U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

BOMBERMAN

Final Project

LCOM 2021/2022

T01G03
Alexandre Correia - up202007042
Duarte Lopes - up202006408
Tiago Branquinho - up202005567
João Sousa - up201904739

Table of Contents

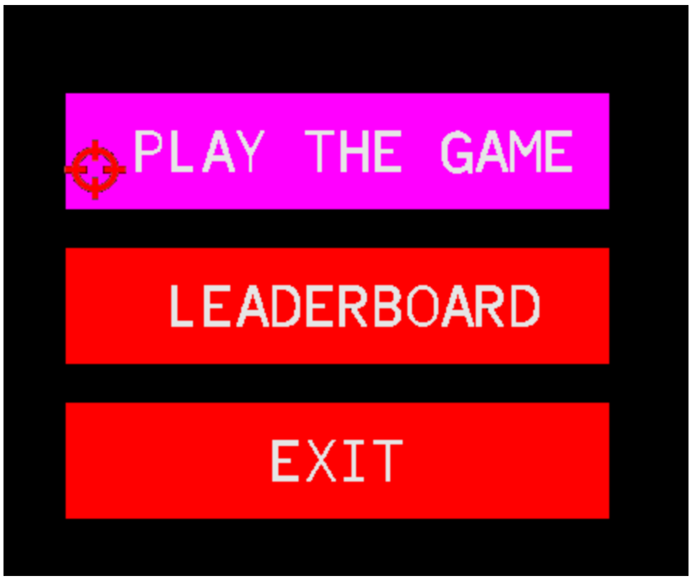
1 User Instructions

1.1 How to play

Throughout this project the controls are:

Menu:

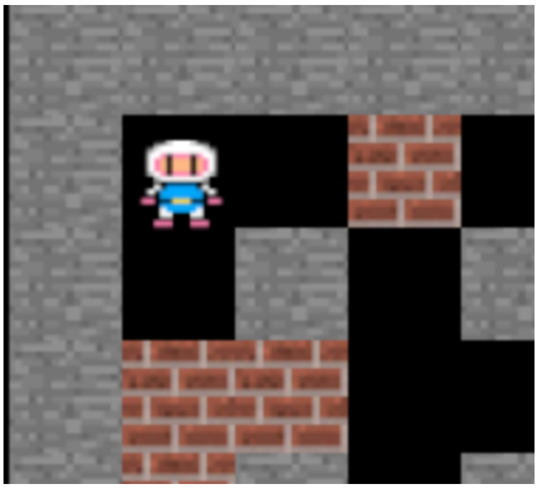
- **Mouse left-click** to select and navigate through the different buttons.



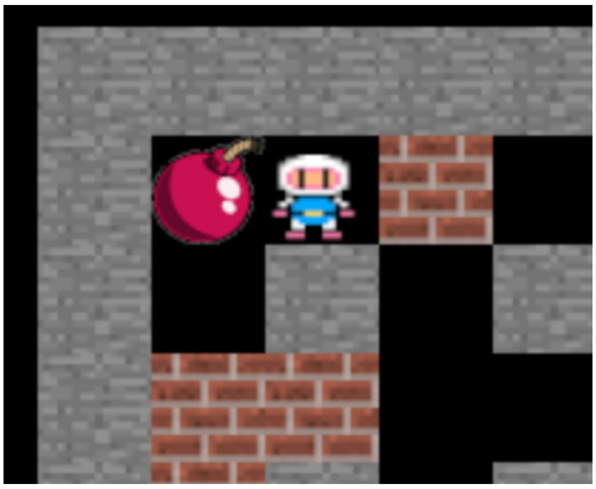
- **ESC** can also be used to exit the program.

Game:

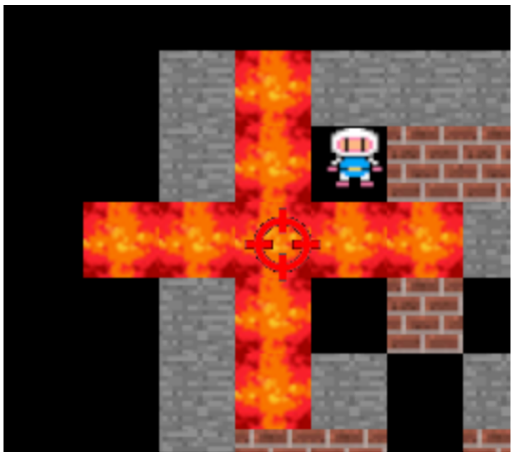
- **WASD** or **ARROW KEYS** to move the player.



