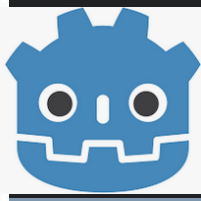
Criando um jogo básico em Godot com C#

Prof. Thiago Felski Pereira, MSc.

Visão Geral

- Porque C# se a linguagem principal é o GDScript?
  - Porque o C# é utilizado em todo lugar, enquanto o GDScript é utilizado apenas no Godot.
    - Você pode utilizar todo “arsenal” do .Net para seu jogo, como por exemplo utilizar bibliotecas de aprendizado de máquina para criar inimigos que aprendem com seus próprios erros.
    - Utilizada em outros ambientes de desenvolvimento de jogos populares: Unity.



Preparando o ambiente

- Crie uma pasta para seu jogo.

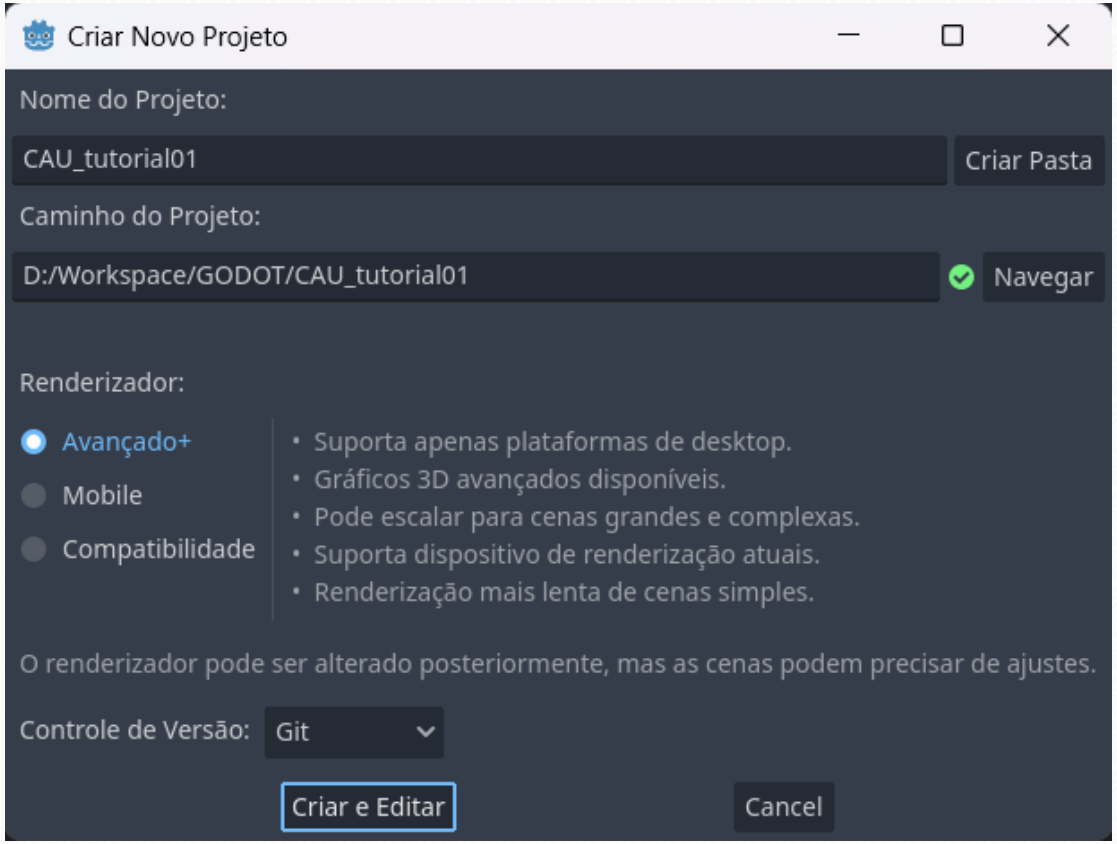
  - Eu chamei a minha de **CAU_tutorial01**, mas vocês podem chamar como quiserem.

