

Desenvolvimento de jogo 2D com C# e Unity

Pedro Denardi Minuzzi

Ciência da computação – Universidade Regional do Alto Uruguai e das Missões (URI)
Jaguari – RS – Brasil

pedrominuzzi21@gmail.com

Abstract. This article discusses the development of 2D game using the C# programming language and the Unity framework. The game includes the choice of characters with special abilities, weapons and upgrades, phases with different scenarios and enemies, a soundtrack system to help define the mood and emotion of the phase. The game will be developed using Unity, a popular and complete framework for game development.

Resumo. Este artigo aborda o desenvolvimento do jogo em 2D utilizando a linguagem de programação C# e o framework Unity. O jogo inclui a escolha de personagens com habilidades especiais, armas e upgrades, fases com cenários e inimigos distintos, um sistema de trilha sonora para ajudar a definir o clima e a emoção da fase. O jogo será desenvolvido através da Unity, um framework popular e completo para desenvolvimento de jogos 2D e 3D.

1. Introdução

O desenvolvimento de jogos em 2D tem se tornado cada vez mais popular entre os desenvolvedores de jogos. Com o uso de tecnologias avançadas, é possível criar jogos com gráficos impressionantes e jogabilidade envolvente. Neste artigo, iremos aprofundar no desenvolvimento de um jogo em 2D, utilizando a linguagem de programação C# e o framework Unity.

2. Storyboard

O jogo em desenvolvimento será composto por um personagem principal que percorrerá o mapa em busca de “loots”, melhorias e novos desafios. Para cada cenário, um inimigo diferente é desenvolvimento, com características peculiares. Também em conjunto, cada inimigo disponibiliza de uma fraqueza ou resistência.



Figura 1. Cenários da fase inicial do jogo

O jogo será dividido em 3 fases, cada uma com um cenário diferente. As fases serão compostas por inimigos distintos, cada um com sua característica de fraqueza ou resistência. O herói terá que derrotar os inimigos para chegar até o chefe e então o mata-lo finalizar o jogo.

3. Menu

Ao iniciar o jogo, o jogador será recebido por um menu inicial. Nessa seção, o jogador encontrará uma lista de personagens disponíveis para escolher. Cada personagem terá suas próprias habilidades, atributos e estilo de jogo único.

Também estará disponível ao lado de seleção dos personagens, os controles do jogo, contendo a descrição e qual botão do teclado/mouse deve ser pressionado.



Figura 2. Menu inicial do jogo

4. Elementos do Personagem

O personagem principal disponibilida características padrões de físicas. Ele tem a habilidade de pular, andar, agachar, atacar, mudar de direção no mapa, receber dano e morrer. Além disso, contém uma barra de vida que pode acumular maior quantidade de vida ao decorrer do jogo, sendo mais difícil a morte do personagem.

4.1. Animações

As animações do personagem foram criadas para trazer vida e fluidez aos seus movimentos. Utilizando o Animator Controller e a funcionalidade Animation, cada animação foi desenvolvida para representar os diferentes estados do personagem.

A primeira animação desenvolvida foi a "Parado" do personagem, na qual a cabeça dele se movimenta para cima e para baixo, criando o efeito de respiração, dando mais vida ao personagem.

A animação de "Andar", transmite a sensação de movimento do personagem enquanto ele percorre o cenário. Com transições e uma sequência de frames definida, essa animação dá a impressão de que o personagem está se deslocando de forma natural.

A animação de "Pular" é responsável por adicionar um elemento de dinamismo ao personagem. Ela mostra o momento em que ele se impulsiona para o ar. Essa animação dá a sensação de gravidade e força ao personagem.

Ao pressionar a tecla de "Agachar", o personagem realiza uma animação correspondente a essa ação. Seus movimentos são adaptados para uma posição mais baixa, com os joelhos dobrados.

Quando o personagem é atingido por um ataque, a animação de "Hit" entra em ação. Ela representa o momento em que o personagem é impactado, exibindo reações de dor. Essa animação inclui movimentos de recuo, adicionando realismo às interações de combate.

A animação de "Morrer" é ativada quando o personagem chega ao seu fim. Ela mostra o personagem recebendo o hit e em sequencia, caindo no chão, indicando que ele não possui mais vida.

