

# Oficina de programação de jogos

Criando o jogo Flappy  
(no motor de jogos Godot)

# Teoria



**Saman Bemel Benrud**  
@samanbb

...

Game developers: with enough if statements and while loops I can do literally anything.

Web developers: I will use graph theory and a hand crafted functional state management framework to create this sign up form.

9:06 PM · 19/02/2021 · Twitter Web App

Adding another programming language  
to my resume after learning how to write  
Hello World in it.



```
func _ready():
    $Label.text = "Hello world!"
```

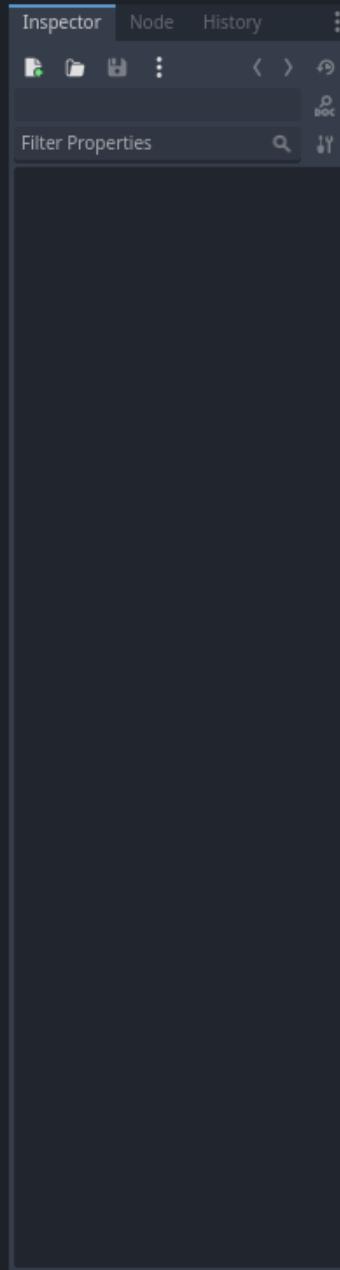
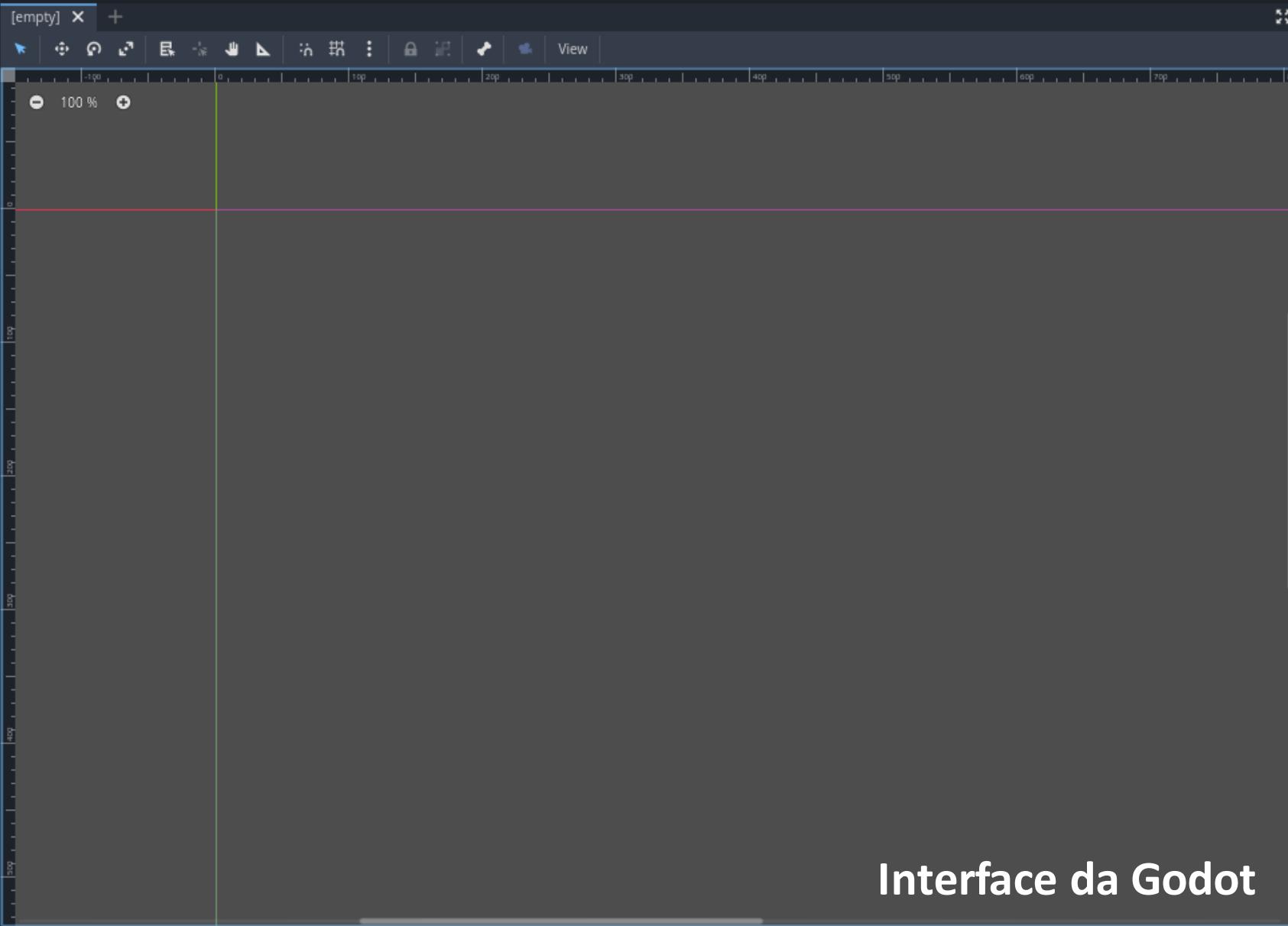
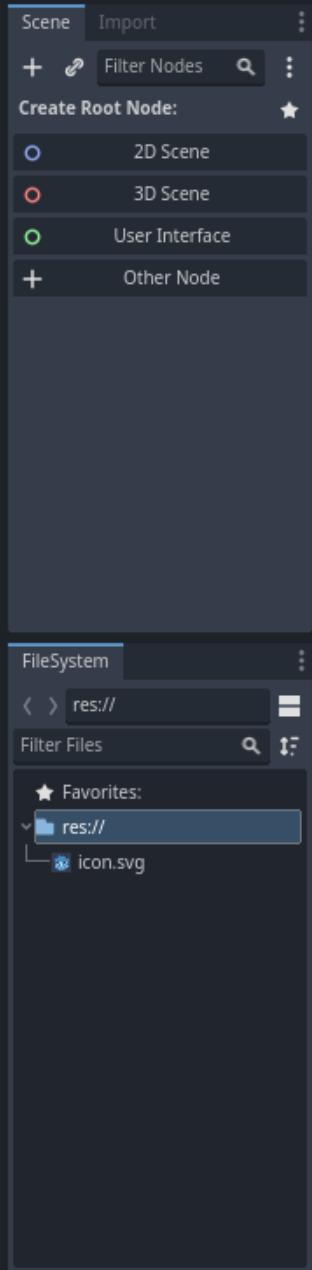
Olá Mundo!

Scene Project Debug Editor Help

2D 3D Script AssetLib

Output Debugger Audio Animation Shader Editor

4.0.beta6



Interface da Godot

## Limiting Healing

### Goals

We have a heal function that adds an amount to the character's health.

Add to the function so the character's health is never greater than 80.

### Hints

### Checks

- Health Does Not Go Above 80
- Health Limit Is Not Too Low
- Health Takes Different Values

```
1 func heal(amount):  
2     health += amount  
3     if health > 80:  
4         health = 80
```



### Output

Run    Pause    Reset  
Solution    Output    Continue

≡ Open Practice List

Aplicativo "Learn GDScript"

Average  
fan



**GDScript**  
enjoyer



Linguagens



```
Heatmap.cpp x
24 #include "Heatmap.h"
23 #include <TileMap.hpp>
22 #include <SceneTree.hpp>
21 #include <Viewport.hpp>
20 #include <deque>
19 #include <TileSet.hpp>
18 #include <algorithm>
17
16 namespace godot {
15     inline float lerp(const float& a, const float& b, const float& t) {
14         return a + t * (b - a);
13     }
12
11 void Heatmap::_register_methods() {
10     //public
9         register_method("_ready", &Heatmap::_ready);
8         register_method("_draw", &Heatmap::_draw);
7         register_method("_process", &Heatmap::_process);
6         register_method("best_direction_for", &Heatmap::best_direction_for);
5         register_method("calculate_point_index", &Heatmap::calculate_point_index);
4         register_method("calculate_point_index_for_world_position",
            &Heatmap::calculate_point_index_for_world_position);
3
2     //semi-private
1         register_method("_on_Events_player_moved", &Heatmap::on_Events_player_moved);
25
1     //properties
2         register_property<Heatmap, NodePath>("pathfinding_tilemap", &Heatmap::m_pathfinding_tilemap,
        NodePath());
1
● 10k godot-platformer-2d/game/src/Native/Heatmap/Heatmap.cpp 25:0 Top7:35AM 0.97 LF UTF-8 C++/l master
```

C++

C#

```
if (Input.GetKeyDown(KeyCode))
{
    "ui_accept"
    "ui_cancel"
    "ui_down"
    "ui_end"
    "ui_focus_next"
    "ui_focus_prev"
    "ui_home"
    "ui_left"
    "ui_page_down"
    "ui_page_up"
    "ui_select"
}
```

NodePath Sprite = ;

- "Camera" NodePath
- "Platform" NodePath
- "Platform/CollisionShape" NodePath
- "Platform/Sprite" NodePath

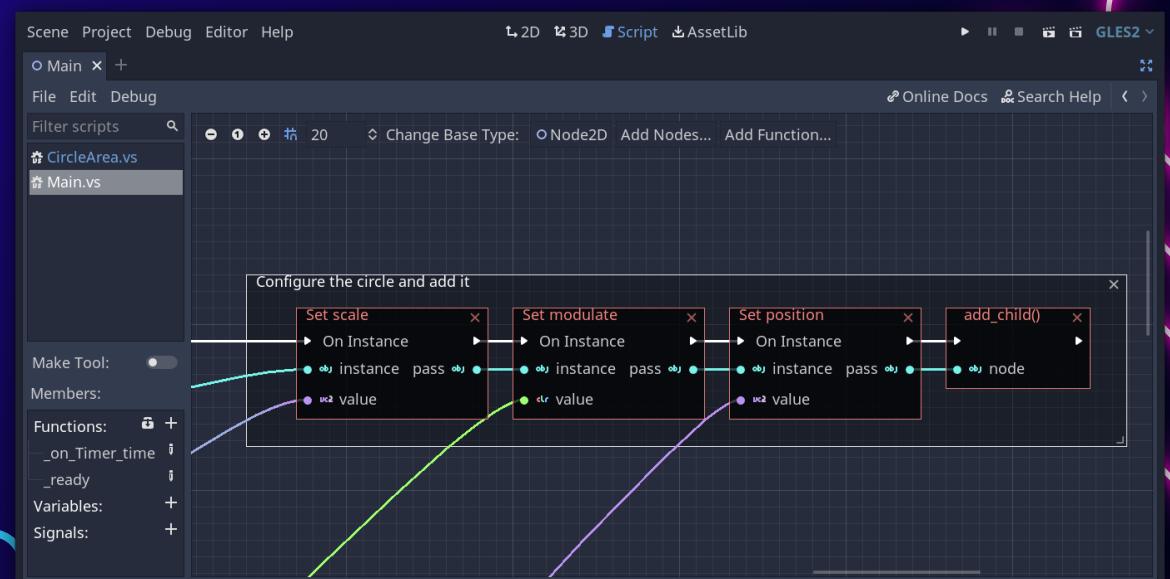
GetNode<Sprite>().Visible = true;

- "Camera" NodePath
- "Platform" NodePath
- "Platform/CollisionShape" NodePath
- "Platform/Sprite" NodePath

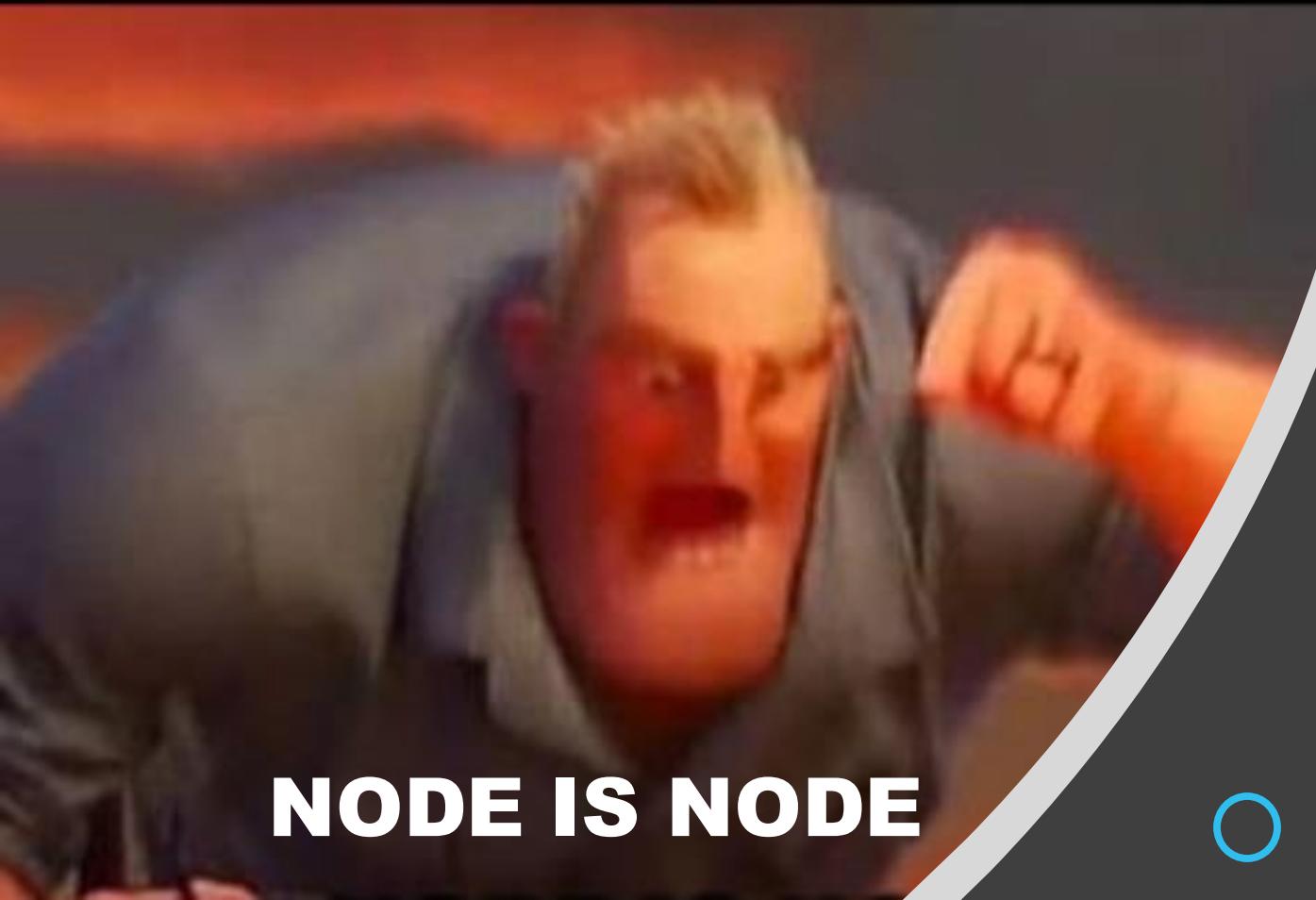
- #if  
- #region  
+ AABB  
+ AcceptDialog  
+ AcceptEvent  
+ AccessViolationException  
+ Action<>  
+ Aabb  
+ AcceptDialog  
+ AcceptEvent  
+ AccessViolationException  
+ Action<>

## GDScript

## Visual Script



**WHEN YOU HEAR OTHER GAME  
DEVS TALK ABOUT GAMEOBJECTS,  
PREFABS, SCENES, BLUEPRINTS  
AND MAPS, BUT YOU USE GODOT:**

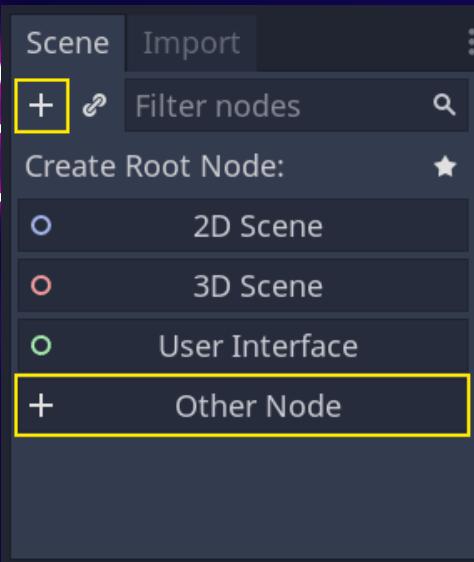


**NODE IS NODE**

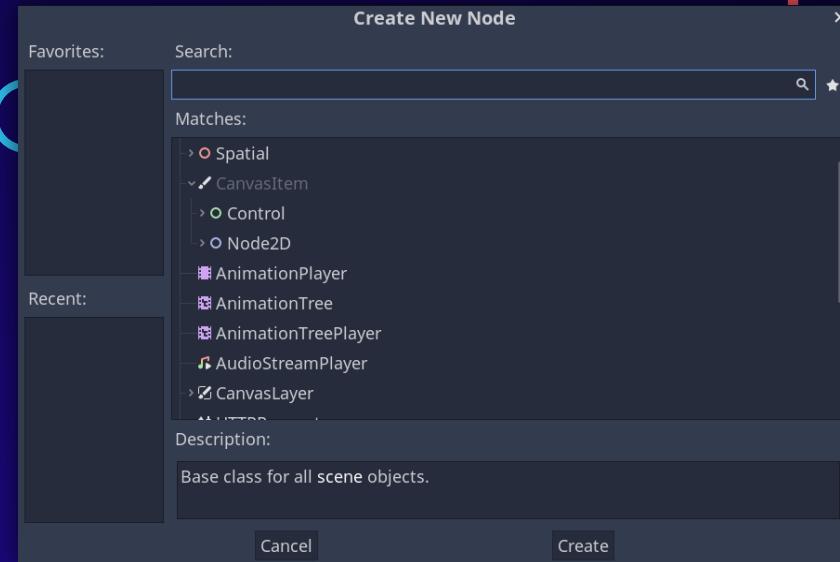
Nodes  
&  
Scenes



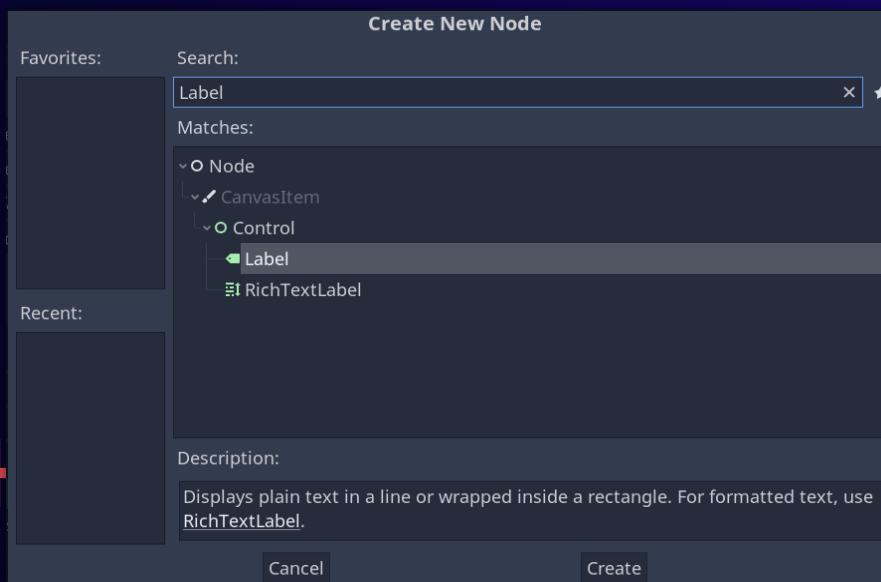
# Criando uma cena



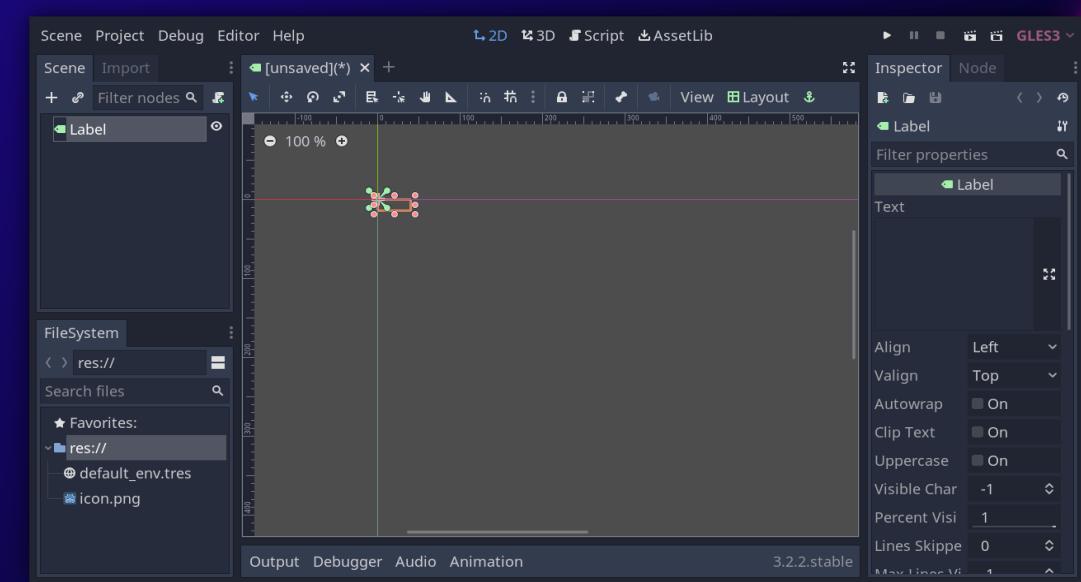
# Selecionar nó

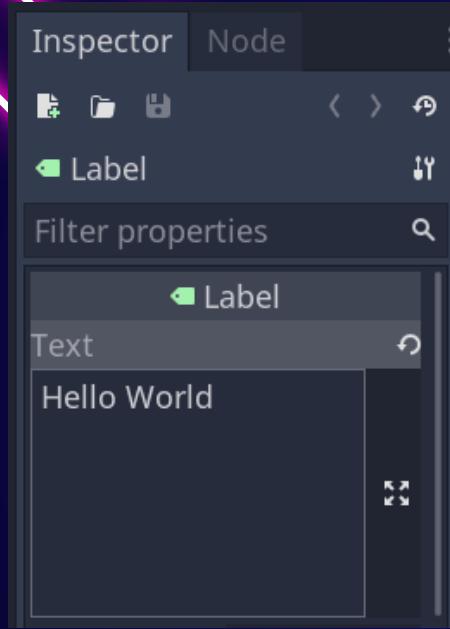


## Escolher nó "Label"

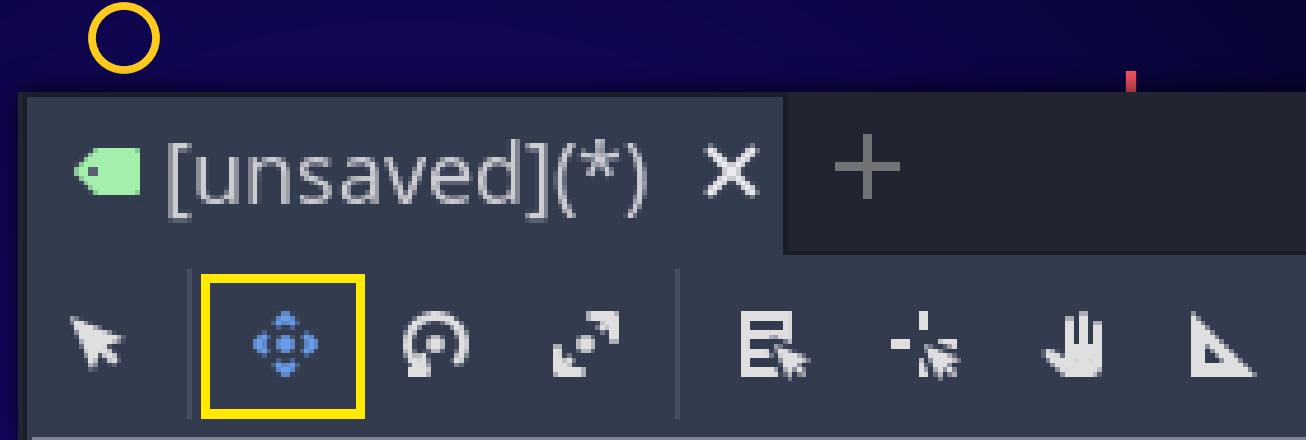


## Label inserida

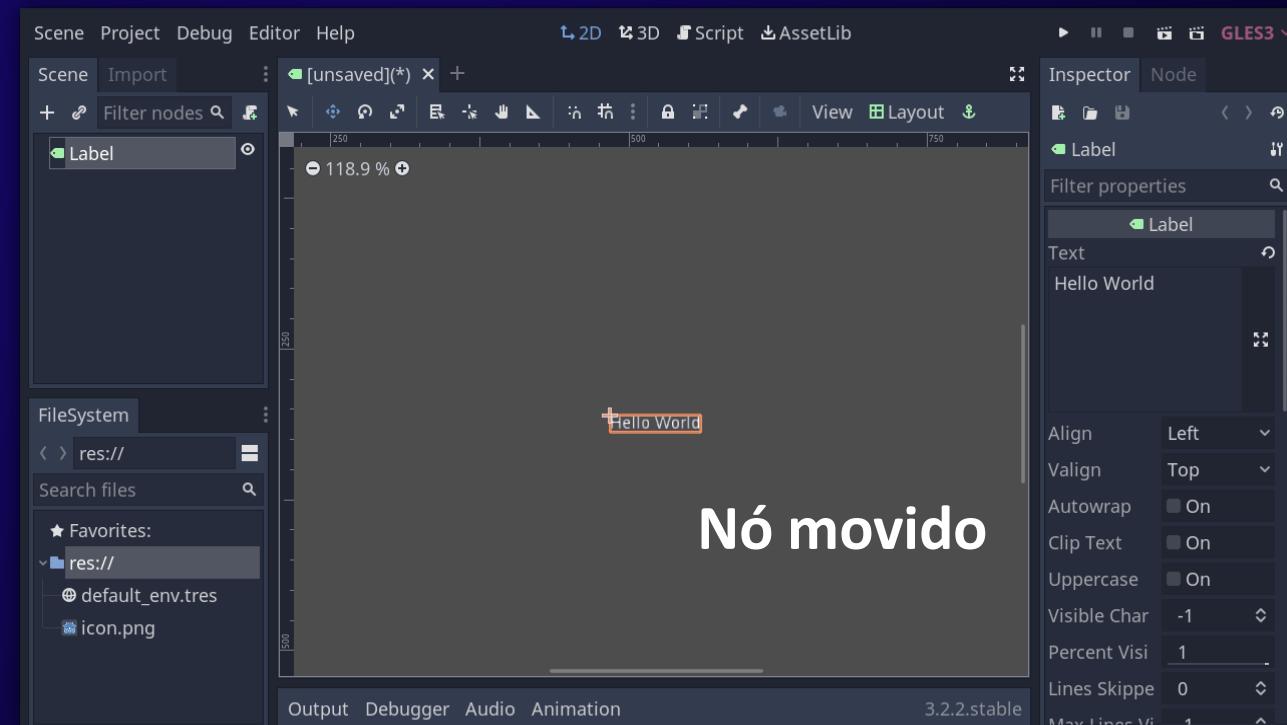




Propriedades do nó  
(Text é um atributo  
do tipo String)



## Ferramenta de movimentação



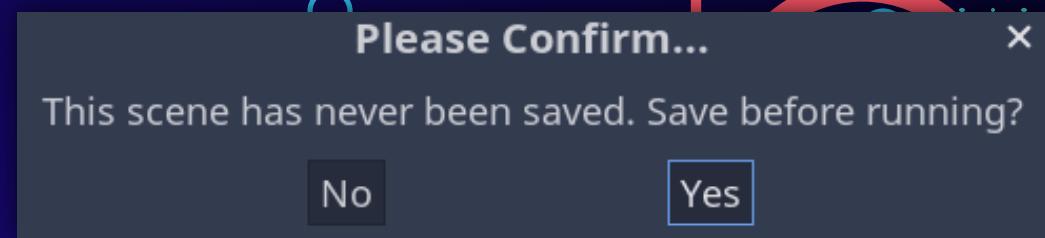
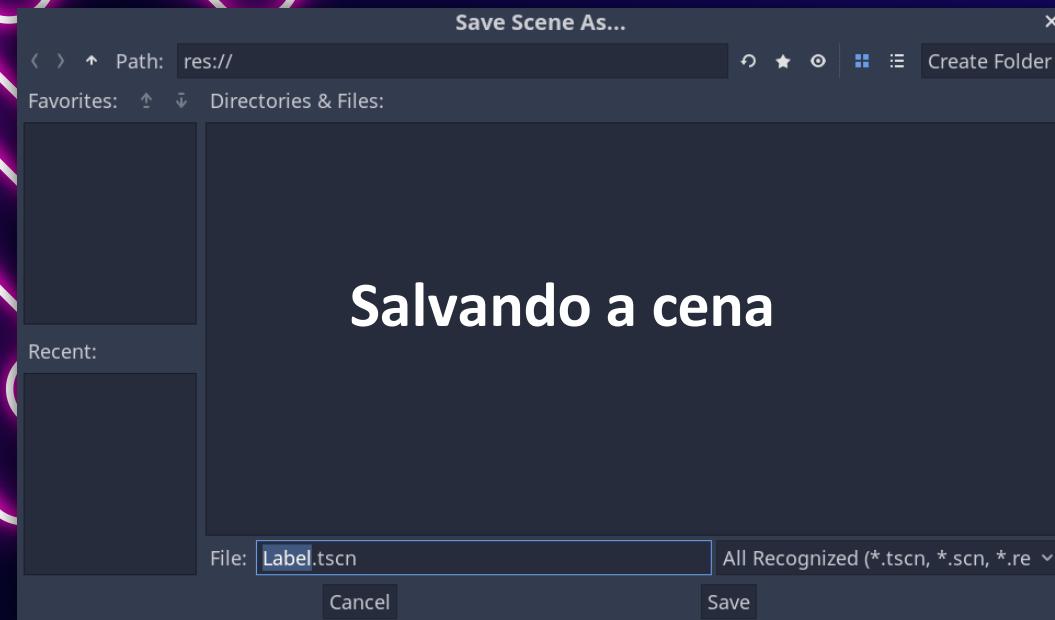
Nó movido

Executar cena

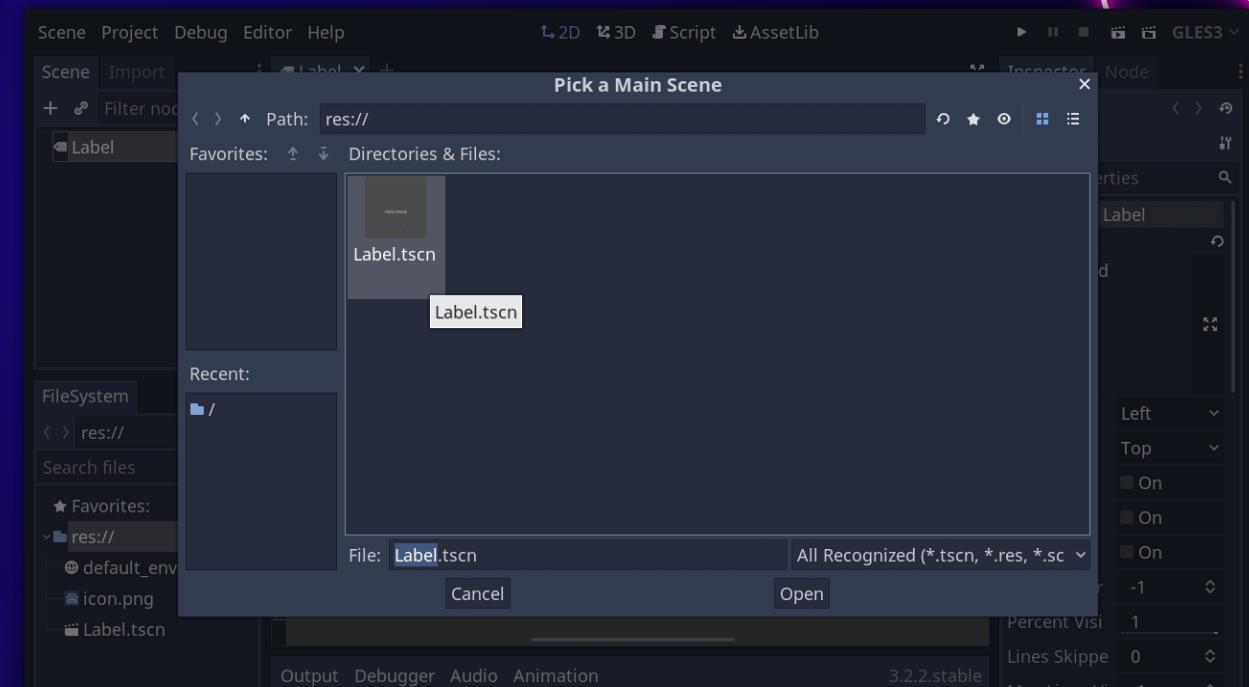


GLES3

Salvando a cena



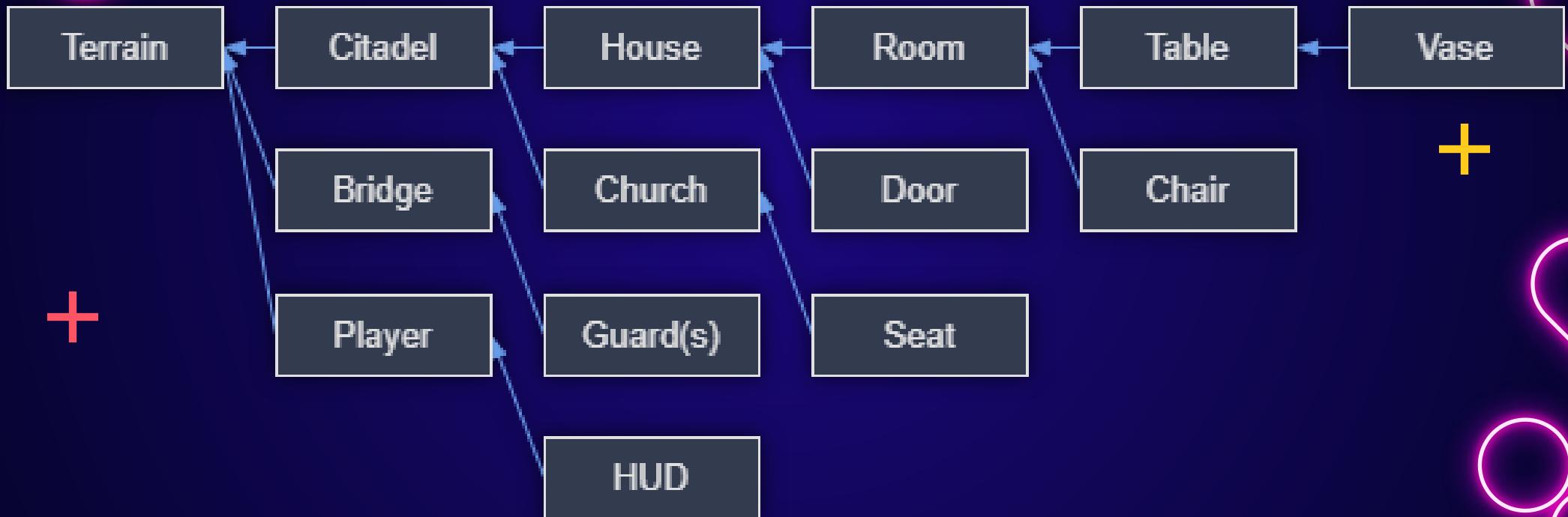
É necessário salvar a cena antes

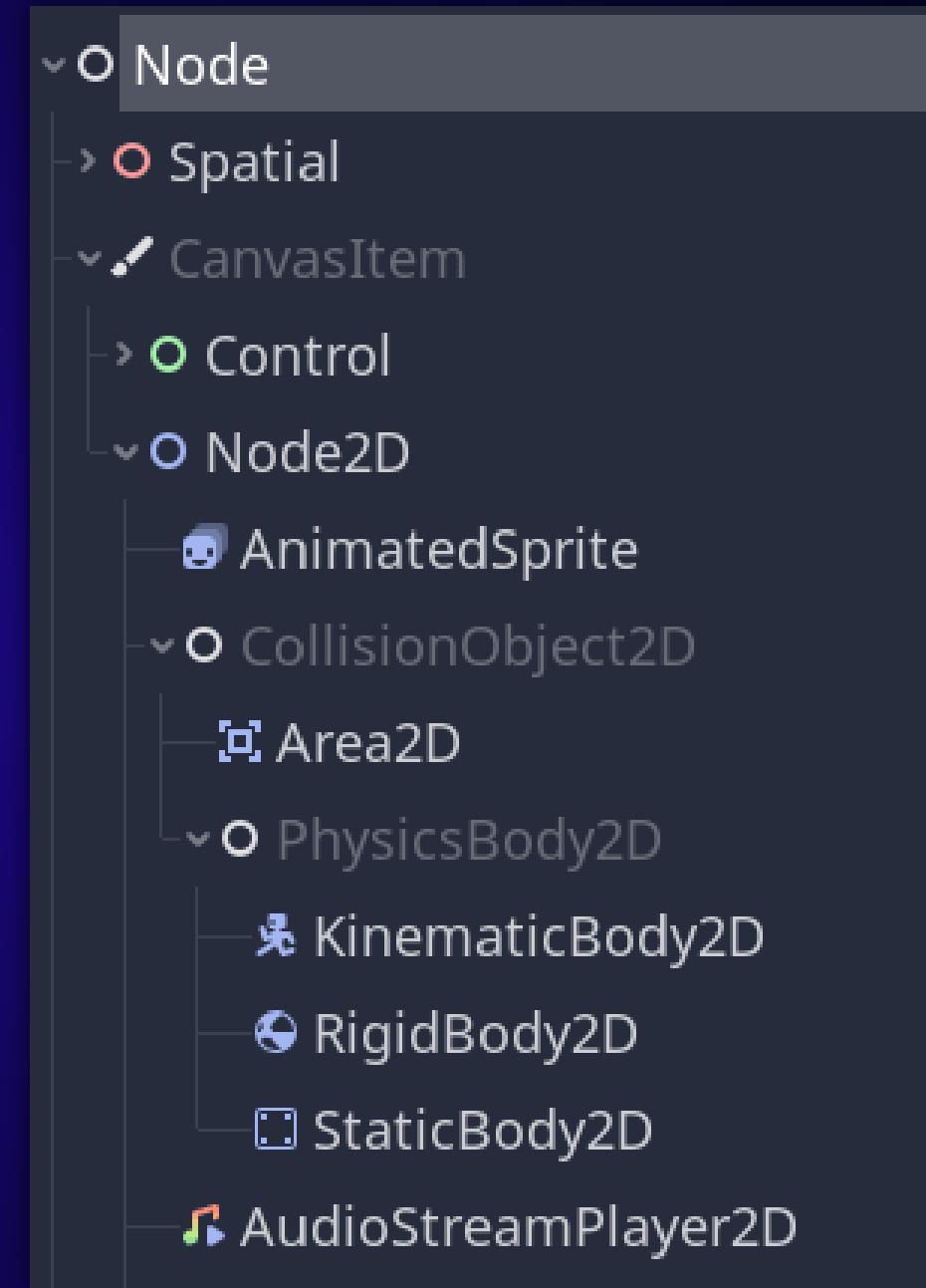
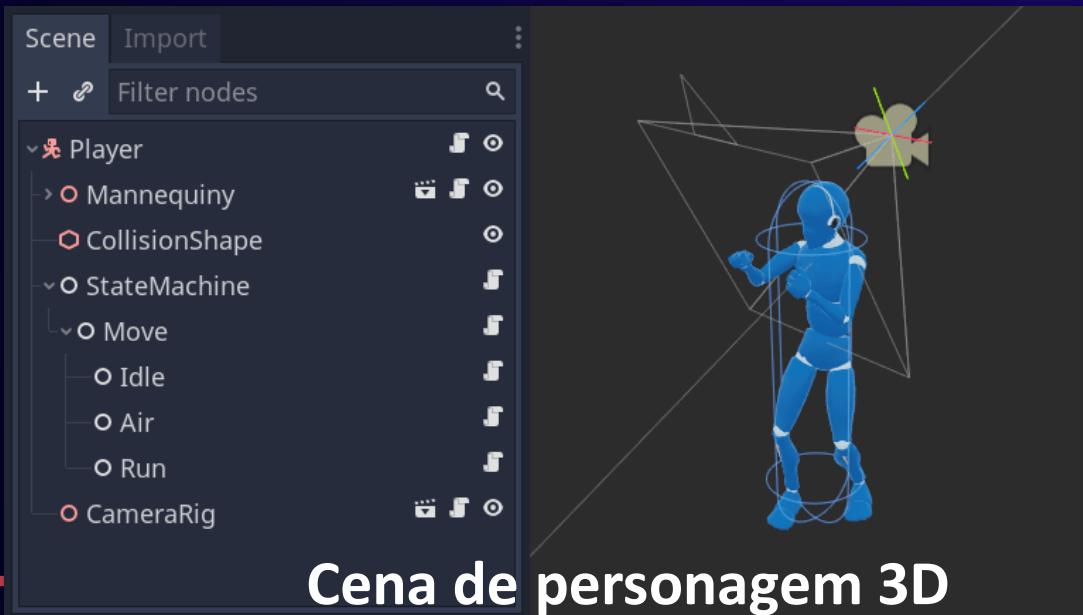


Usamos a cena salva +  
como a principal



A hierarquia das cenas é uma forma de agregação que permite projetar níveis mais intuitivamente





# Prática



Jaeheon Shim  
@jaeheonshim

Theory is when you know everything  
but nothing works.

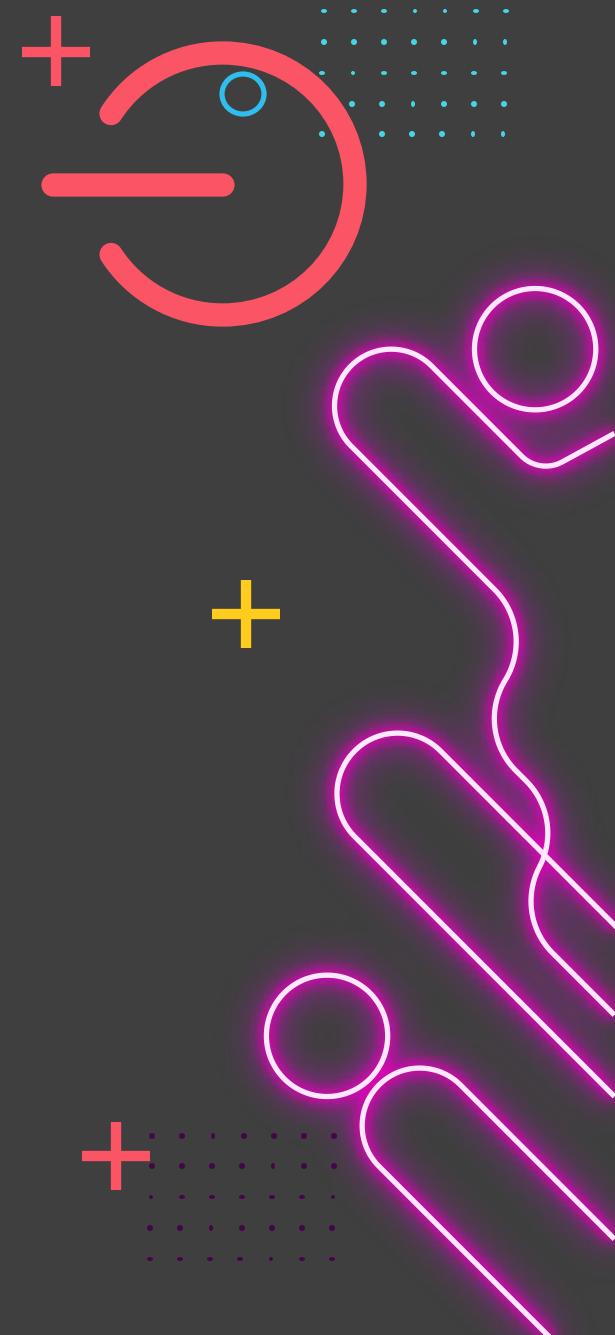
Practice is when everything works but  
nobody knows why.

