Conceitos fundamentais de física e colisão no Unity

Prof. Me. Hélio Lourenço Esperidião Ferreira

Conceitos sobre física

Física em jogos se refere a uma simulação controlada pela própria engine.

No Unity, existem duas engines de física disponíveis, uma 2D, baseada na Box2D, e outra em 3D, chamada PhysX.

Collider

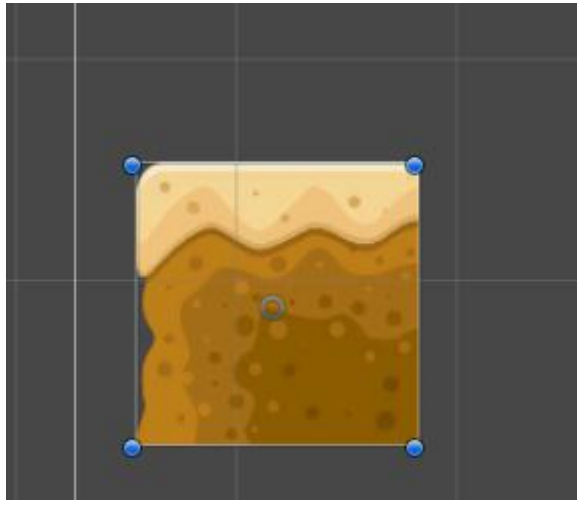
Propriedades que marcam o volume físico de um objeto, assim como o material físico (com atrito e elasticidade) que define parte de seu comportamento.

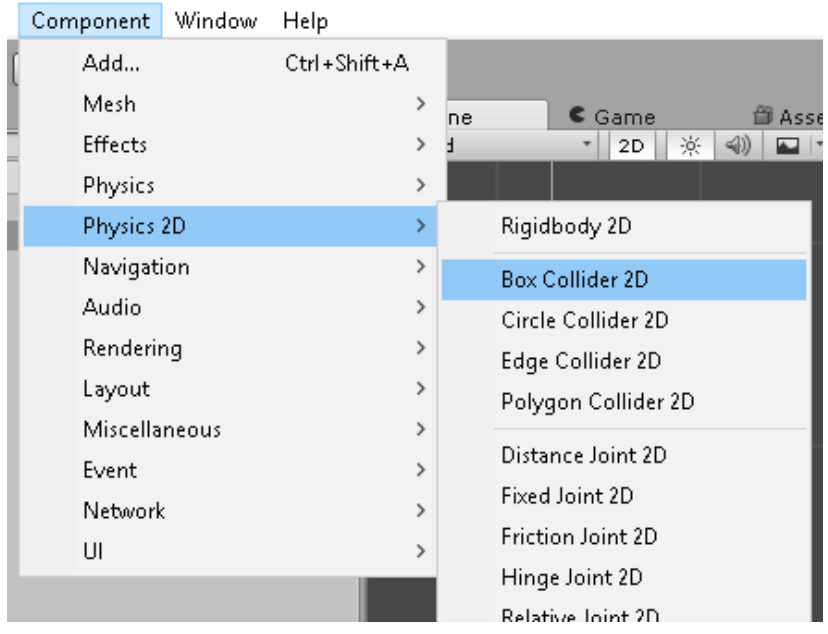
Existem vários tipos de colisores, cada um com formato diferente (por exemplo, BoxCollider, SphereCollider).

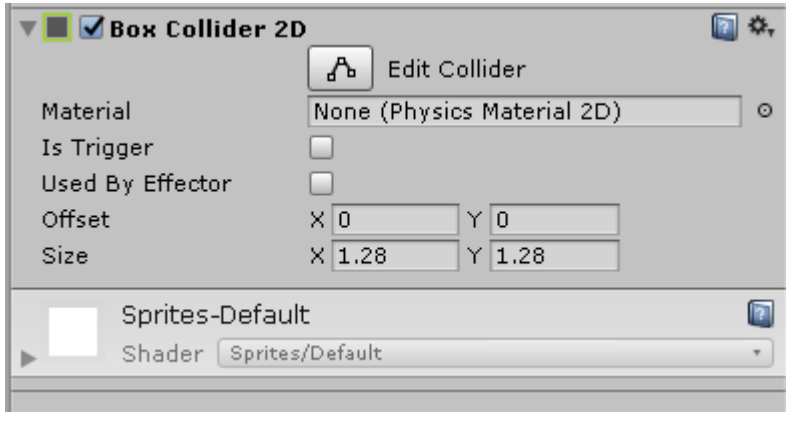
Também mantem informações sobre o tipo de interação que tem com outros objetos (colisão ou sobreposição). Esse componente tem métodos e eventos que utilizamos para criar lógica para interações físicas, como OnCollisionEnter e OnCollisionExit. Para que um script possa utilizar essas funções-evento, o objeto a que está atrelado tem que ter um Collider.

Colisão de elementos

- Acrescente o chão do cenário.
- Clique sobre o componente.
- Vá ao menu Component >> Physic2D>>Box Collider 2D
- Faça para todos os elementos onde pode ocorrer colisão.





