Godot: plataformaPixel: Interações & Extras

Prof. Thiago Felski Pereira, MSc.

Visão Geral

- Nessa continuação iremos melhorar interaja deforma ativa com o mundo.
  - Coletaremos itens no mapa.
  - Ativaremos itens no mapa.
  - Teremos perigos estáticos e móveis.

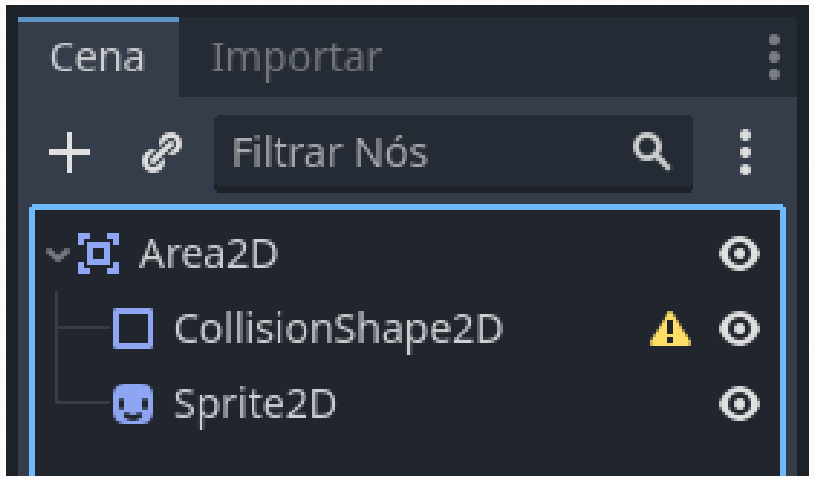
- Bibliografia
  - https://www.kenney.nl/assets/pixel-platformer

UNIVALI

Cena: Espinho

- Vamos criar alguns espinhos que poderão machucar o personagem.
  - Tornando assim nosso jogo mais desafiador.

- Primeiro crie a seguinte estrutura.
  - Area2D é um tipo de nó que permitirá detectar quando alguém colidiu com ele, iremos usar muito esse bloco.

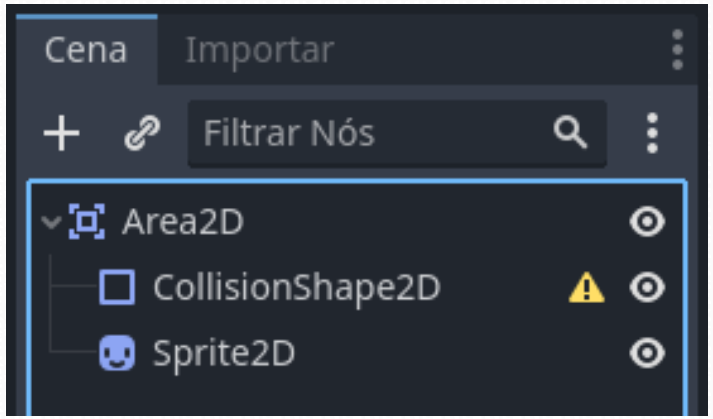


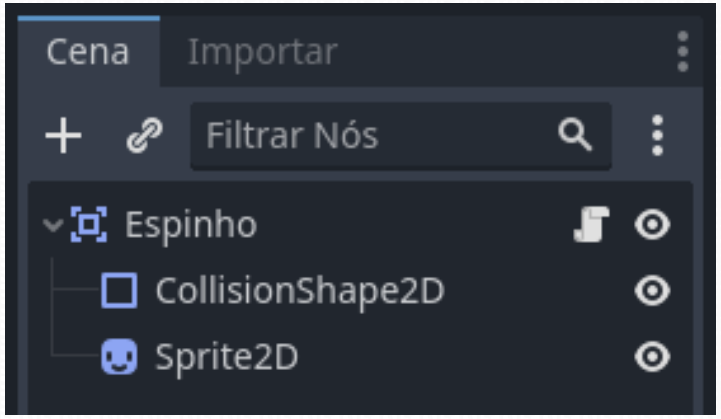
UNIVALI

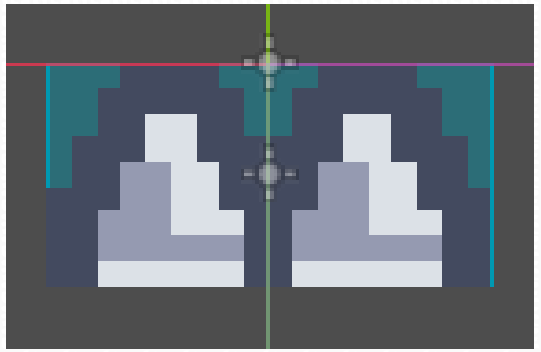
Cena: Espinho

- Vamos criar alguns espinhos que poderão machucar o personagem.
  - Tornando assim nosso jogo mais desafiador.

- Primeiro crie a seguinte estrutura.
  - Area2D é um tipo de nó que permitirá detectar quando alguém colidiu com ele, iremos usar muito esse bloco.
  - Renomeie a Area2D para Espinho e adicione um script a ela.
  - Além disso coloque a imagem de um espinho e defina sua área de colisão.



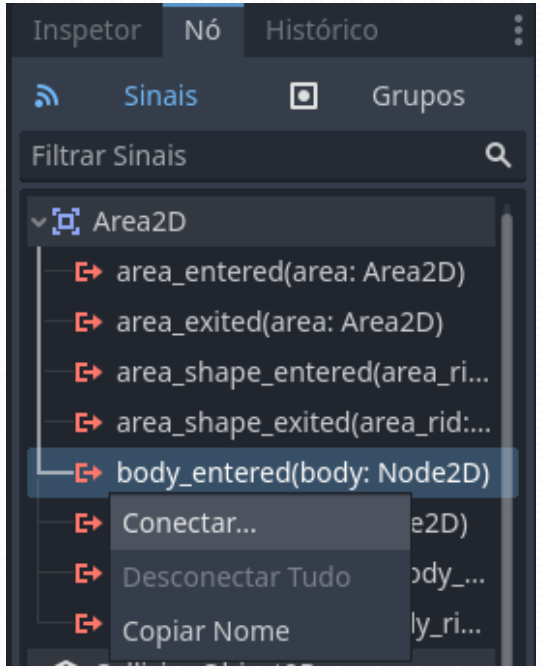


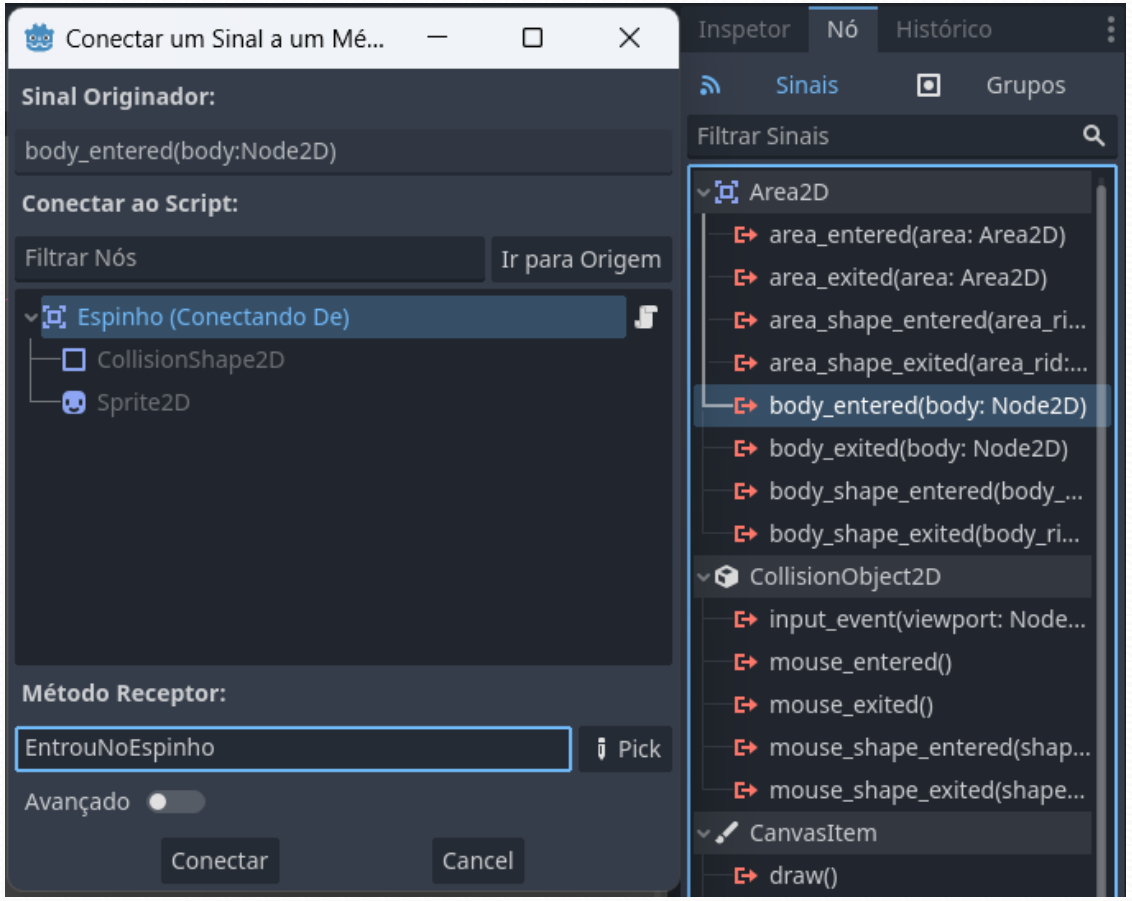


Cena: Espinho

- Vamos criar alguns espinhos que poderão machucar o personagem.
  - Tornando assim nosso jogo mais desafiador.

