🕹️ Documentação do Script do Player em Godot

Este documento descreve o funcionamento do script do player em um jogo 2D, desenvolvido em **Godot Engine** com **GDScript**. Abaixo, você encontrará uma explicação completa do script, com seções destacadas e ícones para melhor navegação.

⚙️ Estrutura Geral

🌲 Classe e Herança

```
extends CharacterBody2D
class_name player
```

A classe `player` herda de `CharacterBody2D`, o que facilita o controle de **movimentos físicos** e **colisões** em um ambiente 2D.

🔑 Variáveis Principais

As variáveis definem **atributos essenciais** do player, como **velocidade de movimento**, **controle de ataque** e **animações**.

```
@export var _speed_pixel: float = 120
@export var _temporizador_de_acoes: Timer
@onready var _animador_do_personagem: AnimationPlayer = $Animation

var direcao: Vector2 = Vector2.ZERO
var _sufixo_da_animacao: String = "_baixo"
var _pode_atacar: bool = true
```

Explicação:

- `_speed_pixel` 🏃: Define a **velocidade** do player em pixels por segundo.
- `_temporizador_de_acoes` ⏳: Controla o **intervalo entre ataques**.
- `_animador_do_personagem` 🎬: Controla a **reprodução das animações** do player.
- `direcao` ➡️: Vetor que armazena a **direção de movimento**.

- `_sufixo_da_animacao` 💪: Guarda o **sufixo da direção** do movimento, usado nas animações.
- `_pode_atacar` ⚔️: Booleano que indica se o player está **habilitado para atacar**.

🚀 Funções Principais

Cada função tem um propósito claro e desempenha uma função específica no comportamento do personagem.

1. `func _ready() -> void` - 🚦 Configurações Iniciais

Chamado assim que o nó é carregado. No momento, não há ações específicas.

```
func _ready() -> void:
    pass
```

---

2. `func _physics_process(_delta: float) -> void` - 🧩 Processamento de Física

Chamado em cada **frame de física** para executar as funções principais de movimento, animação e ataque.

```
func _physics_process(_delta: float) -> void:
    _movimento_personagem()
    _sufixo_da_animacao = _sufixo_do_personagem()
    _atacar()
    _animar()
```

3. `func _movimento_personagem() -> void` - 🕹️ Controle de Movimento

Define a **direção de movimento** com base nas teclas pressionadas.

```
func _movimento_personagem() -> void:
    direcao = Input.get_vector("move_esquerda", "move_direita", "move_cima",
"move_baixo")
```

```
    velocity = direcao * _speed_pixel
    move_and_slide()
```

Explicação:

- `Input.get_vector`: Obtém as teclas pressionadas para determinar a direção do movimento.
- `velocity`: Multiplica a direção pela velocidade para determinar a **velocidade do player**.
- `move_and_slide()`: Move o player e lida com colisões.

---

4. `func _sufixo_do_personagem() -> String` - 🔄 Ajuste de Sufixo para Animações

Define o **sufixo da animação** com base na direção do movimento (esquerda, direita, cima ou baixo).

```
func _sufixo_do_personagem() -> String:
    var acao_horizontal = Input.get_axis("move_esquerda", "move_direita")
    if acao_horizontal == -1:
        return "_esquerda"
    if acao_horizontal == 1:
        return "_direita"

    var acao_vertical = Input.get_axis("move_cima", "move_baixo")
    if acao_vertical == -1:
        return "_cima"
    if acao_vertical == 1:
        return "_baixo"

    return _sufixo_da_animacao
```

Explicação:

- Retorna um **sufixo de animação** baseado nos valores de `Input.get_axis`.
- Esse sufixo é utilizado para chamar a animação correta, como `"move_esquerda"` ou `"parado_direita"`.

---

5. `func _atacar() -> void` - ⚔️ Função de Ataque

Controla a ação de **ataque do player** e usa `_temporizador_de_acoes` para evitar ataques consecutivos.

```
func _atacar() -> void:
    if _pode_atacar == false:
        return
    if Input.is_action_pressed("ataque_normal") and _pode_atacar:
        _animador_do_personagem.play("atacar" + _sufixo_da_animacao)
        _temporizador_de_acoes.start(0.4)
        set_physics_process(false)
        _pode_atacar = false
```

Explicação:

- **Controle de Ataque**: Verifica se `_pode_atacar` é `true`. Se não, bloqueia a ação.
- `_temporizador_de_acoes.start(0.4)` : Inicia um temporizador para retomar o ataque após 0.4 segundos.
- `set_physics_process(false)` : Pausa o movimento para focar na animação de ataque.

---

6. `func _animar() -> void` - 🎬 Controle de Animação

Escolhe a animação de **movimento** ou **parado** com base na direção e velocidade do player.

```
func _animar() -> void:
    if _pode_atacar == false:
        return
    if velocity:
        _animador_do_personagem.play("move" + _sufixo_da_animacao)
        return
    _animador_do_personagem.play("parado" + _sufixo_da_animacao)
```

Explicação:

- **Animação de Movimento**: Exibe a animação "move" caso o player esteja se movendo (`velocity` ≠ 0).
- **Animação de Parado**: Caso contrário, exibe a animação "parado".

---

7. `func _on_temporizador_de_acoes_timeout() -> void` - ⏳ Função de Timeout

Essa função é chamada ao término do temporizador, permitindo que o player volte a **se mover e atacar**.

```
func _on_temporizador_de_acoes_timeout() -> void:
    set_physics_process(true)
    _pode_atacar = true
```

Explicação:

- `set_physics_process(true)` : Retoma o movimento do player.
- **Liberação de Ataque**: Permite que o player ataque novamente.

📜 Resumo Geral

1. **Movimento**: Controlado pela direção e velocidade.
2. **Ataque**: Controlado por um temporizador para evitar ataques consecutivos.
3. **Animações**: Variam com base na direção e ação atual.
4. **Temporizador**: Garante intervalos entre ataques e retoma o movimento.
