Tutorial para Criar um Arcanoid

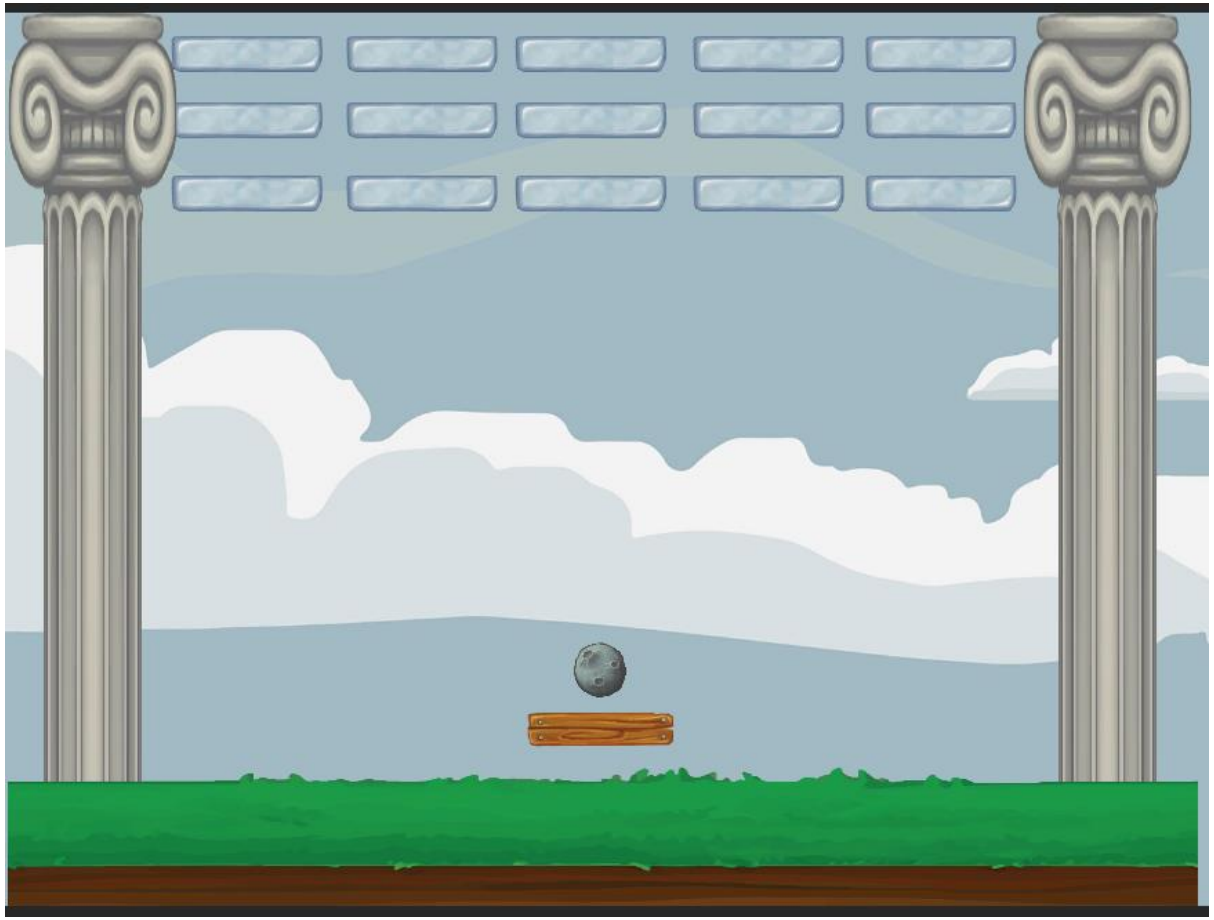
Professor Doutor Flávio Miranda de Farias

