UNIVERSIDADE FEDERAL DE MINAS GERAIS

PROGRAMAÇÃO E DESENVOLVIMENTO DE SOFTWARE 1

MATEUS DE SOUZA GONTIJO

DOCUMENTAÇÃO OFICIAL DO TP ALLEGRO DESENVOLVIDO:

CANYON BOMBER

Belo Horizonte

2023

Sumário

1. INTRODUÇÃO

O presente documento visa fazer uma documentação detalhada do Trabalho Prático, em Allegro, desenvolvido pelo aluno Mateus de Souza Gontijo, para a disciplina de Programação e Desenvolvimento de Software 1.

O jogo desenvolvido - Canyon Bomber -, consiste basicamente em duas naves (cada nave representa um jogador) que percorrem a tela de uma lado ao outro e devem atirar nos alvos abaixo.

Figura 1 - Ilustração do jogo sendo executado.

2. OBJETIVO DO JOGO

O objetivo do jogo, como mencionado na introdução, é atingir os alvos que estão na parte de baixo da tela. Os alvos são representados por uma matriz de blocos, divididos por linhas. São 8 linhas no total e cada linha (corresponde a uma tom de roxo), contém blocos que valem uma certa quantidade de pontos.

Quando um dos jogadores atirar, caso o tiro atinja os blocos, eles serão destruídos e os pontos serão contabilizados para o jogador que atirou. Porém, caso o tiro não atinja nenhum

alvo, o jogador perde 1 vida (cada jogador tem 3 vidas no início da partida). Dito isso, existem duas condições para fazer com que o jogo encerre:

- Caso não hajam mais blocos restantes;
- Caso um dos jogadores chegue a zero pontos de vida.

3. DOCUMENTAÇÃO DO CÓDIGO

3.1. GERAL

O código implementado para o jogo está dividido em diversas seções, separadas visualmente por estruturas como mostra a imagem abaixo:

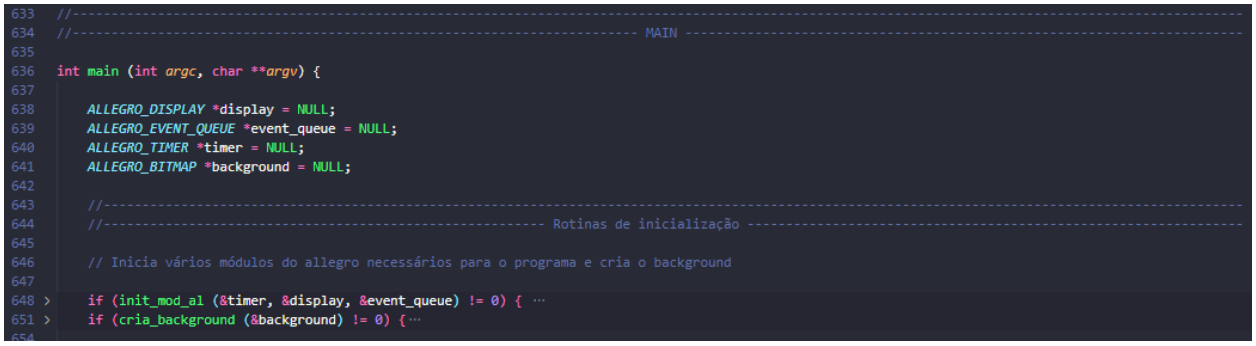

Figura 2 - Ilustração das divisões feitas no código utilizando "// -----..."

No geral, o código consiste em funções diversas, que são chamadas no início, meio e fim da função principal (main). O jogo em si ocorre durante o loop "while (playing) { ... }", que funciona por uma lógica de aguardar eventos enviados pelo jogador (como cliques de mouse, teclado, etc) para fazer determinadas ações. Além disso, durante esse loop mencionado, há uma constante atualização da tela, do background e das naves (usando funções para, por exemplo, atualizar a posição da nave e dos tiros).

Após o loop principal, outras funções são chamadas. Dentre elas as funções que salva no arquivo do jogo, a vitória do jogador que venceu (para manter um histórico de vitórias). Outras funções de destaque são as de encerramento do jogo, para destruir bitmaps criados, etc.

