VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELGAVI-590014

Computer Graphics Laboratory
A Mini-Project Report
On

"MINESWEEPER ALGORITHM"

Submitted in partial fulfillment of the requirements for the 6th semester of
Bachelor of Engineering in Computer Science and Engineering *of Visvesvaraya*
Technological University, Belagavi

Submitted by:

BAPUSAHEB SHANKARAGOUDA PATIL   **1RN15CS033**

Under the Guidance of:

Mrs. S Mamatha Jajur   **Mrs. Sampada K S**
Assistant Professor   **Assistant Professor**
Dept. of CSE   **Dept. of CSE**

Department of Computer Science and Engineering
RNS Institute of Technology
Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560 098
2017-2018

RNS Institute of Technology
Channasandra, Dr.Vishnuvardhan Road,
Bengaluru-560 098

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

Certified that the mini project work entitled **"MINESWEEPER ALGORITHM"** has been successfully carried out by **BAPUSAHEB SHANKARAGOUDA PATIL** bearing USN **1RN15CS033**, bonafide student of **RNS Institute of Technology** in partial fulfillment of the requirements for the **6th semester** of **Bachelor of Engineering** in **Computer Science and Engineering** of **Visvesvaraya Technological University**, Belagavi, during the academic year 2017-18. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the Computer Graphics laboratory requirements of 6th semester BE, CSE.

Mrs. S Mamatha Jajur   **Mrs. Sampada K S**   **Dr. G T Raju**
Assistant Professor   **Assistant Professor**   **Professor and Head**
Dept. of CSE   **Dept. of CSE**   **Dept of CSE**

External Viva:
Name of the Examiners   **Signature with Date:**

1.

2.

ACKNOWLEDGEMENTS

Date: **Bapusaheb Shankaragouda Patil   1RN15CS033**
Place: Bengaluru

ABSTRACT

The Minesweeper Algorithm is based on the early logic of Jerimac Ratliff's Cube, a mainframe puzzle that was quickly succeeded by Relentless Logic, or RLogic, for short. In RLogic, the main goal is to navigate through the minefield, from the top left right angled corner to the bottom right right angled corner (the Command Center). It is not necessary to clear all non-mine squares. Also, there is no mechanism for marking mines or counting the number of mines found. There's no need to clear or flag all the mines at the end.

Building on top of this, Minesweeper Algorithm eliminated the need to navigate in a particular sense of direction in the minefield. Additionally, the number of steps taken to flag all the mines isn't counted in Minesweeper, as opposed to RLogic. However, it is one of the objectives of the algorithm that all the mines be flagged for the algorithm to terminate. The technologies used in this project are OpenGL and GLUT, with the entire codebase written in C++.

The Minesweeper Algorithm was most popularly used in the Windows Minesweeper game by Microsoft, a video game that came shipped with every version of Windows XP, Windows Vista and Windows 8. Another common but infrequent use of the Minesweeper Algorithm is in the Financial Resources Allocation problem, where the resources have to be allocated. Financial Resources Allocation problem is a subset of Constraint Processing and the Minesweeper Algorithm helps in getting a better idea of those constraints and helps solve them by reaching a final solution.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF COMPUTER GRAPHICS

The term computer graphics has been used in a broad sense to describe almost everything on computers that is not text or sound. Typically, the term *computer graphics* refers to several different things:

- The representation and manipulation of image data by a computer
- The various technologies used to create and manipulate images
- The images so produced, and
- The sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content.

Today, computers and computer-generated images touch many aspects of daily life. Computer images are found on television, in newspapers, for example in weather reports, in all kinds of medical investigation and surgical procedures. A well-constructed graph can present complex statistics in a form that is easier to understand and interpret. In the media such graphs are used to illustrate papers, reports, thesis, and other presentation material.Many powerful tools have been developed to visualize data. Computer generated imagery can be categorized into several different types: 2D, 3D, 4D, 7D, and animated graphics.

As technology has improved, 3D computer graphics have become more common. Computer graphics has emerged as a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content. Over the past decade, other specialized fields have been developed like information visualization, and scientific visualization more concerned with the visualization of three dimensional phenomena (architectural, meteorological, medical, biological, etc.), where the emphasis is on realistic renderings of volumes, surfaces, illumination sources, and so forth, perhaps with a dynamic component.

