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

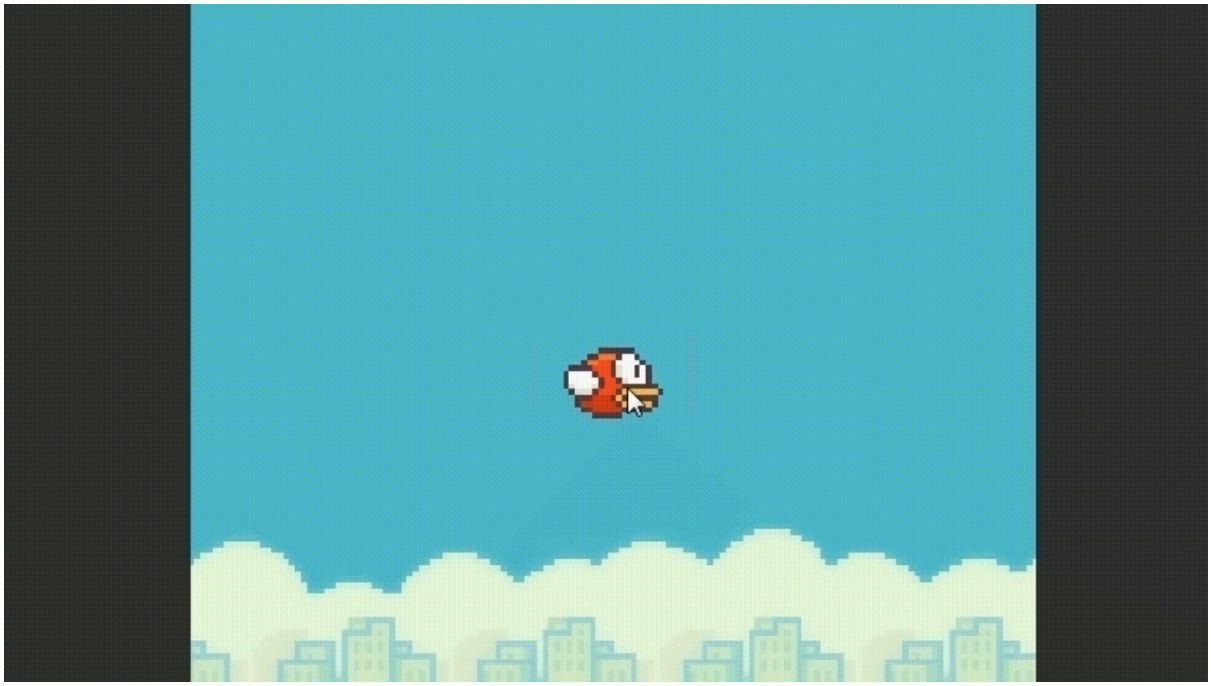
O conceito do jogo é bastante simples. O jogador controla Faby (o pássaro), que se move horizontalmente pela tela enquanto a gravidade o puxa para baixo. O objetivo é evitar que Faby colida com os canos e, a cada cano atravessado com sucesso, o jogador ganha um ponto.

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

Mas nós vamos fazer ainda melhor! Neste tutorial, vamos criar um jogo estilo Flappy Bird em apenas 10 minutos, usando a Unity Game Engine.

Então, sem mais demora, vamos começar! A colaboração acontece online, portanto, a primeira etapa é salvar seu documento no OneDrive.

Andrei Inoue Hirata



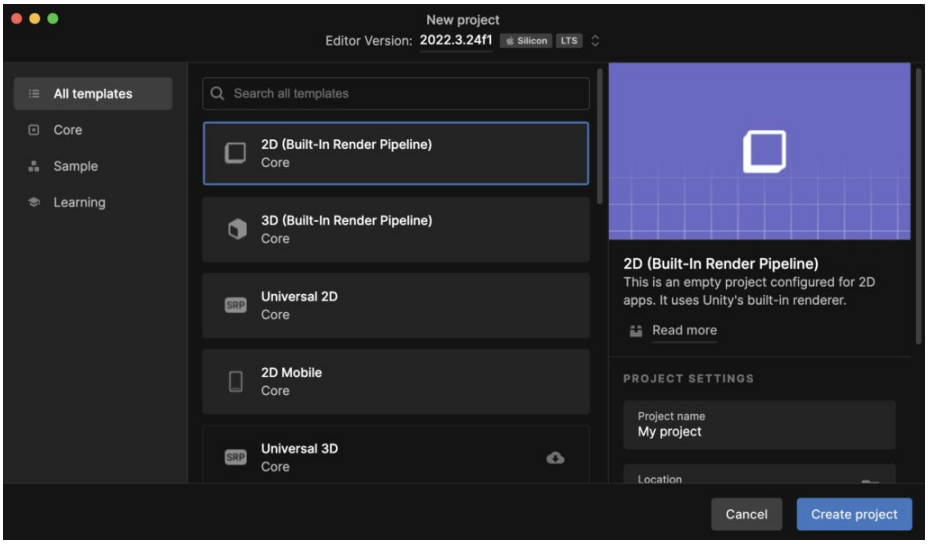
Criando Flappy Bird Usando Unity – Guia Passo a Passo

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.

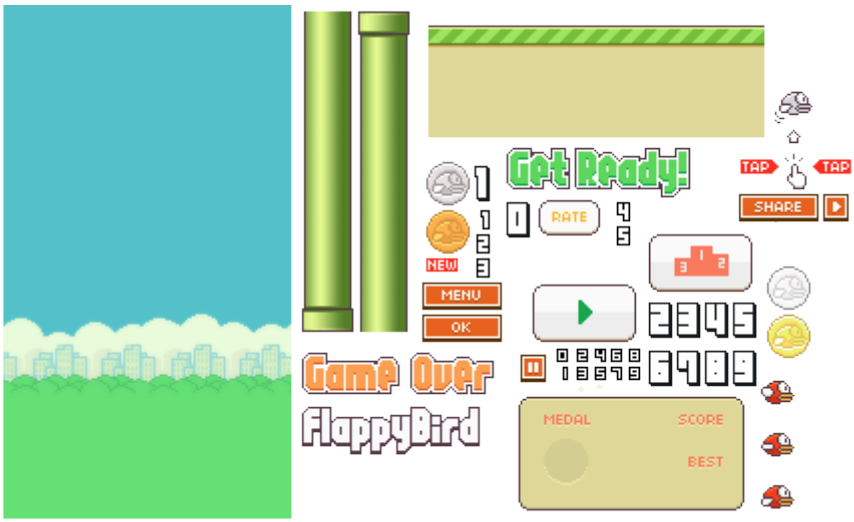
Criando o Projeto na Unity

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



Baixando os Assets

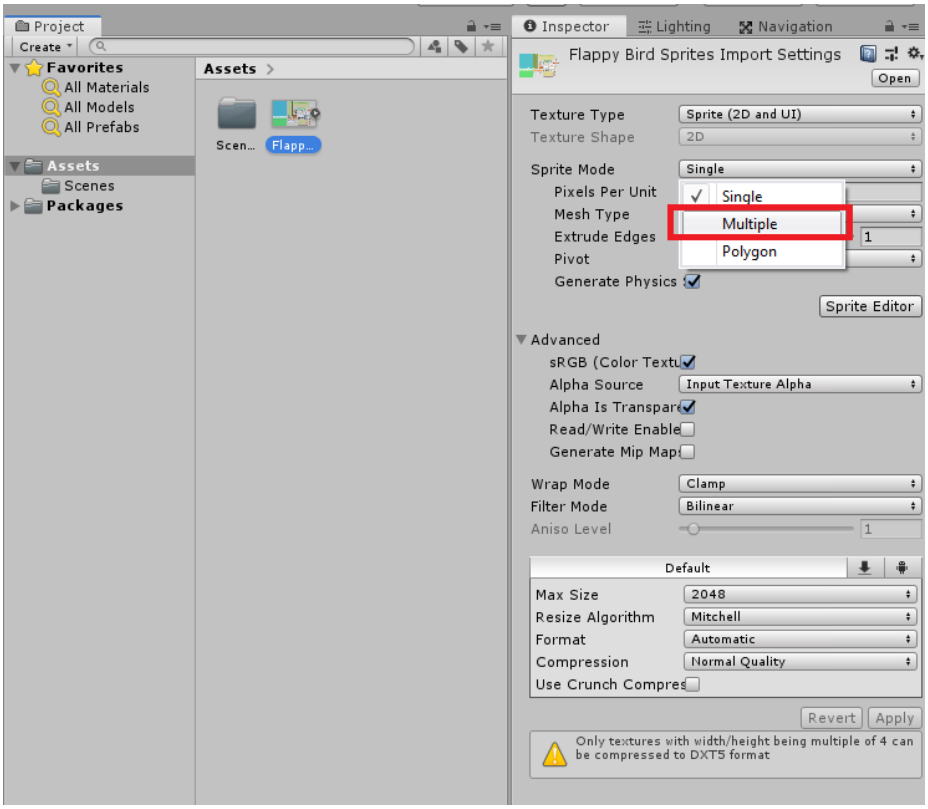
Antes de começarmos a criar o jogo, obviamente precisamos dos assets, como o pássaro, os canos, as interfaces, etc.



Depois de baixar o arquivo, volte para o Unity e importe a imagem.

Depois de importado, selecione o sprite e certifique-se de definir seu Sprite Mode para Multiple, Filter Mode para Point e Compression para None. Isso fará com que seu sprite fique nítido e de alta qualidade.

Não se esqueça de clicar em Apply para salvar as alterações.

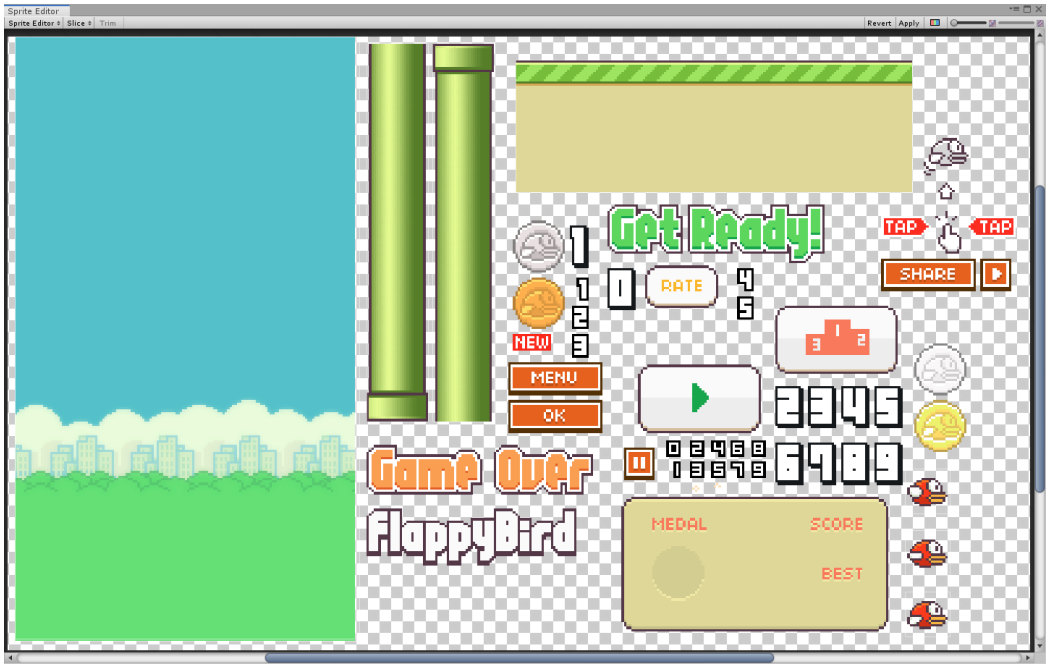


Agora, se você já percebeu, nossos assets estão compilados em um arquivo PNG, que é então convertido em um sprite após ser importado para o Unity.

Precisamos cortar esse sprite usando o Sprite Editor para podermos utilizar os objetos, como o pássaro, os canos, etc.

