Faculdade de Engenharia da Universidade do Porto

Project Report
Laboratório de Computadores
2023/2024

AIR HOCKEY

T17G02

Filipa Geraldes (up202208030@up.pt)

Afonso Domingues (up202207313@up.pt)

Afonso Machado (up202207611@up.pt)

Luís Arruda (up202206970@up.pt)

Contents

User Instructions

Description

Air Hockey is a game where players engage in high-speed matches on a sleek, low-friction table designed to make the disk glide effortlessly. Utilizing handheld pucks, players must hit the disk with precision, aiming to score goals in the opponent's slot.

How to Play

W / S : Navigate through the buttons in Main Menu and Game Menu.

Space : Select a button.

ESC : Exit the Game Menu or Rules Menu.

Mouse Left-Click : Grab the puck and move it.

Grab the puck with the mouse and move it to hit the disk and score a goal. The game ends when a player scores nine points first or when the time ends. If the time ends, wins the player with the most points.

Game Functionalities

Main Menu

When entering the game, its displayed the following screen to the user:

Figure 1 - Main Menu

It is display three options, which can be navigated with the keys W to go up and S to go down. When selecting an option, the user should press the SPACE key.

- **Play**, to start playing the game;
- **Rules**, where the user can learn about the game logic and how to play;
- **Exit**, to leave the game.

On the top left of the window, it is displayed the current time, and on the top right of the window, the current date.

Rules

In this window, it is display the rules of the game. To go back to the Main Menu, press ESC key.

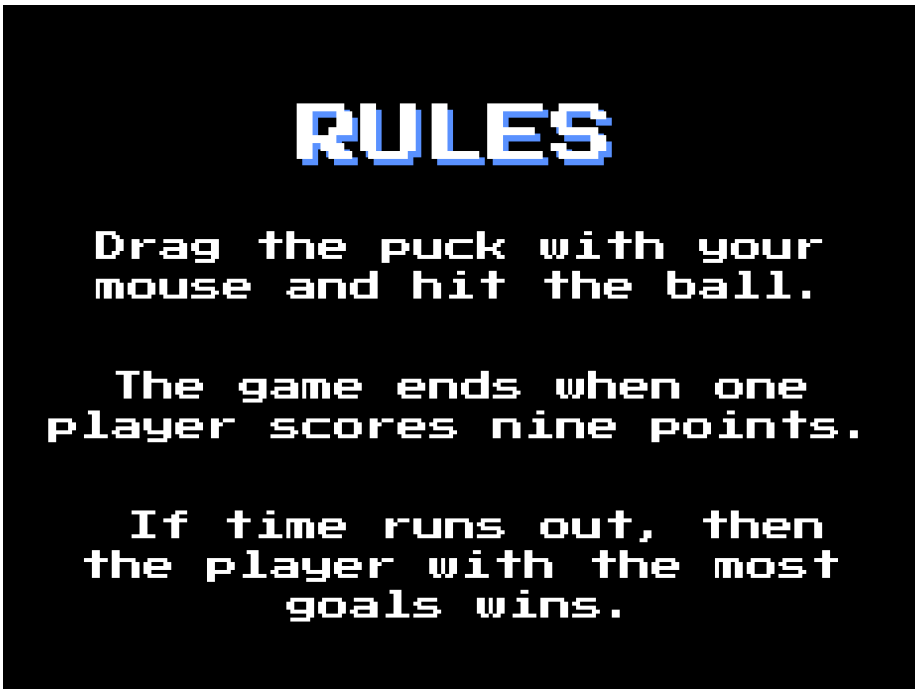


Figure 2 – Rules Menu

Game

On the left side of the window, it is displayed the time left for the game to end.

On the right side of the window, it is displayed the points of each player, represented by its number color.

In the center of the window, is the board of the game. There we can find two pucks, which can move up to the middle green line of its part of the board, and a disk, displayed on the center of the board.

