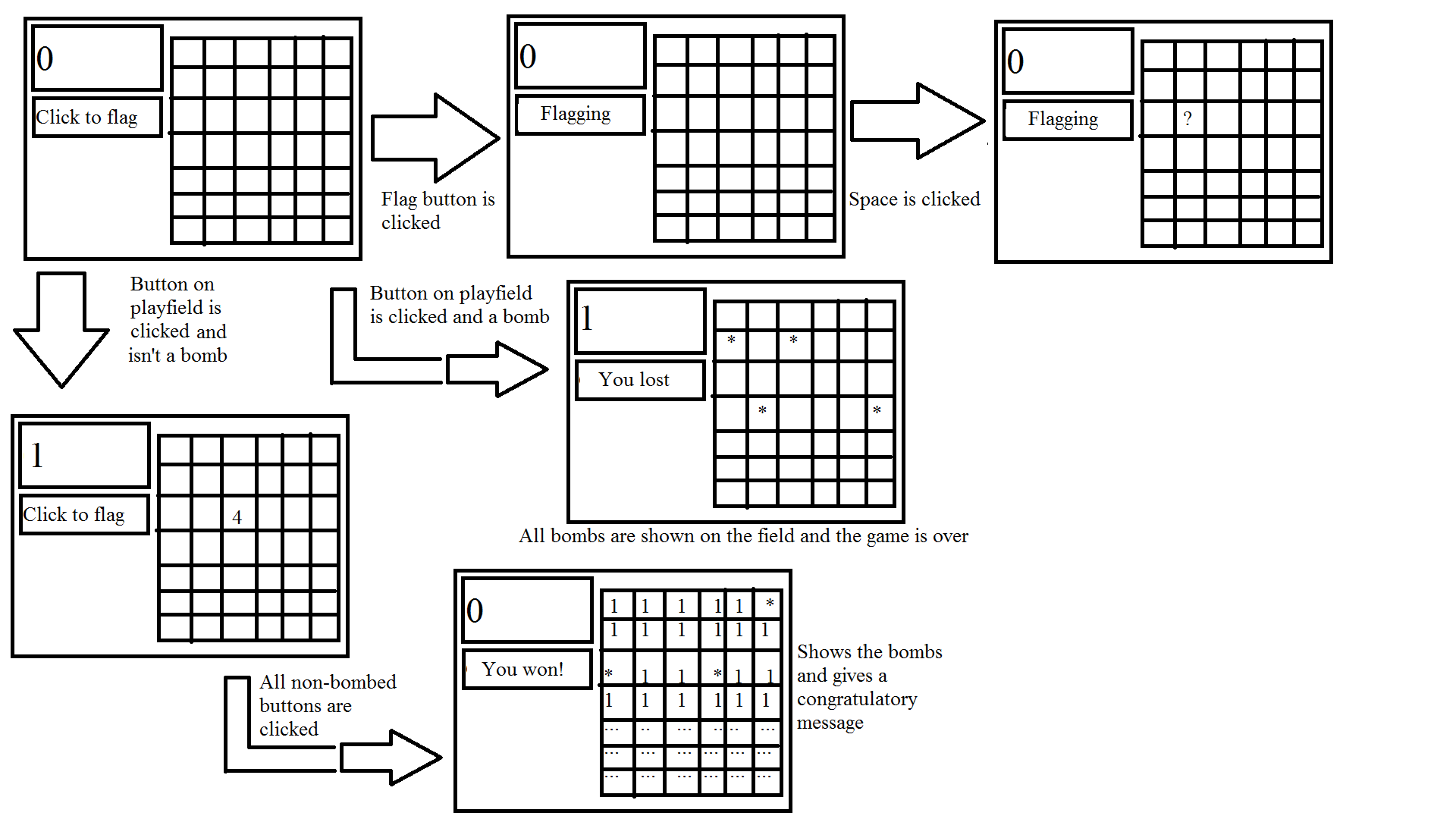
Evan Wilt

4/20/16

Yet Another Minesweeper Clone

**Project Summary:** The title for my project is “Yet Another Minesweeper Clone”, as this is a remake of the original minesweeper game in Java. It uses a GUI, has a turn counter to show how many places the player has clicked, and a flag button to mark potential bombs. The gameplay is a nearly identical remake of minesweeper, where bombs are randomly placed and number within each button indicates how many bombs are within its immediate area. The intended user is anyone interested in playing a game of minesweeper. There isn’t really a problem that this program is trying to solve. Rather, it was a test of my ability and a way to try out higher-leveled concepts within Computer Science such as recursion as well as data handling using ArrayLists. For this project, I used a GUI made with Swift, recursion, OOP, and ArrayLists that contain the bombs on the playfield and the buttons that make up the playfield.

**Use Case Analysis and UI design:** This photo illustrates the use cases of the program and what the UI looks like.



**Data design:** The data for this project is mostly stored in two ArrayLists, one that holds bombs, and one that holds the buttons on the playfield. The bombs are given a random X and Y coordinate within the map size (that don’t overlap with one another). When a space is clicked, if it is a bomb, the user will get a game over. If it isn’t a bomb, a method checks all surrounding spaces to see how many bombs are in the area. I used OOP and created three different classes, one for bombs, one for buttons on the playfield, and one for the GUI which acts as the main. The GUI uses a Grid Layout which because it scales well when you resize the window. It also makes scaling the game much easier because I can just set a global variable for the X and Y size and use those for the Grid Layout dimensions, so it doesn’t matter if there’s 100 buttons or 1000, it will always scale. I’ve made a UML that shows exactly what classes have what variables and methods, which will be included with this document.

**Algorithm:** There is a class for the GUI, a class for the buttons, and a class for the bombs. The GUI creates the layout and any extra parts such as the turn counter and the flag button. It also acts as the “main” class for this project. The button class acts as a button on the playfield and has several variables to identify if it’s flagged, it has a bomb on it, and more. The bomb class takes an X and Y axis to set the bombs within the field. The initializer sets the bombs at (0,0) and when the bombs are created, they’re given random coordinates that fit within the playing field. The coordinates also don’t overlap because if they did, there wouldn’t be enough bombs on the field and the buttons might say there’s more bombs near them then there really are. The most important part of Minesweeper is the algorithm that tells the program to check until every edge has at least one bomb near it when the number of bombs on the button you clicked is zero. The algorithm looks like this:

If the player isn’t flagging and the button isn’t hidden or flagged…

Call the turnCounter method.

Check for a loss or win.

If the game isn’t over, check every space near the button clicked for a bomb and add the number found to nearby bombs.

If there’s no nearby bombs, check run the method on every nearby button that hasn’t already been checked.

If there are nearby bombs, show the number of them on the button clicked and mark it as checked for bombs.

If the player is flagging, mark the button clicked with a question mark if it isn’t flagged or make the button blank if it is currently flagged.

I also mentioned that buttons could be hidden. This is a little bit of a trick to hide those nasty edge cases that could cause a problem for the algorithm. A method called hideEdges checks for buttons with an X and/or Y value at the min or the max and hides them. It looks a little odd on the GUI since it makes some whitespace, but this isn’t a GUI design class so I don’t mind. I also have some other methods including one that shows all the bombs on the playfield, adds bombs to the playfield, and a counter of all the buttons on the playfield which will decrease as buttons are pressed and once it hits zero, the player has won.