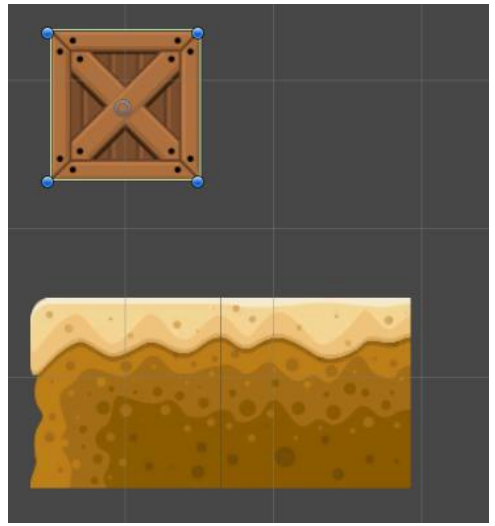


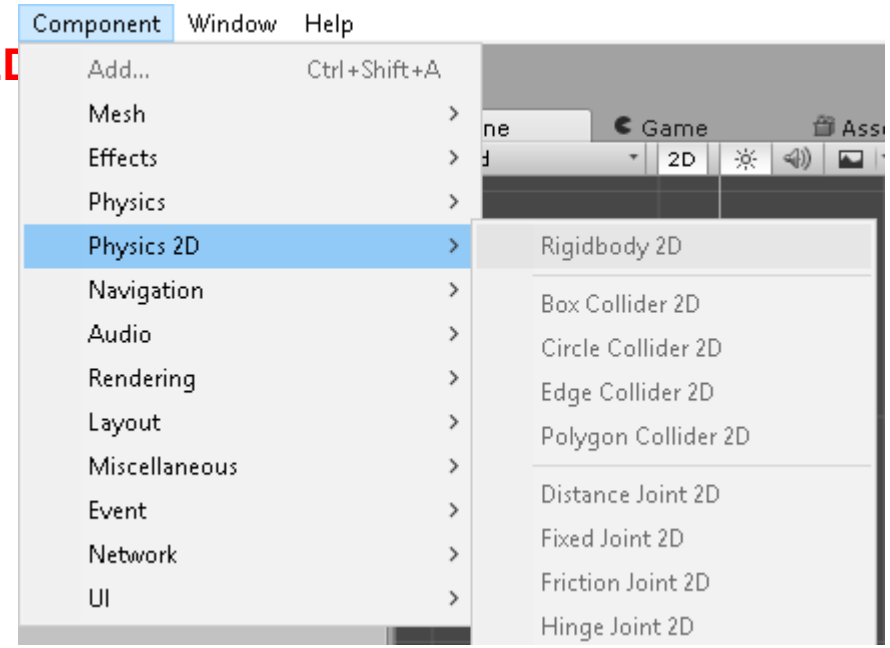
Colisão de elementos

- Quando é adicionada a caixa de colisão a um elemento de cena, a caixa de colisão torna-se uma propriedade do elemento.
- Além das propriedades de **posicionamento** e **cor** agora o elemento possui propriedades de colisão.
- É possível configurar o tamanho da caixa de colisão (**hitbox**) clicando em edit Collider como pode ser viso na figura ao lado.

RigidBody 2D

- Acrescento o caixote
- Defina o como caixa de colisão 2D
  - Vá ao menu Component >> Physic2D>>Box Collider 2D
- Acrescente Física ao de corpo Rígido
  - Vá ao menu menu **Component** >> **Physic2D**>>**RigidBody 2D**
- Como a caixa é um corpo rígido ao rodar o jogo ela vai cair se se colidir com o chão que por sinal é uma caixa de colisão.