Por último, temos a animação de "Atacar". Ela define o comportamento do personagem ao realizar um golpe. Ela envolve 3 movimentos, cima, meio e baixo. Essa animação foi configurada para ser reproduzida em uma sequência de 3 estágios num determinado intervalo de tempo. Dependendo da arma, esse intervalo pode ser maior ou não para dar uma expressão de que a arma é mais pesado ou leve que outra.

Além do controle das animações por meio do Animator Controller e da criação das animações com a Animation, existe o objeto "groundCheck". Esse objeto desempenha um papel fundamental na detecção se o personagem está ou não no chão em tempo de física. Essa informação é essencial para determinar se o personagem pode ou não pular, cair ou executar outras ações específicas.

4.2. Golpes

Os golpes do personagem são compostos por três componentes distintos na Unity, cada um representando uma animação de ataque da arma em diferentes direções: cima, meio e baixo. Essa abordagem permite que o personagem execute golpe que vai conduzir o ataque em um angulo de 180 graus, então se tiver algo em cima ou embaixo quando acontecer a animação, o golpe vai acertar.

Para tornar esses golpes interativos, foi utilizado um CircleCollider2D na ponta da arma, com a função trigger habilitada. Essa configuração permite que o collider da arma não colida fisicamente com outros objetos do cenário, mas ainda é capaz de detectar e retornar informações sobre colisões por meio de triggers.

Ao ativar o trigger da arma ao entrar em contato com um inimigo, é possível capturar essa informação e desencadear a ação de dano. Através da captura desse trigger gerado pela colisão da arma com o inimigo, é executado um script do inimigo que reduz a saúde e aciona animações de reação ao impacto. Quando o trigger é acionado e o inimigo fica sem vida, aciona a animação de morte e retorna loots para o personagem coletar.

4.3. Vida

Durante o decorrer do jogo, o personagem pode sofrer danos, o que resultará na redução da sua vida. No entanto, existem também meios de recuperar e aumentar a vida para enfrentar os inimigos adiante.

Quando o personagem sofre danos, sua vida diminui em uma quantidade determinada ou morre instantaneamente, seja por ataques de inimigos, armadilhas ou se cair fora do cenário.

Para recuperar a vida perdida, o jogador deve coletar loots de baús espalhados pelo mapa. Esses baús contém frascos de vida que, ao ser utilizado, recupera uma porção de vida do personagem.

Se o jogador ficar sem vida, ele automaticamente morre, iniciando o menu de Game Over, com as opções de sair do jogo ou reiniciar a fase.



Figura 3. HUD de vida do personagem

4.4. Mana

A mana é um recurso essencial para o personagem mago, permitindo que ele utilize habilidades mágicas e ataques especiais durante o combate. A seguir, mais detalhes sobre o sistema de mana e sua recuperação através da coleta de loots de baús.

A mana é a energia mágica que o mago utiliza para conjurar feitiços, lançar magias e executar ataques mágicos. Cada ataque consumirá uma quantidade específica de mana, geralmente 1 unidade por ataque.

O jogador deve estar atento ao seu nível de mana durante as batalhas. Quando a mana atinge o valor zero, o mago não poderá mais utilizar habilidades mágicas até que sua mana seja restaurada.

Para recuperar a mana esgotada, o jogador deve coletar loots de baús. Esses baús podem conter itens específicos que restauram a mana do personagem mago. Por exemplo, um baú pode conter uma poção de mana que, ao ser utilizada, restaura uma certa quantidade de mana ao mago.



Figura 4. HUD de mana do personagem

4.5. Classes de armas

As armas disponíveis são divididas em três tipos: Machado, Arco (flecha) e Cajado (magia). Cada tipo de arma possui três níveis: normal, avançado e elite. Cada nível de arma aumenta tanto o dano mínimo quanto o dano máximo que ela causa.

O Machado é um tipo de arma que não possui variações de tipos ou atributos além do aumento de dano proporcionado pelos diferentes níveis. Os três níveis de machados, normal, avançado e elite, aumentam gradualmente o dano que o machado pode causar.

O Arco e flecha, por sua vez, possuem diferentes tipos de flechas. São eles: normal, avançado e elite. Cada tipo de flecha aumenta não apenas o dano máximo e mínimo, mas também a velocidade das flechas disparadas pelo arco.

