Semester 1, 2019

CS12020 Arduino Game Project: Write-Up

by Matthew Hibbin (mjh32)

Introduction

The task for this assignment was to make a basic scrolling game using an Arduino Uno paired with Aberystwyth University's LED Matrix Shield. I used my knowledge from past programming experiences and completing worksheets throughout this module to help me create a solution to the assignment.

Model

I have taken inspiration from the worksheets undertaken in the Game Practical Sessions for the model. Input, update and render were all functions implemented and were called in the main loop. Each one would contain code for each state, call other standalone functions and set variables in order to make the game work correctly. The update function had an interval variable that would dictate the time each frame would appear on the screen. The value of interval would decrease and increase to speed up and slow down the game respectively.

Worksheet three demonstrated the use of a two-dimensional array and random 'block' generation which I interpreted in order to create my game. I reverse engineered the code to make the dots scroll left, instead of down. Regarding the generation, I modified the function so there would be random red dots that would have a one in ten chance of spawning. I then added code so that 1 in 4 red dots would be green via a static counter variable.

For the player, I used an array with two entries for both x and y. In this game, the x always stayed at the value 0 as the player can only move up and down.

I handled collisions by making two functions to return whether the player had hit a red or green dot respectively. The update function would use this information to change the value of the interval variable accordingly.

The states 'start', 'playing', 'pause' and 'end' were implemented using a state machine model. The three main functions have a switch-case statement for each state, so that all three functions will work together to make the game operate correctly.

Issues

I created a draft of the code to initially test all the functions I would implement. In this draft I had quite a few problems, due to poor memory management within the code. One of these problems was when no green dots were appearing on the screen. Whilst trying to find the cause of the issue, I found that printing out the variable that kept a count on the red dots to the serial port made the green dots appear. When writing the final version of the program I found that there was a function that was randomly writing to memory which was causing this issue. I deleted this part of the function and the code operated as expected.

Another issue I encountered was when implementing a feature to speed up the game gradually whilst playing. The problem was that initially the interval variable decreased by one each time update was called. This made the game start to rapidly speed up whilst playing, ending the session abruptly. To counter this, I made the interval variable a float type and divided it by 256 each time, which made the game's speed up consistent.

The pause function also needed to be changed, as at first it would just set the interval variable to a value that made it change frame every 24 hours, which was not truly pausing. A new pause state was added instead with separate code to meet the specification.
The last issue was displaying the score and high score on the end screen. Since displaying the scores as a decimal number is quite troublesome on such a small screen, I instead opted for a different numbering system. I translated the scores into binary and displayed them as a horizontal line of 8 dots on the end screen. It is not the most user intuitive but allows for a less complicated end screen, making it quicker to read than scrolling through a decimal number.

Self-Evaluation

When creating my code, I used the programming techniques that I've learnt over the past ten weeks. I split up my code sensibly into functions that would be called at multiple points throughout the code. I used as little global variables as I could and made sure to manage memory efficiently on the Arduino.
I feel as if it might've been possible to have less global variables, but a lot of values needed to be accessed throughout the program, so these were appropriate.
I also think, that given more time, I could have displayed the scores at the end in decimal rather than binary numbers.
However, I do believe that my code was efficient and used the appropriate methods to create the game to the specification. All the extra features were seamlessly integrated into the game. The game is fully playable and passed all the tests within the test table.

Function List

- playerHit(): Detects if the player has hit a red dot.
- playerPower(): Detects if the player has hit a green dot.
- clearAllBlocks(): Sets the entire blocks[][] array to be empty.
- bitTranslation(unsigned char num): Translates an unsigned char to bits.
- writing(int x, int y, bool special, int luck): Adds red and green dots to the blocks[][] array.
- generation(): Randomly decides if dots are going to spawn.
- scrolling(): Increments all the dots forward by one.
- renderBlocks(): Uses the AberLED.set() function to write the contents of the blocks[][] variable to screen.
- getInput(): Registers valid inputs within the current state to effect the game accordingly.
- updateLCD(): Calls functions and sets variables each interval appropriate for the current state.
- render(): Renders the displays for each state with the use of standalone functions.