U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Projeto Final de Laboratório de Computadores
Turma 13 - Grupo 5

Realizado por:

- Rodrigo Pinto Pesqueira Gaspar Pombo - up202105374
- Liess Pereira Aouimeur - up202206296
- Pedro Jorge das Neves Pinto Vieira - up202206230

Índice:

Introdução……..…3
Sprites……..3
Instruções……4
- Menu inicial…….4
- Quit……4
- Play……5
- Game Over……..6
- High Scores……..7
- Nota adicional……..7

Controlos……8
Estado do projeto……..……9
- Tabela de funcionalidades……9
- Tabela de dispositivos……..……10

Dispositivos……..11
- Rato……11
- Teclado……..……..11
- Timer……..12
- Graphics……..……12
- RTC……13

Organização do código/estrutura…….……14
- KBC module……..……14
- Graphics module……..14
- Keyboard module……..……15
- Mouse module……..……15
- RTC module……..……16
- Timer module……..……..16
- Game state module……..……16
- Utils module……..……17
- Main module……..……17
- Sprites module……..……18

Detalhes de implementação…….……19
Function call graph……20
Conclusões……..21
Índice de figuras……..22
Créditos……22
Demonstração……..22

Introdução

Debates sobre a existência da cidade de Leiria no território português têm vindo a aumentar com o decorrer dos anos, dividindo a população em duas partes, crentes e negacionistas. Este debate deu origem à ideia do nosso projeto “Leiria”. Não sabendo se a cidade realmente existe ou não, o nosso grupo decidiu desenvolver um jogo mostrando como poderia ser a cidade, pelo menos da maneira que nós a imaginamos ser.

O jogo consiste num mapa fechado, a cidade de Leiria, no qual o jogador é representado por um personagem coberto por uma manta vermelha. Em cada canto do mapa se encontram “spawners”, canos de esgoto de onde os monstros, “habitantes de Leiria”, aparecem, e existem múltiplos muros funcionando como obstáculos, tanto para o jogador como para os monstros, espalhados pelo mapa. Estes monstros têm como objetivo atacar o jogador, matando-o em caso de colisão. No entanto, o jogador tem a possibilidade de matar os monstros clicando com o botão esquerdo do rato em cima deles se estes estiverem num bloco adjacente ao mesmo, atribuindo 1 ponto ao jogador por cada monstro.

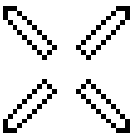
O objetivo do jogo é obter o máximo de pontos possível antes de morrer, podendo ver a pontuação e o tempo decorrido desde o início da partida no topo do ecrã. Estes resultados serão guardados, permitindo ao jogador ver as suas melhores tentativas no menu de “High Scores”.

Sprites

Jogador

Monstro

Ponteiro do rato



Instruções

Menu Inicial

Ao abrir o jogo, deparamo-nos com o menu inicial. Neste, além do logotipo do jogo, temos três opções diferentes na qual o jogador pode clicar: “Play”, “High Scores” e “Quit”.

Figura 1 - Menu inicial

Quit

Ao pressionar o botão esquerdo do rato sobre o botão de “Quit”, o jogo é fechado e voltamos para o modo de escrita do Minix.

Play

Ao pressionar com o botão esquerdo do rato sobre o botão de “Play”, somos enviados para o jogo, começando automaticamente a partida. Além do mapa, do “Time” e do “Score” como explicados na introdução, também podemos observar um botão de “Quit” no topo do ecrã à direita. Ao clicar neste botão o jogador é enviado para o menu inicial. No caso de o jogador clicar no botão de Esc, exatamente como o botão “quit” do menu inicial, o jogo é fechado e voltamos ao modo de escrita do Minix.

Figura 2 - Gameplay do jogo

Game Over

Quando o jogador entra em contacto com um monstro, o jogo acaba, parando a movimentação dos monstros e do jogador e fazendo aparecer uma mensagem de “Game Over” por cima do mapa. No entanto, o tempo e a pontuação do jogador continuam visíveis, tendo também a possibilidade de clicar no botão de “Quit”, sendo enviado para o menu inicial. Continua a ser possível clicar no botão Esc, fechando o jogo e voltando para o menu de escrita do Minix.

Figura 3 - Game Over

High Scores

Ao pressionar com o botão esquerdo do rato sobre o botão de “High Scores”, somos enviados para o menu dos high scores. Este menu mostra a data em que a pontuação foi conseguida, tendo o dia, mês, ano, hora, minuto e segundo, o tempo de jogo para essa pontuação e a pontuação, para cada uma das 5 melhores pontuações que o jogador obteve.

