Movimento Vertical e Horizontal

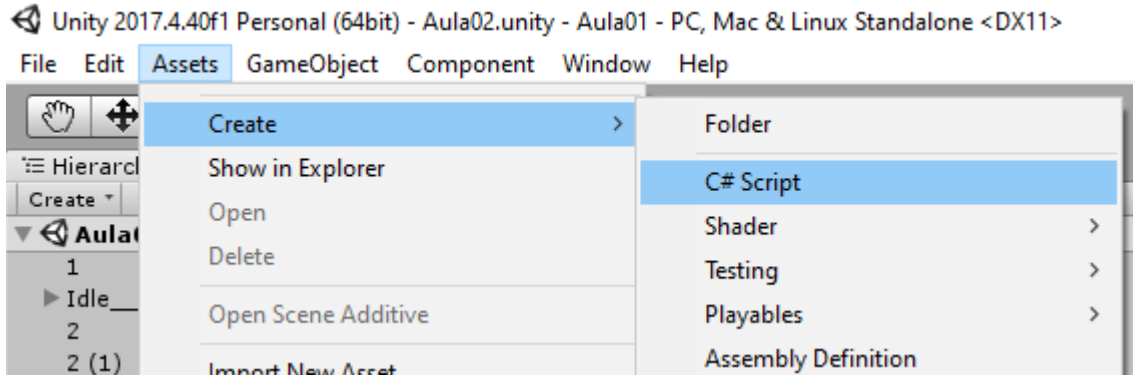
Prof. Me. Hélio Esperidiaõ

Atributos

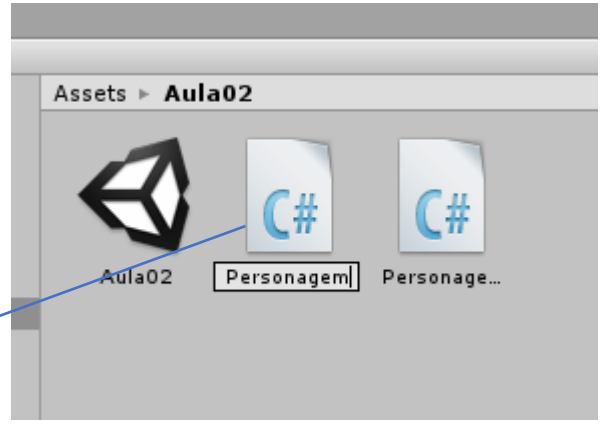
- Os atributos são características.
- São utilizados para descrever alguma coisa.
- Quais os atributos da sala?
- Quais os atributos da mesa?
- Quais os atributos de um personagem de jogo?

Crie um Script

- Será criado um arquivo na sessão de assets do Unity.
- Nomeie o arquivo com as mesmas regras de declaração de variáveis
  - Mais a regra da primeira letra em máscula.
  - Efetue dois cliques no arquivo gerado que o editor de código abrirá o código



Código base




```
using System.Collections;
using UnityEngine;
public class Personagem : MonoBehaviour {

    void Start () {
            //Quando é executado?
    }

    void Update () {
            //Quando é executado?
    }

}
```

Atributos do Personagem

- Os atributos neste momento do curso podem ser considerados variáveis que devem ser utilizadas para representar características do personagem.
- **Vx: velocidade em x do personagem**
- **Vy: velocidade em Y do personagem**
- VelocidadeMovimento: Velocidade de movimentação do personagem.
- **DirecaoHorizontal**: Se o personagem deve se mover para esquerda ou direita.
- **CorpoRigido**: Variável que contem todas as funções de física do personagem.
- Por que do personagem?
- Porque o Script será posicionado e utilizado pelo personagem.

```
using System.Collections;
using UnityEngine;
public class Personagem : MonoBehaviour {
                float Vx;
                float Vy;
                float VelocidadeMovimento;
                float DirecaoHorizontal;
                Rigidbody2D CorpoRigido;

    void Start () {

    }

    void Update () {

    }

}
```

Inicialização de Variáveis

```
using System.Collections;
using UnityEngine;
public class Personagem : MonoBehaviour {
        float Vx;
        float Vy;
        float VelocidadeMovimento;
        float DirecaoHorizontal;
        Rigidbody2D CorpoRigido;
        void Start () {
                VelocidadeMovimento = 5;
                DirecaoHorizontal = 0;
                CorpoRigido = GetComponent<Rigidbody2D> ();
                CorpoRigido.freezeRotation = true;
        }
        void Update () {


        }
}
```

