Project Report

Videogame Homem Bala

FEUP – LCOM 2019/20

Pedro Pacheco (up201806824)
João Carlos Pires (up201806079)

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Index

Introduction 3

User Instructions 4

Project Status 14

Code Organization / Structure 18

Implementation Details 27

Conclusions 28

Hyperlinks 29

Introduction

This document is the report of the Final Project of the Curricular Unit Laboratório de Computadores (LCOM – Computer Laboratory), of the academic year 2019/2020.

We've developed in C programming language a videogame of the iconic circus performance *Homem Bala* (Bullet Man).[1]

User Instructions

The game starts with a main menu, where the user is able to use both the mouse and the keyboard to select from three available options: Play, Instructions and Exit.

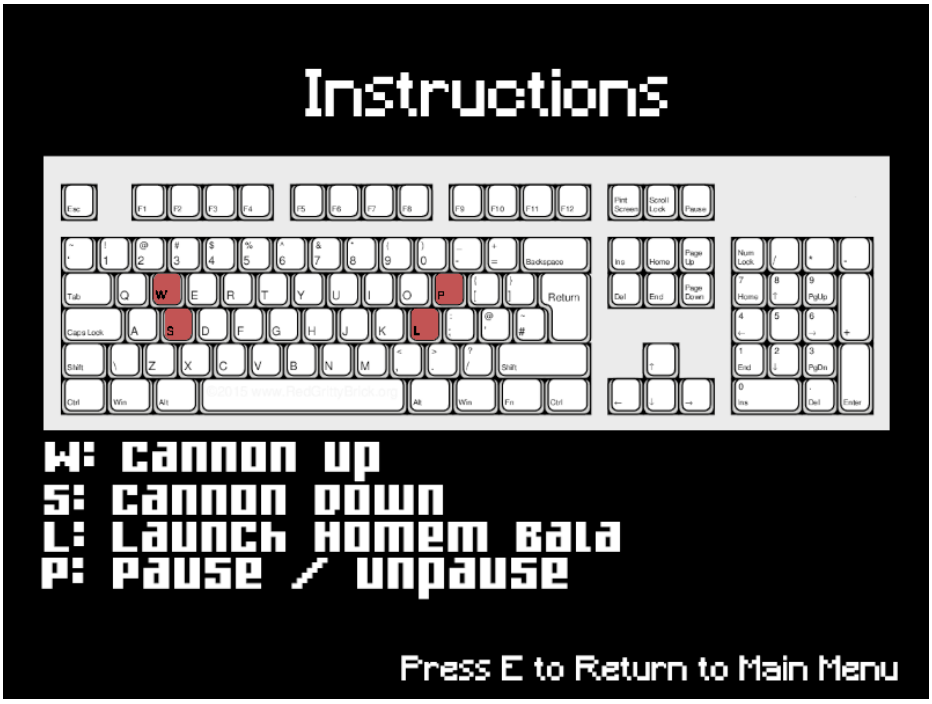
Each option can be accessed through mouse movement and left button click or through the key corresponding to the first letter of the option. For example, if we want to Play, we can press key ‘P’ (same for the other options).

Picture 1 – Main Menu

Instructions

If the user selects the option Instructions, it will lead to another screen, the Instructions Screen, in which the user can see the different keys that allow to control the cannon initial position ('W' and 'S'), to launch the *Homem Bala* (Bullet Man) ('L') and to pause the Game ('P'). To return to the previous menu, the Main Menu, the user can press 'E'.

Picture 2 – Main Menu, Instructions Option



Picture 3 – Instructions Screen

Play

Now, if the user selects the option Play, it will lead to another screen, the Game Screen, and the game starts. (Note that, as shown in the previous paragraphs, the option can be selected through key 'P' of through mouse movement and left click)

Picture 4 – Main Menu, Option Play

The initial Game Screen looks like the one in picture 5. On top, shows the current level, the time (which is a 10 second countdown) and the score, which will increase 50 points every time a level is completed. This can be seen in the picture 6, where the level changed to 2, the time left to complete the level is now just 7 seconds (note that each level has the same time to be completed) and the score is 050, which results of completing the first level. There are 5 levels.