UNIVALI

Preparando o ambiente

- Abra o aplicativo do Godot

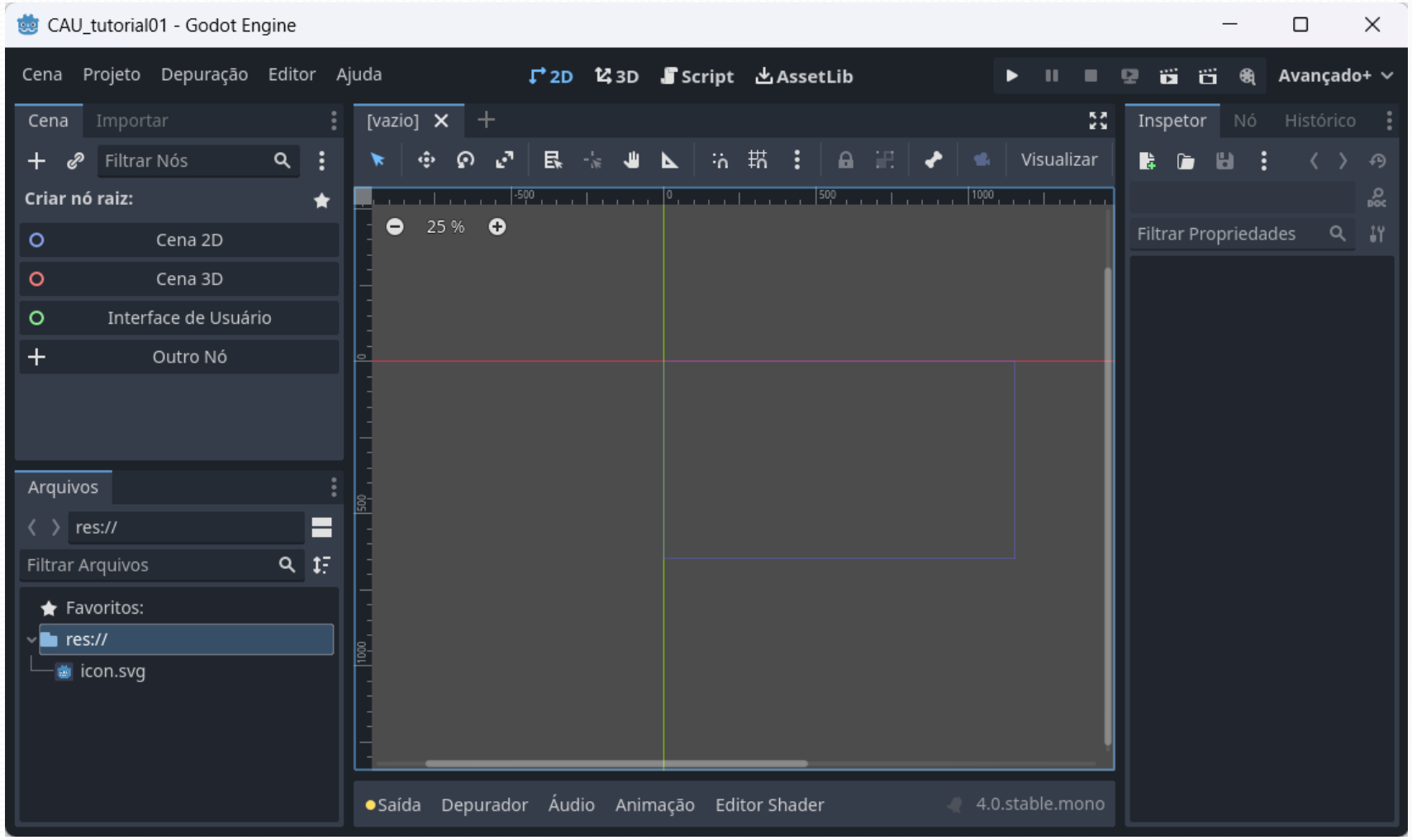
- Crie um novo projeto, escolhendo um nome e a pasta que você criou para ele