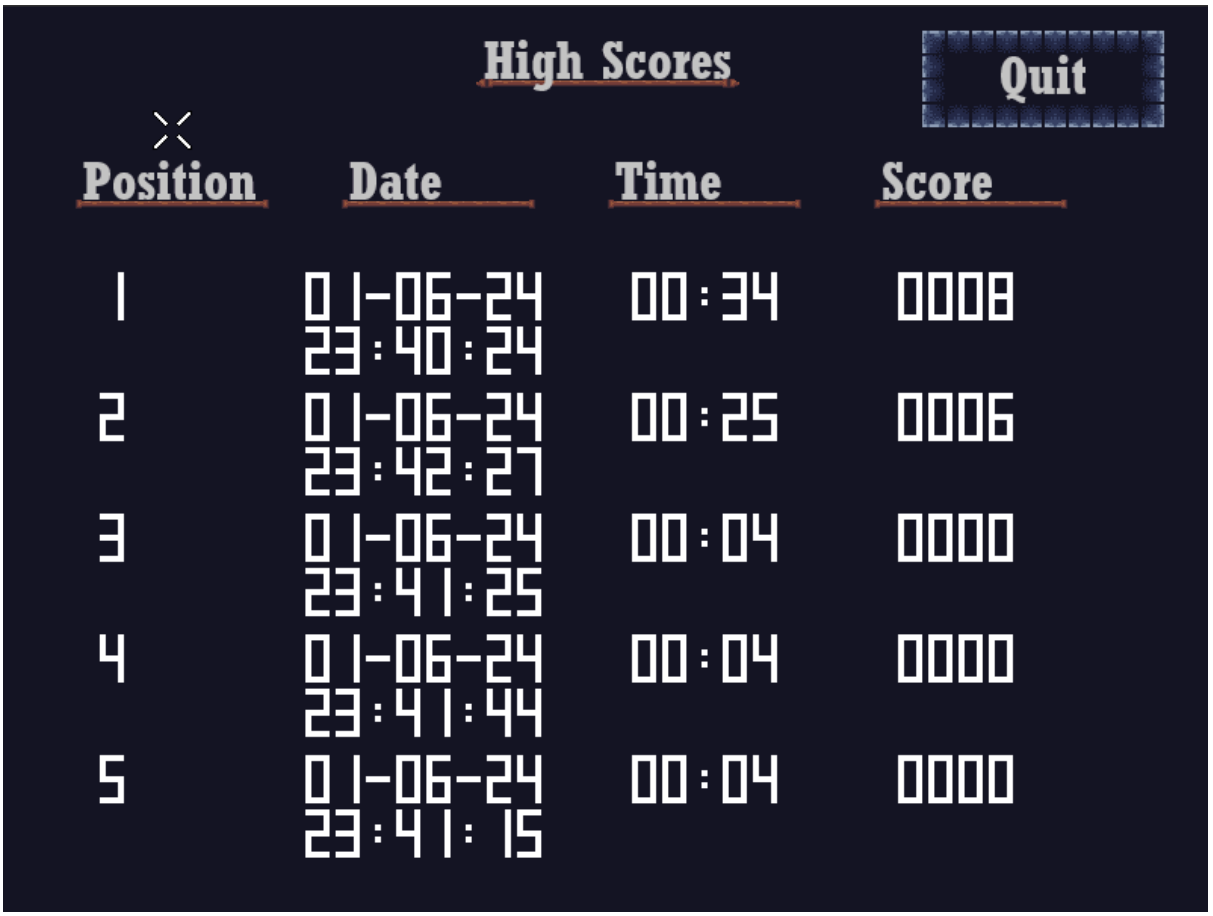


Figura 4 - Menu de High Scores

Nota:

Para o funcionamento dos high scores e do game over é necessário criar um ficheiro de texto e o path desse ficheiro tem de ser definido dentro do ficheiro “game_state/game_state.h” com o nome “HIGH_SCORES_FILE_PATH_NAME”:

```
#ifndef _LCOM_GAME_STATE_H_
#define _LCOM_GAME_STATE_H_
//////////////////////////
///// define high_scores.txt file path name (change this to the correct path name for your computer)
#define HIGH_SCORES_FILE_PATH_NAME "/home/lcom/labs/g5/proj/src/high_scores.txt"
//#define HIGH_SCORES_FILE_PATH_NAME "/home/lcom/labs/proj/src/high_scores.txt"
//////////////////////////
```

Figura 5 - Definição de path do ficheiro de texto do High Score

O ficheiro de texto pode estar vazio e só guarda até 5 linhas, cada linha correspondendo a um high score no leaderboard.