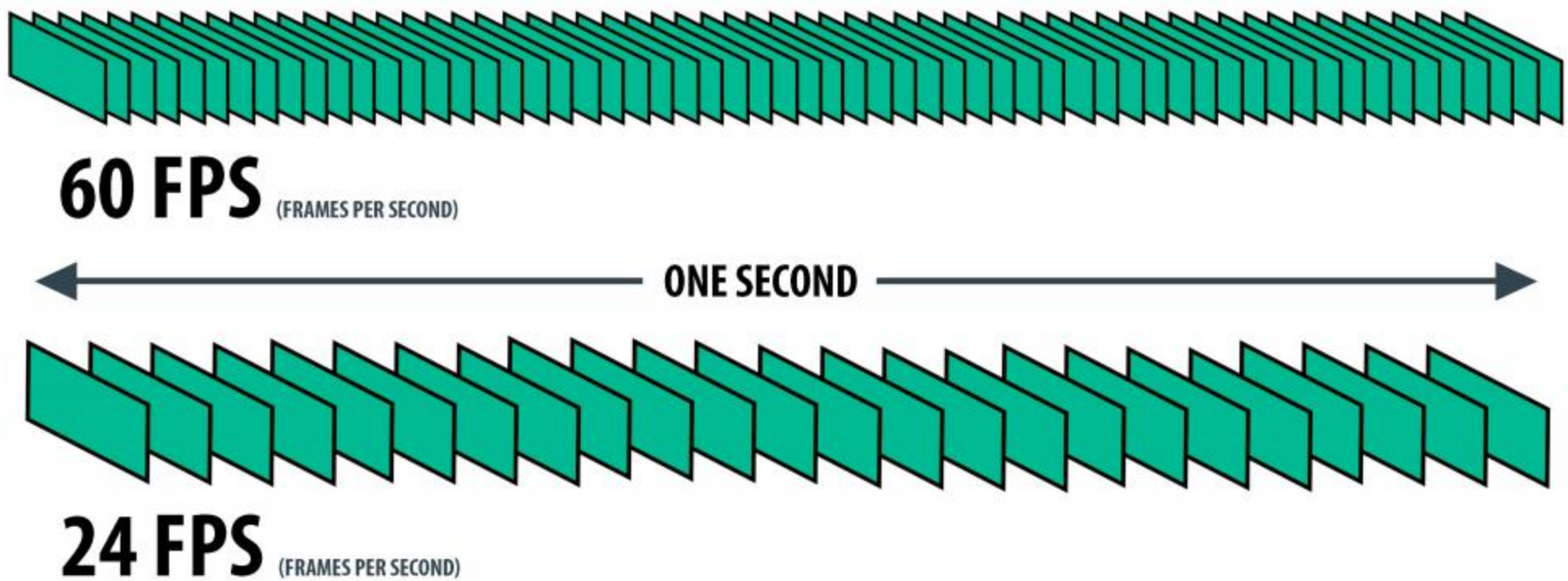


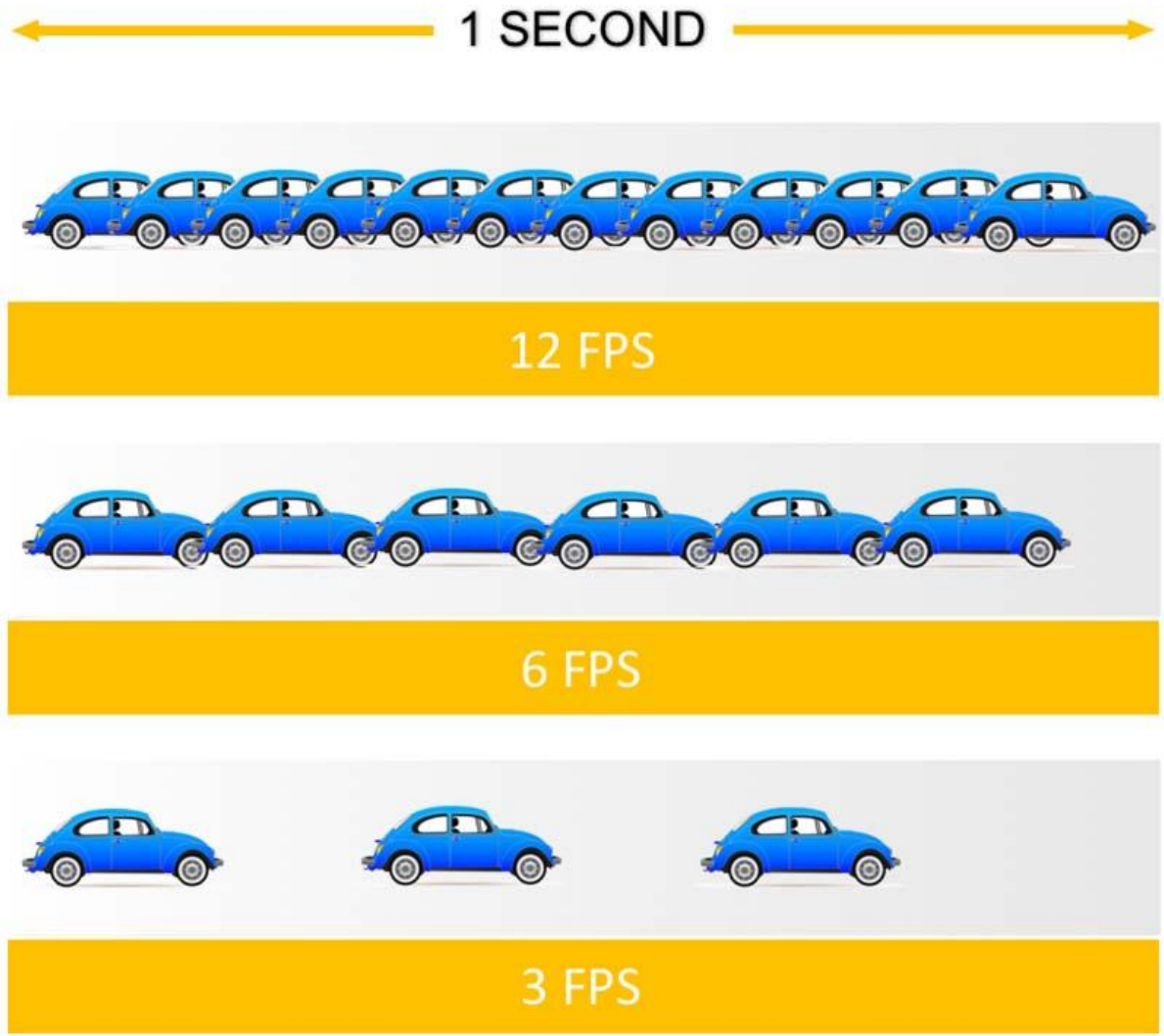
Faça suposições

- o que acontece se o cawboy e o chão forem configurados com colisor?
- o que acontece se o cawboy for configurado como corpo rígido e o chão for configurado como colisor?
- O que acontece se o chão for configurado como corpoRigido?
-

O que é FPS(frame per second)?

