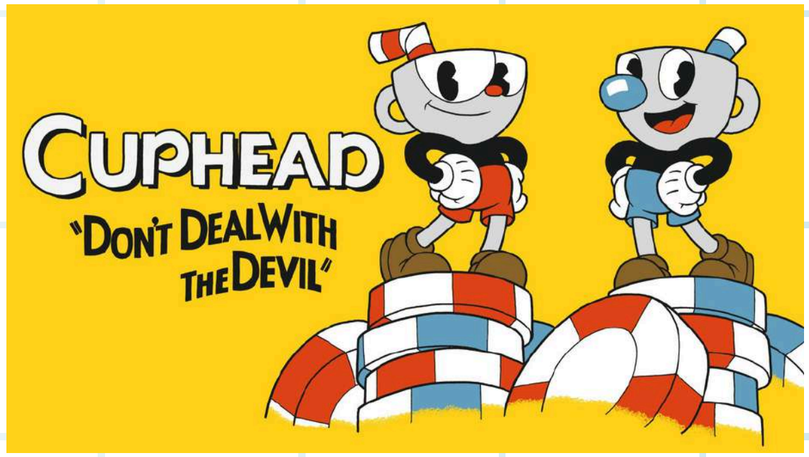
CURSO DE DESENVOLVIMENTO DE JOGOS

CEFET MG - LEOPOLDINA

Instrutores: Guilherme
Thiago

Unity

- **É uma plataforma de desenvolvimento de jogos em tempo real que permite a criação de aplicações 2D e 3D, VR, como jogos, simulações e Apps interativos.**



O que será visto:

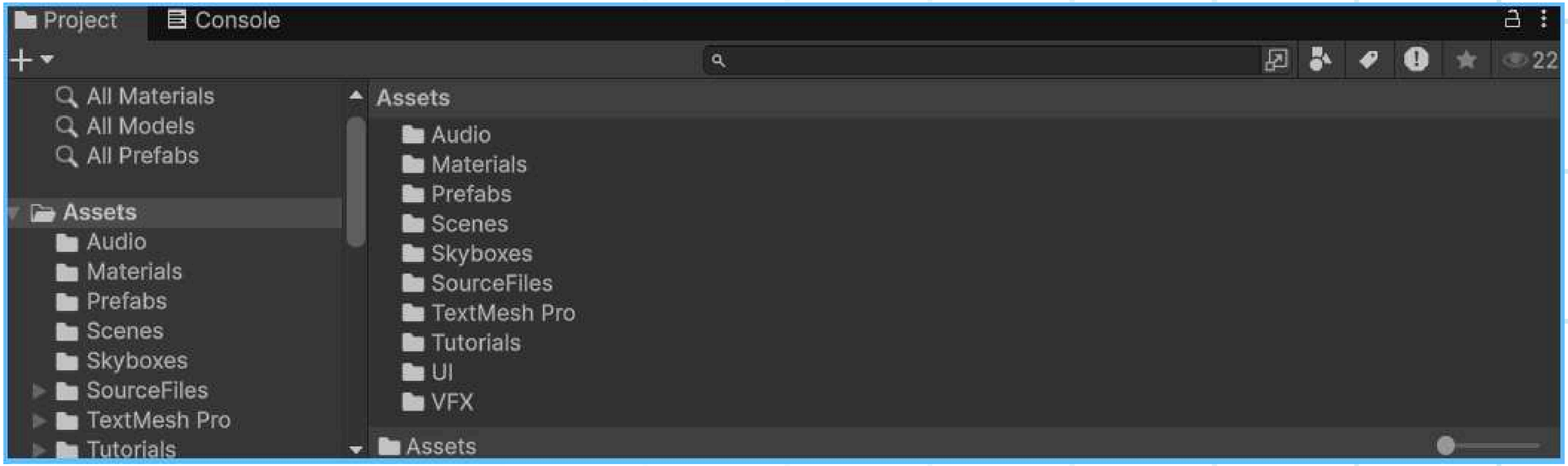
- Interface do Unity (Scene, Game, Project, Inspector).
- Introdução a classes e objetos (MonoBehaviour).
- GameObjects e Components.
- Sprites
- Sistema de Coordenadas
- Prefabs

Interface Unity

- **Projeto:**

Ao criar um projeto na Unity, uma coleção de imagens, modelos, scripts e outros assets que serão usados na criação do Projeto.

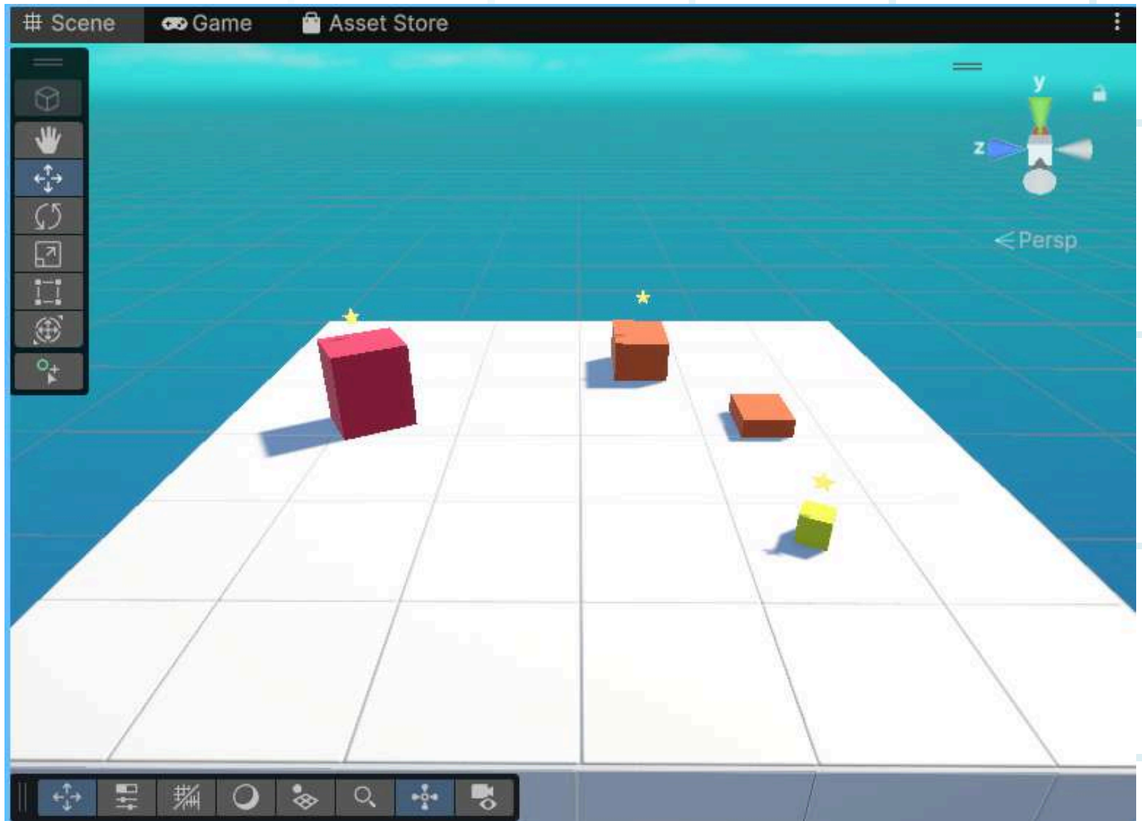
A janela “Project” permite que organizemos esses arquivos



