Instituto Superior de Engenharia de Lisboa (ISEL)
Departamento de Engenharia Eletrónica e de Telecomunicações e Computadores (DEETC)

LEIM
Licenciatura em Engenharia Informática e Multimédia
Unidade Curricular de Projeto

Unity Mobile Game

André Faria 47455

Sara Nobre 47504

Orientador

Professor Pedro Fazenda

september, 2023

RESUMO

Este documento é o relatório final de um projeto que foi realizado no âmbito da unidade curricular de Projeto e tem por objetivo dar a conhecer o trabalho realizado e as diferentes fases do seu desenvolvimento.

O projeto, como o nome indica, é a criação de um videojogo para telemóvel desenvolvido em Unity Game Engine. A ideia surgiu devido à paixão e gosto por videojogos e programação.

O projeto foi divido em três partes:

1. Na implementação do motor de jogo e das funcionalidades que permitem ao utilizador "jogar";

2. À criação do motor que gera o mundo e a interligação deste motor com o do videojogo;

3. No design e implementação de uma interface gráfica que permite melhorar a experiência do utilizador e que conecta todas as funcionalidades da aplicação.

ABSTRACT

This document refers to the final report of the project carried out as part of the Projeto curricular unit and it intends to make the developed work and the many stages of development known.

The project, as the name infers, is the design and development of a video game for mobile using Unity Game Engine. It arose as a result of a combined passion for video games and programming.

The work was split into three sections:

1. Development of the game engine and features that allow the user to "play";
2. Creation of the engine that produces the world and stages, and its integration with the game;
3. Design and implementation of a graphical interface that enhances the user experience and connects all of the application's features.

AGRADECIMENTOS

Este trabalho não ficaria completo sem agradecer a todos que ajudaram na sua concretização.

Em primeiro lugar, queremos agradecer ao Professor Pedro Fazenda pela sua orientação, apoio e disponibilidade.

Ao professor Hélder Bastos pela sua disponibilidade, apoio e por aceitar ser o arguente deste projeto.

Não se pode esquecer dos docentes e colegas do ISEL, e também todos que aceitaram testar e ajudar com o nosso trabalho. Os nossos agradecimentos finais vão para as nossas famílias pelo apoio e incentivo ao longo da realização deste projeto.

ÍNDICE

LISTA DE TABELAS

LISTA DE FIGURAS

ABREVIATURAS

ISEL - Instituto Superior de Engenharia de Lisboa
LEIM - Licenciatura em Engenharia Informática e Multimédia;
SUS - System Usability Scale

CAPÍTULO 1

INTRODUÇÃO

1.1 Contexto

Este documento aborda o projeto realizado para a unidade curricular de Projeto. Este trabalho permitiu o desenvolvimento de um videojogo para Android utilizando-se a plataforma do Unity Game Engine [6].

O Unity permite desenvolver aplicações e/ou simulações para qualquer tipo de sistema (consolas, computadores e mobile). No caso deste projeto, o jogo a desenvolver vai ser um videojogo para Android.

1.1.1 Conceito do videojogo

O projeto consiste em um videojogo de estratégia do estilo *rogue-like* em que o jogador atravessa níveis desafiantes nos quais pode encontrar inimigos. Para os derrotar, irá recorrer a cartas que simulam as ações das personagens pertencentes à sua equipa.

Rogue-like é um conceito de videojogo onde o mundo e níveis são gerados aleatoriamente. A derrota do jogador implica que comece uma nova tentativa desde o início.

1.2 Objetivos do videojogo

O objetivo do jogo é atravessar níveis adquirindo equipamentos para chegar o mais longe possível. Em cada nível, o jogador pode deparar-se com uma escolha ou uma sala com inimigos. No caso da sala com inimigos, o jogador tem no seu arsenal três personagens, cada uma com as suas cartas, utilizando-as para derrotar os inimigos.

O jogo é desafiador para o utilizador incidindo duas vertentes em termos de dificuldade: sorte e perícia.

1.3 Software/Plataformas

Através do Unity consegue-se desenvolver todos os mecanismos e funcionalidades do sistema, tal como a *gameplay* (jogabilidade), a geração do mundo e a interface.

O videojogo vai ser desenhado no estilo de arte pixel (animações, efeitos, personagens, interface, entre outros...) através da ferramenta Aseprite [1]. Os sons vão ser gravados a partir do Audacity [2].

Pretende-se lançar o videojogo através do Google Play [4] após ser avaliado por alguns utilizadores.

1.4 Motivação

Existem milhares de videojogos para Android, mas a maior parte são jogos em que os desenvolvidores focaram-se na monetização. No entanto, este jogo foi desenvolvido de modo a focar-se no utilizador recompensando-o por jogar. Isto é possível criando-se uma jogabilidade interativa e desafiante para o manter entretido.

No futuro pretende-se adicionar novos conteúdos para que o videojogo não se torne monótono.

1.5 Estrutura do documento

O documento é composto por **VII** capítulos no qual se encontra o trabalho realizado ao longo do semestre.

O capítulo **I** aborda de uma forma geral o conceito do trabalho e as suas funcionalidades.

No **II** e **III** capítulos vai ser abordada a fase de análise onde vão ser apresentados os planos do projeto e trabalhos relacionados.

No **IV** vai ser abordado o modelo de desenvolvimento proposto como os requisitos funcionais, fundamentos e a abordagem.

No capítulo**V** vai ser elaborado as partes mais relevantes da implementação do modelo proposto.

O capítulo **VI** relata o estado atual do projeto, bem como resultados a questionários e discussões aos mesmos.

O **VII** capítulo vai abordar as conclusões tiradas da realização do projeto e vai elaborar sobre o percurso de realização, referindo dificuldades e alterações. Também vai definir funcionalidades e otimizações que podem ser implementadas num futuro próximo.

CAPÍTULO 2

ANÁLISE

Este capítulo vai abordar a fase inicial de planeamento e análise que ajuda a conceber o conceito do projeto e as suas funcionalidades. Foi elaborado um Game Design Document (GDD) que permite estabelecer e planear todas as fases do desenvolvimento de um videojogo.

2.1 Game Design Document - Resumo

Um videojogo, tal como aplicações e softwares, tem bastantes fases de desenvolvimento onde é necessário à priori um planeamento de forma a definir os conceitos, funcionalidades e público alvo.

2.1.1 Público Alvo

O público alvo apontado é pessoas entre os jovens e os jovens adultos, uma vez que são estes utilizadores que usufruem mais do seu tempo para jogar videojogos.

2.1.2 Mecânicas do jogo

O jogador inicia um novo jogo no qual vai ser gerado um mundo com níveis aleatórios. Sempre que vence um nível recebe uma recompensa que aumenta as habilidades das suas personagens. No caso de perder, o jogador volta ao menu inicial de modo a começar uma nova aventura.

O jogo é um jogo de cartas por turno onde o jogador utiliza-as para derrotar inimigos. Contém na sua posse três personagens, cada uma com baralhos de cartas diferentes.

À medida que o jogo vai progredindo, os níveis vão aumentando progressivamente a dificuldade de modo a seguir os melhoramentos das personagens, tornando o jogo constantemente desafiante.

Em termos de combate o jogo vai funcionar da seguinte forma:

- Jogador recebe cartas para a mão;
- Troca de personagens e utiliza cartas;
- Chega ao limite de energia do turno;
- Acaba o turno passando à vez do(s) inimigo(s);
- Inimigos atacam personagens;
- Acaba o turno e retorna à vez do jogador.

Este loop termina quando o jogador ganhar (derrotar todos os inimigos do nível) ou perder (todas as personagens morrerem). Se ganhar, recebe uma melhoria e continua a aventura. Se perder, ganha uma recompensa correspondente ao seu desempenho e até onde chegou terminando a aventura.

2.1.3 Personagens

O jogador tem na sua posse três personagens distintas, cada uma é representada por uma cor ou classe. O conjunto das personagens chama-se equipa e esta tem por turno uma energia máxima para usar habilidades (cartas).

