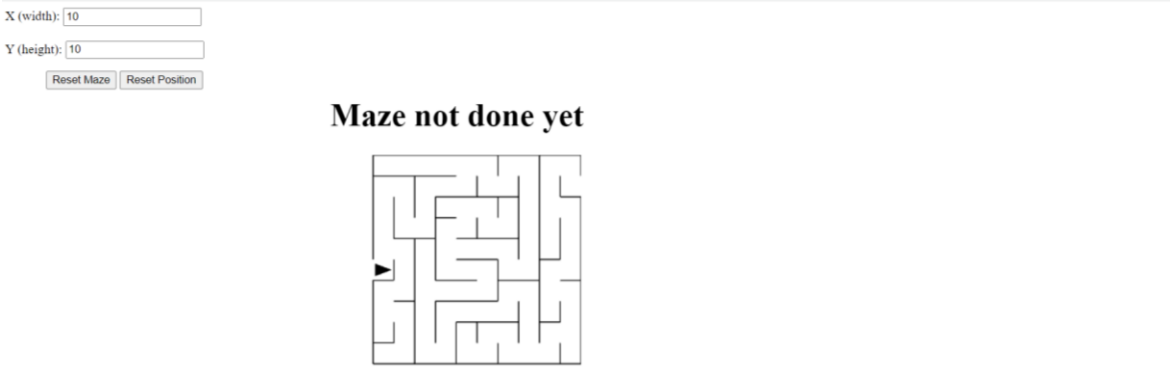
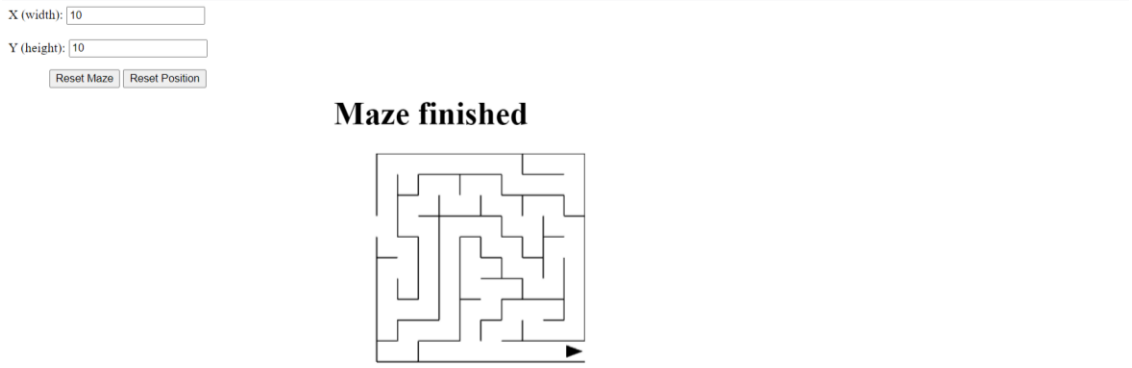


This is an example of a randomly generated maze. The generated maze can have dimensions from 1 to 10 for both x and y. Reset maze will generate a new maze and reset position will return the triangle to the starting position. To move the triangle, use W to move forward, A to turn left, and D to turn right.



The message above the maze will change when the maze is finished, as shown here, and the controls will be disabled until a new maze is generated.

The grid was generated using LINES, with eight points representing a cell. The maze was generated using depth first search, randomly picking a direction to go until there are no available directions. It cannot travel to a cell that has already been visited or out of the boundaries. It will then mark the proper wall as a wall to remove from the MxN grid. Then, as it backtracks, it will remove the position from the stack.

Finally, when the stack is empty, all walls that should be removed are removed from the vertices set, and the vertices set is rendered.

The start and end positions are randomly generated by picking a point from 0 to y and removing the corresponding wall. The start position is added to (0,0) for the triangle's starting position.

The triangle is created by a set of three points which are drawn using TRIANGLE_STRIP after the maze. The code keeps track of its position, and its rotation. A different set of points are used depending on the rotation and those points are added to its position relative to the maze. When W is pressed, the position is changed only if it is not moving into a wall or leaving the boundary. There is also a check for if it is going to cross the exit, and instead of moving it will register that the maze is finished, then after one more input, the program outputs that the maze is finished.

The biggest issues in creating this project were keeping track of what position meant what in the vertices and wall arrays, as well as completing the logic for the depth first search, for example one difficulty was properly defining if a cell was no longer useful and could be removed from the stack. However, I believe I was able to overcome these challenges to produce the intended result.