3.2. SEÇÃO 1 - MACROS DO PROGRAMA

Logo após a chamada de incluir as funções padrão C e as da biblioteca allegro (#include ...), há uma seção que contém todos os macros e variáveis globais do programa. O intuito é reunir as principais variáveis utilizadas ao longo do jogo para tornar a edição e manutenção do código mais viáveis de serem feitas. Alguns exemplos de variáveis globais dessa seção são:

- const float FPS = 75 - Para controlar o FPS do jogo;
- const int SCREEN_W = 960 - Largura da tela;

- const int SCREEN_H = 540 - Altura da tela;
- Dentre várias outras.

3.3. SEÇÃO 2 - STRUCTS DO PROGRAMA

A segunda seção do programa, reúne todos os structs utilizados ao longo do jogo. Eles estão representados na figura abaixo:

```
36   // Struct do tiro
37 > typedef struct tiro {···
44   } Ttiro;
45
46   // Struct da nave
47 > typedef struct nave {···
63   } Tnave;
64
65   // Struct dos alvos da matriz de blocos
66 > typedef struct bloco {···
72   } Tbloco;
73
74   // Struct dos icones visuais (p1/p2/vidas)
75 > typedef struct player_icons {···
86   } Tplayer_icons;
87
88   // Struct de texto
89 > typedef struct text {···
95   } Ttext;
```

Figura 3 - Structs do programa.

- Tnave e Ttiro controlam os principais aspectos da nave dos jogadores (como vida, direção, velocidade, etc) e de seus tiros, respectivamente;
- TBloco é utilizada na matriz de blocos para facilitar alguns processos (como conferir se o tiro encostou ou não no bloco através das coordenadas do bloco);
- Tplayer_icons foi criada para reunir as imagens (bitmap allegro) dos ícones de vida e informação dos jogadores;
- Ttext contém propriedades de cor e estilo de fonte, utilizados no jogo.

3.4. SEÇÃO 3 - FUNÇÕES DO PROGRAMA

A terceira parte é uma das mais importantes, pois contém todas as funções criadas para o jogo (ou seja, são as funções além das que já vêm prontas do Allegro). Ademais, a seção 3 está subdividida em algumas categorias de funções:

- Funções mais gerais:
    - **[init_mod_al]:** inicia diversos módulos allegro;
    - **[encerra_mod_al]:** encerra diversos módulos allegro;
    - **[encerra_player_bitmaps]:** encerra bitmaps (imagens carregadas) especificamente relacionadas aos jogadores;
    - **[init_text]:** inicia as propriedades da struct Ttext.

- Funções relativas aos arquivos:
    - **[ler_vitorias]:** lê dos arquivos do p1 e p2, o número de vitórias registrado;
    - **[salvar_vitorias]:** atualiza os dados dos arquivos, referente ao número de vitórias;

- Funções de game over:
    - **[checar_gameover]:** verifica se uma das condições para o game over ocorreu;
    - **[tela_gameover]:** desenha a tela final de game over, com as informações de pontos da partida e vitórias dos jogadores;
    - **[confere_vencedor]:** atualiza a propriedade ".win" atrelada às naves dos jogadores, de acordo com o player vencedor;

- Funções relativas aos jogadores:
    - **[printar_pontos_e_vida]:** printa no terminal pontos e vida dos jogadores;
    - **[criar_player_icons]:** carrega as imagens da struct Tplayer_icons;
    - **[desenhar_player_icons]:** desenha na tela as vidas e informações dos jogadores (p1/p2);

- Funções relativas ao background e grid de blocos:
    - **[cria_background]:** carrega a imagem de background;
    - **[init_blocos]:** inicia as propriedades de Tbloco para todos os blocos da matriz (como cor, pontos atrelados, etc);
    - **[desenhar_blocos]:** desenha na tela a matriz de blocos;
    - **[conferir_destruicao_grid]:** verifica se todo o grid foi destruído;

- Funções relativas às naves e aos tiros:
    - **[cria_nave_e_tiro]:** inicia as propriedades dos structs Tnave e Ttiro;
    - **[atualiza_nave_e_tiro]:** atualiza as posições da nave e do tiro;
    - **[desenha_nave_e_tiro]:** desenha a nave e o tiro de acordo com suas posições;

3.5. SEÇÃO 4 - MAIN

Na função main do programa, existem algumas seções, como a seção de "Rotinas de Inicialização" e a seção voltada para a criação das naves, grid, etc. Basicamente, essas partes do código irão criar variáveis (usando struct's feitos) e chamar funções de inicialização.

