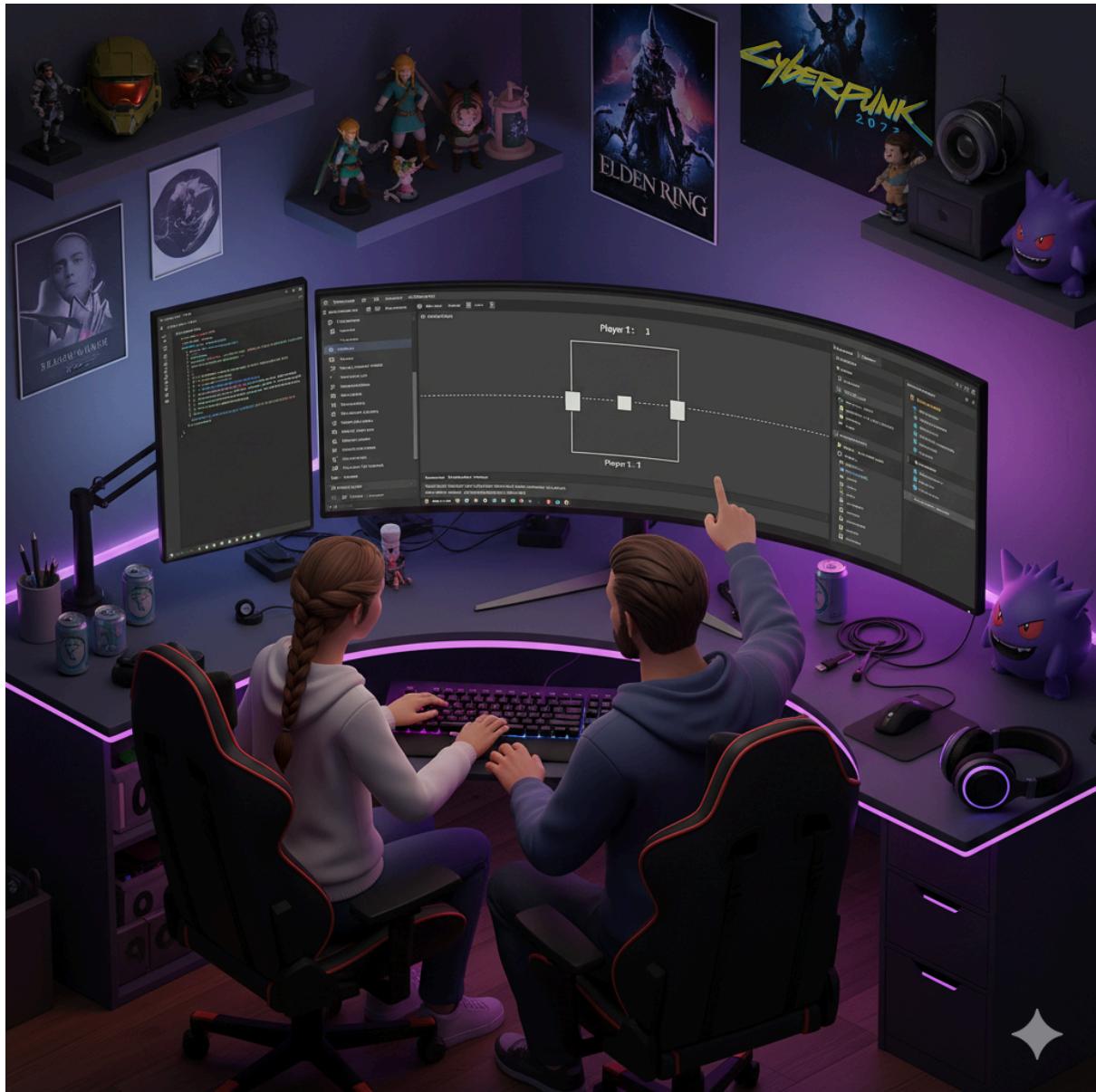


Introdução ao Desenvolvimento de Jogos com Unity



Semana Integrada UniSENAC

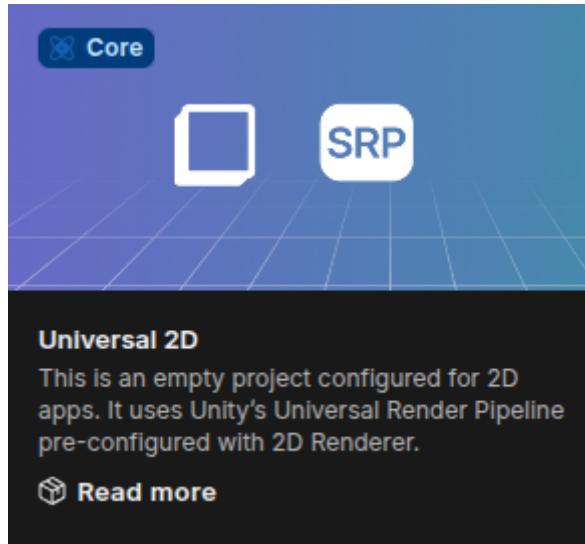
Vamos desenvolver uma versão do jogo Pong.

