CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION TO COMPUTER GRAPHICS:

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. We can create images that are indistinguishable from photographs of real objects. In other terms, Computer Graphics are the graphics created by the computers, and more generally, the representation and manipulation of image data by a computer. The development of computer graphics has been driven both by the needs of the user community and by advances in hardware and software.

Typically, the term Computer Graphics refers to several different things;

1. The representation and manipulation of image data by a computer.
2. The various technologies used to create and manipulate images.
3. The images so produced, and manipulating visual content.

1.2 HISTORY OF COMPUTER GRAPHICS:

The phrase Computer Graphics was coined in 1960 by William Fetter, a graphic designer for Boeing. The field of Computer Graphics developed with the emergence of computer graphics hardware. Early projects like the Whirlwind and SAGE projects introduced the CRT as a viable display and interaction interface and introduced the light pen as an input device. Further advances in computing led to greater advancements in interactive computer graphics. In 1959, the TX-2 computer was developed at MIT Lincoln Laboratory. A light pen could be used to draw sketches on the computer using Ivan Sutherland’s revolutionary Sketchpad software.

Also in 1961 another student at MIT, Steve Russell, created the first video game, Space war. 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. In this computer generated film, Zajac showed how the attitude of a satellite could be altered as it orbits the Earth. Many of the most important early breakthroughs in computer graphics research occurred at the University of Utah in the 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.

Graphics and application processing were increasingly migrated to the intelligence in the workstation, rather than continuing to rely on central mainframe and mini-computers. 3D graphics became more popular in the 1990s in gaming, multimedia and animation. Computer graphics used in films and video games gradually began to be realistic to the point of entering the uncanny valley. Examples include the later Final Fantasy games and animated films like The Polar Express.

1.3 INTRODUCTION TO OpenGL:

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that are used to specify the objects and operations needed to produce interactive three-dimensional applications. OpenGL is designed as a streamlined hardware independent interface to be implemented on many different hardware platforms.

These are certain characteristics of OpenGL;

* OpenGL is a better documented API.
* OpenGL is much easier to learn and program.
* OpenGL has the best demonstrated 3D performance for any API.

The OpenGL specification describes an abstract API for drawing 2D and 3D graphics. Although it's possible for the API to be implemented entirely in software, it's designed to be implemented mostly or entirely in hardware.

In addition to being language-independent, OpenGL is also platform-independent. The specification says nothing on the subject of obtaining, and managing, an OpenGL context, leaving this as a detail of the underlying windowing system. For the same reason, OpenGL is purely concerned with rendering, providing no APIs related to input, audio, or windowing.

OpenGL is an evolving API. New versions of the OpenGL specification are regularly released by the Khronos Group, each of which extends the API to support various new features. In addition to the features required by the core API, GPU vendors may provide additional functionality in the form of extensions.

Extensions may introduce new functions and new constants, and may relax or remove restrictions on existing OpenGL functions. Vendors can use extensions to expose custom APIs without needing support from other vendors or the Khronos Group as a whole, which greatly increases the flexibility of OpenGL. All extensions are collected in, and defined by, the OpenGL Registry.

