



PROGRAMAÇÃO ORIENTADA A OBJETOS COM JAVA

DESENVOLVIMENTO DE JOGOS

MAIO/2024



Apresentação do Professor

Prof. EDUARDO PARETO

eduardo.pareto@uva.br



- ❖ **Mestre em Informática pelo IM-NCE UFRJ**
- ❖ **Graduado em Engenharia Elétrica PUC-RJ**
- ❖ **Graduado em Administração UNESA**
- ❖ **Pós Graduado em Docência Superior**

EMENTA

- ❑ Definição da história e evolução dos jogos eletrônicos.
- ❑ Introduzir a teoria dos Jogos.
- ❑ Conceitos Fundamentais para o planejamento e construção de jogos.
Desenvolver jogos para computadores e outros dispositivos.

COMPETÊNCIAS E HABILIDADES

- ❑ As competências gerais desenvolvidas, ao longo do curso, envolve a capacidade de desenvolver jogos digitais de média e alta complexidade.

OBJETIVOS DA DISCIPLINA

- ❑ Abordar as práticas profissionais relacionadas aos conteúdos técnicos da lógica de programação e aplicação desses conceitos na programação e construção de jogos digitais, através da utilização de técnicas e ferramentas de desenvolvimento.

CONTEÚDO PROGRAMÁTICO

Unidade 1 - História e Evolução dos Jogos Eletrônicos Objetivo

Específico (Habilidades) - Ao final desta Unidade, espera-se que o aluno tenha familiarização com os conceitos fundamentais de programação e desenvolvimento de jogos.

1.1 - Introdução a evolução dos games.

1.2 - Apresentação das Engines. Introdução ao Unity3D.

1.3 - Comparação entre os games engines.

1.4 - Desenvolvimento de jogos digitais: primeiros passos.

CONTEÚDO PROGRAMÁTICO

Unidade 2 - Teoria dos Jogos Objetivo Específico (Habilidades) - Ao final desta Unidade, espera-se que o aluno tenha a capacidade de programação em engine.

2.1 - Conceitos de aplicações com Unity3D.

2.2 - Criação de menu. Manipulação de câmera.

2.3 - Arquivos externos ao Unity3D.

2.4 - Jogos digitais com Unity3D.

CONTEÚDO PROGRAMÁTICO

Unidade 3 - Planejamento e Construção de Jogos Objetivo Específico (Habilidades) - Ao final desta Unidade, espera-se que o aluno tenha a capacidade de observação e representação do movimento.

3.1 - Formas geométricas. Colisão de telas.

3.2 - Construção e games. Tratamento de eventos internos.

3.3 - Tratamento de eventos externos. Controle do Teclado e Mouse.

3.4 - Aplicações de controle de games por eventos.

CONTEÚDO PROGRAMÁTICO

Unidade 4 - Desenvolvimento de Jogos Digitais Objetivo Específico (Habilidades) - Ao final desta Unidade, espera-se que o aluno tenha a capacidade de construir um jogo digital.

4.1 - Projeto de Jogo Digital: Planejamento e estruturação.

4.2 - Projeto de jogo Digital: Construção do jogo digital.

4.3 - Projeto de Jogo Digital: Testes de verificação e validação do jogo.

4.4 - Projeto de Jogo Digital: Desenvolvimento de melhorias do jogo digital. Aferição da qualidade do jogo digital. Entrega do jogo digital.

METODOLOGIA

Aulas expositivas, práticas e dialogadas, podendo contar com o apoio de projeções, além do desenvolvimento de trabalhos, individuais e/ou em grupo, visando ao preparo dos alunos para o mercado de trabalho profissional. Para isso, as atividades propostas favorecem a autonomia do aluno e a construção do conhecimento.

AVALIAÇÃO

Trabalho em grupo – máximo 5 componentes.

Deve ser apresentado um jogo completo com apresentação para turma a ser testado de forma aleatória por alunos da turma.

O jogo deve estar disponível antes da apresentação para ser testado.

A4 – Entrega do jogo e apresentação para turma.

SISTEMA DE AVALIAÇÃO

Trabalho 1 – Planejamento do Projeto do Jogo

- Utilizar o modelo GDD.
- Utilizar o quadro do Github do projeto.
- Tirar o print do quadro de tarefas do GitHub e colocar dentro do GDD do seu jogo.

SISTEMA DE AVALIAÇÃO

Trabalho 2 – Desenvolver o Jogo.

Tema: Livre;

Jogo: pode ser em 2D ou em 3D;

Deve ter um personagem principal.

Deve ter as pastas separadas e organizadas dentro do projeto.

O jogo desenvolvido deve refletir o GDD da Atividade 1.

O grupo da atividade 1 deve ser o mesmo grupo da atividade 2.

O GDD deve ser atualizado e entregue junto com o jogo.

O grupo deve entregar no Teams a pasta completa do Jogo e o GDD.

O grupo deve apresentar o Jogo em sala de aula.

A apresentação vai compor a nota do Jogo.



PLANEJAMENTO DAS AULAS (seg)

Maio	Conteúdo
06	
13	
20	
27	
Junho	Conteúdo
03	
10	Entrega do primeiro trabalho
17	
24	
Julho	Conteúdo
01	
08	Apresentação dos trabalhos – A4

04/7 - Último dia do lançamento da nota A2/A4 - Disciplinas Presenciais e Virtualizadas

13/7 - Último dia do lançamento da A4 - Disciplinas Presenciais e Virtualizadas

16/7 - Final do 2o Trimestre - 2024.2

REFERÊNCIAS

1. NOVAK, Jeannie. Desenvolvimento de Games. São Paulo: Cengage Learning, 2010.
2. RABIN, S. Introdução ao Desenvolvimento de Games. vol. 1. São Paulo: Cengage Learning, 2012.
3. SIMÕES, Alberto. Introdução ao desenvolvimento de jogos com Unity. São Paulo: FCA, 2017.



SUMÁRIO – Construção do Jogo

01. INTRODUÇÃO AO JOGO

02. CICLO DO JOGO

03. ANIMAÇÃO DO JOGO

04. PROGRAMAÇÃO E FÍSICA DO JOGO

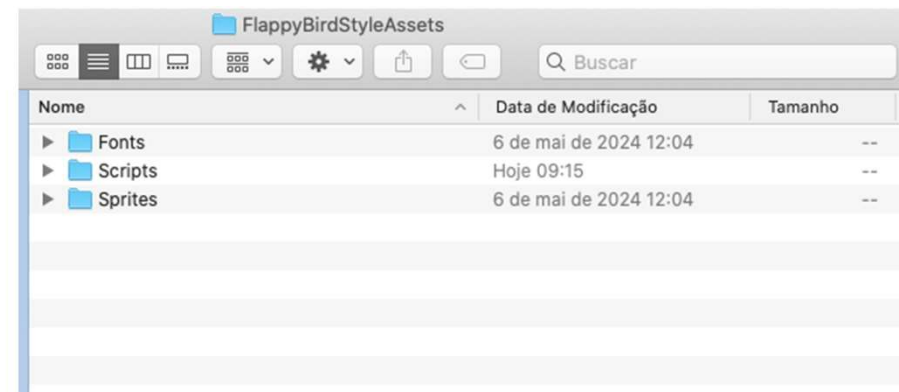
Mais informações em: <https://www.youtube.com/watch?v=WWn4i5u2pWY>



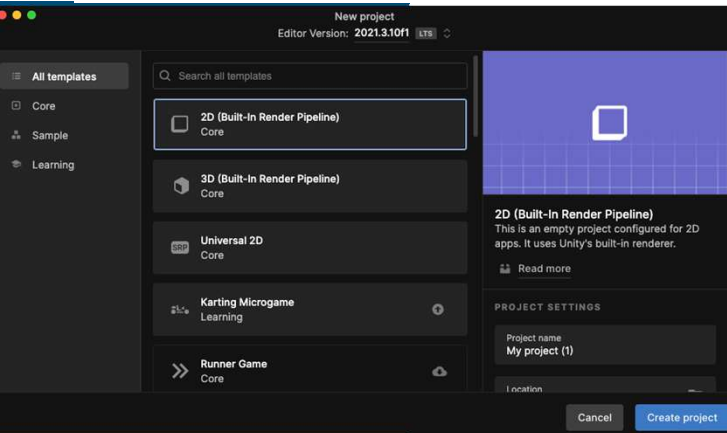
Preparação do Jogo

PREPARAÇÃO DO JOGO?

- INSTALAR UMA VERSÃO COMPLETA DO UNITY
- EDITOR VERSION 2021.3.10f1 LTS
 - Versões LTS - Desenvolvida para criadores que valorizam o que há de melhor em estabilidade e suporte para projetos futuros, a versão LTS reúne os recursos e as melhorias implementadas ao longo do ano em uma única instalação
- COPIAR AS PASTAS DE APOIO
 - Fonts
 - Sprites
 - Scripts



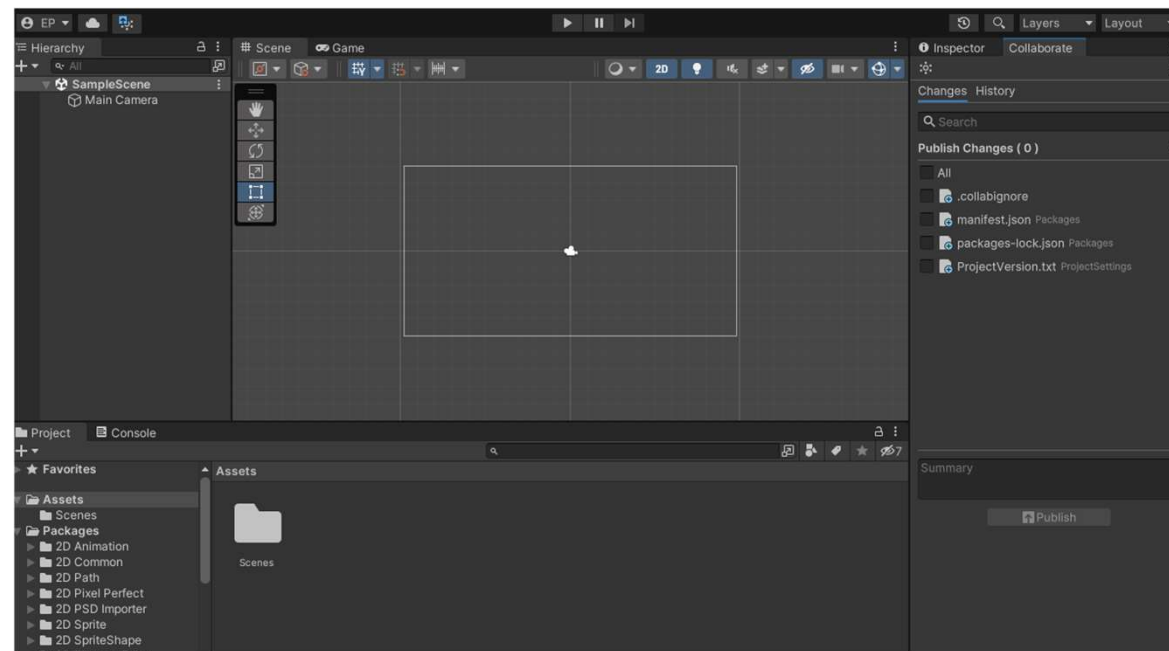
CRIANDO O PROJETO FLAPPY BIRD



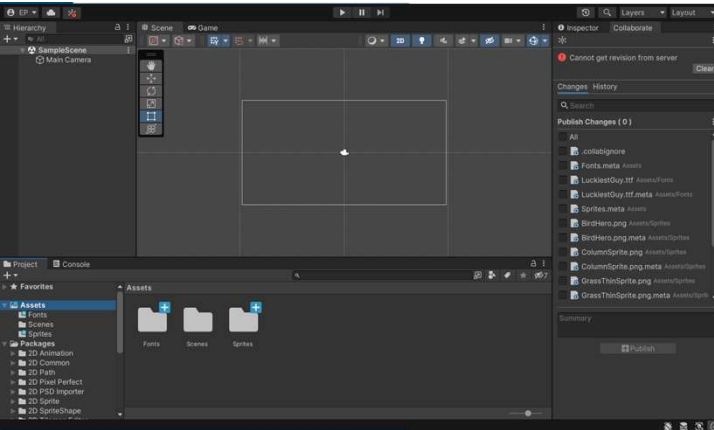
CRIAR UM NOVO 2D - CORE

De o nome de
FlappyBirdLive

Após algum tempo, a tela de construção
do jogo será aberta



INSERINDO AS PASTAS



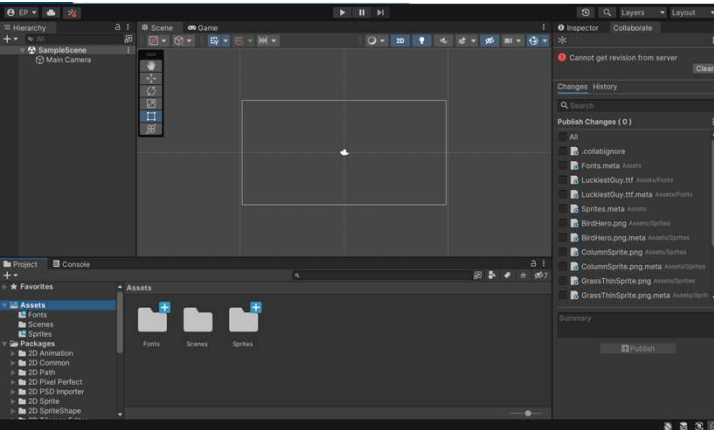
Vamos arrastar as duas pastas:

- Fonts
- Sprites

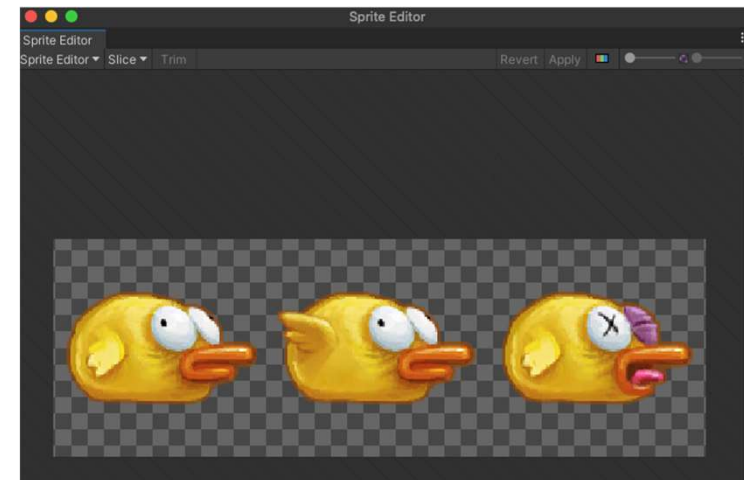
Para dentro da pasta

- Assets

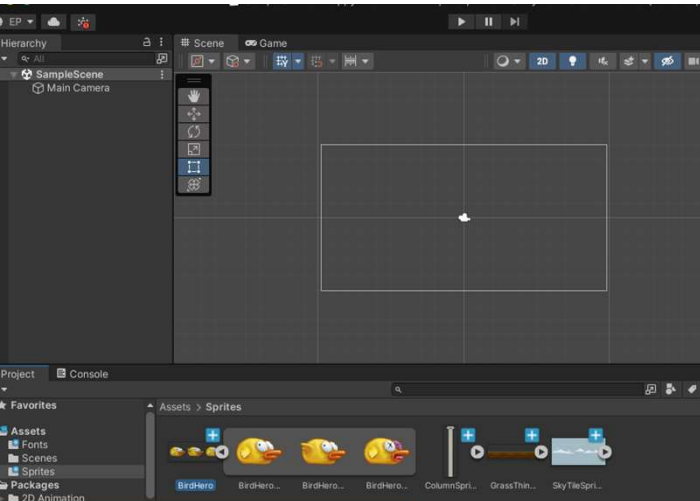
Preparando o Sprinte



Selecionar o Sprite BirdHero
Na janela Inspector, mudar o
Sprite Mode : Multiple
Apply
Clique em Sprite Editor
Slice -> Slice
Apply
Fechar

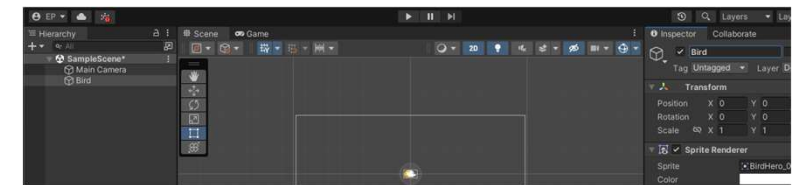
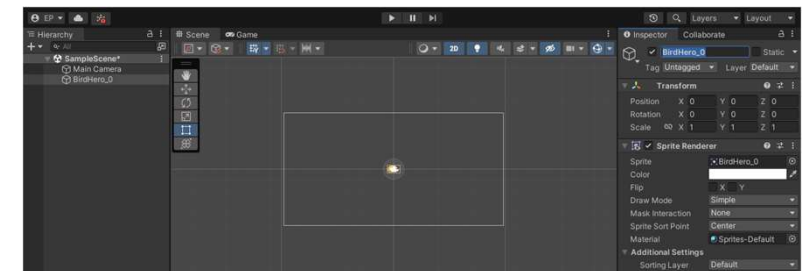


Inserindo o Sprinte

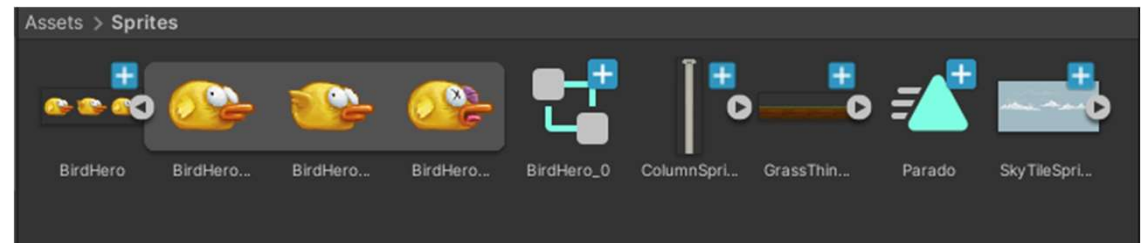
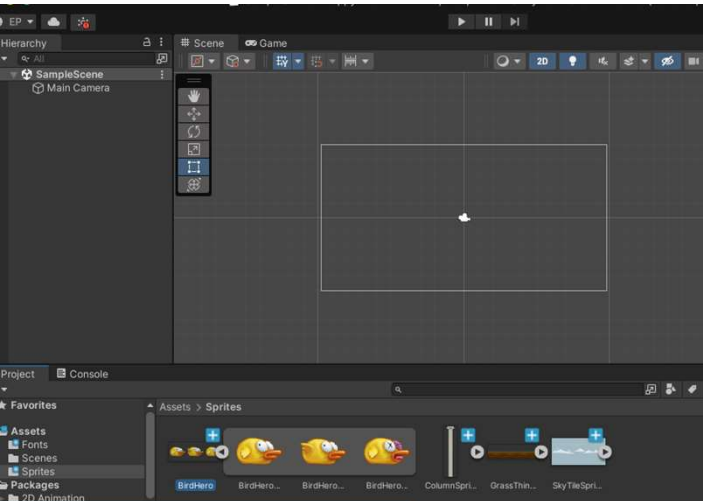


No Sprite BirdHero, clique no botão de *play* para abrir as fatias.

Pegue o primeiro e arraste para embaixo da Câmera do jogo. Troque o nome para Bird



Criando animação do Sprinte

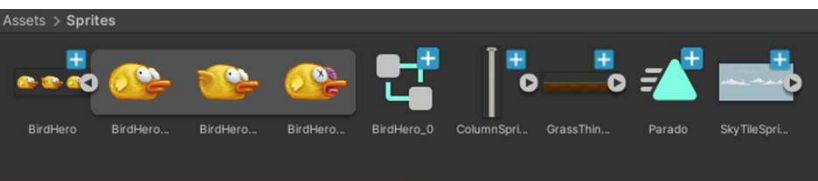


Vamos selecionar o nosso Heroi parado e batendo asas, os dois primeiros Sprites.

Arrastando a seleção múltipla para a hierarquia, ele irá pedir para criar a Animação e irá criar:

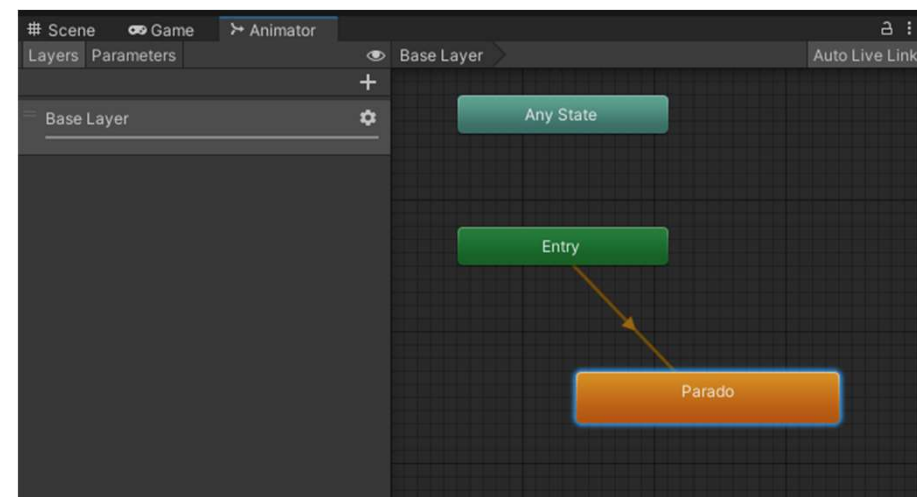
Animator e Animation

É importante lembrar que devemos apagar o Bird colocado no slide anterior e renomear o Sprite que contem a animação.



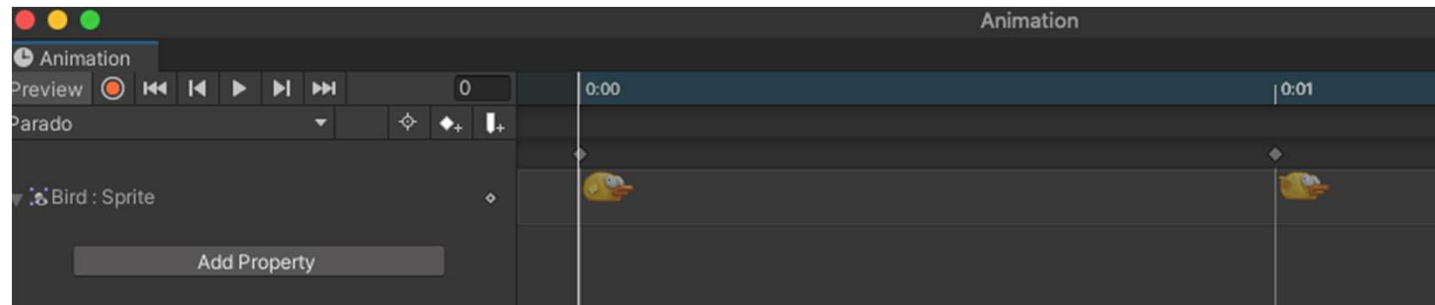
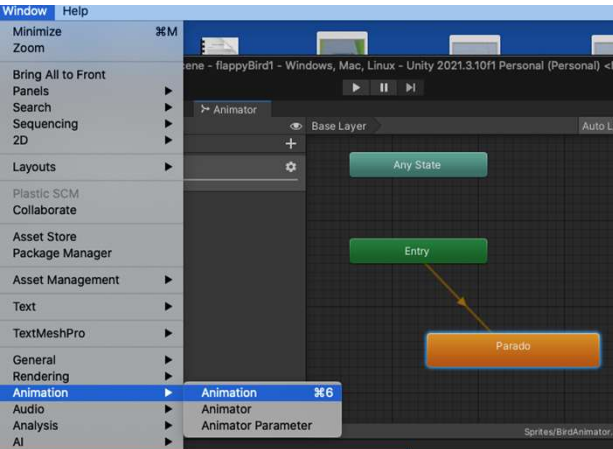
Trabalhando no Animator

Vamos renomear o Animator para BirdAnimator.
Cada estado está associado a uma Animation

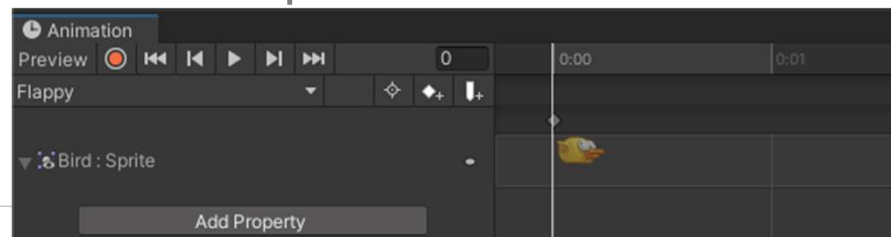


O Animator é uma
Máquina de
Estados

Criando as Animações

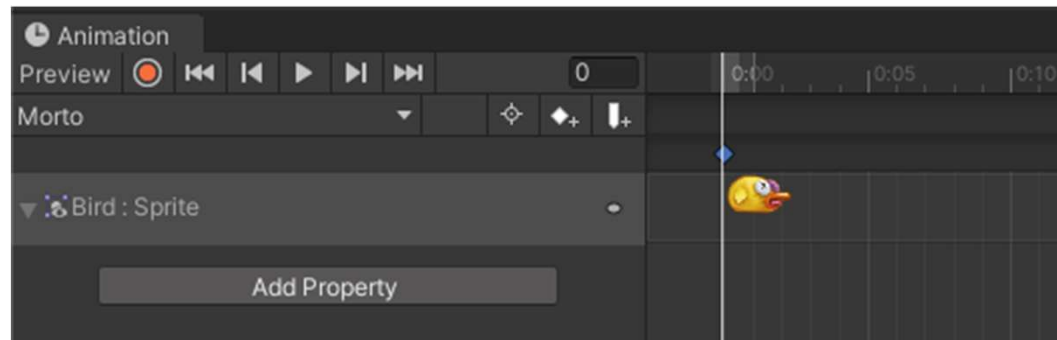


Vamos abrir a janela Animation
 O Objeto Bird deverá estar selecionado para você visualizar;
 Apague o segundo Sprite com a asa aberta
 Vamos criar uma nova Animação e dar o nome de -> Flappy <-
 Um macete que facilita é copiar da animação Parado, o Bird. Em seguida arraste o Sprite com a asa aberta e apague o Sprite anterior



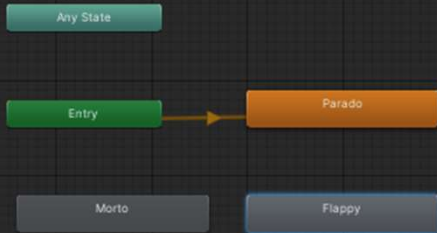
Criando as Animações

Vamos fazer a última animação:
Vamos criar uma nova Animação e dar o nome de -> Morto <-
Um macete que facilita é copiar da animação Parado, o Bird. Em seguida arraste o Sprite Morto e apague o Sprite anterior



Vamos trabalhar
na janela
Animation

Criando as Animações - OBS

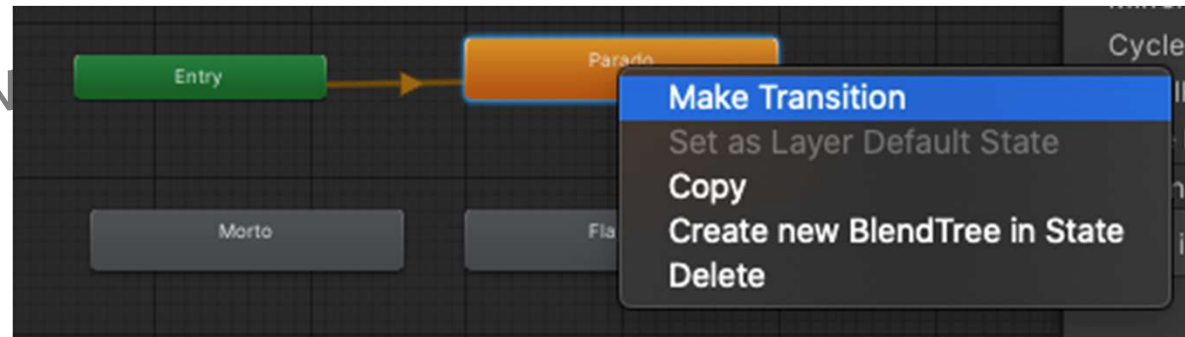


- Para que isso funcione, o objeto Bird precisa estar selecionado
- Os novos estados Morto e Flappy aparecem na máquina de estados – Animator
- Vamos voltar para a tela da Máquina de Estados para programar os eventos e as transições

