

Oficina de Programação de C# e Introdução ao Unity

Gilseone Moraes



Criando o Projeto



Criando o Projeto

Demonstração



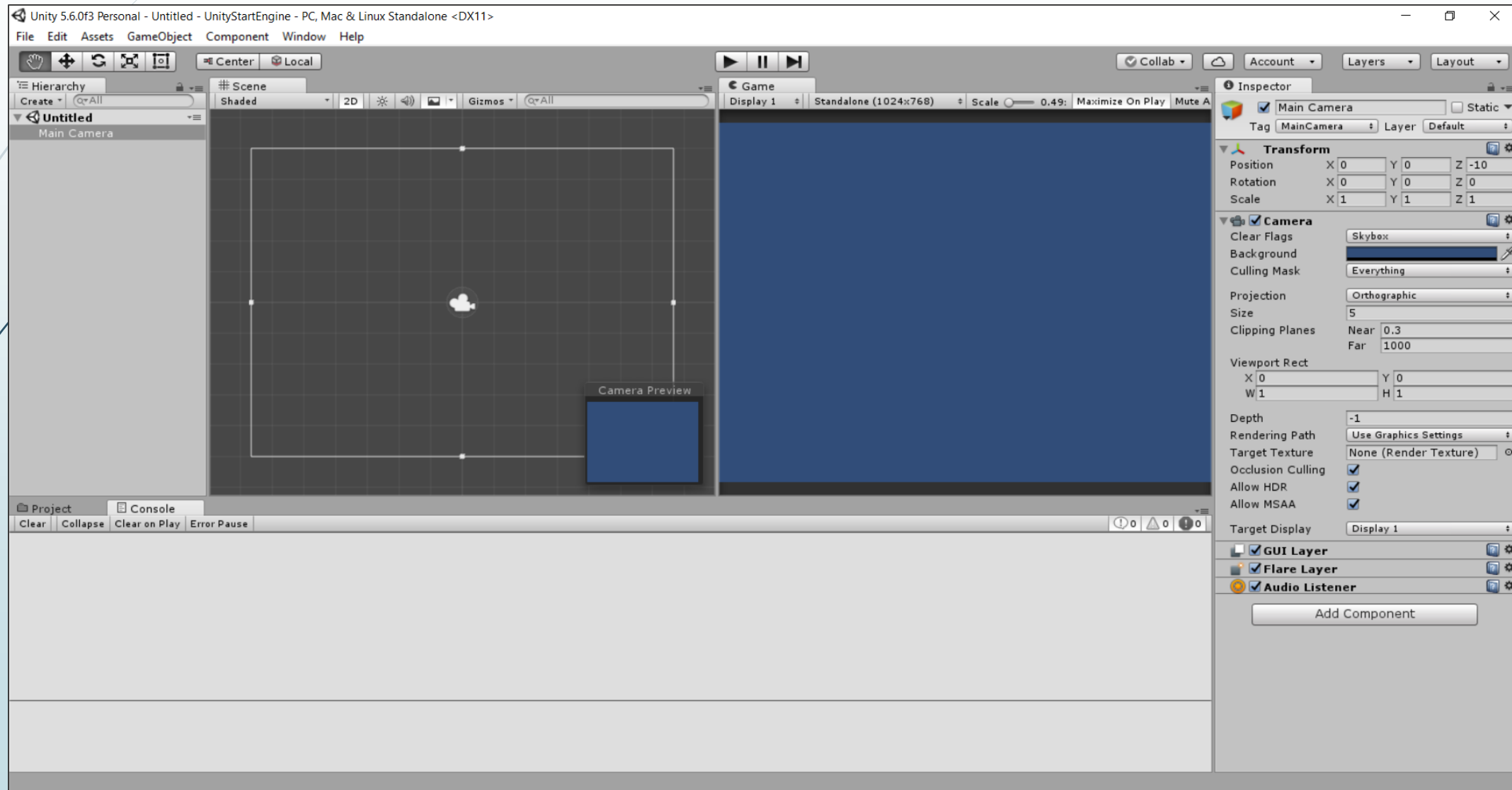
Conhecendo a Engine



Conhecendo a Engine

- Criado o novo projeto você irá ser levado a uma tela parecida com essa:

Conhecendo a Engine





Conhecendo a Engine

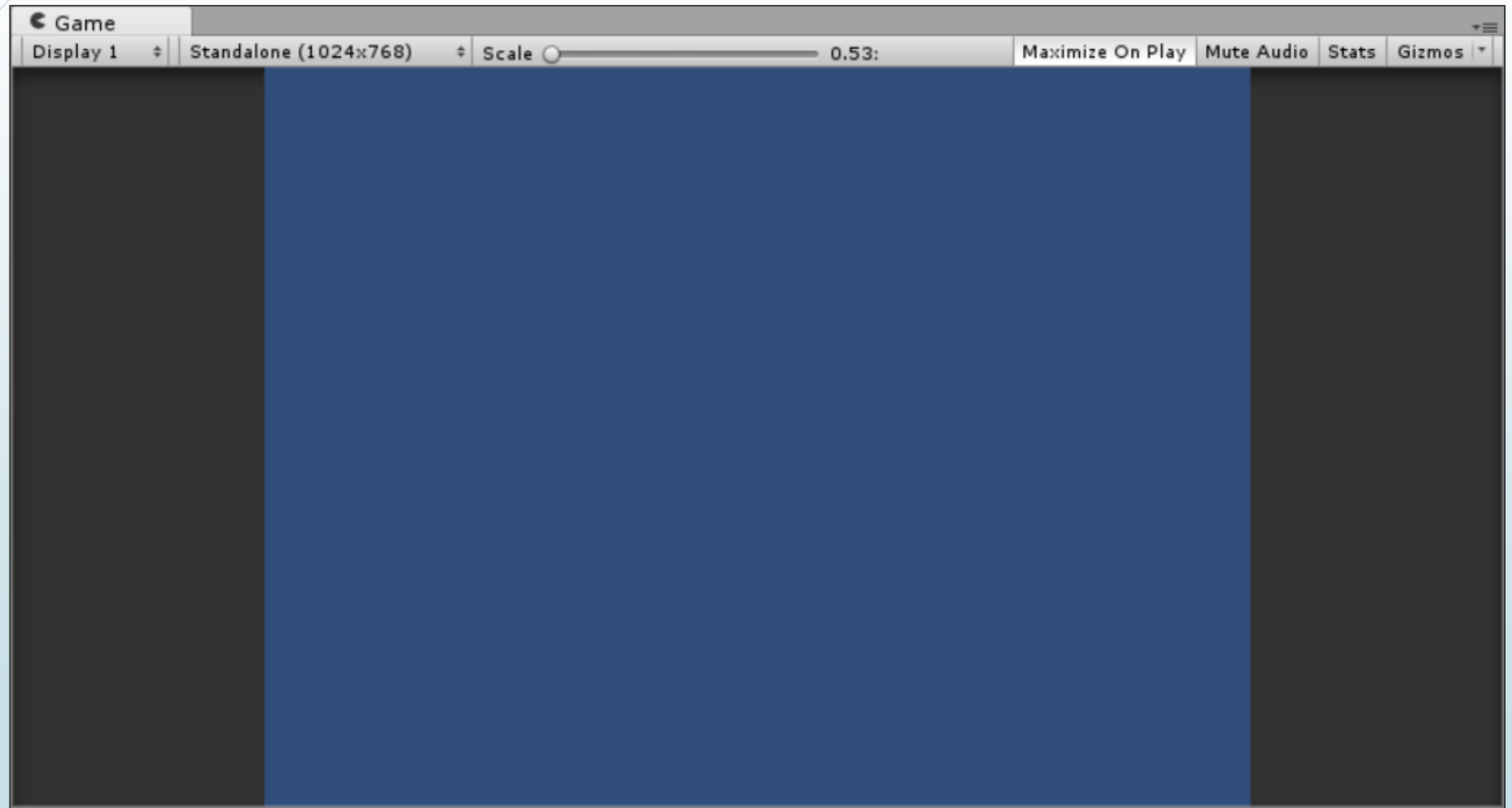
- A localização das abas poderá ser organizada como você achar melhor, mas uma dica é organizar de uma forma que você possa analisar ao mesmo tempo a aba “*Game*”, “*Scene*” e “*Hierarchy*”, pois são de extrema importância para você analisar como o jogo está se comportando.



