Documentação do jogo

O Apocalipse Da Idade Média

Este documento visa explicar a estrutura e lógica do código fonte do jogo para total compreensão de como foi desenvolvido e idealizado.

Estrutura do jogo

O jogo foi desenvolvido inteiramente na linguagem de programação C com auxílio da biblioteca para desenvolvimento de jogos Allegro.

O código fonte foi separado em duas partes principais, sendo a primeira responsável pela execução e controle do fluxo e chamada de funções e na segunda parte (que está guardada na pasta Modulos) estão as definições de todas as funções e estruturas utilizadas no jogo. Esta abordagem de modularização do código foi escolhida para fins organizacionais, visto que todo o código fonte possui mais de 2000 linhas e seria mais difícil compreendê-lo se fosse escrito em apenas um documento. Junto com o jogo vêm algumas pastas com arquivos específicos:

* Imagens: contém os bitmaps das skins de personagens, menus, mapas, etc.
* Audio: contém as músicas que tocam nos menus, fases e telas do jogo.
* Arquivos: contém um arquivo para armazenamento do save do jogador
* Fonte: contém o arquivo com a fonte usada para texto no jogo

Para fins de melhor compreensão, primeiramente serão apresentadas e esmiuçadas as funções e estruturas utilizadas e, só então, será apresentada a seção de execução. São ao todo 23 funções que tratam desde o posicionamento do jogador até a exibição da tela de vitória. Além das funções, também há três structs que representam objetos dentro do jogo. As definições das funções estão localizadas em “funcoesEProcessos.c” e os cabeçalhos de funções e as structs estão em “funcoesEProcessos.h”. Há também diversas variáveis e constantes globais e defines cujos quais a função será explicada durante a explicação da função ou trecho do código que os utiliza. As principais variáveis globais são:

* #define TAMANHO\_TELA: seu valor é sempre 720.
* const\_int TAM\_ZUMBI e TAM\_HEROI: definem os tamanhos do zumbi e do jogador na tela (1/5 de TAMANHO\_TELA).
* int dificuldade\_jogo: é usada para definir diversas configurações.
* Int OBJETIVO: define em quais dos quadrantes estará o objetivo.
* Int som: permite ou barra a execução de música no jogo, pode ser alterada pelo jogador ao bel-prazer.
* Int skin: é usada para definir qual skin será aplicada ao jogador na execução atual do jogo

Structs

Há três structs definidas dentro do jogo, cada uma representando uma entidade básica. A definição das structs se encontram da linha 1 até a linha 30 de “funcoesEProcessos.h”.

Jogador

Esta struct representa o jogador, o “player” e possui algumas variáveis, cada uma sendo usada para um propósito.

* *int pontuação:* armazena a quantidade de pontos acumulada pelo jogador durante a seção atual
* *int vida*: representa a quantidade de vida atual do jogador. O valor inicial da vida do jogador é 200, sendo reiniciada no início de cada fase do jogo
* *int eixoX e eixoY*: são as variáveis que representa, a posição do jogador na tela.
* *int pocaoDeVida*: representa uma das habilidades do jogador. Seus valores possíveis são 0 e 1. Seu valor padrão é 1 e no início de cada fase este valor lhe é novamente atribuído.

Zumbi

Esta struct representa um inimigo ao longo do jogo. Suas variáveis são:

* *int tipoDeZumbi*: representa o tipo de inimigo.
* *int eixoX e eixoY*: representam a posição do zumbi na tela.
* *int quadrante*: indica o quadrante da tela onde este zumbi pode ser gerado. A posição de um determinado zumbi sempre será definida dentro dos limites de seu quadrante.
* *int vivo*: indica o estado atual deste zumbi, 1 (vivo) ou morto (0).

Há três tipos de Zumbi (inimigo) no jogo:

* Comum: Vida -> 100, Dano -> 20 a 30.
* Padre: Vida -> 100, Dano -> 30 a 40.
* Soldado: Vida -> 150, Dano -> 40 a 50.

