UNIVERSITY VISVESWARAYA COLLEGE OF ENGENEERING   
 , KR CIRCLE BANGALORE

COMPUTER SCIENCE AND ENGG.  
 COMPUTER GRAPHICS MINI PROJECT ON  
 “FIGHTER JET ”

- KAVYA S R(18GAEC9028)

**CONTENTS**

* **ABSTRACT**
* **OVERVIEW OF THE PROJECT**
* **INTRODUCTION OF CG**
* **INTRODUCTION OF OPENGL**
* **REQUIREMENTS SPECIFICATIONS**
* **DESIGN AND IMPLEMENTATION**
* **RESULT AND SNAPCHATS**
* **CONCLUSION**

CHAPTER 1

**ABSTRACT**

In this mini-project (FIGHTER JET), the operations performed by the jet are simulated graphically using OpenGL. Specific functions of OpenGL are used to simulate various operations of the jet.

The perspective concept will be implemented to provide a real-world effect. The fighter jet can be moved towards the required direction. It can be rotated in any direction as well. Various functions performed by jet are bullet firing, enabling a protective shield, can increase the throttle as required. The jet can destroy the objects present on the screen through bullet firing.

CHAPTER 2

**OVERVIEW OF PROJECT**

By using the keyboard interface, we perform the required action. The various key functions are:

      “up” arrow key to increase throttle

     “right” arrow key for the clockwise motion of the fighter jet.

     “left” arrow key for the anti-clockwise motion of the fighter jet.

     “Q” or “space bar” key used to fire the bullets

     “S” key used to enable protective shield

     “P” key is to pause the action of the fighter jet.

CHAPTER 3

**INTRODUCTION OF CG**

Computer Graphics involves technology to access. The Process transforms and presents information in a visual form. The role of computer graphics insensible. In today life, computer graphics has now become a common element in user interfaces, T.V. commercial motion pictures.

Computer Graphics is the creation of pictures with the help of a computer. The end product of the computer graphics is a picture it may be a business graph, drawing, and engineering.

In computer graphics, two or three-dimensional pictures can be created that are used for research. Many hardware devices algorithm has been developing for improving the speed of picture generation with the passes of time. It includes the creation storage of models and image of objects. These models for various fields like engineering, mathematical and so on.

Today computer graphics is entirely different from the earlier one. It is not possible. It is an interactive user can control the structure of an object of various input devices.

CHAPTER 4

**INTRODUCTION OF OPENGL**

**OpenGL** (Open **Graphics** Library) is a cross-platform, hardware-accelerated, language-independent, industrial standard API for producing 3D (including 2D) **graphics**. Modern **computers** have dedicated GPU (**Graphics** Processing Unit) with its own memory to speed up **graphics** rendering.

OpenGL is a library for doing computer graphics. By using it, you can create interactive applications which render high-quality color images composed of 2D&3D geometric objects and images. OpenGL is window and operating system independent.

As such, the part of your application which does rendering is platform independent. However, in order for OpenGL to be able to render, it needs a window to draw into. Generally, this is controlled by the windowing system on whatever platform you’re working on.

CHAPTER 6

**REQUIREMENT SPECIFICATION**

The requirement specifications of this project are not perfectly optimized. However, the following hardware and software specifications were done to best of my effort. Here are specifications:

Hardware Requirements:

The hardware Requirements given here is minimal requirements for the project to run even though the project can smoothly on almost all i386 Machines.

* Processor Speed - 512 MHz and above
* Ram Size - 512 MB or above
* Storage Space - 64 GB or above

Software Requirements:

* Operating System - Windows Family, preferably windows 7 , ubuntu and above
* Compiler - CODE::BLOCKS/VS CODE
* Graphics Library - glut.h

Programming language - C using OpenGL

CHAPTER 7

**DESIGN AND IMPLEMENTATION**

This mini project on FIGHTER JET Using OpenGL is a reliable graphics project that provides a menu-based user interface to run the program

Design and Working

Firstly, the working area will be created using windows function. whenever we run a program it shows the details of department, student, guide, and project name after that we press enter button in the system. The graphical view of a home in environment will be display. In the environment mainly contain a jet, objects, firing bullets, protective shield

By using the keyboard interface, we perform the required action. The various key functions are:

      “up” arrow key to increase throttle

     “right” arrow key for the clockwise motion of the fighter jet.