- Criando um **sinal** de alerta para colisão.
  - Com a cena principal selecionada ...





UNIVALI

Cena: Espinho

- Vamos criar alguns espinhos que poderão machucar o personagem.
  - Tornando assim nosso jogo mais desafiador.

- Criando um script no Espinho para tratar a colisão.
  - O script irá detectar se quem colidiu com o Espinho foi o jogador, nesse caso iremos chamar a função espetado no próprio script do jogador

```
public void EntrouNoEspinho(Node2D body) {
    if (body is Jogador) {
        ((Jogador)body).Espetado();

    }
}
```

UNIVALI

Cena: Jogador

- Vamos criar alguns espinhos que poderão machucar o personagem.
  - Tornando assim nosso jogo mais desafiador.

- No script do Jogador iremos criar a função Espetado
  - Essa função irá retornar o jogador ao ponto de salvamento e retirar uma vida do jogador.
  - Precisaremos descobrir como criar esses pontos de salvamento e como definir e trabalhar com a vida.

```
6 references
private int vida;
8 references
private Vector2 pontoDeSalvamento;
```

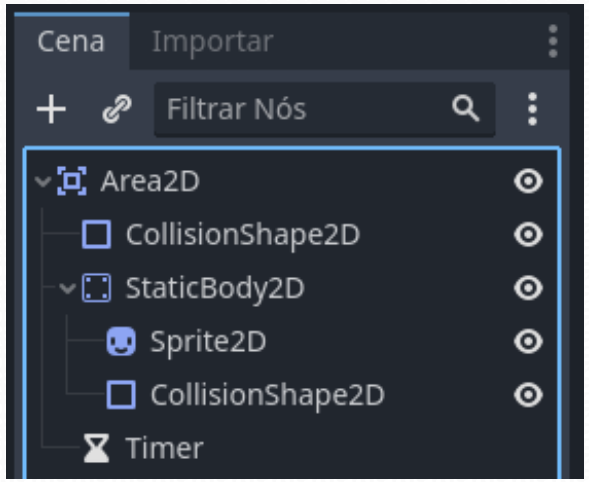
```
public void Espetado() {
    GlobalPosition = pontoDeSalvamento;
    vida = vida - 1;
    GD.Print("Retornou ao ponto salvo");
}
```

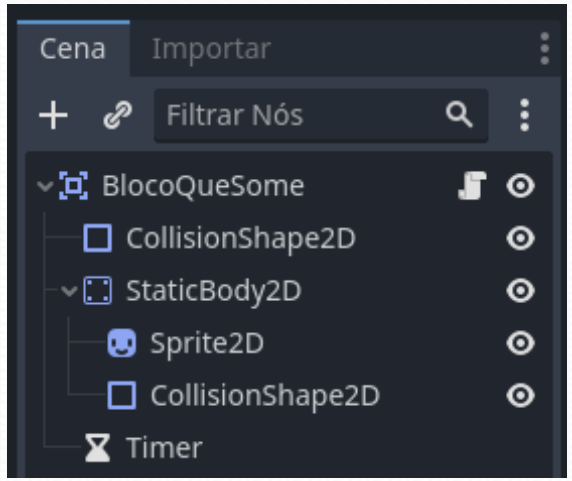
UNIVALI

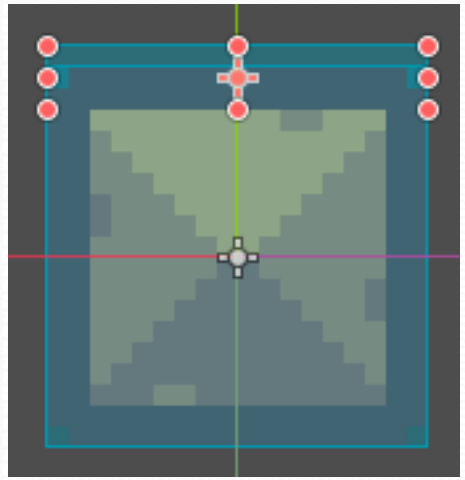
Cena: BlocoQueSome

- Vamos criar alguns bloco que somem com o tempo.

- Primeiro crie a seguinte estrutura.
  - Renomeie a Area2D para BlocoQueSome e adicione um script a ela.
  - Note que a foi colocada uma área de colisão para detectarmos que o jogador entrou nela.
  - Também foi incluído um timer para definir o tempo que o bloco existirá depois de pisado.



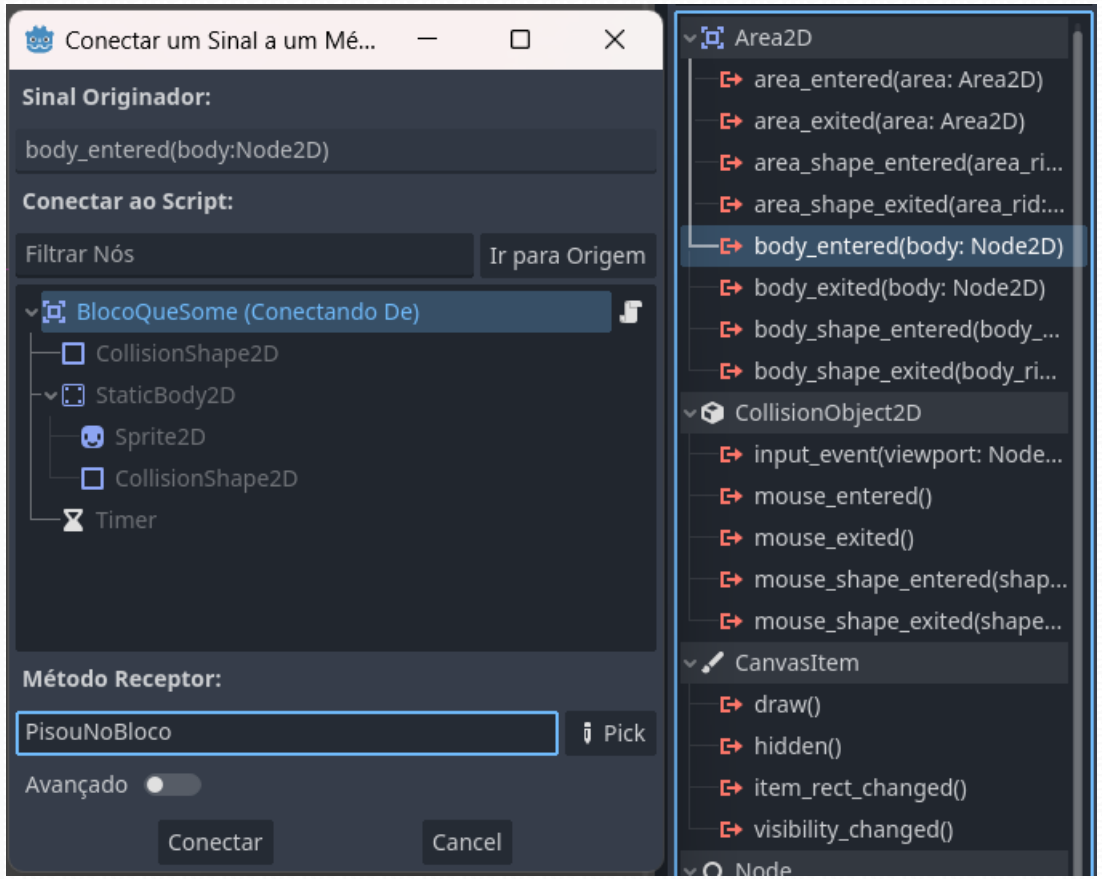




UNIVALI

Cena: BlocoQueSome

- Vamos preparar para detectar a colisão desse bloco com o Jogador.



UNIVALI

Cena: BlocoQueSome

- Vamos programar o script do BlocoQueSome

```
public partial class BlocoQueSome : Area2D {
    7 references
    private Timer tempo;
    6 references
    private bool pisou=false;
    4 references
    public override void _Ready() {
        //tempo recebe o nó Timer
        tempo = GetNode<Timer>("Timer");
    }
    3 references
    public override void _Process(double delta) {
        if (tempo.TimeLeft < 0.1f && pisou) {
            //Remove o bloco do jogo
            QueueFree();
        }
    }
```

```
    public void PisouNoBloco(Node2D body) {
        if (body is Jogador) {
            //Inicia uma Contagem Regressiva de 5 segundos
            //E indica que já pisou
            tempo.Start(5);
            pisou=true;
        }
    }
} //Fim da classe BlocoQueSome
```

UNIVALI

Cena: BlocoQueSome Avançado

- A única diferença para o BlocoQueSome é que nosso script fará o bloco reaparecer após algum tempo.

