais na atualidade.

A Unity atualmente, é uma das principais “Engines” para o mercado, tendo uma vasta possibilidade de plataformas (Playstation, Xbox, PC, etc.) e uma grande comunidade presente, ajudando novas pessoas que se interessam pela Unity, e também, a própria Unity, que tem bastantes recursos disponibilizados, aulas e alguns livros com níveis de complexidade de iniciante, até um veterano na área de desenvolvimento de jogos.

O objetivo do trabalho, é mostrar que qualquer pessoa, com um interesse nessa área, consegue desenvolver seus próprios jogos, e mostrar o potencial da Unity, no desenvolvimento de jogos em nível “Indie”.

Palavras-Chave: Unity, Jogos, Programação.

ABSTRACT

This paper aims to show the stages of game creation, and how is the use of the Unity tool, and its importance in the market of development of digital games nowadays.

Unity currently is one of the leading "Engines" for the market, having a wide range of platforms (Playstation, Xbox, PC, etc.) and a large community present, helping new people interested in Unity, as well as Unity itself, which has plenty of resources available, classes and some books with levels of complexity from beginner to veteran in the area of game development.

The purpose of the work is to show that anyone with an interest in this area can develop their own games, and show the potential of Unity in the development of games at the "Indie" level.

Keywords: Unity, Games, Programming.

Lista de Figuras

Figura 1 - Interface 19
Figura 2 - Project View 20
Figura 3 - Hierarchy View 21
Figura 4 - Scene View 22
Figura 5 - Game View 23
Figura 6 - Inspector View 24
Figura 7- Asset Store 25
Figura 8 - Cenário 1 26
Figura 9 - Cenário 2 27
Figura 10 - Criação do Menu 27
Figura 11 - Player 28
Figura 12 - Inimigo 1 28
Figura 13 - Inimigo 2 28
Figura 14 - Inimigo 3 28
Figura 15 - Scripts 29
Figura 16 - Variáveis do Script Player 30
Figura 17- Funções de movimentação e pulo 31
Figura 18 - Variáveis do Script dos inimigos 31
Figura 19 - Movimentação e o "raio de ataque" do inimgo 32
Figura 20 - Câmera Follow 32
Figura 21 - Gerenciador 33
Figura 22 - Menu Script 34
Figura 23 - Moeda 35
Figura 24 - Armadilha 36

Sumário

1. INTRODUÇÃO 9
   1.1. OBJETIVOS 9
   1.2. MOTIVAÇÔES 10
   1.3. PÚBLICO ALVO 10
   1.4. JUSTIFICATIVA 10
   1.5. ESTRUTURA DO TRABALHO 11
2. INTRODUÇÂO SOBRE JOGOS DIGITAIS 12
   2.1. HISTÓRIA DOS JOGOS DIGITAIS 12
   2.2. ESTRUTURA/CONCEITO DOS JOGOS DIGITAIS 13
      2.2.1. Gênero 13
      2.2.2. Plataformas 15
      2.2.3. Engines 16
      2.2.4. Modos de jogo 17
3. FERRAMENTAS UTILIZADAS 18
   3.1. SOFTWARES AUXILIARES 18
   3.2. UNITY3D 18
      3.2.1. Introdução à Unity 18
      3.2.2. Interface da Unity3D 19
      3.2.3. Project View 20
      3.2.4. Hierarchy View 21
      3.2.5. Scene View 22
      3.2.6. Game View 23
      3.2.7. Inspector View 24
      3.2.8. Asset Store 25
4. DESENVOLVIMENTO DO JOGO 26
   4.1. SOBRE O JOGO 26
   4.2. PERSONAGENS 28
   4.3. SCRIPTS 29
      4.3.1. Player 30
      4.3.2. Inimigo 31
      4.3.3. Câmera Follow 32
      4.3.4. Gerenciador de level 33
      4.3.5. Menu 34
      4.3.6. Moedas 35
      4.3.7. Armadilhas 36
   5. CONSIDERAÇÕES FINAIS 37
   REFERÊNCIAS 38

1. INTRODUÇÃO

Jogos digitais atualmente não são algo mais visto como “coisa de criança” ou apenas um simples entretenimento, é um mercado que rivaliza de igual para igual com a indústria da música e do cinema, e vem evoluindo ano após ano, novas tecnologias vem surgindo, e há quem diga que um dia, será considerado arte.

Pessoas de todas as partes estão deixando de serem só consumidores de jogos, e começando uma “carreira” de criadores de pequenos jogos (indies), é algo que vem se intensificando bastante ao longo do tempo, por apoio e “patrocínio” das duas maiores engines free de game designer, a “Unreal Engine” e a “Unity”, ambos tem um ótimo conteúdo para iniciantes, e uma comunidade ativa.

Foi dessa vontade de querer criar jogos, mais uma engine q facilitasse o desenvolvimento do game, que colaborou para escolha de criar um jogo para o TCC; A engine utilizada será a Unity, por ter maior facilidade e uma maior comunidade na área de jogos 2D, efeitos sonoros, sprites, cenários serão versões free, disponibilizadas pela própria comunidade, e a linguagem utilizada na criação dos scripts será a linguagem C#.

1.1. OBJETIVOS

O Principal objetivo, é construir um jogo do 0, listando algumas dificuldades no decorrer do desenvolvimento, e incentivar para quem sempre teve vontade de criar um jogo, mas não sabe por onde começar, que é possível criar um jogo simples, divertido e que não irá agradar só você, mas também, outras pessoas que compartilham do seu mesmo gosto.

E talvez um dia conseguir trabalhar nessa área que é bem nebulosa ainda no Brasil, por sofrer bastante preconceito e não ser levado à sério

1.2. MOTIVAÇÔES

Sempre me interessei por jogos digitais, e por tão grandiosos e complexos eles ficaram, envolvendo música, roteiristas, designers... E sempre quis entender como era o processo de criação de jogos, e se possível, aprender a criar eles.

1.3. PÚBLICO ALVO

Qualquer tipo de pessoa, de criança ao adulto, pessoas que se interessem em jogos do tipo ação/aventura em 2D.

1.4. JUSTIFICATIVA

Com uma crescente no mercado de jogos digitais em todo mundo, como jogos Triple A (jogos com grandes investimentos), muita gente desiste da área por achar muita elitizada, ou de grande complexidade, mas à contra partida disso, pequenas empresas de jogos estão surgindo, buscando espaço nesse concorrido mercado, porém, um novo nicho de jogos está pouco à pouco crescendo, que são os jogos indies, jogos de baixíssimo custo, que na sua maioria são criados por uma pequena equipe, ou até uma pessoa, e é nesse nicho que será realizado o jogo/TCC, utilizando poucos recursos, mas tentando fazer algo com qualidade.

1.5. ESTRUTURA DO TRABALHO

O Primeiro capítulo é sobre a introdução do projeto, motivações, justificativas, público alvo e objetivo TCC.