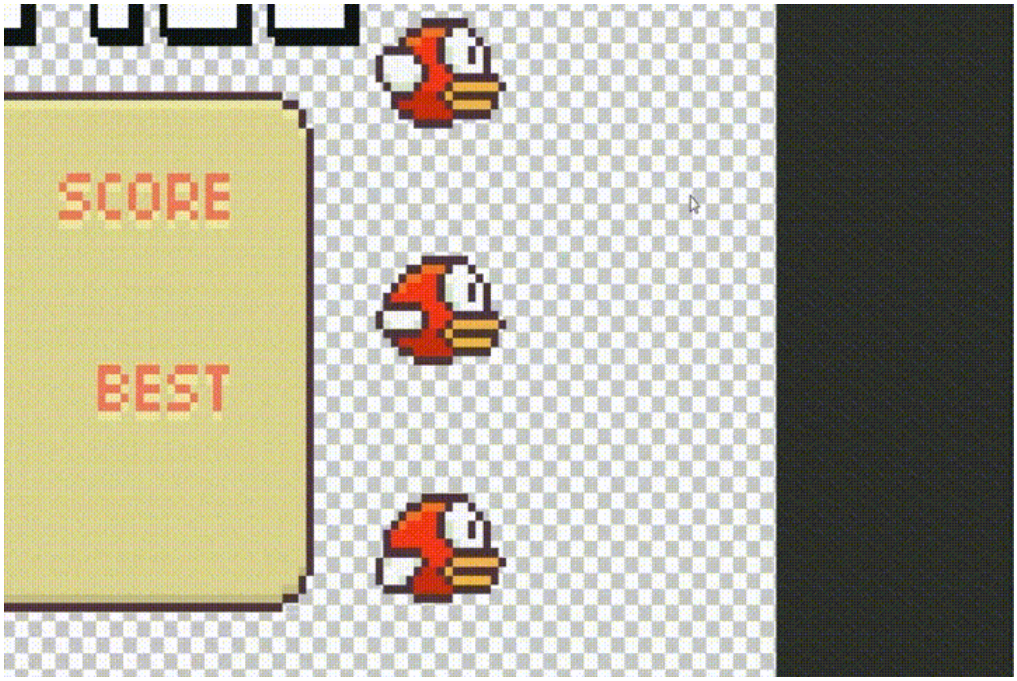
Selecione o sprite novamente e, logo abaixo da opção Generate Physics, clique no botão Sprite Editor.

Isso abrirá a seguinte janela.

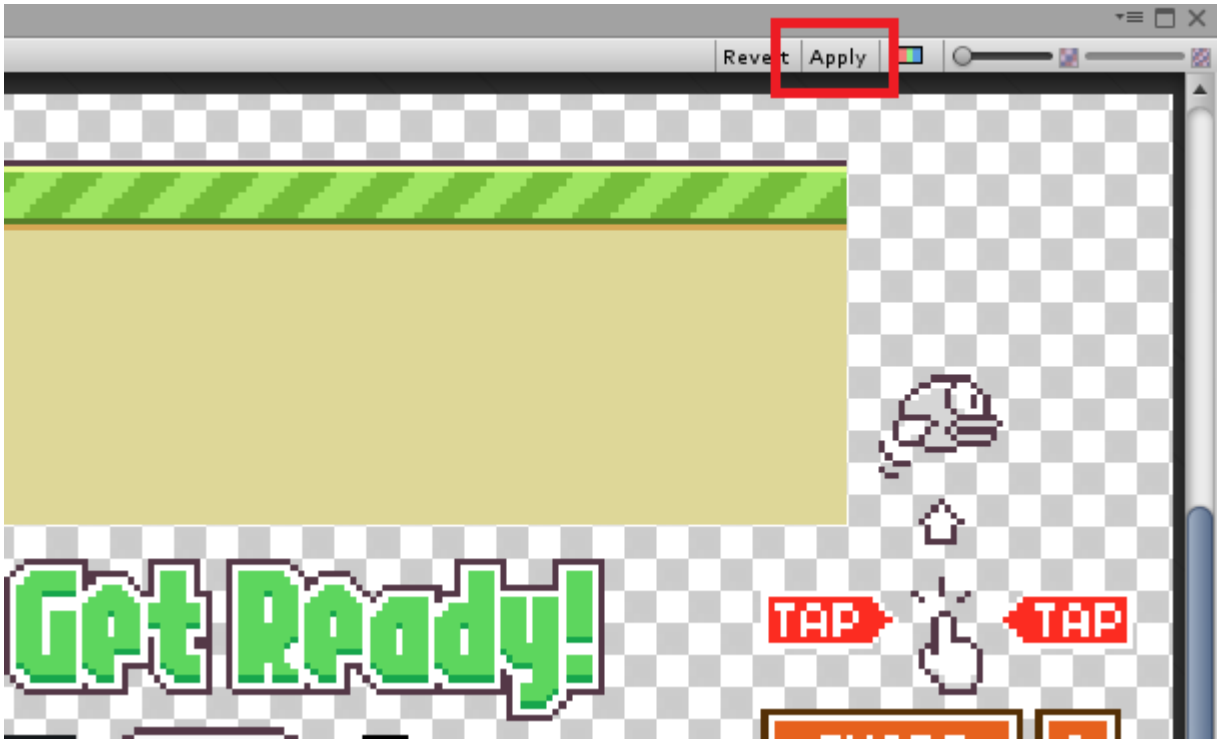


Agora, para criar novos sprites, selecione os objetos arrastando dentro do Sprite Editor.

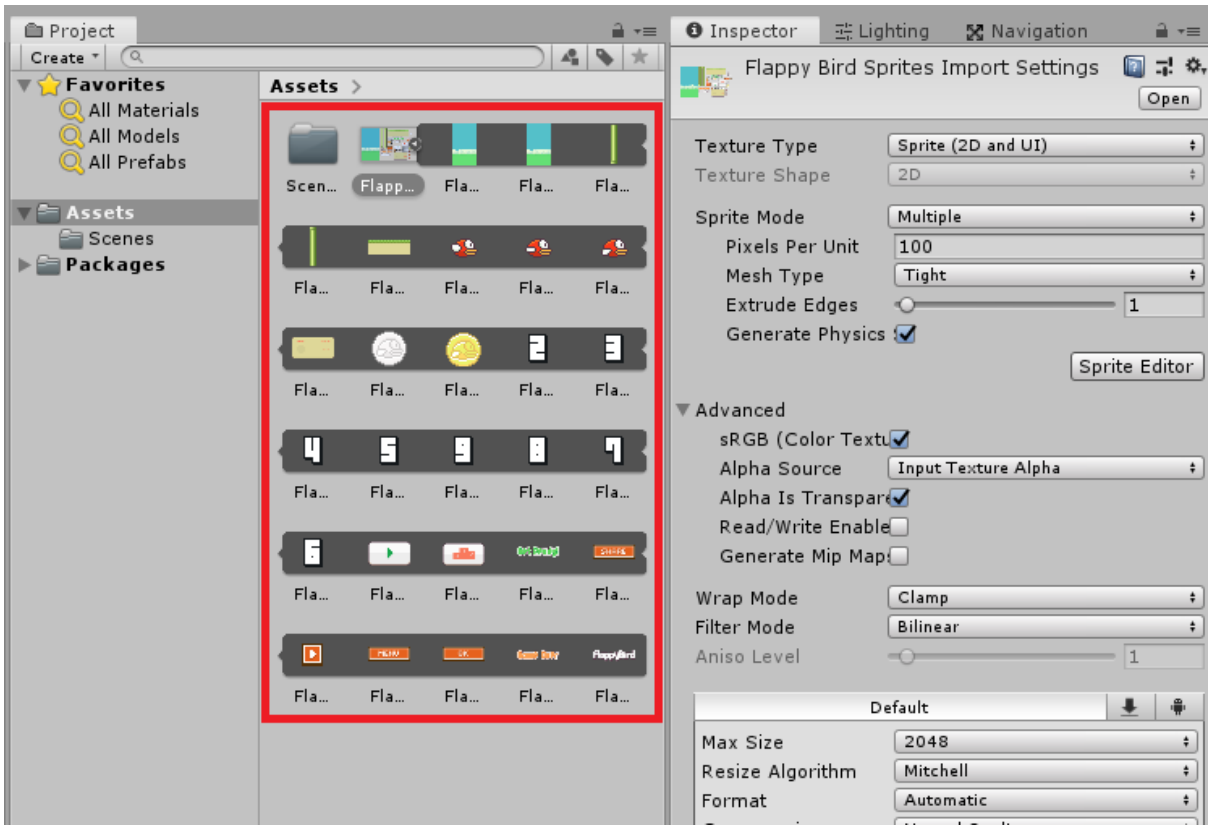
(Imagine que você está usando o Photoshop e selecionando uma imagem.)



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



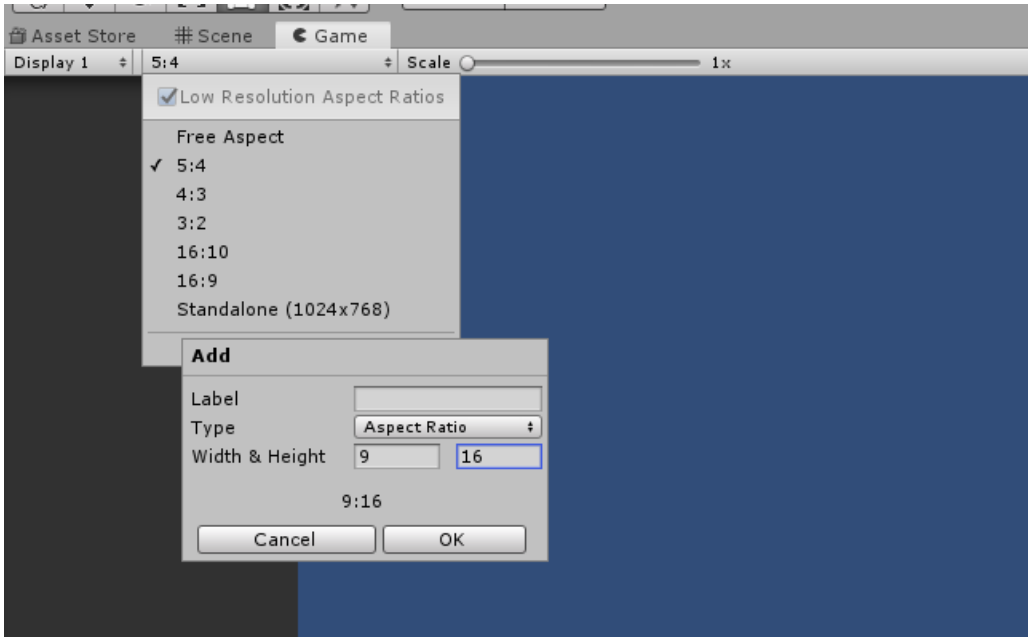
Uma vez concluído com sucesso, você verá todos os sprites cortados dentro do seu asset de sprite do Flappy Bird.



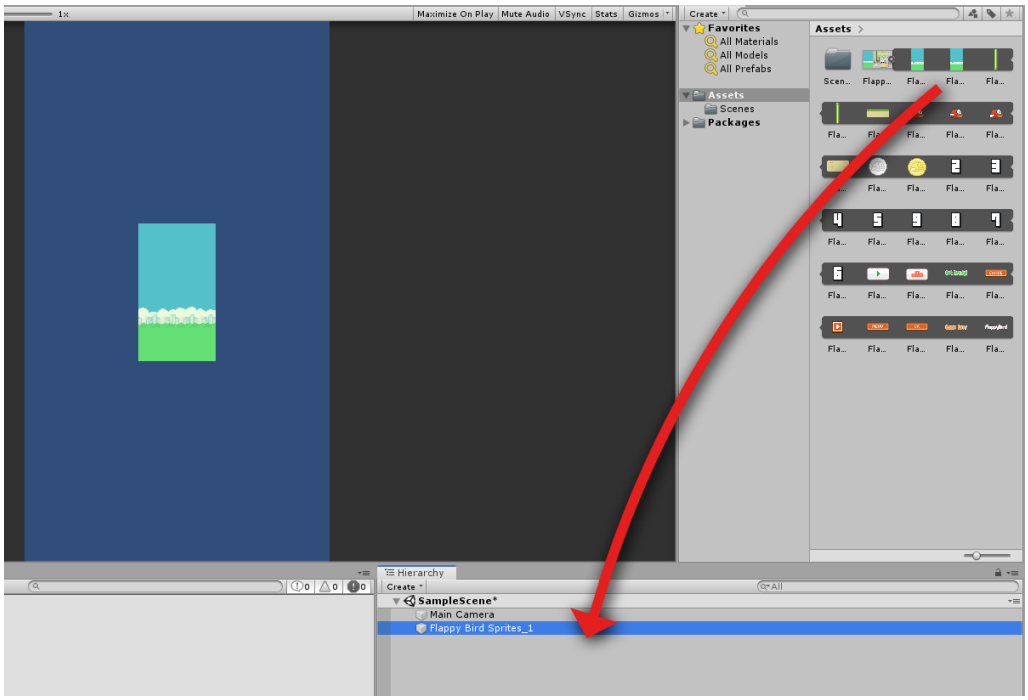
Andrei Inoue Hirata

Configurando o Jogo

Agora que terminamos com nossos assets, vamos iniciar este tutorial ajustando a proporção de tela do nosso jogo para 9:16.

