Tutorial para Criar um Flappy-Bird

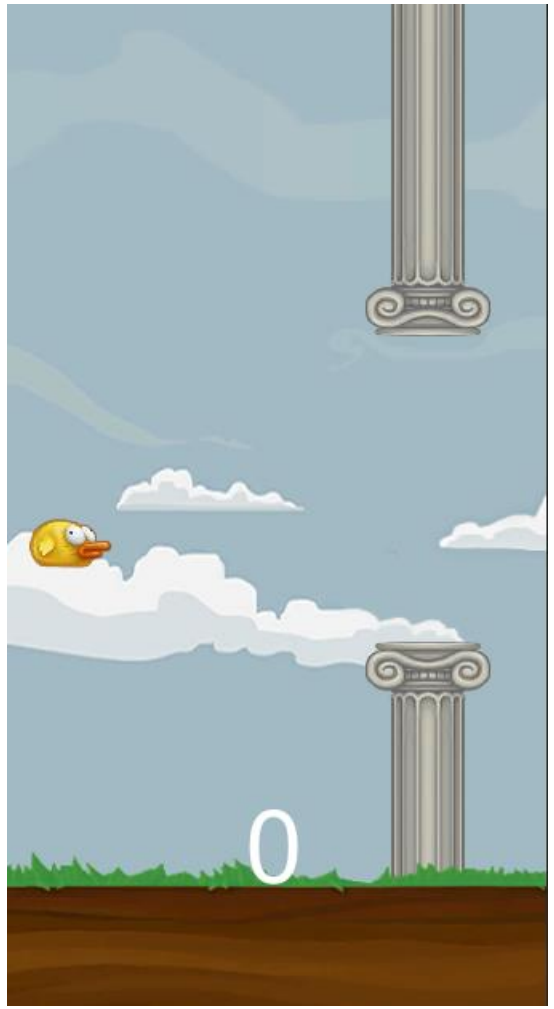
Professor Doutor Flávio Miranda de Farias

