Mini Project

Tic-Tac-Toe Using OpenGL
CSE-304

Gudladhana Harshith
Registration no: 11903248

Submitted to
Dr. Om Prakash Yadav (26121)
Asst. Professor,
School of Computer Science and Engineering

LOVELY PROFESSIONAL UNIVERSITY
Jalandhar – Delhi, Grand Trunk Rd
Phagwara, Punjab, 144001
March, 2023

Abstract:

OpenGL is a widely used graphics API that enables developers to create interactive 2D and 3D graphics applications across a variety of platforms. This project aims to explore the capabilities of OpenGL and create a basic graphics application that can be used as a starting point for more advanced projects. The application will showcase the use of various OpenGL features, such as vertex buffers, shaders, textures, and lighting, and will provide users with the ability to interact with the graphics through keyboard and mouse inputs. The project will also include a documentation outlining the application's design, implementation, and future development possibilities. Through this project, the developers aim to gain a deeper understanding of OpenGL and its potential for creating visually stunning and interactive applications.

Introduction:

OpenGL is a powerful and widely used graphics API that provides developers with a comprehensive set of tools for creating high-quality 2D and 3D graphics applications across various platforms. It is highly optimized for hardware acceleration, making it a popular choice for games, scientific simulations, architectural visualizations, and other real-time graphics applications.

This project aims to explore the capabilities of OpenGL and create a basic graphics application that can serve as a foundation for more advanced projects. The application will demonstrate the use of various OpenGL features, such as vertex buffers, shaders, textures, and lighting, to create visually appealing and interactive graphics. It will also provide users with the ability to interact with the graphics through keyboard and mouse inputs, making the application more engaging and user-friendly.

In addition to the application itself, the project will include a detailed documentation outlining the design, implementation, and potential future development opportunities. This documentation will serve as a valuable resource for developers interested in learning more about OpenGL and its capabilities.

Support Libraries:

OpenGL has three support libraries like GL, GLU, GLUT. The developments of utility libraries have greatly extended the low-level functionality of OpenGL. GL provides lower-level functions for drawing points, lines and polygons. All GL function names start with “gl”. GLU provides functions for drawing more complex primitives of OpenGL such as curves and surfaces. It also functions to help specify 3D views of scenes. All GLU function start with “glu”.

GLUT provides the facilities for interaction that OpenGL lacks. It provides functions for managing windows on the display screen and handling input events from the mouse and keyboard. It provides some rudimentary tools for creating Graphical User Interface (GUIs).

System Requirements:

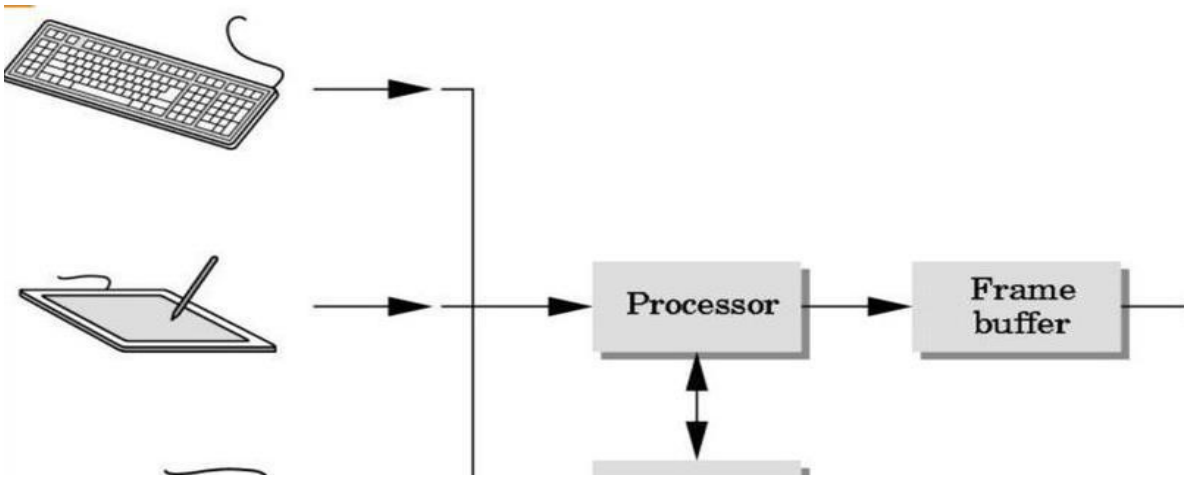
•Input devices: Graphics systems provide a keyboard and another one input device. Most common is the mouse, the joy stick and the data tablet. This provides the positional information