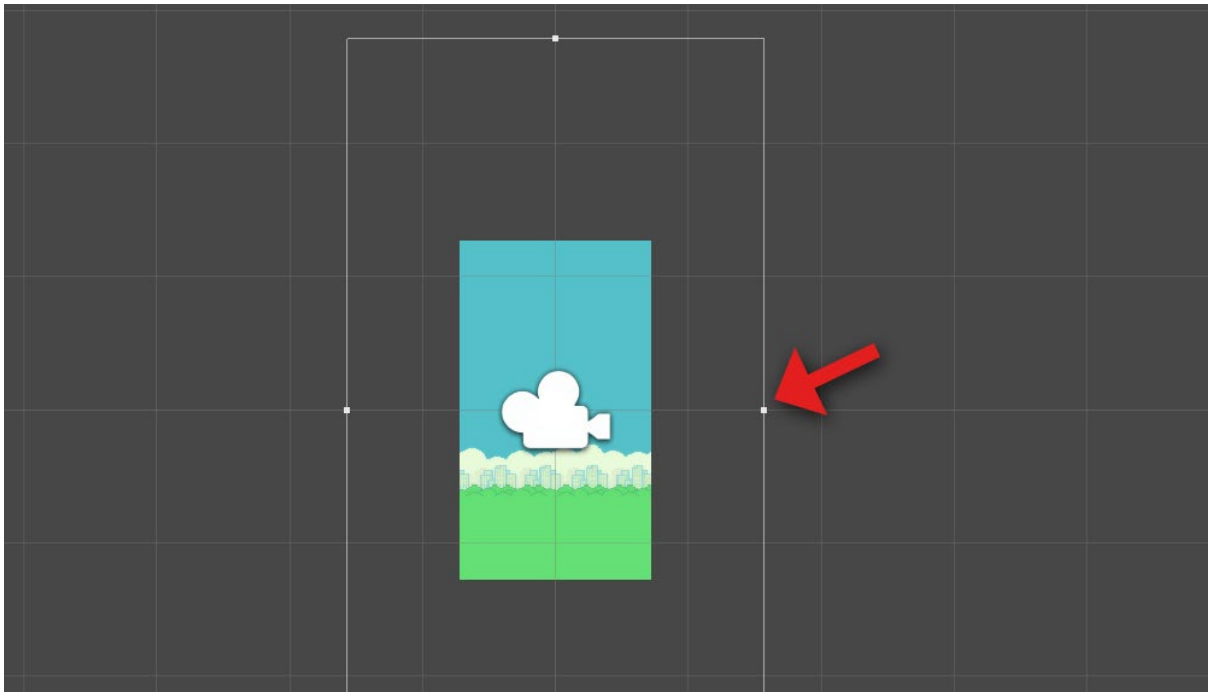


Em seguida, arrastaremos o fundo do jogo para a Hierarquia.



Em seguida, vá para a janela Scene (CTRL + 1), clique na Main Camera dentro da Hierarquia e arraste o ponto branco para baixo até que corresponda ao tamanho do nosso fundo.

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



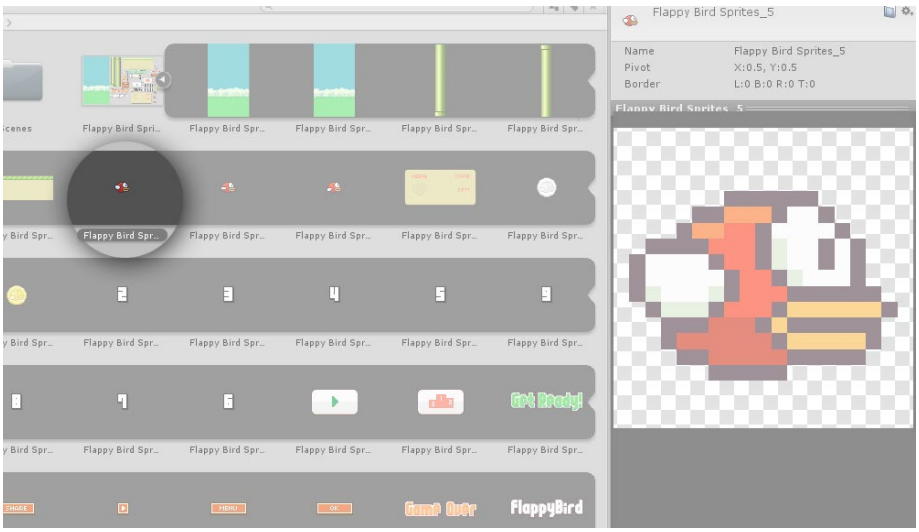
Isso deve deixar a prévia do seu jogo pronta para o jogador.

Criando o Jogador

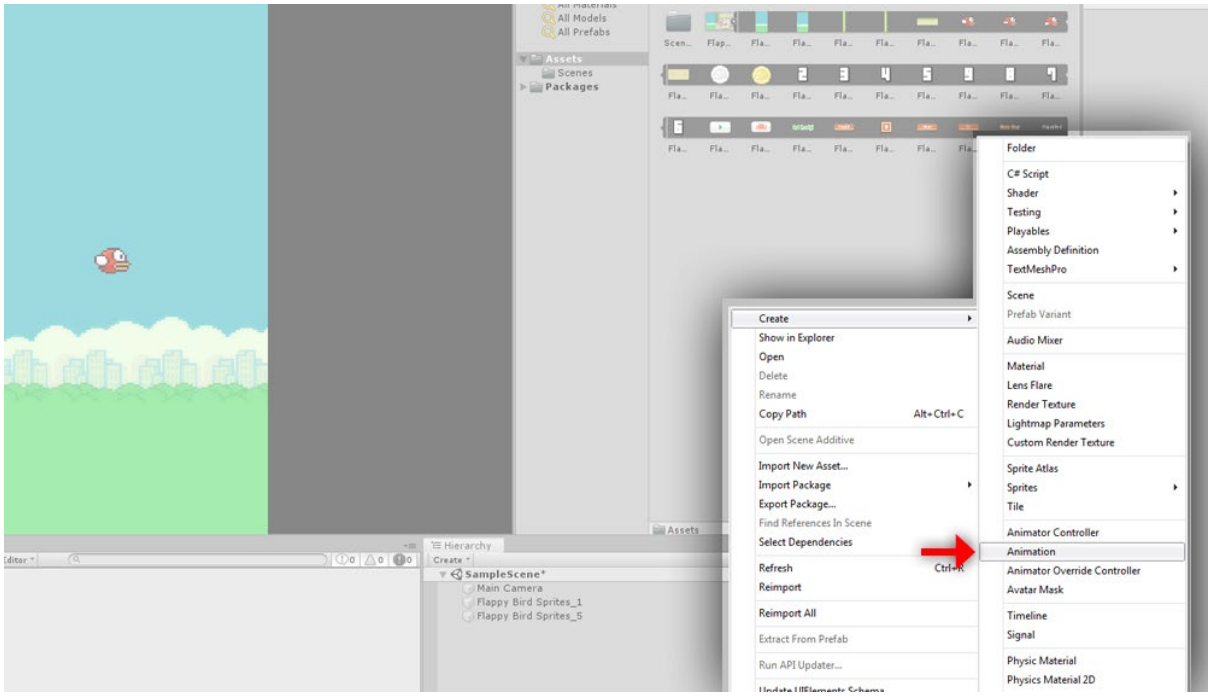
Agora que terminamos com o fundo, o próximo passo é trabalhar no pássaro, que será o nosso jogador.

Volte para os arquivos do projeto, abra o sprite do Flappy Bird novamente e arraste o sprite do pássaro com a asa levantada para a Hierarquia.

Renomeie o GameObject para Player.

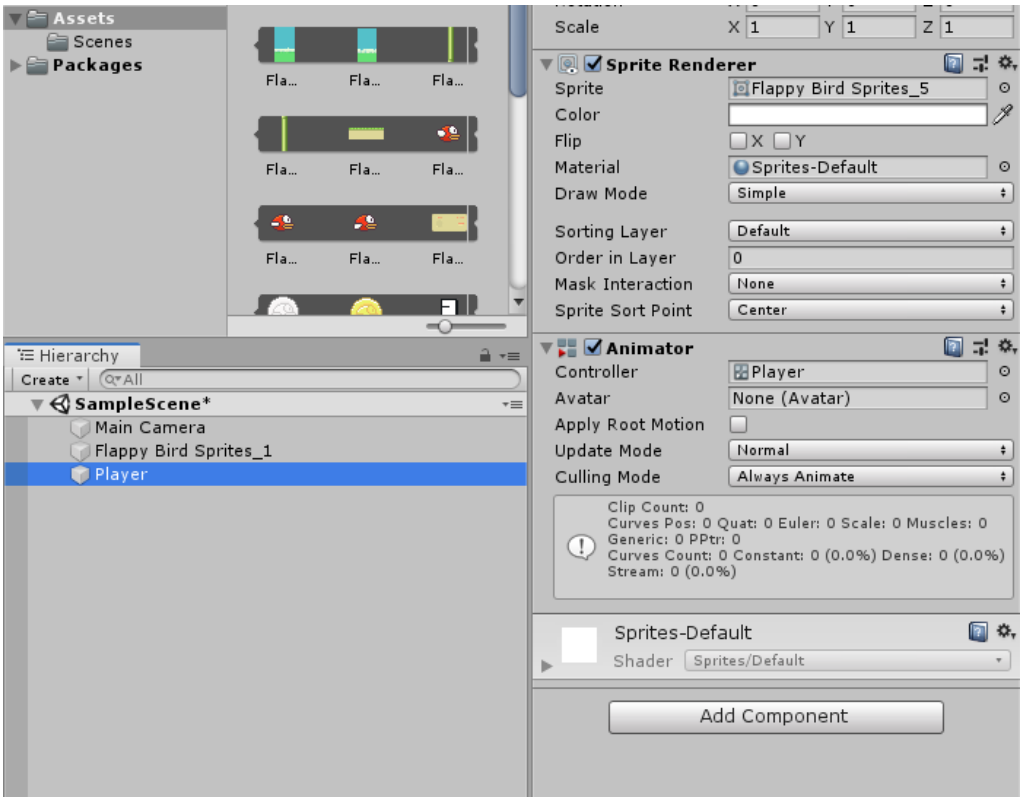


Next, we're going to create a new animation- so go to your project window and right-click- select Create then choose Animation.



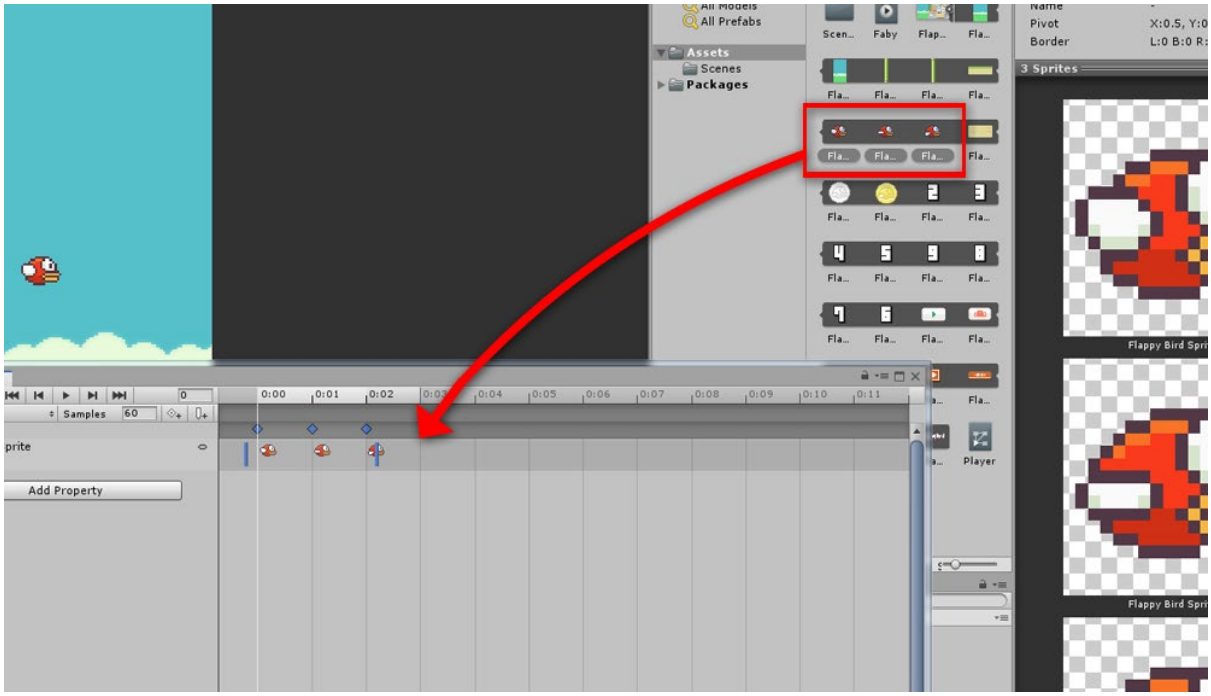
Você pode nomear a animação como quiser. Vamos chamá-la de Faby.

