UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE – UERN
FACULDADE DE CIÊNCIAS EXATAS E NATURAIS – FANAT
DEPARTAMENTO DE INFORMÁTICA – DI

Antonio Fabrício Martins de Lima

Design de Jogo: Cycles Eternal

MOSSORÓ / RN
2023

Antonio Fabrício Martins de Lima

Design de Jogo: Cycles Eternal

Relatório apresentado ao curso de Ciência da Computação da Universidade do Estado do Rio Grande no Norte como requisito da disciplina de Trabalho de Diplomação, sob a orientação do Prof. Dr. Carlos Heitor Pereira Liberalino.

Mossoró / RN
2023

Antonio Fabrício Martins de Lima

Design de Jogo: Cycles Eternal

Relatório apresentado como pré-requisito para obtenção do título de Bacharel em Ciência da Computação da Universidade do Estado do Rio Grande do Norte – UERN, submetida à aprovação da banca examinadora composta pelos seguintes membros:

Aprovado em 28/02/2024

Banca Examinadora

Prof. Dr. Carlos Heitor Pereira Liberalino (Orientador)
Universidade do Estado do Rio Grande do Norte – UERN

Profa. Dra. Ceres Germanna Braga Morais (Examinadora)
Universidade do Estado do Rio Grande do Norte – UERN

Prof. Dr. Francisco Chagas de Lima Junior (Examinador)
Universidade do Estado do Rio Grande do Norte – UERN

Dedico este trabalho a minha mãe Ligía, ao meu pai Edimar, a todos os meus familiares e amigos que estiveram presente em toda essa jornada.

AGRADECIMENTOS

Gostaria de agradecer primeiramente aos meus pais, pela luta, educação, conselhos, amor, gratidão, orgulho e tudo que passamos ao decorrer de todos esses anos, sem eles eu não sei se estaria onde estou e seria o que sou hoje.

Agradeço também a todos os meus professores, mestres e tutores, que me fizeram absorver todo o conhecimento repassado por eles e que me engrandeceram de alguma forma, estes são também responsáveis pela conclusão deste trabalho. Em especial eu agradeço ao professor Heitor, que foi de extrema relevância na elaboração conjunta deste projeto, e por ter acreditado nesse projeto.

Agradeço aos amigos que fiz ao longo de minhas jornadas, sejam estas acadêmicas ou não, que me acompanharam e me deram forças nos mais tardares momentos. Entre estes, o grupo do "Bonde", que são amigos de longa data dos quais pretendo levar até o fim da minha vida; O grupo do "GDP" que embora eu tenha convivido mais recentemente com eles, se mostram sempre presentes e imprescindíveis, e por eles guardo devo muito, e também pretendo levar até o fim da minha vida.

Dou destaque as grandes amizades que fiz nesse estágio acadêmico, pois sem eles eu não sei o que seria de mim, meu muito obrigado em especial a Alber, Isau, Igor, João, Márcio e Liza. Foi graças a vocês que cheguei onde estou, meu muito obrigado!

RESUMO

O desenvolvimento de jogos independentes no Brasil, embora esteja em ascensão, o número de lançamentos(entre 1.000 e 2.000) ainda é baixo em comparação a outros países devido a diversos fatores. Este projeto tem como objetivo incentivar o desenvolvimento do mercado de jogos brasileiros e expandir o estudo academico sobre jogos, além de propor uma forma de entretenimento aos jogadores. Neste trabalho será explorado alguns conceitos do jogo Cycles Eternal desenvolvido na Unity que tornam estes pontos bem vistos. No presente estudo serão citados conceitos e funcionalidades referentes a Sprites, Level Desing, monstros e sistemas de vida dos inimigos e personagens, assim como o aúdio do ambiente.

Palavras chave: Cycles Eternal , Unity ,Sprites, Level Desing.

ABSTRACT

The development of independent games in Brazil, although on the rise, the number of releases (between 1,000 and 2,000) is still low compared to other countries due to several factors. This project aims to encourage the development of the Brazilian games market and expand the academic study of games, in addition to offering a form of entertainment for players. This work will explore some concepts from the game Cycles Eternal developed in Unity that make these points well seen. In this study, concepts and functionalities relating to Sprites, Level Design, monsters and enemy and character life systems will be mentioned, as well as environmental audio.