- O FPS é a sigla que determina a quantidade quadros por segundo em um filme, uma animação ou uma cena de um jogo.
- Quanto mais quadros, mais fluída será a animação e mais detalhado será o movimento
- O cérebro humano de transformar imagens em sequência em um “filme” dentro de nossas cabeças.


60 FPS (FRAMES PER SECOND)
ONE SECOND
24 FPS (FRAMES PER SECOND)

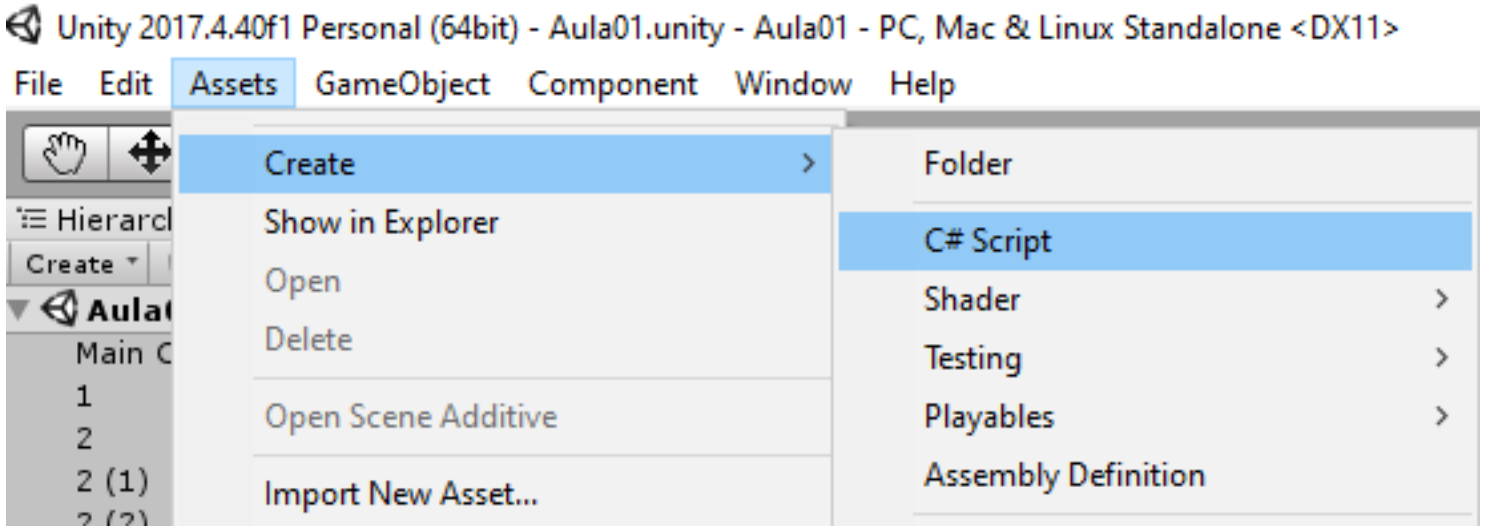

1 SECOND
12 FPS
6 FPS
3 FPS

Script na computação

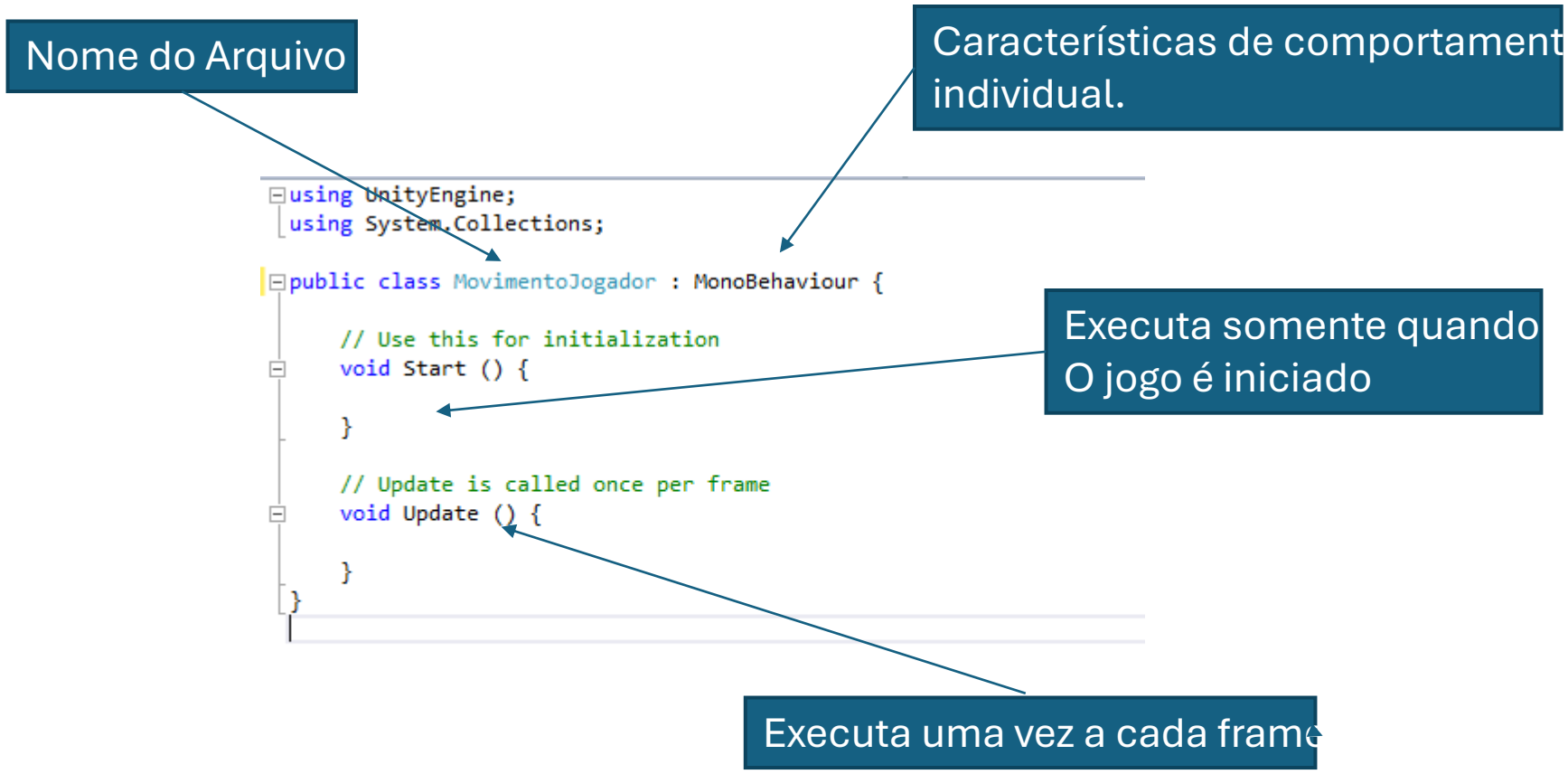
- Script é um texto com uma série de instruções escritas para serem seguidas por um computador.
- O termo é uma redução da palavra inglesa manuscript, que significa “manuscrito”, “escrito à mão”.

Criar um Script para o jogo

- Click no meu “Assets”>>”C# Script”
- Mude o nome do arquivo criado para : **Jogador** e efetue dois cliques para editar o script



Entenda o Script



Atributos

- Os atributos são características.
- São utilizados para descrever alguma coisa.
- Quais os atributos da sala?
- Quais os atributos da mesa?
- Quais os atributos de um personagem de jogo?

Atributos do Personagem

```
using UnityEngine;

// Classe que controla o movimento do personagem
public class PersonagemAula1 : MonoBehaviour {

    // Variáveis para controlar a velocidade do personagem
    float VelocidadeX; // Velocidade no eixo X (horizontal)
    float VelocidadeY; // Velocidade no eixo Y (vertical)
    float VelocidadeHorizontalMaxima; // Velocidade máxima que o personagem pode atingir no eixo X
    float DirecaoHorizontal; // Direção do movimento horizontal (-1 para esquerda, 1 para direita, 0 para parado)
    Vector2 VetorVelocidadePersonagem; // Vetor que armazena a velocidade do personagem em X e Y

    // Referência ao componente Rigidbody2D do personagem
