**CURSO PROFISSIONAL TÉCNICO DE GESTÃO E PROGRAMAÇÃO DE SISTEMAS INFORMÁTICOS – NÍVEL 4**

**PROVA DE APTIDÃO PROFISSIONAL**

**2020-2023**



**Beat em UP**

Professor Acompanhante:

*Milton Aguiar*

Aluno: *Joaquim Ferreira*

Nº Processo: *15348*

Funchal, 21 de junho de 2023

**Beat em Up**

“I choose a lazy person to do a hard job. Because a lazy person will find an easy way to do it” – Bill Gates

**AGRADECIMENTOS**

Durante a criação deste projeto encontrei várias dificuldades e não seria possível fazer sem a ajuda dos meus amigos, nomeadamente o Diogo Alexandre e o André Perreira.

Além disso gostaria de agradecer aos canais [Cobra Code](https://www.youtube.com/@CobraCode), [Coding in Flow](https://www.youtube.com/@codinginflow), [Brackeys](https://www.youtube.com/@Brackeys) e [BMo](https://www.youtube.com/@BMoDev) por os seus vídeos informativos e explicativos no âmbito de criação de jogos.

Também quero agradecer ao [Clembod](https://clembod.itch.io/) por os sprites das personagens, o [Ariel Oliveira](https://oarielg.itch.io/) Pelo UI e ao [Ansimuz](https://ansimuz.itch.io/) pelo tile set.

Finalmente queria agradecer aos professores da área de informática da escola pois durante estes três anos que me deram os conhecimentos que tornaram este projeto possível.

**INDICE**

[INDICE DE FIGURAS ii](#_Toc138272647)

[1. Introdução/Fundamentação 4](#_Toc138272648)

[2. PLANIFICAÇÃO 5](#_Toc138272649)

[3. CRONOGRAMA 7](#_Toc138272650)

[4. METODOLOGIA DO PROJETO 8](#_Toc138272651)

[4.1. PESQUISA 8](#_Toc138272652)

[4.2. objetivos e conteúdos 18](#_Toc138272653)

[4.3. TECNOLOGIAS/linguagens UTILIZADAS 19](#_Toc138272654)

[5. EXECUÇÃO DO PROJETO 22](#_Toc138272655)

[5.1. Criação, gestão e integração dos conteúdos 22](#_Toc138272656)

[5.2. concepção DO DESIGN 45](#_Toc138272657)

[5.3. desenvolvimento e testes realizados 60](#_Toc138272658)

[5.4. IMPLEMENTAÇÃO DO SITE 63](#_Toc138272659)

[6. CONCLUSÃO 67](#_Toc138272660)

[7. REFERÊNCIAS BIBLIOGRÁFICAS 68](#_Toc138272661)

# INDICE DE FIGURAS

[Figura 1-Planificacão 5](#_Toc138279772)

[Figura 2 -Planificação 6](#_Toc138279773)

[Figura 3 -Cronograma 7](#_Toc138279774)

[Figura 4 -Logo do Unreal Engine 19](#_Toc138279775)

[Figura 5 -Logo da Unity 20](#_Toc138279776)

[Figura 6 -Logo do Visual Studio 21](#_Toc138279777)

[Figura 7 -Segmento de Código do Movimento 1 22](#_Toc138279778)

[Figura 8 -Segmento de Código do Movimento 2 23](#_Toc138279779)

[Figura 9 -Segmento de Código do Movimento 3 23](#_Toc138279780)

[Figura 10 -Segmento de Código do Movimento 4 24](#_Toc138279781)

[Figura 11 -Segmento de Código do Movimento 5 25](#_Toc138279782)

[Figura 12 -Código que Controla a Câmera 25](#_Toc138279783)

[Figura 13 -Segmento de Código de Combate 1 26](#_Toc138279784)

[Figura 14 -Segmento de Código de Combate 2 26](#_Toc138279785)

[Figura 15 -Segmento de Código de Combate 3 27](#_Toc138279786)

[Figura 16 -Segmento de Código de Combate 4 27](#_Toc138279787)

[Figura 17 --Segmento de Código de Combate 5 28](#_Toc138279788)

[Figura 18 -Segmento de Código de Combate 6 28](#_Toc138279789)

[Figura 19 -Segmento de Código de HP e Morte 1 29](#_Toc138279790)

[Figura 20 -Segmento de Código de HP e Morte 2 29](#_Toc138279791)

[Figura 21 -Segmento de Código de HP e Morte 3 30](#_Toc138279792)

[Figura 22 -Segmento de Código de HP e Morte 4 30](#_Toc138279793)

[Figura 23 -Segmento de Código da Barra de Vida 1 31](#_Toc138279794)

[Figura 24 -Segmento de Código da Barra de Vida 2 31](#_Toc138279795)

[Figura 25 -Segmento de Código da Barra de Vida 3 32](#_Toc138279796)

[Figura 26 -Segmento de Código do Ataque do Boss 1 32](#_Toc138279797)

[Figura 27 -Segmento de Código do Ataque do Boss 2 33](#_Toc138279798)

[Figura 28 -Segmento de Código do Ataque do Boss 3 33](#_Toc138279799)

[Figura 29 -Segmento de Código do HP do Boss 1 34](#_Toc138279800)

[Figura 30 -Segmento de Código do HP do Boss 2 34](#_Toc138279801)

[Figura 31 -Segmento de Código do HP do Boss 3 35](#_Toc138279802)

[Figura 32 -Segmento de Código do HP do Boss 4 35](#_Toc138279803)

[Figura 33 -Segmento de Código do HP do Boss 5 36](#_Toc138279804)

[Figura 34 -Segmento de Código da Lógica do Boss 1 36](#_Toc138279805)

[Figura 35 -Segmento de Código da Lógica do Boss 2 37](#_Toc138279806)

[Figura 36 -Segmento de Código da Lógica do Boss 3 37](#_Toc138279807)

[Figura 37 -Segmento de Código da Lógica do Boss 4 37](#_Toc138279808)

[Figura 38 - Código da Orientação do Boss 38](#_Toc138279809)

[Figura 39 - Código do Menu Principal 39](#_Toc138279810)

[Figura 40 -Segmento de Código do Menu de Pausa 1 40](#_Toc138279811)

[Figura 41 -Segmento de Código do Menu de Pausa 2 40](#_Toc138279812)

[Figura 42 -Segmento de Código do Menu de Pausa 3 41](#_Toc138279813)

[Figura 43 -Segmento de Código do Menu de Pausa 4 41](#_Toc138279814)

[Figura 44 -Segmento de Código do Menu de Pausa 5 41](#_Toc138279815)

[Figura 45 -Segmento de Código do Menu de Pausa 6 42](#_Toc138279816)

[Figura 46 -Segmento de Código do Menu de Pausa 7 42](#_Toc138279817)

[Figura 47 -Segmento de Código do Ecrã de Morte 1 43](#_Toc138279818)

[Figura 48 -Segmento de Código do Ecrã de Morte 2 43](#_Toc138279819)

[Figura 49 -Segmento de Código do Ecrã de Morte 3 43](#_Toc138279820)

[Figura 50 -Segmento de Código do Ecrã de Vitória 1 44](#_Toc138279821)

[Figura 51 -Segmento de Código do Ecrã de Vitória 2 44](#_Toc138279822)

[Figura 52 -Segmento de Código do Ecrã de Vitória 3 44](#_Toc138279823)

[Figura 53 -Animator 45](#_Toc138279824)

[Figura 54 -Menu de uma Animação 46](#_Toc138279825)

[Figura 55 -Interligação das Animações 46](#_Toc138279826)

[Figura 56 -Menu da Interligação das Animações 46](#_Toc138279827)

[Figura 57 -Scene Manager 47](#_Toc138279828)

[Figura 58 - Gerenciador do Square 47](#_Toc138279829)

[Figura 59 -Animações da Personagem 48](#_Toc138279830)

[Figura 60 -Animator da Personagem 49](#_Toc138279831)

[Figura 61 -Gerenciador da Personagem 50](#_Toc138279832)

[Figura 62 -Animações do Boss 51](#_Toc138279833)

[Figura 63 -Animator do Boss 52](#_Toc138279834)

[Figura 64 -Gerenciador do Boss 53](#_Toc138279835)

[Figura 65 -Grid 54](#_Toc138279836)