Controlos

A movimentação do personagem é controlada através das teclas W, A, S e D, fazendo o personagem ir para cima, para a esquerda, para baixo e para a direita respetivamente.

Para atacar os monstros é necessário a utilização do rato, clicando com o botão esquerdo em cima do monstro que pretende atacar, no entanto só funciona se o monstro estiver num bloco adjacente ao bloco onde se encontra o jogador.

Estado do projeto

Tabela de Funcionalidades

Funcionalidades	Dispositivos	Estado de Implementação
Movimento jogador	Teclado, timer e graphics	Completo
Movimento e ataque monstros	Timer e graphics	Completo
Spawn monstros	Timer e graphics	Completo
Atacar	Rato e graphics	Completo
Mostrar tempo de jogo	Timer e graphics	Completo
Mostrar informações de score e high score	RTC, timer, graphics	Completo
Mostrar menus e jogo	Graphics	Completo
Navegação de menus	Rato e graphics	Completo
Jogar em modo Multijogador	Serial Port e graphics	Não Implementado

Tabela de dispositivos

Dispositivos	**Funcionalidades**	**Modo**
Rato	Atacar os monstros e mudar de menu	Interrupções
Teclado	Movimentação do jogador	Interrupções
Timer	Movimentação dos monstros e do jogador, spawn dos monstros e tempo de jogo	Interrupções
Graphics	Mostrar o jogo, os menus e todas as informações, mostrar o jogador e os monstros e mostrar o ponteiro do rato	Interrupções
RTC	Mostrar data dos high scores	Por pedido

Dispositivos

Rato

O rato é usado para a escolha de opções representadas por botões nos diferentes ecrãs (Começar novo jogo, Highscore e Quit) e para atacar os monstros durante a gameplay.

Em todos os ecrãs, o cursor do rato é representado por um xpm parecido com uma “mira”, aludindo à vertente de combate oferecida pelo jogo, deixando-o visualmente mais apelativo.

A posição do rato é calculada através das coordenadas que o próprio Minix usa, onde o mesmo começa sempre no canto superior esquerdo (x,y) -> (0,0) e a nova posição é calculada em relação à última, através dos atributos delta_x e delta_y do struct mouse_ev.

A implementação de todas as funções relacionadas ao rato está no ficheiro KBC_mouse.c, estando também no ficheiro keyboard_and_mouse.c implementadas funções da KBC que o rato usa.

Teclado

O teclado é usado na escolha da direção de movimentação do jogador durante um jogo e para fechá-lo, caso haja algum problema inesperado com o Minix.

As teclas W, A, S e D realizam os movimentos do personagem, à distância de um quadrado para cima, esquerda, baixo e direita, respetivamente. A tecla ESC permite fechar imediatamente o jogo em qualquer ecrã.

As teclas são reconhecidas através dos seus makecodes, e o seu efeito no jogo está constantemente ativo até que a mesma seja solta, ou seja, quando o seu breakcode é reconhecido. Os makecodes e breakcodes das teclas utilizadas estão inseridos num switch case que

vai fazer o reconhecimento e permitir que o efeito desejado aconteça no ecrã.

Assim como o rato, o teclado tem um ficheiro com as funções que lhe dizem respeito, KBC_keyboard.c e também faz uso do ficheiro keyboard_and_mouse.c para realizar operações com a KBC.

Timer

O timer serve para dar display ao tempo passado na sessão de jogo.

O timer é também usado na vertente de movimentação, fazendo com que a mesma aconteça numa determinada frequência, para o personagem e para os monstros. O personagem move-se a cada segundo, caso seja detetada uma tecla que defina a direção do movimento, e os monstros movem-se constantemente de 0,5 em 0,5 segundos, numa órbita de perseguição ao personagem.

Como o jogo usa uma frequência geral de sistema de 60 Hz, o movimento do personagem é permitido quando o counter, que vai sendo incrementado a cada interrupt do timer é 60 e o movimento do monstro é feito quando o counter é 30.

A implementação de todas as funções relacionadas ao timer está no ficheiro timer.c.

