# Visvesvaraya Technological University

**"JnanaSangama", Belagavi – 590018, Karnataka**

A

A Mini Project Report

on

**"Tic Tac Toe"**

*Submitted in partial fulfillment of the requirement for the Computer Graphics Laboratory with Mini Project (18CSL67) of VI semester*

*Bachelor of Engineering in*
*Computer Science and Engineering*

Submitted By

**Nayan V Bhandari (1GA19CS095)**

Under the Guidance of

**Mrs. Vanishree.M.L**
**Assistant Professor,**
**Dept of CSE**

**Dr. Bhagyashri R Hanji**
**Professor & Head,**
**Dept of CSE**

# GLOBAL ACADEMY OF TECHNOLOGY

## Department of Computer Science and Engineering

**(Accredited by NBA 2019-2022)**

**Rajarajeshwari Nagar, Bengaluru – 560 098**

**2021-2022**

# GLOBAL ACADEMY OF TECHNOLOGY

## Department of Computer Science and Engineering

**(Accredited by NBA 2019-2022)**

**Rajarajeshwari Nagar, Bengaluru – 560 098**

# CERTIFICATE

This is to certify that VI Semester Mini Project entitled "**TIC TAC TOE"** is a bonafide work carried out by **Nayan V Bhandari (1GA19CS095)** as a partial fulfillment for the award of Bachelor's Degree in Computer Science and Engineering for Computer Graphics Laboratory with Mini Project [18CSL67] as prescribed by **Visvesvaraya Technological University**, **Belagavi** during the year 2021-2022.

---

**Mrs. Vanishree M L**
Assistant Professor,
Dept of CSE,
GAT, Bengaluru

---

**Dr. Bhagyashri R Hanji**
Professor & Head,
Dept of CSE,
GAT, Bengaluru

**External Exam**

| Name of the Examiner | Signature with date |
|---|---|
| 1. _____ | _____ |
| 2. _____ | _____ |

# GLOBAL ACADEMY OF TECHNOLOGY

## Department of Computer Science and Engineering

**(Accredited by NBA 2019-2022)**

**Rajarajeshwari Nagar, Bengaluru – 560 098**

# DECLARATION

I ,Nayan V Bhandari, bearing USN 1GA19CS095 ,student of Sixth Semester B.E, Department of Computer Science and Engineering, Global Academy of Technology, Rajarajeshwarinagar Bengaluru, declare that the Mini Project entitled "**Tic Tac Toe"** has been carried out by me and submitted in partial fulfillment of the course requirements for the award of degree in Bachelor of Engineering in Computer Science and Engineering from Visvesvaraya Technological University, Belagavi during the academic year 2021-2022.

**Nayan V Bhandari**

**(1GA19CS095)**

Place: Bengaluru

Date: 15/07/2022

# ABSTRACT

Tic-tac-toe is a game for two players who take turns marking the spaces in a three-by-three grid with X or O. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner. It is a two-player game. This project is the implementation of the same game using openGL functions.

# ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance crowned our efforts with success.

I consider myself proud, to be part of **Global Academy of Technology** family, the institution which stood by our way in endeavors.

I express my deep and sincere thanks to our Principal **Dr. N. Ranapratap Reddy** for his support.

I am grateful to **Dr. Bhagyashri R Hanji**, Professor and HOD, Dept of CSE who is source of inspiration and of invaluable help in channelizing my efforts in right direction.

I wish to thank our internal guide **Prof. Vanishree M L,** Assistant Professor for guiding and correcting various documents with attention and care. She has taken lot of pain to go through the document and make necessary corrections as and when needed.

I would like to thank the faculty members and supporting staff of the Department of CSE, GAT for providing all the support for completing the Project work.

Finally, I am grateful to our parents and friends for their unconditional support and help during the course of our Project work.

**Nayan V Bhandari (1GA19CS095)**

# TABLE OF CONTENTS

# CHAPTER 1

# INTRODUCTION

## 1.1 INTRODUCTION TO COMPUTERGRAPHICS

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. Graphics provides one of the most natural means of communicating within a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and effectively. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television.

**Applications of Computer Graphics**

1. Display of information
2. Design
3. Simulation and animation
4. User interfaces

**The Graphics Architecture**

Graphics Architecture can be made up of seven components:

1. Display processors
2. Pipe linear architectures
3. The graphics pipeline
4. Vertex processing
5. Clipping and primitive assembly
6. Rasterization
7. Fragment processing

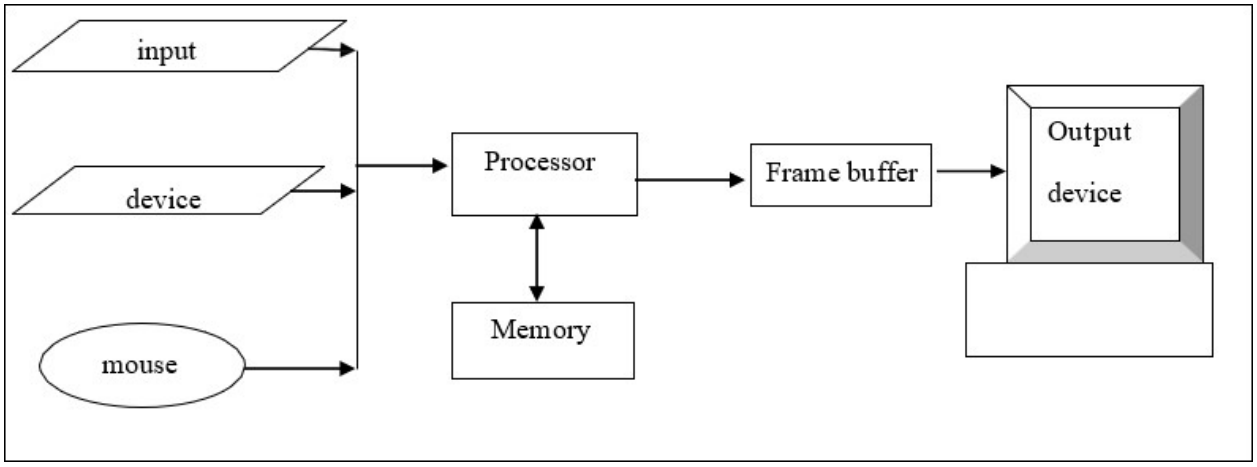


**Figure 1.1: Components of Graphics Architecture and their working**

## 1.2 INTRODUCTION TO OPENGL

OpenGL is software used to implement computer graphics. The structure of OpenGL is similar to that of most modern APIs including Java 3D and DirectX. OpenGL is easy to learn, compared with other.