Rio Branco – Junho de 2024

fmflavio@gmail.com

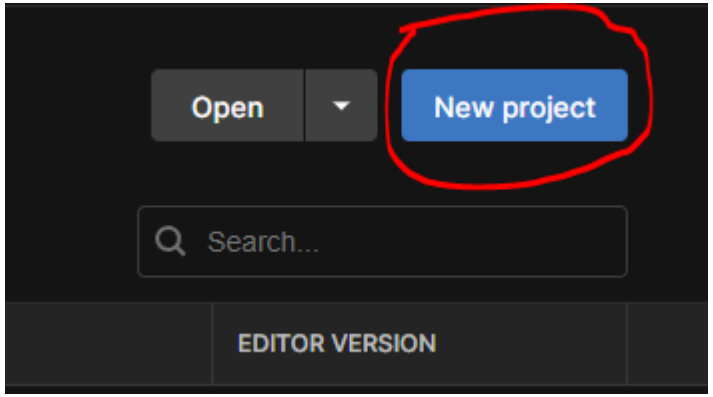
www.fmflavio.com.br

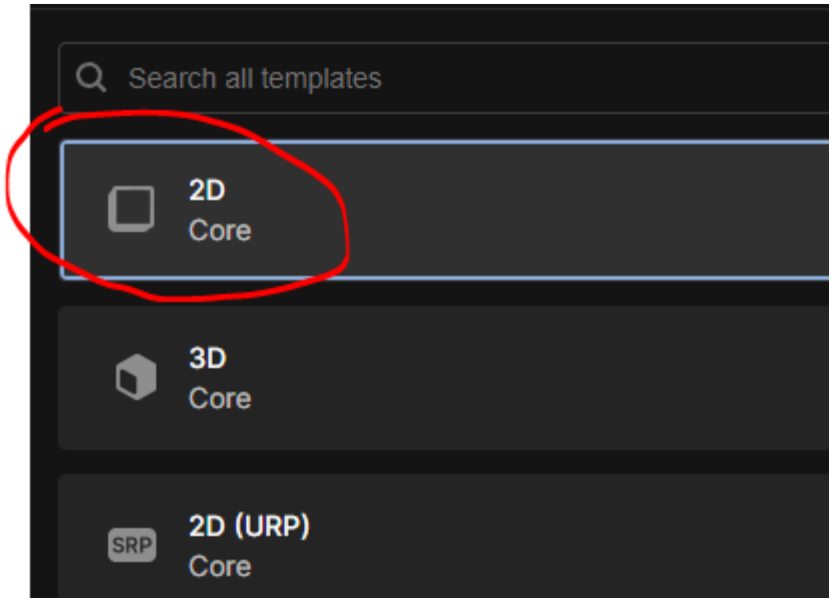
Jogo Nosso Jogo Arcade Arcanoid

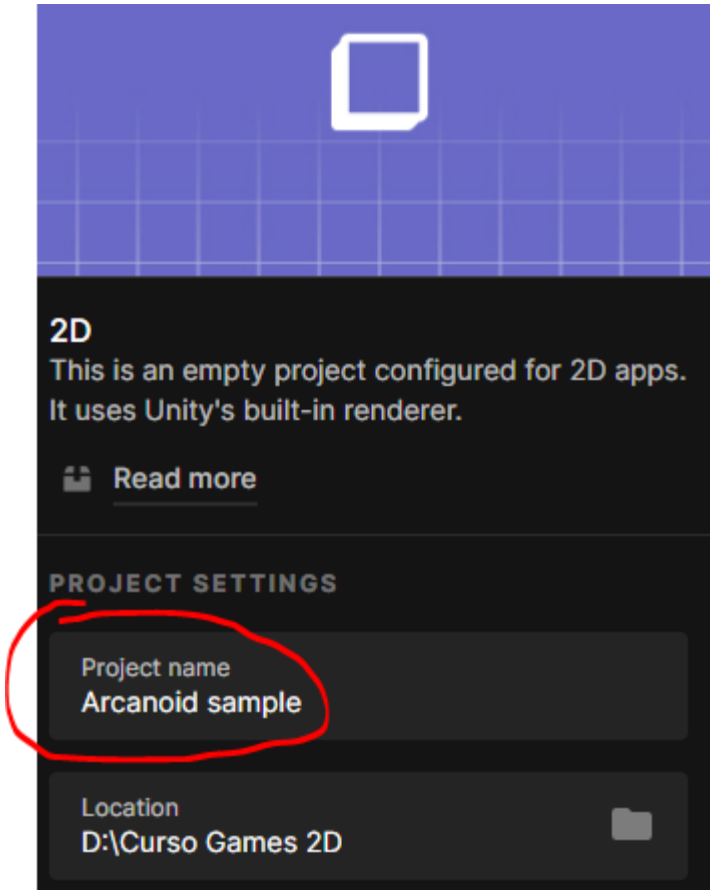


Para baixar o projeto completo deste jogo acesse o repositório:
https://github.com/fmflavio/Arcanoid---treino