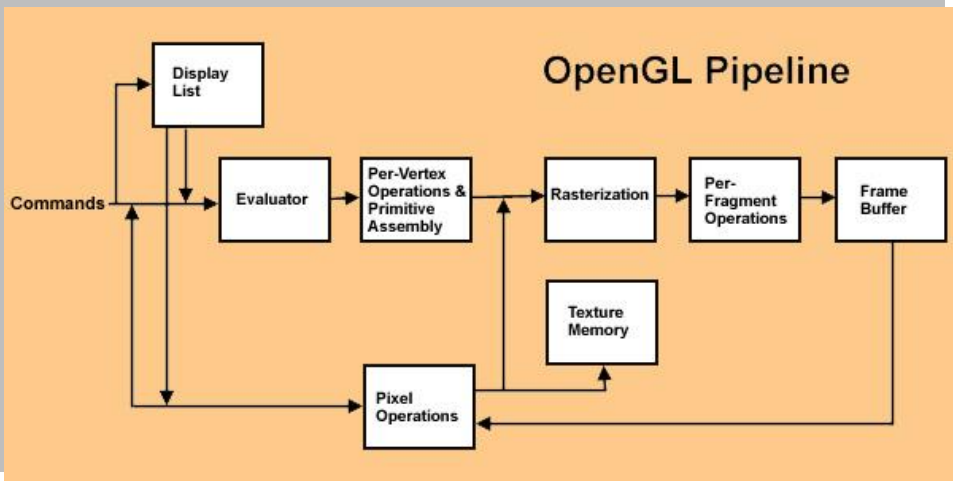


Fig. 1.3: OpenGL Pipeline.

1.4 INTRODUCTION TO GLUT:

GLUT is the OpenGL utility toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing API for OpenGL. GLUT makes it easier to learn about and explore OpenGL programming. GLUT provides a portable API so you can write a single OpenGL program that works across all PC and workstation OS platforms. GLUT is designed for constructing small to medium sized OpenGL programs.

While GLUT is well-suited to learning OpenGL and developing simple OpenGL applications, GLUT is not a full-featured toolkit so large applications requiring sophisticated user interfaces are better off using native window system toolkits. The GLUT library has both C, C++ (same as C), FORTRAN, and ADA programming bindings. The GLUT source code distribution is portable to nearly all OpenGL implementations and platforms.

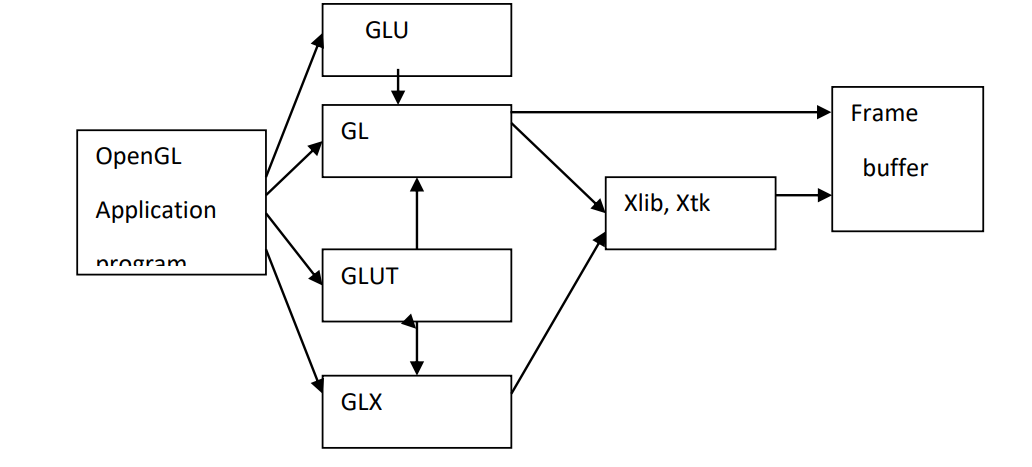


Fig. 1.4: Library Organization of OpenGL.

1.5 APPLICATIONS OF COMPUTER GRAPHICS:

The development of computer graphics has been driven both by the needs of the user community and by advances in hardware and software. The applications of computer graphics are many and varied. We can however divide them into four major areas;

* DISPLAY OF INFORMATION: More than 4000 years ago, the Babylonians developed floor. plans of buildings on stones. Today, the same type of information is generated by architects using computers. Over the past 150 years, workers in the field of statistics have explored techniques for generating plots. Now, we have computer plotting packages. Supercomputers now allow researchers in many areas to solve previously intractable problems. Thus, Computer Graphics has innumerable applications.
* DESIGN: Professions such as engineering and architecture are concerned with design. Today, the use of interactive graphical tools in CAD, in VLSI circuits, characters for animation have developed in a great way.
* SIMULATION AND ANIMATION: One of the most important uses has been in pilots‟ training. Graphical flight simulators have proved to increase safety and reduce expenses. Simulators can be used for designing robots, plan it’s path, etc. Video games and animated movies can now be made with low expenses.
* USER INTERFACES: Our interaction with computers has become dominated by a visual paradigm. The users’ access to internet is through graphical network browsers. Thus Computer Graphics plays a major role in all fields.

