**Lab 9: DS4Maze -- Part 1**

*NOTE: This is a 2 week lab. A working program with parts 1A and 1B below is due before the beginning of lab next week (during the undergraduate TA’s office hours). Demonstration to an undergraduate TA for the first part* ***will be required during their office hours*** *before the beginning of next week’s lab so it is important to work hard in this lab session.*

**The lab report will be due in two weeks.**

**Objectives:**

* Practice Top-Down Program Design, Problem Solving in C
* Work with 2-dimensional arrays
* Develop skills in handling events in a loop

**Starting Point:**

* [*lab9.c*](https://drive.google.com/file/d/0B53msxezjTwYcS1xTUs3SHJ1dlE/view?usp=sharing)

To compile your code you will need to append -lncurses to your gcc command line

gcc -o lab9 lab9.c -lncurses

**Process:**

**Creating a New Folder**

Create a new folder named *lab9* in your *se185labs* folder on the U: drive. You will want to copy over ds4rd.exe to the *lab9* folder.

**Problem**

In this and the following lab, you will develop a simple, real-time game controlled by the DualShock 4.

* 1. Create a random maze of characters on the screen. The variables COLS and ROWS will indicate the size of the window when it is started. To get the right size window screen, right click on the title bar in Cygwin and select Options>Window and change the columns and rows to match your COLS and ROWS setting in the starting code.
  2. Start your avatar (a single character) at the top center of the screen. **A function that lets you write a character to any character location in the window is included in the starting source code. DON’T CHANGE THE FUNCTION.**
  3. Every so often (a delay to be found by you), the avatar will fall one line down the screen.
  4. Character movement
     1. If the DualShock 4 is tilted right, the avatar will move to right.
     2. If the DualShock 4 is titled left, the avatar will move to the left.
     3. You may need to worry about a tolerance value here.
  5. The avatar may not move into locations occupied by the maze or off of the screen.
  6. The avatar wins if it makes it to the bottom of the screen without getting stuck.

There are three fundamental things to worry about in any game: the game state, the rules, and how to update the game state based on user input. We will begin with the game state in this lab and worry about implementing most of the updating and rules code in the following lab. The game state consists of:

* 1. The maze layout
  2. The avatar's location

**Part 1A: Develop the Maze**

You will generate a random maze of characters in a two dimensional array. This will be the same maze that you output to the screen. Using good modular programming techniques, generate the maze and display it on the screen.

The rand() function in stdlib.h will be very useful, as it will return a new random int every time it is called that is from 0 to a very large number. HINT: Taking rand's output % 100 may be useful.

To allow for differing levels of difficulty, your code will be given a difficulty value of 0-100 on the command line. This difficulty is the probability in percent that a given space of the maze will be an obstacle. Hence, difficulty zero indicates a blank maze while 50 indicates that roughly half of the characters will be blank.

**Part 1B: Character Movement**

The second part of the state is the avatar's position. In the sample code provided, we have included a function that lets you write to a character to a specific x,y position on the screen. Be sure to use this function (**draw\_character**). **DO NOT MODIFY THIS FUNCTION**. In a loop, you will make the character begin at the top center of the screen and move downward every so often. Note that if you do this without a delay, the program will complete so fast that you can barely see the avatar move.

To add delay, remember that if the -t option is given to ds4rd.exe, the first argument is the time in milliseconds since the executable starts. Use this data to wait some number of milliseconds (a delay for you to determine), and then implement the avatar moving down the screen with the delay. You may wish to keep track of a moving average of the gryroscope as well here so that you can use them to control the avatar next week.

**Demonstration of Parts 1A/1B due before the beginning of next week's lab during the undergraduate TA’s office hours. A functioning program showing Parts 1A/1B is required for the checkoff.**

PARTS 2A/2B coming next week.

**Turn-In:**

* No report is due until after next week’s lab.
* Demo
  + Generate and draw maze
  + Draw avatar
  + Move down screen with delay