Em seguida, arraste a animação para o GameObject Player. Isso gerará um novo componente Animator para o nosso jogador.



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

Com o Player selecionado, vá para a janela do Projeto, abra o sprite do Flappy Bird e arraste os três sprites do pássaro para a janela de Animação.

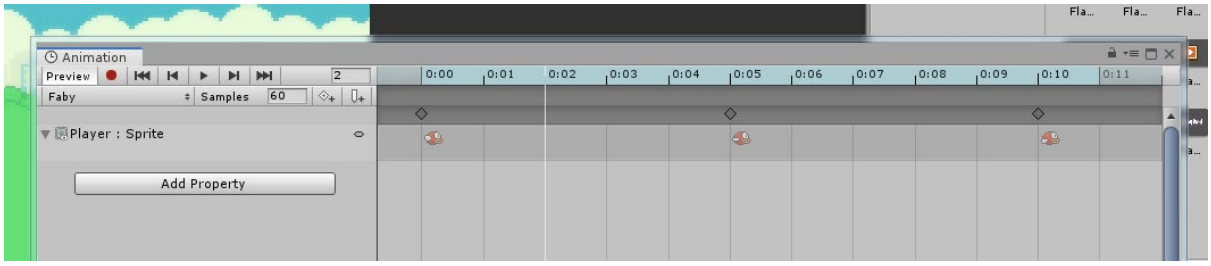


Para melhorar a aparência da animação, ajuste-a arrastando os quadros-chave:

O último keyframe para 0:10.

O segundo keyframe para 0:05.

Deixe o primeiro keyframe em seu lugar original.

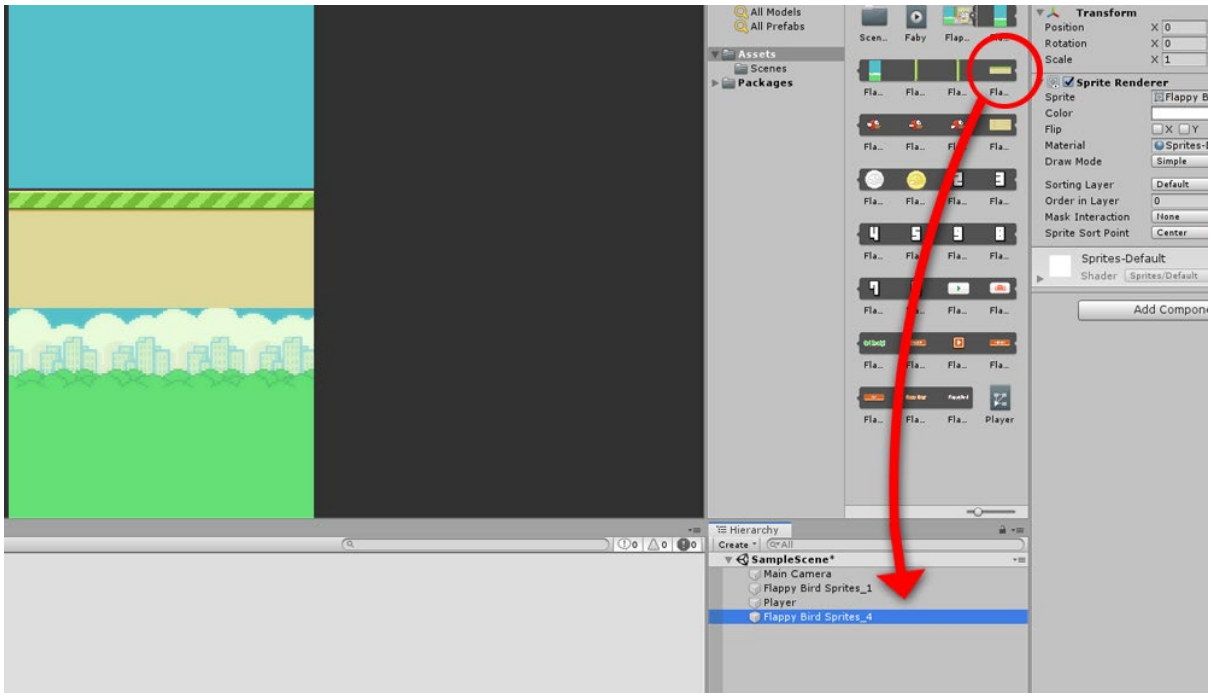


Andrei Inoue Hirata

A Plataforma

O próximo GameObject que precisamos é a plataforma.

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



Agora, obviamente, precisamos arrastar essa plataforma para a parte inferior da tela.

Vá para a janela Scene novamente e arraste a plataforma para baixo.

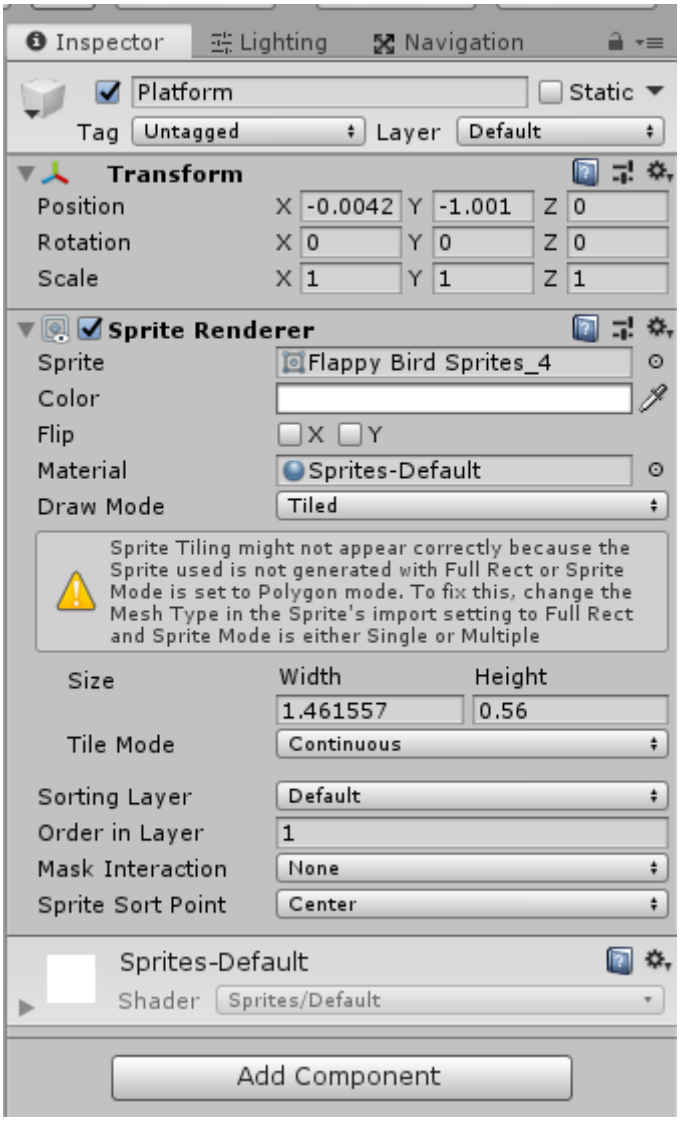
Renomeie o GameObject da plataforma para Platform.

Assim como o fundo, renomeie o fundo para Background.

Selecione o GameObject Platform e altere:

Draw Mode para Tiled.

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:

Selecione o GameObject Platform.

Clique no botão Add Component.

Pesquise por Box Collider 2D e adicione-o.

Certifique-se de que a opção Auto Tiling está definida como true.

Andrei Inoue Hirata

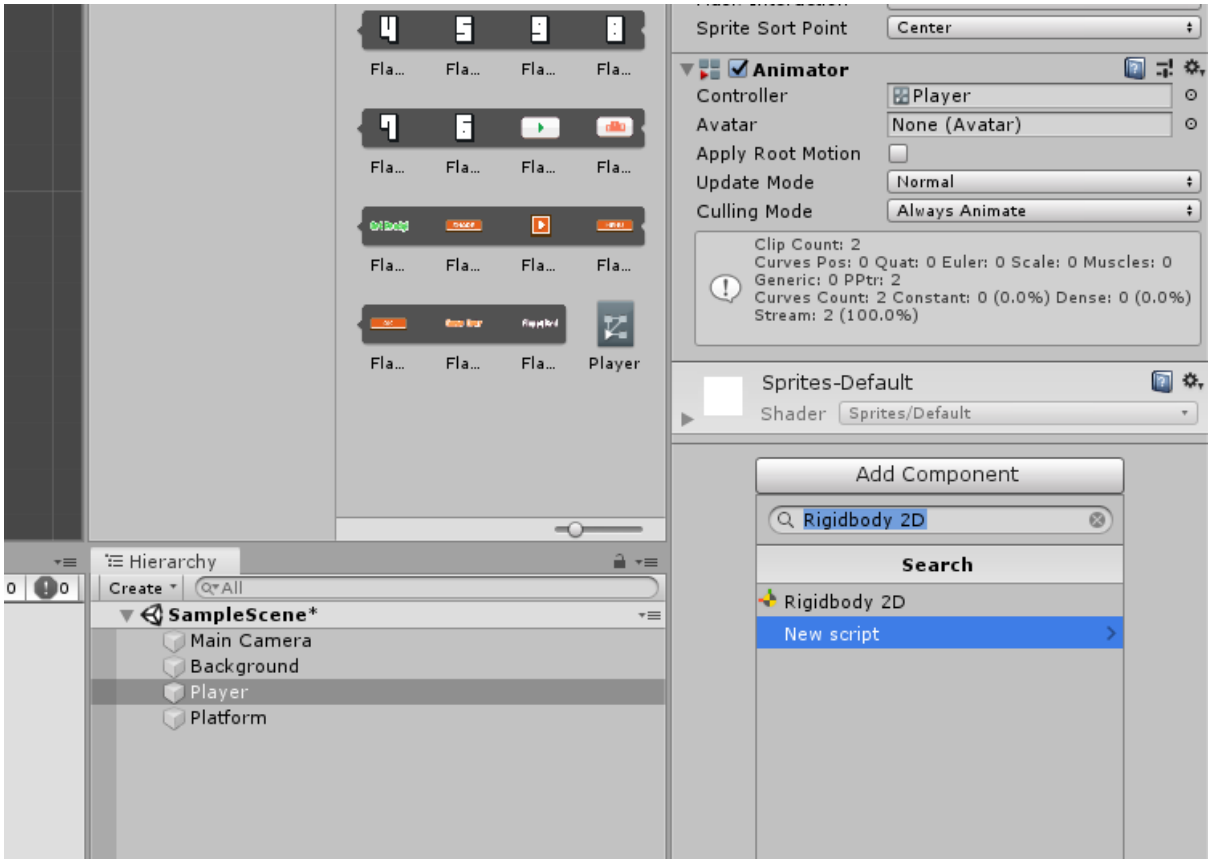
Física do Jogador

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

Com o GameObject Player selecionado, vá para a janela Inspector.

Clique em Add Component.

Pesquise por Rigidbody 2D e adicione-o.