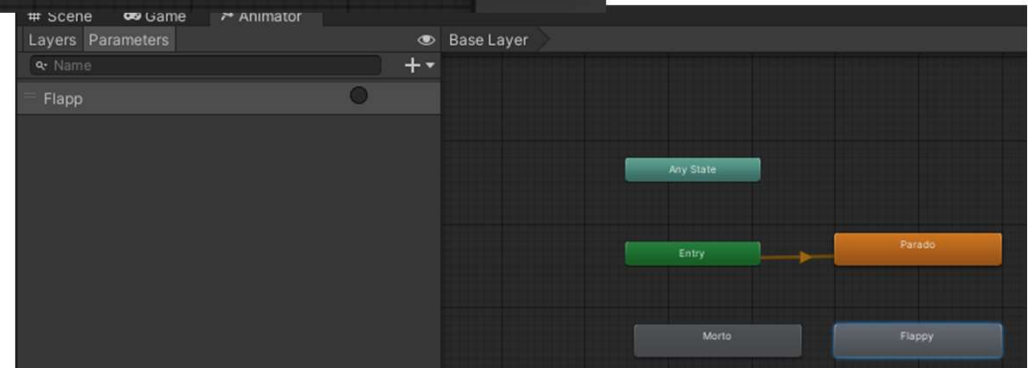
Vamos trabalhar
na janela
Animator

Criando as Transições

- N



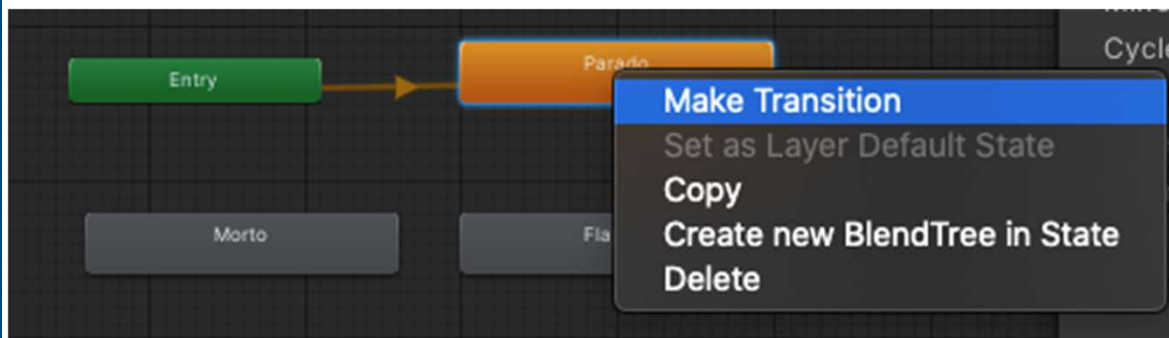
Flapp



Vamos trabalhar
na janela
Animator

Criando as Transições

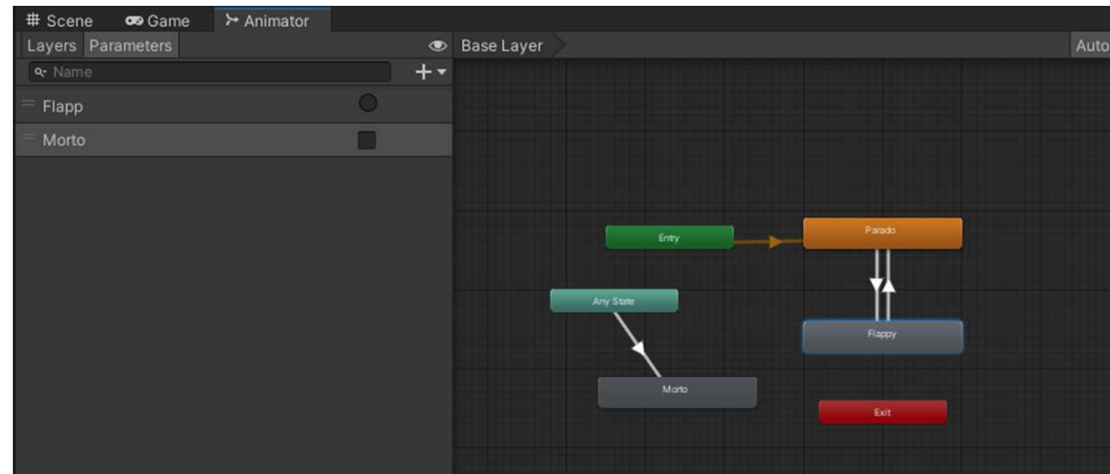
- Vamos clicar com o botão direito em cima do Estado de Origem e selecionamos -> Make Transition <-
- Arrastar para o Estado Final



Vamos trabalhar
na janela
Animator

Criando as Transições

- Transições:
 - De Parado -> Flappy
 - De Flappy -> Parado
 - De Any State -> Morto

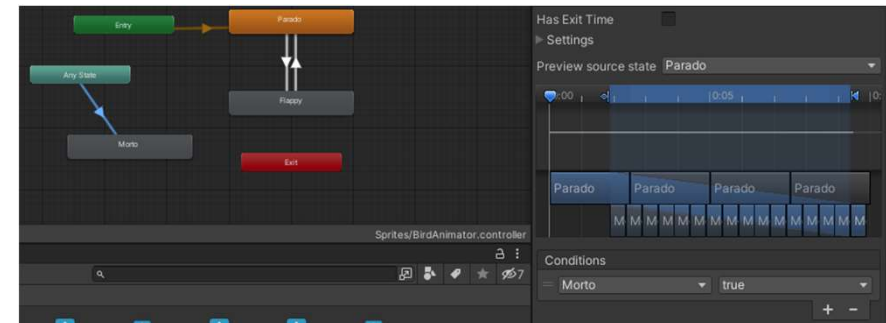
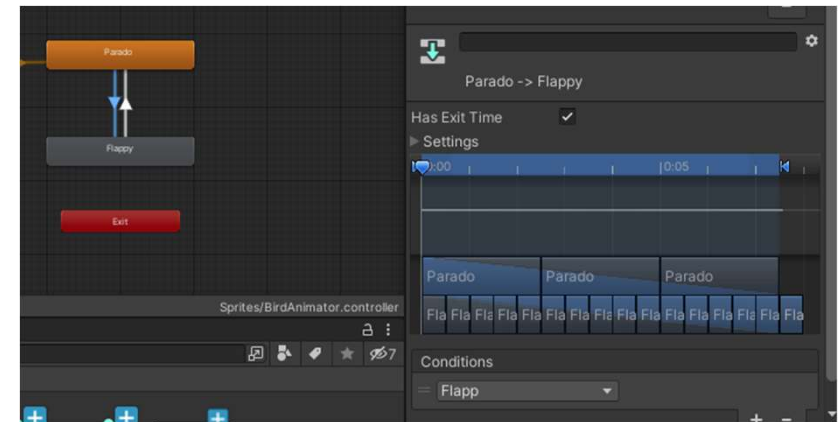


Vamos trabalhar
na janela
Animator

Ajustando as Transições

- Transições:
 - De Parado -> Flappy
 - Adicionar uma Conditions e selecionar Flapp
 - Desmarcar HasExitTime
 - De Any State -> Morto
 - Adicionar uma Conditions e selecionar Morto

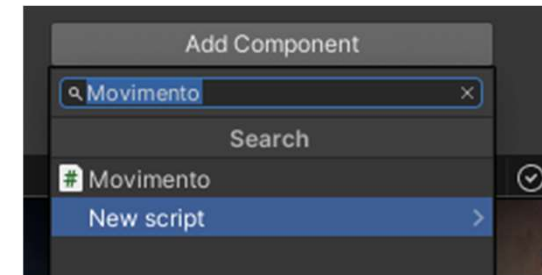
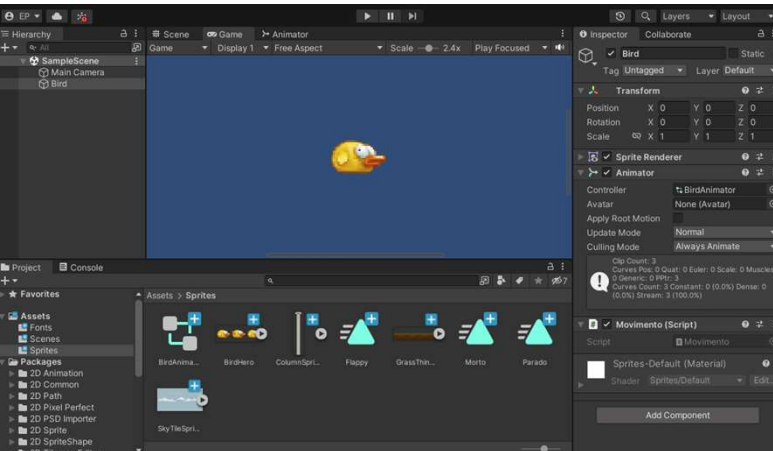
Vamos trabalhar
na janela
Animator



Finalizando o Movimento

- Neste momento, nosso jogo está controlando todo o movimento e os estados

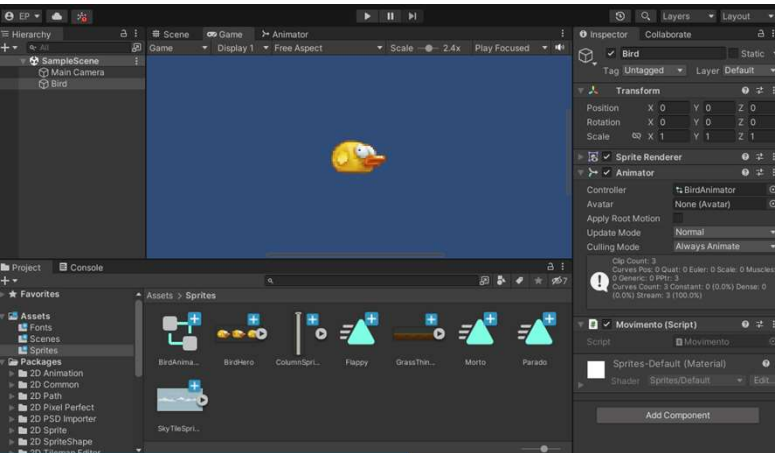
Criando o Script de Movimento do Bird



- Vamos adicionar um Componente
- O nome será Movimento
- O Visual Studio irá abrir

```
Movimento.cs
Nenhuma seleção

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Movimento : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     void Start()
9     {
10
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16
17     }
18 }
19
```



Criando o Script de Movimento do Bird

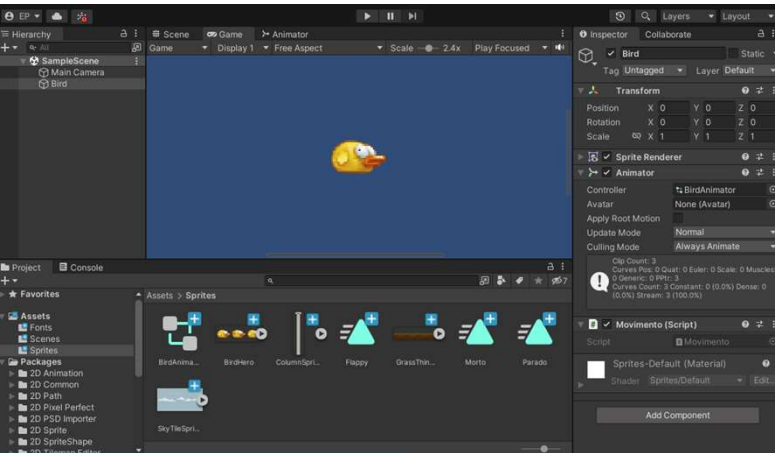
- Adicionamos o controlador
- No método Start(), capturamos o componente que estamos controlando - Bird
- No método Update(), verificamos se a tecla Space foi apertada e disparamos o Trigger Flapp, já programado no Unity
- Salvar as alterações

Vamos trabalhar
na janela Visual
Studio

```

Movimento.cs
Movimento ▶ Update()
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Movimento : MonoBehaviour
6  {
7      private Animator controlador;
8
9      // Start is called before the first frame update
10     void Start()
11     {
12         controlador = GetComponent<Animator>();
13     }
14
15     // Update is called once per frame
16     void Update()
17     {
18         if (Input.GetKey(KeyCode.Space))
19         {
20             controlador.SetTrigger("Flapp");
21         }
22     }
23 }
24
25

```



Testando o Movimento

- Vamos retornar para o Unity;
- Dar Play
- O pássaro deverá bater asas ao clicar na barra de espaço

Vamos trabalhar
na janela Unity

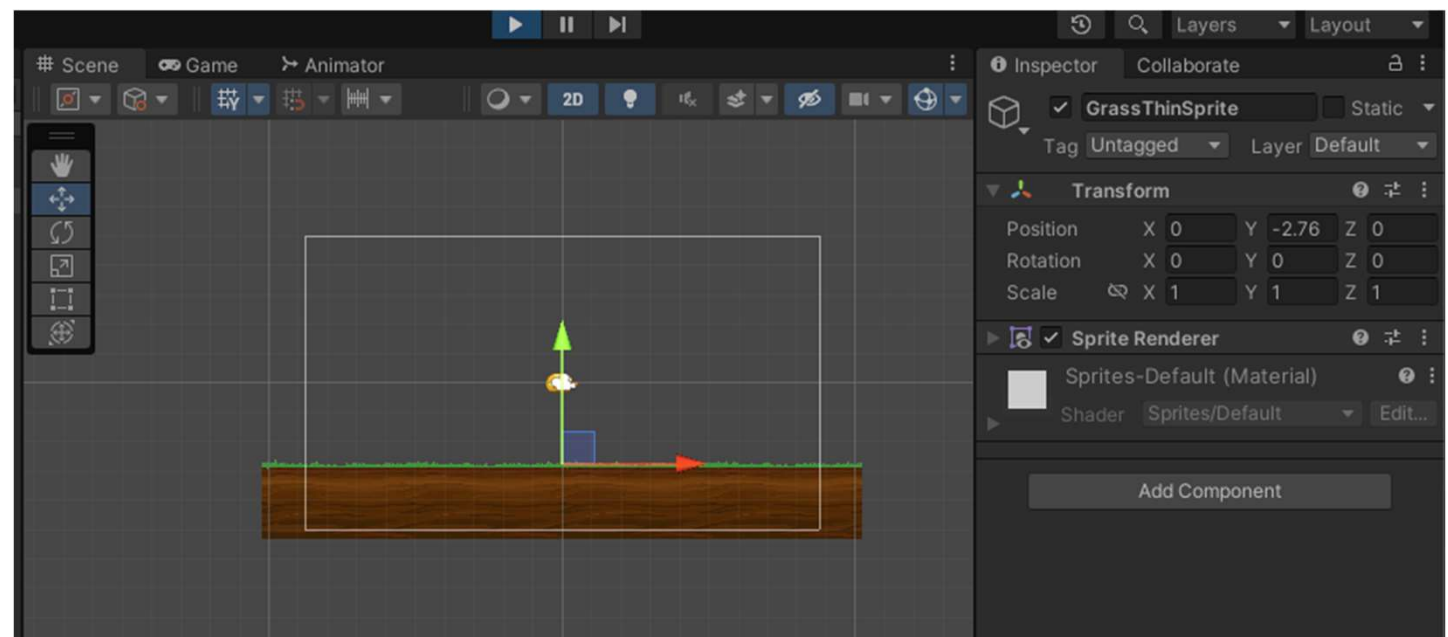


Nenhuma alteração deverá ser feita quando o jogo estiver executando. O Unity não guarda as alterações feitas com o Jogo Executando.