Quadrante

Essa struct representa uma unidade do cenário. A intenção da criação dessa struct foi a garantia de homogeneidade na distribuição de zumbis pelo mapa. As variáveis são:

* *Int X\_1, X\_2, Y\_1 E Y\_2*: são variáveis que indicam o ponto inicial e final do quadrante.
* *Int liberado*: essa variável indica se o quadrante está liberado para alocação de um zumbi nele, isso garante aleatoriedade na distribuição dos zumbis, visto a definição de quais quadrantes estarão liberados é aleatória, dependendo apenas da dificuldade da fase. (Isto será explicado claramente na seção de funções).

Funções

As funções serão apresentadas e explicadas em uma ordem que permita a compreensão de seu funcionamento. Caso uma função seja citada durante a apresentação de outra, o índice dessa função nesta seção será informado ao lado, para que o leitor possa localizá-la facilmente.

**1. void inicia(int default\_ou\_save)**

Primeiramente esta função lê de Record.txt (que está dentro da pasta Arquivos) o record de pontuação para o jogo. Após isto, o comportamento da função dependerá do valor de seu argumento. Se *default\_ou\_save* for igual a 0, a função inicializará o jogo (definição dos quadrantes, posições ou tipos de zumbis e etc) através da configuração inicial padrão. Se for igual a 1, a função puxa todos os dados e configura tudo a partir do save.

No padrão, a vida do jogador é 200, seu *eixoX* é 0 e seu *eixoY* é definido pela conta TAMANHO\_TELA – TAM\_HEROI. A *pocaoDeVida* é sempre definida como 1.

Em seguida, a função cria e armazena os zumbis no array global de zumbis, zumbis[]. A quantidade e tipos de zumbis são definidas em função de *dificuldade\_jogo*. Dependendo dos valores de *dificuldade\_jogo* os zumbis são definidos da seguinte forma:

* Dificuldade 1: 12 zumbis, sendo 6 comuns, 3 padres e 3 soldados.
* Dificuldade 2: 18 zumbis, sendo 8 comuns, 5 padres e 5 soldados.
* Dificuldade 3: 24 zumbis, com 8 de cada tipo.

Após isto a função faz a chamada das funções *embaralhaZumbis 2, iniciaQuadrantes 3* e *posicaoZumbis 4.*

**2. void embaralhaZumbis()**

Esta função embaralha os zumbis dentro em zumbis[]. Esta função garante que o tipo de um determinado zumbi seja sempre imprevisível.

**3. void iniciaQuadrantes(int bloquear)**

Primeiramente esta função cria cada um dos 25 quadrantes 144x144 pixels que repartirão a tela, atribuindo a cada um, suas coordenadas e armazenando-os no array global de Quadrante quadrantes[]. Em seguida a função libera todos os quadrantes (para que possuam zumbis), menos o do objetivo (global *int OBJETIVO*), visto que não pode haver zumbi no objetivo. Então, com auxílio da função *geraRand*5, ela bloqueia aleatoriamente <bloquear> quadrantes. Portanto o argumento bloquear define quais quadrantes vão ser aleatoriamente bloqueados (não terão zumbis). Isto se deve à dificuldade atual. Caso a dificuldade seja 1, só 12 zumbis serão criados na função *inicia*1, portanto faz sentido que os quadrantes sem zumbis (bloqueados) sejam definidos aleatoriamente, para o jogador nunca poder prever onde haverá zumbis. No final da função há uma rotina comentada que imprime no console todas as informações sobre cada um dos quadrantes.

**4. void posicaoZumbis()**

Esta função primeiro aloca à cada um dos zumbis em zumbis[] um quadrante. Apenas os quadrantes liberados são alocados e visto que *embaralhaZumbis*2 já foi chamada, o tipo de zumbi para cada quadrante é aleatório. Após alocar um quadrante para cada zumbi, a função gera, com auxílio de *geraRand*()5, uma posição aleatória para o zumbi, dentro de seu quadrante. Por último, esta função imprime no console as informações de cada um dos zumbis.