As personagens têm a vida inicial e habilidades distintas das outras, isto é, cada personagem foca-se mais em certos efeitos.

2.1.4 Níveis

Por mundo existem níveis que podem ser:

- Combate - Níveis com inimigos:
  - Normal - Inimigos normais;
  - Elite - Inimigos mais difíceis;
  - *Boss* - Último obstáculo do mundo com um inimigo mais difícil.
- Evento - Um obstáculo com escolhas e efeitos para ultrapassar;

2.1.5 Interface

A interface tem de ser adequada ao projeto e ao público alvo. Como se trata de um jogo para telemóvel, a interface tem de ser simples e intuitiva.

Para se tornar adequada ao projeto e ao tema, vai ser desenhada com o mesmo estilo de arte do jogo.

CAPÍTULO 3

TRABALHOS RELACIONADOS

O projeto foi concebido inspirando-se em alguns aspetos de jogos conhecidos.

3.1 Celeste

O Celeste [5] é um jogo de plataforma para PC e consolas no qual o estilo de arte é a arte pixel.

Figura 3.1: Imagem *in-game* de Celeste

3.2 Slay the Spire

O Slay the Spire [3] é um jogo de estratégia por turno para PC e consolas onde a jogabilidade se assemelha ao projeto a ser desenvolvido.

Figura 3.2: Imagem *in-game* do combate de Slay the Spire

CAPÍTULO 4

MODELO PROPOSTO

Neste capítulo vai ser referido os requisitos necessários a implementar, visíveis como invisíveis. Também vai ser elaborada uma introdução ao software utilizado na realização do projeto e por fim vai ser referido a abordagem escolhida para se prosseguir à próxima fase de implementação.

4.1 Requisitos

Para organizar o desenvolvimento do projeto, foi planeado todas as funções (requisitos funcionais) e atributos (requisitos não funcionais) do sistema. Tal como foi lecionado na unidade curricular de projeto, o significado destes termos é o seguinte:

- **Requisitos Funcionais** - Funcionalidades que o sistema deve oferecer (exemplo: o jogador deve puder escolher o próximo nível);
- **Requisitos Não Funcionais** - Características não funcionais que se aplicam ao sistema (exemplo: o jogo deve ser intuitivo).

4.1.1 Requisitos Funcionais (Funções de Sistema)

Uma função de sistema pode ser categorizada nos seguintes aspetos:

- **Evidente** - A função tem de ser realizada e o utilizador toma conhecimento da mesma;
- **Invísivel** - A função tem de ser realizada mas o utilizador não a conhece;
- **Adorno** - Quando a função é opcional e não prejudica as restantes.

Com os conceitos definidos passou-se para o planeamento das funções de sistema a se definir. A seguinte tabela mostra os principais requisitos funcionais definidos:

Requisito Funcional	Categoria
Começar nova aventura	Evidente
Escolher o nível	Evidente
Ter controlo das suas ações	Evidente
Visualizar o seu progresso	Evidente
Criação aleatória do mundo	Evidente
Inicialização dos objetos do jogo	Invísivel
Feedback após ação	Adorno

Tabela 4.1: Requisitos Funcionais

4.1.2 Requisitos Não Funcionais (Atríbutos de Sistema)

Estes requisitos devem ser categorizados em dois tipos:

- **Desejável** - Requisito que deve ser alcançado, mas que não comprometa o desempenho final;
- **Obrigatório** - Requisito essencial para o desempenho final.

A seguinte tabela demonstra os requisitos não funcionais planeados:

Requisito Não Funcional	Categoria
Inicialização rápida dos objetos	Obrigatório
Compatibilidade com Android	Obrigatório
Sistema intuitivo e agradável	Desejável
Monetização (Exemplo: Ads)	Desejável

Tabela 4.2: Requisitos Não Funcionais

4.2 Fundamentos

Este projeto assenta na criação de um ambiente virtual destinado ao público alvo. Esse ambiente é um videojogo 2D para Android e permite a interação do utilizador com o ambiente em tempo real.

4.2.1 Tecnologias utilizadas

Durante o percurso deste projeto, foi necessário recorrer a diversas tecnologias para desenvolver a aplicação ou auxiliar tal processo.

Figma

O Figma é uma **aplicação *web* dedicada ao design de interfaces** com intenções de ajudar os utilizadores a planearem as interfaces para as suas aplicações podendo ainda simular as interações entre as diferentes interfaces/janelas.

Esta aplicação foi escolhida devido ás diversas ferramentas que disponibiliza e o fácil uso da mesma.

Aseprite

Um **editor de imagem** designado em especifico para o desenho de **arte pixel**. Contem diversas ferramentas que permitem a edição e animação de imagens/arte pixel.

O Aseprite foi apenas um de vários programas encontrados que se destacou mais. Além de ser fácil de usar, as ferramentas básicas são impulsivas e rápidas de se habituar.

Firebase

Firebase é um **conjunto de recursos de computação em nuvem** (*cloud computing*) fornecido pela Google. Alguns dos recursos disponibilizados são serviços, base de dados, autenticação e integração que podem ser usados por uma variedade de aplicações como Android, iOS, Java, Unity, PHP entre outros.

Para este projeto, o plano é usar em particular o recurso de base de dados. O projeto terá informação que pode ser útil armazenar numa base de dados como dados de jogo do utilizador (tempo gasto, nível, ...) ou mesmo informação do jogo (por exemplo mundos ou inimigos).

Unity

O Unity é um **motor de jogo** para diversas plataformas desenvolvido pela equipa Unity Technologies. Atualmente suporta diversas plataformas como computador, telemóvel, consola e mesmo realidade virtual sendo possível criar ambientes em 2D ou 3D. É particularmente conhecido no desenvolvimento de jogos de telemóvel sendo bastante popular com os desenvolvidores independentes (*indie games*).

Grande parte do projeto é realizado utilizando este motor. O jogo em si é desenvolvido unicamente em Unity usando a sua programação para gerar as ações desejadas. Quanto aos ficheiros de arte criados pelo Aseprite, o Unity consegue interpretar e aplicá-los a um *canvas* no projeto facilitando consideravelmente a relação da arte com o código.

Google Play

Um **serviço de distribuição digital** desenvolvido e controlado pela Google conhecido por Google Play. É a loja oficial de aplicações para os aparelhos com o sistema operativo Android sendo uma fonte de confiança para instalar aplicações desenvolvidas com o *software* Android.

4.3 Abordagem

Tendo em consideração os requisitos estabelecidos e o objetivo deste projeto, é fundamental articular diversos fatores para a criação da aplicação desejada. Estes fatores podem ser divididos nos seguintes:

4.3.1 Inspiração nas Aplicações Existentes

Inspiração e motivação foi adquirida através de outras aplicações semelhantes de forma a ajudar a modelar os objetivos do projeto.

4.3.2 Uso de Ferramentas

Em seguida, foram usadas várias ferramentas para criar a aplicação. O Figma ajudou a planear a parte gráfica da aplicação. O Aseprite foi usado para criar as sprites e animações. Finalmente, usufruiu-se do Unity implementar o jogo em si.

No futuro, planeia-se utilizar a Firebase para guardar dados do utilizador, objetos gráficos e dados do jogo. Por fim pretende-se lançar o videojogo através da Google Play.

4.3.3 Modelagem Original

Devido à intenção de usar o mínimo de recursos online, uma parte importante foi criar modelos originais. Isto faz a aplicação destacar-se como única, sendo um dos motivos de escolha da arte pixel graças à sua mistura de apelativo e simplicidade.

CAPÍTULO 5

IMPLEMENTAÇÃO DO MODELO

Este capítulo refere-se à implementação do modelo proposto, recorrendo ás tecnologias descritas no capítulo anterior.

Fases de implementação

O desenvolvimento do modelo foi separado em duas fases: fase de desenho e fase de programação.