Criando o Projeto

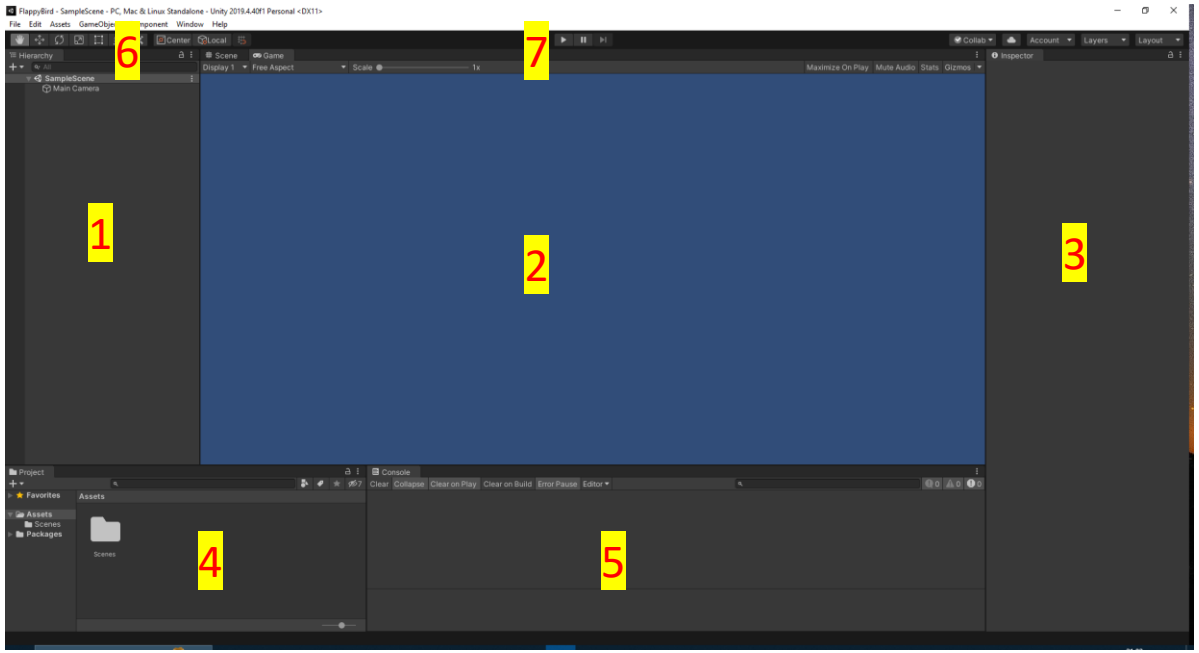






Agora aguardo o projeto ser criado

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;

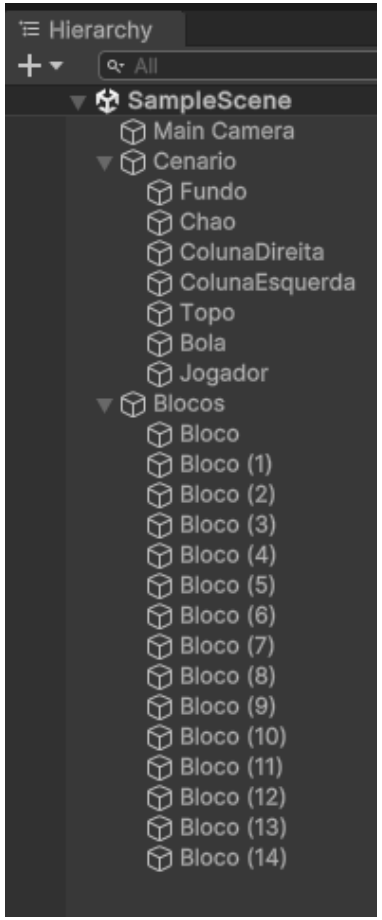
4 – Arvore de arquivos do projeto;

