MINESWEEPER GAME IN CPP

A PROJECT REPORT

Submitted by

Ronak Jain(23BCS10225)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh University

November, 2025

TABLE OF CONTENTS

List of Figures

CHAPTER 1. INTRODUCTION

1.1. Introduction to Project

As a gamming enthusiast, I embarked on developing a customizable Minesweeper

game in C++, a widely recognized puzzle game. O objective is to replicate the mechanics of traditional Minesweeper while enhancing the user experience by adding features like customizable grid size and mine density. The project explores basic programming concepts such as arrays, loops, and functions, but also aims to create a more flexible and engaging game experience. The game involves players attempting to clear a board filled with hidden mines by logically deducing their locations based on numerical hints provided by the revealed cells.

1.2. Identification of Problem

I noticed that many existing Minesweeper games, whether on desktop or mobile platforms, lack options for customization. Players are often restricted to predefined board sizes and fixed numbers of mines. Additionally, many of these implementations are not user-friendly, especially for beginners, leading to a steep learning curve and frustration. Therefore, I set out to address these problems by creating a version of Minesweeper that allows more customization and provides an intuitive, player-friendly interface.

CHAPTER 2. BACKGROUND STUDY

2.1. Existing Solutions

There are several versions of Minesweeper available across various platforms. The most notable examples include:

- **Windows Minesweeper**: A classic version that became synonymous with Windows operating systems.
- **Web-based Minesweeper**: Available on multiple websites offering browser-based gameplay.
- **Mobile Minesweeper Apps**: Various implementations on iOS and Android.

However, despite their popularity, most of these versions share common limitations such as fixed grid sizes, fixed mine counts, and a lack of customization, which reduces the overall experience for users who prefer more control over their game settings.

2.2. Problem Definition

The primary problem we identified is the rigidity in existing Minesweeper games. These games are often not customizable, leaving players unable to adjust key settings like grid size or mine count. Additionally, the interface design in many of these games can be unintuitive, particularly for new players unfamiliar with the game's mechanics.

2.3. Goals/Objectives

To address the limitations of existing solutions, our project aims to:

- Develop a customizable Minesweeper game in C++.
- Implement a user-friendly interface that clearly explains the rules and controls.
- Allow users to choose their grid size and the number of mines before starting the game.
- Provide informative feedback on game progress, with clear victory and defeat conditions.

CHAPTER 3. DESIGN FLOW/PROCESS

3.1. Evaluation & Selection of Specifications/Features

The essential features selected for this project include:

- A **9x9 grid with 10 mines** by default, with an option to customize both the grid size and the number of mines.
- Functions to handle key gameplay aspects such as initializing the board, revealing cells, and checking for game-over conditions.
- A display function that dynamically updates the board's state as the player progresses through the game.

3.2. Analysis of Features and Finalization Subject to Constraints

In our design process, we had to operate within certain constraints:

- **Grid Size**: The maximum allowable grid size is set at 20x20.
- **Mine Count**: The number of mines cannot exceed the total number of grid cells.
- **User Input**: The game must handle invalid inputs gracefully, ensuring no crashes or disruptions occur.