In computer programming, theory and  
practice are combined: nothing works  
and nobody knows why.

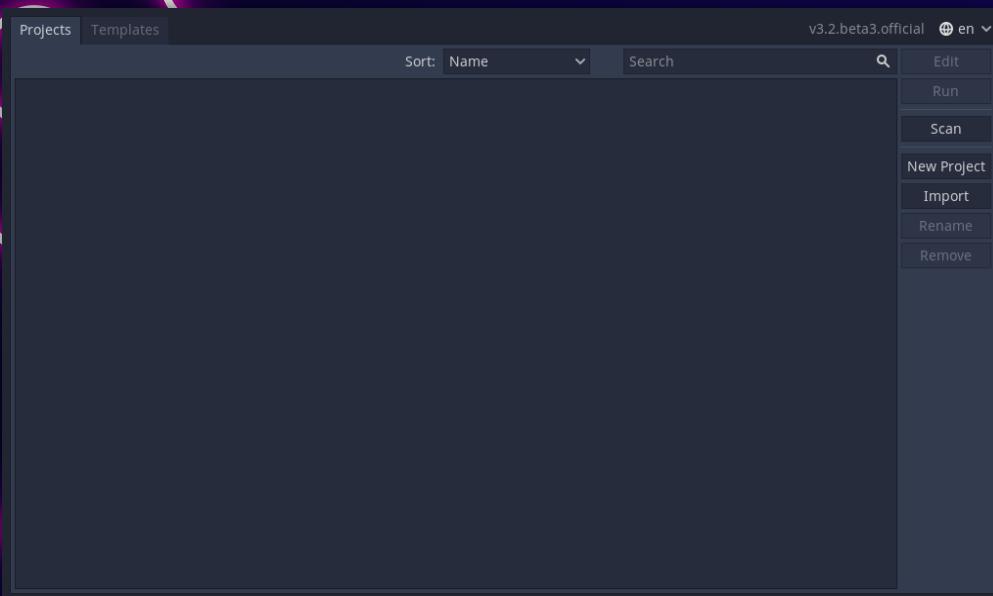
# Friendship ended with unity



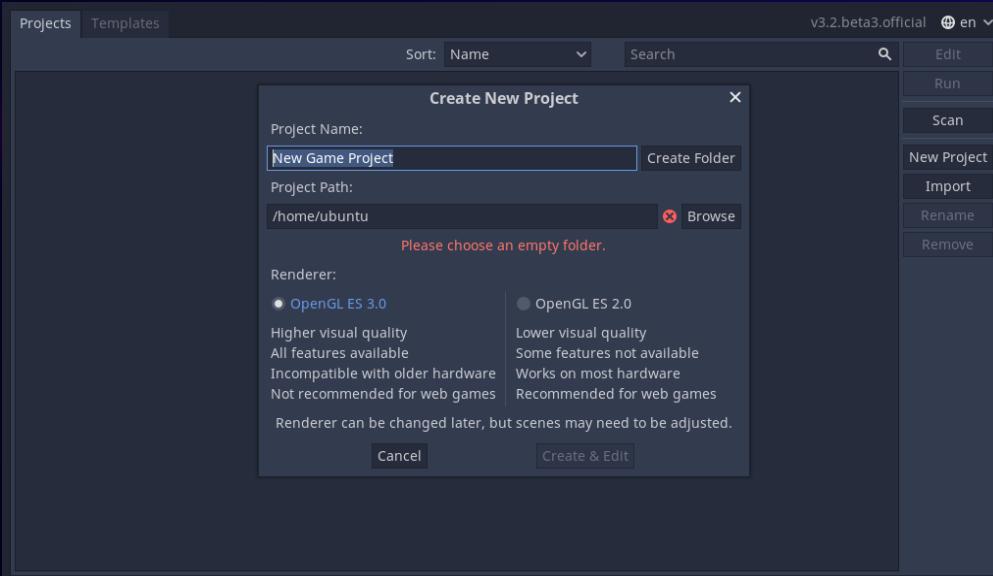
Editor



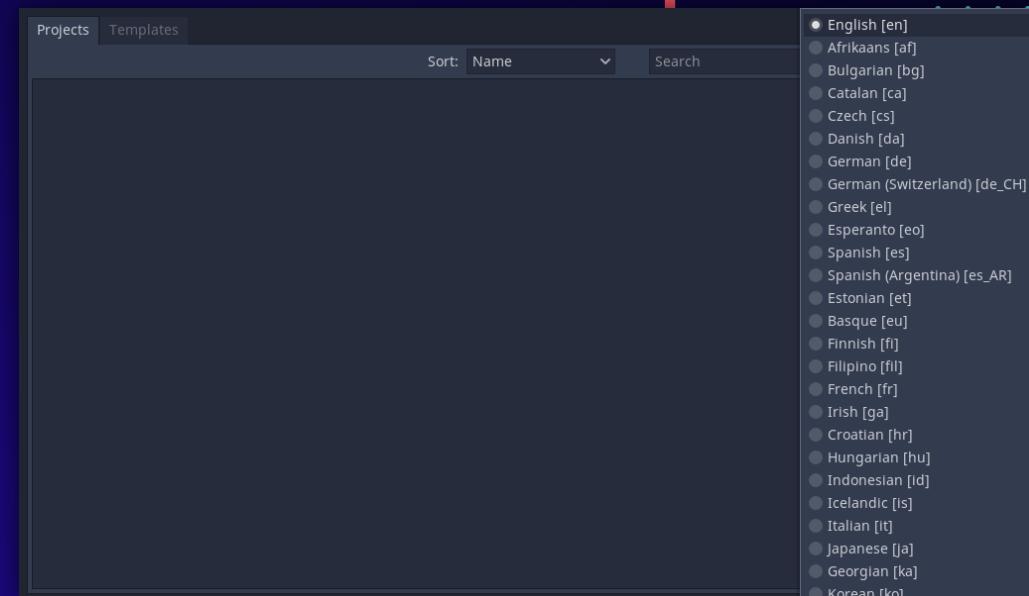
# Gerenciador de projetos



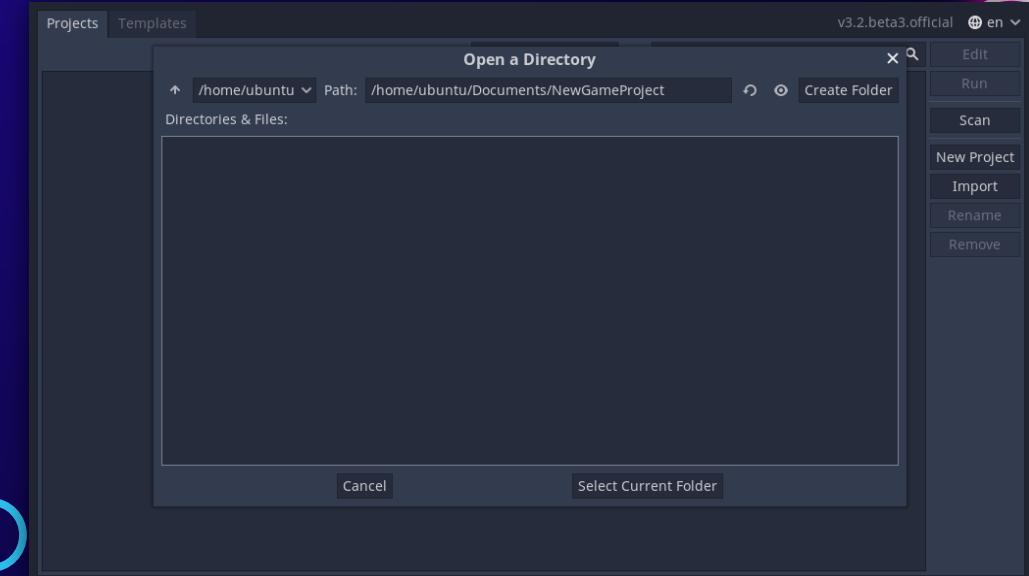
## Novo projeto



# Seleção de idiomas



## Navegador de arquivos





Search

Sort: Recently Updated

Category: All

Site: godotengine.org

Support

First Previous 1 2 3 4 5 6 Next Last



FirstPersonStarter

Templates

Whimfoome

MIT



Oculus Quest Button Testing

Projects

creikey

MIT



Mannequin: Open 3D Mannequin

Templates

gdquest

MIT

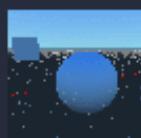


Inventory System Example

Demos

Oen44

MIT



RigidBody Planetary Physics Correct Stand Up

Demos

ComancheAk

MIT



Top Down Twin Stick Shooter

Templates

RedSlimeSkirt

MIT



Controller use Template

Templates

Anne

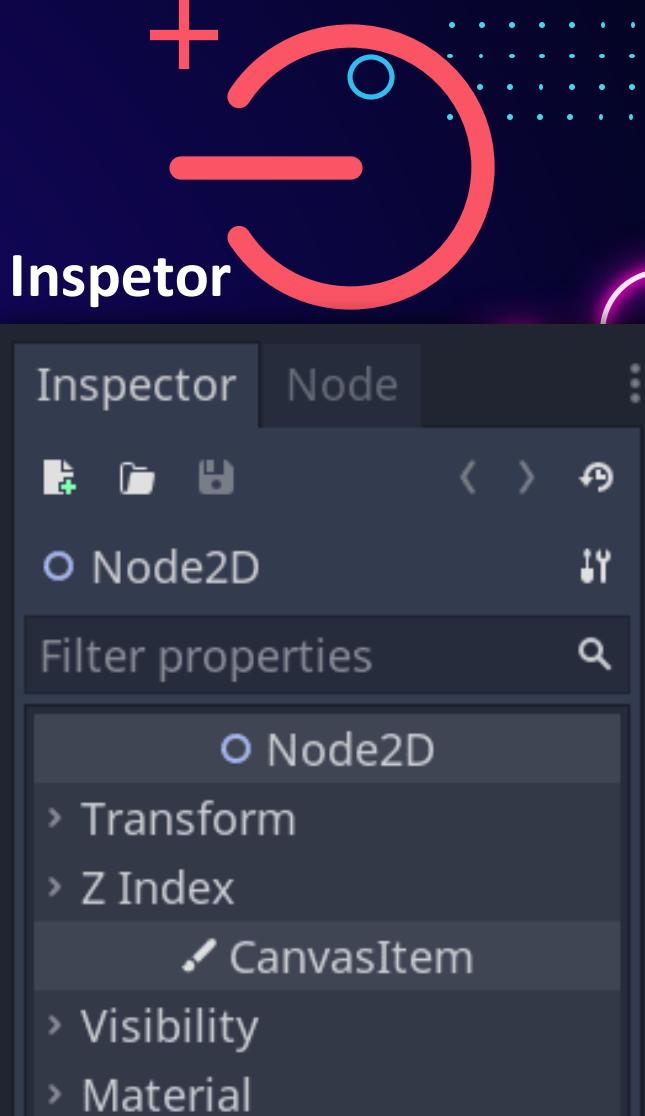
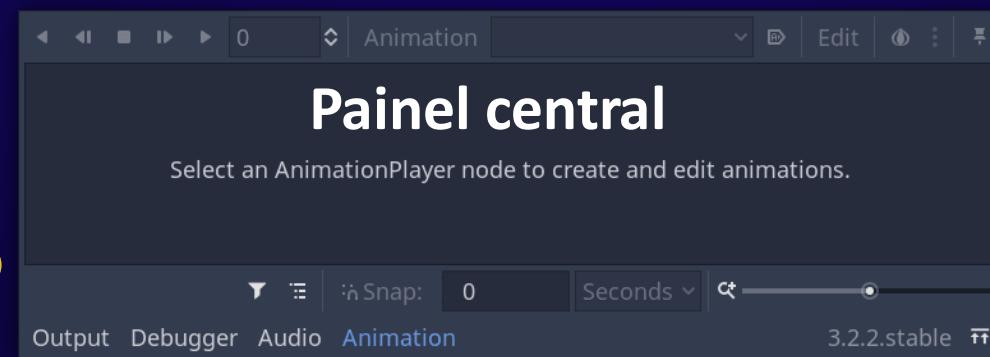
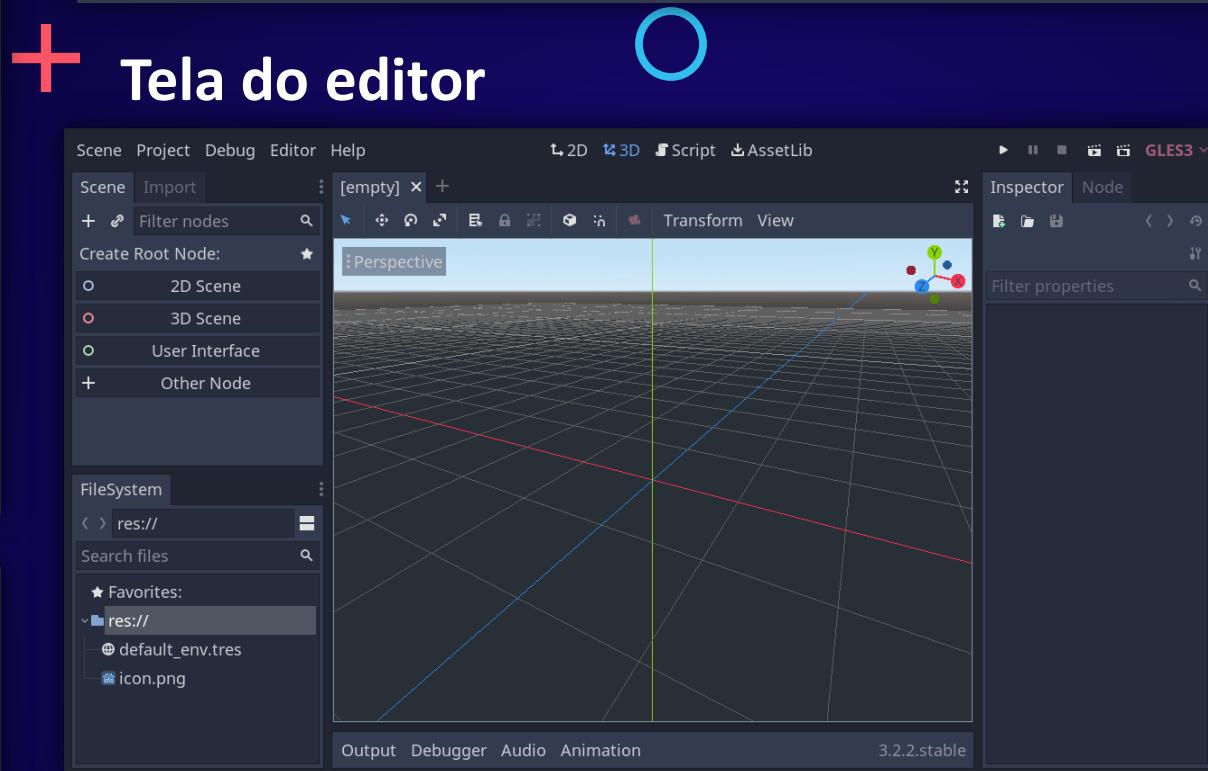
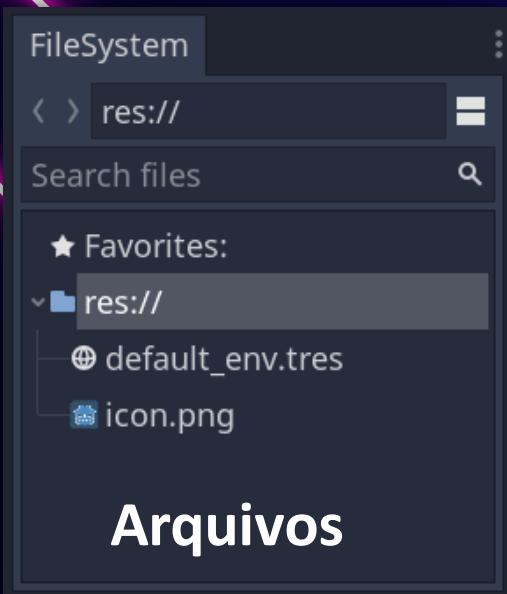
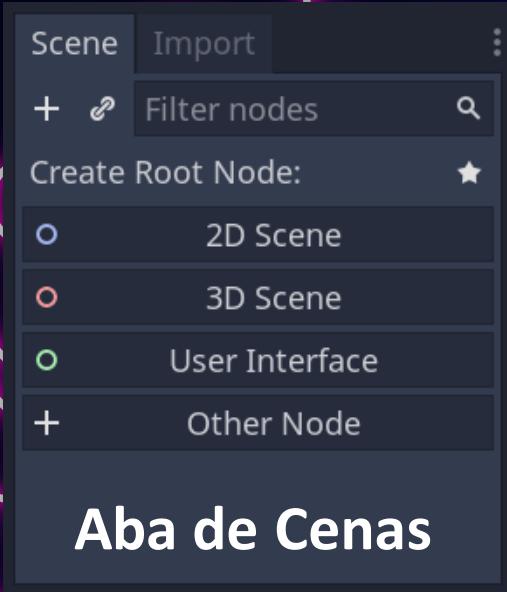


Speed feeling Godot 2D

Templates

Anne

# Barra de ferramentas

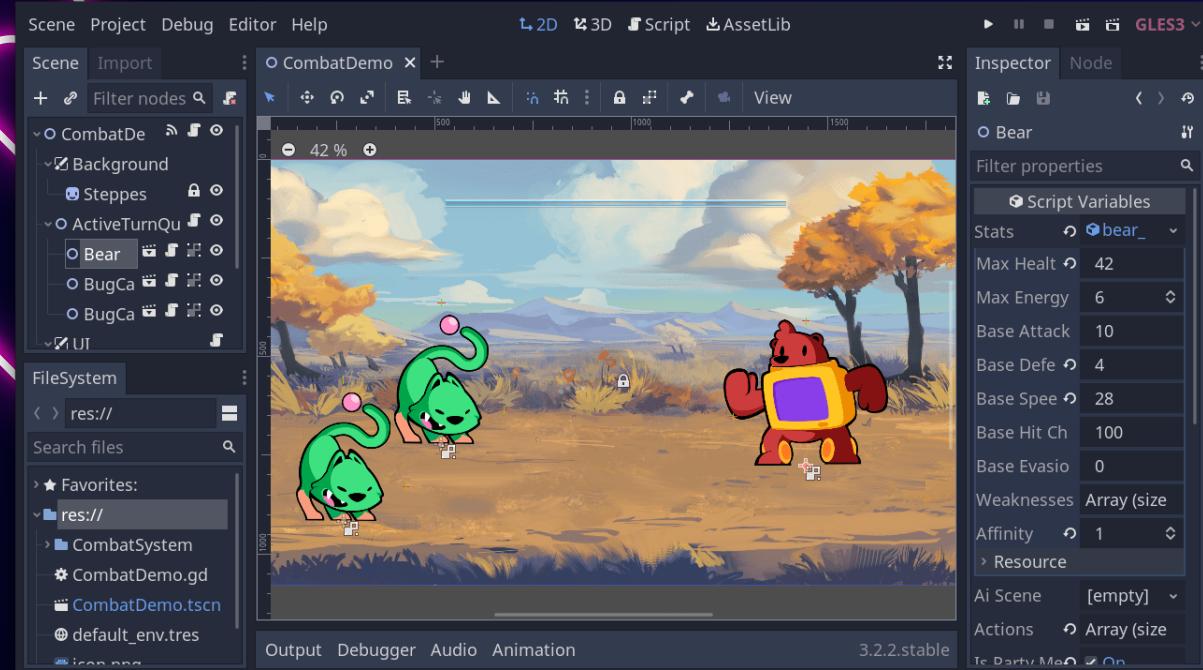


(aba de propriedades)

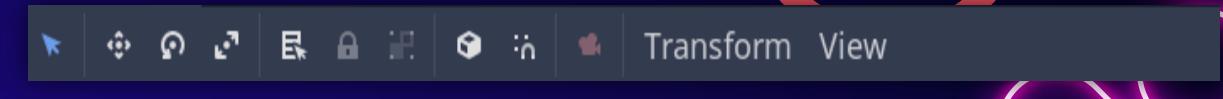
## Barra de ferramentas 2D



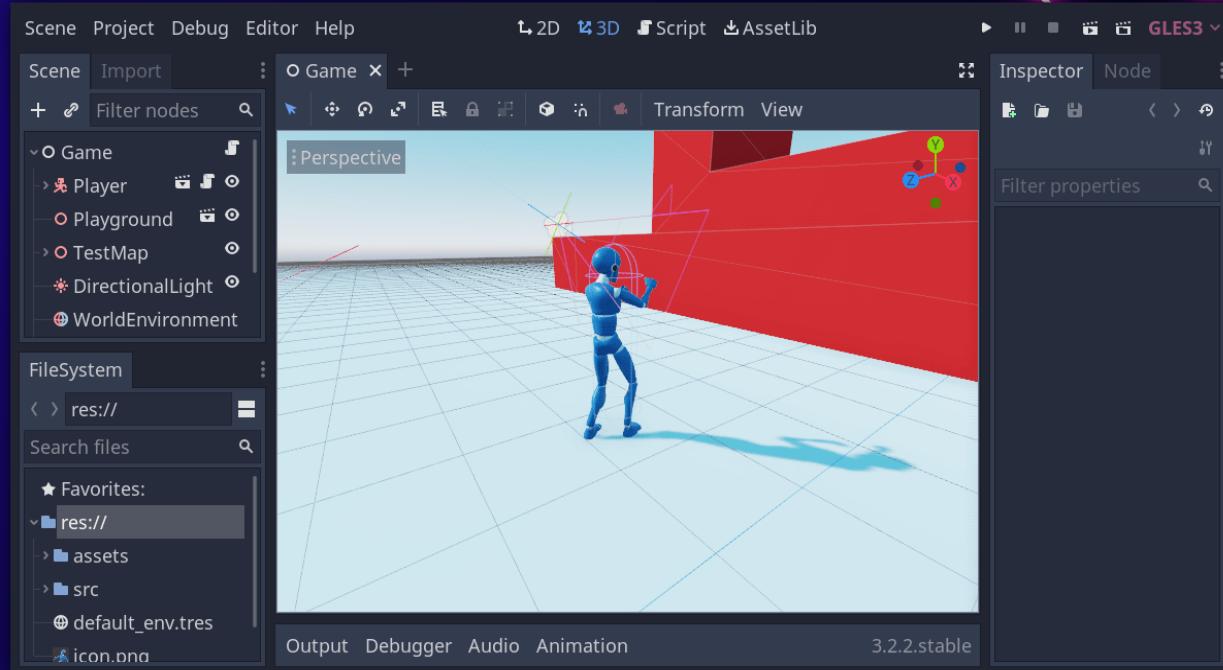
## Editor 2D



## Barra de ferramentas 3D



## Editor 3D



## Editor de texto

```
1 ## Character or monster that's participating in combat.
2 ## Any battler can be given an AI and turn into a
computer-controlled ally or a foe.
3 extends Node2D
4 class_name Battler
5
6 ## Emitted when the battler is ready to take a turn.
7 signal ready_to_act
8 ## Emitted when an animation from `battler_anim` finished
playing.
9 signal animation_finished(anim_name)
10 ## Emitted when the battler finished their action and arrived
back at their rest position.
11 signal action_finished
12 signal readiness_changed(new_value)
13 signal health_depleted
14 signal selection_toggled(value)
15 signal damage_taken(amount)
16 signal hit_missed
```

## Documentação

**Class: AnimatedSprite**  
Inherits: [Node2D](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Description**

Animations are created using a [SpriteFrames](#) resource, which can be configured in the editor via the [SpriteFrames](#) panel.

**Properties**

	SpriteFrames	frames
String	<code>animation</code> [default: "default"]	
int	<code>frame</code> [default: 0]	
float	<code>speed_scale</code> [default: 1.0]	
bool	<code>allow_modifying</code> [default: false]	

# Janela de ajuda

Search Help



Aa



Display All



Animated

Object

Class

Node

Class

CanvasItem

Class

Node2D

Class

AnimatedSprite

Class

animation\_finished

Signal

VisibilityNotifier2D

Class

VisibilityEnabler2D

Class

.c ENABLER\_PAUSE\_ANIMATED\_SPRITES

Constant

.P pause\_animated\_sprites

Property

Spatial

Class

VisualInstance

Class

GeometryInstance

Class

Cancel

Open

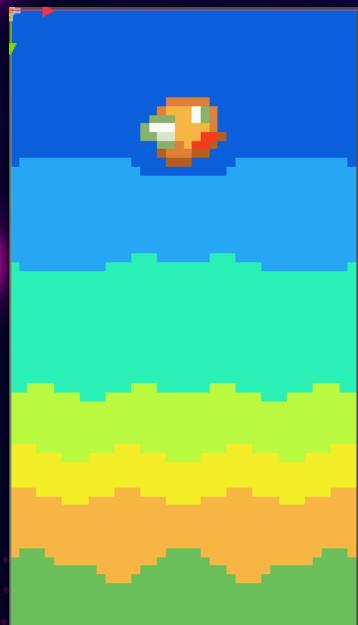
**GODOT IS A CULT  
BUT IT'S NOT A BAD THING.**



**YEAH, RIGHT**

Imagens

# Cenário



OPENGAMEART.ORG

Home Browse Submit Art Collect Forums FAQ Leaderboards 🍀 Donate

Messages Hello, Linky! Log out

ADVANCED SEARCH

SEARCH flappy bird

TITLE

TAGS Is one of

Enter a COMMA SEPARATED list of tags. (example: "sword, weapon, item")

SUBMITTER

ART TYPE

- ✓ 2D Art ✓ 3D Art
- ✓ Concept Art ✓ Texture
- ✓ Music ✓ Sound Effect
- ✓ Document

LICENSE(S)

- ✓ CC-BY 4.0 ✓ CC-BY 3.0
- ✓ CC-BY-SA 4.0 ✓ CC-BY-SA 3.0
- ✓ GPL 3.0 ✓ GPL 2.0
- ✓ OGA-BY 3.0 ✓ CC0
- ✓ LGPL 3.0 ✓ LGPL 2.1

SORT BY Search relevance ORDER Desc

ITEMS PER PAGE 24

COLLECT INTO... Select a collection

SEARCH

CHAT WITH US!

SEARCH ART

LEGAL NOTICE REGARDING NFTs:

WARNING: Taking art from OpenGameArt.org to be sold as NFTs? You may be committing FRAUD. Visit this link for legal details: <https://opengameart.org/content/warning-taking-art-from-opengameartorg-t...>

Note of caution to NFT purchasers or those interested in trading NFTs: You could be getting scammed! Please visit this link for more information: <https://opengameart.org/content/note-of-caution-to-nft-purchasers-or-tho...>

BEVOULIN FREE ... FLAPPY GRUMPY... BEVOULIN GREE... GAME CHARACTE... BEVOULIN FREE ... PINK FLAPPY B...

FLAPPY DRAGON... BEVOULIN FREE ... BEVOULIN FREE ... FAT BIRD SPRITE... FREE GAME ASS... BEVOULIN FREE ...

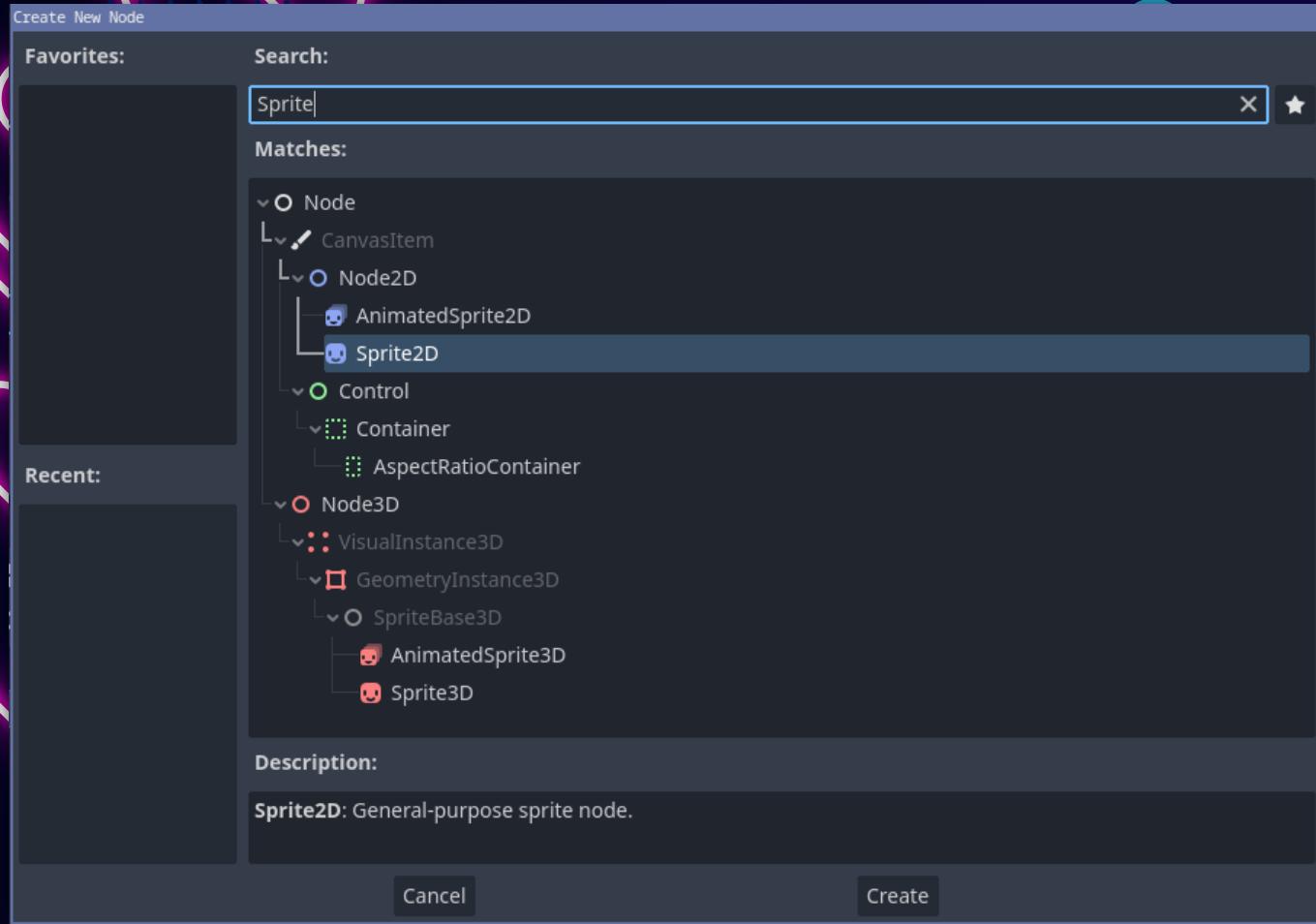
FLAPPY BIRD CL... FLAPPY BIRD BAC... FLAPPY BIRD Sp... FLAPPY BUNNY FLAPPY BEANS

A grid of 20 thumbnail images for various flying bird assets. The thumbnails are arranged in four rows of five. Some of the visible titles include "FLAPPY DRAGON...", "FLAPPY BIRD CL...", and "FLAPPY BUNNY". Each thumbnail shows a different stylized or cartoonish bird character in flight.

# OpenGameArt.org

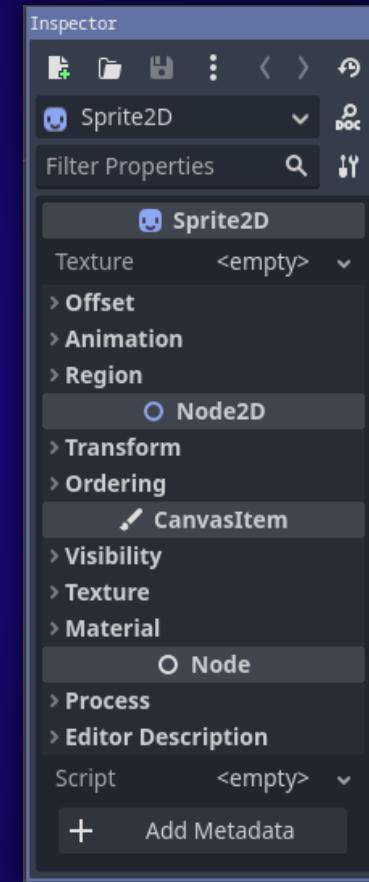
Demo +



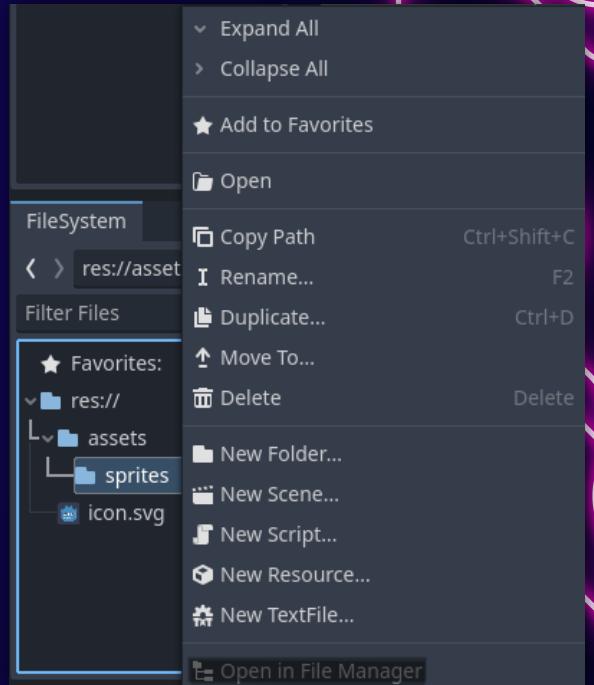


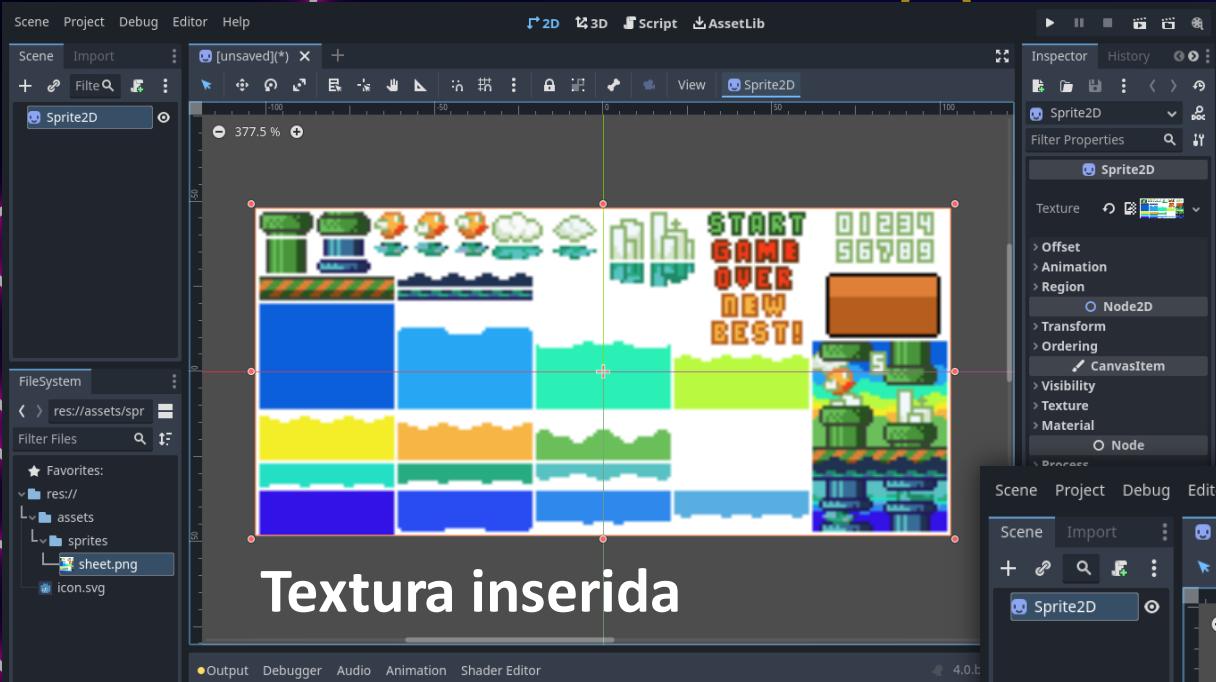
Selecionar nó "Sprite2D"

## Inspector

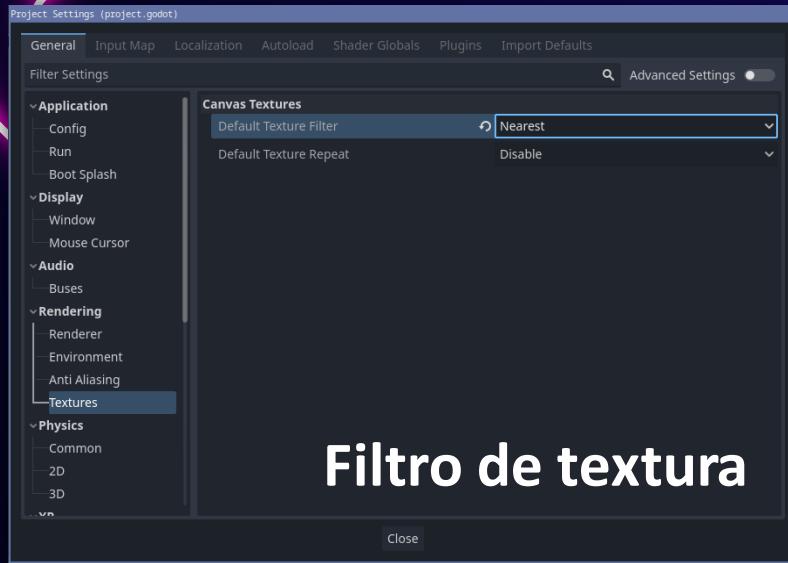
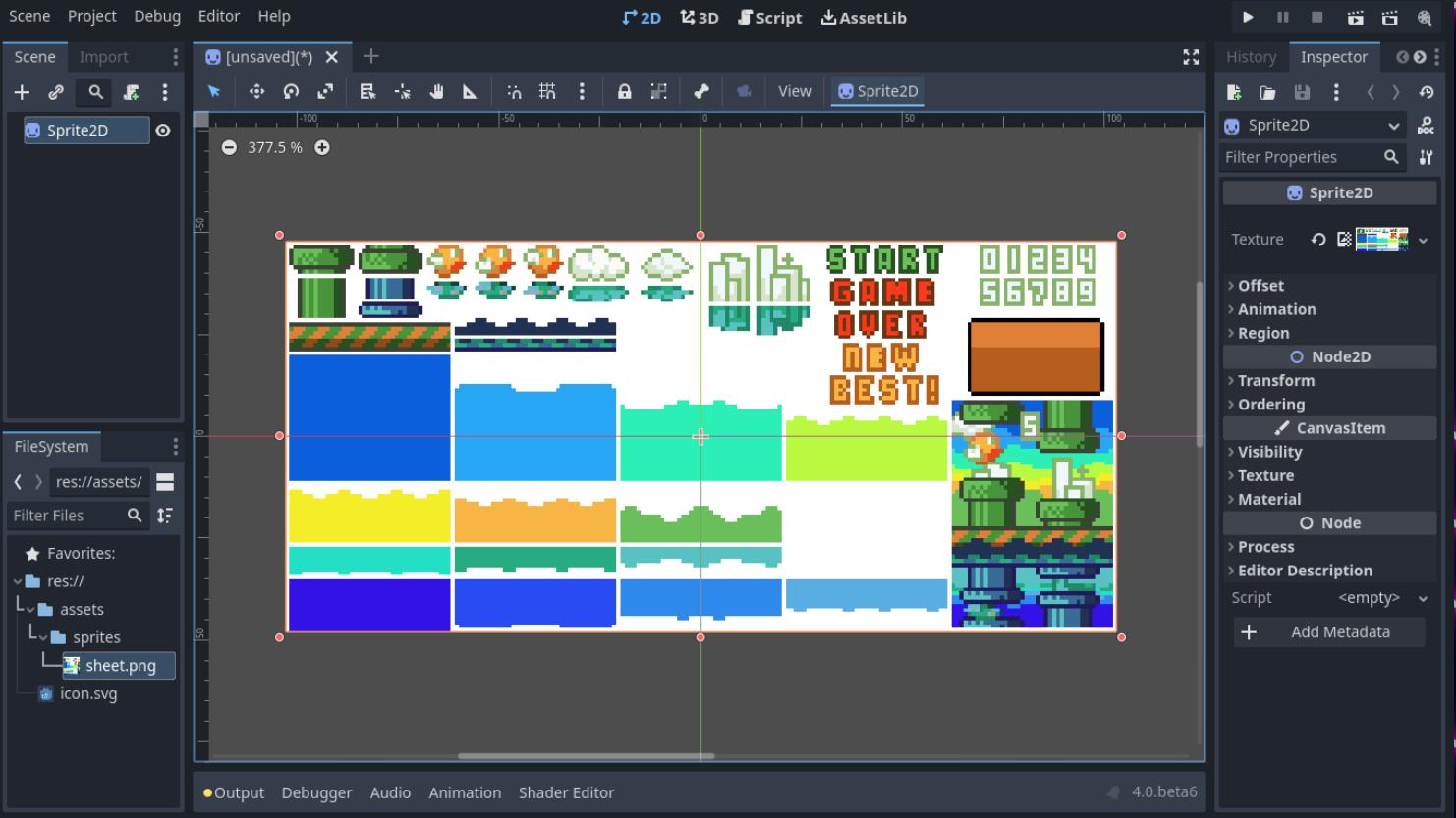