SEMPRE PARE O JOGO

CRIANDO O CENÁRIO

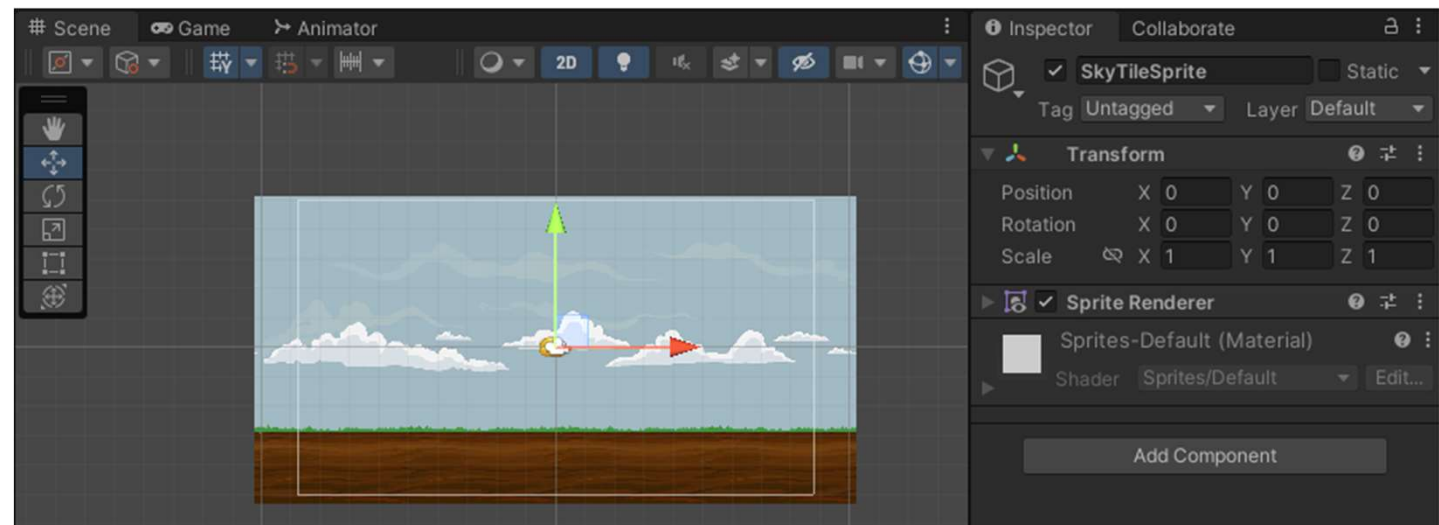
- Vamos adicionar o chão, arrastando o arquivo GrassThinSprite para baixo do Bird;
- Troque o nome para Solo;



Vamos trabalhar
na janela Unity

CRIANDO O CENÁRIO

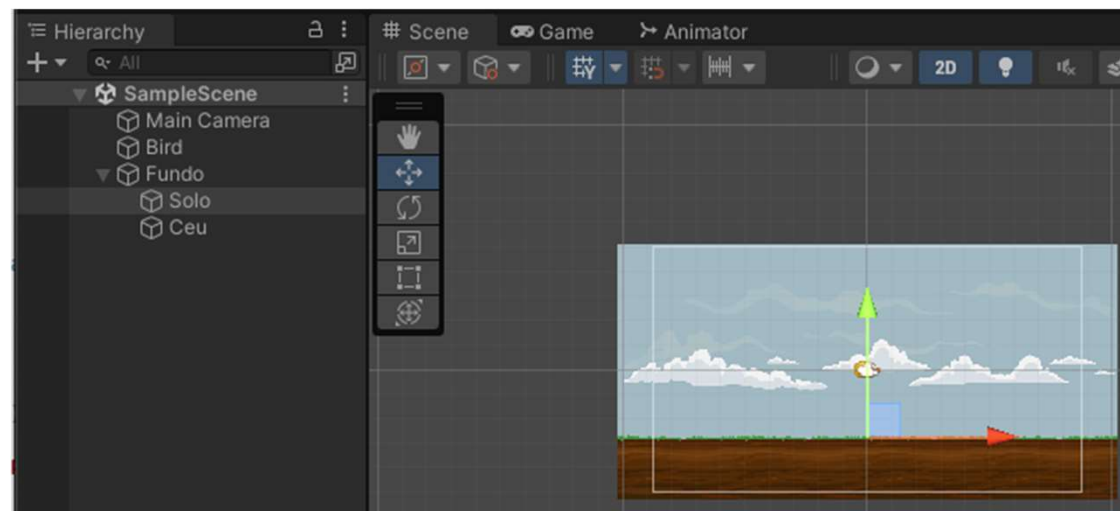
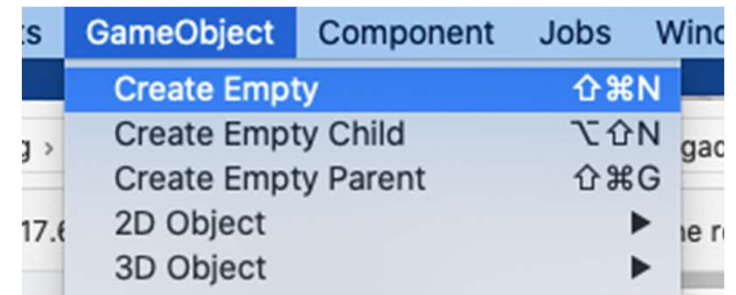
- Vamos adicionar o céu, arrastando o arquivo SkyTileSprite para baixo do Solo;
- Troque o nome para Solo;



Vamos trabalhar
na janela Unity

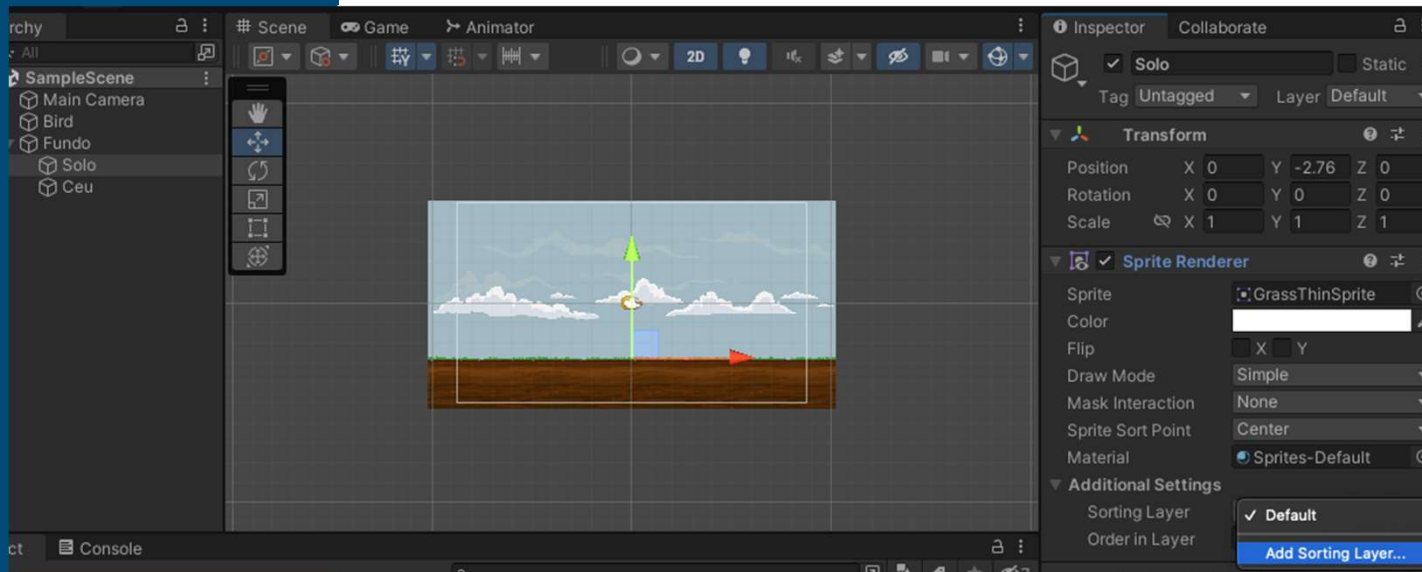
CRIANDO O CENÁRIO

- Vamos criar um Game Object para colocar estes elementos dentro. Ele será chamado de Fundo
- Arraste o Solo e o Ceu para dentro dele;



Vamos trabalhar
na janela Unity

Criando as camadas



Em qualquer Sprite:

- Na Janela Inspector
 - Sprite Renderer
 - Additional Settings
- Sorting Layer

Clicar na seta pra baixo e escolher Add Sorting Layer...

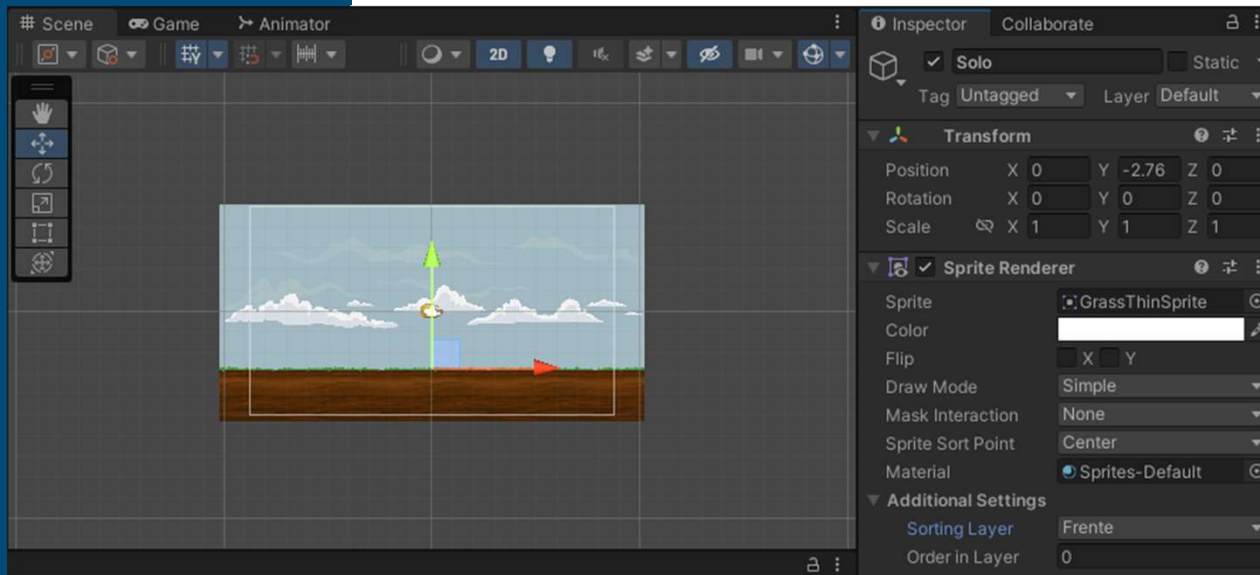


Vamos trabalhar
na janela Unity

A janela Inspector de Tags & Layers é aberta
Adicione as seguintes Camadas:

- Fundo
- Meio
- Frente

Colocando os Sprites nas camadas



As Camadas:

- Solo -> Frente
- Céu -> Fundo
- Bird -> Frente

Vamos trabalhar
na janela Unity

Preparar o Fundo



Vamos Acrescentar um Componente

- Selecione o Fundo
 - Clicar em Add Component
 - Escolher: Rigidbody 2D
- Este componente vem como padrão,
 - Body Type -> DynamicIsso significa que a gravidade atua nele. Para que não atue, mude para Static

Vamos trabalhar
na janela Unity

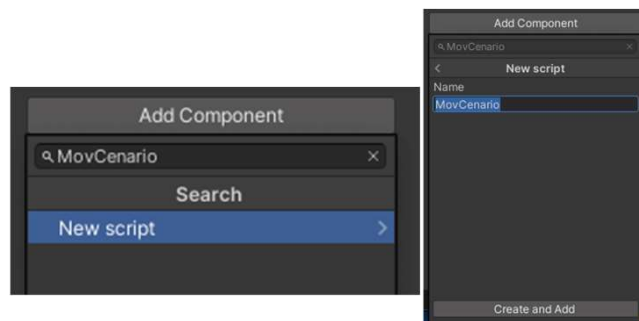
Preparar o Fundo



Vamos Acrescentar um Componente

- Selecione o Fundo
 - Clicar em Add Component
 - Escolher: Rigidbody 2D
 - Este componente vem como padrão,
 - Body Type -> Dynamic
 Isso significa que a gravidade atua nele.
 Para que não atue, mude para Kinematic;
- Quando um componente é Cinemático, as forças atuam, mas não a força da gravidade;
- Agora vamos adicionar um Script que se chama MovCenario