1.1.1 HISTORY

In 1959, the TX-2 computer was developed at MIT's Lincoln Laboratory. The TX-2 integrated a number of new man-machine interfaces. A light pen could be used to draw sketches on the computer using Ivan Sutherland's revolutionary Sketchpad software.[4] Using a light pen, Sketchpad allowed one to draw simple shapes on the computer screen, save them and even recall them later. The light pen itself had a small photoelectric cell in its tip. This cell emitted an electronic pulse whenever it was placed in front of a computer screen and the screen's electron gun fired directly at it. By simply timing the electronic pulse with the current location of the electron gun, it was easy to pinpoint exactly where the pen was on the screen at any given moment. Once that was determined, the computer could then draw a cursor at that location. Also in 1961 another student at MIT, Steve Russell, created the first video game, E. E. Zajac, a scientist at Bell Telephone Laboratory (BTL), created a film called "Simulation of a two-giro gravity attitude control system" in 1963.

During 1970s, the first major advance in 3D computer graphics was created at UU by these early pioneers, the hidden-surface algorithm. In order to draw a representation of a 3D object on the screen, the computer must determine which surfaces are "behind" the object from the viewer's perspective, and thus should be "hidden" when the computer creates (or renders) the image.

In the 1980s, artists and graphic designers began to see the personal computer, particularly the Commodore Amiga and Macintosh, as a serious design tool, one that could save time and draw more accurately than other methods. In the late 1980s, SGI computers were used to create some of the first fully computer-generated short films at Pixar. The Macintosh remains a highly popular tool for computer graphics among graphic design studios and businesses. Modern computers, dating from the 1980s often use graphical user interfaces (GUI) to present data and information with symbols, icons and pictures, rather than text. Graphics are one of the five key elements of multimedia technology.

3D graphics became more popular in the 1990s in gaming, multimedia and animation. In 1996, Quake, one of the first fully 3D games, was released. In 1995, Toy

Story, the first full-length computer-generated animation film, was released in cinemas worldwide. Since then, computer graphics have only become more detailed and realistic, due to more powerful graphics hardware and 3D modeling software.

1.1.2 APPLICATIONS

The applications of computer graphics can be divided into four major areas:

- **Display of information:** Computer graphics has enabled architects, researchers and designers to pictorially interpret the vast quantity of data. Cartographers have developed maps to display the celestial and geographical information. Medical imaging technologies like Computerized Tomography (CT), Magnetic Resonance Imaging (MRI), Ultrasound, Positron Emission Tomography (PET) and many others make use of computer graphics.

- **Design:** Professions such as engineering and architecture are concerned with design. They start with a set of specification; seek cost-effective solutions that satisfy the specification. Designing is an iterative process. Designer generates a possible design, tests it and then uses the results as the basis for exploring other solutions. The use of interactive graphical tools in Computer Aided Design (CAD) pervades the fields including architecture, mechanical engineering, and the design of very-large-scale integrated (VLSI) circuits and creation of characters for animation.

- **Simulation and animation:** Once the graphics system evolved to be capable of generating sophisticated images in real time, engineers and researchers began to use them as simulators. Graphical flight simulators have proved to increase the safety and to reduce the training expenses. The field of virtual reality (VR) has opened many new horizons. A human viewer can be equipped with a display headset that allow him/her to see the images with left eye and right eye which gives the effect of stereoscopic vision. This has further led to motion pictures and interactive video games.

- **User interfaces:** Computer graphics has led to the creation of graphical user interfaces (GUI) using which even naive users are able to interact with a computer. Interaction with the computer has been dominated by a visual paradigm that includes windows, icons, menus and a pointing device such as mouse. Millions of people are internet users; they access the internet through the

- graphical network browsers such as Microsoft internet explorer and Mozilla Firefox.

1.2 DESCRIPTION OF PROJECT

The Minesweeper Algorithm is a widely used algorithm in solving the Financial Resources Allocation problem and in the famous Microsoft game that comes shipped with every version of Windows since Windows XP, Windows Minesweeper.

The OpenGL implementation of this program is a resemblance to the above mentioned Minesweeper program. The main goal of the algorithm is to flag all mines with the help of the numbers allocated to the squares surrounding the mines without encountering a mine, given a grid of largely unexplored squares.

