Escola Secundária de Santa Maria Maior

Curso de Gestão e Programação de Sistemas Informáticos

Ano letivo 2018/2019

Relatório Final do Projeto De Aptidão Profissional
"The Ravenous Tarrare"

Autor: Luís Miguel Rego Torres

Viana do Castelo, junho de 2019

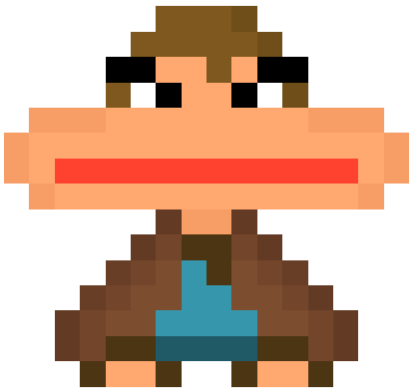
Escola Secundária de Santa Maria Maior

Curso de Gestão e Programação de Sistemas Informáticos

Ano letivo 2018/2019

Relatório Final do Projeto De Aptidão Profissional
"*The Ravenous Tarrare*"

Videojogo para Computador



Autor: Luís Miguel Rego Torres

Orientação: Dalila Dantas

Viana do Castelo, junho de 2019

Agradecimentos

Gostaria de agradecer aos Professores e amigos que apesar de se tratar de um projeto invulgar, estiveram do meu lado no seu desenvolvimento e proporcionaram ajuda e suporte naquilo que era necessário.

Gostaria ainda de agradecer à *Unity Tecnhologies* e à sua comunidade pela documentação incrível, que aborda uma imensidade de conteúdos de forma amigável e orientada a um iniciante. Sem esta ajuda teria sido muito mais difícil encontrar as soluções aos problemas que encontrei.

Por último, gostaria de agradecer a todos aqueles que trabalham e ajudam no desenvolvimento de software *OpenSource.* Foi graças a estas ferramentas que consegui desenvolver este projeto por inteiro.

Índice

Dicionário

Engine: motor de jogo (por norma consiste num conjunto de bibliotecas de programação avançada com uma interface, de forma a simplificar o processo de desenvolvimento de jogos);

Unity: motor de jogo usado no projeto;

Roguelike: trata-se de um subgénero de jogos RPG, em que por norma não existe um conceito de progresso, assemelha-se aos jogos antigos de Arcada;

API'S: Application Programming Interface: Interface de Programação de Aplicações;

NPC's: non-playable-characters: Trata-se de qualquer personagem presente no jogo que o jogador não tenha controlo;

Sprites: objeto gráfico bidimensional ou tridimensional;

Scripts: Documentos de código que são compilados por um software;

Assets: Quaisquer recursos utilizados (sprites, áudio, scripts etc.);

Gameover: Fim de Jogo;

HighScore: Maior Pontuação;

OpenSource: Código aberto. É um modelo de desenvolvimento que promove licenciamento livre;

PixelArt: é uma forma de arte digital na qual as imagens são criadas ou editadas tendo como elemento básico os pixels;

Tarrare: Nome Próprio do personagem histórico em que o jogo se baseia;

SoundEffect: Efeito Sonoro;

GameDevelopment: desenvolvimento de jogos eletrônicos. É o processo na qual um jogo eletrónico é produzido;

IndieDevelopment: desenvolvimento de jogos eletrônicos criados por uma pessoa ou pequenas equipas;

Prefabs: Prefabricação de algo, isto é ter um componente/elemento pronto a utilizar com as devidas configurações prefabricadas

Tiles: Imagem quadrada que contem um certo número de pixéis (16x16 neste caso);

Tilesets: conjunto de Tiles, que por norma são utilizados na criação de mapas;

Layers: camadas;

SpriteSheets: Conjunto de sprites organizadas horizontalmente ou verticalmente;

Idle: Base, ou que se ativa quando não ocorre interação;

Chiptune: é um gênero de música eletrônica sintetizada, produzido por chips de som de antigos;

UI: User-interface: Interface de Utilizador. São todos os objetos que interagem com o utilizador apenas para dispor ou receber informação. (ex.: Botões, Títulos, Ícones etc.)

Colliders: Objetos Colidíeis, ou seja, qualquer objeto que possa colidir.

Spawn: Aparecimento de algo.

Raycast: O algoritmo que lança raios de forma a medir distâncias.

Introdução/Resumo

O meu projeto de aptidão profissional é um Videojogo desenvolvido com o suporte da *Engine Unity*. Este relatório consiste numa explicação de todo o processo de tal desenvolvimento.

Neste videojogo, do tipo *roguelike*, o jogador está sempre faminto. Para não morrer de fome deve procurar *NPCs* no mapa e devorá-los. Só assim consegue manter-se sem fome e saciado. Caso não coma *NPCs*, o jogador acaba por morrer de fome, gameover. Quando o jogador morre é registada a sua pontuação (*highscore*) numa base de dados externa, para futura análise e comparação com outros jogadores deste videojogo.

No que diz respeito ao desenvolvimento deste videojogo, todos os *Assets*, exceto a fonte de texto utilizada em alguns menus, foram desenvolvidos por mim, tal como proposto no anteprojeto. Os *Assets* desenvolvidos foram:

Gráficos: Sprites, Mapa, Ícones, Botões e Planos de fundo;

Scripts: Mecânicas, Interfaces, Conexão a Base de Dados, Pontuações;

Áudio: Melodia de Fundo e vários Efeitos sonoros.

