Lab 8: DS4Maze -- Part 2

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:

As a reminder from last week, the game will:

Lab 8 Part 1

1. Create a random maze of characters on the screen
2. Start your avatar (a single character) at the top center of the screen.
3. Every so often (a delay to be found by you), the avatar will fall one line down the screen.

Lab 8 Part 2

4. Character movement
    a. If the DualShock 4 is tilted right, the avatar will move right.
    b. If the DualShock 4 is tilted left, the avatar will move to the left.
    c. 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.
7. **BONUS 1 - 5 points**: Detect when the character gets stuck and announce a loser when the Avatar is stuck in a position where it cannot move at all.
8. **BONUS 2 - 10 points**: Same as above, except detect when the Avatar is stuck in a position where it can no longer move down, but can move left and right.

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 have already discussed game state (1, 2 and 3 above) in the previous lab and will implement the rules and updating the game state in this lab (4, 5, 6 above).

Process:

Demonstration your work from the prior lab to the undergraduate TA before the beginning of class (in their office hours).

Part 2A:

On the final page is a finite state machine for the event loop process. Think of it as a flow chart where on the arrowed lines, there is a condition that must be met to move forward. This will help you think through your logic for your event loop.

Manually step through the diagram. For each of the conditions (labels on the lines), consider how you will check each in your code.

In your final code, you must comment each point where one of these conditions is checked by adding a relevant label. For instance, where you check to see if the player has won there should be `//Did we win?` .

Part 2B:

Finish your code to implement the game.

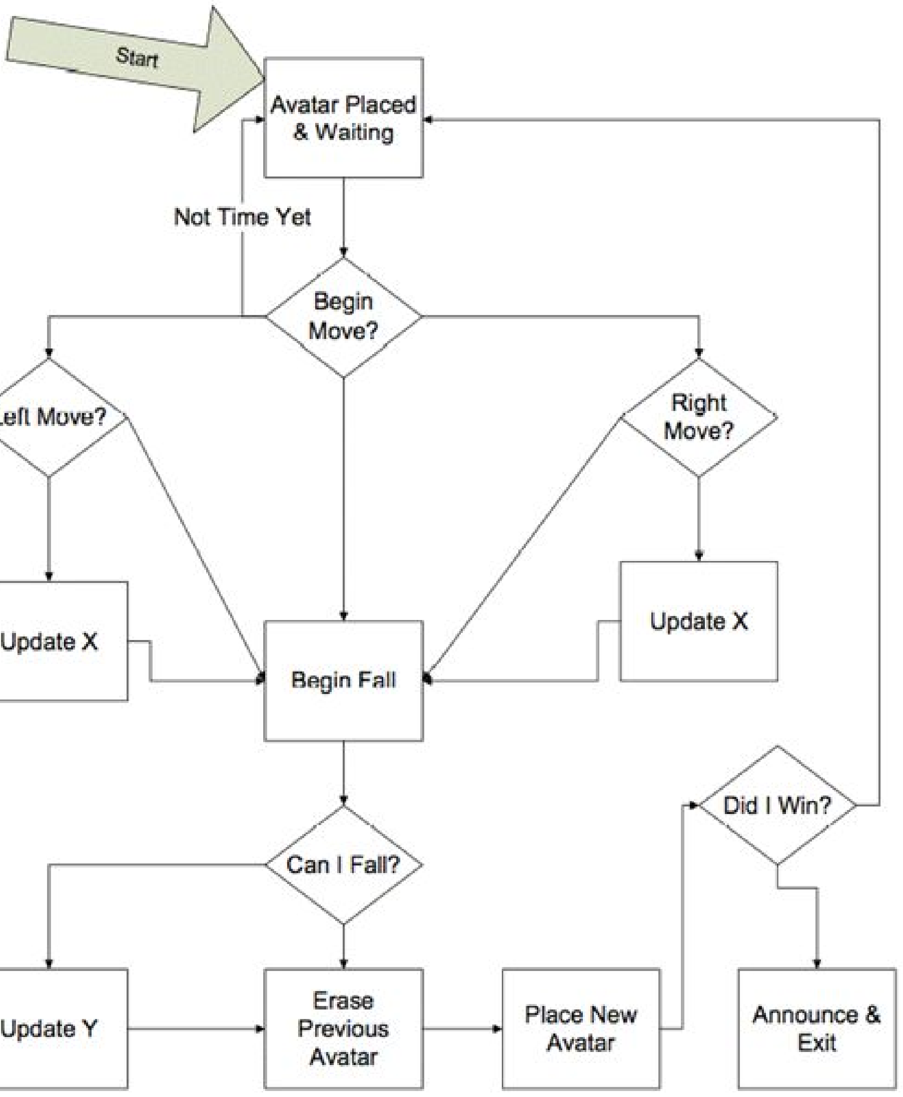
Your undergraduate TA must check it off before the beginning of the following lab session.

Questions and Experiments:

1. In the “Safe to Go RIGHT/LEFT” conditions and the “Can I fall” conditions, document what is checked and how.
2. Describe what would be necessary to check for the player losing the game and how you would add it to the state machine.

Turn-In:

Your lab report including answers to the all questions from part 1 and part 2, as well as all of your source code which has every condition commented. The source code needs to be demonstrated to your undergraduate TA. Don’t forget to explain what would happen for the loser to announced.


Start
Avatar Placed & Waiting
Not Time Yet
Begin Move?
Left Move?
Right Move?
Update X
Update X
Begin Fall
Can I Fall?
Did I Win?
Update Y
Erase Previous Avatar
Place New Avatar
Announce & Exit