    Rigidbody2D CorpoRigidoPersonagem;

   void Start () {

   }

   void Update () {

   }

}
```

```
using UnityEngine;

// Classe que controla o movimento do personagem
public class PersonagemAula1 : MonoBehaviour {

    // Variáveis para controlar a velocidade do personagem
    float VelocidadeX; // Velocidade no eixo X (horizontal)
    float VelocidadeY; // Velocidade no eixo Y (vertical)
    float VelocidadeHorizontalMaxima; // Velocidade máxima que o personagem pode atingir no eixo X
    float DirecaoHorizontal; //(-1 para esquerda, 1 para direita, 0 para parado)
    Vector2 VetorVelocidadePersonagem; // Vetor que armazena a velocidade do personagem em X e Y
    Rigidbody2D CorpoRigidoPersonagem;  // Referência ao componente Rigidbody2D do personagem

    // Método chamado uma vez no início do jogo
    void Start () {

        // Obtém o componente Rigidbody2D do objeto ao qual este script está anexado
        CorpoRigidoPersonagem = GetComponent<Rigidbody2D> ();

        // Define a escala da gravidade para o Rigidbody2D (1 é o valor padrão)
        CorpoRigidoPersonagem.gravityScale = 1f;

        // Congela a rotação do Rigidbody2D para evitar que o personagem gire ao colidir com algo
        CorpoRigidoPersonagem.freezeRotation = true;

        // Inicializa as variáveis de velocidade
        VelocidadeX = 0f; // Começa parado no eixo X
        VelocidadeY = 0f; // Começa parado no eixo Y
        VelocidadeHorizontalMaxima = 2.5f; // Define a velocidade máxima no eixo X
        DirecaoHorizontal = 0f; // Começa sem direção definida

        // Cria um vetor de velocidade inicial para o personagem
        VetorVelocidadePersonagem = new Vector2 (VelocidadeX, VelocidadeY);
    }

  void Update() {


  }
}
```