PONG: CONFRONTO ELEMENTAL

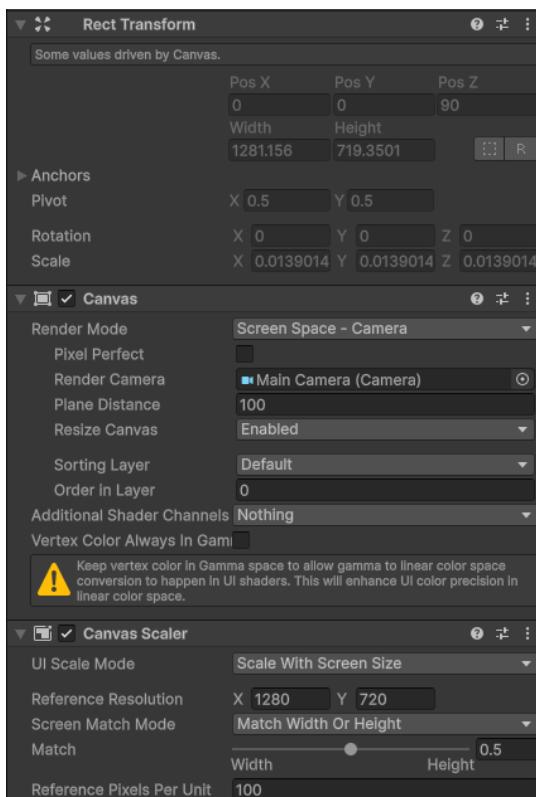


Passo a Passo - Projeto Unity

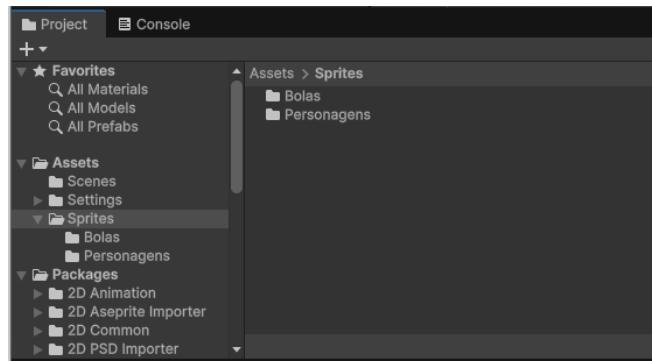
- Criar um novo projeto 2D



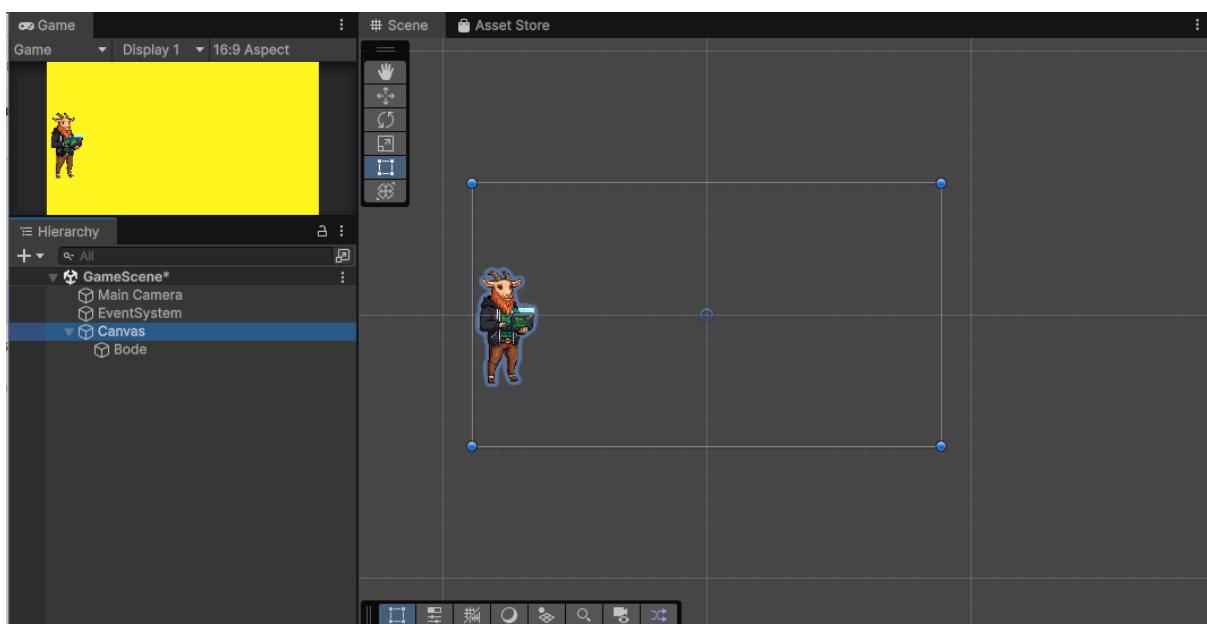
- Criar um CANVA UI>CANVA
- Vincular “Main Camera” no canva
 - Seguir as configurações que estão na imagem abaixo



- Criar uma pasta chamada Sprites
 - Criar uma pasta dentro dela chamada bolas
 - Criar uma pasta dentro dela chamada Personagens



- Agora baixar as imagens deles (na pasta Sprites no Mussum)
- Agora vamo arrastar o personagem escolhido para cena do jogo, e vinculá lo ao canvas



- Agora vamos criar uma pasta chamada scripts