UNIVALI

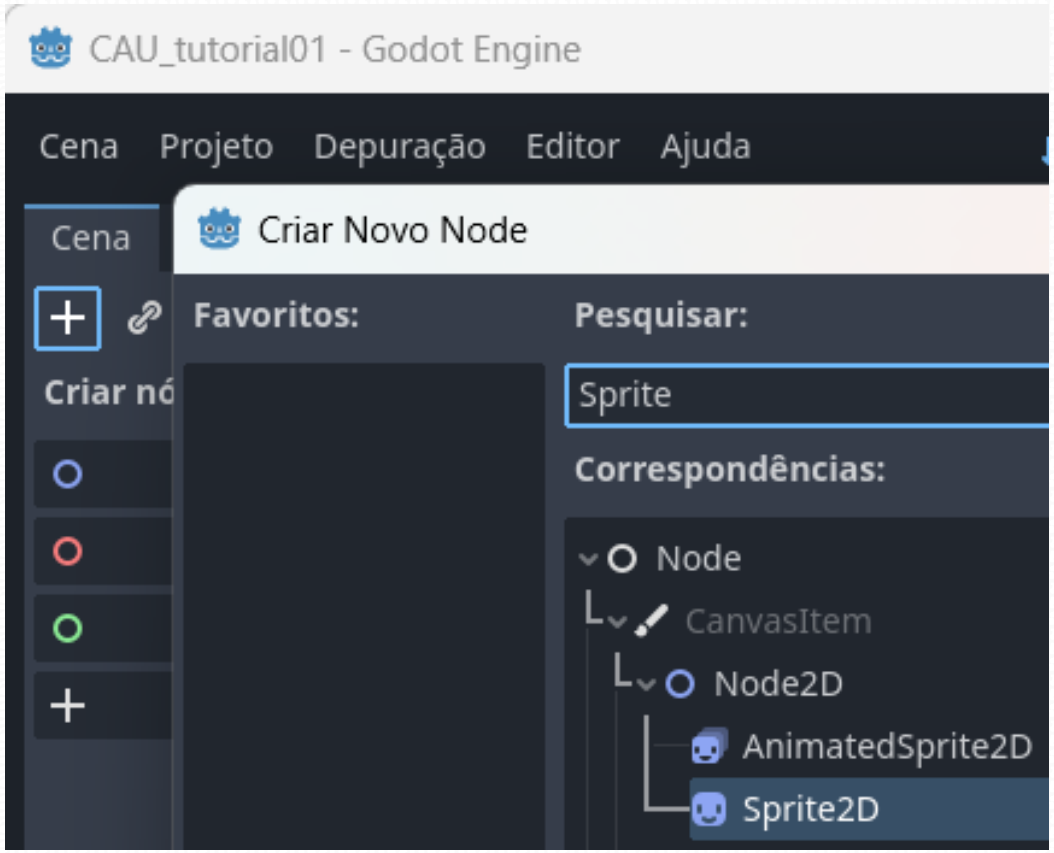
Preparando o ambiente

- Configure o ambiente para 2D, pois assim será nosso primeiro projeto



Desenvolvendo o jogo

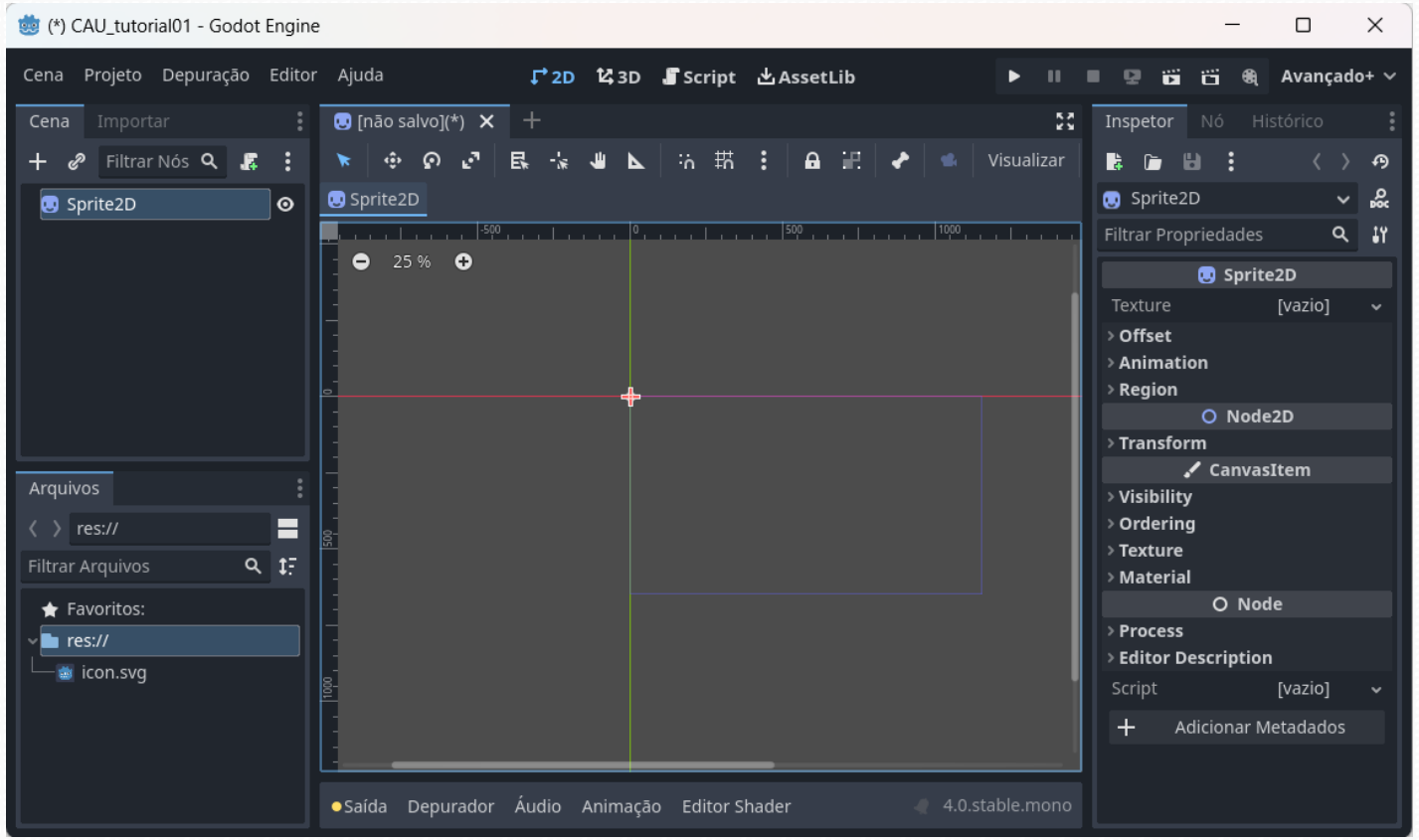
- O Godot é composto de
  - **Nodos**/Nodes: cada objeto individual dentro do nosso jogo
  - **Cenas**/Scenes: um agrupamento de Nodos.
- Crie, seu primeiro Nodo
  - + -> Sprite2D