Por fim, temos o Cajado de magia. Assim como o arco e flecha, o cajado de magia também possui três tipos: normal, avançado e elite. Cada tipo de cajado aumenta dano máximo e mínimo dos ataques mágicos.

4.6. Inventário

O jogo possui um inventário que permite ao jogador visualizar, selecionar, equipar, fazer upgrade e descartar itens. Nesse inventário, além das armas mencionadas anteriormente, é possível encontrar outros itens e recursos relevantes para o jogo.



Figura 4. Inventário do personagem

Ao selecionar um item no inventário, o jogador pode equipá-lo para uso imediato pelo personagem. O upgrade de itens no inventário permite melhorar suas características, o máximo de upgrades possíveis em uma arma são cinco, com o custo de duzentas moedas para cada upgrade. Para cada upgrade, uma característica é melhorada,

como dano da arma ou velocidade da flecha. Também, é possível descartar um item do inventário, removendo-o permanentemente da posse do personagem.



Figura 5. Informações da arma

Por fim, o item equipado, não é possível remover, caso contrario, o personagem ficaria sem arma, impedindo de prosseguir no jogo.

5. Cenários

As fases do jogo serão divididas em cenários diferentes, cada um com seus próprios ambientes e inimigos. Os cenários incluem uma floresta com um clima agradável, logo em seguida, uma ponte que ao passar por ela, tem acesso a escada que leva a mina e por fim, ao sair da mina, o personagem entra no cenário de gelo. Cada cenário será único e terá seus próprios inimigos.

Para criar os cenários, foi utilizado o Tile Palettes. Com essa técnica, é possível criar padrões de tiles e objetos que representam os cenários do ambiente e outros elementos. O uso de diferentes camadas de tiles e objetos permite a criação de um ambiente tridimensional, dando a sensação de profundidade e imersão.

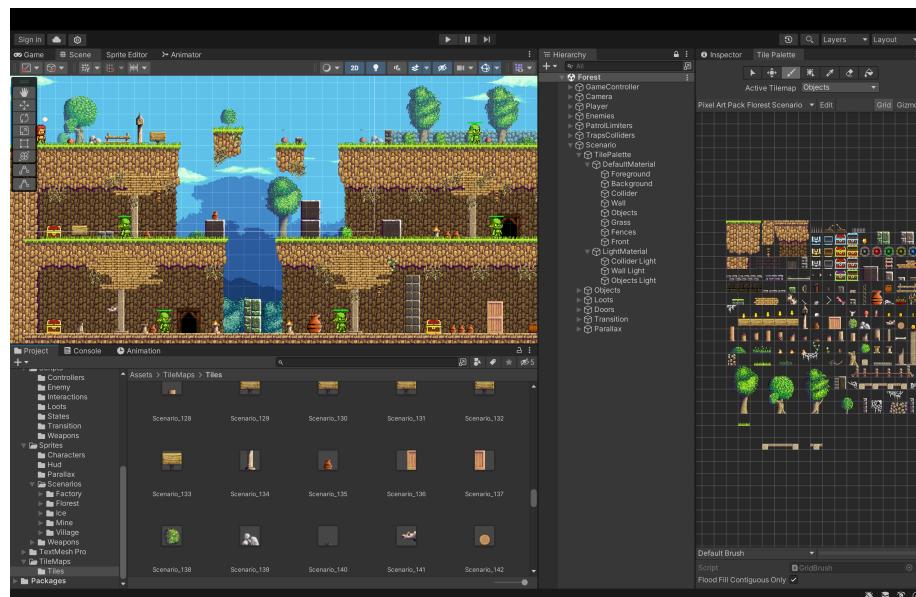


Figura 6. Desenvolvimento do mapa através do Tile Palettes

Para a criação do ambiente tridimensional, foi utilizado a técnica de Layer Sorting, que também desempenha um papel importante na criação dos ambientes, garantindo que os elementos do cenário estejam corretamente ordenados em relação aos personagens e inimigos, para evitar sobreposições visuais inadequadas.

