

Andrei Inoue Hirata



Flappy Birds

Criar Seu Próprio Jogo Flappy Bird em 10 Minutos

Se você é um grande fã de jogos para celular, provavelmente já ouviu falar do jogo Flappy Bird. Talvez você até tenha instalado e jogado no seu celular pelo menos uma vez.

Flappy Bird é um jogo mobile disponível tanto para Android quanto para iOS, desenvolvido pelo desenvolvedor indie vietnamita Dong Nguyen. O jogo foi lançado inicialmente em 24 de maio de 2013, mas foi repentinamente removido da PlayStore por volta de fevereiro de 2014.

O Jogo Flappy Bird

O conceito do jogo é bastante simples. O jogador controla Faby (o pássaro), que se move horizontalmente pela tela enquanto a gravidade o puxa para baixo. O objetivo é evitar que Faby colida com os canos e, a cada cano atravessado com sucesso, o jogador ganha um ponto.

Segundo Dong Nguyen, o jogo foi criado em apenas 3 dias.

Neste tutorial, vamos criar um jogo estilo Flappy Bird, usando a Unity Game Engine.



Andrei Inoue Hirata

Sumário

O Jogo Flappy Bird	1
Criando Flappy Bird Usando Unity – Guia Passo a Passo	3
Criando o Projeto na Unity.....	3
Baixando os Assets	3
Configurando o Jogo.....	6
Criando o Jogador	8
A Plataforma	10
Física do Jogador	11
Player Script.....	12
Os Obstáculos	13
Plataforma Rolante (Scrolling Platform)	16
Interface do Usuário (UI)	17
Gerenciador do Jogo (Game Manager)	19
Morte do Jogador.....	23

Andrei Inoue Hirata

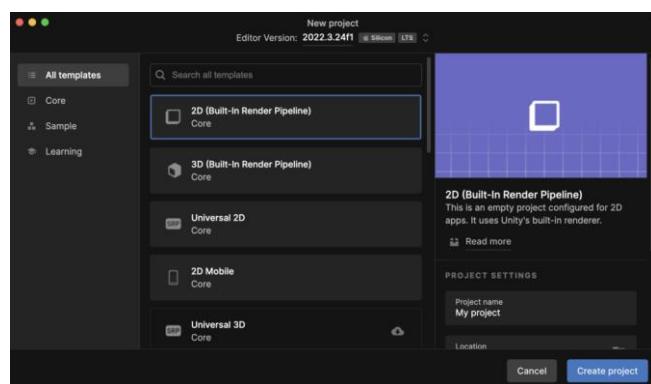
Criando Flappy Bird Usando Unity – Guia Passo a Passo

Antes de começarmos, é fundamental garantir que você tenha todas as ferramentas necessárias para criar seu próprio jogo. Como estamos desenvolvendo um clone do Flappy Bird na Unity, será necessário instalar esse motor de jogo no seu computador.

Se você ainda não tem o Unity instalado, pode baixá-lo no site [oficial da Unity](#). Este é um passo essencial, pois a plataforma da Unity servirá como a base e ferramenta principal para o desenvolvimento do seu jogo.

Criando o Projeto na Unity

1. Depois que o Unity estiver instalado, crie um novo projeto Unity e certifique-se de definir seu template para 2D.



2. Dentro da pasta Assets, crie uma nova pasta chamada de FlappyBirds, pois é nela que teremos outras pastas futuramente do jogo.

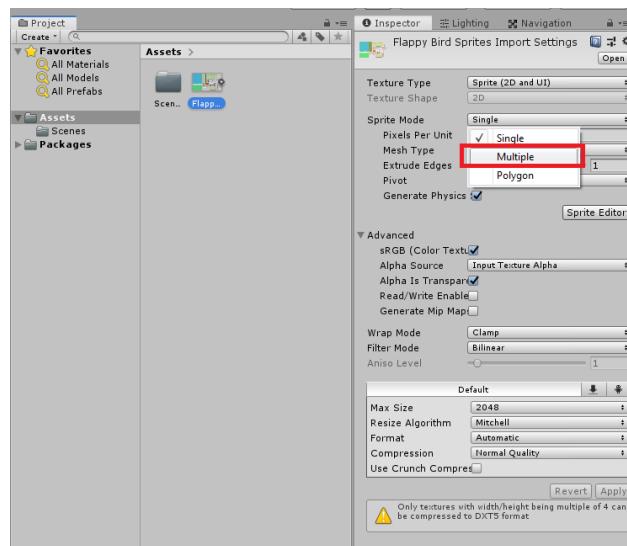
Baixando os Assets

Antes de começarmos a criar o jogo, obviamente precisamos dos assets, como o pássaro, os canos, as interfaces, etc.

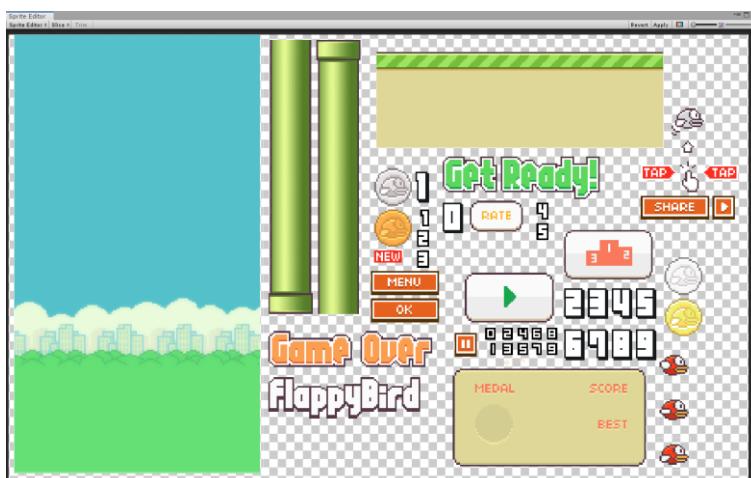
Andrei Inoue Hirata



1. Faça o download dos arquivos e depois volte para o Unity e importe a imagem.
2. Depois de importado, selecione o sprite e certifique-se de definir: Sprite Mode para Multiple; Filter Mode para Point; Compression para None (Isso fará com que seu sprite fique nítido e de alta qualidade.)
3. Depois, clique em Apply para salvar as alterações.



Atenção: Se você já percebeu, os assets estão compilados em um arquivo PNG, que é então convertido em um sprite após ser importado para o Unity. Precisamos cortar esse sprite usando o Sprite Editor para podermos utilizar os objetos, como o pássaro, os canos, etc.

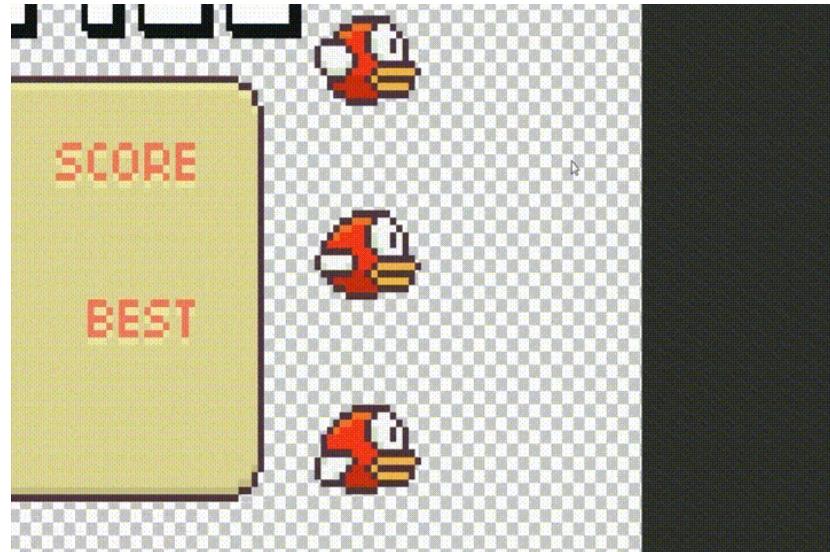


Andrei Inoue Hirata

4. Selecione o sprite novamente e, logo abaixo da opção Generate Physics, clique no botão Sprite Editor.

Isso abrirá uma nova janela. Agora, para criar novos sprites:

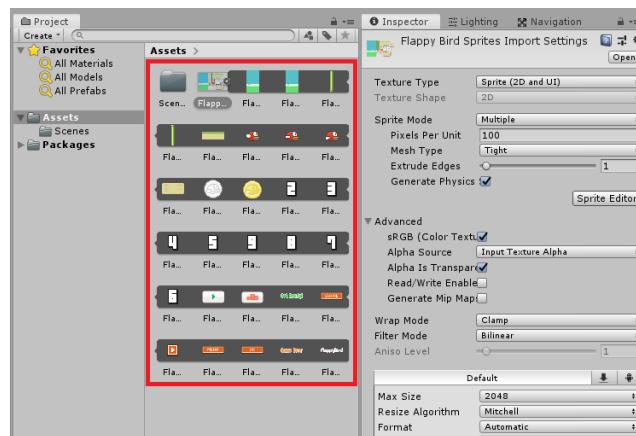
5. Selecione o objeto com botão esquerdo e arraste a área dele no Sprite Editor.



