Peyton Wolf

3/30/2025

Cst-250 Programming in C# ||

Mark Smithers

Milestone 2

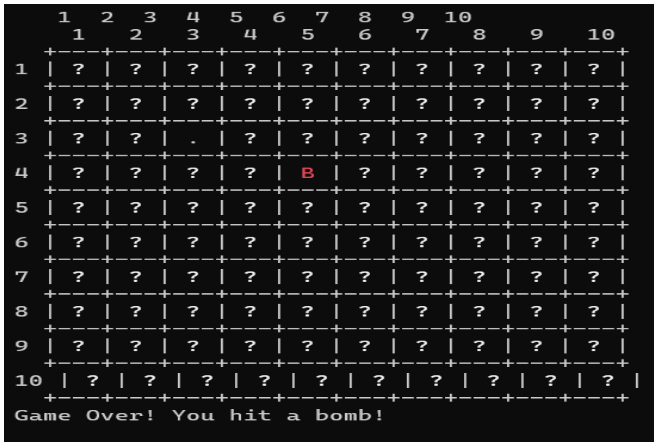
Loom Screencast Videos

https://www.loom.com/share/9a3a3cd67d354feeb34fc836e504f544?sid=53049cb9-6fe5-40c5-bd24-aaf59eb25be3

GitHub Link

https://github.com/KnoxHighStax/CST250/tree/main/Milestone%202

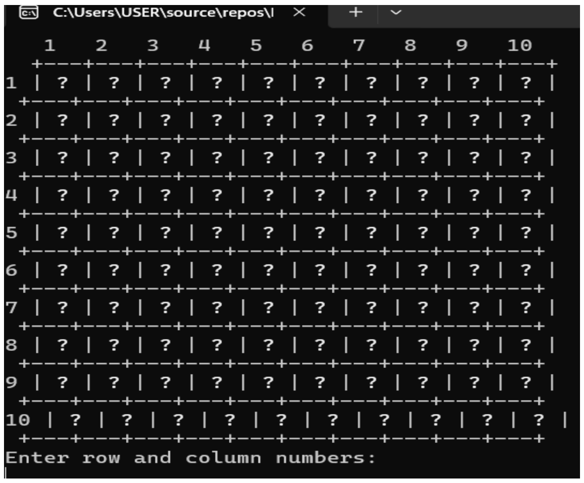
Determine Game State Method



```
    1  2  3  4  5  6  7  8  9  10
      1    2    3    4    5    6    7    8    9    10
   +----+----+----+----+----+----+----+----+----+----+
1  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
   +----+----+----+----+----+----+----+----+----+----+
2  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
   +----+----+----+----+----+----+----+----+----+----+
3  | ?  | ?  | .  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
   +----+----+----+----+----+----+----+----+----+----+
4  | ?  | ?  | ?  | ?  | B  | ?  | ?  | ?  | ?  | ?  |
   +----+----+----+----+----+----+----+----+----+----+
5  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
   +----+----+----+----+----+----+----+----+----+----+
6  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
   +----+----+----+----+----+----+----+----+----+----+
7  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
   +----+----+----+----+----+----+----+----+----+----+
8  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
   +----+----+----+----+----+----+----+----+----+----+
9  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
   +----+----+----+----+----+----+----+----+----+----+
10  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
   +----+----+----+----+----+----+----+----+----+----+
Game Over! You hit a bomb!
```

In this screenshot we are demonstrating the creation of the DetermineGameState method for the GameStatus, which helps to decide whether the game has been Won or Lost by the individual playing. This is something that will be occurring when the cells that are selected as visited have