1.3 OVERVIEW TO OpenGL

OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation.

OpenGL provides a set of commands to render a three dimensional scene. That means you provide the data in an OpenGL-useable form and OpenGL will show this data on the screen (render it). It is developed by many companies and it is free to use. You can develop OpenGL-applications without licensing. OpenGL is a hardware- and system-independent interface. An OpenGL-application will work on every platform, as long as there is an installed implementation. Because it is system independent, there are no functions to create windows etc., but there are helper functions for each platform. A very useful thing is GLUT.

1.4 OpenGL LIBRARIES

Computer Graphics are created using OpenGL, which became a widely accepted standard software system for developing graphics applications. As a software interface for graphics hardware, OpenGL's main purpose is to render two- and three-dimensional objects into a frame buffer. These objects are described as sequences of vertices (which define geometric objects) or pixels (which define images). OpenGL performs several processing steps on this data to convert it to pixels to form the final desired image in the frame buffer.

OpenGL stands for 'open graphics library' graphics library is a collection of API's (Applications Programming Interface). Graphics library functions are:

1. **GL library** (OpenGL in windows) – Main functions for windows.
2. **GLU** (OpenGL utility library) - Creating and viewing objects.
3. **GLUT** (OpenGL utility toolkit)- Functions that help in creating interface of windows

OpenGL draws *primitives*—points, line segments, or polygons—subject to several selectable modes. You can control modes independently of each other; that is, setting one mode doesn't affect whether other modes are set (although many modes may interact to determine what eventually ends up in the frame buffer). Primitives are specified, modes are set, and other OpenGL operations are described by issuing commands in the form of function calls. These libraries are included in the application program using preprocessor directives

#include <GL/glut.h>

OpenGL User Interface Library (**GLUI**) is a C++ user interface library based on the OpenGL Utility Toolkit (GLUT) which provides controls such as buttons, checkboxes, radio buttons, and spinners to OpenGL applications. It is window and operating system independent, relying on GLUT to handle all system-dependent issues, such as window and mouse management.

The **OpenGL Utility Library** (**GLU**) is a computer graphics library. It consists of a number of functions that use the base OpenGL library to provide higher-level drawing routines from the more primitive routines that OpenGL provides. It is usually distributed with the base OpenGL package.

CHAPTER 2

REQUIREMENTS ANALYSIS

The Hardware Requirements are very minimal and the program can be run on most of the machines.

Processor: Intel Core i3 processor

Processor Speed: 1.70 GHz

RAM: 4 GB

Storage Space: 40 GB

Monitor Resolution: 1024x768 or 1336x768 or 1280x1024

2.2 SOFTWARE REQUIREMENTS

Operating System: Windows

IDE: Microsoft Visual Studio with C++ (version 6)

OpenGL libraries, Header Files which includes GL/glut.h, Object File Libraries, glu32.lib,opengl32.lib,glut32.lib,DLLfiles,glu32.dll,glut32.dll,opengl32.dll.

2.3 FUNCTIONAL REQUIREMENTS

Keyboard: The ENTER key is required to navigate the user from the splash screen to the main screen. The ESC key terminates the program and closes the window.

Mouse: The LEFT CLICK is required to reveal the underlying square that is clicked, and it can either be a mine, an empty square or a numbered square.

CHAPTER 3

SYSTEM DESIGN

3.1 MODULES

The description of all the functions used in the program is given below:

- **void glutInitDisplayMode (unsigned int mode)**

This function requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model (GLUT_RGB, GLUT_INDEX) and buffering (GLUT_SINGLE, GLUT_DOUBLE).

- **void glutInitWindowPosition (int x, int y)**

This specifies the initial position of top-left corner of the windows in pixels.

- **void glutInitWindowSize (int width, int height)**

This function specifies the initial height and width of the window in pixels.

- **void glutCreateWindow (char *title)**

This function creates a window on the display the string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.

- **void glutDisplayFunc (void (*func) (void))**

This function registers the display func that is executed when the window needs to be redrawn.

- **void glClearColor(GLclampfr,GLclampf g, GLclampfb,GLclampf a)**

This sets the present RGBA clear colour used when clearing the colour buffer. Variables of type GLclampf are floating point numbers between 0.0 and 1.0.