6. Inimigos

Os inimigos do jogo possuem características distintas e interações interessantes com o jogador. Cada inimigo terá uma fraqueza e resistência específicas, bem como uma quantidade de dano que causa ao acertar o jogador ou uma quantidade de dano recebido do jogador equivalente a arma e ou efeito que utiliza na arma. Além disso, quando o jogador conseguir derrotar um inimigo, ele receberá recursos ou vida, permitindo que prossiga em sua jornada e faça melhorias e upgrades.

É interessante destacar também que os inimigos possuem personalidades semelhantes às do jogador, mas com comportamentos limitados a animações de respiração, andar, mudar de direção e pular. Essas animações ajudam a dar vida aos inimigos e torná-los mais realistas em seu comportamento.

Para dar ênfase a um maior desafio ao enfrentar um inimigo, foi adicionado uma inteligência artificial para que os inimigos possam detectar e identificar o jogador, iniciando uma perseguição ativa.

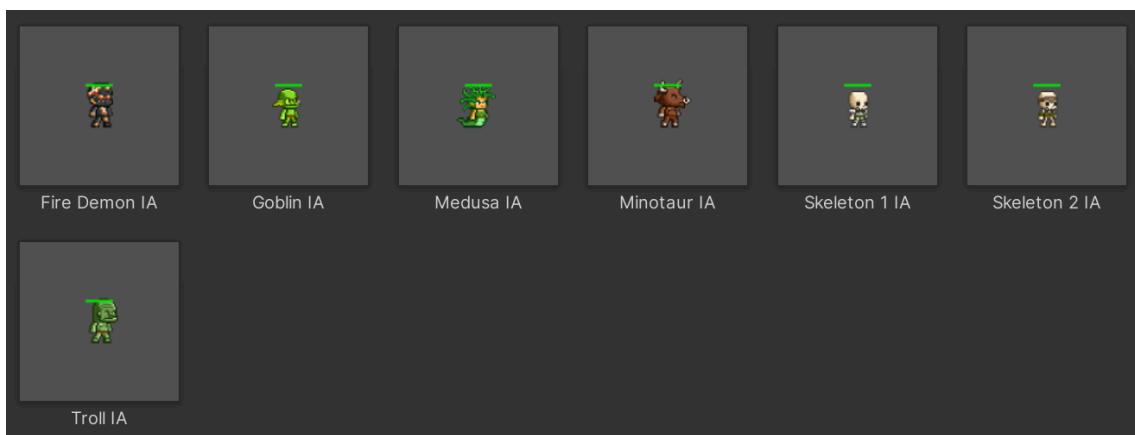


Figura 7. Inimigos configurados com IA

Além disso, os recursos podem ser usados para melhorias e upgrades do personagem.

Por fim, cada inimigo possuí um dos tipos de fraqueza ou resistência, entre elas, estão disponíveis: gelo, fogo e magia.

7. Inteligência Artificial

A inteligência artificial (IA) dos inimigos desempenha um papel crucial para criar desafios e interações interessantes com os jogadores. As ações que os inimigos podem

realizar com base nos comportamentos definidos são: Atacando, patrulha, parado, alerta, recuar.

Quando os inimigos estão em modo de ataque, eles procuram se aproximar do jogador e realizar uma série de ações agressivas. Isso pode incluir atacar diretamente com armas, lançar magias ou usar habilidades especiais. A IA dos inimigos avalia a distância e o estado do jogador para decidir a melhor abordagem de ataque.

Alguns inimigos podem ser designados para patrulhar áreas específicas. Nesse comportamento, eles seguem um trajeto pré-definido ou percorrem uma área delimitada. Esses inimigos estão atentos a qualquer atividade suspeita e podem entrar em alerta caso avistem o jogador ou identifiquem algum sinal de perigo.

Em certas situações, inimigos podem ficar em modo de espera, geralmente protegendo um ponto-chave ou aguardando a aproximação do jogador. Eles permanecem imóveis até que sejam provocados ou detectem uma ameaça. Essa postura pode criar oportunidades estratégicas para o jogador, como realizar ataques furtivos ou evitar conflitos desnecessários.

Quando os inimigos entram em estado de alerta, eles são conscientes da presença do jogador e agem com maior vigilância. Nesse momento, eles podem iniciar uma ação defensiva ou ofensiva.