Figura 4 - Ilustração do início da função main do programa.

Logo após essa parte, o timer, display e keyboard serão adicionados na fila de eventos e o loop principal do jogo se inicia. Basicamente, o loop é controlado pela variável playing: enquanto ela for igual a 1 o jogo continuará ocorrendo. Porém, em certos eventos, o valor de playing mudará para zero (de forma a parar o loop).

Ressalta-se que no loop, há uma lógica implementada para 3 tipos de eventos:

- **Passagem de tempo (segundos):** atualizar e desenhar as naves, tiros, grid, etc é feito dentro dessa parte (de forma a atualizar a tela várias vezes por segundo (FPS));
- **Clicar no botão de fechar a janela:** essa ação causa um encerramento abrupto do jogo e pode ser considerada o caso excepcional de condição de parada. Isso forçará o jogo a ir para a tela de game over;

- **Clicar em alguma tecla do teclado**: nesse caso, as teclas implementadas são as que os jogadores utilizam para atirar ("S" para o player 1 e "K" para o player 2).

```
714     int playing = 1;
715     while (playing) {
716
717         ALLEGRO_EVENT ev;
718         al_wait_for_event (event_queue, &ev); // Espera por um evento e o armazena na variavel de evento ev
719
720         // Se o tipo de evento for um evento do temporizador, ou seja, se o tempo passou de t para t+1
721 >       if (ev.type == ALLEGRO_EVENT_TIMER) {···
749
750         // Se o tipo de evento for o fechamento da tela (clique no x da janela)
751 >       else if (ev.type == ALLEGRO_EVENT_DISPLAY_CLOSE) {···
754
755         // Se o tipo de evento for um pressionar de uma tecla
756 >       else if (ev.type == ALLEGRO_EVENT_KEY_DOWN) {···
780
781     } // Fim do while
```

Figura 5 - Loop principal do jogo

3.6. SEÇÕES 5 E 6- TELA GAME OVER E ENCERRAMENTO

A penúltima parte do código faz a chamada de funções necessárias para atualizar as vitórias de acordo com o jogador que ganhou o jogo e exibir as informações na tela de game over, como mostra a figura abaixo:

```
783     //------------------------------------------------------------------------------------------------------
784     //------------------------------------------------------ Tela de resultados finais ----------------------
785
786     // Printa (no terminal) a pontuação final dos jogadores e atualiza as vitórias dos jogadores
787
788     printar_pontos_e_vidas (p1nave, p2nave);
789     confere_vencedor (&p1nave, &p2nave, &vitorias_p1, &vitorias_p2);
790     salvar_vitorias (vitorias_p1, vitorias_p2, p1nave, p2nave);
791
792     // Exibe a tela final de gameover
793
794     al_draw_bitmap (background, 0, 0, 0);
795 >   if (tela_gameover (&txt, p1nave, p2nave) != 0) {···
800     al_flip_display ();
801     al_rest (6.0);
802
803     //------------------------------------------------------------------------------------------------------
804     //------------------------------- Procedimentos de fim de jogo (fecha a tela, limpa a memória, etc) ----
805
806     encerra_player_bitmaps (&p1nave, &p2nave, &icones_p1_e_p2);
807     encerra_mod_al (timer, display, event_queue, background);
808
809     return 0;
810 }
```

Figura 6 - Parte final do código.

Já a última parte, consiste única e exclusivamente na chamada das funções de encerramento: [encerra_player_bitmaps] e [encerra_mod_al].

4. INFORMAÇÕES SOBRE O ENVIO DO TP

Para o envio do trabalho, não foi possível enviar o arquivo completo via moodle devido ao tamanho em bytes. Portanto, o trabalho foi colocado em um repositório público no github. Abaixo, o link para o repositório:

- https://github.com/ateteu/canyon-bomber-game.git

O trabalho foi desenvolvido pelo aluno Mateus de Souza Gontijo em 2023/2. As imagens utilizadas no trabalho (naves, vidas, etc) são de autoria própria.