6. Depois de terminar de cortar o sprite, clique em Apply.



Uma vez concluído com sucesso, você verá todos os sprites cortados dentro do seu asset de sprite do Flappy Bird.

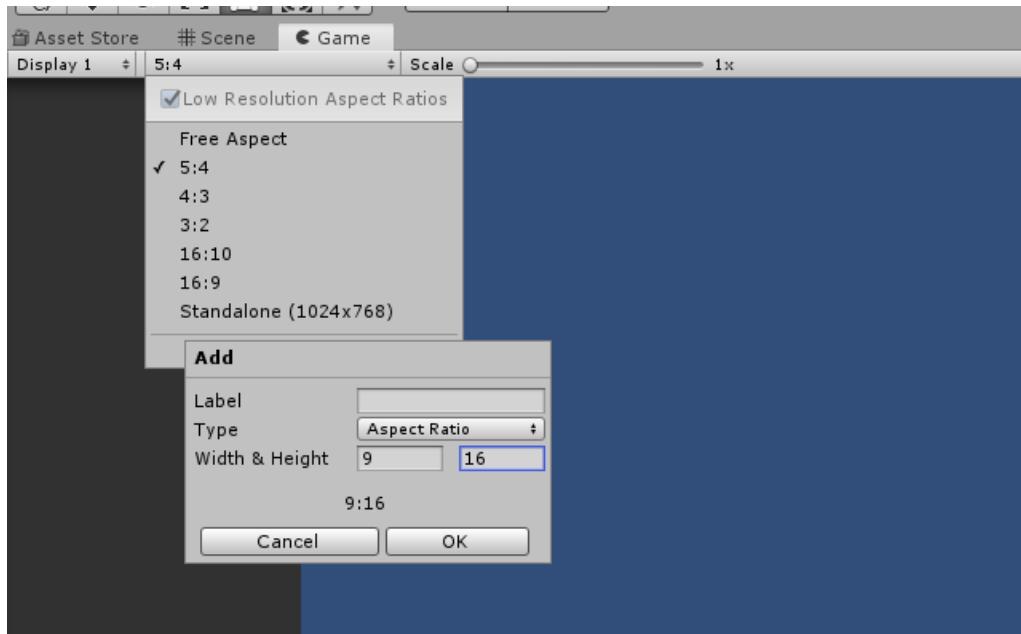


Andrei Inoue Hirata

Configurando o Jogo

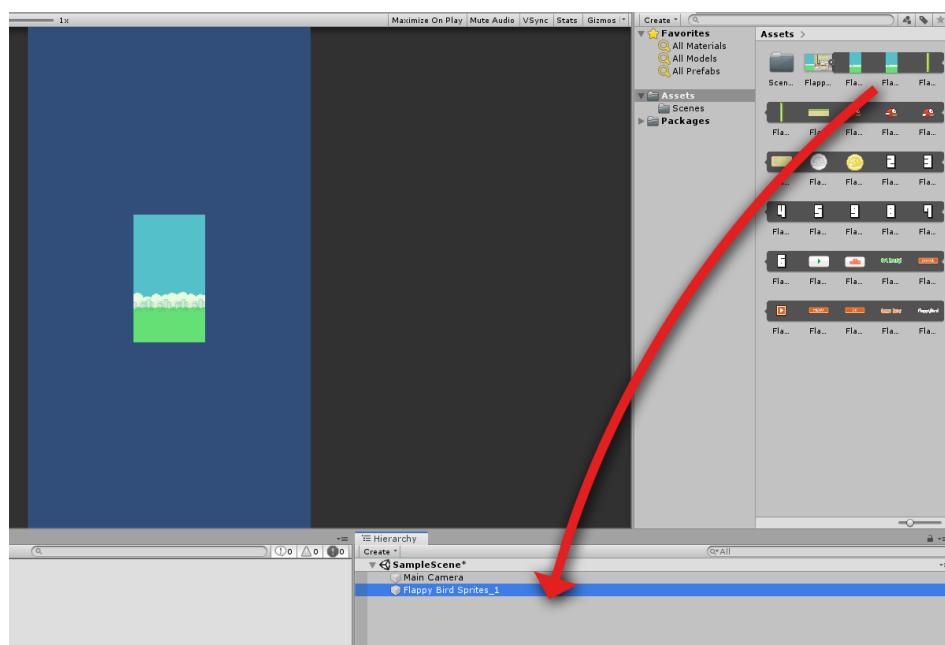
Agora que terminamos com nossos assets, vamos iniciar este tutorial ajustando a proporção de tela do nosso jogo:

1. Clique na janela Game e adicione a proporção para 9:16 e com o nome de Smartphone.
2. Clique em OK para confirmar.



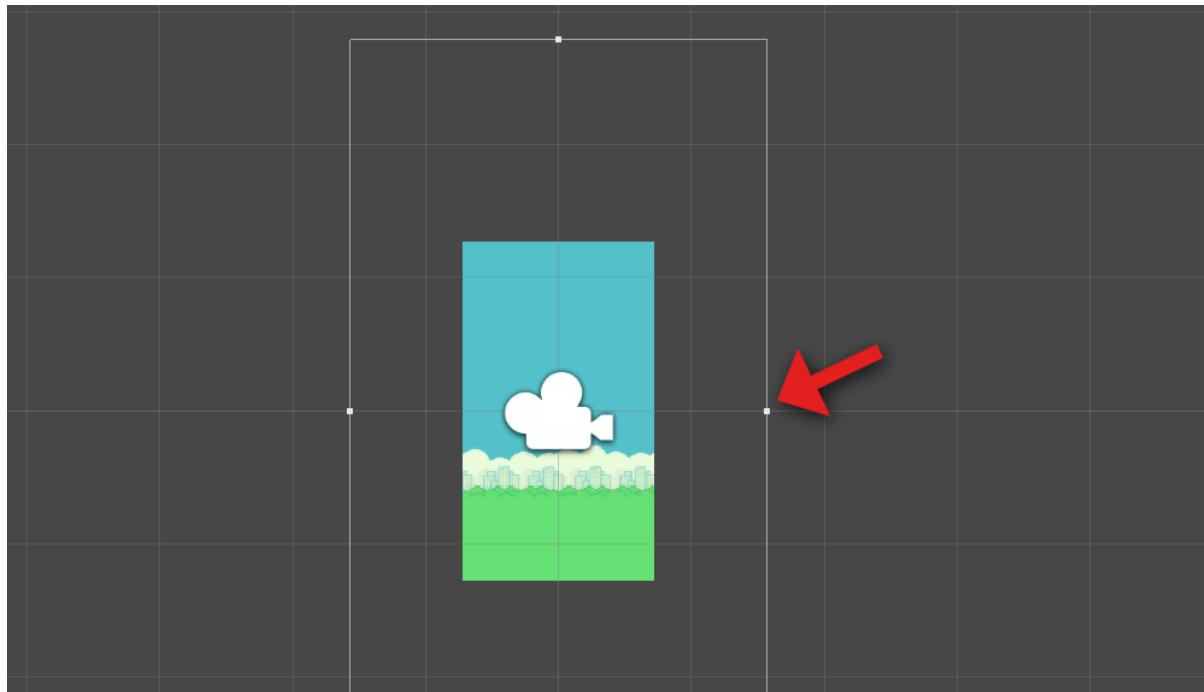
Em seguida, arrastaremos o fundo do jogo para a Hierarquia.