Interface Unity

Scene View:

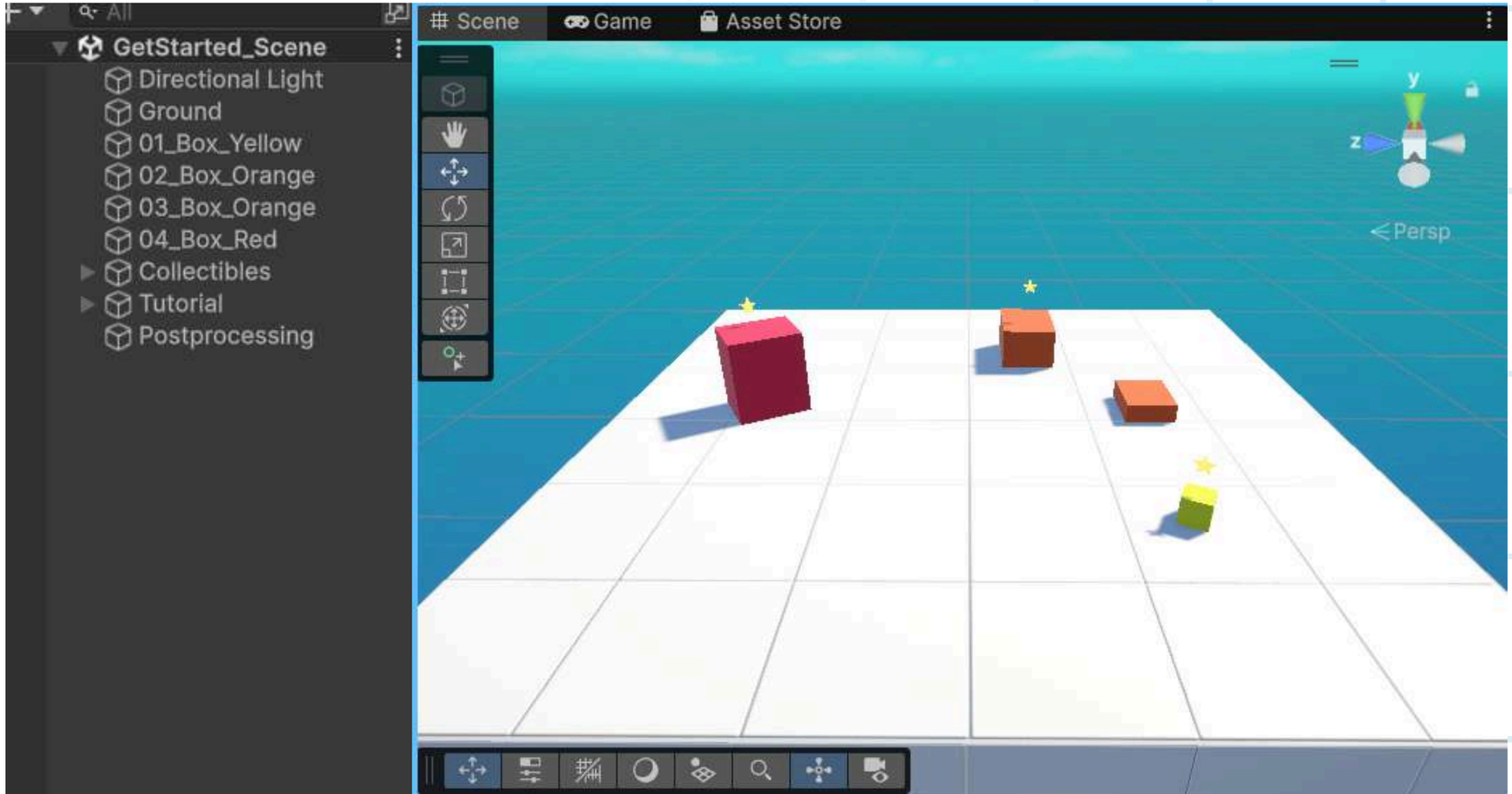
A janela “Scene View” é onde manipularemos os objetos. Aqui podemos visualizar nossos cenários 2D e 3D



