

Aplicando animações no GODOT

Prof. Thiago Felski Pereira, MSc.

Visão Geral

- O **Pixelorama** foi utilizado para criar imagens e animações, mas agora precisamos incluí-las nos nossos jogos.



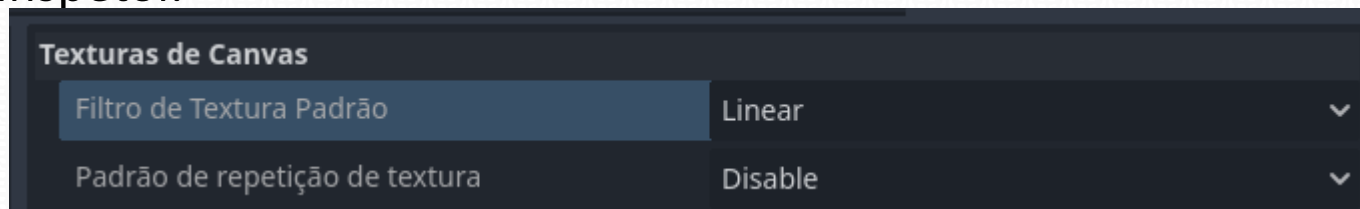
(2)



UNIVALI

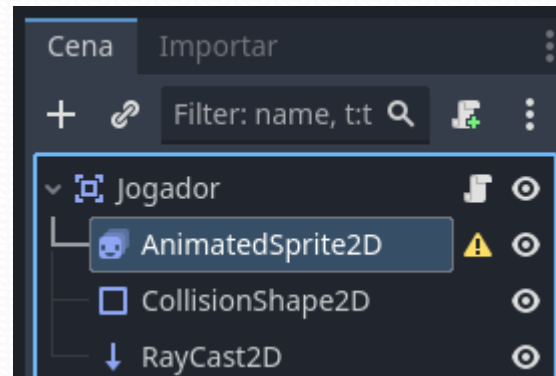
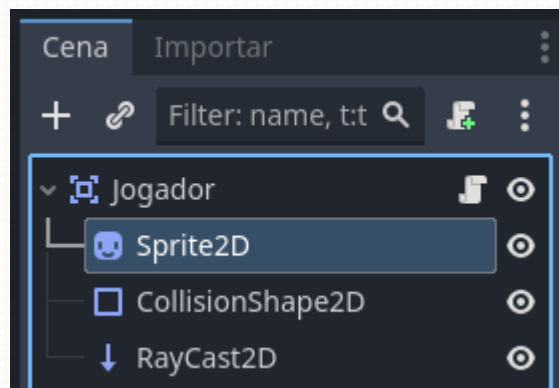
Ajustando a resolução

- As ferramentas tentem a possuir filtros para melhorar as imagens, mas em pixelart essas ferramentas acabam tendo o efeito reverso.
 - No GODOT, o renderizador tende a suavizar as bordas das imagens para não deixa-las quadradas.
 - Mas é justamente esse o efeito que queremos em uma pixelart.
- Para desabilitar o renderizador, precisamos ir em:
 - Menu -> Projeto -> Configurações do Projeto... -> Renderizando -> Texturas
 - E trocar o filtro de texturas padrão para Linear.
 - Com isso todas as texturas do Godot seguirão esse padrão e não precisaremos mudar uma a uma no Inspetor.



