Project 3

There were a few steps to making this work

1. Come up with a data-structure that can easily represent a maze
    a. I have a MazeCell class that stores whether a cell has been visited, and each edge a cell has. An edge in this case represents a wall between two cells in the graph. Say you have two adjacent cells A and B on the left and right respectively. A wall between these two cells would be represented by A.rightEdge = true and B.leftEdge = true. Removing that wall requires modifying both A and B.
    b. An entire maze is stored as a 2d array of MazeCells
2. Setup WebGL to render lines and a square to represent our character (the rat)
    a. Extract edges from the maze, map those edges to an array of points and use WebGL to draw lines between those points.
    b.
3. Add event listeners for keyboard behavior
    a. Add support for arrow keys as well as W, A, S, D
4. Add controls that allow you to resize (set N and M) the map
    a. The map is regenerated when N or M changes.
    b. Each time the map is regenerated the old map is destroyed removing its event listeners to prevent memory leaks.
5. **FIX anti-aliasing**
    a. I had a lot of lines rendering different thicknesses or disappearing as I resized the window (I'm dynamically sizing my HTML canvas element).
    b. I was able to fix this by setting the canvas height width equal to the canvas' offsetHeight and offsetWidth. I also disabled anti-aliasing since I want my maze to render pixelated (since everything is blocky and I don't have any curved edges).

Possible imporvements

Right now, I'm redrawing the entire map every time the rat moves. I think it would be more efficient to only update the cell the rat previously occupied (removing the white square) and the cell the rat is now on (adding the white square). In practice, I didn't notice any performance issues, so I'm leaving it as is.