A fase de desenho corresponde à modelação 2D recorrendo-se ao Aseprite[1] onde foram modelados os diferentes objetos, personagens, itens, ...

A fase de programação corresponde à implementação da aplicação/jogo através do Unity[6]. Nesta fase dividiu-se o desenvolvimento em diferentes sub fases de modo a lentamente ir-se aumentando a complexidade da aplicação.

Diagrama do Projeto

A aplicação pode ser dividida em diferentes partes que comunicam entre si através do motor de jogo (composto pelas principais classes denominadas de "*Engine*").

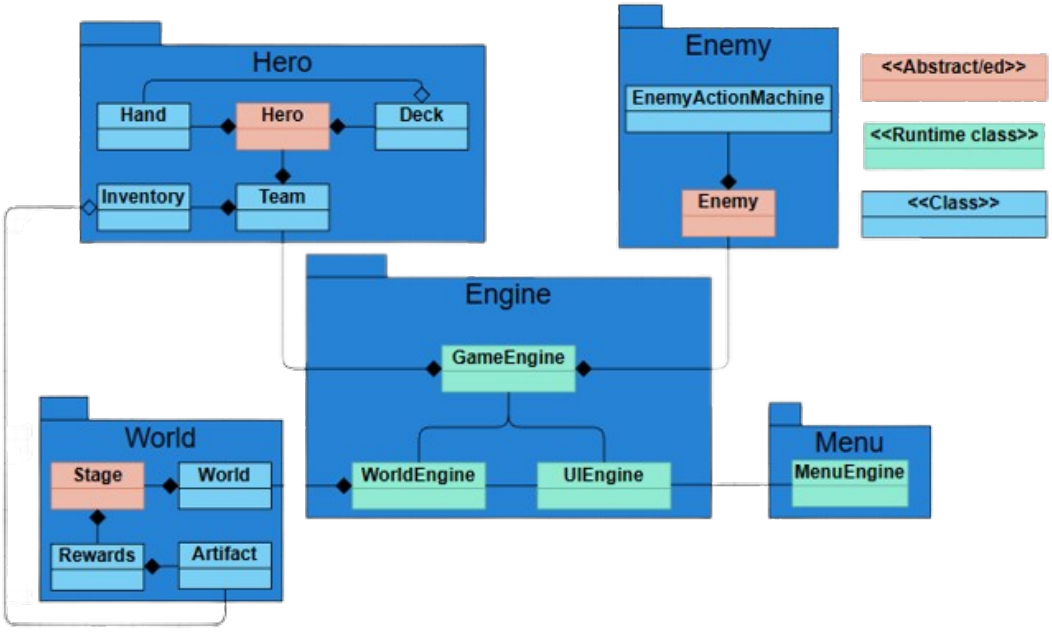

Figura 5.1: Diagrama de classes da implementação (Unity)

5.1 Aseprite - Modelação 2D

Aseprite foi o editor de arte escolhido para o projeto. Este é dedicado à arte pixel e contem várias ferramentas fáceis de usar.

5.1.1 Base da aplicação

Após criar um novo ficheiro ou abrir um existente, depara-se com a interface principal do Aseprite (figura 5.2). Esta está dividida em quatro partes principais:

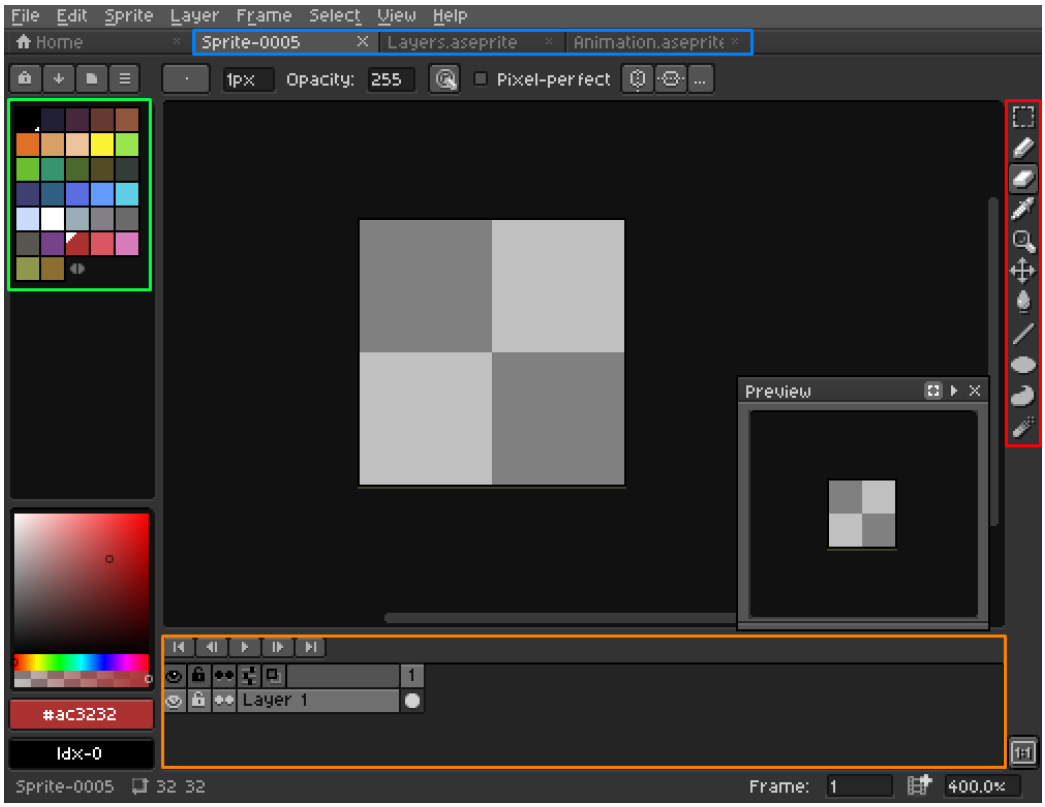

Figura 5.2: Interface principal do Aseprite

Ficheiros (azul)

Atrás dos ficheiros está o menu inicial com informação geral (últimos ficheiros acedidos, atualizações). Sempre que um ficheiro é aberto/criado, aparece ao lado da janela inicial como se pode ver na imagem (o destaque azul contém os três ficheiros criados para esta demonstração)

Ferramentas (vermelho)

No lado direito da janela encontra-se a lista de ferramentas necessárias para o uso do editor. Por ordem descendente: **selecionar** (livre, circulo, retângulo ou zona completa da mesma cor), **caneta** ou spray, **borracha**, selecionar/**copiar cor**, **zoom**, **mover**, **balde** (preenche a área), desenha uma **linha**, desenha um **circulo ou retângulo** (contorno ou preenchidos), desenha o **contorno** da forma e preenche automaticamente, **desfoca** ou mistura pixeis.

Facilmente nota-se que há uma razão para ter estas ferramentas em tão fácil acesso visto que metade delas são usadas com bastante frequência, até para as obras mais simples.

Palete de cores (verde)

O Aseprite disponibiliza várias paletes como amostra bastando alterá-la no menu. Além disto, permite organizar a palete como deseja podendo sempre alterar a cor na palete. Abaixo da palete está a gama de cor contendo um canal dedicado ao *alpha* (transparência da cor) e as duas cores que pode intercalar com um atalho.

Camadas e *frames* (laranja)

Na parte inferior da tela situa-se a lista de camadas e *frames*. Estas serão explicadas em maior detalhe posteriormente.

Neste caso (figura 5.2), existe apenas uma camada "Layer 1". Cada camada tem um *trigger* de visibilidade e um de permissão (cadeado fechado impossibilita qualquer alteração na camada).

5.1.2 Camadas

A gestão das camadas é algo simples mas essencial para o proveito máximo da aplicação. Com a possibilidade de ter várias camadas, isto permite separar diferentes objetos conforme desejar. No entanto, há que ter atenção em que posição estão cada camada. A imagem seguinte demonstra como a posição das camadas afeta o produto final onde cada camada tem apenas uma palavra desenhada nela (camada "Topo"tem a palavra "TOPO", ...).