 - Criar dentro dela um script chamado Jogador.cs

O Script está na próxima página

```

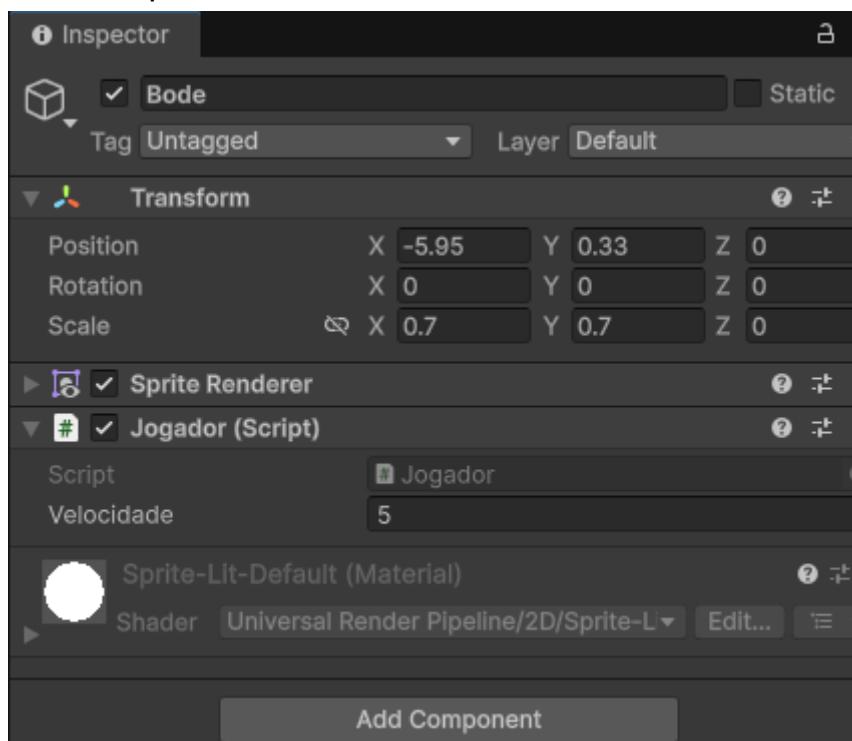
using UnityEngine;
using UnityEngine.InputSystem;

public class Jogador : MonoBehaviour
{
    public float velocidade = 5f;

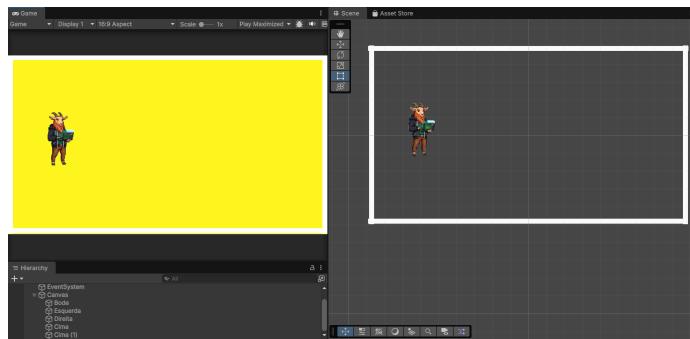
    void Update()
    {
        if (Keyboard.current.upArrowKey.isPressed)
        {
            transform.position += new Vector3(0, velocidade * Time.deltaTime, 0);
        }
        else if (Keyboard.current.downArrowKey.isPressed)
        {
            transform.position += new Vector3(0, -velocidade * Time.deltaTime, 0);
        }
    }
}