Graphics

O graphics é usado no design da parte visual do jogo. Como o Minix está em modo de vídeo, iniciado no setup do jogo, através da leitura de xpm’s específicos para essa matéria, o graphics preenche o frame buffer do jogo com o seu conteúdo e também é responsável pela existência dos sprites de personagem e monstros.

De forma a tornar o jogo mais fluido e eficiente, o grupo escolheu implementar a técnica de triple buffering. O primeiro buffer é o buffer principal, que contém a informação que vai ser mostrada, o segundo é o buffer que contém a informação acerca da posição do cursor do rato e, inerentemente, a sua atualização, e o terceiro é o buffer que contém a informação sobre o jogo, como o seu display e a posição atualizada do personagem, dos monstros, etc.

Portanto, nesta implementação, temos um buffer principal que é a base do que é mostrado no ecrã com dois buffers de apoio que lhe enviam as informações atualizadas sobre o que mostrar, através da função memcpy().

A implementação de todas as funções relacionadas ao graphics está no ficheiro graphics.c.

Funções VBE usadas:

- 01h get VBE mode information
- 02h set VBE mode

RTC

O RTC é utilizado na aba de “Highscores” para registar a data e a hora à qual um novo recorde de pontuação foi alcançado.

Desta forma, conseguimos acrescentar mais detalhe à informação sobre cada “new highscore” individualmente.

A implementação de todas as funções relacionadas ao RTC está no ficheiro rtc.c.

Organização do código/estrutura

KBC module - 5%

devices/common_keyboard_and_mouse/

- keyboard_and_mouse.h
- keyboard_and_mouse.c

Este módulo contém funções do KBC desenvolvidas no Lab3 (e posteriormente reutilizadas no Lab4) que puderam ser reutilizadas no nosso projeto. Estas funções permitem realizar operações de leitura e escrita no KBC.

Contribuidores:

- Pedro Vieira

Graphics module - 10%

- main.c

devices/graphics/

- graphics.h
- graphics.c

Este módulo contém funções desenvolvidas no Lab5 que também puderam ser reutilizadas com algumas modificações e acrescentos de forma a se adaptar ao nosso projeto. Estas funções permitem configurar e gerir os buffers que ditam o que é mostrado no ecrã.

Funções VBE usadas:
01h Get VBE Mode Information
02h Set VBE Mode

Contribuidores:

- Pedro Vieira (fez o lab em si)
- Rodrigo Pombo (adaptou o lab para poder ser usado no projeto, fez a implementação no projeto)

Keyboard module - 7%

- main.c

devices/keyboard/

- KBC_keyboard.h
- KBC_keyboard.c

Este módulo contém funções desenvolvidas no Lab3 especificamente para o teclado, que puderam também ser reutilizadas. Estas funções permitem gerir as interrupções do teclado e, consequentemente, o seu funcionamento no contexto do jogo.

Contribuidores:

- Pedro Vieira (fez o lab em si, tratou da geração de interrupts no projeto)
- Rodrigo Pombo (ajudou a dar debug, fez o resto da implementação no projeto)

Mouse module - 7%

- main.c

devices/mouse/

- KBC_mouse.h
- KBC_mouse.c

Este módulo contém funções desenvolvidas no Lab4 especificamente para o rato, que puderam também ser reutilizadas, apenas com uma alteração na máquina de estados, novamente com o intuito de adaptação. Estas funções permitem gerir as interrupções do rato e, da mesma forma, o seu funcionamento no contexto do jogo.

Contribuidores:

- Pedro Vieira (fez o lab em si, tratou da geração de interrupts no projeto)
- Rodrigo Pombo (ajudou a dar debug, fez o resto da implementação no projeto)

RTC module - 5%

devices/rtc/
- rtc.h
- rtc.c

game_state/
- game_state.h
- game_state.c

Este módulo foi feito totalmente do zero para este projeto com base nas informações presentes na documentação do Lab6. As funções presentes no seu ficheiro correspondente permitem realizar operações de escrita e leitura no RTC e registar um horário quando necessário, após atualizar os seus registos.

Contribuidores:
- Pedro Vieira (fez o lab em si)

Timer module - 5%

- main.c

devices/timer/
- timer.c

Este módulo contém funções desenvolvidas no Lab2 que também puderam ser reutilizadas. Estas funções permitem gerir as interrupções do timer e o seu funcionamento no contexto da gameplay.

