

CS101 PROJECT

MINESWEEPER

Submitted to: Prof. D.B.Phatak

INDEX

1. Acknowledgement
2. Introduction
3. Project Overview
4. Status of Completion
5. Rules and Basics
6. Features
7. Functions Used
8. System Requirements
9. Screenshots

1. ACKNOWLEDGEMENT

In the present scenario, where the world is taking breath at a high pace, computers have now become an integral part of our life. We are given with an excellent opportunity to learn computers by our institute under the guidance of one of the best professor in Computer Science of India Dr.D.B.Phatak. He increased the field of our knowledge by indulging us in such a good project which requires a good knowledge of C++ language and graphics. Lessons by sir during the classes proved to be of great help .We learned great qualities like Professionalism, Team work, Self Assessment ,etc which are sure to play an important role in our life. Last but not the least we will also like to thank our very helping TA Aarif Jindani who was there always to help us in any difficulty and to clear our doubts.

Overall it was a great journey and again a big thanks to our professor.

2.INTRODUCTION

MINESWEEPER is a mind game. The object of the game is to clear an abstract minefield without detonating a mine

Minesweeper cannot always be solved with 100% certainty, and may require the occasional use of probability to flag the square most likely to have a mine. In other words, one must sometimes guess to solve a minesweeper puzzle.

3. PROJECT OVERVIEW

Our Team (Lab Batch 212) was divided into 3 teams each consisting of 2 members. Following are the details of teams and work allocated to them-

Team A: Work Allocated - Algorithm and Logic Part
Ananthkrishnan P - 110260029 (Team Leader)
Ankit Parashar - 110010059

Team B: Work Allocated - Creating a Graphics based user interface using the EzWindows.

Akshay Jain - 110040019
Ankit Yadav - 110040109

Team C: Work Allocated - Processing the mouse input
Ajay kumar - 110040062
Ankit Garg - 115060002

4. STATUS OF COMPLETION

The Project has been completed successfully in an executable condition with three difficulties levels and high score feature.

5. RULES AND BASICS

The objective is to find the empty squares in the minefield while avoiding the mines. The faster you clear the board, the better your score.

The Minesweeper Board

Minesweeper has three standard boards to choose from, each progressively more difficult.

Recruit Level: 7*7 tiles, 8 mines

Regular Level: 11*11 tiles, 20 mines

Veteran Level: 15*15 tiles, 45 mines

HOW TO PLAY

The rules in Minesweeper are simple:

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 eight surrounding squares—information you use to deduce which nearby squares are safe to click.

You win if you correctly flag all the boxes which contains the mines.

Hints and tips

Mark the mines. If you suspect a square conceals a mine, click on it when flag mode is on. This puts a flag on the square.

Study the patterns. If three squares in a row display 2-3-2, then you know three mines are probably lined up beside that row. If a square says 8, every surrounding square is mined.

Explore the unexplored. Not sure where to click next? Try clearing some unexplored territory. You're better off clicking in the middle of unmarked squares than in an area you suspect is mined.

6. FEATURES

Some of the Features that is available to the User:

Option to choose **THREE DIFFICULTY LEVELS**.

The User will get randomly mined maps every time.

The first click will always be on an empty box.

The User can Quit or Restart the game at any point of time(except right after losing the game).

A function to display the elapsed time has been included.

Features that we thought but were not able to implement:

We wanted to implement left click and right click but EzWindows would not permit that.

We wanted to take the name of the user as an input if he/she gets a high score. We added the necessary commands for that in the program. But since the input has to be taken from the terminal we avoided that and removed the codes. The other option available for us was to display the images of all alphabets on the window once a highscore was made, but since it required more effort in the graphics part and due to time constraint we did not implement that either.

7. FUNCTIONS USED

The following are the various code intercepts written by our Team members for various input and output operations. This includes only the main functions that are used. Minor functions have been avoided and details about those functions have been given as comment lines in the program.