3. Clique na imagem do Fundo e arraste para a janela Hierarchy e coloque o nome de ImgBackground.

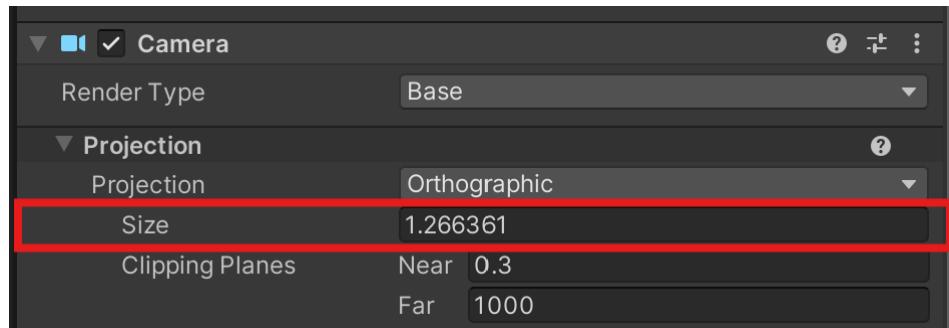


Andrei Inoue Hirata

4. Em seguida, vá para a janela Scene, clique na Main Camera dentro da Hierarquia e arraste o ponto branco para baixo até que corresponda ao tamanho do nosso fundo.

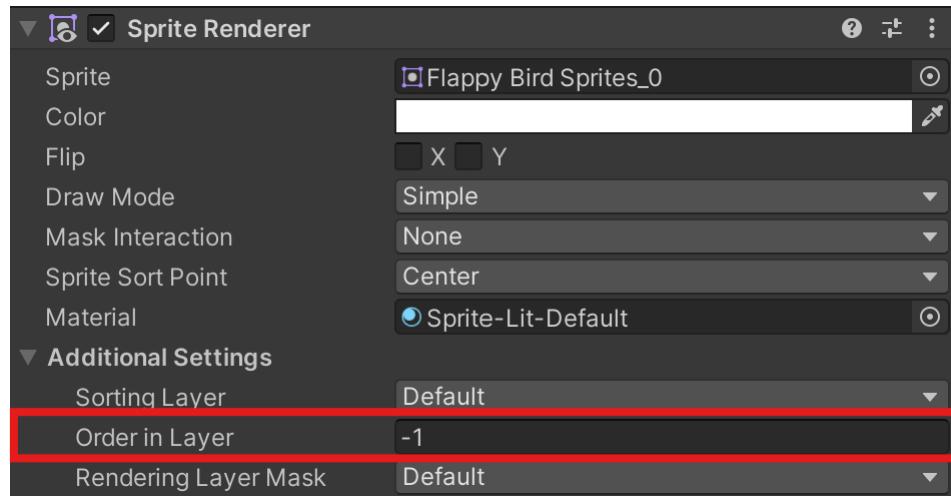


Voce pode observar que o Size irá diminuir

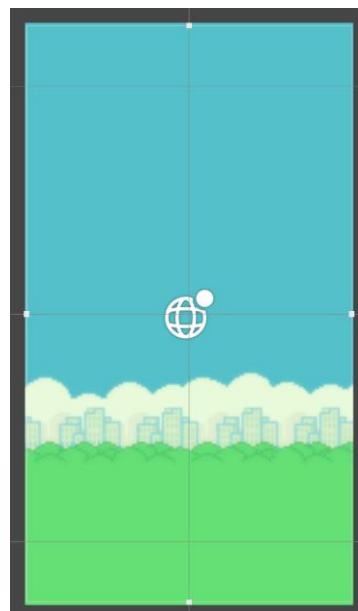


5. Depois, vá para a janela Inspector, selecione a imagem de fundo e defina o valor de Order in Layer para -1.

Andrei Inoue Hirata



Isso deve deixar a prévia do seu jogo pronta para incluir o jogador. O resultado final ficará assim.



Criando o Jogador

Agora que terminamos com o fundo, o próximo passo é trabalhar no pássaro, que será o nosso jogador.

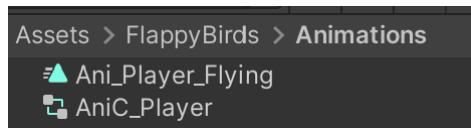
1. Volte para os arquivos do projeto, abra o sprite do Flappy Bird novamente e selecione as 3 imagens do pássaro e arraste eles para a janela Hierarchy.

Andrei Inoue Hirata



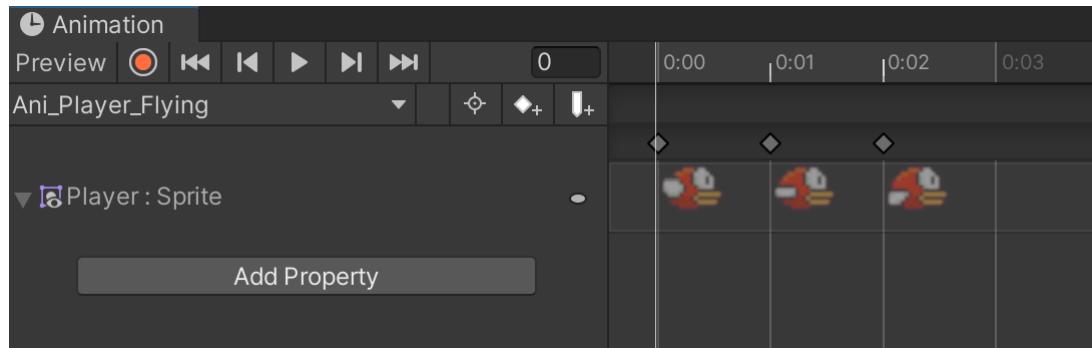
Observe que irá aparecer uma janela para salvar a animação e depois você irá perceber que um animator será criado também de forma automática.

2. Na janela aberta crie uma pasta chamada Animations dentro de Assets/FlappyBirds
3. Mova e renomei o animation para Ani_Player_Flying e o animator para AniC_Player.
4. Renomeie o GameObject para Player.

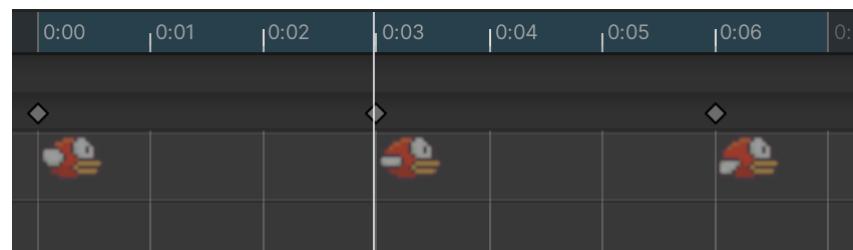


5. Em seguida, abra a janela de Animação pressionando CTRL + 6.

Observe que você irá ver as 3 imagens que foram adicionadas e cada uma delas se encontra no frame.



6. Para melhorar a aparência da animação, ajuste-a arrastando os quadros-chave: O último keyframe para 0:06, o segundo keyframe para 0:03 e o primeiro deixe em 0:00



7. Arraste o Player para um pouco para trás, com valor no eixo x=-0.5

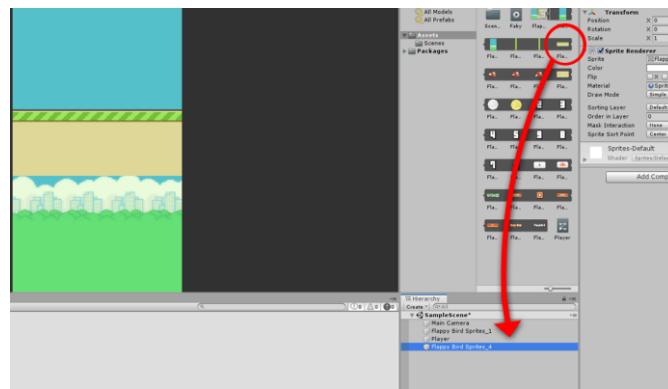
Andrei Inoue Hirata

8. Execute o jogo e veja o pásasso bater as asas.

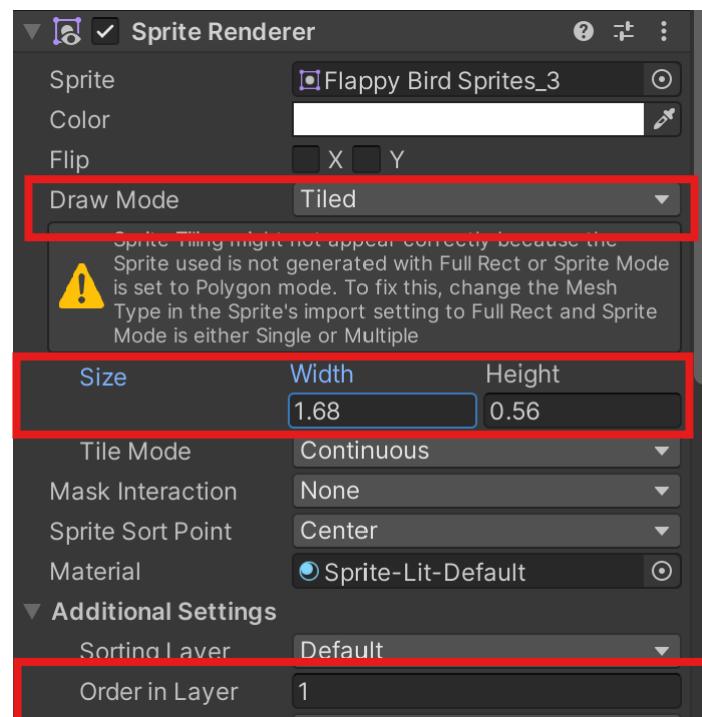
A Plataforma

Agora iremos desenvolver a plataforma que ficará na parte inferior, como se fosse um chão.