Segundo capítulo é um pequeno resumo da história dos jogos digitais, descrevendo sua evolução durante as décadas até os dias atuais.

Terceiro capítulo será descrito as ferramentas utilizadas no desenvolvimento do jogo, e uma breve introdução à engine Unity, mostrando suas ferramentas internas e seu funcionamento.

Quarto capítulo é o desenvolvimento do jogo, mostrando os scripts, progresso do jogo, cenários, sprites e sons.

2. INTRODUÇÂO SOBRE JOGOS DIGITAIS

2.1. HISTÓRIA DOS JOGOS DIGITAIS

A história dos videogames começa por volta da década de 50 quando programadores e professores começaram a desenvolver pequenos jogos, simulações e conceitos de inteligência artificial em suas pesquisas sobre computação, mas somente nos anos 70 com Nolan Bushnell os videogames começaram a ser vistos como forma de entretenimento comercial, mas não fez muito sucesso, devido à necessidade de possuir um computador para jogá-los.

A partir da década de 70 a indústria dos videogames foi evoluindo, com o surgimento das máquinas de “Arcade” e o Atari videogames se espalharam pelo mundo, mas sendo vistos pela maioria do público como brinquedos.

Em 1983 houve uma crise na indústria de videogames na América do Norte, onde muitas empresas declararam falência e houve a ‘‘invasão’’ das empresas Japonesas no cenário americano.

Até o final da década de 90 houve inúmeras inovações em relação aos videogames, incluindo o surgimento de gráficos em 3D e o surgimento de outra geração de jogos, juntamente com o lançamento do PlayStation 2 e outros consoles da época.

A partir dos anos 2000 a indústria dos videogames começou um crescimento estrondoso, com gráficos exuberantes, consoles e computadores ficando cada vez mais comuns, a popularização da internet e dos dispositivos moveis.

Hoje o hardware voltado para videogames, tanto consoles como PCs, são geralmente de alta tecnologia e liderando as inovações em muitos campos, além de infraestruturas como servidores para jogos online suportarem milhões de jogadores.

<https://pt.wikipedia.org/wiki/História_dos_jogos_eletrônicos>

2.2. ESTRUTURA/CONCEITO DOS JOGOS DIGITAIS

Neste capítulo, será abordada toda a estrutura base de um jogo, como gênero, plataformas de desenvolvimento, engine e modos de jogo.

2.2.1. Gênero

Existem inúmeros gêneros de jogos e a cada dia mais gêneros e tipos de jogos são pensados e desenvolvidos, além de quem 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.

A seguir serão descritos alguns dos gêneros de videogames mais comuns, com uma visão mais ampla dos jogos que estão sendo produzidos na atualidade.

Os gêneros mais comuns na atualidade:

- **MMORPG's**

Um jogo de interpretação de personagens online e em massa para multijogadores (*Massively ou Massive Multiplayer Online Role-Playing Game ou Multi massive online Role-Playing Game*) ou MMORPG é um jogo de computador, celulares e/ou videogame que permite a milhares de jogadores criarem personagens em um mundo virtual dinâmico ao mesmo tempo na Internet. MMORPGs são uns subtipos dos Massively Multiplayer Online Game ("*Jogos Online para Multijogadores*").

Ex: "World of Warcraft, Guild Wars 2".

- **Shooters:**

Jogos que envolvem o jogador atirando contra oponentes, dependem do tempo de reação e precisão do jogador para alcançar seus objetivos, existem dezenas de subgêneros.

Ex: “Counter Strike, Quake”.

- **Plataformers**

Jogos em que o jogador atravessa um level que é constituído por ''plataformas'', geralmente o objetivo é atravessar um level de seu começo ao fim passando por diversos obstáculos.

Ex: “Cuphead, Crash”.

- **Simuladores**

Jogos que buscam simular algo do mundo real, gênero extremamente abrangente, que varia desde jogos de futebol até simuladores de corrida.

Ex: “Second Life, FIFA”.

- **Action/Adventure**

Jogos de ação e/ou aventura, que combinam combate, exploração de ambiente, resolução de problemas, quebra-cabeças e destreza com os controles.

Ex: “GTA, Red Dead”.

- **RTS**

"Real time strategy” ou estratégia em tempo real são jogos onde envolvem noções de estratégia militar, gerenciamento de recursos, e um pouco de paciência para poder alcançar os objetivos nas partidas.

Ex: “Age of Empires, Warcraft”.

< https://pt.wikipedia.org/wiki/Gêneros_de_jogos_eletrônicos>

2.2.2. Plataformas

Uma plataforma computacional é, no senso mais geral, qualquer que seja o ambiente pré-existente, um pedaço de software que é projetado para ser executado internamente, obedecendo às suas limitações e fazendo uso das suas instalações.

Plataformas típicas incluem:

- Uma arquitetura de hardware;
- Um Sistema Operacional;
- Bibliotecas de tempo de execução**.**

As Principais plataformas para jogos são:

- Arcades;
- Celulares: (Android e IOS);
- Computadores: (Windows, Linux e Mac);
- Consoles: (X-box One, PlayStation 4 e Nintendo);
- Consoles Portáteis: (Game Boy Advance, Nintendo 3DS e PlayStation Vita);

2.2.3. Engines

Motor de jogos, ou simplesmente engine, é um conjunto de bibliotecas, para simplificar e abstrair o desenvolvimento de jogos eletrônicos ou outras aplicações com gráficos em tempo real, para videogames e/ou computadores rodando sistemas operacionais.

A funcionalidade tipicamente fornecida por um motor de jogo inclui: um motor gráfico para renderizar gráficos 2D e/ou 3D, um motor de física para simular a física ou simplesmente para fazer detecção de colisão, suporte a animação, sons, inteligência artificial, networking, gerência de memória, gerência de arquivos, gerência de linha de execução, suporte a grafos de cena e entidades e, suporte a uma linguagem de script.

As três principais Engines no mercado, voltado para o desenvolvimento “Indie” são:

- **Unreal Engine**

Unreal Engine é um motor de jogo desenvolvido pela Epic Games, usado pela primeira vez em 1998 no jogo de tiro em primeira pessoa Unreal, ele tem sido a base de muitos jogos desde então.

- **Unity**

Unity, também conhecido como Unity 3D, é um motor de jogo 3D proprietário e uma IDE criado pela Unity Technologies. Unity é similar ao Blender, Virtools ou Torque Game Engine, em relação a sua forma primária de autoria de jogos: a sua interface gráfica.

O motor cresceu a partir de uma adição de um suporte para a plataforma Mac OS X e depois se tornou um motor multi-plataforma.

- **GameMaker Studio**

GameMaker: Studio, anteriormente conhecido como Animo, e depois Game Maker, é um motor de jogo proprietário, desenvolvido pela YoYo Games. O motor tem suporte a uma linguagem de script, chamada GML.