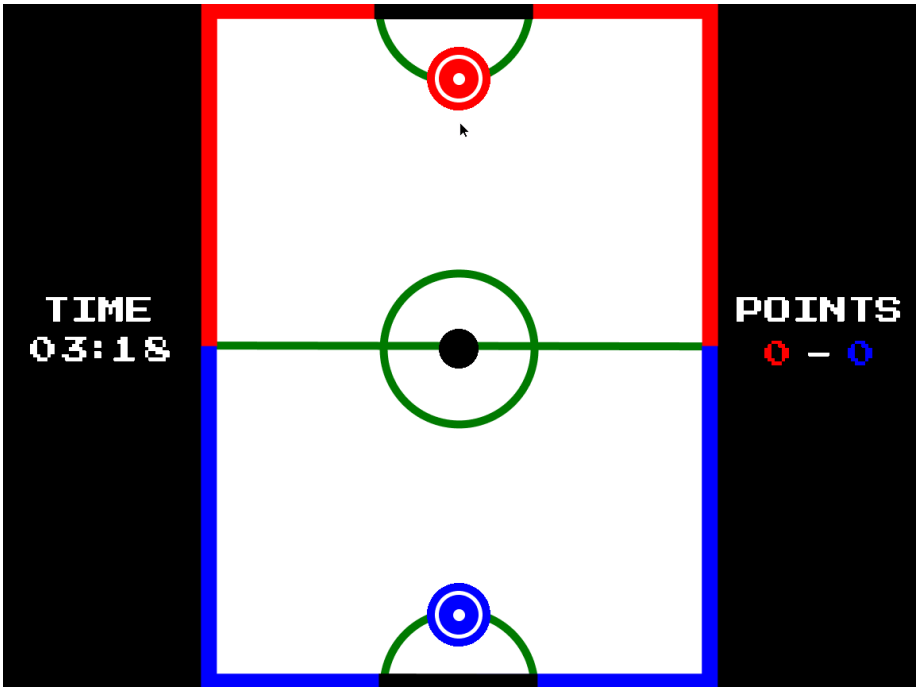


Figure 3 - Game

To win the game, a player must either score nine points or have the most points when time runs out. If the player loses, wins, or if the game ends in a tie, a message will be displayed with options to play again or exit.