It also shows the cannon and the target (which position is different in each level). Note that the *Homem Bala* (Bullet Man) is inside the cannon by now.

Picture 5 – Game Screen (Level 1, Time 00:10, Score 000)

Picture 6 – Game Screen (Level 2, Time 00:07, Score 050)

If the user presses the key ‘P’, the game will pause, showing the message “PAUSED” in the screen. Note for the fact that if the user types ‘P’ while the *Homem Bala* (Bullet Man) is flying, the game will just pause after the movement being completed. If the movement is now completed and succeeded, the game will Pause in the beginning of the next level. Otherwise, it will show not the pause message but the Game Over one, returning to Main Menu.

To back into the game, the user must type ‘P’ again to un-pause the game and continue playing.

Picture 7 – Paused message (In this case, the game was paused before the initial throw)

There's another message that appears only when it's Game Over. It's the message with that name and it appears three times:

1. When the time reaches 00:00 and no launch was made.
2. When the launch was made but the *Homem Bala* (Bullet Man) didn't hit the target.
3. When the user presses 'E' to leave the game back into Main Menu.

This message appears during two seconds and then goes back to Main Menu.

Picture 8 – Game Over due to time finish (00:00)

Picture 9 – Game Over due to launch error – the *Homem Bala* (Bullet Man) didn't hit the target

The movement of the *Homem Bala* (Bullet Man) is predefined for each cannon position. Only one cannon initial position makes the Man hits the target. And, as there are 5 cannon positions and 5 levels… Well, we can't tell how to complete the game!

The following pictures show the Man's movement in three different positions (orientations/directions).

Picture 10 – *Homem Bala* (Bullet Man) in Position 1

Picture 11 – *Homem Bala* (Bullet Man) in Position 2

Picture 12 – *Homem Bala* (Bullet Man) in Position 3

When the user wants to leave the game, must select the option Exit in the Main Menu, using the mouse and left click or pressing the key ‘E’.

Picture 13 – Main Menu, Option Exit

Project Status

In this project, we've used four I/O devices:

1. Timer
2. Keyboard
3. Mouse
4. VBE

I/O Device	Functionality	Interruptions
Timer	Create a countdown timer for each level.	Yes
Keyboard	Allow the user to navigate in the menus and to play the game.	Yes
Mouse	Allow the user to navigate in the Main Menu.	Yes
VBE	Draw Menus and all the other screens of the game.	No

Note: All the subscriptions and cancels of subscriptions, except for the I/O Device Mouse, were made in the file proj.c, as well as the initialization of video mode and, in the final, the return to the text mode.

1. Timer

The I/O Device Timer was only used in the function game_screen(), using Interruptions, and with the main objective

of create a countdown timer (10 seconds) for each level of the game.

We've used, to subscribe, unsubscribe as well as the Interrupt Handler, the functions already created in lab2 – timer_subscribe_int(), timer_unsubscribe_int() and timer_int_handler().

The time, handled by this I/O Device, was then printed in the screen by drawing a XPM, previously created for each time (00:10, 00:09, etc.), using the VBE (see the subsection 4. VBE).

2. Keyboard

The I/O Device Keyboard was used in four functions: home_screen(), instructions_menu(), game_screen() and paused().

By making use of Interruptions, the main goal of this I/O Device was to help the user to play the game, using the keys 'W' and 'S' to move the cannon up and down, respectively, as well as the key 'L' to launch and 'P' to pause the game; to select and option from the Main Menu ('P' for Play, 'I' for Instructions and 'E' for Exit. In this last case, the user can also press 'E' during the game to leave to Main Menu); and, finally, to return from the Instructions Screen to the Main Menu.