- **void glClear(GLbitfield mask)** It clear buffers to present values. The value of mask is determined by the bitwise OR of options GL_COLOR_BUFFER_BIT, GL_DEPTH_BUFFER_BIT

- **void glutPostRedisplay ()**

This function requests that the display callback be executed after the current callback returns.

- **void glutReshapeFunc (void *f (int width, int height)**

This function registers the reshape callback function f. The callback function returns the height and width of the new window. The reshape callback invokes a display callback.

- **void glViewport (int x, int y, GLsizei width, GLsizei height)**

This function specifies a width*height viewport in pixels whose lower left corner is at (x, y) measured from the origin of the window.

- **void glMatrixMode (GLenum mode)**

This function specifies which matrix will be affected by subsequent transformations. Mode can be GL_MODEL_VIEW, GL_PROJECTION, GL_TEXTURE.

- **void glLoadIdentity ()**

This function sets the current transformation matrix to an identity matrix.

- **void gluOrtho2D(GLdoubleleft, GLdouble right, GLdouble bottom, GLdouble top)**

This function defines a two-dimensional viewing rectangle in the plane z=0.

- **void glutMouseFunc (void *f (int button, int state, int x, int y)**

This function registers the mouse callback function f. The callback function returns the button(GLUT_LEFT_BUTTON,GLUT_MIDDLE_BUTTON,GLUT_RIGHT_BUTTON), the state of the button after the event (GLUT_UP, GLUT_DOWN), and the position of the mouse relative to the top-left corner of the window.

- **void glVertex3f(TYPE xcoordinate, TYPE ycoordinate, TYPE zcoordinate)**

void glVertex3fv(TYPE *coordinates)

This specifies the position of a vertex in 3 dimensions. If v is present, the argument is a pointer to an array containing the coordinates.

- **void glBegin(glEnum mode)**

This function initiates a new primitive of type mode and starts the collection of vertices. Values of mode include GL_POINTS, GL_LINES and GL_POLYGON.

- **void glEnd()**

This function terminates a list of vertices.

- **void glutMainLoop()**

This function causes the program to enter an event processing loop. It should be the last statement in main.

- **void glPushMatrix(void);**

Pushes all matrices in the current stack down one level. The current stack is determined by glMatrixMode(). The topmost matrix is copied, so its contents are duplicated in both the top and second-from-the-top matrix. If too many matrices are pushed, an error is generated.

Void glRotate{fd} (TYPE angle, TYPE x, TYPE y, TYPE z);

voidglTranslate{fd} (TYPE x, TYPE y, TYPE z);

Void glScale {fd} (TYPE x, TYPE y, TYPE z);

21. **Void glPopMatrix (void);**

Pops the top matrix off the stack, destroying the contents of the popped matrix. What was the second-from-the-top matrix becomes the top matrix. The current stack is determined byglMatrixMode(). If the stack contains a single matrix, calling glPopMatrix() generates an error.

- **glLoadName(ID) :**

To replace the top of the integer-ID name stack for picking operations

- **glSelectBuffer(size,pickbuff)** : To set up a pick-buffer array

- **glRenderMode(GL_SELECT) :** To activate OpenGL picking operations

- **glInitNames() :** To activate the integer-ID name stack for the picking operations

- **glPushName(ID) :** To place an unsigned interger value on the stack

- **glGet** :** Query function to copy specified state values into an array(requires specification of data type,symbolic name of a state parameter and an array pointer)

- **gluPickMatrix(xPick,yPick,widthPick,heightPick,vpArray) :** To define a pick window within a selected viewport

- **glPopMatrix()** : To destroy the matix on top of the stack and to make the second matrix on the stack become the current matrix

- **glutMouseFunc(mousefcn)** : Specify a mouse callback function that is to be invoked when a mouse button is pressed

- **glutMotionFunc(motionfcn) :** Specify a mouse callback function that is to be invoked when the mouse cursor is moved while a button is pressed

- **glutKeyboardFunc(keyboardfcn)** : Specify a keyboard callback function that is to be invoked when a standard key is pressed

- **glutCreateMenu(menuFunc) :** To create a popup menu and specify the procedure to be invoked when a menu item is selected