< https://www.tecmundo.com.br/video-game-e-jogos/9263-o-que-e-engine-ou-motor-grafico-.htm>

2.2.4. Modos de jogo

- **Single-player**

Um jogo eletrônico para um jogador, também conhecido apenas por single player, é um jogo eletrônico que possibilita a participação de apenas um jogador por partida, geralmente de um jogador humano, e interagindo com a IA do jogo.

- **Multi-player**

Jogos multijogador, também conhecidos como jogos multiplayer, são jogos que permitem que vários jogadores participem simultaneamente de uma mesma partida.

O Multijogador permite que seja desfrutada uma experiência com vários jogadores, podendo ser em forma de disputa, cooperativo ou rivalidade, e proporcionar-lhes uma forma de comunicação social que está quase sempre ausente em jogos Single Player.

3. FERRAMENTAS UTILIZADAS

3.1. SOFTWARES AUXILIARES

A utilização de software extras no desenvolvimento de jogos é de extrema importância, pois são úteis na manipulação e criação dos designs dos objetos e personagens, edição do áudio, etc.

No projeto, serão utilizados os seguintes softwares:

- PhotoShop Cs4: Edição dos objetos, cenário e Sprites;
- Audacity: Edição de Áudio;
- MonoDevelop: Desenvolvimento dos scripts para o jogo;

3.2. UNITY3D

3.2.1. Introdução à Unity

Unity é um framework completo para a criação de jogos, pode ser adquirida em sua versão grátis ou proprietária e permite que o usuário criar seus jogos para diversas plataformas, como Android, Windows Phone, IOS, Web e PCs.

Ela inicialmente era bastante utilizada por empresas independentes, desenvolvedores indies e por pessoas que gostariam de conhecer sobre desenvolvimento de jogos, porém nesses últimos anos, ela vem sendo utilizada por grandes empresas como a Blizzard, ela que criou um grande sucesso do estilo CCG (Collectibles Card Games) o “Hearthstone”, existem muitos outros jogos de grande sucesso, alguns como: Ori and the Blind Forest, Inside e Cuphead.

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.

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.

< Https://producaodejogos.com/unity/>

3.2.2. Interface da Unity3D

O motor de jogos Unity3D possui uma interface bastante simples e amigável que objetiva facilitar o desenvolvimento de jogos de diversos gêneros e outros sistemas de visualização.

Sua área de trabalho é composta de várias janelas chamadas views, cada uma com um propósito específico.

A figura a seguir apresenta a Interface principal da Unity.

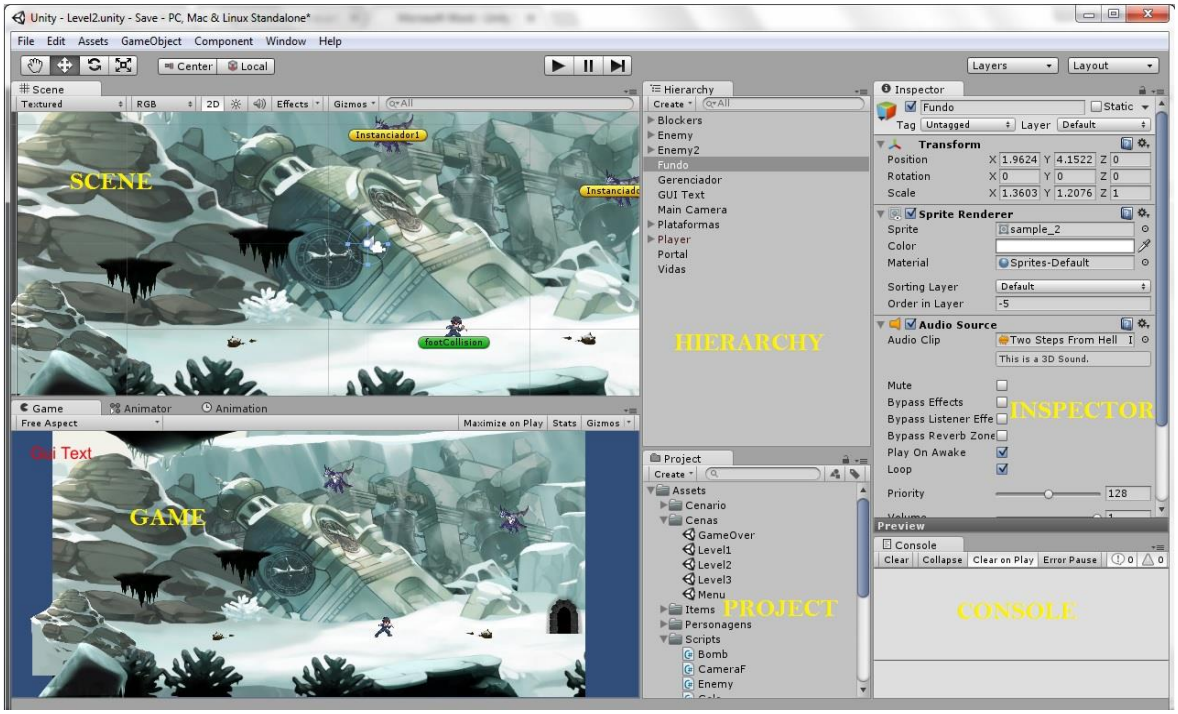

Figura 1 - Interface

3.2.3. Project View

A janela Project é a interface para manipulação e organização dos vários arquivos (Assets) que compõem um projeto tais como scripts, modelos, texturas, efeitos de áudio e Prefabs, os quais serão detalhados mais adiante na seção de Scripting.

A estrutura exibida na janela Project é correspondente à subpasta Assets dentro da pasta do projeto no sistema de arquivos do computador. Recomenda-se que a manipulação de sua estrutura e conteúdo seja efetuada somente dentro da Unity, a fim de manter a integridade dos metadados que são associados a estes elementos.

Entretanto, certas mudanças como atualização de uma textura por um editor de imagens, por exemplo, ou mesmo a adição de novos Assets, pode ser feita de forma segura diretamente no sistema de arquivos.

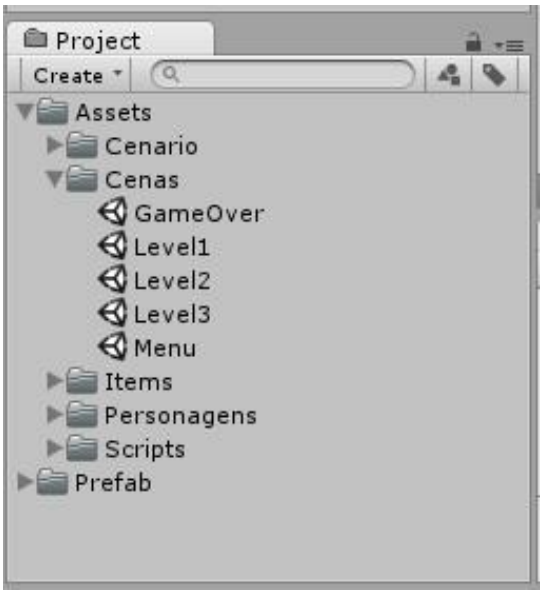

Figura 2 - Project View

3.2.4. Hierarchy View

A janela Hierarchy exibe todos os elementos da cena que se encontram na cena que se está editando. Além disso, nessa janela podemos organizar e visualizar a hierarquia de composição entre os vários objetos que compõem a cena.