Keywords: Cycles Eternal , Unity ,Sprites, Level Desing.

SUMÁRIO

1. INTRODUÇÃO

Este capítulo tem como objetivo nortear o leitor quanto à estrutura do trabalho, além de apresentar o tema de forma objetiva, bem como justificar a escolha da temática abordada.

1.1 Problemática

Embora, este projeto se trate de um desenvolvimento de um jogo para um fim específico (desenvolvimento de um game), esse simples gesto o torna mais abrangente para a comunidade de desenvolvedores do Brasil, trazendo mais visibilidade ao mercado de jogos independentes brasileiro.

1.2 Objetivos

É interessante neste trabalho tratar quais funcionalidades desejáveis para se desenvolver um jogo. Tratando o Cycles Eternal como mecanismo para inspirar um novo projeto que possa preencher o máximo de lacunas e trazer uma agradável e eficiente experiência aos jogadores.

Desse modo, o objetivo deste trabalho é a demonstração das funções e mecânicas do jogo Cycles Eternal, para expandir o mercado e ensino do desenvolvimento de jogos brasileiro.

1.3 Estrutura do Documento

O restante do documento está dividido em capítulos. O segundo capítulo trata-se do referencial teórico. Neste, há uma fundamentação relativa aos mecanismos relevantes na progressão do projeto.

No terceiro capítulo conterá conterá o resultado, ou seja, o desenvolvimento do jogo monstrando suas funções, mecânicas e desings visuais. Por fim, o quarto capítulo conterá uma conclusão para o trabalho, seguido das referências bibliográficas.

2. REFERENCIAL TEÓRICO

Neste capítulo serão incluídas as tecnologias utilizadas no decorrer do trabalho, tanto para basear a pesquisa quanto para modelá-la.

2.1 C#

O C# (“Cê Sharp”) é uma linguagem de programação moderna, orientada a objetos e de alto nível, desenvolvida pela Microsoft. Ela é amplamente utilizada para diversos fins, incluindo o desenvolvimento de jogos, aplicativos web e desktop, e software em geral. C# foi criada em 2000 pela Microsoft como parte da plataforma .NET. O desenvolvimento da linguagem foi liderado por Anders Hejlsberg, um dos principais nomes da indústria de software, também responsável por outras linguagens como Delphi e Turbo Pascal. A criação de C# teve como objetivo principal oferecer uma alternativa moderna e eficiente para o desenvolvimento de software em Windows. A linguagem foi projetada para ser fácil de aprender e usar, ao mesmo tempo que oferece recursos poderosos e flexíveis para atender às necessidades de diversos tipos de aplicações.

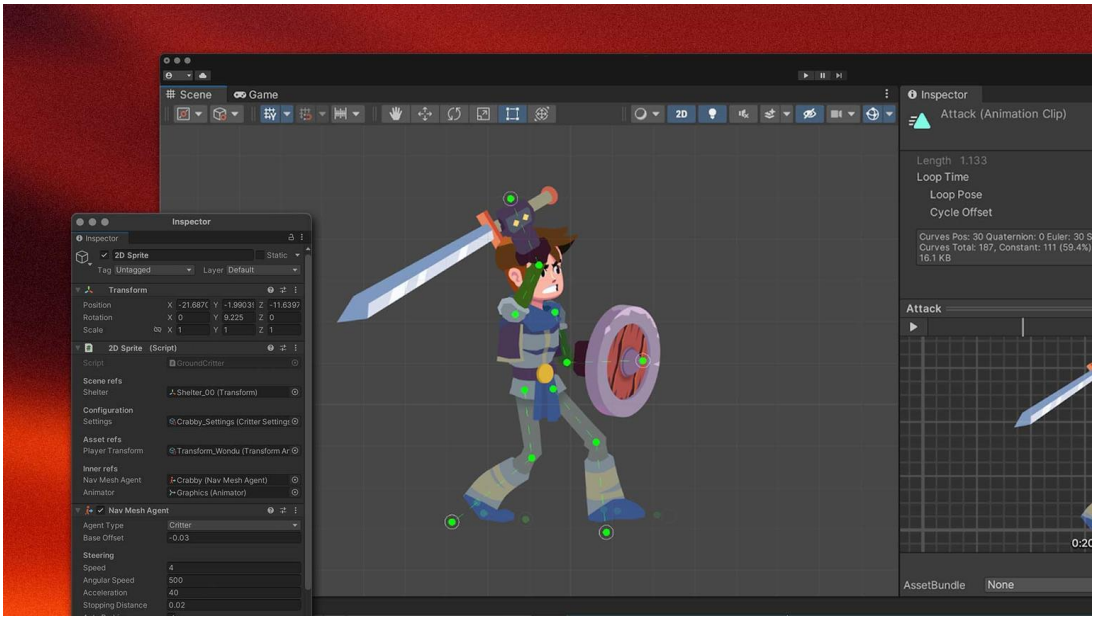
No contexto do desenvolvimento de jogos, C# é uma das linguagens mais populares, especialmente em conjunto com o motor de jogo Unity. Ela é uma linguagem compilada, o que significa que o código é convertido diretamente em linguagem de máquina antes da execução. Isso garante um alto desempenho e escalabilidade, características essenciais para jogos com gráficos complexos e jogabilidade exigente.