Different libraries used

```
#include <iostream>
#include "ezwin.h"
#include <stdio.h>
#include "bitmap.h"
#include "rect.h"
#include <stdlib.h>
#include <string>
#include <cassert>
#include <ctime>
#include <time.h>
```

Programs that are loaded into the main program

```
#include "MSGraphics.cpp"
#include "Highscores.cpp"
#include "Highscoresdisplay.cpp"
#include "Stopwatch.cpp"
```

- void Gameplay()
main function that helps in playing the game
- void displayzero()
function that is used to display blank boxes
- void load_map()
function that loads the minefield
- void displayfield()
function that displays the entire field once you lose
- int timer()
function that displays clock time
- int disprec()
function to view highscore in recruit level
- int dispreg()
function to view highscore in regular level
- int dispvet()
function to view highscore in veteran level
- void displaymine (float x, float y)
function to display image of mine
- void displayflag(float x,float y)
function to display the image of flag
- void displaywin()
function to display the winning screen
- void displayyoulose()
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
- int MouseClick(const Position &p)
function that checks the position of the mouse click in the gameplay screen and follows up with calling the respective function
- load_GUI()
displays the gameplay screen
- load_menu()
displays the game menu
- int high(int newscore,int mines)
updates the highscore
- int disprec()
displays the high score when the highscore screen is loaded

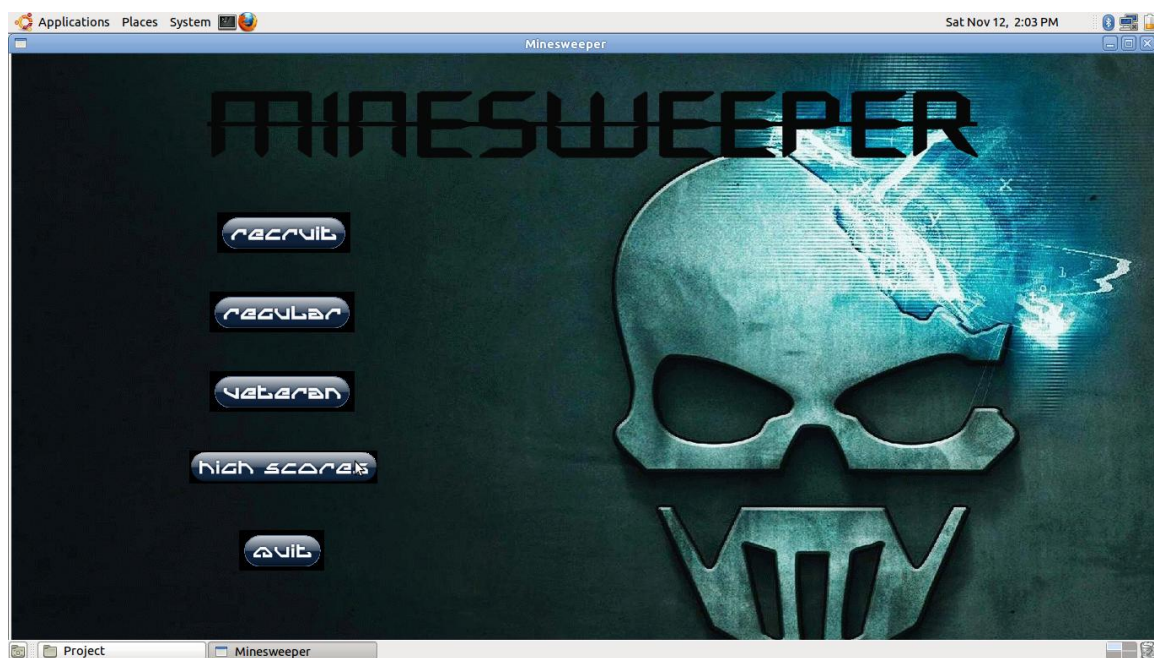
8. SYSTEM REQUIREMENTS

Ubuntu (Linux based OS)

