Version 1

Group: Foxtrot

Maze Program Version: 1.0

Our basic idea for this version is to create the gameboard, the piece spots, and the buttons and to have them all in the correct positions.

Tiles: We are not ready to implement movable tiles, so this will be saved for future versions.

Gameboard: Gridbag layout was difficult to learn, but Buck figured out how to set the pieces in the correct position. Getting these set up and working was simple once we understood the GridBagConstraints function, especially the tile pieces. We also changed the background color to a less vibrant color.

Buttons: Getting the buttons to work was simple enough, as we coded the public class for the buttons, since they will be used throughout the project. The exit button works as intended, but the start and reset button still do nothing other than output to the console. These other buttons will be addressed in future versions.

Sidebars: These were created alongside the Game Board. They are just JPanles right now but will eventually become more and hold the pieces that we need.

Entire Project

Planning for Group Foxtrot:

Create a maze game that has tiles that need to be arranged in a path to win.

Win Condition: To create a win condition for the maze game, a checker needs to be implemented that verifies if the tiles are arranged the correct way for all possible win conditions. This will allow players to know when they have successfully completed the game.

Window Size: Setting a minimum or static window size for the game is important to provide a consistent display experience for players, so that even when resizing it will make sense to the user. This will help ensure that the game can be played on various screen sizes, and accommodate players screen size preferences..

Moving Tiles: The tiles in the game need to be designed to be movable and rotatable within the game board. We will achieve this by creating code that allows the tiles to change positions. We will have to keep track of all current positions of tiles and their rotations, which will eventually be connected with a win checker to determine if everything is in its correct position.

Tile Randomizer: A tile randomizer should be added to the game to create different paths each time it is played. We will do this by having a class that randomizes the rotation of each individual tile piece and which position in the “sidebar” it will be located. It should make sure that every new game looks different to the user.

Win Screen: The win screen should be designed to display when the player has successfully arranged the tiles in the correct path. This will provide a visual confirmation of the player's success and create a sense of accomplishment for them. It will allow them to exit the game or start a new game if they please.

UML Diagram: The UML should be reviewed and be updated as necessary to ensure it accurately reflects the design and implementation of the game for each step of the way.

Bug Fixes: Regular bug fixes should be performed to maintain the stability and functionality of the game. This will ensure that players have a smooth experience while playing the game and that the code looks nice, readable, and maintains Object Oriented principles.