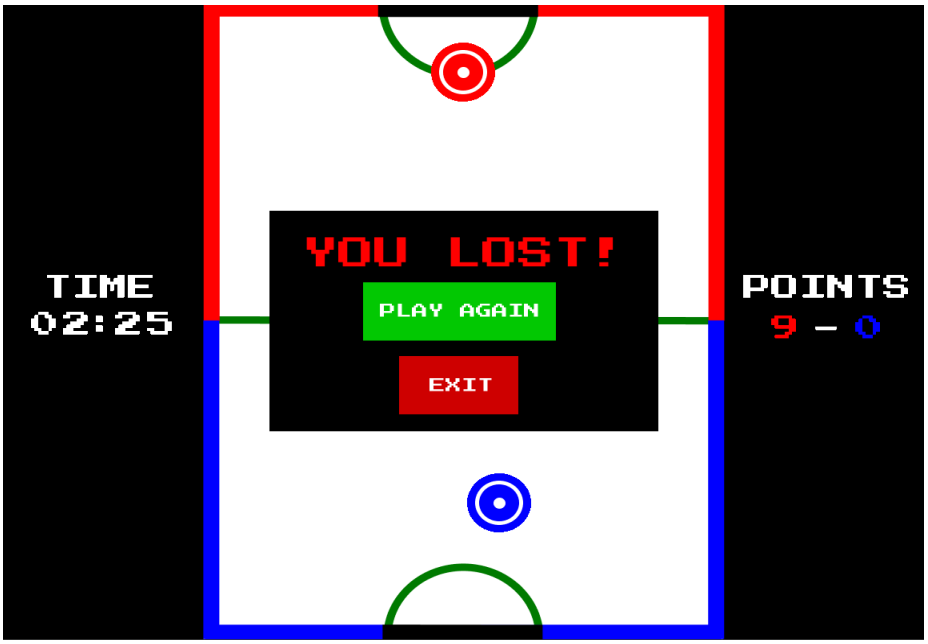


Figure 4 - You Lost

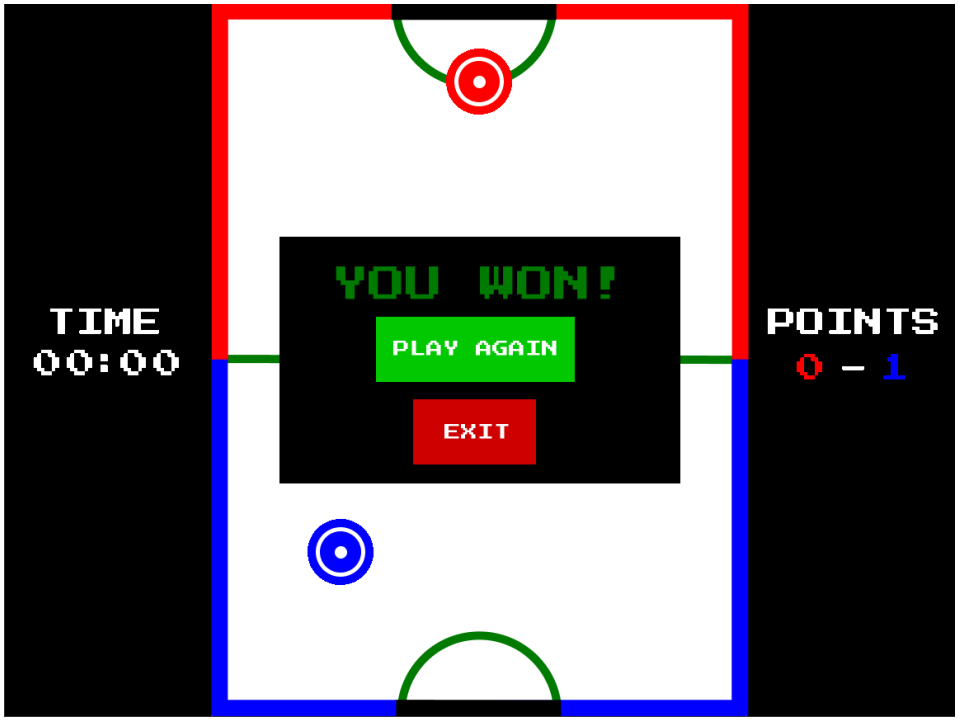


Figure 5 - You Won

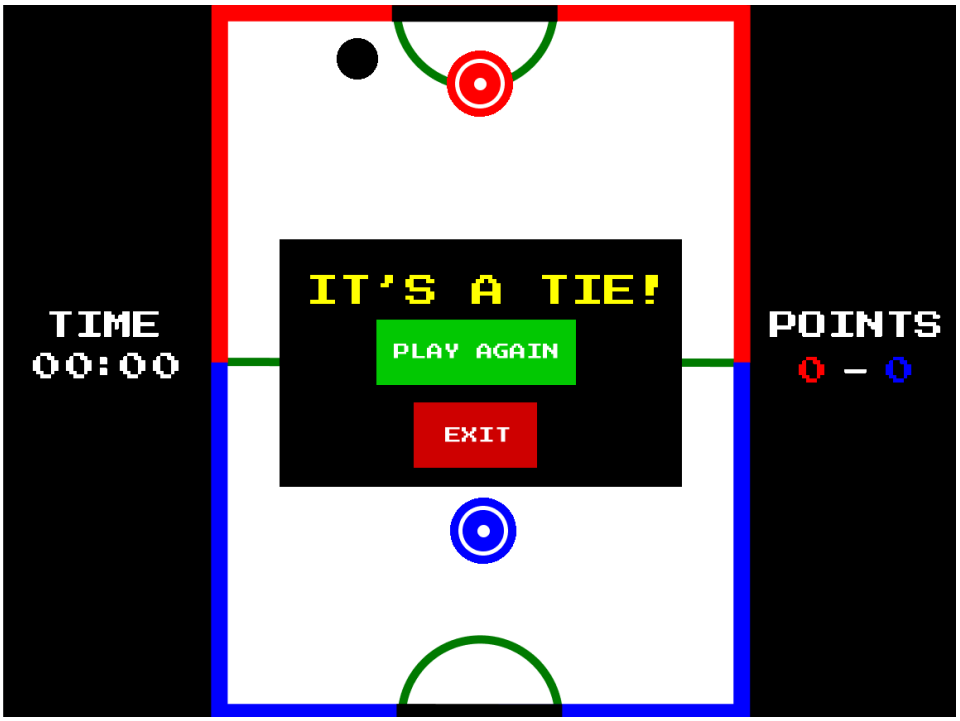


Figure 6 - It's a Tie

Project Status

Functionalities implemented

- Main Menu
- Game Menu
- Rules Menu
- Display real time and date
- Serial port for Multiplayer purposes
- Drag puck with mouse
- Keyboard to navigate through buttons
- Countdown for the game
- Physics of the puck and disk

Functionalities not implemented

- We initially hoped to implement a leaderboard feature, however we didn't have time for that. Instead we implemented a Rules Menu for the user to learn how to play the game.

I/O devices

Device	What for	Interrupt
Timer	Frame rate and game duration	Yes
Keyboard	Navigate through buttons and exit game menu or rules menu	Yes
Mouse	Drag the puck	Yes
Video Card	Game drawing	No
RTC	Date / time display	Yes
Serial Port	Multiplayer	Yes

Timer

The timer is used to generate continuous interrupts with a frequency of 60 frames per second, to ensure that the page is refreshed and the information is updated.

The implementation of the timer can be found in "…/src/controller/timer/timer.c". To deal with timer, we used the functions: timer_subscribe_interrupt, timer_unsubscribe_interrupt, timer_set_frequency, timer_int_handler and timer_get_conf.

Keyboard

The keyboard was implemented to navigate through the buttons in Main Menu and Game Menu through the W and S keys, SPACE key to select the button, and ESC to exit menus.