1. Vá para a janela do Projeto novamente e arraste a plataforma para a Hierarquia.



2. Agora, obviamente, precisamos arrastar essa plataforma para a parte inferior da tela.
3. Renomeie o GameObject da plataforma para ImgPlataform.
4. Selecione o ImgPlataform e altere: Draw Mode para Tiled e Order in Layer para 1.



Isso fará com que nossa plataforma fique contínua e seja exibida acima de tudo. Agora, vamos adicionar um Box Collider a esse GameObject.

5. Selecione o GameObject ImgPlataform.
6. Clique no botão Add Component.

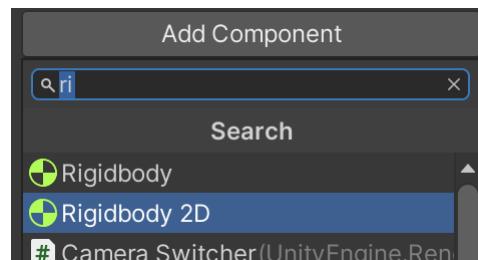
Andrei Inoue Hirata

7. Pesquise por Box Collider 2D e adicione-o.
8. Certifique-se de que a opção Auto Tiling está definida como true.

Física do Jogador

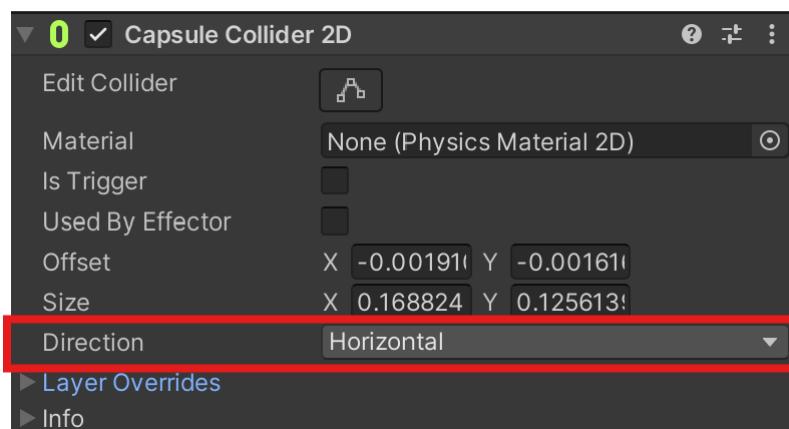
Para que este jogo funcione corretamente, precisamos aplicar física ao nosso jogador.

1. Com o GameObject Player selecionado, vá para a janela Inspector.
2. Clique em Add Component.
3. Pesquise por Rigidbody 2D e adicione-o.

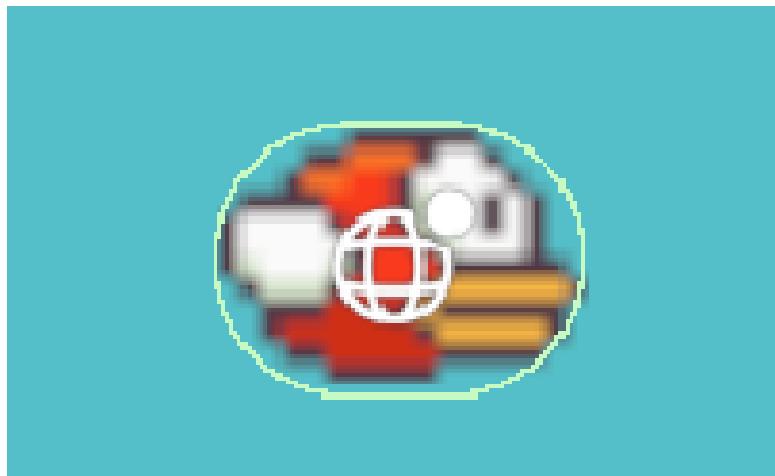


Em seguida, adicionaremos outro componente, o Capsule Collider.

1. Clique no botão Add Component novamente.
2. Pesquise por Capsule Collider 2D e adicione-o.
3. Defina a direção do Capsule Collider 2D para Horizontal.
4. Clique no botão Edit Collider e ajuste o tamanho do Capsule Collider para que se encaixe perfeitamente no seu jogador.



Andrei Inoue Hirata



Player Script

Vamos criar um novo script para fazer nosso jogador pular assim que tocarmos na tela.

1. Vá para a janela do Projeto e crie uma pasta chamada Scripts
2. Dentro dela com com o botão direito, selecione Create → C# Script.
3. Nomeie o script como S_Player.cs.
4. Após a criação, arraste o script para o GameObject Player.
5. Abra o script com o Visual Studio e digite o código.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class S_Player : MonoBehaviour
{
    public float velocity = 2.4f;
    private Rigidbody2D rigidbody;

    // Start is called before the first frame update
    void Start()
    {
        rigidbody = GetComponent<Rigidbody2D>();
    }
}
```

Andrei Inoue Hirata

```
// Update is called once per frame

void Update()
{
    if(Input.GetMouseButtonDown(0))
    {
        rigidbody.velocity = Vector2.up * velocity;
    }
}
```

6. Execute o jogo e veja o jogador pulando ao clicar com o botão direito do mouse

Os Obstáculos

Agora que nosso jogador, plataforma e fundo estão prontos, é hora de criar os obstáculos (conhecidos como os canos).

1. Abra os sprites novamente e arraste os dois canos para a cena.
2. Com os canos selecionados, adicione o componente Box Collider 2D.
3. Alinhe os canos verticalmente, como no jogo original.
4. Crie um GameObject vazio na Hierarquia e renomeie-o para Obstacle Pipes.
5. Arraste os dois canos para dentro desse GameObject vazio, tornando-o o objeto pai dos canos.
6. Certifique-se de posicionar o GameObject vazio entre os dois canos, para que a estrutura fique parecida com o jogo original.



7. Agora, vamos criar um novo script C# chamado Obstacle.cs
8. Vá para a janela do Projeto e clique com o botão direito, selecione Create → C# Script.
9. Nomeie o script como S_Obstacle.cs.
10. Após a criação, arraste o script para o GameObject "Obstacle Pipes".
11. Abra o script com o Visual Studio para começar a programar a lógica dos obstáculos.

