Kelvin Schaun Brose e Maria Eduarda Casali Ricardo

# ZINF

## Infos.h

Armazena algumas estruturas referentes a funcionalidade do jogo:

- enum GameScreen: define as telas possíveis do jogo.
- enum GameMode: define a dificuldade da gameplay.
- struct Infos: define as variáveis do jogo.
    currentState – indica em qual tela o jogo está naquele momento

    currentMode – indica em qual nível de dificuldade o jogo está

    gameArea – define a área ocupada pelo jogo na tela

    closeGame – indica se o jogo deve ser fechado

    requestcloseGame – indica se o jogador solicitou o fechamento do jogo

    currentLevel – indica o nível atual do jogo

    score – indica a pontuação atual do jogador

    num_inimigos – indica o número de inimigos lidos em cada nível

    num_inimigos_mortos – indica o número de inimigos mortos até o momento

## Character.h

Armazena a estrutura referente às características dos personagens:

- struct Character: define as variáveis de cada personagem.
    rec – indica o retângulo que representa o personagem no display do jogo
    width – indica a largura do personagem
    height – indica a altura do personagem
    posXinicial – indica a posição inicial do personagem no eixo x
    posYinicial - indica a posição inicial do personagem no eixo y
    vidas – indica quantas vidas o personagem tem
    attackTimer – indica o tempo de ataque do personagem
    rotacao – indica a rotação do personagem
    velocidade – indica  a velocidade de locomoção do personagem
    collisionUp, collisionDown, collisionRight, collisionLeft - indica se o personagem colidiu com algo naquela direção
    attackUp, attackDown, attackRight, attackLeft – indica se o personagem atacou naquela direção
    walking – indica se o personagem está caminhando

direction – indica o sprite de cada direção daquele personagem
damage – indica o som de dano daquele personagem
sword – indica o som da espada daquele personagem

**Map.h**

Armazena funções referente ao mapa do jogo:

- le_nivel: faz a leitura do arquivo do nível e preenche a matriz do mapa do jogo.
- display_jogo: desenha na tela todas as informações do mapa do jogo e movimentações dos elementos.
- checa_colisao_mapa: verifica se os personagens se chocam com os limites do mapa ou com os obstáculos.

**Menu.c**

Armazena as funções envolvidas no display do menu:

- SetupTriangle – define as características do triangulo de seleção do menu do jogo.
- DrawSelector – desenha o ícone de seleção do menu.
- DrawTitle – desenha o título do jogo no menu.
- DrawMenu – desenha as opções presentes no menu.
- ProgSelector – indica a posição atual do índice de seleção da opção do menu.

**Menu.h**

Armazena as constantes utilizadas no menu e a estrutura do triângulo de seleção:

- struct Triangle: define as variáveis do triângulo de seleção.

  origem – vetor que indica a posição de origem do triângulo.

  sides – indica os lados do triângulo.

  radius – indica o raio do triângulo.

  r – indica a rotação do triângulo.

  cor – indica a cor do triângulo.

**Enemy.h**

Armazena as funções referentes ao inimigo:

- inicializa_pos_inimigo: inicializa a posição dos inimigos no mapa de acordo com o nível.
- Inicializa_inimigo: inicializa as variáveis dos inimigos.
- move_inimigo: move os inimigos de maneira aleatória pelo mapa.

**Espada.h**

Armazena as estruturas e funções referentes a funcionalidade da espada do personagem:

- struct Sword: define as variáveis da espada.
  rec – indica o retângulo que representa a espada no display do jogo
  velocidade – indica a velocidade de locomoção da espada
  rotacao – indica a rotação da espada
  direction - indica o sprite de cada direção da espada
  sound – indica o som da espada
  hitwall – indica o som da espada ao colidir com a parede
- inicializa_espada: inicializa as variáveis da espada.
- move_espada: move a espada de acordo com a sua rotação.
- checa_colisao_espada: verifica se a espada atingiu um inimigo.
- checa_ataque: verifica se o jogador solicitou o ataque e, nesse caso, move ela durante determinado tempo.

**Player.h**

Armazena as funções referentes ao personagem:

- inicializa_pos_jogador - inicializa a posição do personagem no mapa de acordo com o nível.
- inicializa_jogador - inicializa as variáveis do jogador.
- move_jogador – move o jogador pelo mapa de acordo com as teclas apertadas.
- checa_colisao_personagem – verifica se o personagem colidiu com um inimigo e, nesse caso, decresce uma vida do personagem.

**Scoreboard.c**

Armazena as funções referentes ao scoreboard:

- DrawScoreboard – desenha os highscores na tela.
- le_arquivo_score – lê o arquivo bin com os highscores e preenche a matriz highscores.
- atualiza_highscores – atualiza a matriz highscores com os novos dados.
- escreve_arquivo – escreve novamente o arquivo bin com a matriz atualizada.
- prog_save – armazena o nome do jogador atual.

**Scoreboard.h**

Armazena a estrutura scores e as funções de scoreboard:

- struct SCORE: define as variáveis das pontuações.
  nome – indica o nome do joagor.
  score – indica a pontuação do jogador.

**Sprites.h**

Armazena a estrutura das sprites e as funções referentes a elas no jogo:

- struct Textures: define as variáveis das sprites.
- carrega_sprites – carrega as sprites do jogo nas variáveis da estrutura.
- descarrega_sprites – descarrega as sprites utilizadas.

**Main.c**

Armazena o programa principal main, que faz uso de todos os arquivos previamente especificados e faz o jogo rodar com suas funcionalidades.