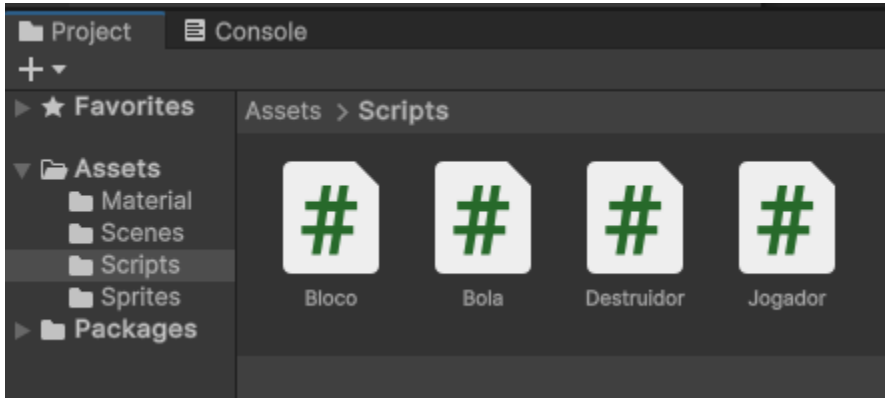
5 – Console para debug do jogo;

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

7 – Compilador/play do jogo.