Rio Branco – Junho de 2024

fmflavio@gmail.com

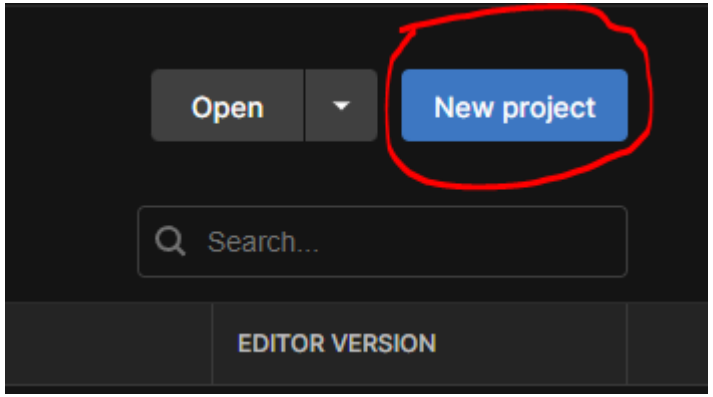
www.fmflavio.com.br

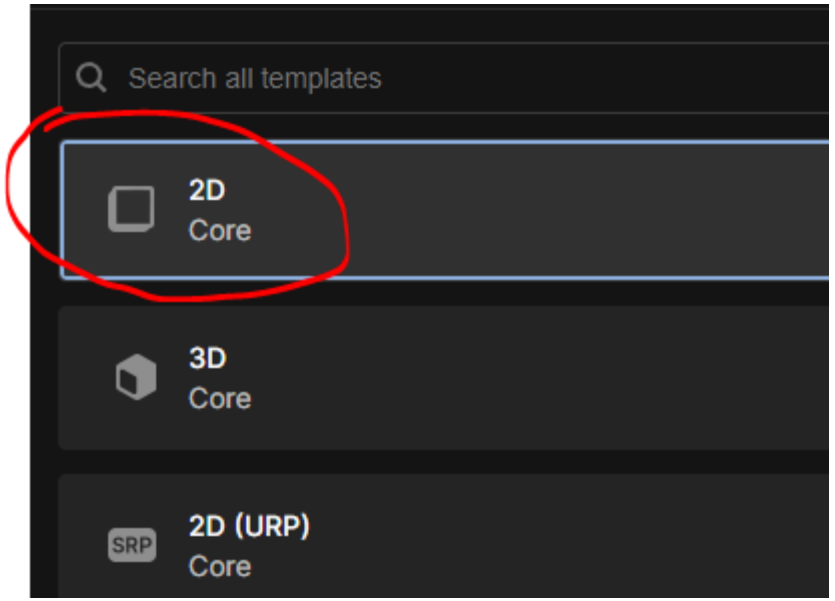
O nosso jogo Flappy-Bird pra celular

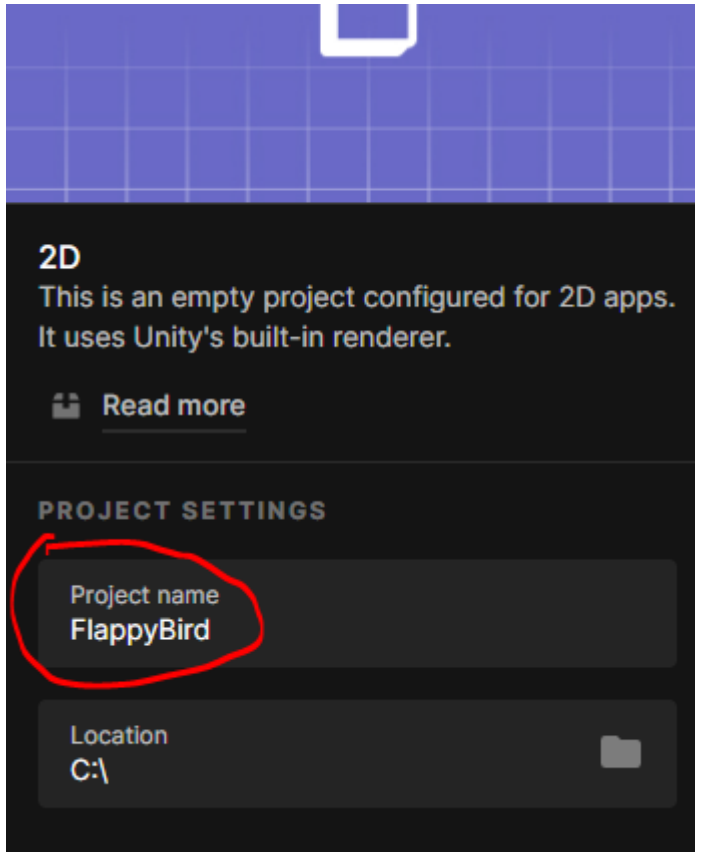


Para baixar o projeto completo deste jogo acesse o repositório:
https://github.com/fmflavio/Flappy-Bird-Exemplo