1.6 APPLICATIONS OF OpenGL:

* OpenGL (Open Graphics Library)is a cross-language, multi-platform API for rendering 2D and 3D computer graphics.
* The API is typically used to interact with a GPU, to achieve hardware accelerated rendering.
* It is widely used in CAD, virtual reality, scientific visualization, information⎫ visualization, flight simulation, and video games.

1.7 OpenGL PRIMITIVES:

OpenGL supports two classes of primitives;

1. Geometric Primitives.
2. Image(Raster) Primitives.

Geometric primitives are specified in the problem domain and include points, line segments, polygons, curves and surfaces.

Raster primitives, such as arrays of pixels pass through a separate parallel pipeline on their way to the frame buffer.

There are ten basic OpenGL primitives;

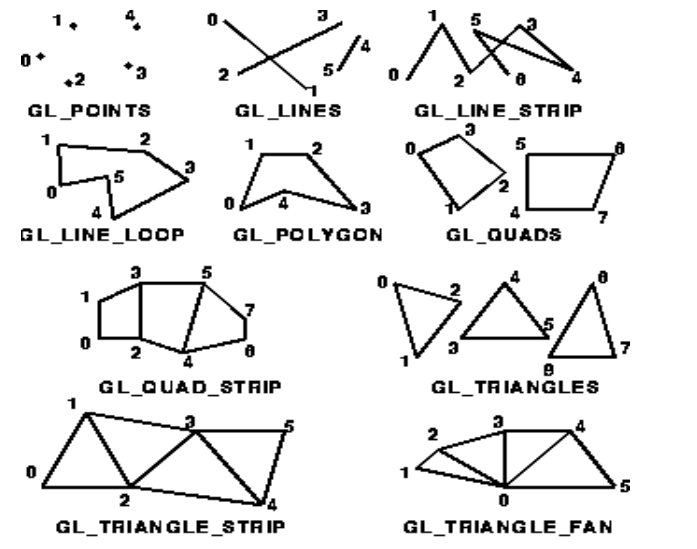


Fig. 1.7: OpenGL Primitives.

CHAPTER-2

REQUIREMENT SPECIFICATION

The graphics editor has been programmed in C. It makes use of Turbo C Graphics library package for creating the user interface. This is a subroutine library for terminal independent screen painting and input event handling.

2.1 HARDWARE REQUIREMENTS:

The standard output device is assumed to be a Color Monitor. It is quite essential for any graphics package to have this, as provision of color options to the user is a must. The mouse, the main input device, has to be functional i.e. used to give input in the game. A keyboard is used for controlling and inputting data in the form of characters, numbers i.e. to change the user views. Minimum Requirements expected are cursor movement, creating objects like lines, squares, rectangles, polygons, etc. Transformations on objects/selected area should be possible. Filling of area with the specified color should be possible.

1. Processor: Intel dual core i5.
2. RAM: 1GB RAM or above.
3. Input devices: Keyboard.
4. Output devices: Monitor.

2.2 SOFTWARE REQUIREMENTS:

The editor has been implemented on the OpenGL platform and mainly requires the appropriate version of Microsoft Visual Studio.

1. Operating System: Microsoft Windows 10.
2. Microsoft Visual Studio 2010.
3. glut.h header file.
4. glut.dll library file.

CHAPTER-3

ABOUT THE PROJECT

It is the same noughts and crosses or the Xs and Os, the other names for Tic-Tac-Toe, you’ve played with paper and pencil. This mini game project is written in C language in a very simple manner; it is complete and totally error-free.

You have probably played the Tic-Tac-Toe game to pass time during school hours. It’s fun when you play with paper and pencil. Here, we have developed a mini project in C Tic-Tac-Toe game-a simple console application with graphics.