Inicialização de Variáveis

- O método **void Start()** deve ser utilizado para iniciar variáveis e propriedades do Personagem.
- Nesse momento o Corpo Rígido que foi adicionado ao personagem (Component>>Physic2D>>RigidBody2D) é inserido na variável **CorpoRigido**, assim é possível modificar propriedades de física utilizando a variável CorpoRigido.
- Observe que a propriedade freezeRotation é modificada para true, isso impede que o personagem rotacione quanto tocar em quinas.

Criar movimento horizontal

```
void Update () {
        // Input.GetAxis("Horizontal") retorna um valor entre -1 e 1:
        // -1 para esquerda, 1 para direita, 0 se nenhuma tecla estiver sendo pressionada
        DirecaoHorizontal = Input.GetAxis ("Horizontal");

        // Calcula a velocidade no eixo X multiplicando a direção pela velocidade máxima
        VelocidadeX = VelocidadeHorizontalMaxima * DirecaoHorizontal;

        // Mantém a velocidade atual no eixo Y (para não interferir na gravidade ou pulo)
        VelocidadeY = CorpoRigidoPersonagem.velocity.y;

        // Cria um novo vetor de velocidade com os valores calculados
        VetorVelocidadePersonagem = new Vector2 (VelocidadeX, VelocidadeY);

        // Aplica o vetor de velocidade ao Rigidbody2D para mover o personagem
        CorpoRigidoPersonagem.velocity = VetorVelocidadePersonagem;

}
```

DirecaoHorizontal Recebe até 1 para direita e até -1 para esquerda.
Quando parado Recebe 0.

= 5 * DirecaoHorizontal.

Determina o módulo, sentido e direção do movimento

Código Completo

```
using UnityEngine;
public class PersonagemAula1 : MonoBehaviour {
    // Variáveis para controlar a velocidade do personagem
    float VelocidadeX; // Velocidade no eixo X (horizontal)
    float VelocidadeY; // Velocidade no eixo Y (vertical)
    float VelocidadeHorizontalMaxima; // Velocidade máxima que o personagem pode atingir no eixo X
    float DirecaoHorizontal; // Direção do movimento horizontal (-1 para esquerda, 1 para direita, 0 para parado)
    Vector2 VetorVelocidadePersonagem; // Vetor que armazena a velocidade do personagem em X e Y
    Rigidbody2D CorpoRigidoPersonagem; // Referência ao componente Rigidbody2D do personagem
    void Start () {
        CorpoRigidoPersonagem = GetComponent<Rigidbody2D> ();// Obtém o Rigidbody2D do objeto ao qual este script está anexado
        // Define a escala da gravidade para o Rigidbody2D (1 é o valor padrão)
        CorpoRigidoPersonagem.gravityScale = 1f;
        // Congela a rotação do Rigidbody2D para evitar que o personagem gire ao colidir com algo
        CorpoRigidoPersonagem.freezeRotation = true;
        VelocidadeX = 0f; // Começa parado no eixo X
        VelocidadeY = 0f; // Começa parado no eixo Y
        VelocidadeHorizontalMaxima = 2.5f; // Define a velocidade máxima no eixo X
        DirecaoHorizontal = 0f; // Começa sem direção definida
        VetorVelocidadePersonagem = new Vector2 (VelocidadeX, VelocidadeY);
        CorpoRigidoPersonagem.velocity = VetorVelocidadePersonagem;
    }
    void Update () {    // Método chamado uma vez por frame (quadro)
        // -1 para esquerda, 1 para direita, 0 se nenhuma tecla estiver sendo pressionada
        DirecaoHorizontal = Input.GetAxis ("Horizontal");
        // Calcula a velocidade no eixo X multiplicando a direção pela velocidade máxima
        VelocidadeX = VelocidadeHorizontalMaxima * DirecaoHorizontal;
        // Mantém a velocidade atual no eixo Y (para não interferir na gravidade ou pulo)
        VelocidadeY = CorpoRigidoPersonagem.velocity.y;
        // Cria um novo vetor de velocidade com os valores calculados
        VetorVelocidadePersonagem = new Vector2 (VelocidadeX, VelocidadeY);
        // Aplica o vetor de velocidade ao Rigidbody2D para mover o personagem
        CorpoRigidoPersonagem.velocity = VetorVelocidadePersonagem;

    }
}
```

Melhorando o código - Organização

- Conforme o andar do desenvolvimento de um jogo é possível verificar que o método Update ganha mais linhas de código conforme o Personagem ganha funcionalidade.
- É possível que ao final do desenvolvimento o jogo possua centenas de linhas de código ou mais.
- Para melhor organizar o que é programado em **void Update()** você pode criar blocos de código separados, onde cada bloco possuí uma funcionalidade diferente.