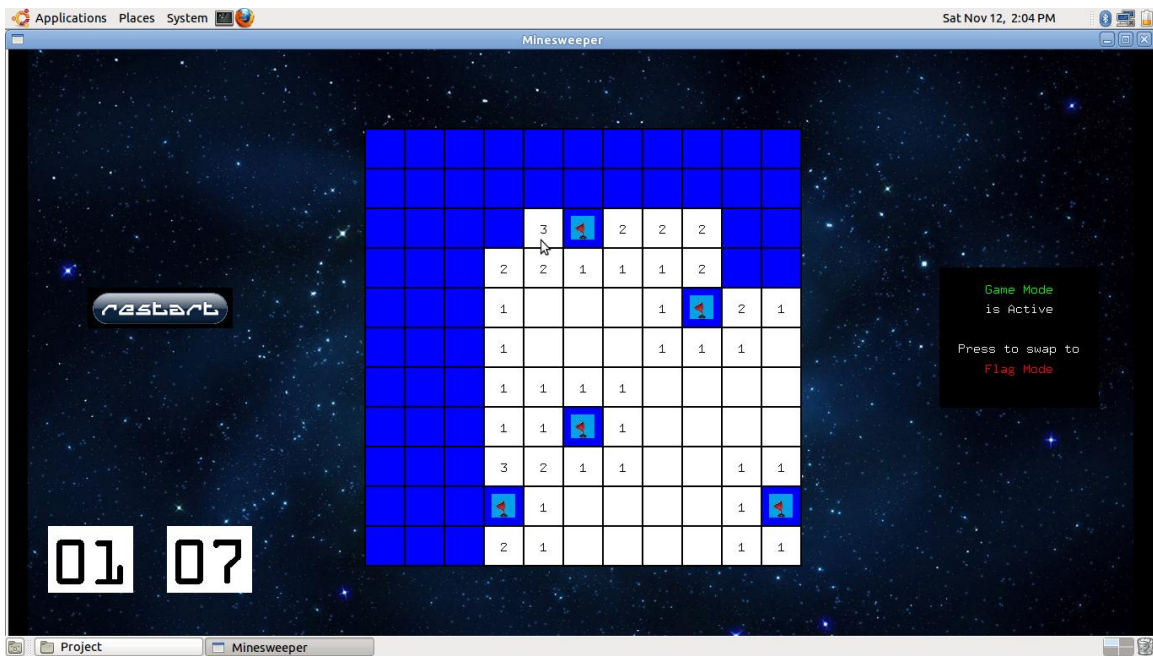
EzWindows Graphics Library

9. SCREENSHOTS

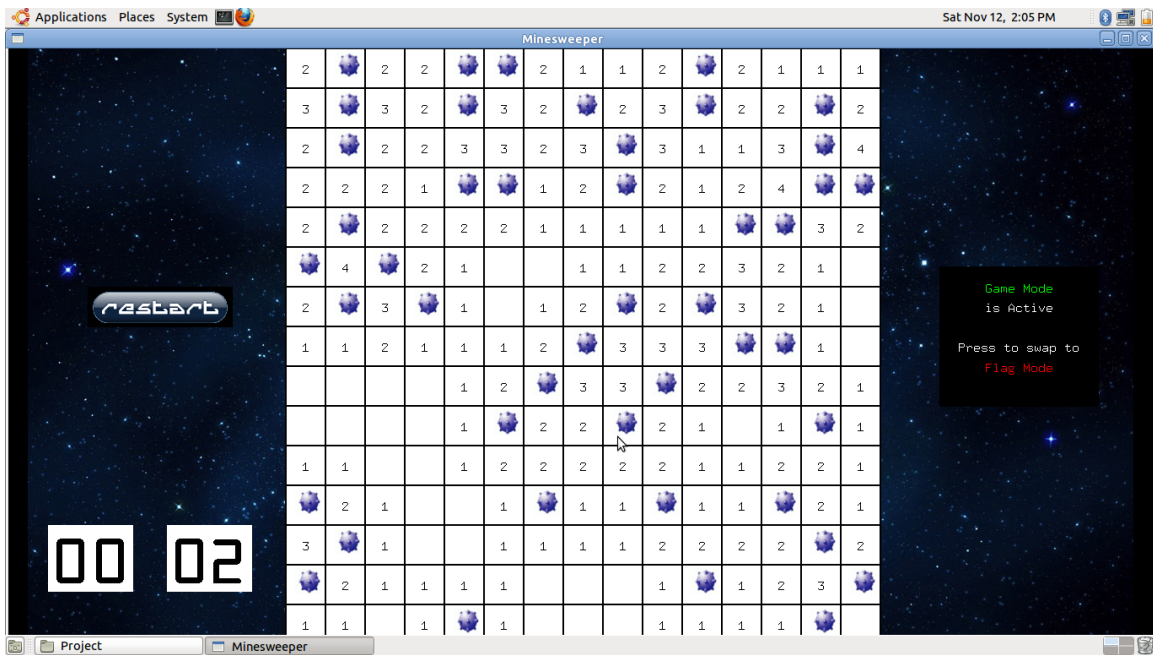