This Tic-Tac-Toe game in C is compiled in Code::Blocks with gcc compiler. The source code is not that long; it is about 400 lines.

While making a Tic-Tac-Toe game using C language, it is important to make use of arrays. The Xs and Os are kept in different arrays, and they are passed between several functions in the code to keep track of how the game goes. With the code here you can play the game choosing either X or O against the computer.

This Tic-Tac-Toe C game is such that you will have to input a numerical character, from 1 to 9, to select a position for X or O into the space you want. For example: if you are playing with O and you input 2, the O will go to first row – second column. If you want to place O in third row – first column, you have to enter 7. And, it is similar for the other positions.

CHAPTER-4

DESIGN

Design of the project includes the initialization and the flow in which the project works. The initialization involves the use of mouse and keyboard functions. Flow of control includes the working pattern of the project.

INITIALIZATION:

1. The game is to be played between two people (in this program between HUMAN and COMPUTER).
2. One of the player chooses “O” and the other “X” to mark their respective cells.
3. The game starts with one of the players and the game ends when one of the players has one whole row/ column/ diagonal filled with his/her respective character (“O” or “X”).
4. If no one wins, then the game is said to be draw.

FLOW OF CONTROL:

A board game (such as Tic-tac-toe) is typically programmed as a state machine. Depending on the current-state and the player's move, the game goes into the next-state. In this example, I use a variable currentState to keep track of the current-state of the game, and define named-constants to denote the various states of the game (PLAYING, DRAW, CROSS\_WON, and NOUGHT\_WON). A method called updateGame() is defined, which will be called after every move to update this currentState, by checking the status of the game-board.

Two methods are defined for printing the game board, printBoard() and printCell(). The printBoard() shall call printCell() to print each of the 9 cells. This seems trivial here, but will be useful in the object-oriented design to separate the board and cells into separate classes.

FLOWCHART:

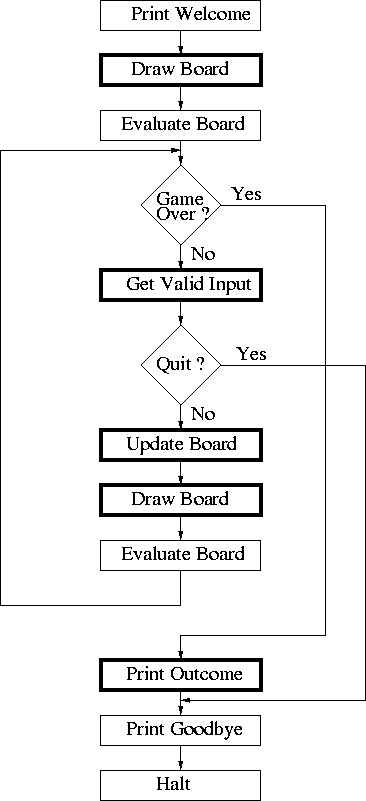


Fig. 4.1: Flowchart of Tic-Tac-Toe Working.

CHAPTER-5

IMPLEMENTATION

5.1 BUILT-IN FUNCTIONS:

1. glColor3f() - Used to display color.
2. glBegin() - Used to accept a single argument.
3. glVertex2f() - Used to specify a line a line, point.
4. glutPostRedisplay() - Marks the current window as needing to be redisplayed.
5. glClearColor(0.0,0.0,0.0,0.0) - Specifies clear values for the color buffers.
6. glEnable(GL\_DEPTH\_TEST) - Enables the OpenGL capabilities, Specifies the conditions under which the pixels will be drawn.
7. glLoadIdentity() - Replaces current matrix with identity matrix.
8. glViewport() - Sets the view port.
9. glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB) - Sets the initial display mode.
10. glutInitWindowSize(500,500) and glutInitWindowPosition(50,50) - Set the initial window size and position respectively.
11. glutCreateWindow() - Creates a top level window with the window name as specified.
12. glutDisplayFunc(display) - Sets the display call back for the current window.
13. glutReshapeFunc(reshape) - Sets the reshape call back for the current window.
14. glutMainLoop() - Enters the GLUT event processing loop. This routine should be called atmost once in a GLUT program. Once called, this routine will never return. It will calls necessary any callbacks that have been registered.