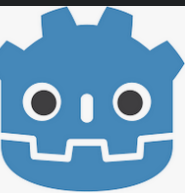
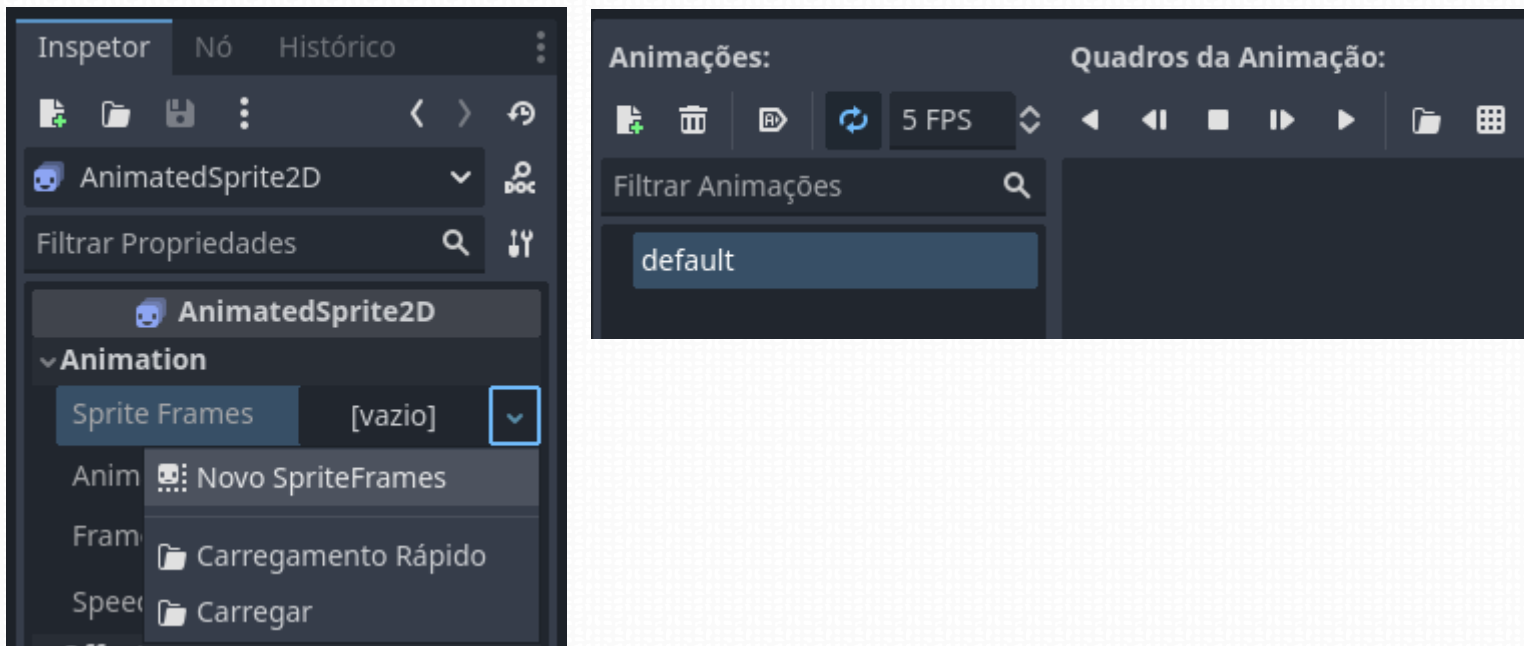
Quadros de animação

- A inclusão de quadros de animação é bem simples de ser feita no GODOT, pois já existe um objeto chamado `AnimatedSprite2D` que nos ajudará com isso.
- Para testar vamos modificar nosso projeto do Sokoban.
 - **Alerta**, se você ainda não apresentou a AVA para o professor, salve uma cópia do projeto por segurança.
- No Jogador, exclua o objeto `Sprite2D` e inclua um `AnimatedSprite2D` no lugar.






Quadros de animação

- No Inspetor, adicione um **Novo Sprite Frames** a Animação.
- Em seguida clique nele para abrir o menu de edição.



Quadros de animação

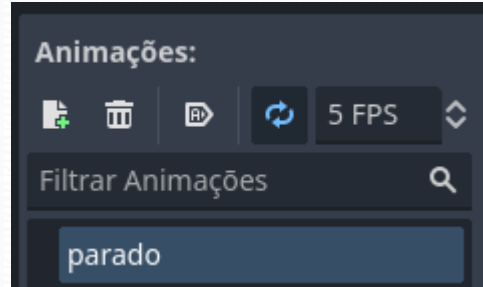
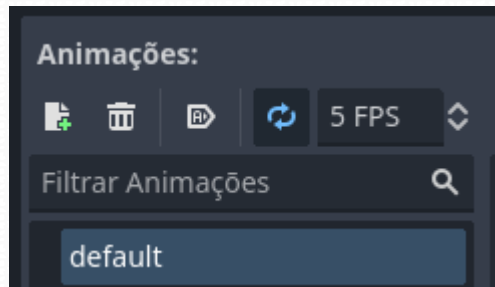
- Incluindo as imagens que criamos no **Pixelorama**.
- **Passo 1:** coloque as imagens no diretório do projeto.
 - Eu coloquei todas as minhas imagens em uma pasta chamada Slime.
 - Lembre-se de colocar as imagens descompactadas para conseguirmos utilizar.

