LCOM - Final Report

Allan Borges de Sousa - 201800149
Juliane de Lima Marubayashi - 201800175

6 de Janeiro de 2020

1 Instruções de Usuário

1.1 Menu

O jogo começa inicialmente apresentando o menu, o qual expõe ao usuário o nome do jogo e três opções:

- Start
- Instructions
- Exit

Fig.1 - Menu

1.2 Ask name

Em seguida ao menu, o jogo pergunta o nome de cada um dos jogadores, para que no final possa ser exibido o nome do vencedor. É possível apagar o nome e reescrevê-lo. Este módulo foi feito de modo que o nome seja sempre mostrado no centro do ecrã.

Fig.2 - Jogo pergunta nome

1.3 Game - Como funciona

O jogo deve ser jogado a dois obrigatoriamente. O objetivo é um canhão acertar o outro três vezes e cada vez que um canhão é atingido, ele perde um coração de vida. O ângulo da trajetória do tiro é calculado em relação a posição do mouse. As regras consistem em:

- Os canhões não podem atirar simultânemente;
- Uma vez que o canhão da direita atire, por exemplo, é preciso esperar que o canhão da esquerda atire para poder lançar a bola novamente e vice-versa;
- Os canhões não podem se aproximar mais do que 300 pixels;
- Se um canhão pode ultrapassar os extremos da tela. Esta é uma medida proposital e defensiva. O canhão pode sair da tela, mas enquanto estiver fora não poderá atirar;
- Canhões podem mover-se ao mesmo tempo;

Fig.3 - GamePlay

1.4 Instruções

Apertando a opção **Instructions** no menu principal, as instruções do jogo são exibidas. Resumidamente temos:

- Letras "a"e "d"movem o canhão da esquerda;
- Setas da direita e esquerda movem o canhão da direita;
- Posição do rato define o ângulo inicial da parábola da bola do canhão;
- Botão esquerdo do rato atira a bola.

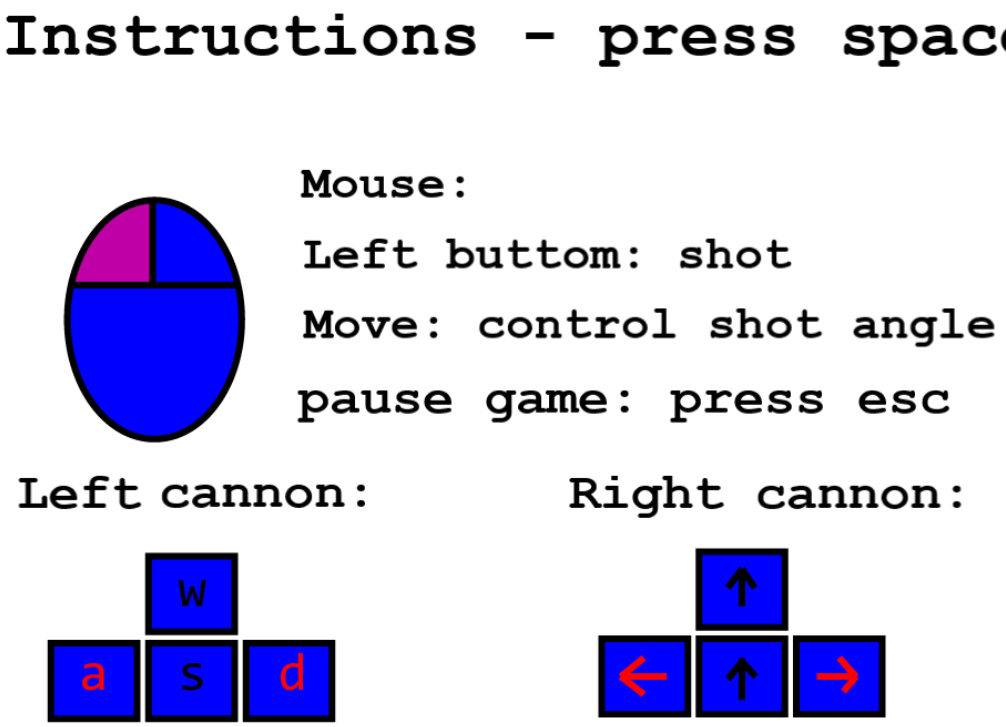

Fig.4 - Intruções

1.5 Fim de jogo

Quando um dos canhões perde todos os corações a tela do ecrã muda para e assim que o usuário aperta space, o jogo é reiniciado.

Fig.5 - Fim de jogo

2 Estado do projeto

Na tabela a seguir é mostrado resumidamente o uso de cada device no projeto. Detalhes estão descritos em subsections seguintes.