**5. int geraRand(int x)** Esta função retorna um inteiro aleatório entre 0 e x.

**6. void salvar(int mostrar\_imagem**)

Esta função salva o estado atual do jogo em Save.txt (dentro da pasta Arquivos). O argumento da função, *mostrar\_imagem*, serve para que quando esta função for chamada pelo jogador no modo de navegação (os modos serão explicados mais adiante), apareça na tela uma imagem onde está escrito “Salvando...”. A função salva os dados em Save.txt da seguinte forma:

* Linha 1: é usada para saber se o Save tem dados. Por padrão, na primeira linha de Save está escrito 0. Porém após o jogador salvar o jogo pela primeira vez, ela é permanentemente alterada para 1.
* Linha 2: salva a dificuldade atual do jogo.
* Linha 3: são salvos os dados atuais do jogador na ordem: pontuação, vida, coordenadas e poção de vida.
* Nas próximas linhas serão salvos os dados de cada um dos zumbis, um zumbi em cada linha. (dados serão salvos na ordem: tipo de zumbi, coordenadas, quadrante e estado –vivo ou morto-).
* Por último são salvos os dados de cada um dos quadrantes em cada linha, primeiro é salvo o estado do quadrante (liberado ou bloqueado) e então as coordenadas de seu ponto inicial e final.

**7. void registraRecord()**

Caso a pontuação do jogador for maior que o *record*, esta função sobrescreve o *record* em Record.txt como o novo valor.

**8. void iniciaAllegro()**

Esta função inicializa os objetos allegro (bitmaps, samples, fontes) nas variáveis globais que foram declaradas no início de "funcoesEProcessos.c".

**9. void iniciaSkin()**

Esta função inicia a skin do jogador dependendo de qual ele escolheu na primeira tela do jogo. (essa função foi criada para não ser necessário carregar todas as skins ao jogo para o jogador escolher qual usar só depois).

**10. void desligaAllegro()**

Esta função libera a memória de todas as estruturas Allegro que foram utilizadas.

**11. void processaTeclas()**

Esta função é responsável pelas interações do jogador via teclado durante o modo de navegação (será explicado mais adiante). Para cada tecla que o jogador pressiona durante o modo de navegação, esta função testa se a tecla é de um tipo específico e então realiza a ação correspondente. O jogador pode se mover em todas as direções (com as setas do teclado), pode apertar S para *salvar*6 o jogo, M para voltar ao menu principal e P para usar a poção de vida da fase (se ele já tiver usado a poção na fase atual, ao apertar P nada acontecerá).

**12. int chegouObjetivo()**

Essa função testa constantemente se as coordenadas do jogador estão dentro dos limites do quadrante marcado como o objetivo. Em cada uma das fases do jogo, chegar ao objetivo resultará em um comportamento diferente:

* Fases 1 e 2: a *dificuldade\_jogo* é incrementada em 1 unidade, a música atual para de tocar para a da nova fase ser iniciada e algumas variáveis de apoio voltam aos valores padrão. Após isso, a função *inicia(0)*1 é chamada para configurar a nova fase à partir das configurações padrão com base na dificuldade atual.
* Fase 3: se o jogador chegar ao objetivo nesta fase isso significa que ele venceu o jogo, portanto a tela de Vitória é desenhada com o *record* atual (caso o *record* tenha sido batido pelo jogador aperecerá "Novo Record: valor") pela função *desenhaTelaWin()*18. Em seguida, a tela de créditos é desenhada, a *dificuldade\_jogo* é setada para 1 e, só então, a função inicia(0)1 é chamada novamente para preparar o jogo para começar do início novamente. Por último, o jogo volta automaticamente ao menu (a forma como o jogo volta ao menu será explicada mais adiante).

Esta função retorna 1 caso o jogador tenha chegado ao objetivo e 0 caso não tenha.

**13. int detectouMonstro()**

Esta função verifica se o jogador colidiu com um monstro vivo (compara as coordenadas atuais do jogador com as coordenadas de cada um dos zumbis de zumbis[]. Caso o jogador tenha colidido com um monstro, a função salvo o índice do monstro (em zumbis[]) na variável global *zumbi\_detectado*, que será usada em outras funções. Depois, a função salva a vida do zumbi na global *vida\_zumbi\_atual* com base no tipo do zumbi (100 de vida para zumbis comuns e padres e 150 para soldados). Então a função dá duas orações (habilidade que o jogador pode usar no modo de batalha apenas duas vezes) ao jogador e, por último desenha a tela de alerta de zumbi. A função retorna 1 se um zumbi tiver sido detectado e 0 se não.