5.2 PSEUDOCODE:

// Tic Tac Toe or X's and O's.

// Keyboard input

// 'v' = view ortho/perspective

// 'l' = lighting on/off

// Include necessary libraries

#include <GL/glut.h>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <string.h>

// Lighting variables

GLfloat LightAmbient[]= { 0.5f, 0.5f, 0.5f, 1.0f };

GLfloat LightDiffuse[]= { 0.5f, 0.5f, 0.5f, 1.0f };

GLfloat LightPosition[]= { 5.0f, 25.0f, 5.0f, 1.0f };

GLfloat mat\_specular[] = { 1.0, 1.0, 1.0, 1.0 };

// Mouse variables

int mouse\_x, mouse\_y, Win\_x, Win\_y, object\_select;

// State variables

static int view\_state = 0, light\_state = 0;

int spin, spinboxes;

int player, computer, win, start\_game;

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}};

int box\_map[9];

int object\_map[9][2] = {{-6,6},{0,6},{6,6},{-6,0},{0,0},{6,0},{-6,-6},{0,-6},{6,-6}};

GLUquadricObj \*Cylinder;

// Initialize game

void init\_game(void) {

int i;

for (i = 0; i < 9; i++) {

box\_map[i] = 0;

}

win = 0;

start\_game = 1;

}

// Check for three in a row/column/diagonal

int check\_move(void) {

int i, t = 0;

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;

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;

}

// 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)) {

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 a corner

int check\_corner(void) {

if (box\_map[0] == 0) {

box\_map[0] = computer;

return 1;

}

if (box\_map[2] == 0) {

box\_map[2] = computer;

return 1;

}

if (box\_map[6] == 0) {

box\_map[6] = computer;

return 1;

}

if (box\_map[8] == 0) {

box\_map[8] = computer;

return 1;

}

return 0;

}

// Check for free space in a 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;

}

// Computer's turn

int computer\_move() {

if (blocking\_win() == 1)

return 1;

if (check\_corner() == 1)

return 1;

if (check\_row() == 1)

return 1;

return 0;

}

// Draw text on screen

void Sprint(int x, int y, char \*st) {

int l, i;

l = strlen(st);

glRasterPos2i(x, y);

for (i = 0; i < l; i++) {

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, st[i]);

}

}

// Spin the cube

static void TimeEvent(int te) {

spin++;

if (spin > 360)

spin = 180;

glutPostRedisplay();

glutTimerFunc(8, TimeEvent, 1);

}

// Initialize OpenGL

void init(void) {

glClearColor(0.6, 0.6, 0.4, 0.0);

glShadeModel(GL\_SMOOTH);

glEnable(GL\_DEPTH\_TEST);

glLightfv(GL\_LIGHT1, GL\_AMBIENT, LightAmbient);

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, LightDiffuse);

glLightfv(GL\_LIGHT1, GL\_POSITION, LightPosition);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT1);

start\_game = 0;

win = 0;

Cylinder = gluNewQuadric();

gluQuadricDrawStyle(Cylinder, GLU\_FILL);

gluQuadricNormals(Cylinder, GLU\_SMOOTH);

gluQuadricOrientation(Cylinder, GLU\_OUTSIDE);

}

// Draw 'O'

void Draw\_O(int x, int y, int z, int a) {

glPushMatrix();

glTranslatef(x, y, z);

glRotatef(a, 1, 0, 0);

glutSolidTorus(0.5, 2.0, 8, 16);

glPopMatrix();

}

// Draw 'X'

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, -3);

gluCylinder(Cylinder, 0.5, 0.5, 6, 16, 16);

glPopMatrix();

glPushMatrix();

glRotatef(a, 1, 0, 0);

glRotatef(90, 0, 1, 0);

glRotatef(315, 1, 0, 0);

glTranslatef(0, 0, -3);

gluCylinder(Cylinder, 0.5, 0.5, 6, 16, 16);

glPopMatrix();

glPopMatrix();