A interligação de todos os componentes e o tempo investido na aprendizagem e desenvolvimento dos mesmos, permitiu tornar um jogo, que à primeira vista parece fácil de desenvolver em algo único, pessoal e complexo. O investimento de muito tempo e recursos nesses componentes permitiu-me conhecer as várias etapas necessárias em múltiplas áreas de conhecimento no desenvolvimento de um jogo.

Fundamentação Teórica

A principal razão que me levou a escolher um jogo como projeto de aptidão profissional, foi a minha inclinação para prosseguir uma carreira nessa mesma área. Foi esta mesma razão que me fez inicialmente entrar em contacto com a programação em geral, sempre fui admirador de *GameDevelopment*, mais especificamente *IndieDevelopment* e tenciono entrar na área profissionalmente.

A ideia do projeto tem o fundamento num personagem histórico denominado por *Tarrare,* quando entrei em contacto com a sua história de vida fiquei encantado com a sua singularidade. Após ter descoberto tal singularidade fiquei impressionado pela falta de conteúdo criativo a explorar tal anomalia, foi isto que me levou a decidir ser pioneiro nessa mesma exploração.

Resumidamente *Tarrare* foi um soldado francês na época de 1780, conhecido por seus hábitos alimentares incomuns. Tendo a capacidade de comer grandes quantidades de carne, ele estava constantemente faminto, o que levou a que os seus pais numa época em que a pobreza governava as ruas de França não o pudessem sustentar. Isto levou a que a sua família o deixasse ainda na adolescência, então *Tarrare* foi para Paris onde trabalhou como um artista de rua ele engolia rolhas, pedras, animais vivos entre muitos outros objetos peculiares. Mais tarde na sua vida *Tarrare* entrou no exército, esta carreira militar acabou por hospitalizá-lo, o que por sua vez levou a sua morte. *Tarrare* viveu entre 1772-1798. Recomendo que consultem o link na bibliografia pois a meu ver trata-se de uma história no mínimo peculiar.

Após ter conhecimento deste personagem comecei por criar um rascunho com a ideia base, queria desenvolver um jogo *roguelike,* em arte pixel 16 por 16, no qual o jogador iria comer animais espalhados pelo mapa, que por sua vez iriam tentar fugir, se o tempo que o jogador sobrevive-se fosse um novo recorde teria opção de guardar esse mesmo recorde, a semelhança das máquinas *Arcade* antigas. Realizei uma pesquisa em busca de uma *Engine* que iria satisfazer os meus requisitos e que tivesse suporte para novos utilizadores, as duas *engines* que prevaleceram foi o *Unity* e a *Godot,* no fim acabei por decidir *Unity* devido a incrível documentação que eles proporcionam na sua página e a comunidade que está constantemente a ajudar os novos utilizadores.

Como foi possível verificar nos *Assets* indicados na introdução, este projeto aborda várias áreas e não apenas a área de informática. Isto levou a que fosse necessário a aprendizagem de vários conteúdos que abrangem áreas como *PixelArt,* desenvolvimento de animações, criação de banda sonora/efeitos sonoros e programação C# com a utilização do *Unity.*

Para a elaboração dos componentes que são abrangidos pelas áreas indicadas acima, eu utilizei várias ferramentas *OpenSource*, o que levou a necessidade de aprender e dominar as mesmas. Segue uma listagem destas ferramentas:

- *Tiled*, ferramenta *OpenSource* de edição de níveis, com suporte para mapas ortogonais, isométricos, hexagonais. Esta ferramenta foi uma ajuda vital na criação do mapa em que o jogador vagueia, as *Tilesets* utilizadas foram desenvolvidas e editadas por mim em *Paint.net.*

Figura 1: Ícone

- *Unity, Engine* escolhida para o desenvolvimento deste projeto, A Unity oferece aos desenvolvedores a capacidade de criar jogos em 2D e 3D, suportando as seguintes APIs: Direct3D no Windows e Xbox 360; OpenGL no MacOS e Linux; OpenGL ES no Android e iOS; WebGL na Internet. (versão 2017.3.0f3)

Figura 2: Ícone

- *Tiled2Unity,* trata-se de uma ferramenta *OpenSource* que proporciona fácil exportação de um ficheiro do *Tiled* para um projeto *Unity,* para isto utiliza o sistema de *Prefabs* que o Unity disponibiliza. (versão mais recente)

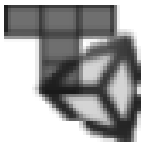


Figura 3: Ícone

- *Paint.net,* ferramenta *OpenSource* que eu utilizei para criar todos os gráficos e animações presentes no projeto, usufrui também das extensões "Animation Helper Plugin" e "Grid Maker Plugin v3.0". Links encontram-se na bibliografia. (versão mais recente)

Figura 4: Ícone

- *Audacity,* programa *OpenSource* de edição digital de áudio, utilizei para a criação de todos os componentes de áudio presentes no projeto. (exceto a música de fundo) (versão mais recente)

Figura 5: Ícone

- *BeepBox,* ferramenta online *OpenSource* de criação de melodias "*ChipTune"*, ou seja, melodias de género eletrónico sintetizado, muitas vezes utilizadas para proporcionar uma experiência retro. (versão mais recente)

Figura 6: Ícone

- *WampServer*, é um software que facilita a configuração de um software interpretador de scripts local e um banco de dados no sistema Windows. Foi utilizado de forma a criar uma simulação de uma página online. (versão mais recente)

Figura 7: Ícone

- *VisualStudio,* é um ambiente de desenvolvimento integrado (IDE) da Microsoft para desenvolvimento de software. Software no qual desenvolvi todo o código C# utilizado no projeto. *(versão mais recente)*