 .godot	25/04/2024 09:08	Pasta de arquivos	
 Slime	25/04/2024 09:10	Pasta de arquivos	
 caixa.tscn	25/04/2024 09:08	Arquivo TSCN	1 KB



Quadros de animação

- Incluindo as imagens que criamos no **Pixelorama**.
- **Passo 2:** Ajuste o nome da animação e o FPS no Godot.
 - Como vou colocar as animações do Slime parado no Godot vou editar o nome de **default** para **parado**.



- Note que teremos que alinhar o FPS dos nossos frames para ficarem equivalentes ao que planejamos no **Pixelorama**.

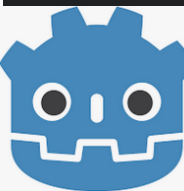
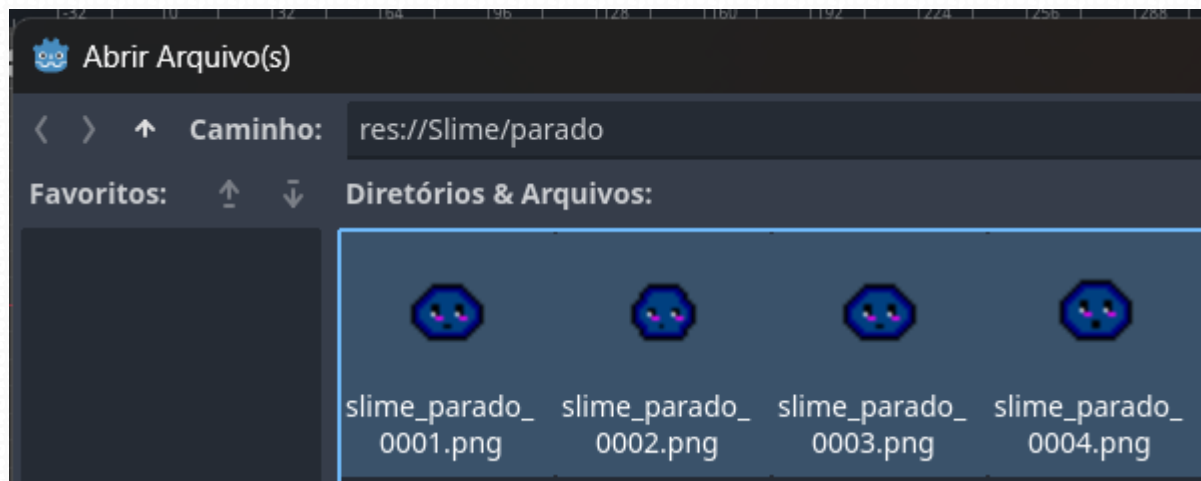


Quadros de animação

- Incluindo as imagens que criamos no **Pixelorama**.
- **Passo 3:** Adicione as imagens de parado.
 - Vamos adicionar as imagens (quadros) que criamos para nossa animação.
 - Clique para abrir os quadros de um arquivo.



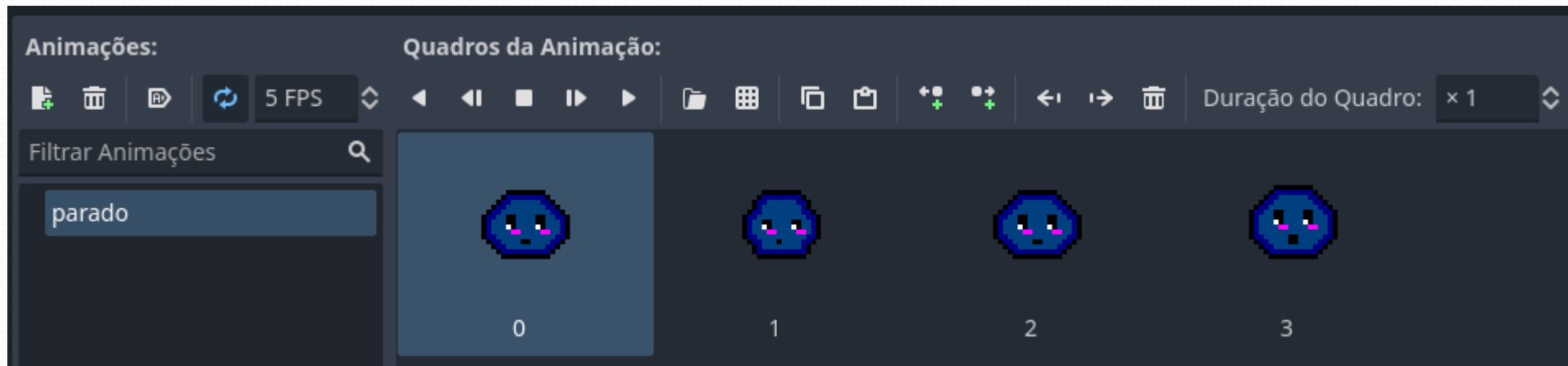
- Você pode incluir um quadro por vez, ou apertar shift e selecionar todos os que quer incluir de uma única vez.