As in the Timer, we've used also functions created during lab3 – kbc_ih(), util_sys_inb() – as well as the instructions to subscribe/unsubscribe.

The user can access the Instructions to play the game, the keys, in the Instructions Menu, available from the Main Menu.

3. Mouse

The I/O Device Mouse, unlike the others used, is only used in one function – home_screen() – and the subscription of interruptions, as well as it's cancelation, is made inside the function. For this, we've used the functions created during lab4 – disable_mouse_ih(), enable_mouse_ih(), mouse_subscribe(), mouse_unsubscribe() and mouse_command_stream().

To handle with mouse packets, we've used the function mouse_ih() (also created during lab4) and an adaptation of function parse() in order to set only the state of the mouse's left button and it's position (coordinates x and y).

With the left button state and the mouse's position, we're able to create, through if conditions, the capacity for the user select options from the Main Menu using the mouse. In other words, for each option, that we already knew the corresponding positions, we've created an if statement that verifies if the left button is pressed and if it's "above" the option at the same time.

The mouse pointer was created using the XPM made for the effect and the I/O Device VBE (see next subsection).

4. VBE

This is probably the I/O Device we've used most in our project. We've optioned to use a set of XPM and the function draw_xpm() created in lab5, as well as two more functions – draw_xpm_transparent() and draw_xpm_partial() – that result of adaptations of the function already mentioned to draw XPMs ignoring the White colour and to draw a XPM without loading it, respectively. In this last case, the function is only used to draw the

XPMs of the Main Menu and the Game Screen. This functions call other two, one also created in lab5 and the other one the adaptation in order to draw only the non-White pixels, vg_draw_pixel() and vg_draw_pixel_transparent().

To initialize the Video Mode, we've used the function video_init() – created during lab5 – and the function vg_exit() to return to text mode if the user selected 'Exit' in the Main Menu. We've used the mode 0x115, which allowed to have a resolution of 800x600 (pixels) and a Direct Colour Mode (24 bits per pixel – 8 R, 8 G, 8 B), which means there were 16 777 216 colours available.

The VBE, specifically the functions to draw the XPM's and the other ones called inside them, is used in all the functions of screen.c, listed below:

- home_screen()
- game_screen()
- gameover_screen()
- draw_time()
- change_cannon()
- draw_man()
- draw_man_1()
- draw_man_2()
- draw_man_3()
- draw_man_4()
- draw_man_5()
- draw_level()
- draw_score()
- draw_target()
- instructions_menu()
- paused()

The main goal of the use of this I/O Device is to draw the Main Menu, the Instructions Menu, the Game Screen and all it's components (time, score, level, cannon, *Homem Bala* (Bullet

Man) and target). The description of the functions mentioned above is in the next chapter.

Note for the fact that the letters of the game were made in the website *Font Memes*[2] and the elements of the game were pixelized using the website *onlinepngtools*[3].

Code Organization / Structure

We've made some changes after the Project Specification. The number of modules is now the same, but the module 'Menu' changed its name to 'Screen' and it now includes some features we taught it would belong to 'VBE' module.

In terms of group member responsibility, for each module, we've made a lot more changes. But we'll list all the contributions and what have each member done in the next paragraphs.

We've followed the Development Plan as expected, making some small changes due to the progress and problems we've obtained during the project making.

The modules of our project are:

1) Timer

2) Keyboard

3) Mouse

4) VBE

5) Screen (old 'Menus')

We'll now describe each one.

1) Timer

In this module, we've used all the functions created during lab2, as mentioned in the previous chapter. The code in this module is the one that allows to subscribe and unsubscribe

interruptions, and to handle with the interruption. The relative weight in the overall project is more or less 10%. As we've only used functions from lab2, and then put them in a certain way to do what was supposed, the group member responsibility was of both of us. Initially was Pedro's responsibility, but we've worked together on lab2 and project.