The implementation of the keyboard can be found in "…/src/controller/keyboard/keyboard.c". To deal with keyboard, we used the

functions: keyboard_get_conf, keyboard_subscribe_int, keyboard_unsubscribe_int and kbc_ih.

Mouse

The mouse is used to move the puck in the game, in order to hit the disk. Press the left button of the mouse to drag the puck and release it to hit the disk.

The implementation of the mouse can be found in "…/src/controller/mouse/mouse.c". To deal with mouse, we used the functions: mouse_get_conf, mouse_subscribe_int, mouse_unsubscribe_int, mouse_ih, kbc_write_cmd, kbc_write_argument, kbc_read_return, write_mouse_cmd, mouse_enable_reporting and mouse_disable_reporting.

Video Card

- Video Mode used: 0x14C
- Color Model: Direct color
- Resolution: 1152x864
- Color: XPM_8_8_8_8
- Double buffering, via copy

The video card is used to draw the UI of the menus and the game, which are generated in xpms and passed through sprites. Whenever we are in a button, it changes the color to orange, for the user to know which option is being selected. Double buffering was used for the game to run more smoothly.

The implementation of the video card can be found in "…/src/controller/graphics/graphics.c". To deal with video card, we used the functions: graphics_set_mode, map_phys_mem, vg_init, vg_draw_hline, create_mask, get_color_component. vg_draw_xpm, vg_draw_sprite, get_h_res and get_v_res.

RTC

The RTC is used to display the current time (hours, minutes and seconds) and date (day, month and year) in the Main Menu.