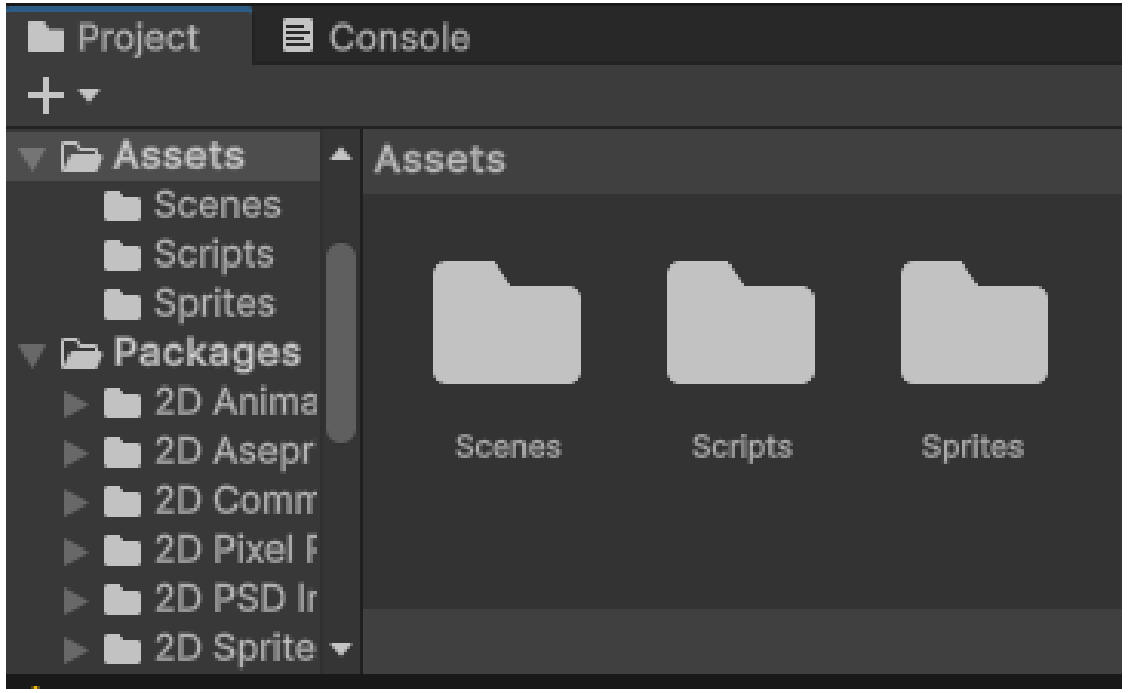
Capturas do Final de Projeto



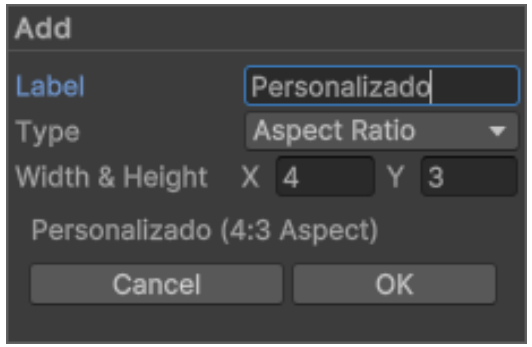


Vamos Importar Manualmente Alguns Arquivos

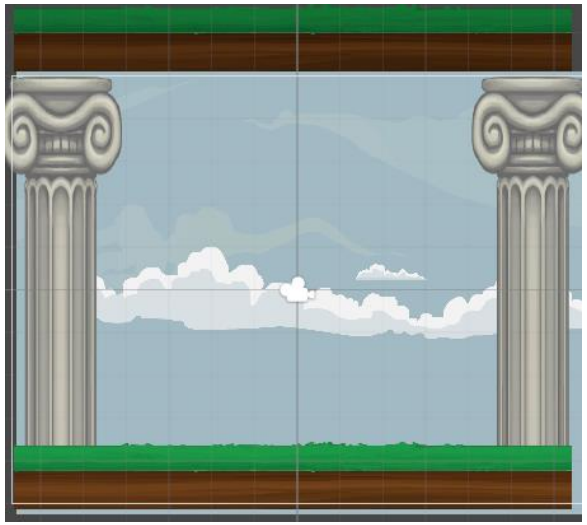
- Em Link dos Arquivos , baixe o pacote em zip e vamos extrair as pastas Sprites e Scripts, para a raiz da arvore do projeto (4).



Vamos Preparar o Cenário



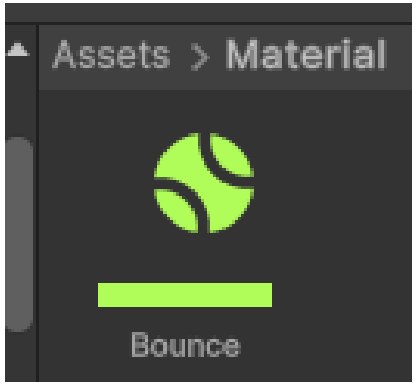
- Na aba Game (em 2) em Free Aspect, mude o aspecto da tela para 4:3, para usarmos o padrão de tela dos antigos arcades;
- Na aba Hierarcht (em 1):
  - Crie um GameObject Empty (vazio), renomeio para “Cenario”;
  - Agora vamos inserir as imagens e ir ajustando na aba Scene pra formar o
  - cenário do jogo;
  - Na pasta Sprite, arraste o sprite “SkySprite” (em 4) para cima do
  - GameObject (Cenario) (em 1) e solte, aperte F2 e renomeie para “Fundo”;



  - Arraste o sprite “GrassSprite” para cima do GameObject (Cenario) e solte, aperte F2 e renomeie para “Chao”, após redimensione até ajustar a tela de jogo;
  - Arraste dois sprite “ColumnShortSprite” para cima do GameObject (Cenario) e solte, aperte F2 e renomeie para “ColunaDireita” e “ColunaEsquerda”, após redimensione até ajustar a tela de jogo;
  - Duplique o Objeto “Chao” em Cenário e renomeie para “Topo” e coloque em cima.
- Recomenda-se ir clicando em play para ver como esta sendo composto o cenário.

