

Faculdade de Engenharia da Universidade do Porto



Universidade do Porto

FEUP Faculdade de Engenharia

Jumping Frog

Laboratório de Computadores

Licenciatura Engenharia Informática e Computação

Pedro Souto
Pedro Brandão
Nuno Paulino

Estudantes e Autores

Grupo T08G04

Ana Carolina Brandão, up202004461@edu.fc.up.pt

André Soares, up202004161@edu.fe.up.pt

Vinícius Macedo, up202001417@up.pt

Maio 2022
Porto, Portugal

Index:

User Instructions	4
About	4
How to use our project	4
Menu	4
Help	5
Gameplay	5
Pause	6
Project Status	8
Timer	8
Code References	8
Keyboard	8
Code References	9
Mouse	9
Code References	9
RTC (Real Time Clock)	9
Code References	9
Graphic Card	9
Code organization	10
draws.c	10
entity.c	10
frog.c	10
game.c	10
gameController.c	11
graphics_card.c	11
keyboard.c	11
log.c	11
mouse.c	11
proj.c	12
rtc.c	12
timer.c	12
turtle.c	12
utils.c	13
vehicle.c	13
Function Call Graph	13
Implementation details	14
Topics covered in the lectures:	14
Topics not covered in the lectures:	14
Conclusions	14
Problems	14
Like to have	14

User Instructions

About

Our game is based on a really popular game, among the 80's culture, called "Frogger", in which basically there is a frog trying to cross obstacles without dying.

How to use our project

To access our project folder, go to: "\$ cd labs/proj/src". After inside the folder, you'll need to write 'make' to compile our game. Now to run the game, just write "lcom_run proj" and it will work.

Menu



After doing everything correctly, this screen should appear to the player. This is our menu to play the game.

Use your mouse to select your desired option on top of each button in the screen. In case of exit, the user can also use "Esc" to leave as well.

Help

Pressing the "Help" option, you should expect to see the instructions and game explanation.

Jumping Frog


Use the 4 arrow keys to move up,down,left and right

Each time you cross the map and level up, the game will become faster, but you won't

You may stay on top of logs and turtles but be careful, the latter like to take a dive from time to time and lets just say that you unfortunately can't breath underwater. Also unless you wanna be a splashed frog I would avoid the vehicles

If you need to take a break all you have to do is press the esc key where you will be taken to the pause menu from where you can continue or go back to the menu (you can use both the arrow up and arrow down keys or the mouse to navigate there) and should you choose the latter and then play again later, the game will restart

You only have 5 lives and should you spend them all, you will go back to the menu



Press the esc key to go back to the menu

Gameplay

Pressing the "Play" option, you should expect to see the game running.



On the bottom of the screen we can see the frog's lives, which can be lost after being trapped by some of the objects involved in the game.

On the right side of the frog's lives, we have the real time of the real world. In case you spend too much time playing our game and lose sense of time!

The player can also confront multiple levels of our game increasing the speed of objects, logically increasing the difficulty!