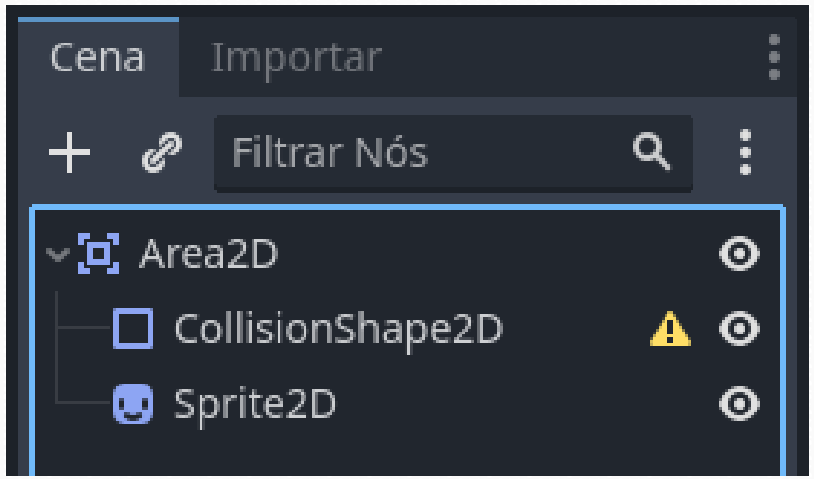
```
public partial class BlocoQueSome : Area2D {
    private StaticBody2D bloco;
    private Timer tempo;
    private bool pisou=false;
    private bool sumiu=false;

    public override void _Ready() {
        //tempo recebe o nó Timer
        tempo = GetNode<Timer>("Timer");
        bloco = GetNode<StaticBody2D>("StaticBody2D");
    }
    public override void _Process(double delta) {
        if (tempo.TimeLeft < 0.1f && pisou) {
            pisou=false;
            sumiu=true;
            RemoveChild(bloco); //Retira o bloco do jogo
            tempo.Start(5); //Tempo para o bloco reaparecer
        } else if (tempo.TimeLeft < 0.1f && sumiu) {
            sumiu=false;
            AddChild(bloco); //recoloca o bloco no jogo
        }
    }
```

```
    public void PisouNoBloco(Node2D body) {
        if (body is Jogador) {
            //Inicia uma Contagem Regressiva de 5 segundos
            //E indica que já pisou
            tempo.Start(5);
            pisou=true;
        }
    }
} //Fim da classe BlocoQueSome
```

Cena: Espinho Móvel

- Vamos criar alguns espinhos móveis que nada mais são do que espinhos um pouco mais avançados.
  - Via de regra eles são como os espinhos normais, mas irão se mover.

- Primeiro crie a seguinte estrutura.
  - Area2D é um tipo de nó que permitirá detectar quando alguém colidiu com ele, iremos usar muito esse bloco.

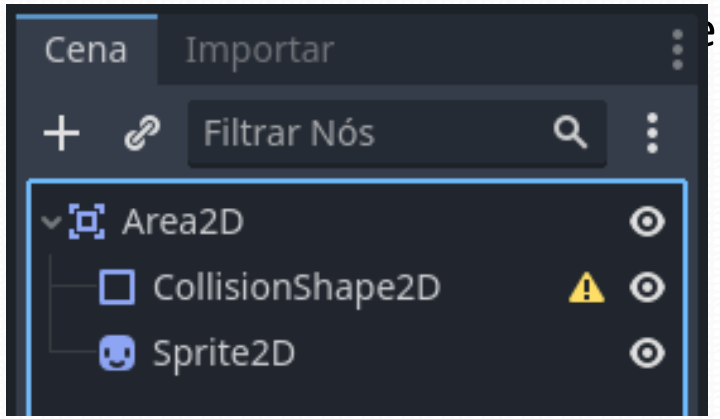


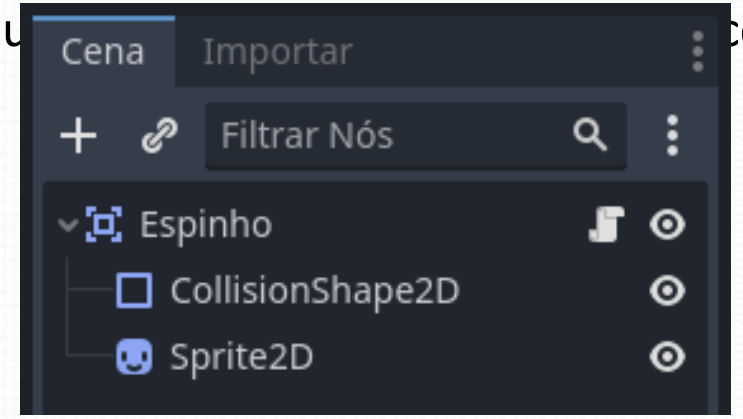
UNIVALI

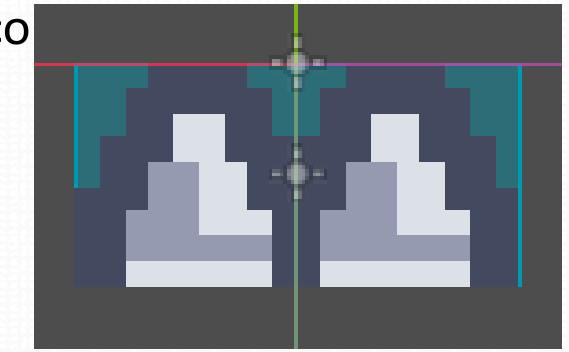
Cena: Espinho Móvel

- Vamos criar alguns espinhos móveis que nada mais são do que espinhos um pouco mais avançados.
  - Via de regra eles são como os espinhos normais, mas irão se mover.

- Primeiro crie a seguinte estrutura.
  - Area2D é um tipo de nó que permitirá detectar quando alguém colidiu com ele, iremos usar muito esse bloco.
  - Renomeie a Area2D para Espinho e adicione um script a ela.
  -



