**CMPE 230, SYSTEM PROGRAMMING MINESWEEPER GAME**

**Yusuf Akdoğan 2021400075**

**Bahadır Kuşcan 2022400210**

**02.06.2024**

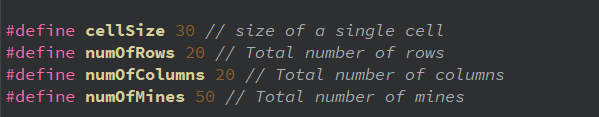
1. **Introduction**

The Minesweeper game is a classic puzzle game where the objective is to clear a grid containing hidden mines without detonating any of them. The grid presents numbers indicating how many mines are adjacent to each cell, allowing the player to strategically deduce safe cells to uncover. This document provides an in-depth look at the problem definition, solution, user interface, execution, program structure, examples, improvements, difficulties encountered, and conclusion of the Minesweeper game project.

1. **Program Interface and Execution**

**Game Initialization:**

* To execute the program first ensure QT is installed and working. Then you can simply open QtCreator and open minesweeper.pro as a new Project.



**Gameplay:**

* The user clicks on cells to reveal them.
* If a mine is clicked, the game ends.
* If a number is revealed, it indicates how many mines are adjacent to that cell.
* If there is no adjacent mine no number will pop up and all the adjacent cells will be revealed
* The user can flag cells suspected of containing mines.
* The number of revealed cells will be score.
* User can get hints by clicking hint button and clicking second time will result revealing the cell.

**Winning the Game:**

* The user wins by revealing all non-mine cells.

**User Interface:**

* The GUI includes buttons for revealing cells and flagging mines.
* There are menu options for starting a new game and taking a hint from system.
* Hint button gives hint in first click and opens the hinted cell in second click

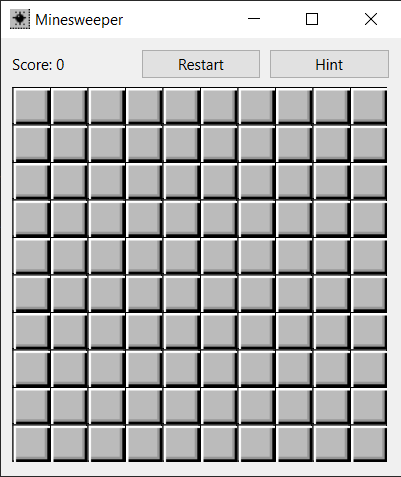
1. **Input and Output**

User can open cells by simply left clicking them. Similarly to restart to game left clicking restart button and to get a hint left clicking hint button will be sufficient

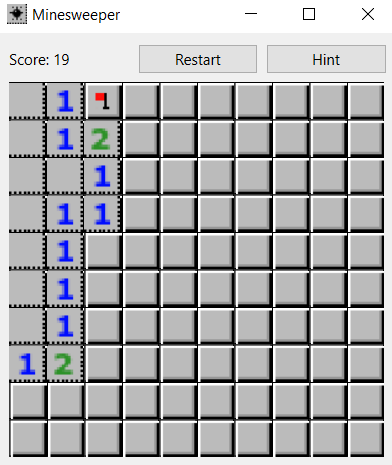
Right clicking will result with a flag. If there already is a flag it is possible to remove it with a second right click.

**Some Examples:**

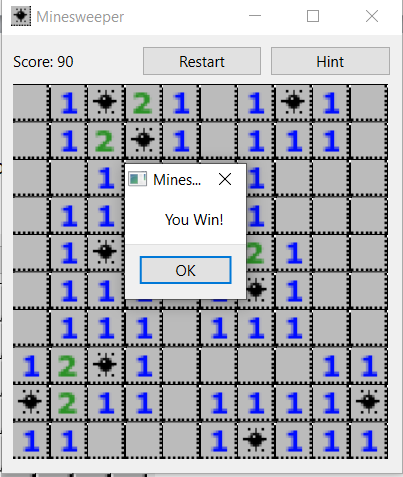
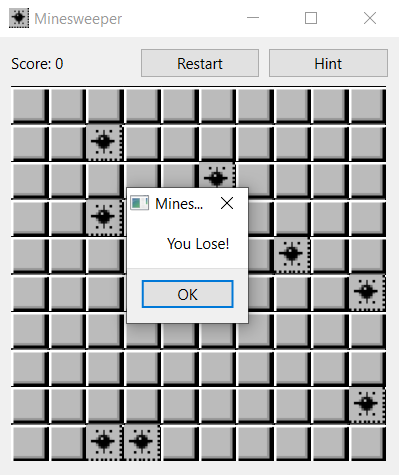
* Initial situation:



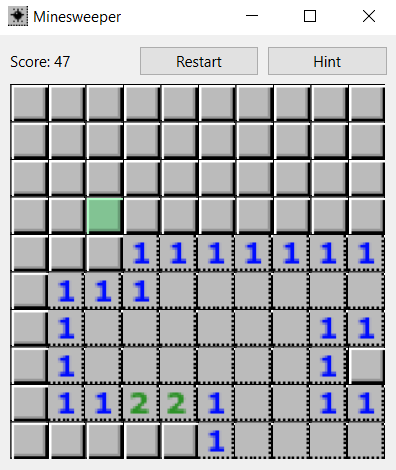
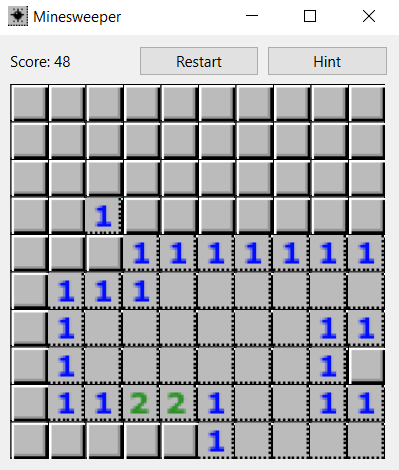
* After left click top left most cell and putting a flag:



* Game lose and game win screens:

* Using hint button twice:

1. **Program Structure**

The Minesweeper game is structured into several modules:

1. **main.cpp:**
   * Initializes the game and handles the main game loop.
   * Manages the graphical user interface, including drawing the grid and handling user interactions.
2. **MineField.cpp:**
   * Contains the logic for grid generation, mine placement, and cell revealing.
3. **Cell.cpp:**
   * Stores data for each cell
4. **Improvements and Extensions**

**Possible Improvements:**

* **Enhanced Graphics:** Improving the visual appeal of the game.
* **Leaderboard:** Implementing a leaderboard to track high scores.
* **Advanced Commands:** The game can include more advanced commands. For example left clicking an opened cell with a status equals to the amount of flags adjacent to it may open all non-flagged cells.
* **Multiplayer Mode:** Allowing multiple players to play cooperatively or competitively.

1. **Difficulties Encountered**

### Challenges:

* **GUI Implementation:** Handling user interactions and drawing the grid were initially challenging. There were problems like having changes in cell sizes and having non-square cells. However those problems got eventually solved by fixing sizes or using similar approaches.
* **Hint Option:** Implementing a perfect algorithm for the hint button that utilizes all the information on the board proved to be challenging and somewhat excessive for our needs. Instead, we implemented a sufficient algorithm that uses a partial set of information. The algorithm works as follows:

1. **Initial Comparison:**
   * The algorithm first compares the status of all opened cells with the number of closed cells adjacent to each opened cell.
   * If the number of adjacent closed cells equals the number indicated by the opened cell, virtual flags are placed on each of these closed cells.
2. **Virtual Flag Check:**
   * The algorithm then scans all the opened cells again.
   * It compares the number of virtual flags with the number indicated by the opened cell.
3. **Safe Cell Identification:**
   * If the number of virtual flags around an opened cell equals the number on the cell, and there is an additional closed cell without a virtual flag, this closed cell is considered safe.
   * This safe cell is then provided as a hint to the player.

This approach provides a practical balance, offering useful hints without the complexity of a perfect algorithm.

1. **Conclusion**

The Minesweeper game project provided valuable experience in GUI programming, algorithm design, QT implementation and project documentation. The final product is a functional game with potential for future enhancements. This project highlighted the importance of planning, problem-solving, and continuous improvement in software development.

1. **Appendicies**

The source code is provided as an extra ZIP file because of its size