```
using System.Collections;
using System.Collections.Generic;
```

Andrei Inoue Hirata

```
using UnityEngine;

public class S_Obstacle : MonoBehaviour
{
    public float speed;
    // Update is called once per frame
    void Update()
    {
        transform.position += ((Vector3.left * speed) * Time.deltaTime);
    }
}
```

12. Certifique-se de arrastar este script para o GameObject "Obstacle Pipes".

Agora que nosso obstáculo está pronto, podemos criar um spawner de obstáculos para que eles apareçam continuamente.

13. Crie um GameObject vazio e renomeie-o para Obstacle Spawner.
14. Redefina sua posição Transform e arraste a posição X para a direita da tela.
15. Crie um novo script C# e nomeie-o como S_Spawner.cs.
16. Arraste este script para o GameObject "Obstacle Spawner" para vinculá-lo.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class S_Spawner : MonoBehaviour
{
    public float queueTime = 1.5f;
    private float time = 0;
    public GameObject obstacle;
    public float height;
    // Update is called once per frame
    void Update()
    {
        if(time > queueTime)
        {
            GameObject go = Instantiate(obstacle);
            go.transform.position = transform.position + new Vector3(0, Random.Range(-height, height), 0);
            time = 0;
        }
        else
            time += Time.deltaTime;
    }
}
```

Andrei Inoue Hirata

```

time = 0;

Destroy(go, 10);

}

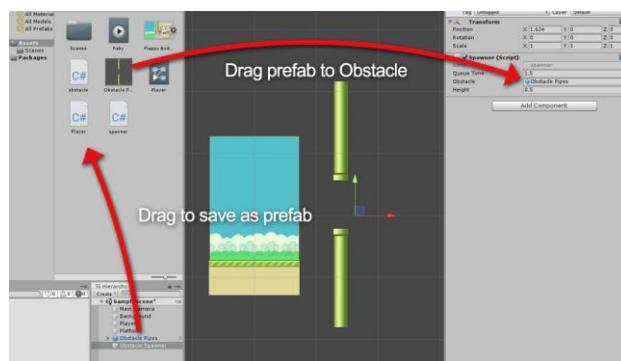
time += Time.deltaTime;

}

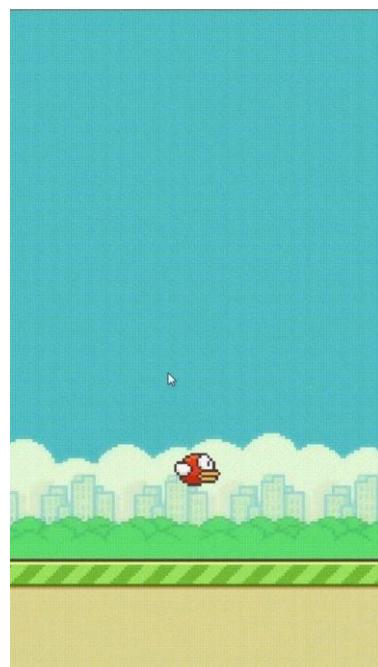
}

```

17. Certifique-se de salvar todos os seus scripts e também a sua cena.
18. Volte para o Unity e agora vamos salvar os Obstacle Pipes como um GameObject Prefab.
19. Crie uma pasta chamada Prefabs
20. Arraste o Obstacle Pipes para a pasta.
21. Selecione o GameObject "Obstacle Spawner".
22. Arraste o prefab "Obstacle Pipes" para a variável Obstacle dentro do script Spawner no Inspector.



Você deve ter os seguintes resultados.

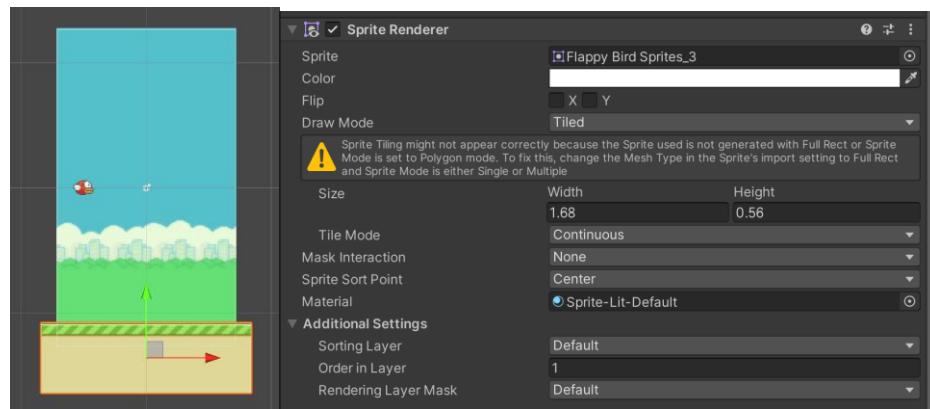


Andrei Inoue Hirata

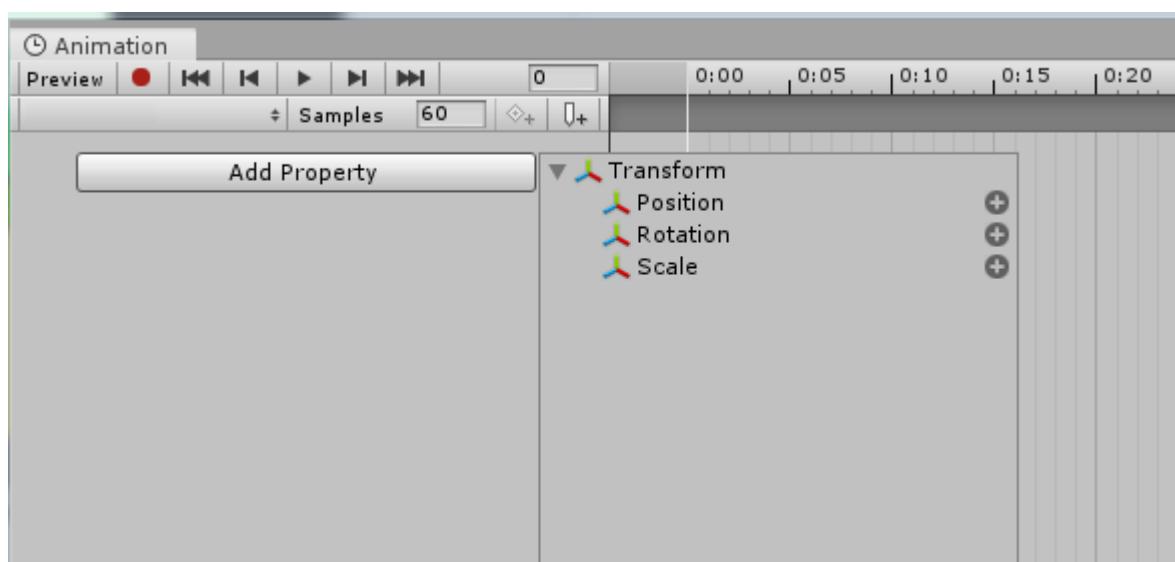
Plataforma Rolante (Scrolling Platform)

Se você olhar de perto, verá que a plataforma não está rolando horizontalmente junto com os canos, o que parece estranho. Então iremos criar esse efeito

1. Crie uma nova animação e nomeie-a como Ani_Plataform_Moving.
2. Arraste essa animação para o GameObject "ImgPlatform".
3. Mova as animações e Animations para a pasta Animations.
4. Renomeie o controlador para "AniC_Plataform"
5. Aumente a escala horizontal da plataforma para que o efeito de rolagem fique mais fluido.



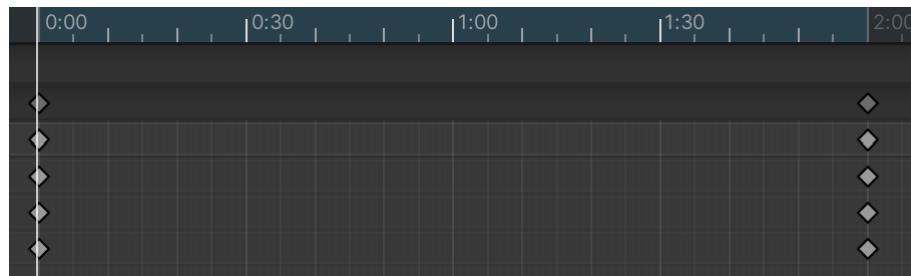
6. Em seguida, abra a janela de Animação pressionando CTRL + 6.
7. Com o GameObject "ImgPlatform" selecionado clique no botão Add Property na janela de Animação.
8. Selecione Transform → Position.



Vamos agora criar um keyframe:

Andrei Inoue Hirata

9. Clique no botão ao lado do campo numérico para adicionar um keyframe em 0:00.
10. Em aproximadamente 2:00 segundos, crie outro keyframe.
11. Desta vez, ajuste o valor de X para que a plataforma deslize horizontalmente.