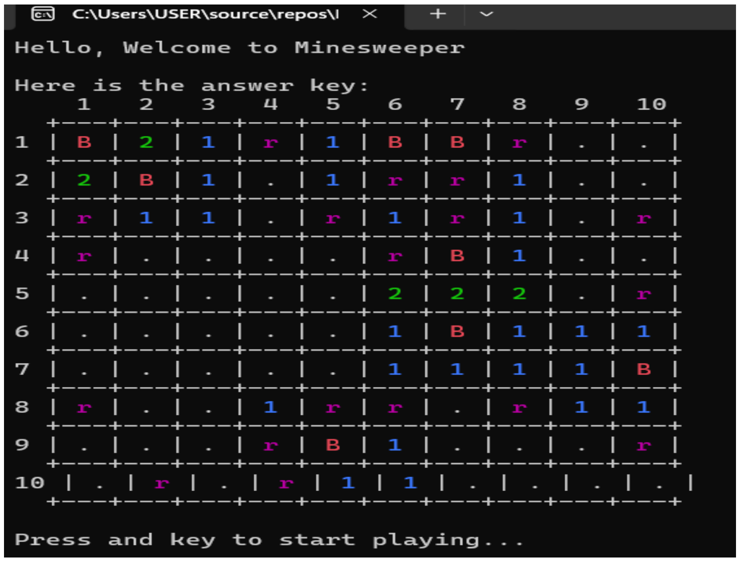
Printing Board



```
C:\Users\USER\source\repos\

    1    2    3    4    5    6    7    8    9    10
  +----+----+----+----+----+----+----+----+----+----+
1 | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
  +----+----+----+----+----+----+----+----+----+----+
2 | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
  +----+----+----+----+----+----+----+----+----+----+
3 | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
  +----+----+----+----+----+----+----+----+----+----+
4 | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
  +----+----+----+----+----+----+----+----+----+----+
5 | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
  +----+----+----+----+----+----+----+----+----+----+
6 | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
  +----+----+----+----+----+----+----+----+----+----+
7 | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
  +----+----+----+----+----+----+----+----+----+----+
8 | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
  +----+----+----+----+----+----+----+----+----+----+
9 | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
  +----+----+----+----+----+----+----+----+----+----+
10 | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  | ?  |
  +----+----+----+----+----+----+----+----+----+----+
Enter row and column numbers:
```

In this screenshot we are showing the program displaying the game board with the cell having already generated bombs, neighbors, rewards and if the cell is safe. Then will give the user the chance to input the coordinates of the cell they would like to select.

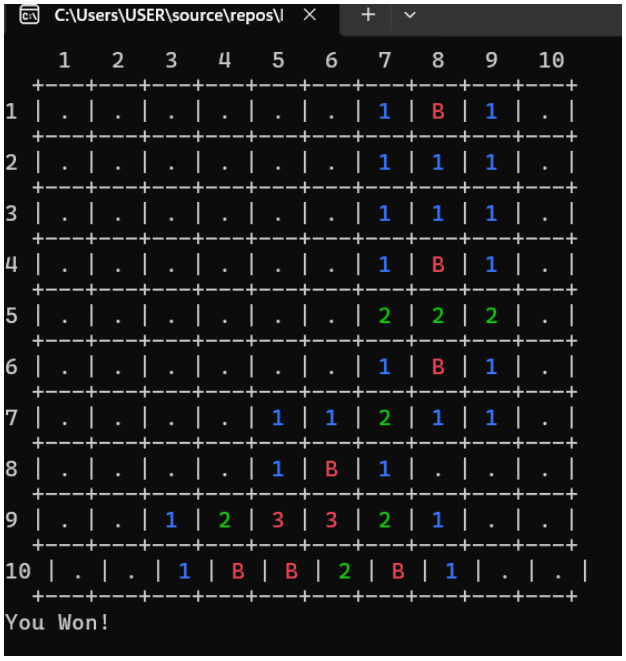
Answer Key Generation



```
C:\Users\USER\source\repos\I

Hello, Welcome to Minesweeper

Here is the answer key:
     1   2   3   4   5   6   7   8   9   10
   +---+---+---+---+---+---+---+---+---+---+
1  | B | 2 | 1 | r | 1 | B | B | r | . | . |
   +---+---+---+---+---+---+---+---+---+---+
2  | 2 | B | 1 | . | 1 | r | r | 1 | . | . |
   +---+---+---+---+---+---+---+---+---+---+
3  | r | 1 | 1 | . | r | 1 | r | 1 | . | r |
   +---+---+---+---+---+---+---+---+---+---+
4  | r | . | . | . | . | r | B | 1 | . | . |
   +---+---+---+---+---+---+---+---+---+---+
5  | . | . | . | . | . | 2 | 2 | 2 | . | r |
   +---+---+---+---+---+---+---+---+---+---+
6  | . | . | . | . | . | 1 | B | 1 | 1 | 1 |
   +---+---+---+---+---+---+---+---+---+---+
7  | . | . | . | . | . | 1 | 1 | 1 | 1 | B |
   +---+---+---+---+---+---+---+---+---+---+
8  | r | . | . | 1 | r | r | . | r | 1 | 1 |
   +---+---+---+---+---+---+---+---+---+---+
9  | . | . | . | r | B | 1 | . | . | . | r |
   +---+---+---+---+---+---+---+---+---+---+
10 | . | r | . | r | 1 | 1 | . | . | . | . |
   +---+---+---+---+---+---+---+---+---+---+

Press and key to start playing...
```

In this screenshot we are demonstrating our Minesweeper program to display the answer key for the game board that is about to be played for the user to review. Here you will be able to see where rewards, bombs, neighbors and safe zones are located. Then the user can press any key, and the game board will refresh to hide all the cell contents, and the game will begin.