Figura 3 - Hierarchy View

3.2.5. Scene View

A janela Scene á a forma principal de manipulação dos elementos visuais no editor de cenas da Unity, possibilitando a orientação e posicionamento desses elementos com um feedback imediato do efeito das alterações efetuadas. Nessa janela, pode-se manipular graficamente os objetos através das opções de arrastar e soltar com o mouse.

Essa manipulação é semelhante à ferramentas de modelagem 3D e pode-se manipular objetos tais como câmeras, cenários, personagens e todos os elementos que compõem a cena.

Figura 4 - Scene View

3.2.6. Game View

A janela "Game" é responsável pela visualização de aplicação em desenvolvimento da forma que ela será exibida quando finalizada. Nessa janela, pode-se rapidamente ter uma prévia de como os elementos estão se comportando dentro da aplicação.

Além disso, a Unity fornece a opção de se paralisar (botão pause) a simulação enquanto ela estiver em depuração, de forma a possibilitar que os parâmetros dos vários elementos possam ser ajustados para experimentação.

Nesta janela, também se pode visualizar várias informações, estatísticas (stats) sobre a simulação, tais como tempo de processamento e número de frames por segundo, número de triângulos e vértices renderizados, memória de textura utilizada, etc. Essa opção é importante para a depuração do desempenho da simulação para um posterior otimização, caso necessário.

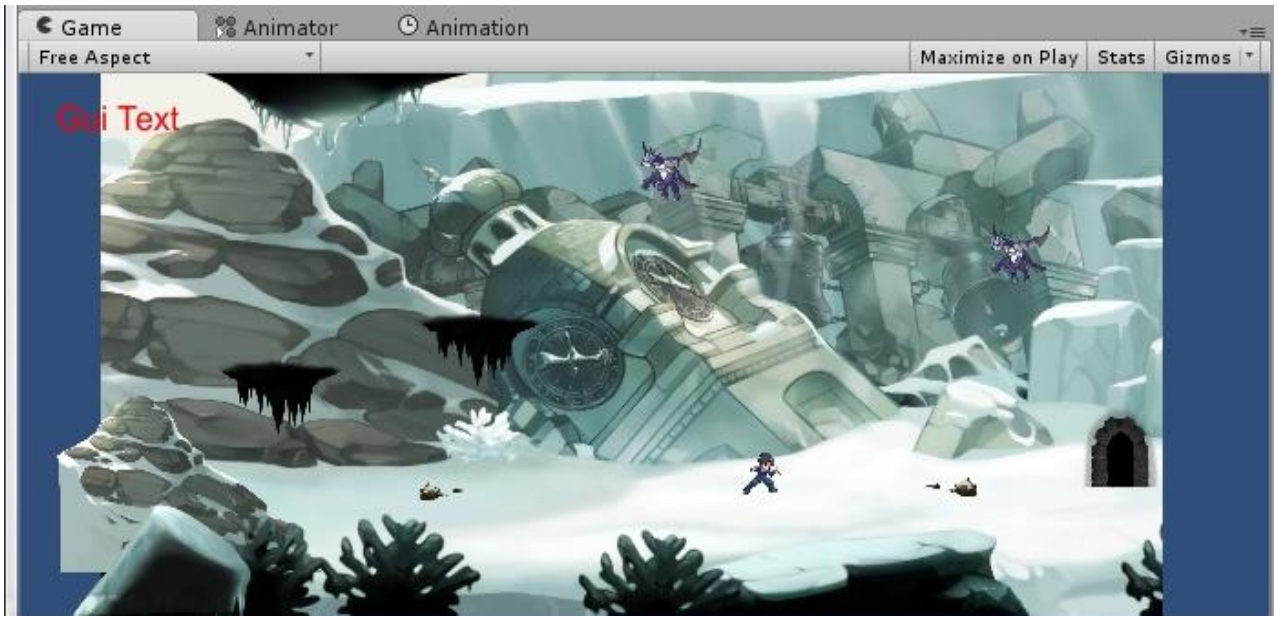

Figura 5 - Game View

3.2.7. Inspector View

Na janela Inspector, tem-se acesso aos vários parâmetros de um objeto presente no cenário, bem como aos atributos de seus componentes (Components).

Ainda na janela Inspector, pode-se ajustar os atributos públicos (parâmetros) de cada componente, inclusive durante a execução da aplicação.

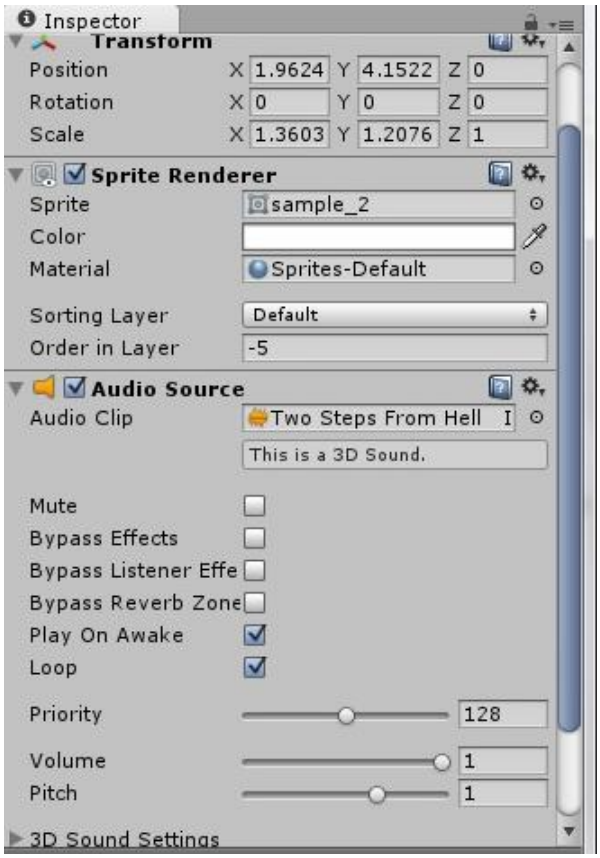

Figura 6 - Inspector View

3.2.8. Asset Store

A Unity oferece uma vasta lista de produtos na sua loja online, contendo sprites, tamplates e texturas, existem alguns produtos grátis, mas a maioria dos recursos são pagos, o que força a grande maioria de pequenos desenvolvedores a buscar conteúdos públicos, ou aprenderem por conta própria modelar seus próprios modelos.