Figura 8: Ícone

- *Notepad++,* é um editor de texto e de código fonte de código aberto sob a licença GPL. Foi neste software que desenvolvi todo o Código PHP. (versão mais recente)

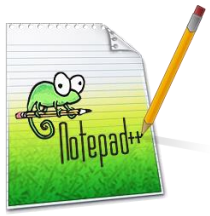


Figura 9: Ícone

Atividades Desenvolvidas

Esta parte do relatório está no Presente de forma a facilitar a leitura passo-a-passo e está organizada pelas fases de desenvolvimento. Isto é importante referir pois o projeto não foi desenvolvido pela ordem que está explicado, acredito que assim seja possível explicar o processo de uma forma mais amigável para o leitor comum. Caso exista necessidade de verificar a evolução cronologia recomendo ler os relatórios periódicos.

Inicialmente eu construí um plano de desenvolvimento que ponho em prática sempre que é necessário implementar algum componente no projeto. Este plano consiste nas fases de:

1. *Design ;*(seja sonoro ou gráfico)
2. Animação (caso seja necessário);
3. Requisitos de pré-implementação (caso seja necessário);

Criação de Assets;

4. Programação;
5. Implementação;
6. Aperfeiçoamento;

Implementação e Utilização de Assets;

Seguindo o plano referido, caso se trate de um elemento gráfico, começo por criar o grafismo do componente em questão para isto utilizo o *Paint.net*. Caso o componente se trate de um personagem é necessário animá-lo, ou seja, é necessário criar *SpriteSheets* que represente o movimento do personagem *frame* por *frame*, para a pré-análise destas *Sprites* em Movimento, eu utilizo a extensão *"Animation Helper Plugin".* Agora seguem as *SpriteSheets* que utilizei para dar vida e animar os personagens do jogo:

Tarrare:

Figura 10: SpriteSheets, Tarrare

Galinha:

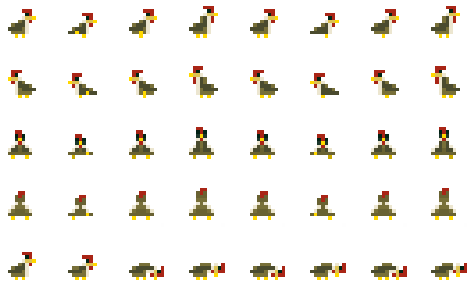


Figura 11: SpriteSheet, Galinha

Ovelha:

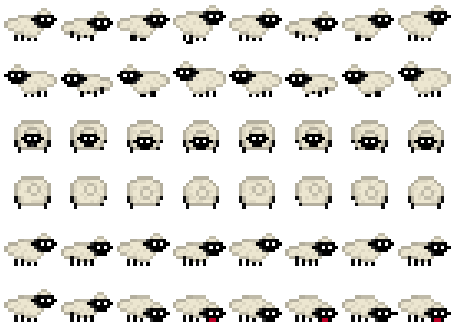
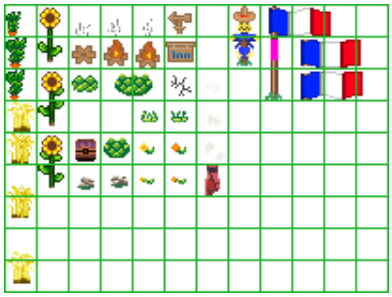
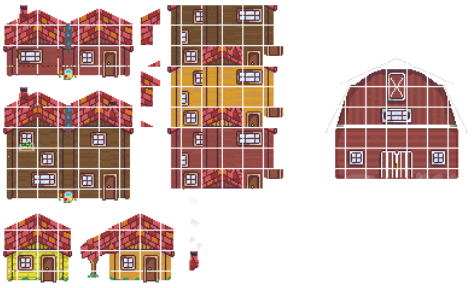


Figura 12: SpriteSheet, Ovelha

Ao contrário dos personagens acima existem componentes, como por exemplo o mapa e a *UI*, que não requerem animações via *SpriteSheet*. Com isto não quero ser mal interpretado pois apesar de não requerer animações o mapa requer alguns requisitos diferentes como a criação de *Tilesets,* que funcionam como peças de um puzzle na montagem do mapa em si. Para a composição destas *Tilesets* é necessário criar os diferentes *tiles* individuais e organizá-los de forma a facilitar a futura manipulação, isto é realizado novamente em *Paint.net*.

Agora seguem os *Tilesets* utilizados na criação do mapa:





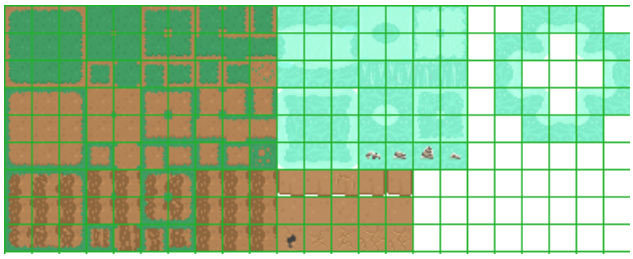


Figura 13: Tilesets

Agora que temos o material para criar o mapa, é altura de utilizar a ferramenta *Tiled* para unificar as peças, esta unificação é possível graças a vários subsistemas que o programa proporciona como por exemplo o sistema de camadas. É também nesta fase que eu elaboro as colisões do mapa, de modo a facilitar trabalho futuro.

