config

```python
"""Configurações"""
LARGURA_JANELA = 960
ALTURA_JANELA = 540
QUADROS_POR_SEGUNDO = 60
COMPRIMENTO_NIVEL = 6000
```

camera

```python
import pygame
"""Camera"""
class Camera:
    def __init__(self, largura: int, altura: int, largura_janela:int):
        self.deslocamento_x = 0
        self.largura = largura
        self.altura = altura
        self.largura_janela = largura_janela
    def seguir(self,retangulo_alvo: pygame.Rect):
        "Centralizar a Camera com o Personagem"
        centro_alvo_x = retangulo_alvo.centerx
        self.deslocamento_x = max(0, self.largura -
self.largura_janela)
        desloc_maximo = max(0, self.largura - self.largura_janela)
        if self.deslocamento_x > desloc_maximo:
            self.deslocamento_x = desloc_maximo
    def aplicar(self, retangulo: pygame.Rect) -> pygame.Rect:
        return pygame.Rect(retangulo.x - self.deslocamento_x,
retangulo.y,
                           retangulo.width, retangulo.height)
```

imagem

```
import pygame
"""Carrega as imagens"""
def carregar_imagem(caminho: str, altura_escala: int | None=None) ->
pygame.Surface:
    """Calcula Proporção"""
    imagem = pygame.image.load(caminho).convert_alpha()
    if altura_escala is None:
        return imagem
    largura_atual, altura_atual = imagem.get_size()
    proporção = altura_escala / altura_atual
```

```
    novo_tamanho = (int(largura_atual * proporção), int(altura_atual *
proporção))
    return pygame.transform.smoothcale(imagem, novo_tamanho)
```

cenario

```
import os
import pygame
from util_imagem import carregar_imagem
"""Constroi o cenario"""
def construir_cenario(dir_arquivos: str, altura_janela: int,
comprimento_nivel: int) -> tuple[list[pygame.Rect],
list[tuple[pygame.Surface, pygame.Rect]]]
solidos: list[pygame.Rect] = []
decoraçoes = list[tuple[pygame.Surface, pygame.Rect]]= []
chao = pygame.Rect(0, altura_altura - 64 conprimento_nivel, 64)
solidos.append(chao)
caminhos_tijolo = os.path.join(dir_arquivos, "tijolo,png")
img_tijolo = carregar_imagem(caminhos_tijolo, altura_escala = 48)
plataforma_tijolo = [
    (300, altura_janela = 200, 5)
    (900, altura_janela = 260, 4)
    (1400, altura_janela = 180, 6)
]
for inicio_x
```

inimigo

```
import os
import pygame
from util_imagem import carregar_imagem
""" Inimigos com patrulha horizontal """
class Inimigo(pygame.sprite.Sprite):
    def __init__(self, posicao_inicial: tuple[int, int], dir_arquivos:
str, limites_patrulha: tuple[int, int]):
        super().__init__()
        caminho_planta = os.path.join(dir_arquivos, "Piranha
Plant.png")
        self.imagem = carregar_imagem(caminho_planta, altura_escala=64)
        self.retangulo = self.imagem.get_rect(topleft=posicao_inicial)
        self.direcao = 1
        self.velocidade = 2
```

```
        self.limites_patrulha = limites_patrulha

    def patrulhar(self):
        esquerdo, direito = self.limites_patrulha
        self.retangulo.x += self.direcao * self.velocidade
        if self.retangulo.left <= esquerdo:
            self.retangulo.left = esquerdo
            self.direcao = 1
        elif self.retangulo.right >= direito:
            self.retangulo.right = direito
            self.direcao = -1

class Inimigo2(Inimigo):
    def __init__(self, posicao_inicial: tuple[int, int], dir_arquivos:
str, limites_patrulha: tuple[int, int]):
        super().__init__(posicao_inicial, dir_arquivos,
limites_patrulha)
        caminho_img = os.path.join(dir_arquivos, "inimigo2.jpeg")