Interface Unity

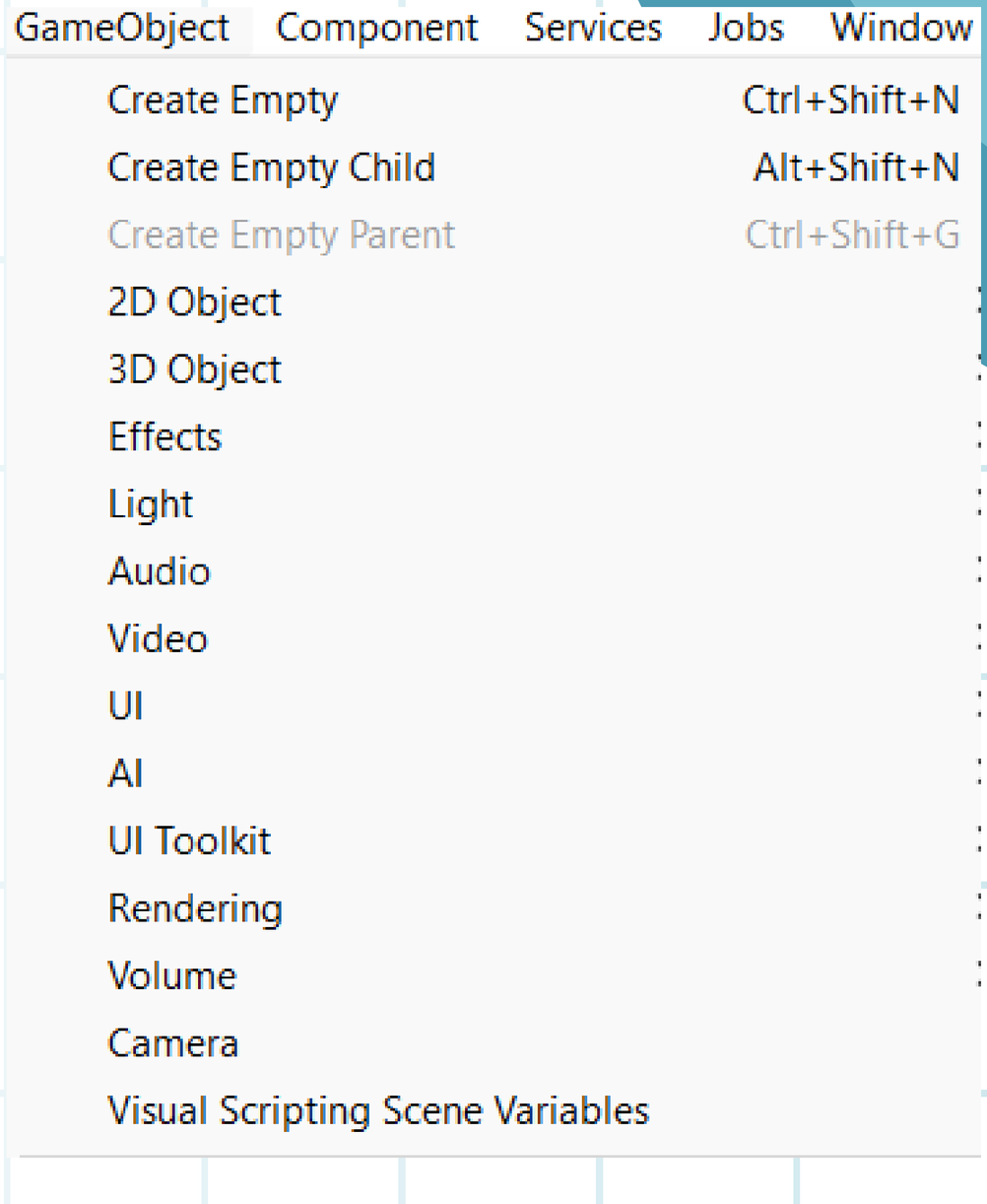
Hierarchy:

A janela de Hierarquia nos mostra todos os GameObjects presentes em nossa Scene. Conseguimos organizar todos os objetos a partir dessa janela



Game Object:

- Tudo que está dentro da cena do Unity (ex: um cubo, um personagem, a camêra, a luz, etc.)
- Um GameObject sozinho não faz nada, é necessário a utilizaçã de Componentes para dar-lhe funcionalidades.



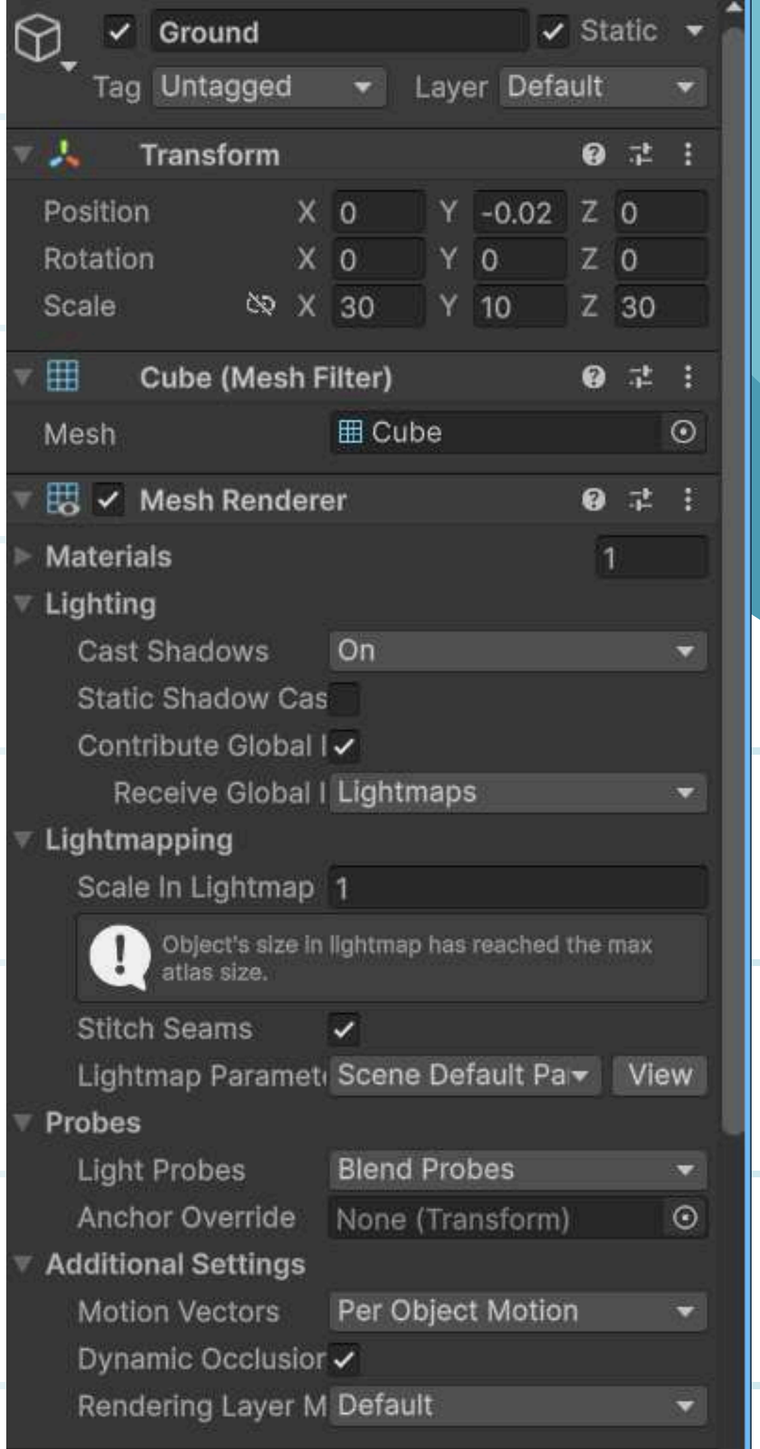
Hierarquia (pais e filhos):