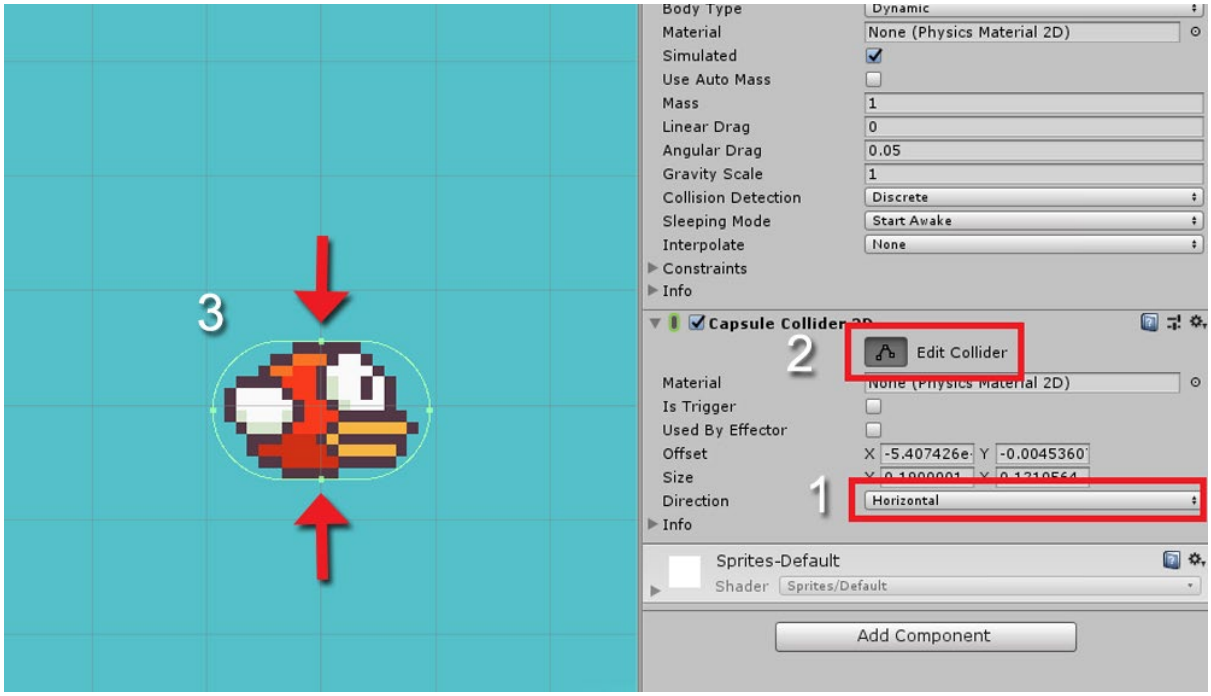
Em seguida, adicionaremos outro componente, o Capsule Collider.

Clique no botão Add Component novamente.

Pesquise por Capsule Collider 2D e adicione-o.

Defina a direção do Capsule Collider 2D para Horizontal.

Clique no botão Edit Collider e ajuste o tamanho do Capsule Collider para que se encaixe perfeitamente no seu jogador.



Player Script

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

Vá para a janela do Projeto.

Clique com o botão direito, selecione Create → C# Script.

Nomeie o script como Player.cs.

Após a criação, arraste o script para o GameObject Player.

Abra o script com o Visual Studio.

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

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

    // Start is called before the first frame update
```

```
    void Start()
    {
        rigidbody = GetComponent<Rigidbody2D>();
    }

    // Update is called once per frame
    void Update()
    {
        if(Input.GetMouseButtonDown(0))
        {
            rigidbody.velocity = Vector2.up * velocity;
        }
    }
}
```

Os Obstáculos

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

Abra os sprites novamente e arraste os dois canos para a cena.

Com os canos selecionados, adicione um componente clicando em Add Component e pesquisando Box Collider 2D.

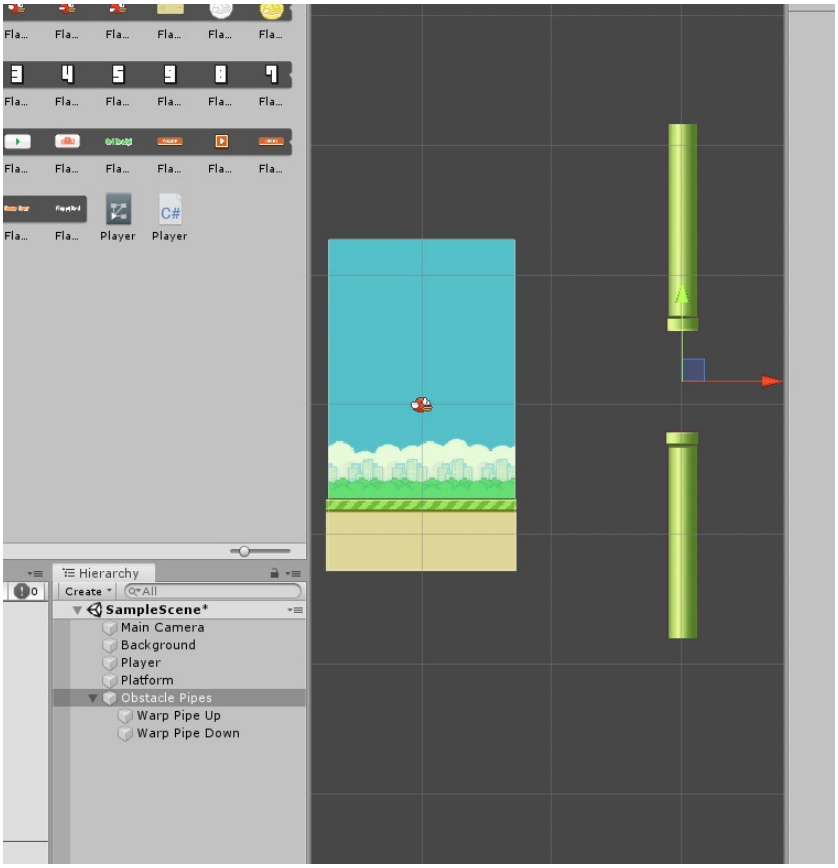
Alinhe os canos verticalmente, como no jogo original.

Crie um GameObject vazio na Hierarquia e renomeie-o para Obstacle Pipes.

Arraste os dois canos para dentro desse GameObject vazio, tornando-o o objeto pai dos canos.

Não parece complicado, certo? 😉

Certifique-se de posicionar o GameObject vazio entre os dois canos, para que a estrutura fique parecida com o jogo original.



Agora, vamos criar um novo script C# chamado Obstacle.cs

Vá para a janela do Projeto.

Clique com o botão direito, selecione Create → C# Script.

Nomeie o script como Obstacle.cs.

Após a criação, arraste o script para o GameObject "Obstacle Pipes".

Abra o script com o Visual Studio para começar a programar a lógica dos obstáculos.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class obstacle : MonoBehaviour
{
    public float speed;
    // Update is called once per frame
    void Update()
    {
        transform.position += ((Vector3.left * speed) * Time.deltaTime);
```

```
    }
}
```

Code language: C# (cs)

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.

Crie um GameObject vazio e renomeie-o para Obstacle Spawner.

Redefina sua posição Transform e arraste a posição X para a direita da tela.

Crie um novo script C# e nomeie-o como Spawner.cs.