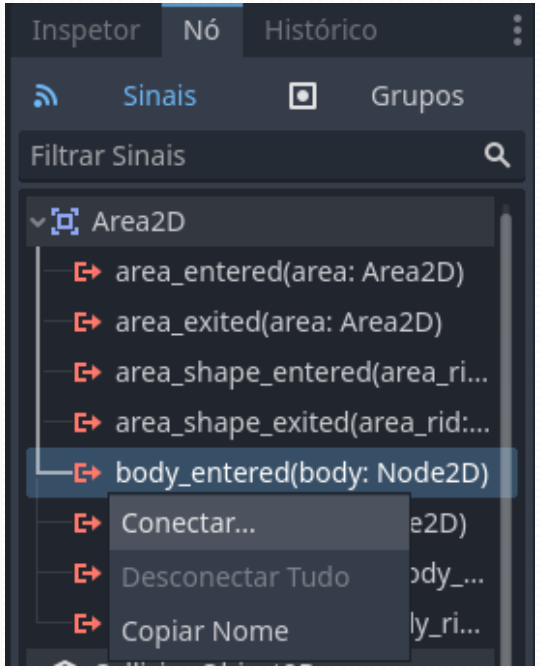


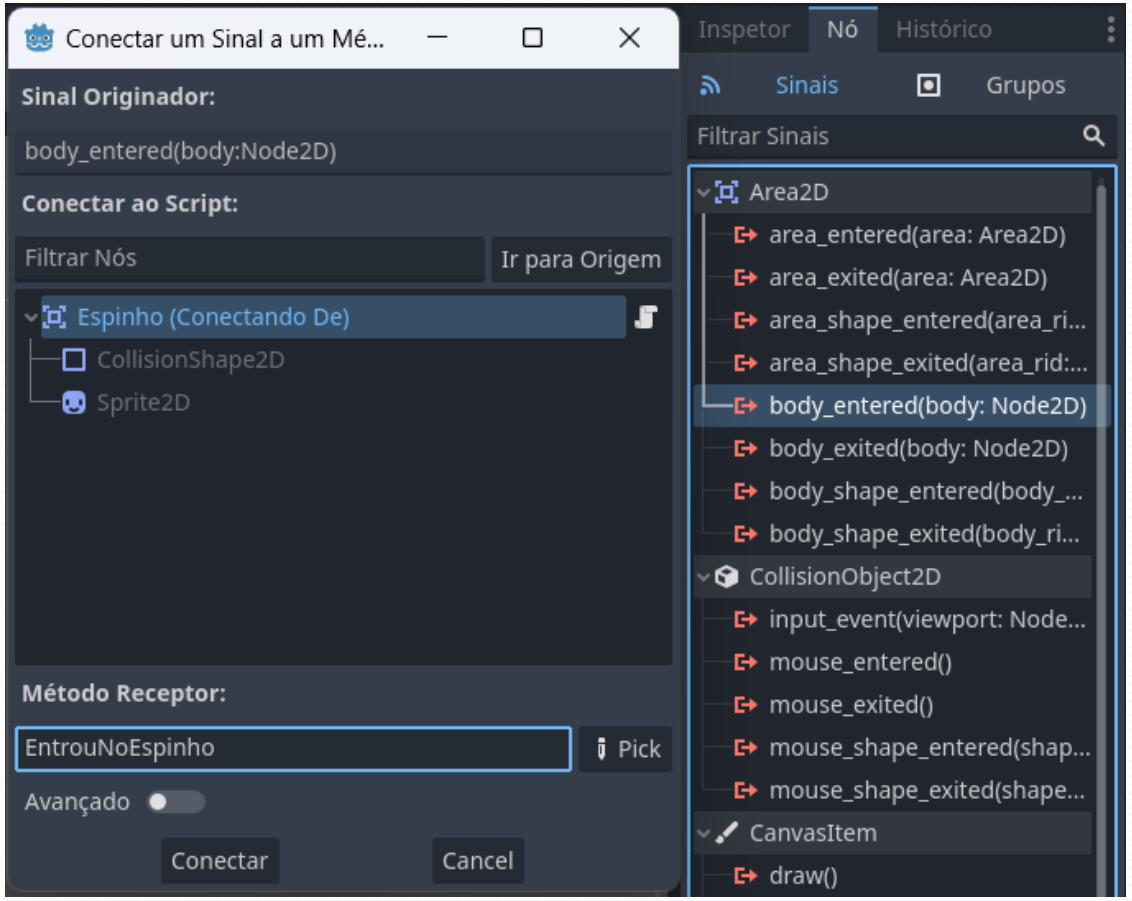


UNIVALI

Cena: Espinho Móvel

- Criando um **sinal** de alerta para colisão.
  - Com a cena principal selecionada ...





UNIVALI

Cena: Espinho Móvel

- Script do espinho móvel.
  - A diretiva `[Export]` foi utilizada para podermos mudar os valores na interface.

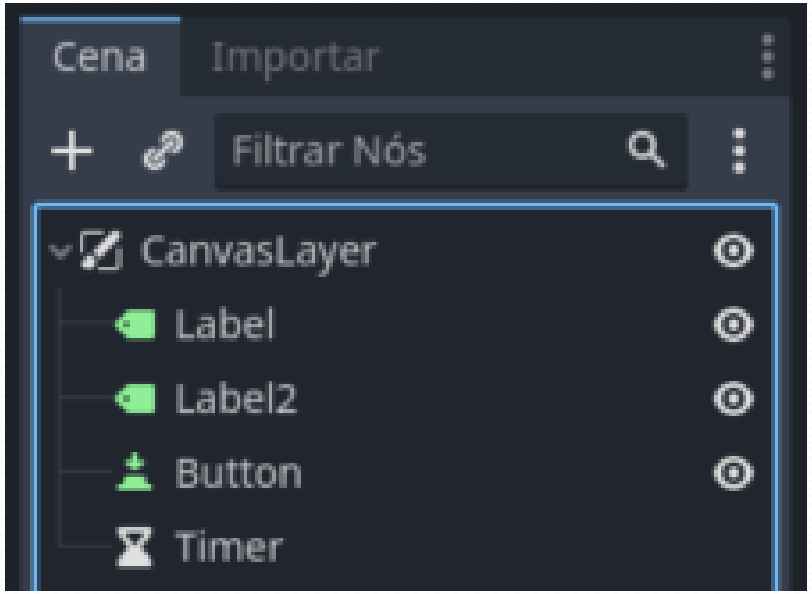
```
public partial class EspinhoMovel : Area2D {
    7 references
    private int contador=0;
    5 references
    [Export]private bool direcao = true;
    5 references
    [Export]private bool progressivo = true;
    5 references
    [Export]private int distanciaMaxima=40;
    9 references
    private int p;

    5 references
    public override void _Ready() {
        if (progressivo) {
            p=1;
        } else {
            p=-1;
        }
    }
```

```
    public override void _Process(double delta) {
        if (direcao) {
            GlobalPosition += new Vector2(p,0);
        } else {
            GlobalPosition += new Vector2(0,p);
        }
        if(contador >= distanciaMaxima) {
            contador=0;
            p *= - 1;
        }
        contador++;
    }
    1 reference
    public void EntrouNoEspinho(Node2D body) {
        if (body is Jogador) {
            ((Jogador)body).Espetado();
        }
    }
}
```

Cena: HUD

- Para criar nossa interface de usuário nós precisaremos seguir alguns passos.

- Primeiro crie a seguinte estrutura.
- Em seguida renomeie de forma a entender o que significa cada coisa.

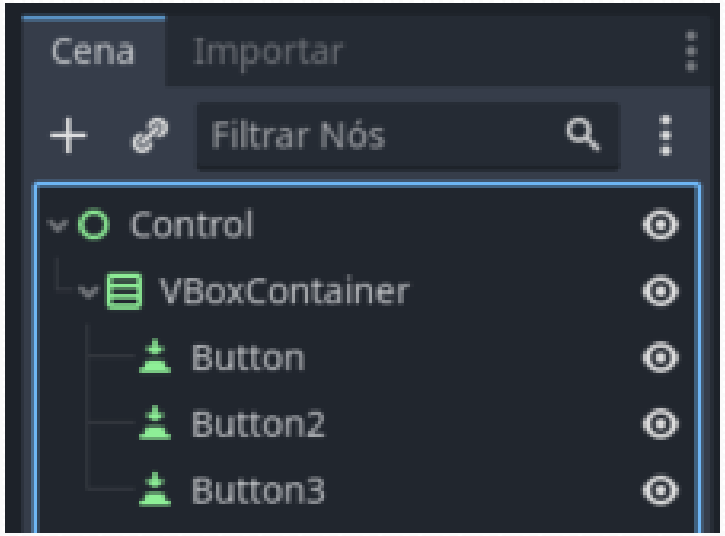


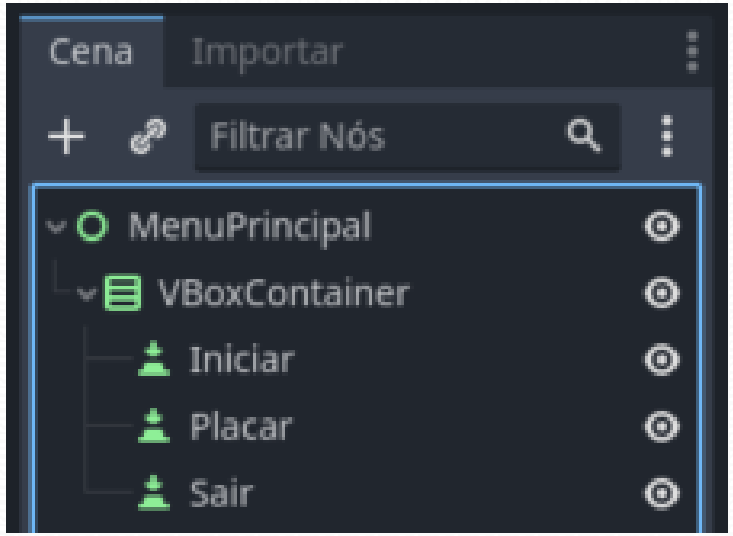
UNIVALI

Cena: Menu Principal

- Vamos criar nosso menu principal.

- Primeiro crie a seguinte estrutura igual a da imagem na esquerda.
- Em seguida, renomeie de forma a ficar igual a imagem da direita.