- **SPACEBAR** to plant a bomb.



- Click a planted bomb using **Mouse left-click** to detonate it.



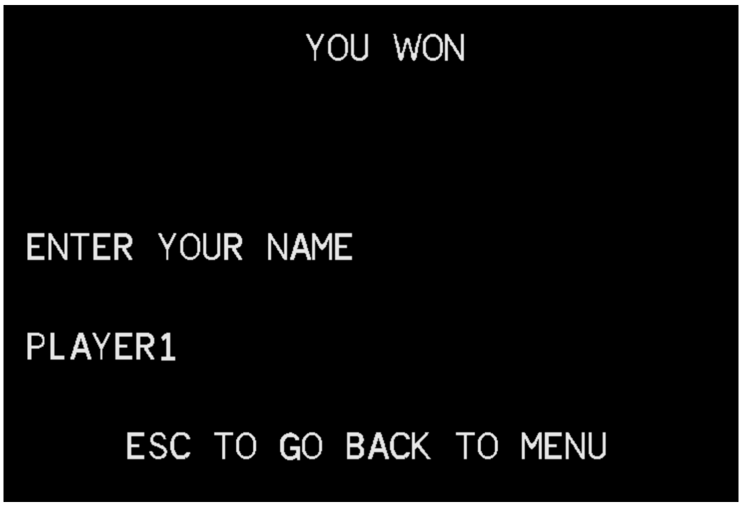
- **ESC** to quit the game and return to the menu.

Leaderboard:

- **ESC** to return to the main menu.

Game Ended (When the player wins or loses the game)**:**

- **KEYBOARD INPUT** to write your name. (When the player wins)