        self.imagem = carregar_imagem(caminho_img, altura_escala=80)
        base = self.retangulo.bottomleft
        self.retangulo = self.imagem.get_rect()
        self.retangulo.bottomleft = base
        self.velocidade = 3
```

placar

```python
import os
""" Leitura, escrita e formatação de tempos do placar """
def ler_tempos(caminho: str) -> list[tuple[int, str]]:
    if not os.path.exists(caminho):
        return []
    resultados: list[tuple[int, str]] = []
    try:
        with open(caminho, "r", encoding="utf-8") as f:
            for linha in f:
                partes = linha.strip().split(";")
                if len(partes) == 2:
                    try:
                        ms = int(partes[0])
                        nome = partes[1]
                        resultados.append((ms, nome))
                    except ValueError:
                        pass
    except OSError:
        return []
    return resultados

def salvar_tempo(caminho: str, ms: int, nome: str) -> None:
    try:
        with open(caminho, "a", encoding="utf-8") as f:
            f.write(f"{ms};{nome}\n")
    except OSError:
        pass

def top5(registros: list[tuple[int, str]]) -> list[tuple[int, str]]:
    return sorted(registros, key=lambda x: x[0])[:5]

def formatar_ms(ms: int) -> str:
    s = ms // 1000
    ms_rem = ms % 1000
    return f"{s}.{ms_rem:03d}s"
```

ui

```python
import pygame
from config import LARGURA_JANELA, ALTURA_JANELA
from placar import formatar_ms, top5


""" Desenho de HUD, telas e botões """


def desenhar_grade(superficie: pygame.Surface, deslocamento_x: int,
comprimento_nivel: int, altura_janela: int):
    cor = (220, 220, 220)
    for x in range(0, comprimento_nivel + 64, 64):
        tela_x = x - deslocamento_x
        pygame.draw.line(superficie, cor, (tela_x, 0), (tela_x,
altura_janela))


def botao_reiniciar_topo(tela: pygame.Surface) -> pygame.Rect:
    btn = pygame.Rect(LARGURA_JANELA - 180, 10, 170, 36)
    pygame.draw.rect(tela, (0, 0, 0), btn, 2)
    texto_btn = pygame.font.SysFont(None, 24).render("REINICIAR (R)",
True, (0, 0, 0))
    tela.blit(texto_btn, texto_btn.get_rect(center=btn.center))
```

```
    return btn


def tela_inicio(tela: pygame.Surface, registros: list[tuple[int,
str]]):
    fonte = pygame.font.SysFont(None, 24)
    titulo = pygame.font.SysFont(None, 64).render("MARIO - SIMPLES",
True, (0, 0, 0))
    ret_titulo = titulo.get_rect(center=(LARGURA_JANELA // 2, 100))
    tela.blit(titulo, ret_titulo)
    lista = top5(registros)
    y = 170
    tela.blit(fonte.render("TOP 5 TEMPOS (s.ms):", True, (0, 0, 0)),
(LARGURA_JANELA // 2 - 140, y))
    y += 28
    if not lista:
        tela.blit(fonte.render("Sem tempos salvos.", True, (0, 0, 0)),
(LARGURA_JANELA // 2 - 100, y))
    else:
        for i, (ms, nome) in enumerate(lista, start=1):
            linha = f"{i}. {formatar_ms(ms)} - {nome}"
            tela.blit(fonte.render(linha, True, (0, 0, 0)),
(LARGURA_JANELA // 2 - 160, y))
            y += 24
    instru = fonte.render("Pressione qualquer tecla para começar",
True, (0, 0, 0))
    ret_instr = instru.get_rect(center=(LARGURA_JANELA // 2,
ALTURA_JANELA - 60))
   ela: pygame.Surface, tempo_ms: int):
    fonte = pygame.font.SysFont(None, 24)
    texto = fonte.render(f"Tempo: {formatar_ms(tempo_ms)}", True, (0,
0, 0))
    tela.blit(texto, (16, 10))


def tela_final(tela: pygame.Surface, tempo_final_ms: int, nome_atual:
str) -> pygame.Rect:
    sobreposicao = pygame.Surface((LARGURA_JANELA, ALTURA_JANELA),
pygame.SRCALPHA)
```

```
    mensagem = fonte_grande.render(f"FIM DO JOGO -
{formatar_ms(tempo_final_ms)}", True, (255, 255, 255))
    ret_mensagem = mensagem.get_rect(center=(LARGURA_JANELA // 2,
ALTURA_JANELA // 2 - 40))
    tela.blit(mensagem, ret_mensagem)
    fonte = pygame.font.SysFont(None, 24)
    instru_nome = fonte.render("Digite seu nome e pressione Enter:",
True, (255, 255, 255))
    tela.blit(instru_nome, (LARGURA_JANELA // 2 - 160, ALTURA_JANELA //
2 + 10))
    caixa_ret = pygame.Rect(LARGURA_JANELA // 2 - 160, ALTURA_JANELA //
2 + 36, 320, 30)
    pygame.draw.rect(tela, (255, 255, 255), caixa_ret, 2)
    texto_nome = fonte.render(nome_atual, True, (255, 255, 255))
    tela.blit(texto_nome, (caixa_ret.x + 8, caixa_ret.y + 5))
    btn_larg, btn_alt = 240, 56
    btn_ret = pygame.Rect((LARGURA_JANELA - btn_larg) // 2,
ALTURA_JANELA // 2 + 80, btn_larg, btn_alt)
    pygame.draw.rect(tela, (255, 255, 255), btn_ret, 2)
    btn_txt = pygame.font.SysFont(None, 36).render("REINICIAR (R)",
True, (255, 255, 255))
    tela.blit(btn_txt, btn_txt.get_rect(center=btn_ret.center))
    return btn_ret tela.blit(instru, ret_instr)


def hud_tempo(tela: pygame.Surface, tempo_ms: int):
    fonte = pygame.font.SysFont(None, 24)
    texto = fonte.render(f"Tempo: {formatar_ms(tempo_ms)}", True, (0,
0, 0))
    tela.blit(texto, (16, 10))


def tela_final(tela: pygame.Surface, tempo_final_ms: int, nome_atual:
str) -> pygame.Rect:
    sobreposicao = pygame.Surface((LARGURA_JANELA, ALTURA_JANELA),
pygame.SRCALPHA)
    sobreposicao.fill((0, 0, 0, 120))
    tela.blit(sobreposicao, (0, 0))
    fonte_grande = pygame.font.SysFont(None, 72)
    mensagem = fonte_grande.render(f"FIM DO JOGO -
{formatar_ms(tempo_final_ms)}", True, (255, 255, 255))
    ret_mensagem = mensagem.get_rect(center=(LARGURA_JANELA // 2,
ALTURA_JANELA // 2 - 40))
```

```
    tela.blit(mensagem, ret_mensagem)
    fonte = pygame.font.SysFont(None, 24)
    instru_nome = fonte.render("Digite seu nome e pressione Enter:",
True, (255, 255, 255))
    tela.blit(instru_nome, (LARGURA_JANELA // 2 - 160, ALTURA_JANELA //
2 + 10))
    caixa_ret = pygame.Rect(LARGURA_JANELA // 2 - 160, ALTURA_JANELA //
2 + 36, 320, 30)
    pygame.draw.rect(tela, (255, 255, 255), caixa_ret, 2)
    texto_nome = fonte.render(nome_atual, True, (255, 255, 255))
    tela.blit(texto_nome, (caixa_ret.x + 8, caixa_ret.y + 5))
    btn_larg, btn_alt = 240, 56
    btn_ret = pygame.Rect((LARGURA_JANELA - btn_larg) // 2,
ALTURA_JANELA // 2 + 80, btn_larg, btn_alt)
    pygame.draw.rect(tela, (255, 255, 255), btn_ret, 2)
    btn_txt = pygame.font.SysFont(None, 36).render("REINICIAR (R)",
True, (255, 255, 255))
    tela.blit(btn_txt, btn_txt.get_rect(center=btn_ret.center))
    return btn_ret
```

jogador

```
import os
import pygame
from util_imagem import carregar_imagem


""" Entidade do jogador e sua física simples """


class Jogador(pygame.sprite.Sprite):
    def __init__(self, posicao_inicial: tuple[int, int], dir_arquivos:
str):
        super().__init__()
        caminho_mario = os.path.join(dir_arquivos, "mario.gif")
        self.imagem = carregar_imagem(caminho_mario, altura_escala=64)
        self.retangulo = self.imagem.get_rect(topleft=posicao_inicial)
        self.velocidade_x = 0.0
        self.velocidade_y = 0.0
        self.no_chao = False

        self.aceleracao_chao = 0.8
        self.aceleracao_ar = 0.5