UNIVALI

Desenvolvendo o jogo

- É possível observar os objetos criados em 2 locais
  - Na Hierarquia
  - Na Cena (não aparece nada lá ainda, pois ainda não associamos nenhuma imagem a nosso Sprite2D)
  - Note que na direita temos o **Inspetor**, que nos permite visualizar as características dos objetos selecionados

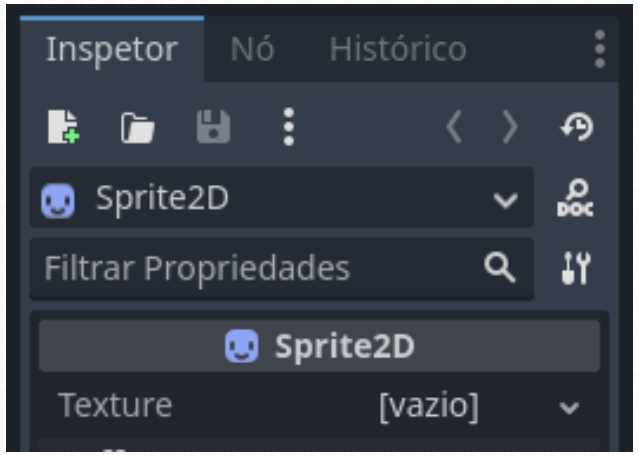


UNIVALI

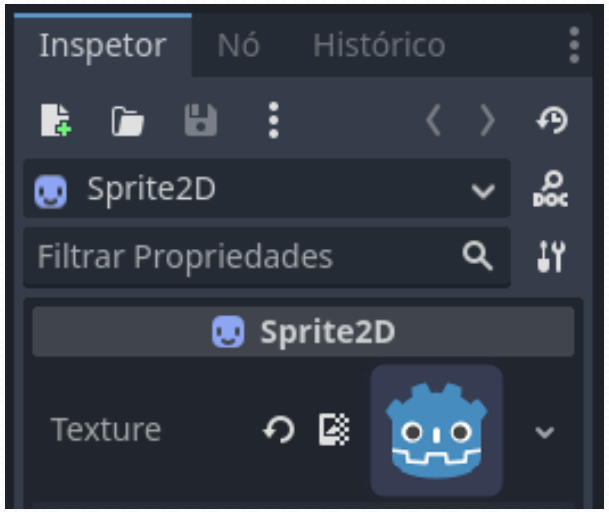
Desenvolvendo o jogo

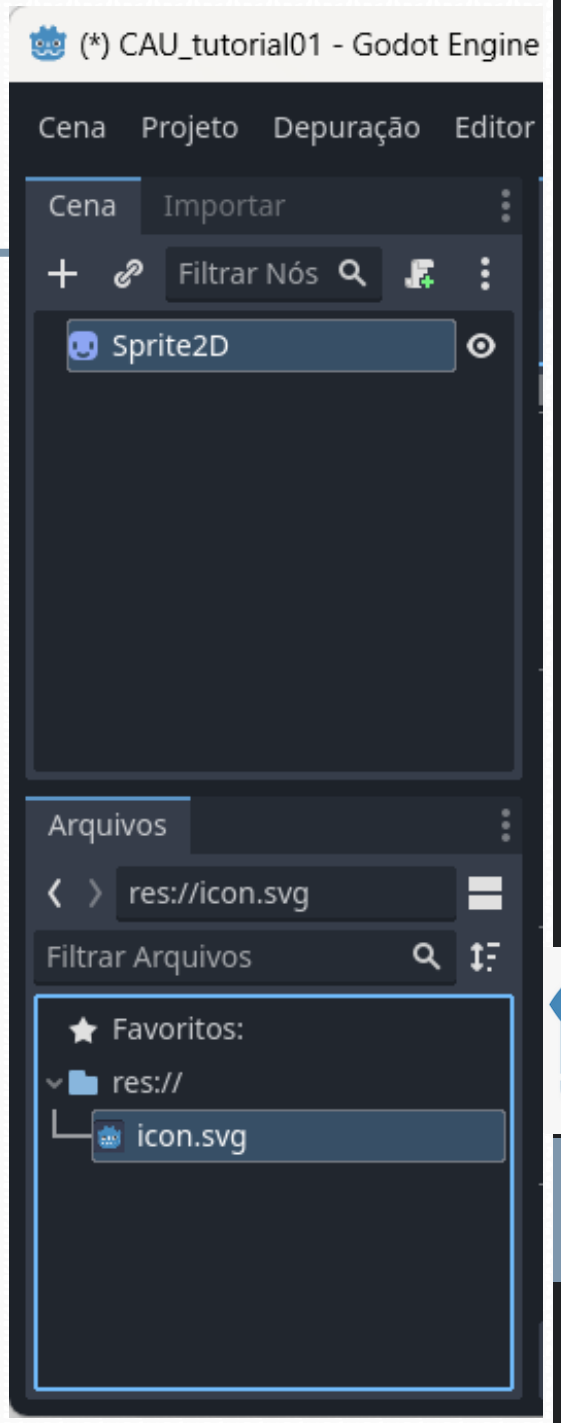
- Se você é bom observador deve ter notado um ícone na pasta raiz do projeto icon.svg
  - Arraste ele para a textura do Sprite2D

