

MineSweeper 3D

Isaiah Martinez
CSUN
Computer Science Department
isaiah.martinez.891@my.csun.edu

03/08/2024

Contents

1	Change History	2
2	Introduction	4
2.1	Background Information	4
2.2	Project Information	4
2.3	Game Information	5
3	Design Goals	6
4	System Behavior	6
5	Logical View	7
5.1	High-Level Design (Scripts)	7
5.2	Mid-Level Design	8
5.3	Detailed Class Design	9
6	Process View	10
7	Development View	10
8	Physical View	10
9	Use Case View	10
10	Reflection	10

1 Change History

Version: 1.00.00

Modifier: Isaiah Martinez

Date: 03/08/24

Description of Change: Build Project and released on Github. Documentation added. Updated Readme with instructions for installation and how to play.

Version: 0.94.35

Modifier: Isaiah Martinez

Date: 03/08/24

Description of Change: Deleted unused sound assets from unity store. Added music to game. Added Sound mechanics. Added ability to adjust volume within menu.

Version: 0.88.80

Modifier: Isaiah Martinez

Date: 03/08/24

Description of Change: Added free sound assets from the unity store.

Version: 0.83.25

Modifier: Isaiah Martinez

Date: 03/08/24

Description of Change: Adjusted bomb creation mechanics and assignment. Deleted stray logs to console.

Version: 0.77.70

Modifier: Isaiah Martinez

Date: 03/07/24

Description of Change: Menu connectivity introduced. Quit button now functional. Mouse controls camera movement.

Version: 0.72.15

Modifier: Isaiah Martinez

Date: 03/07/24

Description of Change: Added new panels representing several menus such as Pause Menu, Settings Menu, and Post-Game Menu. Displayed timer post-game completion.

Version: 0.66.60

Modifier: Isaiah Martinez

Date: 03/07/24

Description of Change: Events added to Stopwatch. Changes affected Game Mechanics, Stopwatch, and UI Manager Script.

Version: 0.61.05

Modifier: Isaiah Martinez

Date: 03/07/24

Description of Change: Stopwatch added.

Version: 0.55.50
Modifier: Isaiah Martinez
Date: 2/21/24
Description of Change: Added a win condition to game logic. Adjusted Post-Game logic to accomodate the different states of the game finishing.

Version: 0.49.95
Modifier: Isaiah Martinez
Date: 03/06/24
Description of Change: Updated design for the main menu.

Version: 0.44.40
Modifier: Isaiah Martinez
Date: 03/06/24
Description of Change: Reviewed changes made to different menus. Removed unnecessary whitespace within the scripts.

Version: 0.38.85
Modifier: Isaiah Martinez
Date: 03/06/24
Description of Change: Added Neighbor Revealing Logic when a 0 is discovered. Added ability to lose game. Added bomb locations upon creation.

Version: 0.33.30
Modifier: Isaiah Martinez
Date: 03/06/24
Description of Change: Added Flag Mechanics to game. Added 3D Flag Elements to the Cell Prefab.

Version: 0.27.75
Modifier: Isaiah Martinez
Date: 03/05/24
Description of Change: Modified the way the cells for the board were developed which utilized updated Cell Prefab.

Version: 0.22.20
Modifier: Isaiah Martinez
Date: 2/28/24
Description of Change: Added ability to update size of board when clicking the different buttons on the main menu. Connected GameMechanics and UI Manager Script together via events. Now able to hide main menu when creating the board.

Version: 0.16.65
Modifier: Isaiah Martinez
Date: 2/23/24
Description of Change: Added UI Manager Script to handle events with the menus.

Version: 0.11.10
Modifier: Isaiah Martinez
Date: 2/21/24
Description of Change: Created Outline of Prefab used for Cells within board. Added Basic Input Script to allow for camera movement with key presses. Added Game Mechanics Script to control the construction of the board. Added Cell Logic Script to describe the cells within the board.

Version: 0.05.55
Modifier: Isaiah Martinez
Date: 2/21/24
Description of Change: Made Git repository and added Readme

2 Introduction

2.1 Background Information

This document describes the architecture, design, and implementation for the popular game MineSweeper. See [Here](#) for more information. The project was done in Unity and utilized the usage of several scripts as well as assets which are acknowledged within the Acknowledgements section at the end of this document.

The title for this project: **MineSweeper 3D** is named so due to the project utilizing 3D elements to construct the game. Each cell within the visible board is a 3D element which casts shadows and reflects light. Not all elements, however, within the game utilize 3D structures, e.g., the menus.