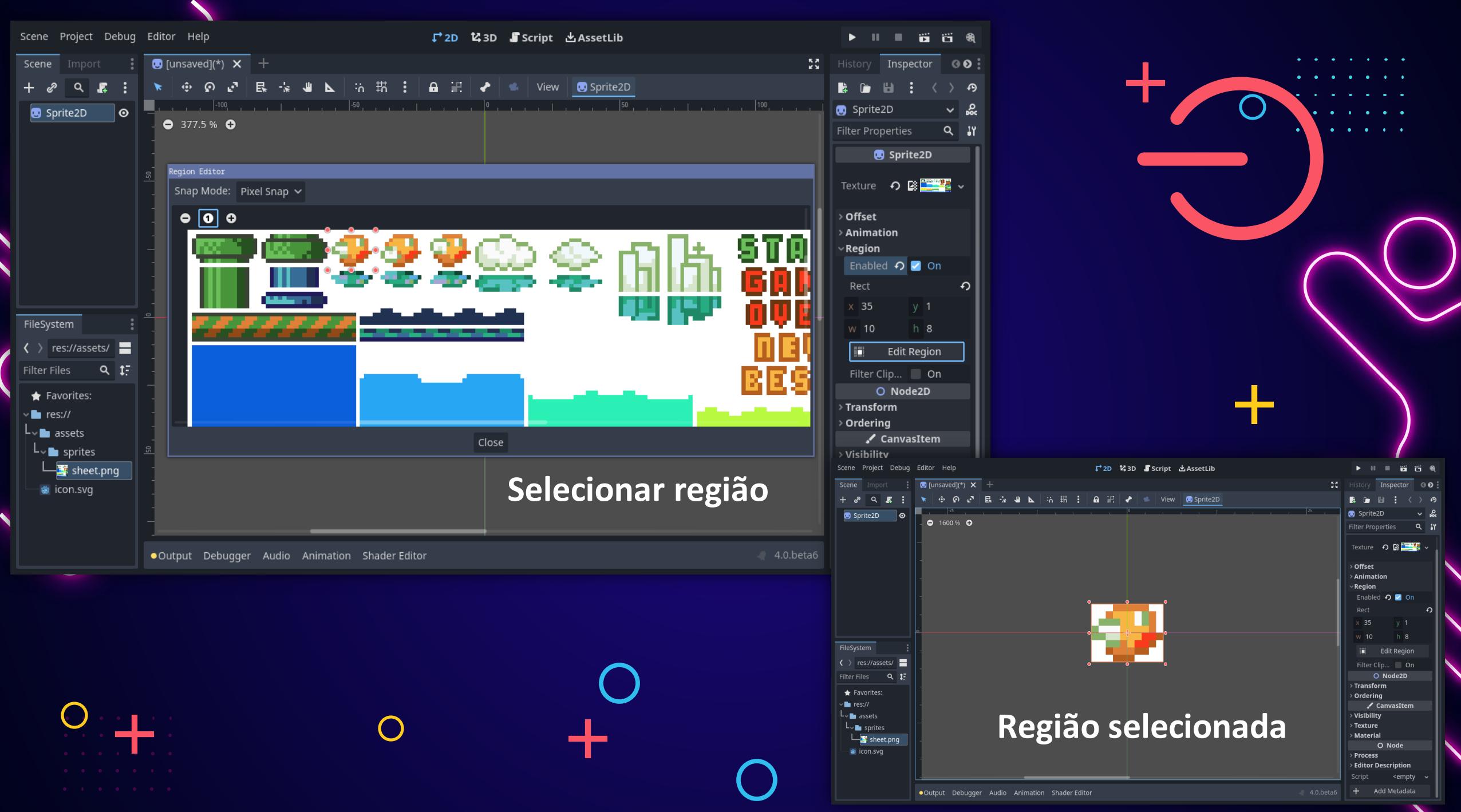
## Mover textura

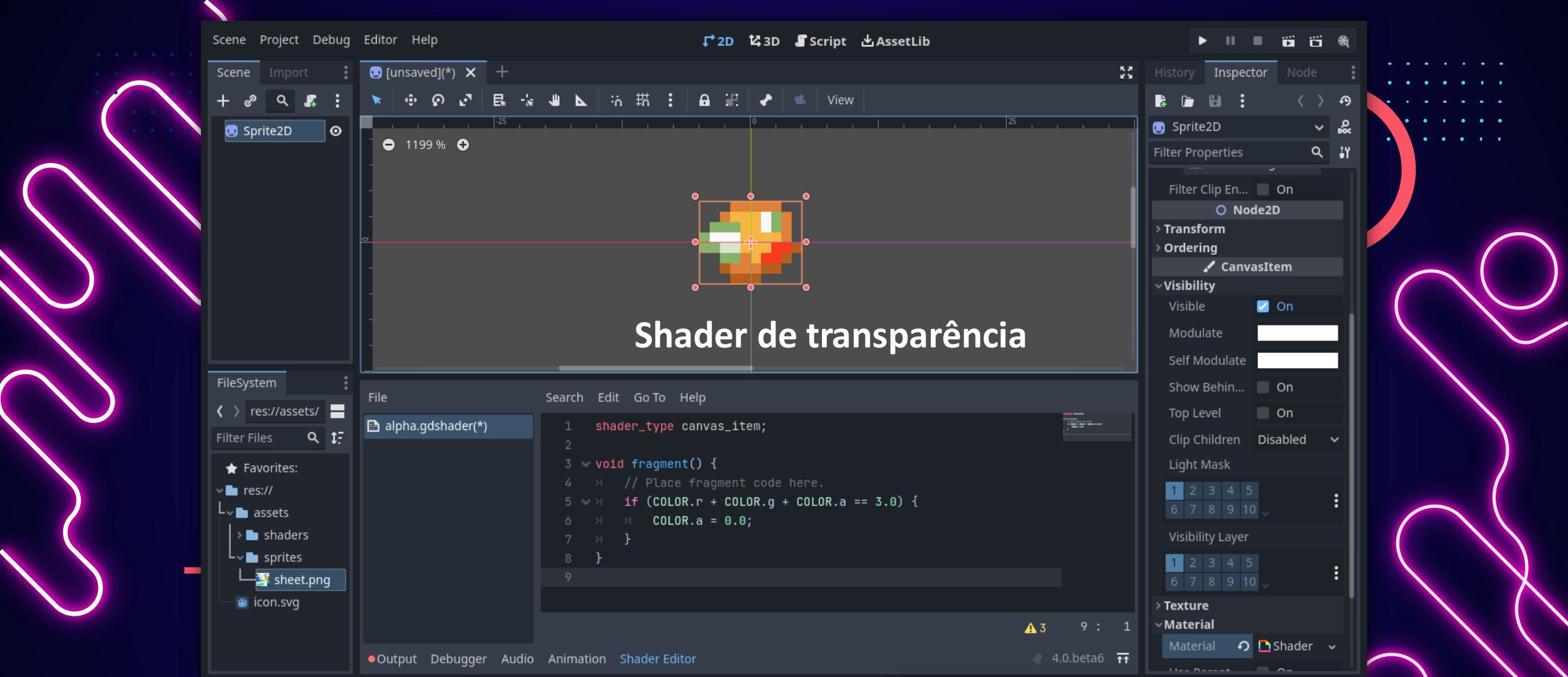




"Nearest"



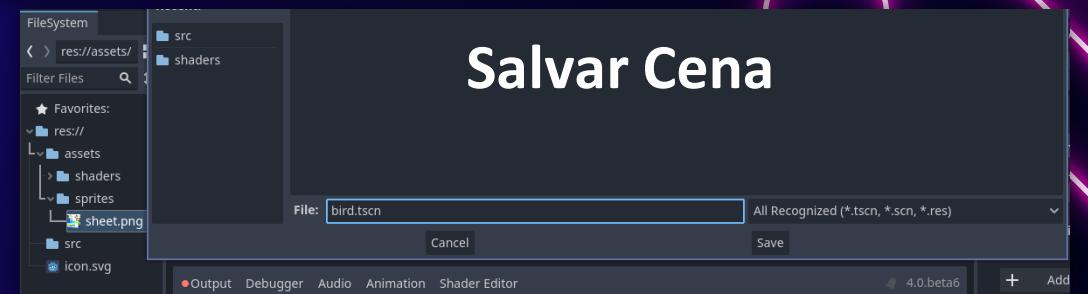


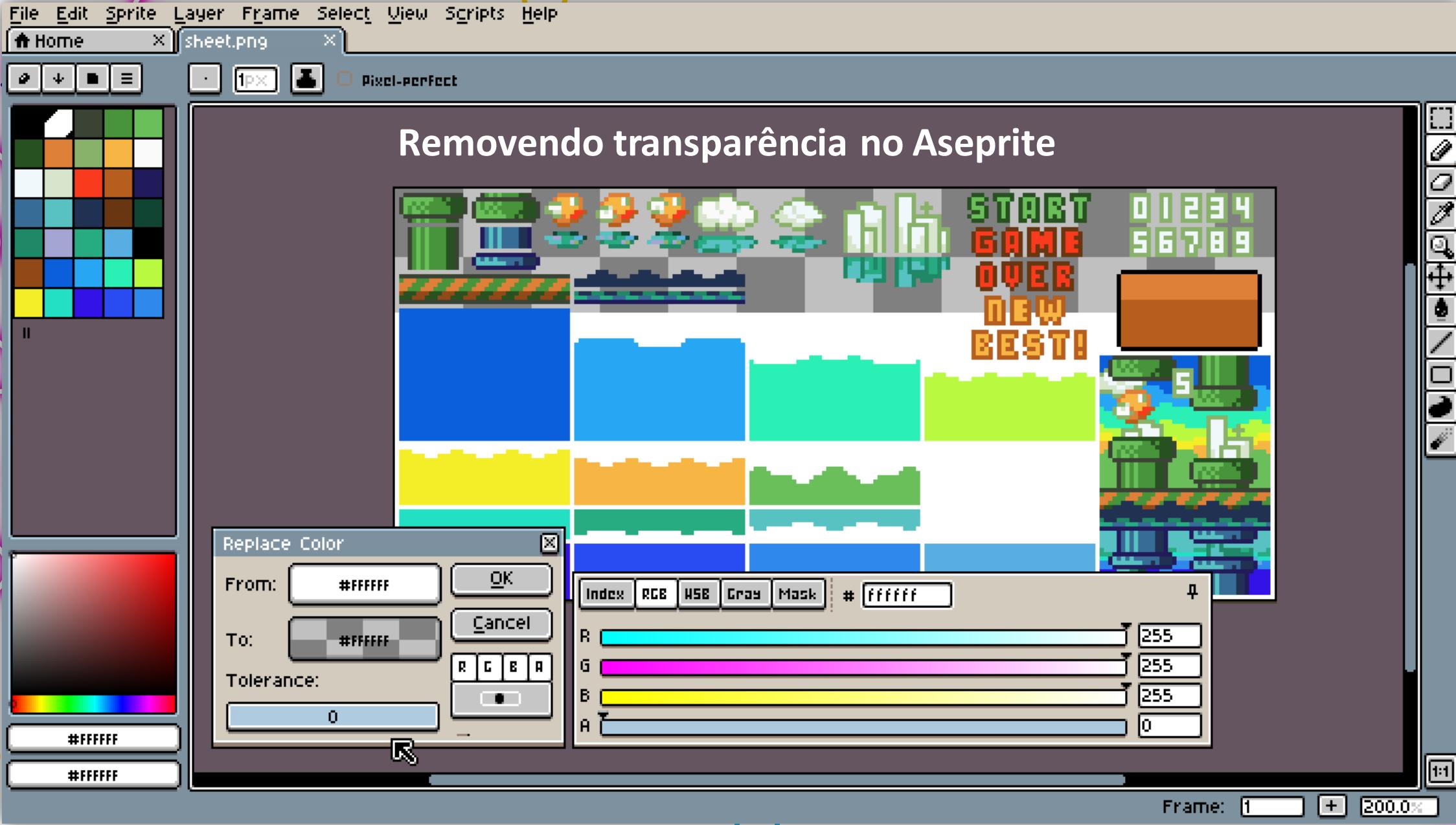


Renomear

Bird

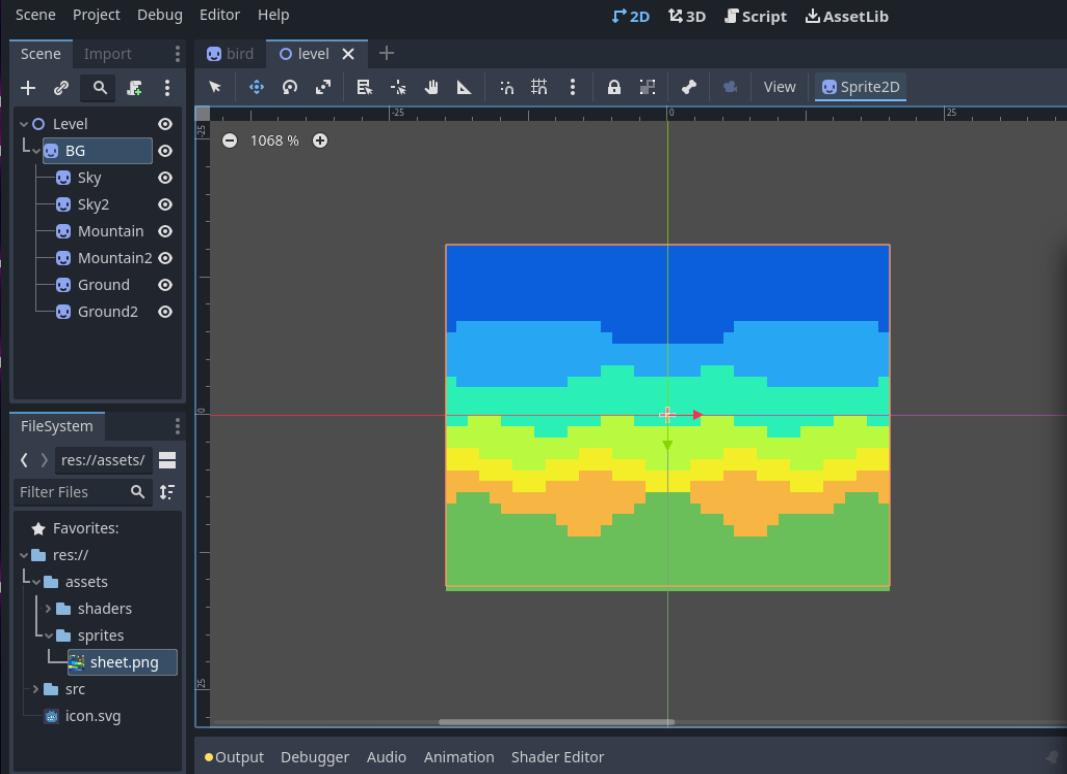
Salvar Cena



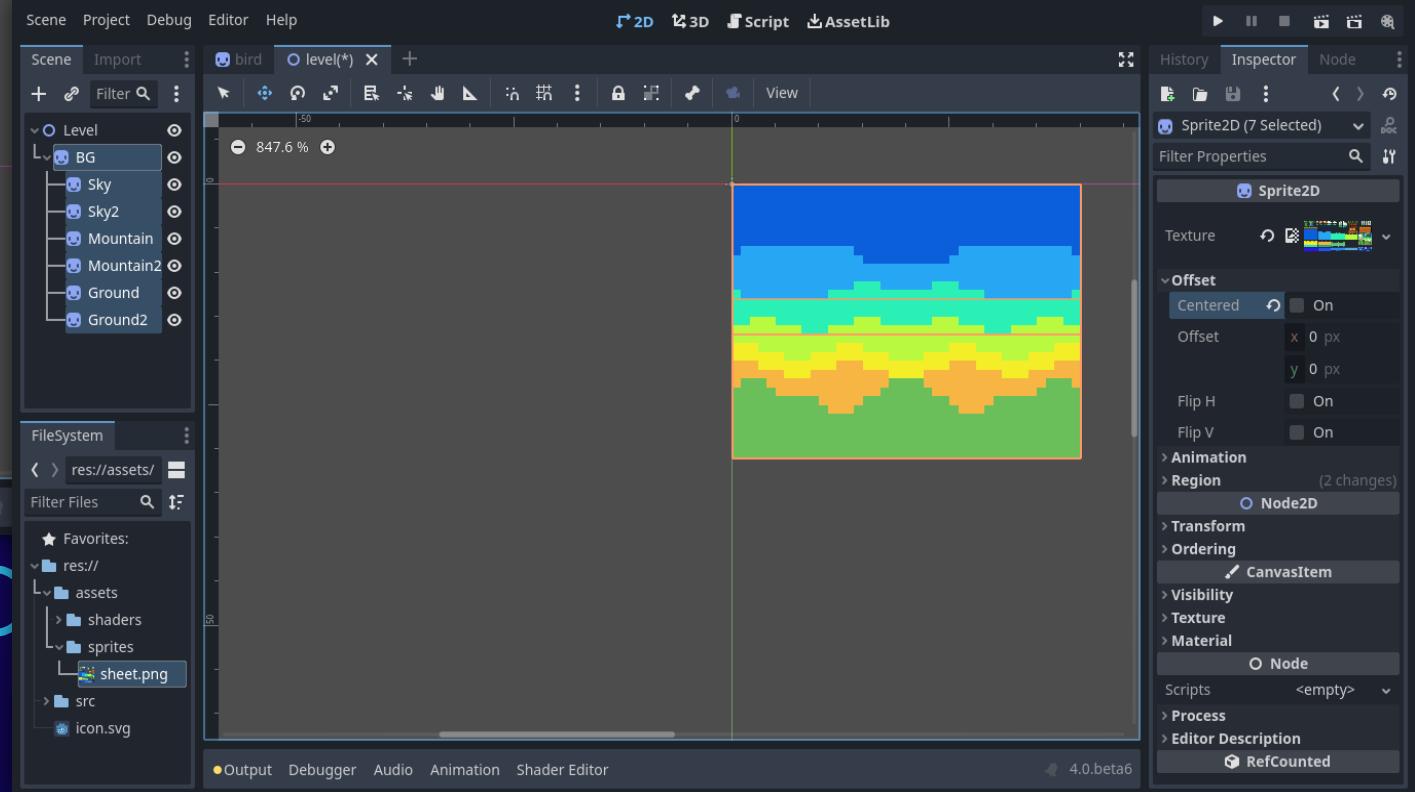
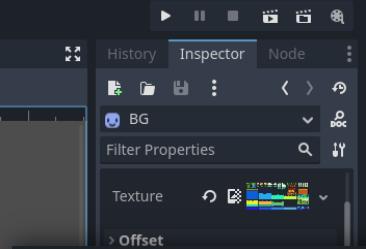




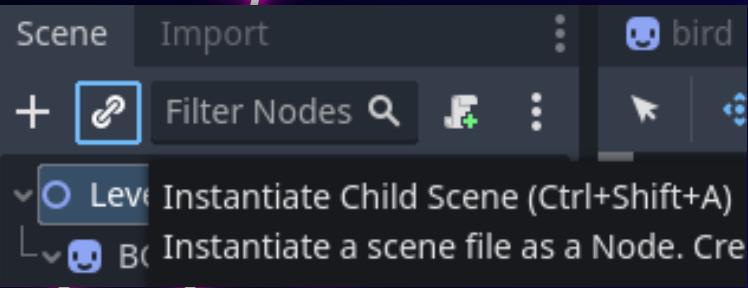
## Nova Cena



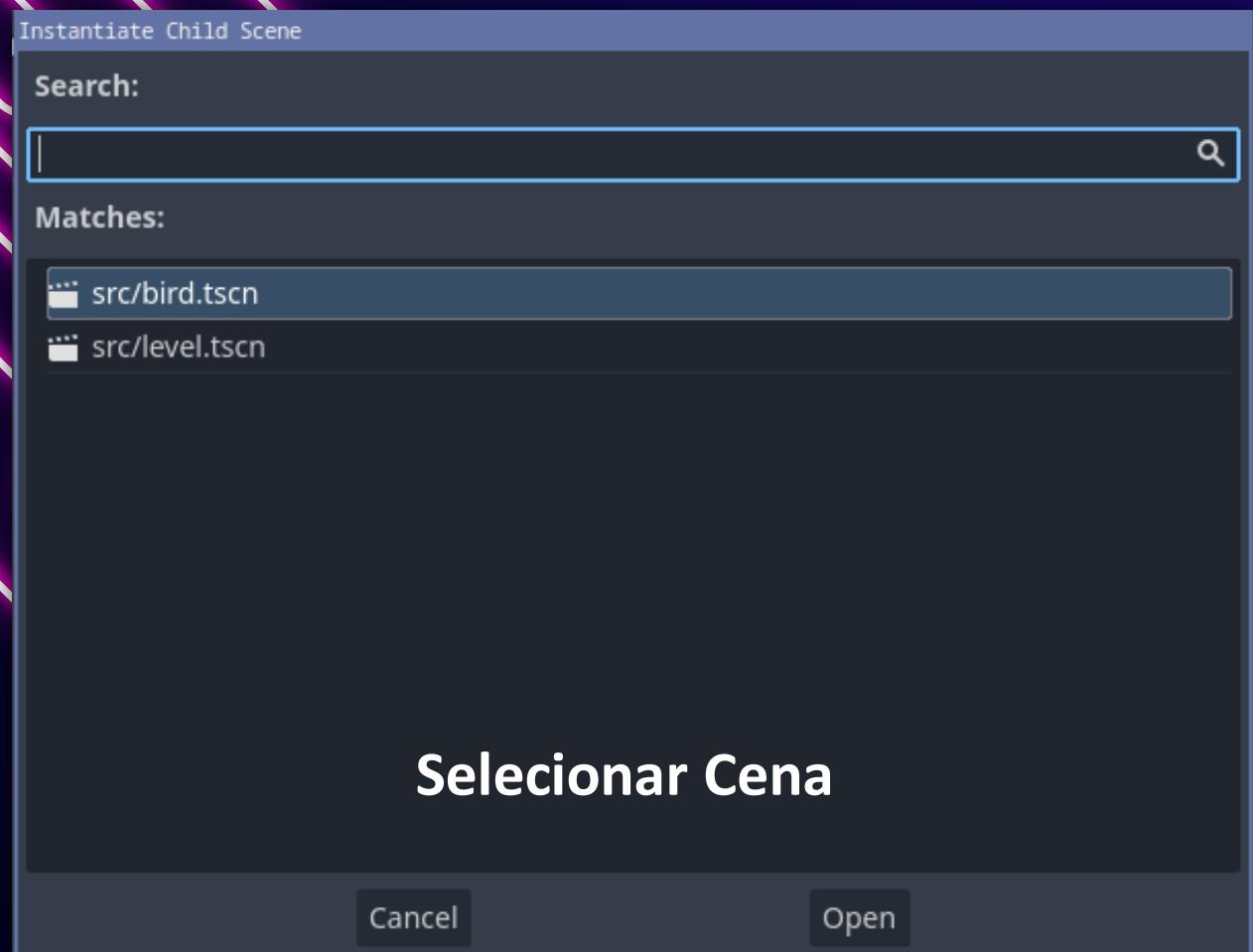
## Background



## Deslocar o centro



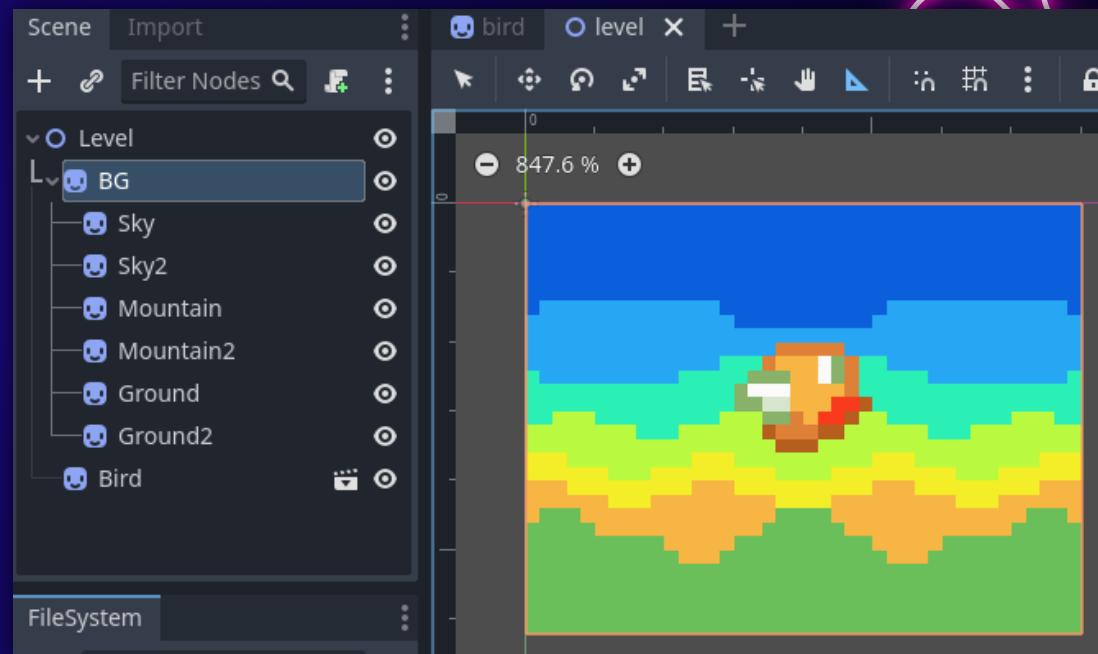
Instanciar cena  
como nó  
filho



Seleccionar Cena

Cancel

Open



Ajustar Cena

Executar cena

Executar projeto

## Selecionar Cena principal

Please Confirm...

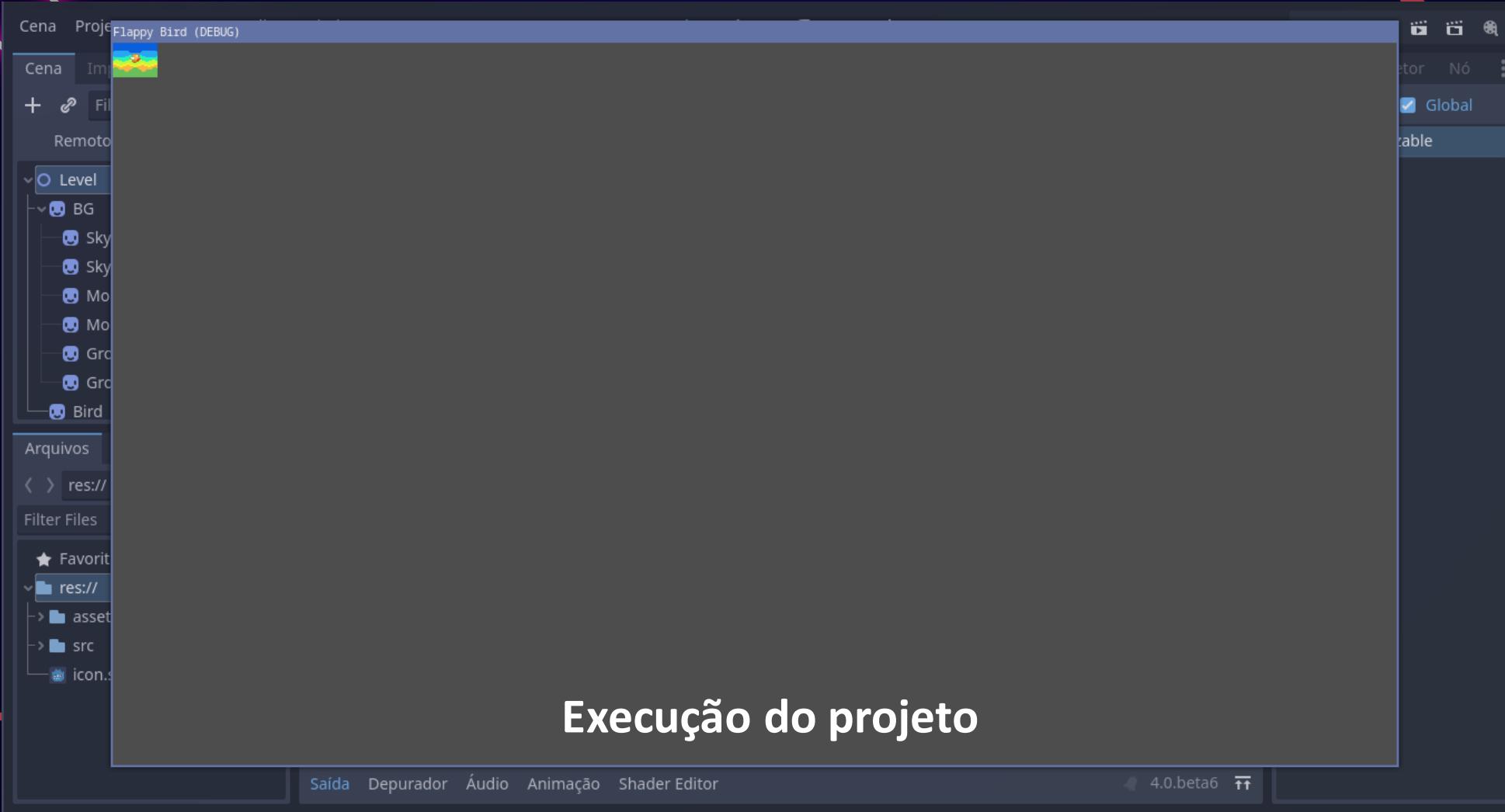
A cena principal não foi definida, selecionar uma?

Você pode alterá-la mais tarde nas "Configurações do Projeto" na categoria 'Application'.

Cancel

Selecionar

Selecionar Atual



Configurações do Projeto (project.godot)

Geral Mapa de Entrada Localização Autoload Shader Globals Plugins Padrões de Importação

Filter Settings

Aplicação

- Configuração
- Rodar
- Imagen de Exibição ao I

Exibição

- Janela
- Cursor do Mouse

Áudio

- Buses

Renderização

- Renderer
- Ambiente
- Anti Aliasing
- Texturas

Física

- Comum
- 2D
- 3D

Tamanho

- Viewport Width: 720
- Viewport Height: 1280
- Modo: Windowed
- Redimensionável: Ativo
- Sem Bordas: Ativo

Portátil (Handheld)

- Orientação: Portrait

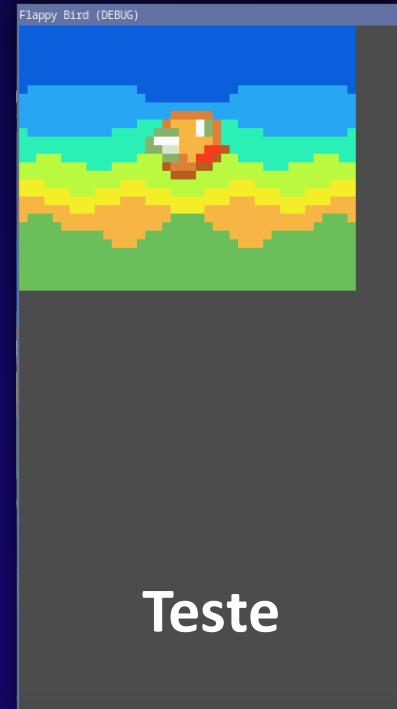
Esticar

- Modo: disabled
- Aspecto: keep
- Escala: 8

Fechar

## Configurações Tamanho de tela

Novas dimensões



Configurações do Projeto (project.godot)

Geral Mapa de Entrada Localização Autoload Shader Globals Plugins Padrões de Importação

Filter Settings

Aplicação

- Configuração
- Rodar
- Imagen de Exibição ao I

Exibição

- Janela
- Cursor do Mouse

Áudio

- Buses

Renderização

- Renderer
- Ambiente
- Anti Aliasing

Tamanho

- Viewport Width: 360
- Viewport Height: 1280/2
- Modo: Windowed
- Redimensionável: Ativo
- Sem Bordas: Ativo

Portátil (Handheld)

- Orientação: Portrait

Esticar

- Modo: viewport
- Aspecto: keep
- Escala: 8

The following table shows which screen resolution is most used in selected countries based on web traffic statistics for Q2 2019.

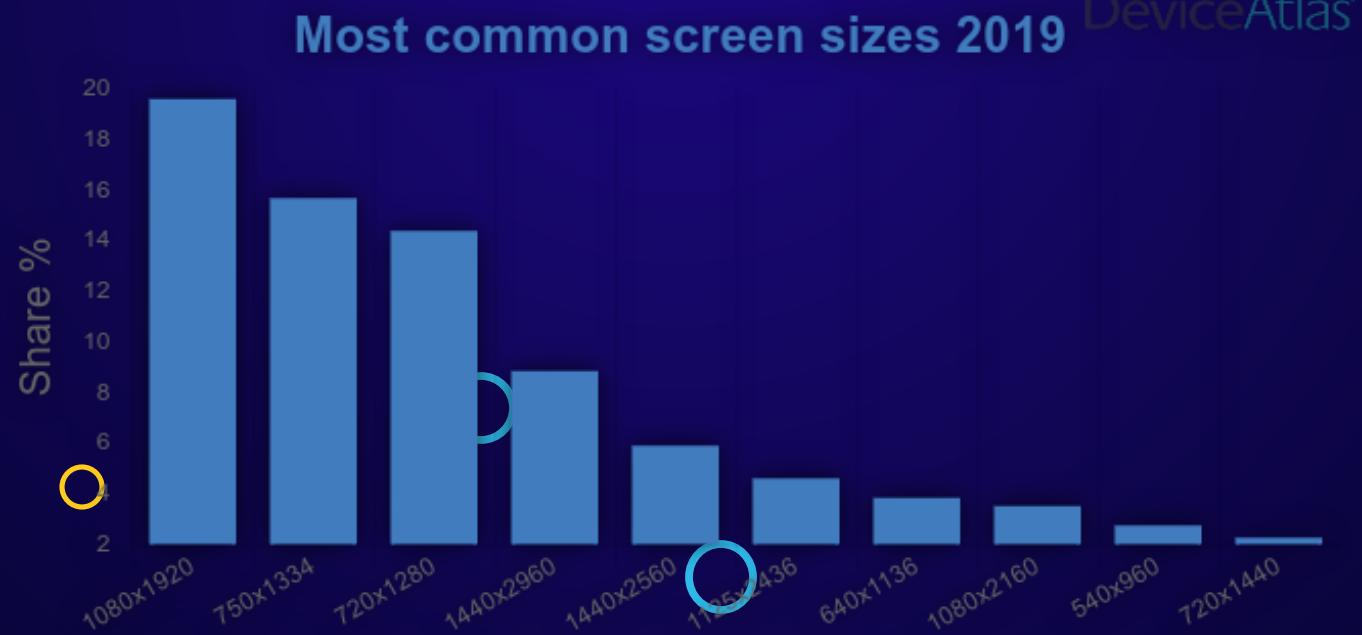
Country	Screen resolution	Share	Change (v Q1)
Argentina	720x1280	33.31%	-1.36%
Australia	750x1334	22.03%	-1.80%
Brazil	720x1280	22.52%	-6.86%
Canada	750x1334	24.77%	-1.33%
Colombia	1080x1920	19.10%	+0.18%
Denmark	750x1334	28.01%	-3.64%
Egypt	720x1280	21.36%	-5.68%
Finland	1080x1920	24.54%	+2.21%
France	750x1334	19.54%	-2.94%
Germany	1080x1920	17.89%	-1.74%

## Resoluções por país

The following table shows which screen resolution is most used in selected countries based on web traffic statistics for Q2 2019.

Country	Screen resolution	Share	Change (v Q1)
Argentina	720x1280	33.31%	-1.36%
Australia	750x1334	22.03%	-1.80%
Brazil	720x1280	22.52%	-6.86%
Canada	750x1334	24.77%	-1.33%
Colombia	1080x1920	19.10%	+0.18%
Denmark	750x1334	28.01%	-3.64%
Egypt	720x1280	21.36%	-5.68%
Finland	1080x1920	24.54%	+2.21%
France	750x1334	19.54%	-2.94%
Germany	1080x1920	17.89%	-1.74%

## Resoluções no Brasil





Parallax



## Criar Novo Node

Favoritos:

Pesquisar:

parallax



Correspondências:

Node

CanvasLayer

└ ParallaxBackground

CanvasItem

Node2D

└ ParallaxLayer

Recente:

Polygon2D

Sprite2D

## Nó ParallaxBackground

Descrição:

**ParallaxBackground:** A node used to create a parallax scrolling background.

Cancel

Criar

## Criar Novo Node

Favoritos:

Pesquisar:

parallax



Correspondências:

Node

└ CanvasLayer

└ ParallaxBackground

CanvasItem

└ Node2D

└ ParallaxLayer

Recente:

ParallaxLayer

ParallaxBackgr...

Polygon2D

Sprite2D

## Nó ParallaxLayer

Descrição:

**ParallaxLayer:** A parallax scrolling layer to be used with [ParallaxBackground](#).

Cancel

Criar

Cena Importar

+

Filter Nodes



bird

level X



Level

ParallaxBackground

ParallaxLayer

BG

Floor

Grass

Grass2

Sky

Cloud

Cloud2

Bird

Arquivos

&lt; &gt; res://

Filter Files



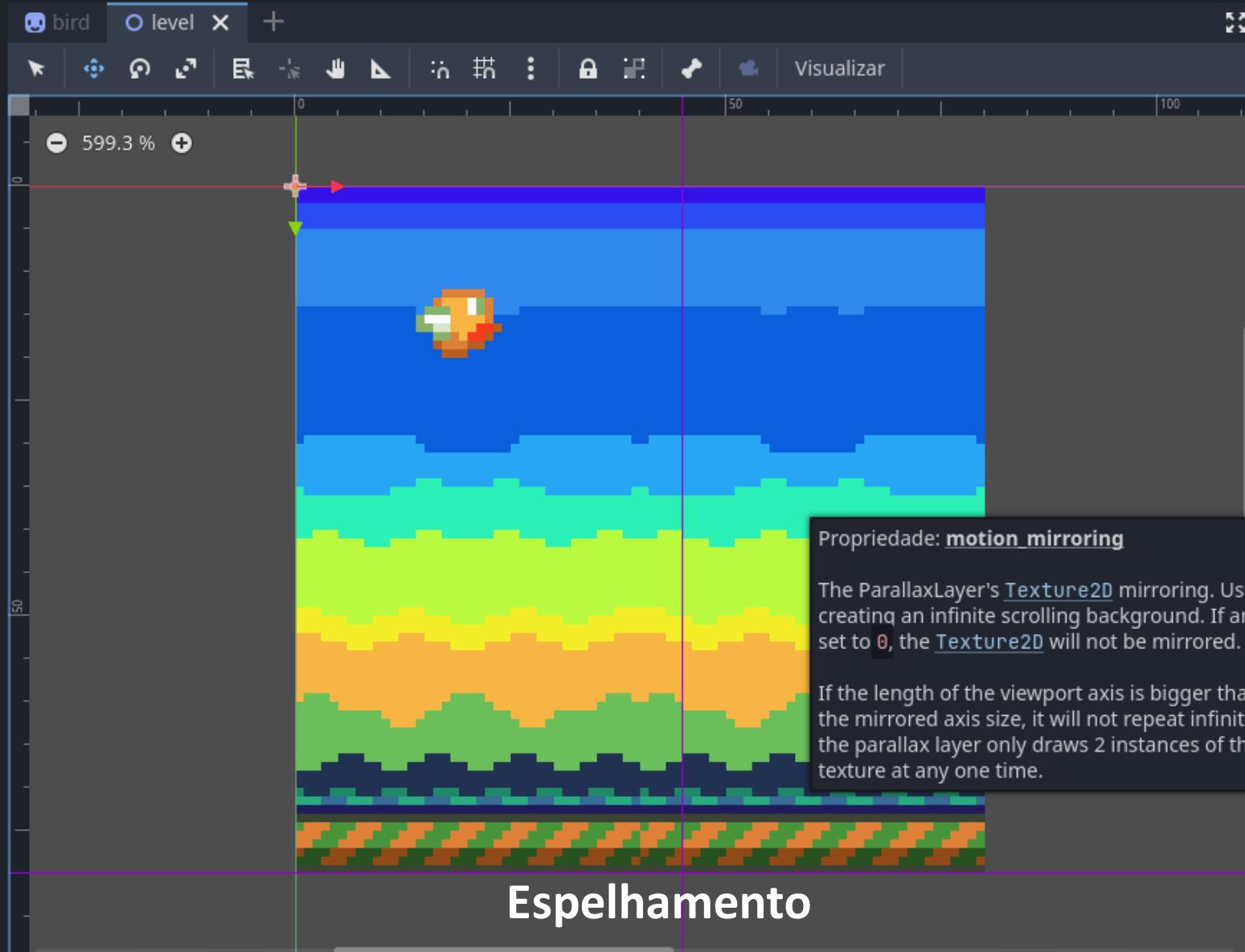
Favoritos:

res://

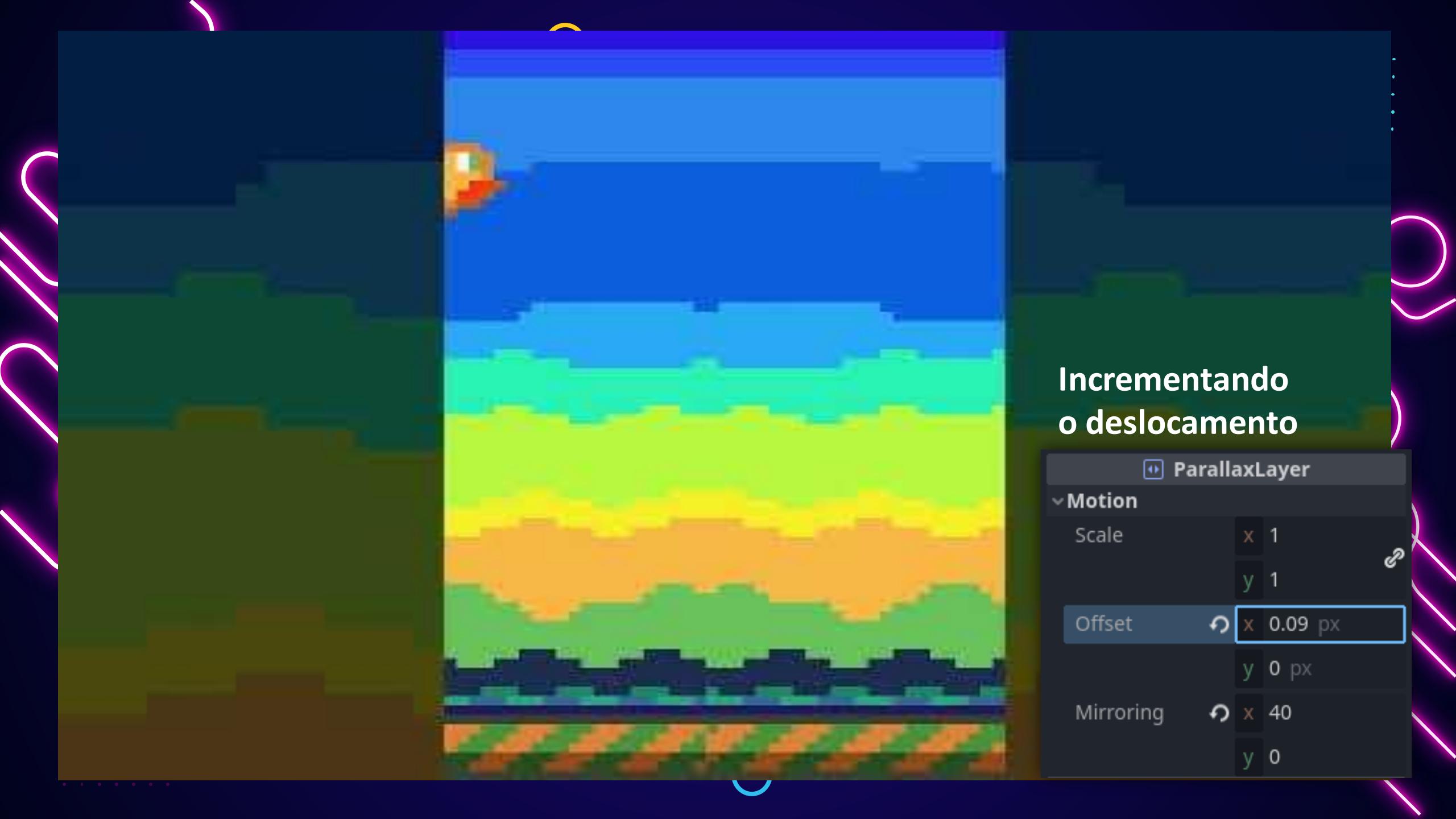
assets

src

icon.svg



Espelhamento



Incrementando  
o deslocamento

ParallaxLayer

Motion

Scale

x 1

y 1

Offset

x 0.09 px

y 0 px

Mirroring

x 40

y 0

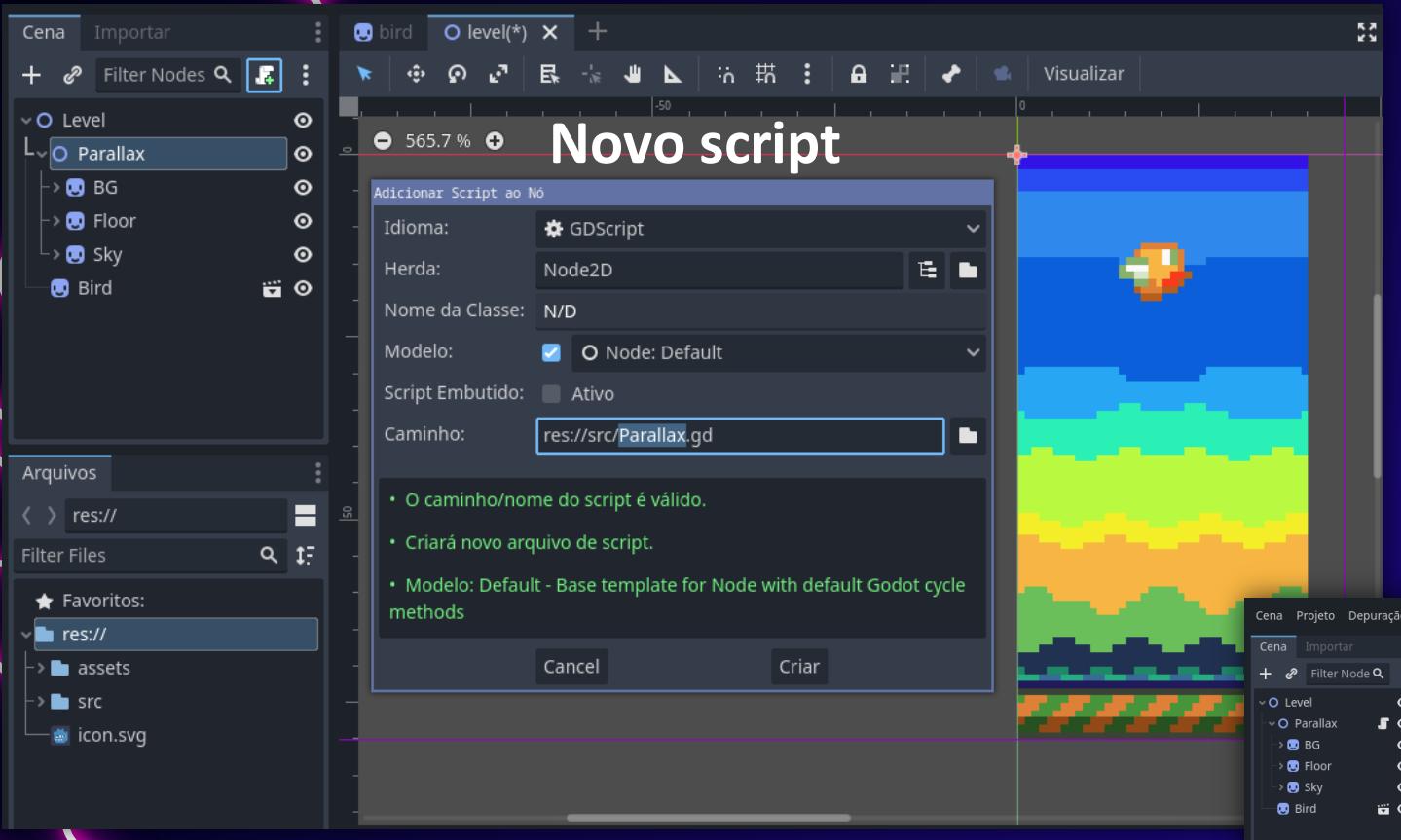


**MORE**

**PARALLAX!**

Parallax: Uma  
abordagem  
diferente





```

1  extends Node2D
2
3  var bg
4  var floor
5  var sky
6
7  # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.
8  func _ready():
9    bg = get_node("BG") # Obtém um nó, via código, por nome
10   floor = $Floor      # Atalho para `get_node`
11   sky = get_child(2)  # Obtém um nó filho, por índice
12