The implementation of the RTC can be found in "…/src/controller/rtc/rtc.c". To deal with RTC, we used the functions: rtc_subscribe, rtc_unsubscribe, rtc_get_config, verify_updating, verify_binary, rtc_convert_binary, rtc_update_info and rtc_read_convert.

Serial Port

The Serial Port is used for Multiplayer purposes. It lets two players compete with each other by sending information across two computers about the movement of the puck and disk, making the game playable and fun.

The implementation of the serial port can be found in "…/src/controller/serialport/serialport.c" and "…/src/controller/serialport/queue.c" . To deal with serial port, we used the functions: sp_subscribe_int, sp_unsubscribe_int, sp_ih, read_char, send_char, serialPort_init, serialPort_exit, serialPort_resetFIFO, transmit_puck_change, transmit_ball_speed, send_signal, read_next_signal, create_queue, destroy_queue, enqueue, dequeue, is_empty, queue_read_int, queue_get_size, queue_is_full and queue_get_capacity.

Code Organization / Structure

controller/

- graphics/ - video card
- keyboard/ - keyboard
- mouse/ - mouse
- rtc/ - real time clock
- serialport/ - serial port
- timer/ - timer
- utils.c – helper functions for sys_inb, lsb and msb

model/

- xpm/ - contains xpms to draw the game
- model.c - logic of game (points, win, lose, tie) and menus
- physics.c – handles game physics for the puck and the disk
- sprites.c – transforms xpm to sprites

view/

- view.c – render of game graphics and elements

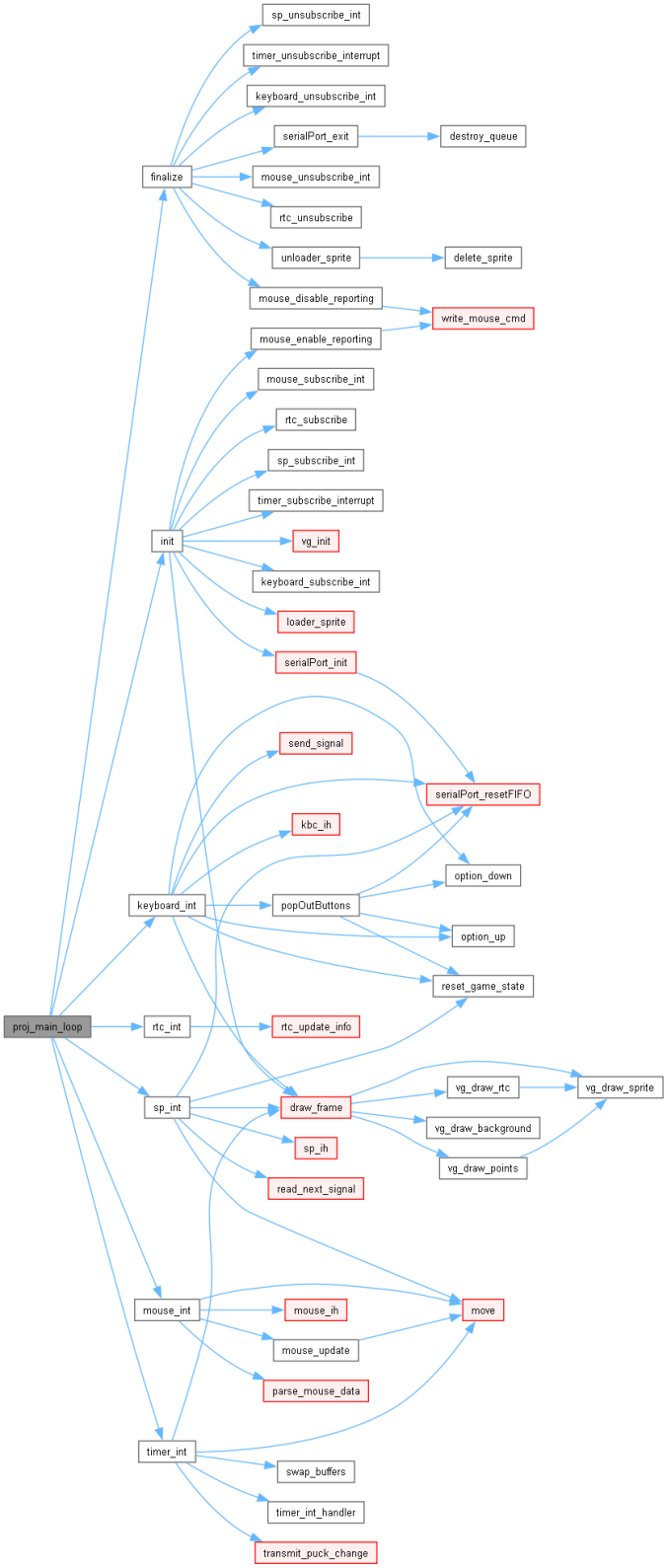
project.c – handles all devices, subscriptions, unsubscriptions and interrupts