This project offers players a new take on the classic game of MineSweeper by expanding into the 3rd dimension.

2.2 Project Information

This project addresses the interests of the major stakeholders including:

- The professor — wants a project that utilizes modern software development techniques, game design strategies, and smart application of the ideas discussed in class. Additionally, there is the desire for quality code design, precise implementation, and unambiguous documentation leading into the fulfillment of Software Development Tasks such as usability, reliability, etc.
- Players of the game — they want a functioning product with thoughtful insight into the design and playability of the game. The game should be as bug-free as possible. Furthermore, the game should be fun.

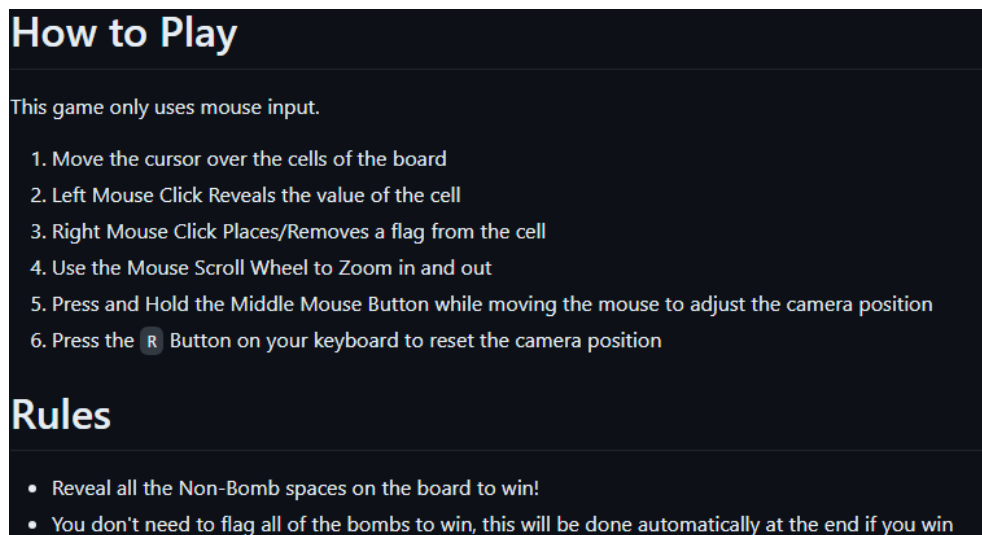
- The Developer — wants the code to be properly structured, commented, and legible. On top of maintaining high quality code, the elements within the various parts of the project not in code (those within the Unity Design Window) should also be held to the same caliber.

The design of this project is highly complicated and necessitates the documentation to describe the project from a variety of perspectives:

1. Logical View — components accompanied by their operations and attributes. Additionally, the relationships between the components amongst other components.
2. Process View — the processes that handle execution for components described in the Logical View
3. Development View — the larger scale view of the project including those described in the Process and Logical Views.
4. Use Case View — describe the end user interactions with the system. These serve as both a guideline for project construction as well as defining the user requirements.

2.3 Game Information

The Rules to play this game can be seen below in Figure 1. These are located on the publicly visible Github Repository seen here. Also located on the Github Repository is a ReadMe file which describes the process for downloading and playing the game for oneself.



How to Play

This game only uses mouse input.

1. Move the cursor over the cells of the board
2. Left Mouse Click Reveals the value of the cell
3. Right Mouse Click Places/Removes a flag from the cell
4. Use the Mouse Scroll Wheel to Zoom in and out
5. Press and Hold the Middle Mouse Button while moving the mouse to adjust the camera position
6. Press the **R** Button on your keyboard to reset the camera position

Rules

- Reveal all the Non-Bomb spaces on the board to win!
- You don't need to flag all of the bombs to win, this will be done automatically at the end if you win

Figure 1: The rules of the game and how to play.

3 Design Goals

The design priorities for the game are as follows:

- The design should be efficient in space complexity. Additionally, the design should aim to be as simplistic, yet effective as possible. This is to reduce design complexity and maximize effort put forth in development.
- This project was developed by one person, so understanding the limitations of my own abilities was paramount when designing and implementing the game.
- Utilize the code given as an outline from the professor to further enhance the quality of the project.

4 System Behavior

The architecture description provided in this section describes the flow of the different menus for the game once started.