APIs are nevertheless powerful. It supports the simple 2D and 3D programs. It also supports the advanced rendering techniques. OpenGL API explains following 3 components

1. Graphics functions
2. Graphics pipeline and state machines
3. The OpenGL interfaces

There are so many polygon types in OpenGL like triangles, quadrilaterals, strips and fans. There are 2 control functions, which will explain OpenGL through,

1. Interaction with window system
2. Aspect ratio and viewport

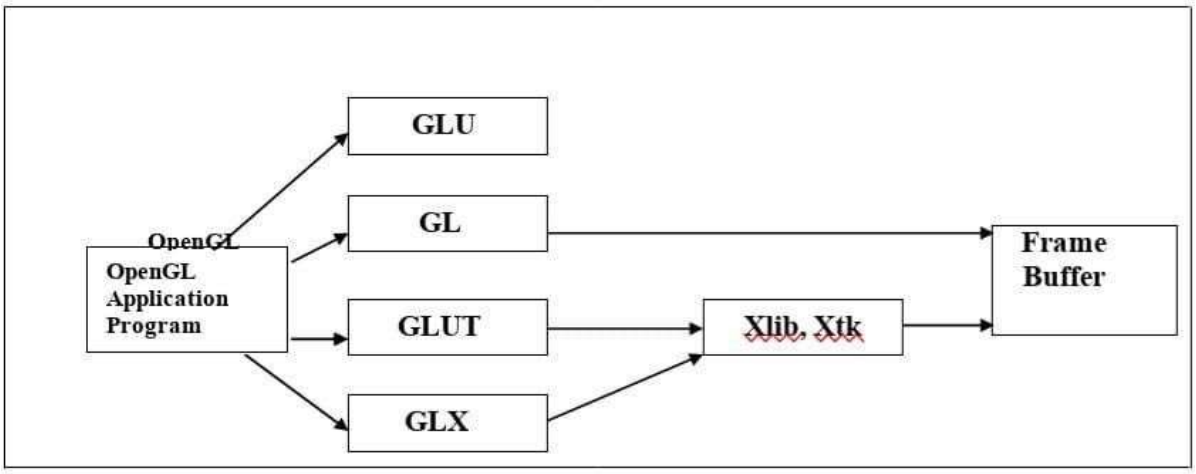


**Figure 1.2: OpenGL Library organization**

Most implementations of OpenGL have a similar order of operations, a series of processing stages called the OpenGL rendering pipeline. This ordering, as shown in Figure 1.2, isnot a strict rule of how OpenGL is implemented but provides a reliable guide for predicting what OpenGL will do. The following diagram shows the assembly line approach, which OpenGL takes to process data. Geometric data (vertices, lines, and polygons) follow the path through the row of boxes that includes evaluators and per-vertex operations, while pixel data (pixels, images, and bitmaps) are treated differently for part of the process. Both types of data undergo the same final steps before the final pixel data is written into the frame buffer.

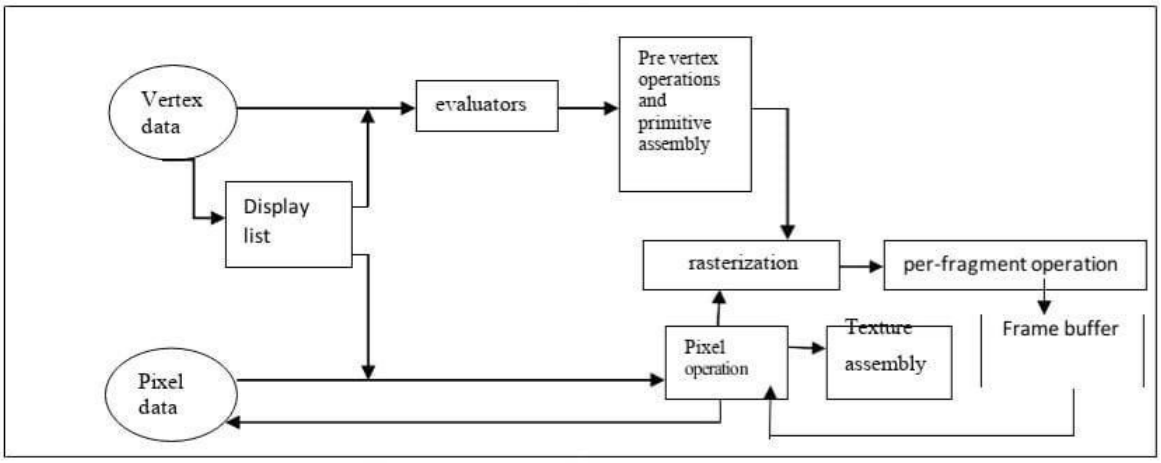


**Figure 1.3: OpenGL Order of Operations**

# CHAPTER 2

## REQUIREMENTS SPECIFICATION

### 2.1 SOFTWAREREQUIREMENTS

- Operating system – Windows10
- Code::Blocks 17.12
- OPENGL library files – GL, GLU,GLUT
- Language used is C/C++

### 2.2 HARDWAREREQUIREMENTS

- Processor –Intel i5 7$^{th}$ Gen
- Memory –8GBRAM
- 1TB Hard Disk Drive
- Mouse or other pointing device
- Keyboard
- Display device

# CHAPTER 3

## SYSTEM DEFINITION

### 3.1 PROJECT DESCRIPTION

Computer graphics involves the designing of objects in different forms which are regular and irregular in shape. The mini project named “Train Animation” creates a scenery with railway tracks and a building. This project is designed and implemented using OpenGL interactive application that basically deals with providing the graphical interface between the user and the system. Through keyboard events the movement of the train can be moved and stopped.

### 3.2 USER DEFINED FUNCTIONS

- **glutInit (int *argc, char **argv):** glutInit is used to initialize the GLUT library.
- **glutInitDisplayMode(unsigned int mode):**glutInitDisplayMode sets the initial display mode.
- **glutInitWindowPosition(intx,int y):** Specifies the initial position of the top-left corner of the window in pixels.
- **glutInitWindowSize(intwidth,int height):** Specifies the initial height and width of the window in pixels.
- **glutKeyboardFunc (void *f(char key,intwidth,int height):**KeyboardFunc sets the keyboard callback for the current window.
- **glClear():**The clear function clears buffers to preset values.
- **glClearColor(GLclampf r, GLclampf g, GLclampf b, GLclampf a):**The glClearColor function specifies clear values for the color buffers.
- **glMatrixMode(GLenum mode):**This function specifies which matrix is the current matrix.
- **glLoadIdentity():** Set the current transformation matrix to an identity matrix.