2.2 Unity

A Unity é um motor de jogo (em inglês, “engine”) multiplataforma líder de mercado, desenvolvido para facilitar a criação de jogos 2D e 3D de alta qualidade. Lançada em 2005, a Unity se tornou uma ferramenta essencial para diversos tipos de desenvolvedores, desde iniciantes até profissionais experientes. Por sua vez a Unity oferece uma interface intuitiva baseada no sistema drag-and-drop (clicar e soltar), permitindo aos usuários criarem jogos sem a necessidade de programar código complexo.

Para maior flexibilidade e controle, a Unity integra-se perfeitamente com a linguagem de programação C#, permitindo aos desenvolvedores criarem scripts personalizados.

Figura 1 – Unity: Ambiente de desenvolvimento



Fonte: Unity(2022)

3. RESULTADOS

Após análises e pesquisas criou-se o jogo, desenvolvido na Unity. O jogo produzido durante esse projeto, denominado Cycles Eternal , trata-se de um plataforma shooter (estilo de jogo relacionado a atirar e bater) , em que o personagem principal tem o objetivo de atravesar as quatro fases, derrotando inimigos e por fim eliminando o chefão deles.

Figura 2 – Menu

Fonte: Autoria própria

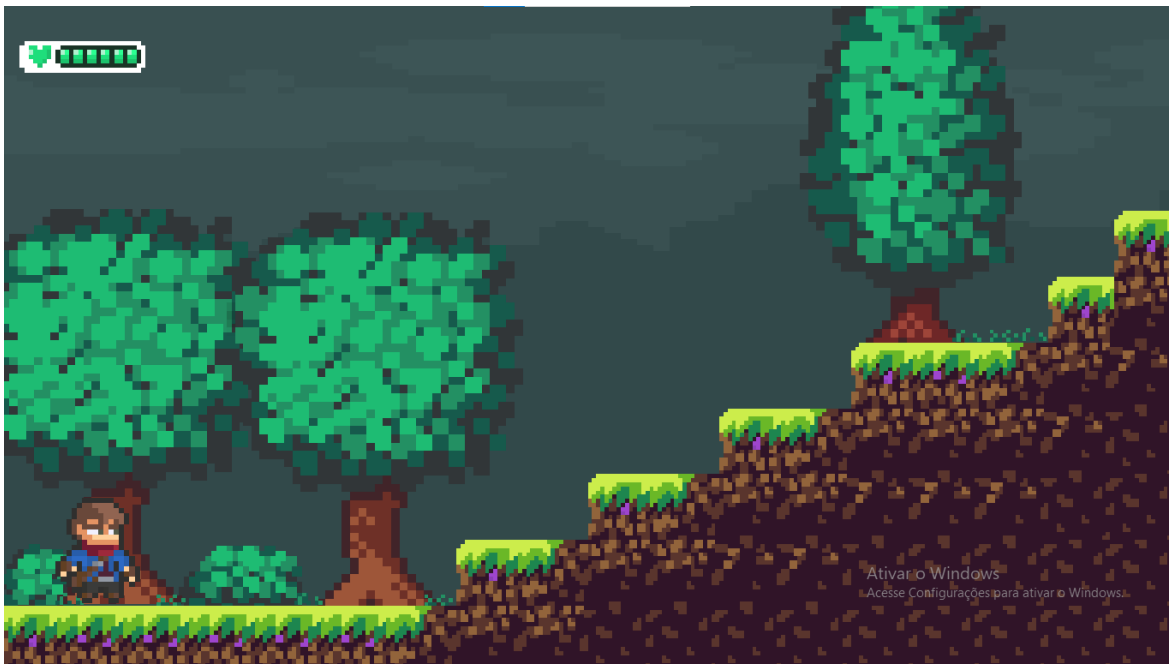
A Figura 2 apresenta o menu inicial do jogo, no qual possui três botões que realizam determinada ações. O botão 1, de jogar , ao clicar inicia o jogo. O botão 2, de opções, mostra os comandos de movimentação e ataque do personagem. O botão 3, de sair, ao clicar fecha o jogo.