2) Keyboard

In this module, we've used all the functions created during lab3, as mentioned in the previous chapter. The code in this module is the one that allows to subscribe and unsubscribe interruptions, and to handle with the interruption. The relative weight in the overall project is more or less 20%. As we've only used functions from lab3, and then put them in a certain way to do what was supposed, the group member responsibility was of both of us. Initially was João's responsibility, but we've worked together on lab3 and project.

3) Mouse

In this module, we've used some functions created during lab4, as mentioned in the previous chapter. The code in this module is the one that allows to subscribe and unsubscribe interruptions, and to handle with the interruption of the mouse, as well as read the packets and set the three variables already mentioned we needed. The relative weight in the overall project is more or less 10%. As we've used functions from lab4, and then put them in a certain way to do what was supposed, the group member responsibility was of both of us, as it was supposed

initially. And in fact, it was what happened. We've just made a slightly variation of the function parse(), whose responsibility was of João.

4) VBE

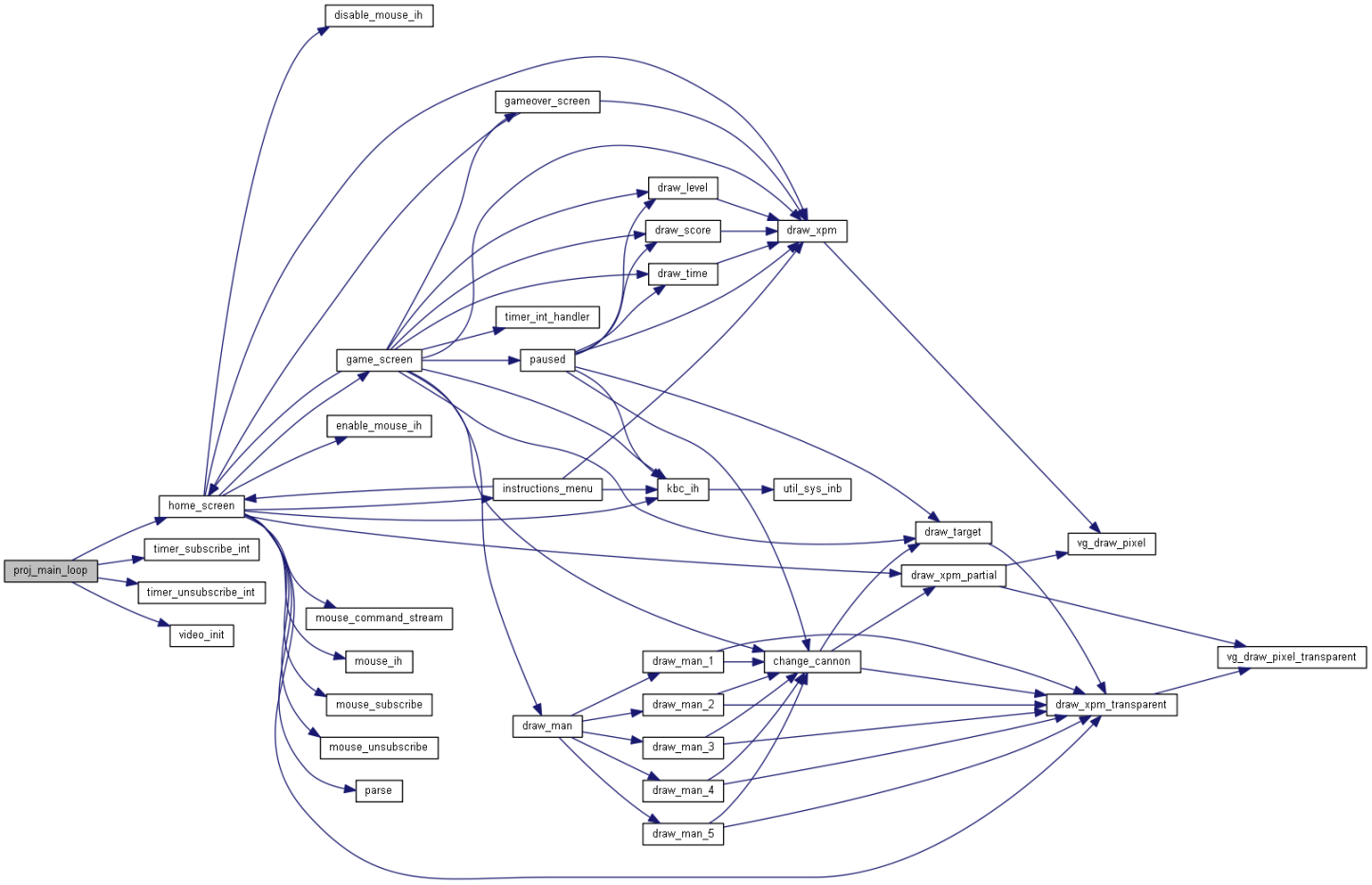
In this module, we've used the functions created during lab5, specially video_init() and vg_draw_pixel(), as mentioned in the previous chapter. The code in this module is the one that allows to enter in video mode and return to text mode by the end, as well as draw the pixels. The relative weight in the overall project is more or less 10%. As we've used functions from lab5, although we've created a new one based on vg_draw_pixel(), called vg_draw_pixel_transparent(), and then put them in a certain way to do what was supposed, the group member responsibility was of both of us. Initially was João's responsibility, but we've worked together on lab5 and project. The new function was the only one made by João, but, in fact, as already said, it was just a slightly variation of the one we already had.