- 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

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

= 5 * DirecaoHorizontal.

```
void Update () {
    DirecaoHorizontal = Input.GetAxis ("Horizontal");
    Vx = VelocidadeMovimento * DirecaoHorizontal;
    Vy = CorpoRigido.velocity.y;
    Vector2 vetorMovimento = new Vector2 (Vx, Vy);
    CorpoRigido.velocity = vetorMovimento;
}
```

Determina o módulo, sentido e direção do movimento

Código Completo

```
using System.Collections;
using UnityEngine;
public class Personagem : MonoBehaviour {
        float Vx; //Velocidade em X do Personagem
        float Vy; // Velocidade em Y do Personagem
        float VelocidadeMovimento;  //Velocidade de movimento do personagem
        float DirecaoHorizontal;    // direção que o personagem deve ser mover (-1,0,1)
        Rigidbody2D CorpoRigido;
        void Start () { //INICIALIZA VARIÁVEIS E ATRIBUTOS
                VelocidadeMovimento = 5;    //Inicializa a variável VelocidadeMovimento
                DirecaoHorizontal = 0;    //Inicializa a variável DirecaoHorizontal
                // Inicializa a variável com o corpo rígido do personagem.
                // É necessário que o corpo rígido tenha sido inserido no personagem.
                CorpoRigido = GetComponent<Rigidbody2D> ();
                CorpoRigido.freezeRotation = true; //Impede que o personagem rotacione no eixo.
        }
        void Update () { // É EXECUTADO UMA VEZ POR FRAME
                //DirecaoHorizontal Recebe até 1 para direita e  até -1 para esquerda. Quando parado Recebe 0.
                DirecaoHorizontal = Input.GetAxis ("Horizontal");
                Vx = VelocidadeMovimento * DirecaoHorizontal; // Gera uma nova velocidade em x
                Vy = CorpoRigido.velocity.y; //Recupera a velocidade em y que já existe
                Vector2 vetorMovimento = new Vector2 (Vx, Vy); //Cria um vetor de velocidade
                // O vetor de velocidade é adicionado a velocidade do corpo rígido do personagem
                CorpoRigido.velocity = vetorMovimento;
        }
}
```

Melhorando o código - Organização

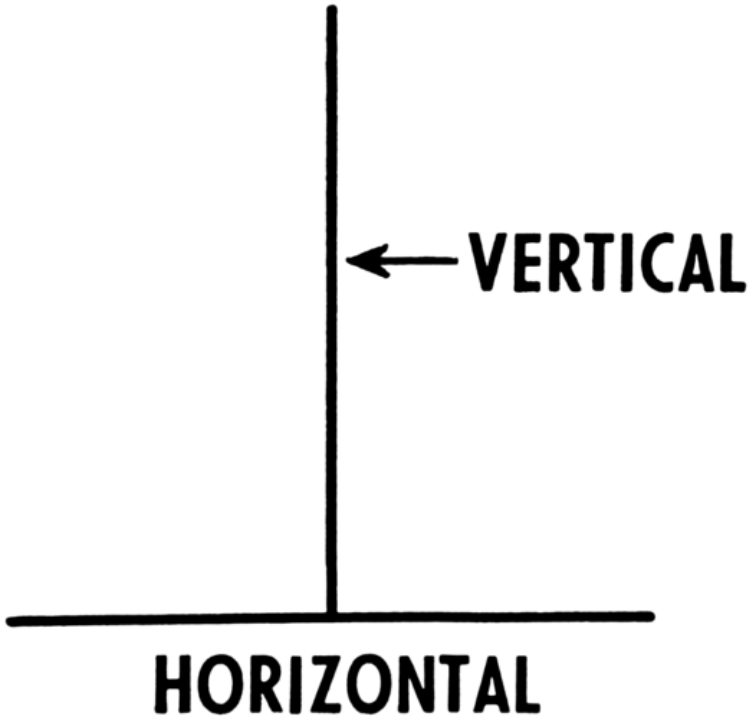
- Conforme o andar do desenvolvimento de um jogo é possível verificar que o método Update ganha mais linhas de código conforme o Personagem ganha funcionalidade.
- É possível que ao final do desenvolvimento o jogo possua centenas de linhas de código ou mais.
- Para melhor organizar o que é programado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