- **glPushMatrix(),glPopMatrix():** This pushes to and pops from the matrix stack corresponding to the current matrix mode.
- **glPointSize():**The **glPointSize** function specifies the diameter of rasterized points.
- **glTranslate[fd](TYPE x,TYPEy,TYPE z):** This function multiplies the current matrix by a translation matrix.
- **gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top):**The **gluOrtho2D** function defines a 2-D orthographic projection matrix.
- **glBegin (glEnum mode):** It initiates a new primitive of type mode and starts the collection of vertices. Values of mode include GL_POINTS, GL_LINES, GL_LINE_STRIP, and GL_POLYGON.
- **glEnd():** It terminates a list of vertices.
- **glFlush ():**It forces any buffered openGL commands to execute.
- **glutMainLoop() :**Itcauses the program to enter an event processing loop.
- **glutDisplayFunc(void (*func)(void)):**It registers the display function 'func' that is executed when the window needs to be redrawn.
- **glutCreateMenu(void (*f)(void)):** It returns an identifier for a top-level menu and registers the callback function f that returns an interger value corresponding to the menu entry selected.

## 3.3 DATA FLOW DIAGRAM

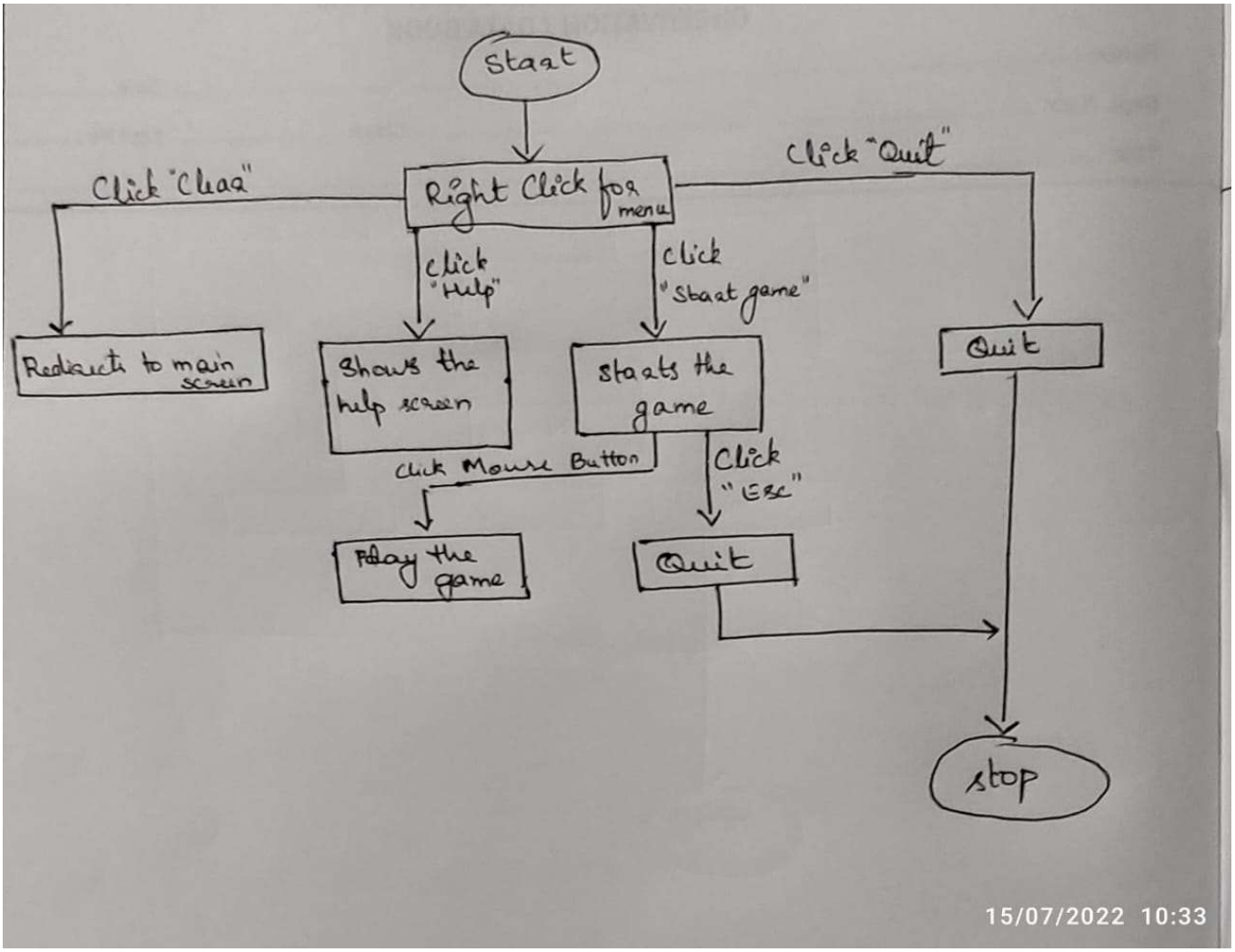


Fig 3.3: Flow Diagram

# CHAPTER 4

## IMPLEMENTATION

### 4.1 SOURCE CODE

```
#include <windows.h>
#include<glut.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
// lighting
GLfloat LightAmbient[] = { 1.0f, 0.0, 0.0f };
GLfloat LightDiffuse[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat LightPosition[] = { 0.0, 1.0, 0.0, 1.0 };
GLfloat LightSpecular[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat mat_shininess = 5.0;
int abc = 0;
// mouse variables: Win = windows size, mouse = mouse position
int mouse_x, mouse_y, Win_x, Win_y, object_select;
// Use to spin X's and O's
int spin, spinboxes;
// Win = 1 player wins, -1 computer wins, 2 tie
//player or computer; 1 = X, -1 = O,
//start_game indicates that game is in play.
int player, computer, win, start_game;
// alignment of boxes in which one can win
// We have 8 possibilities, 3 across, 3 down and 2 diagonally
// 0 | 1 | 2
// 3 | 4 | 5
// 6 | 7 | 8
// row, column, diagonal information
static int box[8][3] = { { 0, 1, 2 }, { 3, 4, 5 }, { 6, 7, 8 }, { 0, 3, 6 }, { 1, 4, 7 }, { 2, 5, 8 }, { 0, 4, 8
}, { 2, 4, 6 } };
```

```
// Storage for our game board
// 1 = X's, -1 = O's, 0 = open space
int box_map[9];
// center x,y location for each box
int object_map[9][2] = { { -6, 6 }, { 0, 6 }, { 6, 6 }, { -6, 0 }, { 0, 0 }, { 6, 0 }, { -6, -6 }, { 0, -6 }, { 6, -6 } }; // quadric pointer for build our X
GLUquadricObj* Cylinder;
// Begin game routine
void init_game(void)
{
        int i;
        // Clear map for new game
        for (i = 0; i < 9; i++)
                box_map[i] = 0;
        win = 0;      // Set 0 for no winner
        start_game = 1;
}
// Check for three in a row/column/diagonal
// returns 1 if there is a winner
int check_move(void)
{
        int i, t = 0;
        //Check for three in a row
        for (i = 0; i < 8; i++)
        {
                t = box_map[box[i][0]] + box_map[box[i][1]] + box_map[box[i][2]];
                if ((t == 3) || (t == -3))
                {
                        spinboxes = i;
                        return (1);
                }
        }
        t = 0;
        // check for tie
        for (i = 0; i < 8; i++)
```

```
                t = t + abs(box_map[box[i][0]]) + abs(box_map[box[i][1]]) +
abs(box_map[box[i][2]]);
        if (t == 24)
                return (2);
        return (0);
}
// Do we need to block other player?
int blocking_win(void)
{
        int i, t;
        for (i = 0; i < 8; i++)
        {
                t = box_map[box[i][0]] + box_map[box[i][1]] + box_map[box[i][2]];
                if ((t == 2) || (t == -2))
                {
                        // Find empty
                        if (box_map[box[i][0]] == 0)
                                box_map[box[i][0]] = computer;
                        if (box_map[box[i][1]] == 0)
                                box_map[box[i][1]] = computer;
                        if (box_map[box[i][2]] == 0)
                                box_map[box[i][2]] = computer;
                        return (1);
                }
        }
        return (0);
}
// check for a free space in corner
int check_corner(void)
{
        int i;
        if (box_map[0] == 0)
        {
                box_map[0] = computer;
                i = 1;
```

```
                return (1);
        }
        if (box_map[2] == 0)
        {
                box_map[2] = computer;
                i = 1;
                return (1);
        }
        if (box_map[6] == 0)
        {
                box_map[6] = computer;
                i = 1;
                return (1);
        }
        if (box_map[8] == 0)
        {
                box_map[8] = computer;
                i = 1;
                return (1);
        }
        return (0);
} // Check for free space in row
int check_row(void)
{
        if (box_map[4] == 0)
        {
                box_map[4] = computer;
                return (1);
        }
        if (box_map[1] == 0)
        {
                box_map[1] = computer;
                return (1);
        }
        if (box_map[3] == 0)
```

```
        {
                box_map[3] = computer;
                return (1);
        }
        if (box_map[5] == 0)
        {
                box_map[5] = computer;
                return (1);
        }
        if (box_map[7] == 0)
        {
                box_map[7] = computer;
                return (1);
        }
        return (0);
}
// logic for computer's turn
int computer_move()
{
        if (blocking_win() == 1)
                return (1);
        if (check_row() == 1)
                return (1);
        if (check_corner() == 1)
                return (1);
        return (0);
}
// I use this to put text on the screen
void Sprint(int x, int y, const char* st)
{
        int l, i;
        l = strlen(st);         // see how many characters are in text string.
        glRasterPos2i(x, y); // location to start printing text
        for (i = 0; i < l; i++) // loop until i is greater then l
