Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CSU44012 Topics in Functional Programming

Assignment 2 Mine Sweeper Game

Minjuan Luo

20313326

1. High-Level Design Choices

The following high-level design choices were made to align with the functional paradigm and to effectively meet the project deliverables:

1.1 Core Game Logic (Mine.hs)

1.1.1 Data Types: Defined custom types like MineElement, Display, BoardCell, Result, and Board to represent the game state robustly.

1.1.2 Game State Management: Utilized pure functions to transform the game state, ensuring immutability and predictable behavior.

1.1.3 Game Actions: Implemented sweepingMines for uncovering cells, and markingMines for flagging potential mines, serving as the primary interactions for the player.

1.1.4 Endgame Detection: The isWin function rigorously checks for winning conditions, and revealAllMines in a loss scenario ensures a clear endgame state is presented.

1.2 Randomization Utility (Lib.hs)

1.2.1 Random Distribution of Mines: Used sampleN, leveraging the Fisher-Yates shuffle algorithm implemented in fisherYatesShuffle, to distribute mines randomly and fairly at the start of each game.

1.2.2 Array Manipulation: swapElements aided the shuffling process by swapping elements in-place, a key aspect in achieving an unbiased mine distribution.

1.3 User Interface and Interaction (Main.hs)

1.3.1 Threepenny GUI Integration: Leveraged Threepenny GUI library to create a user interface, tying the pure logic with a graphical representation.

1.3.2 Event Handling: Managed user events like button clicks and cell selections, translating them into game actions using functions from Mine.hs.

1.3.3 Board Rendering: Developed boardToSVG to draw the board state in a visual format, updating the display based on user interactions.

2. Project Deliverables Completion

The project was successfully completed in two phases:

2.1 Phase 1 Deliverables:

2.1.1 Random Mine Distribution: Achieved through the initBoard function which calls sampleN to place mines on the board at random locations.

2.1.2 Interactive Gameplay: Ensured by integrating sweepOrFlag and autoMove functions from Mine.hs into the UI event handlers in Main.hs, allowing the user to actively play the game.

2.1.3 Endgame Detection: The game state is checked after each move through the isWin function and revealAllMines is used to expose all mines at the end of the game.

2.2 Phase 2 Deliverables:

2.2.1 Safe Move Play: Implemented an 'auto play' feature that uses autoMove to determine a safe move based on the current board state. This function smartly utilizes saveMove and

dangerousMove to evaluate the safety of potential moves.
2.2.2 Heuristic for Risky Moves: When a safe move is not obvious, dangerousMove uses a heuristic approach to calculate the least risky move, thereby enhancing the AI's decision-making process.

3. Reflection on the Implementation of the Code Logic

The implementation of the Minesweeper game logic in Haskell was particularly facilitated by the language's features that support functional programming.

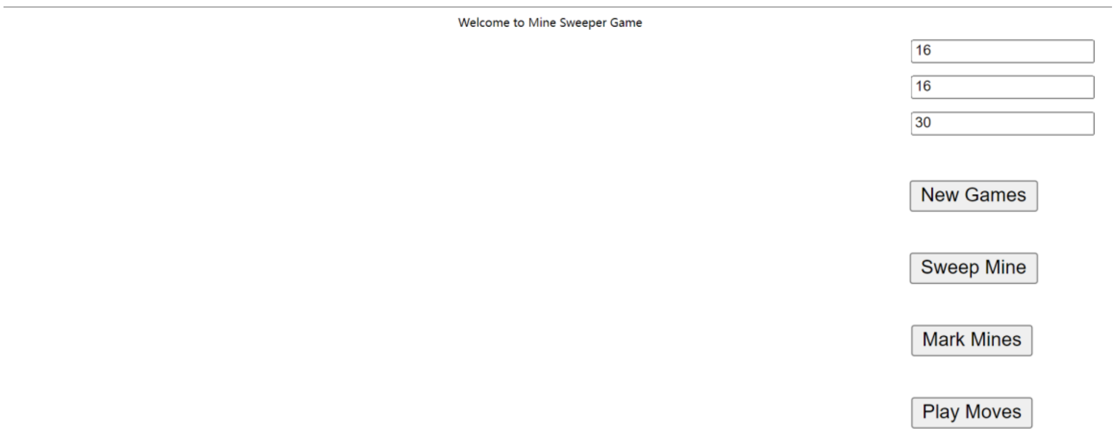
Haskell's pattern matching was instrumental in simplifying the logic within functions like sweepingMines, where different cases for the content of a cell could be easily differentiated and handled. Recursive patterns were expressed in functions like expandEmptyCells, allowing the "zero" rule in Minesweeper, where clicking an empty cell reveals adjacent cells recursively, to be implemented succinctly.