Conhecendo a Engine

- Caso uma dessas abas não estejam aparecendo para você, poderá habilitá-la na barra de ferramentas lá em cima na opção “*Window*”. Tudo ok? Então vamos à explicação do que é cada uma:

Conhecendo a Engine: Aba Game

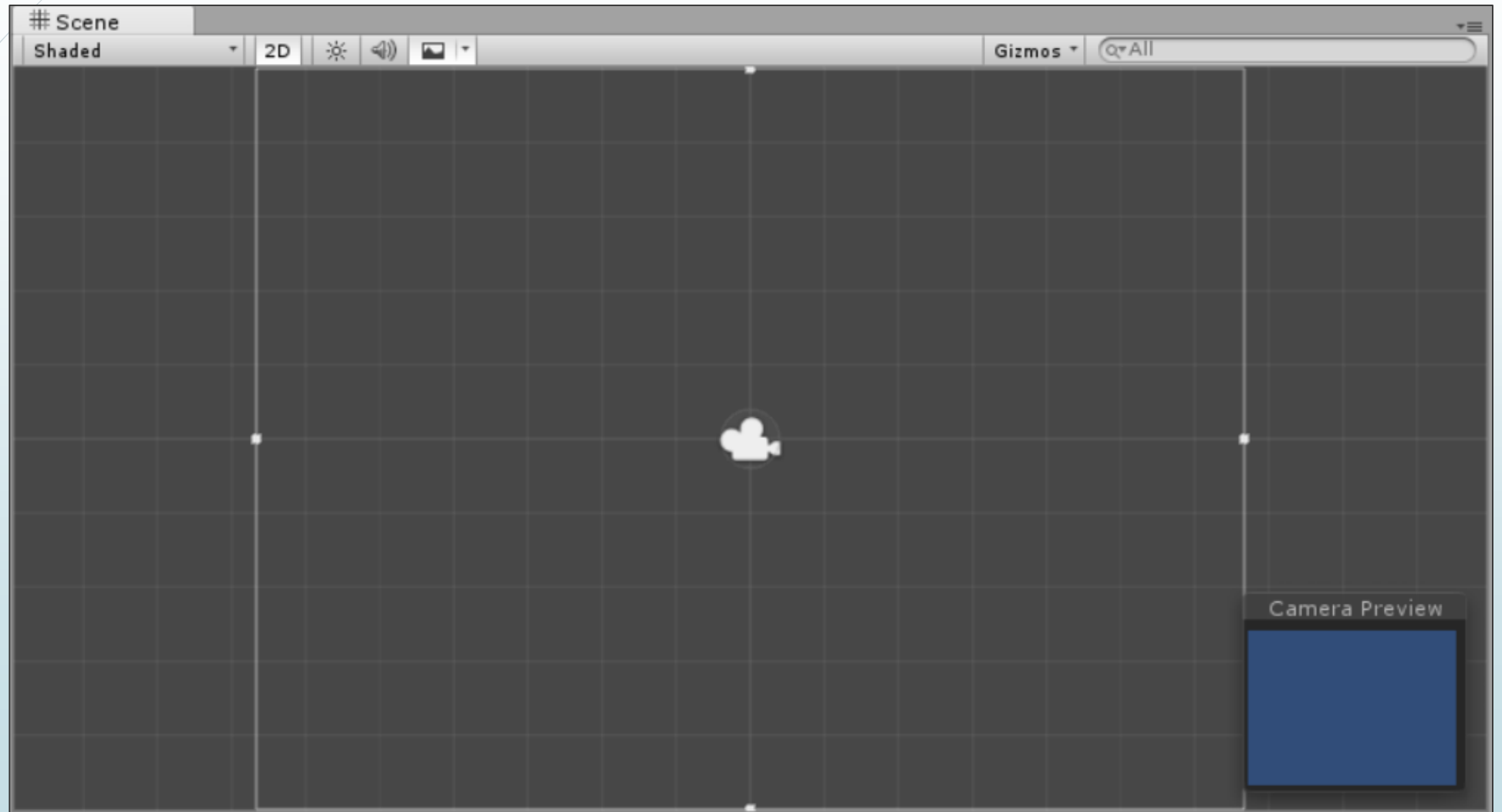




Conhecendo a Engine: Aba Game

- Nesta aba é onde você irá visualizar como o jogo estaria rodando caso estivesse pronto. Ao clicar na opção “*Play*” o jogo iria começar e ser executado nesta aba.