     “left” arrow key for the anti-clockwise motion of the fighter jet.

     “Q” or “space bar” key used to fire the bullets

     “S” key used to enable protective shield

     “P” key is to pause the action of the fighter jet.

**IMPLEMENTATION**

User Defined Functions:-

There are eight user defined functions in the source code of a project in OpenGL.

Void drawsquare();

This function used to draw the squares

Void drawjet();

This function used to draw the jet.

Void drawshield();

This function is used to draw the protective shield.

Void drawbullets();

This function is used to draw bullets.

Void shotbullets();

This function is used to shot bullets towards object.

Void display(void);

This function is used to display the things and objects in sc

**SOURCE CODE**

#include<stdbool.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <GL/glut.h>

#include<gl/gl.h>

#ifndef M\_PI

#define M\_PI 3.14159265358979323846

#endif

void mydisplay(void);

void initWindow(void);

float angle = 0.0;

float b1x1=40, b1x2=35, b1x3=35, b1x4=40;

float b1y1=23,b1y2=23,b1y3=18,b1y4=18;

int left, right;

int leftTime, rightTime;

int thrust, thrustTime;

float x = 20, y = 20, xv, yv, v;

int shield = 0, joyShield = 0, cursor = 1;

int lastTime;

int paused = 0;

int resuming = 1;

int originalWindow = 0, currentWindow;

int flag1=0, flag2=0, flag=0,flag3=0;

void display();

void drawstring(float x,float y,float z,char \*string)

{

char \*c;

glRasterPos3f(x,y,z);

for(c=string;\*c!='\0';c++)

{

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,\*c);

}

}

void screen(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,0.0,0.0);

drawstring(15,37,0.0,"FIGHTER JET");

glColor3f(1.0,0.0,1.0);

drawstring(10,24,0.0,"Kavya S R");

glColor3f( 1.0,0.0,0.0 );

drawstring(10,2,0.0,"Press w to Start");

glFlush();

}

void mydisplay(void)

{ if(flag2==0)

screen ();

if(flag==1 && flag1==0 &&flag2==0)

display();

}

void drawsquare()

{

glColor3f(0.0,1.0,0.0);

glBegin(GL\_POLYGON);

glVertex2f(b1x1,b1y1);

glVertex2f(b1x2,b1y2);

glVertex2f(b1x3,b1y3);

glVertex2f(b1x4,b1y4);

glEnd();

}

typedef struct

{

int inuse;

float x;

float y

float v;

float xv;

float yv;

int expire;

} Bullet;

#define BULLETS 100

Bullet bullet[BULLETS];

int allocBullet(void)

{

int i;

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

if (!bullet[i].inuse) {

return i;

}

}

return -1;

}

void initBullet(int i, int time)

{

float c = cos(angle\*M\_PI/180.0);

float s = sin(angle\*M\_PI/180.0);

bullet[i].inuse = 1;

bullet[i].x = x + 3.5 \* c;

bullet[i].y = y + 3.5 \* s;

bullet[i].v = 0.025;

bullet[i].xv = xv + c \* bullet[i].v;

bullet[i].yv = yv + s \* bullet[i].v;

bullet[i].expire = time + 600;

}

void advanceBullets(int delta, int time)

{

int i;

int a = 0;

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

if (bullet[i].inuse) {

float x, y;

if (time > bullet[i].expire) {

bullet[i].inuse = 0;

continue;

}

x = bullet[i].x + bullet[i].xv \* delta;

y = bullet[i].y + bullet[i].yv \* delta;

x = x / 40.0;

bullet[i].x = (x - floor(x))\*40.0;

y = y / 40.0;

bullet[i].y = (y - floor(y))\*40.0;

if((bullet[i].x<=b1x1 && bullet[i].y<=b1y1 && bullet[i].y>=b1y4 && bullet[i].x>=b1x3)){

a=1;

break;

