Documentação do Projeto

Phantom Dodge

Desenvolvido por: Luiz Henrique Vieira Frandoloso (182354).

Jogo 2D de Desvio de Obstáculos

Descrição do Projeto

Phantom Dodge é um jogo 2D simples onde o jogador controla um personagem que deve desviar de obstáculos gerados automaticamente. O objetivo principal é sobreviver o máximo de tempo possível enquanto coleta pontos e tenta sempre superar sua última pontuação máxima. A dificuldade aumenta gradualmente com o tempo, tornando o jogo mais desafiador.

Arquitetura do Projeto

Estrutura Geral

O projeto utiliza uma arquitetura baseada em **GameObjects** e scripts interligados no Unity. Cada funcionalidade principal (como spawn de obstáculos, controle do jogador, e pontuação) é modular e separada em componentes, permitindo fácil manutenção e escalabilidade.

Componentes Principais

1. **PlayerController**:
    - Controla o movimento do jogador e a lógica de colisão.
    - Gerencia as vidas do jogador e os eventos relacionados à morte.
2. **Spawner**:
    - Responsável por gerar obstáculos em intervalos de tempo definidos.
    - Escala a dificuldade do jogo reduzindo o tempo entre spawns e aumentando a velocidade global.
3. **Obstacle**:
    - Define o comportamento dos obstáculos gerados, como movimento e interação com o jogador.
4. **ScoreManager**:
    - Gerencia a pontuação atual e a pontuação máxima.
    - Salva a pontuação máxima usando o **PlayerPrefs** para persistência entre sessões.
5. **UIManager**:
    - Exibe a pontuação, vidas restantes e outras informações na interface gráfica.

6. **Destroyer**:
    - Destroi automaticamente objetos que saem da tela para economizar memória.

Estrutura de pastas

- **Animation** -> Animação da Câmera , Obstáculos e Jogador.
- **Prefabs** → Modelos reutilizáveis de GameObjects para manter a consistência.
- **Scenes** → Contém as 2 Cenas do jogo (Menu Inicial e o Jogo)
- **Scripts** → Contém todos os scripts.
- **Sprites** → Contém os sprites (Optei por pegar sprites prontos)

Arquitetura do Jogo

- **Câmera e Mundo 2D**:
    - O jogo é projetado em uma perspectiva 2D com uma câmera ortográfica.
    - Os objetos (jogador, obstáculos, fundo) se movem horizontalmente, criando a sensação de progressão infinita.
- **Divisão em Prefabs**:
    - **Player**: Inclui o sprite, colisor e o script de controle.
    - **Obstáculos**: Prefabs com sprites variados e scripts que controlam o movimento e a colisão.
    - **Spawner**: Um GameObject fixo na cena que instancia os obstáculos dinamicamente.

Scripts Desenvolvidos

1. Player

- **Responsabilidades**:
    - Controlar o movimento vertical do jogador.
    - Gerenciar as vidas e exibir visualmente o status no HUD.
    - Detectar colisões e disparar eventos, como a morte do jogador.
- **Métodos Principais**:
    - **`Update`**: Controla o movimento e verifica as teclas pressionadas.
    - **`OnTriggerEnter2D`**: Detecta colisões com obstáculos.

2. Spawner

- **Responsabilidades**:
    - Gerar obstáculos em intervalos definidos.
    - Ajustar dinamicamente a frequência de spawns e a dificuldade.
- **Métodos Principais**:
    - **`Update`**: Verifica o tempo para o próximo spawn e ajusta a dificuldade.
    - **`Start`**: Inicializa o tempo entre os spawns.

3. Obstacle

- **Responsabilidades**:
  - Controlar o movimento dos obstáculos.
  - Detectar colisões com o jogador e aplicar efeitos (dano, explosão).
- **Métodos Principais**:
  - `Update`: Move os obstáculos horizontalmente.
  - `OnTriggerEnter2D`: Trata colisões com o jogador.

4. Score

- **Responsabilidades**:
  - Gerenciar a pontuação do jogador.
  - Salvar e carregar a pontuação máxima usando **PlayerPrefs**.
- **Métodos Principais**:
  - `OnTriggerEnter2D`: Incrementa a pontuação e atualiza a pontuação máxima.
  - `Start`: Carrega a pontuação máxima salva.

5. Destroyer

- **Responsabilidades**:
  - Destruir automaticamente objetos que saem da tela.
- **Métodos Principais**:
  - `Start`: Agendar a destruição do objeto com base em seu tempo de vida.

Ferramentas e Tecnologias

- **Engine**: Unity 2021+
- **Linguagem de Programação**: C#
- **Sistema de Física**: Unity Physics 2D
- **Gerenciamento de Cena**: Unity SceneManager
- **Persistência de Dados**: Unity PlayerPrefs (para pontuação máxima)

Processo de Desenvolvimento

1. Planejamento

- Escolher a mecânica principal: desvio de obstáculos com aumento de dificuldade.
- Definir os recursos principais: jogador, obstáculos, spawn e pontuação.

2. Desenvolvimento Modular

- Cada funcionalidade foi separada em scripts:
  - Movimento do jogador.
  - Geração de obstáculos.

    - Exibição de pontuação e vidas.
- Uso de **Prefabs** para reutilização de objetos.

3. Observações

- A música de fundo eu resolvi não adicionar.
- Adicionei apenas os seguintes efeitos sonoros.
    - Movimentação
    - Colisão
    - Morrer

Conclusão

Phantom Dodge foi desenvolvido com foco na modularidade e na escalabilidade. A mecânica principal é simples, com elementos visuais e de interface integrados. O uso de variáveis dinâmicas para ajustar a dificuldade e salvar a pontuação máxima garante uma experiência mais clara e atrativa para o jogador.