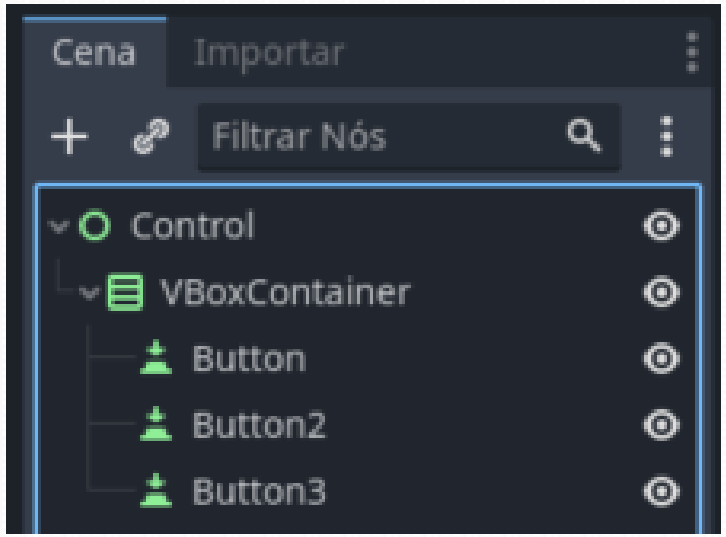


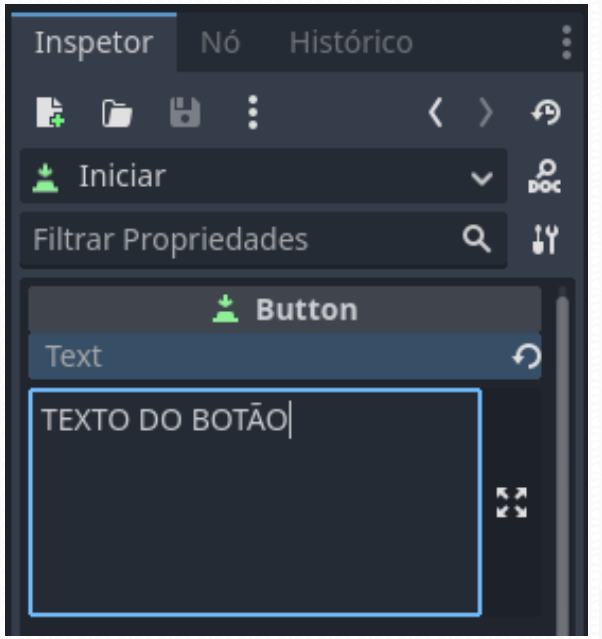


UNIVALI

Cena: Menu Principal

- Renomeando os textos dos botões
  - Imagem da esquerda: estrutura inicial.
  - Imagem central: renomeando os objetos.
  - Imagens da direita: ajustando o nome dos botões





UNIVALI

Cena: Menu Principal

- Passos para adicionar funcionalidade aos botões.
  - Adicione um script ao menu principal.
  - Adicione um sinal de botão pressionado para cada botão e ligue ao menu principal.
  - Escreva os códigos para as funções chamadas pelo sinal.

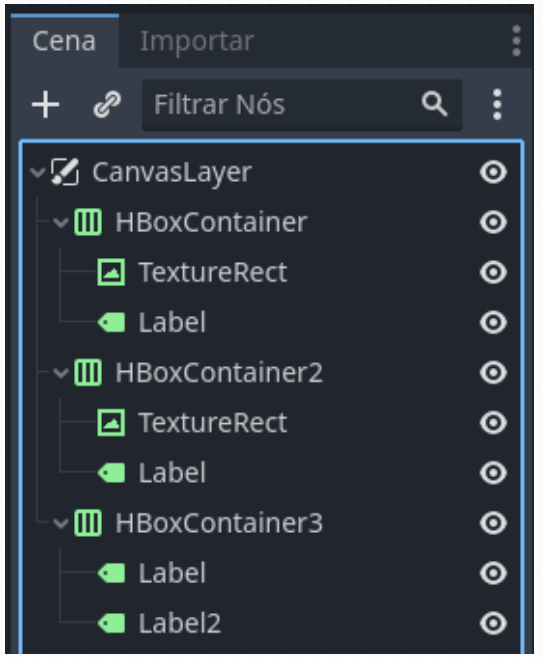
```
public partial class MenuPrincipal : Control
{
    1 reference
    public void IniciarPressionado() {
        GetTree().ChangeSceneToFile("res://mapa_1.tscn");
    }
    1 reference
    public void PlacarPressionado() {
        //Editaremos após registrarmos os melhores tempos
    }
    1 reference
    public void SairPressionado() {
        GetTree().Quit();
    }
}
```

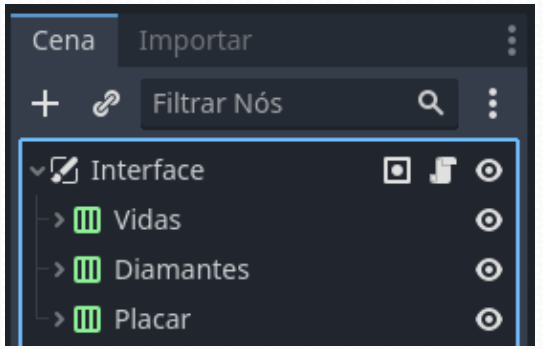
UNIVALI

Cena: Interface de usuário

- Essa interface servirá para dar informações úteis sobre o jogo como:
  - Vidas restantes.
  - Coletáveis pegos.
  - Tempo de jogo.
  - Etc.

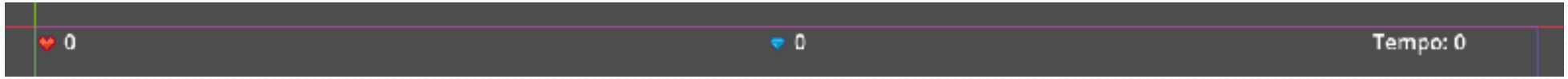
- Para isso iremos utilizar o nó `CanvasLayer` que nos permite desenhar elementos de interface em uma camada acima do resto do jogo, de forma que as informações mostradas não fiquem cobertas por quaisquer elementos do jogo, como o jogador ou os inimigos.





UNIVALI

Cena: Interface de usuário



- Organize sua interface para ficar com a aparência da imagem acima.
- No código da interface iremos substituir o número de corações e os diamantes pelos valores fornecidos no jogo.
- Devemos desabilitar o cronometro quando o mapa terminar.
  - Variáveis da Interface

```
public partial class HUD : CanvasLayer {
    7 references
    private float cronometro;
    7 references
    private bool cronometroAtivo=false;
    9 references | 9 references | 6 references
    private int min, seg, mil;
    6 references | 7 references | 7 references
    private string cronometroFormatado,segundoFormatado, minutoFormatado;
    6 references
    public override void _Ready() {
        cronometro = 0;
        cronometroAtivo = true;
    }
```


UNIVALI

Cena: Interface de usuário

- Funções para iniciar, controlar e imprimir a passagem de tempo.

```
public override void _Ready() {
    cronometro = 0;
    cronometroAtivo = true;
}
5 references
public override void _Process(double delta) {
    if(cronometroAtivo) {
        cronometro +=(float)delta;
        AtualizaPlacar(cronometro);
    }
}
```

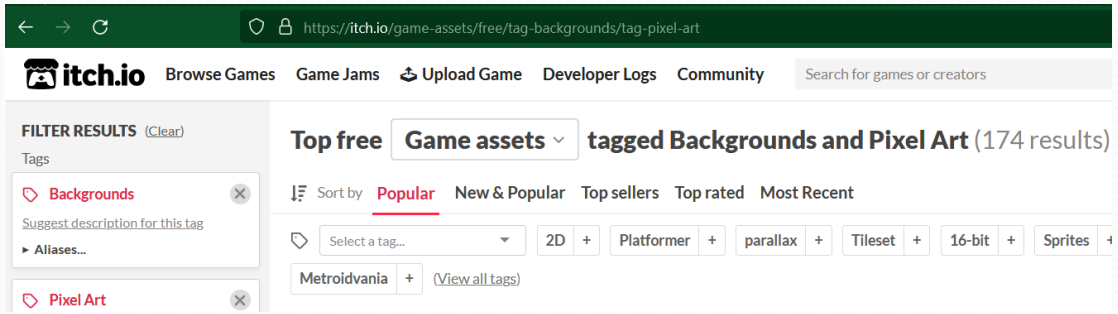
```
public void AtualizaPlacar (float placar) {
    seg = (int)placar%60;
    min = (int)placar/60;
    mil = (int)(100*(placar - min - seg));
    if (min<10){
        minutoFormatado="0"+min;
    } else {
        minutoFormatado=""+min;
    }
    if (seg<10){
        segundoFormatado="0"+seg;
    } else {
        segundoFormatado=""+seg;
    }
    cronometroFormatado = minutoFormatado + ":" + segundoFormatado + ":" + mil;
    GetNode<Label>("Placar/TextoPlacar").Text = cronometroFormatado;
}
```


UNIVALI

Pano de Fundo

- Existem duas maneiras de criar um pano de fundo para o seu jogo.
  - A primeira é colocar uma imagem estática que cubra todo cenário do jogo como fundo de cena.
  - A segunda é colocar uma imagem menor centralizada na câmera do jogador.

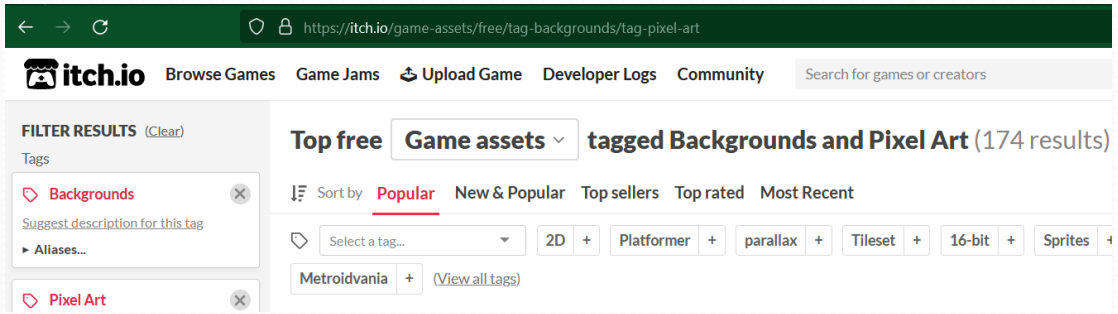
- De qualquer forma o primeiro passo é selecionar uma imagem para ser seu pano de fundo.
  - Eu recomendo pegar imagens que são garantidamente gratuitas, por isso evite busca-las no google.
  - Eu utilizo o `itch.io`, mas pode buscar em qualquer fonte que desejar.