Figura 3 – Menu de comandos

Fonte: Autoria própria

A Figura 3 apresenta o menu de comandos do jogo, no qual a letra “A” move o personagem para a esquerda, a letra “D” o personagem para direita, o “SPACEBAR” faz o personagem pular, o botão esquerdo do mouse realiza um ataque com a espada e o botão direito atira um poder.

Figura 4 – Primeira fase



Fonte: Autoria própria

A Figura 4 apresenta a primeira fase do jogo, onde é iniciado a jornada do personagem principal, mostrando uma floresta com alguns inimigos e plataformas, e exibindo as mêcanicas do jogo, o sistema de vida e dano.

Figura 5 – Barra de vida

Fonte: Autoria própria

O sistema de vida do personagem(Figura 5), demonstrada através de uma barra de vida verde, representa a funcionalidade de sobrevivência do jogador. O jogador possui seis vidas, a cada ataque que o personagem toma de um inimigo, essa vida abaixa. Caso a barra de vida não esteja mais preenchida significa que o jogador perdeu e a fase será reiniciada.

Figura 6 – Código de vida(personagem)

```
1 reference
IEnumerator Damage(){


    for(float i= 0f;i< 1f;i+=0.1f){
        sprite.enabled = false;
        yield return new WaitForSeconds(0.1f);
        sprite.enabled = true;
        yield return new WaitForSeconds(0.1f);
    }

    ivuneravel = false;
}

0 references
public void DamagePlayer(){
    ivuneravel = true;
    vida --;
    StartCoroutine (Damage());
}
```

Fonte: Autoria própria

Nesta tela(Figura 6) o código demonstra como funciona o sistema de dano para a retirada de vida do jogador. Afim de demonstrar que o “sprite” do jogador recebeu dano, o sprite oscila de cor para demonstrar ivunerabilidade e retirada de vida, quando o jogador retorna a sua cor padrão ele poderá receber dano novamente.

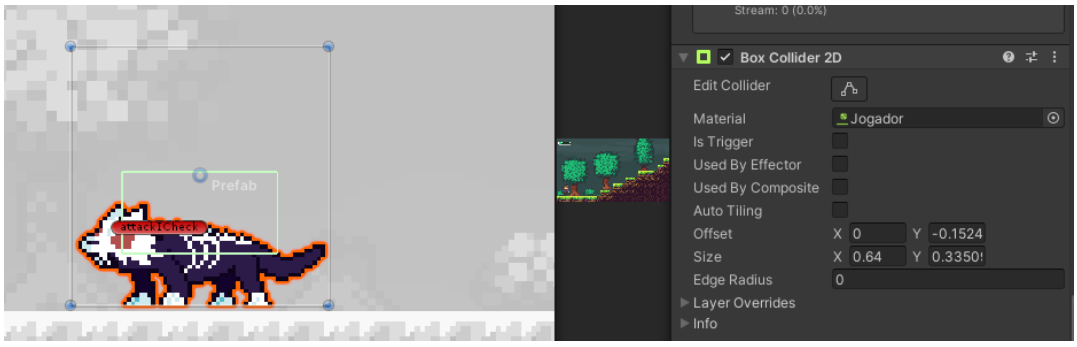
Figura 7 – Código de vida(inimigos)

```
0 references
public void DamageEnemy(){
    vida -= 1;
    StartCoroutine (Damage());

    if (vida < 1){
        Invoke("morte", 1f);
    }
}

1 reference
IEnumerator Damage(){
    sprite.color = Color.red;
    yield return new WaitForSeconds(0.1f);
    sprite.color = Color.white;
}
```

Fonte: Autoria própria

Assim como a Figura 6, o código demonstrado na Figura 7 demonstra o dano recebido pelo inimigo, com o mesmo funcionamento, tendo como diferencial o sprite de invecibilidade ser vermelho e que caso o inimigo esteja com menos que um de vida, ele morra.

