ExMaze

Section T

Larisa Andrews

12/3/2015

Testing:

1. I used the close to function and the tolerance .2 to make the Esplora go to the left of to the right.
2. To know if you are stuck you have to determine whether there is a wall on all sides of you. You do this by checking the left right and below.

```c
// Headers
#include <stdio.h>
#include <math.h>
#include <ncurses/ncurses.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>


// Mathematical constants
#define PI 3.14159

/* Screen geometry
Use NUMROWS and NUMCOLS for the screen height and width (set by system)
*/
#define NUMROWS 80
#define NUMCOLS 100

/* Character definitions */
#define AVATAR 'A'
#define WALL '*'
#define EMPTY_SPACE ' '

/*Number of samples taken to form an average for the accelerometer data
Feel free to tweak this.  You may actually want to use the moving averages
code you created last week along with this number as your windowing size
to get a playable game*/
#define NUM_SAMPLES 10


/* 2D character array which the maze is mapped into
 You should fill this with characters, not numbers. */
char MAZE[NUMROWS][NUMCOLS];


/* PRE: The level of difficulty will be entered on the command line.
You will have to use the argument to the command line to determine how
difficult the maze is (how many maze characters are on the screen).
POST: Generates a random maze structure into MAZE[][]
You will want to use the rand() function and maybe use the output %100. */
void generate_maze(int difficulty);

/* PRE: MAZE[][] has been initialized by generate_maze()
POST: Draws the maze to the screen.  You must use the draw_character
function to print to the screen.  You cannot use printf in curses.  */
void draw_maze(void);

/* PRE: 0 < x < NUMCOLS, 0 < y < NUMROWS, characters are defined above
```

```
POST: Draws character use to the screen and position y,x as in a graph
where x is the horizontal axis and y is the vertical axis.
When using the i and j from the maze, you will want to remember that
i (outer loop) is the rows and corresponds to y, while j is the columns
and corresponds to x.  */
void draw_character(int x, int y, char use);

/* PRE: -1.0 < x_mag < 1.0
POST: Returns tilt magnitude scaled to -1.0 -> 1.0
You may want to reuse the roll function written in previous labs. */

double calc_roll(double x_mag);

void updatebuffer(double buffer[], int length, double new_item);
double avg(double buffer[], int num_items);

// Main - Run with './explore.exe -t -a -b' piped into STDIN
void main(int argc, char* argv[])
{
    int difficulty = 0;
    int t, time2, time3, time, time4, time5;
      int timeSec = 0;
      int timeSet = 0;
      int leftB,upB,downB,rightB;
      int joy;
      int s;
      double ax, ay ,az;
      int i, j;
      double avgX, new;
      double arrayX[1000];
      int x, y;
      int game, win;



      // setup screen
      initscr();
      refresh();

      sscanf(argv[1], "%d", &difficulty);


      generate_maze(difficulty);


              // Generate and draw the maze, with initial avatar

      draw_maze();



      x = 50;
    y = 0;

    draw_character(x,y,AVATAR);

      // Event loop
      do
      {
     scanf("%d, %lf, %lf, %lf, %d, %d, %d, %d, %d, %d", &t, &ax, &ay, &az,
&upB, &downB, &leftB, &rightB, &joy, &s);// Read accelerometer data to get
ready for using moving averages.



      //printf ("%lf\n", avgX);



            time2 = t;
                  if(timeSec == 1){
                        if(time3 < time2){
                              if(MAZE[y+1][x] != WALL){

                                  draw_character(x,y,EMPTY_SPACE);
                                    y++;
                                     // Delete previous A's
                                    draw_character(x,y,AVATAR);
                                    timeSec = 0;
                                 //timeSec = 0;
                              }
                        }
                  }
                  else {
                        time2 = t;
                        time3 = t + 250;
                        timeSec = 1;

                  }

                        updatebuffer(arrayX, 25 ,ax);
                  avgX = avg(arrayX ,25);
                    //new = calc_roll(avgX);


            time4 = t;
                  if(timeSet == 1){
                        if((avgX > .2) && (x != 99)){

                          if(MAZE[y][x+1] != WALL){
                                    if(time5 < time4){


                                            // Move Down the Maze
                                          draw_character(x,y,EMPTY_SPACE);
                                          x++;

                                          draw_character(x,y,AVATAR);
                                          timeSet = 0;

                                    }
                              }
                        }
                if((avgX < -.2) && (x != 0) ){

                          if(MAZE[y][x-1] != WALL){
                                    if(time5 < time4){


                                            // Move Down the Maze
                                          draw_character(x,y,EMPTY_SPACE);
                                          x--;

                                          draw_character(x,y,AVATAR);
                                          timeSet = 0;

                                    }
                              }
                        }
                  }
                  else {
                        time4 = t;
                        time5 = t + 250;
                        timeSet = 1;

                  }


            // Read data, update average

            if((MAZE[y][x-1] == WALL)&&(MAZE[y][x+1] == WALL) &&(MAZE[y+1][x]
== WALL)){
                  game = 1;
                  win = 0;
            }
            if(y == 79){
                  game = 1;
                  win = 1;
            }




       //ay = 0;


            // Is it time to move?  if so, then move avatar

      } while(game != 1); // Change this to end game at right time

      // Print the win message
      endwin();

      if (win == 1){
      printf("YOU WIN!\n");
      }
      else if(win == 0){
            printf("YOU LOSE!\n");
      }
}
```

```
/* PRE: 0 < x < NUMCOLS, 0 < y < NUMROWS, characters are defined above
POST: Draws character use to the screen and position y,x as in a graph
where x is the horizontal axis and y is the vertical axis.
When using the i and j to draw the maze, you will want to remember that
i (outer loop) is the rows and corresponds to y, while j (the inner loop) is
the columns
and corresponds to x.

This code places the Avatar and the maze on the screen.

IT WORKS CORRECTLY AS PROVIDED.
PLEASE DO NOT CHANGE THIS FUNCTION. */

void draw_character(int x, int y, char use)
{
      mvaddch(y,x,use);
      refresh();
}
void generate_maze(int difficulty){
      int count;
      int i;
      int j;
      srand((int)time(0));




      for(i=0; i<NUMROWS; i++){

            for(j=0; j< NUMCOLS; j++){
                        count =     (rand()%100);
                  if(count < difficulty){
                  MAZE[i][j] = WALL;
                  }
                  else{
                  MAZE[i][j] = EMPTY_SPACE;
                  }

            }
      }







}
void draw_maze(void){
      int i;
      int j;
      for(i=0; i<NUMROWS; i++){
```

```
                  for(j=0; j< NUMCOLS; j++){

                  draw_character(j,i,MAZE[i][j]);


                  }
            }
}
double calc_roll(double x_mag){
      double roll;
      roll = asin(x_mag);
      if(roll > 1){
            roll = 1;
      }
      else if(roll < -1){
            roll = -1;
      }
      return roll;
}
void updatebuffer(double buffer[], int length, double new_item){
      int i;
      for(i = 1; i<length; i++){



            buffer[i-1] = buffer[i];


      }
      buffer[length-1] = new_item;
}
double avg(double buffer[], int num_items){
      int i;
    double avg;

      for(i = 0; i < num_items; i++){
      avg = buffer[i] + avg;
      }
      avg = avg/num_items;
      return avg;
}
```