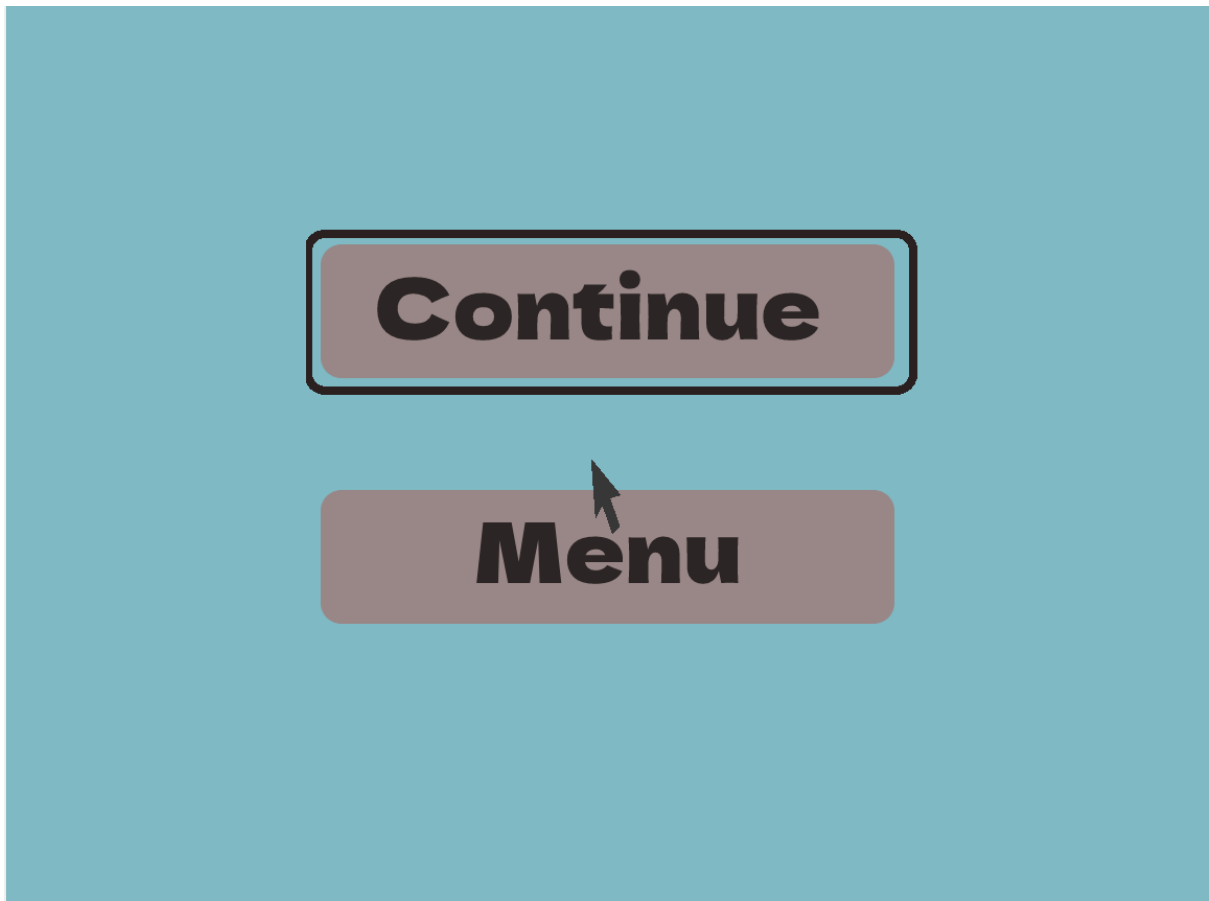
To pass to another level, the player should cross the river to the other side, while still counting the number of lives of past levels.

To move the frog, use arrows on the keyboard, such as: "<-, ->, etc.."

In case the frog lost all of his lives, it will automatically come back to the menu page.

Pause

To use the pause page, press "Esc", and it should work.



To select options, use the keyboard's arrows and press Enter to your desired option or the mouse cursor to select any desired button.

Project Status

Device	What for	Interrupt/polling
Timer	Control of the frame rate	I
KBD	Frog moves, pause selection and pause button.	I
Mouse	Menu selection	I
RTC	Game environment also displays a real time clock.	I
Graphics Card	Drawing of the entire interface and gameplay.	N/A

Timer

Refresh menu and game with a constant frame rate, and such as performing different frame rates to specific entities, like the cars, turtles, to change the animation. Those actions can be affected during some events in the game.

File "timer.c" created during lab2, which has some functions to subscribe and unsubscribe timer, and interrupt handler.

Code References

- gameLoop() - defined in ("game.c")
- deviceHandler() - defined in ("gameController.c")

Keyboard

Register user's input in game, for controlling the frog and selecting buttons in the pause section.

File "keyboard.c" contains code developed in lab3, with methods to subscribe and unsubscribe keyboard, as well the interrupt handler.

Code References

- gameLoop() - defined in ("game.c")
- deviceHandler() - defined in ("gameController.c")

Mouse

Used for the main menu and pause menu, where the user can choose the options available. We read the user's input, both the button press and mouse movement.

In the file mouse.c there are some functions adapted to our game, such as resetting the mouse cursor to our initial display dimension.

Code References

- gameLoop() - defined in ("game.c")
- deviceHandler() - defined in ("gameController.c")

RTC (Real Time Clock)

Here it has some functionalities to display the real time clock, and alter the game environment, like day and night.

Code References

- gameLoop() - defined in ("game.c")
- deviceHandler() - defined in ("gameController.c")

Graphic Card

Used to draw pixels on the screen, xpm's, letters on the screen, at the desired position, and at the buffers.