• Processor: The main function of the graphical processor is to take specification of the graphical primitives. Generated by the application programs and to assign values to the pixels in the frame buffer that best represents these entities.

• Frames Buffer: The frame buffers are the core elements of the graphics system. Its resolution determines the detail of the image we can view.

• Memory: Frame buffer is usually implemented with special type of memories chips that enable fast redisplays on the contents of the frame buffer.

• Output Devices: The dominant type of the display or monitor has been cathode ray tube Flat panel technologies are now more popular.



Software Requirements:

To use OpenGL in computer graphics, you need an operating system that supports it, a graphics card that supports OpenGL, an OpenGL implementation, a programming language that supports OpenGL like C++, a development environment which is Dev C++, OpenGL utility libraries, and debugging tools.

Hardware Requirements:

To use openGL we need hardware like a proper working Monitor with Keyboard, mouse, CPU, RAM

CODE FOR THE Tic-Tac-Toe Using OpenGL:

```
// skk_ttt_v3.0.cpp : Defines the entry point for the console application.
//


#include <iostream>
#include <GL/glut.h>  // Glut (gl utility toolkit) basic windows functions, keyboard, mouse.
#include <stdio.h>    // Standard (I/O library)
#include <stdlib.h>   // Standard library (set of standard C functions
#include <math.h>     // Math library (Higher math functions )
#include <string.h>             // String library
int p[3]={0,0,0};
int ch=0;                       // Color Scheme Choice
int board[3][3];    // Board for gameplay
int turn;                       // Current move
int result;                             // Result of the game
bool over;                              // Is the game Over?
GLfloat v[5][4][3]={
        {{0,0,0},{0.984375,0.289063,0.101563},{0.964844,0.714844,0.199219},{1,1,1}},
        {{0.184314,0.309804,0.309804},{1,1,1},{0,0,0},{0,0,0}},
        {{0,0.74902,1},{0.678431,1,0.184313},{1,1,0},{0.678431,0,0.184314}},
        {{1,0.611765,0.356862},{0.929411,0.188235,0.235294},{0.231372,0.505882,0.513725},{0.960784,0.
388235,0.290196}},
        {{0.941176,0.847059,0.658823},{0.949019,0.839215,0.580392},{0.980392,0.164706,0},{0.525490,0.
721568,0.694117}}
                                        };

char *pint(int i)
{
        switch(i)
        {
         case 0:
                 return " 0 ";
                 break;
         case 1:
                return " 1 ";
                 break;
         case 2:
                return " 2 ";
                 break;
         case 3:
                 return " 3 ";
                 break;
         case 4:
                 return " 4 ";
                 break;
         case 5:
                 return " 5 ";
                 break;
         default:
                 return "NA";
        }
}
//      Sets the board for Tic Tac Toe
void Intialize()
{
        turn=1;
```

```
	for(int i=0;i<3;i++)
		for(int j=0;j<3;j++)
			board[i][j]=0;
}
//	Called when any key from keyboard is pressed
void Keyboard(unsigned char key,int x,int y)
{
	switch(key)
	{
		case 'y':
		if(over==true)
		{
			if(result==0)
				p[0]++;