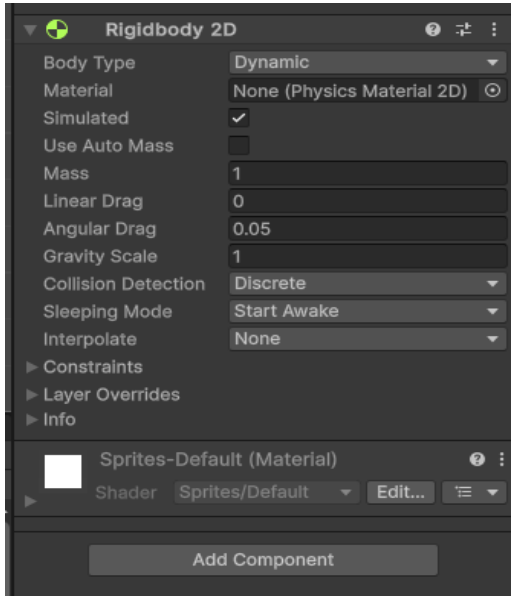
Vamos criar um material físico quicável

- Em Project (em 4), na pasta Assets, vamos criar um material Bounce (quicavel), pra isso na arvore do projeto, crie uma nova pasta (Folder) chamada Material;
- Na pasta Material, crie (botão direito->Create) um novo “Physics Material 2D” em Create->2D e renomeie pra “Bounce”;
- O ajuste de Bounce, fica a seu critério, mas vamos iniciar com os dados Dynamic Friction 0 e Bounciness 1 (com isso temos um material quicavel);



Vamos Criar as Colisões do Topo

- Selecione o Topo em no Inspector (em 3), clique me Add Component para adicionarmos os componentes “Rigidbody 2D” e “Box Collider 2D”;
- Posicione a caixa de colisão a cima do cenário utilizando a propriedade Transform do Inspetor (em 3) ou as ferramentas em 6;
- Em Rigidbody 2D, em Constraints ative X, Y e Z para ficarem fixos;
- Em Rigidbody 2D, em Mass (massa física), coloque 100;
- Em Box Collider 2D, ajuste Size para os limites de colisão desejados ou em Edit Collider.
- Agora clicando em Topo, podemos adicionar o material Bounce através de Box Collider 2D em Material.



Vamos Criar as Colisões das Paredes/Colunas

- Para as paredes, adicione “Rigidbody 2D” e “Box Collider 2D”;
- Em Rigidbody 2D, em Constraints ative X, Y e Z para ficarem fixos;
- Em Rigidbody 2D, insira o material Bounce;
- Em Rigidbody 2D, em Mass (Massa), coloque 1000;
- Em Box Collider 2D, ajuste Size para os limites de colisão desejados ou em Edit Collider.

- **Observação**: Nenhum limite de colisão, deve encontar em outro limite de colisão para evitar bugs.

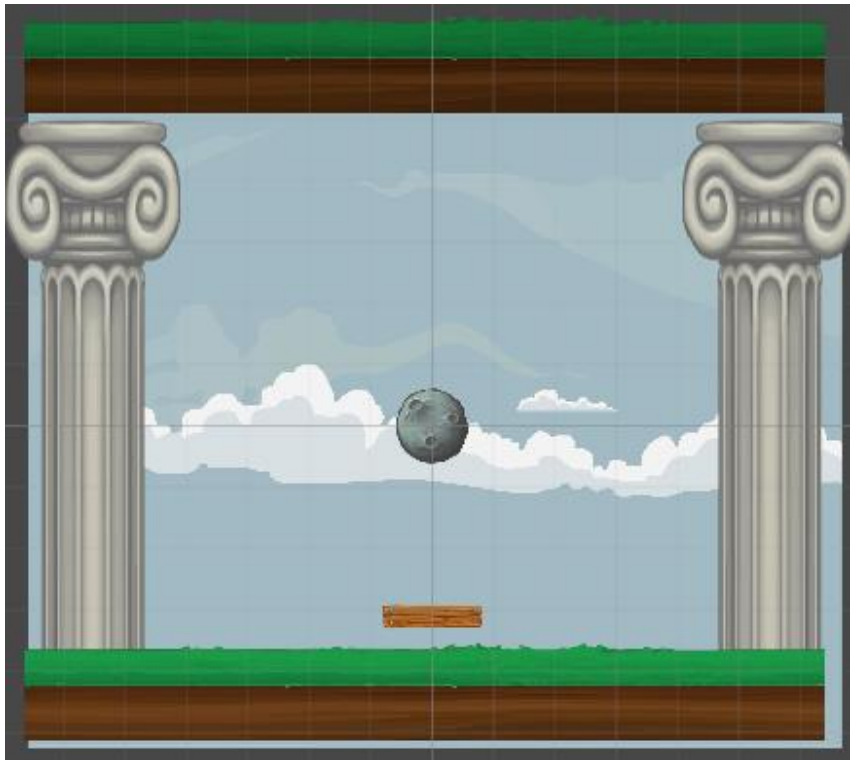
Vamos Criar a Colisão do Piso

- Para o piso, adicione “Rigidbody 2D” e “Box Collider 2D”;
- Em Rigidbody 2D, em Constraints ative X, Y e Z para ficarem fixos;
- Em Box Collider 2D, ajuste Size para os limites de colisão desejados ou em Edit Collider.