YOU WON

ENTER YOUR NAME

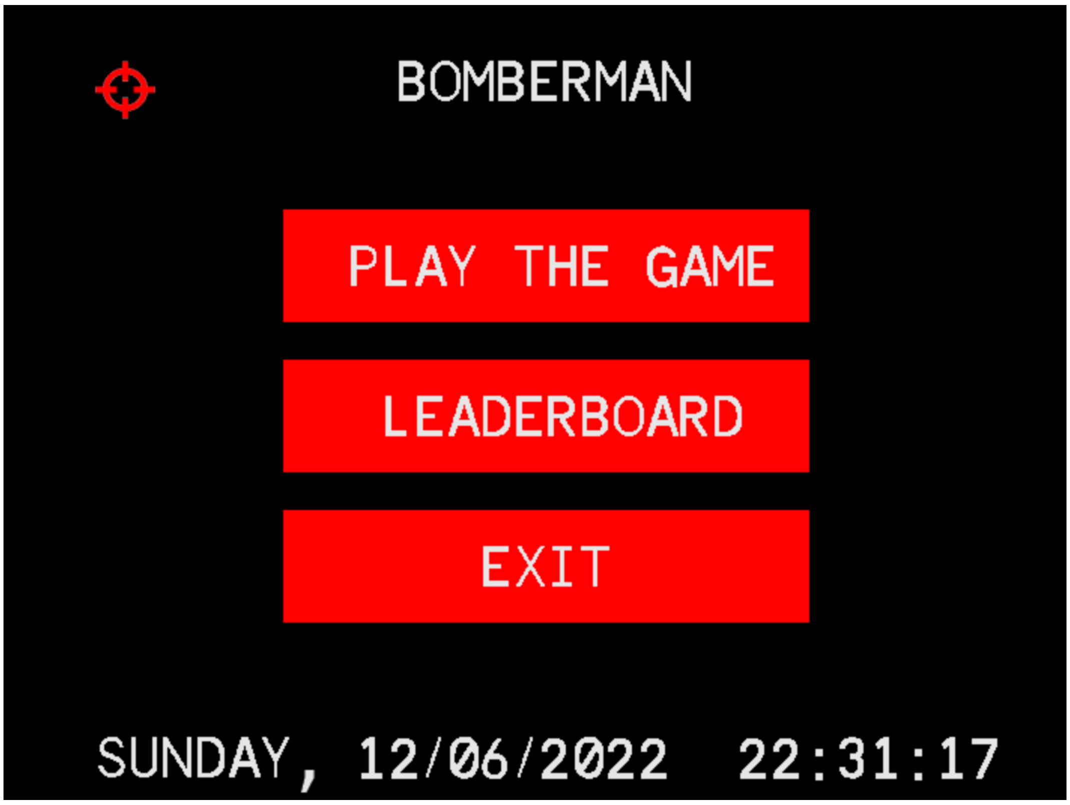
PLAYER1

ESC TO GO BACK TO MENU

- **ENTER** to register your score and return to the main menu. (When the player wins)
- **ESC** to not register your score and return to the main menu.

1.2 Main Menu

Using the mouse movement and clicks the user can select one of the several options available:

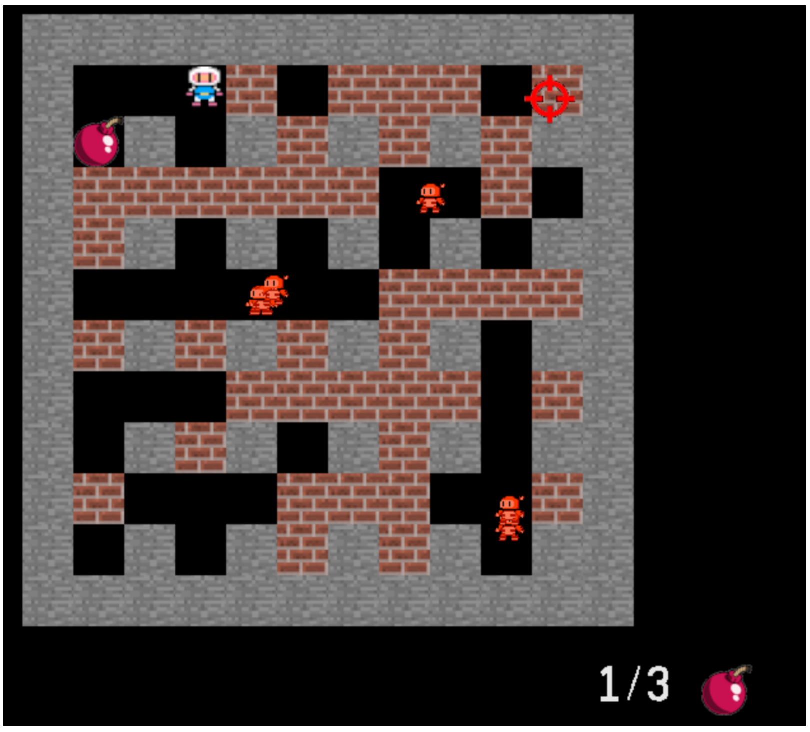


- **Play**
  - Jump into the arena, right into action. Good luck to survive!
- **Leaderboard**
  - Leaderboard with the best registered players' names, times, and when they were achieved.
- **Exit**
  - Exits the program.
  - Does the same as pressing **ESC**.

In addition to that, the real weekday, date and time are also displayed in the corner, as a bonus.