Vamos trabalhar
na janela Unity



Preparar o Fundo

Vamos copiar todo o código e entender:

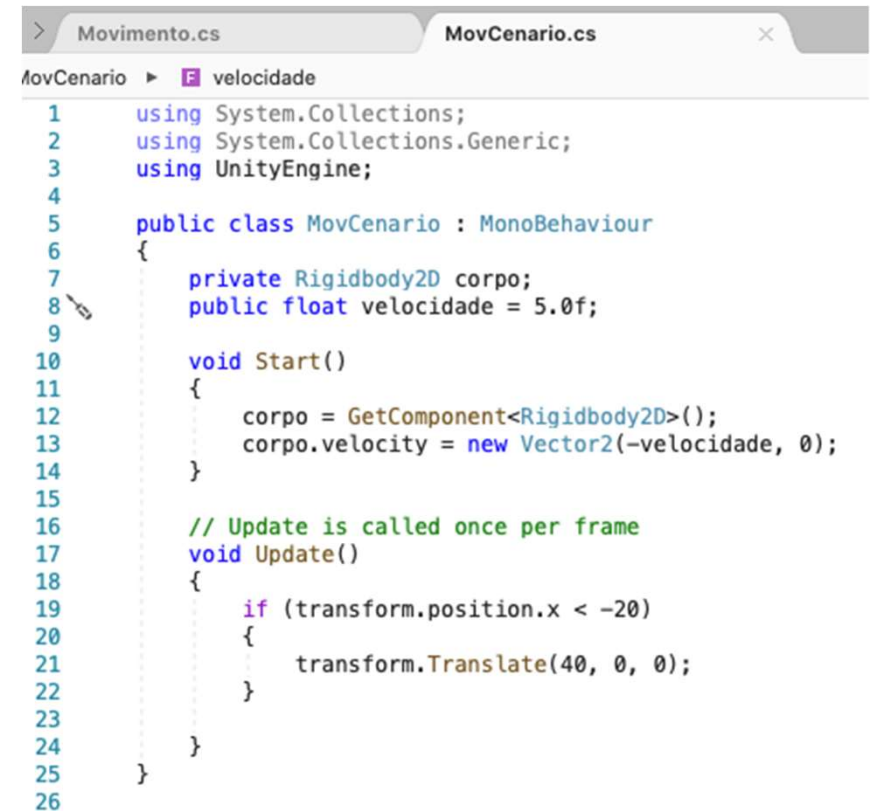
A Variável corpo irá capturar o Rigidbody2D

A variável velocidade definirá a velocidade do cenário e poderá ser modificada no Unity (public)

No Update() – o redesenho da fundo irá transladar sempre que chegar na posição 20.

Este tamanho depende do tamanho da figura.

Vamos trabalhar na janela Visual Studio



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class MovCenario : MonoBehaviour
6 {
7     private Rigidbody2D corpo;
8     public float velocidade = 5.0f;
9
10    void Start()
11    {
12        corpo = GetComponent<Rigidbody2D>();
13        corpo.velocity = new Vector2(-velocidade, 0);
14    }
15
16    // Update is called once per frame
17    void Update()
18    {
19        if (transform.position.x < -20)
20        {
21            transform.Translate(40, 0, 0);
22        }
23    }
24 }
25
26
```

Preparar o Fundo

Vamos copiar todo o código e entender:

A Variável corpo irá capturar o Rigidbody2D

A variável velocidade definirá a velocidade do cenário e poderá ser modificada no Unity (public)

No Update() – o redesenho da fundo irá transladar sempre que chegar na posição 20.

Este tamanho depende do tamanho da figura.



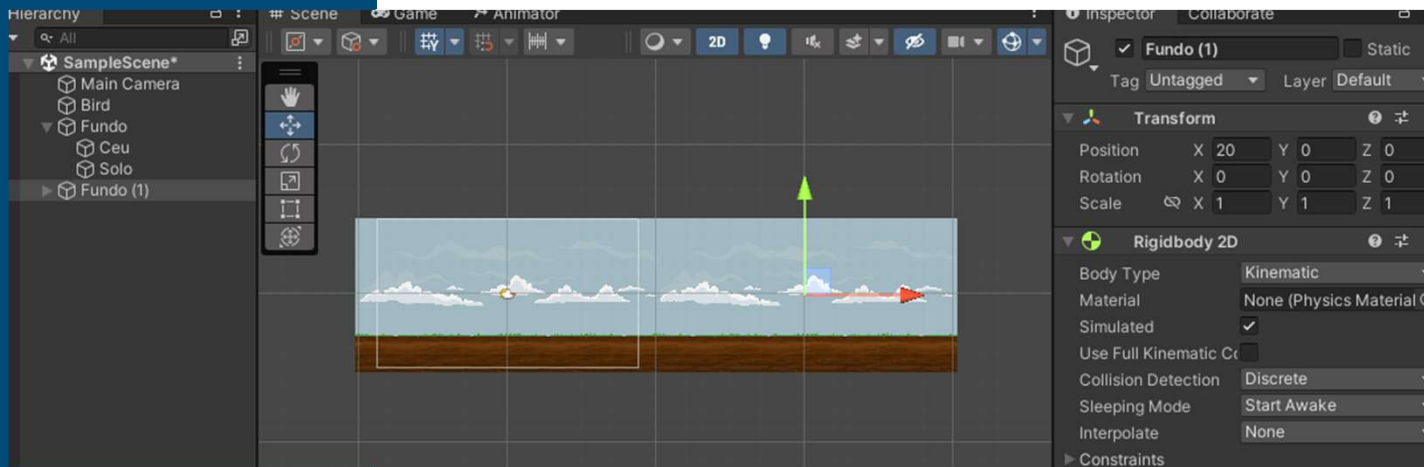
```

Movimento.cs  MovCenario.cs
MovCenario ▶ velocidade
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class MovCenario : MonoBehaviour
6  {
7      private Rigidbody2D corpo;
8      public float velocidade = 5.0f;
9
10     void Start()
11     {
12         corpo = GetComponent<Rigidbody2D>();
13         corpo.velocity = new Vector2(-velocidade, 0);
14     }
15
16     // Update is called once per frame
17     void Update()
18     {
19         if (transform.position.x < -20)
20         {
21             transform.Translate(40, 0, 0);
22         }
23     }
24 }
25
26

```

Vamos trabalhar
na janela Visual
Studio

Preparar o Fundo



Vamos copiar o fundo:
Selecione o Objeto Fundo
Copie e cole
Desloque o objeto Fundo(1) da
posição X – 20

Teste e verifique que o mundo
agora tem continuidade.

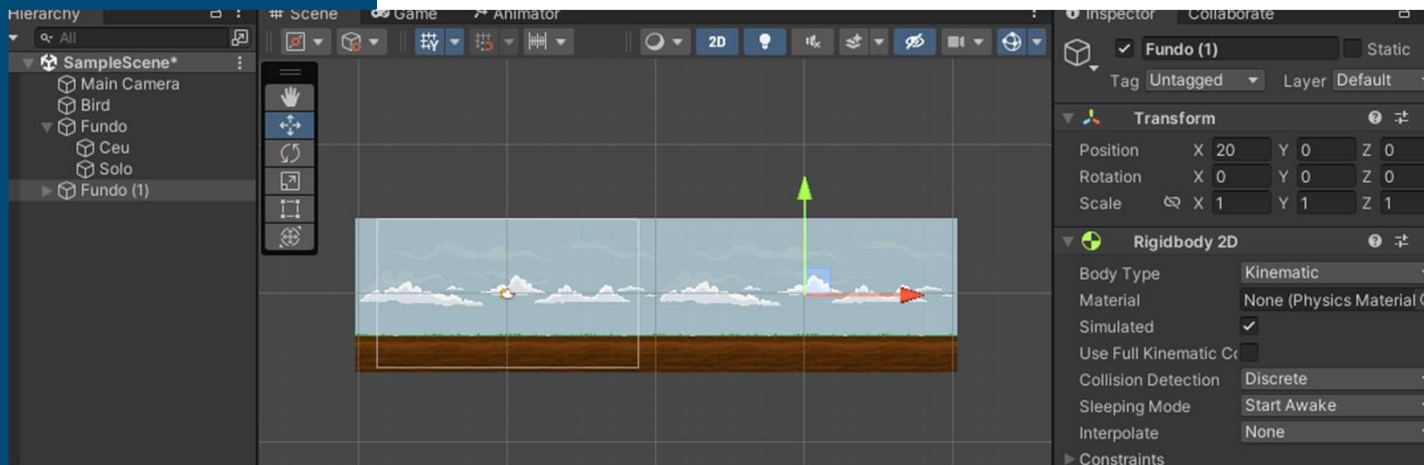


Nenhuma alteração deverá ser feita quando o jogo estiver executando. O Unity não guarda as alterações feitas com o Jogo Executando.

SEMPRE PARE O JOGO

Vamos trabalhar
na janela do Unity

Cenário Pronto



Neste Momento nosso cenário está pronto.

Próximos Passos:
Desenvolver o Inimigo



Nenhuma alteração deverá ser feita quando o jogo estiver executando. O Unity não guarda as alterações feitas com o Jogo Executando.

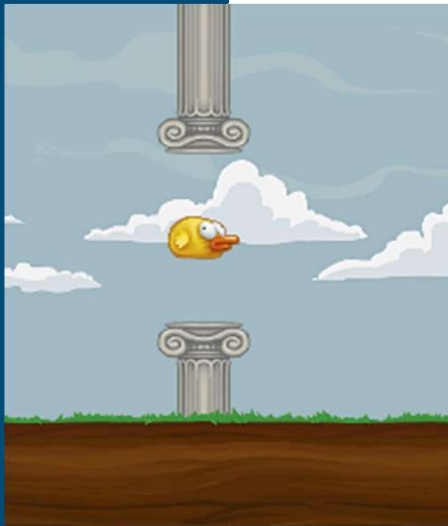
SEMPRE PARE O JOGO

Vamos trabalhar
na janela do Unity

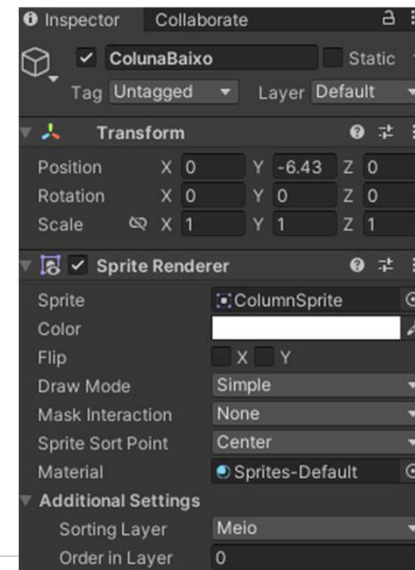
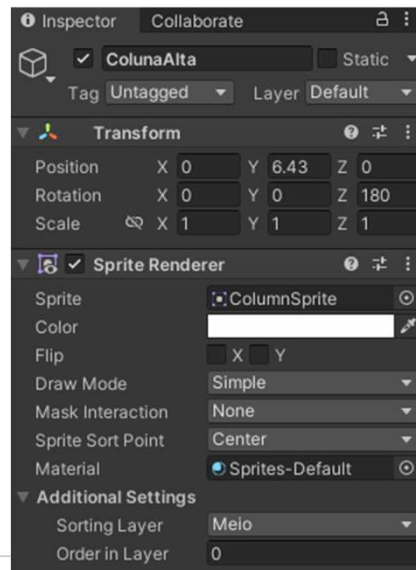
Desenvolvendo o Inimigo

Vamos colocar dois Sprites ColumnSprite

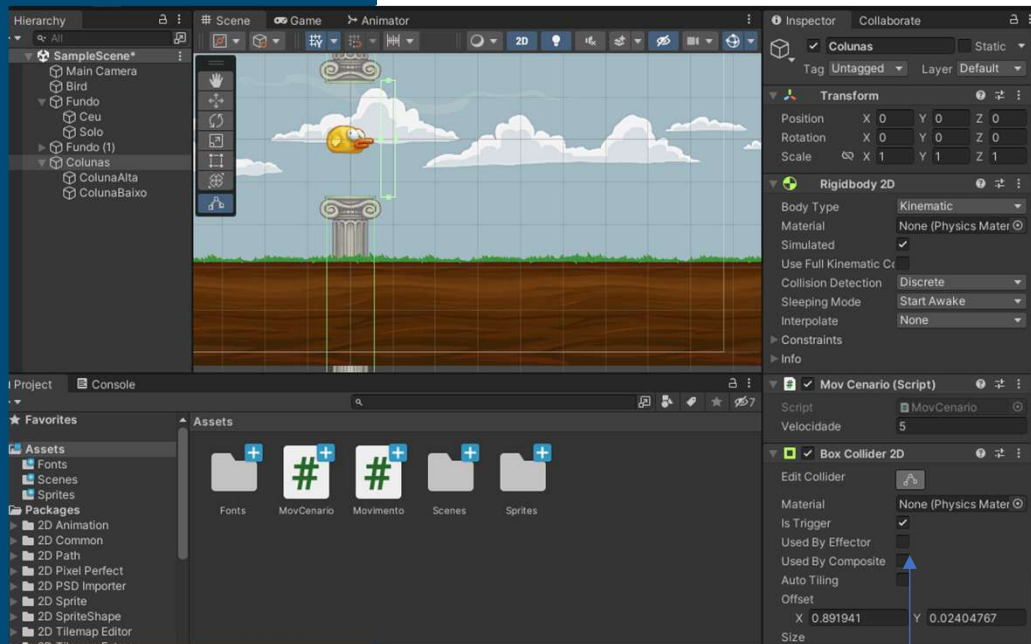
- Posicione os dois como mostra a figura. O segundo deve ser rotacionado de 180 graus no eixo Z
- Deixe um espaço para o pássaro passar
- As colunas devem estar na Sorting Layer -> Meio



Vamos trabalhar
na janela do Unity



Desenvolvendo o Inimigo



Vamos trabalhar
na janela do Unity

Criar um Game Object -> Colunas

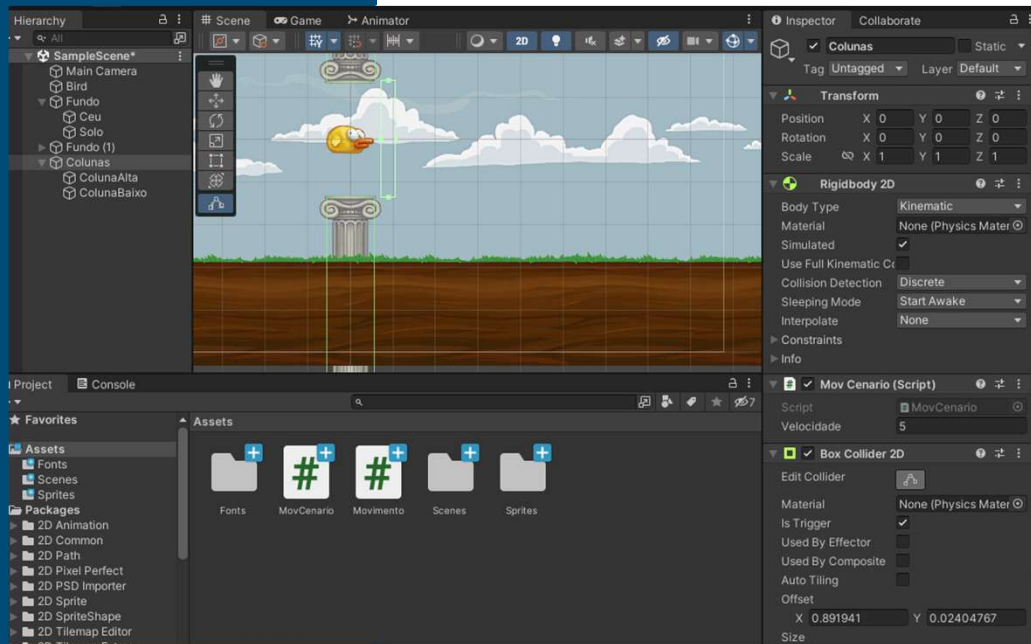
- Acerte a posição do novo objeto para 0,0,0
- Coloque as colunas dentro do novo Objeto
- Adicionar um Componente Rigidbody2D
- Adicionar o MovCenario.cs

Com isso, nosso cenário agora se move com as colunas.

Precisamos agora dos colisores – são 3

- Na ColunaBaixo, adicione o Componente Box Collider 2D;
 - Edite o Collider para ajustar a colisão
- Repita o procedimento para a coluna de cima
- Adicione o último como um portal, que será um Trigger onde usaremos para registrar a pontuação do Herói

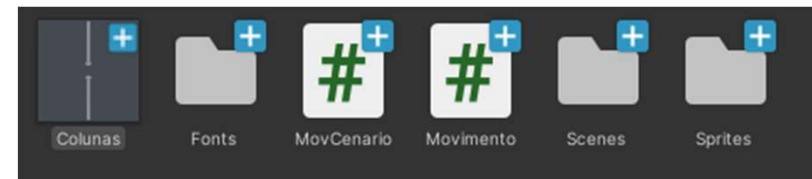
Criando Componentes PreFabs



Vamos trabalhar
na janela do Unity

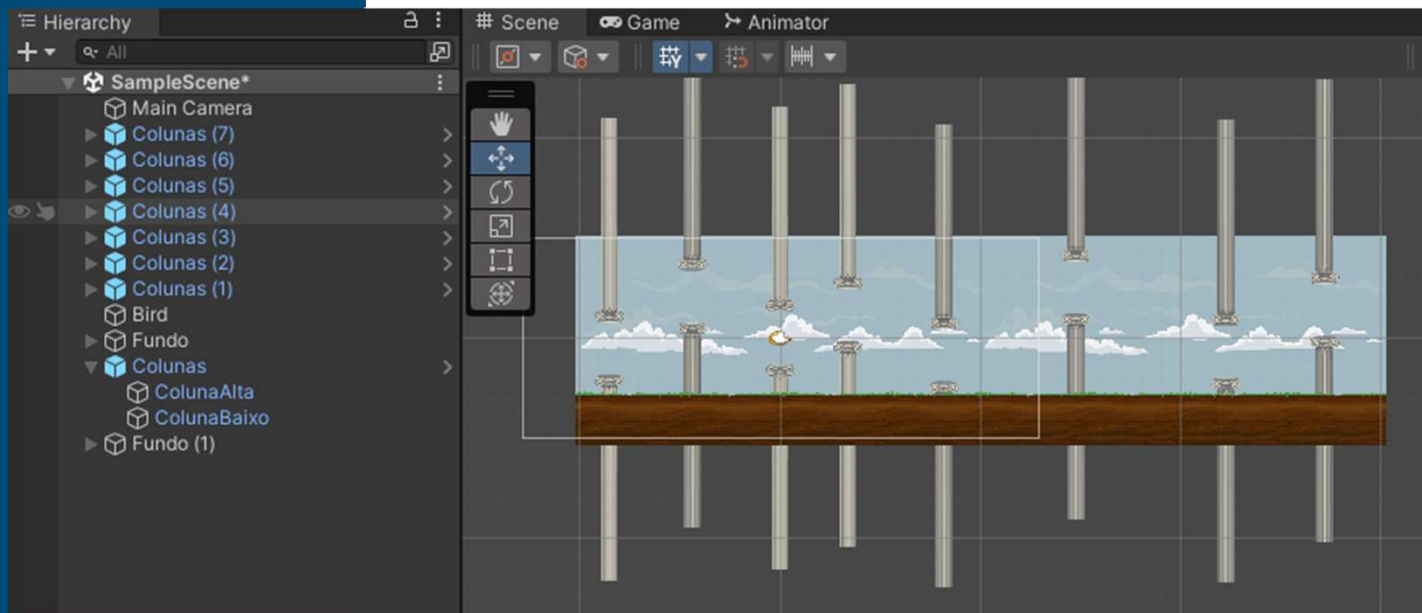
Para fazer um componente:

- Arraste o Objeto para dentro do Assets
- Ele irá criar um PreFab



- Este Objeto pode ser colocado no cenário manualmente ou via programação, como iremos fazer.

Criando Componentes PreFabs



Com os elementos Pré Fabricados, posso colocar quantos quiser, simplesmente arrastando para o cenário.

Vamos apagar todos os obstáculos, visto que faremos isso automaticamente

Vamos trabalhar na janela do Unity

Desenvolver o Gerador de Inimigos

Precisamos gerar os inimigos, que no nosso caso será o portal das colunas, que transformamos em Prefab.

Isso se chama Spawn (gerar)

Faremos um arquivo chamado de Spawner (gerador)

Precisamos criar um Componente para associar este código.

Vamos criar um Game Object -> Controlador

No objeto Controlador, adicionar um Componente:

- ADD Component -> um novo Script chamado Controlador

Vamos copiar todo o código para o Script Controlador.

- ADD Component -> um novo Script chamado Spawner

Vamos trabalhar
na janela do Unity

Trabalhando no Controlador

Vamos comentar todos os componentes visuais ainda não implementados.

Salvar o código todo

Copiar o conteúdo para o Script Spawner

Salvar o código todo