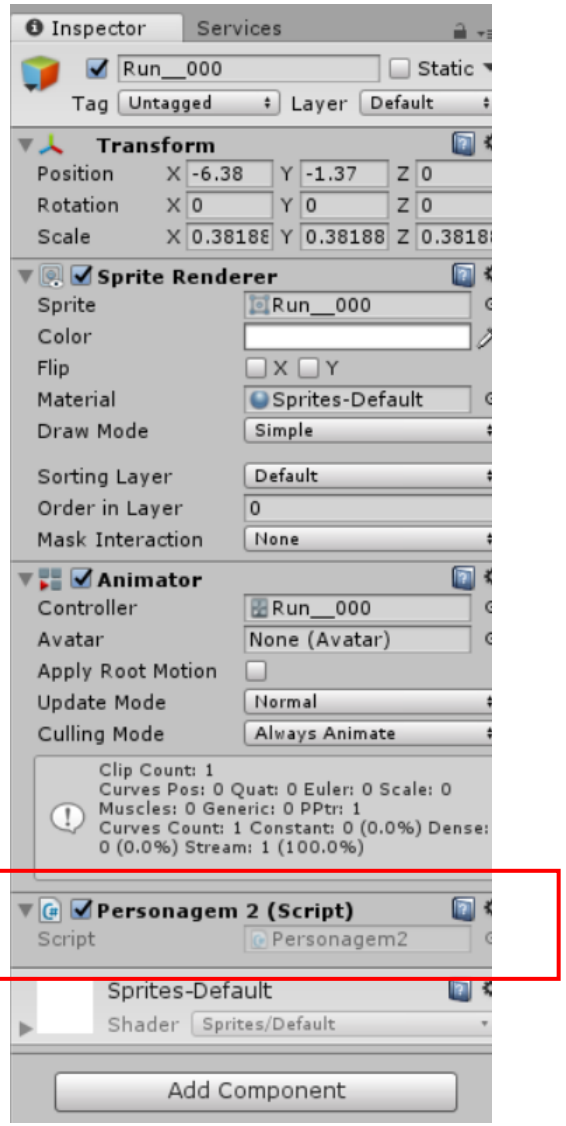
Organização

```
// Classe que controla o movimento do personagem
public class PersonagemAula1 : MonoBehaviour {

    // Variáveis para controlar a velocidade do personagem
    float VelocidadeX; // Velocidade no eixo X (horizontal)
    float VelocidadeY; // Velocidade no eixo Y (vertical)
    float VelocidadeHorizontalMaxima; // Velocidade máxima que o personagem pode atingir no eixo X
    float DirecaoHorizontal; // Direção do movimento horizontal (-1 para esquerda, 1 para direita, 0 para parado)
    Vector2 VetorVelocidadePersonagem; // Vetor que armazena a velocidade do personagem em X e Y
    Rigidbody2D CorpoRigidoPersonagem; // Referência ao componente Rigidbody2D do personagem
    void Start () {    // Método chamado uma vez no início do jogo
        CorpoRigidoPersonagem = GetComponent<Rigidbody2D> (); // Obtém o componente Rigidbody2D do objeto ao qual este script está anexado
        CorpoRigidoPersonagem.gravityScale = 1f;// Define a escala da gravidade para o Rigidbody2D (1 é o valor padrão)
        CorpoRigidoPersonagem.freezeRotation = true; // Congela a rotação do Rigidbody2D para evitar que o personagem gire ao colidir com algo
        // Inicializa as variáveis de velocidade
        VelocidadeX = 0f; // Começa parado no eixo X
        VelocidadeY = 0f; // Começa parado no eixo Y
        VelocidadeHorizontalMaxima = 2.5f; // Define a velocidade máxima no eixo X
        DirecaoHorizontal = 0f; // Começa sem direção definida
        VetorVelocidadePersonagem = new Vector2 (VelocidadeX, VelocidadeY);// Cria um vetor de velocidade inicial para o personagem
        CorpoRigidoPersonagem.velocity = VetorVelocidadePersonagem; // Aplica o vetor de velocidade ao Rigidbody2D para mover o personagem
    }
    void Update () {    // Método chamado uma vez por frame (quadro)
        MovimentoHorizontal (); // Chama o método para controlar o movimento horizontal do personagem
    }
    void MovimentoHorizontal(){ // Método para controlar o movimento horizontal do personagem
        // Obtém a direção horizontal a partir do input do jogador (teclado ou controle)
        // Input.GetAxis("Horizontal") retorna um valor entre -1 e 1:
        // -1 para esquerda, 1 para direita, 0 se nenhuma tecla estiver sendo pressionada
        DirecaoHorizontal = Input.GetAxis ("Horizontal");

        VelocidadeX =VelocidadeHorizontalMaxima * DirecaoHorizontal;// Calcula a velocidade no eixo X multiplicando a direção pela velocidade máxima
        VelocidadeY = CorpoRigidoPersonagem.velocity.y; // Mantém a velocidade atual no eixo Y (para não interferir na gravidade ou pulo)
        VetorVelocidadePersonagem = new Vector2 (VelocidadeX, VelocidadeY); // Cria um novo vetor de velocidade com os valores calculados
        CorpoRigidoPersonagem.velocity = VetorVelocidadePersonagem; // Aplica o vetor de velocidade ao Rigidbody2D para mover o personagem
    }
}
```