```

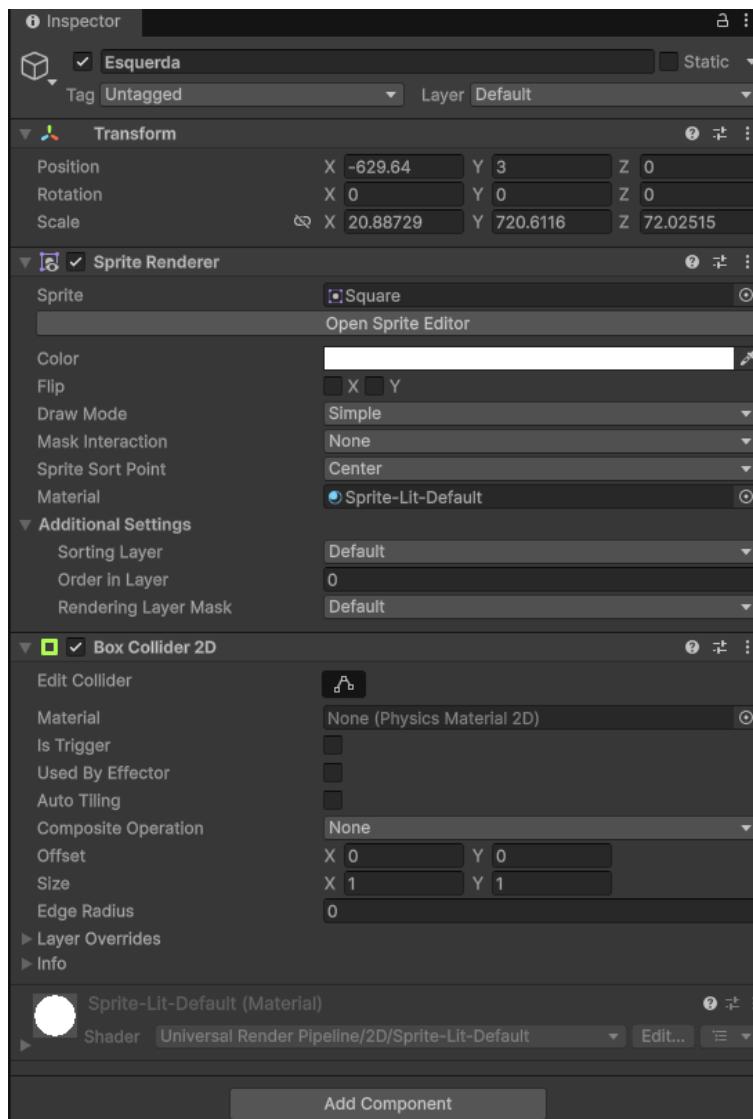
- Vincular esse Script ao Personagem
 - Nas configurações do personagem podemos mexer na velocidade.
 - Vamos aproveitar e diminuir o tamanho da escola dele.