Laziness in Haskell posed interesting challenges, especially in ensuring that the sequence of game state updates occurred correctly. The debugging process, although sometimes counterintuitive due to lazy evaluation, was greatly aided by the use of trace for inspecting function evaluations.

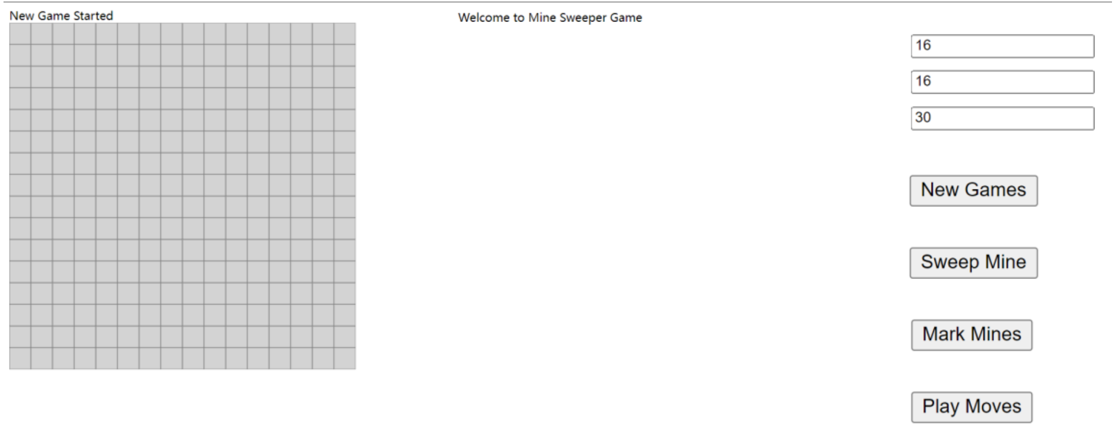
Overall, Haskell's features not only made the code more reliable but also enhanced the development experience. Haskell’s expressiveness and the safety of its type system made it an ideal choice when programming.