Relative Weight (%) and Contributions

		Weight	Done by
Controller	graphics	5%	Labs
	keyboard	3%	Labs
	mouse	5%	Labs
	rtc	6%	Luís Arruda, Filipa Geraldes
	serial port	25%	Afonso Machado, Afonso Domingues
	timer	7%	Labs
	utils	1%	Labs
Model	xpm	5%	Filipa Geraldes
	model	10%	Everyone
	physics	15%	Everyone
	sprites	5%	Afonso Domingues
view	view	10%	Everyone
project	project	3%	Everyone

Function calls graph

We generated various function calls with Doxygen, the graph below shows the root of the function calls.



Implementation Details

- The code developed follows an **Model View Controller** (MVC) design pattern, that divides the related program logic into three interconnected elements
    - the model, responsible for the logic and internal representations of information;
    - the view, the interface that renders and displays the interfaces, for the user to interact with the game;
    - the controller, to handle I/O devices and links model and view.

- For the implementation of the devices:
    - RTC (Real-Time Clock) is was easy to implement, opting for the method using interrupts. We used it to display the current time and data in Main Menu;
    - Serial Port was more difficult to implement, as expected. The information sent should correspond to the data read, and the bit-wise operations were quite hard, namely knowing how many bits we had to send. Performance issues regarding the temporal and space synchronization between the two virtual machines were a big challenge we had ahead of us as well.
    - The other devices were implemented from the labs, with minor adjustments we had to make.

- In order to control the different states of the game, we implement a State Machine. When clicking on different buttons or the ESC key, we switch between states. In each state, the functions called and the graphics are different.

- When we use two virtual machines, the game rules smoothly and the virtual machines sends their puck position to each other. In each interrupt, the virtual machine receives the information and displays it as its opponent in the game board, as well as it points.

- To detect collisions of the pucks with the disk, we used ball_collision. The puck can't move after the middle of its part of the board, handled by detect_middle_field_collision. The collisions of the disk with the board limits were handled by detect_wall_collision, which also changes accordingly the trajectory of the disk. Finally, the collision with the goal are handled by detect_goal_collision, which also increments the points board depending on which player scored.

- To navigate through the options in the Main Menu and Game Menu (when a game ends) , we used option_up function to go up when clicking the W key, and option_down function to go down when clicking the S key.

Conclusion

Main achievements of the project:

- Use of all devices
- Good code structure, modulary and readability
- Documentation using Doxygen
- Menu Buttons
- Mouse to drag the puck
- Physics of the game
- Use of RTC to display time and date
- Use of Serial Port for multiplayer purpose
- Rules Menu
- User Friendly

Appendix

To run our game, through minix do:

“cd ../proj/src”

then run the command:

“make”

which compiles the code and finally do:

“lcom_run proj”

to run it.