Por fim, alguns inimigos podem recuar após um ataque, se distanciando do jogador e entrando em estado de alerta para atacar novamente caso o jogador se aproxime.

8. Cinemachine

O Cinemachine é uma ferramenta poderosa oferecida pela Unity, que permitiu criar cinematografia e controle de câmera de forma intuitiva e flexível. Ele permitiu controle total sobre a posição, rotação e composição da câmera em tempo real, para criar movimentos suaves, transições e alterações de ângulo de forma fácil e intuitiva.

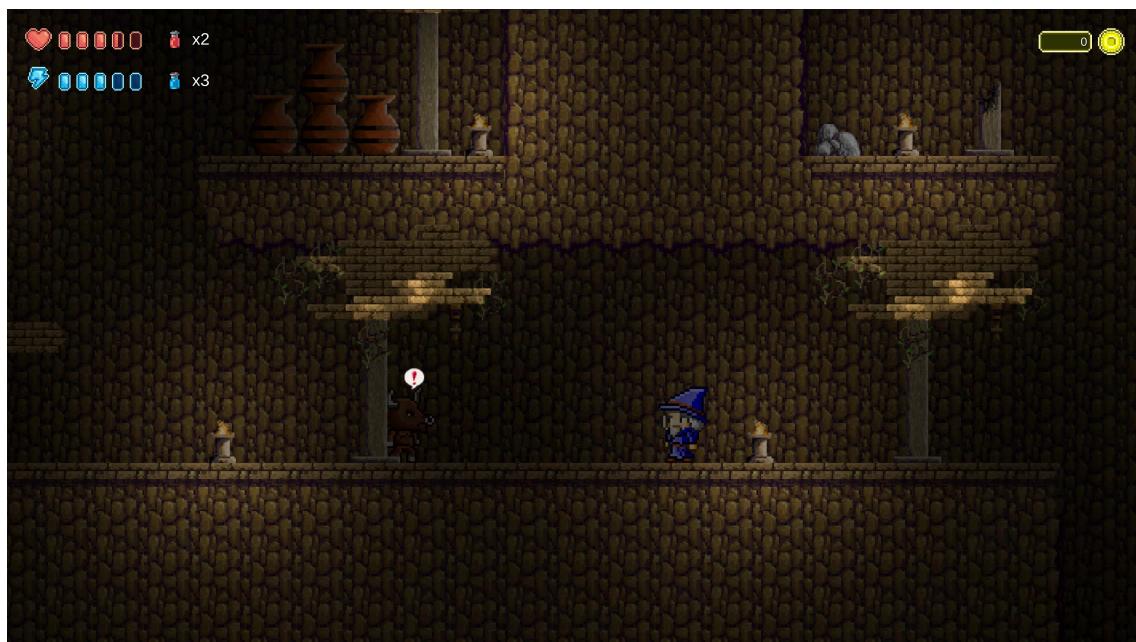


Figura 8. Controle de câmera conforme o personagem se movimenta

9. Parallax

Para dar mais vida ao ambiente, a imagem de fundo do jogo foi dividido em quatro partes, entre elas estão: Nuvens, montanhas, morros e terreno.

Essa divisão foi crucial para criar o efeito de que quando o personagem está se movimentando, o cenário está percorrendo. Para isso, desenvolveu-se um script para controlar para imagem.