			else if(result==1)
				p[1]++;
			else if(result==2)
				p[2]++;
			over=false;
			Intialize();
		}
		break;
		case 'n':
		if(over==true)
			exit(0);
		break;

		default:
			exit(0);
	}
}
//	Called when Mouse is clicked
void Mouse(int button,int state,int x,int y)
{
	if(over==false && button==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
	{
		if(turn==1)
		{
			if(board[(y-50)/100][x/100]==0)
			{
				board[(y-50)/100][x/100]=1;
				turn=2;
			}
		}
		else if(turn==2)
		{
			if(board[(y-50)/100][x/100]==0)
			{
				board[(y-50)/100][x/100]=2;
				turn=1;
			}
		}
	}
}
//	Utility function to draw string
void DrawString(void *font,const char s[],float x,float y)
{
	unsigned int i;
	glColor3fv(v[ch][3]);
	glRasterPos2f(x,y);
```

```
	for(i=0;i<strlen(s);i++)
	{
		glutBitmapCharacter(font,s[i]);
	}

}
//	Function to draw up the horizontal and vertical lines
void DrawLines()
{
	glLineWidth(4.0);
	glBegin(GL_LINES);
	  glColor3fv(v[ch][0]);

	   glVertex2f(2,50);
		glVertex2f(2,380);

		glVertex2f(100,50);
		glVertex2f(100,410);

		glVertex2f(200,50);
		glVertex2f(200,410);

		glVertex2f(299,50);
		glVertex2f(299,380);

		glVertex2f(50,350);
		glVertex2f(50,380);

		glVertex2f(150,350);
		glVertex2f(150,380);

		glVertex2f(250,350);
		glVertex2f(250,380);


	    glVertex2f(0,50);
		 glVertex2f(300,50);

		 glVertex2f(0,150);
		 glVertex2f(300,150);

		 glVertex2f(0,250);
		 glVertex2f(300,250);

		 glVertex2f(0,350);
		 glVertex2f(300,350);

		 glVertex2f(0,380);
		 glVertex2f(300,380);

		 glVertex2f(100,410);
		 glVertex2f(200,410);

	glEnd();
	glLineWidth(1.f);
}
//	Utility function to draw the circle
void DrawCircle(float cx, float cy, float r, int num_segments)
{
	glColor3fv(v[ch][1]);
```

```
        glLineWidth(5.0);
    glBegin(GL_LINE_LOOP);
    for (int i = 0; i < num_segments; i++)
    {
        float theta = 2.0f * 3.1415926f * float(i) / float(num_segments);//get the current angle
        float x = r * cosf(theta);//calculate the x component
        float y = r * sinf(theta);//calculate the y component
        glVertex2f(x + cx, y + cy);//output vertex
    }
    glEnd();
        glLineWidth(1.f);
}
//      Function to draw the cross and circle of Tic Tac Toe
void DrawXO()
{
        for(int i=0;i<3;i++)
        {
                for(int j=0;j<3;j++)
                {
                        if(board[i][j]==1)
                        {
                                glColor3fv(v[ch][2]);
                                glLineWidth(3.0);
                                glBegin(GL_LINES);
                                glVertex2f(50 + j * 100 - 25, 100 + i * 100 - 25);
                                glVertex2f(50 + j * 100 + 25, 100 + i * 100 + 25);
                                glVertex2f(50 + j * 100 - 25, 100 + i * 100 + 25);
                                glVertex2f(50 + j * 100 + 25, 100 + i * 100 - 25);
                                glEnd();
                        }
                        else if(board[i][j]==2)
                        {
                                DrawCircle(50 + j*100 , 100 + i*100 , 25 , 15);
                        }
                }
        }
        glLineWidth(1.f);
}
//      Function to check if there is any winner
bool CheckWinner()