Criando o Projeto

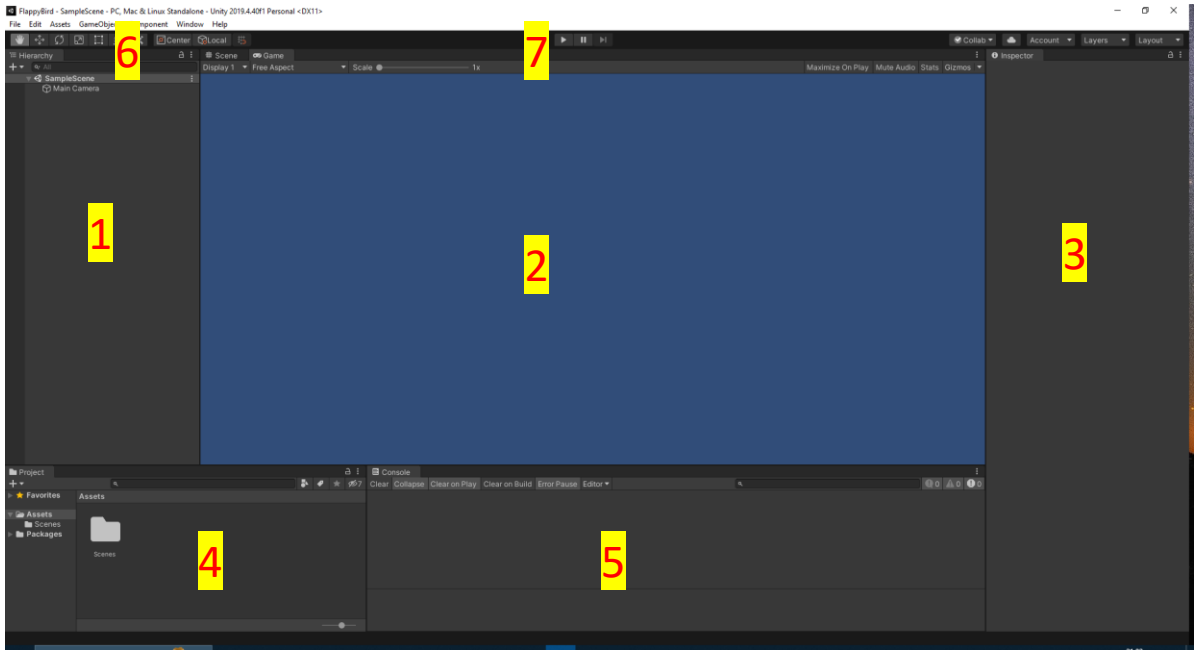






Agora aguardo o projeto ser criado

Com o Projeto Inicial Criado



1 – Painel de Hierarquia dos objetos;

2 – Painel da aba de jogo (**Game**) e aba de gerenciamento de cena (**Scene**) (**podem ser divididos para melhorar a visualização**);

3 – Painel do inspetor de objetos do jogo;

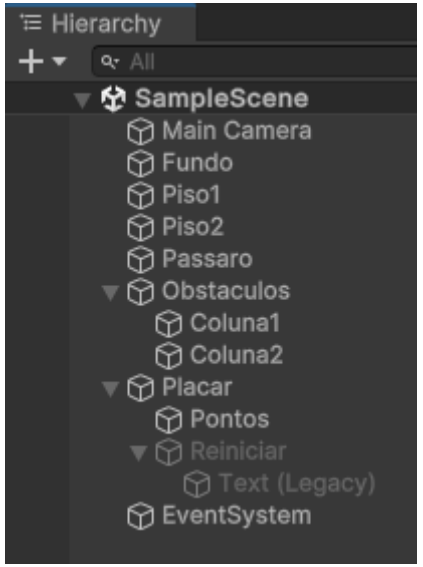
4 – Arvore de arquivos do projeto;

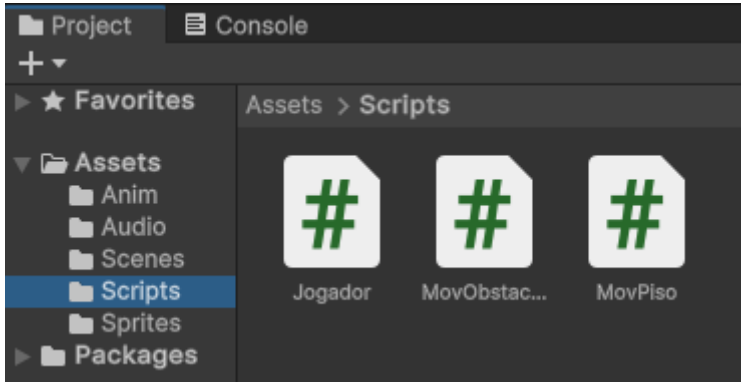
5 – Console para debug do jogo;

6 – Menu da Unity e ferramentas de manipulação da cena; e

7 – Compilador/play do jogo.