5) Screen

In this module, we've used one function created during lab5, draw_xpm(), as mentioned in the previous chapter. The code in this module is the one that allows to draw all the elements of our game. The relative weight in the overall project is more or less 50%. Although the function draw_xpm(), we've created two new variations of this one, draw_xpm_tranparent() and draw_xpm_partial(). We've also created some other functions to draw the Main Menu, the Game Screen and many

others, which will be described in the next paragraphs, and that used, mostly, the draw_xpm() and it's variation functions. All the XPM were pictures took from the Internet or created using *Font Meme*[2] and then we're pixelized using *onlinepngtools*[3]. Some of them needed to be treated in *GIMP* – free software of image edition. Just two of the XPM were created from nothing: the XPM of the *Homem Bala* (Bullet Man) and the XPM of the Target. This one's were Pedro's responsibility, who draw them using a tablet and a pen. Although this module was initially Pedro's responsibility, we've worked together both on XPM and in functions to handle them.

Function Call Graphic



Picture 14 – Function Call Graphic of Project (automatically generated by Doxygen)

Note: Although we've generated this Function Call Graphic automatically from Doxygen, we've submitted all the Doxygen Documentation (.html files) in the SVN repository.

Next, we'll show the list of functions used, including both the ones created during labs and the new ones. We'll also make a small description of what function does, but only for the ones created specifically for this project.

proj.c

- proj_main_loop() – Calls the functions to subscribe and unsubscribe the I/O Devices Timer and Keyboard; initializes video mode and calls vg_exit() by the end to return to text mode; calls the function home_screen().

vbe.c / vbe.h

- video_init()
- vg_draw_pixel()
- vg_draw_pixel_transparent() – Similar to vg_draw_pixel(), except that it doesn't draw the white pixels.

timer.c / timer.h

- timer_subscribe_int()
- timer_unsubscribe_int()
- timer_int_handler()

kbd.c / kbd.h

- kbc_ih()
- util_sys_inb()

mouse.c / mouse.h

- mouse_command_stream()
- mouse_ih()
- mouse_subscribe()
- mouse_unsubscribe()
- disable_mouse_ih()
- enable_mouse_ih()
- parse() – This function is an adaptation of the parse() function of lab5. In this case, we just keep the state of the mouse's left button and of its position (x and y coordinates). These values are set in three global variables we'll use to draw the mouse pointer and to check if any option from the Main Menu is being selected.

screen.c / screen.h

- draw_xpm() – Includes also the set of six global variables, three for Main Menu and other three for Game Screen, that allow draw_xpm_partial() to draw this XPMs without loading them.
- draw_xpm_transparent() – Variation of draw_xpm(), calling vg_draw_pixel_transparent(), which means it ignores the white pixels of the XPM.
- draw_xpm_partial() – Variation of draw_xpm(), drawing a XPM without loading it. It's used only to draw Main Menu and Game Screen, which were previously loaded into global variables the first time they're draw.