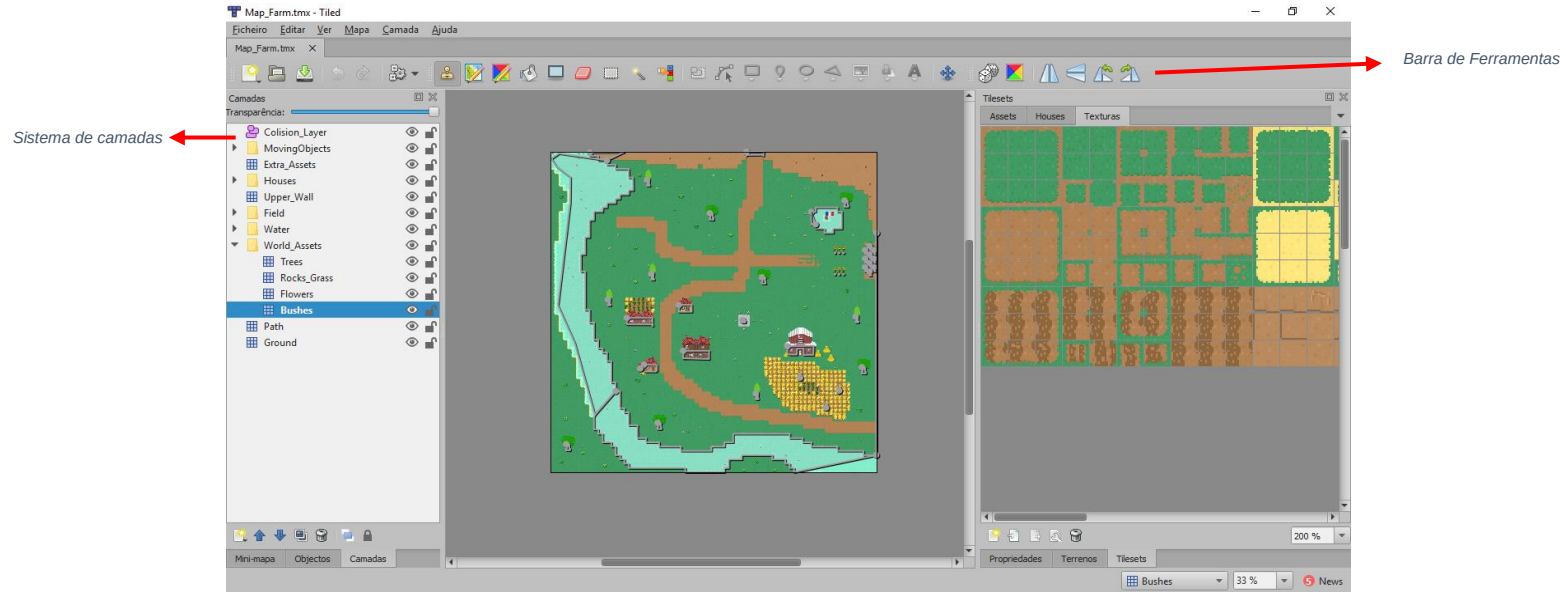


Figura 14: Interface do Tiled, (mapa pré-exportação)

De seguida exporto o mapa do *Tiled* para o *Unity*, para facilitar este processo eu utilizo a ferramenta *Tiled2Unity,* isto leva a criação automática de uma *Prefab* (“Map_farm”), composta por todas as *Layers,* incluindo as colisões referidas acima.