Melhorando o código - Organização

```
using System.Collections;
using UnityEngine;
public class Personagem : MonoBehaviour {
		float Vx;
		float Vy;
		float VelocidadeMovimento;
		float DirecaoHorizontal;
		Rigidbody2D CorpoRigido;
		void Start () {
				VelocidadeMovimento = 5;
				DirecaoHorizontal = 0;
				CorpoRigido = GetComponent<Rigidbody2D> ();
				CorpoRigido.freezeRotation = true;
		}
		void Update () {
				//chama a todo frame O código dentro de MovimentoHorizontal()
				MovimentoHorizontal();
		}
		void MovimentoHorizontal(){
				DirecaoHorizontal = Input.GetAxis ("Horizontal");
				Vx = VelocidadeMovimento * DirecaoHorizontal;
				Vy = CorpoRigido.velocity.y;
				Vector2 vetorMovimento = new Vector2 (Vx, Vy);
				CorpoRigido.velocity = vetorMovimento;
		}
}
```

Nesse momento do curso trataremos como um bloco de código que pode ser chamado quando necessário

Movimento Vertical



Gravity Scale

É uma propriedade que multiplica a gravidade de um corpo rígido por um valor.

O que acontece se a escala de gravidade for igual a 1?

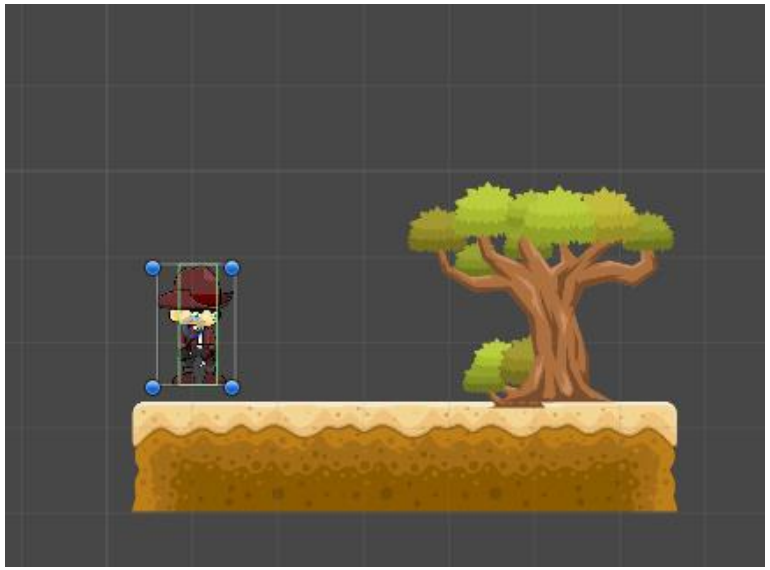
O que acontece se a escala de gravidade for igual a 0?

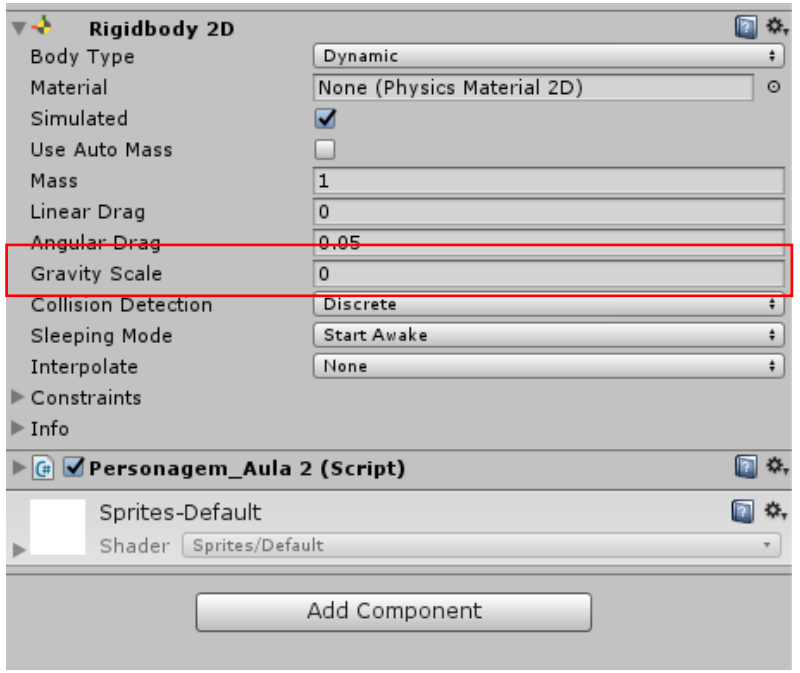
O que acontece se a escala de gravidade for igual a -1?