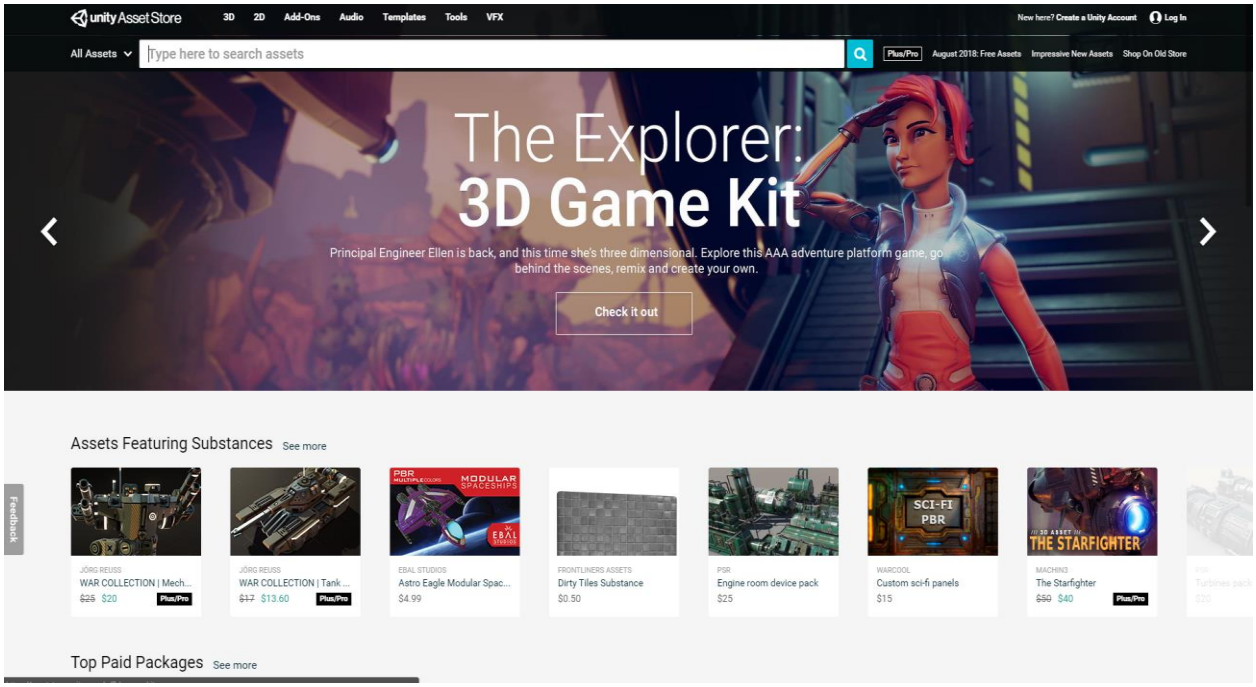

Figura 7- Asset Store

4. DESENVOLVIMENTO DO JOGO

4.1. SOBRE O JOGO

O objetivo do jogo é encontrar e coletar 10 moedas pelo cenário, evitando de ser atingido pelos ataques do inimigos e também, não cair em armadilhas, após coletar a quantidade, você poderá passar para o próximo nível.

A média de inimigos por fase, são 3 e a quantidade de moedas que o jogador deve coletar, é dobrada à cada fase, as armadilhas ficam localizadas nas partes inferiores do cenário, tais como a própria água do jogo, e algumas estacas.

O jogo atualmente, contém 2 fases, com dificuldades, e inimigos diferentes.