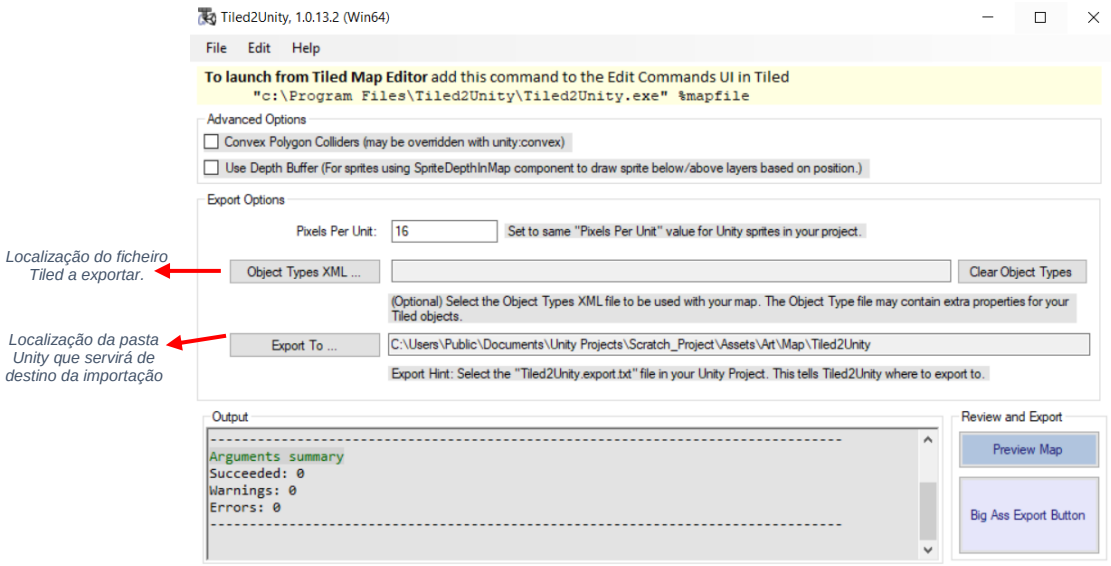


Figura 15: Interface do Tiled2Unity

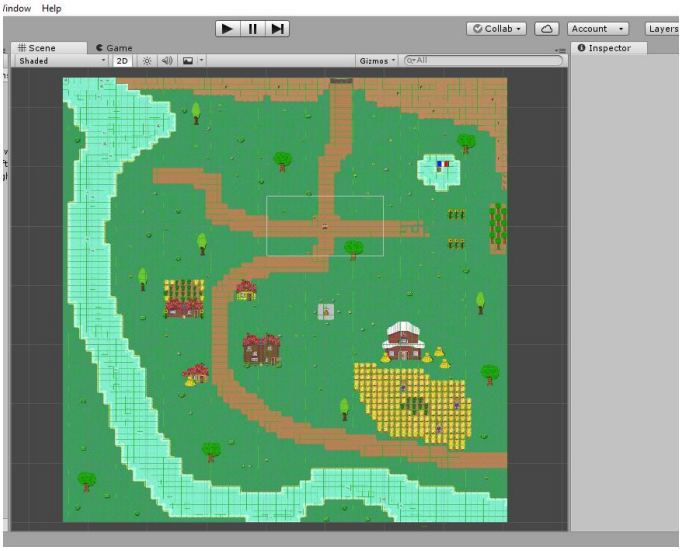


Figura 16: Mapa após importação

Em relação ao *UI*, a criação do mesmo é totalmente em *Paint.net*, utilizo inicialmente uma resolução 16x16 e mais tarde escalo-a para o tamanho necessário. Desta forma mantenho o estilo em todos os aspetos do jogo. Seguem os elementos do UI criados:

Figura 17:UI, utilizado

Por outro lado, caso o componente a adicionar se trate de um componente sonoro, o processo é completamente diferente, ao contrário dos processos acima em que utilizo o *Paint.net*, desta vez não é necessário efetuar *Design* gráfico, mas sim *Design* sonoro, para isto a ferramenta de escolha é o *Audacity*, pois possibilita inúmeras possibilidades de edição e manipulação de som. (Ver sons utilizados nos anexos)

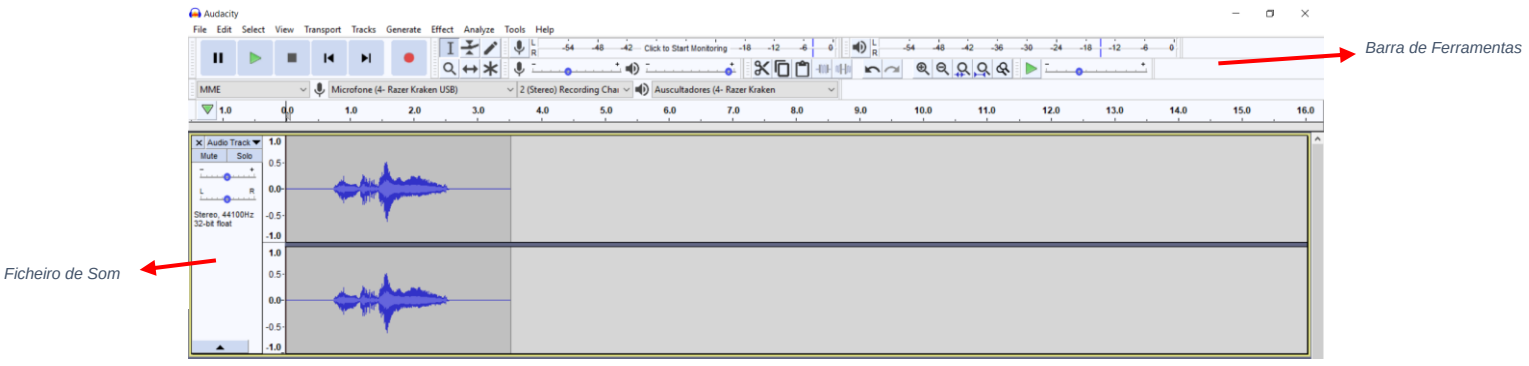


Figura 18: Interface Audacity

Por norma os sons são diretamente gravados e editados no *Audacity* a única exceção é no que diz respeito a melodias, para este efeito escolhi usar o *BeepBox* pois está mais orientado a criar melodias estilo *chiptune* o que é exatamente o que procuro na criação de um jogo deste estilo. O *BeepBox* é totalmente *OpenSource* e dispõe de várias ferramentas e tutoriais criados pela comunidade. (Ver melodias utilizadas nos Anexos)

Figura 19:Interface BeepBox

Após o processo de criação dos *Assets* em geral, desde áudio, animações, mapa, *UI*, entramos na fase de implementação e interligação dos mesmos. Esta implementação varia drasticamente dependendo do componente a implementar. Esta fase exige um bom controlo sobre a interface do *Unity* e as suas propriedades, de forma a tornar a leitura mais compreensiva irei fazer uma pequena análise da interface do *Unity.* (aconselho uma análise mais profunda, documentação na bibliografia do relatório)

