Pong

IS1200

March 5, 2021

Philip Salqvist

phisal@kth.se
CINTE
1992-04-17

Contents

1 Objective & Requirement

The project objective is to develop a pong game on a CHIPKIT Uno32 together with an CHIPKIT Basic I/O shield. The game is rendered on the display embedded in the I/O board and the interaction from users is done using the push buttons on the the same board. The game consists of two players represented by a pong paddle each and a ball that bounces between them as well as the top and bottom limit of the screen. Player 1 is going to be positioned on the left and Player 2 on the right side of the screen. If the ball travels past the left outer limit of the screens x coordinate, player 1 gains a point, and vice versa for player 2 concerning the right outer limit of the screen. First player to gain 4 points, wins the game. The following are the basic requirements that must be met:

- Multiplayer mode
- Clearly describe visually how to start the game
- Score visible on LEDs and updated while playing
- User's can control their respective paddles on the Y-axis using push buttons
- When the ball passes a player's defended limit, the other player gains 1 point
- When the ball hits a player, it's direction along the x-axis is reversed
- The new direction of the ball is dependent on where the ball hits the player
- When a player reaches a defined number of points, the game stops
- Clearly state visually who wins on the display when game is over
- When the game is over, the players are instructed how to restart the game

2 Solution

The game is implemented on a CHIPKIT Uno32 together with a CHIPKIT Basic I/O shield. All of the input is generated using the shields push buttons and the output is displayed using it's display and LED lights. The display is used to display the game itself while the LEDs are used to visualize the current score of the game. To represent the players and the ball two structures called Player and Ball are created. The Player structure holds information about a players x coordinate, y coordinate, it's height and number of points. The Ball structure contains information about it's current position in terms of x and y coordinates and it's

direction along the x axis as well as direction along the y-axis.

Furthermore, the game contains three different states. The start state, play state and end state. The start and end states are rendered on the screen using the function $display\_string$ that was given during the lab sessions. During the play state a vector $game[128 * 4]$ is constantly modified and re-rendered. The players positions are modified using the function $draw\_player$, while the balls position is modified using $draw\_ball$. Both of the functions uses the macro $DRAW\_POINT$, which translates a x and y value to find the correct bit in the $game[128 * 4]$ array.

Moreover, everytime an interrupt occurs in timer2 the interrupt service routine $user_isr$ runs. We control the speed using a prescale of 1:256 as well as a counter inside the interrupt service routine. The speed can be controlled using the macro $SPEED$, which sets the upper limit of this counter. Everytime the counter reaches it's upper limit, we control which that currently is set and perform the necessary operations for that particular state. Concerning the development tools of the project MCB32 was used along with the C programming language.

3 Verification

The tests where performed to verify the requirements stated earlier in this abstract. The results from the tests are visualized in Table 1.

Test	Pass/Fail
Pressing any of the 4 buttons in the start state starts the game	Pass
When the ball hits the upper or lower limit of the display, the y direction is reversed	Pass
When pressing player 1 and 2's respective buttons at the same time, both players move	Pass
When the ball hits player 1 the balls x direction changes	Pass
When the ball hits player 2 the balls x direction changes	Pass
When the ball hits a player in the middle of the racket, the ball moves faster and it's direction angle decreases	Pass
When hitting the players edge, the ball slows down and it's direction angle increases	Pass
When the ball passes the screens left limit, player 2 receives 1 point	Pass
When the ball passes the screens right limit, player 1 receives 1 point	Pass
The points are visualized and updated during the game by the LEDs	Pass
When a point is scored the players and the ball moves to it's initial positions	Pass
When a player receives 4 points, the game moves to the end state	Pass
Pressing a button in the end state restarts the game	Pass
When the game restarts, positions and points are initialized	Pass

Table 1: Documenting performed tests and if they have passed or failed

4 Contributions

Due to the fact that I'm studying at a pace of 150% during this semester I have worked alone during the project to ease planning of the project execution.

5 Reflections

Overall the development process went pretty smoothly. I think this was due to the fact that the labs prior to the project where highly preparatory, especially lab 3 that laid a firm foundation in working with the hardware. Developing the logic of the game wasn't particularly challenging, although developing the functions that interacted with the hardware was. I would say that the task of translating the x and y values representing a particular element in a 2D array to the specific bit in the 1D game array was the most demanding part. One take away from the project is the importance of abstracting away complexity into small functions and the way that it makes the code more readable and easier to debug when things are not working properly.