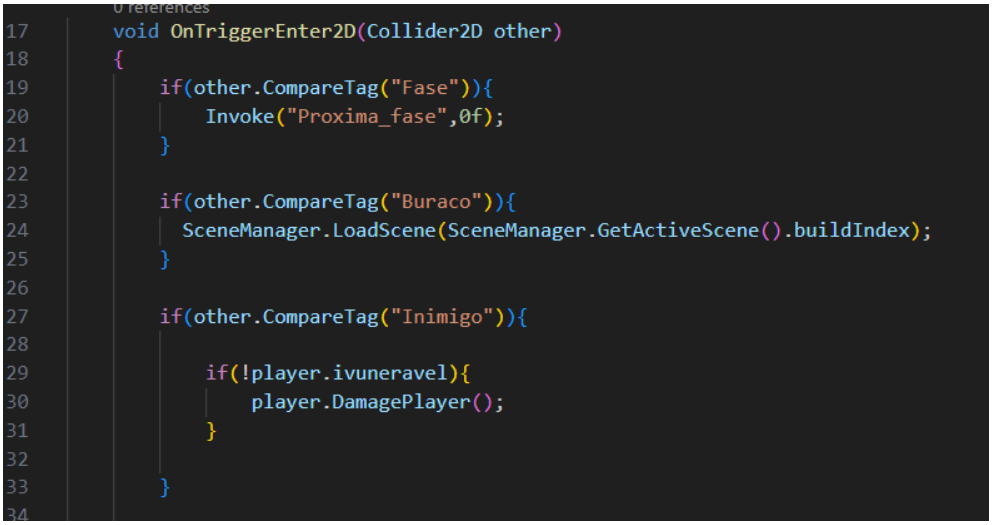
Figura 8 – Sistema de colisão(inimigos)



Fonte: Autoria própria

Para que o personagem realize a função de tomada de dano é preciso que o inimigo o atinja de alguma forma, o que é denotado através do mecanismo de colisão(Figura 8). Esse mecanismo determina como objetos virtuais interagem uns com os outros dentro do ambiente do jogo, caso haja uma colisão com algum elemento ou inimigo o sistema da uma resolução de dano ao jogador, ou seja, ao monstro encostar ou realizar um disparo em que atinja o jogador, o mesmo irá perder uma vida. Essa função é denotada por uma barra verde em volta do elemento ou inimigo.

Figura 9 – Código do sistema de colisão (inimigos)



```
0 references
void OnTriggerEnter2D(Collider2D other)
{
    if(other.CompareTag("Fase")){
        Invoke("Proxima_fase",0f);
    }

    if(other.CompareTag("Buraco")){
      SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    }

    if(other.CompareTag("Inimigo")){

        if(!player.ivuneravel){
            player.DamagePlayer();
        }

    }
```

Fonte: Autoria própria

Colider é uma função já existente no Unity, servindo como um verificador. Na programação acima(Figura 9), a função colider é comparada com a tag definida pelo programador na descrição do projeto. Em caso de êxito, o metódo é invocado a partir da definição do programador, como exemplo do código acima, temos que ao jogador entra em contato com a caixa de colisão do inimigo, a função de “DamagePlayer” é escrito.

Figura 10 – Inimigos comuns

Fonte: itchi.io(2024)

O jogo possui cinco inimigos, sendo quatro deles comuns(Figura 10), portanto, esses quatros aparecem nas três fases iniciais, e o quinto sendo o chefão que aparecerá somente na última fase. A Figura 10 apresenta os inimigos comuns, sendo eles o fantasma, o morcego, o lobo e o diabrete. O fantasma, morcego possuem um padrão de ataque que ao ver o jogador tentam colidir nele, o lobo por sua parte ao ver o jogador tenta ir para cima e mordê-lo, já o diabrete possui a mecânica de disparo, que caso o disparo de energia colida no jogador ele irá tomar dano de um de vida. Cada inimigo, assim como jogador também possue o sistema de vida, caso sejam atingidos os mesmos perdem vida. Tanto o morcego e o diabrete por serem menores e estarem em locais de alto perigo do jogador cair de plataformas, portam um de vida, em contrapartida, o fantasma e o lobo, por serem maiores e aparecerem em lugares menos perigosos, precisam de dois danos para serem eliminados.