[8]

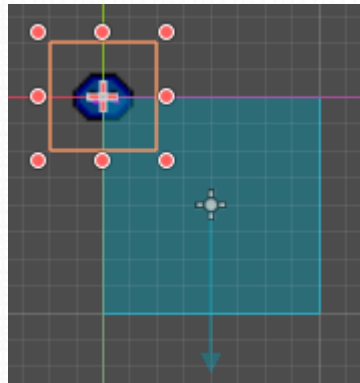
Quadros de animação


- Incluindo as imagens que criamos no **Pixelorama**.
- **Passo 4:** Observe o resultado.
 - Observe que os quadros estão listados, permitindo que você ajuste ou reordene-os no próprio GODOT.

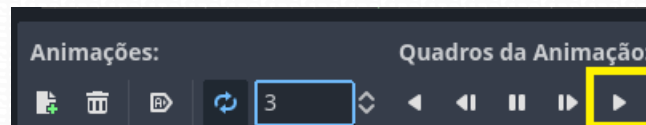


Quadros de animação

- Incluindo as imagens que criamos no **Pixelorama**.
- **Passo 5:** Redimensione e teste sua animação.
 - Minha imagem ficou pequena e deslocada, mas podemos facilmente resolver esse problema.



- Primeiro vá em Inspetor -> Offset e desmarque a opção Centered.
 - Em seguida ajuste o tamanho da imagem para ocupar o quadro.
- 
- Clique em **play** e verifique se a animação está se comportando conforme o planejado.



Script para os quadros de animação

- É a parte mais difícil, precisaremos colocar as regras para ativar essa animação no `script` do Jogador.
- **Passo 1:** Crie uma variável do objeto no código.
 - No script do jogador declare uma variável do tipo que queremos acessar e dê um nome a ela.

```
private AnimatedSprite2D animar;
```



(11)



Script para os quadros de animação

- É a parte mais difícil, precisaremos colocar as regras para ativar essa animação no `script` do Jogador.
- **Passo 2:** Faça o `script` reconhecer o objeto no código.
 - Dentro da função `_Ready()`, ligue o objeto que você quer acessar a variável que acabou de criar.

```
public override void _Ready()  
{  
    ray = GetNode<RayCast2D>("RayCast2D");  
    animar = GetNode<AnimatedSprite2D>("AnimatedSprite2D");  
}
```

- Note que já fizemos isso antes com o `RayCast2D`.



Script para os quadros de animação

- É a parte mais difícil, precisaremos colocar as regras para ativar essa animação no `script` do Jogador.
- **Passo 3:** dar **Play** na animação pelo `script`.
 - Dentro da função `_Ready()`, podemos chamar a função `Play` da nossa animação da seguinte forma.

