Zirui (Felix) Xiao

Lab 4

Software Development Plan

The main game will run in a while loop and implement a finite state machine using a switch statement. The finite state machine is described below.

Finite state machine:

* State 1: startup state
  + Show the splash screen.
  + Plays a sound to signal the start of the game.
  + Calls a method to setup the LCD with all blocks, the platform, and the ball to initial values.
  + Transition: if button C is pressed, go to state 2.
* State 2: game playing state
  + This is the most common state and the game will be here while running.
  + Transition: if the ball comes into contact with a block, go to state 3.
  + Transition: if the user presses button D, go to state 4.
  + Transition: if the ball falls below the platform, go to state 5.
  + Transition: if there are no blocks left on the screen, go to state 6.
  + Transition: if button C is pressed, go to state 1.
* State 3: block is hit state
  + Plays a sound when a block is hit.
  + Calls a method that adds to the player’s score.
  + Transition: if there are no blocks left on the screen, go to state 6.
  + Transition: if there are still blocks left on the screen, go to state 2.
* State 4: power-up activated state
  + Checks if the user has power-ups.
  + If so, plays a sound to signal that a power-up is activated.
  + Sets the activated power-up to false.
  + Transition: if there are no blocks left on the screen, go to state 6.
  + Transition: if there are still blocks left on the screen, go to state 2.
* State 5: ball is below platform state
  + Calls a method to reduce the player’s score.
  + Calls a method to reposition the ball.
  + Transition: if there are no blocks left on the screen, go to state 6.
  + Transition: if there are still blocks left on the screen, go to state 2.
* State 6: game over state
  + Plays a sound to signal that the game is over.
  + Shows a game over screen with the score in a larger font.
  + Transition: if button C is pressed, go to state 1.

How the game will be coded:

* Creating the ball: The bouncing ball algorithm will be modified so that the ball can interact with blocks.
* Creating the blocks: The block will be an object that is shaped like a rectangle and has properties like color, active or not, etc.
* Creating the platform: The platform will also be an object that can be moved to the left and to the right.
* Creating the power-ups: There will be a boolean value for each power-up that will start as false. The temperature sensor code from the last lab will be modified so that the proper boolean is set to true after the temperature is within a range for a certain number of seconds.
* Activating the power-ups: Within the switch statement, if the player chooses to activate a power-up and a power-up boolean value is true, then the boolean will be set to false and the power-up modifications will be applied for a temporary period.
* Things that need to be added: a platform with methods that move it to the left and to the right.
* Other methods that will be added: displaying the score, resetting the score, loading the game by assembling the blocks and positioning the ball before the game begins.

Stages of work:

1. Create objects
   1. Write the code for a block and initialize a block to be placed on the LCD with coordinates and correct properties.
   2. Write the code to place a ball on the screen.
   3. Write the code to place a platform onto the screen.
   4. Tested visually by making sure that objects behave as expected on the LCD.
2. Linking buttons, temperature sensor, speaker
   1. Write the code that calls a method when each of the buttons is pressed.
   2. Write the code that constantly gets the value from the temperature sensor.
   3. Write the code to play a sound in the speaker.
   4. Tested visually by using on-LCD print statements for the temperature and methods being called. Also tested with listening to the speaker.
3. Finite state machine
   1. Implement the FSM detailed above with a while loop and a switch statement.
   2. Tested visually with on-LCD print statements so that I can make sure the proper state is being transitioned to.
4. Assembling the game.
   1. Put the pieces from #1, #2, and #3 together to make a functional game.
   2. Tested by making sure that the game adheres to the rules described above.
5. Testing and customizing
   1. Addressing potential bugs that may arise.
   2. Tested by visually making sure that nothing unexpected happens on the screen.