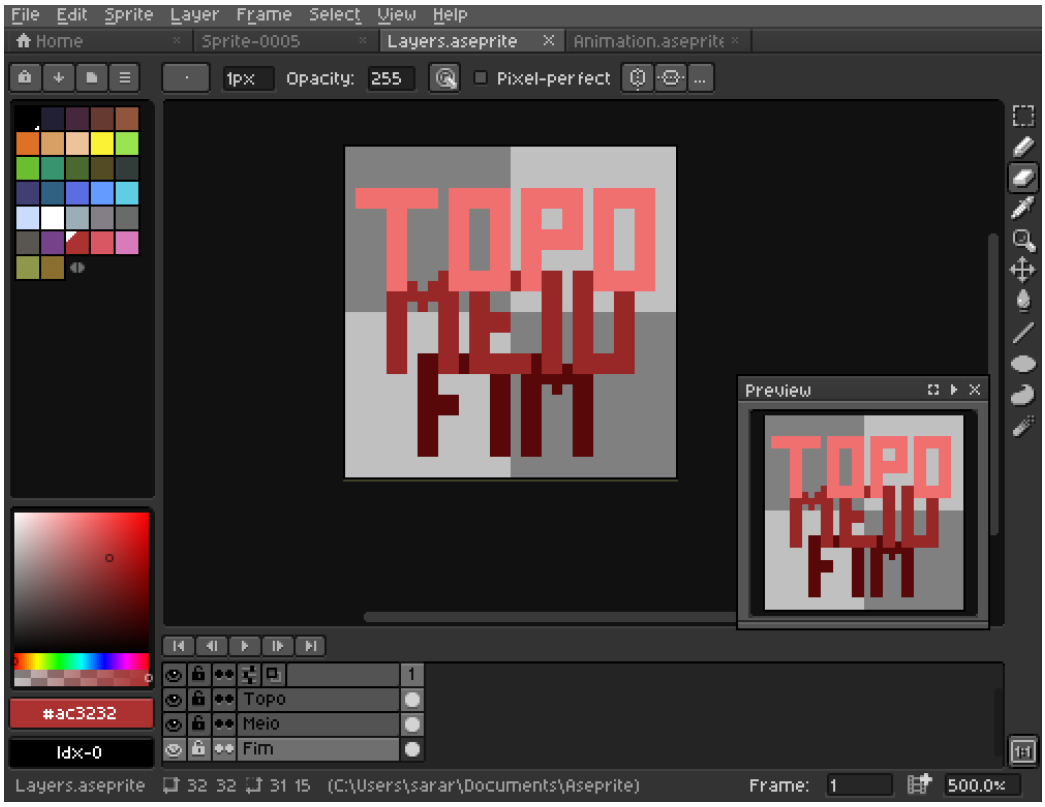

Figura 5.3: Exemplo do uso de camadas

Como o Aseprite considera a primeira camada como a mais superior, quando se visualiza a obra final com todas as camadas visíveis, a camada "Topo"fica sobreposta à camada "Meio"e esta à camada "Fim".

Isto pode não ser extremamente útil num desenho, contudo, numa animação, é onde brilha. A possibilidade de conseguir criar *foreground* e *background* simplifica a animação mais complexa poupando tempo de desenho e planeamento podendo mover as camadas diferentes sem afetar as outras.

5.1.3 *Frames*

As *frames* estão associadas à animação. Cada *frame* fica cronologicamente da esquerda para a direita tendo os números acima da primeira camada a numerá-las. Acima destes pode-se adicionar etiquetas como auxilio para a animação como demonstra o caso seguinte.

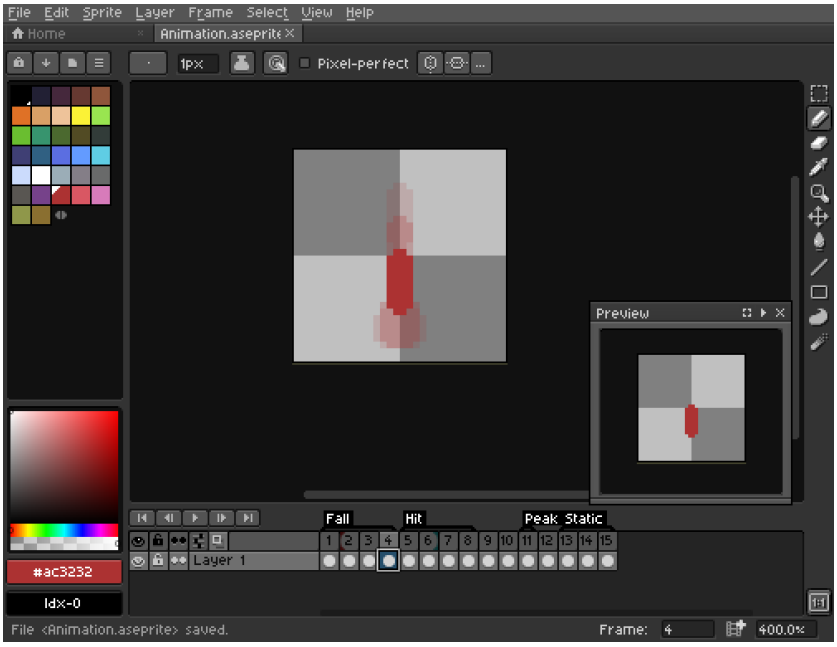

Figura 5.4: Exemplo de uma animação no Aseprite

Por exemplo, uma simples animação de uma bola a cair. Está sinalizado cada parte do movimento: queda (*fall*), impacto (*hit*), pico do ressalto (*peak*) e momento em que fica estática (*static*).

O efeito observado (desfasamento das *frames*) é possível graças ao efeito *onion skin* que facilita a comparação entre *frames*. Neste caso, esta "pele"está entre as *frames* 2 a 6. É possível observar melhor na figura abaixo.

Figura 5.5: Imagem ampliada das *frames* do exemplo

5.1.4 Recursos criados

Devido ao grande volume de recursos a criar, uma organização destes é necessária. Como tal, foi dividido em cinco grandes partes:

- **Artefactos** - Itens considerados como artefactos ou relíquias;
- **Cartas** - Sub-pastas para cada personagem em que cada uma contem as cartas especificas à mesma;
- **Personagens** - Todos os elementos associados diretamente à personagem (ícone de personagem, animações);
- **Interface** - Todos os elementos relacionados com os menus ficam na sub-pasta enquanto o resto dos elementos da interface ficam fora;
- **Mundo** - Haverá uma sub-pasta para cada mundo, esta contendo ainda duas sub-pastas para os inimigos (semelhante à pasta das personagens) e outra para os *wallpapers* que podem intercalar durante os níveis.

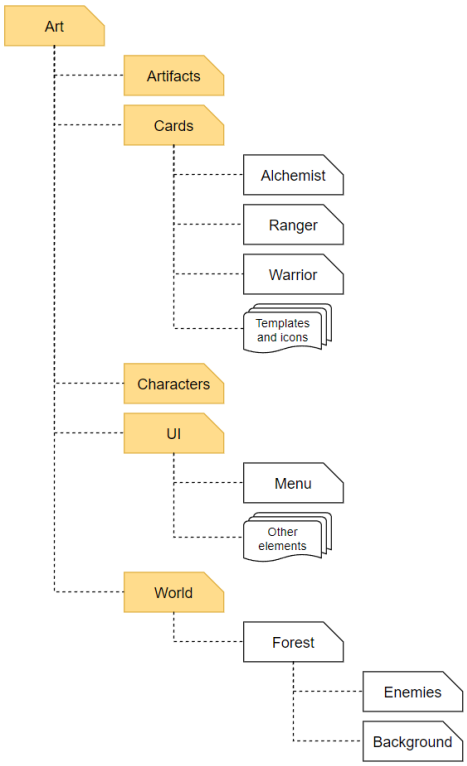

Figura 5.6: Organização dos ficheiros de arte

5.2 Unity - Jogo

Através do Unity é possível desenvolver todas as funcionalidades e componentes do projeto. Este subcapítulo vai focar-se no desenvolvimento/programação do videojogo.