        {
```

```
                glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, st[i]); // Print a
character on the screen
        }
}
// This creates the spinning of the cube.
static void TimeEvent(int te)
{
        spin++; // increase cube rotation by 1
        if (spin > 360)
                spin = 180;             // if over 360 degress, start back at zero.
        glutPostRedisplay();          // Update screen with new rotation data
        glutTimerFunc(8, TimeEvent, 1); // Reset our timmer.
}
// Setup our Opengl world, called once at startup.
void init(void)
{
        glClearColor(0.0, 0.0, 0.0, 0.0); // When screen cleared, use black.
        glShadeModel(GL_FLAT);        // How the object color will be rendered smooth or flat
        glEnable(GL_DEPTH_TEST);        // Check depth when rendering
        //Lighting is added to scene
        glLightfv(GL_LIGHT1, GL_AMBIENT, LightAmbient);
        glLightfv(GL_LIGHT1, GL_DIFFUSE, LightDiffuse);
        glLightfv(GL_LIGHT1, GL_SPECULAR, LightSpecular);
        glLightfv(GL_LIGHT1, GL_POSITION, LightPosition);
        glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);
        glShadeModel(GL_SMOOTH);
        glEnable(GL_NORMALIZE);
        glEnable(GL_LIGHTING); // Turn on lighting
        glEnable(GL_LIGHT1);   // Turn on light 1
        start_game = 0;
        win = 0;
        // Create a new quadric
        Cylinder = gluNewQuadric();
        gluQuadricDrawStyle(Cylinder, GLU_FILL);
        gluQuadricNormals(Cylinder, GLU_SMOOTH);
```

```
        gluQuadricOrientation(Cylinder, GLU_OUTSIDE);
}
void Draw_O(int x, int y, int z, int a)
{
        glPushMatrix();
        glTranslatef(x, y, z);
        glRotatef(a, 1, 0, 0);
        glutSolidTorus(0.3, 1.0, 10, 300);
        glPopMatrix();
}
void Draw_X(int x, int y, int z, int a)
{
        glPushMatrix();
        glTranslatef(x, y, z);
        glPushMatrix();
        glRotatef(a, 1, 0, 0);
        glRotatef(90, 0, 1, 0);
        glRotatef(45, 1, 0, 0);
        glTranslatef(0, 0, -2);
        gluCylinder(Cylinder, 0.3, 0.3, 4, 10, 16);
        glPopMatrix();
        glPushMatrix();
        glRotatef(a, 1, 0, 0);
        glRotatef(90, 0, 1, 0);
        glRotatef(315, 1, 0, 0);
        glTranslatef(0, 0, -2);
        gluCylinder(Cylinder, 0.3, 0.3, 4, 10, 16);
        gluCylinder(Cylinder, 0.3, 0.3, 4, 10, 16);
        glPopMatrix();
        glPopMatrix();
}
// Draw our world
void display(void)
{
        if (abc == 2)
```

```
        {
                glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //Clear the
screen
                glColor3f(0.0, 0.0, 0.0);
                glMatrixMode(GL_PROJECTION);            // Tell opengl that we are doing project
matrix work
                glLoadIdentity();                       // Clear the matrix
                glOrtho(-9.0, 9.0, -9.0, 9.0, 0.0, 30.0); // Setup an Ortho view
                glMatrixMode(GL_MODELVIEW);             // Tell opengl that we are doing model
matrix work. (drawing)
                glLoadIdentity();                       // Clear the model matrix
                glDisable(GL_COLOR_MATERIAL);
                glDisable(GL_LIGHTING);
                glColor3f(1,0,0);
                Sprint(-7, 5, "Project by");
                glColor3f(1,1,1);
                Sprint(-7, 4, "Nayan Bhandari USN-1GA19CS095");
                glColor3f(1,0,0);
                Sprint(-7, 2, "Overview");
                glColor3f(1,1,1);
                Sprint(-7, 1, "Tic Tac Toe is a two-player game where the first player to");
                Sprint(-7, 0, "connect a line of pieces from one side or corner of the board to the
other wins.");
                glColor3f(1,0,0);
                Sprint(-7, -2, "Rules");
                glColor3f(1,1,1);
                Sprint(-7, -3, "1.X moves first.");
                Sprint(-7, -4, "2.A piece may be placed on any empty space.");
                Sprint(-7, -5, "3.A player wins by being the first to connect a line of friendly");
                Sprint(-7, -6, "  pieces from one side or corner of the board to the other.");
                Sprint(-7, -7, "4.The game ends when either one player wins or it is no longer
possible");
                Sprint(-7, -8, "  for a player to win (in which case the result is a draw).");
                glutSwapBuffers();
        }
```

```
        else if (abc == 0 || abc == 4)
        {
                glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //Clear the
screen
                glClearColor(0, 0, 0, 1);
                glMatrixMode(GL_PROJECTION);    // Tell opengl that we are doing project
matrix work
                glLoadIdentity();                       // Clear the matrix
                glOrtho(-9.0, 9.0, -9.0, 9.0, 0.0, 30.0);   // Setup an Ortho view
                glMatrixMode(GL_MODELVIEW);     // Tell opengl that we are doing model
matrix work. (drawing)
                glLoadIdentity();                       // Clear the model matrix
                glDisable(GL_COLOR_MATERIAL);
                glDisable(GL_LIGHTING);
                glColor3f(1, 1, 1);
                Sprint(-2, 7, "Global Academy of Technology");
                Sprint(-3, 6, "Department of Compute Science and Engineering");
                Sprint(-4, 4, "A MINI PROJECT ON COMPUTER GRAPHICS AND
VISUALIZATION: ");
                Sprint(-1, 3, "      Tic Tac Toe");
                Sprint(-2.5, 1, "Student Name : Nayan V Bhandari");
                Sprint(-1.5, 0, "USN : 1GA19CS095");
                Sprint(-1.2, -2, "Under the guidance of");
                Sprint(-3.2, -3, "    Dr.Bhagyashri R Hanji          Prof Vanishree M L");
                Sprint(-3.2, -4, "    Prof and Head            and     Assistant Professor");
                Sprint(-3.2, -5, "    Dept of CSE                      Dept of CSE");
                Sprint(-1, -6, "Academic Year : 2021-22");
                Sprint(3, -8, "RIGHT CLICK FOR MENU");
                glutSwapBuffers();
        }
        else if(abc==1)
        {
                //char txt[30];
                int ix, iy;
                int i;
```

```
        int j;