UNIVALI

Pano de Fundo

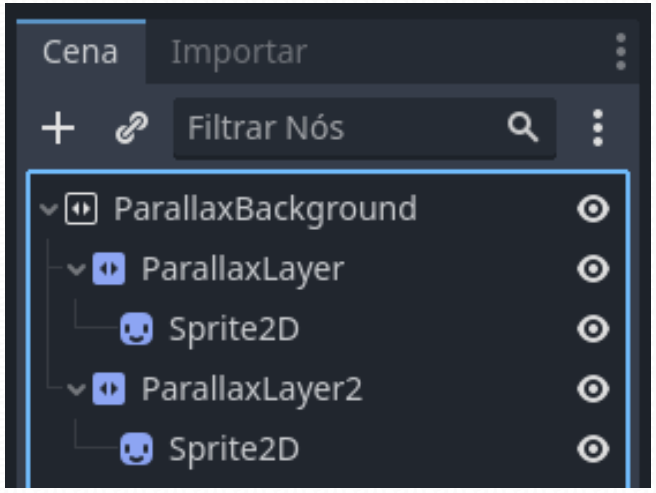
- Existem várias maneiras de criar um pano de fundo para o seu jogo.
  - Colocar uma imagem estática que cubra todo cenário do jogo como fundo de cena.
  - Colocar uma imagem menor centralizada na câmera do jogador.
  - Colocar imagens que se encaixam dando a ilusão de continuidade.
- De qualquer forma o primeiro passo é selecionar uma imagem para ser seu pano de fundo.
  - Eu recomendo pegar imagens que são garantidamente gratuitas, por isso evite busca-las no google.
  - Eu utilizo o `itch.io`, mas pode buscar em qualquer fonte que desejar.

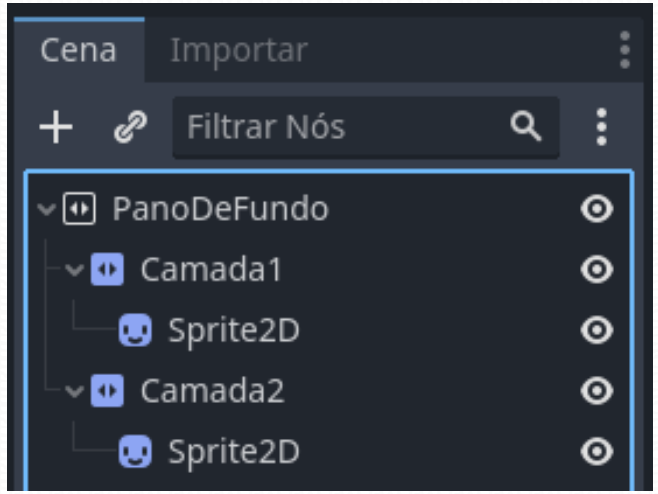


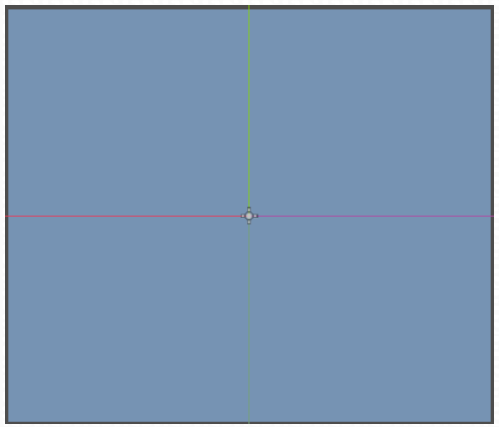
UNIVALI

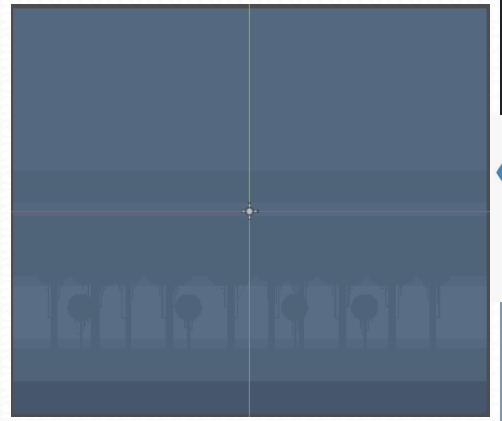
Pano de Fundo

- Nesse exemplo será utilizado um mapa de pixel arte com várias camadas.
  - Free Pixel Art Florest
- Para esse exemplo iremos utilizar uma estrutura de apenas duas camadas (Layers), mas você pode utilizar quantas camadas desejar.
  - Lembre-se que as camadas são desenhadas de cima para baixo, portanto quanto mais em cima na hierarquia, mais no fundo a imagem ficará.





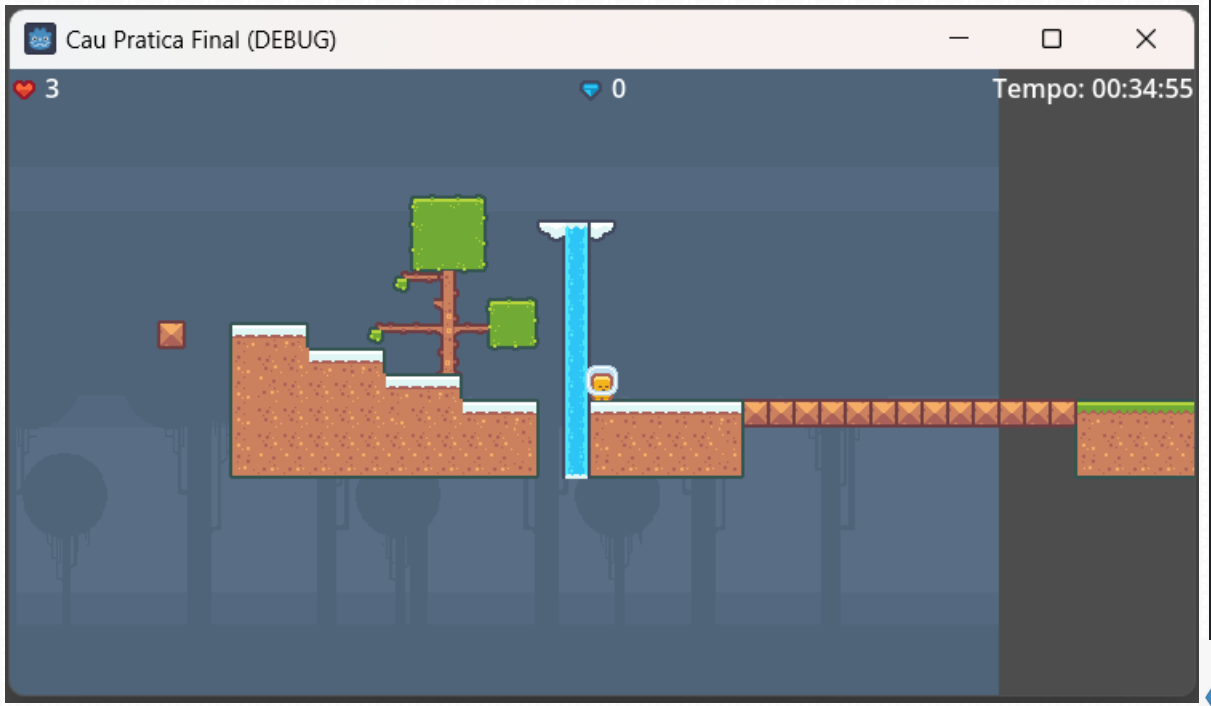




UNIVALI

Pano de Fundo

- Se simplesmente colocarmos a imagem assim ela vai ficar pequena e incompleta.
- Além disso se nos movermos as duas camadas ficam estáticas no fundo e esse não é o efeito desejado.
- Para ajustar o problema de fim de imagem iremos utilizar o espelhamento.