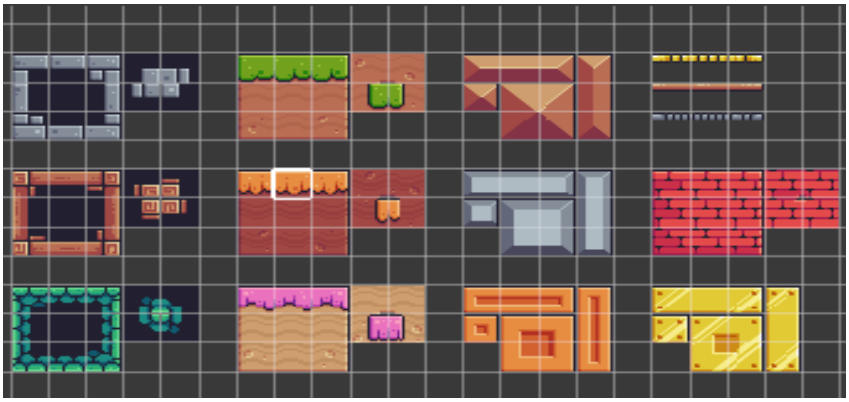
Figura 11 – Inimigo final

Fonte: itch.io(2024)

Este modelo(Figura 11) representa o último inimigo do jogo, o chefão. Por sua vez, o chefão possui três padrões de ataques diversificados. O primeiro é a colisão com o jogador, o segundo são os disparos de facas de pedra e o terceiro um raio de energia, que só sera ativado caso esteja com quatro de vida, ao utiliza-lo este ataque entrará em

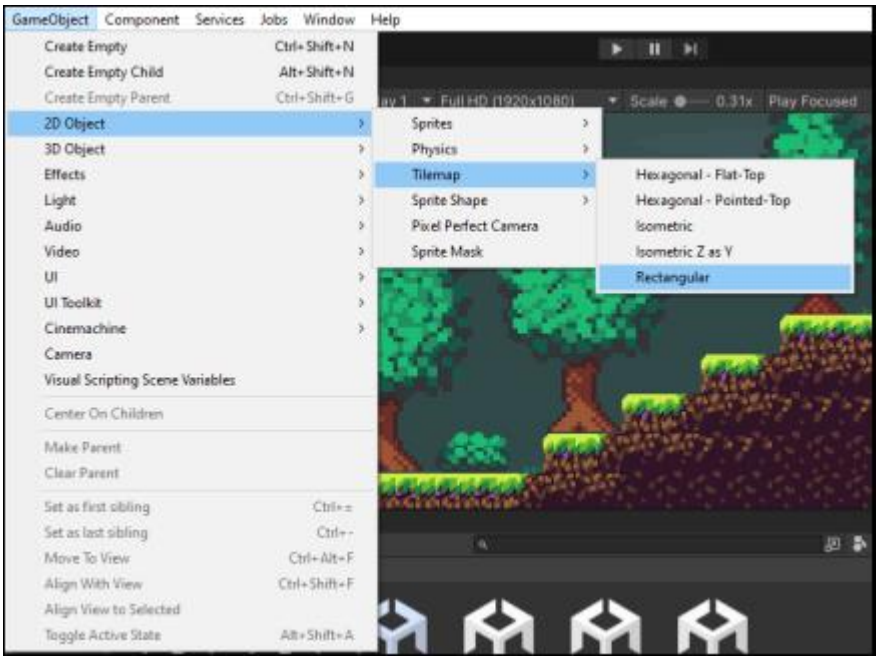
modo de espera de 5 segundos. O chefão portanto, ira escolher o melhor padrão a ser seguido dependendo da distância do jogador e sua vida.