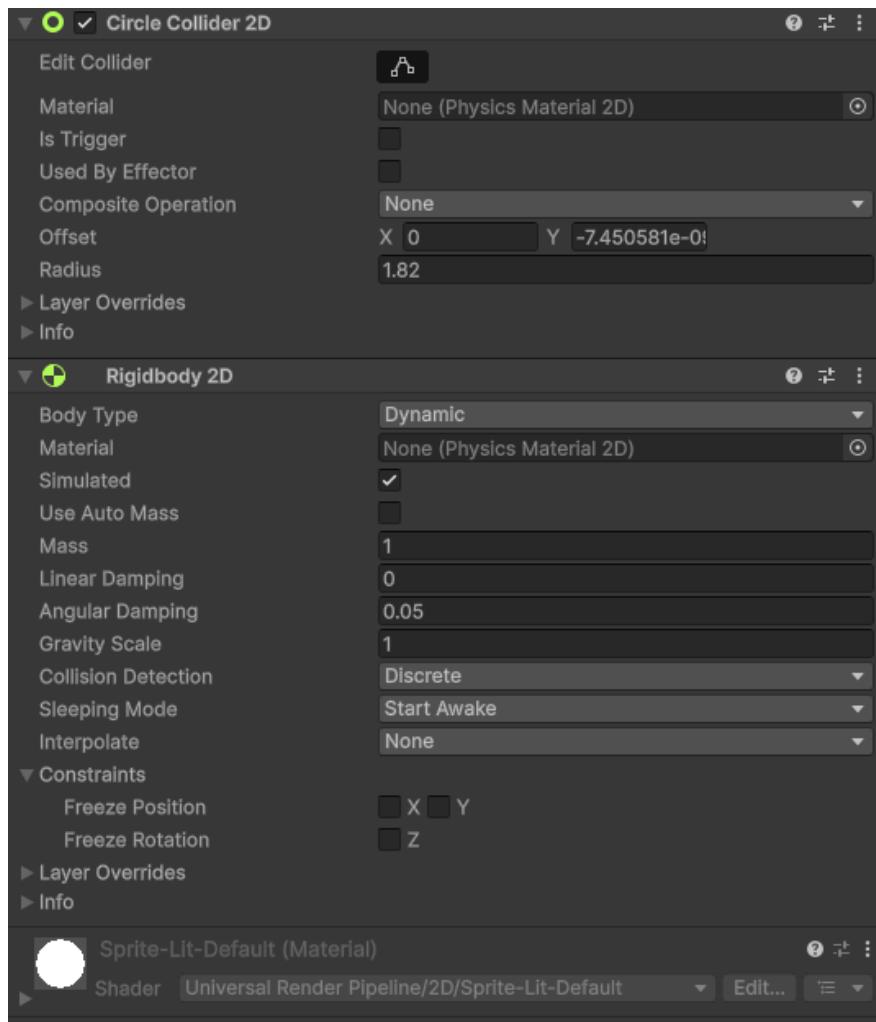
- Agora vamos criar as paredes do cenário
 - Vamos em 2D Object > Sprites > Square
 - Vamos criar 4, para fecharmos um retângulo



- Agora vamos ajustar a “hitbox” e a cor das nossas paredes
 - Clicamos nela e vamos em “Add Component” e selecionamos Box Collider 2D.
 - Para mudar agora na aba de Sprite Renderer tem a opção color



- Agora vamos configurar a bola
 - Selecionamos o Sprite da bola a nossa escolha, e arrastamos para a cena.
 - Ajustamos o tamanho dela de acordo com a necessidade.
 - Vamos em “Add Component” e adicionamos o Circle Collider 2D.
 - Vamos em “Add Component” novamente e adicionamos o Rigidbody 2D



- Agora vamos aproveitar e já fazer o Script da bola
 - Criar um Script chamado Bola.cs dentro da pasta Scripts
 - Vincular os Script na bola

O Script está na próxima página

```

using UnityEngine;

public class Bola : MonoBehaviour
{
    Vector2 direcao;
    private int velocidade;
    private Rigidbody2D rb;

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        velocidade = 5;
        direcao = new Vector2(1, 1).normalized;
    }

    void Update()
    {
        rb.linearVelocity = direcao * velocidade;
    }
}

