Project 1

CS 320

due Mar. 1 at noon

A maze can be looked at as an array of $m \times n$ square cells, where m is the number of rows and n is the number of columns in the maze. Each cell has the property that one to three of its sides (or walls) is present. If two walls are present there is a path into the cell and a path leaving the cell. If three walls are present, the cell is a dead end. It is also possible to have a cell with a single wall. (Note: a maze can be represented by a graph in which each node represents a cell and each edge a connection between cells, i.e., the absence of a wall between adjacent cells.)

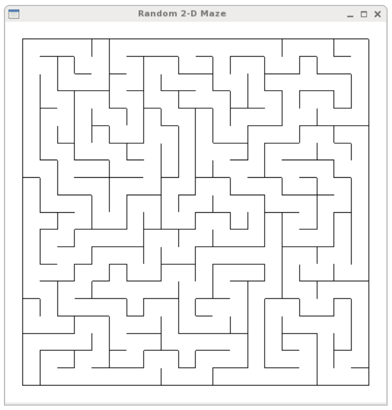

Figure 1: A maze.

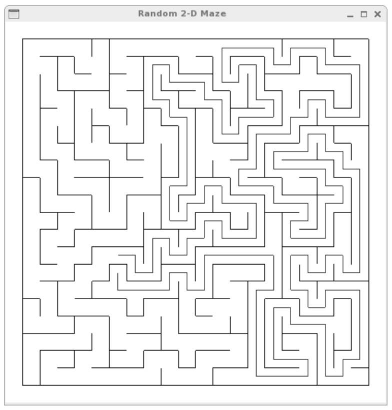

Figure 2: A maze with the path between two cells displayed.

You are to use OpenGL to construct and display a 20×20 maze. The particular type of maze you are to construct is one in which every cell is connected to every other cell and there is exactly one simple path between any pair of cells (a spanning tree).

One method to construct such a maze is to start with an array of cells, each of which has all four walls present. We then remove various walls to construct the desired maze. Consider the following algorithm. Initially all cells are said to be unvisited. We start by selecting (visiting) a random cell. We connect this cell to one of its neighbors by randomly visiting one of its unvisited neighbors recursively. We remove the wall between the two cells to connect them. If any of the other three neighbors has not been visited, we recursively visit them. If we reach a dead end (a cell all of whose neighbors has been visited), we return. When all cells are visited, the first visit call returns and we have a maze. (Note that the algorithm requires a data structure to keep track of which cells have been visited and which have not been visited. This will be discussed in class.)

Once the maze has been drawn, the user can select any two cells with two left mouse clicks. Once the two cells have been selected, the program will then compute and display a path between these two cells. The path should be shown in a color other than the color used to display the maze. Finding the path is another simple, recursive algorithm. Additional pairs of clicks will cause the program to display new paths between the newly-selected cells. (If the total number of clicks is zero or an odd value, no path should be displayed. Otherwise, the path between the last two selected cells should be display.)

A middle mouse click will cause a new maze to be drawn and then the program will again wait for the two left mouse clicks designating the path endpoints, drawing a new path, etc. A right mouse click will exit the program.

If the user reshapes the window, you don't need to match the aspect ratio, but your pick selection code must adjust the the change so that the user can still properly select the cells serving as path endpoints.

Suggested order of work:

1. Display a single maze and exit upon right mouse button click.
2. Using the middle mouse button, display new mazes.
3. Display the path between two selected cells.
4. Adjust to reshape events.

Submitting Your Project

We will follow this procedure for all projects this semester. Your project files are to be emailed to me at kelliher[at]goucher.edu. All source files necessary for building your program (C++ source(s), .h files, vertex shader, and fragment shader) and any documentation files you've created should be sent as a single ZIP archive attachment in a single email. I will build your program from the source files provided and test the resulting program.