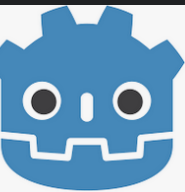
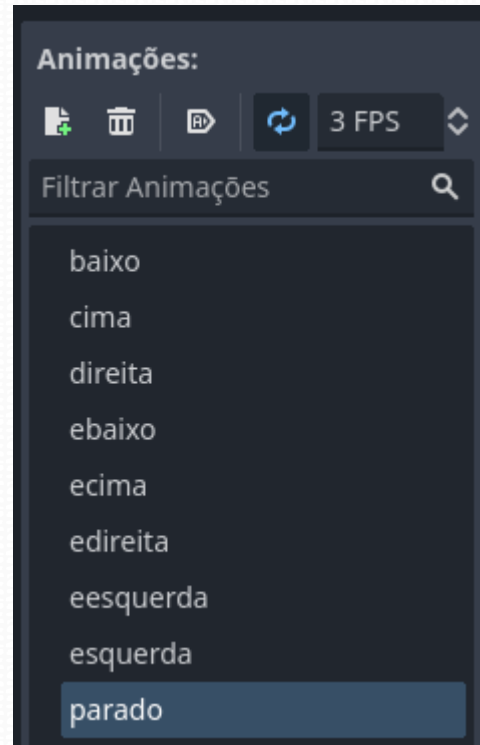
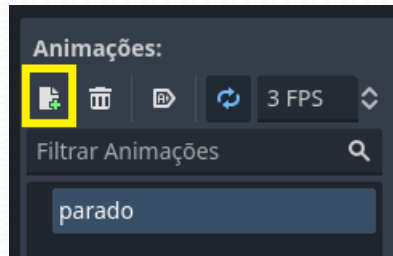
```
public override void _Ready()
{
    ray = GetNode<RayCast2D>("RayCast2D");
    animar = GetNode<AnimatedSprite2D>("AnimatedSprite2D");
    animar.Play("parado");
}
```

- Ao fazer isso, damos **Play** na animação com nome `parado` do nosso `AnimatedSprite2D`.



Quadros de animação atualizados

- Adicionando as demais animações.
 - No AnimatedSprite2D do Jogador, adicione as demais animações utilizando o botão destacado na imagem.



Script para os quadros de animação

- Trocando de animação no `script` do Jogador.
- **Passo 1:** criando a função `Animacao` no `script` do Jogador.
 - Note que no código não podemos dar acento aos nossos nomes, pois são caracteres especiais.

```
public void Animacao()  
{  
  
}
```

- **Alerta:** esse código não pode ficar dentro de nenhuma outra função, se criar ela dentro da função movimento, vai dar erro.



Script para os quadros de animação

- Trocando de animação no `script` do Jogador.
- **Passo 1:** criando a função `Animacao` no `script` do Jogador.
 - Note que no código não podemos dar acento aos nossos nomes, pois são caracteres especiais.

```
public void Animacao()  
{  
  
}
```

- **Alerta:** esse código não pode ficar dentro de nenhuma outra função, se criar ela dentro da função movimento, vai dar erro.



Script para os quadros de animação

- Trocando de animação no `script` do Jogador.
- **Passo 2:** criando a função `Animacao` no `script` do Jogador.
 - Chame a animação corretamente de acordo com a direção.

```
public void Animacao()
{
    if (direcao == new Vector2(0,0)) {
        animar.Play("parado");
    } else if (direcao == new Vector2(-1,0)) {
        animar.Play("esquerda");
    } else if (direcao == new Vector2(1,0)) {
        animar.Play("direita");
    } else if (direcao == new Vector2(0,-1)) {
        animar.Play("cima");
    } else if (direcao == new Vector2(0,1)) {
        animar.Play("baixo");
    }
}
```



Script para os quadros de animação

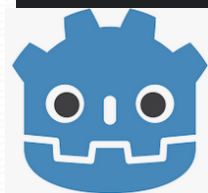
- Trocando de animação no `script` do Jogador.
- **Passo 3:** chame a a função `Animacao()` no `script` do Jogador no momento correto.
 - Se chamarmos a função logo após o `update` do `RayCast2D`, poderemos mudar a animação de acordo com o teste de colisão.

```
//teste se pode mover  
ray.TargetPosition = direcao * area_quadrado;  
ray.ForceRaycastUpdate();  
Animacao();
```



Script para os quadros de animação

- Trocando de animação no `script` do Jogador.
- **Passo 4:** Teste o jogo.
 - Seu personagem deve mudar de direção de acordo com a direção que está indo.



Script para os quadros de animação

- **DESAFIO!!!**
- **Passo 5:** Os movimentos que implementamos foram os movimentos básicos, ou seja, quando o Jogador não está colidindo com nada, mas caso estiver colidindo deveremos utilizar as animações de empurrar.
 - Lembrando que verificamos que não está colidindo com o comando:
 - `if (!ray.IsColliding()) {}`
 - E verificamos que está colidindo com o comando:
 - `if (ray.IsColliding()) {}`
- Boa sorte utilizando esses comandos na função `Animacao()` para executar os movimentos corretos.



Obrigado pela atenção!