```

- Agora vamos criar as Tags, com ela vamos começar a colocar mais lógica no nosso jogo.
 - Vamos selecionar qualquer objeto.
 - Vamos em “Tags” na parte de cima do “inspector” do objeto e clicar em “Add Tag...”
 - Vamos em tags e clicar no símbolo de “+”
 - vamos criar as seguintes Tags
 - Parede
 - Esquerda
 - Direita
 - Jogador
 - Agora vamos Vincular essas Tags a seus respectivos objetos
 - Parede : Cima e Baixo
 - Esquerda: Esquerda
 - Direita: Direita
 - Personagem: Jogador
- Agora vamos adicionar no código da bola para adicionar a lógica de quando a bola bater em determinadas tags algo acontecer
 - Se Bater na tag “Parede”, inverter o eixo Y para a bola mudar de direção na vertical
 - Se bater nas tags “Direita” ou “Jogador” inverter o eixo X para a bola mudar sua direção na horizontal

```

void OnCollisionEnter2D(Collision2D col)
{
    Debug.Log("Colidiu com: " + col.gameObject.name);
    Debug.Log("Direção: " + direcao);

    if (col.gameObject.CompareTag("Parede"))
    {
        Debug.Log("Colidiu com a parede superior/inferior!");
        direcao.y = -direcao.y;
    }

    if (col.gameObject.CompareTag("Jogador"))
    {
        Debug.Log("Colidiu com o jogador!");
        direcao.x = -direcao.x;
    }

    if (col.gameObject.CompareTag("Direita"))
    {
        Debug.Log("Colidiu com a parede Direita!");
        direcao.x = -direcao.x;
    }
}

```

- Agora vamos adicionar pontuação ao jogo ao personagem, adicionando textos na HUD do jogo
 - Vamos adicionar um texto na tela UI>TextMeshPro e escrever Pontos
 - Agora vamos mexer na lógica da bola para fazer com que ao tocar no personagem suba a pontuação e quando tocar na esquerda reseete o jogo

```

using UnityEngine;
using TMPro;
using UnityEngine.SceneManagement;

public class Bola : MonoBehaviour
{
    Vector2 direcao;
    private int velocidade;
    private Rigidbody2D rb;
    private int contador = 0;
    public TextMeshProUGUI TextoPontos;

```

```

void Start()
{
    rb = GetComponent<Rigidbody2D>();
    velocidade = 5;
    direcao = new Vector2(1, 1).normalized;
}

void Update()
{
    rb.linearVelocity = direcao * velocidade;
    TextoPontos.text = "Pontos: " + contador;
}

void OnCollisionEnter2D(Collision2D col)
{
    Debug.Log("Colidiu com: " + col.gameObject.name);
    Debug.Log("Direção: " + direcao);

    if (col.gameObject.CompareTag("Parede"))
    {
        Debug.Log("Colidiu com a parede superior/inferior!");
        direcao.y = -direcao.y;
    }

    if (col.gameObject.CompareTag("Jogador"))
    {
        Debug.Log("Colidiu com o jogador!");
        direcao.x = -direcao.x;
        contador++;
        TextoPontos.text = "Pontos: " + contador;

    }

    if (col.gameObject.CompareTag("Direita"))
    {
        Debug.Log("Colidiu com a parede Direita!");
        direcao.x = -direcao.x;
    }

    if (col.gameObject.CompareTag("Esquerda"))
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }
}