}

// Draw the game board

void display(void) {

if (abc == 3) {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glColor3f(0.0, 1.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-9.0, 9.0, -9.0, 9.0, 0.0, 30.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glDisable(GL\_COLOR\_MATERIAL);

glDisable(GL\_LIGHTING);

glColor3f(0.0, 0.0, 1.0);

Sprint(-2, 0, "Project by");

Sprint(-2, -1, "Gajanan and Nitin");

Sprint(-3, -2, "To Start press right button");

Sprint(-3, -3, "right button for X's");

Sprint(-3, -4, "and left for O's");

glutSwapBuffers();

} else if (abc == 0) {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-9.0, 9.0, -9.0, 9.0, 0.0, 30.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glDisable(GL\_COLOR\_MATERIAL);

glDisable(GL\_LIGHTING);

glColor3f(0.0, 0.0, 1.0);

Sprint(-4, 0, "Project by Gajanana G Bhat and Nitin Kulkarni");

Sprint(-3, -1, "Right Click to Start the Game");

glutSwapBuffers();

} else {

int ix, iy;

int i;

int j;

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-9.0, 9.0, -9.0, 9.0, 0.0, 30.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glDisable(GL\_COLOR\_MATERIAL);

glDisable(GL\_LIGHTING);

glColor3f(1.0, 0.0, 0.0);

if (win == 1)

Sprint(-2, 1, "Congratulations! You win");

if (win == -1)

Sprint(-2, 1, "Computer wins");

if (win == 2)

Sprint(-2, 1, "Tie");

if (view\_state == 1) {

glColor3f(0.0, 0.0, 1.0);

Sprint(-3, 8, "Perspective view");

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60, 1, 1, 30);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

} else {

glColor3f(1.0, 0.0, 0.0);

Sprint(-2, 8, "Ortho view");

}

if (light\_state == 1) {

glDisable(GL\_LIGHTING);

glDisable(GL\_COLOR\_MATERIAL);

} else {

glEnable(GL\_LIGHTING);

glEnable(GL\_COLOR\_MATERIAL);

}

gluLookAt(0, 0, 20, 0, 0, 0, 0, 1, 0);

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();

}

glColor3f(0.0, 0.0, 0.0);

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\_X(object\_map[i][0], object\_map[i][1], -1, j);

if (box\_map[i] == -1)

Draw\_O(object\_map[i][0], object\_map[i][1], -1, j);

}

glutSwapBuffers();

}

}

// Reshape the window

void reshape(int w, int h) {

Win\_x = w;

Win\_y = h;

glViewport(0, 0, (GLsizei)w, (GLsizei)h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

}

// Handle keyboard input

void keyboard(unsigned char key, int x, int y) {

switch (key) {

case 'v':

case 'V':

view\_state = abs(view\_state - 1);

break;

case 'b':

case 'B':

light\_state = abs(light\_state - 1);

break;

case 27:

exit(0);

break;

default:

break;

}

}

// Handle mouse input

void mouse(int button, int state, int x, int y) {

mouse\_x = (18 \* (float)((float)x / (float)Win\_x)) / 6;

mouse\_y = (18 \* (float)((float)y / (float)Win\_y)) / 6;

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;

}

// Handle menu selection

void menu(int choice) {

switch (choice) {

case 1:

abc = 1;

glutMouseFunc(mouse);

break;

case 2:

view\_state = abs(view\_state - 1);

break;

case 3:

abc = 3;

glutMouseFunc(mouse);

break;

case 4:

exit(0);

break;

}

}

// Main program

int main(int argc, char\*\* argv) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(850, 600);

glutInitWindowPosition(10, 10);

glutCreateWindow(argv[0]);

glutSetWindowTitle("X's and O's 3D");

init();

glutCreateMenu(menu);

glutAddMenuEntry("start game", 1);

glutAddMenuEntry("perspective view", 2);

glutAddMenuEntry("help", 3);

glutAddMenuEntry("Quit", 4);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutKeyboardFunc(keyboard);

glutMouseFunc(mouse);

glutTimerFunc(50, TimeEvent, 1);

glutMainLoop();

return 0;

