

SRS DOCUMENT

MINESWEEPER

INTRODUCTION:-

Minesweeper is a single player puzzle game that involves logical thinking. The game was created by Microsoft company to develop analytical skills of people. The game consists of a rectangular grid consisting of multiple tiles some containing mines and rest without mines.

The player has to clear the grid by uncovering the tiles but without uncovering the ones containing the mines.

PROBLEM DEFINITION:-

To create a replica of the game minesweeper using C++ programming language.

GAMEPLAY:-

There are three levels of this game, namely:

- 1.EASY

2.INTERMEDIATE

3.ADVANCED

4.CUSTOM: User dependent

➤ Number of grids in each level:

1.Easy-9x9 rectangular grids

2.Intermediate-16x16 rectangular grids

3.Advanced-16x30 rectangular grids

4.Custom-User Dependent (Anything lesser than 24x30)

➤ Number of mines in each level:

1.Easy-10 mines

2.Intermediate-40 mines

3.Advanced-99 mines

4 .Custom-lesser than 668 mines

- To uncover a tile just press the left mouse button.
- To mark a tile just right click on it, to unmark it press right click again.

RULES:-

- Uncover a mine and the game ends!!!
- Uncover an empty square, and you keep playing.
- Uncover a number and it tells you how many mines lay hidden in the nearest eight tiles.

SCORING :

- The player's score is measured in terms of the time taken to complete the game.
- Lesser the time, higher is the player's score.

ALGORITHM:-

- Select the difficulty level to either easy, intermediate, advanced or custom.
 - According to the difficulty level selected develop a gameboard.
 - While developing the gameboard randomly assign specified number of mines.
 - Dealing with two boards-gameboard and problemboard.
 - Problemboard interacts with the player.
 - With reference to the mines, for the boxes that don't contain mines substitute the number of mines surrounding that particular block.
-
- First Click: In the first click the following cases may occur:
 1. The cell selected doesn't contain mine-

The mines in the nearest 3x3 matrix are shifted to the upper left part of the board. As a result the cell becomes empty.
 2. The cell selected contains a mine then it along with the mines in its 3x3 matrix are shifted to upper left part of the board.

In both the above points(1 and 2) after the shift has taken place number of mines in each of the cells of that 3x3 matrix are shown.

- Accept the row and column number selected by the player which he wants to be either a flag or a non-mine.
- If player selects it to be non-mine-if it is a mine then end the game else reveal the value of the corresponding number on the gameboard.
- If the cell selected by the player is a mine then declare it to be a mine in the problemboard.
- Check if problemboard matches completely with the gameboard.
- If yes, then the player is declared to be the winner!!! Else repeat all the above steps.

FUNCTIONS AND HEADER FILES

Different libraries used :

```
#include <iostream>
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string>
```

```
#include <ctime>
```

```
#include <time.h>
```

Functions used are:

void GamePlay()

- Function that helps in playing the game.

void dispzero()

- function that is used to display blank boxes.'0' represent blank box.

void loadboard()

- function that loads the board

void disploseboard()

- function that displays the entire field once you lose

int timer()

- function that shows clock time.

void dispwin()

-function to display the winning screen(Board,total time taken to solve and new game option)

void displosescreen()

-function to display the losing screen

int Newgame(const Position&);

-function that find what to do after you lose or you win

int Choice(const Position&);

-function that checks the choice you have inputted

int HSMenu(const Position&)

-function that checks the choice high score menu

loadgameplayscreen()

- displays the gameplay screen

load_menu()

- displays the game menu

int high(int newscore,int mines)

- updates the highscore