**ExMaze**

**LAB 9**

**SECTION X**

**SUBMITTED BY:**

**NATHAN SHULL**

**SUBMISSION DATE:**

**DECEMBER 2, 2016**

**Problem**

The purpose of this lab, the ExMaze, is to create a program, which draws a variety of different mazes, based on what the user inputs for a difficulty. The program then reads in accelerometer values to know if the avatar in the maze should move left, right, or go down. The program should not allow an avatar to move into a space occupied by a wall. The point of this lab it to practice working with two-dimensional arrays and develop skills in handling events in a loop.

**Analysis**

For part 1, the problem stated that I must create a random maze, start the avatar at the top center of the screen, and make a delay for the avatar to fall down the screen. In the second part, I had to add 3 more elements to the program. First off, I had to read input from the Esplora that would make the character move to the left or right, depending on the accelerometer values. Second, I had to make sure that the avatar did not move into locations occupied by the maze or off of the screen. Finally, the third element of part 2 was to check if the avatar made it to the bottom of the screen, and if so, print “You win!” I also chose to attempt the bonus one, which was to detect when the avatar was in a position where it could not move left, right, or down, and was in a position that would print “You lose.”

**Design**

To create a random maze, I first iterated through a two dimensional array. Within the array, I used the rand( ) function mod 100, so that a higher difficulty would have a greater chance of drawing a WALL, and a lower difficulty would have a higher change of drawing an EMPTY\_SPACE. To start a character at the top center of the screen I declared and initialized two variables, c(for columns) and r(for rows). I initialized c at 50 and r at 0, making it start at the top center. To create a delay, I subtracted currentTime from lastMove, and compared it with 0.4, so that every four tenths of a second, the avatar would look to move again. I assigned lastMove to currentTime after it went through each loop.

To read accelerometer data, I made a for loop for i being less than the number of samples, and creating an array to which those values were assigned to a\_x. And then to move it left of right, I used a few different if loops. If the space down and to the right was clear, AND if the eslora was close to -1, than the avatar would move down and to the right, and an empty space would be drawn where it previously was. I also wrote similar if statements for if the esplora was tilted to the left, or not even tilted at all. My last if statement was if there was no wall under the character, to add one to row, in order to keep the avatar moving downwards. If the rows got to 81, that means that the avatar reached the bottom, and I printed the you win message.

**Testing**

The biggest issue I had when testing this program was having my avatar go left and right. For a long time, I could only get the avatar to go right, but it would never move to the left. I carefully examined the main function, and couldn’t seem to find anything wrong. I then went looking into the functions I was using to move the avatar left, mainly calc\_roll. It was here that I found my problem. I was comparing x\_mag to positive 0.3 in both if statements, rather than a -0.3 in the one. After changing the sign, I got a program that was able to move an avatar left and right.

**Comments**

1. For the safe to go right/left, what is checked is if there is a wall one row down, either to the left or the right, depending on which direction you want to go. For the can I fall, the program checks if there is a wall or empty space at [c][r+1], or one row down.
2. I actually did check to see if a player loses a game. To check this, you create an if statement that checks if there is a wall directly to the left, right, and below the current avatars positon. If ALL three are true, than the if statement breaks, and the program goes to (r != 81), and prints “You lose.”