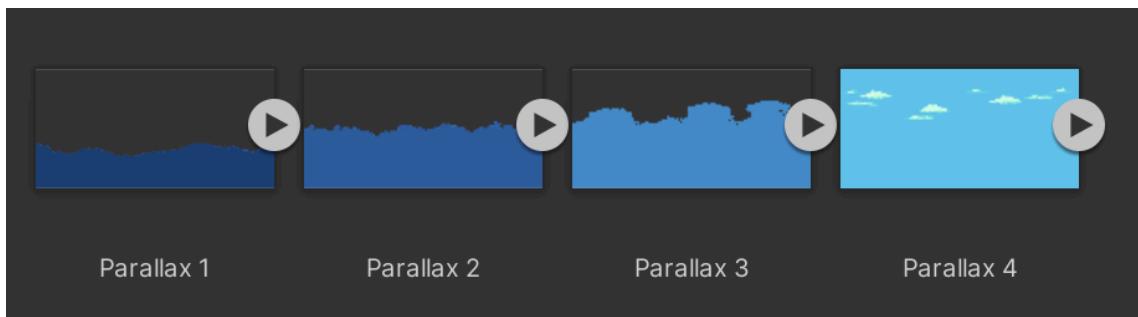


Figura 9. Divisão da imagem de fundo em 4 imagens

Para cada imagem, é alterado o Axis X da imagem multiplicado pela escala da imagem, com isso, a imagem de maior profundidade, as nuvens, são as que menos tem influencia no movimento do cenário, em sequencia as montanhas com um aumento pequeno na velocidade de movimentação e por fim morros e terrenos, que tem uma diferença pequena de movimentação na imagem, para causar o efeito Parallax ao jogo.

10. Boss

O boss foi projetado com base em rotinas programadas para adicionar variedade e imprevisibilidade às suas ações. Cada rotina possui uma porcentagem de chance de ser executada, garantindo que os movimentos não sejam sempre os mesmos.



Figura 10. Sprites do Boss

Essa abordagem é uma maneira inteligente de criar um desafio dinâmico para os jogadores, mantendo-os engajados e surpreendidos durante a batalha. Ao introduzir uma variedade de rotinas com diferentes comportamentos e ataques, o Boss se torna mais imprevisível e exige que os jogadores estejam atentos e se adaptem rapidamente.

11. Trilha Sonora

A trilha sonora do jogo será composta por músicas instrumentais que combinam com o gênero do jogo. Cada fase terá sua própria música de fundo, que ajudará a definir o clima da fase. A música também será usada para indicar quando o jogador alertar um inimigo ou entrar no cenário de Boss.

Também terá efeitos sonoros quando ocorre um hit em inimigo, pulo, ataque e demais animações ou eventos no cenário.

12. Linguagem de Programação e Framework

A Unity é uma poderosa engine de desenvolvimento de jogos que oferece suporte tanto para criação de jogos em 2D quanto em 3D. No contexto do jogo desenvolvido, foi utilizado a plataforma em 2D, utilizando diversas funcionalidades da Unity para criar uma experiência imersiva e interativa.

O C# é a linguagem de programação primária usada na Unity e permite criar a lógica e interações do jogo, com a manipulação através da IDEA do Visual Studio Code e alguns plugins instalados para compilação de auto-correção.

Dentre os recursos disponíveis da Unity, foram utilizados Assets, Components, Physics, Layers e Animation.