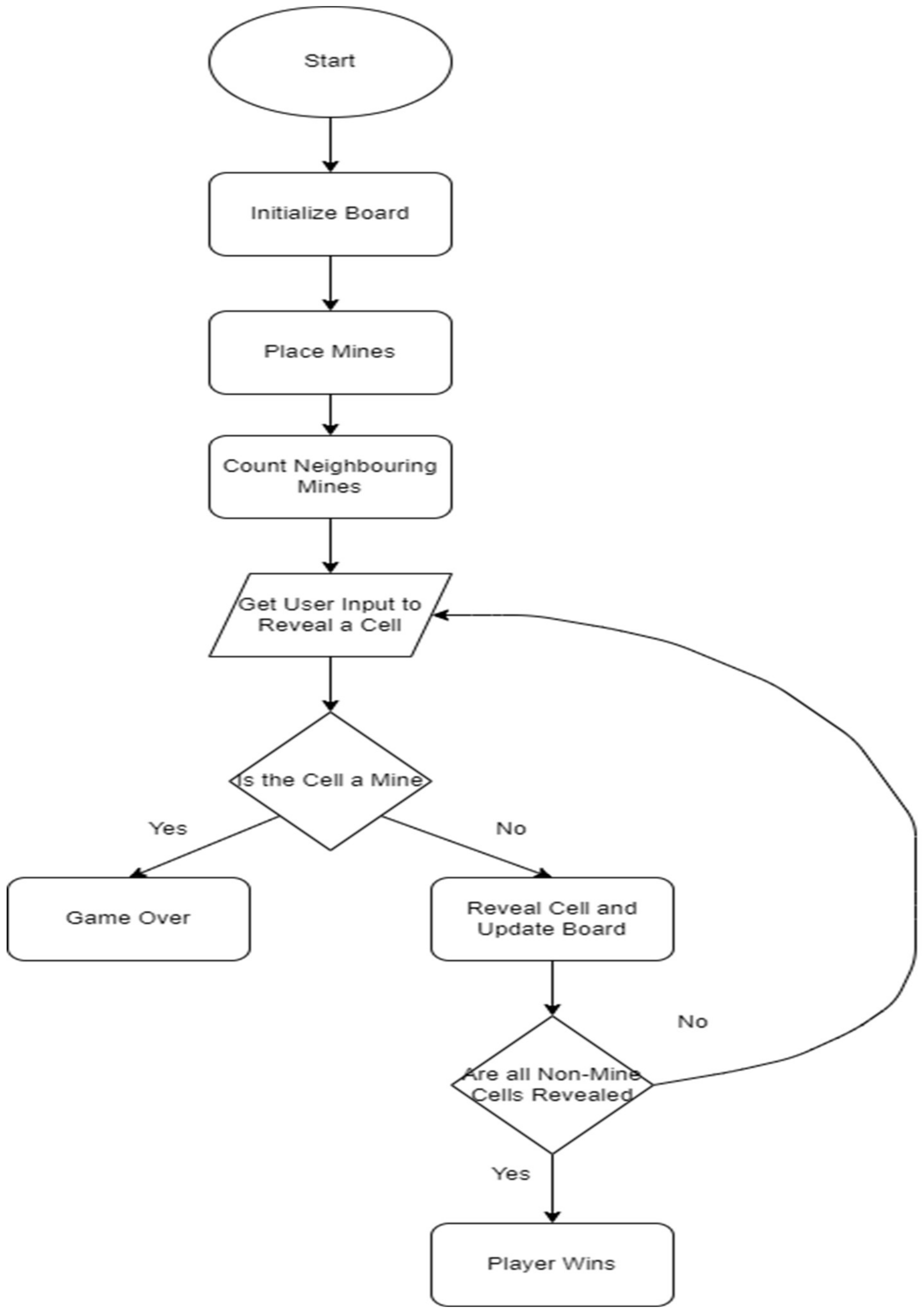
3.3. Design Flow

The design flow of the game follows these steps:

- **Board Initialization**: The board is initialized, and mines are placed randomly.
- **Gameplay Loop**: The player can reveal cells, with the board updating dynamically.
- **Game Logic**: After each move, the game checks for win or loss conditions, depending on whether a mine was revealed or all non-mine cells were cleared.
- **Display**: The board is displayed after every move, showing revealed cells while keeping mines hidden until the game ends.