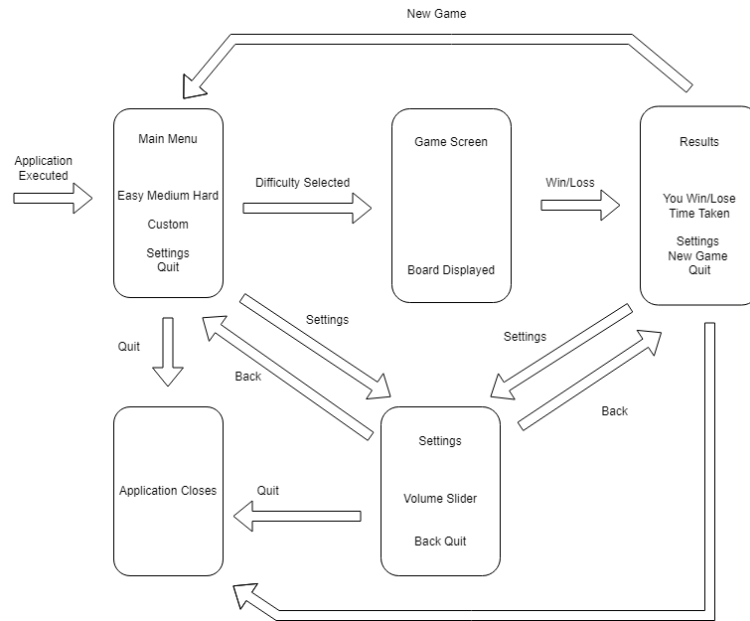


Figure 2: State Diagram describing the System Behavior.

As seen in Figure 2, the game begins when the .exe file is executed. The user is then presented with the Main Menu which allows for the user to choose a difficulty, as well as switch to the settings menu, and also to quit the game. From the Settings menu, a volume slider is displayed, and the user is able to go back to the previous page, either the Main Menu or the Post-Game Menu AKA Results Menu. Should the user have selected a difficulty, then the Game Screen appears with the board with given dimensions based on the difficulty selected.

After the user has won or lost the game, the Post-Game Menu appears with a display of the time taken, a message indicating the result of the game: win/loss, a button to the Settings menu, a button connecting to the Main Menu, and a Quit button. When the Quit button is pressed, the game closes.

5 Logical View

In this section, I will describe the system from a high-level perspective, and get more detailed through each subsection until the entire system is described.

5.1 High-Level Design (Scripts)

The high-level view consists of a series of 6 scripts that communicate via events. Some utilize publicly declared variables to handle logic which will be described later in subsection 5.2.

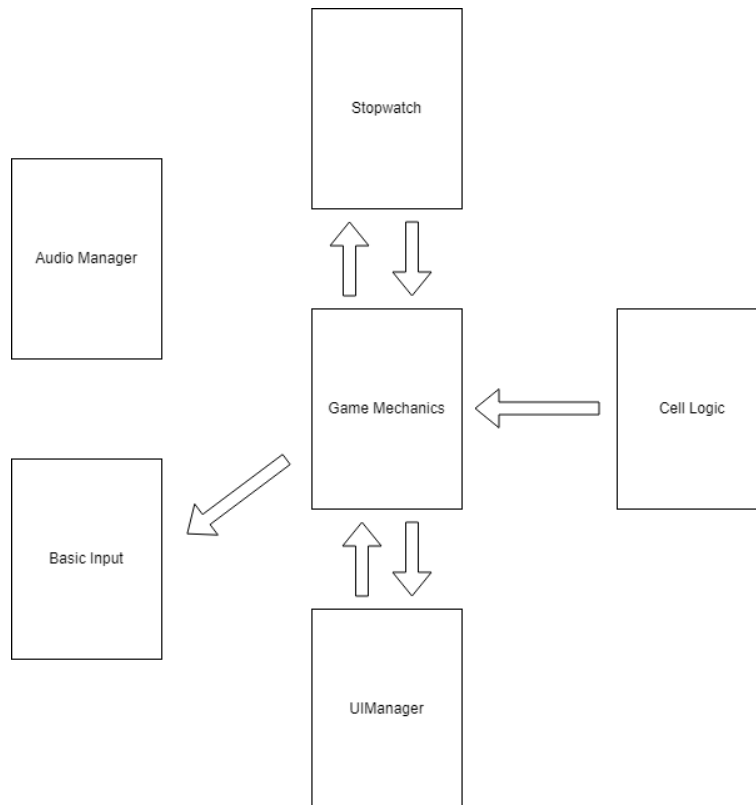


Figure 3: State Diagram describing the connectivity of the scripts.