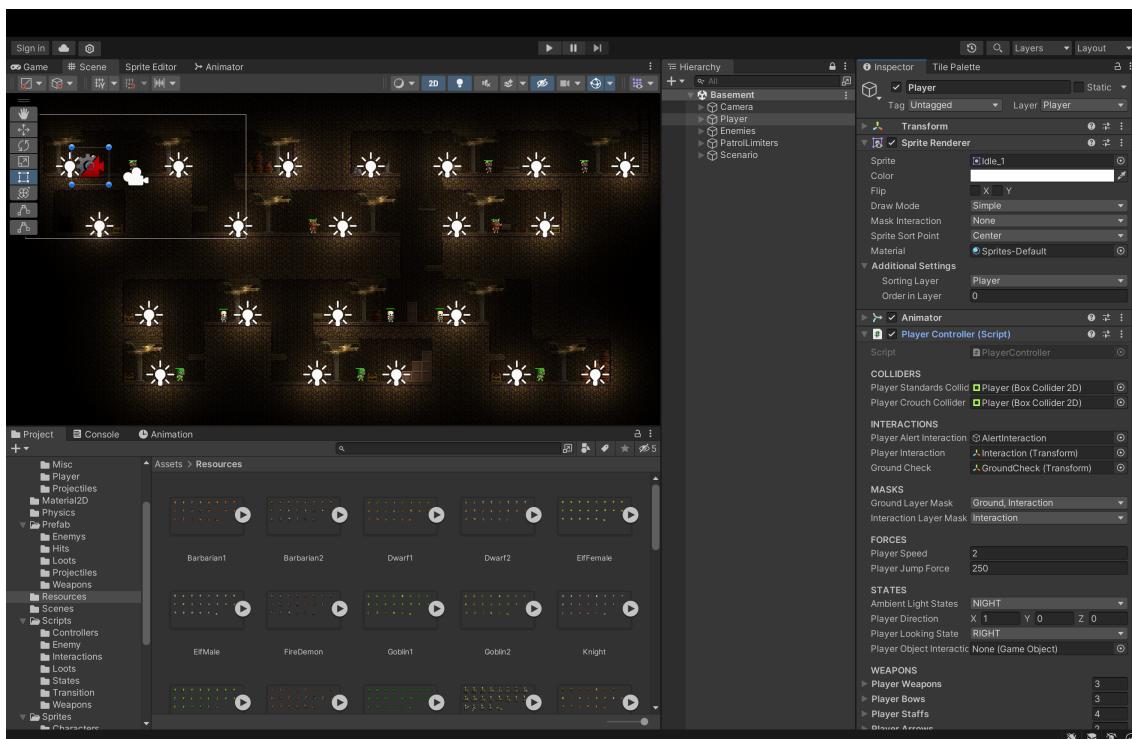


Figura 11. IDEA da Unity

12.1. Assets

Os assets são os recursos usados para construir o jogo, como gráficos, modelos, texturas, áudio e animações. Na criação do jogo em 2D, foram importados spritesheets, que são folhas de sprites contendo várias imagens de um personagem ou cenário em diferentes poses e estados. Alguns spritesheets foram desmembrados em sprites individuais para ser usados em animações ou renderização objetos no jogo

12.2. Components

Os components são os blocos de construção das entidades do jogo. Eles são adicionados aos objetos no Unity para dar a eles funcionalidades específicas. No contexto do jogo em 2D, foi utilizado o Rigidbody2D para adicionar física aos objetos, permitindo que eles interajam com o ambiente. O Collider2D é usado para definir a forma e a área de colisão dos objetos, possibilitando detecção de colisão e resposta física adequada.

12.3. Physics

Foi utilizado o Physics da Unity para simular o comportamento realista de objetos no jogo. O sistema de física 2D da Unity permitiu definir propriedades como gravidade, forças aplicadas, detecção de colisão e resposta física, como rebatimento, deslizamento e rotação de objetos.

12.4. Layers

As layers são usadas para organizar e controlar a renderização e a colisão de objetos no jogo. Elas permitem agrupar objetos relacionados e definir como eles interagem entre si. O jogo está disposto por 6 layers: Ground, Interaction, Attack, Enemy, Player e Loots.

O layer "Ground" é usado para definir a camada onde o chão, as plataformas e outros elementos estáticos do ambiente do jogo estão localizados. Essa camada é fundamental para a detecção de colisões e para garantir que o jogador e outros objetos interajam adequadamente com o chão.

O layer "Interaction" é utilizado para objetos interativos no jogo, como alavancas, interruptores, portas e outros elementos que podem ser ativados ou manipulados pelo jogador. Esse layer permite a interação entre o jogador e esses objetos, seja para abrir caminhos, desbloquear áreas ou acionar eventos específicos.

O layer "Attack" é usado para detectar colisões relacionadas a ataques ou habilidades dos personagens, tanto do jogador quanto dos inimigos. Isso permite interações específicas quando um ataque ou habilidade colidir com outros objetos, como causar dano, acionar animações ou ativar efeitos visuais.

O layer "Enemy" é utilizado para definir os inimigos no jogo. Ele permite a configuração e detecção de colisão e interação entre o jogador e os inimigos, possibilitando o combate, a aplicação de dano e outras mecânicas relacionadas ao confronto com os oponentes.

O layer "Player" é destinado ao personagem controlado pelo jogador. Ele é fundamental para definir a interação do jogador com o ambiente, detectar colisões, permitir movimentação fluida e aplicar lógica específica ao jogador, como coleta de itens e interações.

Por ultimo, o layer "Loots" é utilizado para objetos que representam recompensas ou itens coletáveis no jogo, como baús, moedas, power-ups e armas.

12.5. Animation

Por fim, a Unity oferece uma poderosa ferramenta de animação que permite criar animações complexas para os personagens e objetos do jogo. Através dessa ferramenta, foi disposto todas as animações de movimento, transições entre estados, efeitos visuais e muito mais. Com a utilização dos sprites importados de spritesheets, foi possível criar animações fluídas e expressivas para dar vida aos personagens e elementos do jogo.

