Jaypee University of Information Technology

Subject: Multimedia Lab (18B1WCI575)

Project Title: 2D Helicopter Game using OpenGL

Project Report

Submitted by:

Om Vishal , 211322

Submitted to:

Ms. Seema Rani

Contents:

CHAPTER 1

Introduction

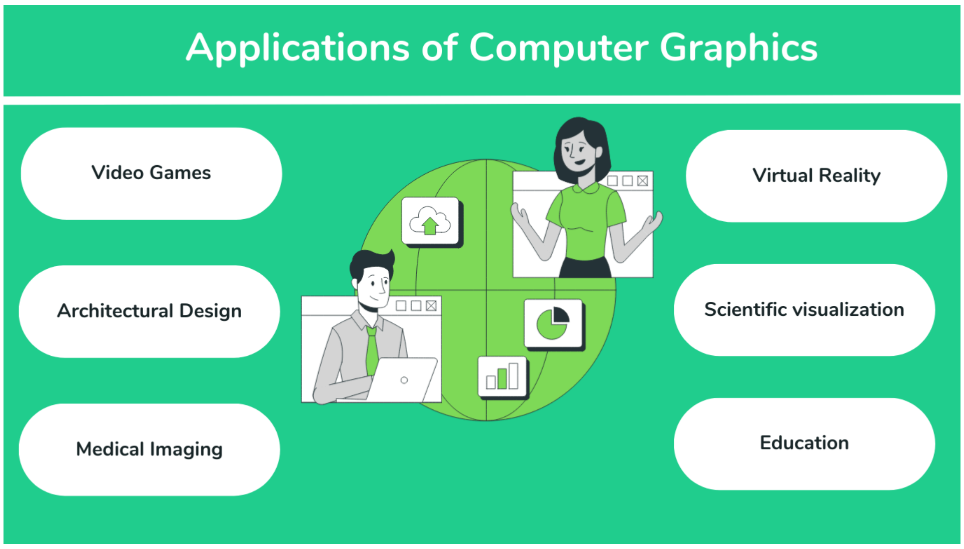
Computer graphics is concerned with all aspects of producing images using a computer. The field began humbly 50 years ago, with the display of few lines on a cathode ray tube (CRT); now, we can create images by computer that are indistinguishable from photographs of real objects.

In this chapter we discuss the application of computer graphics, overview of the graphic system, graphics architectures. In this project we discuss OpenGL- open graphics library API- which is used to develop the application program which is the matter in question.

APPLICATIONS OF COMPUTER GRAPHICS

The applications of computer graphics are many and varied. However we can divide them into four major areas:

- Display of Information
- Design
- Simulation and Animation
- User interface



Display of Information

Classical graphics techniques arose as a medium to convey information among people. Although spoken and written languages serve this purpose, images are easier to communicate with and hence the computer graphics plays an important role.

Information visualization is becoming increasingly important in the field of security, medicine, weather forecasting and mapping. Medical imaging like CT, MRI, ultrasound scan are produced by medical imaging systems but are displayed by computer graphics system.

Design

Professions such as engineering and architecture are concerned with design which is an iterative process. The power of the paradigm of humans interacting with the computer displays was recognized by Ivan Sutherland.

Simulation and Animation

Graphics system, now are capable of generating sophisticated images in real time. Hence engineers and researchers use them as simulators. The most important use has been in the training of pilots.

User interfaces

User interaction with computers has become dominated by a visual paradigm that includes windows, icons, menus and pointing devices. Applications such as office suites, web browsers are accustomed to this style.

1.1 PROBLEM STATEMENT

To design and implement a helicopter game using OpenGL.

1.2 OBJECTIVES

In this project we are going to design a Helicopter game using OpenGL. This project displays a helicopter in flight with its fan and rotor stabilizer moving. The basic idea of the game is to dodge the rectangles and the game ends when the helicopter touches one of the rectangles.

The score is calculated based on the distance covered by the helicopter before the crash.

1.3 SCOPE

It can be used for gaming and entertainment purpose. It can also be extended to the development of android application.

CHAPTER 2

People use the term “computer graphics” to mean different things in different context. Computer graphics are pictures that are generated by a computer. Everywhere you look today, you can find examples, especially in magazines and on television. Some images look so natural you can’t distinguish them from photographs of a real scene. Others have an artificial look, intended to achieve some visual effects.