Conhecendo a Engine: Aba Scene

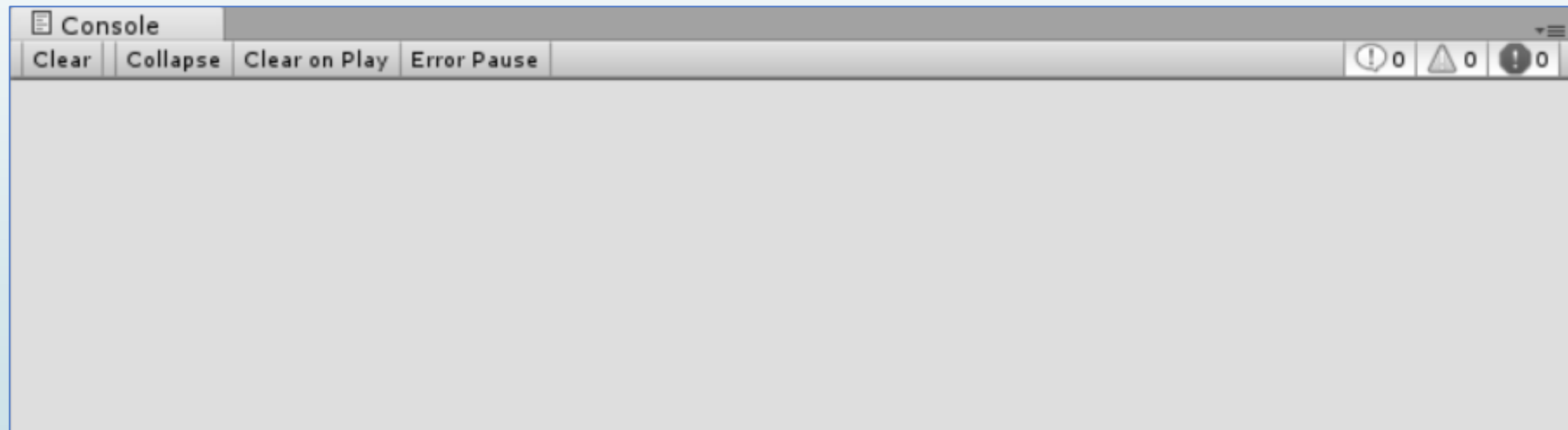




Conhecendo a Engine: Aba Scene

- Esta tela é bem parecida com a aba de Game. Nela você poderá visualizar e editar o comportamento dos objetos no jogo. Por isto que recomendo que você tenha esta aba e aba Game de forma que possa observar as duas. Nesta aba você poderia aumentar o tamanho de um personagem, aproximar a câmera e ver como iria influencia na aba Game.

Conhecendo a Engine: Aba Console





Conhecendo a Engine: Aba Console

- Na Aba console é onde nós vamos ter todas as informações do que esta acontecendo no jogo em forma de texto, os erros que ocorreram e que ocorrem.

Conhecendo a Engine: Aba Hierarchy





Conhecendo a Engine: Aba Hierarchy

- Na aba *Hierarchy* será possível visualizar todos os objetos que estão ativos no jogo. No momento você irá apenas ver o objeto “Main Camera”, responsável pela visão do jogador.