12.6. SpriteRenderer

Componente que permite exibir imagens ou sprites na cena do jogo. Ele é responsável por renderizar e exibir visualmente os elementos 2D, como personagens, objetos e cenários. O SpriteRenderer também oferece recursos para ajustar a cor, transparência e camadas de renderização dos sprites.

12.7. Animator

Componente que permite criar e gerenciar animações. Ele permite a transição suave entre diferentes estados de animação, como andar, pular, atacar, entre outros. O Animator pode ser usado em conjunto com uma máquina de estados para controlar o fluxo das animações com base em certas condições ou eventos.

12.8. BoxCollider2D

Componente que define uma área retangular de colisão em torno de um objeto. Ele é usado para detectar colisões entre objetos no jogo. Quando dois objetos com BoxColliders se sobreponem, uma detecção de colisão é ativada, permitindo ações específicas em resposta a essa colisão, por exemplo, coletar itens no cenário.

12.9. TilePalette

Permite criar e organizar conjuntos de tiles, que podem ser facilmente pintados no cenário do jogo para criar níveis e ambientes. O TilePalette ajudou a agilizar o processo de construção do mundo do jogo, permitindo selecionar e aplicar rapidamente tiles predefinidos.

12.9. Material2D

Recurso que controla as propriedades visuais de um objeto 2D, como cor, textura, brilho e sombra. Ele define como a luz e as texturas interagem com o objeto, determinando seu aspecto visual. O Material2D é aplicado aos sprites ou tiles através do SpriteRenderer, influenciando a sensibilidade à luz dos objetos dentro do cenário.

13. Conclusão

O desenvolvimento de jogos em 2D é uma jornada difícil que envolve a combinação de criatividade, habilidades de programação e o uso de ferramentas poderosas, como a Unity e a linguagem de programação C#. Ao longo do processo de desenvolvimento, é importante considerar os seguintes pontos:

- Planejamento e Design: Antes de começar a escrever código, é essencial ter um plano sólido e um design bem definido para o jogo. Isso inclui a definição dos elementos do jogo, como personagens, cenários, mecânicas e interações. Com a criação dos storyboard, foi possível ter uma visão de como o jogo seria feito.
- Assets e Gráficos: Utilizar ferramentas como spritesheets e Tile Palettes para organizar e otimizar os recursos gráficos, tornando o desenvolvimento mais eficiente.
- Componentes e Funcionalidades: Foi aproveitado os recursos como Rigidbody2D, Collider2D e o sistema de física para criar movimento e interações realistas. Esses componentes foram combinados e personalizados conforme as necessidades do jogo.
- Lógica do Jogo: A implementação da lógica do jogo foi feita através de scripts em C# para controlar as interações entre personagens, inimigos, objetos e elementos do ambiente. Além desses, sistemas como IA, detecção de colisões, pontuação, vidas e outros aspectos fundamentais da jogabilidade.
- Testes e Ajustes: Por fim, no decorrer do jogo houve vários problemas relacionados a bugs e cenários não condizentes com a física do personagem, precisando adequar novamente ou até mesmo recriar.

Sendo assim, com dedicação, criatividade e um sólido conhecimento da Unity e do C#, é capaz criar um jogo 2D envolvente e cativante, capaz de proporcionar uma experiência única aos jogadores.

14. Referencias

Gabriel Sousa. “Desenvolvimento de games 2D mobile com Unity”, <https://www.udemy.com/course/desenvolvimento-de-games-2d-mobile-com-unity/learn/lecture/27444102?start=105#overview>, 26 de Abril de 2023.

Willian Nascimento. “Jogos 3D com Unity + modo multiplayer”, <https://www.udemy.com/course/jogos-3d-com-unity-2017-modo-multiplayer/learn/lecture/7941722?start=15#overview>, 21 de Abril de 2023.

Patrick Muniz. “Aprenda Unity - 2D Definitivo”, <https://www.udemy.com/course/2d-definitivo-unity-2017/learn/lecture/14541120#overview>, 16 de Abril de 2023.