Oktapong Documentação

Projeto desenvolvido por: André Felipe dos Santos
Documento escrito por: André Felipe dos Santos

Sumário:

Introdução:

Este projeto foi desenvolvido durante o processo seletivo para ingressar na Oktagon Games.

O projeto é um jogo de turnos, onde a cada turno, um dos jogadores tenta disparar um projétil contra um adversário.

Os projéteis disparados precisam ricochetear nas paredes, para que acertem um dos alvos.

Ganha o jogador que zerar os pontos de vida do adversário.

Controles:

Os jogadores utilizam as teclas direcionais cima e baixo para movimentar o seu avatar verticalmente, e as teclas direcionais esquerda e direita para mudar o ângulo de disparo.

Projéteis:

Existem 4 projéteis já criados, sendo eles:

Ricochete - Maior número de vezes que pode ricochetear antes de ser destruído.
Heavy Bullet - Dano alto, mas lento.
Speedster - Dano baixo, mas rápido.
Killer Queen - Dano altíssimo, porém, pode causar dano ao jogador que o disparou também.

Após um jogador disparar um projétil, todos os jogadores ficam imóveis e incapazes de atirar, enquanto este projétil não for destruído (seja ricocheteando ou colidindo com outro player).

Isso reforça a lógica de turno, onde, após ser atingido, o jogador tem duas escolhas:

- Permanecer na mesma posição e dar ao adversário a chance de efetuar um disparo mais preciso
- ou mudar de posição, para evadir do próximo tiro, mas precisando ajustar novamente a mira para acertar o adversário.

Frameworks e Softwares utilizados:

Unity Version : 2019.4.1f1
GameEngine

Visual Studio Version: 16.6.6
IDE de desenvolvimento

Unity Api Compatibility Level: .NetStandard 2.0
.NET api utilizada pelo projeto na Unity

Extenjent/Zenject: 9.2.0
Framework de injeção de dependência para Unity3D, com o package importado localizado na pasta localizado na pasta ImportedPackages.

ThomasKomarnicki/GameEventBus: 9.2.0
Sistema de eventos para Unity3D, com o package importado localizado na pasta ImportedPackages.

Estrutura do projeto:

Ao entrar na pasta de assets, nos deparamos com as seguintes pastas:

Images - Armazenam imagens (textures, sprites) do projeto
Plugins - Armazena os frameworks utilizados (Zenject e EventBus)
Prefabs - Armazena todos os prefabs do projeto.
Scenes - Armazena as duas scenes (Main menu e GameScene) do projeto
Scriptables - Dividida em Bullet e Physics, armazena os scriptableObjects que contém informações dos projéteis e das propriedades físicas dos objetos.
Scripts - Dividida em:

EventSystem - Armazena os scripts dos eventos utilizados pelo EventBus

GameLogic - Armazena os scripts de comportamentos de jogo (mover o personagem verticalmente, mirar, efetuar disparo, gerenciar status do personagem) e de HUD.

ManagementScripts - Armazena os scripts de gerenciamento GameManager e HUDManager.

PhysicsSystem - Armazena os scripts referentes a física customizada.

Scriptables - Armazena os scripts que geram os scriptables.

ZenjectInstallers - Armazena os installers (scripts com o binding das classes) do injetor de dependência.

OBS: Diferente dos demais prefabs, o prefab “Bullet” não deve ser colocado no hierarchy diretamente. Ele é utilizado pelo Bullet Factory para instanciar as bullets dentro do Pool de bullets.

ScriptableObjects

Projéteis:

MaxBounces - Quantidade máxima de ricochetes que a bala pode fazer antes de ser destruída.

BulletSpeed - Velocidade de movimento da projetil (valores muito altos podem fazer o projétil atravessar objetos sólidos).

Damage - Dano infligido pelo projétil no alvo.

CanDamageShooter - Define se o projétil pode danificar o próprio atirador (nos casos em que o projétil atinge quem a atirou).

BulletColor - Cor do projétil ao ser disparado.

CustomPhysicsProperties - armazena as informações básicas desta bala.

Propriedades físicas:

isStatic - Define se este objeto se move. Caso não se mova, testes de colisão são feitos apenas para os casos onde outros objetos colidem com ele.

CanBounce - Caso seja verdadeiro, ao colidir com outro objeto, este objeto será impulsionado na direção oposta ao da colisão.

StopOnCollide - Caso seja verdadeiro, ao colidir com outro objeto, este objeto ficará parado na posição onde colidiu não será impulsionado.

Configuração do projeto

Esta seção visa apresentar um passo a passo de como configurar uma cena neste projeto.

Para informações de como configurar o Zenject ou o EventBus, por favor consultar:

https://github.com/modesttree/Zenject
https://medium.com/@tkomarnicki/messaging-architecture-in-unity-6e6409bdda02

respectivamente.

1 - Criar uma cena vazia.
2 - Inserir duas instâncias do prefab “Player” localizado na pasta de prefabs
    2.1 - Colocar cada prefab do Player na posição e rotação Y desejada
3 - Inserir o Prefab “GameManager” localizado na pasta de prefabs
    3.1 - Arrastar os prefabs dos players para a aba “Players” do GameManager
4 - Inserir o Prefab “(Canvas) HUD” para a cena
    4.1 - Arrastar a “MainCamera” para o campo “Render Camera” do component “Canvas”.
    4.2 - Arrastar o Prefab “GameManager” que está na janela hierarchy para o campo “GM” do componente “HUD Manager” que está no inspector.
    4.3 - Selecionar o objeto “(Canvas) HUD > (Panel)ChooseWeapon > WeaponsGroup” e para cada objeto filho (PlayerNWeaponGroup) atribuir um player ao campo “Player Aim And Shoot”.
    4.4 - Selecionar o objeto “(Canvas) HUD > (Panel)ChooseWeapon > ContinueButton” e atribuir um player para cada elemento do campo “Players Aim And Shoot”.

5 - Inserir o Prefab “SceneContext” para a cena
6 - Criar o Enviroment utilizando o prefab “SquareCustomCollider” ou qualquer objeto com o componente “CustomSquareCollider” e a tag “Wall”.
    6.1 - Também é possível utilizar um dos enviroments prontos da pasta Assets/Prefabs/Enviroments.

Para começar, gostaria de mencionar que caso tenha alguma dúvida sobre algum parâmetro

OBS: Para mudar a orientação dos players no inspector, utilizar a rotação Y. Modificar a escala dos objetos pode ocasionar comportamentos inesperados.

Referências e manuais

Zenject: https://github.com/modesttree/Zenject

EventBus:
https://medium.com/@tkomarnicki/messaging-architecture-in-unity-6e6409bdda02 **e** https://github.com/ThomasKomarnicki/GameEventBus

Colisao entre circulos:
http://flatredball.com/documentation/tutorials/math/circle-collision/

Factory: https://www.youtube.com/watch?v=Gp2_8ihvnvA
e
https://github.com/modesttree/Zenject/blob/master/Documentation/Factories.md

Calculo de Bitmasks: https://www.youtube.com/watch?v=VsmgZmsPV6w