```

```
        self.atrito = 0.85
        self.gravidade = 0.9
        self.forca_pulo = -17
        self.velocidade_maxima_x = 8
        self.velocidade_maxima_queda = 22

    def ler_entrada(self):
        teclas = pygame.key.get_pressed()
        desejado = 0
        if teclas[pygame.K_LEFT] or teclas[pygame.K_a]:
            desejado -= 1
        if teclas[pygame.K_RIGHT] or teclas[pygame.K_d]:
            desejado += 1
        aceleracao = self.aceleracao_chao if self.no_chao else
self.aceleracao_ar
        self.velocidade_x += desejado * aceleracao
        if desejado == 0 and self.no_chao:
            self.velocidade_x *= self.atrito
        if self.velocidade_x > self.velocidade_maxima_x:
            self.velocidade_x = self.velocidade_maxima_x
        if self.velocidade_x < -self.velocidade_maxima_x:
            self.velocidade_x = -self.velocidade_maxima_x

    def tentar_pular(self):
        teclas = pygame.key.get_pressed()
        if (teclas[pygame.K_SPACE] or teclas[pygame.K_UP] or
teclas[pygame.K_w]) and self.no_chao:
            self.velocidade_y = self.forca_pulo
            self.no_chao = False

    def aplicar_gravidade(self):
        self.velocidade_y += self.gravidade
        if self.velocidade_y > self.velocidade_maxima_queda:
            self.velocidade_y = self.velocidade_maxima_queda

    def mover_e_colidir(self, solidos: list[pygame.Rect]):
        self.retangulo.x += int(self.velocidade_x)
        for bloco in solidos:
            if self.retangulo.colliderect(bloco):
                # indo para a direita
                if self.velocidade_x > 0:
                    self.retangulo.right = bloco.left
```

```
                elif self.velocidade_x < 0:
                    self.retangulo.left = bloco.right
                self.velocidade_x = 0
        self.retangulo.y += int(self.velocidade_y)
        self.no_chao = False
        for bloco in solidos:
            if self.retangulo.colliderect(bloco):
                # caindo
                if self.velocidade_y > 0:
                    self.retangulo.bottom = bloco.top
                    self.velocidade_y = 0
                    self.no_chao = True
                # subindo
                elif self.velocidade_y < 0:
                    self.retangulo.top = bloco.bottom
                    self.velocidade_y = 0

    def atualizar(self, solidos: list[pygame.Rect]):
        self.ler_entrada()
        self.tentar_pular()
        self.aplicar_gravidade()
        self.mover_e_colidir(solidos)
```

main

```
import os
import sys
import pygame
from config import LARGURA_JANELA, ALTURA_JANELA, QUADROS_POR_SEGUNDO,
COMPRIMENTO_NIVEL
from util_imagem import carregar_imagem
from camera import Camera
from jogador import Jogador
from inimigo import Inimigo, Inimigo2
from cenario import construir_cenario
from estado import EstadoJogo
from ui import desenhar_grade, botao_reiniciar_topo, tela_inicio, hud_tempo, tela_final


def criar_jogo(dir_arquivos: str):
    """Cria todas as entidades do jogo"""
    solidos, decoracoes = construir_cenario(dir_arquivos, ALTURA_JANELA,
COMPRIMENTO_NIVEL)
    jogador = Jogador(posicao_inicial=(100, ALTURA_JANELA - 200),
dir_arquivos=dir_arquivos)
    inimigos = [
        Inimigo((900, ALTURA_JANELA - 160), dir_arquivos, (780, 980)),
        Inimigo2((1300, ALTURA_JANELA - 160), dir_arquivos, (1200, 1500)),
```

```
        Inimigo((1850, ALTURA_JANELA - 160), dir_arquivos, (1750, 2000)),