[Figura 66 -Tilemap 54](#_Toc138279837)

[Figura 67 -Gerenciador do Mapa 55](#_Toc138279838)

[Figura 68 -O Mapa 55](#_Toc138279839)

[Figura 69 -Gerenciador do Botão 56](#_Toc138279840)

[Figura 70 -Gerenciador do Botão (continuação) 57](#_Toc138279841)

[Figura 71 -Menu Principal 58](#_Toc138279842)

[Figura 72 -Menu de Pausa 58](#_Toc138279843)

[Figura 73 -Ecrã de Morte 59](#_Toc138279844)

[Figura 74 -Ecrã de Vitoria 59](#_Toc138279845)

[Figura 75 -Segmento de Código do Site 1 63](#_Toc138279846)

[Figura 76 -Segmento de Código do Site 2 63](#_Toc138279847)

[Figura 77 -Segmento de Código do Site 3 64](#_Toc138279848)

[Figura 78 -Segmento de Código do Site 4 64](#_Toc138279849)

[Figura 79 -Segmento de Código do Site 5 65](#_Toc138279850)

[Figura 80 -Site do Jogo 66](#_Toc138279851)

# Introdução/Fundamentação

No âmbito da Prova de Aptidão Profissional (PAP) do Curso Técnico de Gestão e Programação de Sistemas Informáticos, propõe-se o desenvolvimento de um videojogo de combate 2D acompanhado por um site de instalação. O objetivo do projeto é aplicar os conhecimentos adquiridos ao longo do curso e explorar o interesse pessoal pela programação de jogos, buscando respostas às perguntas que surgem ao jogar videogames e entender como certas funcionalidades são implementadas.

A escolha do tema do projeto baseia-se na vasta experiência em jogar videogames e no constante questionamento sobre as técnicas utilizadas na criação dos mesmos. O projeto surge como uma oportunidade para explorar e adquirir conhecimentos em programação de jogos, além de ser uma chance de compreender e responder a algumas dessas perguntas. A opção por desenvolver um jogo de combate 2D é motivada pela busca por um desafio intermediário, onde se pode explorar a mecânica de combate e ainda manter a estética visual consistente e atrativa.

**Os principais objetivos do projeto são os seguintes:**

* Desenvolver um jogo de combate 2D envolvente, com mecânicas de jogo básicas e funcionais.
* Explorar a programação de jogos, aplicando conceitos e técnicas aprendidas durante o curso.
* Compreender e implementar funcionalidades de um jogo de combate, como movimentação, ataques, habilidades especiais e colisões.
* Criar um site de instalação para disponibilizar o jogo, fornecendo uma plataforma acessível aos jogadores interessados.

**Descrição do Jogo:**

O jogo de combate 2D proposto será ambientado em um mundo fictício, com uma personagem e um chefe. O jogador terá de enfrentar e derrotar este chefe. O jogo contará com mecânicas de combate simples e fáceis de aprender.

**Descrição do Site de Instalação:**

O site de instalação do jogo será desenvolvido para oferecer aos jogadores uma maneira fácil e conveniente de adquirir e instalar o jogo. Ele incluirá uma breve descrição do jogo. O site será projetado de forma a ser simples e acessível a todos os interessados em experimentar o jogo de combate 2D. Além disso também estará disponibilizado um vídeo que serve como tutorial para o jogo.

# PLANIFICAÇÃO

A screenshot of a computer screen

Description automatically generated with low confidence

Figura -Planificacão

A screenshot of a computer

Description automatically generated with medium confidence

Figura -Planificação

# CRONOGRAMA

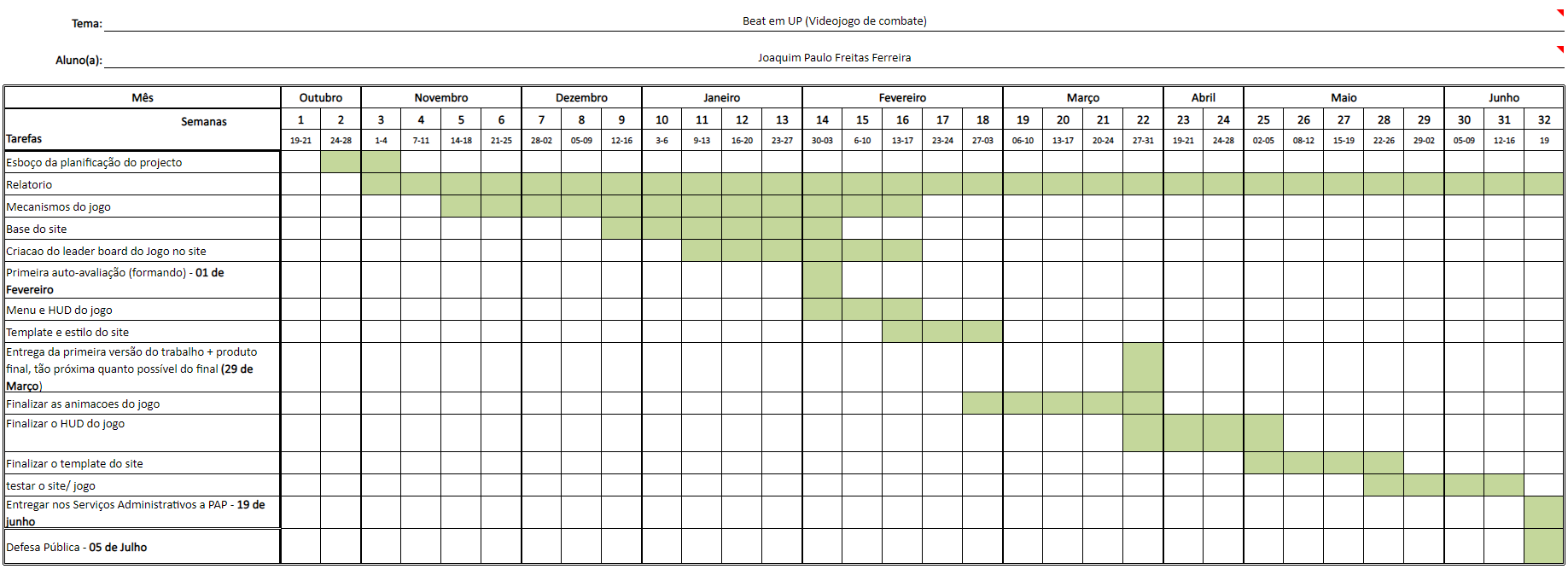


Figura -Cronograma

# METODOLOGIA DO PROJETO

## PESQUISA

**Unreal:**

A minha pesquisa do Unreal Engine 5 (UE5) foi feita primariamente pela Youtube visto como ainda é um software relativamente novo comparado com as outras opções.

O primeiro vídeo que vi era baseado em demonstrar como fazer um jogo 2D simples que facilitou a minha iniciação neste software pois o vídeo estava bem estruturado e ajudou-me a ter os conhecimentos básicos da UE5 demonstrando como criar um projeto novo, criar o mapa, criar um sprite (personagem) fazendo lo mexer e ter animações. O vídeo em questão é este:

[How to make a 2D game in Unreal Engine 5 - Beginner Tutorial](https://www.youtube.com/watch?v=g31NTpq9p-o&list=PLWVtPufZLwg3EQulR_cW5ya7y1PXgTaMI&index=1&ab_channel=CobraCode)

[](https://www.youtube.com/watch?v=g31NTpq9p-o&list=PLWVtPufZLwg3EQulR_cW5ya7y1PXgTaMI&index=1&ab_channel=CobraCode)

Alem de tudo o vídeo apresentou-me a um site que tinha assets (recursos digitais) grátis que facilitou o meu trabalho pois se tivesse de fazer os modelos e as animações do jogo não seria possível acabar a tempo. O site em questão: <https://itch.io/game-assets/free/tag-pixel-art>

Apos de ter uma base para o jogo encontrei um vídeo pela mesma pessoa que fez o anterior, este vídeo explicava como fazem sistema de combate simples que era precisamente o que precisava só que estava a ter problemas de compatibilidade com o código e tive de emendar, mas não consegui fazer tudo. O vídeo em questão:

[2D MELEE COMBAT in Unreal Engine 5 - Beginner Tutorial](https://www.youtube.com/watch?v=JCdRe4kC3FM&list=PLWVtPufZLwg3EQulR_cW5ya7y1PXgTaMI&index=2&ab_channel=CobraCode)

[](https://www.youtube.com/watch?v=JCdRe4kC3FM&list=PLWVtPufZLwg3EQulR_cW5ya7y1PXgTaMI&index=2&ab_channel=CobraCode)

Apos de tentar implementar o código o programa começou a ter mais e mais erros por isso decidi mudar para um software que apoia melhor jogos 2D. O software que encaixava melhor com o que procurava era a Unity pois era relativamente popular fazendo com que pudesse aprender mais facilmente e como já tem muito mais anos do que o unity poderia encontrar soluções para problemas mais facilmente.