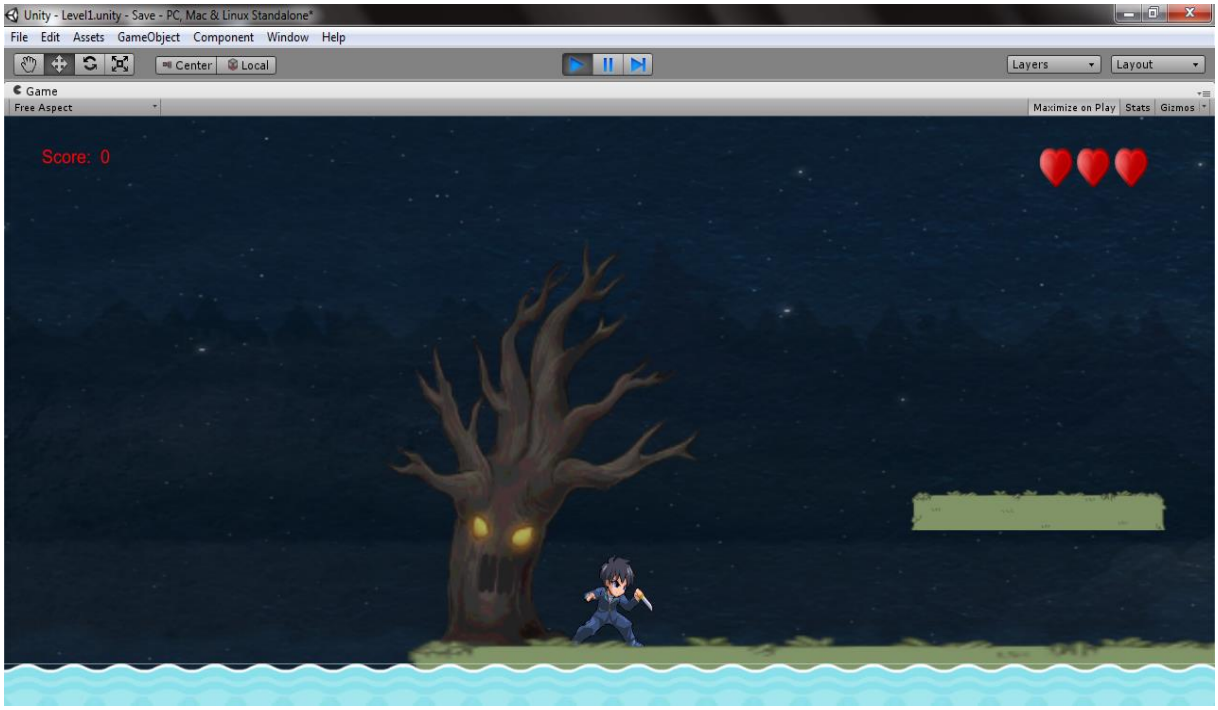

Figura 8 - Cenário 1

Imagem retirada da 1º fase do jogo

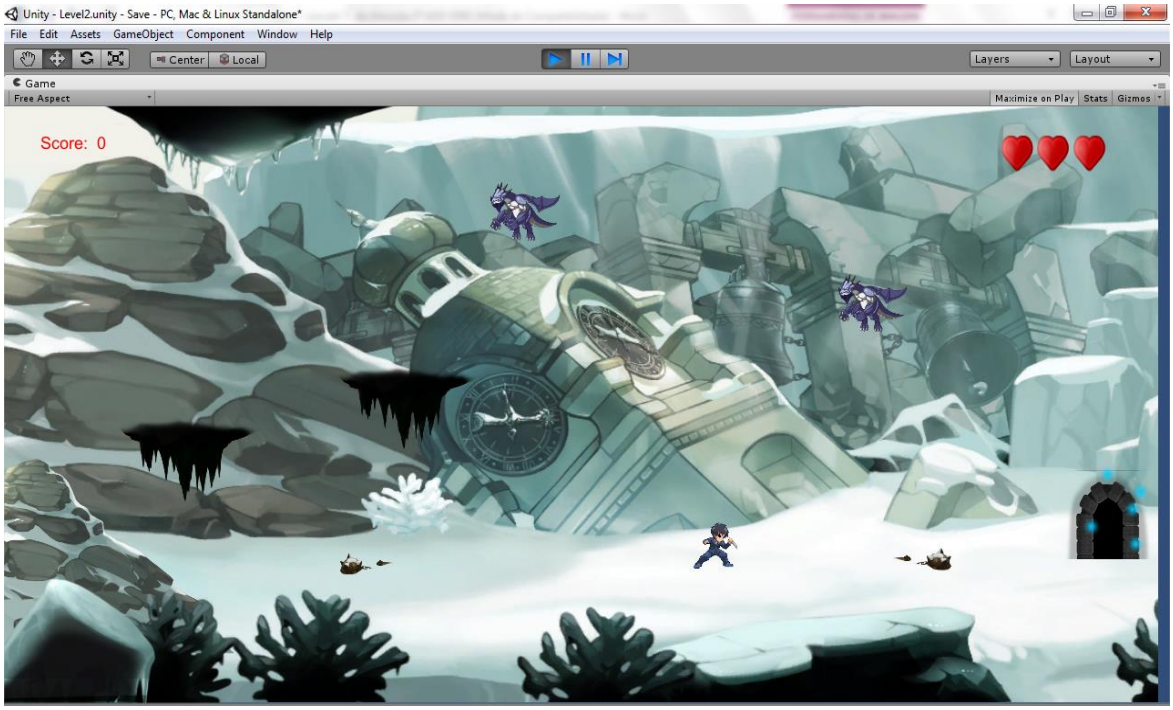

Figura 9 - Cenário 2

Imagem retirada da 2º fase do jogo

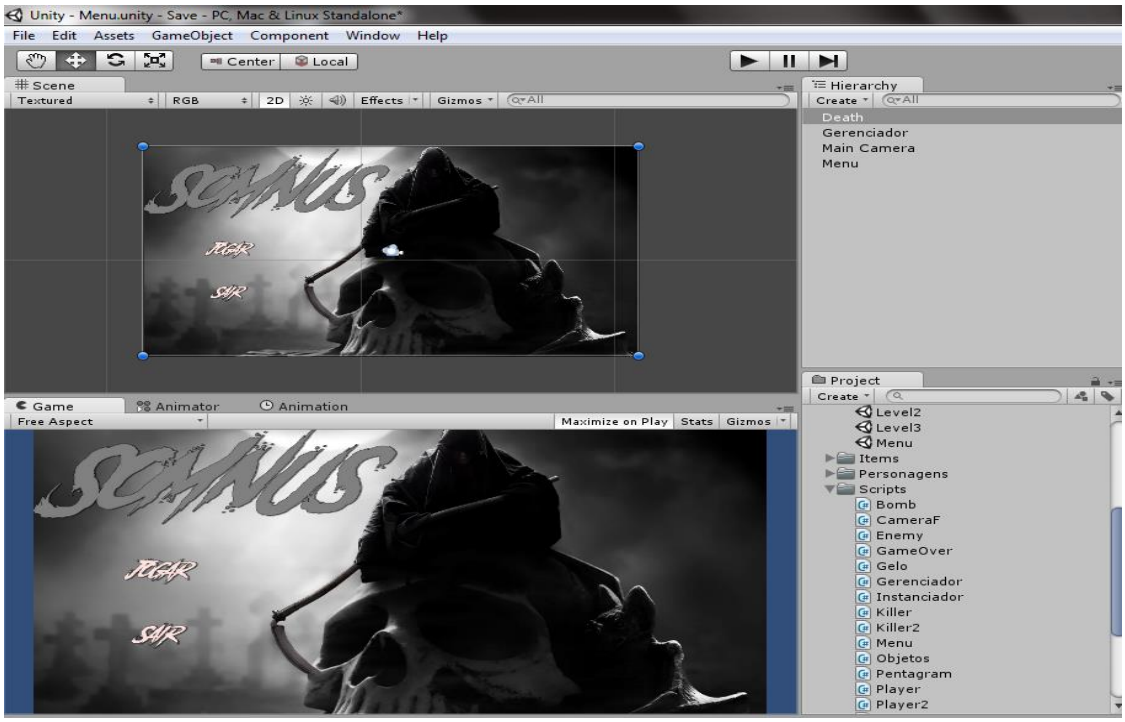

Figura 10 - Criação do Menu

4.2. PERSONAGENS

Esse é o protagonista, chamado Billy e é com esse personagem que o jogador irá jogar.

Figura 11 - Player

Alguns Inimigos que o jogador irá encontrar durante o jogo:

Figura 12 - Inimigo 1

Figura 13 - Inimigo 2

Figura 14 - Inimigo 3

4.3. SCRIPTS

Scripts são códigos que são associados aos “game objects” no jogo, a unity oferece dois tipos de linguagem, C# e javascript, podendo escolher uma das duas ou ambas na criação dos projetos, a utilização do Script é simples, você o codifica e depois somente associa ao game object.