```

**Método get\_node()**

## Editor de texto

**Cena** Projeto Depuração Editor Ajuda

bird level(\*)

Arquivo Editar Pesquisar Ir Para Depuração Documentação Online Pesquisar Ajuda

2D 3D Script AssetLib

Arquivos res:// Filter Scripts :  
Favoritos:  
res://  
assets  
src  
icon.svg

```

1  extends Node2D
2
3
4  # Called when the node enters the scene tree for the first time.
5  func _ready():
6    pass # Replace with function body.
7
8
9  # Called every frame. 'delta' is the elapsed time since the previous frame.
10 func _process(delta):
11    pass
12

```

Parallax.gd Filter Methods :  
\_ready  
\_process

Definir motion\_offset  
Definir motion\_offset  
Remover Nód(s)  
Desfazer: Remover Nód(s)  
Definir motion\_offset  
Definir motion\_offset  
Definir motion\_offset  
Definir motion\_offset  
Remover Nód(s)  
Alternar Visibilidade

Filter Messages

Saída Depurador (1) Resultados de Pesquisa Áudio Animação Shader Editor 4.0.beta6

```
1  extends Node2D
2
3  var bg
4  var floor
5  var sky
6
7  # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.
8  ↳ func _ready():
9    >I  bg = get_node("BG") # Obtém um nó, via código, por nome
10   >I  floor = $Floor      # Atalho para `get_node`
11   >I  sky = get_child(2)  # Obtém um nó filho, por índice
12
13  var bg_copy = bg.duplicate() # duplica uma instância de Nó
14  var floor_copy = floor.duplicate()
15  var sky_copy = sky.duplicate()
16
```

## Ordem de precedência (exemplo com erro)

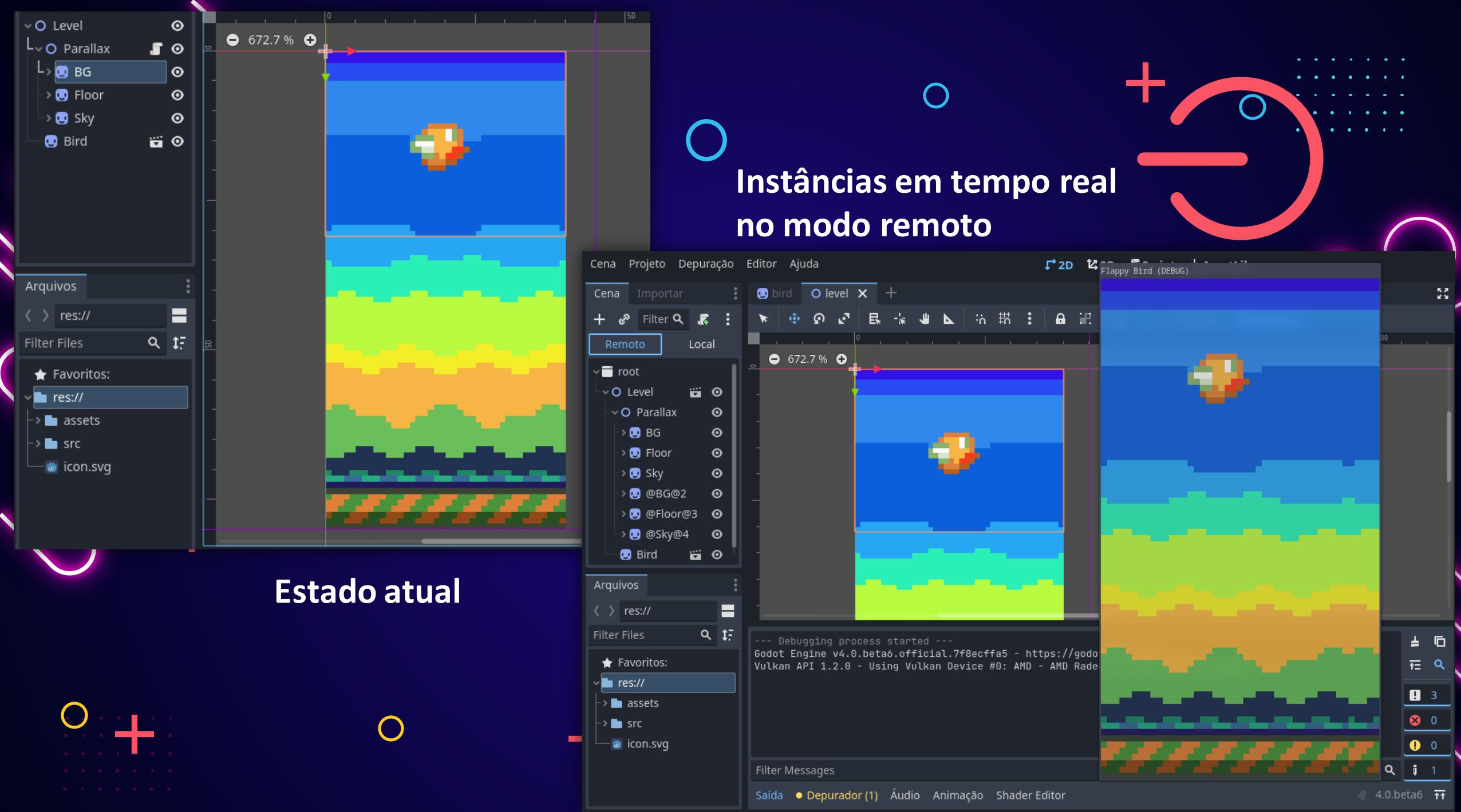
```
1  extends Node2D
2
3  @onready var bg = get_node("BG") # Obtém um nó, via código, por nome
4  @onready var floor = $Floor      # Atalho para `get_node`
5  @onready var sky = get_child(2)  # Obtém um nó filho, por índice
6
7  var bg_copy
8  var floor_copy
9  var sky_copy
10
11 # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.
12 ↳ func _ready():
13   >I  bg_copy = bg.duplicate() # duplica uma instância de Nó
14   >I  floor_copy = floor.duplicate()
15   >I  sky_copy = sky.duplicate()
16
```

@onready +

```
1  extends Node2D
2
3  @onready var bg = get_node("BG") # Obtém um nó, via código, por nome
4  @onready var floor = $Floor      # Atalho para `get_node`
5  @onready var sky = get_child(2)  # Obtém um nó filho, por índice
6
7  var bg_copy
8  var floor_copy
9  var sky_copy
10
11 # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.
12 func _ready():
13     bg_copy = bg.duplicate() # duplica uma instância de Nó
14     floor_copy = floor.duplicate()
15     sky_copy = sky.duplicate()
16
17     bg_copy.position.x = (bg as Sprite2D).region_rect.size.x
18
```

## Intellisense

```
1  extends Node2D
2
3  @onready var bg = get_node("BG") # Obtém um nó, via código, por nome
4  @onready var floor = $Floor      # Atalho para `get_node`
5  @onready var sky = get_child(2)  # Obtém um nó filho, por índice
6
7  var bg_copy
8  var floor_copy
9  var sky_copy
10
11 # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.
12 func _ready():
13     bg_copy = bg.duplicate() # duplica uma instância de Nó
14     floor_copy = floor.duplicate()
15     sky_copy = sky.duplicate()
16
17     bg_copy.position.x = (bg as Sprite2D).region_rect.size.x
18     floor_copy.position.x = (floor as Sprite2D).region_rect.size.x
19     sky_copy.position.x = (sky as Sprite2D).region_rect.size.x
20
21     self.add_child(bg_copy) # Adiciona uma instância de Nó como filho
22     add_child(floor_copy)  # o nó adicionado deve ser órfão antes da chamada
23     add_child(sky_copy)
24
```



Estado atual

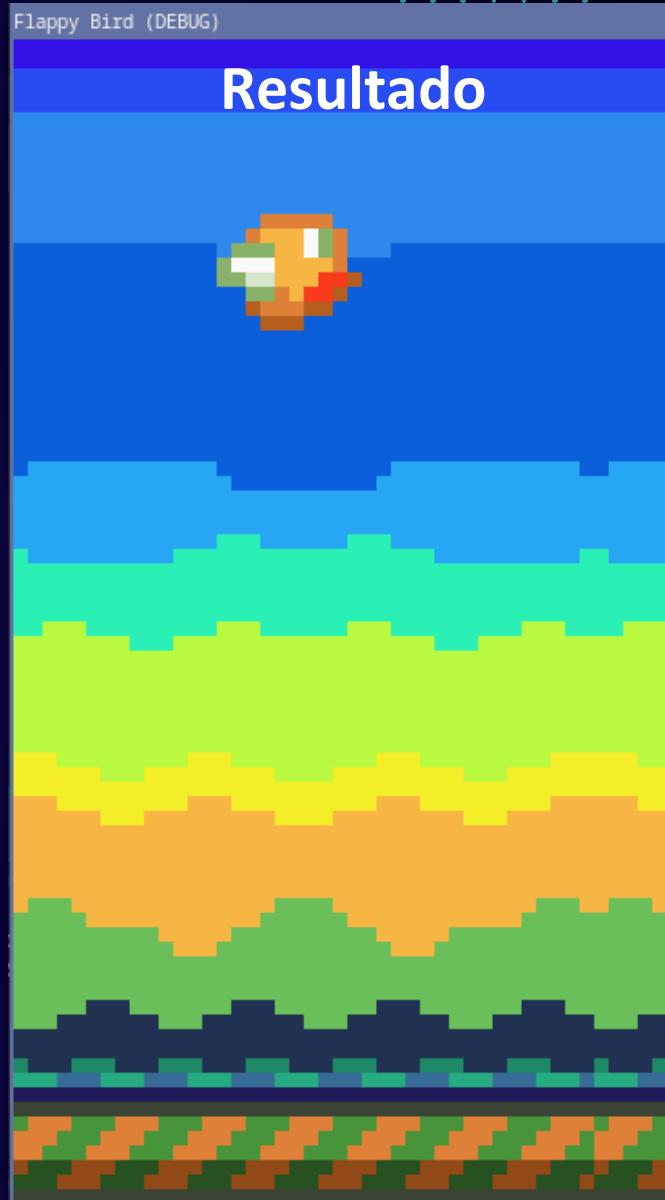
Instâncias em tempo real  
no modo remoto

movement



# Movimentação

```
25 var speed = 1.0 # Taxa de velocidade (rapidez)
26
27 # Called every frame. 'delta' is the elapsed time since the previous frame.
28 func _process(delta):
29     var velocity_x = speed * delta # Velocidade atual do movimento no eixo-x
30
31 # Move os Sprites com base na velocidade atual
32 bg.position.x = bg.position.x - velocity_x
33 sky.position.x = sky.position.x - velocity_x
34 floor.position.x = floor.position.x - velocity_x
35 bg_copy.position.x -= velocity_x
36 sky_copy.position.x -= velocity_x
37 floor_copy.position.x -= velocity_x
38
39 speed += delta # Aumenta a velocidade progressivamente|
```





```

1  extends Node2D
2
3  @onready var bg = get_node("BG") # Obtém um nó, via código, por nome
4  @onready var floor = $Floor      # Atalho para `get_node`
5  @onready var sky = get_child(2)  # Obtém um nó filho, por índice
6
7  @onready var bg_copy = bg.duplicate() # duplica uma instância de Nó
8  @onready var floor_copy = floor.duplicate()
9  @onready var sky_copy = sky.duplicate()
10
11 # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.
12 func _ready():
13     bg_copy.position.x = (bg as Sprite2D).region_rect.size.x
14     floor_copy.position.x = (floor as Sprite2D).region_rect.size.x
15     sky_copy.position.x = (sky as Sprite2D).region_rect.size.x
16
17     self.add_child(bg_copy) # Adiciona uma instância de Nó como filho
18     add_child(floor_copy) # o nó adicionado deve ser órfão antes da chamada
19     add_child(sky_copy)

```

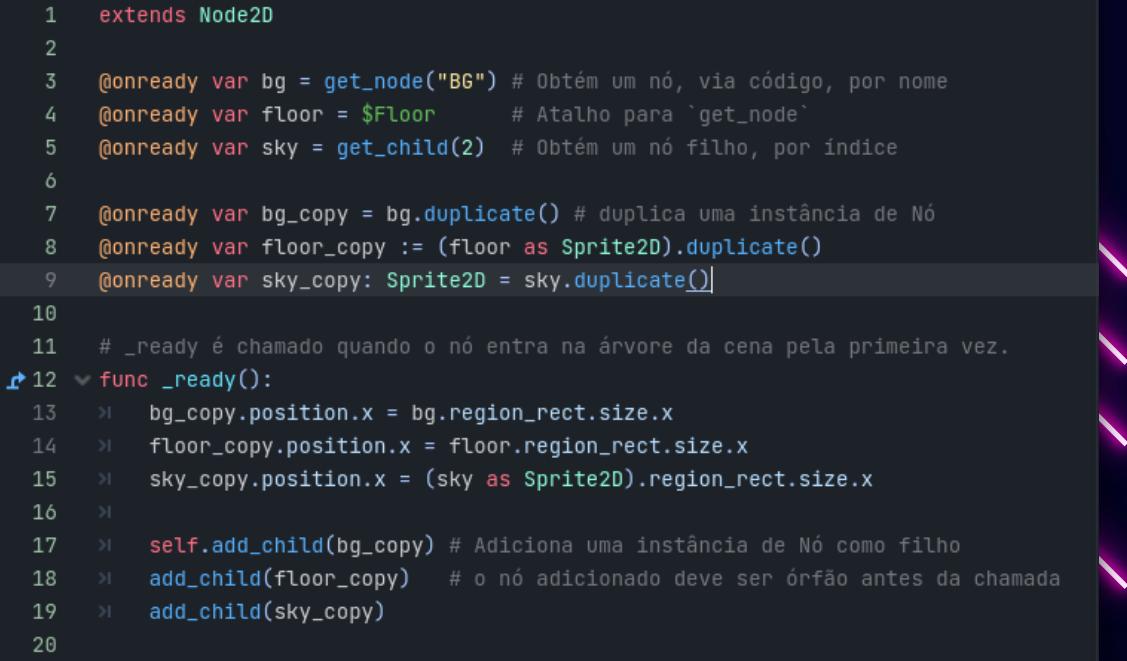
## Ordem de atribuição

```

3  @onready var bg: Sprite2D = get_node("BG") # Obtém um nó, via código, por nome
4  @onready var floor: Sprite2D = $Floor      # Atalho para `get_node`
5  @onready var sky: Sprite2D = get_child(2)  # Obtém um nó filho, por índice
6
7  @onready var bg_copy: Sprite2D = bg.duplicate() # duplica uma instância de Nó
8  @onready var floor_copy: Sprite2D = floor.duplicate()
9  @onready var sky_copy: Sprite2D = sky.duplicate()
10
11 # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.
12 func _ready():
13     bg_copy.position.x = bg.region_rect.size.x
14     floor_copy.position.x = floor.region_rect.size.x
15     sky_copy.position.x = sky.region_rect.size.x
16
17     self.add_child(bg_copy) # Adiciona uma instância de Nó como filho
18     add_child(floor_copy) # o nó adicionado deve ser órfão antes da chamada
19     add_child(sky_copy)
20
21 var speed: float = 1.0 # Taxa de velocidade (rapidez)
22
23 # Called every frame. 'delta' is the elapsed time since the previous frame.
24 func _process(delta):
25     var velocity_x: float # Velocidade atual do movimento no eixo-x
26     velocity_x = speed * delta
27

```

## Refatorando



```

1  extends Node2D
2
3  @onready var bg = get_node("BG") # Obtém um nó, via código, por nome
4  @onready var floor = $Floor      # Atalho para `get_node`
5  @onready var sky = get_child(2)  # Obtém um nó filho, por índice
6
7  @onready var bg_copy = bg.duplicate() # duplica uma instância de Nó
8  @onready var floor_copy := (floor as Sprite2D).duplicate()
9  @onready var sky_copy: Sprite2D = sky.duplicate()
10
11 # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.
12 func _ready():
13     bg_copy.position.x = bg.region_rect.size.x
14     floor_copy.position.x = floor.region_rect.size.x
15     sky_copy.position.x = (sky as Sprite2D).region_rect.size.x
16
17     self.add_child(bg_copy) # Adiciona uma instância de Nó como filho
18     add_child(floor_copy) # o nó adicionado deve ser órfão antes da chamada
19     add_child(sky_copy)
20

```

## Coerção de tipo

```
23 # Chamado a cada frame. 'delta' é o tempo passado desde o último frame.  
24 func _process(delta):  
25     # Velocidade atual do movimento no eixo-x  
26     var velocity_x: float = speed * delta  
27     # Move os Sprites com base na velocidade atual  
28     bg.position.x -= velocity_x  
29     sky.position.x -= velocity_x  
30     floor.position.x -= velocity_x  
31     bg_copy.position.x -= velocity_x  
32     sky_copy.position.x -= velocity_x  
33     floor_copy.position.x -= velocity_x  
34  
35     if bg.position.x + bg.region_rect.size.x < 0:  
36         bg.position.x = bg_copy.position.x + bg_copy.region_rect.size.x  
37     if sky.position.x + sky.region_rect.size.x < 0:  
38         sky.position.x = sky_copy.position.x + sky_copy.region_rect.size.x  
39     if floor.position.x + floor.region_rect.size.x < 0:  
40         floor.position.x = floor_copy.position.x + floor_copy.region_rect.size.x  
41  
42     if bg_copy.position.x + bg_copy.region_rect.size.x < 0:  
43         bg_copy.position.x = bg.position.x + bg.region_rect.size.x  
44     if sky_copy.position.x + sky_copy.region_rect.size.x < 0:  
45         sky_copy.position.x = sky.position.x + sky.region_rect.size.x  
46     if floor_copy.position.x + floor_copy.region_rect.size.x < 0:  
47         floor_copy.position.x = floor.position.x + floor.region_rect.size.x  
48  
49     speed += delta # Aumenta a velocidade progressivamente  
50
```

## Reposicionando BG (Back Ground)

## Repositioning result



```
1  extends Node2D
2
3  ## Retorna a lista de filhos do nó
4  ## Assume que todos os filhos são Sprites
5  @onready var children: Array[Sprite2D] = get_children()
6
7  # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.
8  func _ready():
9    var copies: Array[Sprite2D] = []
10
11   for child in children:
12     var copy: Sprite2D = child.duplicate()      # Duplica um filho
13     child.position.x = child.region_rect.size.x # Posiciona a cópia
14     add_child(copy) # Adiciona a cópia (uma instância de Sprite) como filho
15     # o nó adicionado deve ser órfão antes da chamada
16
17   children.append_array(copies) # Adiciona as cópias como filhos
18
```

## children array



## Rafatorando

```
21  # Chamado a cada frame. 'delta' é o tempo passado desde o último frame.
22  func _process(delta):
23    # Velocidade atual do movimento no eixo-x
24    var velocity_x: float = speed * delta
25    # Move os Sprites com base na velocidade atual
26
27    for child in children:
28      child.position.x -= velocity_x
29
30    if child.position.x + ????.region_rect.size.x < 0:
31      child.position.x = ????.position.x + ????.region_rect.size.x
32
33    speed += delta # Aumenta a velocidade progressivamente
34
```

## children processing

```
23 var speed: float = 1.0 # Taxa de velocidade (rapidez)
24
25 # Chamado a cada frame. 'delta' é o tempo passado desde o último frame.
26 func _process(delta):
27     # Velocidade atual do movimento no eixo-x
28     var velocity_x: float = speed * delta
29     # Move os Sprites com base na velocidade atual
30
31     for child in children:
32         var twin: Sprite2D = child.get_meta("twin")
33         child.position.x -= velocity_x
34
35         if child.position.x + twin.region_rect.size.x < 0:
36             child.position.x = twin.position.x + twin.region_rect.size.x
37
38     speed += delta # Aumenta a velocidade progressivamente
39
```

## Usando nó "twin" como metadado

## Rafatorando

```
7     # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.
8     func _ready():
9         var copies: Array[Sprite2D] = []
10
11        for child in children:
12            var copy: Sprite2D = child.duplicate()          # Duplica um filho
13
14            child.position.x = child.region_rect.size.x # Posiciona a cópia
15            add_child(copy) # Adiciona a cópia (uma instância de Sprite) como filho
16
17            # o nó adicionado deve ser órfão antes da chamada
18
19            child.set_meta("twin", copy) # Salva a "instância gêmea" em metadados
20            copy.set_meta("twin", child)
```

bird level +

Arquivo Editar Pesquisar Ir Para Depuração

Filter Scripts

Parallax.gd

Object

TreeItem

Parallax.gd

Filter Methods

\_ready

\_process

```
1 extends Node2D
2
3 ## Retorna a lista de filhos do nó
4 ## Assume que todos os filhos são Sprites
5 @onready var children: Array[Sprite2D] = get_children()
6
7 # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.
8 func _ready():
9     var copies: Array[Sprite2D] = []
10
11 for child in children:
12     var copy: Sprite2D = child.duplicate()      # Duplica um filho
13
14     child.position.x = child.region_rect.size.x # Posiciona a cópia
```

## Erro de tipo

Depurador • Erros (1) Profilador Visual Profiler Monitores Memória de Vídeo Misc Perfis de rede

Trying to assign a typed array with an array of different type.'

Pilha de Quadros  
0 - res://src/Parallax.gd:5 - at function: \_ready

Filter Stack Variables

## Members

self  
children  
speed

Saída • Depurador (1) Resultados de Pesquisa Áudio Animação Shader Editor



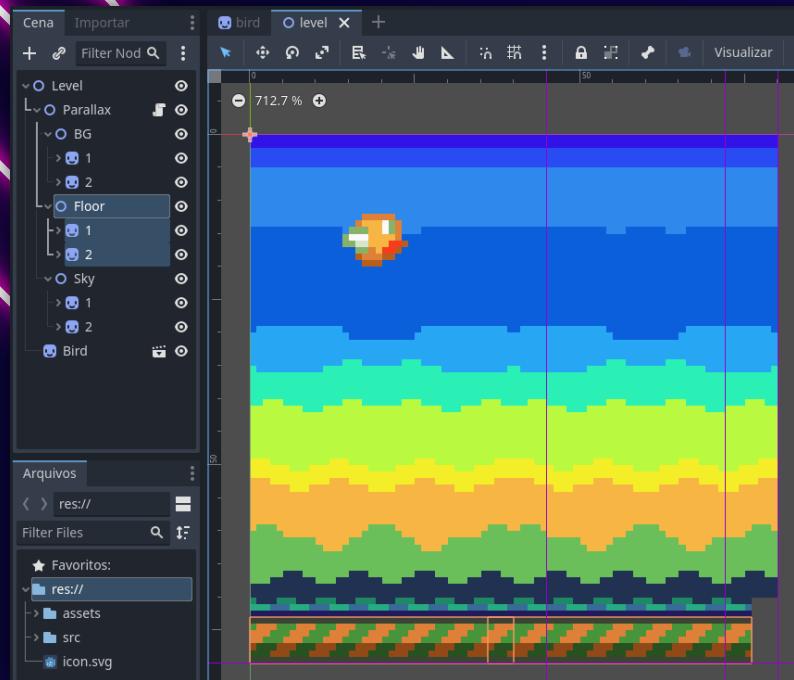
## Consertando erro

```
1 extends Node2D
2
3 ## Retorna a lista de filhos do nó
4 ## Assume que todos os filhos são Sprites
5 @onready var children: Array[Sprite2D] = []
6
7 # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.
8 func _ready():
9
10 for child in get_children():
11     var copy: Sprite2D = child.duplicate()      # Duplica um filho
12
13     child.position.x = child.region_rect.size.x # Posiciona a cópia
14     add_child(copy) # Adiciona a cópia (uma instância de Sprite) como filho
15
16     # o nó adicionado deve ser órfão antes da chamada
17
18     child.set_meta("twin", copy) # Salva a "instância gêmea" em metadados
19     copy.set_meta("twin", child)
20
21     children.append(child) # Salva os filhos na lista `children` |
```

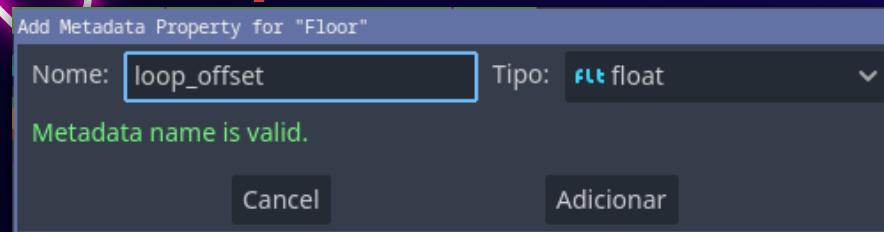
Refactor result



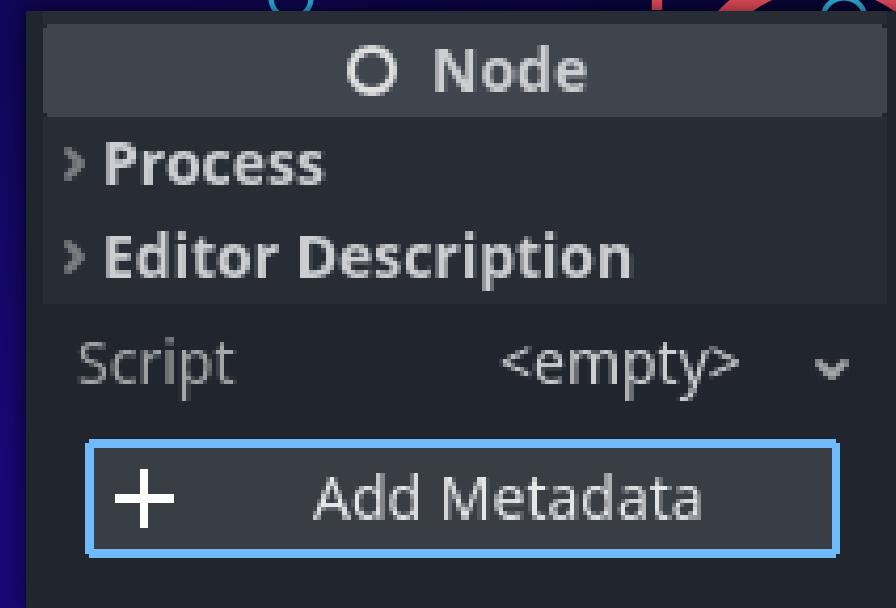
## Duplicando



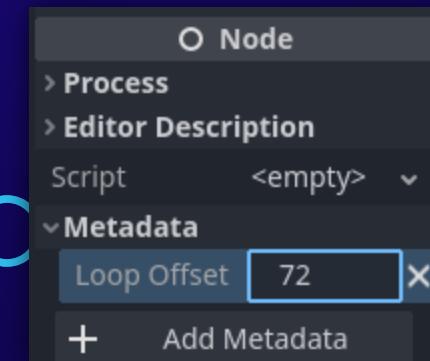
## Metadata



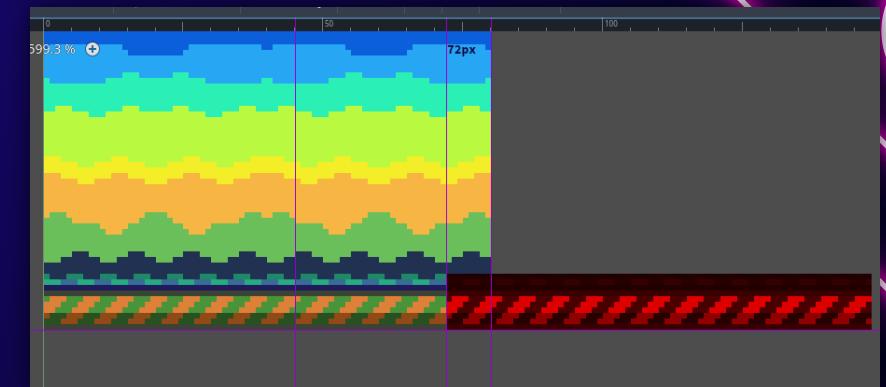
## Add metadata



## Set offset



## Loop offset



## Refactor: ready

```
1  extends Node2D
2
3  ## Retorna a lista de filhos do nó
4  ## Assume que todos os filhos são Sprites
5  @onready var children: Array[Node2D] = []
6
7  # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.
8  func _ready():
9
10 for child in get_children():
11     var copy: Node2D = child.duplicate()      # Duplica um filho
12
13     copy.position.x = child.get_meta("loop_offset") # Posiciona a cópia
14     add_child(copy) # Adiciona a cópia (uma instância de Sprite) como filho
15     # o nó adicionado deve ser órfão antes da chamada
16
17     #child.set_meta("twin", copy) # Salva a "instância gêmea" em metadados
18     #copy.set_meta("twin", child)
19
20     children.append(child) # Salva os filhos na lista 'children'
21     children.append(copy)
22
```

## Refactor: process

```
23  var speed: float = 1.0 ## Taxa de velocidade (rapidez)
24
25  # Chamado a cada frame. 'delta' é o tempo passado desde o último frame.
26  func _process(delta):
27      # Velocidade atual do movimento no eixo-x
28      var velocity_x: float = speed * delta
29      # Move os Sprites com base na velocidade atual
30
31      for child in children:
32          child.position.x -= velocity_x
33
34      for child in children:
35          #var twin: Node2D = child.get_meta("twin")
36          var offset: float = child.get_meta("loop_offset")
37          var end_pos: float = child.position.x + offset
38
39          if end_pos < 0:
40              # end_pos será o quanto de espaço em x ultrapassou o eixo y
41              child.position.x = offset + end_pos
42
43          speed += delta # Aumenta a velocidade progressivamente
44
```



# Virtual pos

## Metadata

Loop Offset 80

Virtual Pos 0

+ Add Metadata

Fix pixel precision

```
1  extends Node2D
2
3  ## Retorna a lista de filhos do nó
4  ## Assume que todos os filhos são Sprites
5  @onready var children: Array[Node2D] = []
6
7  # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.
8  func _ready():
9    for child in get_children():
10      var copy: Node2D = child.duplicate()      # Duplica um filho
11      var offset: float = child.get_meta("loop_offset")
12
13      copy.set_meta("virtual_pos", offset) # Posição "virtual"
14      copy.position.x = floorf(offset)      # Posiciona a cópia (fixed-pixel)
15
16      add_child(copy) # Adiciona a cópia (uma instância de Sprite) como filho
17      # o nó adicionado deve ser órfão antes da chamada
18
19      #child.set_meta("twin", copy) # Salva a "instância gêmea" em metadados
20      #copy.set_meta("twin", child)
21
22      children.append(child) # Salva os filhos na lista `children`
23      children.append(copy)
24
25
```

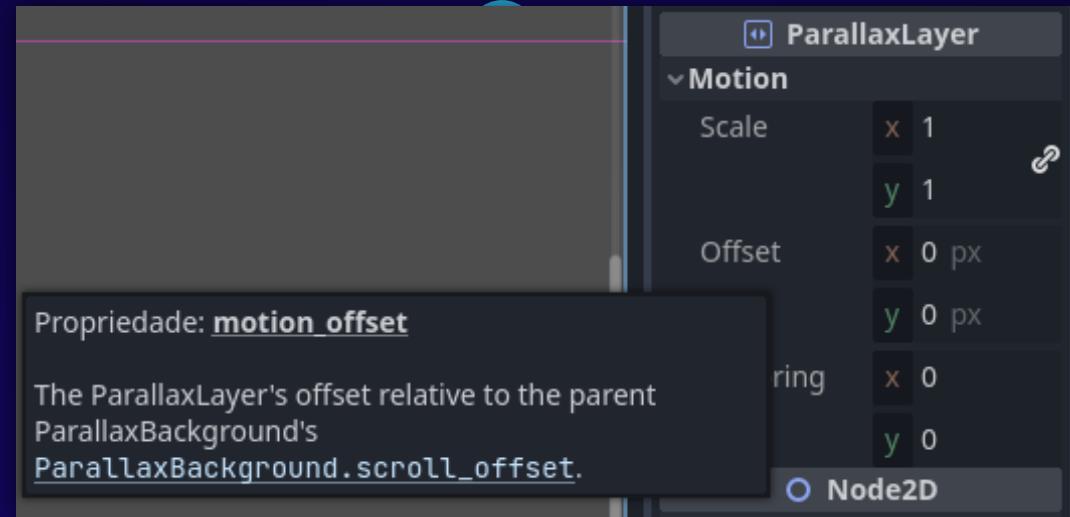
Fix position

```
26  var speed: float = 1.0 ## Taxa de velocidade (rapidez)
27
28  # Chamado a cada frame. 'delta' é o tempo passado desde o último frame.
29  func _process(delta):
30    # Velocidade atual do movimento no eixo-x
31    var velocity_x: float = speed * delta
32    # Move os Sprites com base na velocidade atual
33
34    for child in children:
35      var new_pos := (child.get_meta("virtual_pos") as float) - velocity_x
36      child.set_meta("virtual_pos", new_pos)
37      child.position.x = roundf(new_pos)
38      #child.position.x -= velocity_x
39
40    for child in children:
41      var twin: Node2D = child.get_meta("twin")
42      var offset: float = child.get_meta("loop_offset")
43      var end_pos: float = (child.get_meta("virtual_pos") as float) + offset
44
45      if end_pos < 0:
46        # end_pos será o quanto de espaço em x ultrapassou o eixo y
47        child.set_meta("virtual_pos", offset + end_pos)
48        child.position.x = roundf(offset + end_pos)
49
50    speed += delta # Aumenta a velocidade progressivamente
51
```

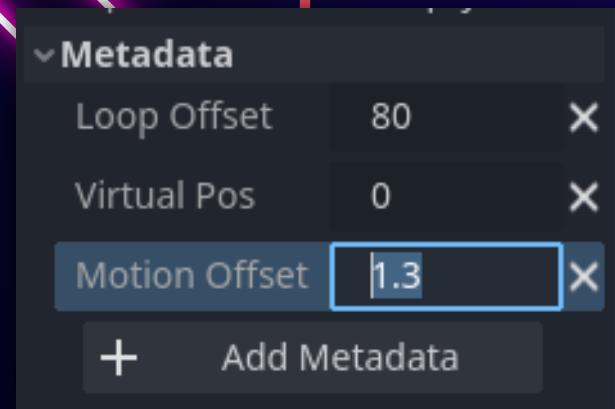
Fix gap final



## Parallax offset



## Motion offset



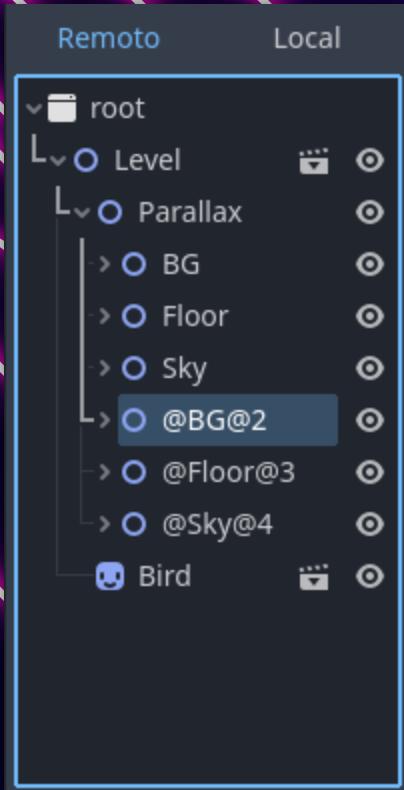
## Apply offset

```
34    for child in children:  
35        var new_pos := child.get_meta("virtual_pos") as float - \  
36            velocity_x * child.get_meta("motion_offset") as float  
37        child.set_meta("virtual_pos", new_pos)  
38        child.position.x = roundf(new_pos)  
39        #child.position.x -= velocity_x  
40
```

# Order problem



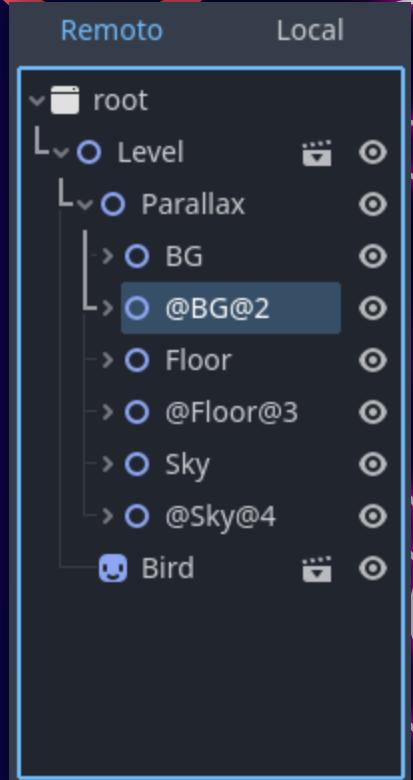
## Remote state



## Node as sibling

```
    7  # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.  
  ↪ 8  ↘ func _ready():  
    9    ↳  
   10   ↳    for child in get_children():  
   11     ↳    var copy: Node2D = child.duplicate()      # Duplica um filho  
   12     ↳    var offset: float = child.get_meta("loop_offset")  
   13     ↳    ↳  
   14     ↳    ↳    copy.set_meta("virtual_pos", offset) # Posição "virtual"  
   15     ↳    ↳    copy.position.x = floorf(offset)      # Posiciona a cópia (fixed-pixel)  
   16     ↳    ↳  
   17     ↳    ↳    child.add_sibling(copy) # Adiciona a cópia no mesmo nível do filho  
   18     ↳    ↳    # o nó adicionado deve ser órfão antes da chamada  
   19     ↳    ↳
```

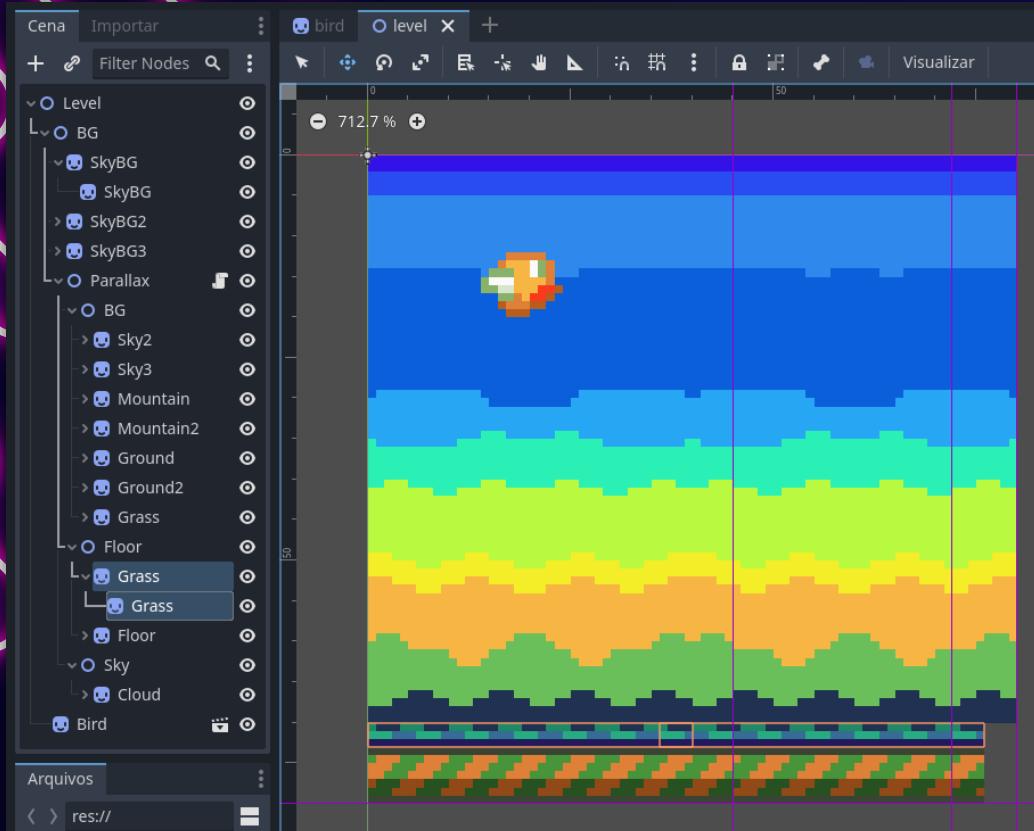
## Remote fixed



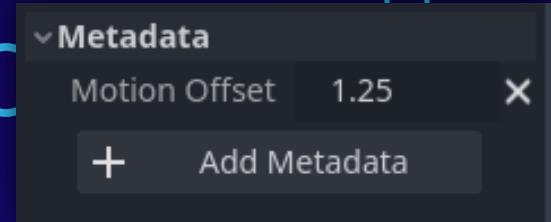
Order problem fixed



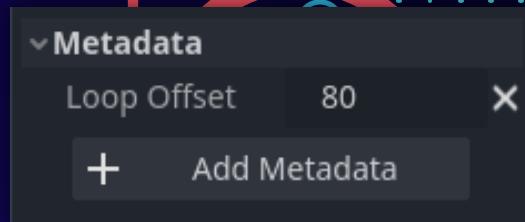
Nova hierarquia +



## Metadados internos



Remove metadata +



## Grandchildren

```
1  extends Node2D
2
3  ## A lista de filhos do nó
4  ## Assume que todos os filhos são Node2D
5  @onready var children: Array[Node2D] = []
6
7  # _ready é chamado quando o nó entra na árvore da cena pela primeira vez.
8  func _ready():
9    for child in get_children(): # filhos
10      var offset: float = child.get_meta("loop_offset")
11
12      for grandchild in child.get_children(): # netos
13        var copy: Node2D = grandchild.duplicate() # Duplica um neto
14
15        grandchild.set_meta("virtual_posx", grandchild.position.x)
16        copy.set_meta("virtual_posx", offset) # Posição "virtual"
17        copy.position.x = floorf(offset)      # Posiciona a cópia (fixed-pxl)
18
19        grandchild.add_sibling(copy)
20
21      # Adiciona a cópia como um "irmão" logo abaixo do neto
22      # o nó adicionado deve ser órfão antes da chamada
23
24      children.append(child) # Salva os filhos na lista 'children'
25
```

## Refactor: loop

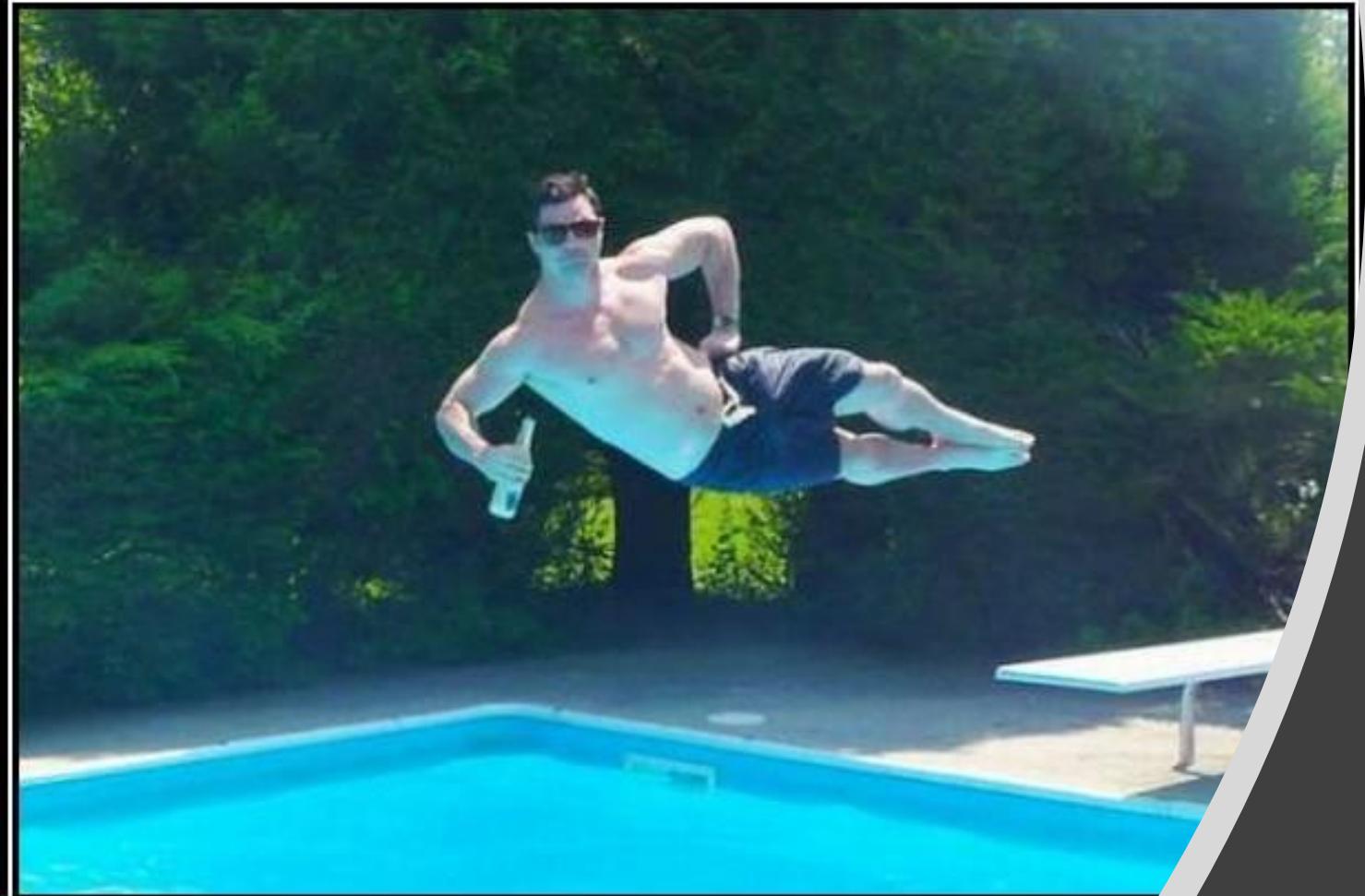


```
26  var speed: float = 1.0 ## Taxa de velocidade (rapidez)
27
28  # Chamado a cada frame. 'delta' é o tempo passado desde o último frame.
29  func _process(delta):
30      # Velocidade atual do movimento no eixo-x
31      var velocity_x: float = speed * delta
32      # Move os Sprites com base na velocidade atual
33
34      for child in children:
35          var offset: float = child.get_meta("loop_offset")
36
37          for grandchild in child.get_children():
38              var new_pos := grandchild.get_meta("virtual_posx") as float - \
39                  velocity_x * grandchild.get_meta("motion_offset") as float
40              var end_pos := new_pos + offset
41
42              if end_pos < 0:
43                  # end_pos será o quanto de espaço em x ultrapassou o eixo y
44                  new_pos = offset + end_pos
45
46              grandchild.set_meta("virtual_posx", new_pos)
47              grandchild.position.x = roundf(new_pos)
48
49      speed += delta # Aumenta a velocidade progressivamente
50
```

## Refactor: process



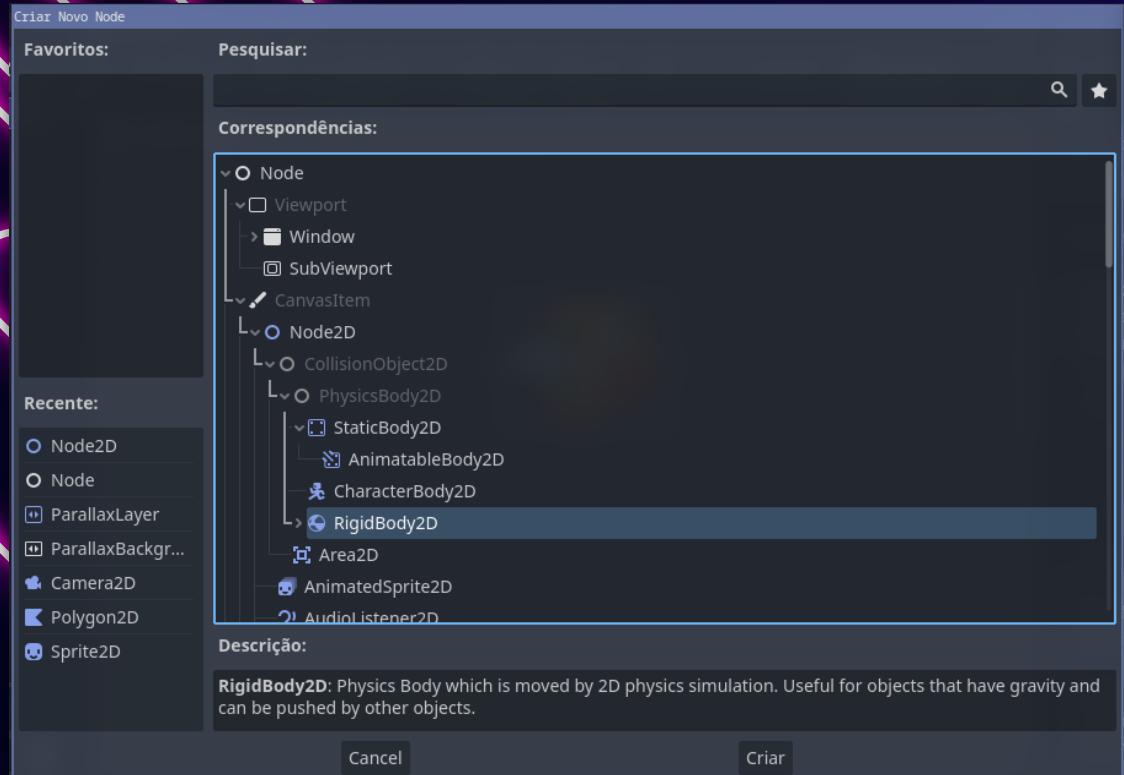
```
28  # Chamado a cada frame. 'delta' é o tempo passado desde o último frame.
29  func _process(delta):
30      # Velocidade atual do movimento no eixo-x
31      var velocity_x: float = speed * delta
32      # Move os Sprites com base na velocidade atual
33
34      for child in children:
35          for grandchild in child.get_children():
36              var new_pos := grandchild.get_meta("virtual_posx") as float - \
37                  velocity_x * grandchild.get_meta("motion_offset") as float
38              grandchild.set_meta("virtual_posx", new_pos)
39              grandchild.position.x = roundf(new_pos)
40
41      for child in children:
42          var offset: float = child.get_meta("loop_offset")
43
44          for grandchild in child.get_children():
45              var end_pos := (grandchild.get_meta("virtual_posx") as float) + offset
46
47              if end_pos < 0:
48                  # end_pos será o quanto de espaço em x ultrapassou o eixo y
49                  grandchild.set_meta("virtual_posx", offset + end_pos)
50                  grandchild.position.x = roundf(offset + end_pos)
51
52      speed += delta # Aumenta a velocidade progressivamente
```



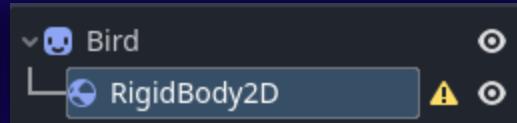
**GRAVITY**  
Just a theory

Gravidade

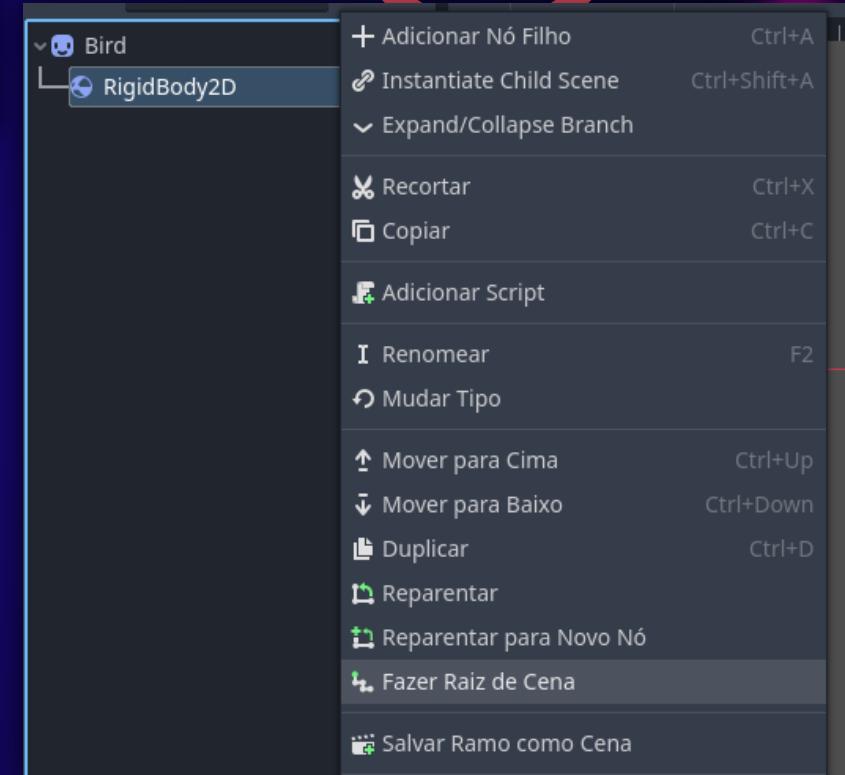
# Add Rigidbody2D



# Rigidbody2D tree

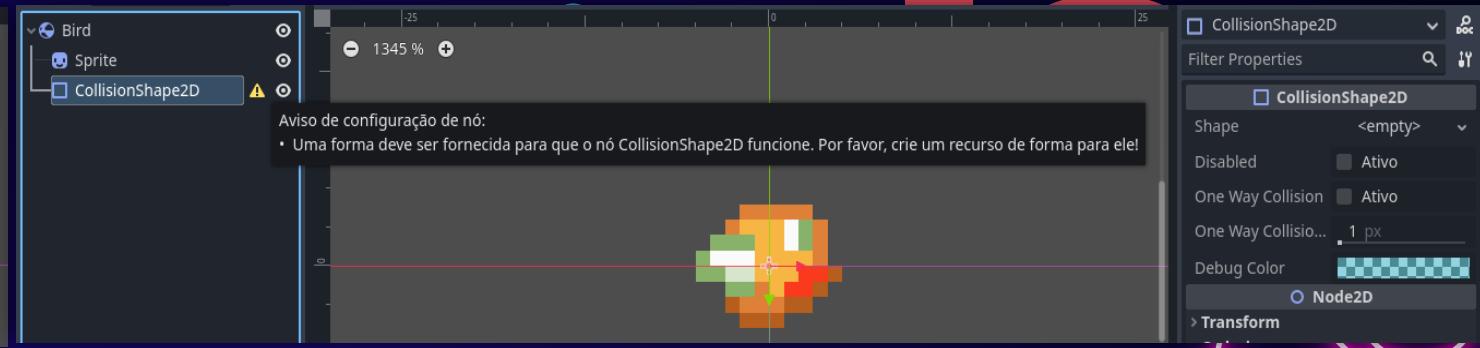
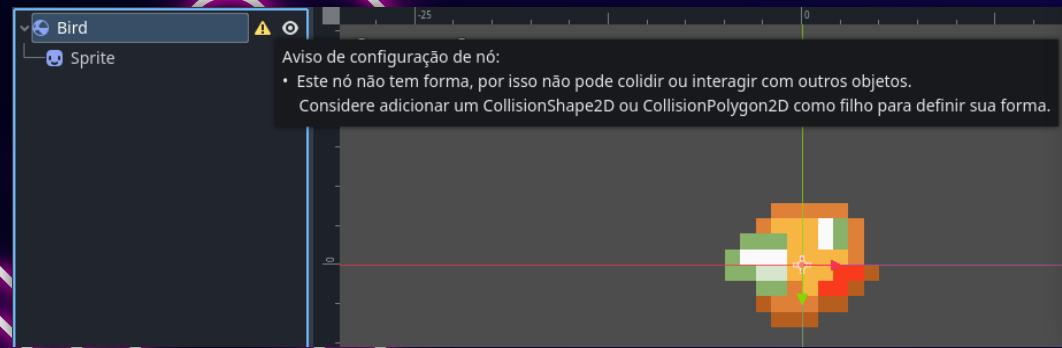


# Tornar raiz

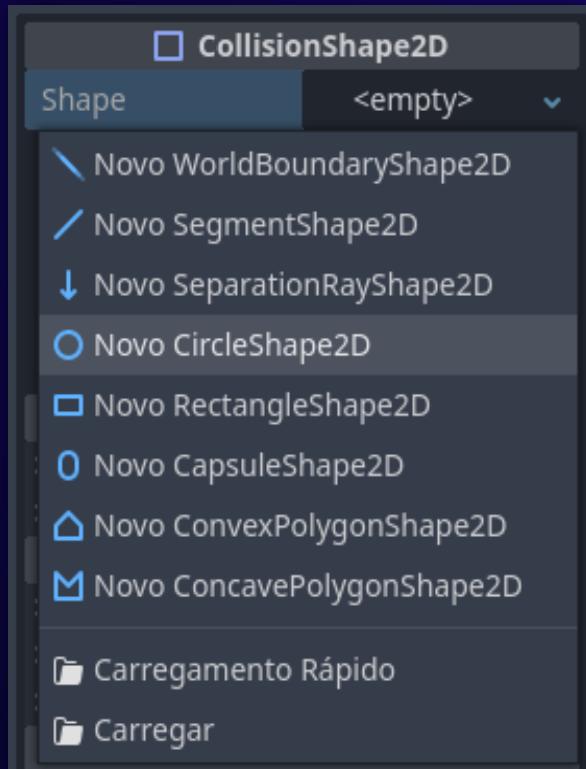


# Warning

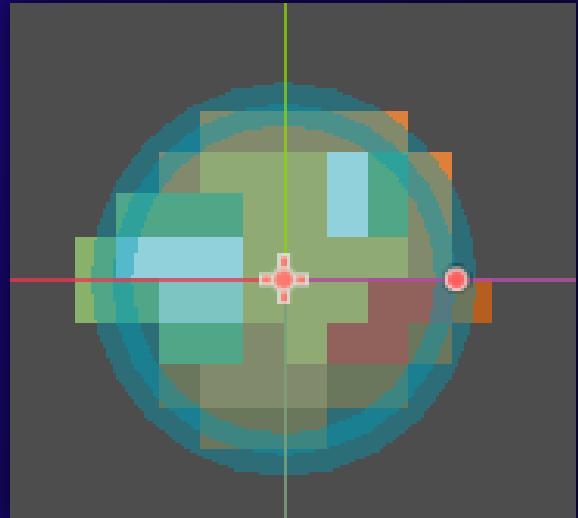
# Collision shape



## CircleShape



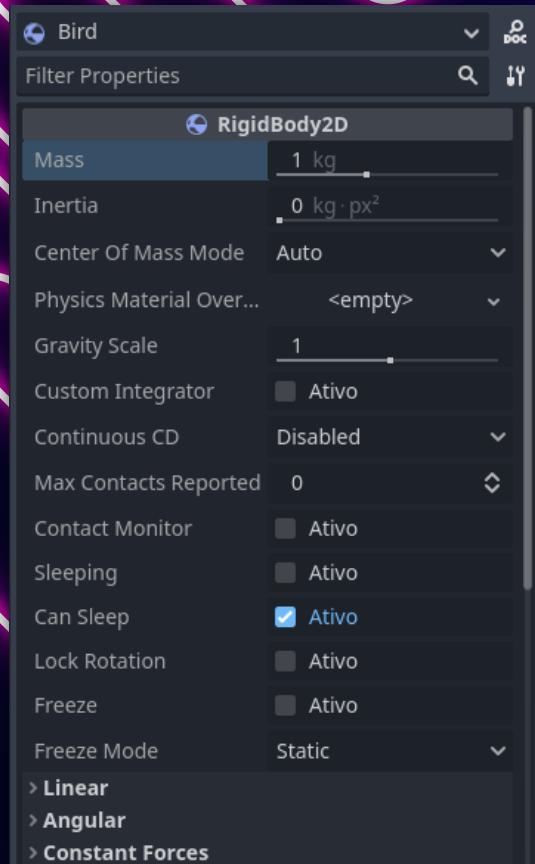
## Fit shape



Gravity



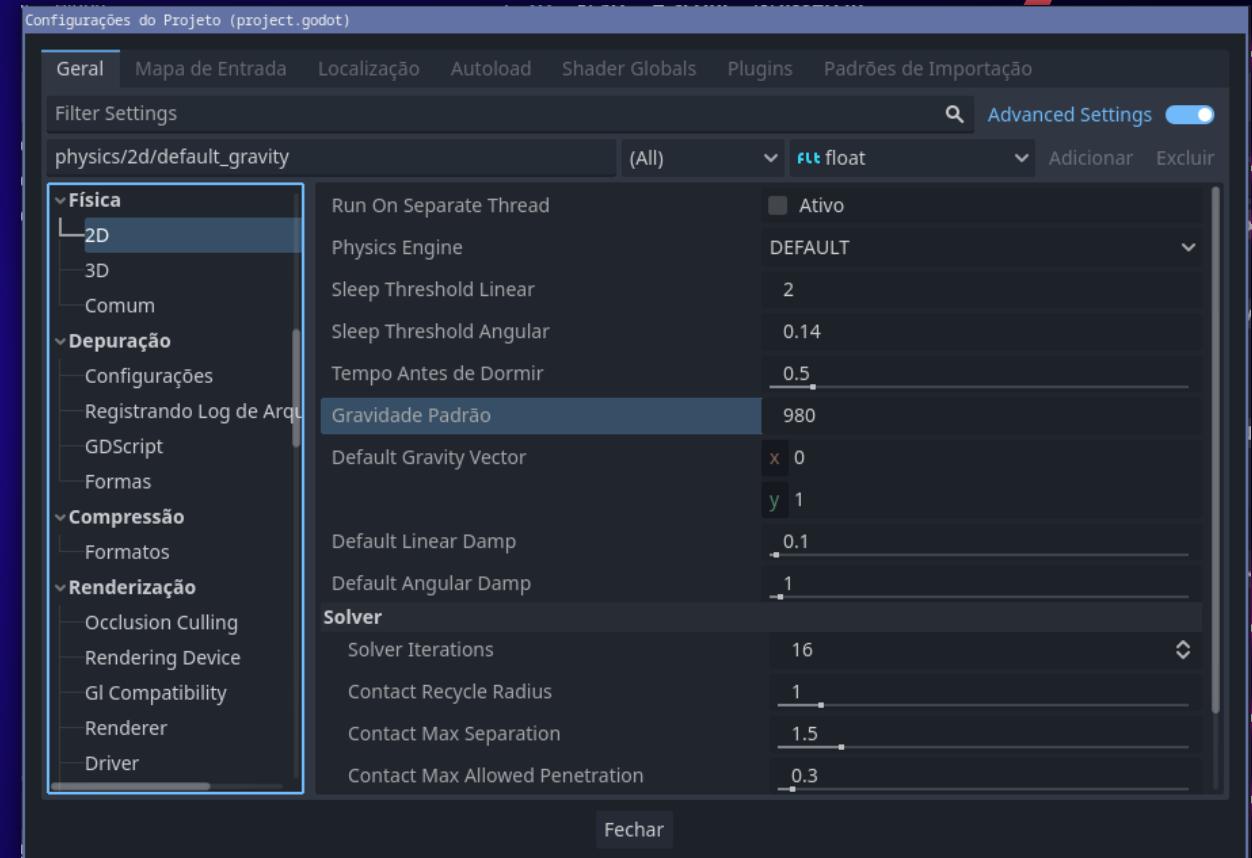
## Set mass



## Gravity scale



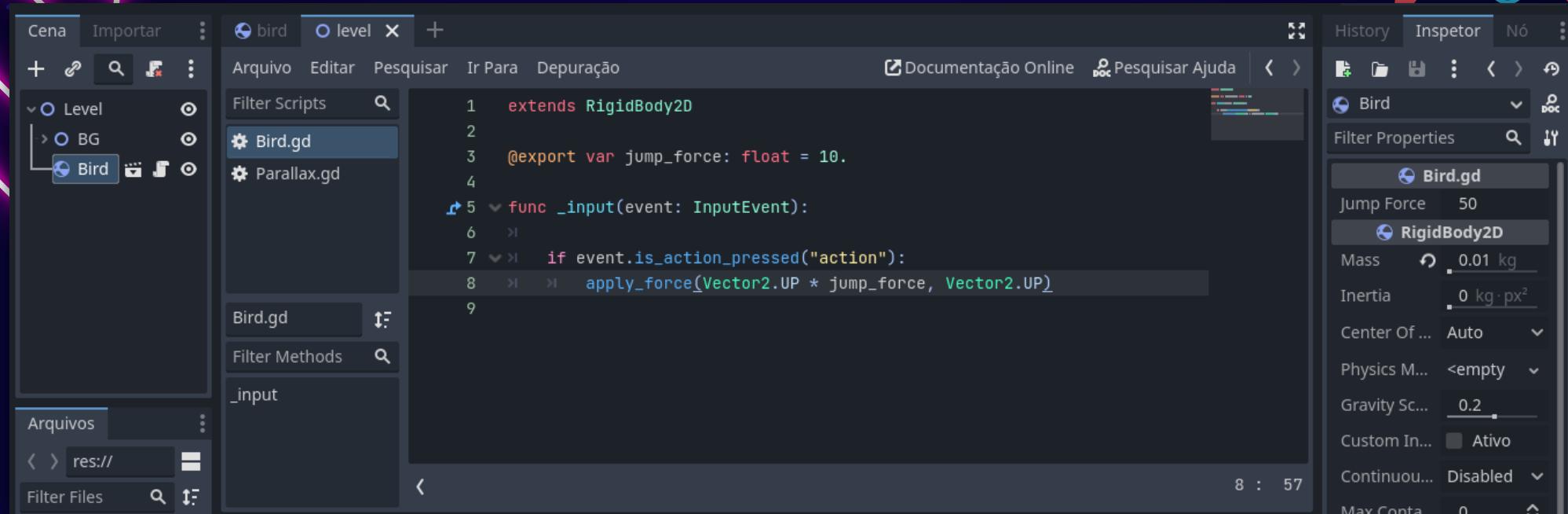
## Defaults



Gravity fixed



# Script do pássaro



```
extends RigidBody2D

@export var jump_force: float = 10.

func _input(event: InputEvent):
    if event.is_action_pressed("action"):
        apply_force(Vector2.UP * jump_force, Vector2.UP)
```

The screenshot shows the Godot Engine's script editor interface. On the left is the Project Tree with nodes like 'Cena', 'Importar', 'bird', 'level', 'Level', 'BG', and 'Bird'. The main area displays the 'Bird.gd' script. The right side shows the Inspector panel for the 'Bird' node, which is a 'RigidBody2D' with properties like 'Jump Force' set to 50, 'Mass' set to 0.01 kg, and 'Gravity Scale' set to 0.2.

## Apply force Docs

- `void apply_force(force: Vector2, position: Vector2 = Vector2(0, 0))`

Applies a positioned force to the body. A force is time dependent and meant to be applied every physics update.

`position` is the offset from the body origin in global coordinates.

# Mapa de entrada

Configurações do Projeto (project.godot)

Geral Mapa de Entrada Localização Autoload Shader Globals Plugins Padrões de Importação

Filter by name... Filter by event... Clear All

Add New Action Adicionar Show Built-in Actions

Ação	Zona morta	
action	0.5	+
Space (Físico)		i
Left Mouse Button - Todos os dispositivos		i
Joypad Button 0 (Bottom Action, Sony Cross, Xbox A, Nintendo B) - Todos os dispositivos		i

Fechar

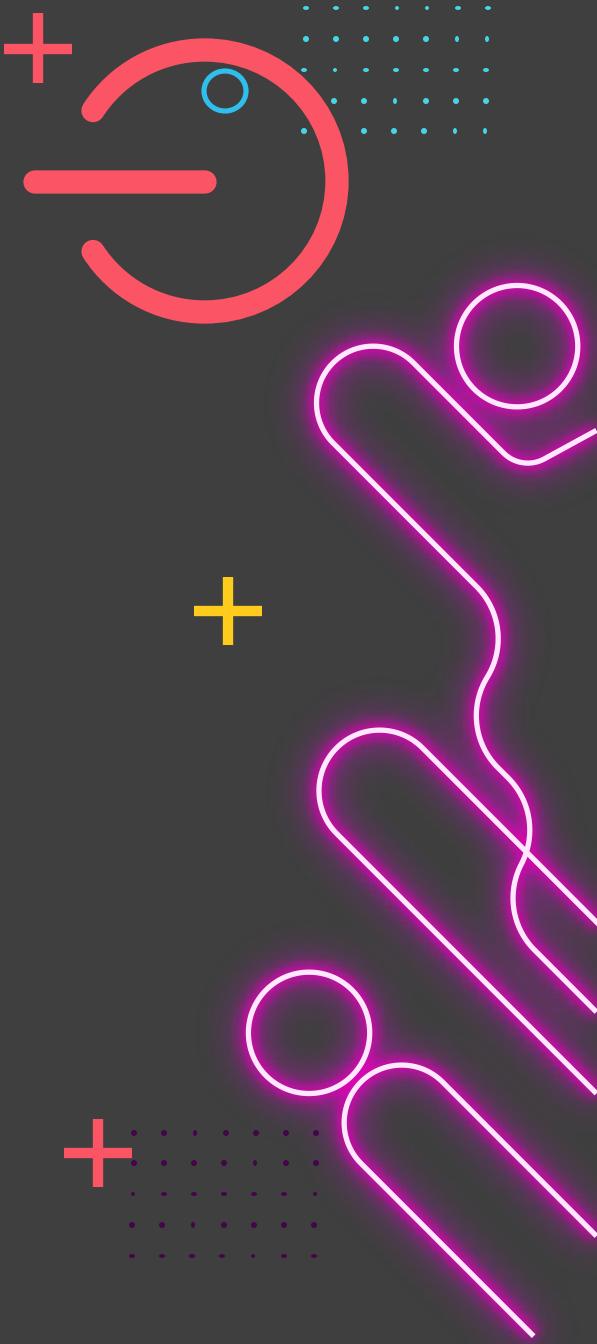
Fly



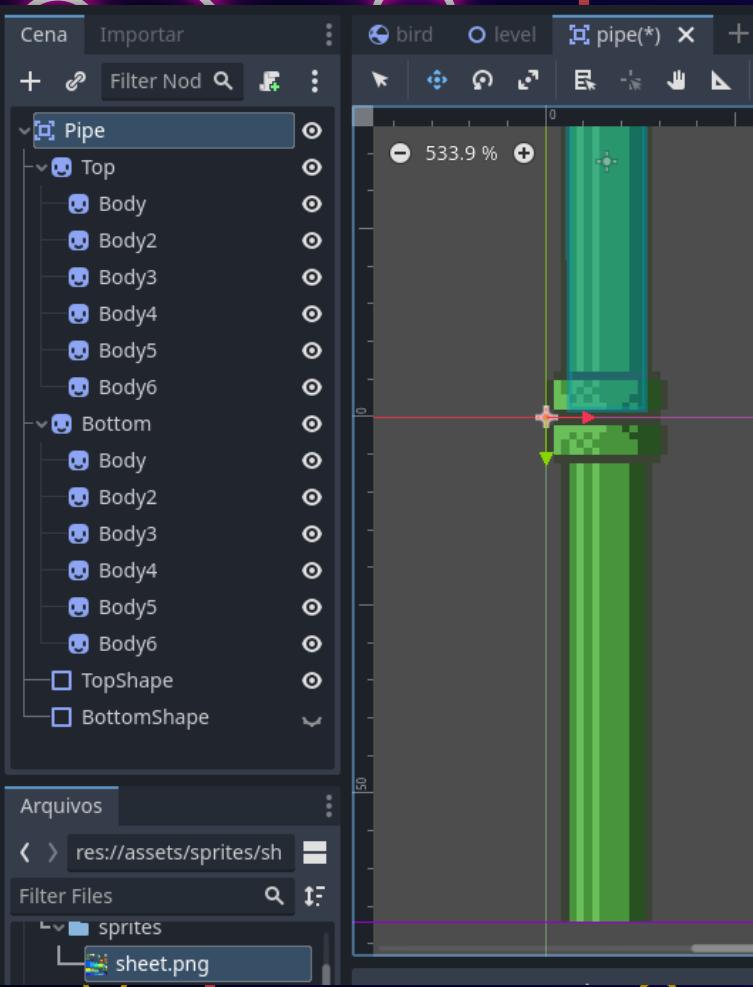
Who the hell  
are you?



Canos



Cano



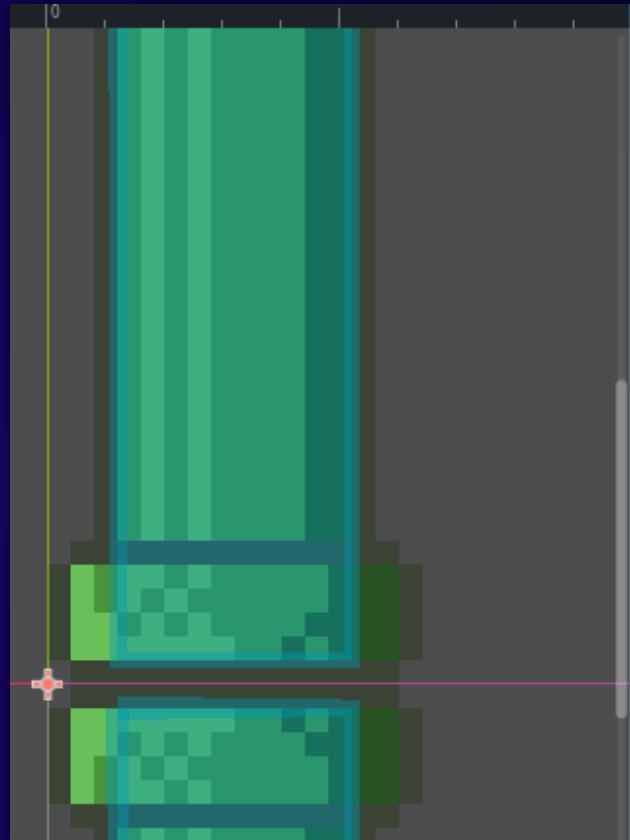
## Script tool

```
1  @tool # Torna o código executável no editor
2  extends Area2D
3
4  ## Distância entre canos
5  @export_range(0., 67.) var gap: float
6
7
8  func _ready():
9      var top: Sprite2D = $Top
10     var top_shape: CollisionShape2D = $TopShape
11
12     top.position.y = -(gap + top.region_rect.size.y)
13     top_shape.position.y = -(gap + top_shape.shape.size.y / 2 + 1)
```

# Setter

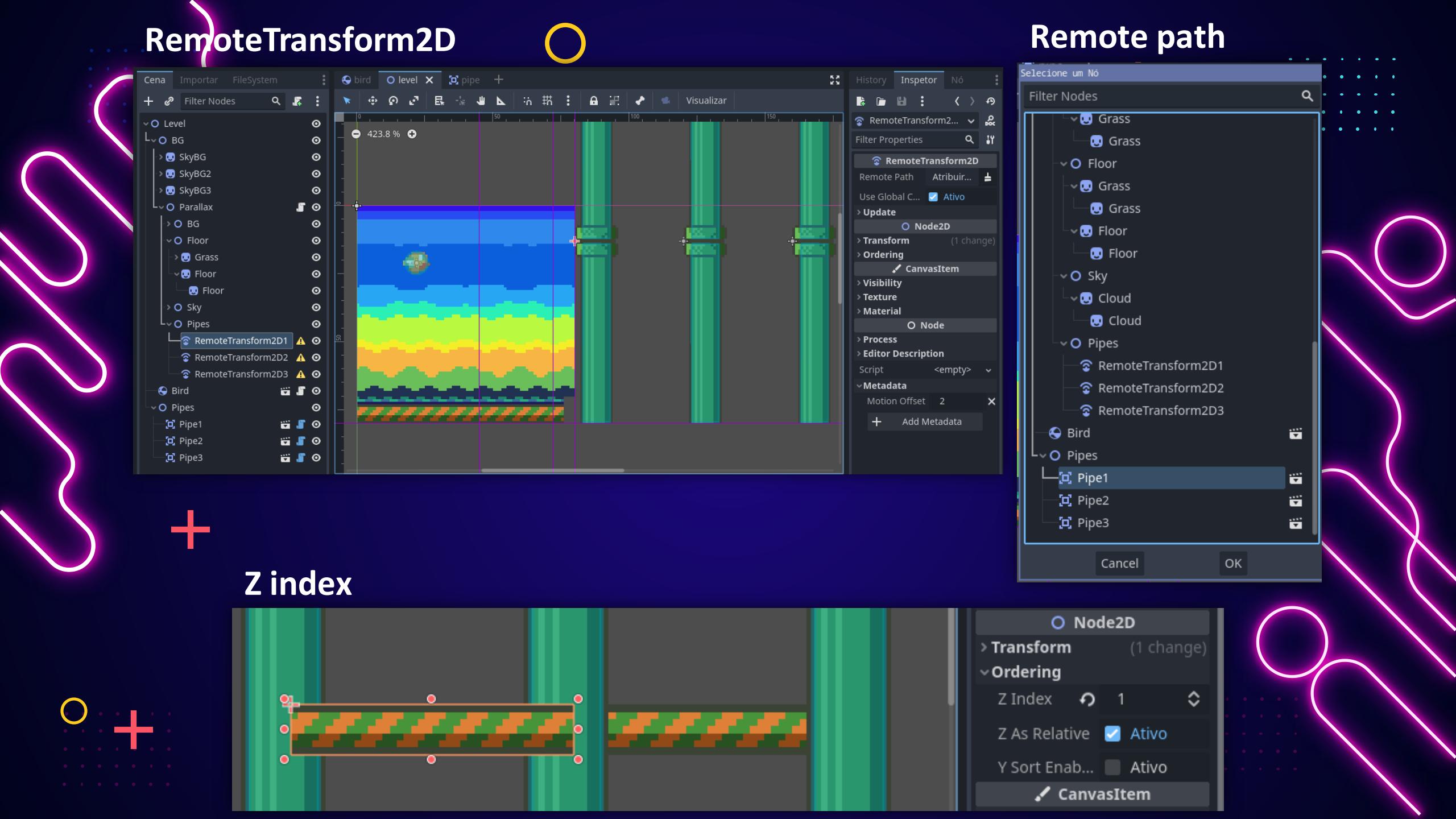
```
1  @tool # Torna o código executável no editor
2  extends Area2D
3
4  ## Variável do comportamento que atualiza a cena com base no valor de `gap`
5  var _gap_update: Callable = func(): pass
6
7  ## Distância entre canos
8  @export_range(0., 67.) var gap: float = 0:
9  set(value):
10     _gap_update.call(value)
11     gap = value
12
13  @onready var top: Sprite2D = $Top
14  @onready var top_shape: CollisionShape2D = $TopShape
15
16
17  func _ready():
18      _gap_update = update_gap
19      update_gap(gap)
20
21
22  func update_gap(value: float):
23      top.position.y = -(value + top.region_rect.size.y)
24      top_shape.position.y = -(value + top_shape.shape.size.y / 2 + 1)
25
```

# Gap



# RemoteTransform2D

# Remote path



Pipes



NEED HELP WITH LIFE  
DECISIONS?

RANDOM NUMBER  
GENERATOR

Geração +  
aleatoriedade

# Result



# Individual logic sequence

```
15 func _ready():
16     _gap_update = update_gap
17     update_gap(gap)
18     # Desativa o processamento no Editor
19     set_process(not Engine.is_editor_hint())
20
21
22 const MOTION_OFFSET: float = 2.
23 var speed: float = 1.0
24 @onready var virtual_posx = position.x
25
26 func _process(delta: float):
27     var velocity_x := speed * delta * MOTION_OFFSET
28     virtual_posx -= velocity_x
29     position.x = floorf(virtual_posx)
30
31 if position.x < -top.region_rect.size.x:
32     queue_free() # Deleta a instância de Pipe
33
34 speed += delta
35
```

## Spawner

```
Cena Importar FileSystem + Filter Nodes Arquivo Editar Pesquisar Ir Para Depuração Documentação Online Pesquisar Ajuda
+ Level BG SkyBG SkyBG2 SkyBG3 Parallax BG Floor Sky Bird PipeSpawner Pipe1 Pipe2 Pipe3
Filter Scripts
Bird.gd
Parallax.gd
Pipe.gd
PipeSpawner.gd(*)
1 extends Node2D
2
3 const SPAWN_POS: float = 80.
4 var pipeScn: PackedScene = preload("res://src/pipe.tscn")
5
6
7 func spawn_pipe():
8     var new_pipe = pipeScn.instantiate()
9
10    new_pipe.position.x = SPAWN_POS
11    add_child(new_pipe)
12
```

# Sinal

Arquivo Editar Pesquisar Ir Para Depuração Documentação Online Pesquisar Ajuda Sinais Grupos

Filter Scripts  Pipe.gd

Filter Signals  Pipe.gd deadzone\_reached()

Bird.gd Parallax.gd Pipe.gd PipeSpawner.gd

Pipe.gd

```
21 signal deadzone_reached
22 var speed: float = 1.0
23
24 func _process(delta: float):
25     var velocity_x := speed * delta
26     position.x -= velocity_x * 2.
27
28     if position.x < -top.region_rect.size.x:
29         deadzone_reached.emit()
30         queue_free() # Deleta a instância de Pipe
31
32     speed += delta
33
34
```

Conectar...

