Minesweeper Design Manual

I. Introduction

The system utilizes the MVC design pattern to enable a user to play Minesweeper on a graphical user interface. The program revolves around the creation of Cell objects which hold either a bombs or a numerical value based on the number of surrounding bombs. The model generates the board and is in charge of the logic behind gameplay. The view creates a graphical user interface, integrating a top pane and a board of squares. Each square in the board is a visual representation of a cell object, so when the user left or right clicks different Cell methods are called. The top pane contains options for users to select their desired preferences, like the color scheme and difficulty level. The controller binds the model and view together, allowing the user to interact with the visuals to actually play the game. We also took into consideration different types of users and their potential motives to play Minesweeper. This encouraged us to include other features like a timer, multiple challenge levels brought out through different board dimensions, and different color schemes.

II. User Stories

Our project has been inspired by four key user stories: the first time player, the personal player, the bored player, and the social player. While each type of user has different wants and needs for their gameplay experience, we were able to prioritize tasks that covered their shared needs in creating this project. Overall, their shared need was simply having a game that they could play that mimics the classic Minesweeper game. Once this need was met in the initial sprints, we focussed on more individual needs to allow our game to be enjoyed by a variety of people.

1. First Time Player

   A first time player is one who is interested in playing the game but is unsure of the rules or how to start playing. Minesweeper is often not self-explanatory and we found that many existing Minesweeper games lack instructions, leaving users with no prior knowledge of the game struggling. To aid users that fit this description we included an instructions feature that outlines the goal of the game and how to play. The player can

hover over the question mark button at the top of the screen to easily access the instructions.

2. Personal Player

    A personal player is a user who wants to play the game for their own enjoyment or growth. For example, a busy person who finds brain-teasers fun may be interested in playing Minesweeper, as it is quick, fun, and challenging. To improve this type of user's experience we included a feature that allows them to change the level of difficulty. If the user wants something more challenging, they can select the hard level, but if they want something quick and fun, they can select the easy or regular difficulty levels.

3. Bored Player

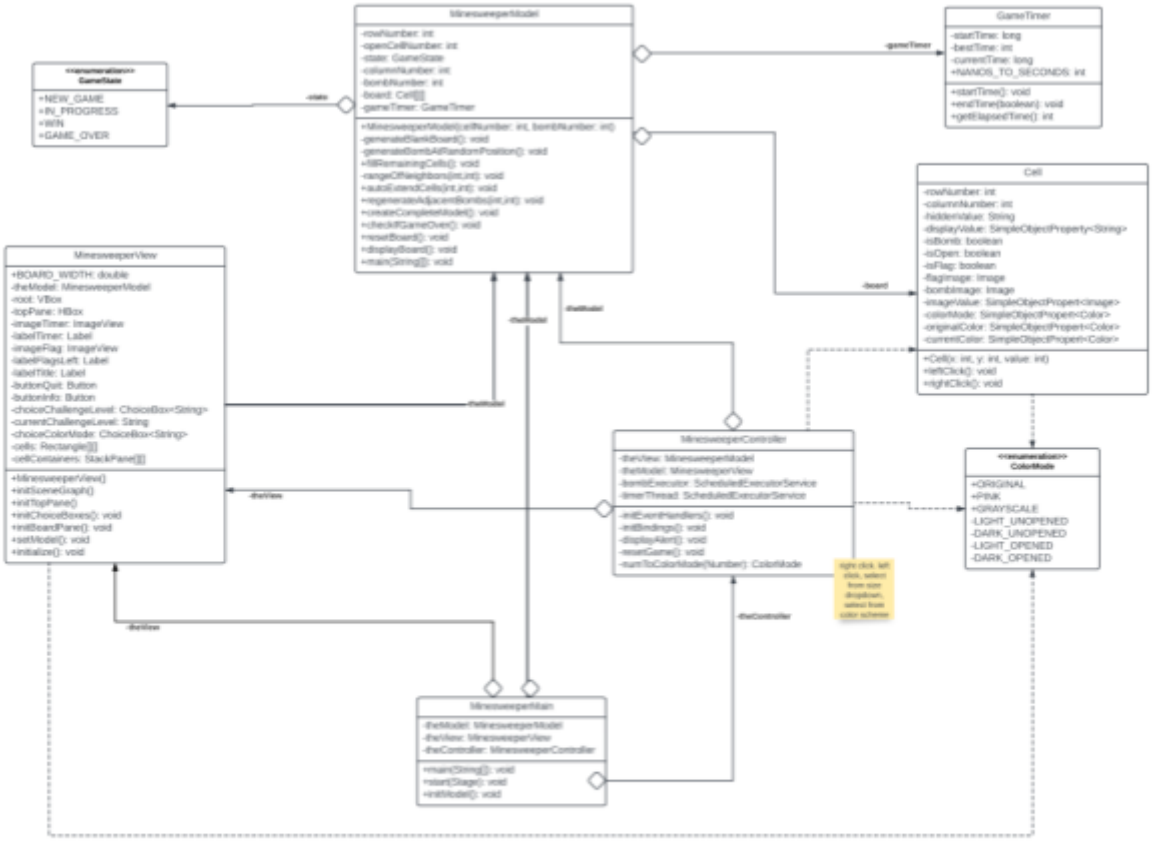
    A bored player is one who wants to play the game for entertainment. A simple game is not something that would peak their interest or satisfy their boredom. To attract bored players to our game we included a variety of fun graphics to draw them in and keep them playing. We added a feature that allows the user to change the color scheme of the board, so the user can choose a theme they like the best or even change it each time they play. We also included graphics for the flags and bombs, making gameplay more fun and easier to decipher what is happening on the screen.