**Unity:**

O primeiro vídeo que vi sobre a unity ajudou-me a aprender a base e serviu para me orientar neste programa que estava a utilizar pela primeira vez. O vídeo é um de um serie de vídeos que facilitou a minha aprendizagem das mecânicas básicas da unity. Este vídeo detalha como criar um “tilemap”, ou seja, um mapa onde as personagens possam lutar, além disto também demostra como fazer o sprite/modelo da personagem e um salto básico. O vídeo em questão:

[Tilemap & Tile Palette | Build a 2D Platformer Game in Unity #2](https://www.youtube.com/watch?v=QkbGr1rAya8)

[](https://www.youtube.com/watch?v=QkbGr1rAya8)

Após esse vídeo utilizei os próximos quatro vídeos da serie para fazer o movimento da personagem e as suas várias animações.

O vídeo em baixo ofereceu uma explicação sobre como programar o movimento da personagem, além disso também demostra como colar a câmera a personagem e fazer com que a personagem fica fixado na orientação correta, ou seja, não permite que a personagem inclina.

[Player Movement | Build a 2D Platformer Game in Unity #3](https://youtu.be/Uv5tfMSKlnU)

[](https://youtu.be/Uv5tfMSKlnU)

O seguinte vídeo contém informações sobre como gerir animações na unity. Neles é explicado como criar uma animação através de sprites, como fazer a lógica das animações e também como controlar a velocidade e a altura dos saltos da personagem.

[Animation & Animator | Build a 2D Platformer Game in Unity #4](https://youtu.be/GChUpPnOSkg)

[](https://youtu.be/GChUpPnOSkg)

[Multiple Animations | Build a 2D Platformer Game in Unity #5](https://youtu.be/65E-q0JxYwU)

[](https://youtu.be/65E-q0JxYwU)

Neste vídeo é explicado como fazer com que a personagem esteja limitada a apenas um salto. Isto é feito através de uma caixa de colisão que deteta se está a tocar no chão do mapa.

[Grounding Check Using Boxcast | Build a 2D Platformer Game in Unity #6](https://youtu.be/LEUhxe9vUOM)

[](https://youtu.be/LEUhxe9vUOM)

Este vídeo serviu como base para o combate do jogo. É também utilizando este vídeo que programei a vida da personagem, fazendo com que ela perca vida se for atacada e vise versa. Além disso também fiz um attackPoint que é um ponto que regista o inimigo e aplica o dano definido no código.

[MELEE COMBAT in Unity](https://www.youtube.com/watch?v=sPiVz1k-fEs&t=1094s)

[](https://www.youtube.com/watch?v=sPiVz1k-fEs&t=1094s)

Neste vídeo aprendi como fazer uma barra de vida para a minha personagem, alem disso serviu como uma introdução para a criação de UI (User Interface, ou seja, menus e coisa desse género).

[How to make a HEALTH BAR in Unity!](https://www.youtube.com/watch?v=BLfNP4Sc_iA&t=1000s)

[](https://www.youtube.com/watch?v=BLfNP4Sc_iA&t=1000s)

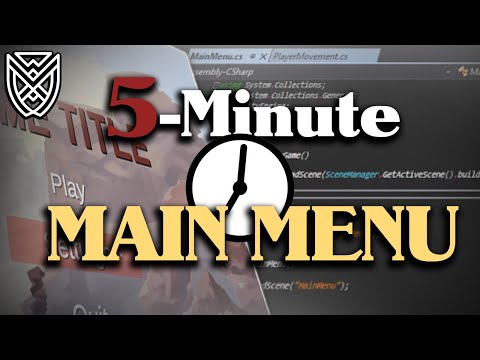
No vídeo representado abaixo aprendi como criar um boss para a minha personagem combater. É também neste vídeo em que aprendi que poderia utilizar o gestor das animações para aplicar lógica ao inimigo fazendo com que ele faz determinadas ações em determinadas circunstâncias.

[How to make a BOSS in Unity!](https://www.youtube.com/watch?v=AD4JIXQDw0s)

[](https://www.youtube.com/watch?v=AD4JIXQDw0s)

Nos seguintes vídeos aprendi como fazer um UI intragável que serviu para fazer os menus do jogo e no segundo vídeo aprendi como pausar o jogo quando o menu aparecia.

[5 Minute MAIN MENU Unity Tutorial](https://www.youtube.com/watch?v=-GWjA6dixV4)

[](https://www.youtube.com/watch?v=-GWjA6dixV4)

[6 Minute PAUSE MENU Unity Tutorial](https://www.youtube.com/watch?v=9dYDBomQpBQ)

## objetivos e conteúdos

O objetivo final do projeto é desenvolver um jogo de combate 2D envolvente, com mecânicas de jogo básicas e funcionais em que o jogador terá de matar um chefe. Para este meio será necessário:

* Criar a personagem principal, para tal terei de utilizar vários componentes para o criar e tornar o num objeto físico com lógica.
* Criar o chefe/boss, para tal terei de novamente utilizar vários componentes para o criar e tornar o num objeto físico com lógica.
* Criar o mapa, para tal terei de utilizar um “grid” para poder colocar as peças do mapa, após isto terei de utilizar componentes para o renderizar e o tornar num objeto físico.
* Criar um menu principal, para tal necessito de criar um “canvas” para colocar o design do menu, neste design é necessário ter um botão para jogar e um para sair do jogo, estes botões serão manipulados por um script.
* Criar um menu de pausa, para tal necessito de outro “canvas” para colocar o design deste menu, neste design é necessário ter um botão para resumir o jogo, outro para voltar ao menu principal e finalmente um para sair do jogo, estes botões serão manipulados por um script.
* Criar um ecrã de vitoria, para tal necessito de mais um “canvas” para colocar o design deste ecrã, se clicar na barra de espaço no teclado voltara ao menu principal
* Criar um ecrã de morte, para tal necessito de o último “canvas” para colocar o design deste ecrã, se clicar na barra de espaço no teclado voltara ao menu principal
* Criar um site para a instalação do jogo, o site terá o logotipo do jogo juntamente com o nome, um carrossel com imagens do jogo, uma secção com um breve resumo do jogo, um vídeo de tutorial e um botão para efetuar o download seguido pelo um rodapé.

## TECNOLOGIAS/linguagens UTILIZADAS



Figura -Logo do Unreal Engine

A Unreal Engine 5 (UE5) é a mais recente versão do mecanismo de jogo da EPIC Games, que fornece uma estrutura para desenvolvimento de jogos e outras experiências interativas em tempo real.

Ele é usado por desenvolvedores de jogos em todo o mundo para criar jogos de alta qualidade e com gráficos impressionantes em várias plataformas, como PC, consoles, dispositivos móveis e VR.

A UE5 apresenta novas tecnologias, como Nanite e Lumen, que permitem renderizar objetos detalhados em tempo real e iluminação global em tempo real para criar uma experiência visual mais realista e imersiva.

Além disso, a UE5 também vem com melhorias no fluxo de trabalho do desenvolvedor e uma série de ferramentas para criar personagens digitais realistas e detalhados.



Figura -Logo da Unity

Unity é um mecanismo de jogo multiplataforma usado para desenvolver jogos, aplicativos e experiências interativas em tempo real para uma variedade de plataformas, como PC, consoles, dispositivos móveis e VR.

É uma ferramenta popular entre desenvolvedores de jogos de todos os níveis de habilidade, desde iniciantes até desenvolvedores experientes, pois permite que eles criem jogos rapidamente e com eficiência.

O Unity oferece uma ampla variedade de recursos, como renderização em tempo real, física, animação, deteção de colisão e suporte a linguagens de programação como C# e JavaScript.

Ele também tem uma ampla comunidade de desenvolvedores ativos que compartilham dicas, truques e recursos úteis, bem como uma ampla variedade de plug-ins e extensões para ajudar os desenvolvedores a estender e personalizar o mecanismo.

O Unity é amplamente usado na indústria de jogos, mas também é usado em outras áreas, como arquitetura, simulação, treinamento, visualização e design de produtos.



Figura -Logo do Visual Studio

O Visual Studio é um ambiente de desenvolvimento integrado (IDE) criado pela Microsoft, utilizado principalmente para desenvolvimento de software.

Ele oferece uma variedade de recursos e ferramentas para programadores, permitindo a criação, edição, depuração e compilação de código em várias linguagens de programação, como C#, C++, Visual Basic, entre outras.

O Visual Studio possui uma interface intuitiva e personalizável, que facilita a escrita e organização do código-fonte. Ele oferece recursos avançados de depuração, como pontos de interrupção e inspeção de variáveis em tempo real, para facilitar a identificação e correção de erros no código.

Além disso, o Visual Studio possui integração com outras ferramentas e serviços, como controle de versão, testes automatizados e suporte a projetos em equipe. Ele também suporta a criação de aplicativos para diferentes plataformas, como Windows, Android e iOS, e oferece suporte a tecnologias e frameworks populares, como o .NET.

No geral, o Visual Studio é uma ferramenta poderosa e versátil para desenvolvedores, oferecendo um ambiente completo para o desenvolvimento de software com eficiência e produtividade.

# EXECUÇÃO DO PROJETO

## Criação, gestão e integração dos conteúdos

**Movimento:**

A picture containing text, screenshot, software, multimedia software

Description automatically generated

Figura -Segmento de Código do Movimento 1

Primeiramente, são importadas as bibliotecas necessárias: System.Collections, System.Collections.Generic e UnityEngine.

A classe PlayerMovement é declarada e herda da classe MonoBehaviour, que é uma classe base da Unity para comportamentos de objetos do jogo.

Em seguida, são declaradas as variáveis privadas que serão utilizadas no script, como Rigidbody2D, BoxCollider2D, SpriteRenderer, Animator e um script chamado Die.

A variável jumpableGround é uma LayerMask que define quais camadas de terreno podem ser consideradas "chão" para o personagem pular.

As variáveis dirx, movespeed e jumpforce são declarados e atribuídos valores padrão. dirx é a direção do movimento do jogador, movespeed é a velocidade de movimento e jumpforce é a força do salto.

É declarado um enum chamado MovementState, que representa os estados possíveis de movimento do jogador (end, idle, running, jumping, falling).

A screen shot of a computer program

Description automatically generated with low confidence

Figura -Segmento de Código do Movimento 2

No método Start (), são obtidas as referências aos componentes do objeto do jogador, como o Rigidbody2D, BoxCollider2D, SpriteRenderer, Animator e o script Die.

A screen shot of a computer code

Description automatically generated with low confidence

Figura -Segmento de Código do Movimento 3

No método Update (), que é chamado a cada quadro do jogo, são atualizadas as entradas do jogador, a velocidade do Rigidbody2D e o estado da animação.

A screen shot of a computer program

Description automatically generated with low confidence

Figura -Segmento de Código do Movimento 4

O método UpdateAnimationState () é responsável por atualizar o estado da animação do jogador com base nas entradas e na velocidade do Rigidbody2D.

Dependendo da direção do movimento, o estado é definido como "running" e o sprite pode ser espelhado horizontalmente.

Caso não haja movimento, o estado é definido como "idle".

Se o jogador estiver morto, é definido o estado "IsDead" na animação.

Se o jogador estiver a saltar ou caindo, o estado é definido como "jumping" ou "falling", respetivamente.

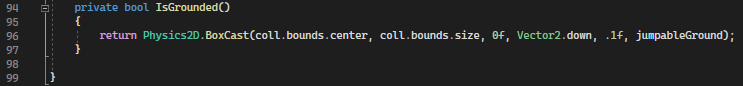


Figura -Segmento de Código do Movimento 5

O método IsGrounded () verifica se o jogador está no chão.

Isso é feito através de um BoxCast que lança um raio para baixo a partir do centro do colisor do jogador e verifica se colide com algum objeto na camada definida pela LayerMask jumpableGround.

**Controle da Câmera:**

A screen shot of a computer program

Description automatically generated with low confidence

Figura -Código que Controla a Câmera

Primeiramente, são importadas as bibliotecas necessárias: System.Collections, System.Collections.Generic e UnityEngine.

A classe CameraController é declarada e herda da classe MonoBehaviour, que é uma classe base da Unity para comportamentos de objetos do jogo.

É declarada uma variável privada chamada player, do tipo Transform. Essa variável é serializada, o que significa que ela pode ser acedida e configurada no Editor da Unity.

No método Update (), que é chamado a cada quadro do jogo, a posição da câmara é atualizada. A posição da câmara é definida como um novo Vector3, utilizando as coordenadas x e y da posição do jogador e a coordenada z da posição atual da câmera. Dessa forma, a câmara acompanha o movimento do jogador apenas no eixo x e y, mantendo a sua posição z inalterada.

**Combate:**

A screen shot of a computer

Description automatically generated with medium confidence

Figura -Segmento de Código de Combate 1

O código define uma classe chamada "PlayerCombat" que é derivada da classe MonoBehaviour, permitindo que seja anexada a um objeto no Unity.

Existem várias variáveis públicas declaradas, como "animator" (responsável por controlar as animações do jogador), "attackPoint" (uma posição onde o ataque será realizado), "enemyLayers" (uma camada que contém os inimigos), "attackRange" (alcance do ataque), "attackDamage" (dano do ataque) e "attackRate" (taxa de ataque).

A black background with white text

Description automatically generated with low confidence

Figura -Segmento de Código de Combate 2

A função "Awake" é chamada quando o objeto é inicializado e atribui a referência do componente SpriteRenderer a uma variável chamada "spriteRenderer"

A screen shot of a computer program

Description automatically generated with low confidence

Figura -Segmento de Código de Combate 3

A função "Update" é chamada a cada frame e verifica se é hora de realizar um ataque. Se a tecla "E" for pressionada e o tempo atual for maior ou igual ao próximo tempo de ataque, a função "Attack" é chamada e o tempo de próximo ataque é atualizado.

A picture containing text, screenshot, software, multimedia software

Description automatically generated

Figura -Segmento de Código de Combate 4

A função "Attack" reproduz a animação de ataque definida no componente "animator" e realiza uma deteção de colisão na área de ataque usando a função "Physics2D.OverlapCircleAll". Em seguida, causa dano aos inimigos atingidos, verificando se eles possuem um componente "BossHealth" e chamando a função "TakeDamage".

A screen shot of a computer code

Description automatically generated with low confidence

Figura --Segmento de Código de Combate 5

A função "OnDrawGizmosSelected" desenha uma esfera no editor do Unity para representar a área de ataque quando o objeto estiver selecionado.

A picture containing text, screenshot

Description automatically generated

Figura -Segmento de Código de Combate 6

A função "LateUpdate" é chamada após o "Update" e ajusta a posição do ponto de ataque com base na direção do personagem, invertendo-o se o SpriteRenderer estiver com a propriedade "flipX" ativada.

**HP e Morte:**

A screen shot of a computer program

Description automatically generated with medium confidence

Figura -Segmento de Código de HP e Morte 1

O código começa com a declaração de algumas variáveis públicas, como "animator" do tipo Animator, "maxHealth" e "currentHealth" do tipo inteiro, "healthBar" do tipo HealthBar (provavelmente uma classe customizada para exibir a barra de vida) e "deathMenu" do tipo GameObject.

A screen shot of a computer program

Description automatically generated with low confidence

Figura -Segmento de Código de HP e Morte 2

No método Start (), que é chamado antes da primeira renderização do jogo, a variável "currentHealth" é inicializada com o valor de "maxHealth" e a função SetMaxHealth () é chamada na variável "healthBar" para definir o valor máximo da barra de vida.

A picture containing text, screenshot, font, software

Description automatically generated

Figura -Segmento de Código de HP e Morte 3

O método TakeDamage (int damage) é responsável por subtrair a quantidade de "damage" da variável "currentHealth". Em seguida, a função SetHealth () é chamada na variável "healthBar" para atualizar a exibição da barra de vida. O método também aciona o gatilho "Hurt" no "animator" para reproduzir a animação de dano. Se a "currentHealth" for menor ou igual a zero, o método RIP () é chamado

A screen shot of a computer code

Description automatically generated with medium confidence

Figura -Segmento de Código de HP e Morte 4

No método RIP (), a animação de morte é reproduzida ao definir o parâmetro "IsDead" como verdadeiro no "animator". O movimento do objeto é desabilitado ao congelar as posições do Rigidbody2D. O colisor do objeto é desabilitado ao definir "enabled" como falso no Collider2D. O "deathMenu" é ativado, tornando-o visível para o jogador. Por fim, o script é desabilitado, tornando-o inativo, isto é feito para evitar com que o jogar move-se após a morte.

**Barra de Vida:**

A screen shot of a computer program

Description automatically generated with medium confidence

Figura -Segmento de Código da Barra de Vida 1

O código declara algumas variáveis públicas, como "slider" do tipo Slider (que é um componente UI usado para exibir a barra de vida), "gradient" do tipo Gradient (que é usado para definir a cor da barra de acordo com o valor da vida) e "fill" do tipo Image (que é usado para preencher a barra de vida).

A screen shot of a computer code

Description automatically generated with low confidence

Figura -Segmento de Código da Barra de Vida 2

O método SetMaxHealth (int health) é responsável por configurar o valor máximo da barra de vida. Ele define o valor máximo do "slider" como "health" e também define o valor atual como "health". Em seguida, ele define a cor do "fill" usando o método Evaluate () do "gradient", passando o valor 1f (o valor máximo normalizado da barra).

A picture containing text, screenshot, font

Description automatically generated

Figura -Segmento de Código da Barra de Vida 3

O método SetHealth (int health) é responsável por atualizar o valor atual da barra de vida. Ele define o valor do "slider" como "health" e, em seguida, define a cor do "fill" usando o método Evaluate () do "gradient", passando o valor normalizado do "slider".

**Ataque do Boss/Chefe:**

A screen shot of a computer program

Description automatically generated with low confidence

Figura -Segmento de Código do Ataque do Boss 1

O código declara algumas variáveis públicas, como "attackDamage" do tipo inteiro (que representa o dano do ataque), "attackOffset" do tipo Vector3 (que provavelmente é usado para definir a posição de ataque da arma), "attackRange" do tipo float (que define o alcance do ataque), "enemyLayers" do tipo LayerMask (que representa as camadas dos inimigos que podem ser atingidos pelo ataque) e "attackPoint" do tipo Transform (que é o ponto de origem do ataque).

A picture containing text, software, multimedia software, graphics software

Description automatically generated

Figura -Segmento de Código do Ataque do Boss 2

O método Attack () é responsável por executar o ataque da arma. Ele utiliza a função OverlapCircleAll () da classe Physics2D para detectar os inimigos na área de ataque. Os inimigos detectados são armazenados em um array de Colliders chamado "hitEnemies". Em seguida, o método percorre todos os inimigos detectados no loop "foreach" e chama a função TakeDamage () da classe "Die" (provavelmente uma classe que controla a vida e morte dos inimigos) para causar dano igual a "attackDamage".

A screen shot of a computer code

Description automatically generated with low confidence

Figura -Segmento de Código do Ataque do Boss 3

O método OnDrawGizmosSelected () é usado para desenhar uma representação visual da área de ataque da arma. Ele calcula a posição com base no objeto transformado, adicionando um deslocamento de ataque especificado por "attackOffset". Em seguida, ele desenha uma esfera de arame usando a função DrawWireSphere () da classe Gizmos, com a posição e o alcance do ataque.

**HP do Boss/Chefe:**

A screen shot of a computer program

Description automatically generated with medium confidence

Figura -Segmento de Código do HP do Boss 1

O código declara algumas variáveis públicas, como "maxHealth" e "currentHealth" do tipo inteiro (que representam a saúde máxima e atual do chefe), "deathEffect" do tipo GameObject (que é um efeito visual reproduzido quando o chefe morre), "isInvulnerable" do tipo bool (que determina se o chefe é invulnerável a ataques), "healthBar" do tipo HealthBar (provavelmente uma classe customizada para exibir a barra de vida), "animator" do tipo Animator (que controla as animações do chefe) e "victoryMenu" do tipo GameObject (que é ativado quando o chefe morre).

A picture containing text, font, screenshot

Description automatically generated

Figura -Segmento de Código do HP do Boss 2

No método Start (), que é chamado antes da primeira renderização do jogo, a variável "currentHealth" é inicializada com o valor de "maxHealth" e a função SetMaxHealth () é chamada na variável "healthBar" para definir o valor máximo da barra de vida. O componente Animator é obtido do objeto através da função GetComponent <Animator> ().

A picture containing text, screenshot, software, font

Description automatically generated

Figura -Segmento de Código do HP do Boss 3

O método TakeDamage (int damage) é responsável por subtrair a quantidade de "damage" da variável "currentHealth". Ele também chama a função SetHealth () na variável "healthBar" para atualizar a exibição da barra de vida. Se a "currentHealth" for menor ou igual a zero, o método Die () é chamado para iniciar o processo de morte do chefe. Além disso, o "victoryMenu" é ativado, tornando-o visível para o jogador.

A picture containing text, screenshot, software, multimedia software

Description automatically generated

Figura -Segmento de Código do HP do Boss 4

O método Die () é responsável por executar as ações necessárias quando o chefe morre. Primeiro, ele verifica se o chefe já está morto para evitar repetições. Em seguida, define a variável "isDead" como verdadeira para indicar que o chefe está morto. Se existir um "deathEffect" definido, ele é instanciado na posição do chefe. Se existir um "animator" definido, é acionado o gatilho "Die" no "animator" para reproduzir a animação de morte. Caso contrário, o objeto do chefe é destruído diretamente.

A screen shot of a computer

Description automatically generated with medium confidence

Figura -Segmento de Código do HP do Boss 5

O método DestroyGameObject () é chamado a partir de um evento de animação de morte e é responsável por destruir o objeto do chefe.

**Lógica do Boss/Chefe:**

A picture containing text, screenshot, software, multimedia software

Description automatically generated

Figura -Segmento de Código da Lógica do Boss 1

O código declara algumas variáveis públicas, como "speed" do tipo float (que representa a velocidade de corrida do chefe) e "attackRange" do tipo float (que define a distância em que o chefe atacará o jogador).

As variáveis "player", "rb" e "boss" são declaradas para armazenar referências ao objeto do jogador, ao Rigidbody2D do chefe e à classe "Boss" (provavelmente uma classe que controla o comportamento geral do chefe).

A screen shot of a computer program

Description automatically generated with low confidence

Figura -Segmento de Código da Lógica do Boss 2

No método OnStateEnter (), que é chamado quando a transição para este estado começa, as referências do jogador, do Rigidbody2D e da classe "Boss" são obtidas.

A screen shot of a computer program

Description automatically generated with low confidence

Figura -Segmento de Código da Lógica do Boss 3

No método OnStateUpdate (), que é chamado a cada quadro de atualização entre os callbacks OnStateEnter e OnStateExit, o método boss.LookAtPlayer () é chamado para fazer o chefe olhar para o jogador. Em seguida, é calculado um novo ponto de destino "target" com base na posição do jogador, mantendo a altura do Rigidbody2D do chefe. A posição do chefe é então movida em direção a esse destino usando o método MoveTowards () e a velocidade definida. Em seguida, o método MovePosition () é usado para atualizar a posição do Rigidbody2D com a nova posição calculada.

Se a distância entre a posição do jogador e a posição do chefe for menor ou igual à "attackRange", o gatilho "Attack" é acionado no animator para iniciar o ataque.

A screen shot of a computer

Description automatically generated with low confidence

Figura -Segmento de Código da Lógica do Boss 4

No método OnStateExit (), que é chamado quando a transição para este estado termina, o gatilho "Attack" é redefinido para evitar repetições indesejadas.

**Orientação do Boss/Chefe:**

A computer screen shot of a program

Description automatically generated with low confidence

Figura - Código da Orientação do Boss

O código declara algumas variáveis públicas, como "player" do tipo Transform (que representa a posição do jogador) e "isFlipped" do tipo bool (que indica se o chefe está virado para a esquerda ou direita).