- Um GameObject pode conter outros dentro dele.
- EX: Jogador (pai) contem - *modelo 3D (filho)* - *Arma (filho)* - *Câmera (filho)*
- Isso ajuda na organização (caso mova o JOGADOR (pai), todo o resto acompanha-o) e utilização de recursos.

Interface Unity

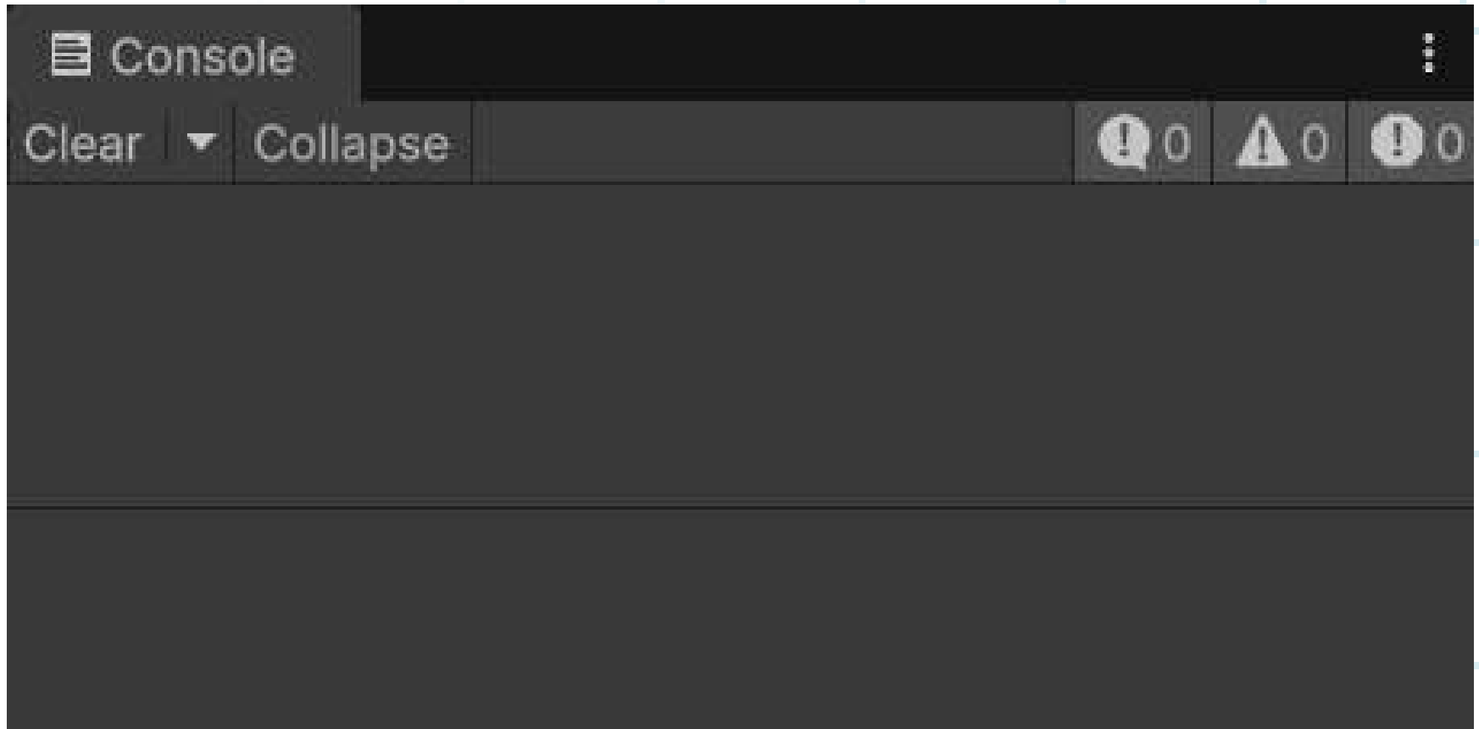
Inspector:
Na janela Inspector podemos visualizar e editar as propriedades de praticamente tudo em nossa Scene.