5.2.1 Introdução à implementação

Nesta secção, vai ser discutido a abordagem adotada durante o desenvolvimento do projeto. A implementação da aplicação foi cuidadosamente planeada e dividida em fases distintas, com base nas funcionalidades mais importantes e nas dependências que essas funcionalidades têm umas com as outras. Esta estratégia de desenvolvimento foi adotada por várias razões importantes:

1. **Priorização de Funcionalidades**: A escolha de dividir o desenvolvimento em fases permitiu um aumento na concentração nas funcionalidades mais cruciais, o que garante que as partes mais essenciais da aplicação estivessem prontas e testadas mais cedo no processo de desenvolvimento.

2. **Gestão de Recursos**: A alocação eficiente de recursos, incluindo tempo e mão-de-obra, foi fundamental durante a fase de desenvolvimento. Ao dividir o desenvolvimento em fases, conseguiu alocar-se recursos de forma mais eficaz, evitando desperdícios e garantindo um uso otimizado dos recursos disponíveis.

Nas próximas secções deste subcapítulo, vai ser descrito detalhadamente cada uma das fases de desenvolvimento, destacando as funcionalidades específicas que foram implementadas em cada fase e como essas funcionalidades se relacionam para criar a aplicação final.

5.2.2 Motor de jogo

O motor de jogo é o conjunto de todas as classes responsáveis por manipular o jogo. Estas classes são chamadas de *scripts* e são associados a "*GameObjects*"(objetos no mundo). Os *scripts* são iniciados quando o Unity entra em "*Play Mode*"ou quando o jogo é iniciado no dispositivo, sendo denominados de *scripts "Runtime"* por correrem em um ambiente em execução.

Comunicando entre si, estes *scripts* manipulam toda a aplicação e permitem uma sincronização entre todas as funcionalidades do sistema. Os scripts que funcionam como motor de jogo são:

- *GameEngine* - Responsável por controlar os mecanismos do jogo;
- *WorldEngine* - Cria e carrega mundos e níveis, bem como as recompensas;
- *UIEngine* - Controla a interface a mudança de ecrã;
- *MenuEngine* - Manipula o menu principal do jogo.

Por dentro de cada motor, existem outros scripts de forma a dividir funcionalidades e reduzir o acoplamento do sistema.

5.2.3 Tratamento dos ficheiros de dados

Como a aplicação se trata de um jogo aleatório, a maioria do conteúdo tem de ser gerado dinamicamente. Para se obter esse resultado recorreu-se a ficheiros de dados do tipo **JSON** que permitem guardar as informações dos objetos (personagens, itens, mundos, ...) além de permitirem que a aplicação seja escalável, ou seja, que se possa adicionar novas conteúdos sem ter de se alterar código.

5.2.4 Níveis de combate

Nesta secção vai ser descrito o processo de desenvolvimento do coração da aplicação: o nível de combate. Este inclui a inicialização das personagens e inimigos, o sistema de cartas e a criação do jogo por turnos.

Para o nível de combate foi necessário implementar os seguintes conceitos:

- Inicialização das personagens;
- Inicialização dos inimigos;
- Sistema de cartas;
- Implementação do jogo por turnos;
- Máquina de decisão de ataques dos inimigos;
- Aumento de dificuldade.

Inicialização das personagens

Como referido na secção anterior, as personagens vão ser lidas e carregadas a partir de um ficheiro JSON. Através da classe *HeroInitializer* que desserializa o ficheiro num dicionário para criar os objetos *"Hero"*. Estes objetos são depois enviados para o motor de jogo para serem carregados visualmente.

Inicialização dos inimigos

Os inimigos seguem o mesmo formato que as personagens, a única alteração é que carregam também os dados dos seus ataques. Para tal, a classe *"Enemy"* cria também o objeto *"EnemyActionMachine"* que é a maquina de decisões responsável em decidir os ataques dos inimigos.

Sistema de cartas

Durante o nível de combate o jogador derrota os inimigos utilizando cartas que simulam ações das personagens. Estas cartas, como o resto dos objetos, têm de ser carregadas dinamicamente. Implementou-se uma classe que inicializa as cartas recorrendo ao ficheiro da *"Deck"* da personagem, sendo este um ficheiro do tipo JSON com as informações das cartas que a personagem possui (incluindo custo, efeitos, quantidade, imagens,...).

Cada personagem tem as suas respetivas cartas e cada carta tem o seu respetivo custo e efeito. Para estas cartas serem utilizadas basta que o jogador arraste-as para o respetivo alvo. Para melhorar a experiência do mesmo adicionou-se uma mira nas cartas e um efeito visual quando a carta está sobre um alvo, como mostra a seguinte figura 5.7.

Figura 5.7: Mira na carta selecionada pelo jogador

Foram criados seis objetos no mundo que vão suportar as cartas na mão da personagem selecionada (o jogador terá no máximo seis cartas na mão por personagem). Cada objeto contém o *script "CardEngine"* que contém as funcionalidades necessárias para o jogador utilizar as cartas. Sempre que o jogador altera de personagem, o motor de jogo altera os objetos das cartas nos *scripts* referidos anteriormente, atualizando visualmente o jogo.

Implementação do jogo por turnos

Completando o nível de combate, falta referir a implementação do aspeto ***"turn based"***: o jogador vai utilizar as suas personagens e respetivas cartas e quando estiver sem energia, termina o seu turno, passando à vez dos inimigos.

O jogador pode decidir terminar o seu turno quando quiser sendo apenas forçado a tal quando ficar sem energia. No turno dos inimigos estes realizam os seus ataques e acabam o seu turno, retornando ao turno do jogador.

Máquina de decisão dos inimigos

A máquina de estado dos inimigos é uma classe que é responsável por **armazenar**, **decidir** e **executar** os ataques dos inimigos. Isto é realizado através de uma máquina de decisões que recebe informações sobre o estado atual do jogo e verifica algumas condições, decidindo o ataque do inimigo com base no resultado das mesmas. Essas condições são:

- **Verificar defesa**: se a sua vida é inferior a um limite, tem mais chance de realizar ações de defesa;

- **Verificar estados**: se as personagens adversárias não têm nenhum estado negativo (veneno, eletrocutado, ...), tem mais chance de usar ações que provocam estados negativo.

No caso de nenhuma condição ser atingida, o inimigo tem mais chance de realizar ataques normais.

Os próximos ataques podem ser visualizados em cima de cada inimigo com a informação do valor e do tipo de ação. Por exemplo, a figura ao lado (figura 5.8) demonstra um ataque com oito de dano.

Figura 5.8: Próximo ataque de um inimigo.

Aumento de dificuldades

Durante o jogo existem dois aumentos de dificuldade:

- **Dificuldade do nível**: À medida que os turnos vão passando, os ataques dos inimigos ficam mais fortes, isto promove a que o jogador não fique infinitamente no mesmo nível.
- **Dificuldade progressiva**: Sempre que o jogador conclui um nível ou mundo a dificuldade aumenta, isto para que os inimigos consigam acompanhar o crescimento das personagens e para tornar o jogo mais desafiante.

5.2.5 Níveis de escolhas (Eventos)

Para tornar o jogo menos monótono e mais interativo, foi implementado um nível de escolhas que atribui ao jogador um desafio e três possíveis decisões. Para se implementar agrupou-se os requisitos nos seguintes conceitos:

- Inicialização do desafio e escolhas;
- Determinação da recompensa ou penálti;
- Representação gráfica.

Inicialização do desafio e escolhas

Usufruindo do mesmo método referido anteriormente, o desafio e escolhas são carregados a partir de um ficheiro do tipo JSON, que é escolhido aleatoriamente dependendo do mundo atual.