Arraste este script para o GameObject "Obstacle Spawner" para vinculá-lo.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class spawner : MonoBehaviour
{
    public float queueTime = 1.5;
    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;
            Destroy(go, 10);
        }
        time += Time.deltaTime;
    }
}
```

Andrei Inoue Hirata

Code language: C# (cs)

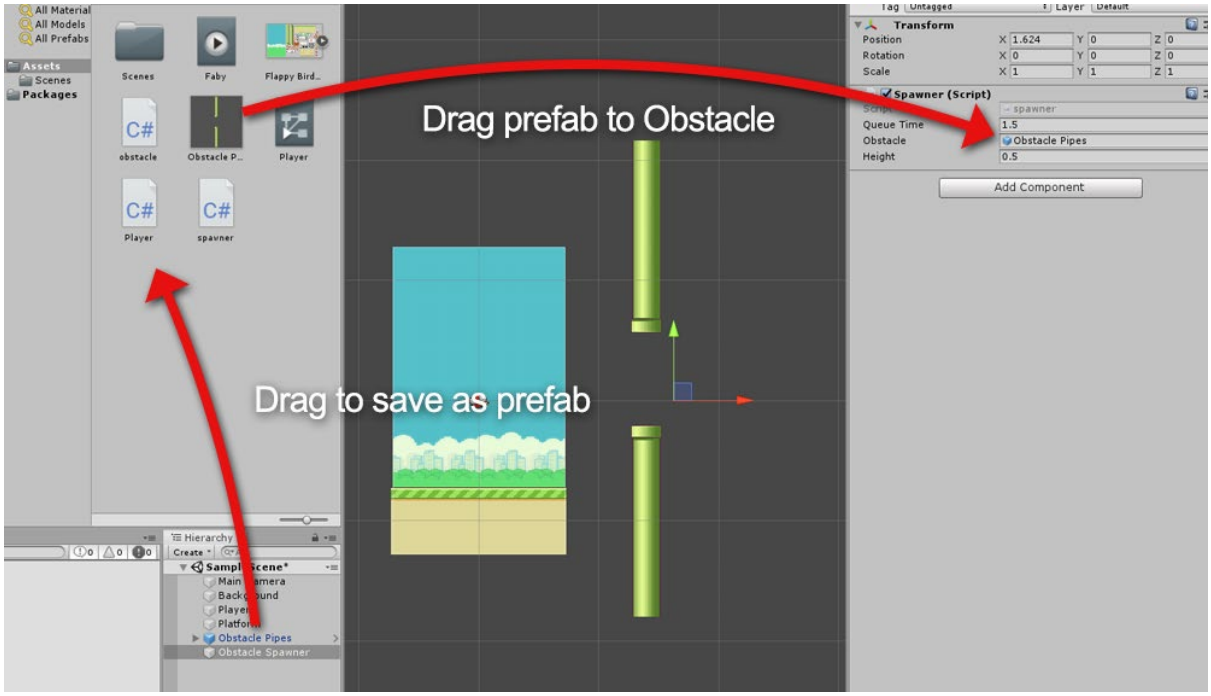
Certifique-se de salvar todos os seus scripts e também a sua cena.

Volte para o Unity e agora vamos salvar os Obstacle Pipes como um GameObject Prefab.

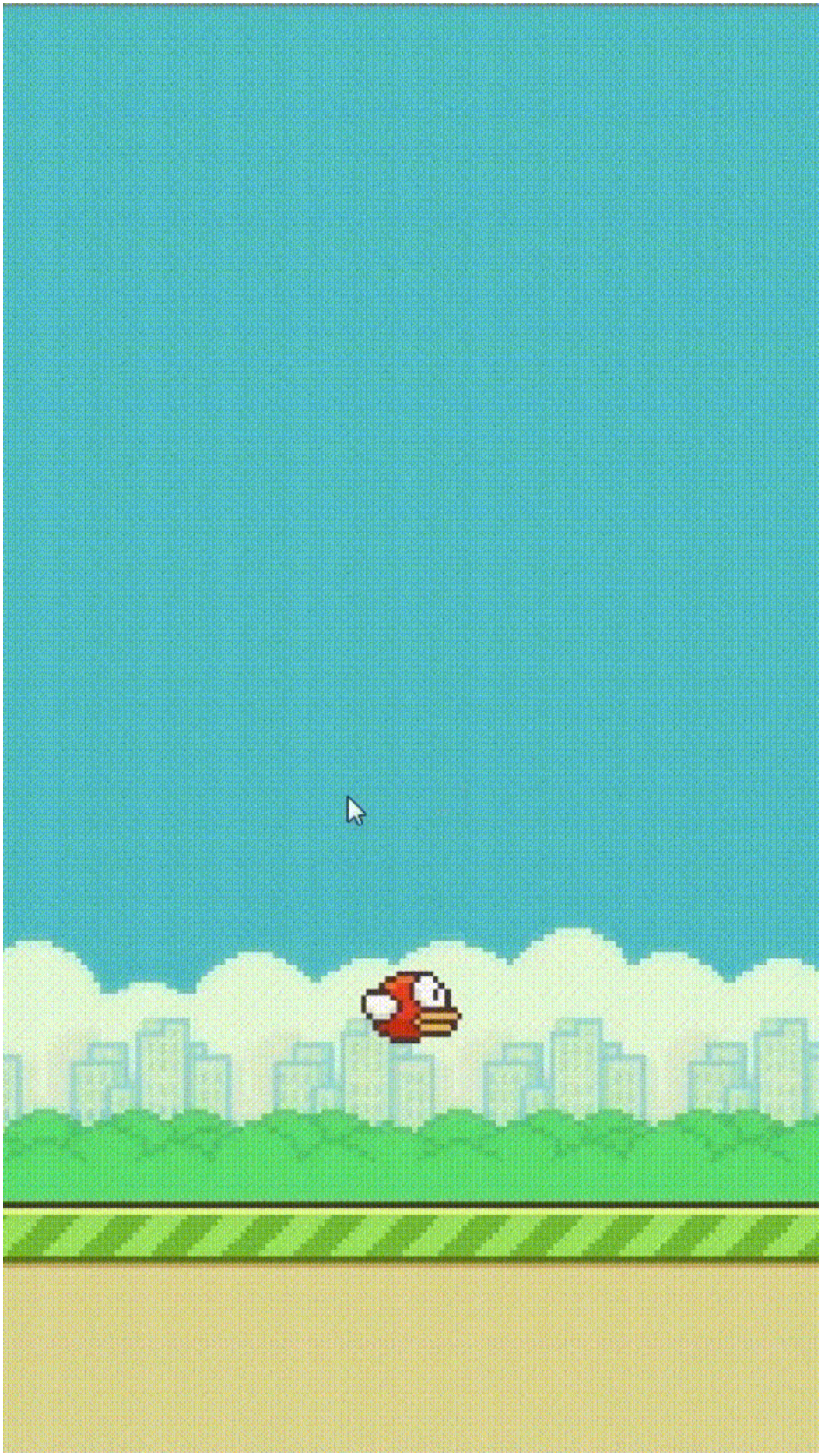
Para isso, arraste o Obstacle Pipes para a pasta "Assets".

Selecione o GameObject "Obstacle Spawner".

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)

Estamos quase terminando!

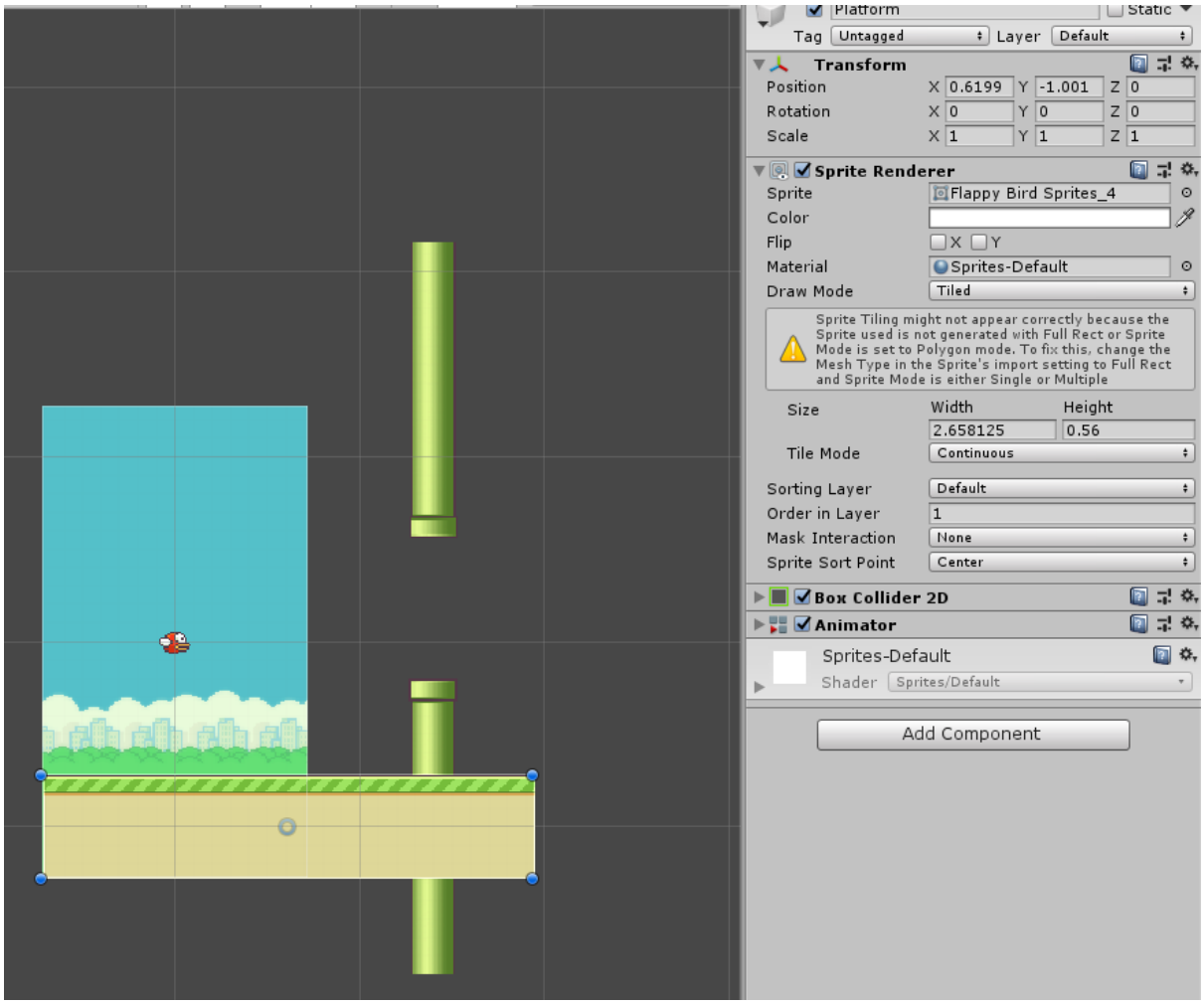
No entanto, se você olhar de perto, verá que a plataforma não está rolando horizontalmente junto com os canos, o que parece estranho.

Para corrigir isso:

Crie uma nova animação e nomeie-a como Platform.

Arraste essa animação para o GameObject "Platform".

Aumente a escala horizontal da plataforma para que o efeito de rolagem fique mais fluido.

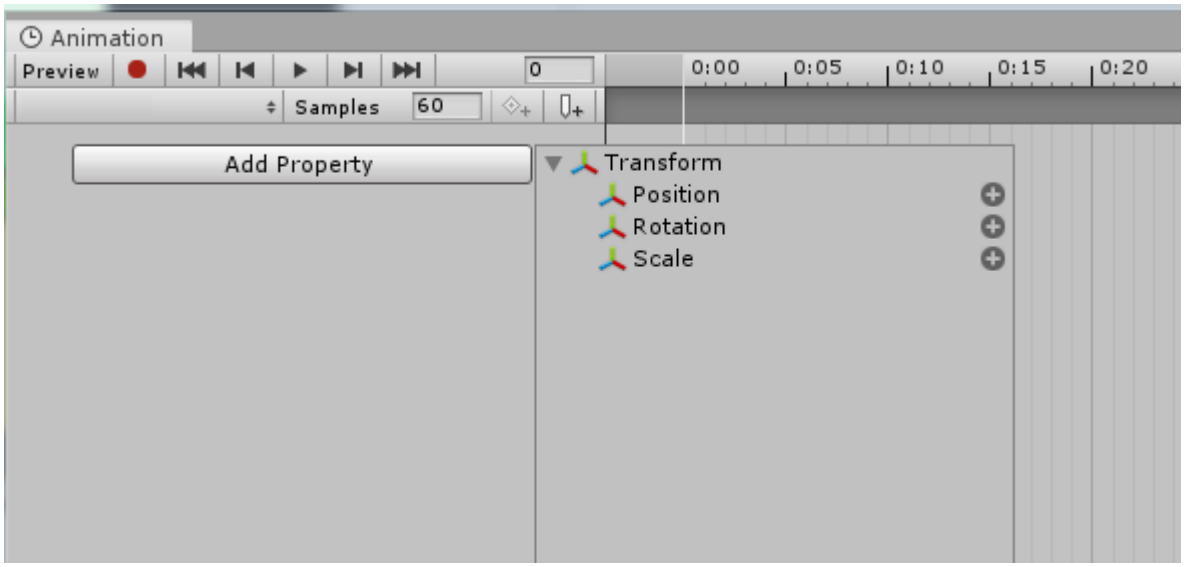


Em seguida, abra a janela de Animação pressionando CTRL + 6.

Com o GameObject "Platform" selecionado,

Clique no botão Add Property na janela de Animação.

Selecione Transform → Position.

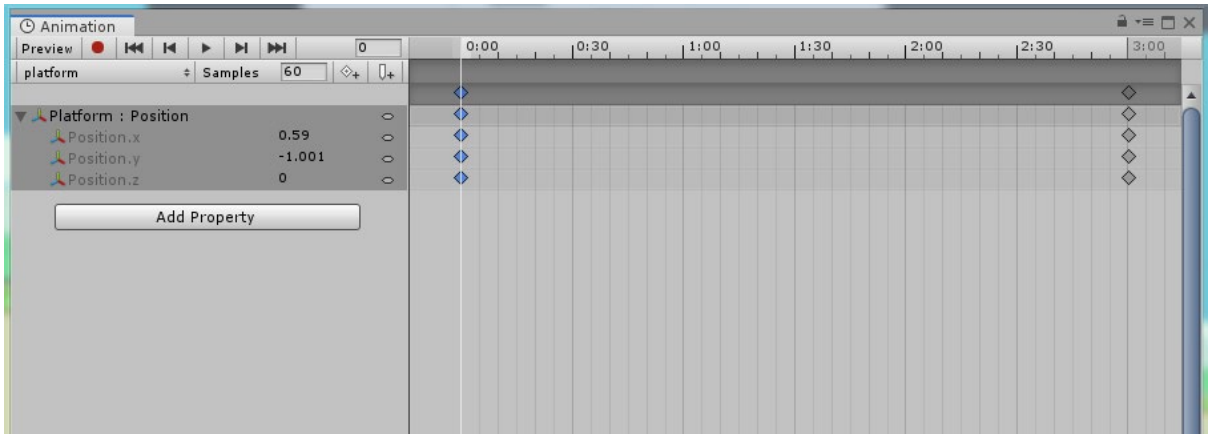


Para criar um keyframe:

Clique no botão ao lado do campo numérico para adicionar um keyframe.

Em aproximadamente 3:00 segundos, crie outro keyframe.

Desta vez, ajuste o valor de X para que a plataforma deslize horizontalmente.



Por exemplo:

No primeiro keyframe, defina a posição X como 0.59.

No segundo keyframe, defina a posição X como -0.59.

Isso fará com que a plataforma role da posição 0.59 para -0.59 a cada 3 segundos.

Andrei Inoue Hirata

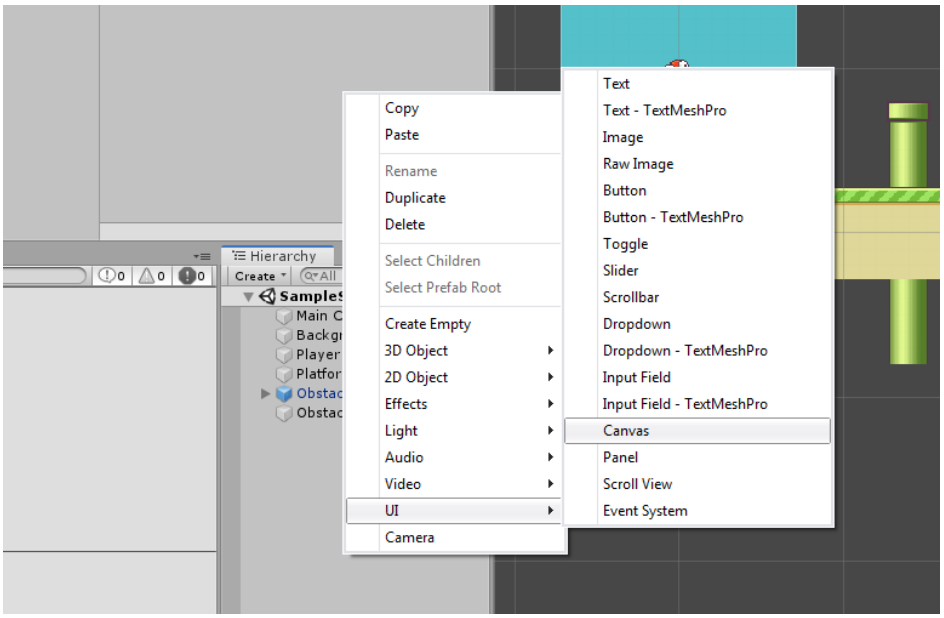
Interface do Usuário (UI)

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

Para começar:

Crie um novo Canvas:

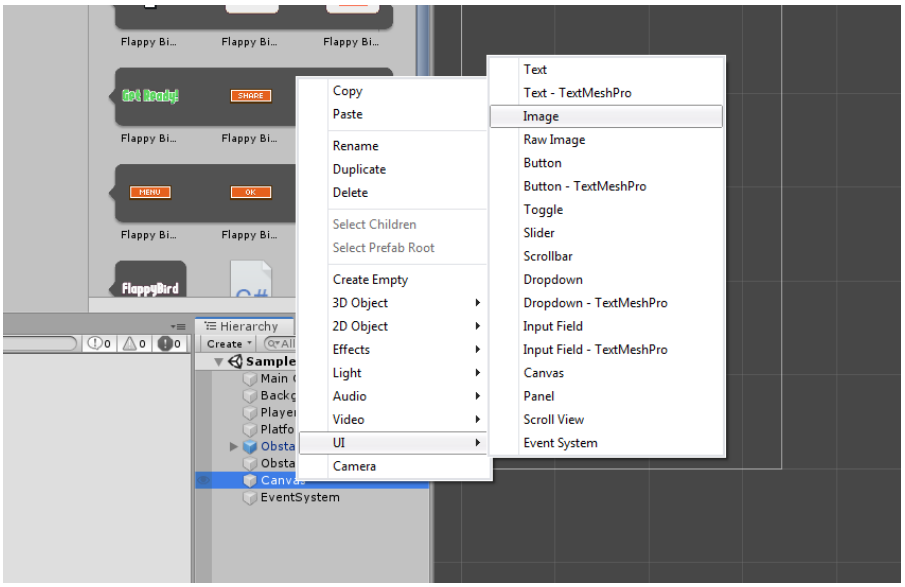
Vá para a Hierarquia, clique com o botão direito, selecione UI → Canvas.



Em seguida:

Clique com o botão direito no GameObject Canvas que você acabou de criar.

Selecione UI → Image.

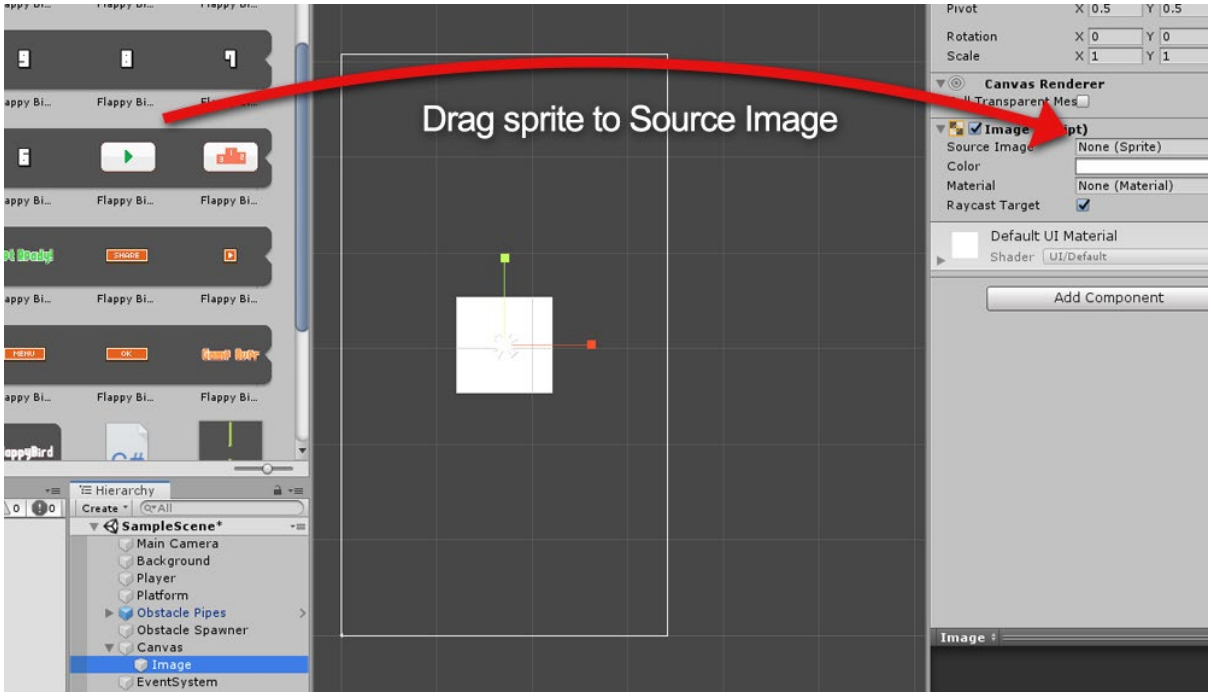


Renomeie este GameObject de Imagem para "Restart Button".

Abra o asset do Flappy Bird Sprite novamente.

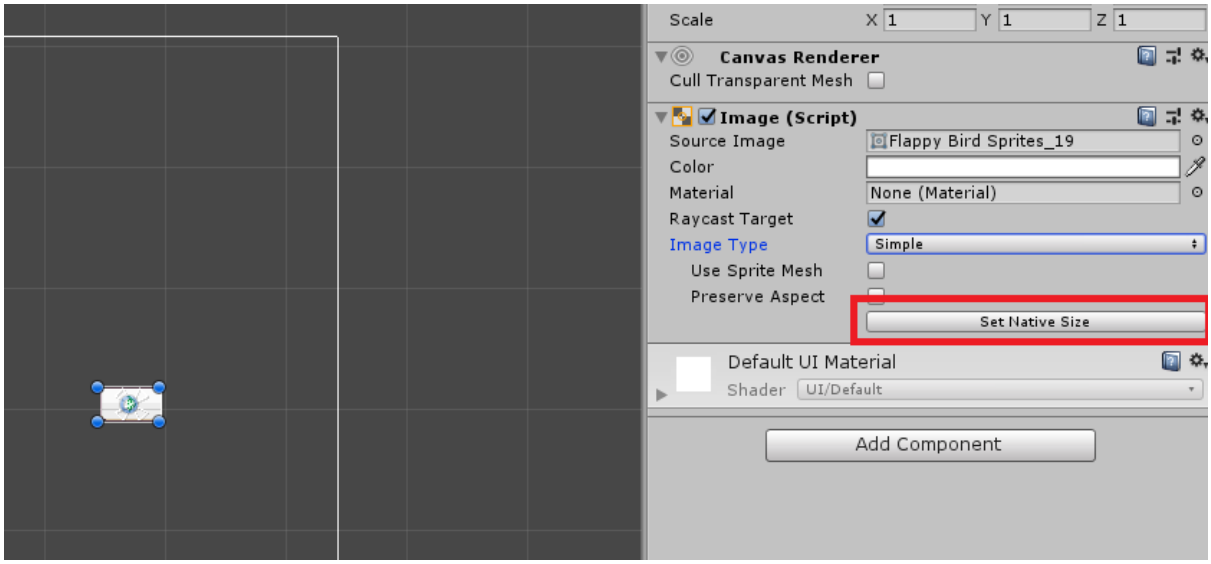
Procure pelo botão de play.

Arraste essa imagem para a propriedade "Source Image" do GameObject Restart Button.



Depois de aplicar a imagem:

Clique no botão Set Native Size para ajustar o tamanho original da imagem.



Se desejar, você pode aumentar o tamanho da imagem para o tamanho desejado.

No nosso caso, escalaremos a imagem para 1.85 nos eixos X, Y e Z.

Gerenciador do Jogo (Game Manager)

Criando o Gerenciador do Jogo (Game Manager)

Agora, vamos criar um script para gerenciar o jogo. Ele deve:

Congelar o jogo no início.

Descongelar quando o jogador clicar no botão de início.

Congelar o jogo novamente quando o jogador morrer e iniciar uma contagem regressiva para reiniciar.

Passos:

Crie um novo GameObject e nomeie-o como GameManager.

Crie um novo script C# e nomeie-o como GameManager.cs.

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

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEditor.SceneManagement;
public class GameManager : MonoBehaviour
{
    public GameObject startButton;
    public 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;
    }
    public void GameOver()
    {
        Time.timeScale = 0;
    }
    public void RestartGame()
    {
        EditorSceneManager.LoadScene(0);
    }
}
```