A GRAPHICS SYSTEM

A computer graphics system is a computer system which has five major elements:

1. Input devices
2. Processor
3. Memory
4. Frame buffer
5. Output devices

This model is general enough to include workstations and personal computers, interactive game systems, and sophisticated image-generating systems. Although all the components, with the exception of the frame buffer, are present in standard computer, it is the way each element is specialized for computer graphics that characteristics this diagram as a portrait of graphics system.

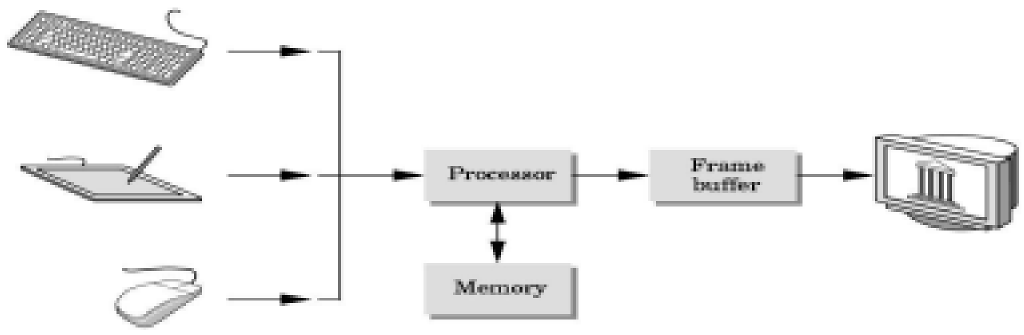


Fig 1.1: A Graphics System

2.1 HISTORY

OpenGL was developed by **'Silicon Graphics Inc'**(SGI) on 1992 and is popular in the gaming industry where it competes with the Direct3D in the Microsoft Windows platform. OpenGL is broadly used in CAD (Computer Aided Design), virtual reality, scientific visualization, information visualization, flight simulation and video games development.

2.2 CHARACTERISTICS

• OpenGL is a better documented API.

• OpenGL is also a cleaner API and much easier to learn and program.

• OpenGL has the best demonstrated 3D performance for any API.

2.3 COMPUTER GRAPHICS LIBRARY ORGANISATION

OpenGL stands for Open Source Graphics Library. Graphics Library is a collection of APIs (Application Programming Interfaces).

Graphics Library functions are divided in three libraries. They are as follows

i. GL Library (OpenGL in Windows)
ii. GLU (OpenGL Utility Library)
iii. GLUT (OpenGL Utility Toolkit)

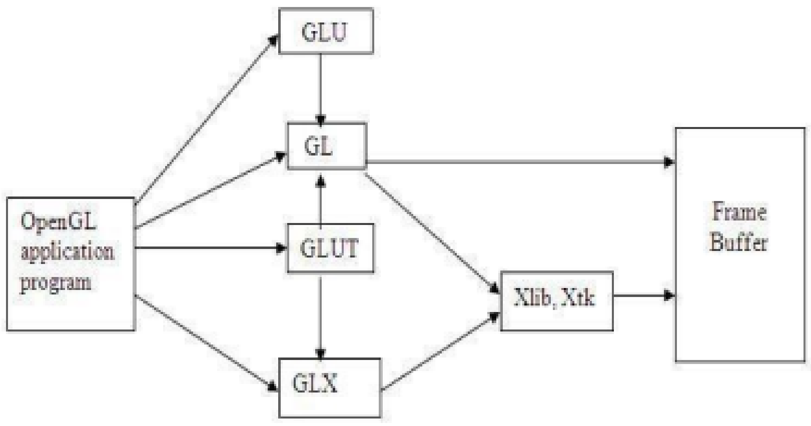


Fig 2.2 Library Organization

CHAPTER 3

System Requirements

Requirements analysis is critical for project development. Requirements must be documented, actionable, measurable, testable and defined to a level of detail sufficient for system design. Requirements can be architectural, structural, behavioural, functional, and non-functional. A software requirements specification (SRS) is a comprehensive description of the intended purpose and the environment for software under development.

