Extended Abstract for Mini Project (IS1200)
Nina Shamaya (19990224-6280)
Azer Hojlas (19991003-3530)
2021-03-08

Pong Project

Description:

Our goal with the project is to create a pong game that follows the description of the rule set described at the pong wikipedia page: https://en.wikipedia.org/wiki/Pong. We have tried, to the extent of our capabilities, to create a pong game that meets the advanced project requirements (we failed at a few though), which are copied from the project page and displayed below:

- *[Highscore]* When a game ends, the player can enter their initials (at least three capital letters) in a high-score list. This high-score list can later be found through a menu system.
- You implement a program or a game using the basic I/O shield using the OLED graphical display, where you have graphics moving around on the whole screen, both in the X and Y directions. Your program must interact with the whole screen pixel-by-pixel and not just using the image and text example code from the labs (graphical objects are moving over pages/segments of the screen). This means that you need to have at least one object of the minimal size of 2x2 pixels (for instance a small image) that can move in **both** X and Y direction, **one pixel in each frame update**. The game/program should also be rather advanced, with several modes (1 and 2 players, high score list, etc.). Please see the project suggestions in 4.1 to get a better idea of what is required.
- There are both one and two-player modes, where the one-player plays against the computer (controlled by an AI). There are different difficulty levels for the AI. We did not manage to implement the two-player mode.
- The bounce angle is determined both by where the ball hits the paddle, and the current trajectory of the ball. If the ball hits the middle of the paddle, its angle of reflection is not modified. Towards the sides of the paddle, the angle of reflection is increasingly modified (in that direction). The bounce angle must be continuous (e.g., by representing the ball velocity with floating-point numbers). It is **NOT** OK to have a small discrete number of bounce angles that are, e.g., computed using a large set of if-statements.
- The ball can hit the paddle from the Y direction as well (resulting in a heavily modified angle of reflection). We did not manage to implement this.

Solution

This project is developed using only the supplied ChipKIT Uno32 board together with the Basic I/O shield. The OLED graphical display is used to present the users with a start menu, from where the player can choose to proceed with different options that either lead to a scoreboard or ultimately a game with a predefined set of rules (aka a game mode). We have used a number of predefined functions to create our game, mainly display_string, display_image and display_update. String and image do exactly what is implied in the function name, they are however not executed until display_update has been called. The project has been written in the C language code, with the help of the MCB32tools and the MSYS2 environment. The I/O devices that we use besides the screen are buttons and switches, more specifically buttons 2 through 4 and switches 3 and 4.

Verification

Verification has been done through a somewhat random and painstaking trial and error process. After each milestone of functionality had been completed (start logo, then start screen, then game mechanics etc), tests have been run to ensure that everything is working properly. This way, various bugs, issues and errors have unraveled themselves that have been solved along the way. Although a slightly inefficient approach, we have managed to produce a complete game that works smoothly.

Contributions

Due to the Coronavirus, we have not enjoyed the same degree of access to lab assistants and students as we would have liked under normal circumstances. However, the Q&A meetings have been extremely helpful, as well as the various github projects that can be found from previous students. Without these resources, finishing the game would be impossible. Below follows a detailed list of who made what functions:

Functions defined in mipslab.c that were modified or created by Nina Shamaya:

- display_image
- display_verticalSeg
- print_pong
- make_image
- start_game
- pause_game
- player_movement
- ai_movement
- move_body

- ball_adjustment
- is_touching
- lower_half
- upper_half
- goal
- show_highscore

Functions defined in mipslab.c that were modified or created by Azer Hojlas:

- reverse_bits
- clear_board
- create_body
- print_menu
- menu
- default_board
- create_middle_line
- get_random
- get_random_sign
- throw_ball
- move_ball
- print_mode
- choose_mode

Reflections

This mini project has been both fun and instructive. Although very challenging, overcoming said challenges has been both rewarding and informative, and our knowledge about the C language and especially the hardware has increased significantly. Especially rewarding is the fact that we have been able to make our own decisions pertaining to how the implementation of game mechanics would be executed, i.e we have through this project been given a high degree of creativity which has not been present in earlier courses. Also, we have experienced first-hand how our code works in external hardware (the kit), which is refreshing in the way that we have been able to physically interact with our programming. All in all we are very pleased with our project, and proud of the game that we have made.