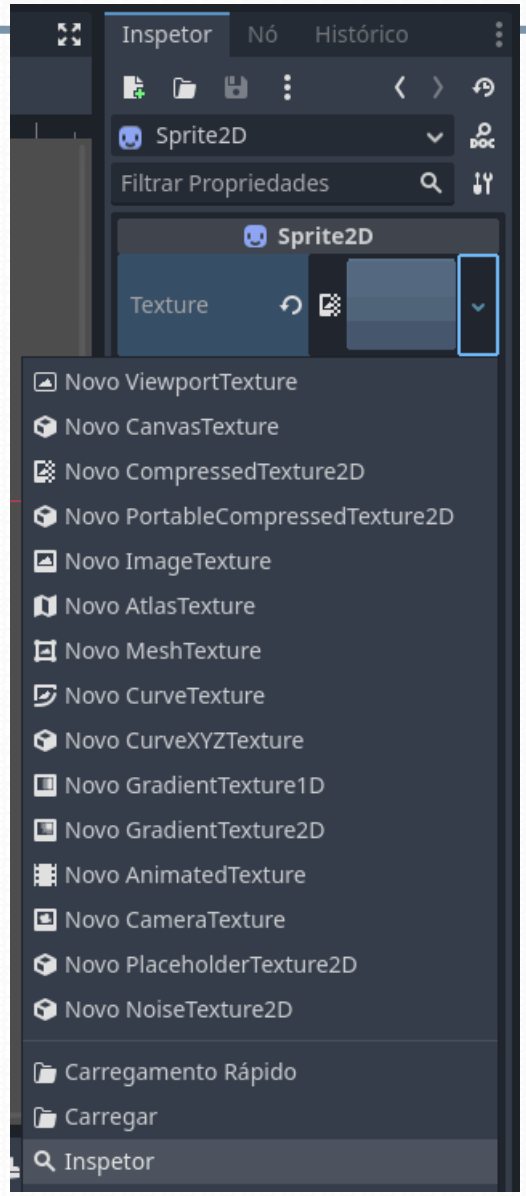


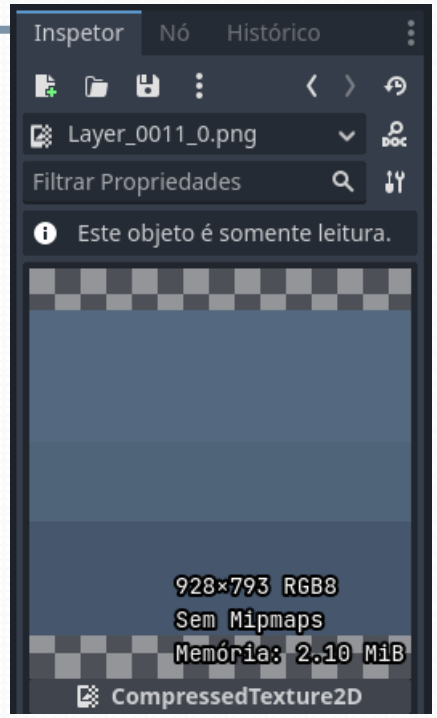
UNIVALI

Pano de Fundo

- Mesmo ajustando a escala para 2 ainda encontramos o final do mapa em algum momento.

- Ajustando o espelhamento das Camadas.
  - Para saber o tamanho das imagens, podemos verificar suas propriedades no inspetor
  - Como a imagem é de 928x793 basta colocar esses valores na propriedade de espelhamento que deseja espelhar.

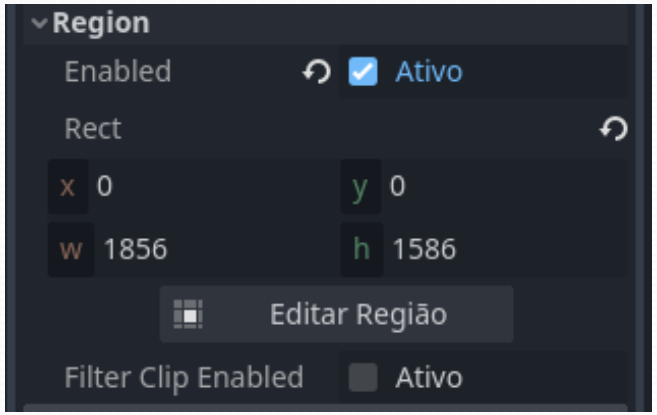


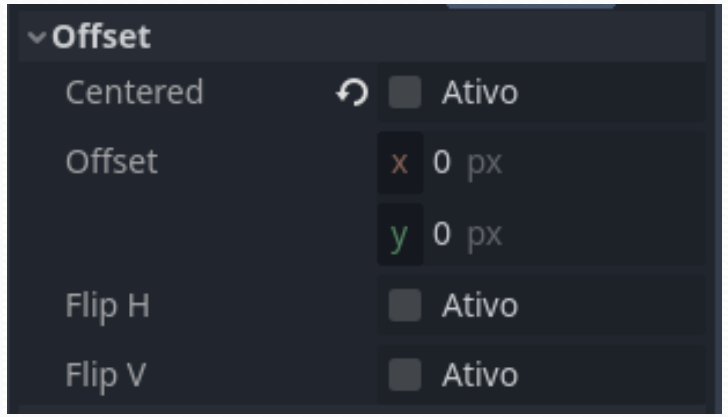


UNIVALI

Pano de Fundo

- Como a imagem selecionada é menor que a área do jogo, foi necessário adaptar o a região visível para que não ficasse visível o espelhamento.
  - A região foi definida como o dobro do tamanho original da imagem.
- Foi desmarcado o Offset de Centered para facilitar o posicionamento do pano de fundo.
- De fora para dentro, foi marcado um valor menor para o Motion Scale X em cada camada.
  - Isso faz com que as camadas de trás se movam mais rápido do que as da frente dando fluidez ao pano de fundo

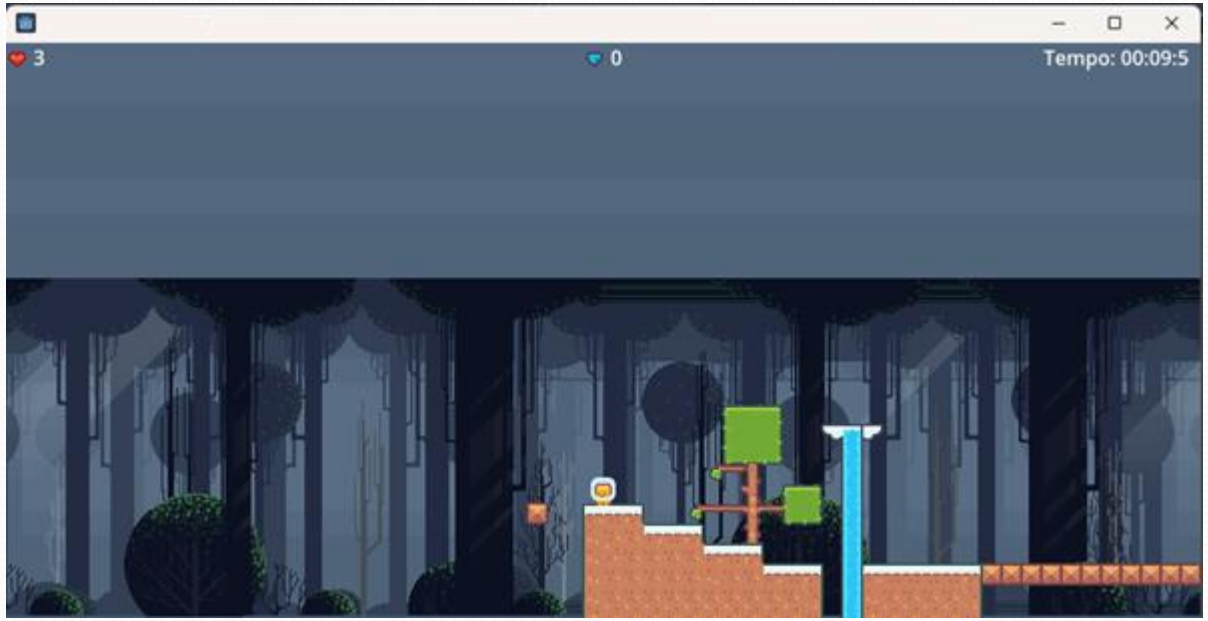




UNIVALI

Pano de Fundo

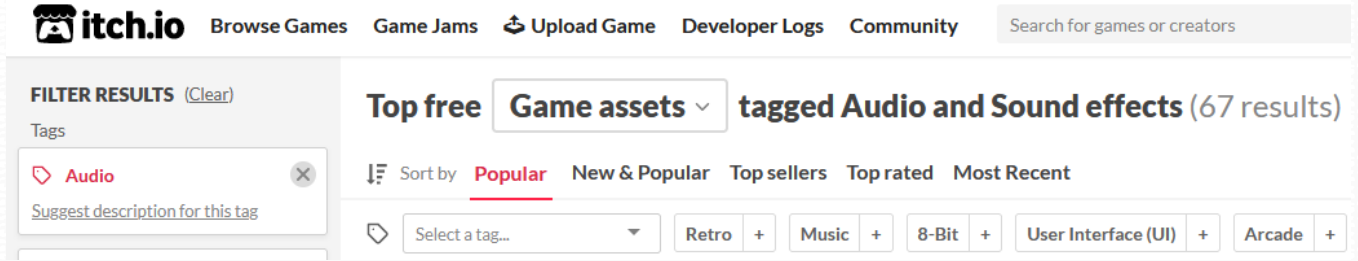
- Esse é o resultado final do pano de fundo com 6 camadas.