4. Demo

4.1 initialization



4.2 create new game


New Game Started

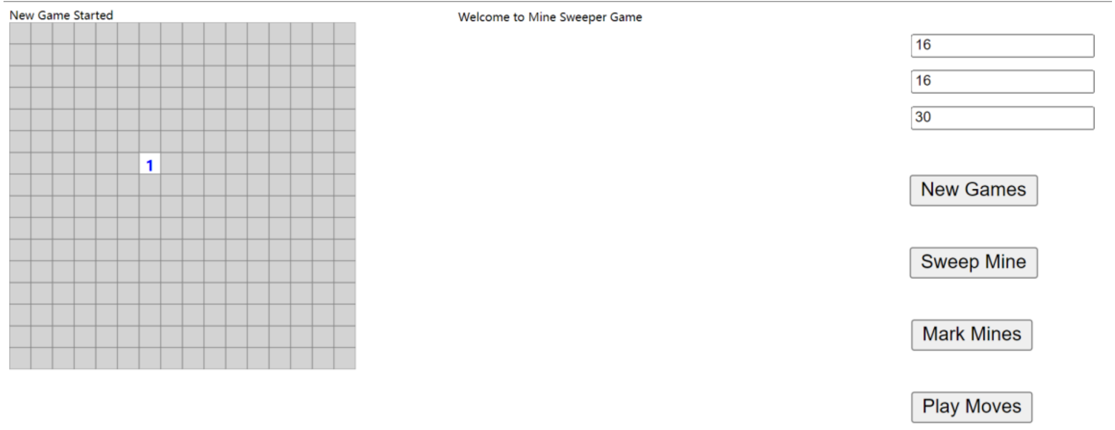
Welcome to Mine Sweeper Game

16

16

30

4.3 sweep mines


New Game Started

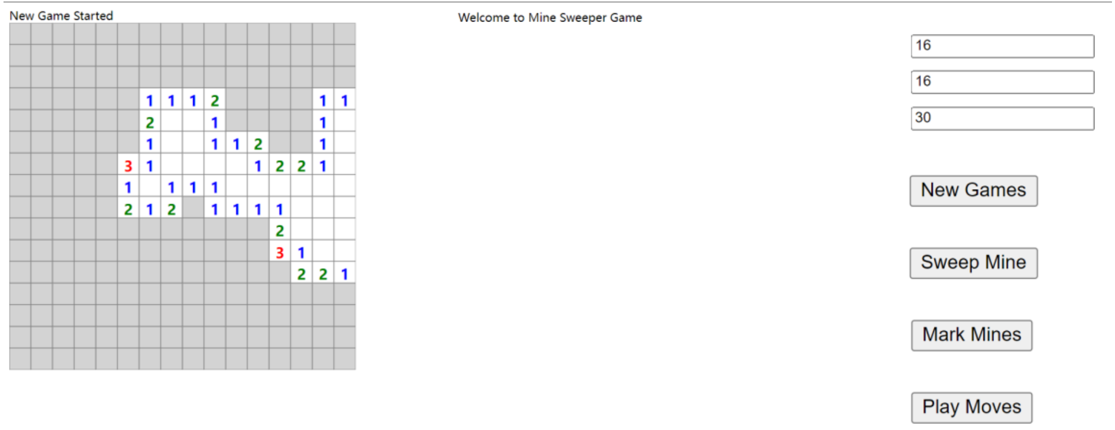
Welcome to Mine Sweeper Game

16

16

30

4.4 expand blank cells till reach mine cell’s neighbor


New Game Started

