Desenvolvendo "Foguinho e Aguinha" na Unity 3D

Criar, atualizar e personalizar um sumário

O jogo Foguinho e Aguinha é um jogo de plataforma cooperativo onde dois personagens, um de fogo e outro de água, precisam colaborar para superar desafios. Este tutorial guiará você pelo processo de desenvolvimento na Unity 3D (versão 6000), usando C# e a biblioteca padrão da Unity.

# Configuração do Projeto

1. Abra a Unity e crie um novo projeto 2D.
2. No menu Game, configure a resolução para 640x480 (ou outra de sua preferência).
3. Importar Sprites e Assets: Adicione sprites para os personagens, plataformas, portais e botões.

# Configuração do Input System

O novo Input System da Unity é a melhor escolha se você vai ter dois (ou mais) jogadores usando controles diferentes ao mesmo tempo. Se você está usando um controle de Xbox para PC na Unity, os botões e eixos do controle são acessados via Input.GetAxis ou Input.GetButton, não via KeyCode.

🎯 Por que usar o novo Input System?

* 🎮 Suporte real a múltiplos controles (Xbox, PS, teclado, etc).
* 🔀 Cada jogador pode ter seu próprio dispositivo e esquema de input.
* 🧩 Você pode usar PlayerInput + Input Actions para instanciar jogadores com controle atribuído automaticamente.
* 🔌 Detecta dinamicamente quem está controlando quem.
* ✅ Ideal para local multiplayer, coop, jogos de luta, etc.

✅ Recomendado pra você se:

* Está fazendo um jogo multiplayer local com dois personagens.
* Quer que cada um use seu controle sem confusão.
* Quer um sistema limpo e extensível no futuro.

🧩 Estrutura geral

* PlayerInput para detectar o controle e associar ao personagem.
* Input Actions com ações Move (Vector2) e Jump (Button).
* Cada personagem vai usar o mesmo script, mas com PlayerInput configurado individualmente.
* PlayerInputManager vai instanciar automaticamente os jogadores (até 4).

## Instale o Input System

Vamos instalar o pacote na Unity3D.

1. Vá ao Menu, selecione Windows → Via Package Manager → Input System (se ainda não estiver instalado).

Texto

O conteúdo gerado por IA pode estar incorreto.

1. Aceite a reinicialização da Unity.

## Crie o Input Actions

Vamos agora configurar o controle.

1. Crie uma pasta e chama de InputActions
2. Clique direito na pasta e no meno escolha → Create → Input Actions
3. Nomeie como: PlayerControls.inputactions
4. Abra o arquivo clicando em “Edit in Project Setting Windows”

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

1. Configure como demonstra a tabela.
   * Action Map: Gameplay
     + Action: Move
       - Type: Value
       - Control Type: Vector2
       - Bindings:
         * <Gamepad>/leftStick
         * <Keyboard>/a + d
     + Action: Jump
       - Type: Button
       - Bindings:
         * <Gamepad>/buttonSouth (botão A do Xbox)
         * <Keyboard>/space
2. Clique em "Save Asset".

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

# Configuração dos Sprites

1. Abra o sprite e renomeie como informado

Imagem em preto e branco

O conteúdo gerado por IA pode estar incorreto.

Coloque o nome de UI\_Player\_WaterGirl\_Walking\_01 a UI\_Player\_WaterGirl\_Walking\_08

# Criando os Personagens

Vamos criar os dois personagens do jogo

## Criar o Foguinho e a Aguinha

Repita os passos acima para a Aguinha. No Sumário, selecione uma entrada de Nível 1 inteira

1. No Hierarchy, crie um objeto vazio e renomeie como Foguinho.
2. Crie outros dois objetos vazios e coloque o nome de Head e Body
3. Arraste os dois objetos para o gameobject pai Foguinho

Interface gráfica do usuário, Texto

O conteúdo gerado por IA pode estar incorreto.

1. Adicione um Sprite Renderer no Head e Body e atribua a imagem correspondente na qual o personagem fique de frente.

Interface gráfica do usuário, Texto

O conteúdo gerado por IA pode estar incorreto.

1. Adicione um Rigidbody2D (modo "Dynamic") no gameobject Foguinho.
2. Adicione um BoxCollider2D e modifique a colisão para cobrir toda a área do personagem.

Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

1. Repita os passos acima para a Aguinha. Para facilitar faça uma cópia de Foguinho e renomeie o objeto para Aguinha e mude os sprites. Se o corpo ficar muito para baixo, puxe para cima.

Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

## Criando o Script de Movimento e Animação

Os personagens devem responder a controles específicos e ter animações.