Por exemplo:

12. No primeiro keyframe, defina a posição X como 0.127.
13. No segundo keyframe, defina a posição X como -0.109.

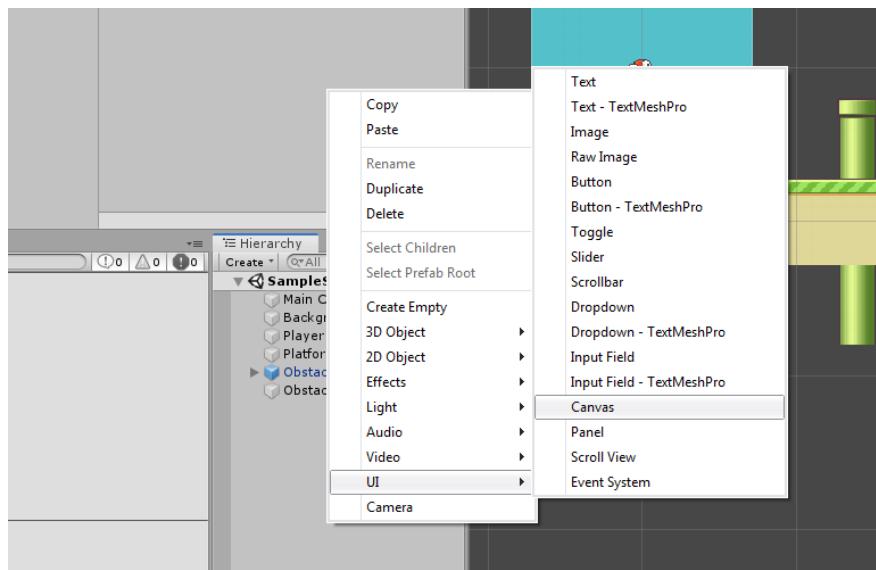
Isso fará com que a plataforma role da posição 0.127 para -0.109 a cada 2 segundos. Esses valores foram escolhidos considerando que a movimentação ocorra e visualmente de uma impressão de movimento bem suave. Dependendo do seu jogo você deve deixar de uma maneira que de essa impressão.

Interface do Usuário (UI)

O jogo está quase pronto, mas ainda falta configurar a morte do jogador e a interface do usuário quando o jogador morre.

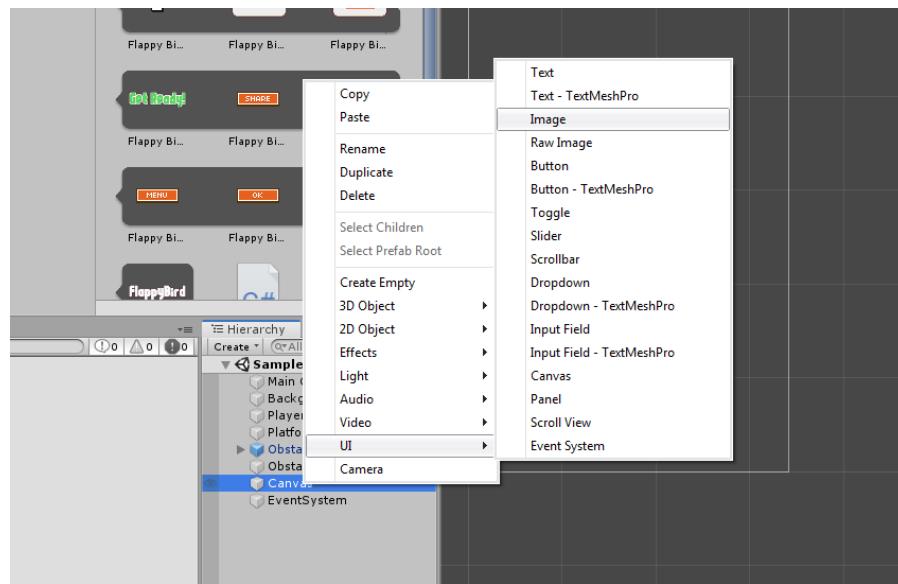
Para começar:

1. Crie um novo Canvas:
2. Vá para a Hierarquia, clique com o botão direito, selecione UI → Canvas.

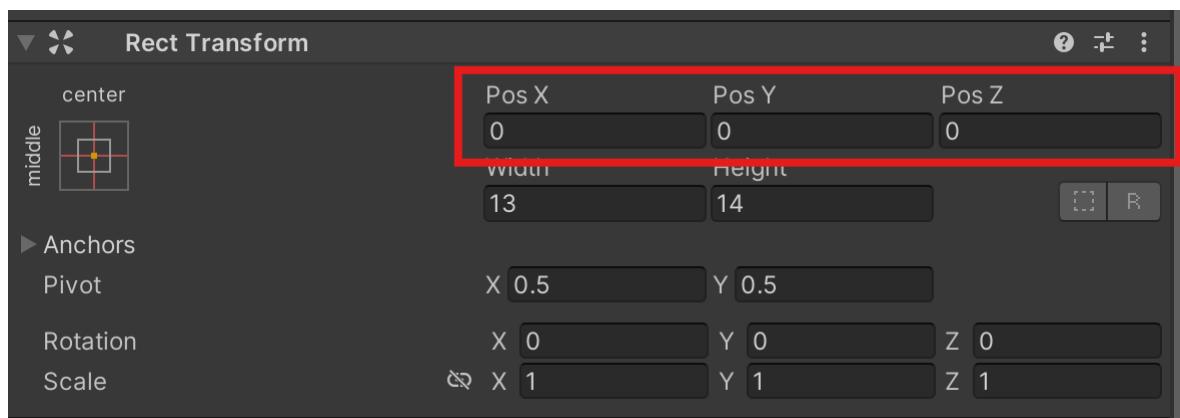


Andrei Inoue Hirata

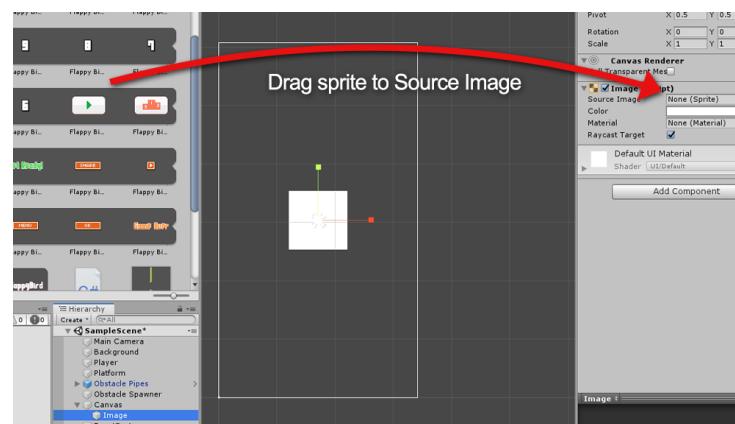
3. Clique com o botão direito no GameObject Canvas que você acabou de criar.
4. Selecione UI → Image.



5. Renomeie este GameObject de Image para "Restart Button".
6. Deixe as posições X,Y, Z para 0

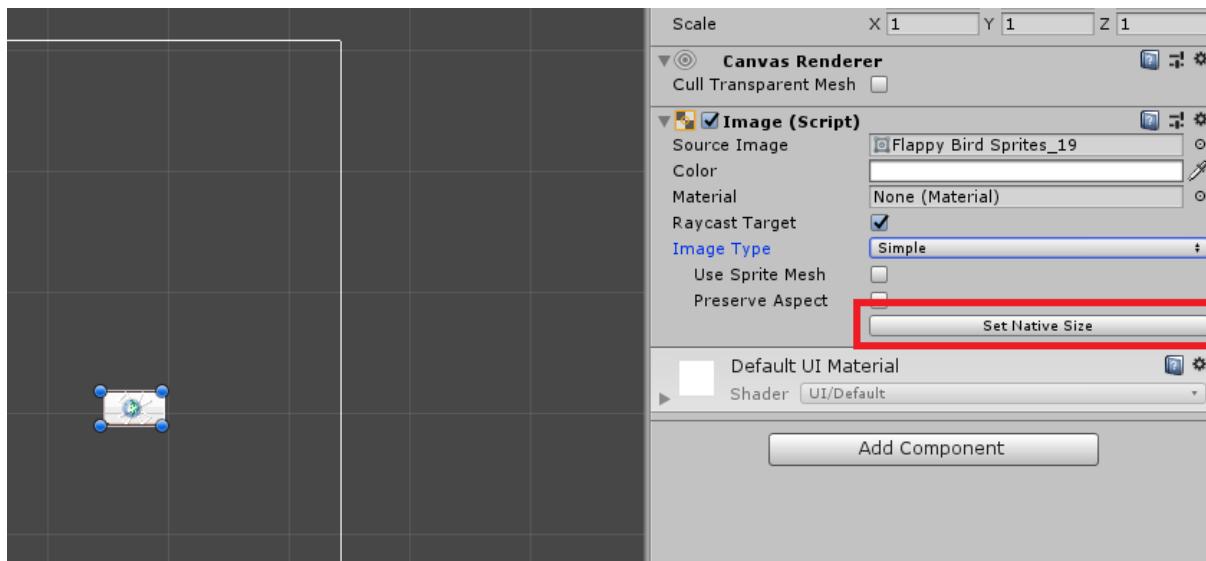


7. Abra o asset do Flappy Bird Sprite novamente.
8. Procure pelo botão de Play.
9. Arraste essa imagem para a propriedade "Source Image" do GameObject Restart Button.