```
void Pause()
{
    Time.timeScale = 0f;
    player.enabled = false;
}

public void Play()
{
    score = 0;
    /*    scoreText.text = score.ToString();

    playButton.SetActive(false);
    gameOver.SetActive(false);
    */
    Time.timeScale = 1f;
    player.enabled = true;
}

public void GameOver()
{
    /*    playButton.SetActive(true);
    gameOver.SetActive(true);
    */
    Pause();
}

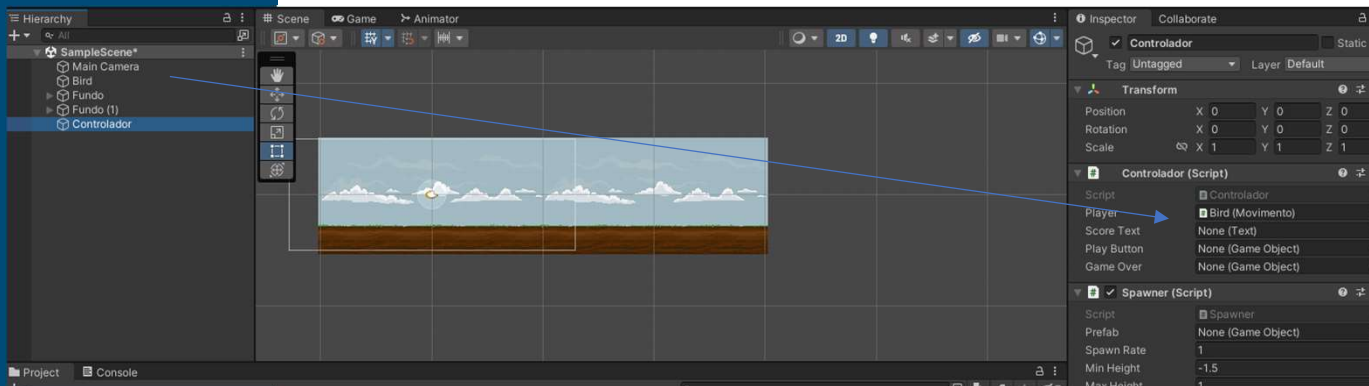
public void BirdScored()
{
    score++;
    // scoreText.text = score.ToString();
}

public void RestartGame()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
}
```

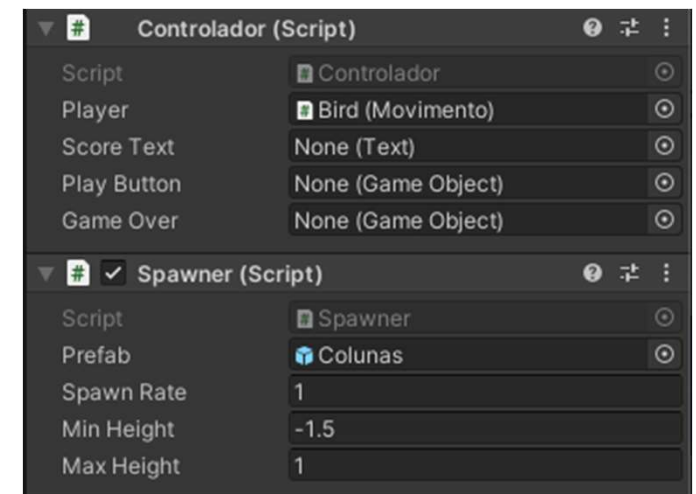
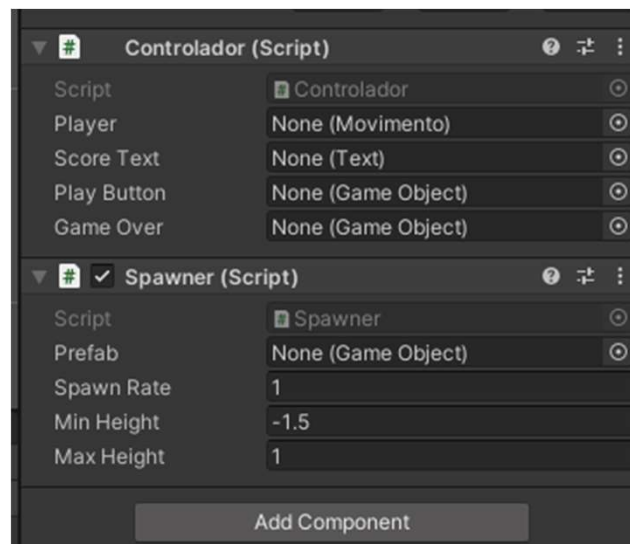
Vamos trabalhar
na janela do Unity

Trabalhando no Controlador

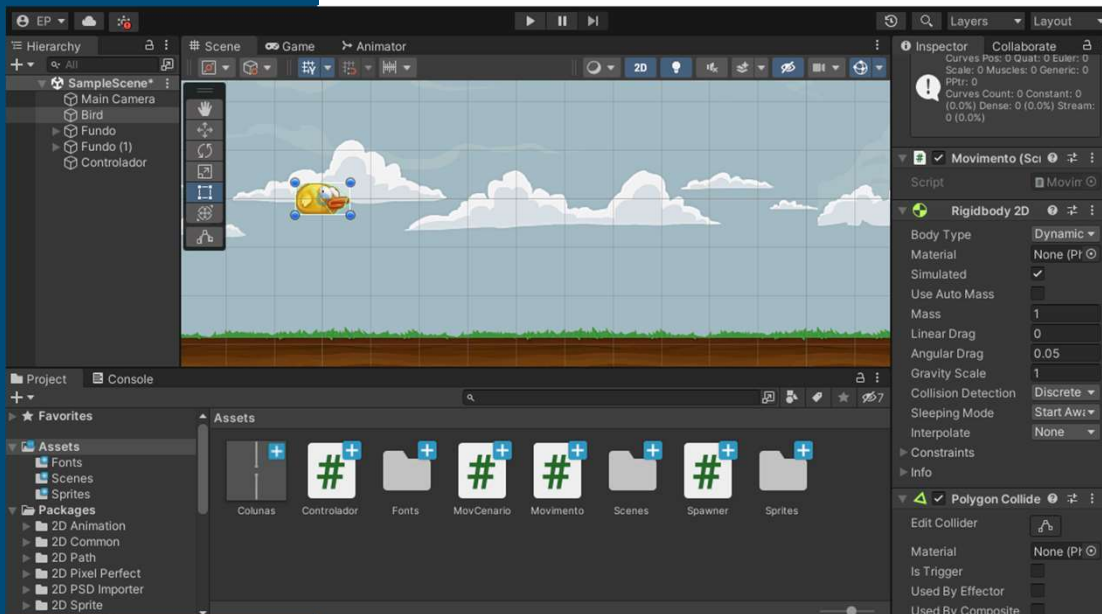
Vamos Associar os objetos ao Controlador:
Player -> Bird
Prefab -> Colunas
Salvar e testar!



Vamos trabalhar
na janela do Unity



Trabalhando no Passaro



Adicionar um componente:

- Rigidbody2d

Ele deverá ser Dynamic, pois sofre o efeito da gravidade;

- Polygon Collider 2D

Este componente já irá envolver todo o pássaro

Agora iremos completar o código do Movimento

Vamos trabalhar
na janela do Unity

Trabalhando no Movimento

Acrescentar as linhas no Script Movimento para ficar igual a figura ao lado.

Salvar as alterações

Vamos testar o jogo

Perceba que o pássaro irá cair direto, pois a gravidade está atuando.

Temos que colocar algumas coisas

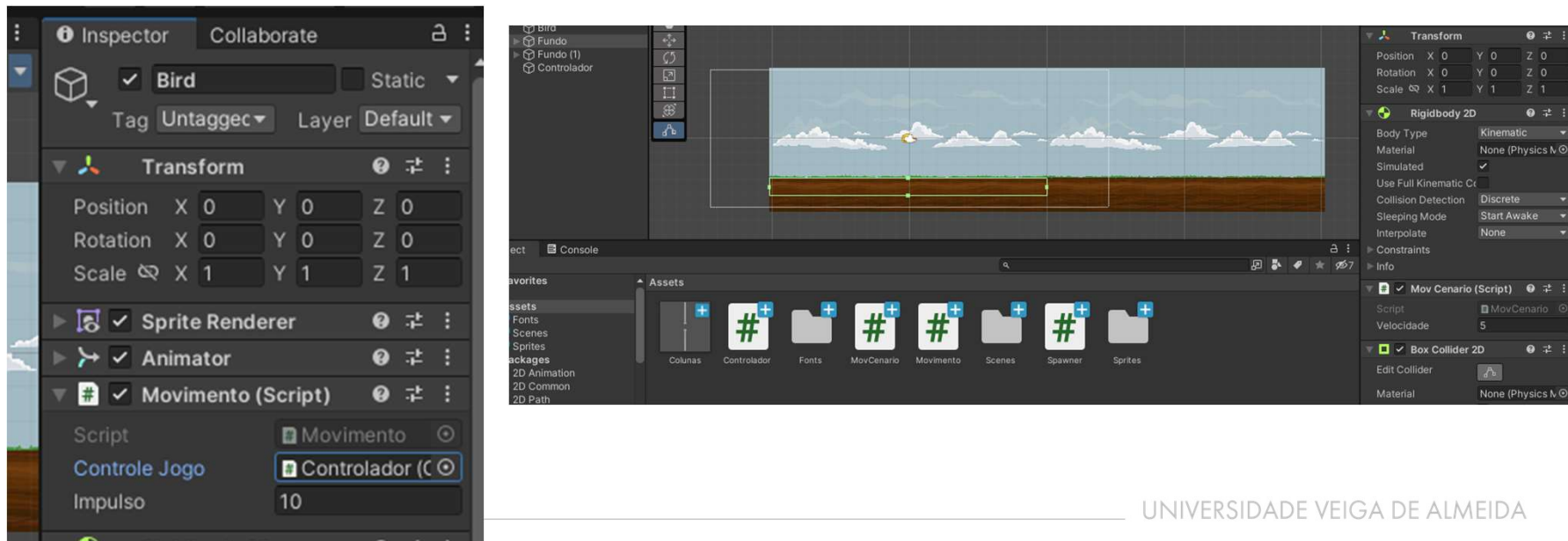
Vamos trabalhar na janela do Visual Studio

