

# 🌟 Guia Completo de GDScript no Godot

Este guia cobre a utilização de métodos de entrada, variáveis, estruturas de fluxo, listas, dicionários, além de configurações recomendadas para **Pixel Art** e mapeamento de **Inputs** no **Godot Engine**.

---

## 🕒 Métodos de Entrada

### ◆ `Input.get_vector()`

Este método é uma maneira rápida e eficiente de obter **direções de movimento** com as teclas de entrada configuradas no editor. Ele recebe os parâmetros das **teclas de movimento** e retorna um vetor que pode ser usado para calcular a direção do movimento.

```
var direcao: Vector2 = Input.get_vector("move_esquerda", "move_direita",
"move_cima", "move_baixo")
velocity = direcao * velocidade_personagem
```

### ◆ `Input.get_axis()`

Outro método de entrada que **captura valores de eixo** para uma direção específica (horizontal ou vertical). Muito usado para capturar valores em **controles analógicos** de gamepads.

```
var horizontal: float = Input.get_axis("move_esquerda", "move_direita")
var vertical: float = Input.get_axis("move_cima", "move_baixo")
direcao = Vector2(horizontal, vertical)
```

💡 **Dica:** Para criar uma entrada responsiva no seu jogo, use `Input.get_vector()` para movimentos e `Input.get_axis()` para entradas analógicas, como nos controles.

---

## 🌀 Referências com \$

O `$` é um atalho para referenciar nós na árvore de cena. É equivalente a `get_node("NodePath")`.


```
@onready var animacao = $AnimationPlayer
@onready var sprite = $Sprite
```

Esse comando torna o código mais limpo e acessa rapidamente nós importantes.

## Constantes

As constantes são declaradas com a palavra-chave `const` e são **valores imutáveis** que não podem ser alterados após a definição. Elas ajudam a tornar o código mais claro e eficiente.

```
const VELOCIDADE_MAXIMA: int = 100
const GRAVIDADE: float = 9.8
```

 **Uso Prático:** Defina valores como velocidade ou limites de tela usando constantes para garantir que esses valores nunca sejam alterados acidentalmente.

## Estruturas de Dados: Listas e Dicionários

### Listas

Listas são coleções ordenadas e mutáveis de itens, que você pode acessar por índices.

```
var dialogos_npc = ["Olá, aventureiro!", "Posso te ajudar?", "Boa sorte na sua jornada!"]
```

**Aplicação:** Use listas para armazenar **diálogos de NPCs** e criar interações dinâmicas em seu jogo.


```
func dialogo_npc(indice: int):
    return dialogos_npc[indice]
```

### Dicionários

Dicionários são coleções de **pares chave-valor** que permitem acessar itens por suas chaves.

```
var inventario = {
    "pocoas": 5,
```

```
"ouro": 100
}
```

 **Dica:** Use dicionários para armazenar dados de inventário ou características dos personagens.

## Estruturas de Controle de Fluxo

### **if e else**

Controla o fluxo do código executando blocos específicos se a condição for verdadeira.

```
if vida < 0:
    game_over()
else:
    atualizar_vida()
```

### **for**


O **for** permite iterar sobre listas, dicionários e intervalos.

```
for item in inventario:
    print(item + ": " + str(inventario[item]))
```

### **match**

O **match** é uma estrutura similar ao **switch** em outras linguagens e é usada para comparar variáveis com valores específicos.

```
match direcao:
    Vector2.UP:
        print("Movendo para cima")
    Vector2.DOWN:
        print("Movendo para baixo")
    _:
        print("Parado")
```

 **Aplicação Prática:** **match** é útil para determinar comportamentos específicos com base na direção ou status do personagem.

## ◆ while

Um laço `while` continua executando enquanto a condição for verdadeira.

```
var contador = 0
while contador < 10:
    print(contador)
    contador += 1
```

📌 **Aplicação Prática:** Use `while` para situações de espera ou loops dependentes de condições, como aguardar o jogador se aproximar de um ponto específico.

## 🎮 Mapeamento de Inputs

O Godot permite que você mapeie inputs facilmente através do menu **Projeto > Configurações do Projeto > Entrada**. Esse mapeamento cria **atalhos para inputs** personalizados que você pode utilizar diretamente no código.

1. Vá para **Projeto > Configurações do Projeto > Entrada**.
2. Crie uma nova ação ( ex: `movimento_esquerda` ).
3. Atribua uma tecla ou botão de controle para cada ação.
4. No código, chame as ações definidas.

```
if Input.is_action_pressed("movimento_esquerda"):
    mover_para_esquerda()
```

🔧 **Dica:** Ao configurar inputs no editor, você pode facilmente mudar teclas sem precisar alterar o código.

## 🎨 Configurações para Pixel Art

### ◆ Configurações de Textura

Para jogos em Pixel Art, defina o filtro da textura como **Nearest**. Isso preserva os pixels e evita o efeito de borrado.

- Abra o **Configurações do Projeto > Renderização > Textura > Filtro**.

- Marque a opção **Nearest**.

## ◆ Configurações de Tela e Resolução

Se o jogo for renderizado em resolução baixa e escalado para HD, siga os passos abaixo:

1. Defina a resolução para **320 x 192** no editor.
2. Para escalar para resolução HD, ajuste para o valor **4x**, ficando **1280 x 768**.
3. Marque a opção `canvas_item`.

Essas configurações garantem que o jogo mantenha o visual de Pixel Art.

---

## 📖 Exemplos de Uso em Jogos

### 1. Diálogos de NPCs com Listas

```
var dialogos_npc = ["Bem-vindo!", "O que deseja saber?", "Boa sorte!"]

func mostrar_dialogo(indice: int) -> String:
    if indice < dialogos_npc.size():
        return dialogos_npc[indice]
    return ""
```

### 2. Inventário com Dicionário

```
var inventario = {
    "poção": 3,
    "ouro": 200
}

func usar_item(item: String):
    if inventario.has(item) && inventario[item] > 0:
        inventario[item] -= 1
        print("Usou " + item)
    else:
        print("Item não disponível")
```

### 3. Movimento Direcional com `Input.get_vector()`

```
func _physics_process(delta):  
    var direcao = Input.get_vector("move_esquerda", "move_direita",  
    "move_cima", "move_baixo")  
    velocidade = direcao * velocidade_personagem  
    move_and_slide(velocidade)
```

---

## Conclusão e Recomendações

Este guia apresenta os fundamentos do GDScript, incluindo **variáveis, estruturas de fluxo, mapeamento de inputs e configurações para jogos em Pixel Art**. Com essas práticas, você poderá desenvolver jogos bem estruturados, organizados e adaptados às configurações do **Godot Engine**.

---