Figura 12 – Tiles



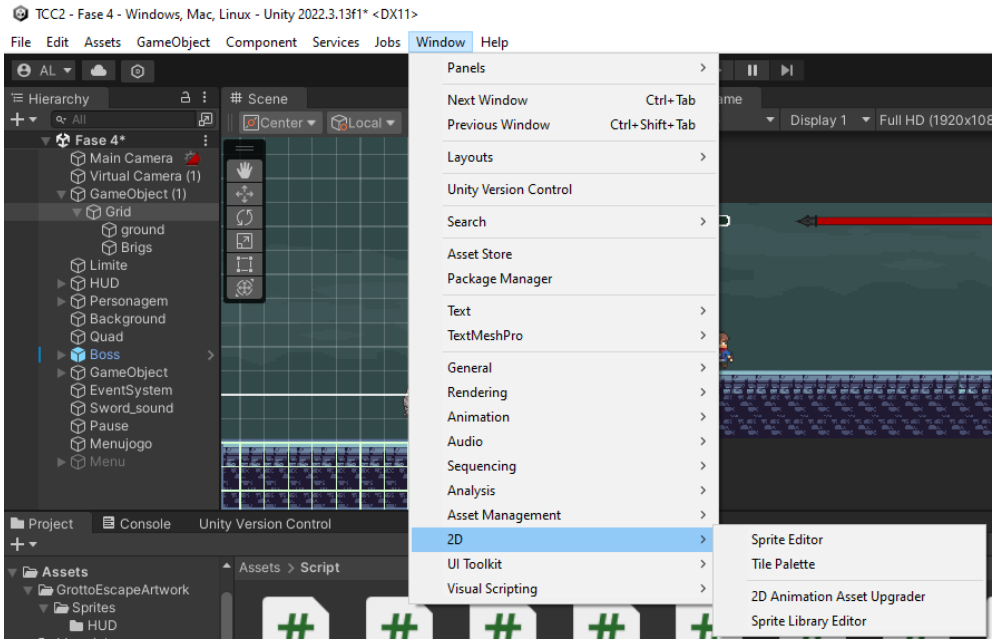
Fonte: Unity(2024)

Figura 13 – Grid



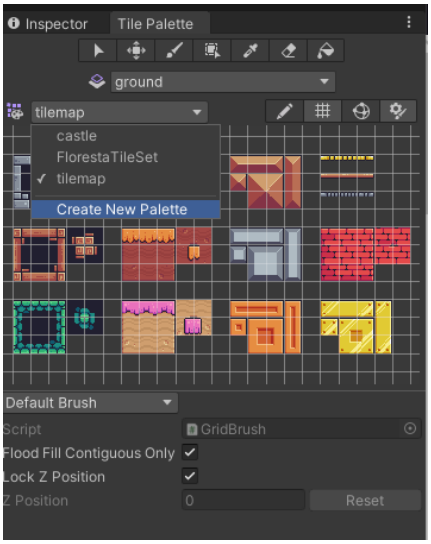
Fonte: Unity(2024)

Figura 14 – Janela de criação do Tile Pallete



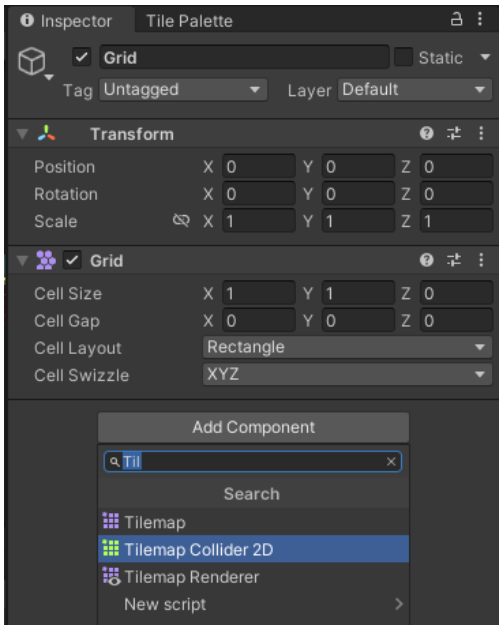
Fonte: Unity(2024)

Figura 15 – Janela do Tile Palette com suas funções



Fonte: Unity(2024)

Figura 16 – Janela do Inspector



Fonte: Unity(2024)