```
Movimento.cs  MovCenario.cs  Controlador.cs
Movimento ▶ OnCollisionEnter2D(Collision2D coll)
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Movimento : MonoBehaviour
6  {
7      private Animator controlador;
8
9      private Rigidbody2D corpo;
10     public Controlador controleJogo;
11     public float impulso = 10;
12
13     // Start is called before the first frame update
14     void Start()
15     {
16         controlador = GetComponent<Animator>();
17         corpo = GetComponent<Rigidbody2D>();
18     }
19     // Update is called once per frame
20     void Update()
21     {
22         if (Input.GetKey(KeyCode.Space))
23         {
24             controlador.SetTrigger("Flapp");
25             corpo.AddForce(Vector2.up * impulso);
26         }
27     }
28     void OnCollisionEnter2D(Collision2D coll)
29     {
30         controlador.SetBool("Morto", true); //Ver o nome dado na variável
31         //controleJogo.GameOver();
32     }
33     void OnTriggerEnter2D(Collider2D coll)
34     {
35         controleJogo.BirdScored();
36     }
37 }
```

Ajustando alguns problemas

Selecionar o Bird, verificar que o Controle Jogo -> None
Vamos adicionar um colisor no solo. Selecione Fundo e add
Component -> Box Collider 2D / Ajustar o Edit Collider
Faça o mesmo para o outro Fundo -> Fundo (1)
Agora nosso Heroi ao cair, bate no chão e morre!!!!

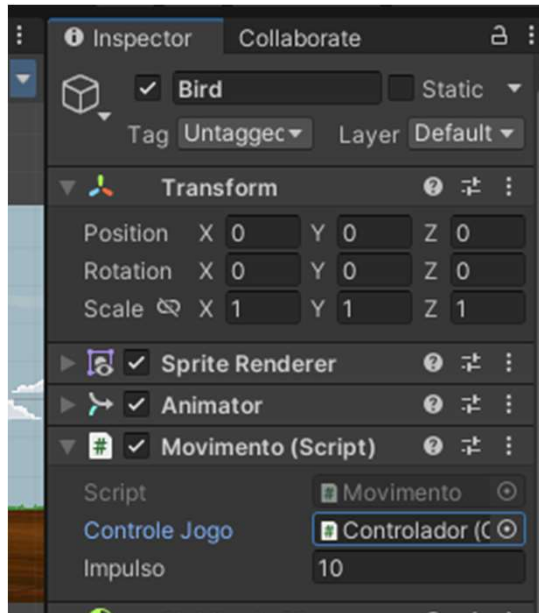
Vamos trabalhar
na janela do Unity



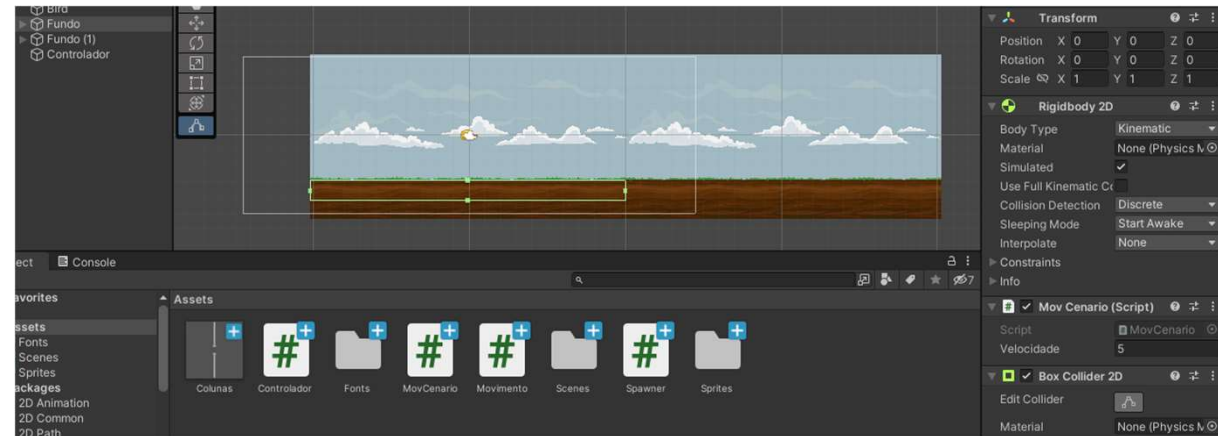
Ajustando alguns problemas

Selecionar o Bird, verificar que o Controle Jogo -> None
Vamos adicionar um colisor no solo. Selecione Fundo e add
Component -> Box Collider 2D / Ajustar o Edit Collider
Faça o mesmo para o outro Fundo -> Fundo (1)
Agora nosso Heroi ao cair, bate no chão e morre!!!!

Vamos trabalhar
na janela do Unity



Ajuste o Impulso do Bird para controlar melhor o voo



Ajustando alguns problemas

Vamos criar uma colisão com o chão

Add Component -> ColisaoChao

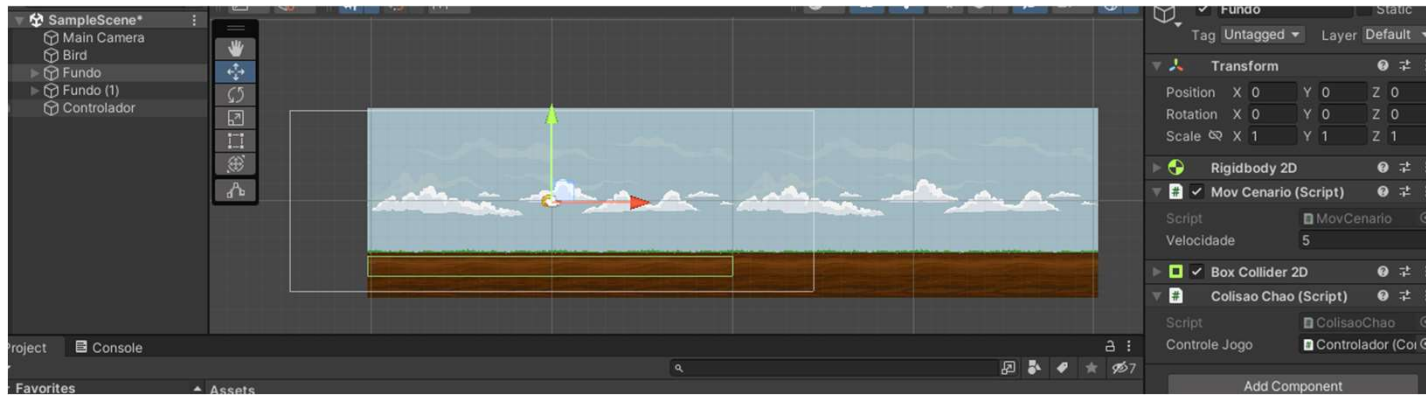
Este Script chama o GameOver

Este macete resolve o problema de quando o pássaro bate na coluna e sim quando ele bater no chão.

Associe o Controlador Jogo a Controlador

Para o segundo Fundo, apenas puxar o ColisaoChao e associar ao jogo

Vamos trabalhar
na janela do Unity



Últimas alterações

Vamos resolver o Controlador

Tirar os 3 comentários:

- Dentro do método Play()
- Dentro do método GameOver()
- Dentro do método BirdScored()

```
private void Awake()
{
    //    spawner = GetComponent<Spawner>();

    Play();
}
void Pause()
{
    Time.timeScale = 0f;
    player.enabled = false;
}

public void Play()
{
    score = 0;
    scoreText.text = score.ToString();

    playButton.SetActive(false);
    gameOver.SetActive(false);

    Time.timeScale = 1f;
    player.enabled = true;
}

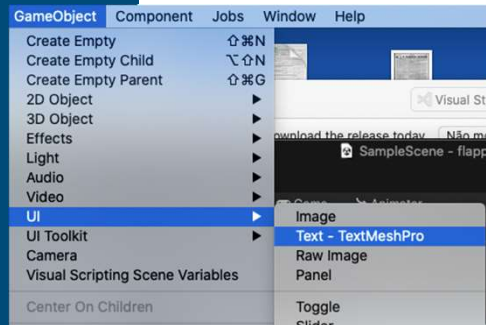
public void GameOver()
{
    playButton.SetActive(true);
    gameOver.SetActive(true);
    Pause();
}

public void BirdScored()
{
    score++;
    scoreText.text = score.ToString();
}

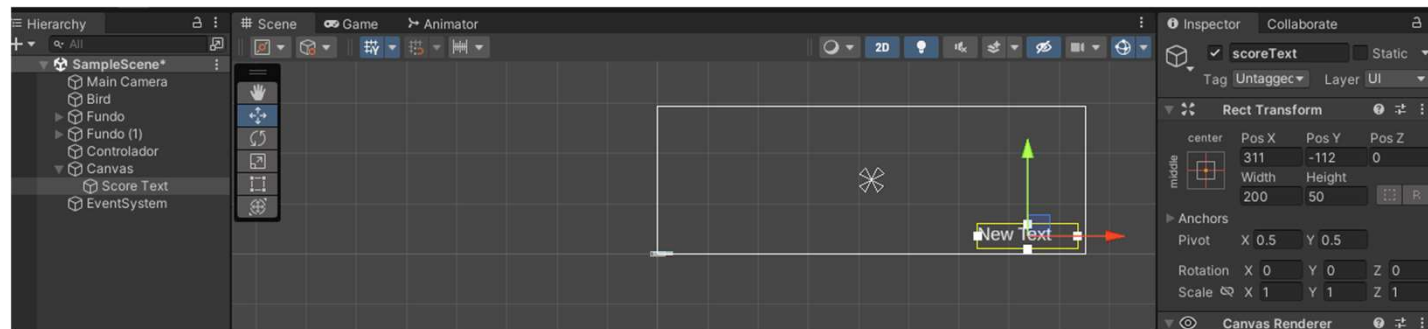
public void RestartGame()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
}
```

Vamos trabalhar
na janela do
Visual Studio

Criando os Components Visuais



Adicione um GameObject ->UI-> Text – TMP
Este componente será responsável por receber a pontuação.
Troque o nome para scoreText



Vamos trabalhar
na janela do Unity

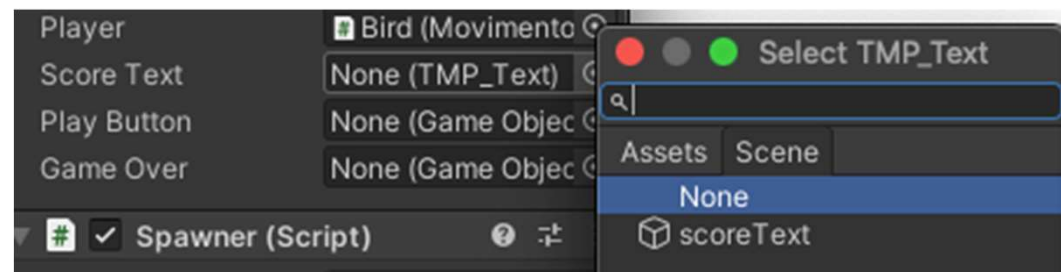
Criando os Components Visuais

Vamos acertar o tipo do texto no Controlador
O Unity descontinuou o Text passando a usar o TMP_Text.

Voltar para tela do Unity e na propriedade do Controlador Score Text, escolher scoreText;

Ainda Falta:

- Play Button
- Game Over



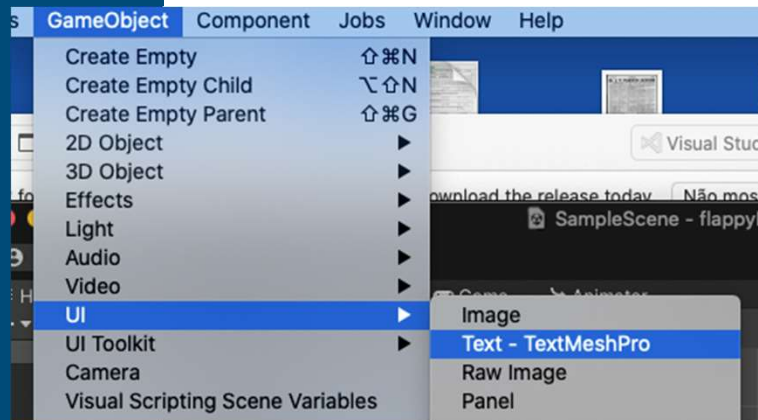
```