3.1 Hardware Requirement

• Minimum of 2GB of main memory
• Minimum of 3GB of storage
• Keyboard
• Mouse
• Display Unit
• Dual-Core or AMD with minimum of 1.5GHz speed

3.2 Software Requirement

• Windows – 11
• Codeblocks Version 20.03
• OpenGL Files
• DirectX 8.0 and above versions

Header Files

• glut.h

Object File Libraries

• glut32.lib

DLL files

• glut32.dll

CHAPTER 4

Code Snippet:

```
#include<stdlib.h>
#include<GL/glut.h>
#include<time.h>
//#include<dos.h>
#include<stdio.h>
//#include<iostream.h>
//#include<windows.h>
int win1, win2;
void Write(float x, float y, float z, float scale, char *s)
{
	int i, l = strlen(s);
	glPushMatrix();
	glTranslatef(x, y, z);
	glScalef(scale, scale, scale);
	for (i = 0; i < l; i++)
		glutStrokeCharacter(GLUT_STROKE_ROMAN, s[i]);
	glPopMatrix();
}

void frontsheet(void)
{
	glClearColor(0, 0, 0, 1);
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.0, 1.0, 0.0);
	Write(-0.50, 0.9, 1, 0.0007, (char*)"Jaypee University Of Information Technology");
	Write(-0.55, 0.8, 1, 0.0006, (char*)"     Department of CSE");
	glColor3f(1.0, 0.0, 0.0);
```

```
        Write(-0.45, 0.6, 0.0, 0.0007, (char*) " 2D Helicopter Game");
        glColor3f(1.0, 1.0, 0.5);
        Write(-0.4, -0.8, 0.0, 0.0006, (char*) "Press 'C' to continue");
        glColor3f(1, 1, 0.0);
        Write(-1.0, 0.1, 0.0, 0.0007, (char*)" Submitted BY:");
        glColor3f(1.0, 1.0, 1.0);
        Write(-1.0, -0.13, 0.0, 0.0006, (char*) "2. Om Vishal: 211322");
        glColor3f(1, 1, 0.0);
        Write(-1.0, -0.4, 0.0, 0.0007, (char*) " Under the guidance of: ");
        glColor3f(1.0, 1.0, 1.0);
        Write(0.15, -0.515, 0.0, 0.0006, (char*) " 2.Ms. Seema Rani");
        glFlush();
}

float bspd = 0.02; // block dx value

//char name[25];

float b1x = 50.0, b1y = 0;//block 1 init position

float hm = 0.0;//copter moving dy value

int i = 0, sci = 1; float scf = 1; // for increment score score_int score_flag

char scs[20], slevel[20];
//to store score_string using itoa() and level as well
int level = 1, lflag = 1, wflag = 1; //level_flag & welcome_flag init w/1
void init(void)
{
        srand(time(0));//it generates a random no each time code is executed
        b1y = (rand() % 45) + 10;//b/w 10 to 44
        glClearColor(0.0, 0.0, 0.0, 0.0);
        glShadeModel(GL_SMOOTH);
        glLoadIdentity();
        glOrtho(0.0, 100.0, 0.0, 100.0, -1.0, .0);
}

void drawcopter()
{
        glColor3f(0.5, 1.0, 0.3);
        glRectf(10, 49.8, 19.8, 44.8);//body
        glRectf(2, 46, 10, 48);//tail
        glRectf(2, 46, 4, 51);//tail up
        glRectf(14, 49.8, 15.8, 52.2);//propeller stand
        glRectf(7, 53.6, 22.8, 52.2);//propeller*/
}

void renderBitmapString(float x, float y, float z, void *font, const char*string)
{
        const char *c;
        glRasterPos3f(x, y, z);
        for (c = string; *c != '\0'; c++)
        {
                glutBitmapCharacter(font, *c);
        }
}

void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT);
```

```
    //GameOver Checking
            if((i == 730 || i == -700)
            //top and bottom checking


            ||
            (((int)b1x == 10 || (int)b1x == 7 || (int)b1x == 4 || (int)b1x == 1) && (int)b1y < 53 +
(int)hm && (int)b1y + 35>53 + (int)hm)
            // propeller front checking


            ||
            (((int)b1x == 9 || (int)b1x == 3 || (int)b1x == 6) && (int)b1y < 45 + (int)hm && (int)b1y
+ 35>45 + (int)hm)
            //lower body checking


            ||
            (((int)b1x == 0) && (int)b1y < 46 + (int)hm && (int)b1y + 35>46 + (int)hm))
            // lower tail checking
    {
            glColor3f(0.0, 0.0, 1.0);
            glRectf(0.0, 0.0, 100.0, 100.0);
            glColor3f(1.0, 0.0, 0.0);
            renderBitmapString(40, 70, 0, GLUT_BITMAP_HELVETICA_18, "GAME OVER!!!");
            glColor3f(1.0, 1.0, 1.0);
            renderBitmapString(30, 58, 0, GLUT_BITMAP_TIMES_ROMAN_24, "THANKS
FOR PLAYING THE GAME!!");
            //renderBitmapString(45, 58, 0, GLUT_BITMAP_TIMES_ROMAN_24, "scored:");
            renderBitmapString(70, 58, 0, GLUT_BITMAP_TIMES_ROMAN_24, scs);
            glutSwapBuffers();
            glFlush();
            printf("\nGAME OVER\n\n");
            system("pause");
```