}

}

}

if(a==1){

bullet[i].inuse = 0;

int zx1=(rand()%39);

int zy1=(rand()%34);

b1x1 = zx1;

b1x4 = zx1;

b1x2 = zx1-5;

b1x3 = zx1-5;

b1y1=zy1;

b1y2=zy1;

b1y3=zy1-5;

b1y4=zy1-5;

}

}

void shotBullet(void)

{

int entry;

entry = allocBullet();

if (entry >= 0)

{

initBullet(entry, glutGet(GLUT\_ELAPSED\_TIME));

}

}

void drawBullets(void)

{

int i;

glBegin(GL\_POINTS);

glColor3f(1.0, 0.0, 1.0);

for (i=0; i<BULLETS; i++)

{

if (bullet[i].inuse)

{

glVertex2f(bullet[i].x, bullet[i].y);

}

}

glEnd();

}

void drawshield(void)

{

float rad;

glColor3f(0.1, 0.1, 1.0);

glBegin(GL\_LINE\_LOOP);

for (rad=0.0; rad<11.0; rad += 1.0) {

glVertex2f(3.75 \* cos(2\*rad/M\_PI)+0.4, 3.0 \* sin(2\*rad/M\_PI));}

glEnd();

}

void drawJet(float angle)

{

glPushMatrix();

glTranslatef(x, y, 0.0);

glRotatef(angle, 0.0, 0.0, 1.0);

if (thrust)

{

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_LINE\_STRIP);

glVertex2f(-0.75, -0.5);

glVertex2f(-1.75, 0);

glVertex2f(-0.75, 0.5);

glEnd();

}

glColor3f(1.0, 1.0, 0.0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(3.5,0.0);

glVertex2f(0.5, -0.25);

glVertex2f(-1.0, -1.0);

glVertex2f(-0.5, 0.0);

glVertex2f(-1.0, 1.0);

glVertex2f(0.5,0.25);

glVertex2f(3.5, 0.0);

glEnd();

if (shield) {

drawshield();

}

glPopMatrix();

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glClearColor(0.0,0.0,0.0,0.0);

if(flag2==0)

{

screen();

}

else

{

drawJet(angle);

drawBullets();

drawsquare();

glFlush();

}

glutSwapBuffers();

}

void idle(void)

{

int time, delta;

time = glutGet(GLUT\_ELAPSED\_TIME);

if (resuming) {

lastTime = time;

resuming = 0;

}

if (left) {

delta = time - leftTime;

angle = angle + delta \* 0.4;

leftTime = time;

}

if (right) {

delta = time - rightTime;

angle = angle - delta \* 0.4;

rightTime = time;

}

if (thrust) {

delta = time - thrustTime;

v = delta \* 0.000004;

xv = xv + cos(angle\*M\_PI/180.0) \* v;

yv = yv + sin(angle\*M\_PI/180.0) \* v;

thrustTime = time;

}

delta = time - lastTime;

x = x + xv \* delta;

y = y + yv \* delta;

x = x / 40.0;

x = (x - floor(x))\*40.0;

y = y / 40.0;

y = (y - floor(y))\*40.0;

lastTime = time;

advanceBullets(delta, time);

glutPostWindowRedisplay(currentWindow);

}

void visible(int vis)

{

if (vis == GLUT\_VISIBLE) {

if (!paused) {

glutIdleFunc(idle);

}

} else {

glutIdleFunc(NULL);

}

}

void key(unsigned char key, int px, int py)

{

switch (key) {

case 27:

exit(0);

break;

case 'S':

case 's':

shield = 1;

break;

case 'C':

case 'c':

cursor = !cursor;

glutSetCursor(

cursor ? GLUT\_CURSOR\_INHERIT : GLUT\_CURSOR\_NONE);

break;

case 'z':

case 'Z':

x = 20;

y = 20;

xv = 0;

yv = 0;

break;

case 'f':

case 'F':

glutGameModeString("400x300:16@60");

glutEnterGameMode();

initWindow();

break;

case 'l':

if (originalWindow != 0 && currentWindow != originalWindow) {

glutLeaveGameMode();

currentWindow = originalWindow;

}

break;

case 'P':

case 'p':

paused = !paused;

if (paused)