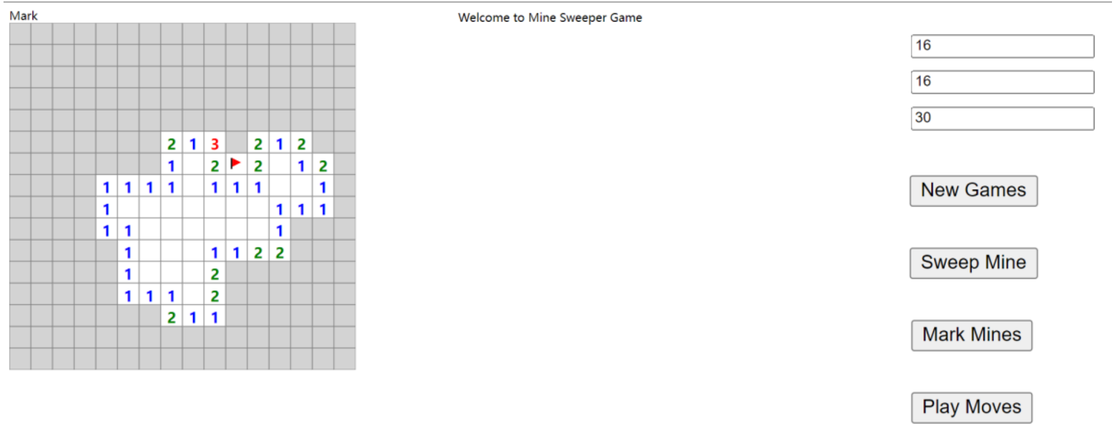
Welcome to Mine Sweeper Game

16

16

30

4.5 mark/flag mines and unmark/unflag them

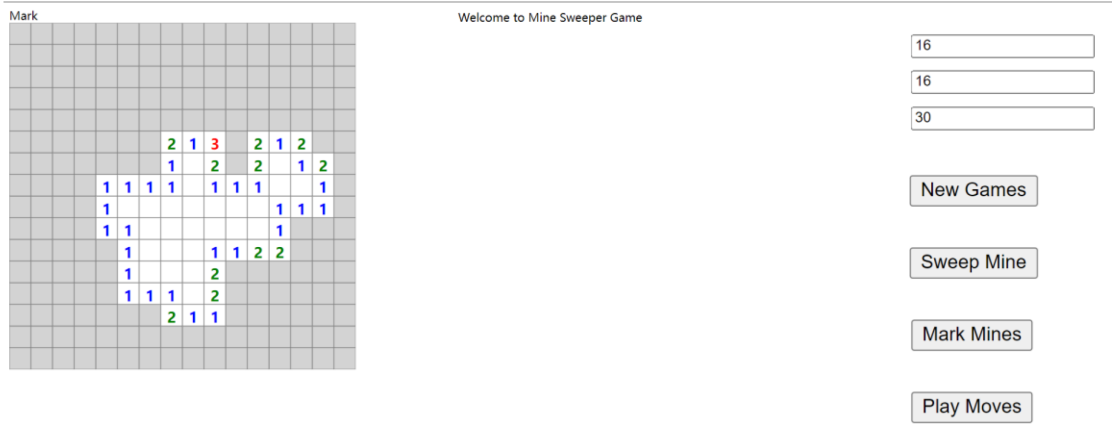

Mark

Welcome to Mine Sweeper Game

16

16

30


Mark

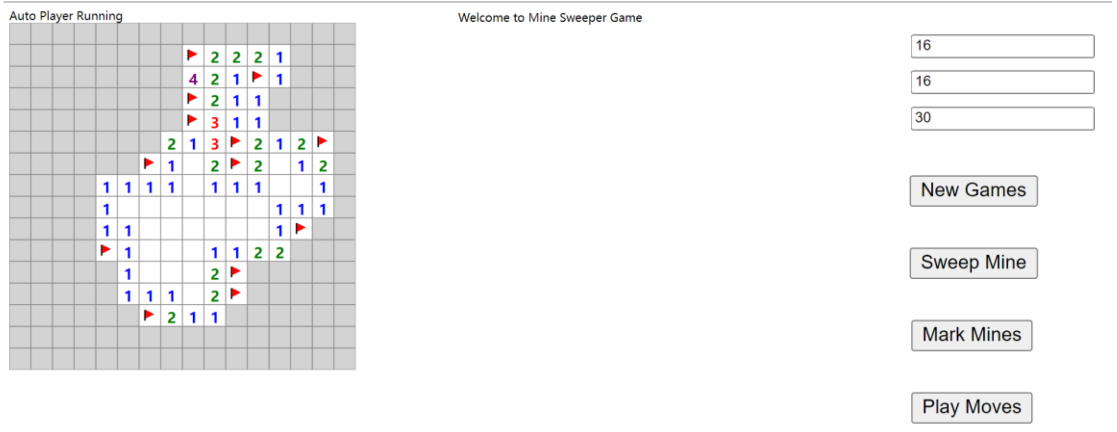
Welcome to Mine Sweeper Game

16

16

30

4.6 auto player running

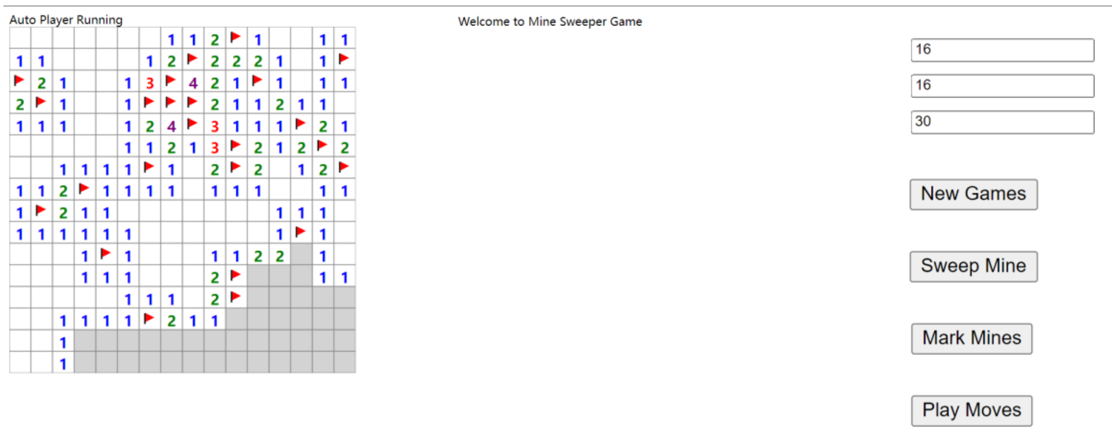