Configurando o Projeto



Configurando o Projeto

Demonstração





Configurando o Aspect da tela do Game



Configurando o Aspect do Game

Demonstração



Importando e Configurando Assets de Imagens



Importando e Configurando Assets de Imagens

Demonstração



Criando Sprites



Criando Sprites

Demonstração



Adicionando Textura aos Sprites criados




Adicionando Textura aos Sprites criados

Demonstração



**Criando, configurando e
vinculando Materiais**



Criando, configurando e
vinculando Materiais aos Sprites

Demonstração



**Criando, configurando uma
imagem de fundo no jogo**



Criando, configurando
uma imagem de fundo no jogo

Demonstração



**Criando, configurando
o asfalto**



Criando, configurando
o Asfalto

Demonstração



Criando o *Player*



Criando o *Player*

Demonstração



Configurando o *Player*

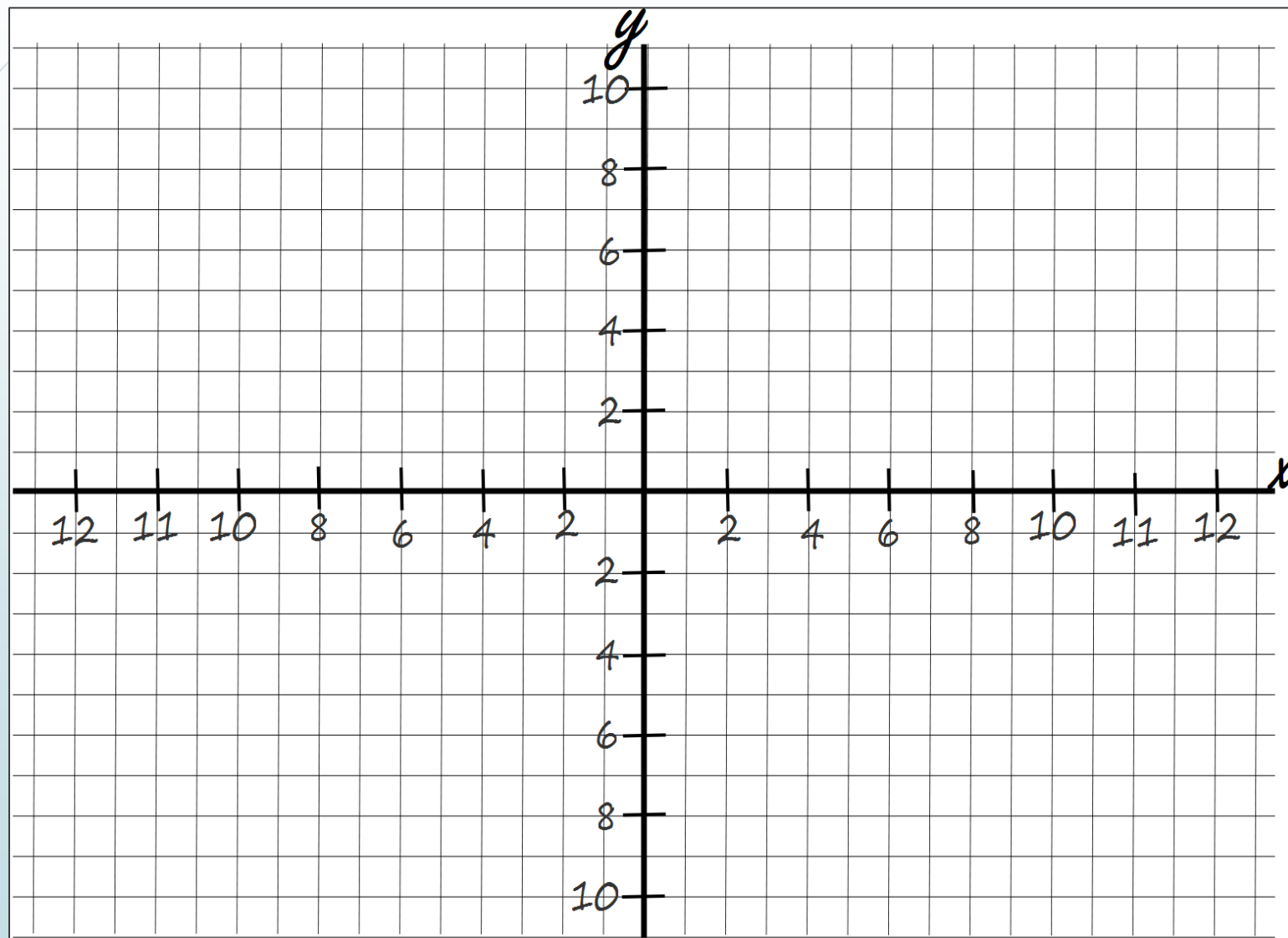
Demonstração



Criando e vinculando a tag “*Player*”