- **glutAddMenuEntry(charString,menuItemNumber) :** To list the name and position for each option

- **glutAttachMenu(button)** : To specify a mouse button that is to be used to select a menu option

3.2 PSEUDOCODE & LOGIC
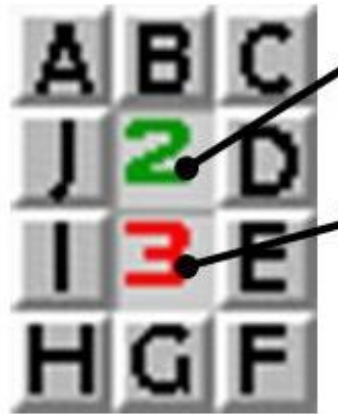
Minesweeper Algorithm is based on solving the Constraint Satisfaction Problem and explore the application of constraint propagation techniques to interactively determine safe and mined squares.

Every square is a variable with two possible values: Safe or Mined.

Every safe square gives a "sum" constraint over its 8 neighbors. For example, a square labeled 3 yields a constraint stating the square be surrounded by 3 mines.

From the above diagram, the constraints can be represented as:

- Constraint C1:
  Scope = {A, B, C, D, E, I, J} – Exactly 2 must have mines

- Constraint C2:
  Scope = {D, E, F, G, H, I, J} - Exactly 3 must have mines

CHAPTER 4

IMPLEMENTATION

- **Calculation of Adjacent Mines**

```c
void calc_adjacent_mines(int i,int j,int dim)
{
  //row above mines
  if(i-1>=0)
  {
    if(j-1>=0)
      if(board[i-1][j-1].content==MINE)
        board[i][j].content++;
    if(board[i-1][j].content==MINE)
      board[i][j].content++;

    if(j+1<dim)
      if(board[i-1][j+1].content==MINE)
        board[i][j].content++;
  }
  if(j-1>=0)
    if(board[i][j-1].content==MINE)
      board[i][j].content++;
  if(j+1<dim)
    if(board[i][j+1].content==MINE)
      board[i][j].content++;

  if(i+1<dim)
  {
    if(j-1>=0)
      if(board[i+1][j-1].content==MINE)
```

```
        board[i][j].content++;
    if(board[i+1][j].content==MINE)
        board[i][j].content++;
    if(j+1<dim)
        if(board[i+1][j+1].content==MINE)
            board[i][j].content++;
  }
}
```

● **Laying Out The Mines**

```
void init_mines(int num,int dim)
{
  int i=0,j=0,count=0;
  while(count!=num)
  {

    i=rand()%dim;
    j=rand()%dim;

    if(board[i][j].content!=MINE)
    {
      board[i][j].content=MINE;
      count++;
      game_stats.add_mine(i,j);
    }
  }
}
```

CHAPTER 5

TESTING AND RESULTS

5.1 TEST CASES

In unit testing, the program modules that make up the system are tested individually. Unit testing focuses to locate errors in the working modules that are independent to each other. This enables to detect errors in coding and the logic within the module alone. This testing is also used to ensure the integrity of the data stored. The various routines were checked by passing the inputs and the corresponding output is tested. Test cases used in the project as follows:

Serial No	Metric	Description	Observation
1	**Keyboard Function**	Key ENTER to start the program Key ESC to exit the program	Results obtained as expected
2	**Display Function**	Board initialization, mines initialization, Squares initialization	Results obtained as expected
3	**Animation Effect**	Squares are uncovered to reveal if the underlying element is a mine or not.	Results obtained as expected