{
        int i, j;
        for(i=0;i<3;i++)// horizontal check
        {
                for(j=1;j<3;j++)
                {
                        if(board[i][0]!=0 && board[i][0]==board[i][j])
                        {
                                if(j==2)
                                {
                                        return true;
                                }
                        }
                        else
                                break;
                }
        }
        for(i=0;i<3;i++)// vertical check
        {
                for(j=1;j<3;j++)
```

```
			{
				if(board[0][i]!=0 && board[0][i]==board[j][i])
				{
					if(j==2)
						return true;
				}
				else
					break;
			}
		}
		if(  (board[0][0]!=0 && board[0][0]==board[1][1] && board[0][0]==board[2][2])
		  || (board[2][0]!=0 && board[2][0]==board[1][1] && board[2][0]==board[0][2])  )  // Diagonal check
			return true;
		return false;
}
//		Function to check if there is draw
bool CheckIfDraw()
{
		int i, j;
		bool draw;
		for(i=0;i<3;i++)
		{
			for(j=0;j<3;j++)
			{
				if(board[i][j]==0)
					return false;
			}
		}
		return true;
}
//		Function to display up everything
void Display()
{

		glClear(GL_COLOR_BUFFER_BIT);
		if(turn == 1)
			DrawString(GLUT_BITMAP_TIMES_ROMAN_24, "X's Turn", 105, 20);
		else
			DrawString(GLUT_BITMAP_TIMES_ROMAN_24, "O's Turn", 105, 40);
		DrawLines();
		DrawXO();
		if(CheckWinner() == true)
		{
			if(turn == 1)
			{
				over = true;
				result = 2;
			}
			else
			{
				over = true;
				result = 1;
			}
		}
		else if(CheckIfDraw() == true)
		{
			over = true;
			result = 0;
		}
		if(over == true)
```

```
	{
		if(result == 0)
			DrawString(GLUT_BITMAP_HELVETICA_18, "It's a draw", 107, 400);
		if(result == 1)
			DrawString(GLUT_BITMAP_HELVETICA_18, "X wins", 125, 400);
		if(result == 2)
			DrawString(GLUT_BITMAP_HELVETICA_18, "O wins", 125, 400);
		DrawString(GLUT_BITMAP_HELVETICA_18, "Press 'y' to Continue | 'n' to Exit ", 20, 430);
	}
		DrawString(GLUT_BITMAP_TIMES_ROMAN_24,"X",18,372);
		//DrawString(GLUT_BITMAP_HELVETICA_18,pint(p[1]),62,370);
		DrawString(GLUT_BITMAP_TIMES_ROMAN_24,"O",118,372);
		DrawString(GLUT_BITMAP_HELVETICA_18,pint(p[2]),162,370);
		DrawString(GLUT_BITMAP_TIMES_ROMAN_24,"D",218,372);
		DrawString(GLUT_BITMAP_HELVETICA_18,pint(p[0]),262,370);
		//DrawString(GLUT_BITMAP_TIMES_ROMAN_24, "|" 300,400);// Code for Name
		//DrawString(GLUT_BITMAP_TIMES_ROMAN_24, "11903248", 300, 402);//Code for Reg
No

glBegin(GL_LINES);
glColor3f(1.0, 0.0, 0.0);
/* 1 */
glVertex2i(300, 400);
glVertex2i(300, 450);
glEnd();

glBegin(GL_LINES);
/* 1 */
glVertex2i(325, 400);
glVertex2i(325, 450);
glEnd();
glBegin(GL_LINES);
/* 9 */
glVertex2i(350, 400);
glVertex2i(350, 425);
glVertex2i(350, 450);
glVertex2i(375, 450);
glVertex2i(375, 450);
glVertex2i(375, 400);
glVertex2i(375, 400);
glVertex2i(350, 400);
glVertex2i(350, 425);
glVertex2i(375, 425);
glEnd();
glBegin(GL_LINES);
/* 0 */
glVertex2i(400, 400);
glVertex2i(400, 450);
glVertex2i(400, 450);
glVertex2i(425, 450);
glVertex2i(425, 450);
glVertex2i(425, 400);
glVertex2i(425, 400);
glVertex2i(400, 400);
glEnd();
glBegin(GL_LINES);
/* 3 */
glVertex2i(450, 400);
glVertex2i(475, 400);
glVertex2i(475, 400);
```