**14. void golpes(int quemFoi)**

Esta função recebe como argumento um inteiro que indica que está aplicando o ataque, o valor 0 indica que é o jogador, 1 zumbi comum, 2 zumbi padre e 3 zumbi soldado. Caso o cursos esteja com o valor 0, o golpe "Agua Benta" do jogador será aplicado, para 2 o golpe "Ácido Mycar" e 3 o jogador usa oração. Os golpes dos Zumbis são definidos em função do tipo de zumbi.

**15. void botoes()**

Desenha os botões do menu inicial e do menu de seleção de golpes do modo batalha, de acordo com a posição do *cursor* (variável global cujos valores podem variar de 0 a 2). Caso o cursor esteja em um botão, ele aparecerá mais claro e os outros botões aparecerão um pouco mais escuros. Caso o jogador já tenha usado as duas orações disponíveis por batalha, o botão de orações ficará bloqueado e um pouco mais escuro, indicando que o jogador não pode mais usar orações nessa batalha.

**16. void pontua()**

Esta função atribui pontos ao jogador dependendo do valor do tipo do zumbi da batalha atual (10 pontos por matar um zumbi comum, 15 por um padre e 20 por um soldado).

**17. void menus(ALLEGRO\_EVENT\_TYPE *ev*, int *tipo\_evento*)**

Cuida da movimentação do cursor ao no menu de skins, no menu inicial e no menu de batalha. O primeiro argumento é um evento (nesse caso ele vai ser sempre uma interação via teclado). E o segundo é o tipo de evento. Esse segundo argumento serve para indicar que a função deve desenhar o cursor da tela de skins (*tipo\_evento == 0*). O jogador deve controlar qualquer um dos menus utilizando as setas do teclado e, caso aperte Enter, o estado do cursor (pode variar de menu pra menu) indicará à função o que deve ser feito.

No menu de skins, apertar Enter definirá uma skin para o jogador. No menu inicial, primeira opção iniciará o jogo na fase 1, a segunda iniciará o jogo à partir do save e a terceira mostrará a tela de "Como Jogar", com as instruções do jogo. No modo de Batalha, apertar Enter na primeira opção utilizará o poder "Água Benta", na segunda, o poder "Ácido Mycar" e, na terceira, a oração.

**18. void desenhaTelaWin()**

Desenha a tela de vitória. Caso o *record* tenha sido batido, essa função escreve "Novo Record: <valor>" na tela, caso contrário a função escreve "Record: <valor>".

**19. void desenhaVida()**

No modo Navegação, esta função desenha a barra de vida em cima do jogador. No modo Batalha esta função desenha a barra de vida em cima do zumbi e do jogador.

**20. void desenhaCenario()**

Esta função utilizará a variável global *int modo\_jogo*para definir qual tipo de cenário será carregado. Os valores possíveis para os modos de jogo estão definidos com *#defines* na parte superior dos arquivos "funcoesEProcessos.c" e "ApocalipseMedieval.c". A forma como cada cenário é carregado varia muito, mas geralmente primeiro a função desenha o bitmap que representa o cenário (o background de qualquer tela), depois a função desenha coisas que devem aparecer por cima do cenário, como o objetivo e botões. Esta função está inteira explicada por comentários no código, leia-os.

**21. void desenhaHeroiNavegacao()**

Esta função desenha o bitmap da skin do jogador (definida pela função *iniciaSkin*9). Ela usa a variável global *int movendo* para definir qual estado do personagem será exibido (olhando pra direita, esquerda, cima, baixo ou parado). Esta variável é constantemente modificada durante as interações do usuário via teclado durante o modo Navegação.

**22. void desenhaZumbiNavegacao()**

Desenha o bitmap de zumbi morto nas coordenadas de cada zumbi morto, no modo Navegação.