1 using System.Collections;
2 using System.Collections.Generic;
3 using TMPro;
4 using UnityEngine;
5 using UnityEngine.SceneManagement;
6 using UnityEngine.UI;
7
8 public class Controlador : MonoBehaviour
9 {
10     public Movimento player;
11     // private Spawner spawner;
12
13     public TMP_Text scoreText;
14     public GameObject playButton;
15     public GameObject gameOver;
16     public int score { get; private set; }
17
18     private void Awake()
19     {
20
21         // spawner = GetComponent<Spawner>();
22
23         Play();
24     }
25     void Pause()
26     {
27         Time.timeScale = 0f;
28         player.enabled = false;
29     }
30
31     public void Play()
32     {
33         score = 0;
34         scoreText.text = score.ToString();
35
36         playButton.SetActive(false);
37         gameOver.SetActive(false);
38
39         Time.timeScale = 1f;
40         player.enabled = true;
41

```

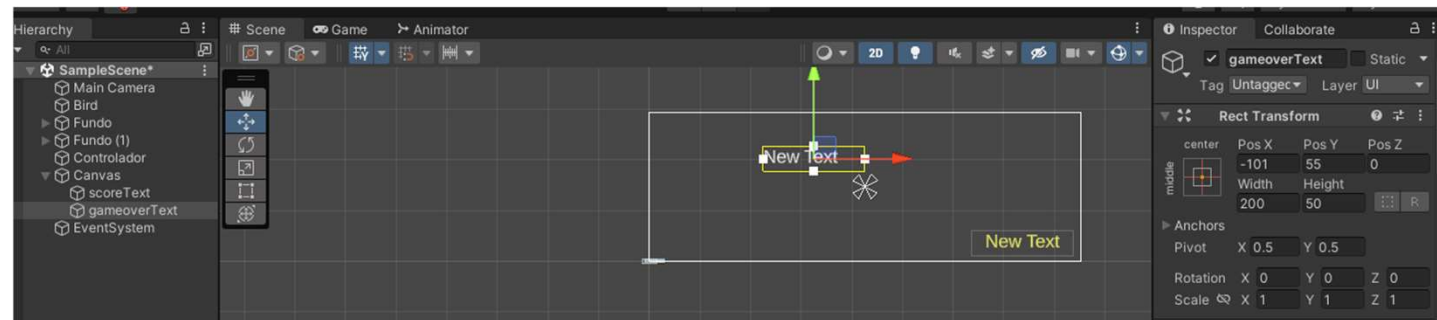
Vamos trabalhar
na janela do
Visual Studio

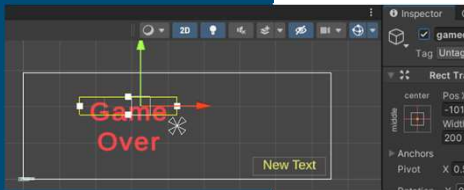
Criando os Components Visuais



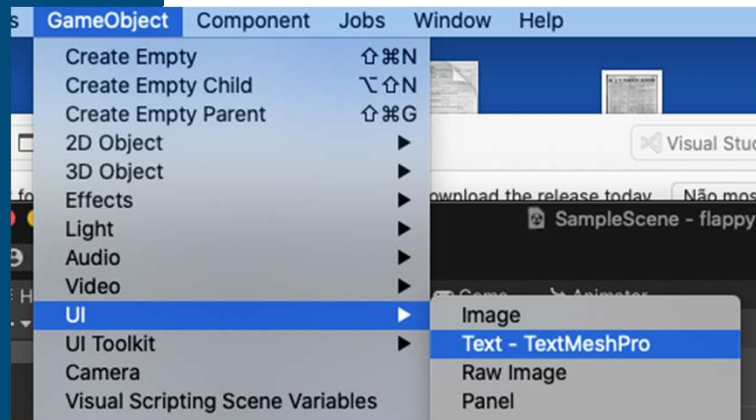
Vamos adicionar um novo GameObject, com o Canvas selecionado
Este Componente deve se chamar gameOverText

Vamos trabalhar
na janela do Unity

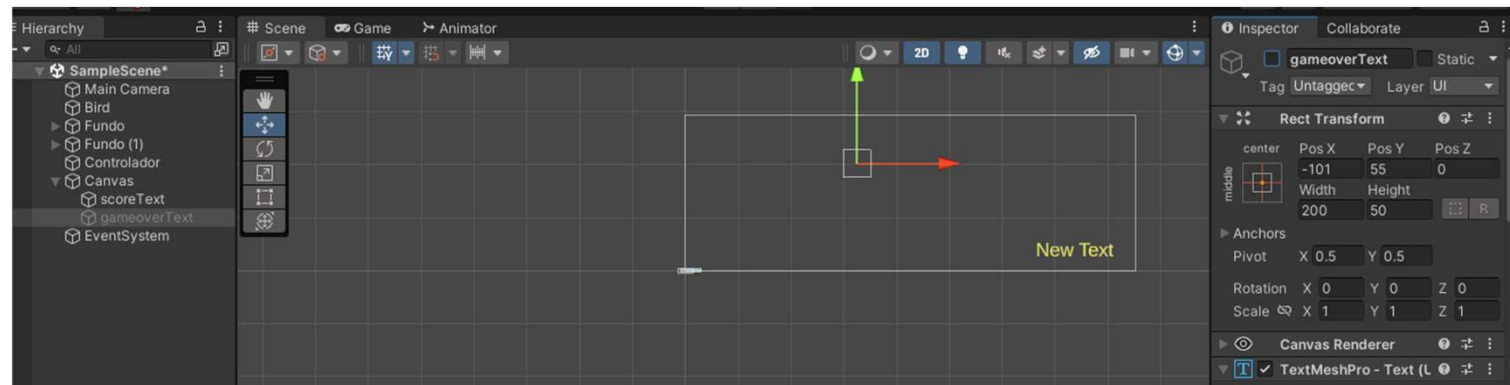




Criando os Components Visuais

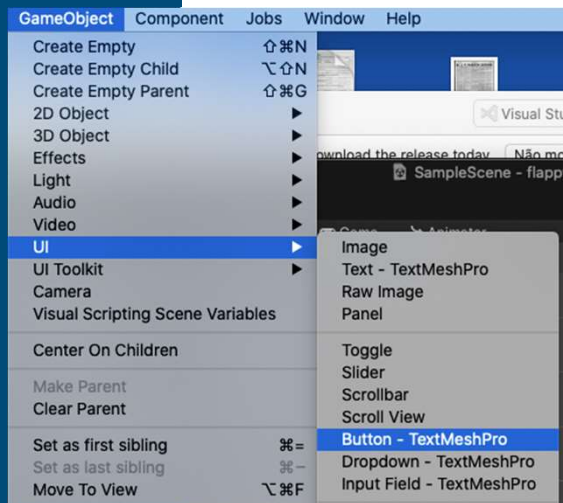


Vamos adicionar um novo GameObject, com o Canvas selecionado
 Este Componente deve se chamar gameOverText
 Em seu texto, colocar Game Over
 Ajuste o tamanho da fonte, cor e características.
 Ao terminar de configurar a fonte, desabilitar para ele não aparecer o tempo todo.



Vamos trabalhar
na janela do Unity

Criando os Components Visuais



Vamos adicionar um novo GameObject, com o Canvas selecionado
Este Componente deve se chamar gameOverText
Dentro do Button, existe um texto que deve ser alterado para RESTART
Deixar tanto o Button quanto o gameOverText desabilitado

Vamos trabalhar
na janela do Unity



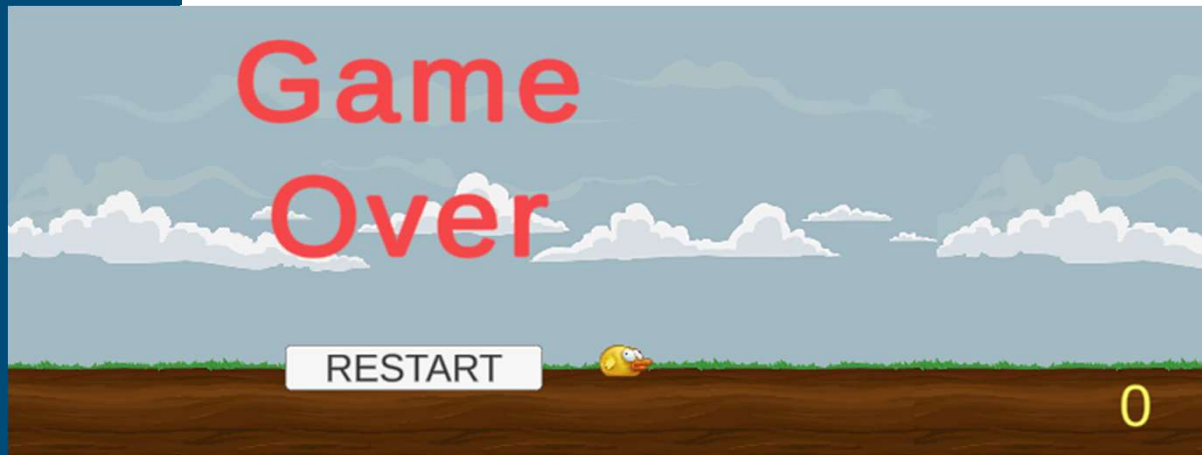
Associando os Components Visuais



Associar:
Play Button -> Button
Game Over -> gameoverText

Vamos trabalhar
na janela do Unity

O que Falta??

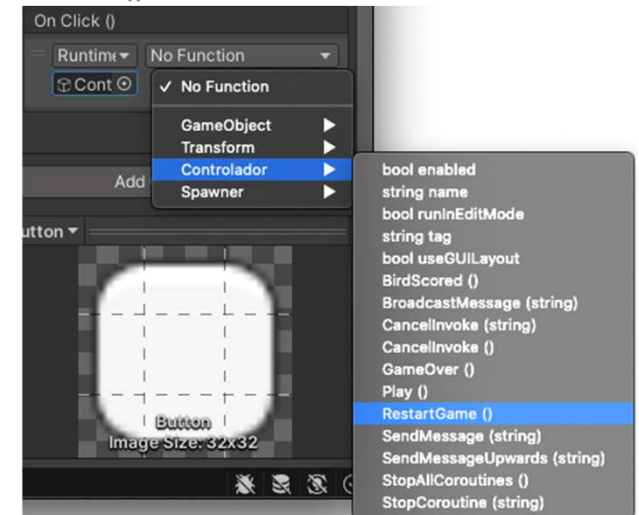
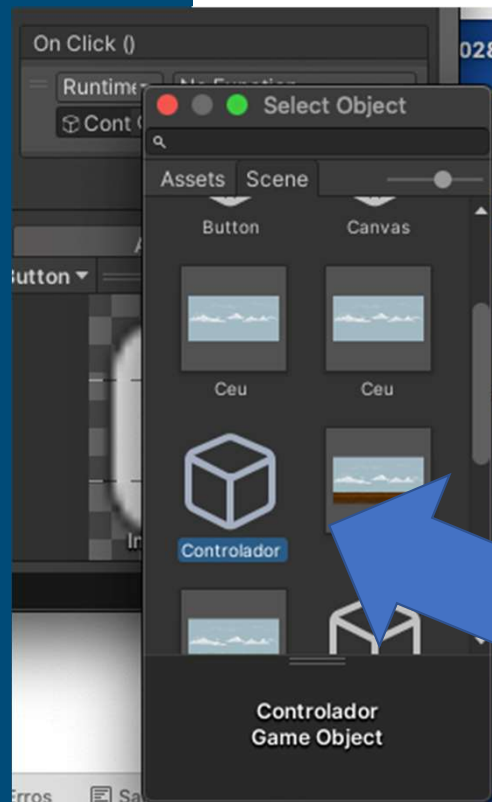


A pontuação já aparece
Quando morremos, já aparece a mensagem
Só falta o botão RESTART

Vamos trabalhar
na janela do Unity

Programar o Evento do Botão

No Botão, rolar o Inspector até os eventos;
No Evento On Click, adicionar mais um;
Tipo Runtime
Clique na bolinha para escolher da Scene, o Game
Object Controlador
Agora, selecione o método RestartGame() do
Controlador

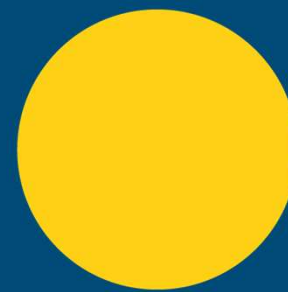


Vamos trabalhar
na janela do Unity

O jogo está Completo!!!

O Botão RESTART não está funcionando mas todo o resto do jogo flui perfeito

UVA



RECONHECIMENTOS:

