

Godot Fighter!

Domine o Código, Vença o Jogo



Victor Yukio Watanabe

01

NODE2D A BASE DE TUDO NO 2D

Node2D — A base de tudo no 2D

Todo jogo 2D começa com o **Node2D**. Ele serve como um ponto de origem para posicionar e agrupar outros nodes. Pense nele como o “pai” de tudo que existe na sua cena.

Exemplo real:

```
extends Node2D

func _ready():
    position = Vector2(100, 200)
    print("Personagem posicionado em:", position)
```

Use Node2D quando quiser criar **objetos do mundo, inimigos, itens ou partes do cenário**.



GODOT
Game engine

02

SPRITE2D DANDO CARA AO JOGO

Sprite2D — Dando Cara ao Jogo

- O **Sprite2D** é o node responsável por **mostrar imagens** no jogo.

É com ele que você exibe o personagem, o cenário, ou qualquer elemento visual.

- **Exemplo real:**

```
extends Sprite2D

func _ready():
    texture = load("res://sprites/player.png")
```

- **Dica:** combine o Sprite2D com um AnimationPlayer para animar o personagem sem precisar trocar de cena.



GODOT
Game engine

03

AREA2D DETECTANDO COLISÕES E EVENTOS

Area2D — Detectando Colisões e Eventos

- O **Area2D** é um dos nodes mais úteis. Ele detecta **quando algo entra, sai ou está dentro da sua área de colisão.**

- **Exemplo real:**

```
extends Area2D

func _on_body_entered(body):
    print("Colisão detectada com:", body.name)
```

- Use o **Area2D** para criar **zonas de dano, coleta de itens, portas e gatilhos de eventos.**



GODOT
Game engine

04

**RIGIDBODY2D
FÍSICA REALISTA
SEM DOR DE CABEÇA**

RigidBody2D — Física Realista Sem Dor de Cabeça

Quer um objeto que reaja à **gravidade** e **colisões** automaticamente?
O RigidBody2D é a escolha certa.

Exemplo real:

```
extends RigidBody2D

func _integrate_forces(state):
    if Input.is_action_pressed("ui_right"):
        apply_force(Vector2(100, 0))
```

Ideal para jogos com **física**, como **plataformas**, **quebra-cabeças** ou **fliperamas**.



GODOT
Game engine

05

CHARACTERBODY2D **O NODE DO JOGADOR**

CharacterBody2D — O Node do Jogador

Desde a Godot 4, o **CharacterBody2D** substitui o antigo KinematicBody2D. Ele foi feito para personagens com **movimento controlado por código**.

Exemplo real:

```
extends CharacterBody2D

@export var speed = 200

func _physics_process(delta):
    var direction = Input.get_axis("ui_left", "ui_right")
    velocity.x = direction * speed
    move_and_slide()
```

Esse node é **perfeito para o jogador principal** ou NPCs que se movem.



GODOT
Game engine

06

ANIMATIONPLAYER DANDO VIDA ÀS CENAS

CharacterBody2D — O Node do Jogador

Com o **AnimationPlayer**, você cria animações de qualquer propriedade: posição, rotação, opacidade e até scripts.

Exemplo real:

```
extends Node2D

func _ready():
    $AnimationPlayer.play("andar")
```

Use-o para **animações de personagens, transições de tela e efeitos visuais.**



GODOT
Game engine

07

**SCRIPT GLOBAL
(SINGLETON)
MEMÓRIA
COMPARTILHADA**

Script Global (Singleton) — Memória Compartilhada

Quer guardar dados que fiquem disponíveis em todas as cenas?

Use um **autoload** (script global). Ele é como um “cérebro” do seu jogo.

Exemplo real (res://global.gd):

```
extends Node  
  
var score = 0
```

Excelente para **guardar pontuação, vida,**
ou **configurações do jogador.**



GODOT
Game engine

08

TIMER CONTROLANDO O TEMPO COM PRECISÃO

Script Global (Singleton) — Memória Compartilhada

O **Timer** dispara um sinal depois de um tempo definido.

Perfeito para **spawn de inimigos**,
cooldowns e **efeitos temporários**.

Exemplo real:

```
extends Node

▼ func _ready():
    $Timer.wait_time = 2.0
    $Timer.start()

▼ func _on_Timer_timeout():
    print("2 segundos se passaram!")
```

Combine **Timer** com **Signals** para criar lógica de jogo elegante e sem “gambiarras”.



GODOT
Game engine

09

SIGNALS COMUNICAÇÃO ENTRE NODES

Signals — Comunicação Entre Nodes

Os **signals** permitem que nodes conversem entre si sem precisar de código acoplado.

Exemplo real:

```
# Em um botão (Button.gd)
signal clicked

func _on_pressed():
    emit_signal("clicked")
```

```
# Em outro node
func _ready():
    $Button.connect("clicked", Callable(self, "_on_button_clicked"))

func _on_button_clicked():
    print("Botão clicado!")
```

Use signals para **menus, interações, sistemas de evento e UI.**



GODOT
Game engine

10

SCENETREE O CORAÇÃO DO JOGO

SceneTree — O Coração do Jogo

A **SceneTree** é a estrutura onde todas as cenas do seu jogo vivem.

Com ela, você pode mudar de fase, reiniciar o jogo ou carregar novas telas.

Exemplo real:

```
extends Control
```

```
func _on_flee_button_pressed():
```

```
>| get_tree().change_scene_to_file("res://scenes/fase2.tscn")
```

Use o SceneTree para **controle de fluxo e transições entre cenas**.



CONCLUSÃO

MONTE SEU PRÓPRIO ARCADE!

Conclusão

Com esses nodes e scripts, você já pode construir praticamente qualquer tipo de jogo 2D na Godot 4 — de um **plataformer clássico** a um **beat ‘em up estilo Final Fighter**.

A mágica está em **entender como os nodes se conectam** e em **experimentar muito**.

Dica final: Comece pequeno, adicione um node por vez e veja seu jogo ganhar vida!



AGRADECIMENTOS

Obrigado!

Esse Ebook foi gerado por IA, e diagramado por humano.

O passo a passo se encontra no meu Github

Esse conteúdo foi gerado com fins didáticos de construção, não foi realizado uma validação cuidadosa humana no conteúdo e pode conter erros gerados por uma IA.



<https://github.com/yukiow55/prompts-recipe-to-create-a-ebook/>