{

glutIdleFunc(NULL);

} else

{

glutIdleFunc(idle);

resuming = 1;

}

break;

case 'Q':

case 'q':

case ' ':

shotBullet();

break;

case 'w':

case 'W':

flag2=1;

break;

case 'a':

flag3=1;

break;

}

}

void keyup(unsigned char key, int x, int y)

{

switch (key) {

case 'S':

case 's':

shield = 0;

break;

}

}

void special(int key, int x, int y)

{

switch (key) {

case GLUT\_KEY\_UP:

thrust = 1;

thrustTime = glutGet(GLUT\_ELAPSED\_TIME);

break;

case GLUT\_KEY\_LEFT:

left = 1;

leftTime = glutGet(GLUT\_ELAPSED\_TIME);

break;

case GLUT\_KEY\_RIGHT:

right = 1;

rightTime = glutGet(GLUT\_ELAPSED\_TIME);

break;

}

}

void

specialup(int key, int x, int y)

{

switch (key) {

case GLUT\_KEY\_UP:

thrust = 0;

break;

case GLUT\_KEY\_LEFT:

left = 0;

break;

case GLUT\_KEY\_RIGHT:

right = 0;

break;

}

}

void initWindow(void)

{

glutDisplayFunc(display);

glutVisibilityFunc(visible);

glutKeyboardFunc(key);

glutKeyboardUpFunc(keyup);

glutSpecialFunc(special);

glutSpecialUpFunc(specialup);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0, 40, 0, 40, 0, 40);

glMatrixMode(GL\_MODELVIEW);

glPointSize(3.0);

currentWindow = glutGetWindow();

}

void reshape (int w, int h)

{

glViewport(0,0,w,h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if(w<=h)

gluOrtho2D(0.0,40.0,0.0\*(GLfloat)h/(GLfloat)w,40\*(GLfloat)h/(GLfloat)w);

else

gluOrtho2D(0.0\*(GLfloat)w/(GLfloat)h,40.0\*(GLfloat)w/(GLfloat)h,0.0,40.0);

glMatrixMode(GL\_MODELVIEW);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB | GLUT\_DOUBLE);

if (argc > 1 && !strcmp(argv[1], "-fullscreen"))

{

glutGameModeString("640x480:16@60");

glutEnterGameMode();

} else

{

originalWindow = glutCreateWindow("Fighter Jet");

}

glutReshapeFunc(reshape);

glutIdleFunc(idle);

initWindow();