Contribuidores:
- Pedro Vieira (fez o lab em si)
- Rodrigo Pombo (ajudou com debug, fez o resto da implementação no projeto)

Game state module - 25%

- main.c

game_state/
- game_state.h
- game_state.c

Este módulo interage com o main module e grande parte da lógica do jogo. Consiste no fundo de funções que juntamente com as funções de main modificam e no fundo “fazem avançar” o estado do jogo, em geral recebem o estado do jogo e os interrupts gerados como argumento e modificam o estado consoante. Isto inclui também uma grande parte da lógica de onde pôr os sprites.

Contribuidores:

- Rodrigo Pombo (fez o módulo em si)

Utils module - 1%

utils/

- utils.c

Este módulo contém funções desenvolvidas no Lab2 que também puderam ser reutilizadas. Estas funções são utilizadas nos outros módulos, como por exemplo na conversão para 1 byte de um resultado de uma operação sys_inb().

Contribuidores:

- Pedro Vieira

Main module - 22%

- main.c

Este módulo é o cérebro de todo o projeto, no qual foi implementado um algoritmo que harmoniza os outros módulos que, em conjunto, fazem tudo acontecer, trata de coordenar os diferentes menus de receber os interrupts e de os enviar para o game state module para serem tratados em conjunto com ele. É também guardado aqui todo o estado do programa.

Contribuidores:

- Rodrigo Pombo (fez o módulo em si, ajudou com o debug a integrar os dispositivos, integrou o dispositivo da placa gráfica)
- Pedro Vieira (integrou os outros dispositivos (subscrição de interrupts, pôr no modo correto))
- Liess (escreveu o array do estado inicial do jogo e ajudou com as coordenadas)

Sprites module - 13%

- main.c

xpm_files/

- LCOM_0.xpm
- LCOM_1.xpm
- LCOM_2.xpm
- LCOM_3.xpm
- LCOM_4.xpm
- LCOM_5.xpm
- LCOM_6.xpm
- LCOM_7.xpm
- LCOM_8.xpm
- LCOM_9.xpm
- LCOM_-.xpm
- LCOM_2_pontos.xpm
- LCOM_character.xpm
- LCOM_cursor.xpm
- LCOM_game_over.xpm
- LCOM_high_scores.xpm
- LCOM_mapa.xpm
- LCOM_menu.xpm
- LCOM_monster.xpm

Este módulo é responsável pela parte visual do jogo, tanto os menus e os backgrounds, como os characters e até as fonts utilizadas.

Contribuidores:

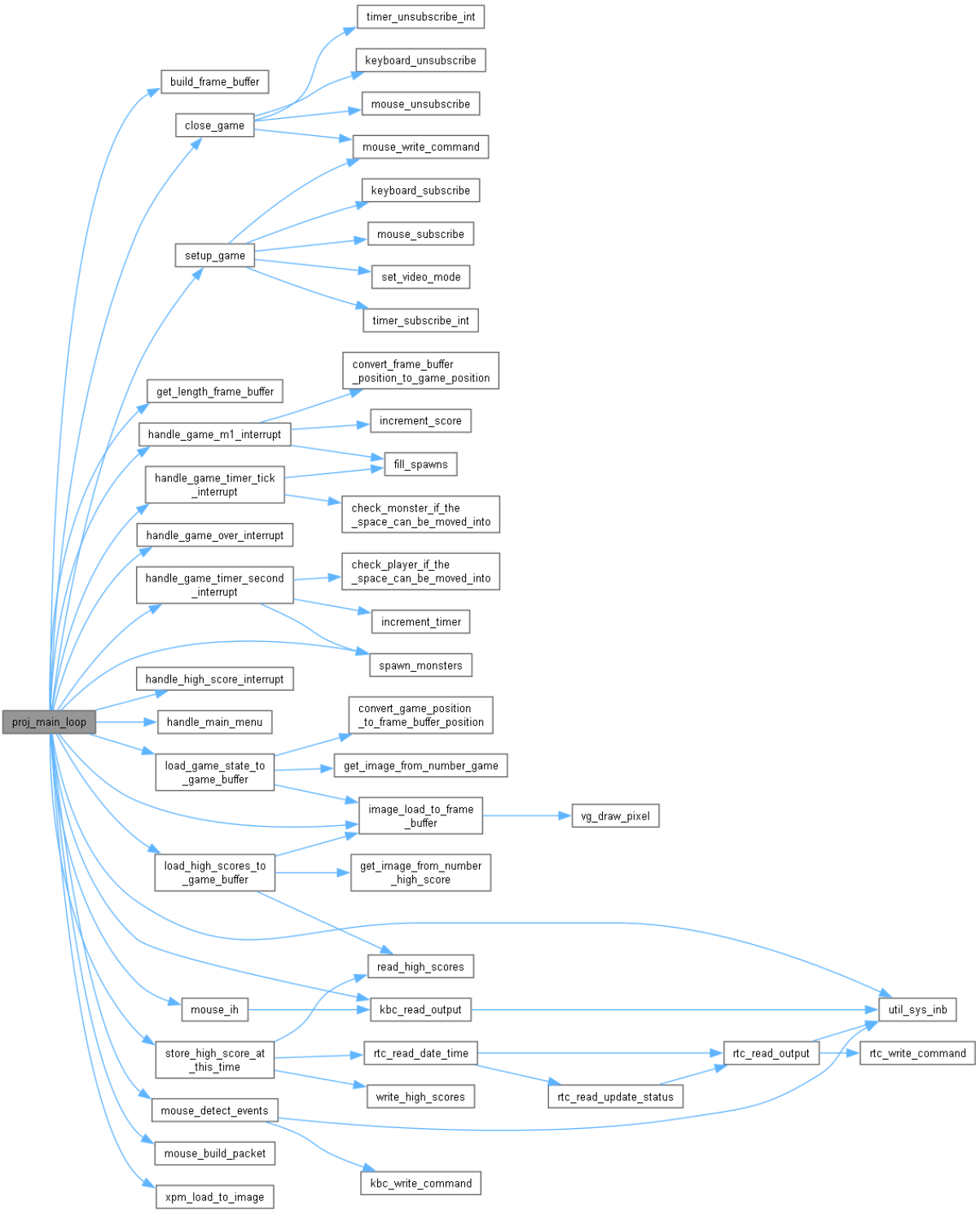
- Liess Aouimeur (tratou da totalidade dos sprites, design dos mapas e menus e fonts utilizadas)
- Rodrigo Pombo (fez mockups para o design geral da aplicação, tratou de fazer a aplicação usar os sprites criados)