Nesse momento do curso trataremos como um bloco de código que pode será chamado quando acontecer um update

Adicione o scritp no personagem

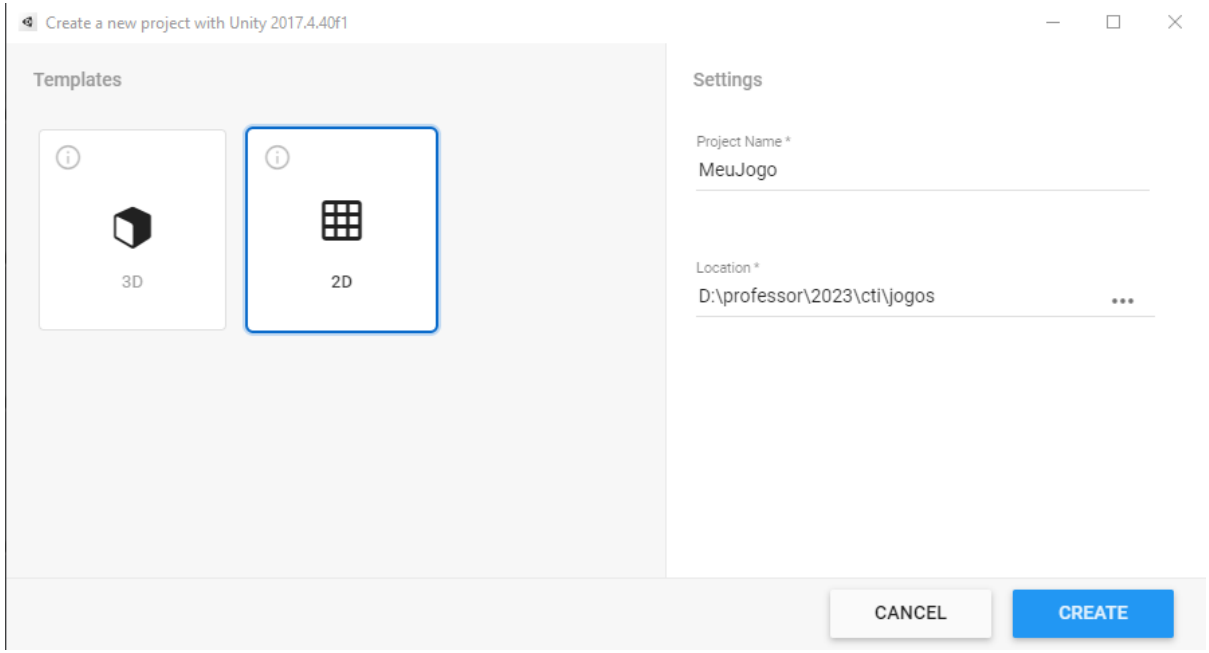
- Para que o script comece a funcionar é necessário que ele seja inserido dentro do personagem.
  - Click no arquivo do script e arraste para dentro do **asset** do personagem localizado na cena.
    - Se não existir nenhum problema no scritp o mesmo aparecerá na lista de propriedades do personagem no inspector



Script inserido no asset do personagem

Salvar o projeto

- Lembre-se da pasta que você escolheu para salvar os arquivos do projeto 2d.
- No exemplo ao lado os arquivos serão salvos na pasta:
  - **d:\professor\2023\cti\jogos\meujogo**
- Antes de fechar o jogo salve a cena.
- Menu:
  - File>> Save Scene>> de um nome para o cenário
- Salve todo o conteúdo da pasta:
  - **d:\professor\2023\cti\jogos\meujogo**
- **Veja**: https://youtu.be/mT122XNuAPU



Salvar e Abrir Projeto de jogo no Unity

https://youtu.be/mT122XNuAPU

Git Hub da Disciplina

https://github.com/helioesperidiao/Desenvolvimento-de-jogos

Tutoriais sobre github

- https://www.youtube.com/watch?v=EGmzAs1G0z0
- https://www.youtube.com/watch?v=nAHVEzDBVeo
- https://www.youtube.com/watch?v=P9xXbEhqhqA