Device	Uso	Int.
Timer	Controle do frame rate	S
KBD	Movimento dos canhões	S
Mouse	Movimento do mouse, cáculo de angulo e tiro	S
Video Card	Exibição de imagens	S
RTC	Lê data/hora e faz o update a cada frame	N

2.1 Timer

O timer foi basicamente usado na função `int(proj_main_loop)(int argc, char *argv[])`. Nesta função, lidamos com os interrupts e a frequência do timer é posta a 64 com a função `timer_set_frequency(0, frequency);`.
Assim, a cada interrupt do timer a informação do ecrã é atualizada.
Além disso, o timer é responsável por calcular quantos segundos se passaram desde o "inicio"de um tiro, a fim de usar o tempo na fórmula da trajetória parabólica do tiro, nomeadamete:

$$s_x = vt$$

$$s_y = s_0 + v_0 t + \frac{gt^2}{2}$$

$$g = 9.8$$

O cálculo dos segundos é feito dentro do interrupt do timer.

2.2 KBD

Foi usado para:

- **Aplicação de controle**
- **Text input**

Resumidamente, o kbd tem as seguintes funções:

Função de cóodigo	Uso	Arquivo
`void handleControls`	Lidar com os movimentos dos canhões	game.c
`int(proj_main_loop)`	Lidar com mudança de estados do jogo (pause - esc)	proj.c
`void write_string`	Escreve a string dada na tela	textInput.c
`void askname1`	Constroi a string digitada pelo usuário	textInput.c
`char get_letter`	Retorna a letra digitada pelo usuário	textInput.c e proj.c

2.3 Mouse

Para o mouse foram usados:

- **Posição** (1)
- **Botões** (2)

Função de código	Uso	Arquivo	Tipo
`void animateMouse`	Obter a nova posição do mouse	sprite.c	1
`void get_angle`	Obter angulo entre a posição do mouse e o canhão	game.c	1
`void check_start *`	Checar se botão esquerdo foi apertado na área indicada	proj.c	2
`int(proj_main_loop)`	Iniciar tiro do canhão	proj.c	2

* void check_exit e check_instructions tem basicamente a mesma função

2.4 Video Card

Sobre as características do video card:

Modo	Resolução do ecrã	Modo da cor	Numero de cores
0x105	1024 x 768	Indexado	64

Para a sua implementação foram usados:

- **Double buffer** (1)
- **Movimento de objetos (detecção de colisões, animação de sprites)** (2)
- **Extra: rotação e inversão de sprites** (3)
- **Exibição de texto e fontes** (4)

Funções utilizadas:

Função de código	Uso	Arquivo	Tipo
`void memvideo_cpy`	Passa informalão do buffer para mem	graphic.c	1
`void malloc_buffer`	Aloca espaço para o buffer	graphic.c	1
`void clear_buffer`	Limpa o buffer	graphic.c	1
`int drawPixel`	Desenha pixel no buffer	graphic.c	1
`int check_start`	Vê se a animação do menu inicia deve ser feita	menu.c	2
`void cal_trajetory`	Desenho parabólico do tiro	game.c	2
`void draw_p1_hp`	Animação dos corações de vida	game.c	2
`int(proj_main_loop)`	Animação dos sprites do menu inicial	proj.c	2
`bool checkCollision`	Checa colisão entre bola e canhão	game.c	2
`int drawSpriteInvertedRotated`	Inverter sprite e rotacionar	sprite.c	3
`int drawSpriteRotated`	Rotacionar sprites	sprite.c	3
`void drawInvertedSprite`	Inverter sprites	sprite.c	3
`void drawMenu`	Desenhar texto do menu inicial	sprite.c	4

2.5 RTC

Todas as funções do RTC estão implementadas no arquivo rtc.c Para sua implementação foram usados:

- **Leitura da data/hora** (1)

Função de código	Uso	Arquivo	Tipo
`int read_rtc`	Requisição de leitura da data/ horario	rtc.c	1
`int(proj_main_loop)`	Chamada da função read_rtc	proj.c	1
`int gameDraw`	Desenha sol ou lua de acordo com data/ hora	game.c	1

2.6 UART

O uart não foi utilizado no trabalho

3 Organização e estrutura do código

Atenção Módulos não apresentados como timer.c e mouse.c não foram considerados relevantes o suficiente para estarem representados em subssecções, uma vez que não sofreram alterações significativas em relação aos arquivos produzidos nos labs das aulas práticas.

3.1 menu.c

Este submódulo lida com o menu inicial do jogo.
A função principal é a `void menuDraw` que é responsável por gerir as atividades visuais do menu que se resumem em:

- Desenhar sprites;
- Gerir animação dos sprites do menu;
- Animar mouse;
- Transferir informção do buffer para a memória de vídeo.