Determinação da recompensa ou penálti

As recompensas ou penáltis são extraídos do mesmo ficheiro e são atribuídos conforme a decisão do jogador perante o desafio.

Representação gráfica

Associando-se cada escolha a um objeto no mundo permite que o jogador consiga pressionar e visualizar as opções.

5.2.6 Geração do mundo

Um mundo é composto por um conjunto de *Stages* que podem ser de combate ou de escolhas, sempre que o jogador completa um recebe uma recompensa.

Nesta secção vai ser referido como é que o mundo é gerado e como é que é determinado o que é cada *stage*.

Para se implementar esta funcionalidade separou-se o conceito nas seguintes fases:

- Geração do mundo e dos níveis;
- Progressão do jogador

Geração do mundo e dos níveis

Num ficheiro do tipo JSON existe uma lista de mundos possíveis, que são extraídos pelo motor de jogo de forma a criar níveis *"Stages"*. Cada nível é composto por um a três *stages* atribuindo a opção de escolha ao jogador.

Um mundo é gerado sempre que o jogador completa um mundo ou começa um jogo novo. Os *stages* são escolhidos de forma aleatória a partir de probabilidades fixas.

Os parâmetros definidos aleatoriamente são:

- Quantidade de *"Stages"* por nível;
- Tipo de *"Stage"* (Combate ou Evento);
- Recompensas;
- Combate:
  - Dificuldade;
  - Inimigos
- Evento:
  - Tipo de evento

5.2.7 Sistema de recompensas

Sempre que o jogador completa um nível é necessário recompensá-lo, uma vez que a dificuldade vai aumentando progressivamente, o jogador precisa de uma forma de acompanhar.

Como recompensas o jogador pode obter dinheiro e itens, o dinheiro servirá no futuro para melhorar as personagens permanentemente e os equipamentos (**Artefactos**) melhoram as personagens durante a aventura atual. Para se implementar este sistema dividiu-se nas seguintes fases:

- Criação das classes Inventário e Artefacto;
- Inicialização dos Artefactos;
- Ligação com os níveis

Criação das classes Inventário e Artefacto

Para guardar cada artefacto foi criado uma classe *"Inventory"* responsável por armazenar e executar os *Artefactos*. Como cada artefacto tem o seu efeito especial foi necessário criar uma classe especial onde são criados os métodos dos artefactos.

Cada artefacto só é executado numa ação definida, que pode ser:

- Inicio do nível;
- Inicio do turno;
- Carta é jogada;
- Fim do turno;
- Fim do nível;
- Passivo (ativa quando é adquirido).

Inicialização dos Artefactos

Repetindo o mesmo formato tornando a adição de artefactos um processo simples, o jogo extrai os mesmos de um ficheiro do tipo JSON de forma a criar os objetos.

Ligação com os níveis

Após cada nível ser gerado, os artefactos são criados ao mesmo tempo e preparados. Um nível tem na sua posse a informação de todas as recompensas/penáltis que pode atribuir. Quando são completados são gerados objetos gráficos, carregados com a informação da recompensa, para o jogador escolher e/ou ser informado da mesma.

5.2.8 Interface e menus - UI

Um jogo é uma aplicação gráfica, ou seja, deve-se focar no aspeto gráfico e na experiência do utilizador. O design e implementação de toda a interface do jogo foi de acordo com os seguintes conceitos:

- Público alvo;
- Conceito do jogo;
- Software/Dispositivo.

Isto permitiu desenvolver uma interface simples e dedicada ao jogo concebido.

Menu de inventário

O menu principal desenvolvido que apoia o jogador durante o seu jogo é o menu de inventário que lhe permite consultar o seguinte:

- Menu de opções (Options Menu): Composto por informação de utilizador, algumas estatísticas e opções de som;
- Menu de mala (Bag Menu): Que contem todos os artefactos adquiridos pelo jogador e as suas informações;
- Menu de barulho (Deck Menu): Composto por conter todos as cartas de todas as personagens e a informação detalhada de cada carta;
- Menu de mapa (Map Menu): Que contém o mapa do mundo atual e a opção para desistir.

Figura 5.9: Menu do jogo.

5.3 Conclusões

Nas conclusões deste capítulo de implementação do modelo, podemos destacar alguns pontos-chave:

1. **Fases de Implementação**: A divisão do desenvolvimento em fases, como a fase de desenho e a fase de programação, permitiu uma abordagem mais organizada e eficiente na implementação do modelo. Isso ajudou a priorizar as funcionalidades essenciais e a gerir os recursos de forma mais eficaz.

2. **Motor de Jogo**: A implementação do motor de jogo é vital, pois controla todas as interações e lógica do jogo. A divisão das funcionalidades em diferentes *scripts*, como *GameEngine*, *WorldEngine* e *UIEngine*, ajuda a manter um código mais organizado e modular.

3. **Tratamento de Dados**: O uso de arquivos *JSON* para armazenar dados do jogo, como personagens, cartas e artefactos, permite uma fácil expansão do conteúdo do jogo sem a necessidade de alterar o código. Isso torna o jogo mais escalável.

Em resumo, a implementação do modelo de jogo envolveu uma série de fases e elementos cruciais para criar uma experiência de jogo envolvente e escalável. A divisão cuidadosa das funcionalidades e o uso de ferramentas adequadas desempenharam um papel fundamental na realização deste projeto.

CAPÍTULO 6

RESULTADOS E DISCUSSÕES

Neste capítulo vai ser apresentado o estado atual do jogo e os resultados a alguns questionários efetuados a utilizadores. Isto permitiu obter avaliações sobre o estado atual do jogo de forma a melhorar futuramente com base nos resultados obtidos.

6.1 Estado do jogo

A implementação do jogo encontra-se na fase de *"alpha"*, na qual as mecânicas do jogo estão definidas, mas ainda falta algumas funcionalidades e conteúdo.

6.2 Avaliações com utilizadores

A avaliação foi realizada através do ***Google Forms*** e os utilizadores foram submetidos a um questionário simples e anónimo.

6.2.1 Informação do Utilizador

Primeiro foram submetidos a algumas questões de forma a serem caracterizados:

Qual o seu género

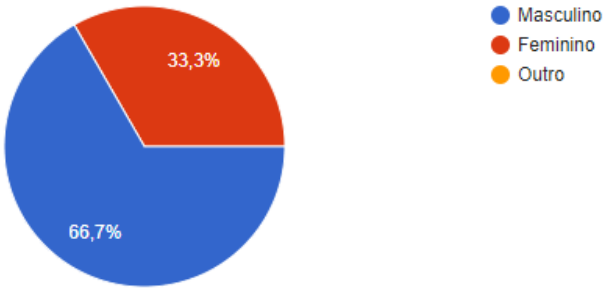

Figura 6.1: Gráfico das respostas ao género

Qual a sua idade

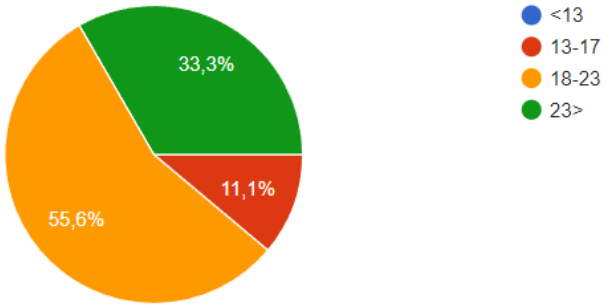

Figura 6.2: Gráfico das respostas à idade

Com que frequência utiliza o telemóvel

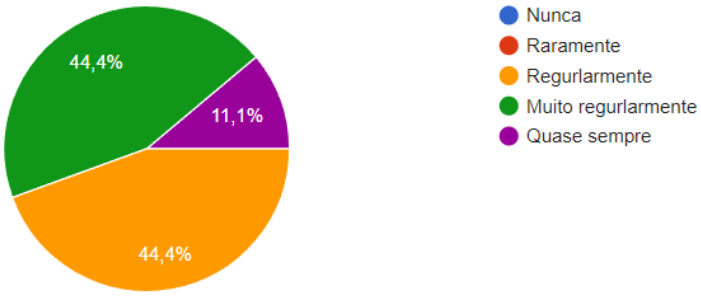

Figura 6.3: Gráfico das respostas à frequência do uso do telemóvel

Com que frequência joga videojogos no telemóvel

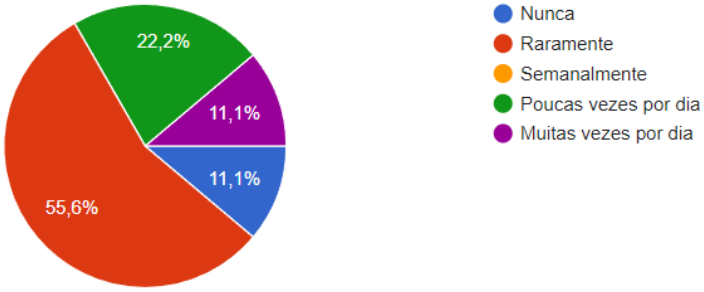

Figura 6.4: Gráfico das respostas à frequência que joga no telemóvel

6.2.2 Sistema de Usabilidade (SUS)

Para se avaliar a qualidade da aplicação foi proposto aos utilizadores responderem ao questionário de usabilidade, *System Scale Usability*. Nesta conformidade, obtiveram-se os valores que constam na seguinte tabela: Posto os

System Usability Scale										
Questão	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Total	35	16	37	12	36	13	33	13	35	17
Total p/ utilizador	3.9	1.8	4.1	1.3	4.0	1.4	3.7	1.4	3.9	1.9

Tabela 6.1: Resultados das questões do **SUS**.

resultados, de forma a calcular a pontuação final é necessário somar os valores da linha *"Total p/ utilizador"* e multiplicar por **2.5** obtendo-se o resultado final de **79.2**.

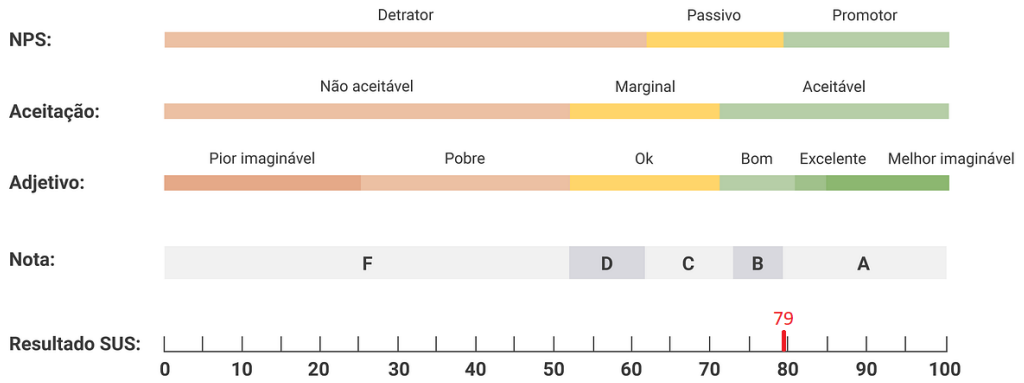

Figura 6.5: Classificação do sistema de usabilidade (SUS)

Consultando-se o gráfico de medida do Sistema de Usabilidade, apresentado em cima, conclui-se que a aplicação se encontra na gama **B**, avaliada como *Good*.

CAPÍTULO 7

CONCLUSÕES E TRABALHO FUTURO

Neste capítulo vai ser referido as conclusões retiradas da realização do projeto. Vai ser apresentado um plano do trabalho futuro a ser realizado, este plano inclui melhoramentos em termos de otimizações e novas funcionalidades.

7.1 Conclusões

Em conclusão, este projeto de desenvolvimento de um videojogo passou por várias fases, desde o planeamento inicial até à implementação concreta do modelo. Foi adotada uma abordagem organizada, dividindo o processo em fases de análise, desenho, implementação e avaliação, o que permitiu priorizar as funcionalidades essenciais e otimizar a gestão de recursos.

Alguns aspetos a salientar da implementação do modelo são:

O motor de jogo desempenhou um papel central, controlando todas as interações e lógica do jogo. A sua estrutura modular, com diferentes *scripts* dedicados a diferentes partes do jogo, ajudou a manter o código organizado e facilmente gerenciável.

O uso de arquivos JSON para armazenar dados do jogo revelou-se uma escolha acertada, tornando o jogo escalável e permitindo a adição de novo conteúdo sem a necessidade de modificar o código existente.

Em resumo, este projeto representou um esforço dedicado na criação de um videojogo envolvente, voltado para um público jovem e jovem adulto. A implementação cuidadosa das mecânicas de jogo, dos níveis, das recompensas e da interface resultou num produto final que visa proporcionar mesmo não completo, uma experiência de entretenimento cativante e desafiante para os jogadores.

7.2 Trabalho Futuro

Nesta secção como referido na introdução vai ser apresentado um plano com algumas adições e/ou otimizações a serem realizados no futuro.

7.2.1 Unity - Otimizações e conteúdo

Quanto ao jogo em si, este precisa de algumas funcionalidades que são críticas para tornar o jogo interessante, como por exemplo, o melhoramento das personagens e a sua troca, a criação de um menu principal e a criação de um tutorial.

Tutorial das mecânicas do jogo

Um tutorial pode ser criado de diferentes formas, um vídeo a explicar as diferentes mecânicas, um nível básico fácil que ajuda o jogador a perceber o jogo de forma *hands-on* ou a apresentação de *tooltips*.

Menu principal - Funcionalidades

O menu principal precisa de ter as seguintes funcionalidades:

- Começar aventura (já implementado);
- Trocar de personagem;
- Loja e *upgrades*;
- Informação sobre o jogo/créditos;
- Opções.

Novos modos de jogo

Para tornar o jogo mais complexo, é possível ser desenvolvido novos modos de jogo que possam motivar os jogadores a uma experiência para além do jogo principal.

Novos tipos de níveis

A criação de novos tipos de níveis permite tornar o jogo menos monótono, estes níveis podem trazer novos desafios ao jogador. Alguns exemplos são níveis de puzzles, níveis multi-combate (com várias fases) e níveis de vendedores (onde o jogador pode gastar o seu dinheiro adquirindo equipamentos).

Otimizações

O projeto implementado pode ser otimizado aumentando a qualidade de execução e gráfica. Algumas otimizações poderão ser:

- **Seleção de níveis** - Criação de caminhos que promove a escolha e planeamento por parte do jogador, tornando o jogo minimamente mais complexo, mas promovendo as suas decisões.
- **Menu de inventário** - Implementação de um sistema de páginas para certos menus de forma a reduzir o acoplamento visual e permitindo uma melhor experiência para o utilizador.

7.2.2 Modelos e animações

O jogo foi implementado de modo a favorecer a escalabilidade, com isto é possível adicionar novos conteúdos como mundos, inimigos e itens, sem ter de se alterar o código. Novos **modelos e animações** permitem dar ao jogador uma nova experiência enquanto joga descobrindo novas mecânicas e desafios.

7.2.3 Som

Qualquer aplicação deste tipo necessita de ter som que pode ser categorizado em dois tipos:

- Efeitos sonoros - Sons que funcionam como feedback a ações do jogador e/ou do sistema;

- Música - Que acompanha o jogador na sua aventura e ajuda a melhor a aplicação face à experiência do utilizador, tornando-a mais agradável e motivante.

7.2.4 Ligação com o Google Play

A ligação com o **Google Play** permite ao utilizador guardar as informações das suas sessões na sua conta, o que permite aos *Game Developers* receber *feedback* no caso de problemas no jogo e permite ao utilizador ter sempre o seu progresso salvo.