4. Social Player

    A social player is a user who likes to play games with other people. They enjoy friendly-competition and sharing their best scores with their friends. While Minesweeper is a single-player game, we were able to include features to make it more competitive so it can be played in a social setting. We added a time feature so the user can track their best time and challenge their friends to beat their fastest time to finish the game. The social player may also enjoy the different levels as they can not only challenge their friends to see who can complete the regular board the fastest, but also who has the best times for the easy and hard boards.

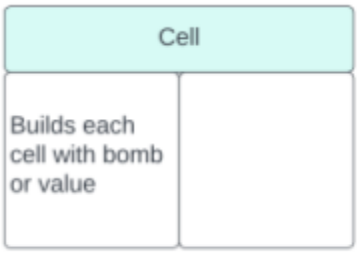
III. Object Oriented Design

Our design involves six classes that implement different aspects of the system, but when combined we get a complete, working Minesweeper game. The classes revolve around the creation of cell objects which contain attributes and methods that are essential for gameplay. We included MinesweeperModel, MinesweeperView, and MinesweeperController classes, executing the MVC design pattern. We also have classes called MinesweeperMain, Cell, and GameTimer adding to the functionality of the MVC classes and meeting the needs of different users. Figure 1 displays the connections and dependencies between all of these classes.

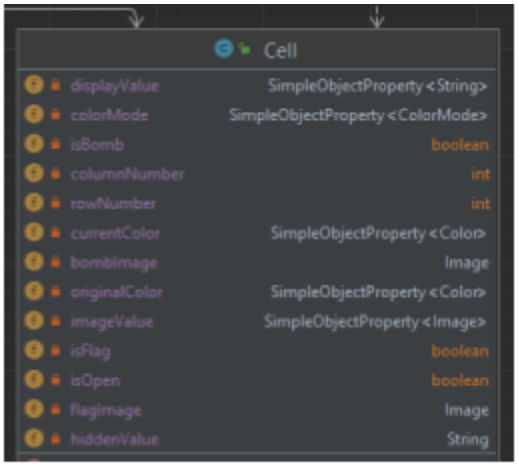


(Figure 1) The UML Diagram exhibiting the connections between all of the classes in the system.