No exemplo ao ao lado estão as propriedades do Chão



Interface Unity

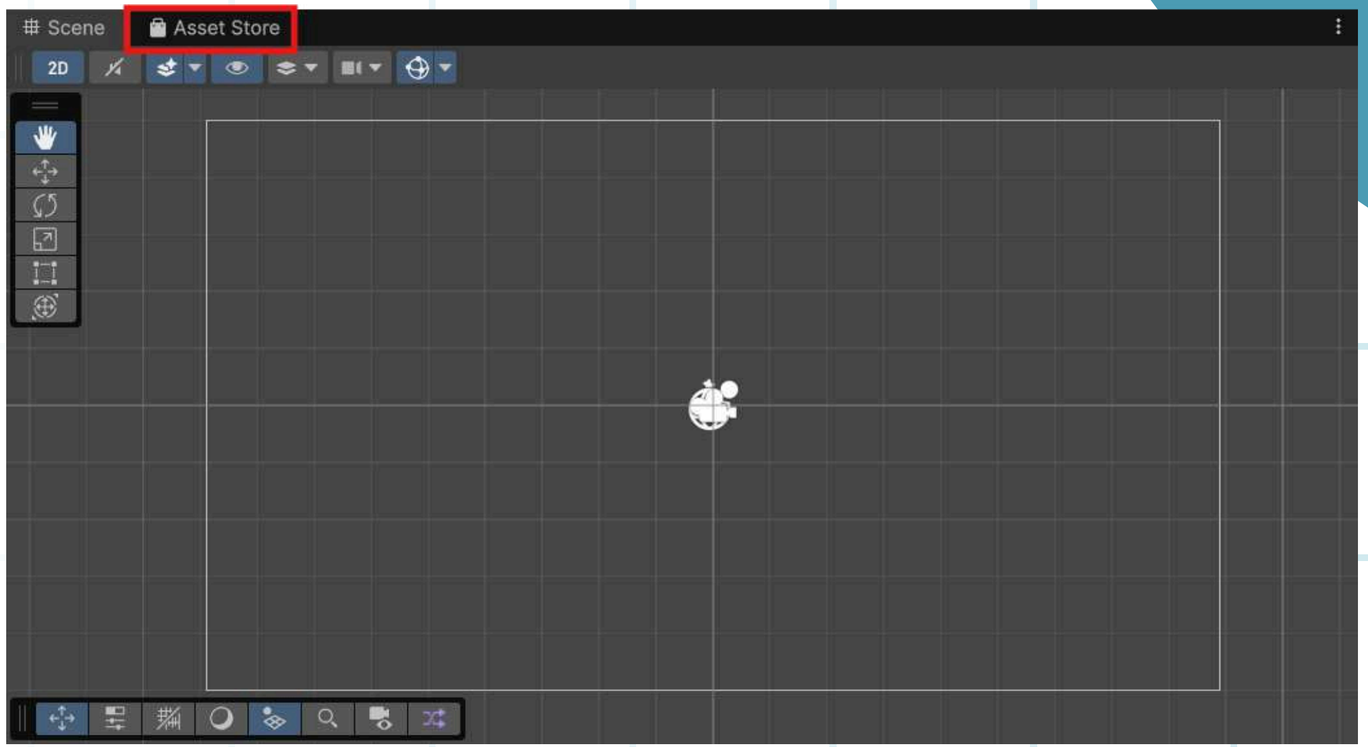
Console: A janela de console nos mostra mensagens de erro, avisos e informações do projeto. Ela será util para encontrarmos problemas em nosso projeto.



Importando Assets

- Podemos importar assets da Assets Store para o nosso projeto

- Estaremos utilizando o “Pixel Adventure 1”

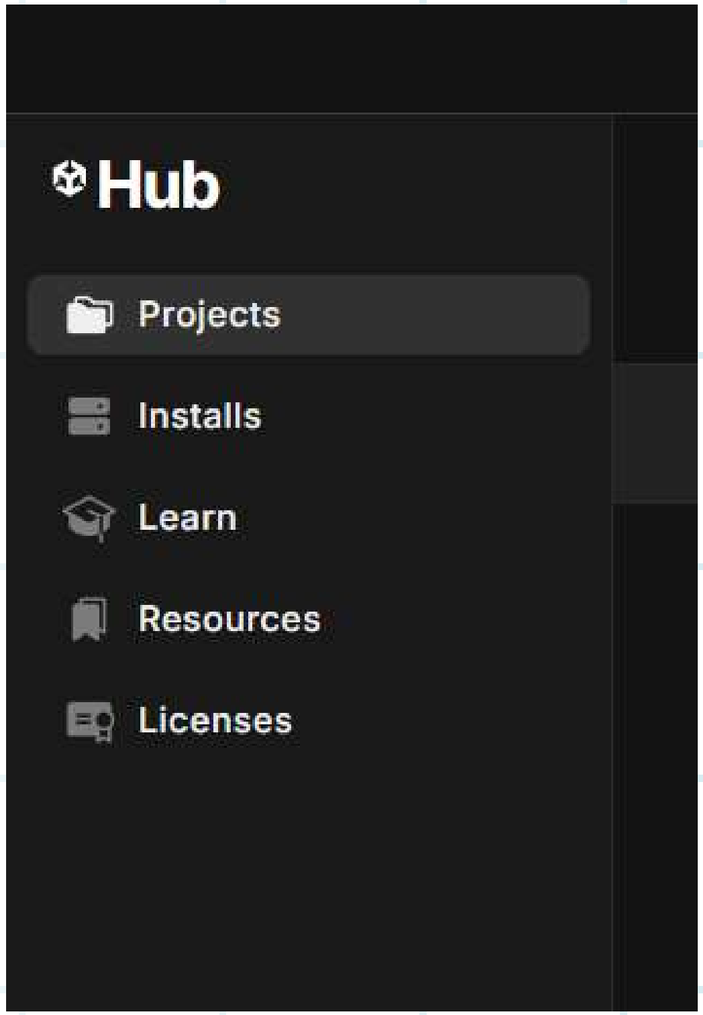


Prefabs:

- São moldes reutilizáveis.
- Crie um objeto (ex: um inimigo genérico)
- Transforma em um Prefab.
- Agora ele pode ser utilizado quantas vezes quiser sem ter que criar do zero.