O que acontece se a escala de gravidade for igual a 3?

Escala de gravidade

- Configure o personagem como **colisor**.
- Configure o personagem como corpo **rígido**.
- Clique no personagem e procure a propriedade Gravity Scale.
  - Mude o *Gravity Scale* para zero





Atributos do Personagem

- Os atributos neste momento do curso podem ser considerados variáveis que devem ser utilizadas para representar características do personagem.
- **DirecaoHorizontal**: Se o personagem deve se mover para esquerda ou direita.
- **DirecaoVertical:** Se o personagem deve se mover para cima ou para baixo

```
using System.Collections;
using UnityEngine;
public class Personagem : MonoBehaviour {
        float Vx;
        float Vy;
        float VelocidadeMovimento;
        float DirecaoHorizontal;
        float DirecaoVertical;
        Rigidbody2D CorpoRigido;

    void Start () {

    }

    void Update () {

    }

}
```

void Start ()

```
public class Personagem : MonoBehaviour {
    float Vx;
    float Vy;
    float VelocidadeMovimento;
    float DirecaoHorizontal;
    float DirecaoVertical;
    Rigidbody2D CorpoRigido;
    void Start () {
        VelocidadeMovimento = 5;
        DirecaoHorizontal   = 0;
        DirecaoVertical     = 0;
        CorpoRigido = GetComponent<Rigidbody2D> ();
        CorpoRigido.gravityScale=0;
        CorpoRigido.freezeRotation = true;
}
```

- Observe que a propriedade **gravityScale** é modificada para 0, isso impede que o personagem sofra ação da gravidade.
- Isso permite que ele se movimente para cima ou para baixo.

void MovimentoVertical()

DirecaoVertical Recebe até 1 para cima e até -1 para baixo.
Quando parado Recebe 0.

```
void MovimentoVertical(){
    DirecaoVertical = Input.GetAxis ("Vertical");
    Vx = CorpoRigido.velocity.x;
    Vy = VelocidadeMovimento * DirecaoVertical;
    Vector2 vetorMovimento = new Vector2 (Vx, Vy);
    CorpoRigido.velocity = vetorMovimento;
}
```

Vy= 5 * **DirecaoVertical**.

Determina o módulo, sentido e direção do movimento

Código completo

```
using System.Collections;
using UnityEngine;
public class Personagem_Aula2 : MonoBehaviour {
    float Vx;    //Velocidade em X do Personagem
    float Vy;    // Velocidade  em Y do Personagem
    float VelocidadeMovimento; //Velocidade de movimento do personagem
    float DirecaoHorizontal;// direção que o personagem deve ser mover (-1,0,1)
    float DirecaoVertical;    // direção que o personagem deve ser mover (-1,0,1)
    Rigidbody2D CorpoRigido;
    void Start () {
        //inicializa variáveis
        VelocidadeMovimento = 5;
        DirecaoHorizontal   = 0;
        DirecaoVertical     = 0;
        CorpoRigido = GetComponent<Rigidbody2D> ();
        CorpoRigido.gravityScale=0;
        CorpoRigido.freezeRotation = true;
    }
    void Update () {
        //chama a todo momento Os blocos:
        MovimentoHorizontal();
        MovimentoVertical();
    }
    void MovimentoVertical(){
        DirecaoVertical = Input.GetAxis ("Vertical");
        Vx = CorpoRigido.velocity.x;
        Vy = VelocidadeMovimento * DirecaoVertical;
        Vector2 vetorMovimento = new Vector2 (Vx, Vy);
        CorpoRigido.velocity = vetorMovimento;
    }
    void MovimentoHorizontal(){
        DirecaoHorizontal = Input.GetAxis ("Horizontal");
        Vx = VelocidadeMovimento * DirecaoHorizontal;
        Vy = CorpoRigido.velocity.y;
        Vector2 vetorMovimento = new Vector2 (Vx, Vy);
        CorpoRigido.velocity = vetorMovimento;
    }
}
```

- Dentro do update antes de cada frame é chamado o “bloco” **MovimentoHorizontal()** e o “bloco” **MovimentoVertical()**