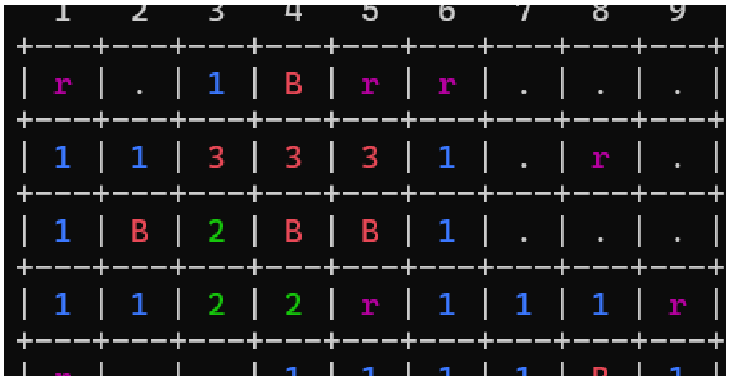
Winning Game



```
    1    2    3    4    5    6    7    8    9    10
  +----+----+----+----+----+----+----+----+----+----+
1 | .  | .  | .  | .  | .  | .  | 1  | B  | 1  | .  |
  +----+----+----+----+----+----+----+----+----+----+
2 | .  | .  | .  | .  | .  | .  | 1  | 1  | 1  | .  |
  +----+----+----+----+----+----+----+----+----+----+
3 | .  | .  | .  | .  | .  | .  | 1  | 1  | 1  | .  |
  +----+----+----+----+----+----+----+----+----+----+
4 | .  | .  | .  | .  | .  | .  | 1  | B  | 1  | .  |
  +----+----+----+----+----+----+----+----+----+----+
5 | .  | .  | .  | .  | .  | .  | 2  | 2  | 2  | .  |
  +----+----+----+----+----+----+----+----+----+----+
6 | .  | .  | .  | .  | .  | .  | 1  | B  | 1  | .  |
  +----+----+----+----+----+----+----+----+----+----+
7 | .  | .  | .  | .  | 1  | 1  | 2  | 1  | 1  | .  |
  +----+----+----+----+----+----+----+----+----+----+
8 | .  | .  | .  | .  | 1  | B  | 1  | .  | .  | .  |
  +----+----+----+----+----+----+----+----+----+----+
9 | .  | .  | 1  | 2  | 3  | 3  | 2  | 1  | .  | .  |
  +----+----+----+----+----+----+----+----+----+----+
10 | .  | .  | 1  | B  | B  | 2  | B  | 1  | .  | .  |
  +----+----+----+----+----+----+----+----+----+----+
You Won!
```

In this screenshot we are demonstrating the ability to win/lose in our Minesweeper application. Once the user has flagged all the bombs and visited all safe cells then the program will display the message you see above indicating to the player if they have won or lost. From here the user can close out the application and rerun it to play again, this time a new board will be generated!

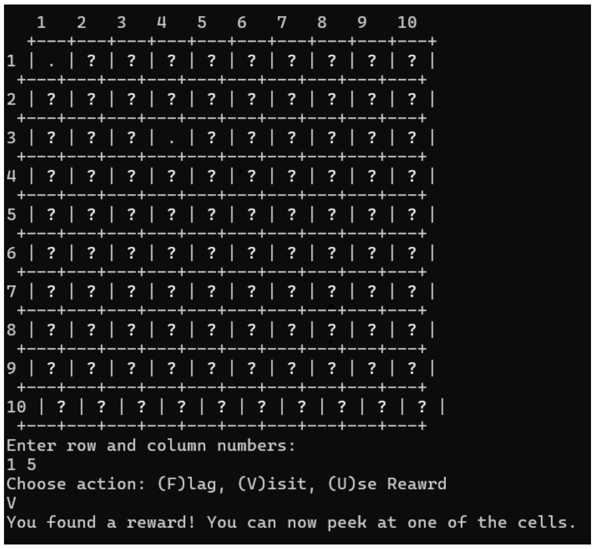
<u>Random Reward Generation</u>



```
  1    2    3    4    5    6    7    8    9
+----+----+----+----+----+----+----+----+----+
| r  | .  | 1  | B  | r  | r  | .  | .  | .  |
+----+----+----+----+----+----+----+----+----+
| 1  | 1  | 3  | 3  | 3  | 1  | .  | r  | .  |
+----+----+----+----+----+----+----+----+----+
| 1  | B  | 2  | B  | B  | 1  | .  | .  | .  |
+----+----+----+----+----+----+----+----+----+
| 1  | 1  | 2  | 2  | r  | 1  | 1  | 1  | r  |
+----+----+----+----+----+----+----+----+----+
```