CHAPTER 4. RESULTS ANALYSIS AND VALIDATION

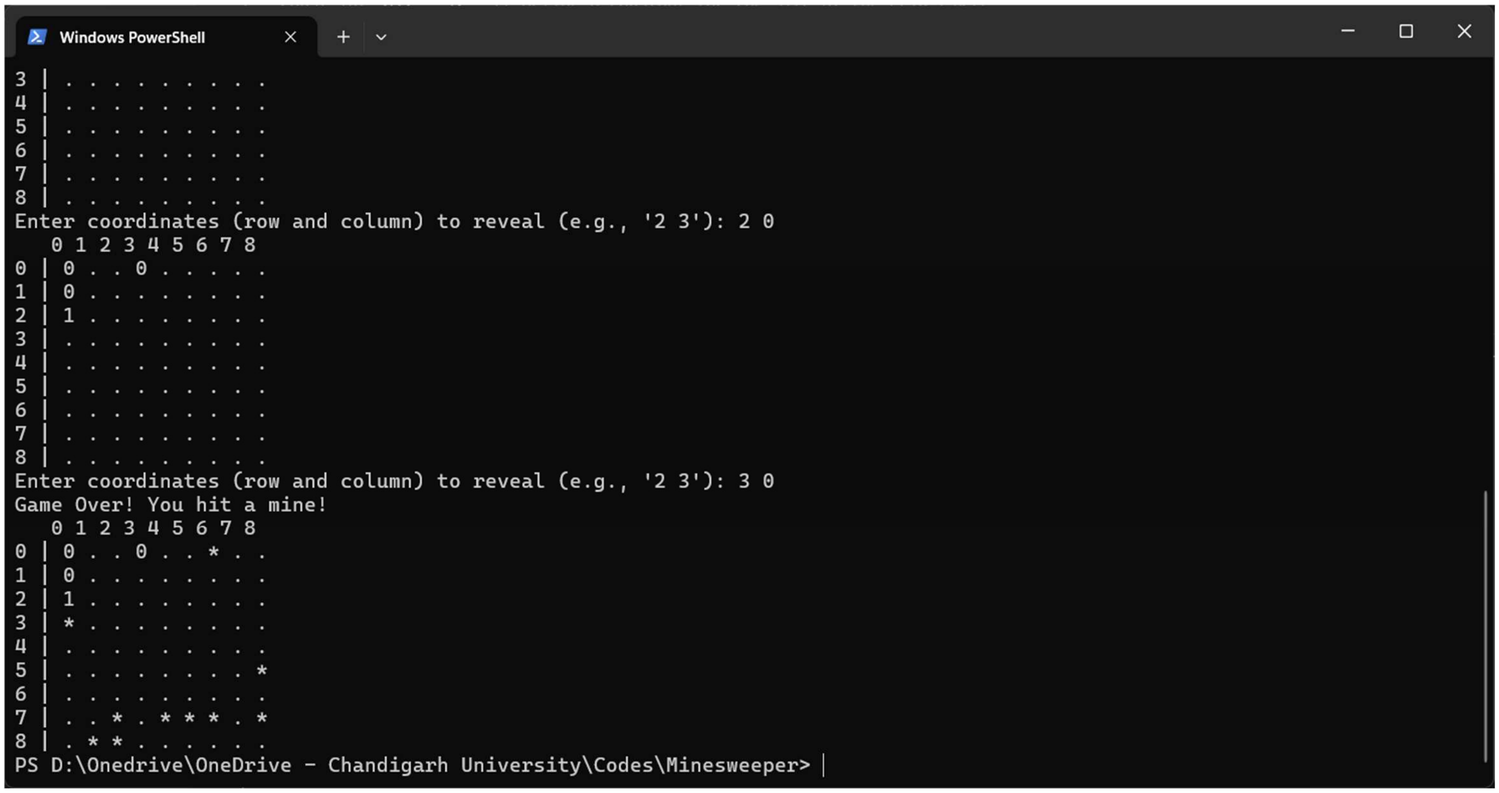
4.1. Implementation of Solution: The Minesweeper game was successfully implemented in C++, adhering to the design flow outlined in Chapter 3. Below is the flowchart of how the design is made:



4.2. Output:

The output of our Minesweeper game is displayed directly in the console. The game starts by presenting the player with a blank grid, where cells are represented by symbols such as asterisks (*) for unrevealed cells. As the player makes selections, the chosen cells are revealed, displaying numbers that indicate the count of neighbouring mines, or a mine symbol if the cell contains a mine.

In the case of winning the game (when all non-mine cells are revealed), a congratulatory message is displayed, and the board shows the locations of all remaining mines. If the player triggers a mine, the game ends with a "Game Over" message, and the board reveals the positions of all hidden mines.



```
3 | . . . . . . . . .
4 | . . . . . . . . .
5 | . . . . . . . . .
6 | . . . . . . . . .
7 | . . . . . . . . .
8 | . . . . . . . . .
Enter coordinates (row and column) to reveal (e.g., '2 3'): 2 0
   0 1 2 3 4 5 6 7 8
0 | 0 . . 0 . . . . .
1 | 0 . . . . . . . .
2 | 1 . . . . . . . .
3 | . . . . . . . . .
4 | . . . . . . . . .
5 | . . . . . . . . .
6 | . . . . . . . . .
7 | . . . . . . . . .
8 | . . . . . . . . .
Enter coordinates (row and column) to reveal (e.g., '2 3'): 3 0
Game Over! You hit a mine!
   0 1 2 3 4 5 6 7 8
0 | 0 . . 0 . . * . .
1 | 0 . . . . . . . .
2 | 1 . . . . . . . .
3 | * . . . . . . . .
4 | . . . . . . . . .
5 | . . . . . . . . *
6 | . . . . . . . . .
7 | . . * . * * * . *
8 | . * * . . . . . .
PS D:\Onedrive\OneDrive - Chandigarh University\Codes\Minesweeper> |
```

CHAPTER 5. CONCLUSION AND FUTURE WORK

5.1. Conclusion

Our Minesweeper game project demonstrates a customizable version of the classic puzzle game. By incorporating C++ programming concepts, we successfully created a game that enhances user experience, allowing players to set their own grid size and number of mines. This project highlights logical thinking, user interaction, and efficient programming techniques.

5.2. Future Work

To further improve the game, future work could include:

- Adding **difficulty levels** (easy, medium, hard) to automatically adjust grid size and mine density.
- Implementing a **timer** to track the player's performance.
- Introducing a **graphical user interface (GUI)** using libraries such as SFML or SDL.
- Exploring **multiplayer functionality** for competitive gameplay between users.