5.2 SCREENSHOTS/SNAPSHOTS
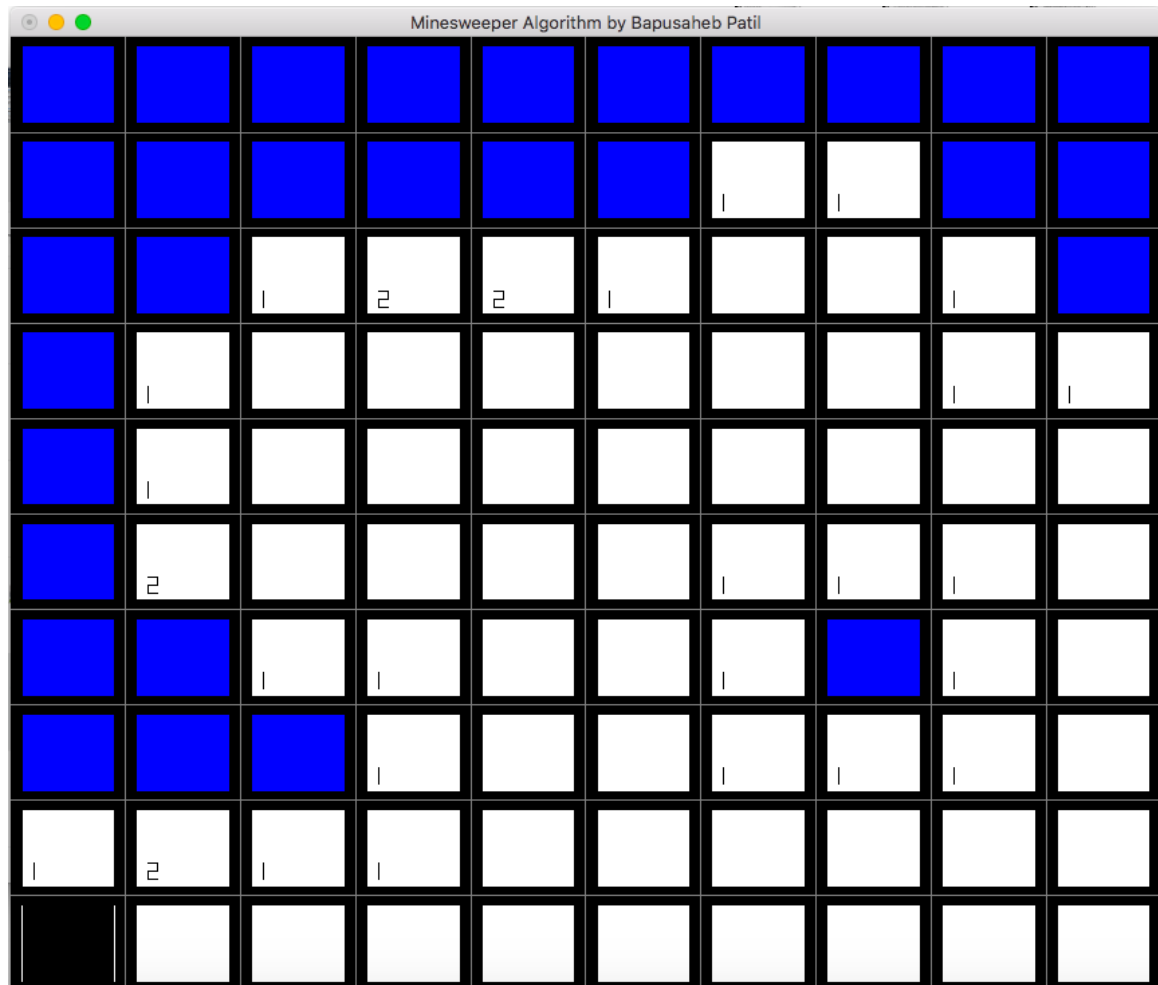
Main Screen of the program

Figure 1: Main Screen of the program

This is a screenshot of what a running instance of the Minesweeper Algorithm looks like in the OpenGL implementation of the algorithm.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENTS

The current OpenGL implementation of the Minesweeper Algorithm uses a fixed dimension of the square grids. An improvement could be made in this regard in the future, where the user can select the difficulty level and select the size of the grid himself/herself.

Additionally, the highest number of flagged mines can be stored as a 'best score', not unlike a high score in a video game, which shows a list of users who have flagged the highest number of mines in consecutive sessions.

Apart from all this, the number of mines that are laid out on the grid is also something that could be interfaced on the GUI and made that particular customization available to the user, as opposed to making it customizable only via the program code.

REFERENCES

1. W3Schools, C++

2. MIT OpenGL Tutorials

3. Harvard Handbook of OpenGL Tips

4. www.opengl.org.

5. www.cs.uiowo.edu/~cwyman/classes/common

6. www.graphicsonline.com

7. www.stackoverflow.com