Detalhes de implementação

Decidimos utilizar um método que não tinha sido explorado durante as aulas a fim de guardar os high scores e apresentá-los no menu de High Scores, guardamos todas as informações atuais, com a função store_high_score_at_this_time(), num ficheiro à parte que podemos depois acessar para construir a lista dos high scores, utilizando as funções write_high_scores() e read_high_scores().

Um grande problema de implementação com o qual nos deparamos foi o lag causada pela grande quantidade de printf que eram guardados no output.txt e embora óbvio, demorámos mais tempo do que gostávamos de admitir a descobrir isso.

Function call graph



Conclusões

Devido a dificuldades de implementação e falta de tempo para lidar com as mesmas a serial port não foi implementada, no entanto o resto dos dispositivos e funcionalidades foram implementadas com sucesso.

Se o trabalho no jogo fosse continuado, gostaríamos de adicionar a função de jogar em modo de multijogador que tinha sido proposta inicialmente. Outras adições possíveis seriam o uso de sprites com animações de movimento e até mesmo quando o personagem se encontra parado, a mudança de cor de fundo e/ou cenário dependendo da escolha do jogador ou do momento do dia usando RTC, e a utilização de um BFS para a movimentação dos monstros.

O trabalho foi em grande parte bem sucedido, ficamos então satisfeitos com o funcionamento do jogo em geral.

A realização deste projeto mostrou-se útil para a interiorização e aplicação de conceitos aprendidos durante as aulas mas também, de certa forma, para a compreensão de como funciona a criação de um jogo, não só na escrita do código, como também na parte visual e de criação dos mapas e menus.

Índice de figuras

- Figura 1 - Menu inicial……4
- Figura 2 - Gameplay do jogo……5
- Figura 3 - Game Over……6
- Figura 4 - Menu de High Scores……...7
- Figura 5 - Definição de path do ficheiro de texto do High Score….8

Créditos

- Character do jogador: Penzilla (https://penzilla.itch.io/hooded-protagonist)
- Character do monstro: Craftpix (https://free-game-assets.itch.io/free-chaos-monsters-3232-icon-pack)
- Versão base do ponteiro do rato: Kab Games (https://kaboff.itch.io/160-cursors-crosshairs-pack-32x32)
- Tileset usado para construir o jogo e menus: David G (https://incolgames.itch.io/dungeon-platformer-tile-set-pixel-art)

Demonstração

- Vídeo da demonstração do projeto: https://www.youtube.com/watch?v=f7aD94bUty0