## Conectar

Conectar um Sinal a um Método

Sinal Origem: deadzone\_reached

Conectar ao Script:

- Floor
- Sky
- Cloud
- Bird
- PipeSpawner
  - Pipe1
  - Pipe2
  - Pipe3 (Conectando de)

Selecionar Método:

- \_on\_pipe\_3\_deadzone\_reached

Avançado

Cancelar Conectar

Arquivo Editar Pesquisar Ir Para Depuração Documentação Online Pesquisar Ajuda Sinais Grupos

Filter Scripts  PipeSpawner.gd

Filter Signals  Pipe.gd deadzone\_reached()

Bird.gd Parallax.gd Pipe.gd PipeSpawner.gd

PipeSpawner.gd

```
14 func spawn_pipe():
15     var new_pipe = pipeScn.instantiate()
16
17     new_pipe.position.x = SPAWN_POS
18     add_child(new_pipe)
19
```

Conexões com o método:

Origem	Sinal	Destino
Pipe3	deadzone_reached	PipeSpawner

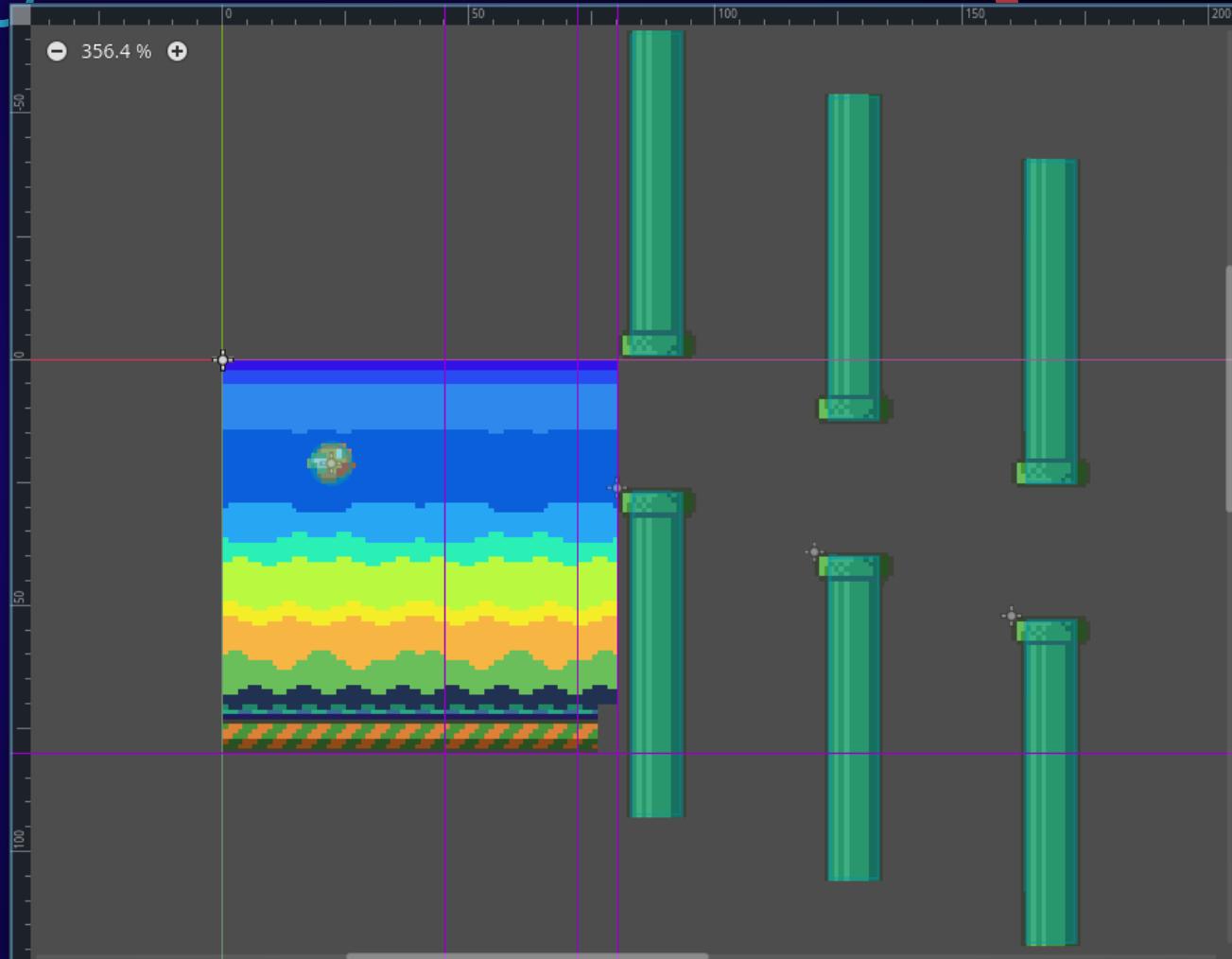
OK Desconectar

## Conexão

## Geração aleatória

```
1  extends Node2D
2
3  const SPAWN_POS: float = 80.
4  const Pipe = preload("res://src/Pipe.gd")
5  const PIPE_SCN: PackedScene = preload("res://src/pipe.tscn")
6
7  @export var gap_reduction: float = 0.1
8  @export var low_gap: float = 13.
9  @export var high_gap: float = 26.
10 @onready var current_gap: float = high_gap
11
12 @export var min_position: float = 26.
13 @export var max_position: float = 80. - 7. # Screen.height - Floor
14
15 func _ready():
16     min_position = high_gap
17
18 func spawn_pipe():
19     var new_pipe: Pipe = PIPE_SCN.instantiate()
20
21     new_pipe.position.x = SPAWN_POS
22     new_pipe.position.y = floorf(randf_range(min_position, max_position))
23     new_pipe.gap = floorf(current_gap)
24
25     # Update Gap
26     current_gap = maxf(lerpf(current_gap, low_gap, gap_reduction), low_gap)
27
28     add_child(new_pipe)
```

## Cena dos canos



When you wake up at 6:59 and  
you set your timer at 7:00



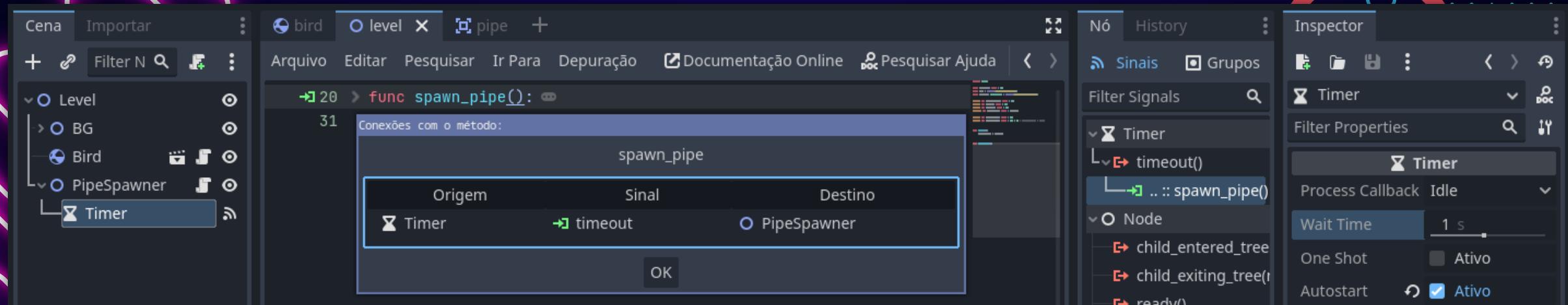
Timer



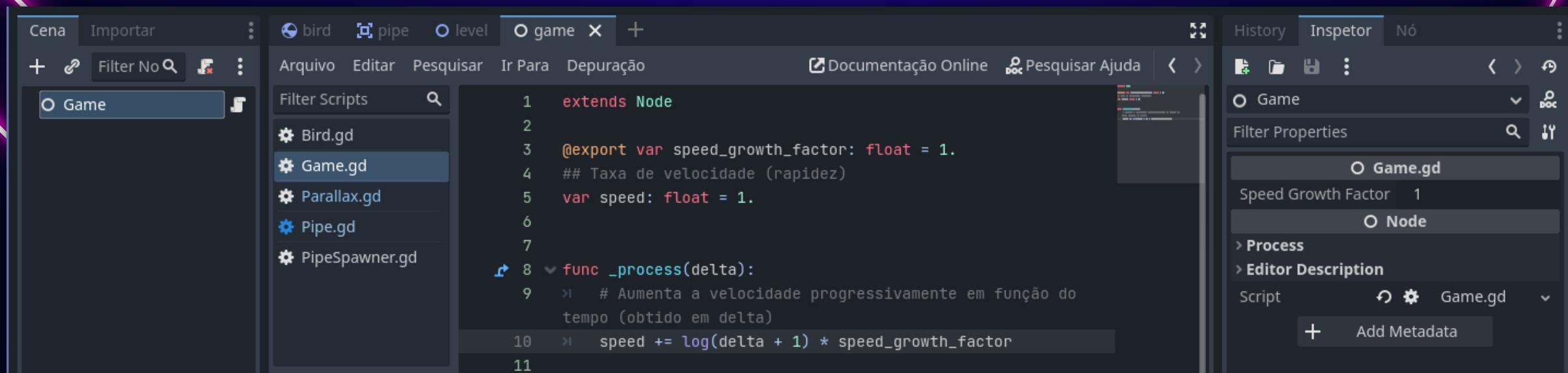
Result



# Timer connection



# Script "Game"



# Autoload

Configurações do Projeto (project.godot)

Geral Mapa de Entrada Localização Autoload Shader Globals Plugins Padrões de Importação

Caminho: Set path or press "Adicionar" to create a s

Nome	Caminho	Variável Global
Game	res://src/game.tscn	<input checked="" type="checkbox"/> Habilitar <input type="button" value=""/>

## Refactor: pipe speed

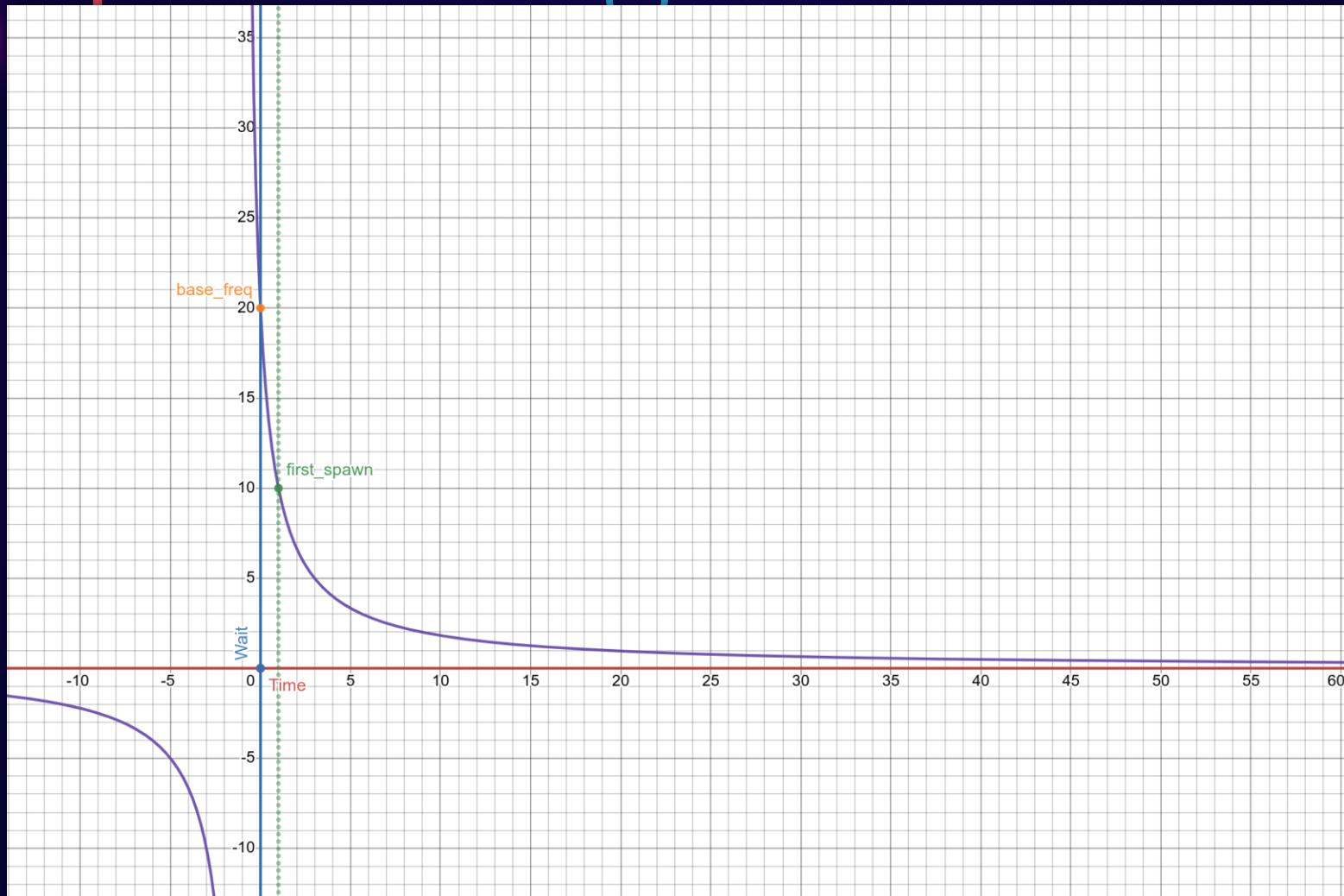
```
22 const MOTION_OFFSET: float = 2.  
23 @onready var virtual_posx = position.x  
24  
25 func _process(delta: float):  
26     var velocity_x := Game.speed * delta * MOTION_OFFSET  
27     # Velocidade atual do movimento no eixo-x  
28  
29     virtual_posx -= velocity_x  
30     position.x = roundf(virtual_posx)  
31  
32 if position.x < -top.region_rect.size.x:  
33     queue_free() # Deleta a instância de Pipe  
34
```

## Refactor: parallax speed

```
28 func _process(delta):  
29     # Velocidade atual do movimento no eixo-x  
30     var velocity_x: float = Game.speed * delta  
31  
32 for child in children:  
33     var offset: float = child.get_meta("loop_offset")  
34  
35 for grandchild in child.get_children():  
36     # Move os Sprites com base na velocidade atual  
37     var new_pos := grandchild.get_meta("virtual_posx") as float - \  
38     velocity_x * grandchild.get_meta("motion_offset") as float  
39     var end_pos := new_pos + offset  
40  
41 if end_pos < 0:  
42     # end_pos será o quanto de espaço em x ultrapassou o eixo y  
43     new_pos = offset + end_pos  
44  
45 grandchild.set_meta("virtual_posx", new_pos)  
46 grandchild.position.x = roundf(new_pos)  
47
```

```
20 @onready var timer: Timer = $Timer
21
22 const BASE_SPAWN_FREQ: float = 20.
23 const FREQ_DEC_DEG: float = 1.
24
25 func spawn_pipe():
26     var new_pipe: Pipe = PIPE_SCN.instantiate()
27
28     new_pipe.position.x = SPAWN_POS
29     new_pipe.position.y = roundf(randf_range(min_position, max_position))
30     new_pipe.gap = roundf(current_gap)
31
32     # Update Gap
33     current_gap = maxf(lerpf(current_gap, low_gap, gap_reduction), low_gap)
34
35     add_child(new_pipe)
36     timer.start(BASE_SPAWN_FREQ * FREQ_DEC_DEG / (Game.speed + FREQ_DEC_DEG))
37
```

## Wait time function



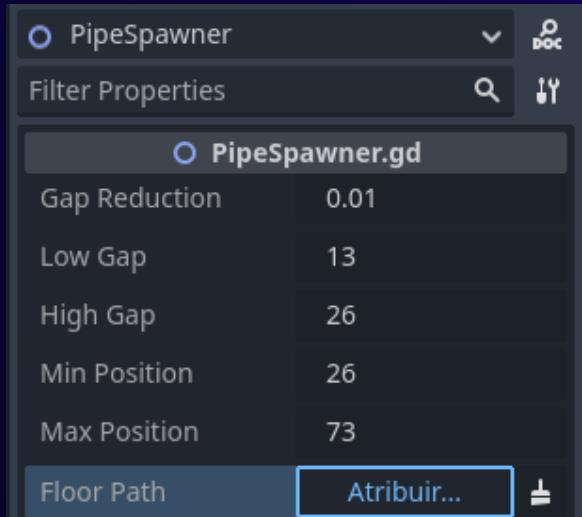
Timer restart



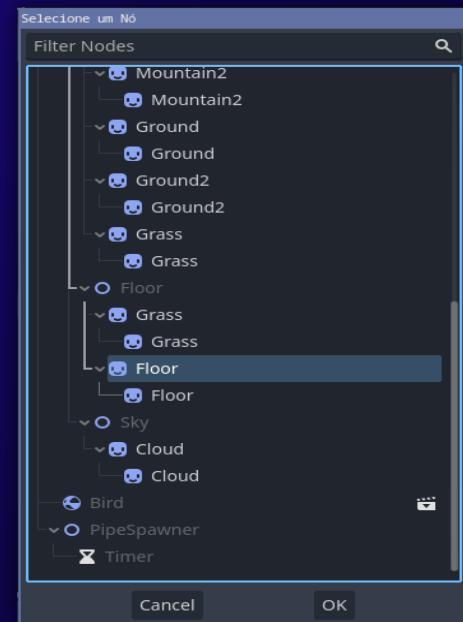
# Fix flicker

```
25 @export_node_path(Sprite2D)
26 var floor_path: NodePath
27 @onready var floor := get_node(floor_path) as Sprite2D
28
29 func spawn_pipe():
30     var new_pipe: Pipe = PIPE_SCN.instantiate()
31     var floor_virtual_posx := floor.get_meta("virtual_posx") as float
32
33     new_pipe.position.x = SPAWN_POS + \
34         floor_virtual_posx - int(floor_virtual_posx)
35     new_pipe.position.y = roundf(randf_range(min_position, max_position))
36     new_pipe.gap = roundf(current_gap)
37
38     # Update Gap
39     current_gap = maxf(lerpf(current_gap, low_gap, gap_reduction), low_gap)
40
41     add_child(new_pipe)
42     timer.start(BASE_SPAWN_FREQ * FREQ_DEC_DEG / (Game.speed + FREQ_DEC_DEG))
43
```

## Path inspector



## Path to floor



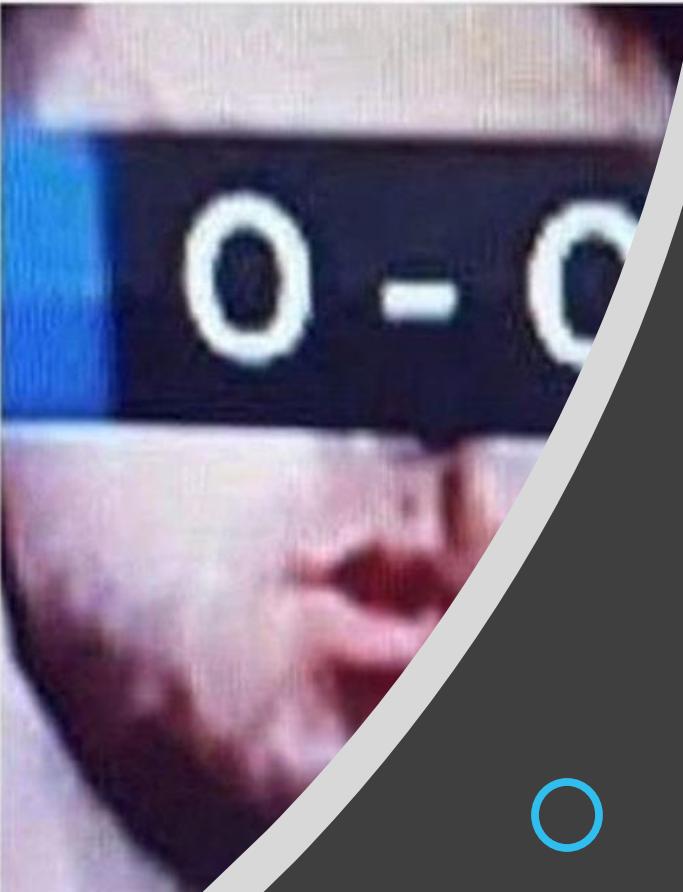
Final result





Tom  
@TomFoins

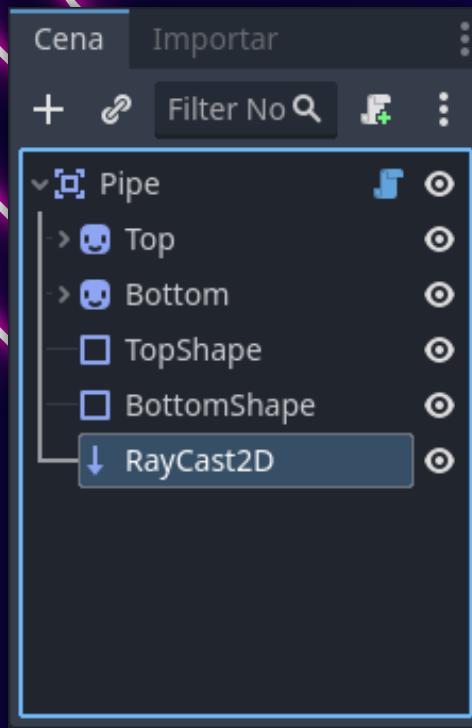
Spent a long time laughing at this



Score,  
Raycasting<sup>+</sup>&  
HUD



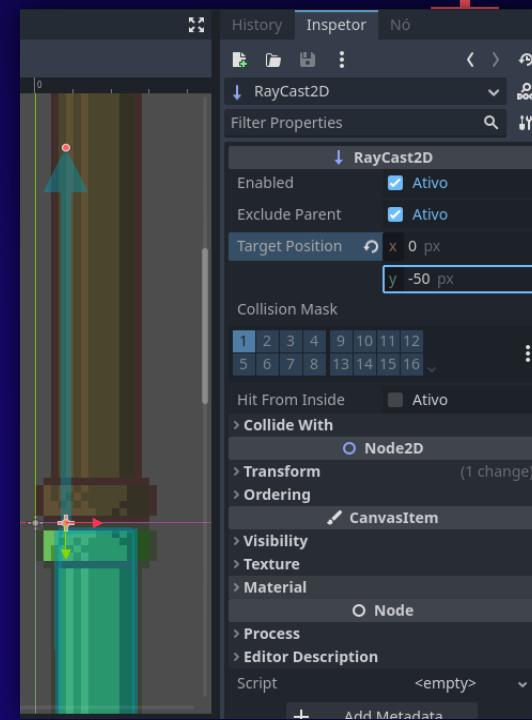
## Add RayCast2D



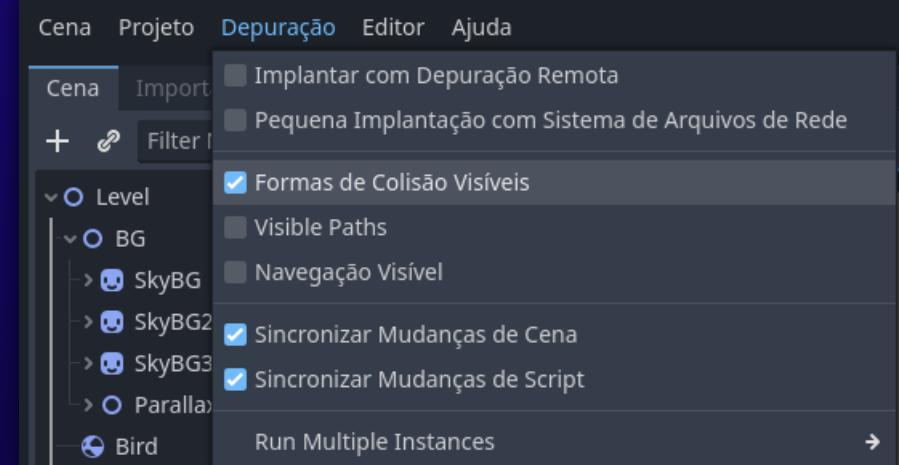
Pause

```
36 @onready var raycast := $RayCast2D as RayCast2D
37
38 func _physics_process(delta: float):
39     if raycast.is_colliding():
40         get_tree().paused = true # Pausa o jogo
41
```

## Target position



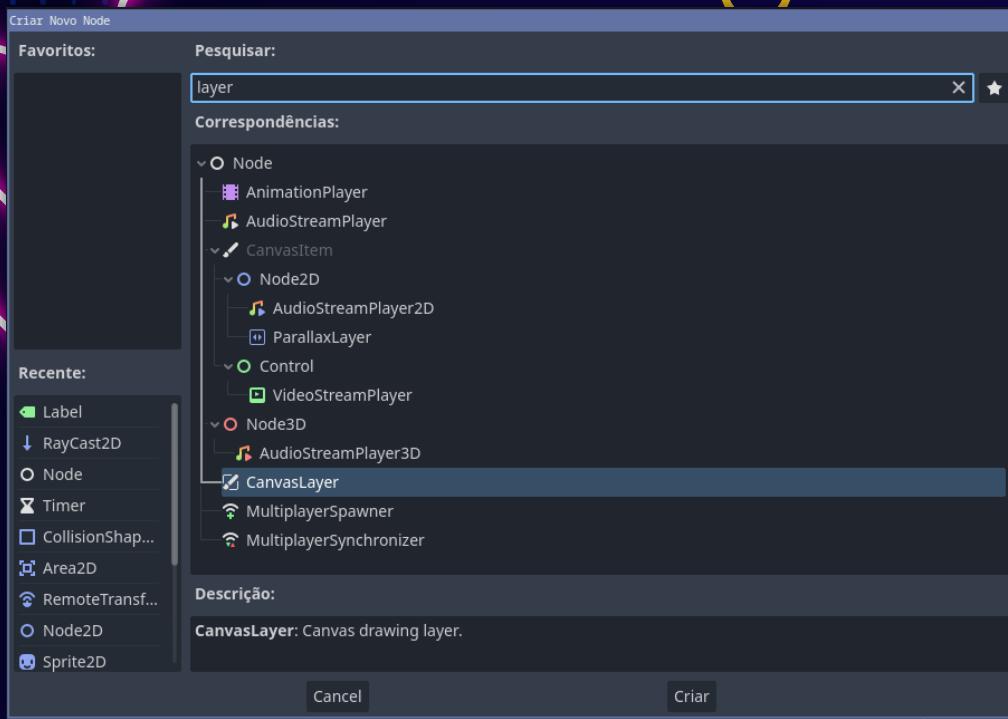
Visible collisions



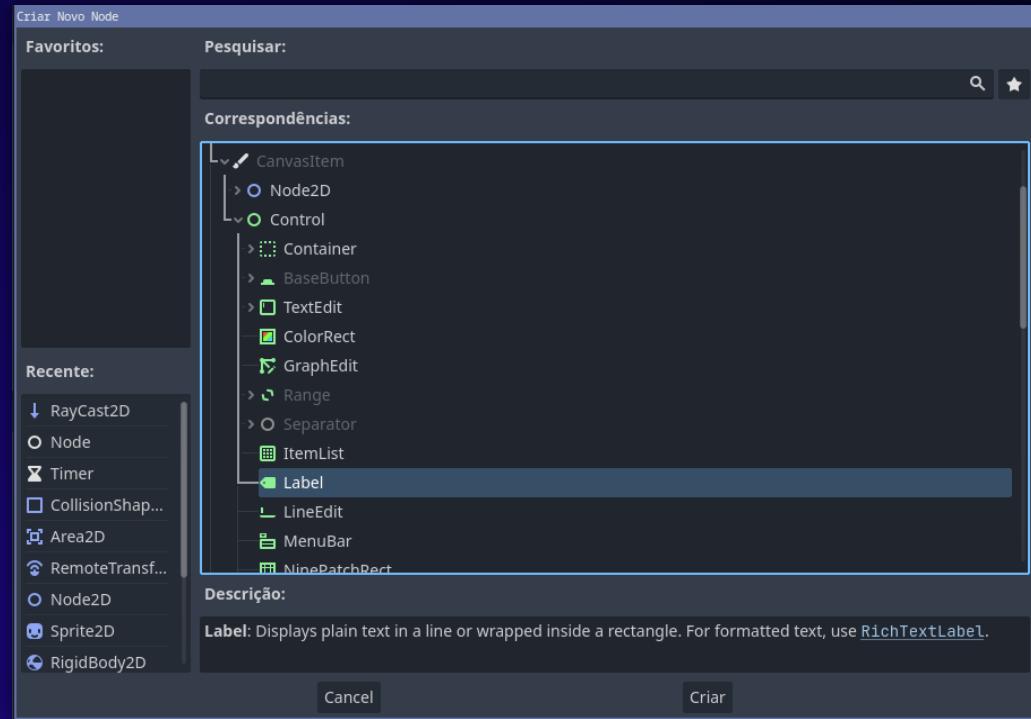
RayCast



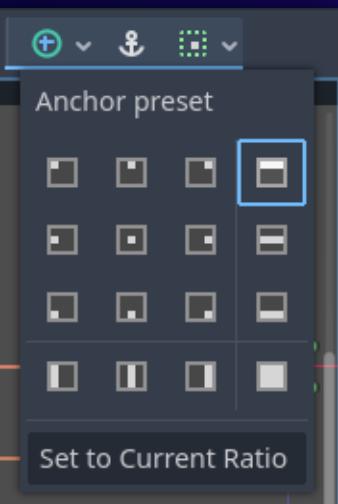
# CanvasLayer



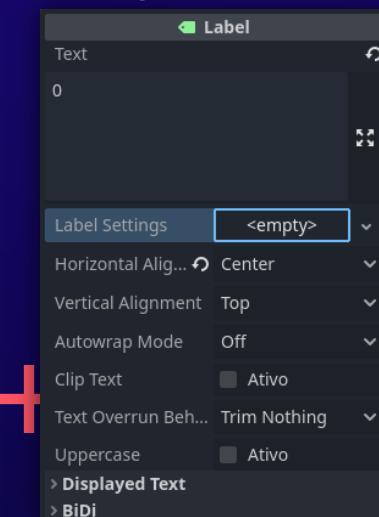
# Label



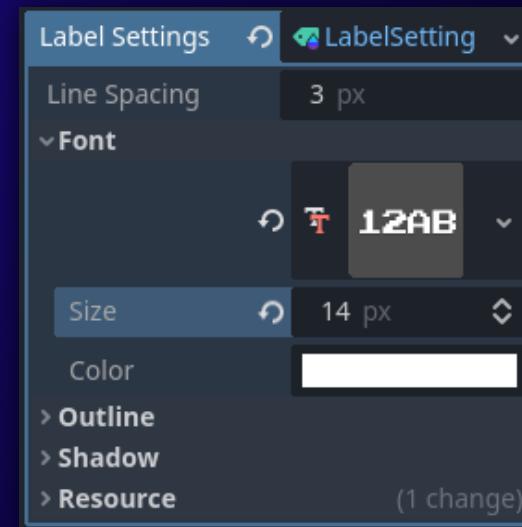
# Anchors

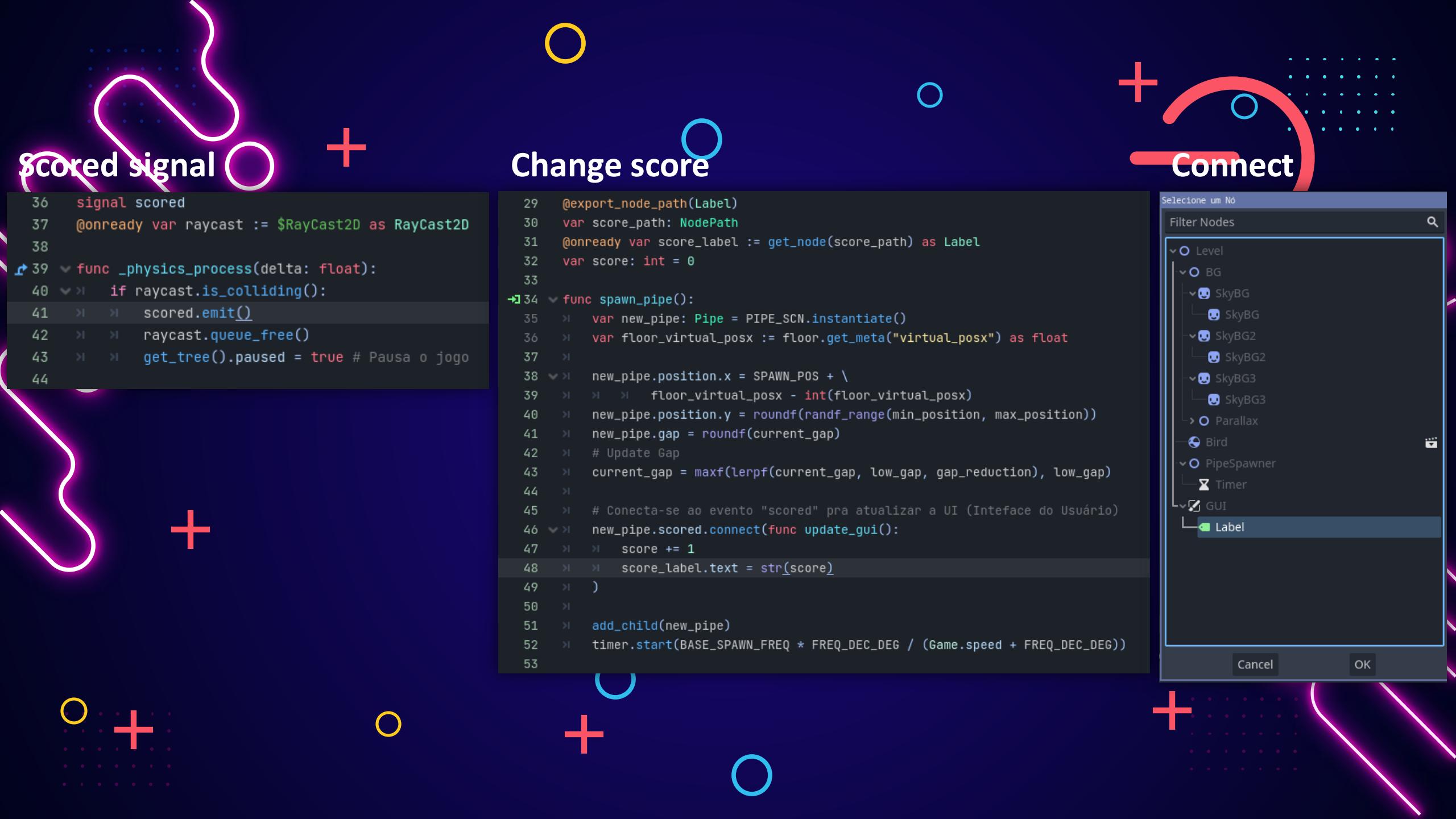


# Inspector



# Settings

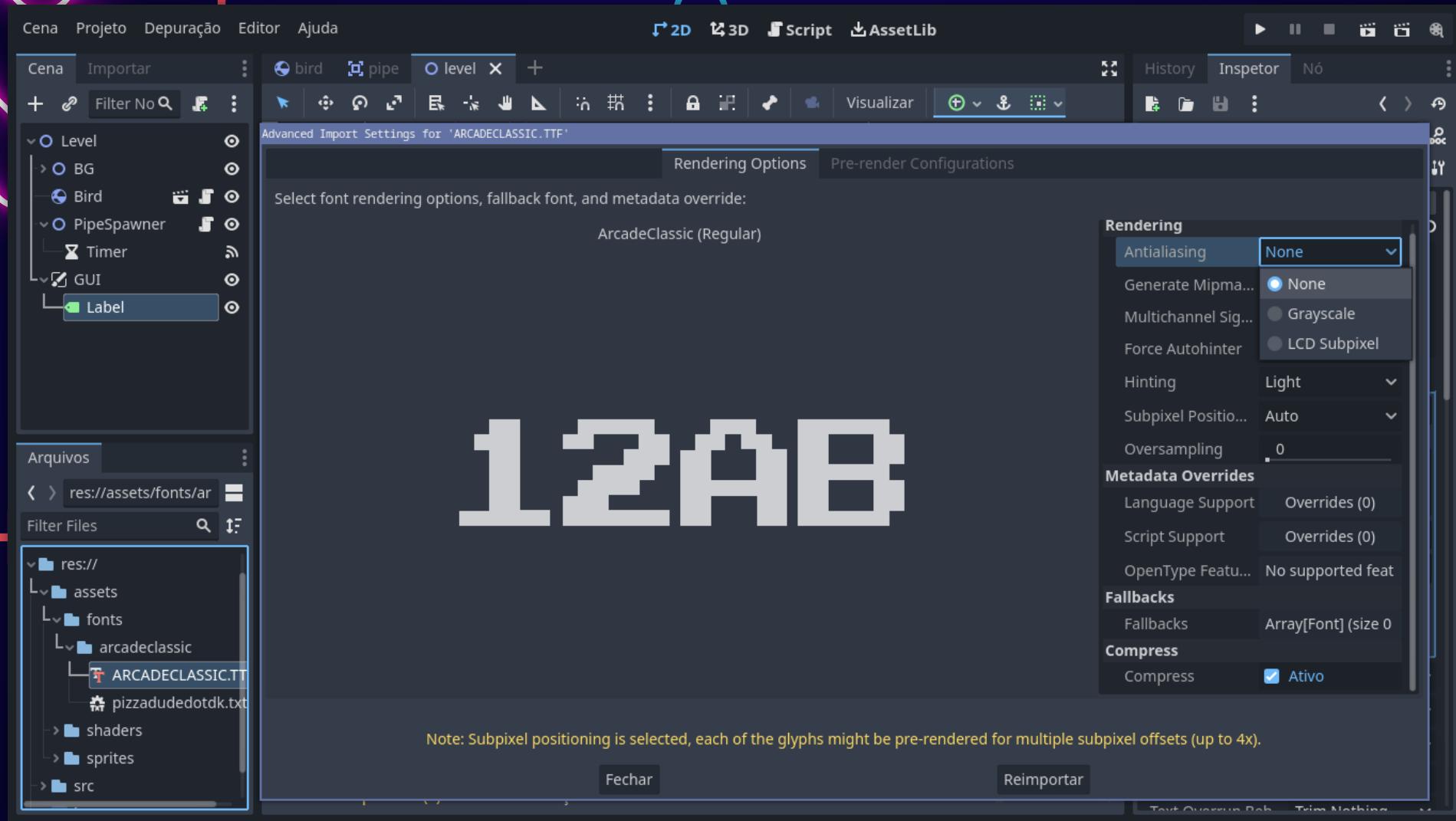




HUD



# Fix aliasing



Final result



# BUGGY COLLISION



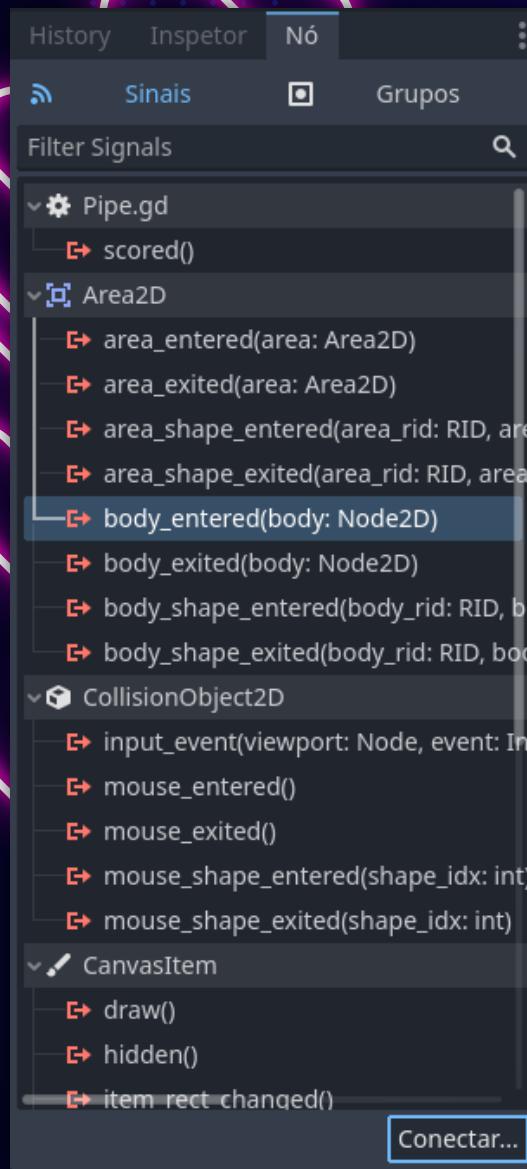
BUGGY COLLISION

EVERYWHERE

Colisões &  
Gameplay<sup>+</sup>  
loop



Body entered



## Export nodes

```
24 # No editor será exportado o path usado pra carregar a instância automaticamente
25 @export var score_label: Label
26 var score: int = 0
27
28 const Bird = preload("res://src/Bird.gd")
29 @export_node_path("PhysicsBody2D")
30 var bird_path: NodePath
31 @onready var bird := get_node(bird_path) as Bird
32
33 @export var freeze_node_paths: Array[NodePath]
34 @export var freeze_nodes: Array[Node] = []
35
36 func _ready():
37     for path in freeze_node_paths:
38         freeze_nodes.append(get_node(path))
39
40     min_position = high_gap
41
```

## Freeze node

PipeSpawner.gd

Gap Reduction	0.01
Low Gap	13
High Gap	26
Min Position	26
Max Position	73
Floor Path	Floor
Score Label	Label
Bird Path	Bird
Freeze Node ...	Array[NodePath]
Tamanho:	2
0	./BG
1	./BG

+ Add Element

## Pause nodes

```
44 func spawn_pipe():
45     var new_pipe: Pipe = PIPE_SCN.instantiate()
46     var floor_virtual_posx := floor_sprite.get_meta("virtual_posx") as float
47     new_pipe.position.x = SPAWN_POS + \ 
48     new_pipe.position.y = roundf(randf_range(min_position, max_position))
49     new_pipe.gap = roundf(current_gap)
50     # Update Gap
51     current_gap = maxf(lerpf(current_gap, low_gap, gap_reduction), low_gap)
52     # Conecta-se ao evento "scored" pra atualizar a UI (Interface do Usuário)
53     new_pipe.scored.connect(func update_gui()):
54         score += 1
55         score_label.text = str(score)
56     )
57     # Conecta-se ao evento "body entered" para resetar o Game Loop
58     new_pipe.body_entered.connect(func restart(_arg):
59         bird.set_process_input(false)
60         bird.apply_impulse(Vector2.LEFT * .2, Vector2.LEFT)
61         await get_tree().physics_frame # Aguarda a física ser processada
62         for node in freeze_nodes:
63             node.process_mode = Node.PROCESS_MODE_DISABLED
64             # Para de processar o nó como se o jogo estivesse pausado para ele
65         )
66         add_child(new_pipe)
67         timer.start(BASE_SPAWN_FREQ * FREQ_DEC_DEG / (Game.speed + FREQ_DEC_DEG))
68     )
69 
```

Collision



# StaticBody2D

Criar Novo Node

Favoritos: Pesquisar: body

Correspondências:

- Node
  - CanvasItem
  - Node2D
    - CollisionObject2D
      - PhysicsBody2D
        - StaticBody2D
          - AnimatableBody2D
          - CharacterBody2D
          - RigidBody2D

Recente:

  - StaticBody2D
  - CanvasLayer
  - Label
  - RayCast2D
  - Node
  - Timer
  - CollisionShape2D
  - Area2D
  - RemoteTransf...
  - Node2D

Descrição:

StaticBody2D: Physics body for 2D physics which is static or moves only by script. Useful for floor and walls.

Cancel Criar

# CollisionShape2D

Criar Novo Node

Favoritos: Pesquisar: collis

Correspondências:

- Node
  - CanvasItem
  - Node2D
    - CollisionPolygon2D
      - CollisionShape2D
- Node3D
  - CollisionPolygon3D
  - CollisionShape3D
- VisualInstance3D
  - GPUParticlesCollision3D
    - GPUParticlesCollisionBox3D
    - GPUParticlesCollisionHeightField3D
    - GPUParticlesCollisionSDF3D
    - GPUParticlesCollisionSphere3D

Recente:

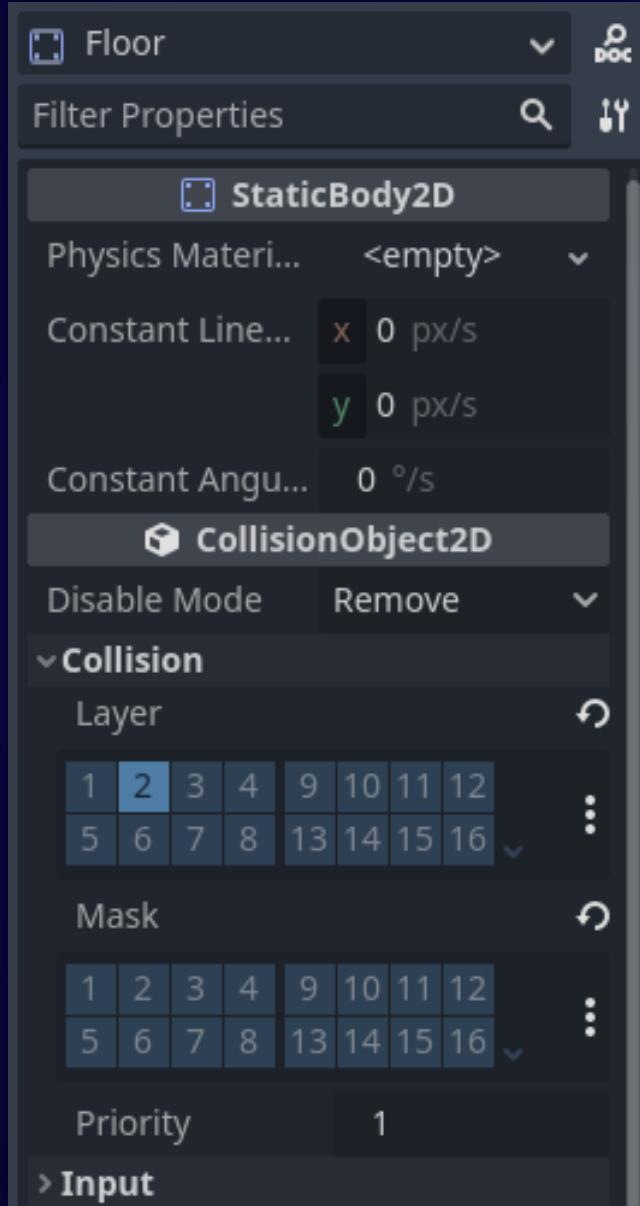
- StaticBody2D
- CanvasLayer
- Label
- RayCast2D
- Node
- Timer
- CollisionShape2D
- Area2D
- RemoteTransf...
- Node2D

Descrição:

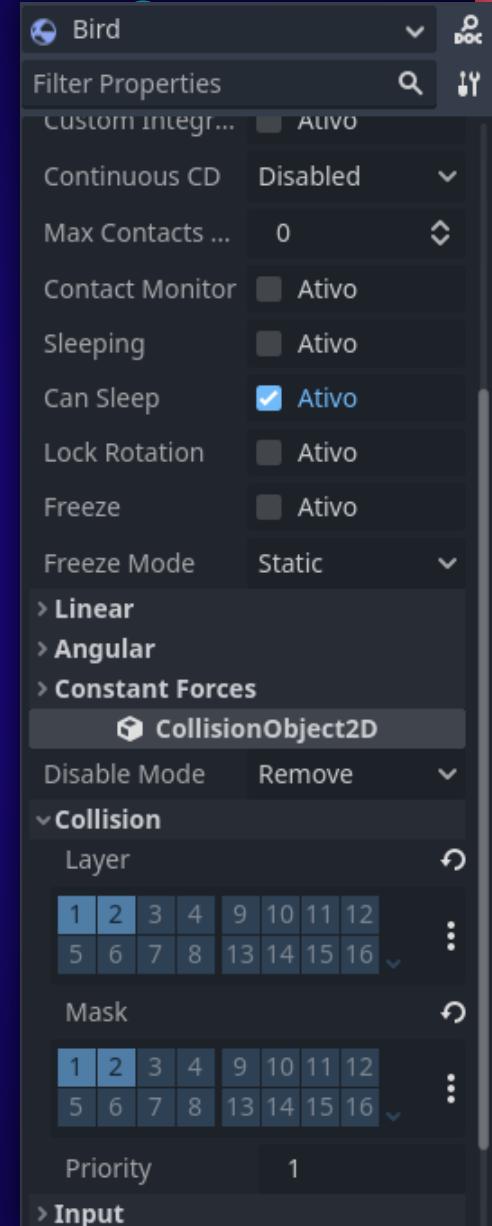
CollisionShape2D: Node that represents collision shape data in 2D space.