- home_screen() – Function that includes an interruption cycle for Mouse and Keyboard and that calls function game_screen() if the user selects the option 'Play', calls the function instructions_menu() if the user selects the option 'Instructions' or exit the game, returning 1, if the user selects the option 'Exit'. It also draws the XPM of the Main Menu and the Mouse Pointer.
- game_screen() – Draws all the game elements, calling the functions responsible for time, score, level, *Homem Bala* (Bullet Man), target and cannon, as well as the background of the game. Includes and interruption cycle for Keyboard and Timer.
- gameover_screen() – Draws the XPM of Game Over during 2 seconds and then calls the home_screen() function.
- draw_time() – For each time passed as argument, draws the corresponding XPM.
- change_cannon() – For each cannon initial position passed as an integer argument, draws the corresponding XPM.
- draw_man() – Calls the draw_man_int() function, where int is the number passed as argument and corresponds to the *Homem Bala* (Bullet Man) movement for each position of the cannon.
- draw_man_1(), draw_man_2(), draw_man_3(), draw_man_4() and draw_man_5() – Each of this functions has, at least, one for cycle, that draws the XPM of the *Homem Bala* (Bullet Man) in the successive positions corresponding to its movement.

- draw_target() – Draws the target in the position corresponding to the one passed as argument. There's one position for each level.
- instructions_menu() – Draws the XPM of the Instructions Menu and includes an interruption cycle that only ends when the user presses 'E' to return to Main Menu.
- paused() – Draws the XPM of Paused and wait until the user types 'P' again to un-pause, which is handle in an interruption cycle. By the end, draws back the Game Screen with all the elements as they were right before the pause.
- draw_score() – Draws the XPM of the score corresponding to the one passed as argument.
- draw_level() – Draws the XPM of the level corresponding to the one passed as argument.

Implementation Details

During this project, we've used mostly functions created during labs. The only thing we've created from scratch were the game elements, like the XPM. We've also created other new functions but using the programming concepts we already had by learning not just in LCOM as in other Curricular Units. We've just recurred to Internet to solve some compilation problems or errors that appeared during the realization of this project as well as learn some C programming useful for our code.

We had a special taste for interruptions, so we've used for Timer, Keyboard and Mouse this method to receive and send information. We've also choose XPM because we're able to quickly create them using GIMP and them loading them into the project recurring to draw_xpm() function and the other variations of this one.

We don't have any function to detect collisions, because we've defined an initial and only acceptable position for each level and target position, which means the program already know if the launch will succeed or not when the *Homem Bala* (Bullet Man) is throwed.

The movement is created by drawing inside a for cycle the element, then all that's in the current screen and then again the element in the new position, repeating this until the end of the movement. We've used this approach both on mouse pointer and on *Homem Bala* (Bullet Man) movement.

Conclusions

During this project, we've worked both to make it happen and to reach the Game we had in our minds since the very beginning. For us, it's 95% complete, since we've not implemented Double Buffering, which made our game a little bit scintillating.

We've enjoyed the project, specially because it was challenging but also fun!

In terms of contribution, as already said, we've worked both on all (code, doxygen, report and demo video), helping each other to fix some problems.

Hyperlinks

[1] - https://www.youtube.com/watch?v=unQtXYfzxO4

[2] - https://fontmeme.com/pixel-fonts/

[3] - https://onlinepngtools.com/pixelate-png