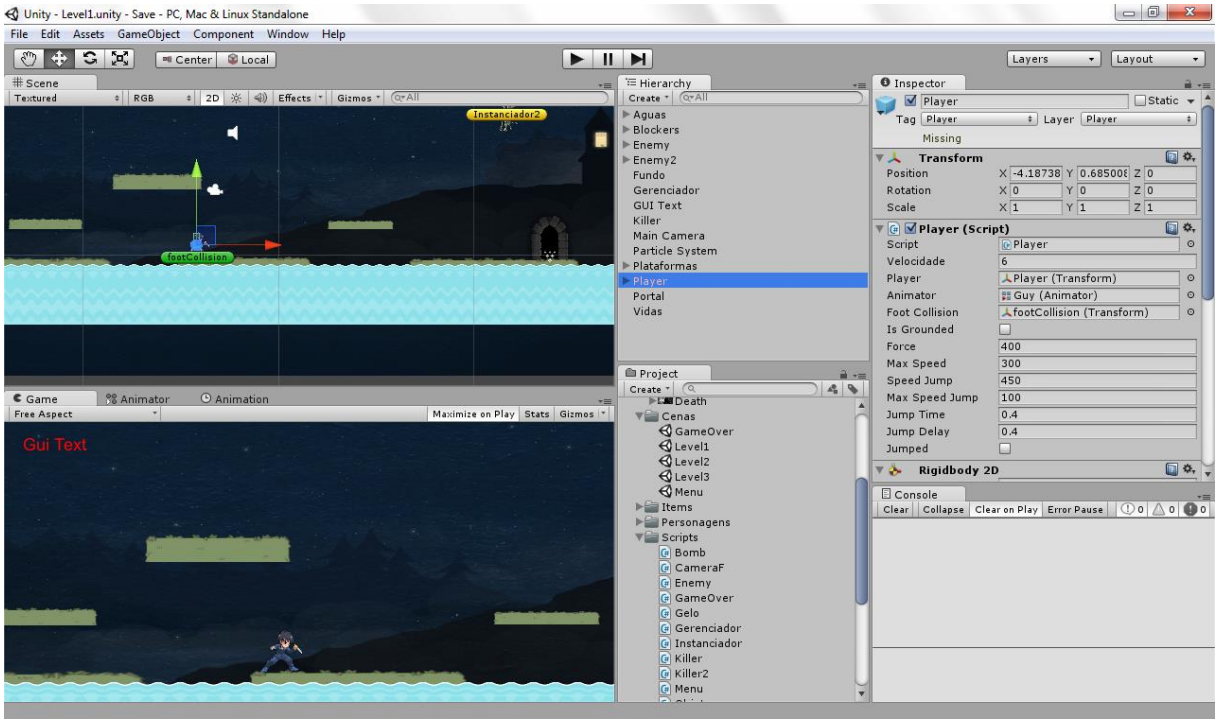

Figura 15 - Scripts

Ao lado direito é possível observar o script do player (personagem), com o arquivo script associado, e com todas as variáveis do tipo public preenchidas, animações e o transform (personagem).

4.3.1. Player

Contém todos os códigos de movimento, pulo, animações, variáveis do personagem principal e interações com script vida.

```
4  public class Player : MonoBehaviour {
5
6
7      public float velocidade;
8
9      public Transform player;
10     public Animator animator;
11
12     public Transform footCollision;
13     public bool isGrounded = true;
14     public float force;
15     public float maxSpeed;
16     public float speedJump;
17     public float maxSpeedJump;
18     public float jumpTime = 0.4f;
19     public float jumpDelay = 0.4f;
20     public bool jumped = false;
21
22
23     private Gerenciador gerenciador;
24     private Rigidbody2D rigidbodyPlayer;
25
```

Figura 16 - Variáveis do Script Player

```
42    void Movimentar()
43    {
44
45        isGrounded = Physics2D.Linecast (this.transform.position, footCollision.position, 1 << LayerMask.NameToLayer ("Ground"))
46
47        animator.SetFloat("run",Mathf.Abs(Input.GetAxisRaw("Horizontal")));
48
49        if (Input.GetAxisRaw ("Horizontal") > 0) {
50
51
52            transform.Translate(Vector2.right * velocidade * Time.deltaTime);
53            transform.eulerAngles = new Vector2(0,0);
54        }
55
56        if (Input.GetAxisRaw ("Horizontal") < 0) {
57
58            transform.Translate(Vector2.right * velocidade * Time.deltaTime);
59            transform.eulerAngles = new Vector2(0,180);
60        }
61
62        if (Input.GetKeyDown (KeyCode.Space) && isGrounded && !jumped) {
63
64            rigidbody2D.AddForce (transform.up * force);
65            jumpTime = jumpDelay;
66            animator.SetTrigger ("jump");
67            jumped = true;
68
69        }
70
71        jumpTime -= Time.deltaTime;
72
73
74        if (jumpTime <= 0 && isGrounded && jumped) {
75
76            //animator.SetTrigger("ground");
77            jumped = false;
78        }
79
80    }
```

Figura 17- Funções de movimentação e pulo

4.3.2. Inimigo

Contém animações, variáveis, movimentação, instanciadores e interação com o personagem como: causar dano.

```
3
4  public class Enemy : MonoBehaviour {
5
6      public GameObject objeto;
7      private bool Esquerda = true;
8      public float velocidade = 5f;
9      public float mxDelay;
10     private float timeMovimento = 0f;
11
12     public Transform verticeInicial;
13     public Transform verticeFinal;
14     public bool Alvo;
15
16     private float mxDelayObjeto = 0.001f;
17     private float timeObjeto = 10f;
```

Figura 18 - Variáveis do Script dos inimigos

```
32      void RayCasting()
33      {
34          Debug.DrawLine (verticeInicial.position, verticeFinal.position, Color.red);
35          Alvo = Physics2D.Linecast (verticeInicial.position, verticeFinal.position,
36                                     1 << LayerMask.NameToLayer ("Player"));
37
38      }
39
40      void Behaviours(){
41
42          if (Alvo) {
43
44              if(timeObjeto <= mxDelayObjeto){
45
46
47                  timeObjeto +=Time.deltaTime;
48
49                  Instantiate(objeto,verticeInicial.position, objeto.transform.rotation);
50
51              }
52
53          }else{
54
55              timeObjeto = 0;
56          }
57      }
```

Figura 19 - Movimentação e o "raio de ataque" do inimgo

4.3.3. Câmera Follow

Script utilizado para a câmera seguir o personagem conforme ele se movimenta no cenário.

```
4  public class CameraF : MonoBehaviour {
5
6      public Transform player;
7      public float smooth = 0.5f;
8      public Vector2 velocidade;
9
10
11
12     // Use this for initialization
13     void Start () {
14
15         velocidade = new Vector2 (0.5f, 0.5f);
16
17     }
18
19     // Update is called once per frame
20     void Update () {
21
22         Vector2 novaPosicao2D = Vector2.zero;
23
24         novaPosicao2D.x = Mathf.SmoothDamp (transform.position.x, player.position.x, ref velocidade.x, smooth);
25         novaPosicao2D.y = Mathf.SmoothDamp (transform.position.y, player.position.y+2f, ref velocidade.y, smooth);
26
27         Vector3 novaPosicao = new Vector3 (novaPosicao2D.x, novaPosicao2D.y, transform.position.z);
28
29         transform.position = Vector3.Slerp (transform.position, novaPosicao, Time.time);
30
31     }
```

Figura 20 - Câmera Follow

4.3.4. Gerenciador de level

A função do gerenciador, é “setar” a quantidade max de moedas que o player deve coletar, direcionar ele pra o próximo level, quando ele encostar no portal e auxiliar no game over e no menu.

```
3
4  public class Gerenciador : MonoBehaviour {
5
6      public Vector2 posicaoInicial;
7      public Transform player;
8      public int levelAtual;
9      public int proxLevel;
10     public int qntColetada = 0;
11     private int qntMax ;
12     public GUIText recorde;
13
14
15     // Use this for initialization
16     void Awake () {
17
18         if (player != null) {
19                 posicaoInicial = player.position;
20
21         }
22
23         qntMax = 10;
24
25         if (recorde != null) {
26
27             recorde.text = PlayerPrefs.GetInt("recorde").ToString();
28         }
29     }
30
31     public bool isColetado()
32     {
33         if (qntColetada >= qntMax) {
34
35                     return true;
36
37             } else {
38
39                     return false;
```