- Antes



Depois





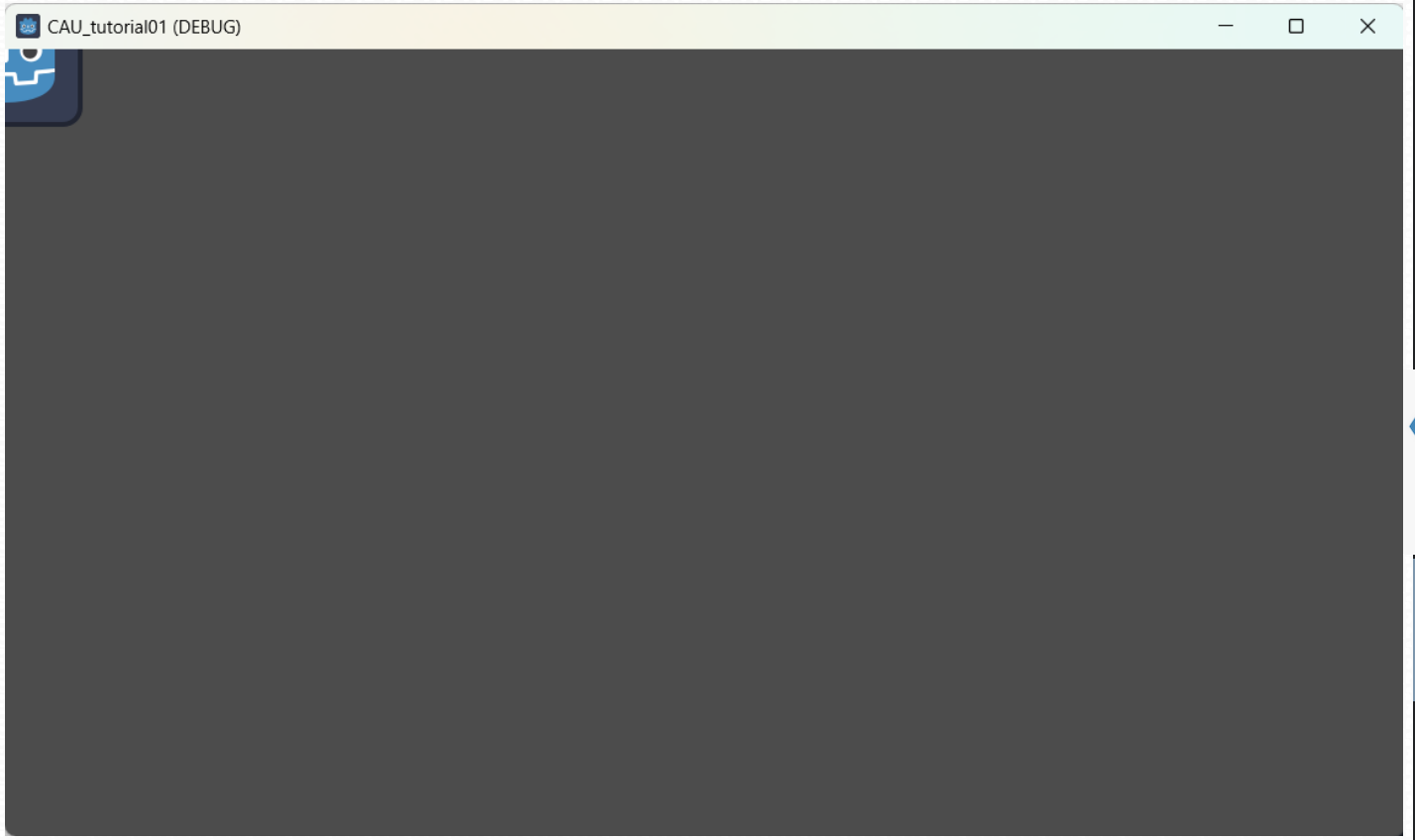
UNIVALI

Desenvolvendo o jogo

- Salve seu Nodo: **CTRL+S**

  - Note que mesmo tendo um único Nodo ele salva esse nodo como uma Cena.

- Execute a cena atual salva: **F6**



UNIVALI

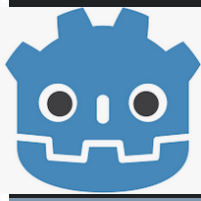
Desenvolvendo o jogo

- Nosso personagem ficou fora da área visível.

- Podemos corrigir isso manualmente.
  - Clicando e arrastando mais para o centro.

- Podemos corrigir isso nos atributos do Inspetor.
  - Inspetor -> Node2D -> Transform -> Position
  - E ajustar ele no eixo (X,Y) do nosso plano cartesiano.
  - Esse é o atributo que iremos alterar quando formos movimentar o personagem no jogo.