On this screenshot we are demonstrating the functionality of the Reward function that we have created. Here you can see the rewards (“r”) are randomly generated and each time the game is rerun the rewards will appear in different cells each time, should more than likely not get the same exact board.

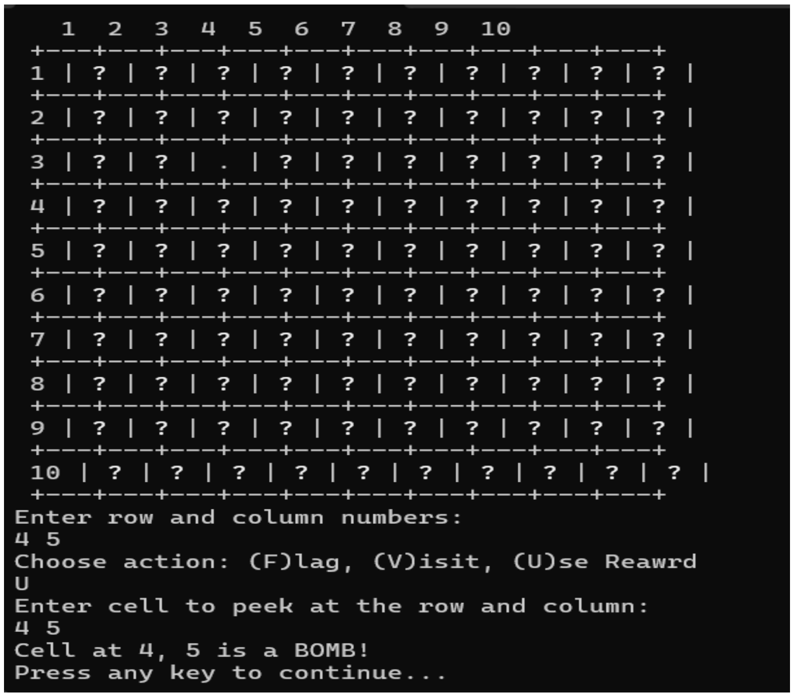
<u>Found Reward</u>



```
   1   2   3   4   5   6   7   8   9   10
  +---+---+---+---+---+---+---+---+---+---+
1 | . | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
2 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
3 | ? | ? | ? | . | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
4 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
5 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
6 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
7 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
8 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
9 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
10 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
Enter row and column numbers:
1 5
Choose action: (F)lag, (V)isit, (U)se Reawrd
V
You found a reward! You can now peek at one of the cells.
```

In this screenshot we are demonstrating the found Reward function to show what it will look like when a player finds a reward. When the game board generates the player will be given the option to select the coordinates for the row and column of the cell that they wish is Visit. Once a cell has become visited, if it has a reward the console will display the message above congratulating the player on finding the reward then will refresh the game board and fill the cell with a ".".