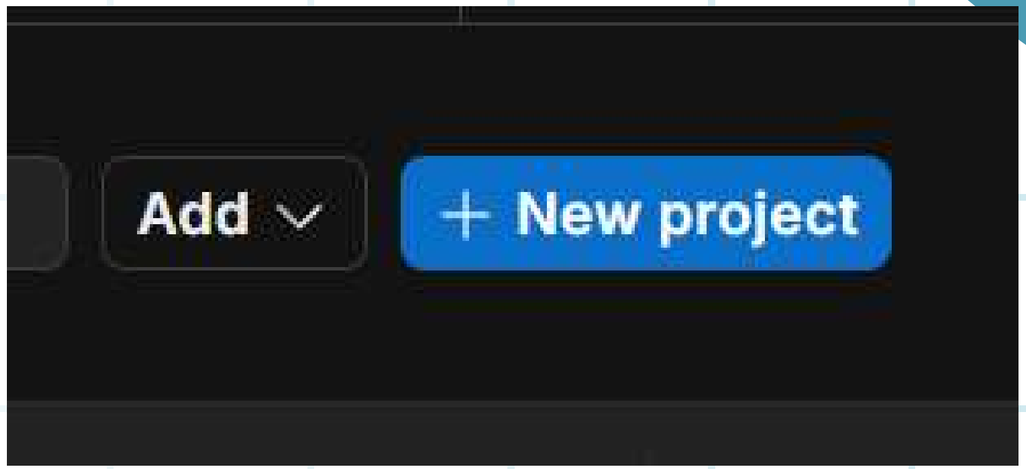
Criando projeto no Unity:

- Abra o App.
- Selecione Projetos (lado esquerdo).
- Novo Projeto.
- Escolha o tipo de projeto (2D ou 3D) e o nomeie (pode levar alguns segundos na primeira vez).
-



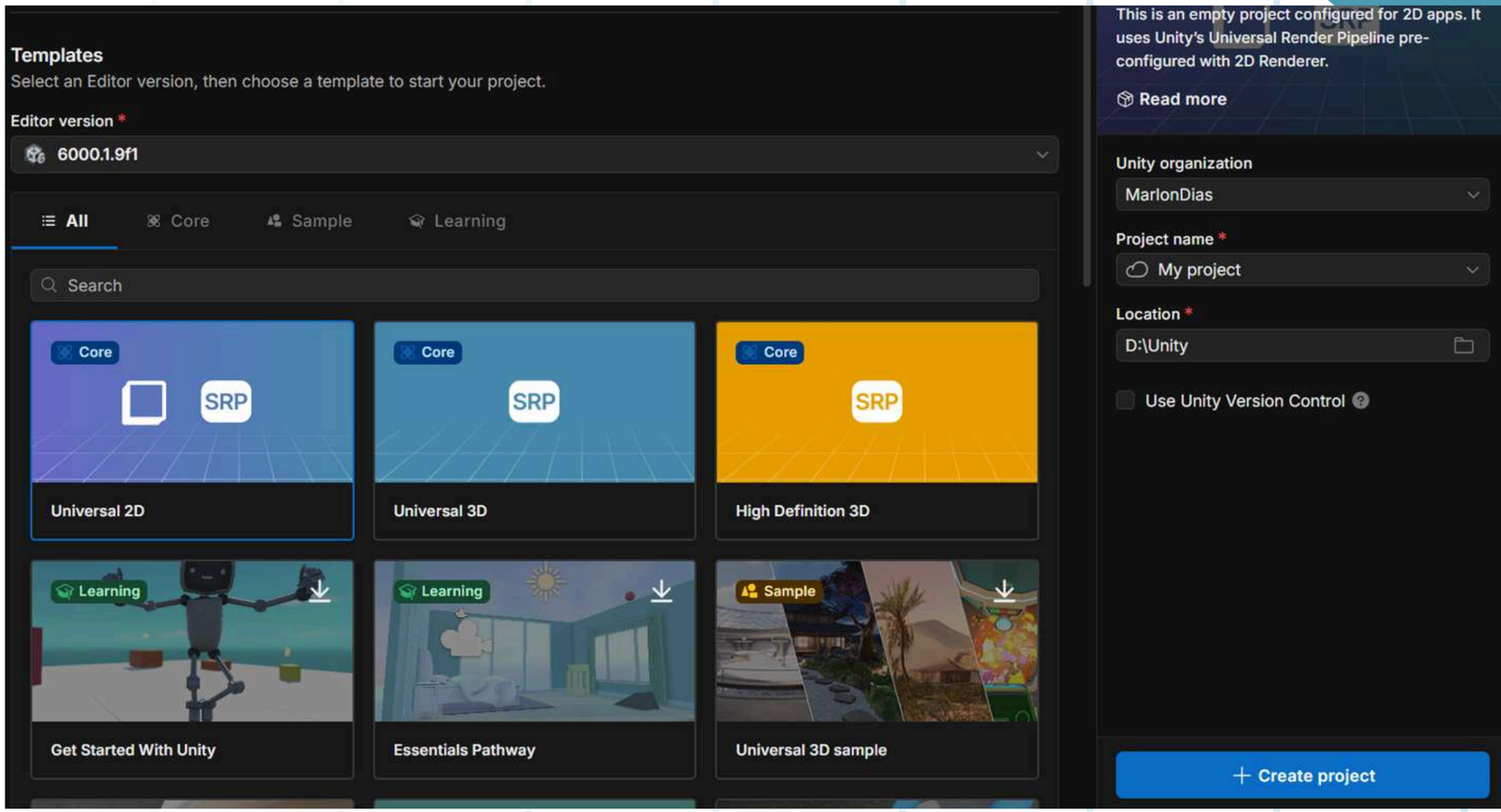
Criando projeto no Unity:

- Abra o App.
- Selecione Projetos (lado esquerdo).
- Novo Projeto (canto superior direito).
- Escolha o tipo de projeto (2D ou 3D) e o nomeie (pode levar alguns segundos na primeira vez).



