Mini game console

When I first read this task I instinctively leaned towards making a snake game because we had been talking about it in earlier lectures. Snake also seemed relatively simple to program since I have some hobby experience with game development from before. Snake is also a game that is relatively flexible, as it can run comfortably on low frame rates yet still be enjoyable.

Starting the project I immediately opened “Arbeidskrav 2” where we previously worked with the same TFT screen and set it up accordingly. After that I looked up a guide on how to implement a joystick, and ended up following this guide by “Lastminuteengineers”. The guide Is pretty in depth on how joysticks work, and from my understanding a joystick is essentially two potentiometers, a button, and a gimbal which holds everything together.

By using my experience from the previous assignment I was able to make the TFT-screen feel a lot more responsive. During the previous assignment I filled the entire screen with the background color before writing the desired text, but this ends up being really, really slow with the Arduino Uno. To prevent the slowdown from affecting gameplay, I decided to only change the pixels who need to be changed, for example removing the tail of the snake by simply “writing” that area in the background color instead of the whole screen. Doing it this way is much faster because what the “FillScreen” does under the hood is using “WritePixel” on every pixel of the screen, which takes ages relative to only writing to the pixels which are being altered.

After being done with everything and playing Snake for a while I was thinking about what more I could add, after which I ended up adding a score and high score system. The high score is only saved in memory though. After that I was thinking of what more I could add to the breadboard to show my proficiency. I was considering adding some LEDs but thought that was a little boring, I also considered adding a potentiometer which would be mapped to for example 50 – 250 and would control the game ticks. After thinking it through I ended up dropping the idea the potentiometer because I considered it going against the idea of a high score and competition.

Ultimately I ended up with the idea of adding a IR remote control, which ended up being a bit difficult for a silly reason. I ended up following this guide from “Circuitbasics”, which is a good and detailed article on how IR remotes work. The only issue was that the IRremote library’s newest version, which is the one I initially used, is 4.1.X. After an embarrassingly long while of hair pulling and frustration of wondering why my remote was only sending the “FF FF FF” signal I found out that the library code isn’t always backwards compatible. In the end I found out after some digging in the IRremote github that the code I was using was from the 2.X.X version, and because I couldn’t get the 4.X code to work, I ended up just changing the version to 2.8. Because the remote I have doesn’t have any dedicated arrow buttons, I used the 2,4,6,8 buttons on the numpad for the directions.

In the end I’m happy with the game I ended up with. The gameplay is smooth, responsive, and addictive. Unfortunately, I didn’t have time nor energy to create the multiple challenge modes I had in my head, like for example having a hard mode where the tick-rate was twice as fast, or an easier mode with multiple fruits.

Sources:

Lastminuteengineers: https://lastminuteengineers.com/joystick-interfacing-arduino-processing/

Circuitbasics: https://www.circuitbasics.com/arduino-ir-remote-receiver-tutorial/

IRremote Github: https://github.com/Arduino-IRremote/Arduino-IRremote