Demonstração

Posicionamento do Player



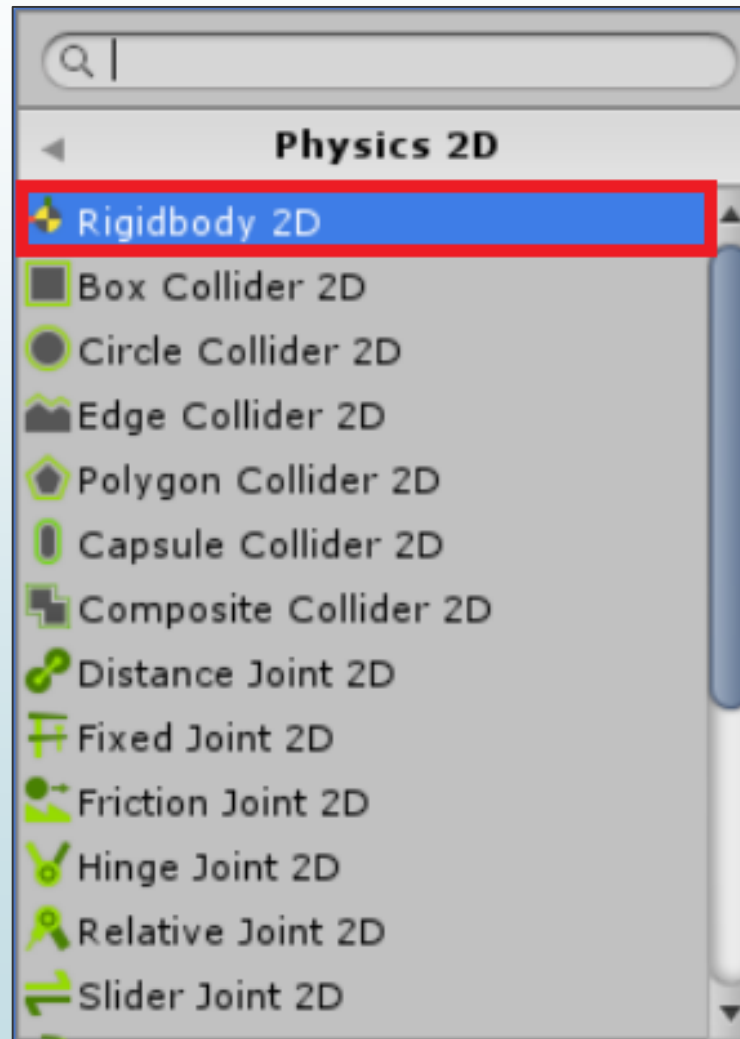


Simulando física no Unity

Simulando física no Unity: Rigidbody

- Rigidbody é um componente que, quando adicionado a um objeto, habilita forças físicas a atuarem sobre ele.
- Por exemplo: Você cria um sprite no meio da cena. Ele vai ficar flutuando lá, sem mover-se, pois, a gravidade, que é uma força física, não atua sobre ele. Mas, ao adicionar a propriedade Rigidbody para este sprite, a gravidade começa a atuar sobre o objeto e o sprite cai.
- Para adicionar um Rigidbody, selecione o objeto e clique em *Add Component* → *Physics 2D* → *Rigidbody 2D* (ou apenas pesquise por *Rigidbody 2D* no campo de busca).