Agora Vamos Adicionar a Bola e Jogador

- Na pasta Sprites, arraste o sprite “AsteroidSprite” para cima do GameObject (Cenario) e solte, aperte F2 e renomeie para “Bola”;
- Arraste o sprite “PlankSprite_0” para cima do GameObject (Cenario) e solte, aperte F2 e renomeie para “Jogador”;
- Posicione e redimensione ambos como achar interessante.

- **Ao compilar com play, já será possível ver algumas propriedades físicas funcionando.**



Vamos Configurar o Jogador

- Para o Jogador, adicione “Rigidbody 2D” e “Box Collider 2D”;
- Em Rigidbody 2D, em Constraints **ative somente Y e Z** para ficarem fixos, pois o X será alterado pelo jogador;
- Em Rigidbody 2D, insira o material Bounce;
- Em Rigidbody 2D, em Mass (massa física), coloque 100;
- Em Box Collider 2D, ajuste Size para os limites de colisão desejados.

Vamos Configurar a Bola

- Para a Bola, adicione “Rigidbody 2D” e “Box Collider 2D ou Circle Collider 2D”;
- Em Rigidbody 2D, em Constraints **NÃO** ative X, Y e Z, pois a bola deverar andar livremente pelo cenário;
- Em Rigidbody 2D, em Mass (massa física), coloque 0,1;
- Em Rigidbody 2D, em **Gravity Scale** (gravidade), coloque 0;
- Em Circle Collider 2D, ajuste Size para os limites de colisão desejados.
- Em Circle Collider 2D, insira o material Bounce;

Com o Cenário Pronto Vamos para os Blocos

- Por questão de organização, na raiz da Hierarchy (em 1), vamos criar um GameObject Empty (vazio), renomeio para Blocos;
- Arraste o sprite “IceSprite” para cima do GameObject (Blocos) e solte, aperte F2 e renomeie para “Bloco”, posicionando em cima no cenário;
- Para o Bloco, adicione “Rigidbody 2D” e “Box Collider 2D”;
- Em Rigidbody 2D, em Constraints ative X, Y e Z para ficarem fixos;
- Em Rigidbody 2D, insira o material Bounce;
- Em Box Collider 2D, ajuste Size para os limites de colisão desejados, observe a caixa de colisão.

Pronto, Agora Vamos Criar os Scripts do Jogo

- Em Project (em 4), crie uma nova pasta chamada Scripts;
- Nesta pasta criaremos 4 scripts, com nome de Bloco, Bola, Destruidor e Jogador, os nomes já sugere as funções;
- Para cria-los, basta clicar com o botão direito na pasta Scripts escolher Create > C# Script;
- 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 Bola

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

public class Bola : MonoBehaviour {
    public Vector2 Velocidade;

    void Start(){
        Rigidbody2D rigidbody = GetComponent<Rigidbody2D>();
        rigidbody.AddForce(Velocidade);
    }
    void Update() {
        if (Input.GetKeyDown(KeyCode.R))
            SceneManager.LoadScene("SampleScene");
    }
}
```

- As linhas 1 a 3 são padrão e não precisam ser alteradas;
- Linha 5 deve conter o nome da Classe Bola igual a do arquivo;
- Linha 6 refere-se a velocidade X e Y usada na Bola;
- Linha 8 instancia um objeto com rigidez de nome rigidbody;
- Linha 9 adicionamos a grandeza força X e Y a este objeto Bola.

Script Jogador

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

public class Jogador : MonoBehaviour{
    public float Velocidade = 10.0f;
    public float HorizontalAxix;
    public Rigidbody2D rigidbody;
    private void Start(){
        rigidbody = GetComponent<Rigidbody2D>();
    }
    void Update(){
        HorizontalAxix = Input.GetAxis("Horizontal");
        rigidbody.velocity = new Vector2(Velocidade *
HorizontalAxix, 0);
    }
}
```

- As linhas 6 a 8 criam variáveis publicas globais;
- Linha 10 instancia um objeto com rigidez de nome rigidbody;
- Linha 13 adicionamos uma entrada das setas direita e esquerda do teclado a variável responsável pela posição horizontal do Jogador;
- Linha 14 adiciona o deslocamento horizontal ao objeto do Jogador.