- The Audio Manager Script operates alone and handles volume control of the entire game.
- The Basic Input Script allows for the player to adjust the camera position on the game screen. It relies on a variable stored within Game Mechanics

Script describing the player being in a game or not. This is because the player should only be allowed to move the camera when they are playing a game, not on a menu.

- The Stopwatch Script contains the logic used to maintain a stopwatch for the player when they enter the game and are playing. The Stopwatch only starts when the game is started, and is returned when the game is finished. This information gets passed to the Game Mechanics Script.
- The Game Mechanics Script is the heavy lifter for the entire game. This handles board generation and utilizes events from other scripts to operate, whether it be screen/menu management, game state, board creation/deletion, and more.
- The UI Manager Script handles logic for button presses and events for menu switching.
- The Cell Logic Script holds the information necessary for the cells to be used within the board created within the Game Mechanics Script.

5.2 Mid-Level Design

The mid-level shows the functionality of each of the scripts and how they connect together.

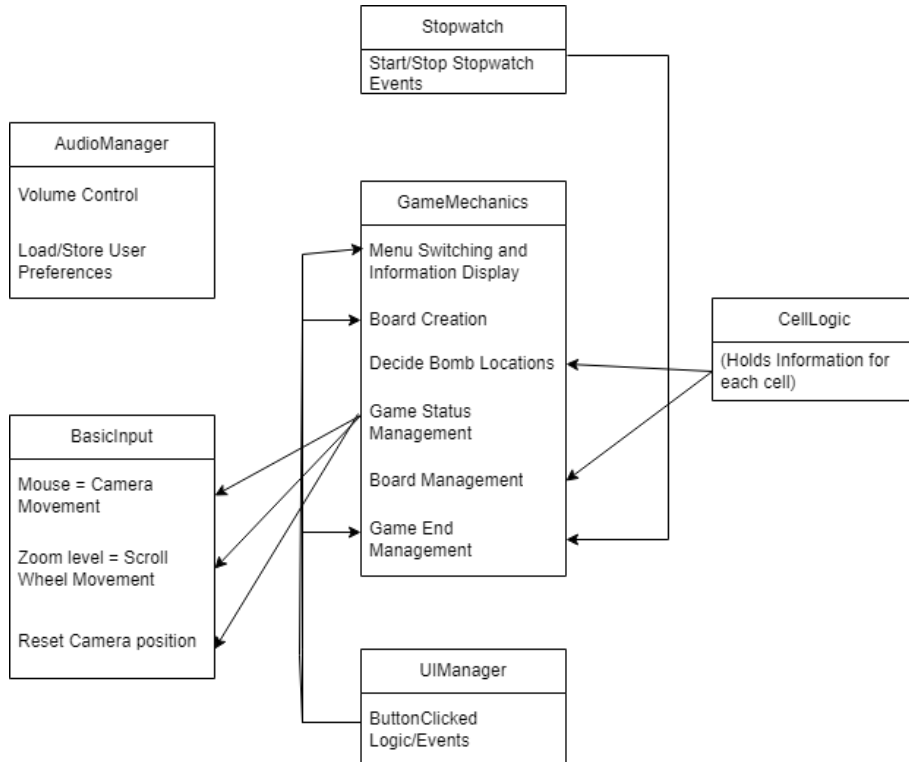


Figure 4: UML Diagram describing Mid-Level connectivity between scripts.