Simulando física no Unity: Rigidbody

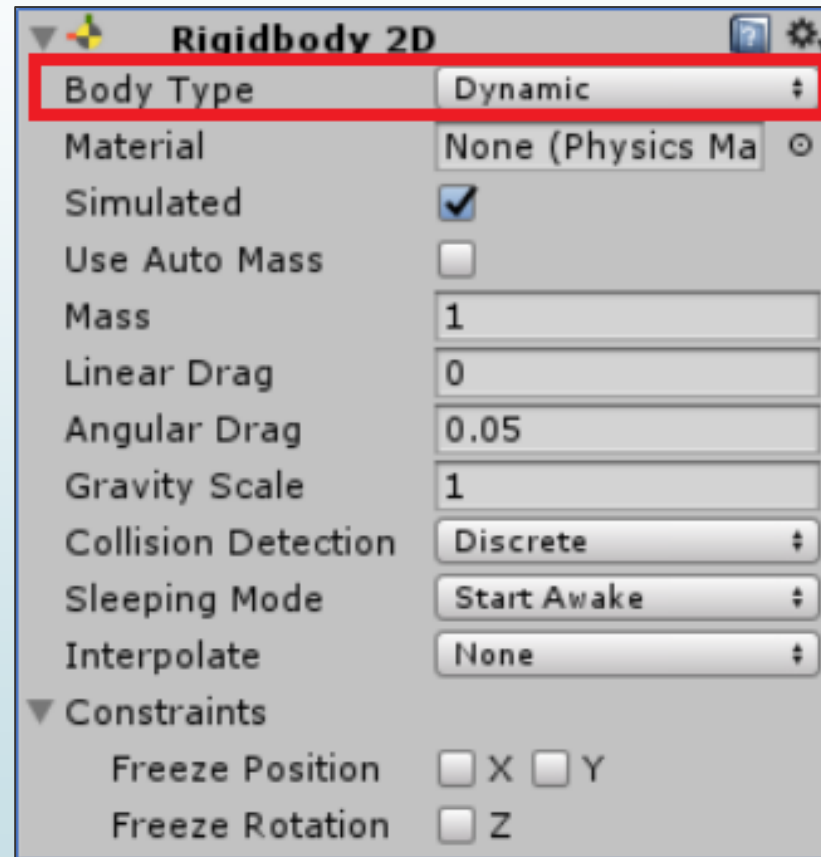




Simulando física no Unity: Rigidbody

- Note que temos várias propriedades no Rigidbody, e estas dependem do “*Body Type*” selecionado. Para este projeto iremos alterar apenas esta propriedade:

Simulando física no Unity: Rigidbody





Simulando física no Unity: Rigidbody

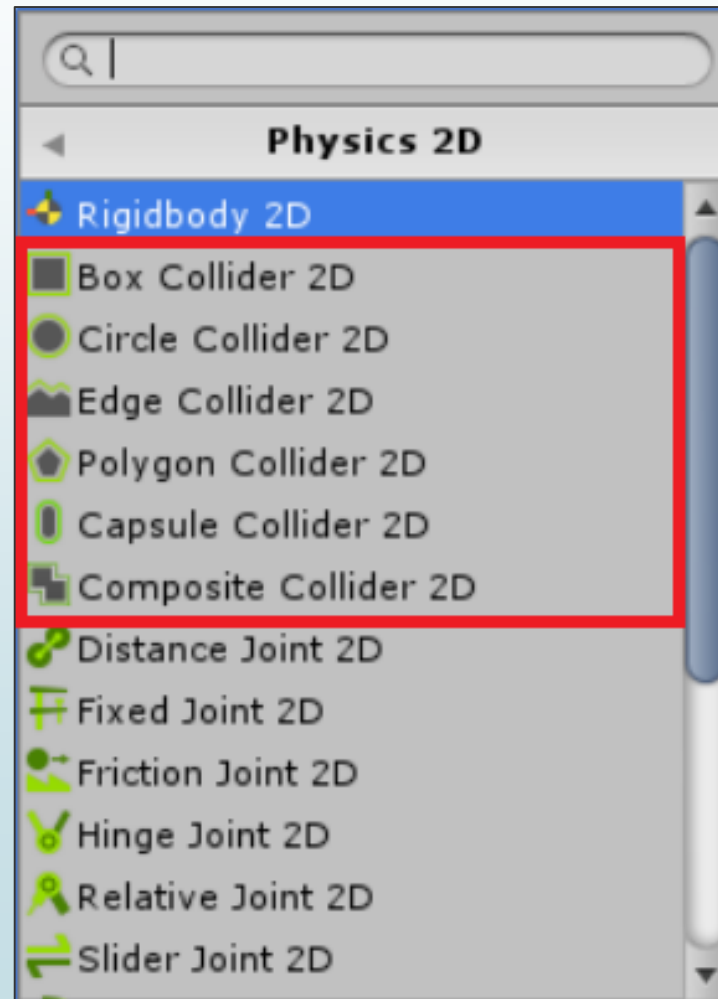
Demonstração



Simulando física no Unity: Colliders

- Um collider 2D, ou colisor 2D, define a forma de um objeto para colisões físicas. Você pode adicionar quantos colliders 2D forem necessários para deixar a colisão o mais próximo possível do modelo do objeto.
- Para adicionar um collider 2D, selecione um objeto e clique em *Add Component* → *Physics 2D* → *Collider 2D*. Note que você verá uma lista com vários *colliders 2D* diferentes, conforme a imagem abaixo:

Simulando física no Unity: Colliders




Simulando física no Unity: Colliders

- Cada colisor possui as suas próprias propriedades. Geralmente elas definem o tamanho e a forma do colisor. Porém, existem algumas propriedades que são comuns entre os colisores. São elas:
 - **Is Trigger:** quando essa propriedade está marcada, os objetos que passam pelo colisor não são afetados por ele. Mas é possível saber quando o objeto passou. Por exemplo: Em um jogo de corrida, você quer saber quando o carro passa pela linha de chegada. Colocamos um collider 2D nela e marcamos a opção Is Trigger para que o carro não pare quando colidir com ele;
 - **Material:** atribuímos um material para um colisor quando queremos simular diferentes superfícies.



