

PROJECT REPORT

PROJECT NAME: Minesweeper game

TEAM DETAILS: Team 1, Group15, Slot 6

TEAM MEMBERS:

1. S Surya(Team leader)
2. Mohammad Imran Ansari
3. Rakesh Chandra Siyag
4. Rajkumar Das

MINSWEEPER GAME: The object of the game is to open all those cells on the board which do not contain mines in them. The game is lost if the player opens a cell with a mine in it.

WORKING OF THE GAME

The game is played on a board displayed on the terminal. The board is represented internally as a three dimensional matrix called $A[16][16][2]$. It can be thought of as two two dimensional matrices.

$A[16][16][0]$ contains a number which represents the number of mines around the cell.

$A[16][16][1]$ can be equal to 0, 1, or 2.

- 0 - The cell is closed
- 1 - The cell is open
- 2 - The cell has been flagged

The following functions are used

1. `void change(int A[16][16][2],int i,int j,int n)`
This function is called when the function “putNumbersInTheGrid” locates a cell in the grid with mine. The “change” function increments the number of the eight cells surrounding it by one. At the end it produces a matrix where each cell's value represents either a mine or the number of mines in the eight cells adjacent to it.
2. `void showAllZero(int A[16][16][2],int n,int x,int y,int &openedGridCount)`
This function is called when a cell containing zero mines around it is opened. It proceeds to call itself to open all those cells around it that contain zeroes until it reaches a cell that contains a non-zero number.
3. `void show(int A[16][16][2],int i,int j,int n,int &openedGridCount)`
This function is used to open those cells that have been set to be opened. It is called by showAll.
4. `void generateBoard(int A[16][16][2],int n)`
This function generates initializes all values in the 3 dimensional matrix to zero. In the program,

it is followed by the "putMines" function.

5. `void putMines(int A[16][16][2],int n)`
This function puts mines into the board at random positions. Two random numbers are generated. These numbers are then used as x and y coordinates of the cell where a mine is placed.
6. `void putNumbersInTheGrid(int A[16][16][2],int n)`
This function searches for mines in the board. When it finds one, it calls the "change" function.
7. `void showboard(int A[16][16][2],int n,int &openedGridCount,int &flagCount)`
This function is called after every single move. It is responsible for outputting characters onto the screen so that a board is displayed after every turn.
8. `void openMine(int A[16][16][2],int n)`
This function is used to open all mines on the board if the user chooses to open any cell with a mine in it.
9. `void showAll(int A[16][16][2], int n, int &openedGridCount)`
This function is invoked after every move. It searches for cells that have been marked to be opened and calls show to open them. It also displays how many cells have been opened.
10. `void flag(int A[16][16][2],int n,int &openedGridCount,int &flagCount)`
This function is invoked when the user wishes to place or remove a flag. It accepts an input from the user also displays the board after processing the input.
11. `void printTitle()`
This function prints the word "MINESWEEPER" on top of the console screen.
12. `void easyScore(double newScore)`
This function is called when the user wins a game at the easy level. It takes in the time taken by the player as the parameter. It compares the score to the previous smallest score. If the current score is smaller than the previous one, it will delete the previous one and store the new score. It will also prompt the user to enter their name and will store this as a string.
13. `void hardScore(double newScore)`
This function is called when the user wins a game at the hard level. It takes in the time taken by the player as the parameter. It compares the score to the previous smallest score. If the current score is smaller than the previous one, it will delete the previous one and store the new score. It will also prompt the user to enter their name and will store this as a string.

14. `void showEasyScore()`

This function displays the current best score and the person behind that score.

15. `void showHardScore()`

This function displays the current best score and the person behind that score.

SCOPE FOR IMPROVEMENT:

1. The game is played on a terminal window. A GUI, that could handle inputs from a mouse, will improve the user experience.
2. Currently, only one high score can be stored for each level. This can be improved.
3. The timing for the game starts when the board is first displayed. In most other versions of the game, the timer starts when the players make their first move.