No script Player.cs, adicione:

using UnityEngine;

public class Player : MonoBehaviour

{

public float speed = 3f;

public float jumpForce = 5f;

private Rigidbody2D rb;

private bool isGrounded;

public KeyCode leftKey;

public KeyCode rightKey;

public KeyCode jumpKey;

public string characterType;

private Animator animator;

void Start()

{

rb = GetComponent<Rigidbody2D>();

animator = GetComponent<Animator>();

}

void Update()

{

float move = 0;

if (Input.GetKey(leftKey))

{

move = -speed;

animator.SetBool("isWalking", true);

transform.localScale = new Vector3(-1, 1, 1);

}

else if (Input.GetKey(rightKey))

{

move = speed;

animator.SetBool("isWalking", true);

transform.localScale = new Vector3(1, 1, 1);

}

else

{

animator.SetBool("isWalking", false);

}

rb.velocity = new Vector2(move, rb.velocity.y);

if (Input.GetKeyDown(jumpKey) && isGrounded)

{

rb.velocity = new Vector2(rb.velocity.x, jumpForce);

isGrounded = false;

animator.SetTrigger("Jump");

}

}

private void OnCollisionEnter2D(Collision2D collision)

{

if (collision.gameObject.CompareTag("Ground"))

{

isGrounded = true;

}

}

private void OnTriggerEnter2D(Collider2D collision)

{

if ((characterType == "Fogo" && collision.CompareTag("Agua")) ||

(characterType == "Agua" && collision.CompareTag("Fogo")) ||

collision.CompareTag("Goo"))

{

animator.SetTrigger("Dying");

Destroy(gameObject, 0.5f); // Pequeno atraso para mostrar animação de morte

}

}

}

1. Associe o script ao Foguinho e Aguinha.
2. No Inspector, configure leftKey, rightKey e jumpKey para os respectivos controles.
3. Criação das Animações: Adicione um Animator a cada personagem.
4. Crie os seguintes estados no Animator Controller:Idle (Parado), Walking (Andando), Jumping (Pulando), Dying (Morrendo), Configure os triggers e bools de acordo com o script acima.

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Tela de computador com jogo

O conteúdo gerado por IA pode estar incorreto.Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

Uma imagem contendo tela, quarto, colorido, pendurado

O conteúdo gerado por IA pode estar incorreto. Desenho de personagem de desenho animado

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

# Criando o Cenário

Vamos agora criar a sua fase

## Criar o chão

Será o chão do seu jjogo

1. No Hierarchy, crie um novo GameObject, adicione um BoxCollider2D e renomeie para Chão.
2. Marque a tag Ground.

## Criar Plataformas e Obstáculos

Será a plataforma e obstáculos

1. Use objetos 2D para criar plataformas.
2. Adicione BoxCollider2D para colisões.

# Criando os Portais e Portas

Criação dos portais e portas do jogo

1. No Hierarchy, crie dois objetos vazios chamados PortalFogo e PortalAgua.
2. Adicione BoxCollider2D e marque como "Trigger".
3. No script Doors.cs, adicione:

using UnityEngine;

public class Doors : MonoBehaviour

{

public string requiredTag;

private bool isOpen = false;

private void OnTriggerEnter2D(Collider2D collision)

{

if (collision.gameObject.CompareTag(requiredTag))

{

isOpen = true;

}

}

}

1. Associe o script às portas e configure requiredTag conforme necessário.

# Criando Botões para Portões

Criação dos botões e portões

1. Crie um objeto vazio chamado Botão e adicione BoxCollider2D (como "Trigger").
2. No script Gates.cs, adicione:

using UnityEngine;

public class Gates : MonoBehaviour

{

public GameObject gate;

private bool platePressed = false;

private void OnTriggerEnter2D(Collider2D collision)

{

if (collision.CompareTag("Fogo") || collision.CompareTag("Agua"))

{

platePressed = true;

gate.SetActive(false);

}

}

}

1. Associe o script ao botão e arraste a porta no Inspector.

# Criando o Sistema de Reinício e Vitória

Essa parte o jogo irá reiniciar e poderá ganhar o jogo

1. Adicione um botão no UI para reiniciar o jogo.
2. No script GameManager.cs, adicione:

using UnityEngine;

public class Gates : MonoBehaviour

{

public GameObject gate;

private bool platePressed = false;

private void OnTriggerEnter2D(Collider2D collision)

{

if (collision.CompareTag("Fogo") || collision.CompareTag("Agua"))

{

platePressed = true;

gate.SetActive(false);

}

}

}

1. Associe o script ao botão e arraste a porta no Inspector.