Cancel Criar

## Floor layer



## Bird layer



# Layered Collision



# Floor Collision



# Ceil Collision

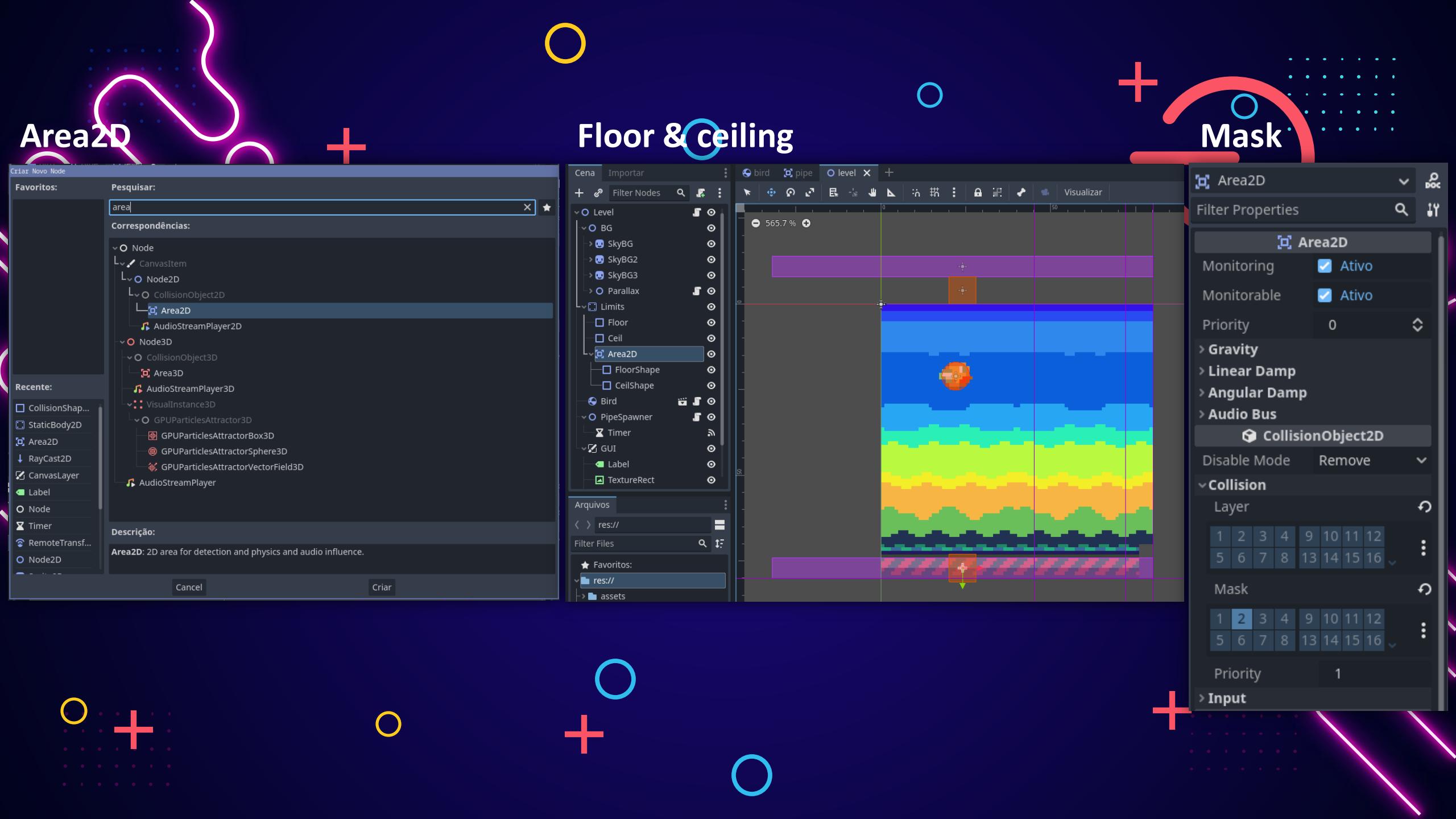


# Pipe Collision



# Pipe tip Collision





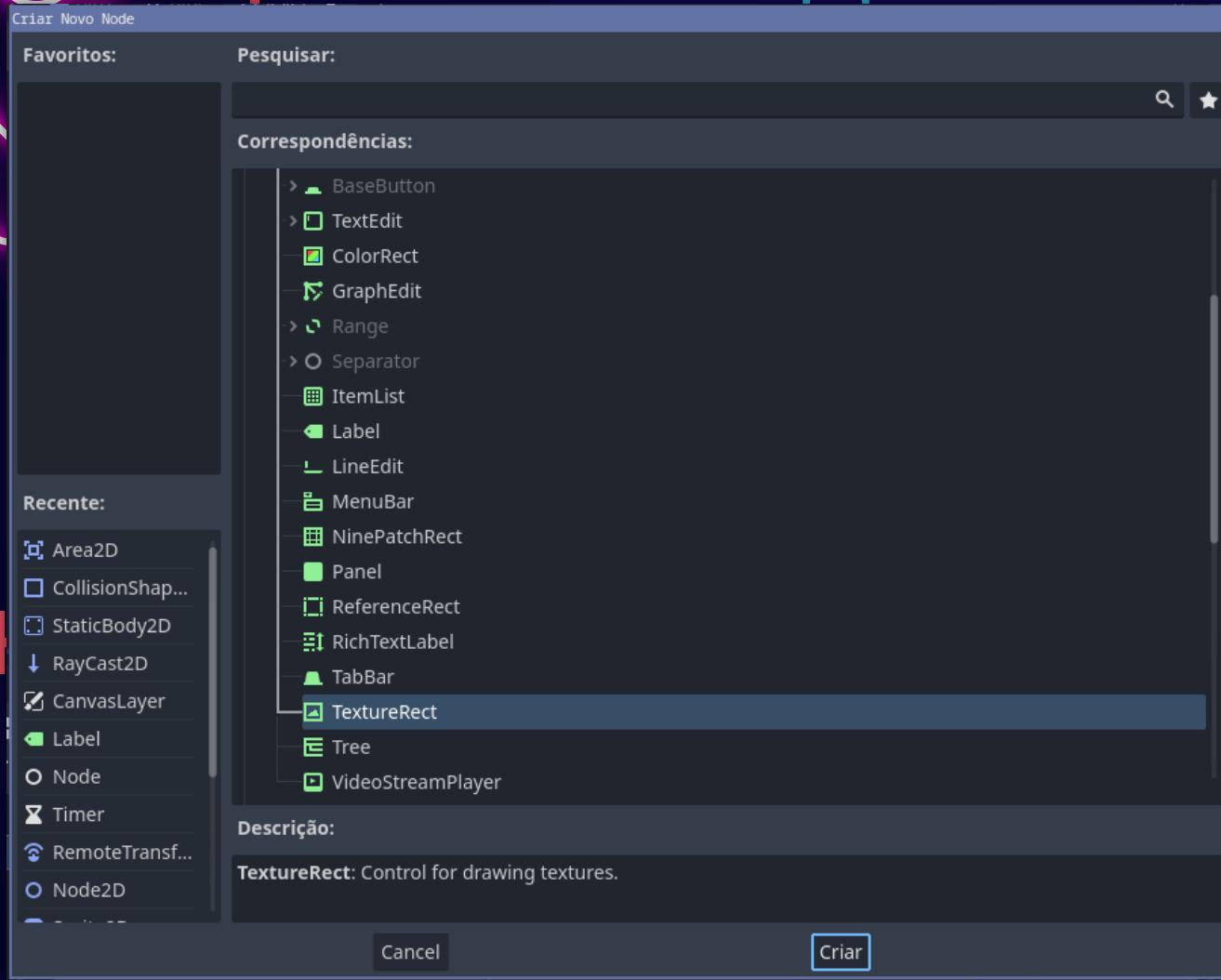
## Function calls

```
43  @export var floor_area: Area2D
44
45  func spawn_pipe():
46    var new_pipe: Pipe = PIPE_SCN.instantiate()
47    var floor_virtual_posx := floor_sprite.get_meta("virtual_posx") as float
48    new_pipe.position.x = SPAWN_POS + \(\)
49    new_pipe.position.y = roundf(randf_range(min_position, max_position))
50    new_pipe.gap = roundf(current_gap)
51    # Update Gap
52    current_gap = maxf(lerpf(current_gap, low_gap, gap_reduction), low_gap)
53    # Conecta-se ao evento "scored" pra atualizar a UI (Interface do Usuário)
54    new_pipe.scored.connect(func update_gui()):
55      score += 1
56      score_label.text = str(score)
57    )
58    # Conecta-se ao evento "body entered" para resetar o Game Loop
59    new_pipe.body_entered.connect(func(_arg):
60      bird.apply_impulse(Vector2.LEFT * .2, Vector2.LEFT)
61      restart()
62    )
63    floor_area.body_entered.connect(func(_arg):
64      bird.apply_impulse(Vector2.UP * .2, Vector2(randf_range(-1, 1), -1))
65      bird.set_process_input(false)
66      floor_area.queue_free()
67      restart()
68    )
69    add_child(new_pipe)
70    timer.start(BASE_SPAWN_FREQ * FREQ_DEC_DEG / (Game.speed + FREQ_DEC_DEG))
71
72
```

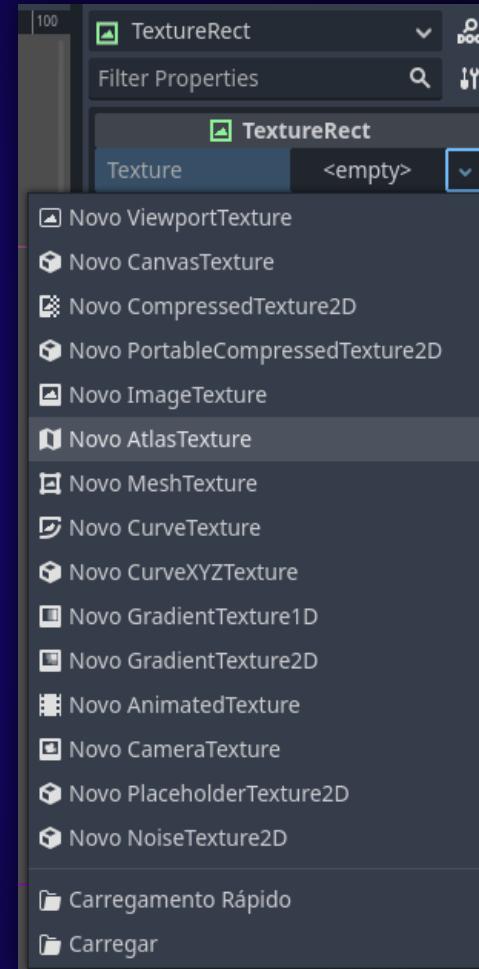
## Restart function

```
72  func restart():
73    bird.set_process_input(false)
74    await get_tree().physics_frame # Aguarda a física ser processada
75
76    for node in freeze_nodes:
77      node.process_mode = Node.PROCESS_MODE_DISABLED
78      # Para de processar o nó como se o jogo estivesse pausado para ele
79
```

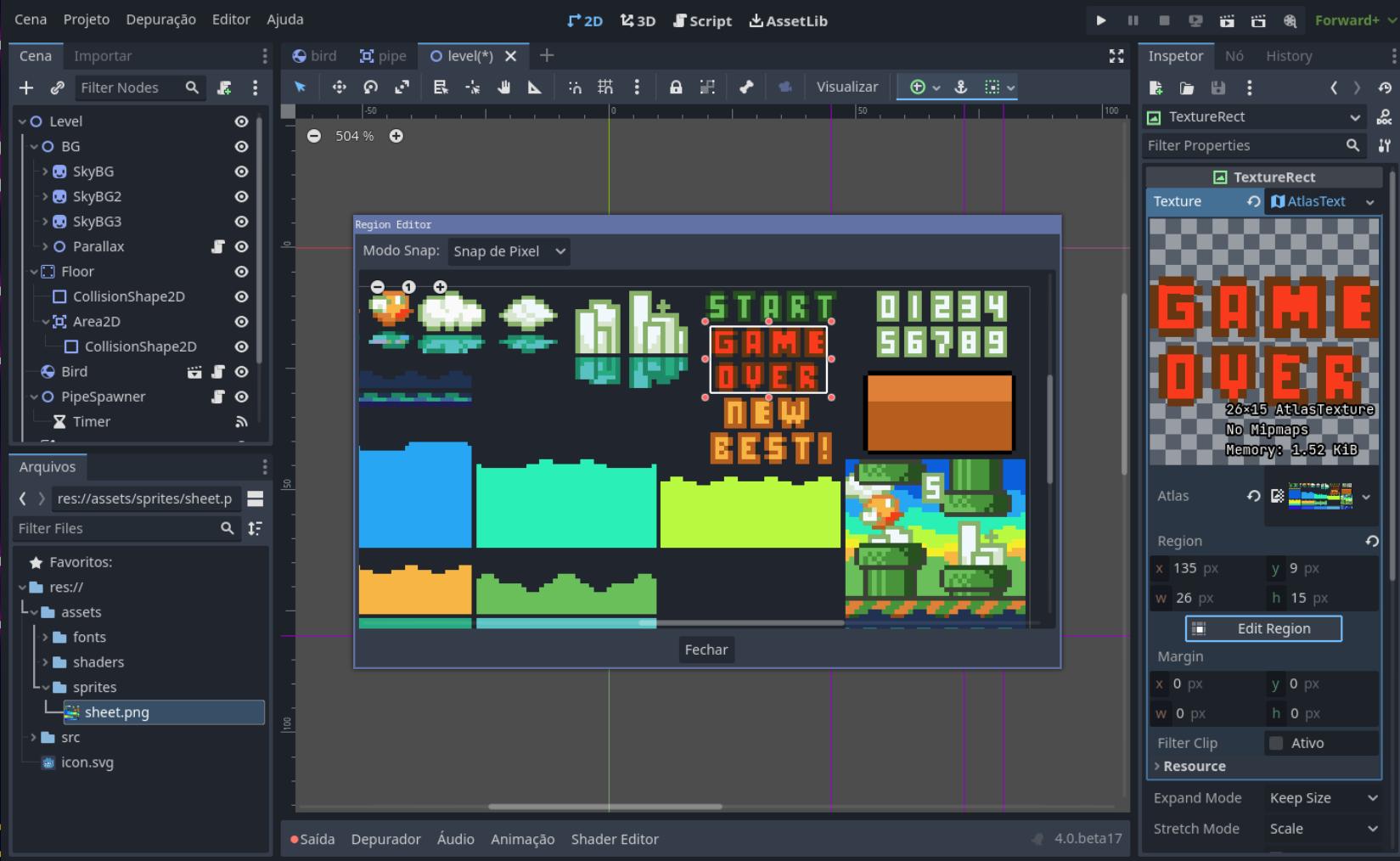
# TextureRect



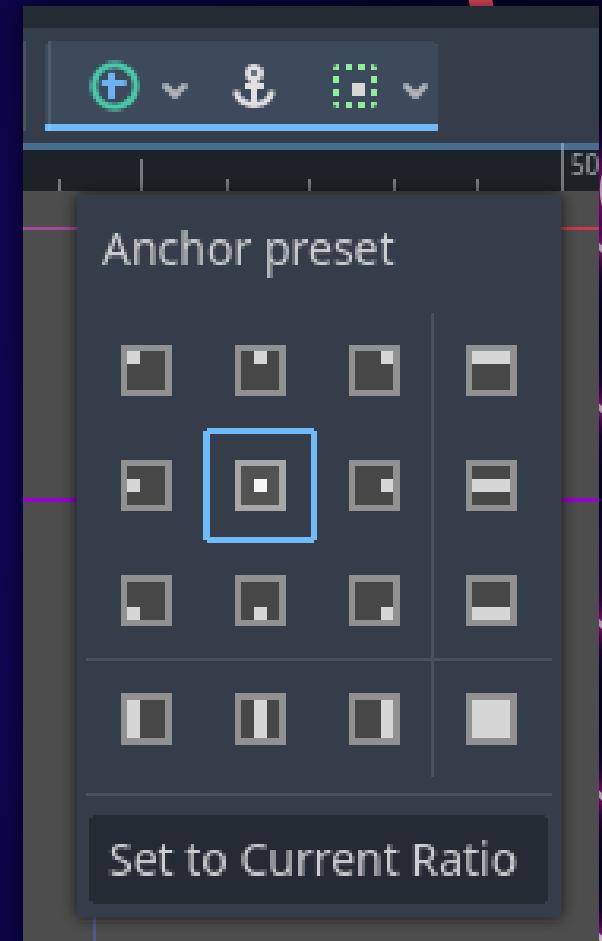
# AtlasTexture



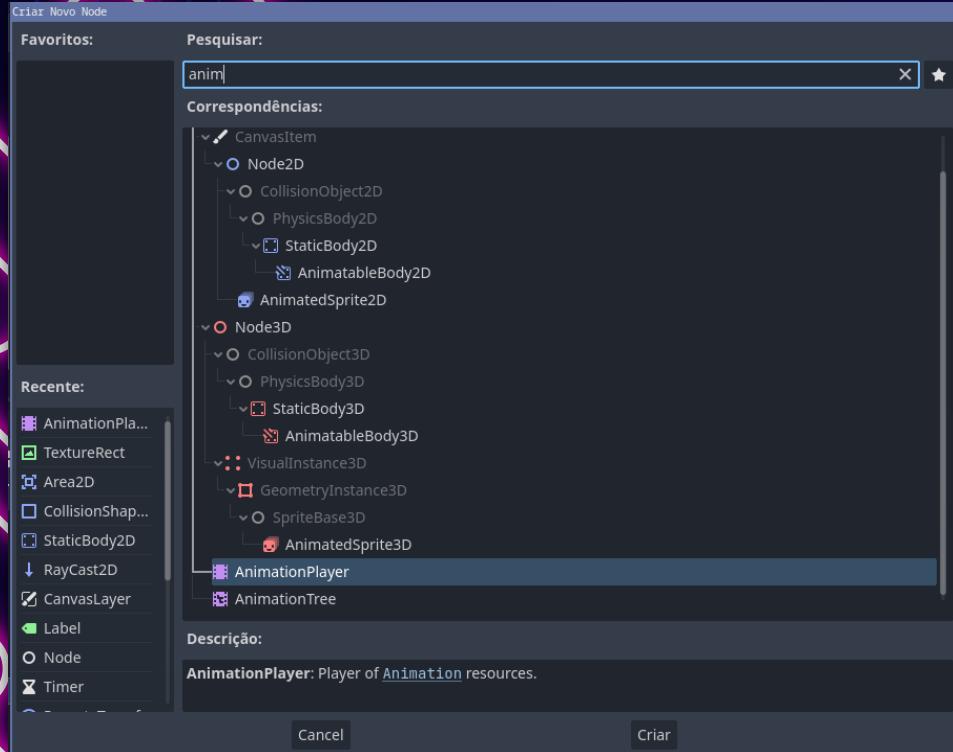
# Edit Region



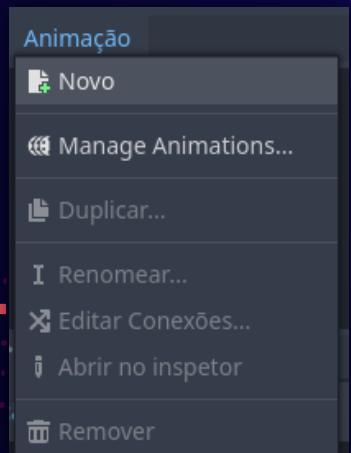
# Center



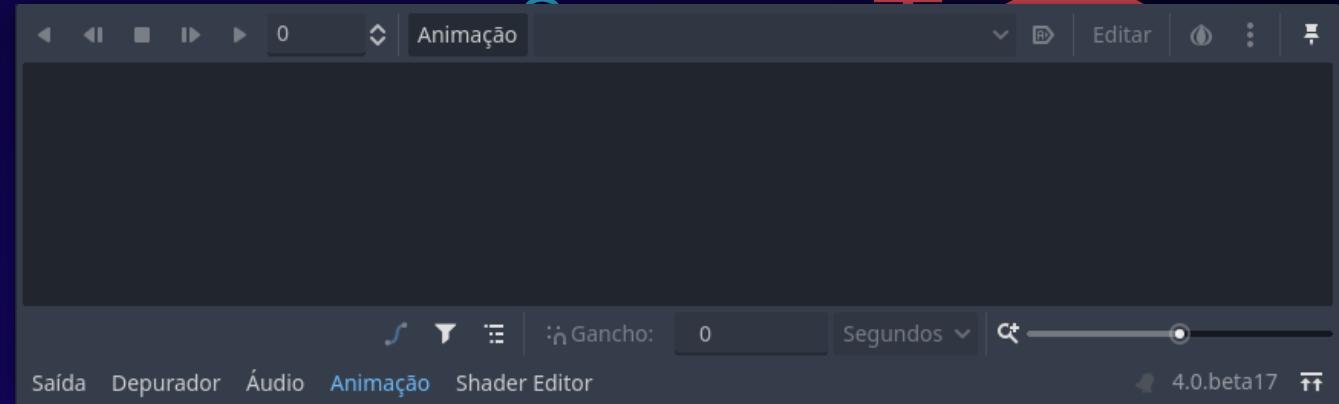
# AnimationPlayer



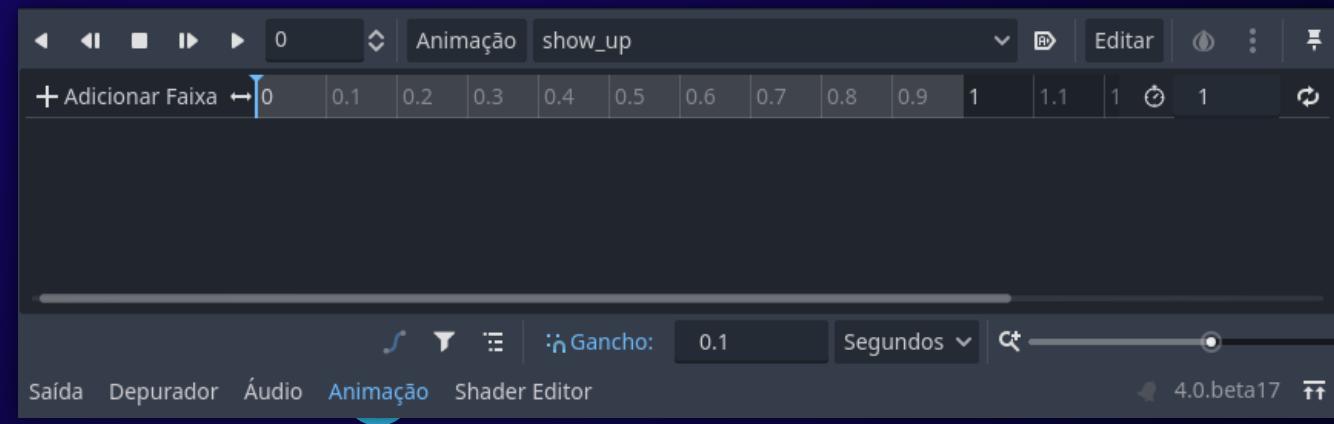
Nova animação



# Animation dock



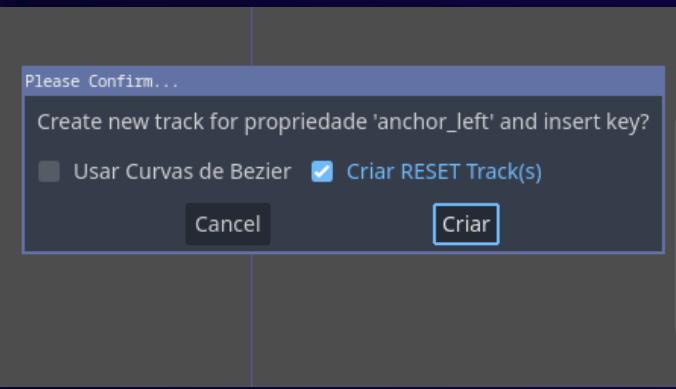
Animation track



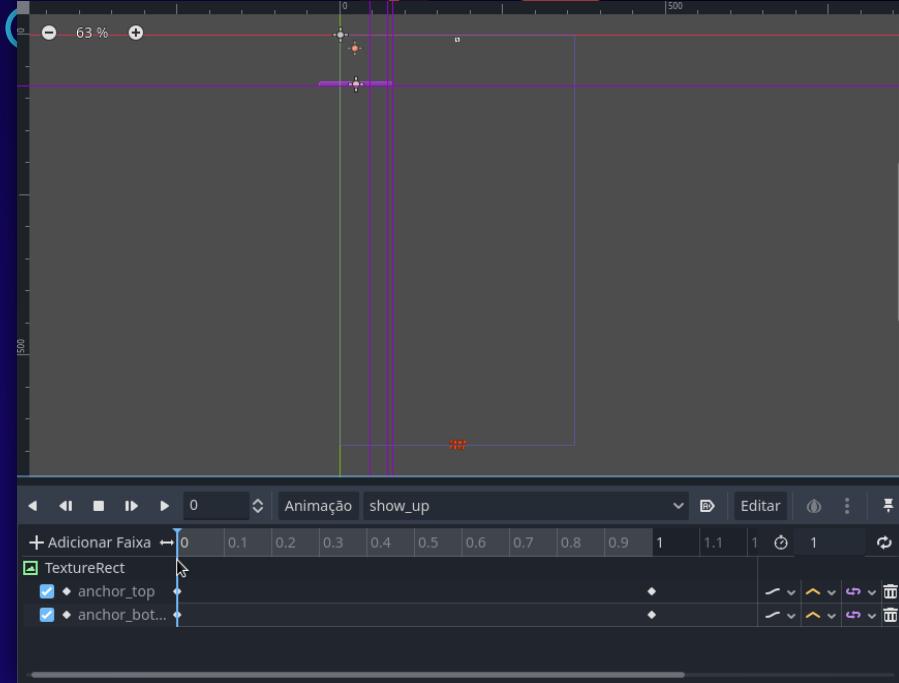
## Anchors



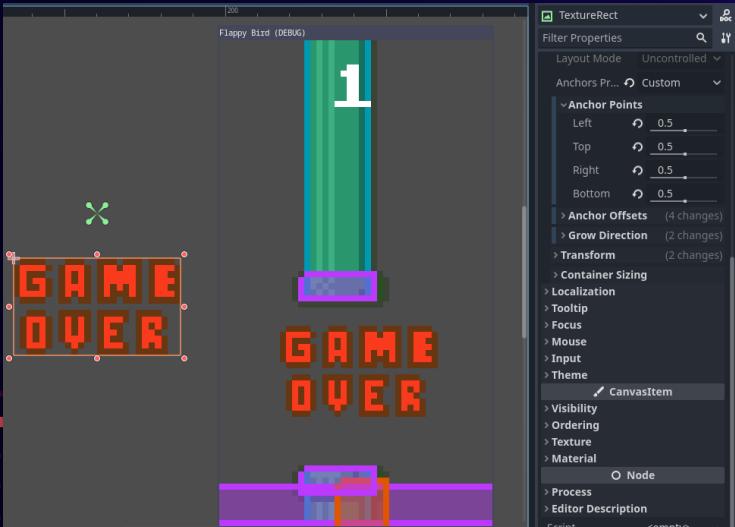
## Keyframe



## "Show up" animation



## Offset



## Play animation

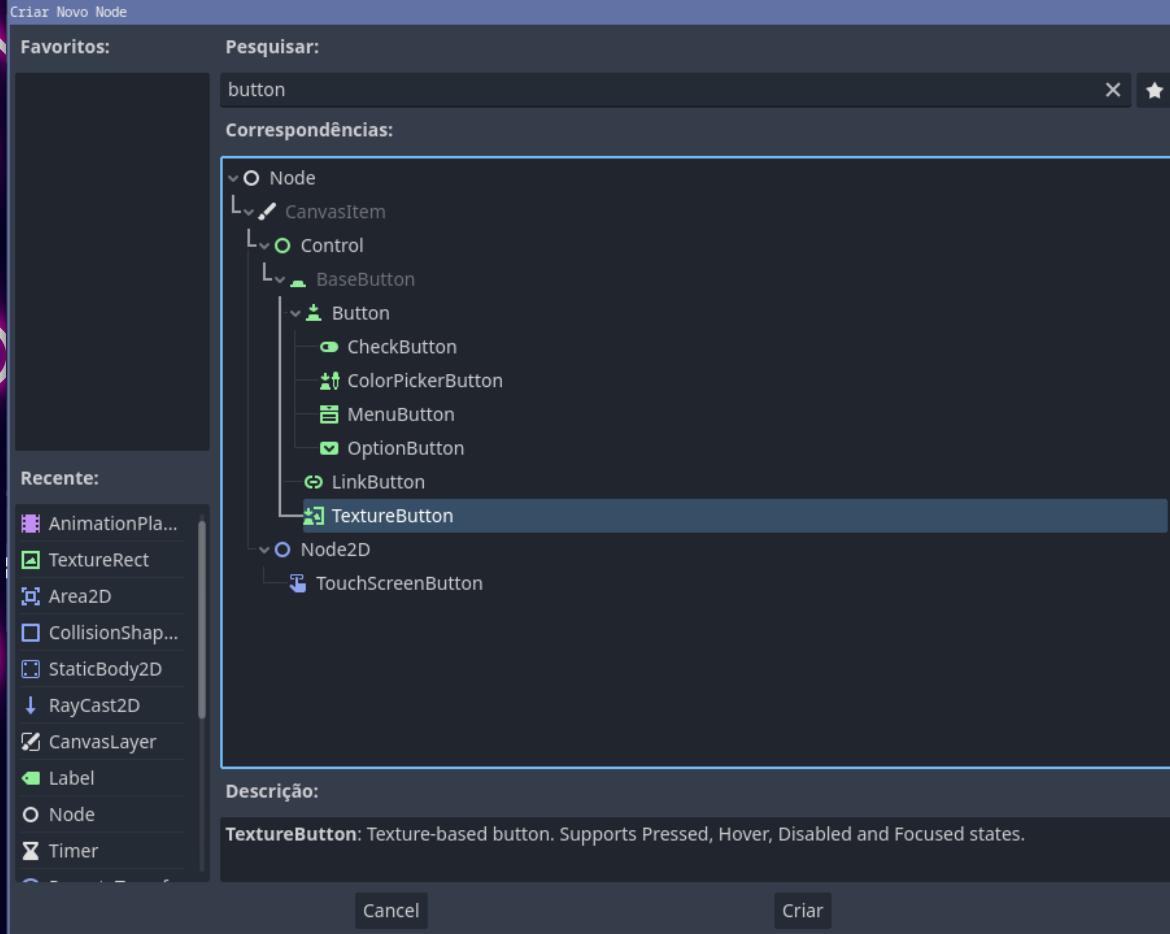
```
74 @export var anim_player: AnimationPlayer
75
76 func restart():
77     bird.set_process_input(false)
78     bird.jump_auto = func(_arg): pass
79
80     await get_tree().physics_frame # Aguarda a física ser processada
81
82     for node in freeze_nodes:
83         node.process_mode = Node.PROCESS_MODE_DISABLED
84         # Para de processar o nó como se o jogo estivesse pausado para ele
85
86     anim_player.play("show_up")
```



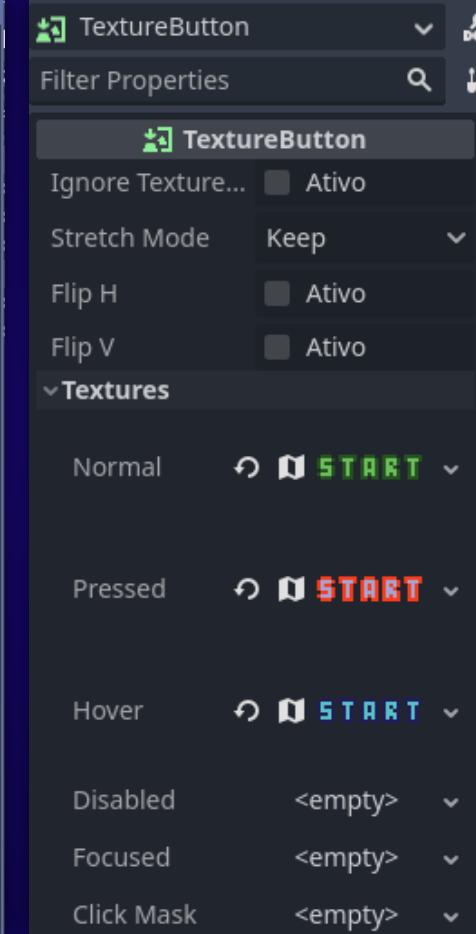
Gameover



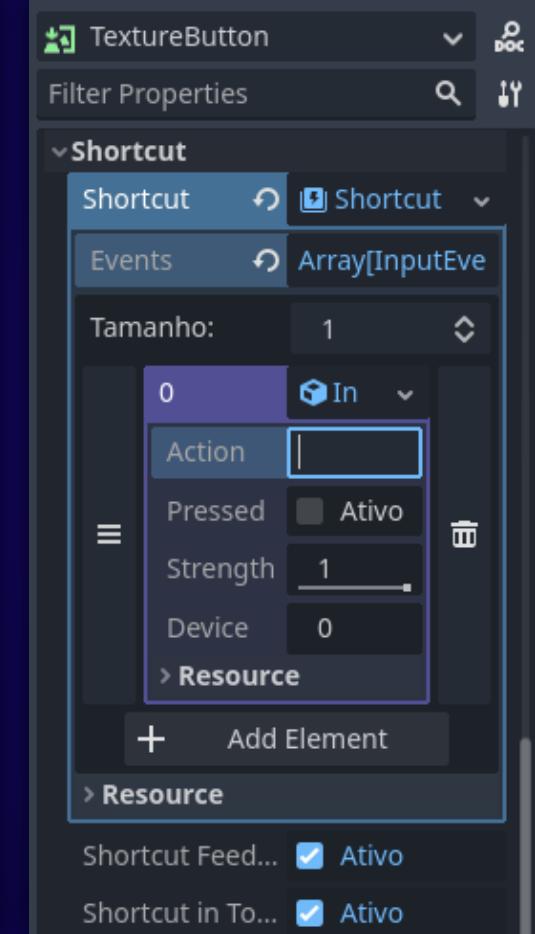
# TextureButton+



# Texture



# Shortcut



# Input Actions

Configurações do Projeto (project.godot)

Geral Mapa de Entrada Localização Autoload Shader Globals Plugins Padrões de Importação

Filter by name... Filter by event... Clear All

Add New Action Adicionar Show Built-in Actions

Ação	Zona morta
ui_accept	0.5
Enter	
Kp Enter	
Space	
ui_select	0.5
Joypad Button 3 (Top Action, Sony Triangle, Xbox Y, Nintendo X) - Dispositivo 0	
Space	
ui_cancel	0.5
Escape	
ui_focus_next	0.5
Tab	
ui_focus_prev	0.5
Shift+Tab	
ui_left	0.5
Left	
Joypad Button 13 (D-pad Left) - Dispositivo 0	

Fechar

ui\_accept

StartButton

Filter Properties

Shortcut

Shortcut Events Array[InputEvent] (s)

Tamanho: 1

0

Action ui\_accept

Pressed Ativo

Strength 1

Device 0

> Resource (1 change)

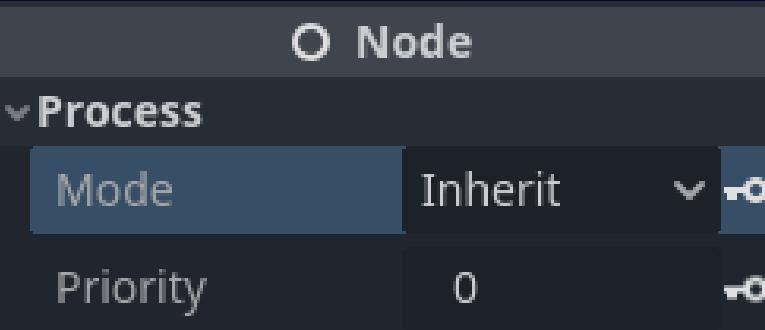
+ Add Element

> Resource (1 change)

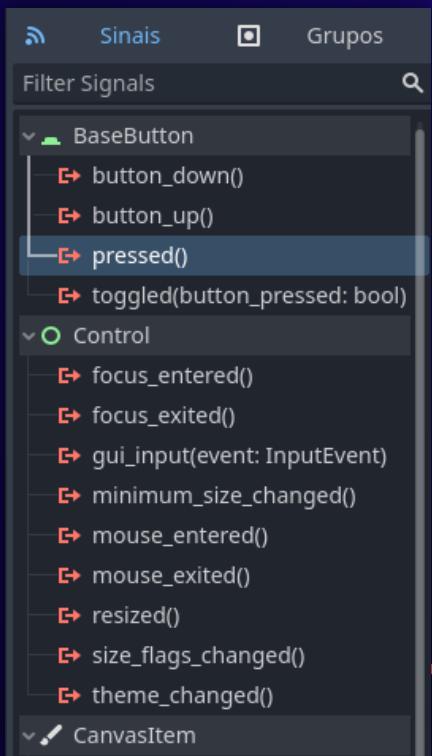
Shortcut Feedback Ativo

Shortcut in Tooltip Ativo

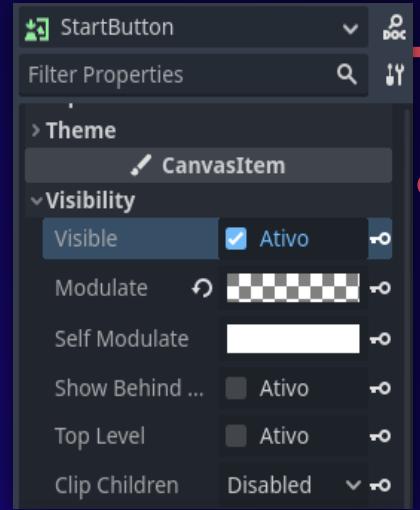
# Process mode



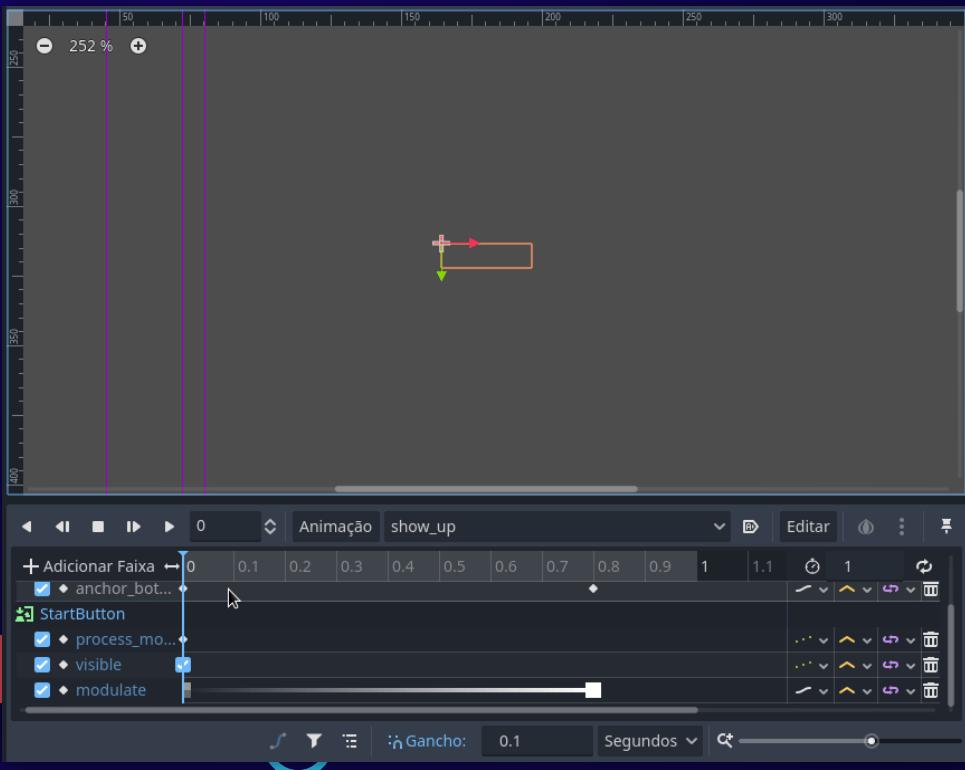
# Signal



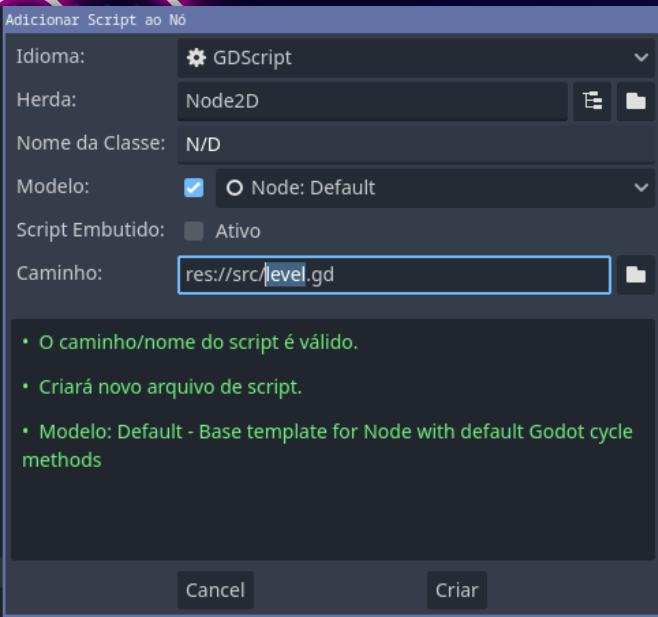
# Visibility



# Animation



# Level script



## New game

```
1 extends Node2D
2
3
4 func new_game():
5   Game.reset()
6   get_tree().reload_current_scene()
7
```

Conexões com o método:  
new\_game  
Origem Sinal Destino  
StartButton pressed Level

# Game reset

```
1 extends Node
2
3 @export var speed_growth_factor: float = 1.
4 ## Taxa de velocidade (rapidez)
5 @export var default_speed: float = 1.
6 @onready var speed: float = default_speed
7
8
9 > func _process(delta):
10
11
12
13
14 ## Reseta os parâmetros do jogo
15 < func reset():
16   speed = default_speed
17
```

