CSCI 200 - Fall 2023 Foundational Programming Concepts & Design

Lab 6C - SFML: Maze Drawer

Home HW Sets Schedule Files Help Links

This lab is due by Thursday, December 07, 2023, 11:59 PM.
As with all labs you may, and are encouraged, to pair program a solution to this lab. If you choose to pair program a solution, be sure that you individually understand how to generate the correct solution.

Jump To: **Rubric Submission**

The process of determing if a maze can be solved will be done in two phases. First, we will read in and visualize the maze. Then we will look for a path from start to end (if one exists).

Maze Files

A **maze pack** has been provided with many sample mazes. The format of the maze file is:

```
R C
#####
#S#E#
#...#
#####
```

The first line specifies the size of the maze as `R` rows and `C` columns. Next, an `R` x `C` 2D array of characters follow. The following properties describe the makeup of the character array:

- `#` - signifies a wall. These spaces are unable to be moved to.
- `.` - signifies an open space. These spaces may be moved to.

- `S` - signifies the starting space. You will begin your search from here. This space may be moved to.
- `E` - signifies the end space. You will end your search here. This space may be moved to.
- The maze will always be well-formed. Meaning, it will always be of size `R` x `C` and only contain the four characters specified above.
- `S` and `E` will each always appear exactly one time in the maze.
- The size of the maze will be bound by `3 ≤ R ≤ 102` and `4 ≤ C ≤ 102`.

Loading the Maze

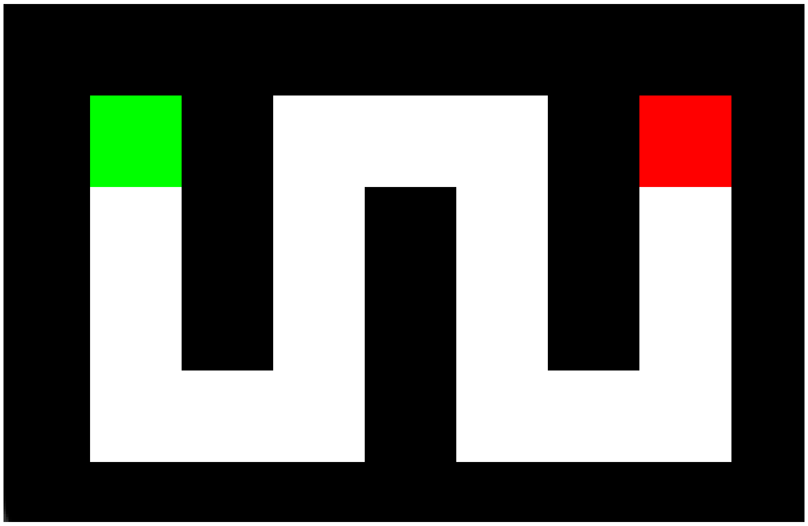
Begin by checking if a command line argument was specified with the filename. If not, prompt the user to enter a file containing the maze. As you read the file, create a 2D list of characters that matches the size of the maze. Read the maze contents into your 2D list. Be sure to note what location the `S` character is in to start your upcoming search.

Drawing the Maze

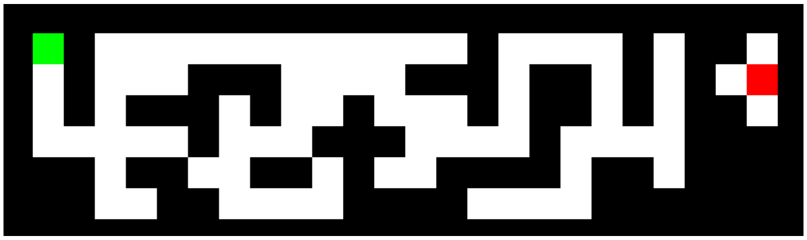
Each space in our maze will be a 15x15 rectangle. Create an SFML window that has a width of `15*C` and height of `15*R`. Inside of the draw loop, loop over every row & column in the maze array. Create a `RectangleShape` that is sized and positioned accordingly. Color the rectangle based on the value in the maze at that position:

- if the cell is `'S'`, then color it Green
- if the cell is `'E'`, then color it Red
- if the cell is `'#'`, then color it Black
- if the cell is `'.'`, then color it White

The visualization of Maze 1 is shown below:



The visualization of Maze 4 is shown below:



Grading Rubric

Your submission will be graded according to the following rubric:

Points	Requirement Description
0.70	Fully meets specifications
0.15	Submitted correctly by Thursday, December 07, 2023, 11:59 PM
0.15	**Best Practices** and **Style Guide** followed
1.00	**Total Points**

Lab Submission

Always, **always**, **ALWAYS** update the header comments at the top of your `main.cpp` file. And if you ever get stuck, remember that there is LOTS of **help** available.

Zip together your `main.cpp`, `Makefile`, `*.h`, `*.cpp`, `*.hpp` files and name the zip file `L6C.zip`. Upload this zip file to Canvas under L6C.

This lab is due by Thursday, December 07, 2023, 11:59 PM.

As with all labs you may, and are encouraged, to pair program a solution to this lab. If you choose to pair program a solution, be sure that you individually understand how to generate the correct solution.

Last Updated: 10/06/23 14:50

Any questions, comments, corrections, or request for use please contact jpaone {at} mines {dot} edu.

[Jump to Top] [Site Map]