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //Clear the
screen
        glMatrixMode(GL_PROJECTION);    // Tell opengl that we are doing project
matrix work
        glLoadIdentity();                       // Clear the matrix
        glOrtho(-9.0, 9.0, -9.0, 9.0, 0.0, 30.0);    // Setup an Ortho view
        glMatrixMode(GL_MODELVIEW);   // Tell opengl that we are doing model matrix
work. (drawing)
        glLoadIdentity();                       // Clear the model matrix
        glDisable(GL_COLOR_MATERIAL);
        glDisable(GL_LIGHTING);
        glColor3f(1,1,1);
        if (win == 1)
                Sprint(-1.5, 8, "Congratulations! You Won");
        if (win == -1)
                Sprint(0, 8, "Computer Won");
        if (win == 2)
                Sprint(0, 8, "Draw");
        // Setup view, and print view state on screen
        glColor3f(0.0, 0.0, 1.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(60, 1, 1, 30);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        glEnable(GL_LIGHTING);
        gluLookAt(0, 0, 20, 0, 0, 0, 0, 1, 0);
        // Draw Grid
        for (ix = 0; ix < 4; ix++)
        {
                glPushMatrix();
                glColor3f(1, 1, 1);
                glBegin(GL_LINES);
                glVertex2i(-9, -9 + ix * 6);
```

```
                glVertex2i(9, -9 + ix * 6);
                glEnd();
                glPopMatrix();
        }
        for (iy = 0; iy < 4; iy++)
        {
                glPushMatrix();
                glColor3f(1, 1, 1);
                glBegin(GL_LINES);
                glVertex2i(-9 + iy * 6, 9);
                glVertex2i(-9 + iy * 6, -9);
                glEnd();
                glPopMatrix();
        }
        for (i = 0; i < 9; i++)
        {
                j = 0;
                if (abs(win) == 1)
                {
                        if ((i == box[spinboxes][0]) || (i == box[spinboxes][1]) || (i ==
box[spinboxes][2]))
                        {
                                j = spin;
                        }
                        else
                                j = 0;
                }
                if (box_map[i] == 1)
                {
                        Draw_O(object_map[i][0], object_map[i][1], -1, j);
                }
                if (box_map[i] == -1)
                {
                        Draw_X(object_map[i][0], object_map[i][1], -1, j);
                }
```

```
                }
                glutSwapBuffers();
        }
}
// This is called when the window has been resized.
void reshape(int w, int h)
{
        Win_x = w;
        Win_y = h;
        glViewport(0, 0, (GLsizei)w, (GLsizei)h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
} // Read the keyboard
void keyboard(unsigned char key, int x, int y)
{
        switch (key)
        {
        case 27:
                exit(0); // exit program when [ESC] key presseed
                break;
        default:
                break;
        }
}
void mouse(int button, int state, int x, int y)
{
        // We convert windows mouse coords to out openGL coords
        mouse_x = (18 * (float)((float)x / (float)Win_x)) / 6;
        mouse_y = (18 * (float)((float)y / (float)Win_y)) / 6;
        // What square have they clicked in?
        object_select = mouse_x + mouse_y * 3;
        if (start_game == 0)
        {
                if ((button == GLUT_RIGHT_BUTTON) && (state == GLUT_DOWN))
                {
```

```
                player = 1;
                computer = -1;
                init_game();
                computer_move();
                return;
        }
        if ((button == GLUT_LEFT_BUTTON) && (state == GLUT_DOWN))
        {
                player = -1;
                computer = 1;
                init_game();
                return;
        }
}
if (start_game == 1)
{
        if ((button == GLUT_LEFT_BUTTON) && (state == GLUT_DOWN))
        {
                if (win == 0)
                {
                        if (box_map[object_select] == 0)
                        {
                                box_map[object_select] = player;
                                win = check_move();
                                if (win == 1)
                                {
                                        start_game = 0;
                                        return;
                                }
                                computer_move();
                                win = check_move();
                                if (win == 1)
                                {
                                        win = -1;
                                        start_game = 0;
```

```
                                        }
                                    }
                                }
                        }
                }
                if (win == 2)
                        start_game = 0;
        }
        void menu(int choice)
        {
                switch (choice)
                {
                case 1:
                        abc = 1;
                        glutMouseFunc(mouse);
                        break;
                case 2:
                        abc = 2;
                        glutMouseFunc(mouse);
                        break;
                case 3:
                        exit(0);
                        break;
                case 4:
                        abc = 4;
                        break;
                }
        }
        int main(int argc, char** argv)
        {
                glutInit(&argc, argv);
                glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
                glutInitWindowSize(1500, 800);
                glutInitWindowPosition(10, 10);
                glutCreateWindow("Tic Tac Toe");
```

```
        init();
        glutCreateMenu(menu);
        glutAddMenuEntry("Start game", 1);
        glutAddMenuEntry("Help", 2);
        glutAddMenuEntry("Quit", 3);
        glutAddMenuEntry("Clear", 4);
        glutAttachMenu(GLUT_RIGHT_BUTTON);
        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        glutKeyboardFunc(keyboard);
        glutTimerFunc(50, TimeEvent, 1);
        glutMainLoop();
        return 0;
}
```