Figura 21 - Gerenciador

4.3.5. Menu

Um script simples, q utiliza o gerenciador para direcionar para level 1, ou um botão caso o jogador queira sair.

```
11      private Gerenciador gerenciador;
12
13      // Use this for initialization
14      void Start () {
15
16          gerenciador = FindObjectOfType (typeof(Gerenciador)) as Gerenciador;
17
18      }
19
20      // Update is called once per frame
21      void Update () {
22
23      }
24
25      void OnGUI()
26      {
27
28          GUI.skin = skinMenu;
29
30          bool play = GUI.Button(new Rect(Screen.width-1100,Screen.height-500,
31                                          160,64), btnPlay);
32
33          bool sair = GUI.Button(new Rect(Screen.width-1100,Screen.height-300,
34                                          160,64), btnSair);
35
36
37          if (play) {
38
39              gerenciador.ProxLevel(gerenciador.proxLevel);
40              Score.Inicializar();
41
42          }
43
44          if (sair) {
```

Figura 22 - Menu Script

4.3.6. Moedas

Esse é o script associado às moedas q o jogador deve coletar durante o jogo, contendo a função de somar pontos, “life time” e associa os pontos ao score.

```
11      private Gerenciador gerenciador;
12
13      void Awake()
14      {
15
16          score = GameObject.FindGameObjectWithTag("Pontos").GetComponent<Score>() as Score;
17          gerenciador = FindObjectOfType (typeof(Gerenciador)) as Gerenciador;
18
19      }
20      // Use this for initialization
21      void Start () {
22
23          timeLife = 0;
24      }
25
26      // Update is called once per frame
27      void Update () {
28
29          timeLife += Time.deltaTime;
30          if (timeLife >= tempMaxLife) {
31
32              Destroy(gameObject);
33              timeLife = 0;
34
35          }
36      }
37      void OnCollisionEnter2D(Collision2D colisor)
38      {
39
40          if (colisor.gameObject.tag == "Player") {
41
42              score.SomarPonto(ponto);
43              score.Recorde();
44              gerenciador.AddQuantidade(1);
45              Destroy(gameObject);
46
47          }
```

Figura 23 - Moeda

4.3.7. Armadilhas

Esse Script tem a função de sempre que o player cai ou encosta nas armadilhas do jogo, perderá um coração, que equivale à uma vida.

```
using UnityEngine;
using System.Collections;

public class Killer : MonoBehaviour {

    private Vidas vida;
    private Gerenciador gerenciador;



    // Use this for initialization
    void Start () {

        gerenciador = FindObjectOfType (typeof(Gerenciador)) as Gerenciador;
    }

    void OnTriggerEnter2D(Collider2D colisor)
    {
        if (colisor.gameObject.tag == "Player") {

            vida = GameObject.FindGameObjectWithTag("Vidas").GetComponent<Vidas>() as Vidas;
            if(vida.excluirVida()){

                gerenciador.StartGame();

            }
            else
            {

                gerenciador.GameOver("GameOver");

            }
        }
    }

```

Figura 24 - Armadilha

5. CONSIDERAÇÕES FINAIS

Gostaria de agradecer à toda FEMA, e pelo apoio do orientador Célio, foi um projeto que teve várias alterações por causa da limitação de material à disposição, porém o resultado foi gratificante, mesmo com as dificuldades.

Atualmente quem gosta de algo criativo, modelagem e até mesmo de programação, e se identifica com jogos digitais, tem a possibilidade de criar seus próprios jogos, com seus personagens favoritos e cenários ao seu gosto.

A respeito sobre investir na profissão de “game designer” ou apenas criar jogos por hobbie, é de escolha sua, é um mercado extremamente competitivo onde é importante saber programar, desenhar e modelar, mas se preferir criar jogos por hobbie, também é um bom caminho, hoje com plataformas como a steam e a google play, e projetos de “ground founding” é possível conseguir patrocínio, se tiver uma boa ideia e publicar nessas plataformas, para versões Desktop (Steam) e para produções voltadas para mobile (Google Play).

A respeito do projeto, pretendo expandir ele, colocando novos “levels”, melhorar a IA dos inimigos, colocar para as duas plataformas (Desktop e Mobile) e por fim, publicá-lo.

REFERÊNCIAS

Dias, Raphael. “Unity Guia completo sobre a Engine.” Disponível em:

< Https://producaodejogos.com/unity/> Acessado 18 de março 2018.

Holmes, Mike. “Jogos Indie: para ficar, ou moda passageira?”. Disponível em:

<https://www.gamereactor.pt/especiais/20054/Jogos+Indie+Para+Ficar+ou+Moda+Passageira/ Acessado 16 de março 2018.

SMITH, M & QUEIROZ, C. Unity 5 cookbook, Editora Packt Publishing Ltd., 2015.

Tecmundo: “O que é engine ou motor gráfico?” Disponível em:

< https://www.tecmundo.com.br/video-game-e-jogos/9263-o-que-e-engine-ou-motor-grafico-.htm> Acessado 18 de março 2018.

THORN, A. Learn Unity for 2D Game Development, Editora Friendsoft Apress, 2013.

THORN, A. Pro Unity Game Development with C#, Editora Friendsoft Apress, 2014.

Unity 3D: Introdução ao desenvolvimento de games. Disponível em:
<https://www.devmedia.com.br/unity-3d-introducao-ao-desenvolvimento-de-games/30653> Acessado 18 de março 2018.

Unity 3D: Aprenda a Unity. Disponível em:

< https://unity3d.com/pt/learn> Acessado 14 de março 2018.

Unity 3D: Tutoriais. Disponível em:

< https://unity3d.com/pt/learn/tutorials> Acessado 14 de março 2018.

Unity 3D: Comunidade. Disponível em:

< https://unity3d.com/pt/community> Acessado 16 de março 2018.

Vidor, George. “Mercado dos games”. Disponível em:

< https://oglobo.globo.com/economia/o-mercado-de-games-no-mundo-fatura-mais-que-cinema-musica-somados-16251427> Acessado 16 de março 2018.

Wikipédia: Gênero de jogos eletrônicos. Disponível em:

< https://pt.wikipedia.org/wiki/Gêneros_de_jogos_eletrônicos>

Acessado 18 de março 2018.

Wikipédia: História dos jogos eletrônicos. Disponível em:

< https://pt.wikipedia.org/wiki/História_dos_jogos_eletrônicos> Acessado 15 de março 2018.