Podemos dizer que este representa 20% do trabalho.
Segue a divisão de trabalho para este módulo considerando apenas as funções principais:

Funções/Componentes	**Allan Borges**	**Juliane Marubayashi**
`menuDraw`		X
`check functions`	X	
`init_menuSprites`	X	X

3.2 game.c

Este módulo gere o jogo.
A sua função principal é `int gameDraw` que é responsável por organizar a lógica de jogo e a interface gráfica.
Sua estrutura de dados principal (definida em sprite.h) é a struct Sprite, que armazena principalmente a velocidade e a posição dos canhões na interface de jogo.
Atenção Esta função não lida totalmente com a jogabilidade. Algumas decisões são tomadas nos interrupts presentes no módulo proj.c.
Sendo assim, seu percurso programacional é:

- Checar o fim de jogo;
- Atualizar posição dos canhões;
- Verificar a interface deve ser dia ou noite;
- Desenhar no buffer a interface do jogo: canhões, chão, mouse, céu, sol/lua;
- Desenhar bola de tiro se necessário;
- Mover informação do buffer para a memória de vídeo

Podemos dizer que este módulo representa algo em torno de 40% do trabalho, uma vez que se trata da tarefa principal do jogo.

Funções/Componentes	Allan Borges	Juliane Marubayashi
`init_gameSprite`	X	X
`draw_p1_hp`	X	
`checkCollision`	X	
`check_for_endgame`	X	
`gameDraw`		X
`cannon1_moviment`		X
`handleControls`		X
`draw_interface`		X
`cal_trajectory`		X
`handle_shot`	X	X
`get_angle`	X	

3.3 textInput.c

Este módulo é responsável por gerenciar o programa quando pede-se os nomes dos jogadores.
Suas funções principais são `void askname1()` e `void askname1()`.
As funções deste módulo foram implementadas por Juliane Marubayashi. Consideramos que este módulo representa cerca de 10% do trabalho total.

3.4 proj.c

Este módulo é responsável pro gerenciar os modos de jogo com base em estados. Além disso, nele ocorre a chamada de interrupts e leitura de dados do keyboard e mouse.
Fora as funções mencionadas acima, ainda ocorre:

- Iniciar sprites do menu incial e do jogo;
- Definir se o modo do jogo deve ser dia ou noite;
- Lidar com interrupts:
  - Cálculo dos segundos desde o início de um tiro até seu fim [TIMER];
  - Se state = MENU, desenhar o menu inicial [TIMER];
  - Se state = GAME, desenhar o jogo [TIMER] ;
  - Fazer update do timer a cada frame [TIMER]
  - Se state = PAUSE, o jogo está em pausa e tem a mesma visão do menu principal, mas ao apertar start não é perguntado o nome dos jogadores novamente [TIMER]
  - Verificar se ESC foi apertado, a fim de voltar ao menu inicial [KBC];
  - Colher dados do KBC [KBC] ;
  - Se state = NAME1 ou NAME2 é perguntado o nome dos jogadores [KBC]
  - Colher dados do rato [MOUSE];
  - Verificar opções apertadas no menu inicial [MOUSE];
  - Verificar se um tiro deve ser feito [MOUSE]

Este módulo representa algo em torno de %30 do trabalho.
Divisão do trabalho:

Tarefas/Componentes	Allan Borges	Juliane Marubayashi
Ler rtc data/ hora		X
Estados		X
Interrupts	X	X

3.5 xpm.h

Este arquivo possui todos os xpm utilizados no trabalho.
Para os xpms para os quais foram adquiridos na internet, as fontes estão no fim deste relatória ne sessão de fontes.

Este arquivo representa cerca de 10% considerando principalmente o tempo que levamos para criar os xpms.
Os xpm's foram criados em conjuntos. Cada componente da dupla compôs cerca de metade dos xpm's presentes no arquivo.

4 Detalhes de implementação

Para o projeto, usamos e adquirimos mais conhecimento relativo aos seguintes tópicos:

- State machine: usado para mudança de estados dos jogo entre gamemode, menu e gameover.
- Frame generation
- Rotação de sprites utilizando matriz de rotação
- Inversão de sprites
- Aprendemos a criar sprites a partir de pixel art utilizando outros softwares
- Colisão de sprites
- Animação de sprites
- Orientação de objetos, uma vez que a informação dos canhões era armazenada em structs.
- Escrita em latex

4.1 Detalhes sobre o rtc

O jogo se baseia nas estações no ano e na hora para definir se a interface gráfica deve ser noite ou dia, uma vez que há divergencia na hora em que o sol se põe e nasce entre as estações do ano. Assim, no arquivo proj.c lidamos com tais divergencias, para que a interface dia-noite seja o mais coerente possível.

5 Conclusão

Com o fim do projeto, tivemos grande aproveitamento da cadeira e sentimos que com este jogo, fomos capazes de testar nossos conhecimentos adquiridos com os trabalhos laboratoriais.

5.1 Sugestões de melhoria

Acreditamos que as aulas teóricas poderiam ter mais exemplos de demostração de código, uma vez que ajudaria no engajamento da matéria.
Sugerimos também abordar o UART em aulas laboratorias.

6 Fontes

6.1 Imagens

Canhão: https://www.pixilart.com/art/angry-sun-mario-3-294e69de4b0e53d
Sol: https://www.pixilart.com/art/angry-sun-mario-3-294e69de4b0e53d

6.2 Ferramentas Online

PIXELART : https://www.pixilart.com/draw