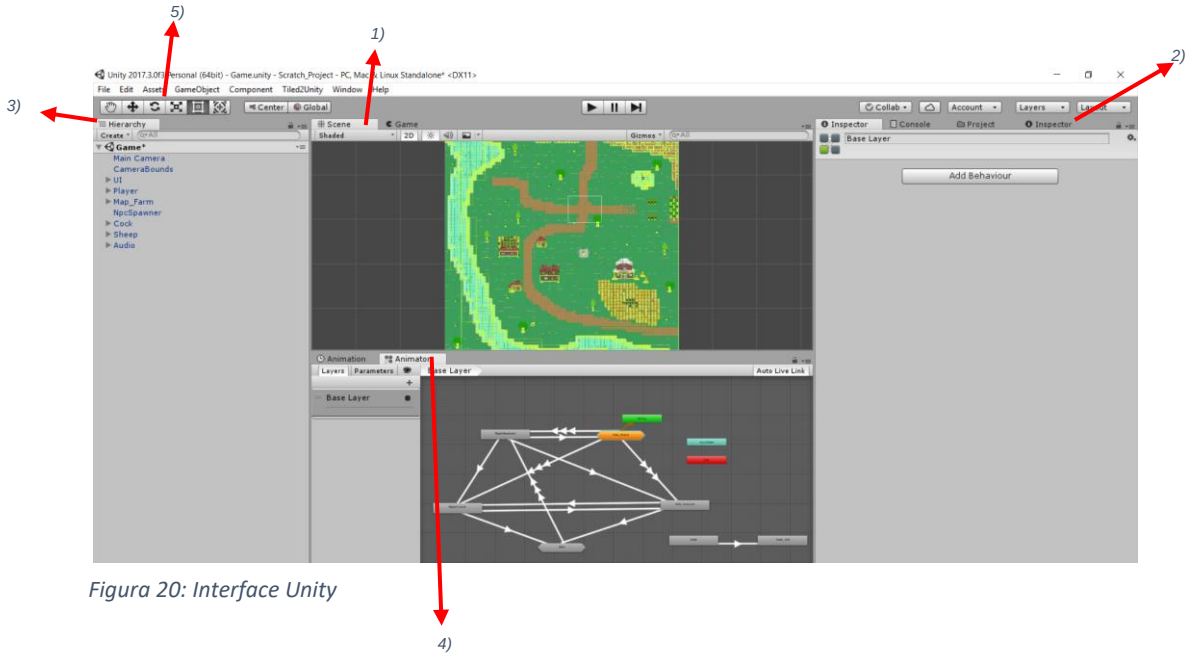


Figura 20: Interface Unity

- 1) Janela de cenas, é responsável por mostrar a posição/escala dos objetos a nível gráfico;
- 2) Janela do inspetor, é responsável por mostrar todas a propriedades do objeto selecionado;
- 3) Hierarquia é onde se pode verificar a existência dos objetos a nível hierárquico, isto a relação que tem entre si;
- 4) Animador, sistema do Unity responsável por interligar as animações criadas através das *SpriteSheets;*
- 5) Barra de ferramentas;

Nesta fase crucial é necessário utilizar ferramentas como o VisualStudio, e o inspetor do *Unity*. A nível de programação acredito que seja mais simples e amigável para o leitor explicar apenas movimento dos personagens. Para estes *scripts* eu escolhi basear-me em componentes que o *Unity* dispõe como:

- *RigidBody2D*, de forma a usufruir de uma simulação de forças como a gravidade, massa, resistências etc.;
- *Colliders,* para possibilitar a deteção de colisões entre objetos;
- *Sprite Renderer,* renderiza *Sprites* com parâmetros customizáveis;
- *Animation,* possibilita criar *clips* de animação a partir de *SpriteSheets*;
- *Animator,* possibilita interligar diferentes *clips* de animação;

Com a ajuda destes componentes, e de uma forma simplificada o código de movimento funciona da seguinte forma, os objetos movem-se livremente num plano 2D (Horizontalmente, Verticalmente e Diagonalmente), caso se trate do *Tarrare* este movimento é controlado pelo *Input* que o jogador efetua nas teclas direcionais (WASD ou Setas). Por outro lado, um *NPC* caso não colida com o jogador (através dos Colliders) efetua o movimento de forma randomizada, utiliza parâmetros fornecidos pelo programador para calcular a distância, direção e velocidade de cada movimento. Caso colida com o jogador o movimento efetuado é oposto ao do jogador no momento da colisão (aparenta fugir). Em caso de curiosidade aconselho ver os comentários presentes no código do projeto. (consultar os anexos para ver o código)

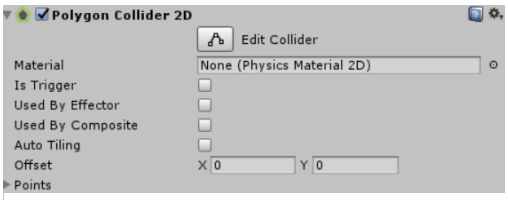


Figura 21: Exemplo de Collider

Com o código desenvolvido é necessário dar vida ao mesmo, ou seja, adicionar as animações. para isto é necessário juntar ao objeto que possui o código e os *Colliders* um *Sprite Renderer* que tem função de renderizar uma imagem na posição do objeto. Com a devida imagem em posição é necessário torná-la num *clip* de várias imagens. Para tal efeito é necessário utilizar o *Animator,* após cortar devidamente as *SpriteSheets* anteriormente criadas, o *Animator* permite organizá-las criando um *clip* de imagens. (é neste mesmo *Animator* que o sistema de eventos do Unity pode ser manipulado algo importar-te no desenvolvimento futuro de código)

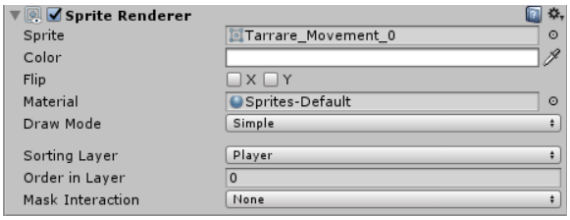


Figura 22: Exemplo de Sprite Renderer

Agora que temos os *clips* criados é necessário interligá-los, um *Animator* com os devidos parâmetros criados irá fazer exatamente isso. O método que eu utilizei foi interligar os parâmetros do *script* de movimento aos parâmetros do *Animator,* desta forma os parâmetros estão em constante atualização uns com os outros.