Andrei Inoue Hirata

10. Depois clique no botão Set Native Size para ajustar o tamanho original da imagem.



11. Aumente o tamanho da imagem para o tamanho desejado que fique bom. No nosso caso, escalaremos a imagem para 8 nos eixos X, Y e Z.



Gerenciador do Jogo (Game Manager)

Agora será criado o Gerenciador do Jogo (Game Manager) que irá controlar todo o jogo. Ele terá recursos como: congelar o jogo no início, descongelar quando o jogador clicar no botão de início e congelar o jogo novamente quando o jogador morrer e iniciar uma contagem regressiva para reiniciar.

1. Crie um novo GameObject e nomeie-o como GameManager e coloque nos eixos X,Y,Z = 0.
2. Crie um novo script C# e nomeie-o como S_GameManager.cs.

Andrei Inoue Hirata

3. Arraste o script para o GameObject "GameManager" para vinculá-lo.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class S_GamerManager : MonoBehaviour
{
    public GameObject startButton;
    public S_Player player;
    public Text gameOverCountdown;
    public float countTimer = 5;
    // Start is called before the first frame update
    void Start()
    {
        gameOverCountdown.gameObject.SetActive(false);
        Time.timeScale = 0;
    }
    private void Update()
    {
        //if( player.isDead )
        //{
        //    //gameOverCountdown.gameObject.SetActive(true);
        //    //countTimer -= Time.unscaledDeltaTime;
        //}
        gameOverCountdown.text = "Restarting in " + (countTimer).ToString("0");
        if(countTimer < 0)
        {
            RestartGame();
        }
    }
    public void StartGame()
    {
        startButton.SetActive(false);
        Time.timeScale = 1;
    }
}
```

Andrei Inoue Hirata

```
public void GameOver()
{
    Time.timeScale = 0;
}

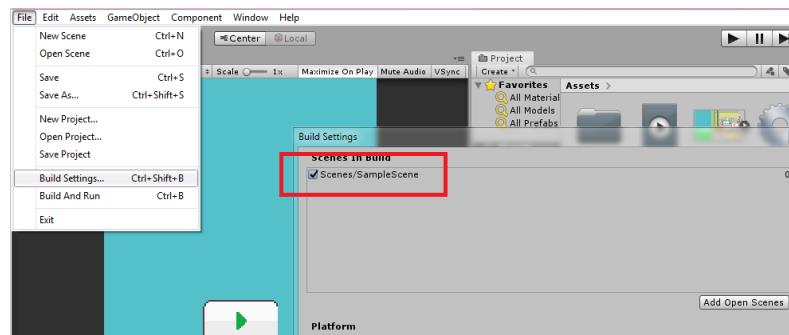
public void RestartGame()
{
    SceneManager.LoadScene(0);
}

}
```

Usando o script acima, certifique-se de que sua cena está adicionada nas Configurações de Build. Caso contrário, você receberá um erro ao iniciar o jogo. Os Passos para adicionar a cena ao Build Settings:

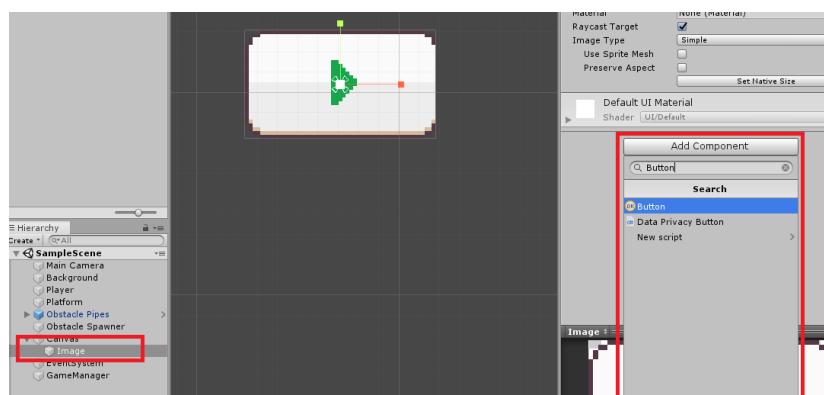
4. Vá para File → Build Settings.
5. Na janela que abrir, clique em Add Open Scenes para adicionar sua cena atual.
6. Certifique-se de que a cena está na posição 0 da lista (a cena principal do jogo).

Isso garantirá que o jogo funcione corretamente sem erros ao reiniciar.



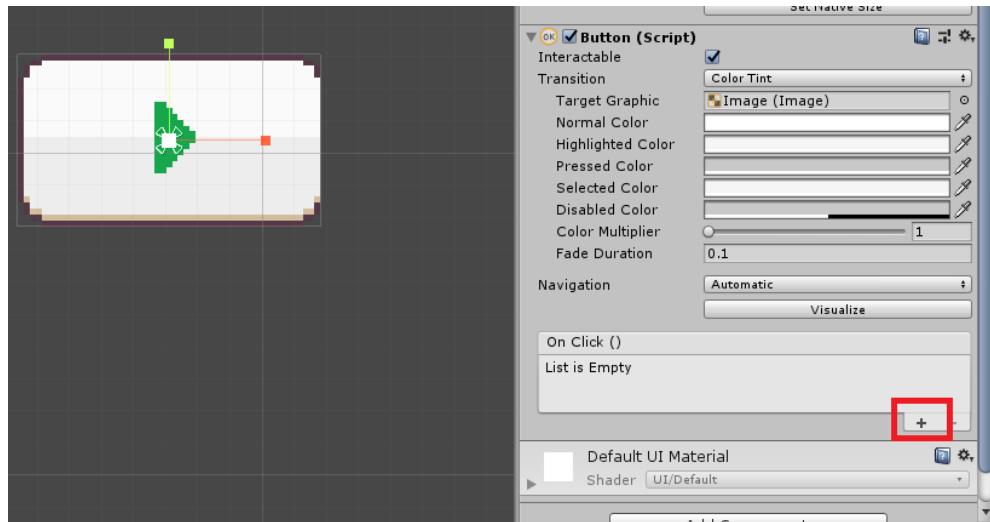
Em seguida, adicionaremos um componente de botão à imagem do botão Restart Button.

7. Selecione o GameObject "Restart Button" na Hierarquia.
8. No Inspector, clique em Add Component.
9. Pesquise por Button e adicione o componente.

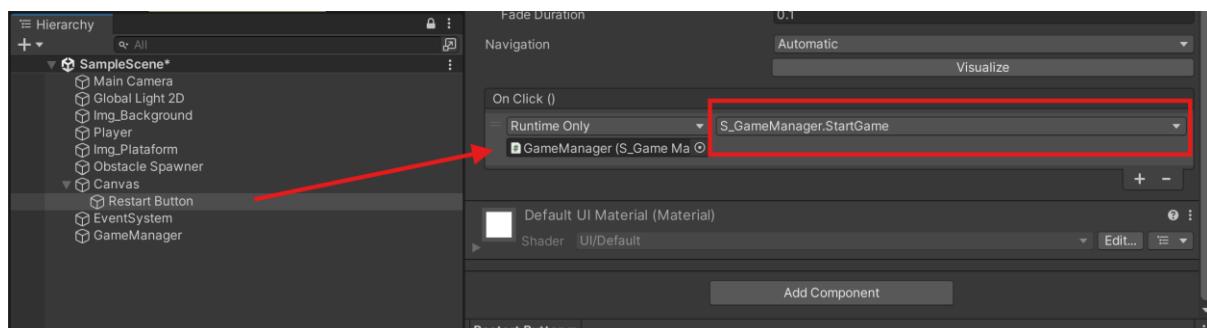


Andrei Inoue Hirata

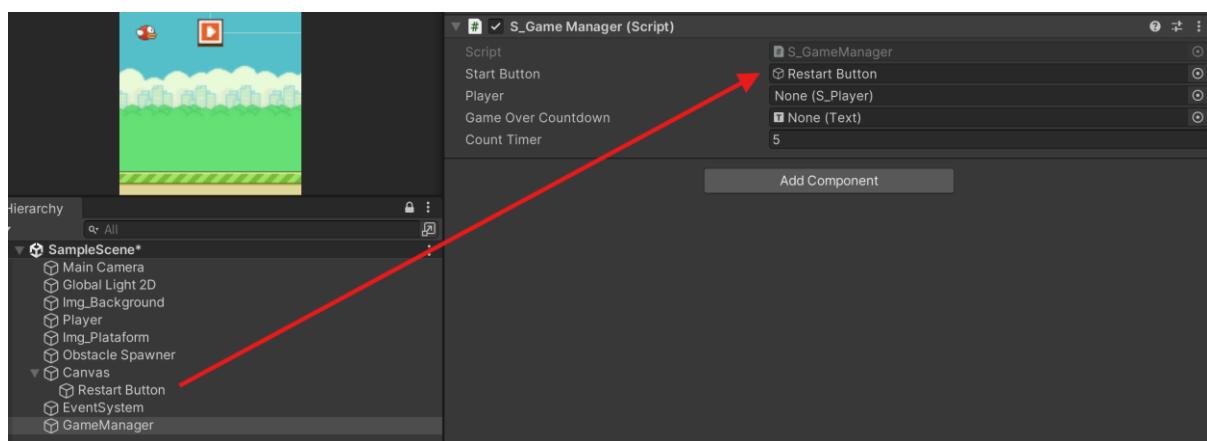
10. Em seguida, dentro do componente Button, navegue até a seção On Click(). Clique no botão "+" no canto inferior direito para adicionar uma nova ação à lista.



11. Em seguida, arraste o GameObject "GameManager" para a caixa logo abaixo do menu suspenso "Runtime Only".
12. Em seguida, clique no menu suspenso "No Function" e altere para GameManager → StartGame().



13. Em seguida, selecione o GameObject "GameManager" e arraste o GameObject "Restart_Button" para o campo "Start Button".



Andrei Inoue Hirata

Morte do Jogador

Agora que temos um botão de início, o último passo é fazer com que nosso jogador morra ao tocar no chão ou nos canos.

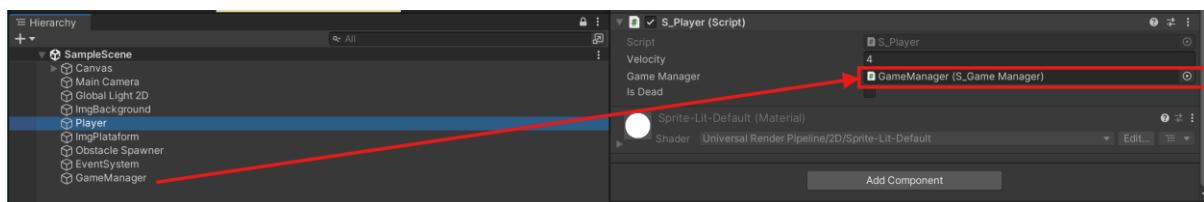
1. Para isso, volte ao script S_GameManager e descomente essa linha