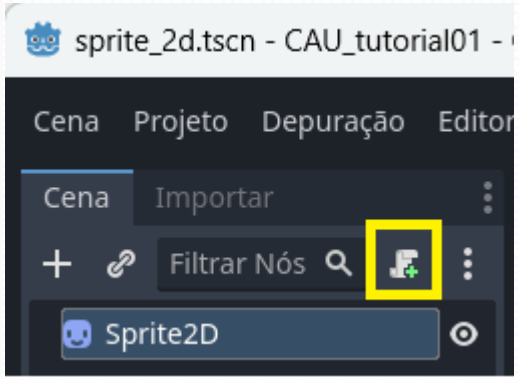
- Faça alguns testes e observe esses atributos com cuidado.
  - Pode ir executando a Cena para observar essas mudanças lá também.



UNIVALI

Programando o jogo

- Programando a movimentação do personagem.
- Crie um **Script** para o nodo, ou seja, um programa.

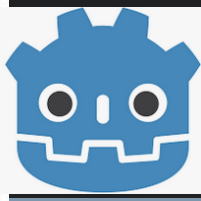


- Troque a linguagem para C# e deixe o resto como está.

Programando o jogo

- Estrutura normal de um Script C# para um Nodo

```
1   using Godot;
2   using System;
3
4   public partial class sprite_2d : Sprite2D
5   {
6
7       // Chamado quando o Nodo entra na Cena pela primeira vez.
8       public override void _Ready()
9       {
10      }
11
12      // Chamado a cada frame. 'delta' é o tempo aproximado de um frame
13      public override void _Process(double delta)
14      {
15
16      }
17  }
```



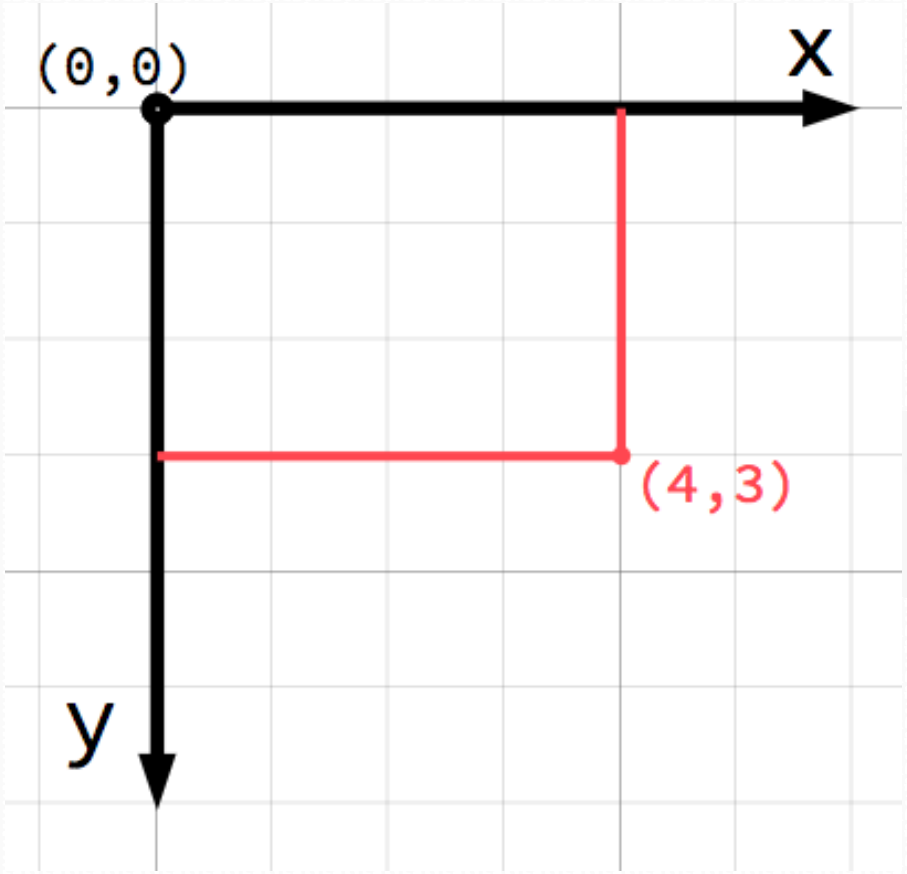
UNIVALI

Programando o jogo

- O código a seguir permite redefinir a posição do nosso objeto na tela

```
public override void _Ready()
{
    GlobalPosition = new Vector2(100,100);
}
```

  - `GlobalPosition` é uma variável que diz qual a posição global do nosso objeto no cenário.
  - Essa posição é definida por um plano cartesiano como o apresentado na imagem.
  - Note também que o ponto 0,0 está posicionado no canto superior esquerdo da nossa tela visível.



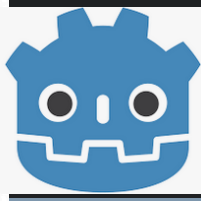
UNIVALI

Programando o jogo

- O código a seguir permite mover o objeto na tela.

```
public override void _Process(double delta)
{
    GlobalPosition = GlobalPosition + new Vector2(1,0);
}
```

- Mas porque move?
  - Esse comando foi colocado no `_Process()` o que faz com que o comando seja chamado a cada frame. `delta` é o tempo aproximado de um frame
  - Dessa forma a função
  - `GlobalPosition = GlobalPosition + new Vector2(1,0);`
  - Atribui a posição global do objeto a sua posição anterior + (1,0) na posição x,y do objeto.
  - Ou seja, a posição fica a anterior mais 1 no eixo x e 0 no eixo y.