MAIN MENU



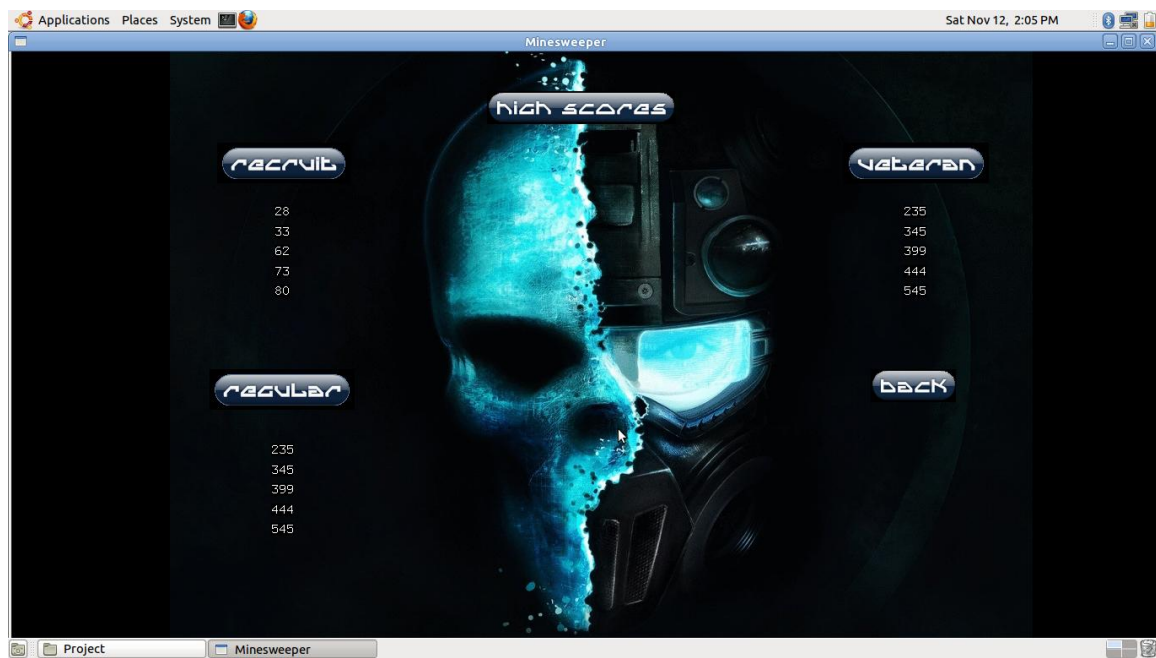
GAME SCREEN



GAME OVER



HIGH SCORE MENU



© Lab Batch 212