```
//if (player.isDead)
//{
//gameOverCountdown.gameObject.SetActive(true);
//countTimer -= Time.unscaledDeltaTime;
//}
```

2. Depois precisamos voltar ao script Player.cs e adicionar o seguinte código:

```
public S_GameManager gameManager;
public bool isDead = false;
private void OnCollisionEnter2D(Collision2D collision)
{
    isDead = true;
    gameManager.GameOver();
}
```

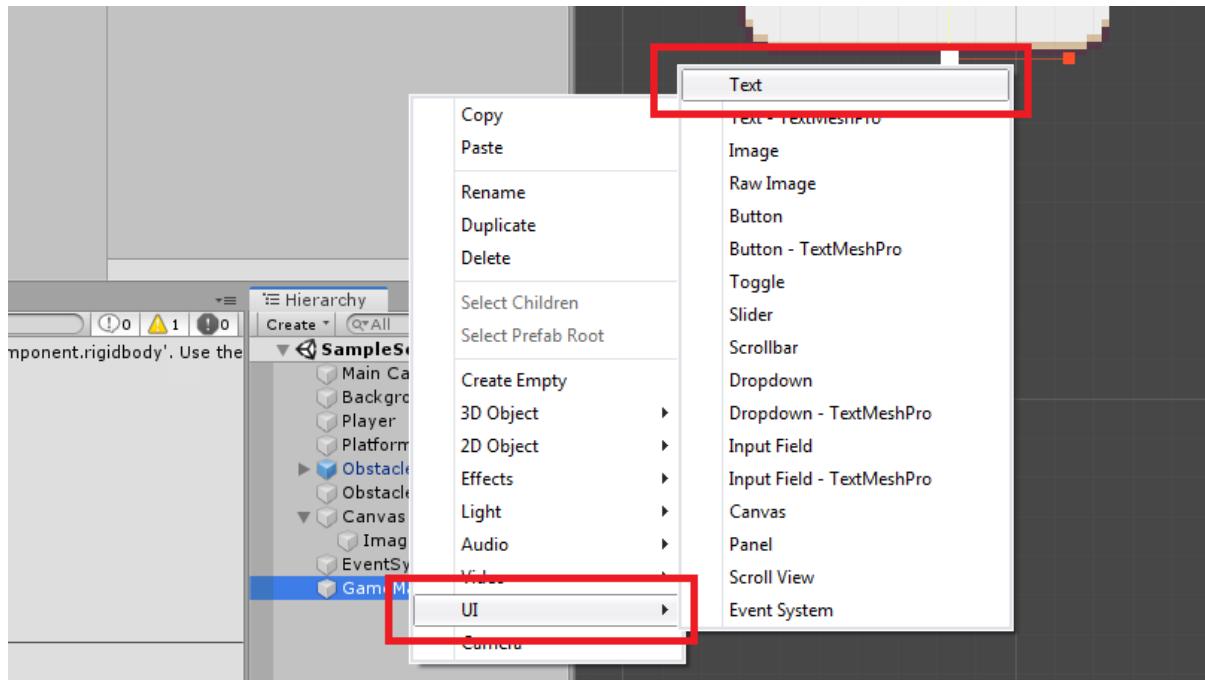
3. Salve o script e volte para o Unity.
4. Selecione o GameObject Player e Arraste para GameManager no componente S_Player.



Por fim, vamos criar um GameObject de Texto e vinculá-lo ao nosso script GameManager.

5. Clique com o botão direito na Hierarquia e selecione UI → Text.
6. Coloque o nome de "TxtCount".

Andrei Inoue Hirata



7. Altere as Configurações do Texto no UI:

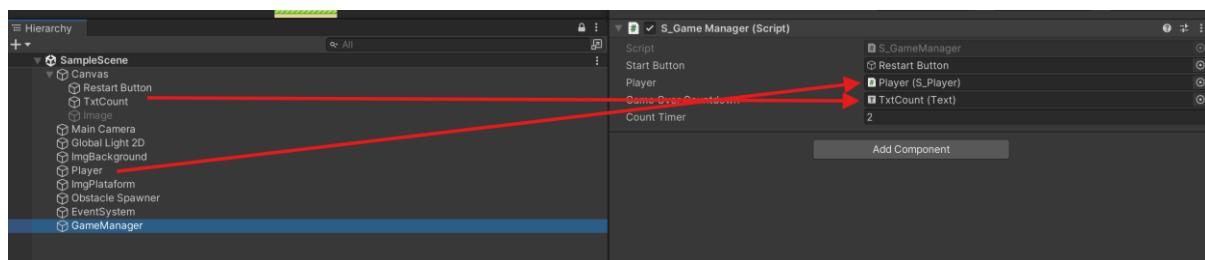
- Alinhe o texto ao centro,
- Marque a opção "Best Fit".
- Defina o tamanho da fonte para 25 e o estilo para negrito (Bold).
- Arraste o texto para cima no eixo Y até 125.65.

8. Adicionando Sombra ao Texto, para isso no Inspector, clique em Add Component. E pesquise por Shadow e adicione ao texto.

Finalizando a Configuração no GameManager

9. Selecione o GameObject "GameManager".

10. Arraste o "Player" e o "TxtCount" para os campos correspondentes no script GameManager.



11. Execute o jogo e aproveite.

Andrei Inoue Hirata