# CHAPTER 5

# TESTING AND RESULTS

## 5.1 DIFFERENT TYPES OF TESTING

### 1. Unit Testing

Individual components are tested to ensure that they operate correctly. Each component is tested independently, without other system components.

### 2. Module Testing

A module is a collection of dependent components such as a object class, an abstract Data type or some looser collection of procedures and functions. A module related Components, so can be tested without other system modules.

### 3. System Testing

This is concerned with finding errors that result from unanticipated interaction between Sub-system interface problems.

### 4. Acceptance Testing

The system is tested with data supplied by the system customer rather than simulated test data.

## 5.2 TEST CASES

The test cases provided here test the most important features of the project.

**Table 5.2.1: Test Case**

| SL No | Test Input | Expected Results | Observed Results | Remarks |
|---|---|---|---|---|
| 1 | Right click the mouse | Menu should appear | Menu appears | Pass |
| 2 | Click on menu item “HELP” | Shows the help section | Shows Help Section | Pass |
| 3 | Click on menu item “START GAME” | It should Start the game | The game starts | Pass |
| 4 | Left Click the mouse on empty boxes | Should fill it with ‘X’ | Fills with ‘X’ | Pass |
| 5 | Left Click after game ends | Should start a new game | Starts a new game | Pass |
| 6 | Click the button ‘Esc’ | Should Quit | Quits | Pass |
| 7 | Click on menu item “CLEAR” | Clears screen and comes to Main Page | Screen cleared and comes to main page | Pass |
| 8 | Click on menu item “QUIT” | Exit | Exit | Pass |

