CS12020 Writeup

From reading the initial briefing, I had already decided on a basic structure for my game. I decided to create a struct for each of the obstacles, containing their position on screen and their color. I then wrote a function to populate an array with these obstacles. They are initialised as black (color=0, background color) and with an x position starting at 8, and then increasing by one every 8th obstacle. This allows them to be "placed" off the left of the screen initially.

I then wrote my function to move the obstacles, which moved them left. It would also check if they were off-screen, if they were, it would move them to x=7 (appearing in the right hand column). I had some issues with this initially, I started off moving them to x=8, hoping to modify their values offscreen. I then realised that I wasn't rendering them until all of the obstacles had ticked anyway so I changed it to x=7. This also solved an issue I had where there was always one empty row cycling round the screen. Then, it would check all of the obstacles in x = 7 and randomly assign them colors, a y position, and figure out if there were too many obstacles in the column. It then checks if any obstacles are in the players column, and checks if its in the same y value as the player. When it collides, it looks up the color of the player and adjusts the move interval accordingly, with an inline if statement on the increase to ensure it never goes above 250ms.

Then I designed my render function, its very basic. It checks if any of the objects are off screen, or assigned a non-visible colour (black) and skips those, the rest it calls AberLED.set(). My player is drawn in this way too, and is always drawn last to ensure it is on top.

My main loop is simply a switch statement. With the 4-states (I added a state for being paused) I felt it cleaner to include all of the actions from a state into its own function. I check for button presses outside of this switch statement, and have another switch in the button press function to decide what action to take in the context of the current state.

The gamePlay function moves the obstacles once every time the move interval is passed. It renders all of the objects as often as it is called however. The pause function is similar, but omits the moveObstacles() function call and only shows the player once every other frame, to make it flash. (To clarify, I say "frame", when really its a set time difference when it is shown or hidden. I couldn't think of a way to say it without explanation)

The EEPROM was the final thing I worked on. I wrote a array of buffer-valid unsigned shorts to be read, corresponding to each number. I initially had an issue where each buffer row is 16 bits (2 bytes) so it would have to be stored, essentially, fragmented. I then realised that I could trim 4 bits from either side, making it 1 byte. I accommodate for this by right shifting the read values 4 bits. Storing the array in the EEPROM was slightly unnecessary, but it was more fun than just using a 2D array.

I store the high score in byte 80, as this is the next free byte after the array of numbers.

I have attempted to adjust all my variables based on their expected range to use the most efficient type I could, despite barely hitting 40% of program storage space or dynamic memory.

I only realised I had completed too many extra features after spending a good 2 days working on them, but it was a good learning experience anyway.

I think I did pretty well in this assignment, and I think I have decently robust code, good commenting and efficient solutions to things. Having read "Appendix AA - Assessment Criteria for Development", I believe I would fall in the 60-69% Range.

Tasks Completed:

- [x] ~~Part One~~
- [x] ~~Part Two~~
- [x] ~~Part Three~~
- [x] ~~Part 4:~~
    - [x] ~~increase the obstacle count over time (5%);~~
    - [x] ~~keep a score (number of squares travelled? Time?) and output it to the serial port on transition to the end state (5%);~~
    - [x] ~~add a pause function to the Playing state, so that pushing FIRE stops the scrolling and starts the yellow player dot flashing. Pushing FIRE should start the game again (5%);~~
    - [x] ~~modify the End state to display the score (perhaps an array of images of digits, which are shown on the screen) (10%);~~

- [x] ~~store the high score on the EEPROM and display that in a new state between End and its return to Start (15%).~~

The program I used to write the numbers to memory:

```
#include <EEPROM.h>

/*
(if you Ctrl-F '10', you can see the numbers in their whole glory.)
They had to be reversed because of how the function writes them to the
screen.
*/
uint8_t numbers[10][8] = {
  { 0b00000000,
    0b00101000,
    0b10000010,
    0b10000010,
    0b10000010,
    0b10000010,
    0b00101000,
    0b00000000 }, //0
  { 0b00000000,
    0b00001000,
    0b00001010,
    0b00001000,
    0b00001000,
    0b00001000,
    0b00101010,
    0b00000000 }, //1
  { 0b00000000,
    0b00101000,
    0b10000010,
    0b00100000,
    0b00001000,
    0b00000010,
    0b10101010,
    0b00000000 }, //2
  { 0b00000000,
    0b00101000,
    0b10000010,
    0b00100000,
    0b00100000,
    0b10000010,
    0b00101000,
    0b00000000 }, //3
  { 0b00000000,
    0b00100000,
    0b00101000,
    0b00100010,
    0b10101010,
    0b00100000,
    0b00000000,
    0b00000000 }, //4
  { 0b00000000,
    0b10101010,
    0b00000010,
    0b00000010,
    0b00101010,
    0b10000000,
    0b10000010,
    0b00101000 }, //5
  { 0b00000000,
    0b00101000,
    0b00000010,
    0b00000010,
    0b00101000,
    0b10000010,
    0b10000010,
    0b00101000 }, //6
  { 0b00000000,
    0b10101010,
    0b100000,
    0b00100000,
    0b00100000,
    0b00001000,
    0b00001000,
    0b00000000 }, //7
  { 0b00000000,
    0b00101000,
    0b10000010,
    0b10000010,
    0b00101000,
    0b10000010,
    0b00101000,
    0b00000000 }, //8
  { 0b00000000,
    0b00101000,
    0b10000010,
    0b10000010,
    0b00101000,
    0b10000000,
    0b00101000,
    0b00000000 } //9
};

void setup() {
  for(int i = 0; i < 10; i++){
    for (int j = 0; j < 8; j++){
      EEPROM.update(j + i*8, numbers[i][j]);
    }
  }
}

void loop() {
  return;
}
```