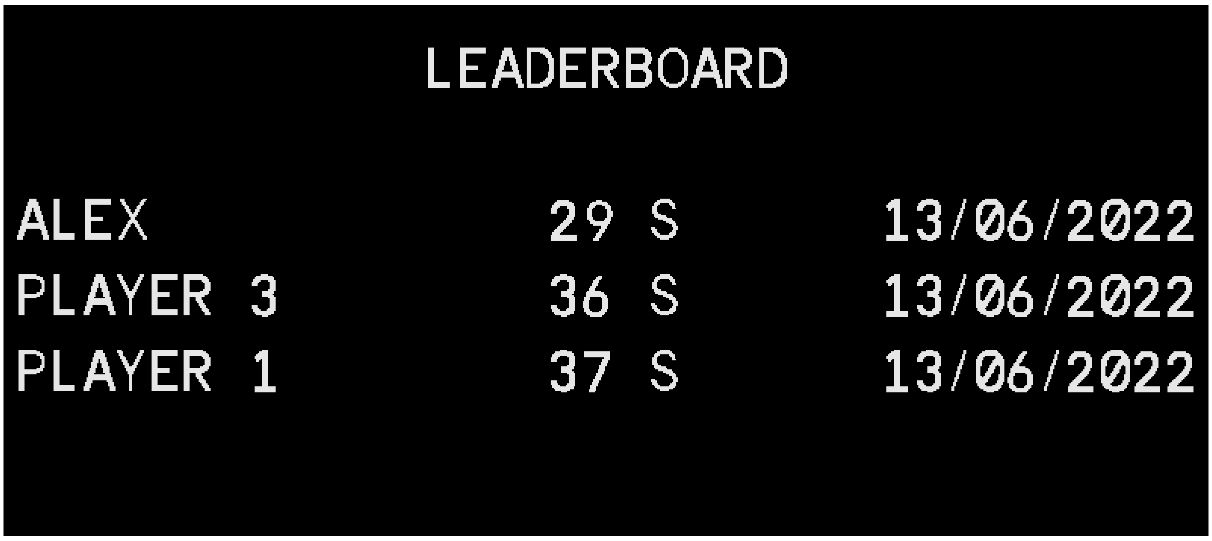
1.3 Playing the Game

The goal of the game is to find a gate to exit the arena without being caught by any zombie and without blowing yourself up. The arena walls and zombies are randomly placed, and there is a limited number of bombs you can place to break walls and kill enemies. The zombies will always be moving, so be careful!



1.4 Leaderboard

Anyone can check the top players, their times and the date they were achieved. It is a good addition to the project, since it promotes competition. Plus, it was not hard to develop, since we already had the menu, font, rtc, and general designs implemented. The player writes his name in real time, being able to delete letters if needed, and the times displayed are ordered.



LEADERBOARD

ALEX	29 S	13/06/2022
PLAYER 3	36 S	13/06/2022
PLAYER 1	37 S	13/06/2022

1.5 Game Ended

Depending on the result of the game, the player can choose either to insert (or not) their name on the leaderboard, or to try again. Returning to the main menu is always an option, pictured before.

2 Project Status

DEVICE	FUNCTIONALITY	INTERRUPTS
Timer	Frame Rate	Yes
Video Card	Showing Game Interface	No
Keyboard	Player movement	Yes
Mouse	Detonating Bombs, Selecting buttons	Yes
RTC	Current Date and Time, Leaderboard	No

2.1 Timer

Timer0 interrupts are used to regulate the screen refreshing rate which happens 60 times per second (60 fps). During the program these interrupts serve to process and update all the game information such as: collisions, movement, writing to the screen, among others. To handle these interrupts functions such as **timer_subscribe_int**, **timer_unsubscribe_int, timer_int_handler, timer_get_no_interrupts** and **timer_reset_no_interrupts** were used.

2.2 Video Card

In this project we used video mode 0x115, which has a resolution of 800x600 with colors being defined as “R:G:B” (8:8:8) bits.

The technique of double buffering was also implemented. Basically we used two buffers, **video_mem** and **video_buf**. When we draw something, we do it in **video_buf**, and then, to display the information, when the timer0 raises an interrupt (60 per second), we copy the information from **video_buf** to **video_mem**, and display this last one. To clear the screen we simply clear the memory from the **video_buf** and repeat the above process.

All images displayed consist of XPMs obtained from the Internet. They were created using the *Gimp* image editor, converting PNGs.

2.3 Keyboard

The keyboard generates an interrupt upon a key pressed or released. The keyboard is used in the different menus where *ESC* can be used to go back or to quit the game. However, it's main function is to move the player in game and to place bombs. It is also used to read the user input when asked for the name when the game ends. To handle these interrupts functions such as **kbd_subscribe_int**, **kbd_unsubscribe_int, kbc_ih, is_two_byte, kbd_error_code, keyboard_get_key, keyboard_check_esc, kbd_get_size_bb** and **update_keys** were used.

