Ball collision pong game w/ wall

By Edvin Frosterud and Yas Asghari

2022-02-07

Objective and Requirements.

The intent of our ***advanced project*** is to make a game similar in logistics to pong. We call it bubble pong. We are developing this game on our embedded platform(chipkit), with very basic knowledge of C. In this game, the objective is to break down the wall protecting your opponent by bouncing your ball back and forth at the wall. Just like regular pong, the player whose ball passes their "racket" loses. The challenge of the game is to both break down the walls protecting your opponent whilst simultaneously not dropping your own ball. Once the walls have an open path, the objective changes. Now you want to bounce the ball into your opponent's side of the battlefield, therefore making them have two balls. Juggling two balls is exponentially harder than one, and therefore will result in a loss if not dealt with. The main must requirements for the game are as follows:

- Supports multiplayer, 1vs1 on a single 128x32 screen
- Must support different difficulty levels in the form of the balls/rackets moving faster.
- The user must be able to control their "racket" with the onboard buttons on the ChipKIT io-shield.

If time allows for more features, we would like to add the following:

- An objective in the middle of the wall, see image 1. This objective would entice players to dig straight into the middle, but also give the opponent an easier way to get to your side.
- High score list, sorted by least amount of bounces required to get to the opponent's side.
- External buttons to control the game are easier, since the ChipKIT IO-shield has a few small buttons closely together.
- AI-mode.

Solution. We aim to develop our project on the ChipKIT UNO32 together with the basic I/O shield. Using the display on the I/O Shield, we will display the game and use the switches to move the player bar horizontally left and right in order to hit the projectile motion of the ball back towards the center of the display. We will use interrupts triggered by the built in timer to update the screen and control the speed of the game. All the development is done using the MCB32tools and all code is written in C, and any other libraries we deem necessary during development.

Verification. The project will be verified through a variety of examinations. For instance guessing the projectile motion of the ball and how it will bounce based on how it hits the walls through human intuition. To make sure our math works, we will write up testing situations with printed angles and a starting angle, therefore we will be able to predict the proper exit angle after bouncing, and if that corresponds to print, then our bouncing works.

Contributions. We hope that this project will allow us to teach eachother important skills within programming, and therefore we intend to do a lot of coding in a pair. We have however decided to make the projects in separate parts, like inputs, graphics, logics and game engine. Yas Asghari will be focusing primarily on graphics and input whilst Edin Frosterud will focus on game engine and logic for the game. The final division of work will be explained in the final abstract.

Reflections. The project will be discussed and reflected upon in the final abstract.

Image 1: Prototype, Potential look

The grid pattern will hopefully resolve on a small screen to look like a gray, to hopefully give an illusion of a third color.