- Video Mode: 0x14C, 1152x864 resolution and a direct color mode with 32 bits per pixel.
- Double buffer

- Moving objects: Entities in game like: log, vehicle, frog, turtle, that also have a collision system.
- "arial black" for the game, "impact" for the help section, letters in game were drawn with xpm, and in menu and help, with xpm ready images.
- VBE functions: 0x02, put the desired graphic mode.

Code organization

draws.c

Used to load xpm and do almost every drawing in our project. Draw frog's lives, pause option, background in game, do the transparency of turtle, given the impression is under water, draw menu and the cursor as well, also the rtc (real time clock) is drawn here.

Relative weight of module: 12%

Developed by: André Soares, Vinícius Macedo, Ana Brandão

entity.c

Here all entities are created. We have cases for: frog, log, turtle and vehicle. There is a position for each entity, which is created in this file.c, and after a movement, the update drawing is done with the functions declared here. As RTC indicates if it's night or not, it switches the turtle day, making it appear darker depending on the hour. Collisions and interactions between the frog and other entities also are declared.

Relative weight of module: 14%

Developed by: André Soares, Vinícius Macedo, Ana Brandão

frog.c

A struct to the entity frog, which initiates the frog in a default position and loads it xpm. Also, when the frog dies, it resets to its original position. Finally, it checks if the frog is out of boundaries of the screen, making it not surpassing the area beyond the resolution.

Relative weight of module: 5%

Developed by: André Soares, Vinícius Macedo

game.c

All I/O devices interruptions are subscribed inside this file and enables mouse stream mode. After an interruption is done, depending on the I/O device, it is handled and changes its state machine during each game state.

Relative weight of module: 5%

Developed by: André Soares, Vinícius Macedo, Ana Brandão

gameController.c

Entities initiations and game controller, how many frog's lives and the default level. State Machines to change state to menu, pause, if it's alive or not. Updates mouse buffer during menu selection, checking also, if it's inside the resolution screen. Each device is handled according to its state. According to each entity, they can also change their speed movement during the game, by some game events.

Relative weight of module: 15%

Developed by: André Soares, Vinícius Macedo, Ana Brandão

graphics_card.c

functions developed during lab5, which set the graphic mode, map the video memory, and get the information about the resolution. Draws rectangles, lines, and pixels according to the position given and color.

Relative weight of module: 10%

Developed by: André Soares, Vinícius Macedo, Ana Brandão

keyboard.c

functions developed during lab3, which have definitions to subscribe keyboard, unsubscribe and handle the interruption.

Relative weight of module: 5%

Developed by: André Soares

log.c

A struct to the entity log, which initiates the logs in a random position and loads it xpm. This random position checks if there is a space between each log, to not overlap the log. Also, each starting log has a default speed during each level.

Relative weight of module: 5%

Developed by: André Soares, Vinícius Macedo

mouse.c

functions developed during lab4, which have definitions to subscribe mouse, unsubscribe and handle the interruption. Also, there are more functions added to develop this project, which we can quote the initiation mouse default position, build and get mouse packet.

Relative weight of module: 5%

Developed by: André Soares

proj.c

given file to compile our project. Basically calls the main functions to set graphics mode, game loop, and put back minix to graphics text mode.

Relative weight of module: 3%

Developed by: André Soares

rtc.c

functions that subscribe rtc, unsubscribe and handle the interruption. Also, do the conversion of hexadecimal to integer for the time in game, and do the update of the time.

Relative weight of module: 5%

Developed by: André Soares

timer.c

functions developed during lab2, which have definitions to subscribe timer, unsubscribe and handle the interruption.

Relative weight of module: 5%

Developed by: André Soares, Ana Brandão

turtle.c

A struct to the entity turtle, which initiates the turtle in a random position and loads it xpm. This random position checks if there is a space between each turtle, and checks during the game if the turtle is underwater or not.

Relative weight of module: 5%

Developed by: André Soares, Vinícius Macedo

utils.c

functions developed during lab2, which have a function that does the conversion of 32-bit to 8-bit for reading ports. Also, read the MSB and LSB.