Capturas do Final de Projeto

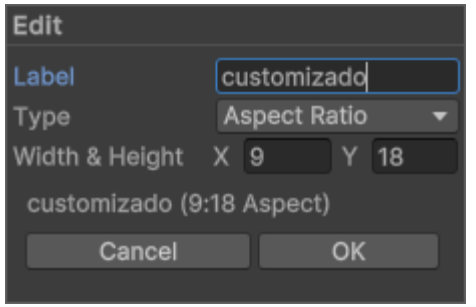




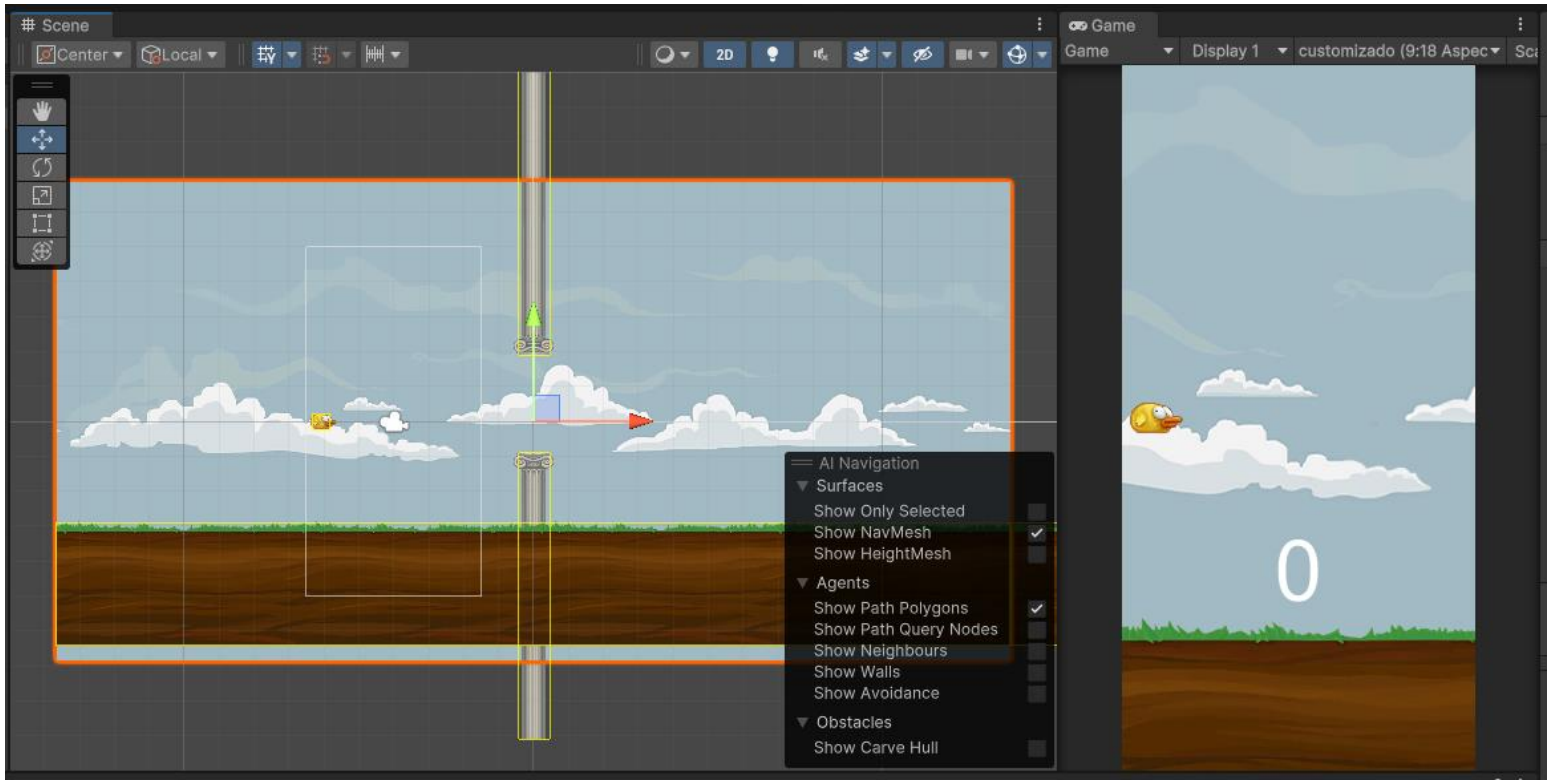
Vamos Importar Manualmente Alguns Arquivos

- Em Link dos Arquivos, baixe o pacote em zip e vamos extrair a pastas Sprites e Audio, para a raiz da arvore do projeto (4).

Vamos Preparar a Tela



- Na aba Game (em 2) em Free Aspect, mude o aspecto da tela para 9:18;
- Para este projeto, recomenda-se dividir (2) em duas telas para interagir melhor e ver como esta se comportamento os objetos do jogo.



Vamos Preparar o Cenário

- Para a aba Scene (em 2):
  - (Em 1) Crie um GameObject Empty (vazio), renomeio para “Cenario”;
  - Arraste o sprite “SkyTileSprite” (em 4) para Hierarchy (em 1) e solte, aperte F2 e renomeie para “Fundo”;
  - Arraste o sprite “GrassThinSprite” para Hierarchy e solte e renomeie para “Piso1”;
  - Duplique o Piso1 (em 1) e renomeie com “Piso2”, posicionando ao lado de Piso1;
  - Arraste o sprite “BirdHeroSprite” para Hierarchy e solte e renomeie para “Passaro”;
  - Crie um GameObject Empty (vazio) (em 1), renomeio para “Obstaculos”;
    - Para Obstaculos, arraste o sprite “ColumnSprite” e solte e renomeie para “Coluna1”, posicionando em baixo;
    - Duplique Coluna1 e crie “Coluna2” posicionando em cima invertido (Rotation 180);
  - Ao final defina os valores de Z em Transform->Position (em 3) para que os Sprites fiquem distribuídos segundo as camadas desejadas.