O método LookAtPlayer () é responsável por fazer o chefe olhar na direção do jogador. Ele verifica a posição do chefe em relação à posição do jogador e decide se o chefe precisa ser virado. Se a posição do chefe for maior do que a posição do jogador e o chefe estiver virado, ele inverte a escala do objeto no eixo z, rotaciona-o em 180 graus e atualiza o valor de "isFlipped" para false. Por outro lado, se a posição do chefe for menor do que a posição do jogador e o chefe não estiver virado, ele inverte a escala do objeto no eixo z, rotaciona-o em 180 graus e atualiza o valor de "isFlipped" para true.

**Menu Principal:**

A screen shot of a computer program

Description automatically generated with medium confidence

Figura - Código do Menu Principal

O código declara duas variáveis públicas do tipo GameObject: "mainMenu" e "game". A variável "mainMenu" representa o objeto do menu principal do jogo, enquanto a variável "game" representa o objeto do jogo em si.

No método Start (), que é chamado antes da primeira renderização do jogo, o objeto "game" é desativado, tornando-o invisível.

O método PlayGame () é chamado quando o jogador seleciona a opção de jogar no menu inicial. Ele desativa o objeto "mainMenu", tornando-o invisível, e ativa o objeto "game", tornando-o visível para iniciar o jogo.

O método QuitGame () é chamado quando o jogador seleciona a opção de sair no menu inicial. Ele chama a função Application.Quit() para encerrar o jogo.

**Menu de Pausa:**

A screen shot of a computer program

Description automatically generated with medium confidence

Figura -Segmento de Código do Menu de Pausa 1

O código declara três variáveis públicas do tipo GameObject: "pauseMenu", "mainMenu" e "game". A variável "pauseMenu" representa o objeto do menu de pausa, "mainMenu" representa o objeto do menu principal e "game" representa o objeto do jogo em si. A variável estática "isPaused" é declarada para controlar o estado de pausa do jogo.

A screen shot of a computer code

Description automatically generated with low confidence

Figura -Segmento de Código do Menu de Pausa 2

No método Start (), que é chamado antes da primeira renderização do jogo, o objeto "pauseMenu" é desativado, tornando-o invisível.

A screen shot of a computer program

Description automatically generated with medium confidence

Figura -Segmento de Código do Menu de Pausa 3

No método Update (), que é chamado a cada quadro de atualização, é verificado se a tecla Esc foi pressionada. Se o jogo estiver pausado, a função ResumeGame () é chamada para retomar o jogo. Caso contrário, a função PauseGame () é chamada para pausar o jogo.

A screen shot of a computer program

Description automatically generated with low confidence

Figura -Segmento de Código do Menu de Pausa 4

O método PauseGame () é responsável por pausar o jogo. Ele define o Time.timeScale como 0 para parar a contagem do tempo do jogo, ativa o objeto "pauseMenu" para exibir o menu de pausa e atualiza a variável "isPaused" para true.

A screen shot of a computer program

Description automatically generated with low confidence

Figura -Segmento de Código do Menu de Pausa 5

O método ResumeGame () é responsável por retomar o jogo a partir do menu de pausa. Ele desativa o objeto "pauseMenu", define o Time.timeScale como 1 para retomar a contagem do tempo do jogo e atualiza a variável "isPaused" para false.

A screen shot of a computer program

Description automatically generated with low confidence

Figura -Segmento de Código do Menu de Pausa 6

O método GoToMainMenu () é chamado quando o jogador seleciona a opção de voltar ao menu principal no menu de pausa. Ele desativa o objeto "game" e ativa o objeto "mainMenu", permitindo que o jogador retorne ao menu principal.

A picture containing text, screenshot, font, number

Description automatically generated

Figura -Segmento de Código do Menu de Pausa 7

O método QuitGame () é chamado quando o jogador seleciona a opção de sair no menu de pausa. Ele chama a função Application.Quit() para encerrar o jogo.

**Ecrã de Morte:**

A screen shot of a computer program

Description automatically generated with low confidence

Figura -Segmento de Código do Ecrã de Morte 1

O código declara duas variáveis públicas do tipo GameObject: "deathMenu" e "mainMenu". A variável "deathMenu" representa o objeto do menu de morte, enquanto a variável "mainMenu" representa o objeto do menu principal.

A picture containing text, screenshot, font

Description automatically generated

Figura -Segmento de Código do Ecrã de Morte 2

No método Start (), que é chamado antes da primeira renderização do jogo, o objeto "deathMenu" é desativado, tornando-o invisível.

A screen shot of a computer program

Description automatically generated with low confidence

Figura -Segmento de Código do Ecrã de Morte 3

No método Update (), que é chamado a cada quadro de atualização, é verificado se a tecla Espaço foi pressionada. Se a tecla for pressionada, o objeto "mainMenu" é ativado, tornando-o visível, e o objeto "deathMenu" é desativado, tornando-o invisível. Além disso, a função SceneManager.LoadScene() é chamada para recarregar a cena atual, reiniciando o jogo.

**Ecrã de Vitória:**

A screen shot of a computer program

Description automatically generated with medium confidence

Figura -Segmento de Código do Ecrã de Vitória 1

O código declara duas variáveis públicas do tipo GameObject: "victoryMenu" e "mainMenu". A variável "victoryMenu" representa o objeto do menu de vitória, enquanto a variável "mainMenu" representa o objeto do menu principal.

A screen shot of a computer program

Description automatically generated with low confidence

Figura -Segmento de Código do Ecrã de Vitória 2

No método Start (), que é chamado antes da primeira renderização do jogo, o objeto "victoryMenu" é desativado, tornando-o invisível.

A screen shot of a computer program

Description automatically generated with low confidence

Figura -Segmento de Código do Ecrã de Vitória 3

No método Update (), que é chamado a cada quadro de atualização, é verificado se a tecla Espaço foi pressionada. Se a tecla for pressionada, o objeto "mainMenu" é ativado, tornando-o visível, e o objeto "victoryMenu" é desativado, tornando-o invisível. Além disso, a função SceneManager.LoadScene() é chamada para recarregar a cena atual, reiniciando o jogo.

## concepção DO DESIGN

Para este jogo utilizei assets grátis que ajudaram bastante a definir o estilo do jogo. Fiquei mais inclinado a usar os assets que tinham personagens com Pixel Art pois era o que tinha em abundância tornando mais fácil manter o jogo esteticamente consistente.

Para compreender como fiz as animações das personagens será necessária uma breve explicação sobre como o Animator Funciona.

A screenshot of a computer

Description automatically generated with medium confidence

Figura -Animator

Utilizei este painel para configurar as animações da personagem, em termos de funcionalidade e muito semelhante a um fluxograma.

No canto superior esquerdo estão demonstrados os parâmetros de controlam as animações, existem três tipos diferentes: Float, Int, Trigger e Bool e estes parâmetros podem ser manipulados através dos scripts.

O retângulo verde chamado Entry é o que aponta para a animação principal, outro retângulo ciano chamado Any State serve para fazer a transição de qualquer animação para outra.

Para armazenar uma animação no retângulo apenas temos de clicar no retângulo desejado e arrastar o ficheiro para o local especificado na imagem abaixo.

A screenshot of a computer

Description automatically generated with medium confidence

Figura -Menu de uma Animação

Se clicar nas setas que interligam as animações, demonstrado abaixo, ira abrir um menu diferente.

A picture containing screenshot, businesscard, line, design

Description automatically generated

Figura -Interligação das Animações

A screenshot of a computer

Description automatically generated with medium confidence

Figura -Menu da Interligação das Animações

No inferior deste menu intitulado Conditions podemos adicionar os parâmetros criados para controlar as animações.

Alem de explicar como gerir as animações também irei explicar como eu criei as personagens.

A screenshot of a game

Description automatically generated with medium confidence

Figura -Scene Manager

Apos de criar um projeto ira aparecer um menu como ilustrado em cima no lado esquerdo do ecrã. Se clicar no espaço branco com o botão direito do rato ira aparecer outro menu neste menu é necessário escolher 2D Object>Sprite>Square após terminar isto terá a sua base.

A screenshot of a computer

Description automatically generated

Figura - Gerenciador do Square

Neste menu é que podemos escolher o modelo da personagem arrastando o ficheiro com o sprite para o local marcado acima. Alem disso se clicar no “Add Component” poderá adicionar componentes para dar funcionalidades as personagens.

**Personagem Principal:**

A pixel art of a person

Description automatically generated with low confidenceA picture containing pixel, cartoon, animation, animated cartoon

Description automatically generatedA pixel art of a person holding a sword

Description automatically generated with medium confidenceA pixel art of a person

Description automatically generated with low confidenceA picture containing pixel, LEGO, cartoon, animation

Description automatically generated

Figura -Animações da Personagem

Decidi usar este sprite sheet para criar a personagem sobre qual o jogador terá o controlo. A razão pela escolha era simples, primeiramente é pixel art que era uma obrigação, segundamente as animações estão extremamente bem feitas e finalmente não utiliza armas com projetais como por exemplo pistolas ou arco e flecha.

A screenshot of a computer

Description automatically generated with medium confidence

Figura -Animator da Personagem

No Animator da minha personagem principal criei quatro parâmetros:

* State - que é um int que controla o Run, Fall, Jump e o Idle, ele faz isto mudando a animação baseado no número em que o código para o movimento da personagem define.
* Attack – que é um trigger que ativa o Attack fazendo com que faz a animação de atacar quando é ativado pelo código de combate da personagem.
* Hurt – que também é um trigger que ativa o Hurt fazendo com que faz a animação de ser magoada quando é ativado pelo código de HP da personagem.
* IsDead – que é uma booleana que ativa o Death fazendo com que a personagem ativa a animação de morrer isto também é controlado pelo código de HP da personagem.

O retângulo Stay\_Dead é um loop com só uma imagem que foi criado para manter a personagem deitada apos a sua morte.

A screenshot of a computer program

Description automatically generated with medium confidence

Figura -Gerenciador da Personagem

Nesta imagem demostro a base da minha personagem que contem os seguintes componentes:

* Transform – que controla a posição da personagem;
* Sprite Renderer – que armazena o sprite;
* Rigidbody 2D – que serve para fazer com que a personagem seja um objeto físico no mapa;
* Box Collider 2D – que deteta as colisões da personagem;
* Player Movement – que é um script que controla o movimento da personagem;
* Animator – que gere as animações da personagem;
* Player Combat – que é o script que controla o combate da personagem;
* Die – que também é um script que gere o HP e a morte da personagem.

**Boss/Chefe:**

A picture containing clothing, cartoon, magenta, screenshot

Description automatically generatedA pixel art of a character

Description automatically generated with low confidence

A picture containing clothing, footwear, cartoon, person

Description automatically generatedA picture containing cartoon, anime

Description automatically generated

Figura -Animações do Boss

Apos de encontrar a personagem principal decidi ver os outros sprites que o criador tinha feito e foi assim que encontrei este. As razoes pelo qual escolhi este são as mesmas que antes só que desta vez estava a procurar por algo que pudesse ser o Chefe do jogo tornando o estilo deste sprite ideal.

A screenshot of a computer

Description automatically generated with medium confidence

Figura -Animator do Boss

No Animator do meu chefe criei quatro parâmetros:

* Attack – que é um trigger que ativa o Attack fazendo com que a animação do ataque seja executada, este trigger é detetado quando o código deteta a personagem.
* Die – Que ativa a Death que faz com que executa a animação de morrer, este trigger é ativado pelo código da vida do chefe, neste caso não seria necessário o stay-dead pois o chefe desaparece após morrer, mas decidi deixar como segurança.

A screenshot of a computer program

Description automatically generated with medium confidence

Figura -Gerenciador do Boss

Nesta imagem demostro a base do meu chefe que contem os seguintes componentes:

* Transform – que controla a posição do chefe;
* Sprite Renderer – que armazena o sprite;
* Rigidbody 2D – que serve para fazer com o chefe seja um objeto físico no mapa;
* Box Collider 2D – que deteta as colisões do chefe;
* Animator – que gere as animações do chefe;
* Boss Health – que é um script que gere o HP e a morte do chefe.
* Player Combat – que é o script que controla quando o chefe ataca;
* Boss – é um script para orientar o chefe para que esteja sempre a olhar para a personagem.

**Mapa:**

Da mesma maneira que criei as personagens criei uma grid que serviu como uma base para criar um mapa. Cliquei no espaço branco no menu de novo e segui o caminho 2D Object>Tilemap>Rectangular que criou uma grid onde arrastei os meus tiles (pecas do mapa) para criar um mapa.

A picture containing pattern, rectangle, square

Description automatically generated

Figura -Grid

A screenshot of a video game

Description automatically generated

Figura -Tilemap

A screenshot of a computer program

Description automatically generated with low confidence

Figura -Gerenciador do Mapa

A imagem em cima demostra a minha base para o mapa em que o componente Transform apenas serve para mudar a posição do mapa, os componentes a roxo servem para renderizar e demonstrar o mapa enquanto os que os componentes verdes sevem para tornar o mapa num objeto físico e gerir as suas colisões.

A picture containing screenshot, water, nature, art

Description automatically generated

Figura -O Mapa

A imagem em cima ilustra o mapa do meu jogo, decidi manter o mapa plano pois se houvesse plataformas ou locais onde a personagem poderia saltar o chefe não iria conseguir o alcançar.

**Menus:**

Da mesma maneira que criei as personagens criei uma “Panel” que serviu como uma base para criar um menu. Cliquei no espaço branco no menu de novo e segui o caminho UI>Panel que criou um painel. Após criar o painel criei uma “Image” (UI>Image) que guardava uma imagem, utilizando esta tática criei 4 menus. Além disso também criei botões (UI>Button) para o menu principal e para o menu de pausa.

A screenshot of a computer

Description automatically generated

Figura -Gerenciador do Botão

A imagem em cima ilustra o menu que controla as funcionalidades dos botões. No local apontado na imagem podemos arrastar a imagem que desejamos e o botão tomara o aspeto da imagem.

A screenshot of a computer

Description automatically generated with medium confidence

Figura -Gerenciador do Botão (continuação)

O primeiro item apontado na imagem serve para guardar uma imagem para o botão. Este item ira mudar a aparência do botão para o que lá estiver guardado se estiver o rato sobre o botão.

O segundo item apontado é igual ao primeiro só que muda o botão se ele for clicado.

O método On Click () que se apresenta no inferior da imagem serve para arrastar e colocar o script que o botão ira utilizar.

A screenshot of a video game

Description automatically generated

Figura -Menu Principal

O menu principal foi feito utilizando o mesmo fundo que o mapa do jogo alem disso tambem inseri outra camada com arvores que vinha com o tileset do mapa.

No centro do menu coloquei o meu logotipo juntamente com dois botões, o botão “Play” faz com que o jogo comece e o botao “Quit” faz com que o jogo feicha, ambos estes botoes utilizam o script Menu Principal mencionado em cima.



Figura -Menu de Pausa

Este menu é accessivel precionando Esc. Foi feito utilizando um “Panel” semi-transparente como o fundo isto permite ver onde ficou antes de pausar o jogo. O pequeno quadrado serve como um fundo para destacar os botoes, dentro deste quadrado coloquei um texto a dizer Pause para saberem que o jogo esta pausado, abaixo disto coloquei dois botoes o laranja serve para resumir o jogo (tambem podera fazer isto a clicar esc de novo), o botao verde o levara de volta para o menu principal. No inferior do quadrado apresenta se um botao “Quit” que sai do jogo.

A picture containing sky, screenshot, cartoon, aqua

Description automatically generated

Figura -Ecrã de Morte

Este ecra aparece se o jogador morrer. Decidi imitar um estilo iconco para o ecra de morte, digo que o estilo é iconico pois apresnta se em varios jogos populares como DarkSouls, Elden Ring e GTA V. Este estilo envolve escorecer o ecra, que faco com um “Panel”, e ter uma barra preta com letras vermelhas com uma forma criativa de dizer “moreste”.

A picture containing screenshot, pc game, cartoon, video game software

Description automatically generated

Figura -Ecrã de Vitoria

Para o ecrã de vitória decidi copiar o de morte, mas mudar as letras para verde e apresentar uma mensagem de vitória.

## desenvolvimento e testes realizados

Durante este projeto efetuei vários testes para verificar as várias funcionalidades do programa encontrei que estes testes foram de extrema importância para o melhoramento do jogo e também alertou a necessidade de mudar o software que estava a usar.