Simulando física no Unity: Colliders

Demonstração



Controlando movimento do Player na cena



Script Player.cs: Criação

Demonstração



Script Player.cs: Vinculação

Demonstração

Script Player.cs: mover *Player* (carro)

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Player : MonoBehaviour
6  {
7      public Vector2 esquerda = new Vector2(-5f, 0);
8      public Vector2 direita = new Vector2(5f, 0);
9
10     private Rigidbody2D rb;
11
12     void Start()
13     {
14         rb = GetComponent<Rigidbody2D>();
15     }
16
17     void Update () {
18         //Move para a Esquerda
19         if (Input.GetKey(KeyCode.LeftArrow))
20             rb.AddForce(esquerda);
21         //Move para a Direita
22         else if (Input.GetKey(KeyCode.RightArrow))
23             rb.AddForce(direita);
24         //Parar movimento
25         else if (Input.GetKeyUp(KeyCode.LeftArrow) || Input.GetKeyUp(KeyCode.RightArrow))
26             rb.velocity = Vector2.zero;
27     }
28 }
```

Acessar a propriedade
Rigidbody2D do objeto.

Controla o movimento
horizontal do objeto



Criando o Efeito de Movimento do cenário



Script Movimento.cs: Criação

Demonstração



Script Movimento.cs: Vinculação

Demonstração

Script Movimento.cs: Mover Cenário

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Movimento : MonoBehaviour {
6
7      private Material materialAtual;
8      public float velocidade;
9      private float deslocamento;
10
11      // Use this for initialization
12      void Start()
13      {
14          materialAtual = GetComponent<Renderer>().material;
15      }
16
17      // Update is called once per frame
18      void Update()
19      {
20          deslocamento += 0.001f;
21          materialAtual.SetTextureOffset("_MainTex", new Vector2(0, deslocamento * velocidade));
22      }
23  }
```

Acessar a propriedade Material do Renderer do objeto.

Controla o movimento Vertical do objeto



Criando os Prefabs de Moedas Alvo



Criando a Moeda

Demonstração



Configurando a Moeda

Demonstração



Script Moeda.cs: Criação

Demonstração



Script Moeda.cs: Vinculação

Demonstração



Criando um Prefab da Moeda

Demonstração

Script Moeda.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Moeda : MonoBehaviour {
6      public Vector3 velocidade = new Vector3(0, -5);
7
8      void Start () {
9          transform.position = new Vector3(Random.Range(-2.6f, 2.6f), transform.position.y, transform.position.z);
10     }
11
12     // Update is called once per frame
13     void Update () {
14         transform.position += velocidade * Time.deltaTime;
15     }
16 }
```



Criando um Gerador de Moedas



Criando o Gerador

Demonstração



Script Gerador.cs: Criação

Demonstração



Script Gerador.cs: Vinculação

Demonstração

Script Gerador.cs

```
1  using System.Collections;
2      using System.Collections.Generic;
3      using UnityEngine;
4
5  public class Gerador : MonoBehaviour {
6
7      public GameObject prefabMoeda;
8      void Start () {
9          InvokeRepeating("CriarMoeda", 1f, 1.5f);
10     }
11
12     void CriarMoeda()
13     {
14         Instantiate(prefabMoeda);
15     }
16 }
```



Detectando Colisões



Script Player.cs

```
void OnCollisionEnter2D(Collision2D colisor)
{
    if (colisor.gameObject.tag == "Moeda")
    {
        Dinheiro++;
        Destroy(colisor.gameObject);
    }
}
```



Criando Pontuação

Script Player.cs

```
void OnGUI()  
{  
    GUI.color = Color.black;  
    GUILayout.Label("Dinhieor $$: " + Dinheiro.ToString());  
}
```

Precisa ser declarado
e inicializado.



Testando o Game