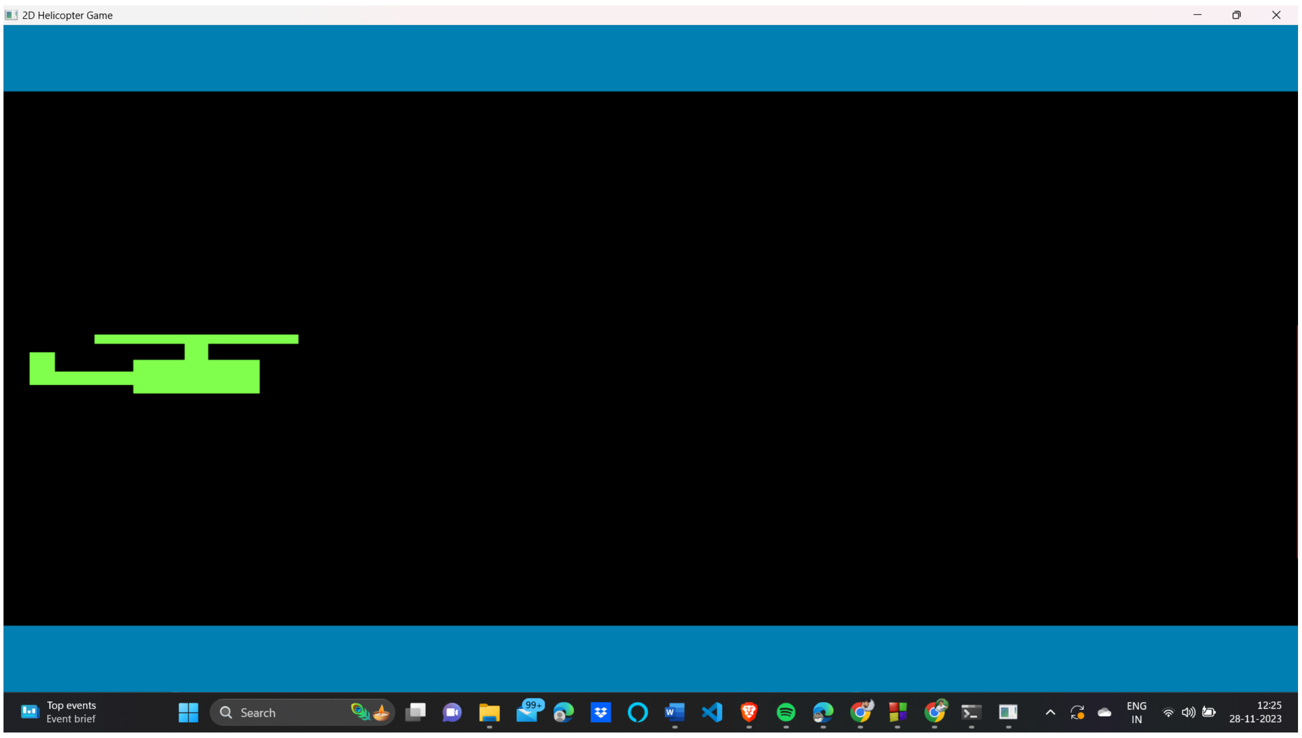
```
        //printf("%s\You scored  %s", name, scs);
        printf("\n\nClose the console window to exit...\n");
        exit(0);
        //getch();
}

else
{
        //on every increase by 50 in score in each level
        if (sci % 50 == 0 && lflag == 1)
        {
                lflag = 0; //make level_flag=0
                level++;//increase level by 1
                bspd += 0.01;//increase block_dx_speed by 0.01
        }
        //within every level make level_flag=1
        else if (sci % 50 != 0 && lflag != 1)
        {
                lflag = 1;
        }
        glPushMatrix();
        glColor3f(0.0, 0.5, 0.7);
        glRectf(0.0, 0.0, 100.0, 10.0);      //ceil
        glRectf(0.0, 100.0, 100.0, 90.0);    //floor

        glColor3f(0.0, 0.0, 0.0);   //score
//      renderBitmapString(1, 3, 0, GLUT_BITMAP_TIMES_ROMAN_24, "Distance:");
        //glColor3f(0.7,0.7,0.7);

        printf(slevel, "%d", level);   //level
```

```
//      renderBitmapString(80, 3, 0, GLUT_BITMAP_TIMES_ROMAN_24, "Level:");
        renderBitmapString(93, 3, 0, GLUT_BITMAP_TIMES_ROMAN_24, slevel);

        scf += 0.025;           //so less as program run very fast
        sci = (int)scf;
        printf(scs, "%d", sci);
        //from int to char convertion to display score
        renderBitmapString(20, 3, 0, GLUT_BITMAP_TIMES_ROMAN_24, scs);
        glTranslatef(0.0, hm, 0.0);
        // hm(=dy) changes occur by mouse func
        drawcopter();
        //code for helicopter
        //if wall move towards left & get out of projection volume
        if (b1x < -10)
        {
                b1x = 50;          //total width is 50
                b1y = (rand() % 25) + 20;
                //10 for selling+10 for giving enough space
                // block bottom limit 0+20 & top limit 24+20=44
        }

        else
                b1x -= bspd;
        //within the projection volume dec its x value by block_speed

        glTranslatef(b1x, -hm, 0.0);

        glColor3f(1.0, 0.0, 0.0);
        glRectf(b1x, b1y, b1x + 5, b1y + 35);//block 1
```

```
                glPopMatrix();

                glutSwapBuffers();
                glFlush();
        }
}

void moveHeliU(void)
{

        hm += 0.05;
        i++;
        glutPostRedisplay();

}

void moveHeliD()
{

        hm -= 0.05;
        i--;
        glutPostRedisplay();

}
void mouse(int button, int state, int x, int y)
{
        switch (button)
        {
        case GLUT_LEFT_BUTTON:
            if (state == GLUT_DOWN)
                glutIdleFunc(moveHeliU);

            else if (state == GLUT_UP)
                glutIdleFunc(moveHeliD);
            break;
        default: break;
        }
}
void keys(unsigned char key, int x, int y)
{
        if (key == 'w') glutIdleFunc(moveHeliU);
        if (key == 'm') glutIdleFunc(moveHeliD);
}
void keyboards(unsigned char key, int x4, int y4)
{
        if (key == 'c' || key == 'C')
        {
            glutDestroyWindow(win1);
            glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
            win2 = glutCreateWindow("2D Helicopter Game");
            glClearColor(0.0, 0.0, 0.0, 0.0);
            glFlush();
            glutDisplayFunc(display);
            gluOrtho2D(-1000, 1000, 0, 1000);
            init();
            glutMouseFunc(mouse);
            glutKeyboardFunc(keys);
        }
}
```

```
int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        glutInitWindowSize(800, 600);
        glutInitWindowPosition(200, 200);
      win1=glutCreateWindow("Mini Project");
        glFlush();
        glutDisplayFunc(frontsheet);
        glutKeyboardFunc(keyboards);
        glutMainLoop();
    return 0;
}
```