**Unreal Engine:**

**Teste 1:**

Após terminar o primeiro vídeo referido na pesquisa efetuei o primeiro teste em que tudo estava a correr como planeado exceto algumas falhas no mapa. Estas falhas eram devido ao método que usei para fazer com que as texturas do mapa fossem um objeto físico. O erro ocorria porque não fechava o polígono que dava a colisão ao objeto fazendo com que houvesse pequenos espaços entre as texturas e a personagem caia pelo mapa.

**Teste 2:**

Durante o segundo vídeo mencionado na pesquisa fiz um teste logo depois de ter configurado as animações de atacar e não estava a funcionar. Isto era devido ao facto de que o código antigo fazia um update a animação da personagem cada tick.

**Teste 3:**

Apos de ter emendado o código com um simples “if” testei de novo e estava a funcionar, agora só faltava programar a vida e o dano da personagem.

**Teste 4:**

Continuei a ver o vídeo até uma parte em que era para testar se a personagem estava a dar dano, isto foi feito através de um “print” em que o sistema imprimia o valor do dano ao lado e estava a funcionar.

**Teste 5:**

Quando terminei o vídeo testei para ver se a personagem iria morrer se a vida chegasse a 0 por isso testei e não deu. Passei horas a tentar descobrir o erro e descobri que o código estava a executar perfeitamente apenas não queria fazer a animação e não sei o porquê.

**Unity:**

**Teste 1:**

Após terminar os primeiros três tutoriais que vi sobre a Unity efetuei o meu primeiro teste do jogo, o movimento da minha personagem estava a funcionar como deveria e não encontrei nenhuns erros.

**Teste 2:**

Após visualizar os próximos dois vídeos, segui os passos e testei de novo, concluindo fiquei com as animações da personagem a funcionar.

**Teste 3:**

Seguidamente visualizei o próximo vídeo e segui os passos fazendo alterações onde achei necessário, quando testei estava funcional pois a minha personagem já não tinha saltos infinitos.

**Testes 4 e 5:**

Depois de terminar o seguinte vídeo sobre criar um sistema de combate, apliquei o conhecimento que tinha adquirido e testei só que não tinha alvo para verificar se estava a funcionar e terminei o teste. Necessitava um alvo para testar o combate por isso copiei a minha personagem só que tirei o componente do movimento, isto efetivamente criou o meu alvo e também permitia testar o que iria acontecer a personagem se levasse dano, testei novamente e estava tudo funcional.

**Teste 6:**

Visualizei o vídeo sobre como criar uma barra de vida e criei uma com os recursos que tinha coloquei esta barra no meu alvo de teste e verifiquei que estava funcional

**Teste 7 e 8:**

O seguinte vídeo que segui demostrou como criar um boss/chefe e apos de visualizar e seguir os passos, testei. O teste foi um sucesso com o único erro sendo que o chefe as vezes teletransportava para trás da personagem, mas decidi não arranjar isto pois achei um mecanismo interessante. Seguidamente copiei a barra de vida da personagem e colei no chefe, com algumas alterações, apos de testar também estava funcional.

**Teste 9:**

Após visualizar o vídeo sobre como criar um menu principal, especificado na minha pesquisa, segui as suas instruções e criei um menu como as minhas especificações. Após ter o feito testei e confirmei que estava funcional.

**Teste 10:**

O seguinte vídeo que vi explicou como criar um menu de pausa e após visualizar e seguir os pacos o meu estava feito. Quando efetuei o teste o menu pausava o jogo e era visível, mas não conseguia interagir com os botões, após alguns minutos de pesquisa e troubleshooting cheguei a conclusão que mesmo não sendo visível o menu principal estava a bloquear os botões. A minha solução para este problema era simplesmente desativar o menu que não estava a ser utilizado por isso meti todos os menus na mesma “scene” e programei a lógica.

**Teste 11 e 12:**

Para estes testes tinha criado o ecrã de vitoria e o de morte utilizando o que tinha aprendido. Iniciei o jogo e morri para confirmar se um estava a funcionar e para o outro iniciei o jogo e matei o chefe, ambos os ecrãs estavam funcionais.

## IMPLEMENTAÇÃO DO SITE

**Código:**

A picture containing text, screenshot, font

Description automatically generated

Figura -Segmento de Código do Site 1

A seção do cabeçalho contém a tag <title> que define o título da página exibido na guia do navegador. Além disso, há links para um arquivo CSS externo chamado "styles.css" e os estilos do plugin Slick Carousel.

A picture containing text, screenshot, font, number

Description automatically generated

Figura -Segmento de Código do Site 2

Esta seção do documento contém um cabeçalho com um logotipo e o título do jogo.

A screen shot of a computer

Description automatically generated with medium confidence

Figura -Segmento de Código do Site 3

Esta seção do carrossel exibe uma série de slides usando o plugin Slick Carousel. Os slides são imagens representadas por tags <img>.

A screenshot of a computer program

Description automatically generated with low confidence

Figura -Segmento de Código do Site 4

Esta seção "about" apresenta informações sobre o jogo em texto, incluindo uma breve descrição, um “embed” com o tutorial par o jogo e um botão de download.

A picture containing screenshot, text, multimedia software, software

Description automatically generated

Figura -Segmento de Código do Site 5

O “footer” (rodapé) contém um o nome do jogo e uma mensagem declarando que não tenho os direitos autorize sobre os assets do jogo.

Os scripts JavaScript no final do corpo do documento incluem as bibliotecas jQuery e Slick Carousel, além de um código que inicializa o carrossel usando a função slick().

**Design do Site:**

A screenshot of a video game

Description automatically generated

A screenshot of a video game

Description automatically generated

Figura -Site do Jogo

Decidi manter o site relativamente simples pois não era o foco deste projeto. Para as cores principais do site decidi basear me no meu logotipo e segmentar o site pelas cores.

O cabeçalho do meu site demostra o logotipo do jogo e o seu nome. O corpo contem um carrossel para ilustrar algumas imagens do jogo seguido pelo uma breve explicação do que é o jogo, após isto tenho um vídeo de tutorial e um botão para instalar o jogo seguido pelo rodapé.

# CONCLUSÃO

Este projeto foi uma experiência. Digo isto pois teve os seus altos e baixos, mas no geral acho que acabou por ser uma experiência boa para a vida.

Ao longo deste projeto tive vários momentos em que pensava que não iria conseguir acabar e devido a isso perdia a minha postura e ficava miserável, sentia me insuficiente, mas o importante é que aprendi a não desistir e acabei por ganhar um enorme respeito pelos criadores de vídeo jogos pois é preciso ter muita paciência as vezes.

Em termos do meu jogo acho que poderia estar melhor e simplesmente não esta, isto é, devido a minha natureza de deixar tudo para a última e depois acabo por ter de fazer várias coisas ao mesmo tempo, mas apos este projeto acho que aprendi a minha lição.

Concluindo terminei este projeto com novos conhecimentos adquiridos sobre a criação de vídeo jogos e consegui responder a algumas das minhas respostas a cerca deles, alem disto aprendi a não deixar tudo para a última da hora.

# REFERÊNCIAS BIBLIOGRÁFICAS

Personagem:

<https://clembod.itch.io/warrior-free-animation-set>

Boss:

<https://clembod.itch.io/bringer-of-death-free>

TileSet:

<https://ansimuz.itch.io/sunny-land-pixel-game-art>

UI:

<https://oarielg.itch.io/2d-simple-ui-pack>

Vídeos:

<https://www.youtube.com/watch?v=Ii-scMenaOQ>

[https://www.youtube.com/watch?v=QkbGr1rAya8&t](https://www.youtube.com/watch?v=QkbGr1rAya8&t=2109s)

<https://www.youtube.com/watch?v=Uv5tfMSKlnU>

[https://www.youtube.com/watch?v=GChUpPnOSkg&t](https://www.youtube.com/watch?v=GChUpPnOSkg&t=1575s)

[https://www.youtube.com/watch?v=65E-q0JxYwU&t](https://www.youtube.com/watch?v=65E-q0JxYwU&t=979s)

<https://www.youtube.com/watch?v=LEUhxe9vUOM&t>

<https://www.youtube.com/watch?v=BLfNP4Sc_iA>

<https://www.youtube.com/watch?v=AD4JIXQDw0s&t>

<https://www.youtube.com/watch?v=-GWjA6dixV4&t>

<https://www.youtube.com/watch?v=9dYDBomQpBQ>