Criando projeto no Unity:

- Escolha onde salvar o arquivo.
- Nomeie o arquivo.
- Por fim, clique em Criar Projeto.



Fisica e Interações

Mesmo criando um cenário, não há interações entre qualquer personagem e o fundo. Portanto é preciso adcionar as colisões

O Unity possui um sistema de física , que permite simular movimentos, colisões e interações entre objetos no jogo.

Fisica e Interações

Rigidbody:

- **Dá “vida” ao objeto, permitindo que ele seja afetado por forças (gravidade, empurrões, quedas).**

Sem Rigidbody, o objeto não reage fisicamente.

Collider:

- **Define a área de colisão do objeto (ex.: Box Collider, Sphere Collider, Mesh Collider).**

Podem ser triggers, que detectam entrada/saída sem colisão física.

Forças e Movimento:

- **Podemos aplicar forças, torque (rotação) ou mover objetos por script.**

Ex.: empurrar uma bola ou simular

Fisica e Interações

Rigidbody:

- **Dá “vida” ao objeto, permitindo que ele seja afetado por forças (gravidade, empurrões, quedas).**

Sem Rigidbody, o objeto não reage fisicamente.

Collider:

- **Define a área de colisão do objeto (ex.: Box Collider, Sphere Collider, Mesh Collider).**

Podem ser triggers, que detectam entrada/saída sem colisão física.

Forças e Movimento:

- **Podemos aplicar forças, torque (rotação) ou mover objetos por script.**

Ex.: empurrar uma bola ou simular

Mas como os objetos se movimetam? Como os cenários mudam?

Para isso precisamos aplicar os Scripts. Os Scripts são “roteiros” que cada objeto irá realizar

O Unity usa C# para programar a lógica do jogo e implementar os scripts

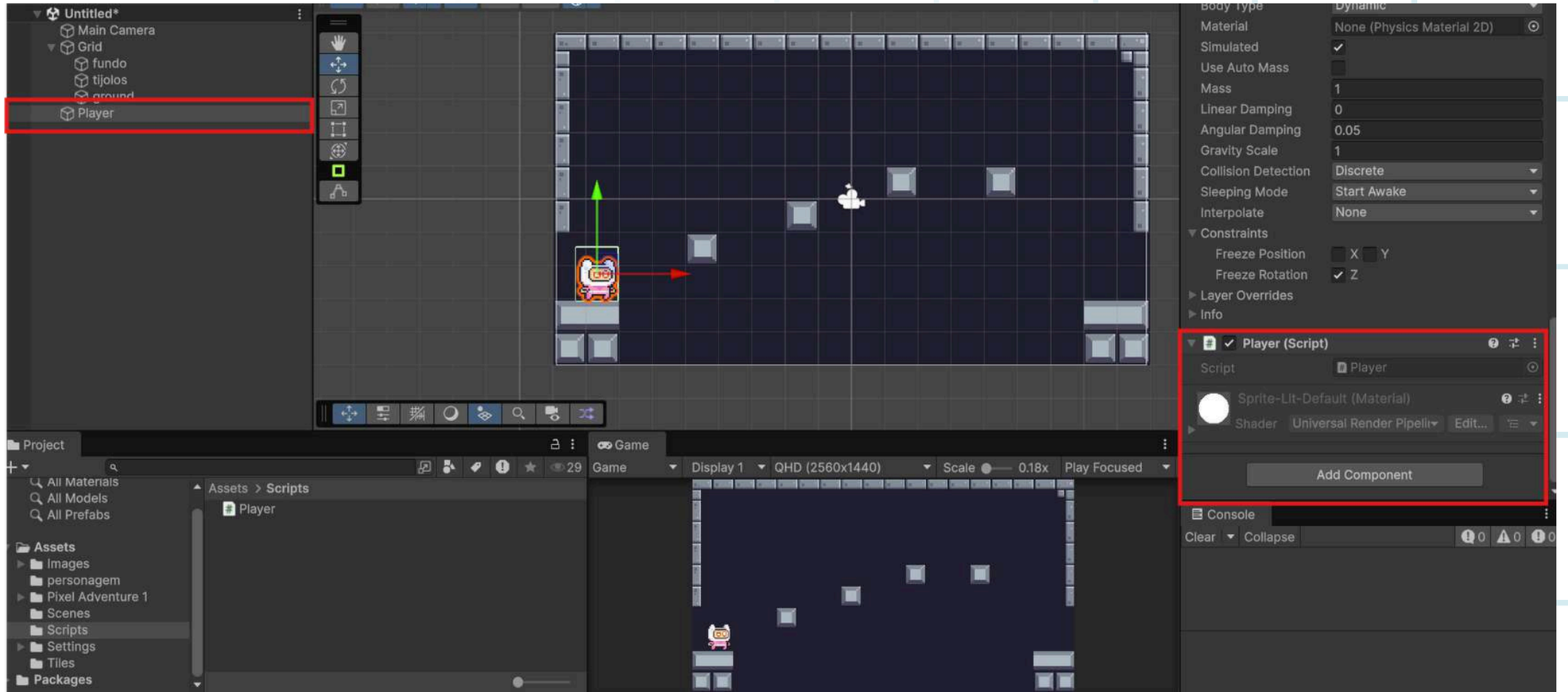
O que é C# no Unity:

- C# é a linguagem usada para dar comportamento aos objetos do jogo.
- O código ao lado mostra no Console a mensagem “Olá Unity” quando iniciar o jogo.

```
using UnityEngine;

0 references
public class HelloWorld : MonoBehaviour
{
    // Start é chamado uma vez quando o jogo começa
    0 references
    void Start()
    {
        Debug.Log("Olá Unity");
    }

    // Update é chamado s cada fram (60 vezes por segundo, por exemplo)
    0 references
    void Update()
    {
        // Aqui entra o código que deve rodar o tempo todo
    }

}
```

Criando um Script:

- Clique no objeto desejado.
- Vá até Add Component.
- Escolha New script.