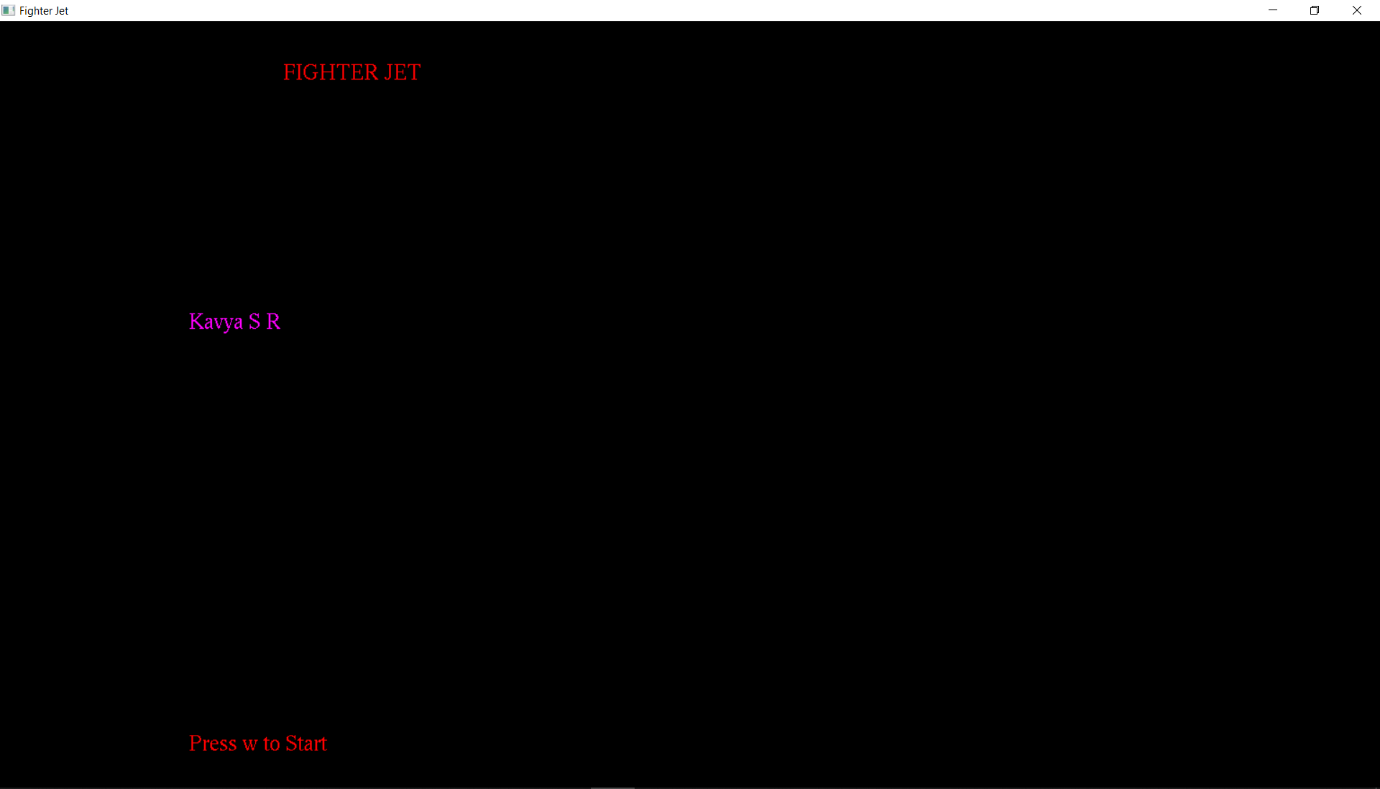
glutMainLoop();

return 0;

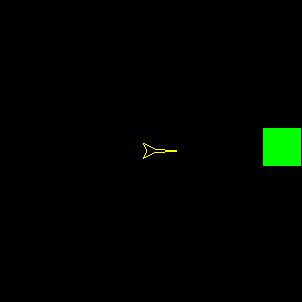
}

Chapter 8

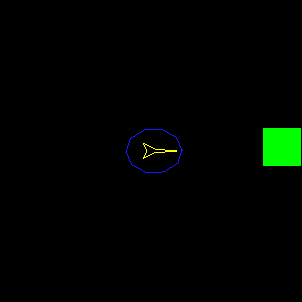
**SNAPSHOTS**

****

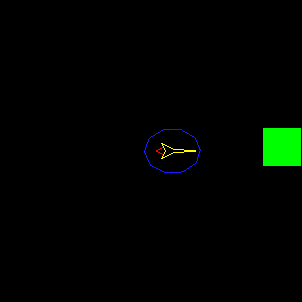
This above snapshot shows the project title and student name

****

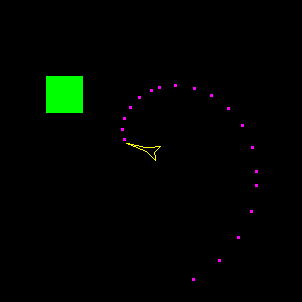
The above snapshot shows Jet and object



This above snapshot shows jet with protective shield



This above snapshot shows jet with protective shield and fir



This above snapshot shows Jet with firing bullets to destroy the object

Chapter 9

**CONCLUSION**

This mini project on FIGHTER JET using OpenGL is a reliable graphics project that provides a menu-based user interface to run the program. The graphics creation and manipulation algorithm in the project have been implemented and tested to ensure the efficiency of operation and they were found to be quite satisfactory. The graphics editor has a good and user-friendly interface which enable the user to get better and easier interaction with the software. After the completion of the development and study of this project we came to conclusion that the computer graphics in C using OpenGL can be used to develop much better and complex software's that include 2D,3D image processing and animation. The program is also mainly concerned with animation and more of animation implementations. This type of implementations is now a days used in game development.