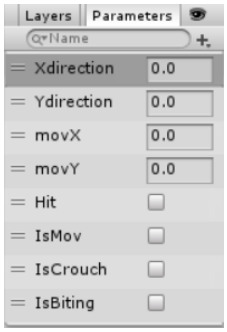


Figura 23: Exemplo de Parâmetros do Animator

```
anim.SetFloat("movX", normalizedDirection.x);
anim.SetFloat("movY", normalizedDirection.y);
anim.SetBool("IsFleeing", IsFleeing);
anim.SetBool("IsMov", IsMov);
```

Figura 24: Interligação entre parâmetros

Após adicionar as animações aos objetos, é preciso dar um pouco mais de realismo aos mesmos, uma boa forma de conquistar este realismo é através do som. Com os efeitos sonoros que desenvolvemos na fase anterior um método rápido e simples de os adicionar ao jogo, é através de código, o método que eu utilizei foi adicionar o som a hierarquia e depois quando quero que o som atue chamo no código.

Figura 25: Exemplo do som na hierarquia

```
if (isDying == true)
{
    DieSound.Play();
    CanControl = false;
    isDying = false;
    isDead = true;
    anim.Play("dead");

}
```

Figura 26: Exemplo de uso de Som

O processo descrito da página 9 a 16 é um exemplo de fácil leitura do trabalho que foi realizado para todos os Assets que foram sendo demostrados ao longo do relatório e anexados. Agora segue uma análise mais detalhada da função dos 14 Scripts em C.

- Administração do ataque: "*Bite_Manager*";
- Controlador da câmara: "*Camera_Controller*";
- Inteligência artificial dos *NPC*: "*Flee_AI"*;
- Administração da fome: "*Hunger_Manager*";
- Controlador do menu: "*MainMenu_Controller*";
- *Spawn* de NPC's: "*NpcSpawner*";
- Controlador do Jogador: "*Player_Controller*";
- Controlador do *Spawn* do Player: "*Player_Spawner*";
- Administrador do alcance da visão dos NPC's: "*Range_Manager*";
- Administração do sistema de *Raycast* para detetar colisões: "Raycast_Manager";
- Controlador da Pontuação: "Score_Controller";
- Administração do *UI*: "UI_Manager";

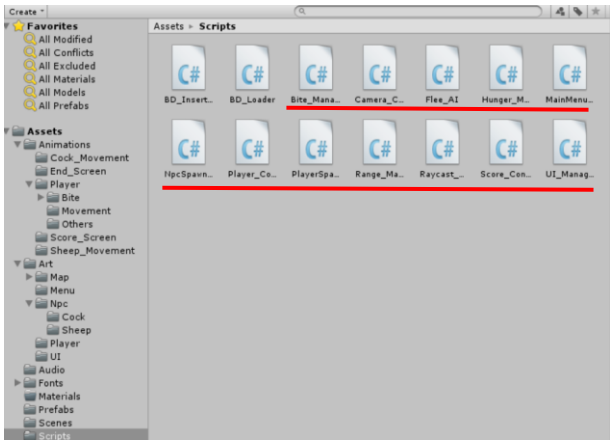


Figura 27: Scripts em C

A única exceção a regra foi o sistema de scores pois este foi desenvolvido em PHP e implementado posteriormente ao Unity.

O sistema de pontuações começou pela criação de um Base de Dados com *MySQL*, foi aqui que utilizei as ferramentas que ainda não foram abordadas no relatório o *WampServer* e o *Notepad ++*, a base de dados resume-se a uma única tabela com os campos de *id, nickname, score, date.*

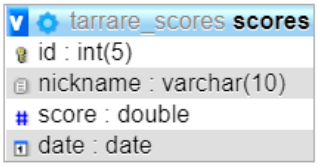


Figura 28: Tabela da Base de Dados

Após ter criado esta tabela foi uma questão de desenvolver um *script* que se conectasse a Base da Dados e que permitisse consultar, inserir dados e calcular o score máximo atual na base de dados, desta forma eu teria toda a informação que iria precisar para identificar se o jogador tinha conseguido bater o *HighScore*. A solução que cheguei foi a criação de 4 *scripts PHP* cada um responsável por uma função diferente:

- Conexão: *"tarrare_connect.php";*
- Consulta de dados: *"tarrare_data.php";*
- Cálculo do HighScore: *"tarrare_highScore.php";*
- Inserção de dados: "*tarrare_insert_data.php";*

Agora que tinha a base de dados, e a conexão a mesma criada foi necessário implementar este código no *Unity*, para isso criei 2 *Scripts* em C# que utilizam os métodos *WWW* do *Unity* de forma a comunicarem com um *URL (*neste caso os ficheiros PHP*)*, um script tem a função de inserir na base de dados e outro tem a função de carregar dados.