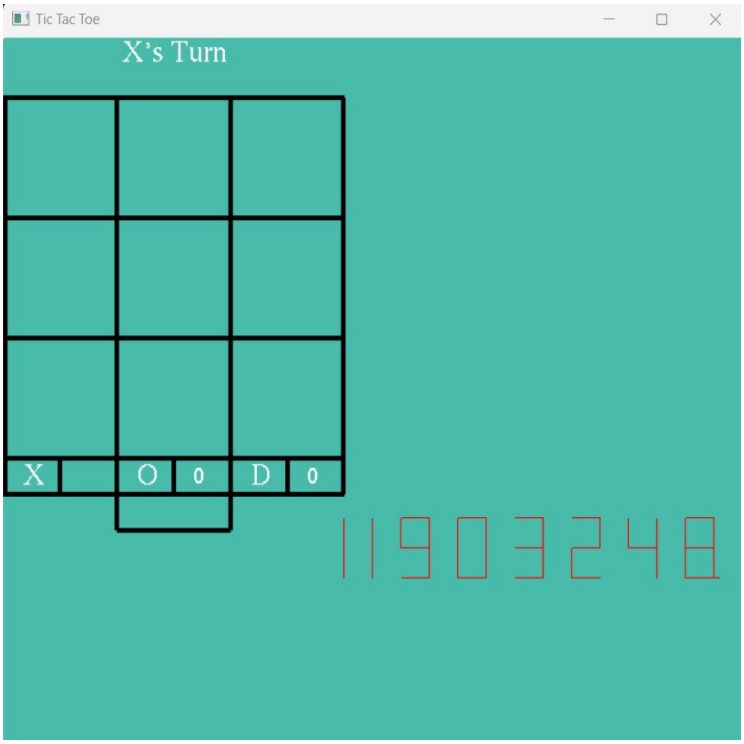
```
glVertex2i(475, 425);
glVertex2i(475, 425);
glVertex2i(450, 425);
glVertex2i(450, 425);
glVertex2i(475, 425);
//
glVertex2i(450, 425);
glVertex2i(475, 425);
glVertex2i(475, 425);
glVertex2i(475, 450);
glVertex2i(475, 450);
glVertex2i(450, 450);
glVertex2i(450, 450);
glVertex2i(475, 450);
glEnd();
glBegin(GL_LINES);
/* 2 */
glVertex2i(500, 400);
glVertex2i(525, 400);
glVertex2i(525, 400);
glVertex2i(525, 425);
glVertex2i(525, 425);
glVertex2i(500, 425);
glVertex2i(500, 425);
glVertex2i(500, 450);
glVertex2i(500, 450);
glVertex2i(525, 450);
glEnd();
glBegin(GL_LINES);
/* 4 */
glVertex2i(550, 400);
glVertex2i(550, 425);
glVertex2i(550, 425);
glVertex2i(575, 425);
glVertex2i(575, 400);
glVertex2i(575, 450);
glEnd();
glBegin(GL_LINES);
/* 8 */
glVertex2i(600, 400);
glVertex2i(625, 400);
glVertex2i(625, 400);
glVertex2i(625, 450);
glVertex2i(630, 450);
glVertex2i(600, 450);
glVertex2i(600, 450);
glVertex2i(600, 400);
glVertex2i(600, 425);
glVertex2i(625, 425);
glEnd();
/* End */
   glEnd();



   // Set line color to white
  glColor3f(1.0, 1.0, 1.0);
  // Draw 1
  glBegin(GL_LINES);
```

```
  glVertex2f(0.1, 0.1);
  glVertex2f(0.1, 0.9);
  glEnd();
  // Draw 1
  glBegin(GL_LINES);
  glVertex2f(0.2, 0.1);
  glVertex2f(0.2, 0.9);
  glEnd();
  // Draw 9
  glBegin(GL_LINES);
  glVertex2f(0.3, 0.1);
  glVertex2f(0.5, 0.9);
  glEnd();
  // Draw 0
  glBegin(GL_LINES);
  glVertex2f(0.6, 0.1);
  glVertex2f(0.6, 0.9);
  glEnd();
  // Draw 3
  glBegin(GL_LINES);
  glVertex2f(0.7, 0.1);
  glVertex2f(0.9, 0.9);
  glEnd();
  // Draw 2
  glBegin(GL_LINES);
  glVertex2f(1.0, 0.1);
  glVertex2f(0.8, 0.5);
  glVertex2f(0.8, 0.5);
  glVertex2f(1.0, 0.9);
  glVertex2f(1.0, 0.9);
  glVertex2f(0.8, 0.9);
  glVertex2f(0.8, 0.9);
  glVertex2f(1.0, 0.5);
  glVertex2f(1.0, 0.5);
  glVertex2f(0.8, 0.1);
  glVertex2f(0.8, 0.1);
  glVertex2f(1.0, 0.1);
  glEnd();
        glutSwapBuffers();
        glEnd();
        glColor3f(1.0, 1.0, 1.0);
   glRasterPos2i(35,5);
}
//      Function to reshape
void Reshape(int x, int y)
{
        glViewport(0, 0, x, y);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(0, x, y, 0, 0, 1);
        glMatrixMode(GL_MODELVIEW);
}
//      Function for menu
void game(int id)
{
 switch(id)
 {
  case 1:
         ch=0;
         glClearColor(0.287933,0.735398,0.669251,1.0);
```

```
            break;
 case 2:
            ch=1;
            glClearColor(0,0.501961,0.501961,1.0);
            break;
 case 3:
            ch=2;
            glClearColor(1,0.411765,0.705883,1.0);
            break;
 case 4:
            ch=3;
            glClearColor(0.980392,0.815686,0.537255,1.0);
            break;
 case 5:
            ch=4;
            glClearColor(0.239216,0.109804,0,1.0);
            break;
 case 6:
            exit(0);
 }

}



int main(int argc, char **argv)
{
            Intialize();
            glutInit(&argc,argv);
            glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE);
            glutInitWindowPosition(550,200);
            glutInitWindowSize(650,650);

            glutCreateWindow("Tic Tac Toe");
            glutCreateMenu(game);
            glutAddMenuEntry("Color Change Menu ",0);
            glutAddMenuEntry("Default Color",1);
            glutAddMenuEntry("O in White and X in Red",2);
            glutAddMenuEntry("Pink Color",3);
            glutAddMenuEntry("Yellow Color",4);
            glutAddMenuEntry("Brown Color",5);
            glutAddMenuEntry("Exit",6);
            glutAttachMenu(GLUT_RIGHT_BUTTON);

            glutReshapeFunc(Reshape);
            glutDisplayFunc(Display);
            glutKeyboardFunc(Keyboard);
            glutMouseFunc(Mouse);
            glutIdleFunc(Display);

            glClearColor(0.287933,0.735398,0.669251,1.0);

   glutMainLoop();

   return 0;
}
```