Script Bloco

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

public class Bloco : MonoBehaviour{
    public Rigidbody2D rigidbody;
    public BoxCollider2D colider;
    private void Start(){
        rigidbody = GetComponent<Rigidbody2D> ();
        colider = GetComponent<BoxCollider2D> ();
    }
    private void OnCollisionEnter2D(Collision2D collision){
        rigidbody.constraints = new RigidbodyConstraints2D();
        colider.isTrigger = true;
    }
}
```

- As linhas 6 e 7 criam variáveis publicas globais;
- Linha 9 instancia um objeto com rigidez de nome rigidbody;
- Linha 10 instancia um objeto com colisão de nome colider;
- Linha 12 cria-se um método que reconhece a colisão de objetos 2D;
- Linha 13 adicionamos um novo gerenciador de eixos do Rigidbody;
- Linha 14 seta o acionador (Trigger) pra verdadeiro.

Script Destruidor

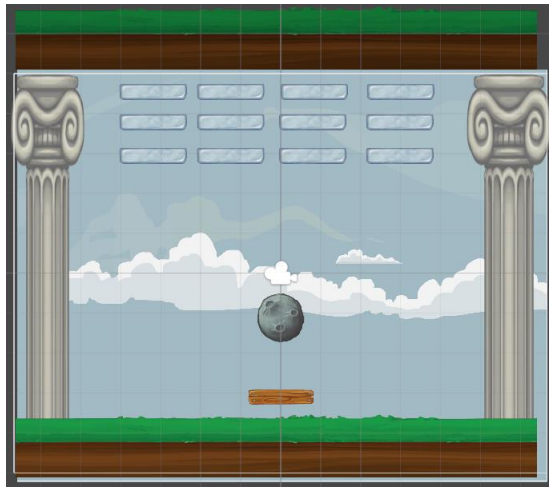
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Destruidor : MonoBehaviour{
    private void OnCollisionEnter2D(Collision2D
collision){
        GameObject collider =
collision.collider.gameObject;
        GameObject.Destroy(collider);
    }
    private void OnTriggerEnter2D(Collider2D
collision){

GameObject.Destroy(collision.gameObject);
    }
}
```

- Linha 6 cria-se um método que reconhece a colisão de objetos 2D;
- Linha 7 cria-se uma referencia com o objeto (bola ou bloco) que colidiu com o Chao;
- Linha 8 destrói o objeto que colidiu com o Chao;
- Linha 10 cria-se um método que verifica se acionador (Trigger) recebeu colisão;
- Linha 11 destoe o que acionou o método.

Agora Vamos Incluir os Scripts aos Objetos

- Clique no Objeto Chao, depois clique em Add Component, digite Destruidor e confirme;
- Clique no Objeto Bola, depois clique em Add Component, digite Bola e confirme;
  - Em velocidade coloque 20 para X e 20 para Y;
- Clique no Objeto Jogador, depois clique em Add Component, digite Jogador e confirme;
- Clique no Objeto Bloco, depois clique em Add Component, digite Bloco e confirme;

Por Fim



- Clone os objetos Bloco e distribua pelo cenário como desejar.
- Um erro comum é não alterar/ajustar a área de colisão dos objetos, caso uma área de colisão se encontre com outra, um elemento pode sumir do cenário ao iniciar o jogo.

- Bonus:
  - Para recarregar o cenário, inclua as seguintes linhas em algum Update() em qualquer script:

```
if (Input.GetKeyDown(KeyCode.R))
            SceneManager.LoadScene("SampleScene");
```

Atividade Extra

Melhore seu jogo!

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