```
extends CharacterBody2D
class_name Player

# Variáveis exportadas para configuração no editor
@export var _speed_pixel: float = 120
@export var _temporizador_de_acoes: Timer
@export var _area_de_ataque: Area2D
@export var _texto_armar_atual: Label

# Variáveis de controle
var _arma_atual: String = "_machado"
var direcao: Vector2 = Vector2.ZERO
var _sufixo_da_animacao: String = "_baixo" # Define a animação atual
var _pode_atacar: bool = true

# Variáveis de nó atribuídas no script
@onready var _animador_do_personagem: AnimationPlayer = $Animation

# Função executada ao inicializar a cena
func _ready() -> void:
	pass

# Função chamada a cada frame para gerenciar animações, ataques e movimentos
func _process(_delta: float) -> void:
	_animar()
	_atacar()
	_sufixo_da_animacao = _sufixo_do_personagem()
	_definir_arma_atual()
	_movimento_personagem()

# Função que controla o movimento do personagem
func _movimento_personagem() -> void:
	direcao = Input.get_vector("move_esquerda", "move_direita", "move_cima",
"move_baixo")
	velocity = direcao * _speed_pixel
	move_and_slide()

# Função que retorna o sufixo da animação com base na direção do movimento
func _sufixo_do_personagem() -> String:
	var acao_horizontal = Input.get_axis("move_esquerda", "move_direita")
	if acao_horizontal == -1:
		_area_de_ataque.position = Vector2(-17, 0)
		return "_esquerda"
```

```
        if acao_horizontal == 1:
                _area_de_ataque.position = Vector2(16, 0)
                return "_direita"

        var acao_vertical = Input.get_axis("move_cima", "move_baixo")
        if acao_vertical == -1:
                _area_de_ataque.position = Vector2(0, -12)
                return "_cima"
        if acao_vertical == 1:
                _area_de_ataque.position = Vector2(0, 17)
                return "_baixo"

        return _sufixo_da_animacao

# Define a arma atual e retorna o tipo de ação com base na entrada do jogador
func _definir_arma_atual() -> String:
        var tipo_acao: String = "ataque_normal"

        if Input.is_action_pressed("machado_ataque"):
                _arma_atual = "_machado"
                tipo_acao = "machado_ataque"
        if Input.is_action_pressed("ataque_normal"):
                _arma_atual = "_espada"
                tipo_acao = "ataque_normal"
        if Input.is_action_pressed("picareta_ataque"):
                _arma_atual = "_picareta"
                tipo_acao = "picareta_ataque"
        if Input.is_action_pressed("enxada_ataque"):
                _arma_atual = "_enxada"
                tipo_acao = "enxada_ataque"
        if Input.is_action_pressed("regador_ataque"):
                _arma_atual = "_regador"
                tipo_acao = "regador_ataque"

        _texto_armar_atual.text = _arma_atual
        return tipo_acao

# Função responsável por executar o ataque
func _atacar() -> void:
        var acao: String = _definir_arma_atual()

        if Input.is_action_pressed(acao) and _pode_atacar:
                _animador_do_personagem.play("atacar" + _arma_atual +
_sufixo_da_animacao)
                _temporizador_de_acoes.start(0.4)
                set_process(false)
```

```
                _pode_atacar = false

# Gerencia as animações do personagem
func _animar() -> void:
        if not _pode_atacar:
                return
        if velocity:
                _animador_do_personagem.play("move" + _sufixo_da_animacao)
        else:
                _animador_do_personagem.play("parado" + _sufixo_da_animacao)

# Reativa o processamento e permite o ataque após o timeout
func _on_temporizador_de_acoes_timeout() -> void:
        set_process(true)
        _pode_atacar = true
```

Explicação

1. **Configuração de Variáveis e Nó**:
   - Variáveis exportadas (`@export`) podem ser ajustadas diretamente no editor.
   - Variáveis `@onready` são atribuídas quando a cena é carregada.
2. **Execução do Código**:
   - `_process()` é chamado a cada frame, controlando o fluxo do jogo.
   - Chama funções para movimento, animação, ataques e atualização de estados.
3. **Movimento**:
   - `_movimento_personagem()` calcula a direção com `Input.get_vector` e aplica o movimento usando `move_and_slide()` .
4. **Animação**:
   - `_sufixo_do_personagem()` ajusta o sufixo da animação e a posição do ataque.
   - `_animar()` toca as animações corretas dependendo do estado (movendo, parado ou atacando).
5. **Ataque**:
   - `_definir_arma_atual()` atualiza a arma ativa com base na entrada.
   - `_atacar()` executa a animação do ataque e impede múltiplos ataques simultâneos até que o temporizador termine.
6. **Temporizador**:
   - `_on_temporizador_de_acoes_timeout()` reativa o processamento e permite novos ataques após o tempo definido.