Auto Player Running

Welcome to Mine Sweeper Game

16

16

30


Auto Player Running

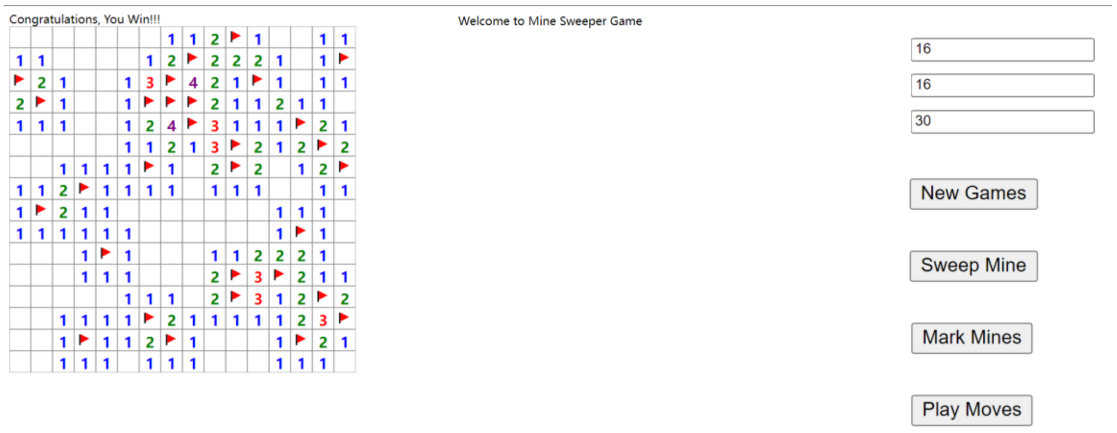
Welcome to Mine Sweeper Game

16

16

30

4.7 end game detection

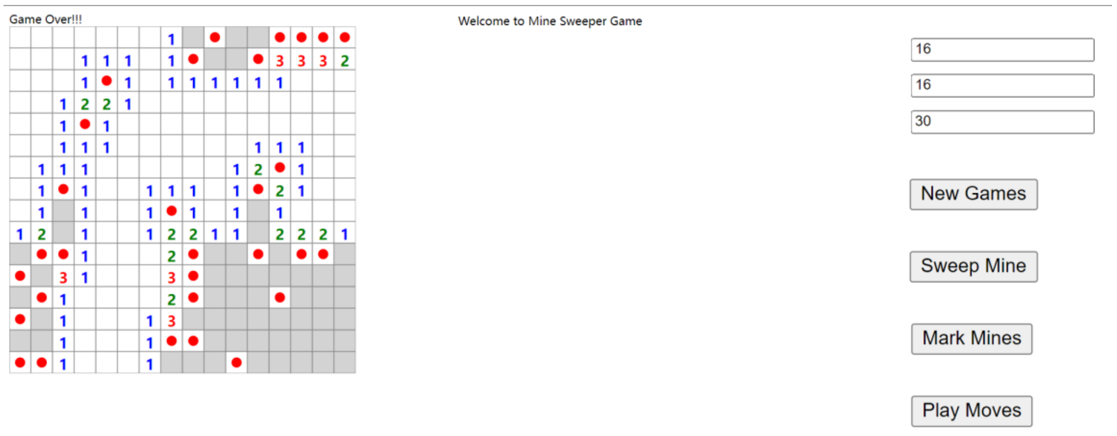

Congratulations, You Win!!!

Welcome to Mine Sweeper Game

16

16

30


Game Over!!!

Welcome to Mine Sweeper Game

16

16

30