Vamos Criar as Colisões do Piso

- Para os Pisos, adicione “Box Collider 2D” para Piso1 e Piso2;
- Em Box Collider 2D, ajuste Size para os limites de colisão desejados ou em Edit Collider;
- Para Piso1 e Piso2 vamos criar um apelido (em 3), em Tag->Add Tag clique em + e salve um novo rótulo “Obstaculo”;
  - Va em Piso1 e Piso2 e altere os Tag para Obstaculo.

Vamos Criar as Colisões do Objeto Passaro

- Para o Jogador, adicione “Rigidbody 2D” e “Circle Collider 2D”;
- Em Circle Collider 2D, ajuste Size para os limites de colisão desejados.

Vamos Criar as Colisões dos Obstaculos

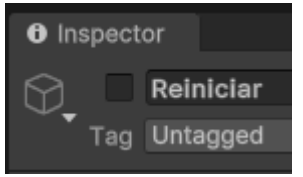
- Para os Obstaculos, adicione “Box Collider 2D” para Coluna1 e Coluna2;
- Em Box Collider 2D, ajuste Size para os limites de colisão desejados ou em Edit Collider;
- Caso não tenha criado ainda, para Coluna1 e Coluna2 vamos criar um apelido (em 3), em Tag->Add Tag clique em + e salve um novo rótulo “Obstaculo”;
  - Va em Coluna1 e Coluna2 e altere os Tag para Obstaculo.

Vamos Animar o Pássaro

- Em Project (em 4), crie uma nova pasta chamada Anim;
- Vamos fazer da forma mais fácil, vá em Sprits (em 4) selecione o sprite do pássaro e copie ou mova para Animin (em 4);
- Agora, vá em Animin (em 4) e selecione as 2 sprites do pássaro voando, com elas selecionadas arraste para cima do Passaro (em 1) e solte;
  - Se tudo ocorrer certo, vai abrir uma janela para salvar uma animação, salve com o nome de “Voar”;
  - Com isso será criado uma animação e controlador com nome Voar em Animin;
  - Além disso será criado um componente Animator (em 3);
  - Com isso ao clicar em play será possível ver a animação de voou.

Vamos Criar a Interface do Usuário (IU)

- Para a pontuação (em 1), clique com direito UI->Text ou UI->Legacy->Text, renomeie Text para “Pontos” e Canvas para “Placar”;
  - Clique em Pontos e altere o texto para 0, adapte o posicionamento do texto e a formatação.
- Para o reiniciar após o fim de Jogo (em 1), clique com direito UI->Button ou UI->Legacy-> Button, renomeie Button para “Reiniciar”;
  - Clique em Reiniciar, adapte o posicionamento do botão em Transform X=0 e y=0 para centralizar;
  - Ainda em Reiniciar (em 3), em On Click () clique em +, arraste Passaro (em 1) para caixa em On Click (), após isso, na caixa ao lado adicione o método Jogador->reiniciar();
  - Clique em Reiniciar->Text e altere o texto para “REINICIAR”, adapte a formatação do texto;
  - Clique em Placar->Reiniciar, e desabilite a visualização no Inspector (em 3).



Pronto, Agora Vamos Criar os Scripts do Jogo

- Em Project (em 4), crie uma nova pasta chamada Scripts;
- Nesta pasta criaremos 3 scripts, com nome de MovPiso, MovObstaculos e Jogador, os nomes já sugere as funções;
- Para cria-los, basta clicar com o botão direito na pasta Scripts escolher Create > C# Script;
- No Script MovObstaculo será necessário importar `using UnityEngine.UI;` e no Jogador `using UnityEngine.SceneManagement;`
- Agora vamos ver cada um dos códigos, tendo em mente que:
  - Os métodos Start() serão sempre executados no início e apenas 1 vez;
  - Os métodos Update() funcionam após o Start() em loop infinito e com frequência variável.