        Inimigo2((2400, ALTURA_JANELA - 160), dir_arquivos, (2250, 2550)),
        Inimigo((3100, ALTURA_JANELA - 160), dir_arquivos, (2950, 3300)),
        Inimigo2((3750, ALTURA_JANELA - 160), dir_arquivos, (3600, 3900)),
        Inimigo((4450, ALTURA_JANELA - 160), dir_arquivos, (4300, 4550)),
        Inimigo2((5200, ALTURA_JANELA - 160), dir_arquivos, (5050, 5350)),
        Inimigo2((5600, ALTURA_JANELA - 160), dir_arquivos, (5500, 5800)),
        Inimigo((5900, ALTURA_JANELA - 160), dir_arquivos, (5850, 6000)),
    ]
    camera = Camera(largura=COMPRIMENTO_NIVEL, altura=ALTURA_JANELA,
largura_janela=LARGURA_JANELA)
    return solidos, decoracoes, jogador, inimigos, camera


def processar_colisoes(jogador: Jogador, inimigos: list, estado: EstadoJogo):
    """Processa colisões entre jogador e inimigos"""
    if estado.estado == "jogando":
        for inimigo in list(inimigos):
            if jogador.retangulo.colliderect(inimigo.retangulo):
                # Se pisar em cima, remove inimigo; senão, reinicia jogador
                if jogador.velocidade_y > 0 and jogador.retangulo.bottom <= inimigo.retangulo.top
+ 10:
                    inimigo.retangulo.topleft = (COMPRIMENTO_NIVEL + 1000, -1000)
                    try:
                        inimigos.remove(inimigo)
                    except ValueError:
                        pass
                    jogador.velocidade_y = -12
                else:
                    jogador.retangulo.topleft = (100, ALTURA_JANELA - 200)
                    jogador.velocidade_x = 0
                    jogador.velocidade_y = 0


def renderizar_cenario(tela: pygame.Surface, camera: Camera, solidos: list, decoracoes: list,
img_nuvens, nuvens: list):
    """Renderiza o cenário (fundo, nuvens, decorações, chão)"""
    cor_ceu = (135, 206, 235)
    tela.fill(cor_ceu)

    # Nuvens com paralaxe
    if img_nuvens:
        for x, y in nuvens:
            tela.blit(img_nuvens, (x - int(camera.deslocamento_x * 0.5), y))

    # Grade de referência
    desenhar_grade(tela, camera.deslocamento_x, COMPRIMENTO_NIVEL,
ALTURA_JANELA)
```

```
    # Decorações (tijolos, canos)
    for img, ret in decoracoes:
        tela.blit(img, camera.aplicar(ret))

    # Chão (barra verde)
    cor_chao = (60, 179, 113)
    for solido in solidos:
        if solido.height >= 64 and solido.width > 500:
            pygame.draw.rect(tela, cor_chao, camera.aplicar(solido))


def executar_jogo():
    pygame.init()
    tela = pygame.display.set_mode((LARGURA_JANELA, ALTURA_JANELA))
    pygame.display.set_caption("Mario - Simples")
    relogio = pygame.time.Clock()

    # Configuração de arquivos
    dir_base = os.path.dirname(os.path.abspath(__file__))
    dir_arquivos = os.path.join(dir_base, "imagens")
    caminho_tempos = os.path.join(dir_base, "tempos.txt")
    caminho_nuvens = os.path.join(dir_arquivos, "nuvens.png")

    # Estado do jogo
    estado = EstadoJogo(caminho_tempos)
    solidos, decoracoes, jogador, inimigos, camera = criar_jogo(dir_arquivos)
    img_nuvens = carregar_imagem(caminho_nuvens, altura_escala=120) if
os.path.exists(caminho_nuvens) else None
    nuvens = [(x, y) for x in range(0, COMPRIMENTO_NIVEL, 480) for y in (40, 90, 140)] if
img_nuvens else []

    executando = True
    while executando:
        # Processamento de eventos
        for evento in pygame.event.get():
            if evento.type == pygame.QUIT:
                executando = False

            # Reiniciar com tecla R
```