Peek Function



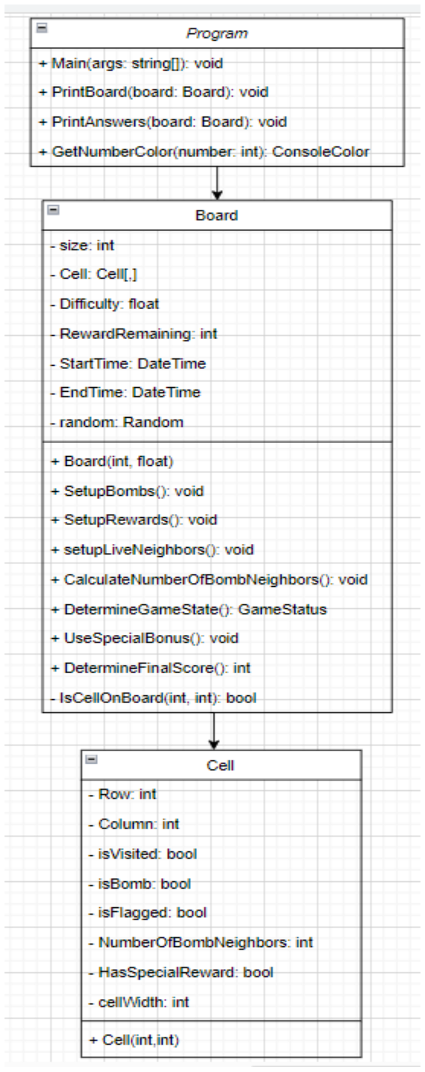
```
   1   2   3   4   5   6   7   8   9   10
  +---+---+---+---+---+---+---+---+---+---+
  1 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
  2 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
  3 | ? | ? | . | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
  4 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
  5 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
  6 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
  7 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
  8 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
  9 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
  10 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |
  +---+---+---+---+---+---+---+---+---+---+
Enter row and column numbers:
4 5
Choose action: (F)lag, (V)isit, (U)se Reawrd
U
Enter cell to peek at the row and column:
4 5
Cell at 4, 5 is a BOMB!
Press any key to continue...
```

In the screenshot we are demonstrating the peek reward ability, something that is granted to the player when they select a cell that has a reward in it, the rewad is the ability to peek at a cell to see what the contents of the cell are before visiting it. This is something that is usually done in a more strategic style of playing to ensure that the player doesn't hit a bomb.

4. The bold are the key words introduced in this lesson.

In this project we have demonstrated key object-oriented programming concepts like **class design** (like for Board and Cell classes), **encapsulation** like through properties and methods that help to control access to the games state. Also do not want to forget about **game state management** for tracking the visited cells, flags and the rewards. We utilized a **2D array** structure model for the game grid, while **randomization** handles bomb/reward placement. For our key gameplay elements, they include **neighbor analysis** (for calculating adjacent bombs), **conditional logic** (to determine the win/lose condition) and **user input validation**. Within our console UI we have showcased the use of **string formatting** and **color coding** for visualization, with **modular methods** for handling distinct tasks like board printing and cell revelation.

UML


Program
+ Main(args: string[]): void
+ PrintBoard(board: Board): void
+ PrintAnswers(board: Board): void
+ GetNumberColor(number: int): ConsoleColor
Board
- size: int
- Cell: Cell[,]
- Difficulty: float
- RewardRemaining: int
- StartTime: DateTime
- EndTime: DateTime
- random: Random
+ Board(int, float)
+ SetupBombs(): void
+ SetupRewards(): void
+ setupLiveNeighbors(): void
+ CalculateNumberOfBombNeighbors(): void
+ DetermineGameState(): GameStatus
+ UseSpecialBonus(): void
+ DetermineFinalScore(): int
- IsCellOnBoard(int, int): bool
Cell
- Row: int
- Column: int
- isVisited: bool
- isBomb: bool
- isFlagged: bool
- NumberOfBombNeighbors: int
- HasSpecialReward: bool
- cellWidth: int
+ Cell(int,int)

Questions

1. What was challenging?
   I would have to say that the reward system was a bit challenging and added a lot of complexity to the game state, which required additional validation checks when the player uses their reward. Debugging for the win/lose conditions proved to be tricky as well, especially when trying to ensure all non-bomb cells being revealed properly to trigger a winning game state.

2. What did you learn?

   In this project I have gained a deeper understanding of object-oriented design principles with encapsulation and separation of concerns of the Board and Cell classes. This project has reinforced my knowledge of 2D array manipulation and neighbor checking algorithms common for grid-based games.

1. How would you improve on the project?
   To improve this project, I could implement a proper scoring system based on completion time and difficulty level. The game could benefit from having the ability for the user to change difficulty levels or maybe have a challenge section where the difficulty starts out small and as the user completes the game boards a new, harder board will generate. Have this continue recursively until the player fails and then give it a leaderboard for tracking scores and winners.

2. How can you use what you learned on the job?

   The skills gained with this project have numerous professional applications, like object-oriented design experience translates directly to enterprise-level application development. With the state management techniques can be transferred to any system running complex state tracking, such as workflow engines or e-commerce systems.

__Day-to-Day__

Monday

Start:6pm End:9pm Activity: Read Chapter 6

Start: End: Activity:

Start: End: Activity:

Tuesday

Nothing

Wednesday

Start:6pm End:8pm Activity: Complete Reading Chapter 6

Start:8pm End:9pm Activity: Activity DQ1

Thursday:

Start:6pm End:9pm Activity: Started on Assignment 2/ChessBoardApplication

Start: 5pmEnd:6pm Activity: Tutoring session

Friday

Start:6pm End:7pm Activity: Activity DQ2

Start:8pm End:10pm Activity: Started on Milestone 2

Saturday

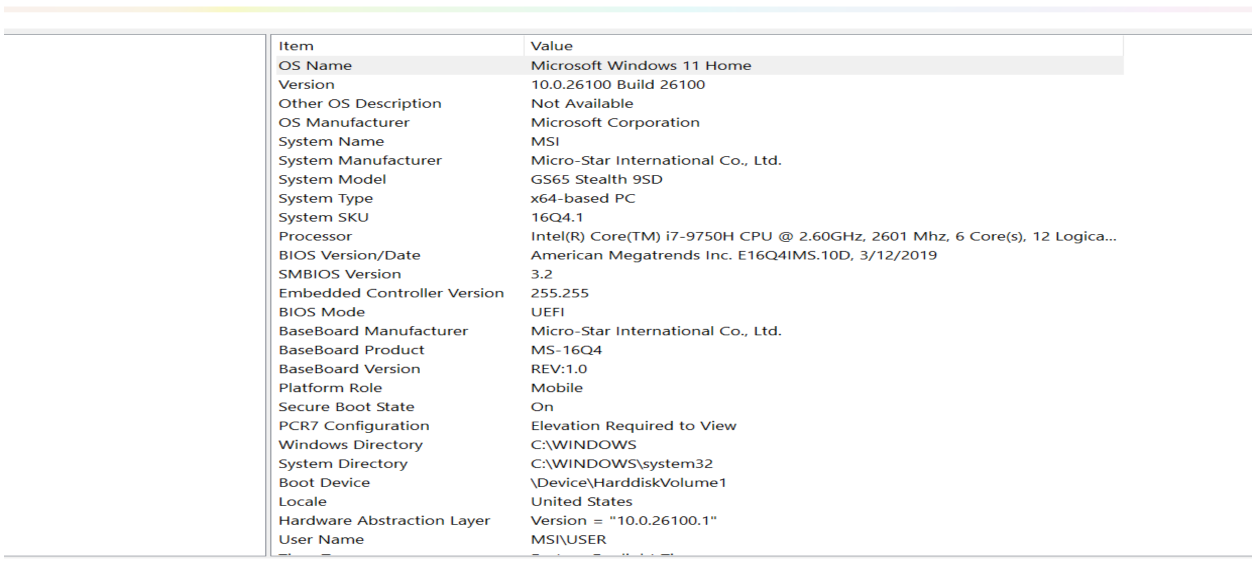
Start: 12pm End:4pm Activity: Completed Assignment 2/ChessBoardApplication

Sunday

Start:11am End:3pm Activity: Complete Milestone 2 Project

Start:6pm End:8pm Activity: Complete Milestone 2 Project

<u>Specs</u>



Item	Value
OS Name	Microsoft Windows 11 Home
Version	10.0.26100 Build 26100
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	MSI
System Manufacturer	Micro-Star International Co., Ltd.
System Model	GS65 Stealth 9SD
System Type	x64-based PC
System SKU	16Q4.1
Processor	Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 2601 Mhz, 6 Core(s), 12 Logica...
BIOS Version/Date	American Megatrends Inc. E16Q4IMS.10D, 3/12/2019
SMBIOS Version	3.2
Embedded Controller Version	255.255
BIOS Mode	UEFI
BaseBoard Manufacturer	Micro-Star International Co., Ltd.
BaseBoard Product	MS-16Q4
BaseBoard Version	REV:1.0
Platform Role	Mobile
Secure Boot State	On
PCR7 Configuration	Elevation Required to View
Windows Directory	C:\WINDOWS
System Directory	C:\WINDOWS\system32
Boot Device	\Device\HarddiskVolume1
Locale	United States
Hardware Abstraction Layer	Version = "10.0.26100.1"
User Name	MSI\USER