**23. void animaGolpes()**

Faz a animação do projétil do ataque indo em direção ao alvo. Esta função utiliza as variáveis *int xgolpe* e *ygolpe* para desenhar o projétil e também atualiza as coordenadas durante a movimentação e ao final, para na inversão de turno de ataque o golpe vir no sentido oposto. Quando o projétil atinge um personagem, ele fica vermelho por alguns instantes para indicar que recebeu dano.

Execução

À seguir irei explicar como funciona a execução do jogo, que acontece dentro da função *main,* que está no arquivo "ApocalipseMedieval.c".

Das linhas 1 à 48 estão os includes, defines e variáveis globais utilizadas. Na linha 49 começa a função *main.*

**Main**

Em um primeiro momento, a função main instala as funcionalidades Allegro que serão utilizadas durante a execução. Então a função cria e configura a fila de eventos e inicia o timer. A função *iniciaAllegro8* Após esses processos inicia-se a execução propriamente dita.

A execução inteira está dentro de um *while(playing == 1)*, caso o jogador feche o jogo (clicando no x da tela), 0 é atribuído à playing e então o loop while é quebrado. Caso o loop seja quebrado, a função *desligaAllegro*10 é chamada e a execução termina.

Dentro do while há um *switch* que recebe a variável global *int modo\_jogo,* é essa variável que controlará qual tela será mostrada dependendo do modo de jogo atual. Há 5 modos de jogo possíveis (que estão definidos em defines no topo da página):

* **NAVEGACAO**: modo de navegação e exploração do cenário. Dentro modo o jogador pode usar as setas para se locomover apertar P para curar toda a sua vida (uma vez por fase). Apertar M fará o usuário voltar ao menu principal e apertar S salvará o jogo. O *modo\_jogo* é trocado para o modo batalha se a função *detectouMonstro*13 retornar 1, indicando que o jogador colidiu com um monstro. Caso o jogador chegue ao objetivo, a função *chegouObjetivo*12 definirá o comportamento do jogo dependendo de *dificuldade\_jogo.* A definição desse comportamento foi explicada no item 12 da seção "Funções" deste documento.
* **BATALHA:** o modo de batalha é iniciado quando o jogador colide com um inimigo. O modo de batalha é iniciado no turno do jogador, que deve utilizar o cursor (setas do teclado) e o botão Enter para escolher qual golpe que aplicar no imigo. Quando o jogador escolhe o golpe, a função *menus*17 é chamada e ela chama outras funções que são responsáveis por processos como aplicação de dano e pontuação ao jogador. Caso o zumbi morra, o jogo volta ao modo de navegação e o corpo do zumbi agora aparece no mapa, e os pontos referentes ao tipo do zumbi são adicionados à pontuação do jogador. Caso o zumbi não morra, o turno passa então para o zumbi que ataca o jogado. Caso o jogador não morra, o turno passa para o jogador novamente e o ciclo recomeça. Caso o jogador morra, a tela de game over é exibida e depois de 5 segundos o jogador volta ao menu principal. Antes de voltar, a *dificuldade\_jogo* é mudada para 1 e a função *inicia(0)1* é chamada para gerar novos zumbis em posições aleatórias. Isto faz com que na próxima vez que o jogador inicie um novo jogo no menu, o jogo recomece com zumbis em novas posições e tipos.
* **MENUSKINS, MENU e COMOJOGAR:** a primeira é responsável por controlar a tela de seleção de skins (primeira tela que aparece, portanto, *modo\_jogo* está sempre com o valor *MENUSKINS* por padrão) e a segunda é responsável por controlar o menu inicial. O modo *COMOJOGAR* é apenas responsável pela tela do último botão do menu, esse modo explica o funcionamento do jogo e dá informações extras.

Durante os modos de BATALHA, MENU e NAVEGACAO é possível desabilitar ou habilitar a música clicando no ícone superior de "auto falante", esse procedimento é controlado pela variável global **int som,** se seu valor for igual a 0, ela barra a música, se for igual a 1, ela libera e se for igual a 2, ela indica que há uma música tocando.