}

CHAPTER-6

RESULT

SNAPSHOTS:

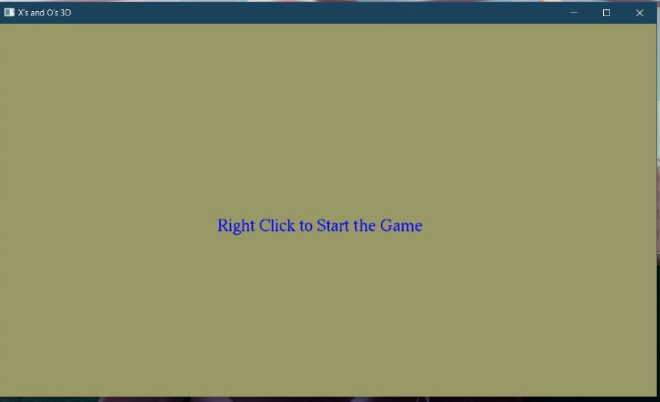


Fig. 6.1: Displaying the Instruction to Start the Game.

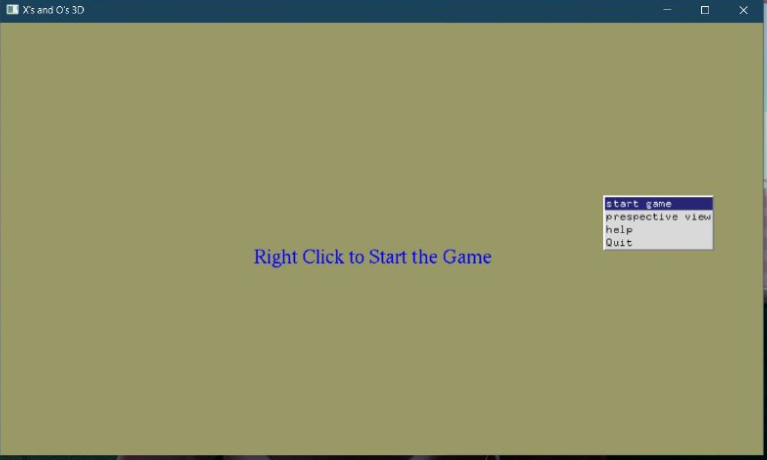


Fig. 6.2: Snapshot Showing to Start the Game.

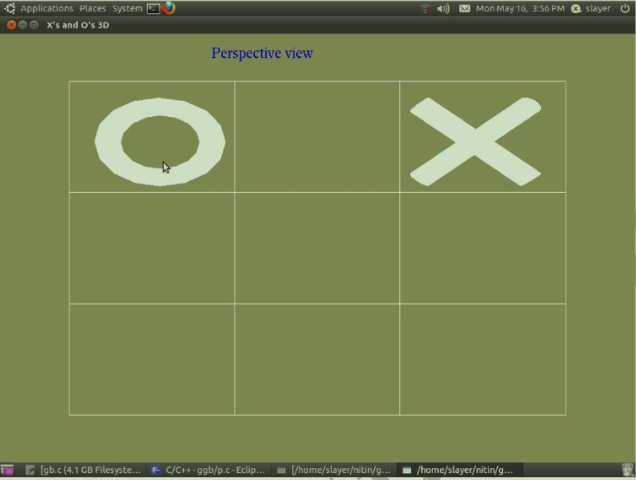


Fig. 6.3: Snapshot Showing O’s and X’s as the Input.

