Group H

Planning Document

Program #1
Maze Program Version 1.0
Planned on meeting 2 times a week on average. Our github is up and running and on our machines individually.

Tiles: These definitely need their own class to feed into the center and the side panels. Our goal was to get these up and running because the whole project revolves around them. Each one is going to need to be able to be individually examined.

GameBoard: This is the game board which means it's going to need to host all the different objects that we make and show inside the game window. That gridbag is gonna be the thing to host all our tiles in it.

Buttons: No changes this time around.

Program #2
Maze Program version 2.0

Quit Button: Must exit the program, should be relatively straight forward.

Tiles: Time to make these movable, we have a couple ideas going forward, one just redraws the tiles and the other involves making a tile wrapper. We will come up with these ideas on our own then compare which one we like more during the next group meeting.This involves the tiles keeping their identity when moved around. Going to need a mouse listener to pick up the mouse so we can actually use it to select tiles.

Extra: Make tiles known that they are selected, and signify that with something like a colored border. We also need to make a swap function, which is difficult when java is kinda pass by value, this is the reason we will need the tile wrapper. We are also going to try and have everyone try to code each part themselves, and then we will compare all our programs to see if any one or the other is more elegant or has any clever ideas.

Program#3
Maze Program version 3.0

Its drawing time

Default.mze: we have to figure out how to read this file type and get the ints and floats to our tiles. It should be number of tiles then tile number then lines.We need to find a way to pass that data to the tiles once we get it, whether we should read the file then pass it to the tiles, or read the file AS we make the tiles.

Tiles: Tiles are now going to need a drawing function within and keep all those drawn lines on that tile. This means we need to store the line data in the tile class and not in the game board class.

Reset: The reset button needs to function by putting everything back to its starting positions. Now if this is just reading the same file you could just reboot the game with the same text file but if not then it will need to memorize beginning locations. Instead we will likely have each placeholder/wrapper remember it's original tile and reset itself to “hold” that tile when reset is called.

Program # 4
Maze Program 4.0

Illegal move indication: we plan on changing the border color to red when the player attempts to perform an illegal move.

Rotating: Tiles should be fairly straight forward to rotate, its just manipulating the x1,y1,x2,y2 values. We will need a function to get this done but once again it should be fairly easy

Randomization: We will need a way to make the tiles appear on the board at random, this will probably entail storing the data and then shuffling it tile by tile into the board (keeping all the data for each tile together). This way it can be saved for our reset function

Change “\” (backslashes) to forward “/” (forward slashes). -> Main.java line 49.

Button Fix: Double check names don't get cut off for buttons

Double check windows size is (900, 1000)

Actually add the pop up window… :)

Program #5

Make tile selection more obvious.

Save game to .mze file.
Change Button Label from “New Game” -> “File”d
Button will open up a menu (or a drop down)
    (contains Load or Save options)
User can load a previous game’s save with rotation and position intact.

Save can save the games state into a .mze file.
    Including tile positions both on and off board, as well as tile rotation degree

Reset button
    Tiles go back to original position and rotation.
    Resets the “modified attribute”

Clicking Quit / Load results in a pop up menu asking for user to save game
    Assuming they have made modifications (modified attribute)

We need to implement a modified attribute.

Implement pop up error in the event the first four bytes are incorrect.
    Present game window with no maze loaded.
    Give user choice to load to quit.