## Best score

```
15 signal new_best_achieved
16 var best_score: int = 0
17 < var score: int = 0:
18 < set(value):
19   score = value
20 < if score > best_score:
21     best_score = score
22     new_best_achieved.emit()
23
24
25 ## Reseta os parâmetros do jogo
26 < func reset():
27   speed = default_speed
28   score = 0
29
```

Final result 1



Final result 2



**DO YOUR CODE WITH STATE  
MACHINES, THEY SAID**

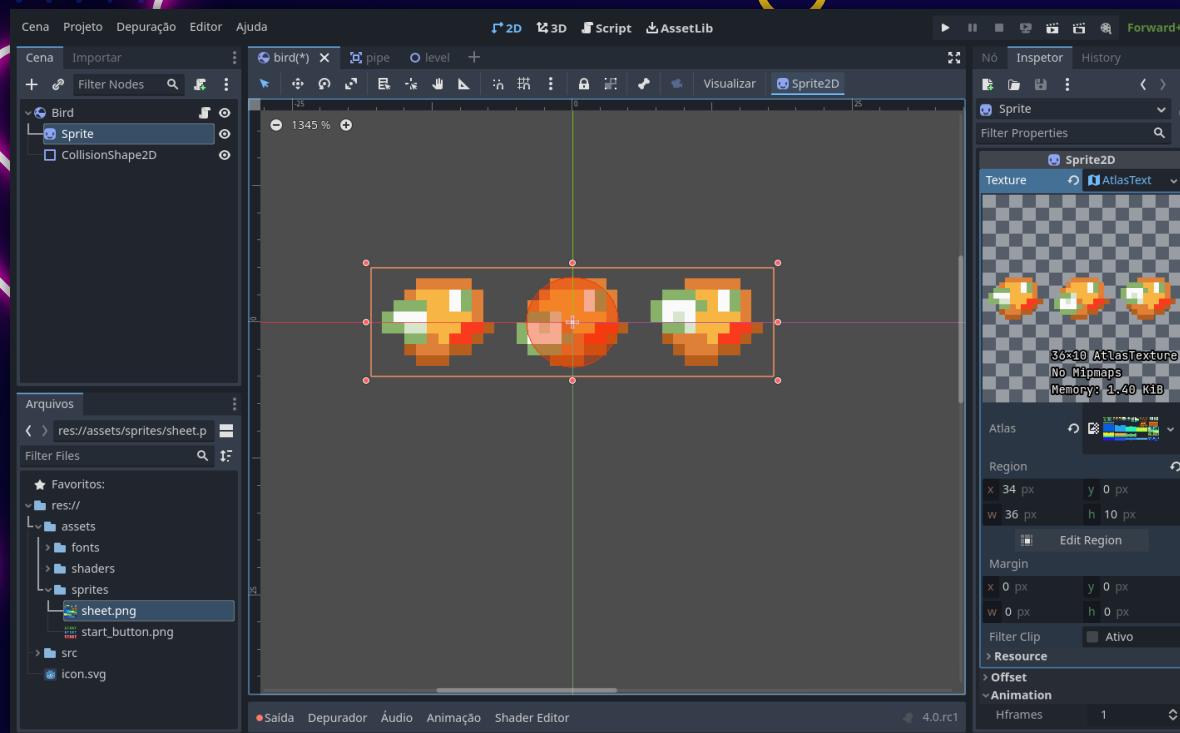


**IT WILL BE SIMPLER, THEY  
SAID**

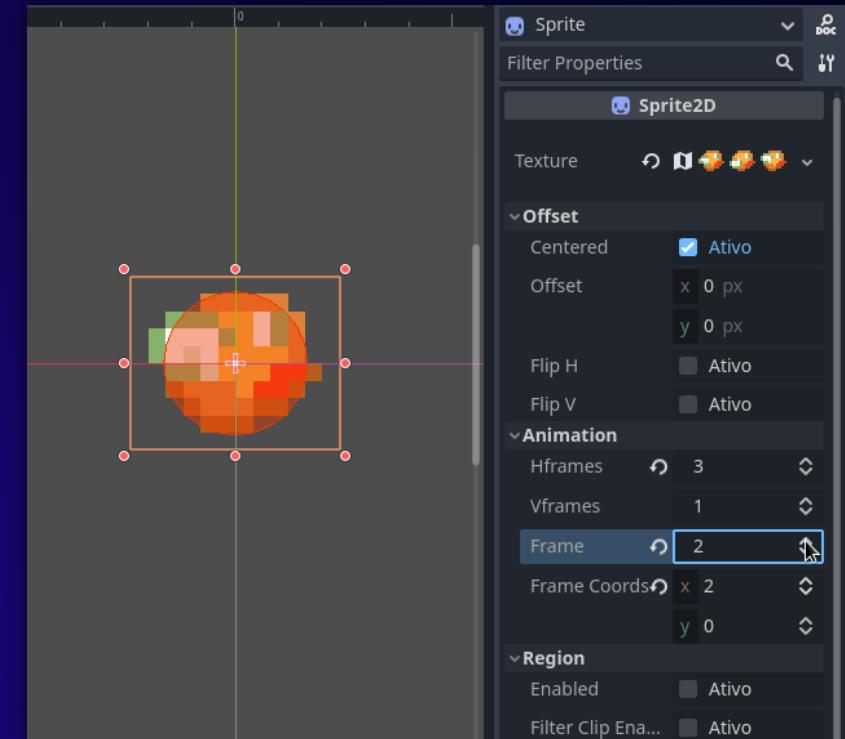
Máquina de  
estado finito  
& áudio



# AtlasTexture



# Animation frames



## State machine

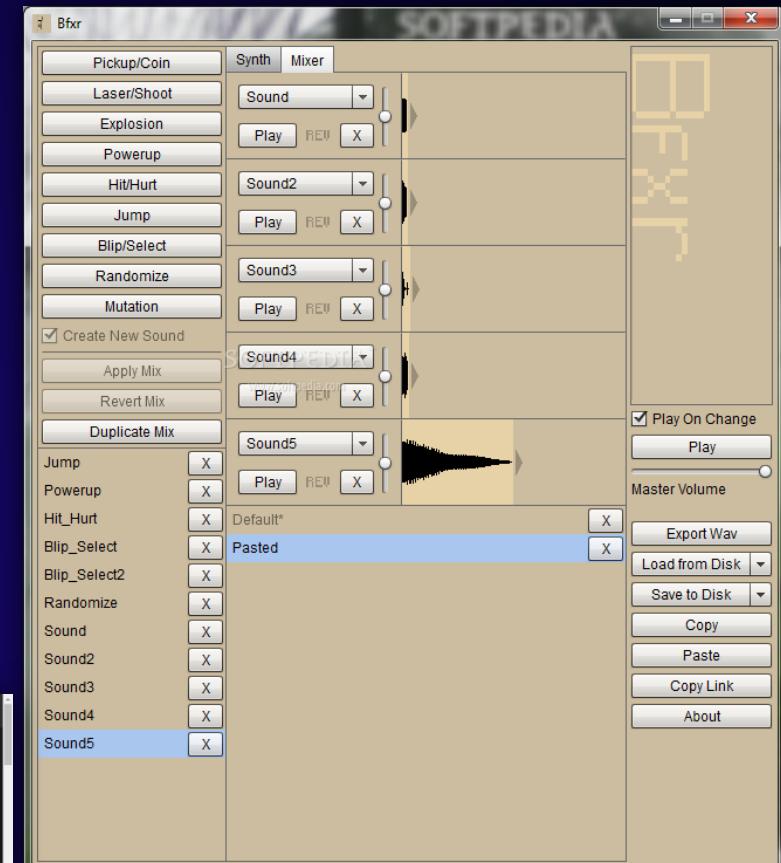
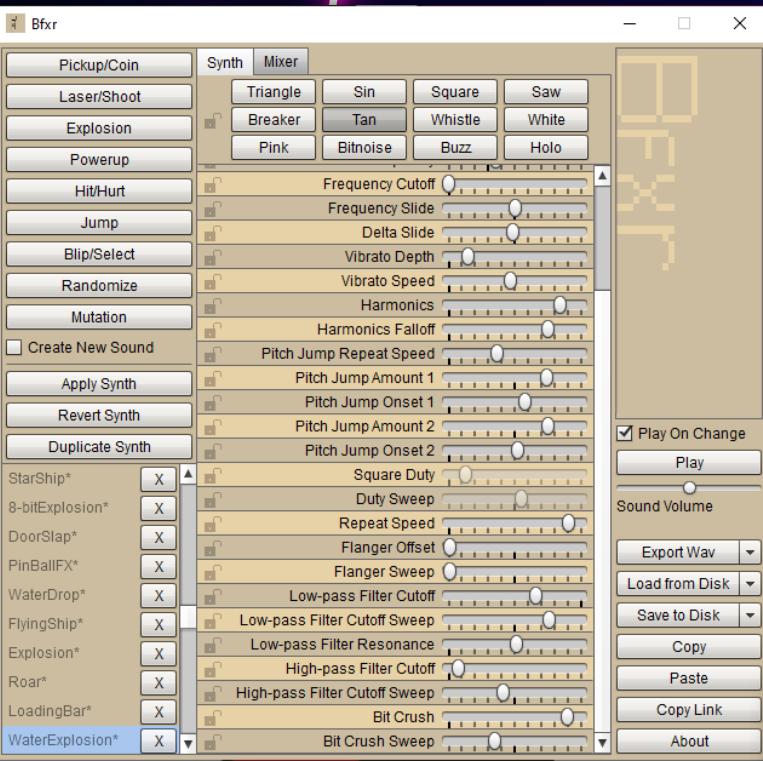
```
26 const ANIM_THRESHOLD: float = 3.
27 enum States { IDLE = 0, FALL, JUMP }
28 @onready var sprite := $Sprite as Sprite2D
29
30 func _physics_process(delta):
31     jump_auto.call(delta)
32
33     if linear_velocity.y < -ANIM_THRESHOLD:
34         sprite.frame = int(States.FALL)
35     elif linear_velocity.y > ANIM_THRESHOLD:
36         sprite.frame = int(States.JUMP)
37     else:
38         sprite.frame = int(States.IDLE)
39
```

# State machine

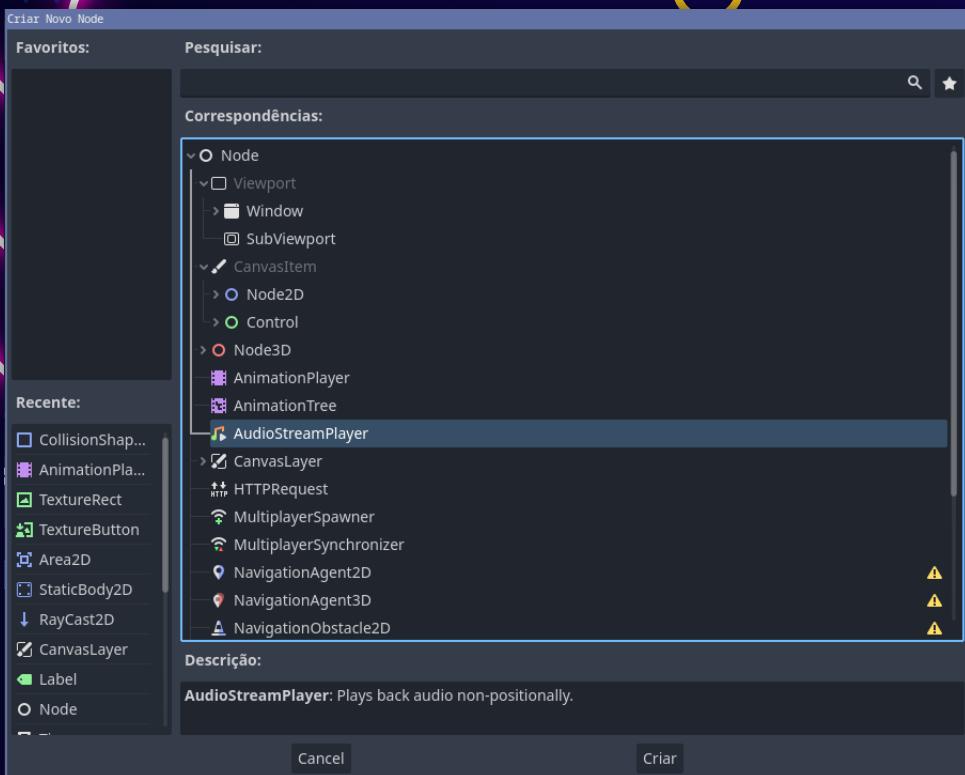


# BFXR

# BFXR mixer



# AudioStreamPlayer



# Add audio



## Play sfx

```
3  const MIN_PITCH: float = .75
4  const MAX_PITCH: float = 1.2
5
6  @export var jump_force: float = 10.
7  @onready var sfx_player := $SFX as AudioStreamPlayer
8
9  func _input(event: InputEvent):
10
11     if event.is_action_pressed("action"): # Jump
12         apply_force(Vector2.UP * jump_force, Vector2.UP)
13         sfx_player.pitch_scale = randf_range(MIN_PITCH, MAX_PITCH)
14         sfx_player.play()
15
```

Jump sfx



## Collide sfxs

```
43 func spawn_pipe():
44     var new_pipe: Pipe = PIPE_SCN.instantiate()
45     var floor_virtual_posx := floor_sprite.get_meta("virtual_posx") as float
46
47     new_pipe.position.x = SPAWN_POS + \o
48     new_pipe.position.y = roundf(randf_range(min_position, max_position))
49     new_pipe.gap = roundf(current_gap)
50
51     # Update Gap
52     current_gap = maxf(lerpf(current_gap, low_gap, gap_reduction), low_gap)
53
54     # Conecta-se ao evento "scored" pra atualizar a UI (Interface do Usuário)
55     new_pipe.scored.connect(func update_gui()): \o
56
57     new_pipe.body_entered.connect(func(_arg):
58         bird.apply_impulse(Vector2.LEFT * .2, Vector2.LEFT)
59         play_sfx(pipe_collide_sfx)
60         bird_died.emit()
61
62         floor_area.body_entered.connect(func(_arg):
63             bird.apply_impulse(Vector2.UP * .2, Vector2(randf_range(-1, 1), -1))
64             bird.set_process_input(false)
65             floor_area.queue_free()
66             play_sfx(floor_collide_sfx)
67             bird_died.emit()
68
69         )
70
71     )
72
73
74     add_child(new_pipe)
75     timer.start(BASE_SPAWN_FREQ * FREQ_DEC_DEG / (Game.speed + FREQ_DEC_DEG))
76
```

## Play stream

```
78     const MIN_PITCH: float = .75
79     const MAX_PITCH: float = 1.2
80
81     @onready var sfx_player := $SFX as AudioStreamPlayer
82
83     func play_sfx(stream: AudioStreamWAV):
84         sfx_player.stream = stream
85         sfx_player.pitch_scale = randf_range(MIN_PITCH, MAX_PITCH)
86         sfx_player.play()
87
```

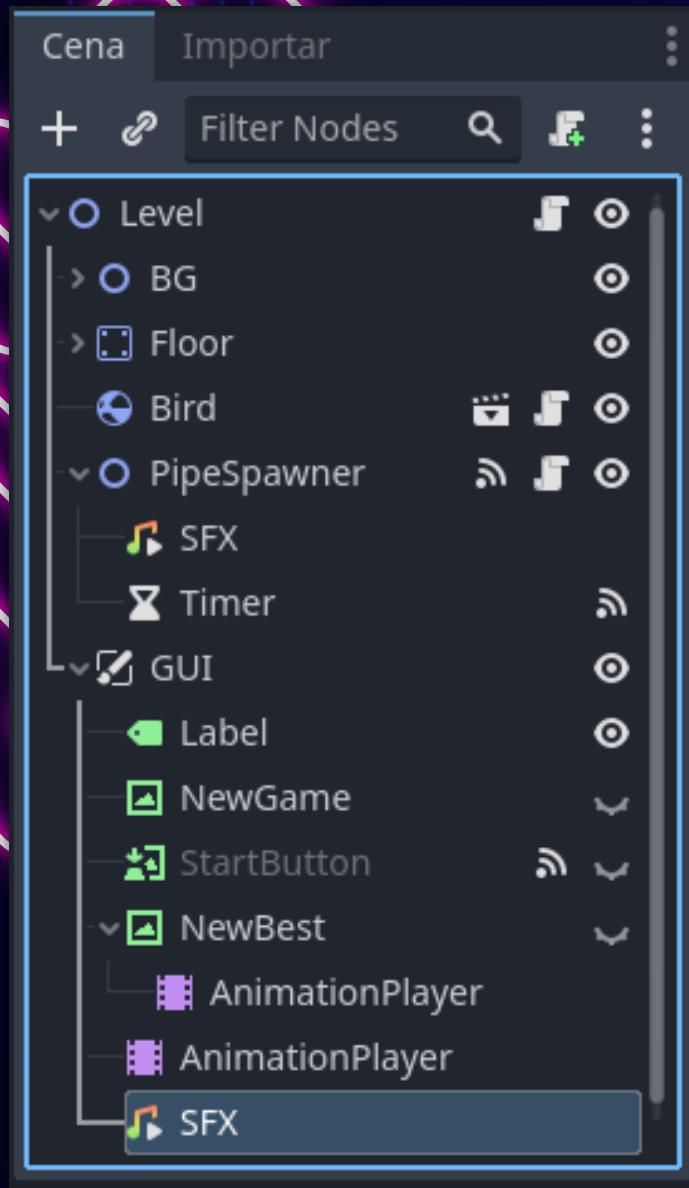
Collide floor sfx



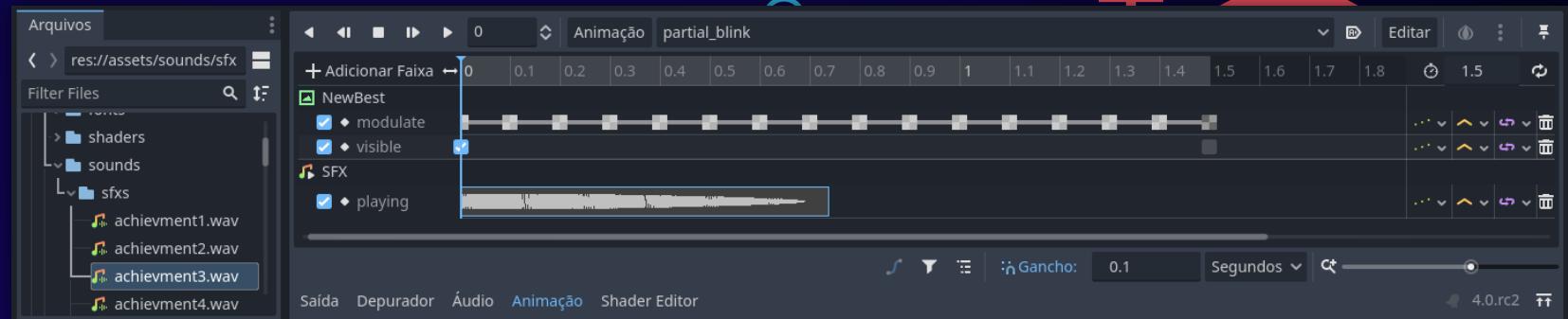
Collide pipe sfx



# Add sfx to animation



# Add achievement sfx



## Play new score sfx

```
20 @export var new_score_sfx: AudioStreamWAV
21 @export var game_over_sfx: AudioStreamWAV
22
23 func _on_bird_died():
24     var bird = $Bird
25     bird.set_process_input(false)
26
27     await get_tree().physics_frame # Aguarda a fisica ser processada
28
29     for node in freeze_nodes:
30         node.process_mode = Node.PROCESS_MODE_DISABLED
31         # Para de processar o nó como se o jogo estivesse pausado para ele
32
33         $GUI/AnimationPlayer.play("show_up")
34         if is_new_best:
35             $GUI/NewBest/AnimationPlayer.play("blink")
36             $GUI/SFX.stream = new_score_sfx
37             is_new_best = false
38         else:
39             $GUI/SFX.stream = game_over_sfx
40
41         $GUI/SFX.play()
```

All sfx's

1

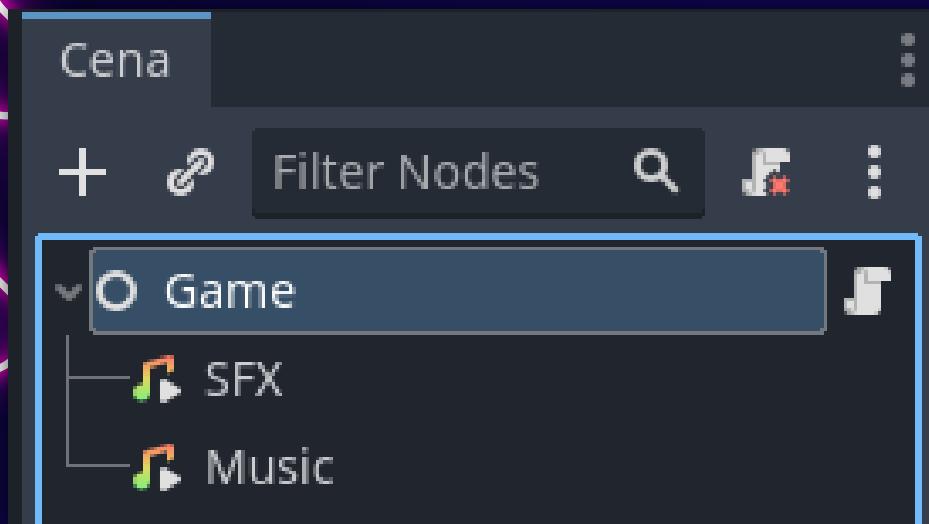
NEW  
BEST!

START

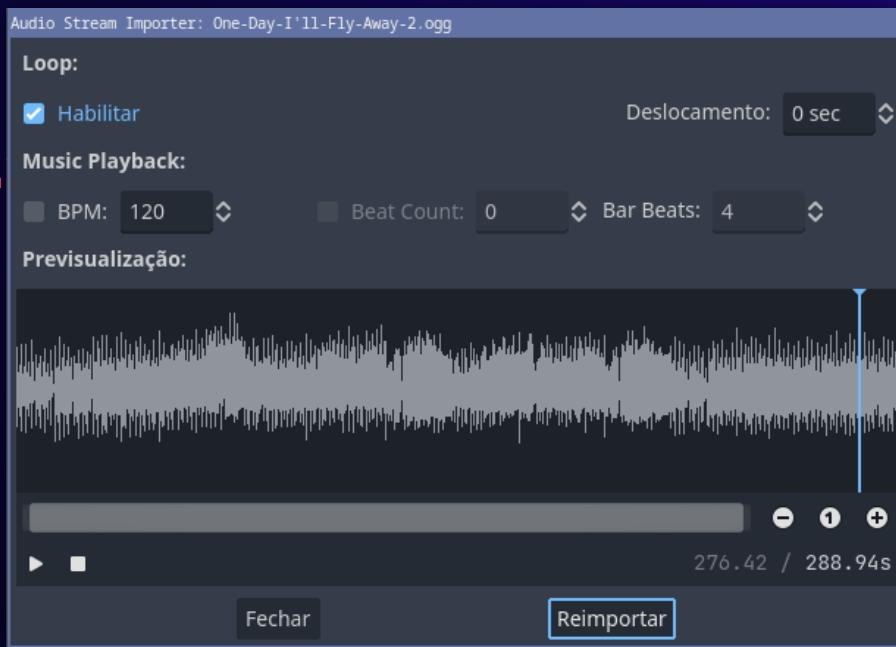
GAME  
OVER



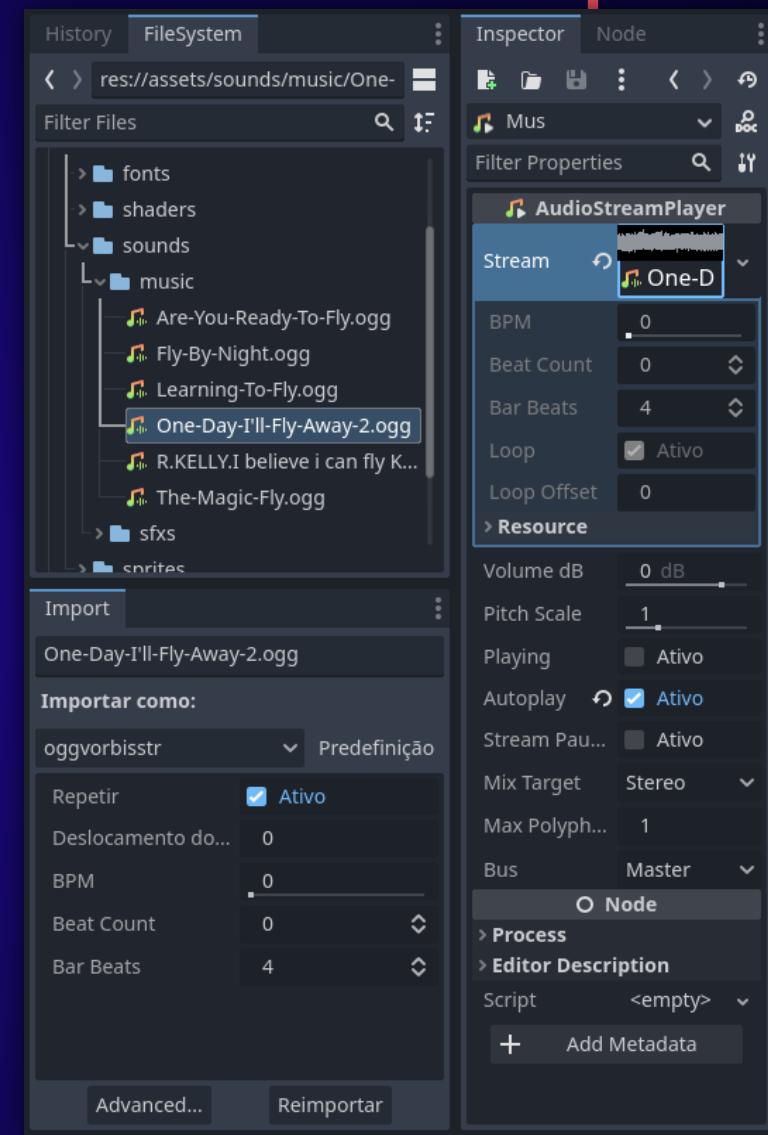
## Game scene



## Music loop



## Music stream

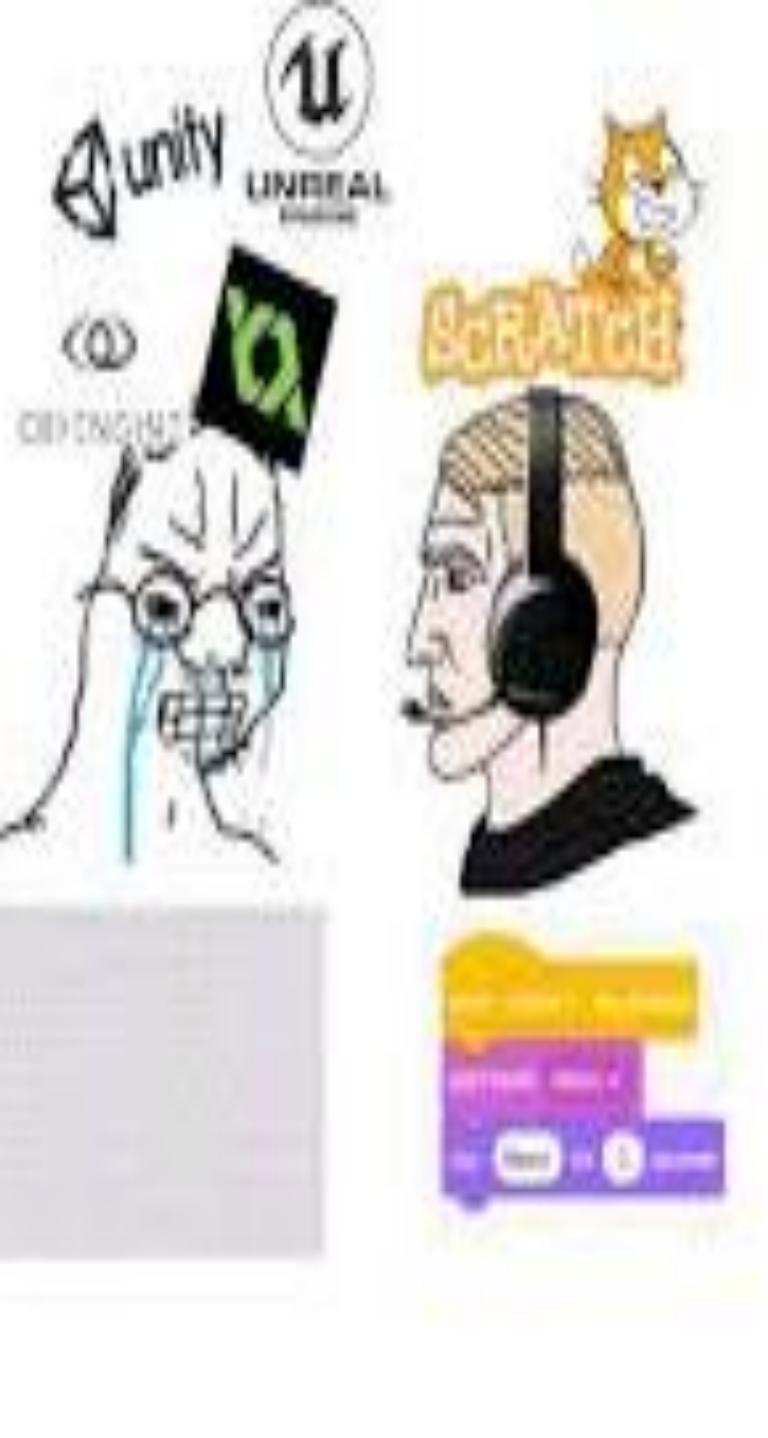


## Play score sfx

```
14    signal new_best_achieved
15
16    const MIN_PITCH: float = 1.
17    const MAX_PITCH: float = 1.75
18
19    var best_score: int = 0
20    var score: int = 0
21    @onready var score_sfx := $SFX as AudioStreamPlayer
22
23    func score_up():
24        score += 1
25        score_sfx.pitch_scale = randf_range(MIN_PITCH, MAX_PITCH)
26        score_sfx.play()
27
28        if score > best_score:
29            best_score = score
30            new_best_achieved.emit()
31
```

# Gameplay





Exportando



# Simular touch

Configurações do Projeto (project.godot)

Geral

Mapa de Entrada Localização Autoload Shader Globals Plugins Padrões de Importação

Filter Settings

Advanced Settings

input\_devices/pointing/emulate\_touch\_from\_mouse

(All)

bool

bool

Adicionar Excluir

threading

Worker Pool

Dispositivos de Entrada

Apontando

Pen Tablet

Buffering

XR

OpenXR

Shaders

Layer Names

2D Render

3D Render

2D Physics

2D Navigation

3D Physics

3D Navigation

Arquivos

Importar

Simular Toque à Partir do Mouse

Ativo

Simular Mouse à Partir do Touch

Ativo

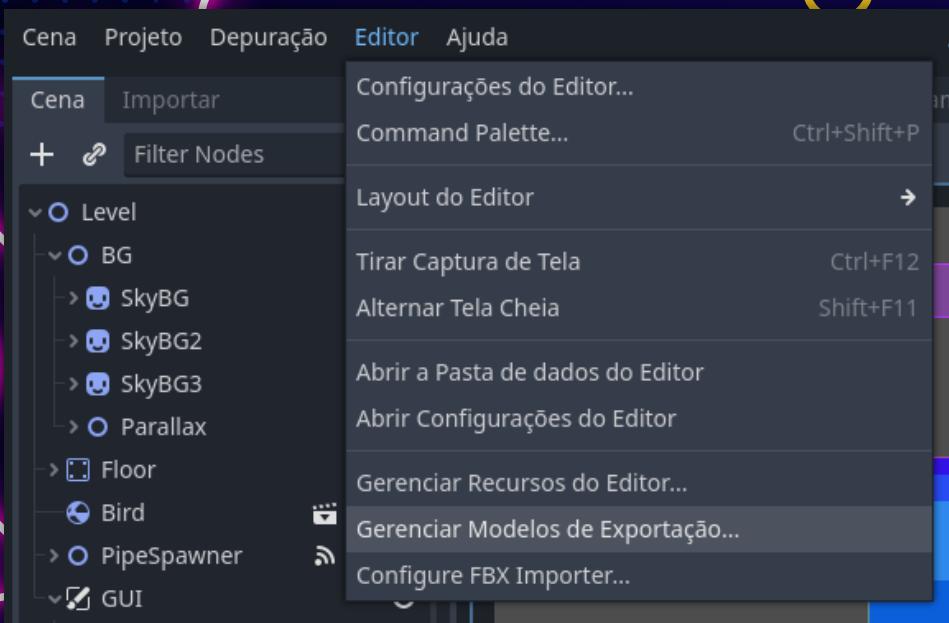
iOS

Touch Delay

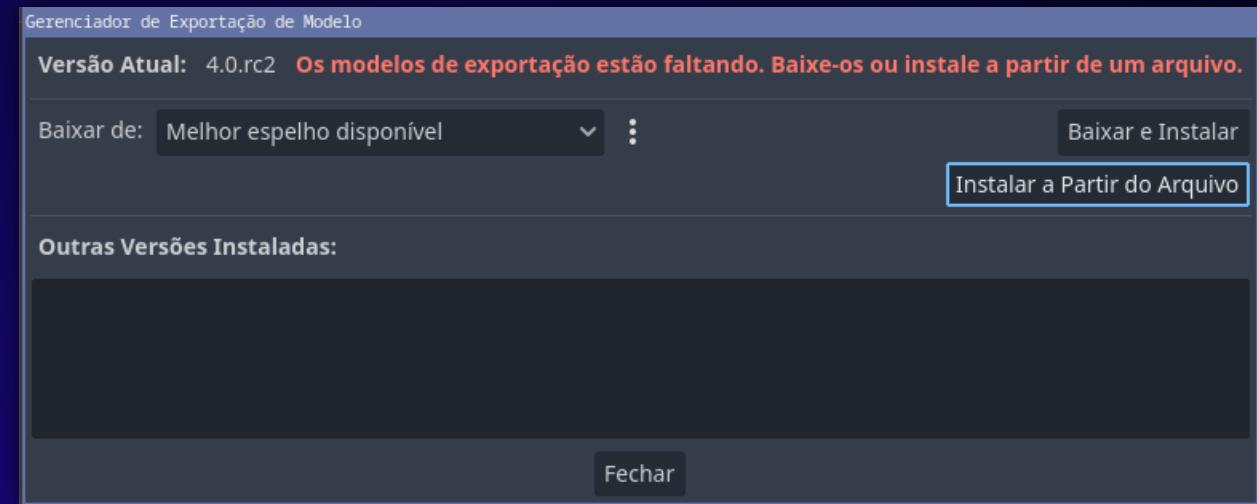
0.15

Fechar

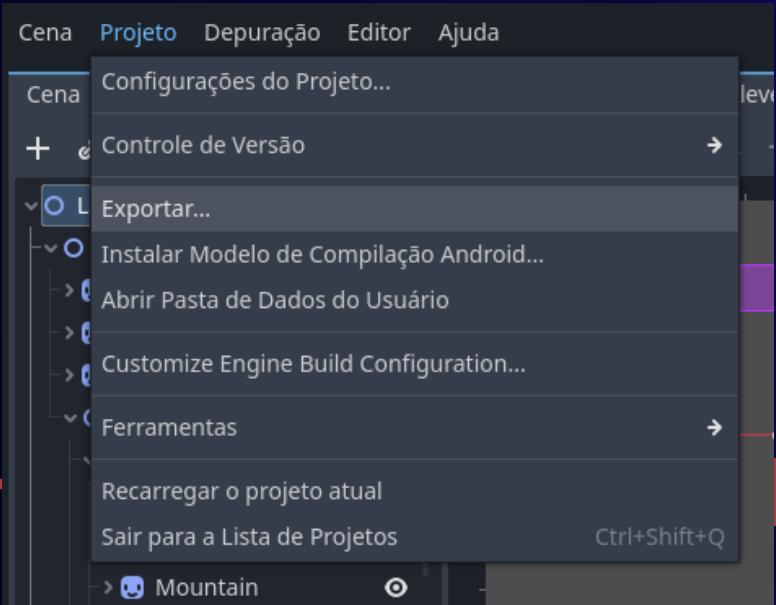
# Modelos de exportação



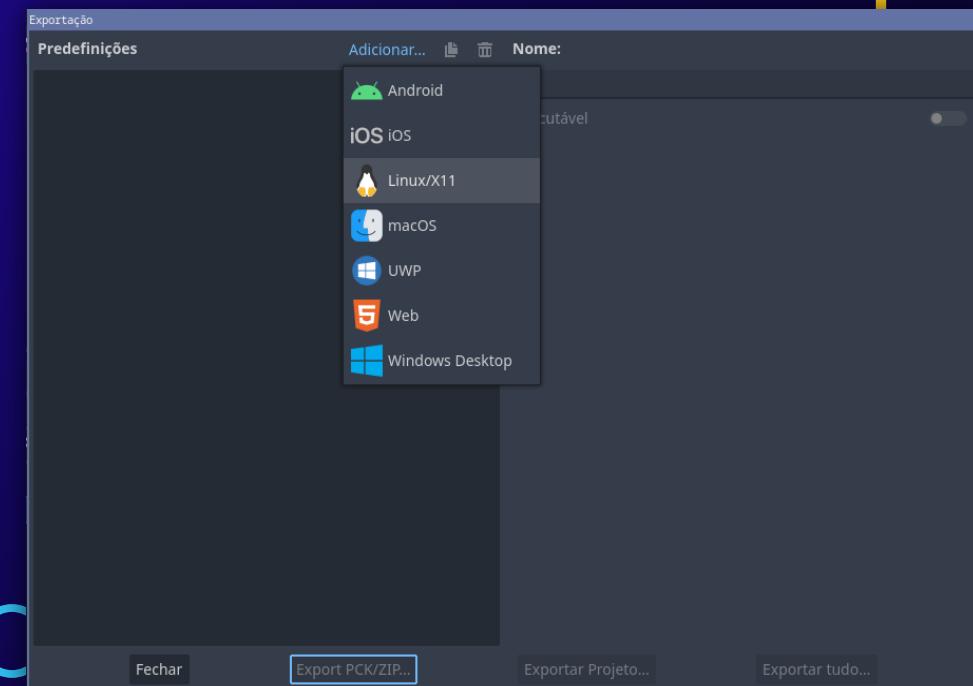
# Gerenciador

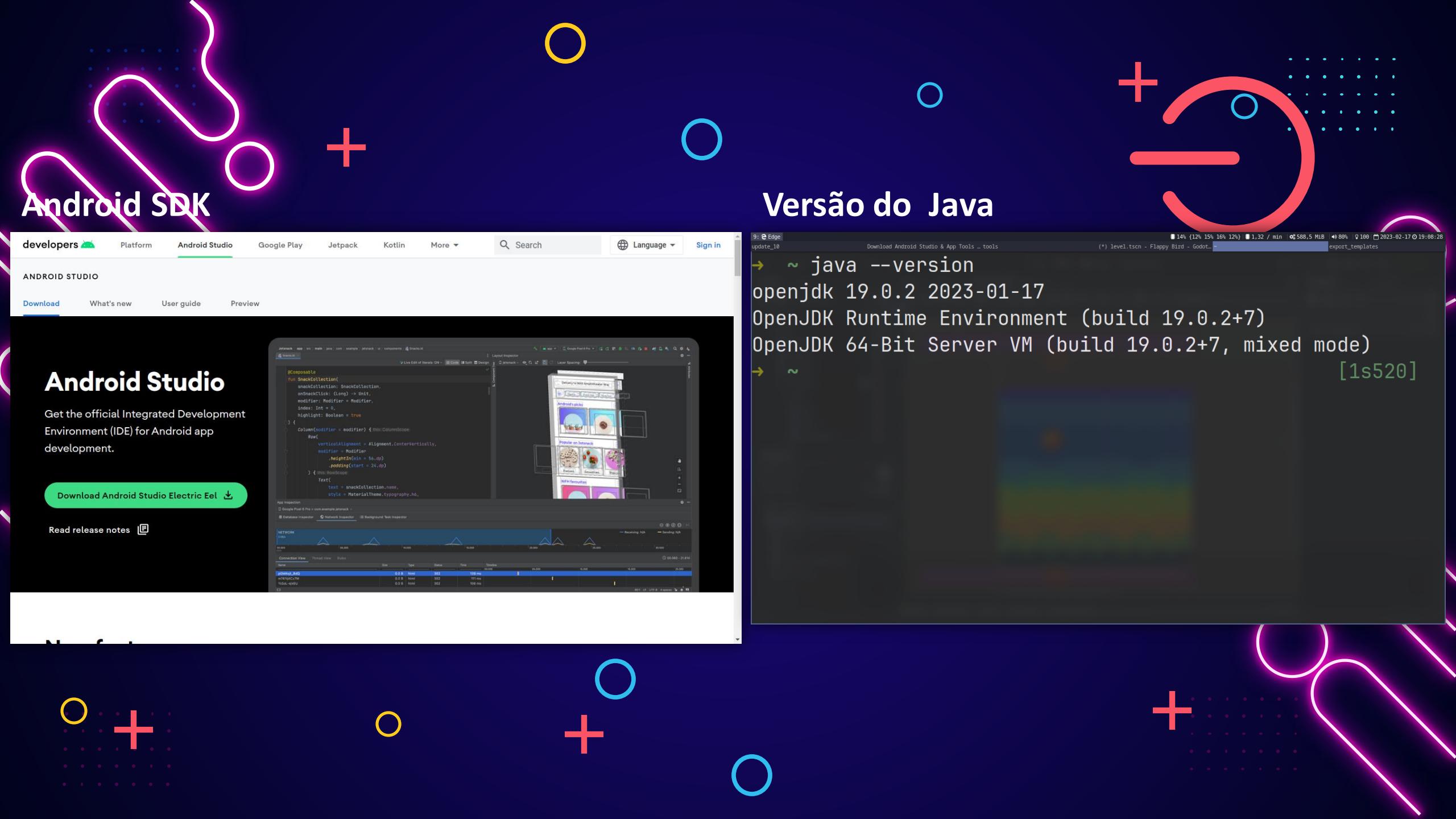


# Exportar



# Plataformas







stable

Search docs

## Exporting for Android

[Install OpenJDK 11](#)

[Download the Android SDK](#)

[Create a debug.keystore](#)

[Setting it up in Godot](#)

[Providing launcher icons](#)

[Exporting for Google Play Store](#)

[Optimizing the APK size](#)

[Troubleshooting rendering issues](#)

[Read the Docs](#)

v: stable ▾

# Exporting for Android

## See also

This page describes how to export a Godot project to Android. If you're looking to compile export template binaries from source instead, read [Compiling for Android](#).

Exporting for Android has fewer requirements than compiling Godot for Android. The following steps detail what is needed to set up the Android SDK and the engine.

## Install OpenJDK 11

Download and install [OpenJDK 11](#).

## Download the Android SDK

Download and install the Android SDK.

- You can install it using [Android Studio version 4.1 or later](#).
  - Run it once to complete the SDK setup using these [instructions](#).
  - Ensure that the [required packages](#) are installed as well.
    - Android SDK Platform-Tools version 30.0.5 or later
    - Android SDK Build-Tools version 30.0.3
    - Android SDK Platform 31
    - Android SDK Command-line Tools (latest)
    - CMake version 3.10.2-4988404

# Referências

Os links de materiais usados nesta apresentação estão disponíveis em  
[github.com/GersonFeDutra/gd\\_start](https://github.com/GersonFeDutra/gd_start)

## Agradecimentos:

Daniel Nascimento

<https://github.com/danielnasc>

Miguel Reis de Araújo

<https://github.com/reisaraujo-miguel>