<u>Test Cases</u>

Test Case Id	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/ Fail
Test1	Visit safe cell	1. Initialize Board 2. Call VisitCell(3,4)	Row=3 Col=4 Cell(3,4).IsBomb=false	Cell marked visited neighbor count displayed	Cell(3,4)IsVisited=true Showed "2" neighbors	Pass
Test2	Visit Flagged Cell	1. Initialize Board 2. Flag cell(2,2) 3. Call VisitCell(2,2)	Row=2 Col=2 Cell(2,2).IsFlagged=true	Error: "Cell" is flagged!	Error message displayed	Pass

Test3	Win condition Test	1. Initialize 3X3 Board 2. Place Bombs at (0,0) 3. Reveal all other cells	Bombs=1at (0,0) Other cells are safe	GameState=Won “You Won!” message	Game ended with win message	Pass

Programming Conventions

1. We will be using the naming convention of camelCase for variables and parameters, like for “isVisited” and “numberOfBombs” and PascelCase for methods and classes, like for “CountBombsNearby(), GameBoard making sure that constants in UPPER_CASE (MAX_BOARD_SIZE).
2. When it comes to our code's readability through utilizing a 4-space indentation and ensuring that we have proper brace placement and strategic spacing around operators and commas.
3. Ensuring comprehensive documentation practices with comments for all public members to be able to review and making sure we have concise inline comments to explain complex algorithms or non-obvious decisions, while trying to avoid any redundant explanations of simpler parts of the code.

Use Case Diagram

Description: A player selects a cell to reveal its contents during an active Minesweeper game, triggering either a safe reveal, showing the neighbor bomb count, or ending the game if a bomb is uncovered.

Primary Actor: Player

Goals: - Successfully reveal a non-bomb cell to progress.

- Avoid revealing a bomb or get game over.

- Discover rewards, if any.

Stakeholders: - Player, wants to win.

- Developer, ensures logic works.

Pre-conditions: - Game board initialized

- Bombs and rewards randomly placed

- Player has not yet lost/won

Post-Conditions: - Cell marked as visited

- Reward claimed, if available

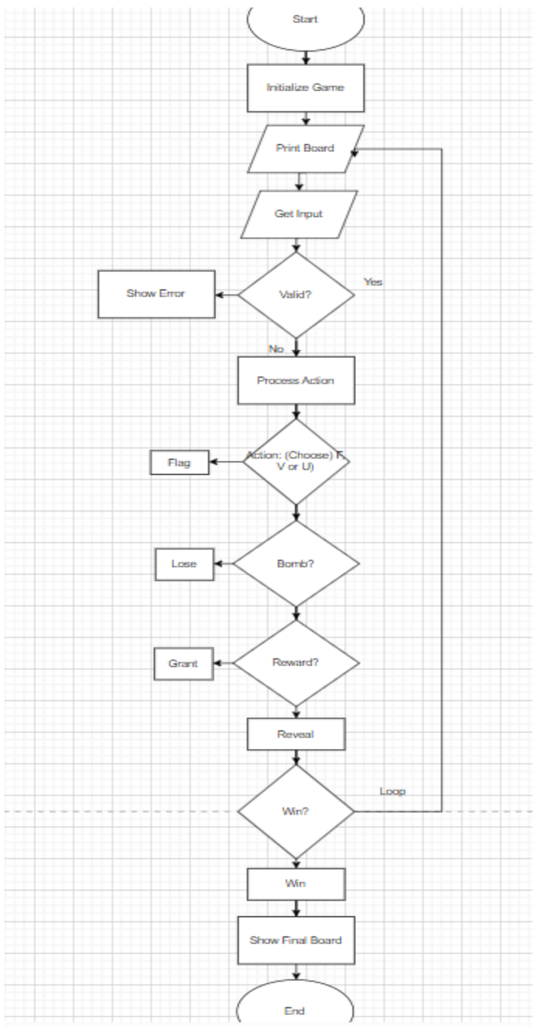
- Game state updated, continue/win/lose

Basic Flow:

1. Player chooses "Visit" action, enters coordinates.
2. System validates input, checks to make sure the cell exists and isn't already flagged.
3. System reveals the cell: If Bomb game ends, If safe will display neighbor bomb count and If reward will grant player the reward to use.
4. System updates board.
5. Player continues gameplay.

Alternative Paths:

- Cell is flagged and system rejects action
- Input of invalid coordinates, System will prompt re-entry



Bug Report

Nothing at this time.