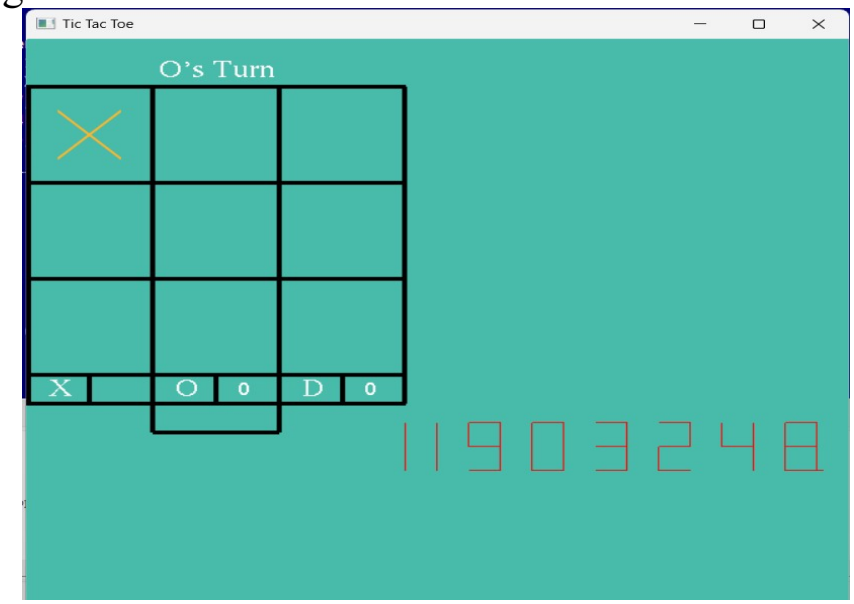
The following is the output of the code which is the Tic-Tac-Toe Using OpenGL:

Snapshot1:



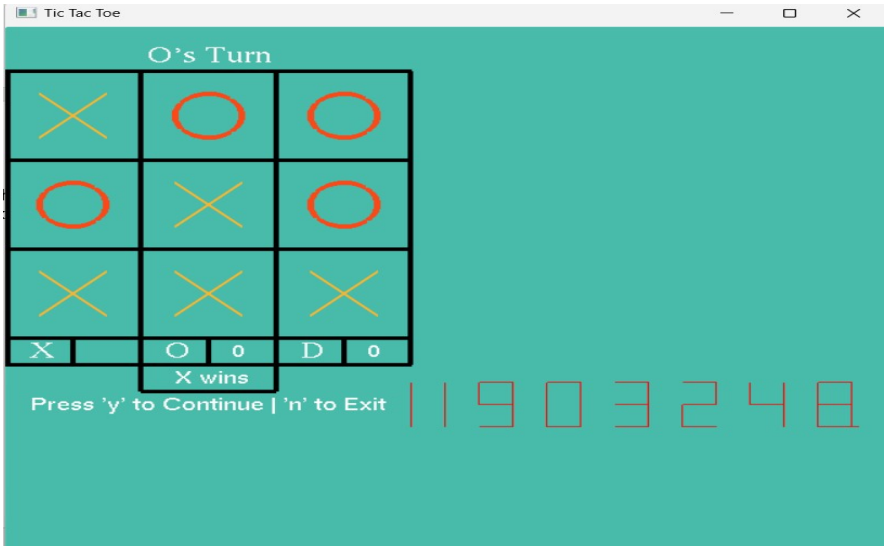
Snapshot2:

Here we can see that X' player game and mark show up when we touch the grid



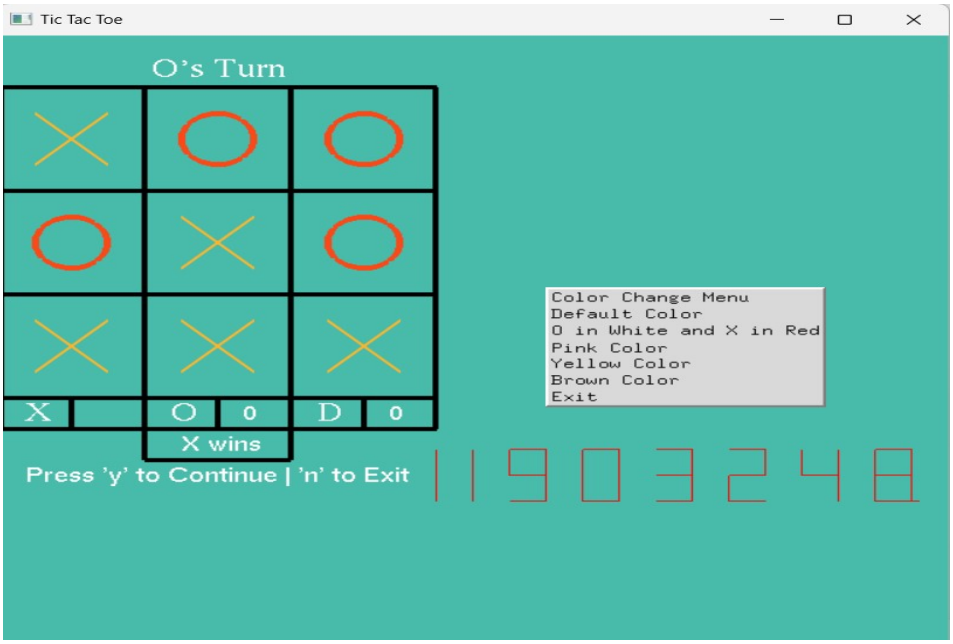
Snapshot 3:

And in snapshot 3 we can see that there is a game between the both players it displays who is winner



Snapshot 4:

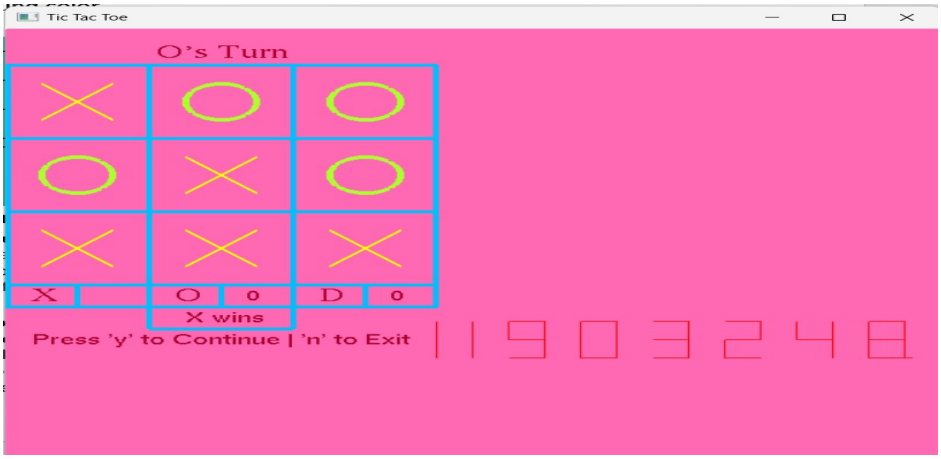
In the snapshot 4 there is an menu option that can be used to change the background color



Snapshot 5:

Menu options are:

1.Default Color
2.O in white and X in Red
3.Pink Color
4.Yellow Color
5.Brown Color
6.Exit



In snapshot 5 we can see that the background color is changed

Conclusion:

In conclusion, OpenGL is a popular graphics API (Application Programming Interface) that offers a standardised method of communicating with graphics hardware. Applications for it range from simulations and video games to CAD/CAM software and scientific visualisation.

Developers have a lot of freedom and control over the rendering process thanks to OpenGL's low-level interface, which enables them to directly manage the graphics hardware. It is a potent tool for producing realistic 3D images since it includes complex rendering features including shading, lighting, and texture mapping. Overall, OpenGL has made a substantial contribution to the advancement of computer graphics, and its impact can still be felt in a wide variety of modern applications.

Future Enhancement:

We can add many other features like some home structures and also can add river, trees and some other structures so as to make the view more attractive and more productive. We can advance this project by using various opengl functions that makes the view still more beautiful and accurate. These are some future enhancements of the project

References:

*Computer Graphics with OpenGL by Hearn, Baker and Carithers 4th edition, Published by Prentice Hall, 2011

*OpenGL 3.0 programming Guide by Dan, 2014