Os “Tiles”(Figura 12) são as imagens individuais que compõem o cenário, podendo ser chão, paredes, plataformas, etc. Cada bloco na Figura 12 é um “Tile”. Assim como outras ferramentas utilizadas da unity, o level design foi modelado utilizando a ferramenta de “Tile Map”(Figura 13) que é a base do nível, denominada “Grid”, que pode ser criada na opção “2D Object”. Para utilizar o sistema de “Grid” é necessário a criação da “Tille Pallete”(Figura 14 e 15) que pode ser gerada na opção “Window” do Unity , onde irá criar uma janela permitira a inserção da imagem apenas arrastando ela até o espaço vazio, que consequentemente denominará como “Tiles”. A “Tille Pallete” possui funcionalidades de organizar e gerenciar tiles para uso em seus níveis de mapa de tiles 2D. Ela utiliza uma forma visual de selecionar e pintar tiles em seu mapa de tiles, similar a usar um pincel e uma paleta de cores.

Para a utilização de colisão com o cenário utiliza-se a opção de “Inspector” (Figura 16) que está localizado ao lado da aba de “Tile Pallete”. Nesta opção pode-se adicionar outra ferramente do Unity, o “Tilemap Collider 2D” que realiza automaticamente a inserção de colisões aos blocos e estruturas colocadas pelo “Tile Pallete” selecionado. Para decoração utiliza-se o mesmo sistema de “Tile Pallete”, mas sem a inserção da

ferramente “Tilemap Collider 2D”.

Com isso foi feito o “Level Design” das fases, sendo a primeira, segunda e terceira com um ambiente mais sombrio e desolado com vales e árvores mortas , e por fim a quarta com o fim de tudo, remetendo a somente o inimigo final.

Para cada animação foi utilizado aúdios de uso gratuitos, assim como também para a ambientação, onde foi utilizado duas músicas, uma centrada para as fases comuns(da primeira a terceira), e outra para a fase do chefão(quarta).

4. CONCLUSÃO

Com este trabalho, foi feito um produto de entrenteinimento, capaz de divertir pessoas, gerar inovações de jogos independentes brasileiros internacionais, incentivar o estudo de desenvolvimento de jogos acadêmicos, trabalho em grupo e afins.

Para o desenvolvimento deste projeto foi utilizado a engine Unity, assim como as ferramentas da mesma e sua linguagem de programação C# (Cê Sharp), o que proporcionou uma maior desenvoltura na progessão do projeto.

Com isso foi produzido testes para a desenvoltura e polimento de erros do jogos, que foram destribuídos para o público em janeiro de 2024.

Como trabalho futuro, pretende-se desenvolver outras mecânicas, podendo existir outros sistemas, de modo que haja meio de mais de um jogador poderem trabalharem juntos em novas jornadas enfrentando diversos inimigos.

5. REFERÊNCIAS BIBLIOGRÁFICAS

Unity Technologies. Aprenda a criar jogos com Unity. Unity, 2023. Disponível em: https://unity.com/pt. Acesso em: 14 de dezembro de 2023

KRONOVI, Darkpixel. Mecha Golem Free. itch.io, 2023. Disponível em: https://darkpixel-kronovi.itch.io/mecha-golem-free. Acesso em: 27 de janeiro de 2024.

BOY, Atari. Skull Wolf Pixel Art. itch.io, 2023. Disponível em: https://atari-boy.itch.io/skull-wolf-pixel-art. Acesso em: 6 de janeiro de 2023.

ELTHEN. Bat Sprite Pack. itch.io, 2023. Disponível em: https://elthen.itch.io/bat-sprite-pack. Acesso em: 20 de dezembro de 2023.

BIGGERMANJD. Platformer Tileset Pixelart Grasslands. itch.io, 2023. Disponível em: https://biggermanjd.itch.io/platformer-tileset-pixelart-grasslands. Acesso em: 3 de janeiro de 2023.

ELTHEN. Pixel Art Adventurer Sprites. itch.io, 2023. Disponível em: https://elthen.itch.io/pixel-art-adventurer-sprites. Acesso em: 14 de dezembro de 2023.

CAINOS. Pixel Art Platformer Village Props. itch.io, 2023. Disponível em: https://cainos.itch.io/pixel-art-platformer-village-props. Acesso em: 13 de janeiro de 2023.

XDEVIRUCHI. 8-Bit Fantasy & Adventure Music Pack. itch.io, 2023. Disponível em: https://xdeviruchi.itch.io/8-bit-fantasy-adventure-music-pack. Acesso em: 14 de dezembro de 2023.