5.3 Detailed Class Design

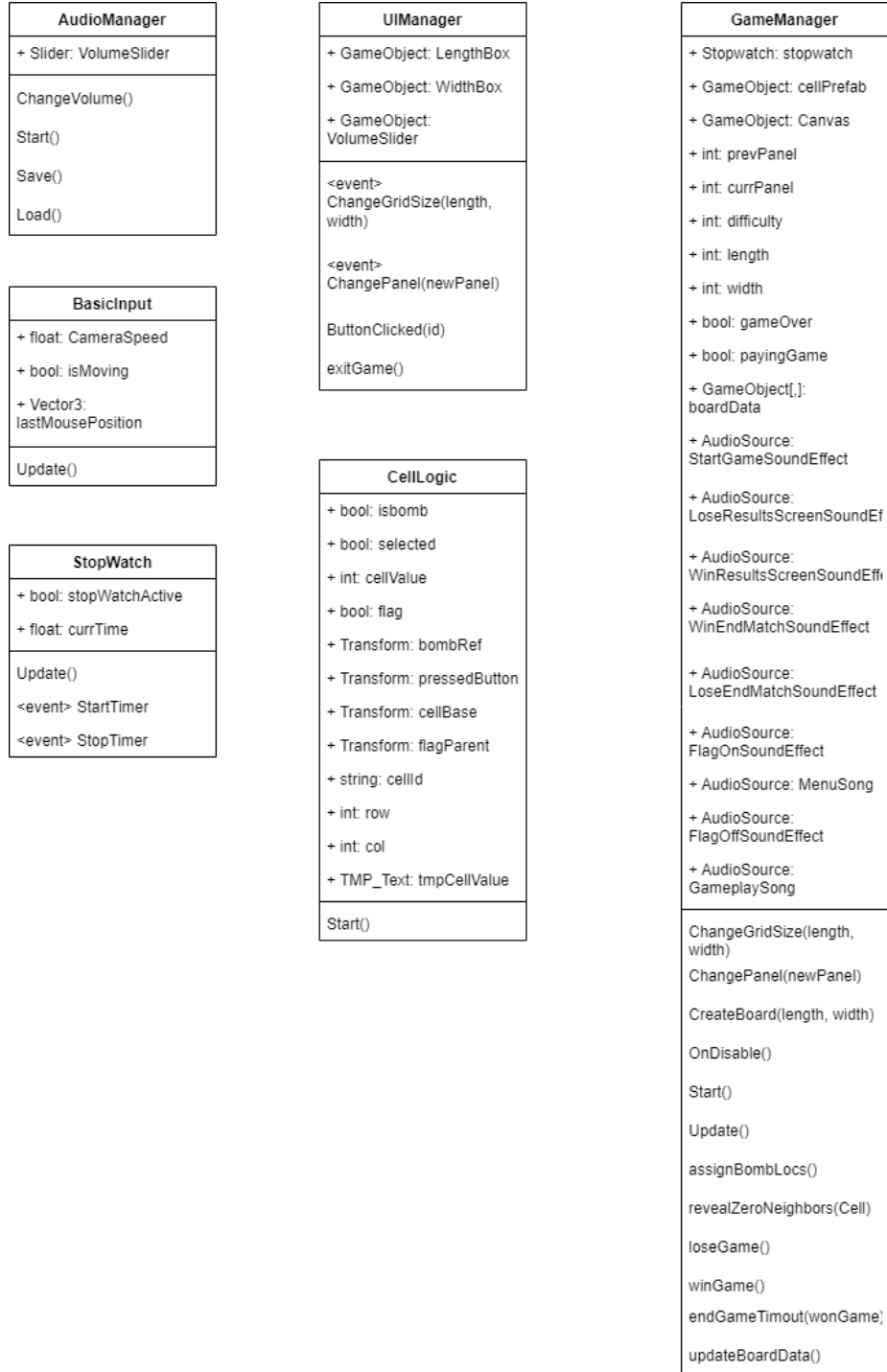


Figure 5: UML Class Diagram for each script.

6 Process View

The threads of control in the application are depicted within the following figure.

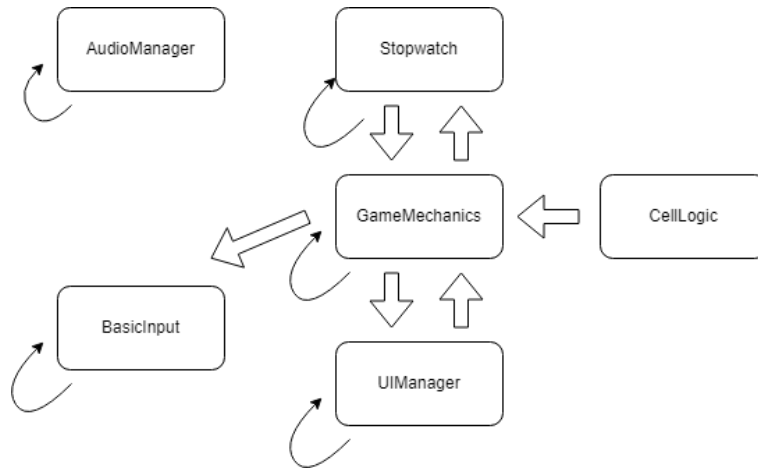


Figure 6: The threads of control for each of the scripts.

7 Development View

8 Physical View

9 Use Case View

10 Reflection