```

```
    }  
  
}
```

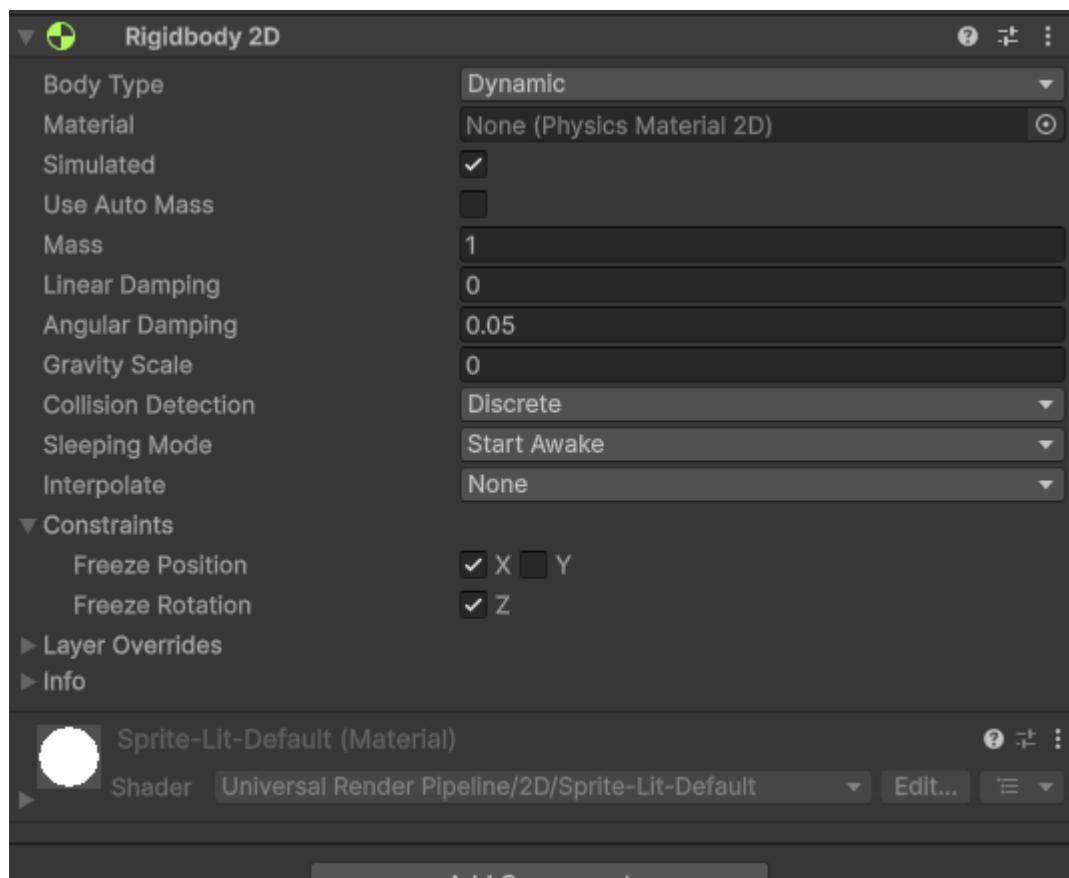
- Agora vamos fazer com que cada vez que a bola bater no personagem ela aumente a velocidade.
 - Vamos criar uma variável de “velocidade inicial” e deixar um valor padrão ex: 5
 - vamos atribuir a velocidade inicial a nossa variável de velocidade, que assim sempre que perdemos ou iniciarmos o jogo novamente, a velocidade volte para a velocidade inicial
 - Podemos também colocar um limitador de velocidade para ela não aumentar infinitamente, no código abaixo vou deixar as 2 opções porém com a parte de limitar comentada.

```
private int velocidade;  
private int velocidadeInicial = 5;
```

```
if (col.gameObject.CompareTag("Jogador"))  
{  
    Debug.Log("Colidiu com o jogador!");  
    direcao.x = -direcao.x;  
    contador++;  
    TextoPontos.text = "Pontos: " + contador;  
    velocidade += 1  
    // velocidade = Mathf.Min(velocidade + 1, 20);  
}
```

- Agora vamos ajustar um bug do personagem, que ele atravessa as paredes
 - Vamos adicionar nele um Rigidbody2D
 - Vamos travar selecionar o “Freeze Position X” e Freeze Position Z
 - Vamos remover a Escala de gravidade em “Gravity Scale”
 - E vamos no Rigidbody2D da bola e reduzir a massa dela para 0, para evitar que ela influencie na movimentação do personagem em altas velocidades
 - Na print abaixo temos os configurações de Rigidbody2D do personagem