- Carregar dados: *"BD_Loader";*
- Inserir dados: "*BD_Inserter*";

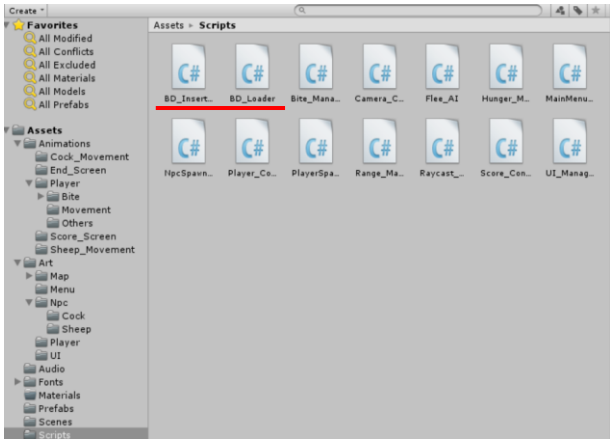


Figura 29: Scripts de Conexão a Base de Dados

No desenvolvimento do Sistema de pontos foi necessário criar um *script* que conta-se o tempo passado desde o começo do jogo até ao eventual *GameOver.* Depois deste script estar criado foi necessário interligá-lo com o "*BD_Inserter",* que por sua vez estava interligado ao ficheiro PHP que inseria ou verificava os dados na Base de Dados.

De forma a finalizar este sistema de pontos foi necessário criar também um *UI* que combina com a arte restante do jogo e mostra o Score da sessão de jogo.

Reflexão Crítica e Autoavaliação

Após ter finalmente terminado o projeto que me propus a fazer, acredito que seja algo original e único, apesar do trabalho árduo consegui atingir o meu principal objetivo que era desenvolver um jogo na sua totalidade, posso afirmar que me sinto realizado com o trabalho que desenvolvi ao longo destes meses. Apesar disto tenho noção que o projeto ainda não está pronto para venda comercial, algo que já sabia que iria acontecer desde o início, pois trata-se de um projeto que requer muito tempo no seu desenvolvimento e ainda mais tempo no seu aperfeiçoamento.

Acredito também ter atingido todos os pontos chave presentes no anteprojeto, e ter implementado mecânicas e componentes que apesar de não ter referido no anteprojeto tem o seu valor, pois tornam a experiência muito mais polida e divertida.

No que diz respeito a documentação, acredito que seja agradável, tentei orientá-la ao leitor comum escrevendo frases simples e evitando vocabulário técnico. O que por um lado torna difícil a elaboração de um desenvolvimento muito detalhado, mas em geral acredito ter encontrado um bom meio termo.

A nível geral, acho que este jogo revela a minha aptidão profissional, a nível de persistência, exigência pessoal, a capacidade de ser autodidata e a capacidade de finalizar aquilo que me proponho a realizar.

Posto isto, a minha autoavaliação seria de 19 valores de 20.

Bibliografia e Web grafia

Documentação Geral do Unity. Unity – Manual: Unity User Manual.
< https://docs.unity3d.com/Manual/index.html >. Acesso em 4 de junho de 2019.

Plugin de Animações para Paint.net.
< https://pixelbyte.itch.io/paint-net-sprite-plugin >. Acesso em 4 de junho de 2019.

Documento do Personagem Historio *Tarrare*
< https://pt.wikipedia.org/wiki/Tarrare >. Acesso em 4 de junho de 2019.

Plugin de Grelhas para Paint.net
< https://forums.getpaint.net/topic/1964-grid-maker-plugin-v30-updated-july-2-2014/ >.
Acesso em 4 de junho de 2019.

Tiled2Unity, Site
< https://seanba.com/tiled2unity > Acesso em 4 de junho de 2019.

Paint.net, Site
< https://www.getpaint.net/ > Acesso em 4 de junho de 2019.

BeepBox, Site
< https://www.beepbox.co > Acesso em 4 de junho de 2019.

Tiled, Site
< https://www.mapeditor.org/ > Acesso em 4 de junho de 2019.

WampServer, Download
< https://sourceforge.net/projects/wampserver/ > Acesso em 4 de junho de 2019.

VisualStudio, Site
< https://visualstudio.microsoft.com/pt-br/?rr=https%3A%2F%2Fwww.google.com%2F >
Acesso em 4 de junho de 2019.

Notepad++, Site
< https://notepad-plus-plus.org/download/v7.7.html >
Acesso em 4 de junho de 2019.

Quora, Fórum, questão
<https://www.quora.com/When-developing-my-video-game-should-I-use-a-game-engine-or-program-everything-with-C++>
Acesso em 4 de junho de 2019.

Gamasutra, Fórum
<https://www.gamasutra.com/blogs/ChrisHildenbrand/20111015/8669/2D_Game_Art_for_Programmers__part_1.php>
Acesso em 4 de junho de 2019.

Agora segue um conjunto de membros da comunidade, que foram cruciais no desenvolvimento deste projeto:

< https://www.youtube.com/channel/UCLyVUwlB_Hahir_VsKkGPIA>

< https://www.youtube.com/channel/UCIabPXjvT5BVTxRDPCBBOOQ>

< https://www.youtube.com/user/Brackeys>

< https://www.youtube.com/user/Unity3D>

< https://www.youtube.com/user/SykooTV>

< https://www.youtube.com/channel/UCHIKDALSPFzE3NhZ6k35pDQ>

< https://www.youtube.com/channel/UCd_lJ4zSp9wZDNyeKCWUstg>

Anexos

Scripts C: (Visual Studio)


BD_Inserter.cs


BD_Loader.cs


Bite_Manager.cs


Camera_Controller.cs


Flee_AI.cs


Hunger_Manager.cs


MainMenu_Controller.cs


NpcSpawner.cs


Player_Controller.cs


PlayerSpawner.cs


Range_Manager.cs


Raycast_Manager.cs


Score_Controller.cs


UI_Manager.cs

Scripts PHP: (notepad ++)


tarrare_connect.php


tarrare_highscore.php


tarrare_data.php


tarrare_insert_data.php