Código de movimento:

- Criando o código de movimentação.
- Update() => Roda várias vezes por segundo.
- transform.position() => move o objeto.
- Time.deltaTime => deixa o movimento suave.

```
public class Player : MonoBehaviour
{

    1 reference
    public float velocidade;

    // Start is called once before the first execution of Update after the MonoBehavio
    0 references
    void Start()
    {

    }

    // Update is called once per frame
    0 references
    void Update()
    {

    }

    0 references
    void Move()
    {
        Vector3 movimento = new Vector3(Input.GetAxis("Horizontal"), 0f, 0f);
        transform.position += movimento * Time.deltaTime * velocidade;
    }
}
```

Podemos adcionar o pulo também

O pulo é afetado diretamente pelo Rigidbody. É preciso então ser possível alterar o rigidbody através da função “Jump”.

Vamos criar uma variavel “rig” do tipo Rigidbody2D, privada, além de uma variavel para alterar a intensidade do pulo

```
public class Player : MonoBehaviour
{

    1 reference
    public float velocidade;
    2 references
    private Rigidbody2D rig;
    1 reference
    public float forca_pulo;

    // Start is called once before the first execution of Update
    0 references
    void Start()
    {
        rig = GetComponent<Rigidbody2D>();
    }

    // Update is called once per frame
```

Pulo

A função Jump captura a tecla “Jump” (No unity é o espaço), onde é adicionada uma força no eixo y, do tipo impulso.

Após isso podemos chamar a função no Update().

```
    void Update()
    {
        Move();
        Jump();
    }

    1 reference
    void Move()
    {
        Vector3 movimento = new Vector3(Input.GetAxis("Horizontal"), 0f, 0f);
        transform.position += movimento * Time.deltaTime * velocidade;
    }

    1 reference
    void Jump()
    {
        if (Input.GetButtonDown("Jump"))
        {
            rig.AddForce(new Vector2(0f,forca_pulo), ForceMode2D.Impulse);
        }

    }
}
```

Correção do pulo e pulo duplo

O pulo já está funcionando, porém se apertarmos Espaço várias vezes o personagem começará a voar. Para corrigir isso

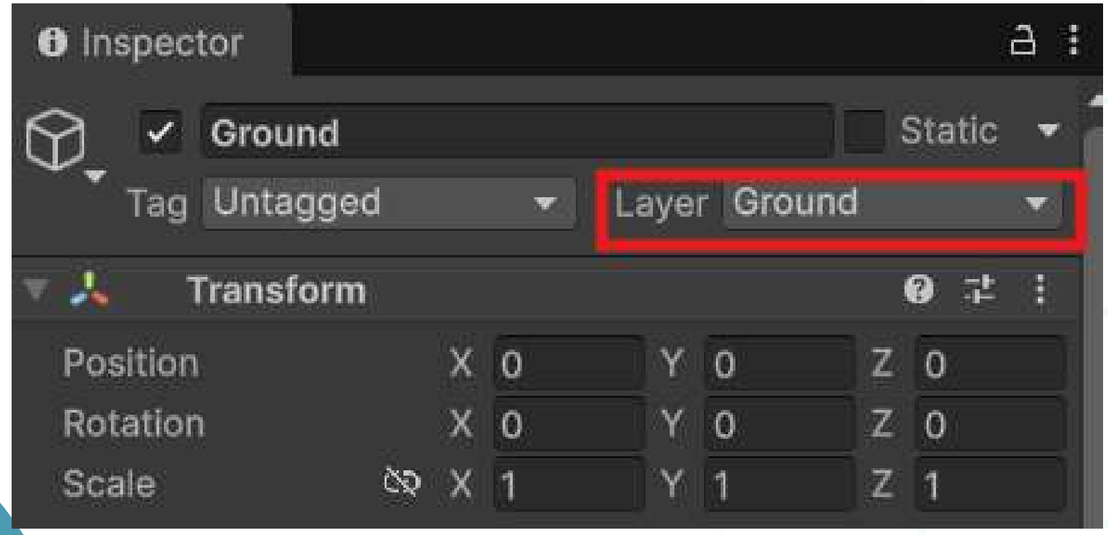
Vamos criar duas variáveis, uma para verificar se o personagem esta pulando e outra para o pulo duplo

```
public class Player : MonoBehaviour
{

    1 reference
    public float velocidade;
    2 references
    private Rigidbody2D rig;
    1 reference
    public float forca_pulo;

    3 references
    public bool isJumping;

    0 references
    public bool doubleJump;
```

Correção do pulo e pulo duplo

Após isso, criemos duas funções que irão verificar se o personagem está pulando ou não. Não só isso mas criemos uma camada chamada Ground na layer 8, e vamos atribuí-la ao nosso chão.

Na função Jump haverá outra condição agora




```
void Jump()
{
    if (Input.GetButtonDown("Jump") && isJumping == false)
    {
        rig.AddForce(new Vector2(0f, forca_pulo), ForceMode2D.Impulse);
    }

}

0 references
void OnCollisionEnter2D(Collision2D collision)
{
    if (collision.gameObject.layer == 8)
    {
        isJumping = false;
    }
}
0 references
void OnCollisionExit2D(Collision2D collision)
{
    if (collision.gameObject.layer == 8)
    {
        isJumping = true;
    }
}
```

Correção do pulo e pulo duplo

Se quisermos adicionar um pulo duplo, basta modificar a condição do pulo e atribuir true quando ele estiver no ar e false no chão

```
void Jump()
{
    if (Input.GetButtonDown("Jump"))
    {
        if (isJumping == false)
        {
            rig.AddForce(new Vector2(0f, forca_pulo), ForceMode2D.Impulse);
            doubleJump = true;
        }
        else
        {
            if (doubleJump == true)
            {
                rig.AddForce(new Vector2(0f, forca_pulo), ForceMode2D.Impulse);
                doubleJump = false;
            }
        }
    }
```