Script MovPiso

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

public class MovPiso : MonoBehaviour {
    public float velocidadePiso;
    void Update() {
        transform.Translate(velocidadePiso *
Time.deltaTime * Vector2.left);
        if (transform.position.x < -10.0f)
            transform.position = new
Vector2(10.0f, -3);
    }
}
```

- As linhas 1 a 3 são padrão e não precisam ser alteradas;
- Linha 5 deve conter o nome da Classe MovPiso igual a do arquivo;
- Linha 6 refere-se a velocidade horizontal de deslocamento do piso;
- Linha 7 função de repetição variável;
- Linha 8 altera a posição no eixo em decorrência da variação do tempo para esquerda;
- Linhas 9 e 10 verifica se o piso alcançou uma posição no eixo x e reinicia a exibição;
- Os valores de x e y vão variar de conforme a posição inicial de Piso2.

Script MovObstaculos

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


public class MovObstaculos : MonoBehaviour {
    public float velocidadeObstaculos;
    public int pontuacao = 0;
    public Text textoPontuacao;
    void Update() {
        if (transform.position.x < -4f) {
            transform.position = new Vector2(3.0f,
Random.Range(-7f, -11f));
            pontuacao++;
            textoPontuacao.text = pontuacao.ToString();
        }
        transform.Translate(velocidadeObstaculos *
Time.deltaTime * Vector2.left);
    }
}
```

- As linhas 7 a 9 cria-se variáveis publicas globais;
- As linhas 8 e 9 serão usadas na Interface para Usuário (IU);
- As linhas 11 e 12 verifica se os obstáculos alcançaram uma posição no eixo x e reinicia a exibição;
- As linhas 13 e 14 atualizam a Interface para Usuário;
- A linha 16 altera a posição no eixo em decorrência da variação do tempo para esquerda;
- Os valores nas linhas 11 e 12 variam e devem ser ajustado para seu projeto.

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

public class Jogador : MonoBehaviour {
    public float subir;
    public Rigidbody2D rigid;
    public Animator anim;
    public GameObject UIgameover;
    void Start() {
        Time.timeScale = 1.0f;
        rigid = GetComponent<Rigidbody2D>();
    }
    void Update() {
        if (Input.GetMouseButtonDown(0))
            rigid.velocity = Vector2.up * subir;
    }
    private void OnCollisionEnter2D(Collision2D collision){
        if (collision.gameObject.tag == "Obstaculo")
            GameOver();
    }
    public void GameOver() {
        anim.SetTrigger("Fim de Jogo");
        UIgameover.gameObject.SetActive(true);
        Time.timeScale = 0.0f;
    }
    public void Reiniciar() {
        SceneManager.LoadScene("SampleScene");
    }
}
```

Script Jogador

- As linhas 7 a 10 cria-se variáveis publicas globais;
- As linhas 11 a 10 inicia a função construtora, defini o tempo inicial de contagem em 1 e instancia um objeto com rigidez de nome rigid;
- As linhas 15 a 18 responde ao clique do mouse com uma impulsão para cima do objeto rigid;
- As linhas 19 a 22 verificam se houve uma colisão com algum objeto de rótulo Obstaculo e chama o método próprio para o tratamento;
- As linhas 13 a 27 tratam a colisão mudando a animação do personagem, exibindo a tela de IU e reiniciando o tempo;
- As linhas 28 a 30 recarrega a cena quando invocado o método.

Agora Vamos Incluir os Scripts aos Objetos

- Clique no Piso1, depois clique em Add Component, digite Movpiso e confirme, modifique para velocidade 2 (em 3), repita o processo para Piso2;
- Clique no Obstaculos, depois clique em Add Component, digite MovObstaculos e confirme, (em 3) modifique para subir 5, adicione em Anim adicione Passaro e Pontos em Texto Pontuacao;
- Clique no Jogador, depois clique em Add Component, digite Jogador e confirme, (em 3) modifique para subir 5 e adicione em Anim adicione Passaro.

Inserindo Música de Fundo

- Para inserir uma música de fundo, abra a pasta Audio (em 4), clique no áudio e arraste para cima da Main Camera (em 1) e solte;
  - Será criado um Audio Source (em 3) vinculado a câmera;
  - No Audio Source ative o Loop.

Atividade Extra

Melhore seu jogo!

Professor Doutor Flávio Miranda de Farias
Rio Branco – Junho de 2024
fmflavio@gmail.com
www.fmflavio.com.br