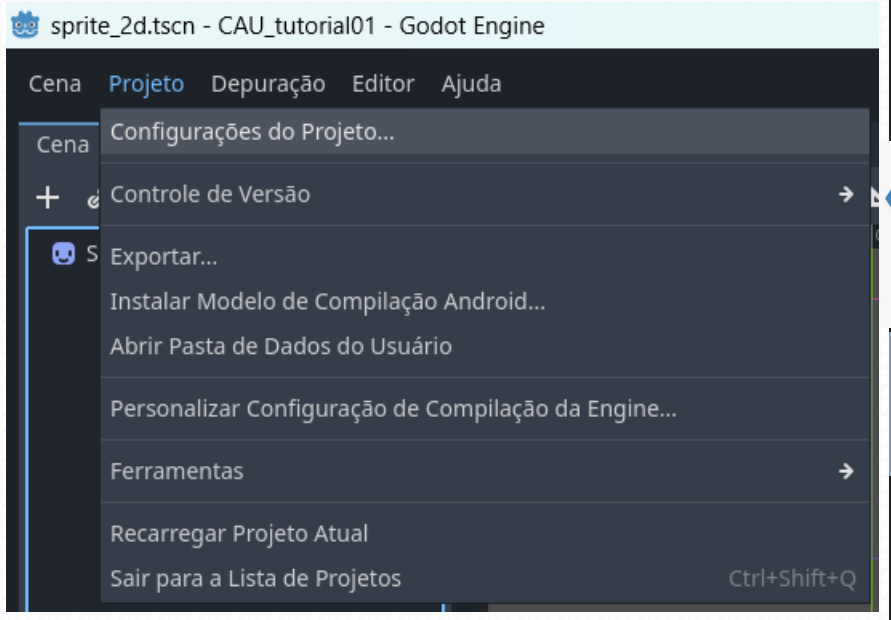


UNIVALI

Programando o jogo (Desvios)

- Vamos nos preparar para aplicar a técnica de desvios no nosso jogo.
- Nosso primeiro uso será executar um comado somente quando uma tecla for pressionada.

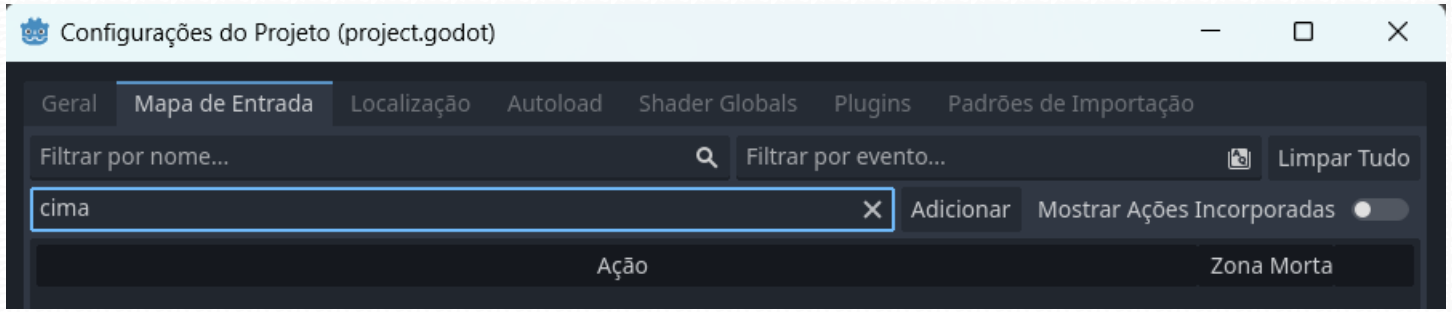
- O primeiro passo para isso será fazer o mapeamento de entradas no Godot
  - `Projeto -> Configurações do Projeto...`



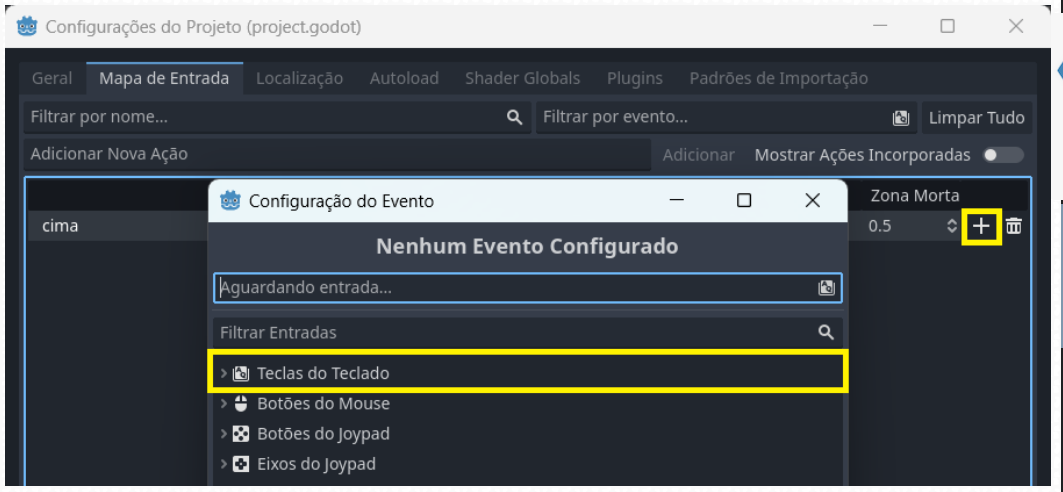
UNIVALI

Programando o jogo (Desvios)