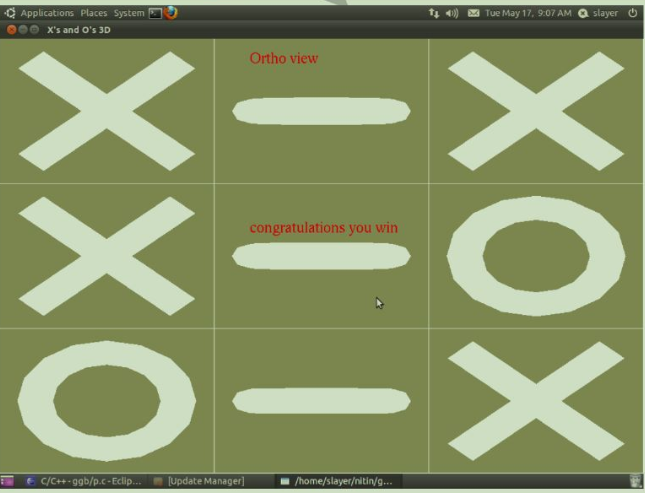


Fig. 6.4: Snapshot Showing if Player Wins.

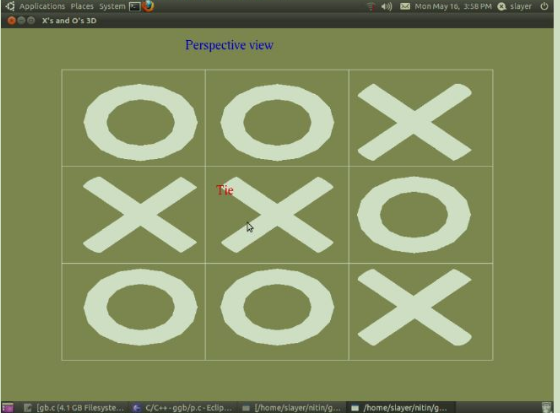


Fig. 6.5: Snapshot Showing a Tie.

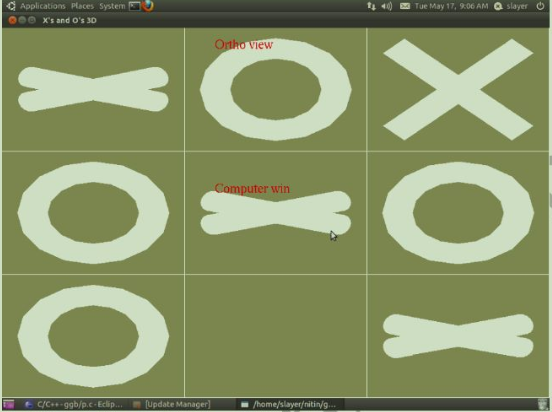


Fig. 6.6: Snapshot Showing if Computer Wins.

CONCLUSION

In conclusion, our computer graphics mini project on tic-tac-toe successfully applied computer graphics principles to create a visually appealing and interactive game. We developed a responsive game board using rendering techniques and implemented smooth animations to enhance the user experience. The intuitive user interface allowed for easy navigation and gameplay. Additionally, the incorporation of an AI opponent added a challenging element to the game. Through this project, we gained valuable knowledge and hands-on experience in implementing computer graphics concepts in the context of a popular game.

FUTURE ENHANCEMENT

This project has been designed such that it works on the windows platform. The project can be designed using different languages and better graphical interfaces. The following features can be incorporated;

* This project may be useful to follow the architecture of fountain and its further development.
* We can show that the water color of the fountain can be changed by the interaction of the mouse.
* We can add some more graphics which show the simulation of other objects including the sound effects which can give the project an attractive look.

REFERENCES

[1] Edward Angel, “Interactive Computer Graphics”, 5th edition, Pearson Education,2005.

[2] Jackie L. Neider, Mark Warhol, Tom R. Davies,” OpenGL Red Book”, 2nd Revised Edition, 2005.

[3] Donald D Hearn and M. Pauline Baker, “Computer Graphics with OpenGL”, 3rd Edition.

[4] F.S. Hill and Stephen M. Kelly, “Computer Graphics using OpenGL”, 3rd Edition.

WEBSITES:

* <http://www.opengl.org>
* <http://www.wikipedia.com>
* <http://basic4gl.wikispaces.com>
* <http://openglprojects.in>