Code language: C# (cs)

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.

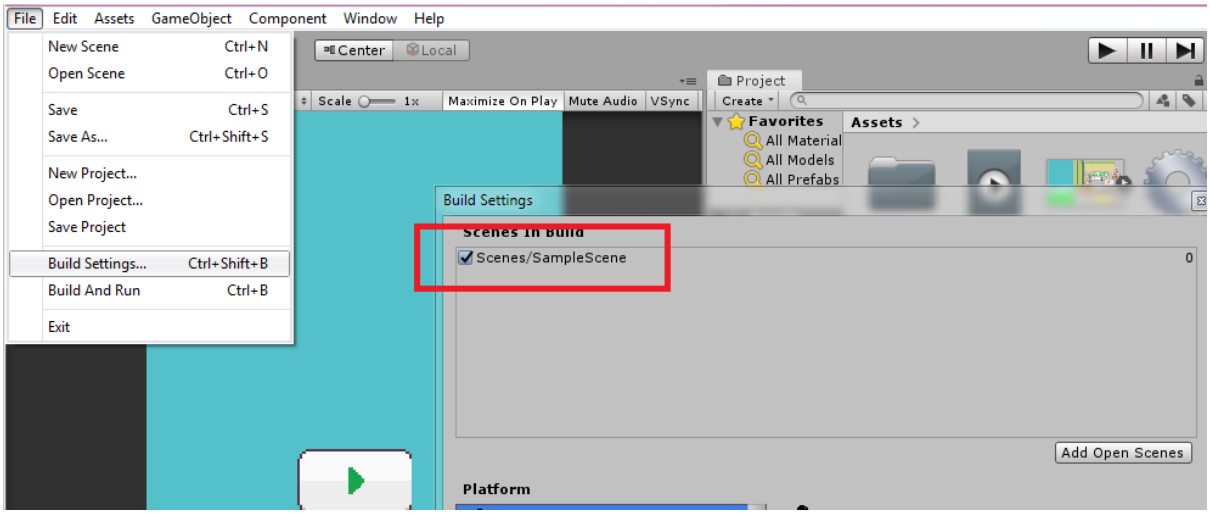
Passos para adicionar a cena ao Build Settings:

Vá para File → Build Settings.

Na janela que abrir, clique em Add Open Scenes para adicionar sua cena atual.

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. 🚀



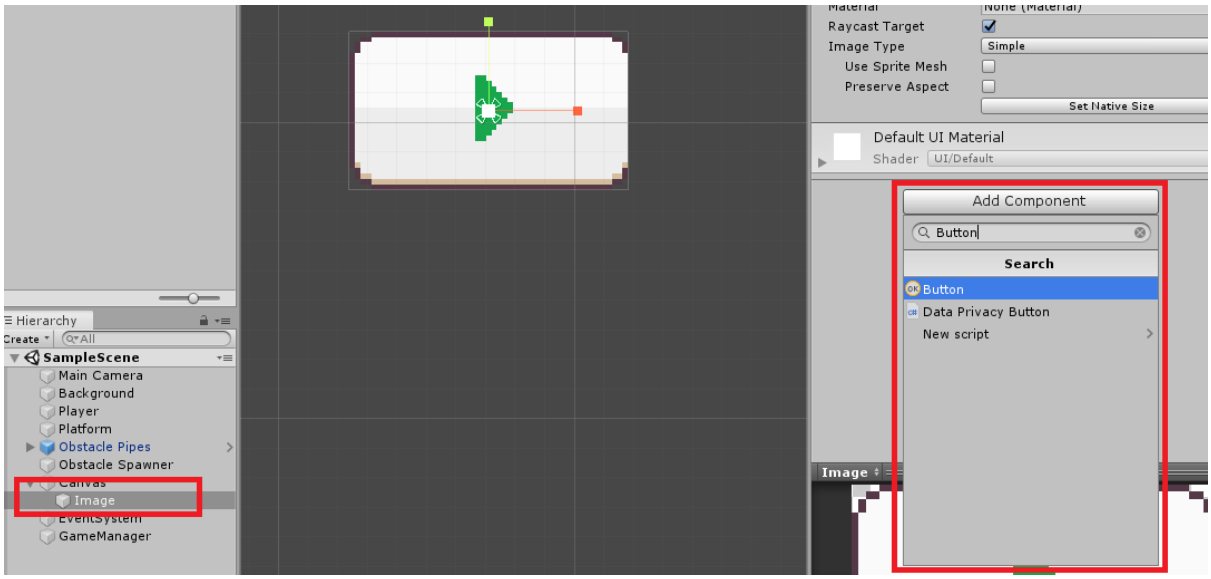
Em seguida, adicionaremos um componente de botão à imagem do botão Play.

1. **Selecione o GameObject "Play Button"** na Hierarquia.