- A seguir selecione a aba `Mapa de Entrada`
  - Adicione uma nova ação chamada `cima`



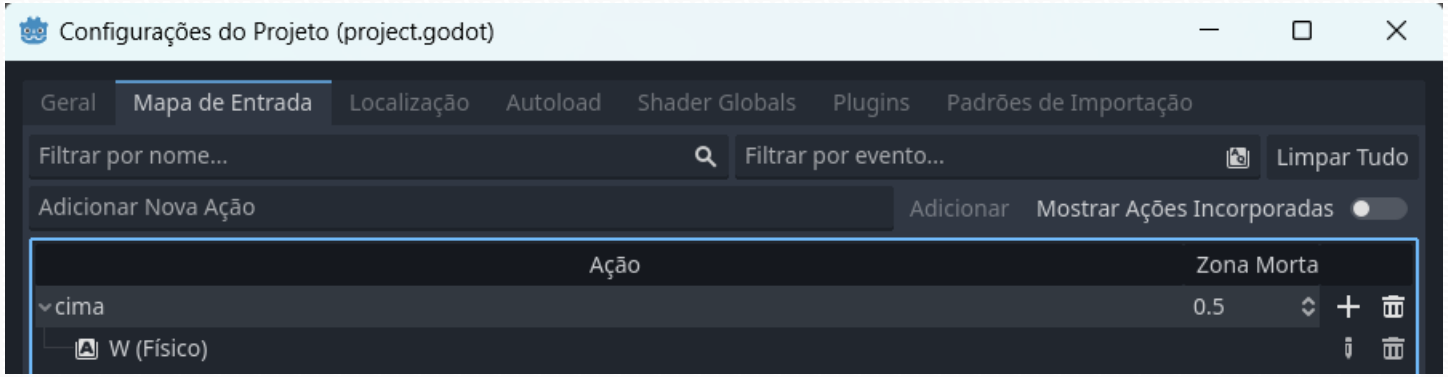
  - Adicione um evento na Ação cima clicando no `+`
  - Em seguida expanda `Teclas do Teclado`
  - Adicione a tecla `W`



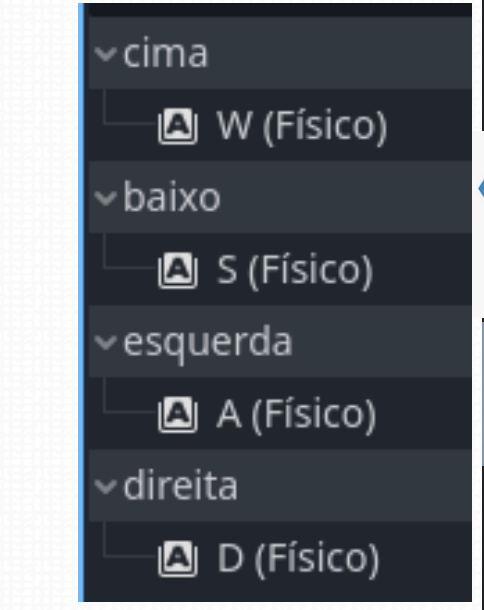
UNIVALI

Programando o jogo (Desvios)

- Se tudo deu certo você terá uma tela parecida com essa



- Repita o processo para `baixo`, `esquerda` e `direita`
  - Respectivamente `S`, `A` e `D`



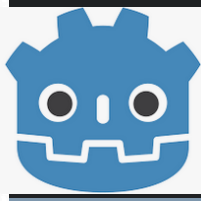
UNIVALI

Programando o jogo (Desvios)

- Com as teclas mapeadas podemos voltar ao `script` e fazer nosso primeiro desvio condicional

```
public override void _Process(double delta)
{
    if(Input.IsActionPressed("cima")) {
        GlobalPosition = GlobalPosition + new Vector2(0,-1);
    }
}
```

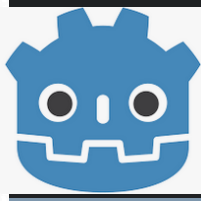
  - `Input.IsActionPressed("cima")` é um comando que retorna verdadeiro quando a tecla que mapeamos com o nome "`cima`" for pressionada
  - Sendo assim, se o usuário pressionar a tecla `W` que mapeamos para "`cima`" ele irá entrar na condição e ajustar a posição global do objeto uma posição para cima



UNIVALI

Programando o jogo (Desvios)

- **Exercícios**
  - Implemente desvios para todas as teclas mapeadas.
  - Teste seu jogo.



UNIVALI