2.4 Mouse

The mouse generates an interrupt when a button is pressed or it has movement. The mouse is mainly used in the different menus by hovering and clicking the diferent buttons and it is also used to click on the bombs during the game to make them explode. To handle these interrupts functions such as **mouse_subscribe_int**, **mouse_unsubscribe_int, mouse_ih, mouse_enable_data_reporting, get_ih_counter, mouse_parse_packet** and **update_mouse** were used.

2.5 Real-Time Clock

In order to read the real time, date, and weekday, we don't make use of interrupts in this device. We simply read the RTC registers whenever we display the information, and update them every second. That is done by the following functions: **rtc_update_real_time**, **rtc_get_real_time**, and **font_draw_string** (which isn't related to the RTC, but draws the data string).

3 Code Organization/Structure

3.1 Graph Module (22%)

3.1.1 Font (2%) - Handles functions to load glyphs and draw text on the screen. Made by Tiago.

3.1.2 Menu (5%) - Represents the interactive menu itself. Made by Tiago

3.1.3 Rectangle (2%) - Represents the buttons present in the menu. Made by Tiago

3.1.4 Sprite (4%) - Handles functions to move, draw and handle sprites in the game. Most part was done by Alexandre, and Tiago had some contributions

3.1.5 Video Card (9%) - Functions present in lab 5 to initialize and operate minix in Graphics Mode. Alexandre, Duarte and Tiago contributed to this module, emphasizing functions to draw xpms that were made by Alexandre and Duarte, which was responsible for the double buffering.

3.2 KBC Module (17%)

3.2.1 Keyboard (5%) - Functions developed in Lab3 to operate and handle interrupts from keyboard. All members contributed.

3.2.2 Keys (4%) - Functions to handle key presses. Tiago took care of this part.

3.2.3 Mouse (8%) - Functions developed in Lab4 to operate and handle interrupts from mouse. Tiago was responsible for implementing this to the project, making some changes to improve performance.

3.3 RTC Module (8%)

Functions to handle the Real-Time Clock. Implemented by Duarte. Alexandre and Tiago contributed equally to display current date and time in the main menu using those functions.

3.4 Timer Module (14%)

Functions developed in Lab2. No additional work was needed to implement it in the project. All members contributed equally to this module.

3.5 Utils Module (2%)

Functions developed in Labs, especially to read from ports and get specific parts of data structures. All members contributed equally.

3.6 Media Module (9%)

Module containing all XPMs used throughout the program. All XPM files were made by João.

3.7 Project Module (28%)

3.7.1 Entities (9%) - Functions to handle eveything about game entites, such as Bomb, Player, Wall, Bot, Explosion, Map and Door. Alexandre and Duarte contributed to this section. Alexandre did everything, apart from the Bots and Bombs, which were made my Duarte.