Relative weight of module: 1%

Developed by: André Soares

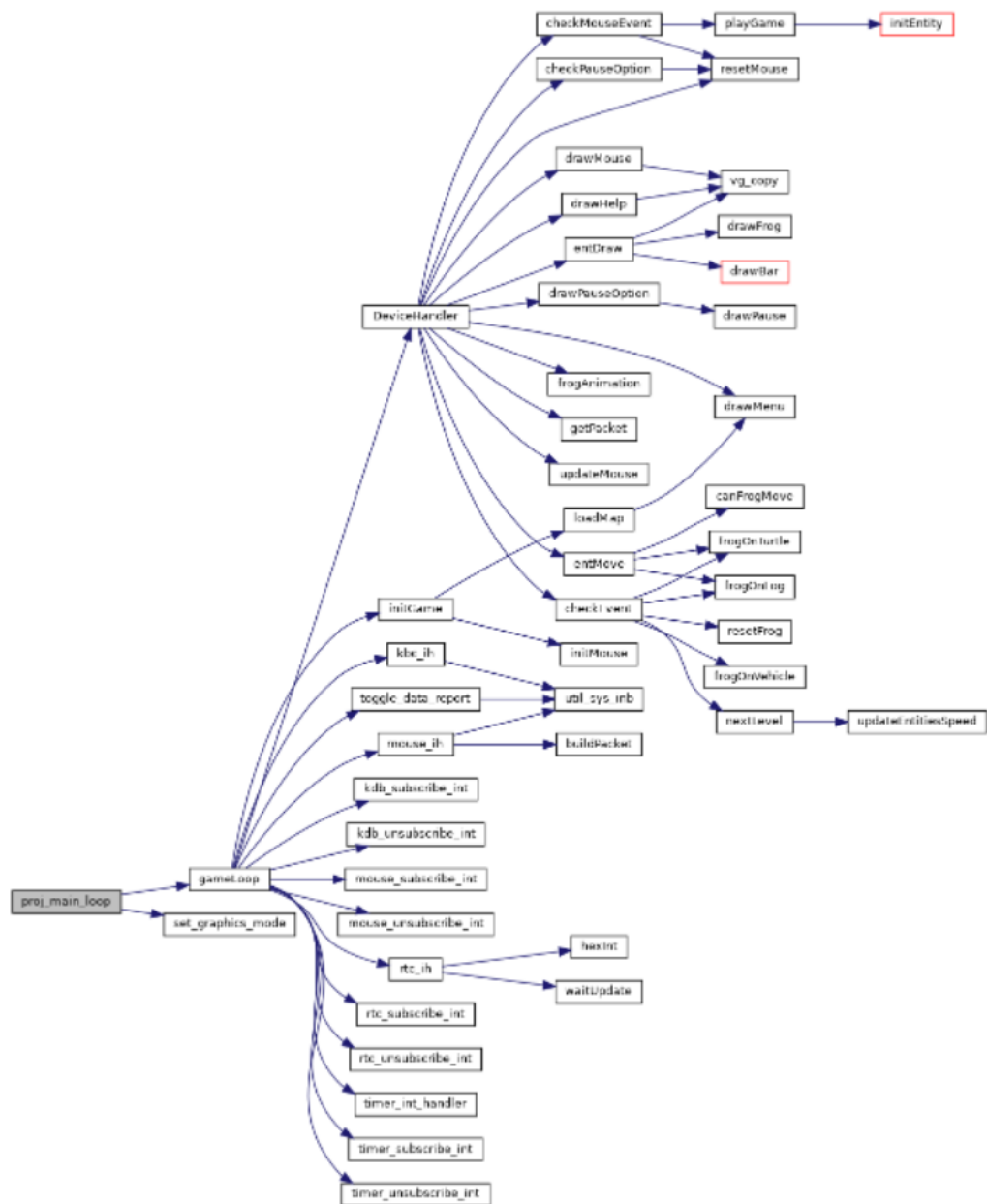
vehicle.c

A struct to the entity vehicle, which initiates the vehicle in a random position and loads it xpm. This random position checks if there is a space between each vehicle, and each vehicle has its own speed.

Relative weight of module: 5%

Developed by: André Soares, Vinícius Macedo

Function Call Graph



Implementation details

Topics covered in the lectures:

- State machines
- Event driven code

Topics not covered in the lectures:

- collision

Conclusions

Problems

- Problem while developing the mouse with the timer, when the timer couldn't process every packet of the mouse.

Like to have

- Like to have a serial port implementation, which would make it possible to play with a friend.

Main achievements

- Able to use all main I/O devices.
- Able to do the same features in our project like in the inspired game.
- Able to do smooth transitions between states and movements of the player.
- Able to change the speed of each entity.