# CHAPTER 6

# SNAPSHOTS

## Snapshots of Tic Tac Toe:

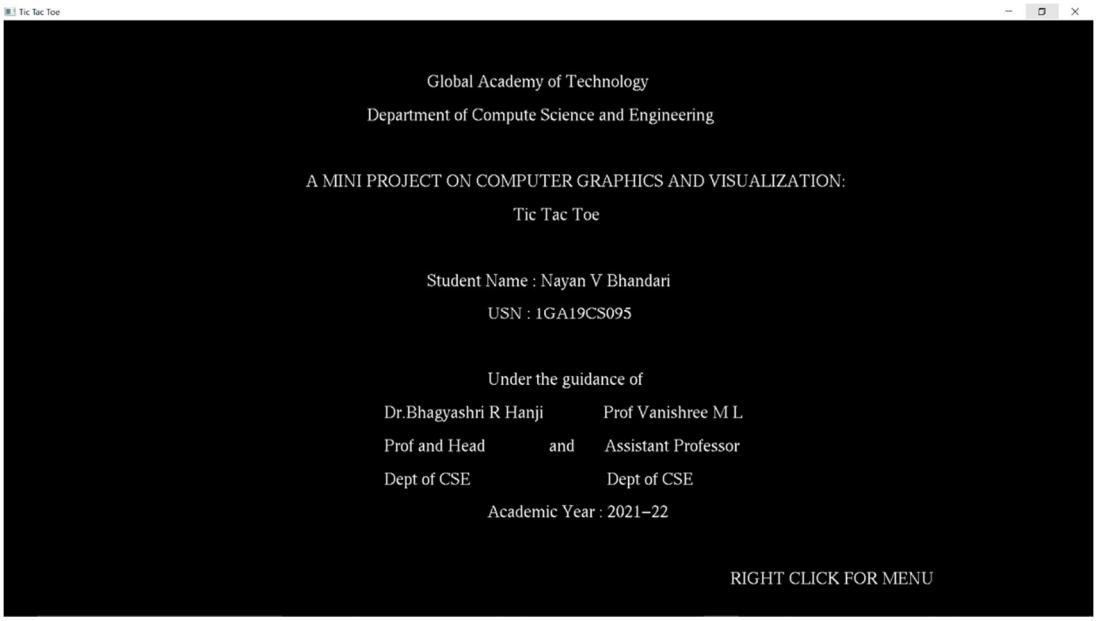


Fig 6.1: STARTING PAGE

A snapshot of the main page with menu

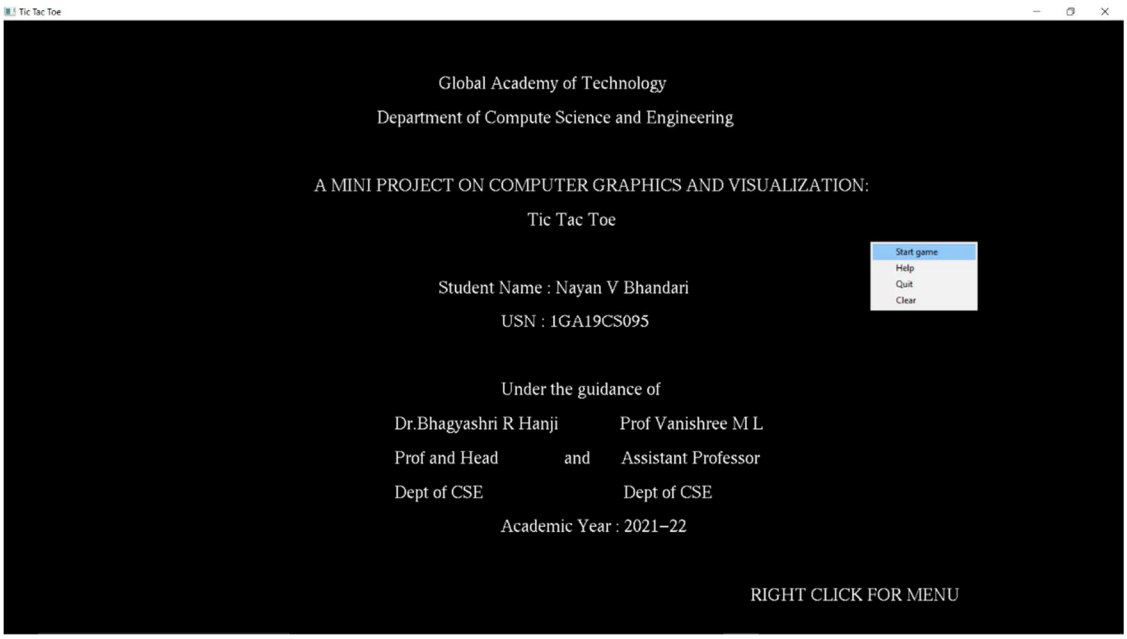


Fig 6.2: STARTING PAGE WITH MENU

A snapshot of empty tic tac toe board

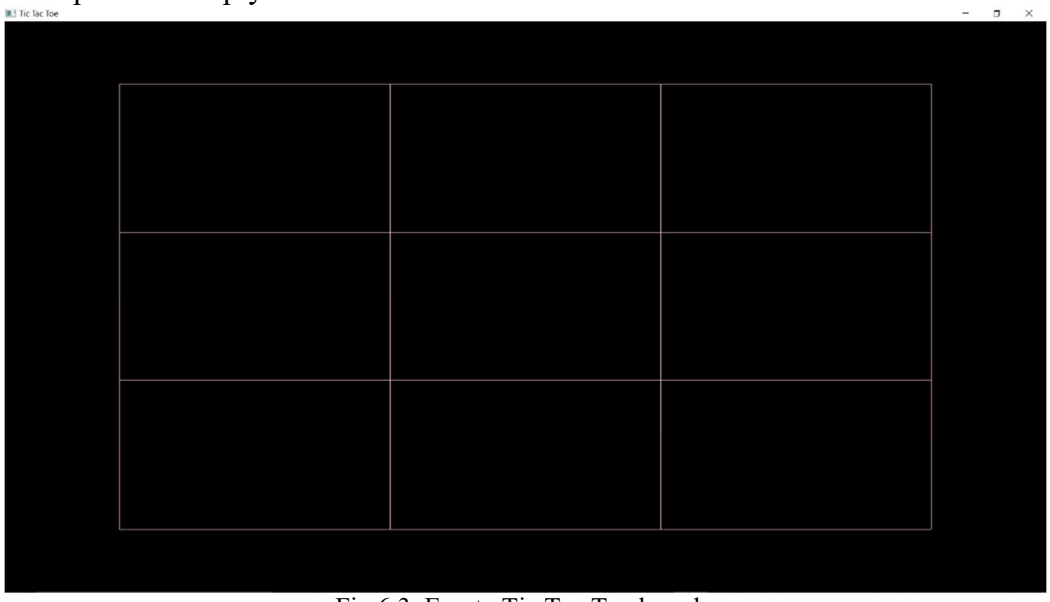


Fig 6.3: Empty Tic Tac Toe board

The snapshot were game is being played

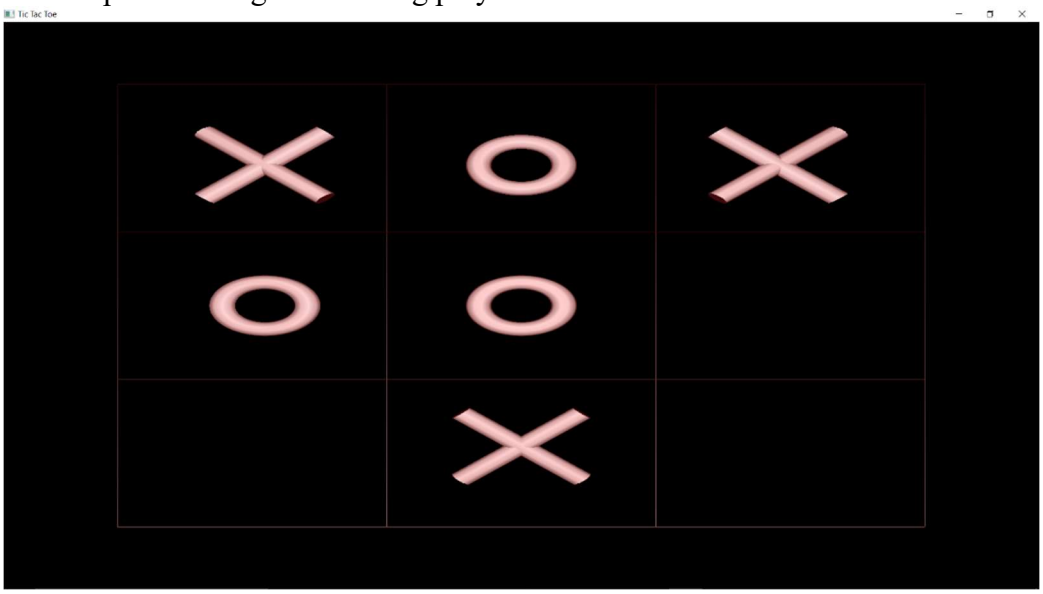


Fig 6.4: Game being played

This snapshot shows when Computer wins the game

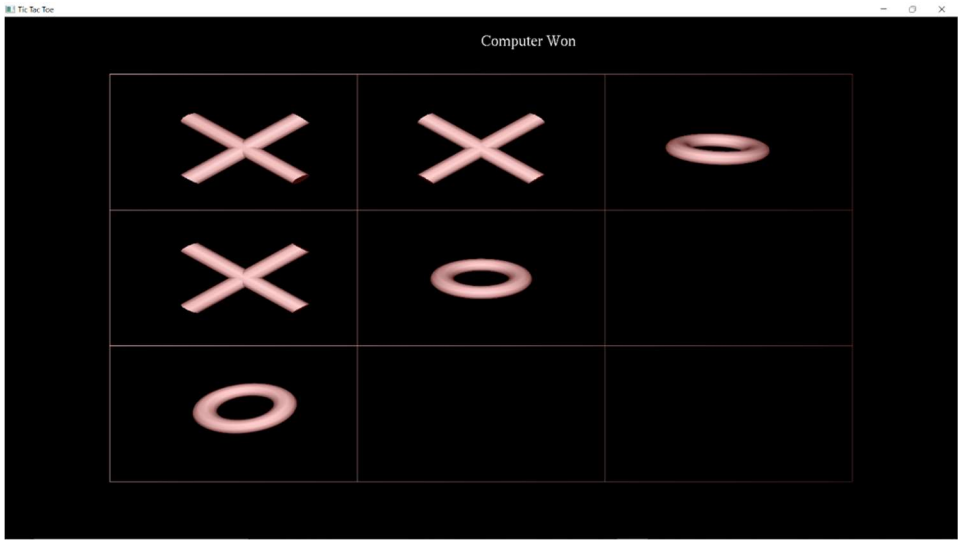


Fig 6.5: Computer wins the game

The snapshot when the game is a draw

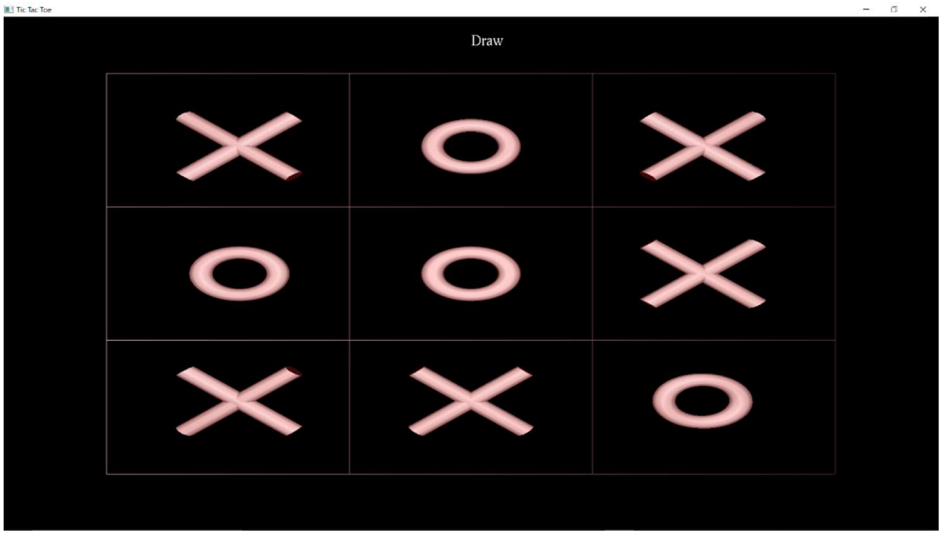


Fig 6.6: A draw game

This snapshot shows the user wins the game

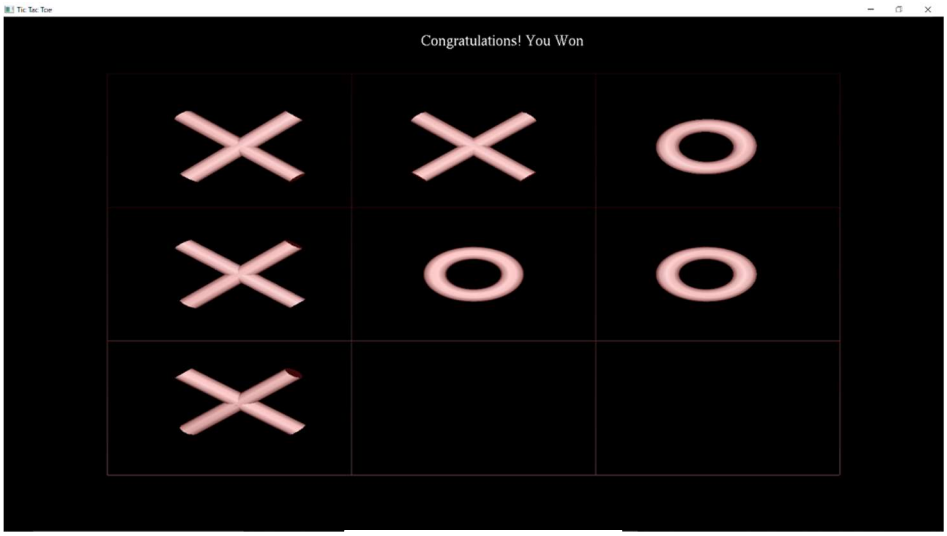


Fig 6.7: User wins

The snapshot shows the help button in menu

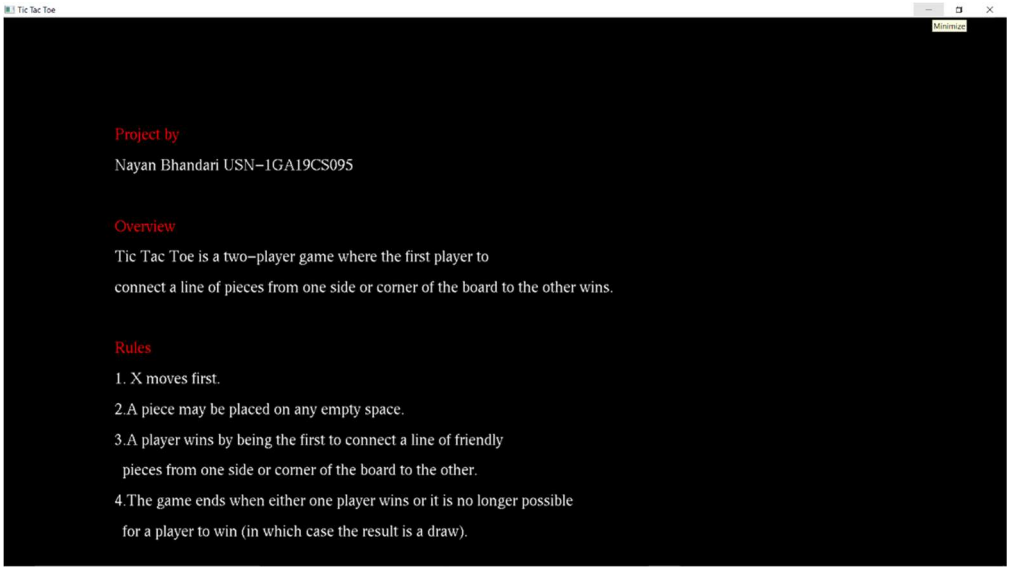


Fig 6.8: Help in the Menu button

# CONCLUSION

The Tic Tac Toe Graphics package has been developed Using OpenGL. The main idea of the program is to play the tic tac toe game. It is a game versus the computer. The user starts the game and by clicking the empty boxes he places an ‘X’ there. Once the game ends it shows the result saying whether the user won, or the computer or it was a draw. Along with this it also shows the implementation of Menu in OpenGL.

# BIBLIOGRAPHY

## References

1. Edward Angel: Interactive Computer Graphics: A Top-Down Approach with 5th Edition, Addison-wesley,2008.

2. James D Foley, Andries Van Dam, Steven K Feiner, Jhon F Hughes: - Computer Graphics, Addison-Wesley 1997.

3. Yashavanth Kanetkar, "Graphics under opengl", Published by BPB Publications.

4. Athul P.Godse, "Computer Graphics",Technical Publications Pune, 2 nd Revised Edition.

5. James D.foley, Andries Van Dam, Steven K.Feiner, John F.Hughes, "Computer Graphics Principle & Practice" Published by Pearson Education Pte.ltd, 2nd Edition.