Cell (Figure 2/3) is a class that allows the creation of the smallest unit of the game. The Cell constructor creates cell objects that each hold a bomb or numerical value. They also hold a row and column value, representing their location on the board, as well as their color and an image if the cell contains a bomb or flag. The Cell class is independent as it relies on none of the other classes to exist, however it is a key component of the model class.



(Figure 2) CRC diagram for the Cell class



(Figure 3) The fields of the Cell class.

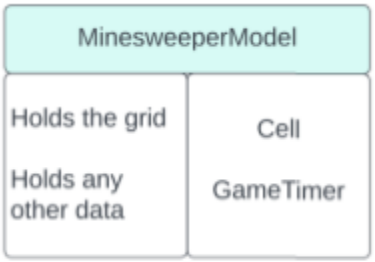
The GameTimer (Figure 4/5) is also an independent class that creates a GameTimer object which mimics a real life stopwatch. It stores the current time on a timer and the best time. The object also has methods that can start and end the timer. This class is used in the model class to create an object representing a timer that is used to track the elapsed time from when the user makes their first click on the board, until they win or lose the game.

(Figure 4) CRC Card for GameTimer

(Figure 5) Select fields and methods of the GameTimer class.

The MinesweeperModel class (Figure 6/7) is responsible for creating a board of a certain size, storing the information about the interface options selected by the user, and the basic logic behind the gameplay. This class has an aggregate relationship with the Cell class as it creates a board of cell objects which each contain their own values. Within the model, a certain number of cells are randomly assigned bombs based on the difficulty level, with easy having the fewest number of bombs and hard having the most. The remaining cells are given a numerical value depending on the number of bombs they are touching. The model also determines when the game

is complete, and additionally collaborates with the GameTimer class to create the timer used during gameplay.



(Figure 6) CRC Card for the MinesweeperModel class

(Figure 7) Select fields and methods of the MinesweeperModel

The MinesweeperView class (Figure 8/9) is responsible for initializing all of the nodes on the scene graph to create a visually appealing graphical user interface. FXML was used to design and implement the different features of the scene graph, including the top pane and the board contained in a VBox root. Included within the topPane are dropdowns, buttons, and labels that allow the user to choose their gameplay preferences. These features rely on collaboration with the MinesweeperController or the MinesweeperModel to function.

(Figure 8) CRC Card for the MinesweeperView class

(Figure 9) The three main components of the MinesweeperView

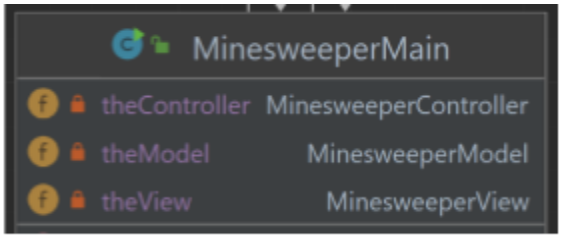
The MinesweeperController class (Figure 10/11) is in charge of how the user's interactions with the game affect the application. This class contains the event handlers for right and left clicks on cells, user selections on the drop-down menus, the quit button, and all hovering features like the color changes of the cells and the instructions tooltip. It also has the resetGame method which resets all aspects of the game when the user clicks the play again button. The MinesweeperController collaborates with the MinesweeperModel and the MinesweeperView as it contains the bindings between the game's logic and the GUI.

(Figure 10) CRC Card for the MinesweeperController

(Figure 11) Key methods of the MinesweeperController

MinesweeperMain (Figure 12/13) is the class where all of the classes come together to run a cohesive game. This class loads in the model, view, and controller in the start method. When the class is run, a window pops up with the complete working game.

(Figure 12) CRC Card for the MinesweeperMain class



(Figure 13) MinesweeperMain brings together the model, view, and controller