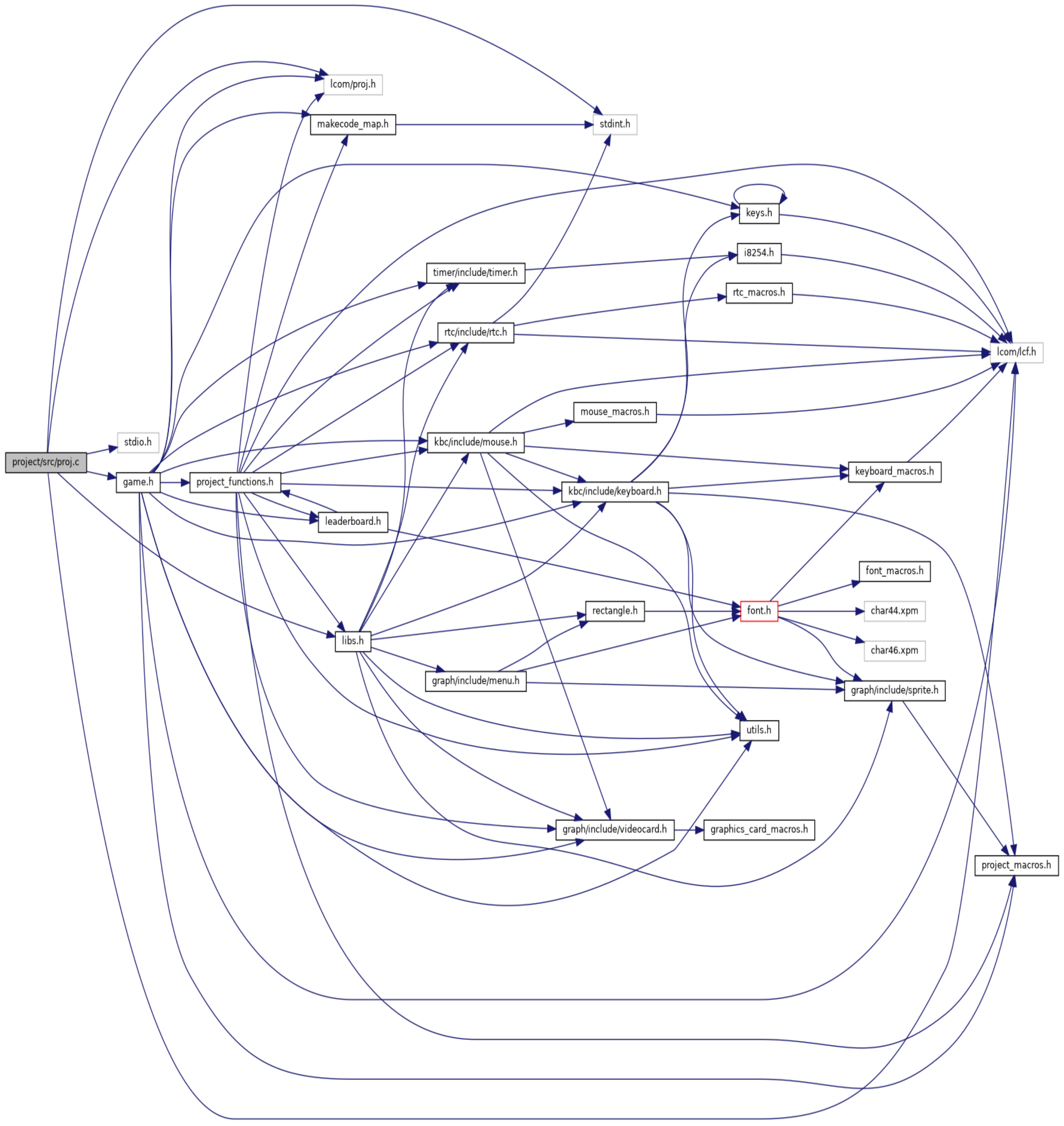
3.7.2 Game (13%) - Project's main loop, where interrupts, and everything about the game is handled. Alexandre, Duarte and Tiago contributed to this section equally.

3.7.3 Leaderboard (3%) - Contains leaderboard file, and functions to display and order it. Duarte took care of all parts of the section, apart from the time sorting, which was handled by Alexandre.

3.7.4 Makecode Map (1%) - Map that binds key codes to respective keys, used to write player name to screen, to then be added to the leaderboard. Made by Duarte.

3.7.5 Project Functions (2%) - Auxiliary functions used in entities structures. This section was made by Alexandre and Duarte, both working equally.

3.8 Function Call Graph



4 Implementation Details

We focused on object oriented programming to develop our game. It has a lot of benefits regarding code organization, readability and its development. The “classes” were achieved using struct pointers. RTC implementation was a bit tricky, since it was not covered in the labs. Code organization in different files as well as the Makefile were challenging as well.

5 Conclusions

We managed to overcome those difficulties and managed to do almost everything we planned. For instance, we planned to make different levels, with different difficulties, but we didn’t have time to do so. If we had more time to develop this game, we would have added that. Furthermore we would have made the indestructible walls stop explosions as well.

On the other hand we added more features that weren’t planned, like the bomb limit and the indestructible walls.

So, given the time we had and the amount of work from other subjects, we see this project as a success.

Finally, the Report was made by Duarte and Tiago, working equally and the doxygen documentation was handled by Alexandre. All labs were made by every member, equally.