OUTPUT :



CHAPTER 5

References:

[1]. Interactive Computer Graphics: A Top- down Approach Using OpenGL, Fifth Edition by Edward Angel, Pearson education, 2009.

[2]. Computer Graphics with OpenGL, Third Edition, by Hearn & Baker, Pearson education.

[3]. http://www.cs.uccs.edu/~ssemwal/glman.html

[4]. http://www.opengl.czweb.org/ewtoc.html

[5]. http://www.opengl.org

[6]. http://www.en.wikipedia.org/wiki/OpenGL

[7]. https://www.opengl.org/documentation/specs/glut/spec3/node49.html

CHAPTER 6

Conclusion:

6.1 Conclusion of the Project

We have successfully implemented a simple game in this project using OpenGL. OpenGl supports enormous flexibility in the design and the use of OpenGl graphics programs. The presence of many built in classes methods take care of much functionality and reduce the job of coding as well as makes the implementation simpler.

This game shows the use of computer graphics throughout an application especially when it comes to interaction of computers with humans. In this program, we saw how alphabetical characters and stored data like scores are rendered on screen. We also saw how the game or animated characters- like the helicopter in this game- are created, and how objects can be moved from one co-ordinate position to another representing the movement of the object.

6.2 Future Enhancements

Every game has one property in common, that is they can be never ending. The games can blossom like trees spanning new levels and new avatars. Similarly the enhancements that can be made for this game are:

- We can create a new background that makes the game look more colourful.
- We can change the obstacles from rectangular blocks to more realistic objects like buildings or other flying objects.
- We can include more levels and increase the difficulty level as the player advances.
- We can include the feature of Multi-player to this game.