APÊNDICE A

LINK PARA O REPOSITÓRIO DO PROJETO

Repositório - Unity Mobile Game
https://github.com/FariaXD/UnityMobileGame-FinalProject

APÊNDICE B

SISTEMA DE USABILIDADE (SUS) - DETALHADO

Acho que gostaria de usar esta aplicação com frequência.

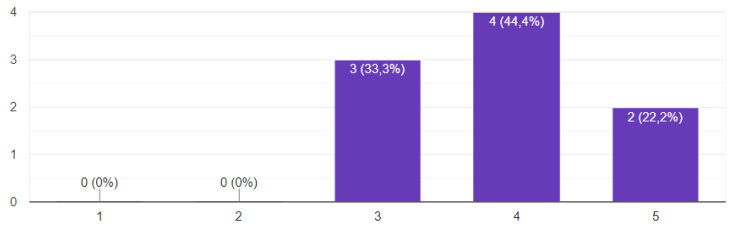

Figura B.1

Considerei a aplicação mais complexa do que o necessário

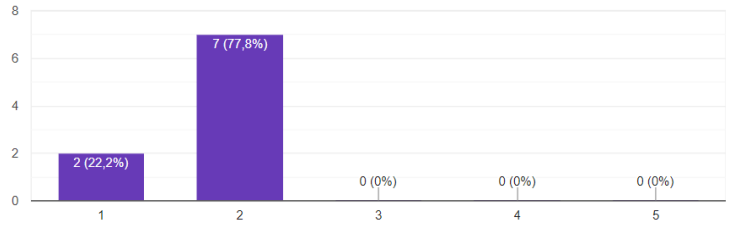

Figura B.2: Gráfico das respostas ao género

Achei a aplicação fácil de usar

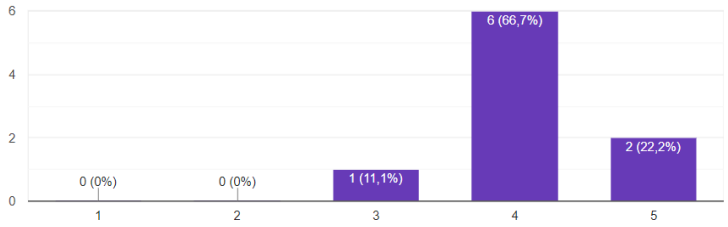

Figura B.3: Gráfico das respostas ao género

Acho que necessitaria de ajuda de um técnico para conseguir utilizar esta aplicação

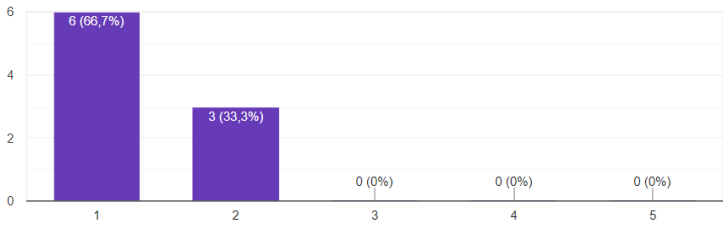

Figura B.4: Gráfico das respostas ao género

Considerei que as várias funcionalidades desta aplicação estavam bem integradas

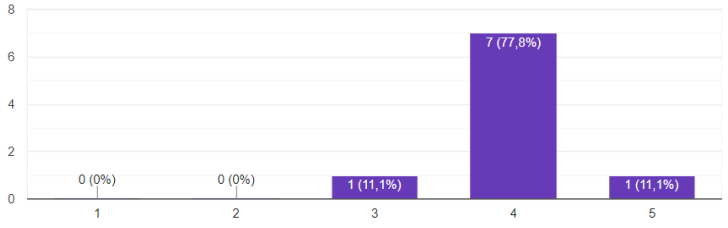

Figura B.5: Gráfico das respostas ao género

Achei que esta aplicação tinha muitas inconsistências

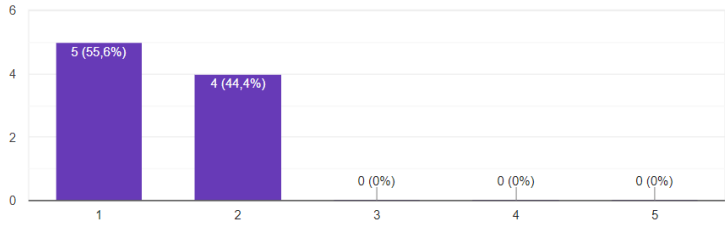

Figura B.6: Gráfico das respostas ao género

Acho que a maioria das pessoas aprenderia a utilizar rapidamente esta aplicação

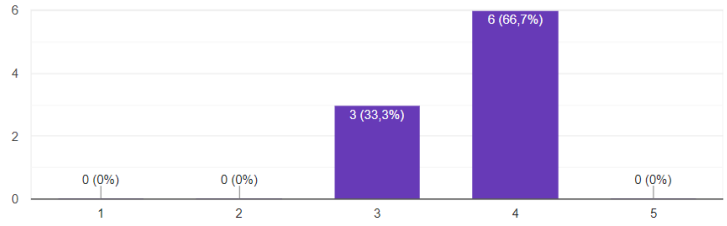

Figura B.7: Gráfico das respostas ao género

Considerei a aplicação complicada de utilizar

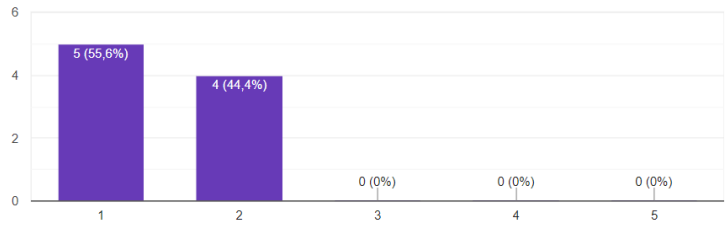

Figura B.8: Gráfico das respostas ao género

Senti-me confiante a utilizar esta aplicação

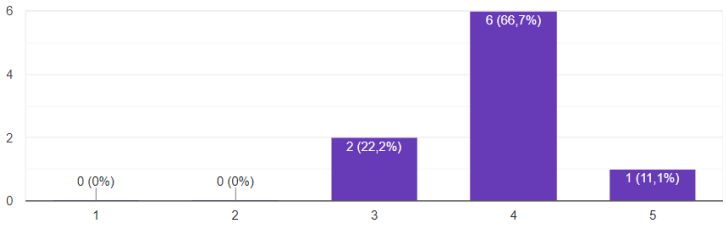

Figura B.9: Gráfico das respostas ao género

Tive que aprender muito antes de conseguir lidar com esta aplicação

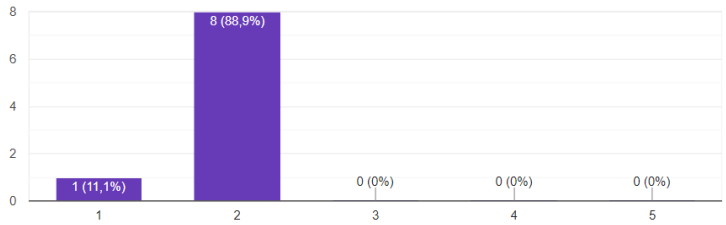

Figura B.10: Gráfico das respostas ao género

BIBLIOGRAFIA

[1] Aseprite. *Aseprite*. URL: https://www.aseprite.org/.

[2] Audacity. *Audacity*. URL: https://www.audacityteam.org/download/.

[3] Mega Crit Games. *Slay the Spire*. URL: https://store.steampowered.com/app/646570/Slay_the_Spire/.

[4] Google. *Google Play*. URL: https://play.google.com/googleplaygames.

[5] Maddy Makes Games Inc. *Celeste*. URL: https://store.steampowered.com/app/504230/Celeste/.

[6] Unity. *Unity*. URL: https://unity.com/.