2. No **Inspector**, clique em **Add Component**.

3. Pesquise por **Button** e adicione o componente.

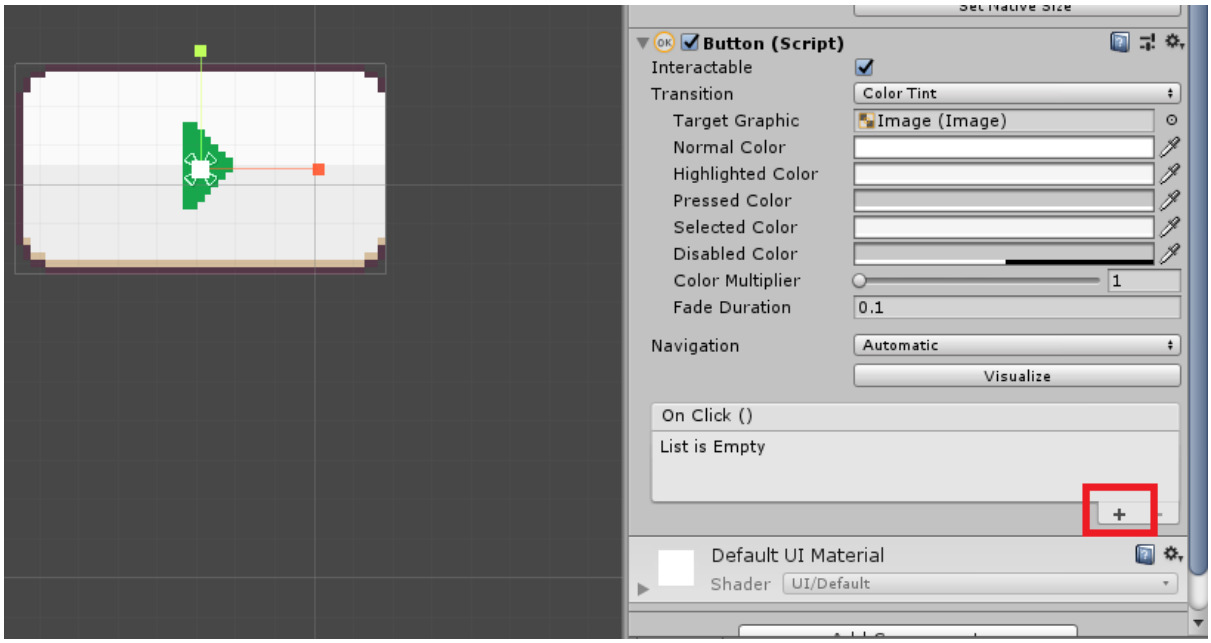
Clique no GameObject de Imagem, clique em Add Component e pesquise por Button.



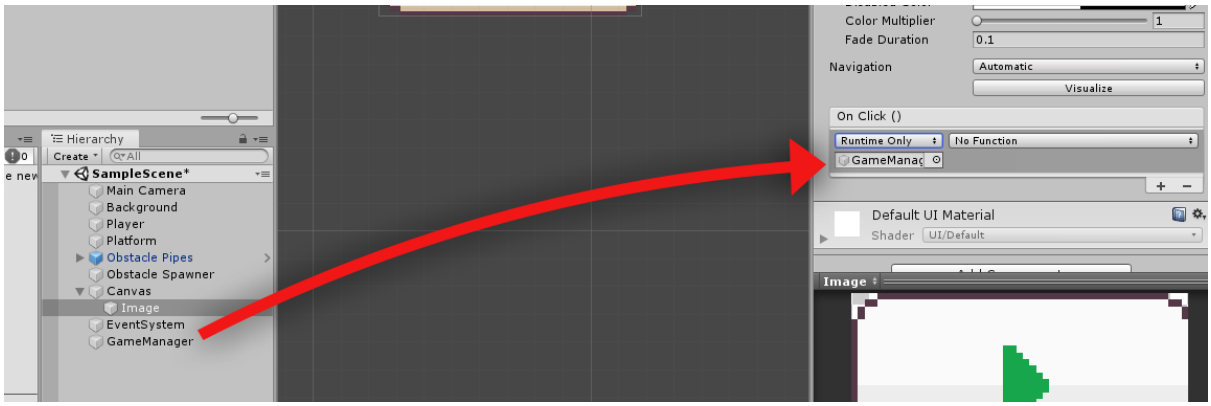
Em seguida, dentro do componente Button:

Navegue até a seção On Click().

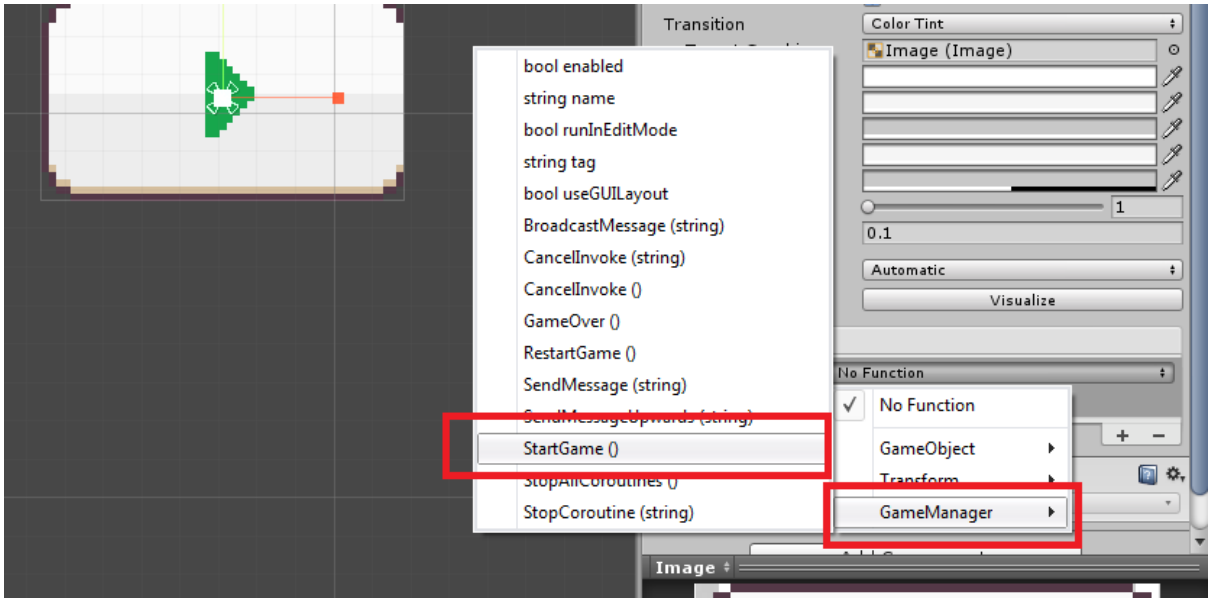
Clique no botão de "+" no canto inferior direito para adicionar uma nova ação à lista.



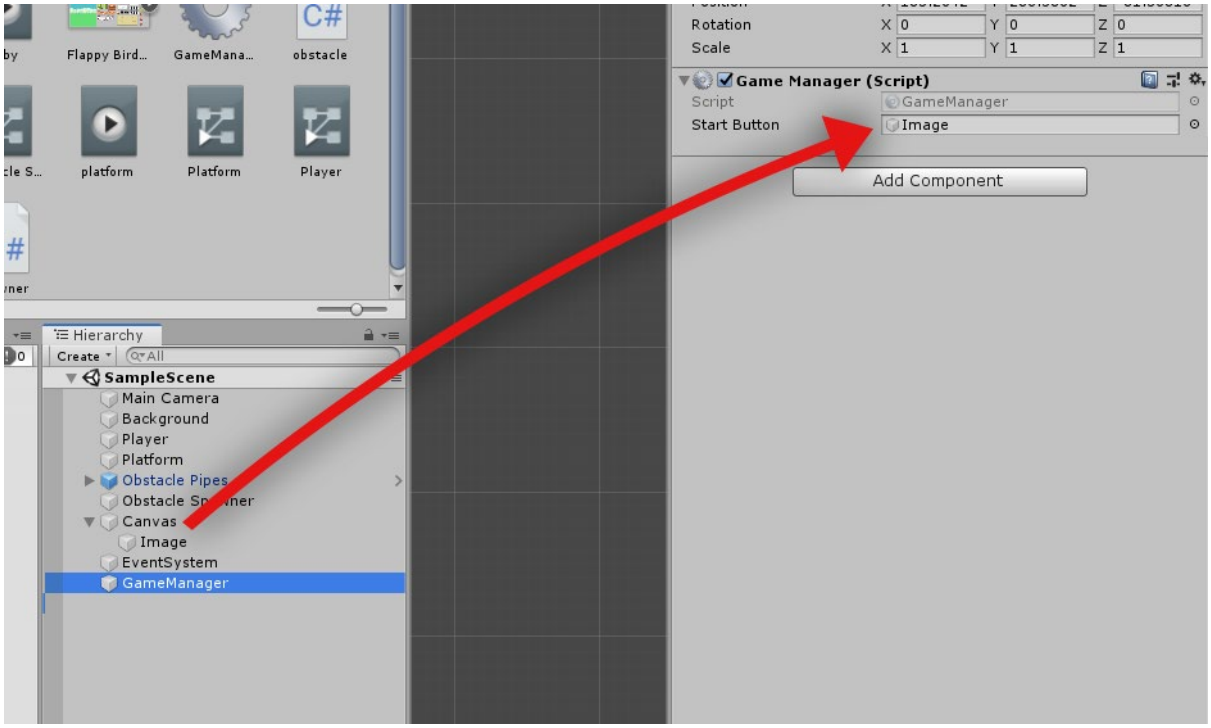
Em seguida, arraste o GameObject "GameManager" para a caixa logo abaixo do menu suspenso "Runtime Only".



Em seguida, clique no menu suspenso "No Function" e altere para GameManager → StartGame().



Em seguida, selecione o GameObject "GameManager" e arraste o GameObject da imagem para o campo "Start Button".



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.

Para isso, precisamos voltar ao script Player.cs e adicionar o seguinte código:

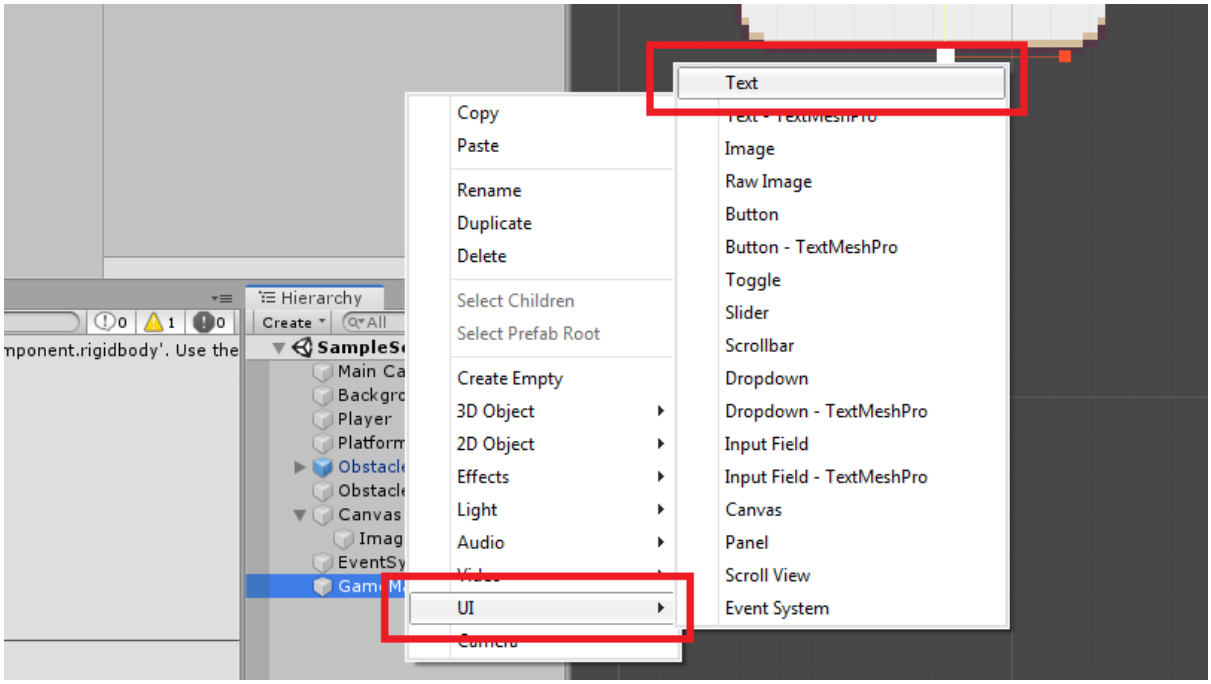
```
public GameManager gameManager;

public bool isDead = false;

private void OnCollisionEnter2D(Collision2D collision)

{

    isDead = true;

    gameManager.GameOver();

}Code language: C# (cs)
```

Salve o script e volte para o Unity.

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

Clique com o botão direito na Hierarquia.

Selecione UI → Text.



Configuração 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.

Adicionando Sombra ao Texto

No Inspector, clique em Add Component.

Pesquise por Shadow e adicione ao texto.

Andrei Inoue Hirata

Finalizando a Configuração no GameManager

Selecione o GameObject "GameManager".

Arraste o "Player" e o "Text UI" para os campos correspondentes no script GameManager.