O Print está na próxima página



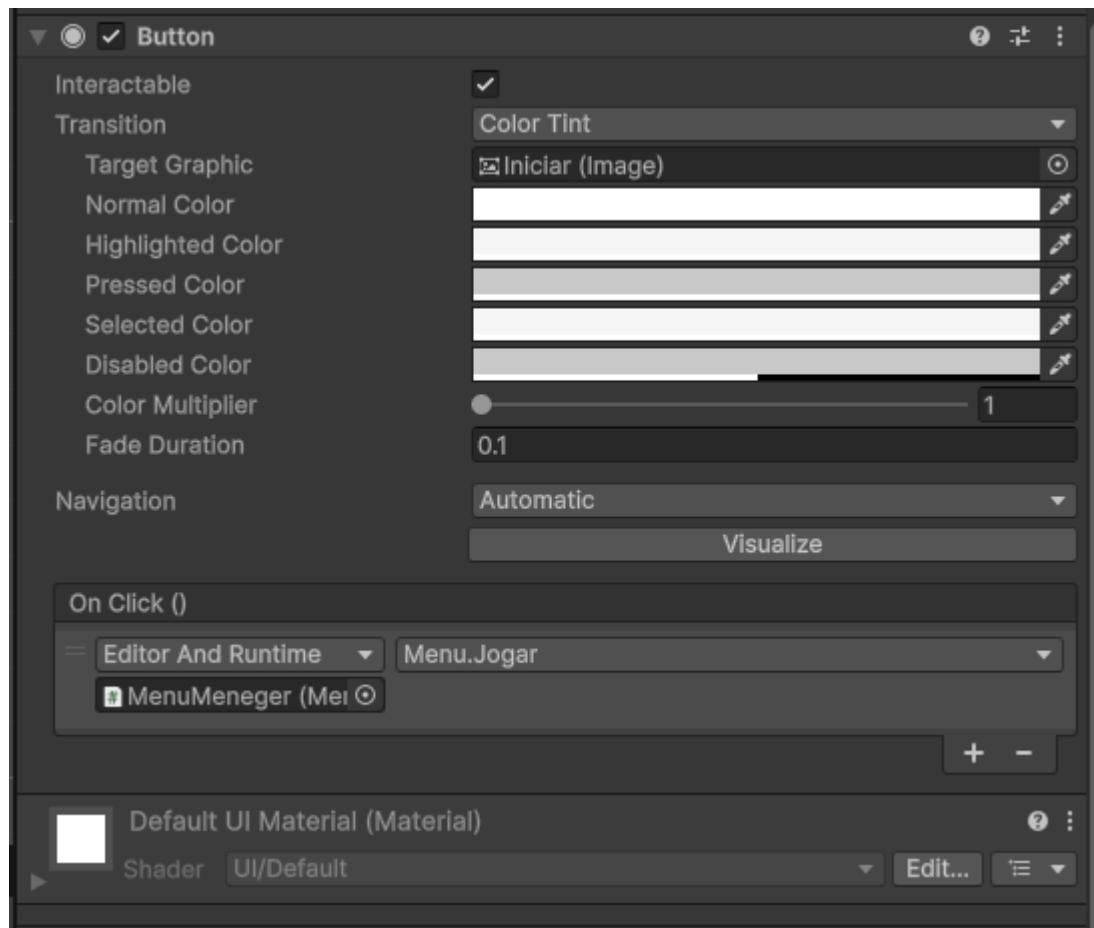
- Vamos agora criar uma Cena de menu inicial para o nosso jogo
 - Vamos clicar em File no canto superior da página e selecionar new scene
- Agora vamos fazer igual a cena do jogo e adicionar a o canvas e configurar ele.
- Vamos agora para a customização do menu.
 - Vamos adicionar um fundo à nossa página UI>Painel.
 - Vamos adicionar um botão UI>Button(Text Mesh Pro) e vamos customizar ele, é possível importar imagens para o botão.
 - Vamos fazer o mesmo com o botão de sair do jogo.
 - Vamos adicionar a cena a build do jogo, file>Build Profiles>Open scene list>Add open scenes
 - Vamos criar um Script para programarmos o funcionamento desses botões.

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class Menu : MonoBehaviour
{
    public void Jogar()
    {
        SceneManager.LoadScene("GameScene");
    }

    public void SairJogo()
    {
        Application.Quit();
    }
}
```

- Agora vamos criar um elemento vazio na nossa cena, “Create Empty” e dar o nome de “MenuManager”
 - Adicionamos o script de Menu.cs nele.
 - Vamos vincular o método de iniciar jogo que criamos ao botão de “Iniciar Jogo”, arrastando o MenuManager no evento de On Click () do botão conforme a print abaixo.
 - Vamos fazer o mesmo com o botão de sair.



- Agora podemos fazer a build do nosso jogo.
- Desafios
 - Criar sistemas de Vida, iniciando em 3 e voltar para o menu após terminar todas as vidas
 - Lembre de não reiniciar a velocidade da bola sem perder as 3 vidas.
 - Utilizar `transform.position = Vector2.zero;` ao invés de `SceneManager.LoadScene(SceneManager.GetActiveScene().name);` Pois ele reinicia a posição da bola, sem reiniciar a partida.