UNIVALI

Som

- Som é um efeito especial presente em quase todos os jogos.
  - Esse, é outro recurso especialmente difícil para programadores.
  - Por isso, nesse momento de aprendizado, iremos buscar conteúdos gratuitos na internet.



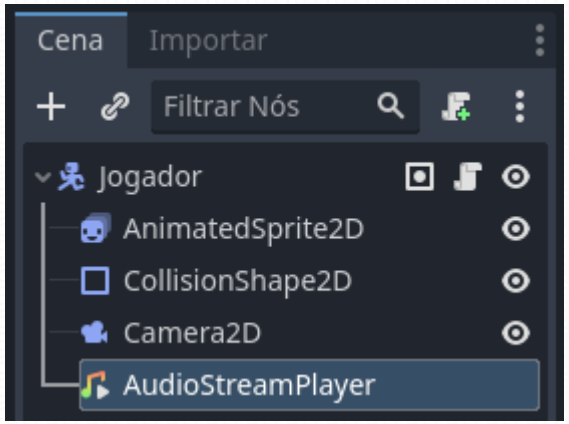
- Reproduzir áudios no jogo é um processo muito similar ao utilizado para ativar as animações no jogador.

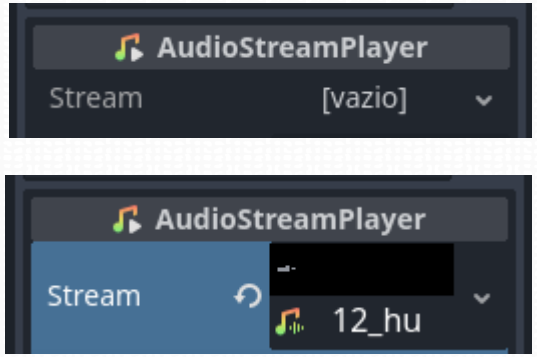
- Para testar serão incluídos alguns efeitos sonoros a cena do jogador.

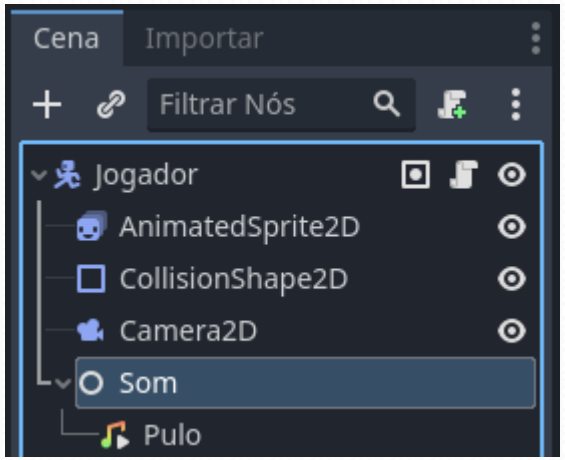
UNIVALI

Som

- O primeiro passo é incluir um `AudioStreamPlayer` ao Jogador.
  - Existe também o `AudioStreamPlayer2D` e o `AudioStreamPlayer3D` esses áudio permitem projetar o som dentro do eixo 2D ou 3D do jogo. (Não usaremos esses)
- Segundo passo é arrastar um áudio para o stream [vazio]
- Se seu plano é definir vários efeitos sonoros, recomendo agrupar todos em um nodo.







UNIVALI

Som

- Programando os sons.
  - Cria-se uma variável privada na classe Jogador:

```
private AudioStreamPlayer somDePulo;
```

  - Atribui a variável criada o nó do som correspondente no método `_Ready()`

```
somDePulo = GetNode<AudioStreamPlayer>("Som/Pulo");
```

  - Escolhe-se o momento em que se quer reproduzir o áudio. (Quando o jogador pular)

```
if (Input.IsActionJustPressed("ui_accept") && IsOnFloor()) {
    velocity.Y = JumpVelocity;
    somDePulo.Play();
}
```

  - Pronto, nosso jogador já produz som quando pula.

UNIVALI

Som

- O som ambiente é muito mais fácil de incluir, pois nós queremos ele tocando sempre e não precisamos colocar condições.
  - Para incluir a música de fundo, basta incluir o `AudioStreamPlayer`, colocar a música desejada e ativar o `AutoPlay`.

UNIVALI

Interagir com uma área

- Algumas ações só devem estar habilitadas em determinadas situações. Por exemplo:
  - O jogador só deverá abrir uma porta se estiver na frente da porta quando clicar no botão de ação.
  - O pulo será ampliado se o jogador pular de cima de um trampolim.
  - O jogador precisará estar em frente a alavanca para poder puxá-la.

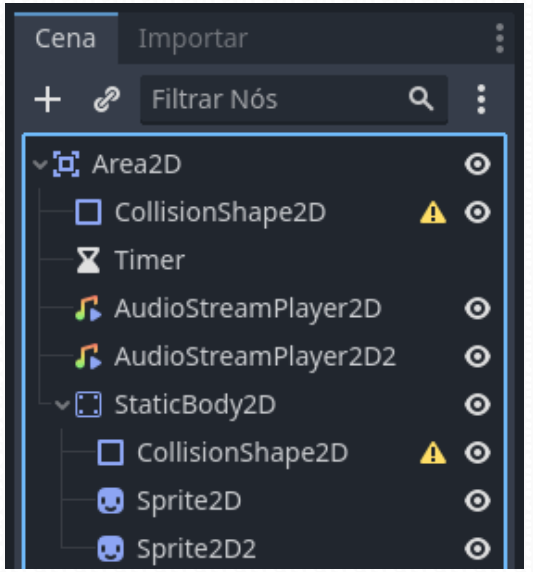
- Não temos um sinal que fique nos indicando quando o tempo todo que o jogador estiver dentro de uma área, mas podemos fazer isso ativando ações quando o jogador entrar na área e desativando-as quando o jogador sair da área.

- Para testar esse conceito será implementado um trampolim que se o jogador pular quando estiver em cima dele o pulo será muito maior.

UNIVALI

Trampolim: super pulo

- Se o jogador estiver em um trampolim, seu pulo será muito mais forte.
- Definindo as regras da cena.
  - O jogador poderá disparar o trampolim se estiver, em cima dele e apertar o botão de pulo.
  - Uma vez disparado o trampolim fica desativado por um tempo e a imagem troca para indicar isso.
  - Passado o tempo o trampolim fica habilitado para um novo pulo.
  - Um som deve ser tocado quando o trampolim ativar e outro som deve ser tocado para indicar que está armado novamente.



UNIVALI

Script Trampolim

```
public partial class Trampolim : Area2D {
    8 references
    private bool ativo = true;
    6 references | 6 references
    private AudioStreamPlayer2D disparo, recarga;
    7 references | 7 references
    private Sprite2D imagemAtivado, imagemDesativado;
    7 references
    private Timer tempo;
    7 references
    public override void _Ready() {
        disparo = GetNode<AudioStreamPlayer2D>("SomDisparo");
        recarga = GetNode<AudioStreamPlayer2D>("SomRecarga");
        imagemAtivado = GetNode<Sprite2D>("BlocoEstatico/ImagemAtivado");
        imagemDesativado = GetNode<Sprite2D>("BlocoEstatico/ImagemDesativado");
        tempo = GetNode<Timer>("Timer");
    }
    6 references
    public override void _Process(double delta) {
        if (tempo.TimeLeft < 0.1 && !ativo) {
            ativo = true;
            recarga.Play();
            imagemAtivado.Show();
            imagemDesativado.Hide();
        }
    }
    public void EntrouNoTrampolim(Node2D body) {
        if (body is Jogador && ativo) {
            ((Jogador)body).Trampolim(true);
            GD.Print("Entrou no Trampolim");
        }
    }

    1 reference
    public void SaiuDoTrampolim(Node2D body) {
        if (body is Jogador) {
            ((Jogador)body).Trampolim(false);
            GD.Print("Saiu do Trampolim");
        }
    }

    1 reference
    public void DisparoTrampolim() {
        ativo=false;
        GD.Print("Yahooo!");
        disparo.Play();
        imagemAtivado.Hide();
        imagemDesativado.Show();
        tempo.Start(5);
    }
}
```

UNIVALI

Script Jogador: interação com Trampolim

- Variável para detectar se está no trampolim.

```
private bool noTrampolim=false;
```

- Super pulo se tiver no trampolim.

```
if (Input.IsActionJustPressed("ui_accept") && IsOnFloor() && !noTrampolim) {
    velocity.Y = JumpVelocity;
    somDePulo.Play();
} else if (Input.IsActionJustPressed("ui_accept") && IsOnFloor() && noTrampolim) {
    velocity.Y = 2.5f*JumpVelocity;
    GetTree().CallGroup("GrupoTrampolim", "DisparoTrampolim");
}
```

- Função chamada pelo trampolim para atualizar a variável noTrampolim.

```
public void Trampolim (bool t) {
    noTrampolim = t;
    GD.Print("Trampolim = ", noTrampolim);
}
```

UNIVALI