UFO VS SPACECRAFT

Project Report submitted to

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

in partial fulfillment of the requirements for the
Computer Graphics Laboratory (18CSL67) of 6th semester
for the degree

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

Submitted by

SURAKSHA RACHANA 1KG19CS103

Under the Guidance of

Mrs.Amitha S
Assistant Professor
K S School of Engineering and Management

Department of Computer Science and Engineering
K.S. School of Engineering and Management
No. 15, Mallasandra, off Kanakapura Road, Bengaluru-560109

2021-2022

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the mini-project entitled **UFO VS SPACECRAFT** is a bonafide work carried out by

SURAKSHA RACHANA 1KG19CS103

in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of **Visvesvaraya Technological University, Belagavi,** during the year 2020-2021. It is certified that all the suggestions indicated during internal assessment have been incorporated in the report and this report satisfies the academic requirement with respect to **Mini-project of Computer Graphics Laboratory (18CSL67) of VI semester** prescribed for the degree.

Mrs Amitha S	**Dr Venkata Rao**	**Dr K Rama Narasimha**
Assistant Professor	Professor and Head	Principal
Project Guide	Dept. of CSE	KSSEM

University Examiners:

Name and Signature of Examiner-1

Name and Signature of Examiner-2

K.S. SCHOOL OF ENGINEERING AND MANAGEMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

I

SURAKSHA RACHANA 1KG19CS103

the students of **BE VI Semester (Computer Science and Engineering)** declare that the mini-project entitled **"UFO vs SPACECRAFT"** is carried out by us as a partial fulfillment of academic requirement of degree under **Visvesvaraya Technological University.** The content in the report are original and are free from plagiarism and other academic dishonesty and are not submitted to any other University either partially or wholly for the award of any other degree.

NAME	USN	Signature
SURAKSHA RACHANA	1KG19CS103	

Date:

Place: Bengaluru

ACKNOWLEDGEMENT

The successful completion of this project was made possible with the help of guidance received from our faculty members. I would like to avail this opportunity to express our sincere thanks and gratitude to all of them.

I am grateful to our management for providing the necessary infrastructure and an ambience environment to work. I express my profound gratitude to **Dr. K Rama Narasimha, Principal and Dr Venkata Rao, Head, Department of Computer Science and Engineering, KSSEM, Bengaluru** for giving us opportunity and support to complete the mini-project.

I am grateful to my guide **Mrs. Amitha S, Assistant Professor, Department of Computer Science and Engineering, KSSEM, Bengaluru** for her valuable suggestions and advice throughout my work period.

I would also like to thank all the staff members of Department of Computer Science and Engineering for their support and encouragement. Finally, I would like to thank all of my friends without whose help and encouragement this project would have been impossible.

Definitely most, I want to thank my family. Words cannot express i thanks to my family members for all their love and encouragement.

Suraksha Rachana 1KG19CS103

ABSTRACT

The aim of this project is to develop a graphics package which supports basic operation of creating menu, translation, rotation etc on the objects. This project uses the Conceptual framework model for an interactive graphics system.

This is about a simple animation where there is a UFO in red and a spacecraft, in pink color .Here UFO is preparing to invade and destroy our planet earth. On the other hand the spacecraft has departed with a mission the protect the earth. The spacecraft is allowed to shoot the UFO with a laser light and even the vise versa i.e UFO is also allowed to invade the spacecraft with laser light. It also has features of moving objects in all the four directions, These movements can be made using the arrow keys on the keyboard.

The instructions are very simple to interpret and the interface is user friendly. Since this uses Opengl platform, it becomes portable.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Problem statement:

This is about a simple animation where there is a UFO in red and a spacecraft, in pink color .Here UFO is preparing to invade and destroy our planet earth. On the other hand the spacecraft has departed with a mission ti protect the earth. The spacecraft is allowed to shoot the UFO with a laser light and even the vise versa i.e UFO is also allowed to invade the spacecraft with laser light. It also has features of moving objects in all the four directions, These movements can be made using the arrow keys on the keboard.

1.2 Scope of the project:

This project uses the Conceptual framework model for an interactive graphics system. It mainly consists of Application Model, Application Program, and Graphics system and hardware components. Each of these systems interacts to produce an effective graphics display.

1.3 Description of project:

This is about a simple animation where there is a UFO in red and a spacecraft, in pink color Here UFO is preparing to invade and destroy our planet earth. On the other hand the spacecraft has departed with a mission ti protect the earth. The spacecraft is allowed to shoot the UFO with a laser light and even the vise versa i.e UFO is also allowed to invade the spacecraft with laser light. It also has features of moving objects in all the four directions, These movements can be made using the arrow keys on the keboard.

The actual implementations are explained in detail along with the functions in this project. The controls are:

- ‘w’ - > To move up.
- ‘s’ - > To move down.
- ‘d‘ - > To move right.
- ‘a’ - > To move left.
- ‘c’ - > To shoot.

Chapter 2

HARDWARE AND SOFTWARE REQUIREMENTS

2.1 Hardware Requirements

NAME OF THE COMPONENT	SPECIFICATION
Processor	Intel 386 onwards Compatible Hardware.
RAM	128Mb RAM
Hard Disk	40 GB
Monitor	1024*786 display resolution
Keyboard	Standard 101 key Keyboard

Table 2.1: Hardware requirements

2.2 Software Requirements

NAME OF THE COMPONENT	SPECIFICATION
Operating System	UBNTU 10.10,WINDOWS 10
Language Tool	C/C++ using OpenGL
Compiler	gcc version 4.5.1
Libraries	Supporting OpenGL & 32 bit color resolution
Documentation	MS-Word.

Table 2.2: Software requirements

Chapter 3

DESIGN

3.1 Flow Chart

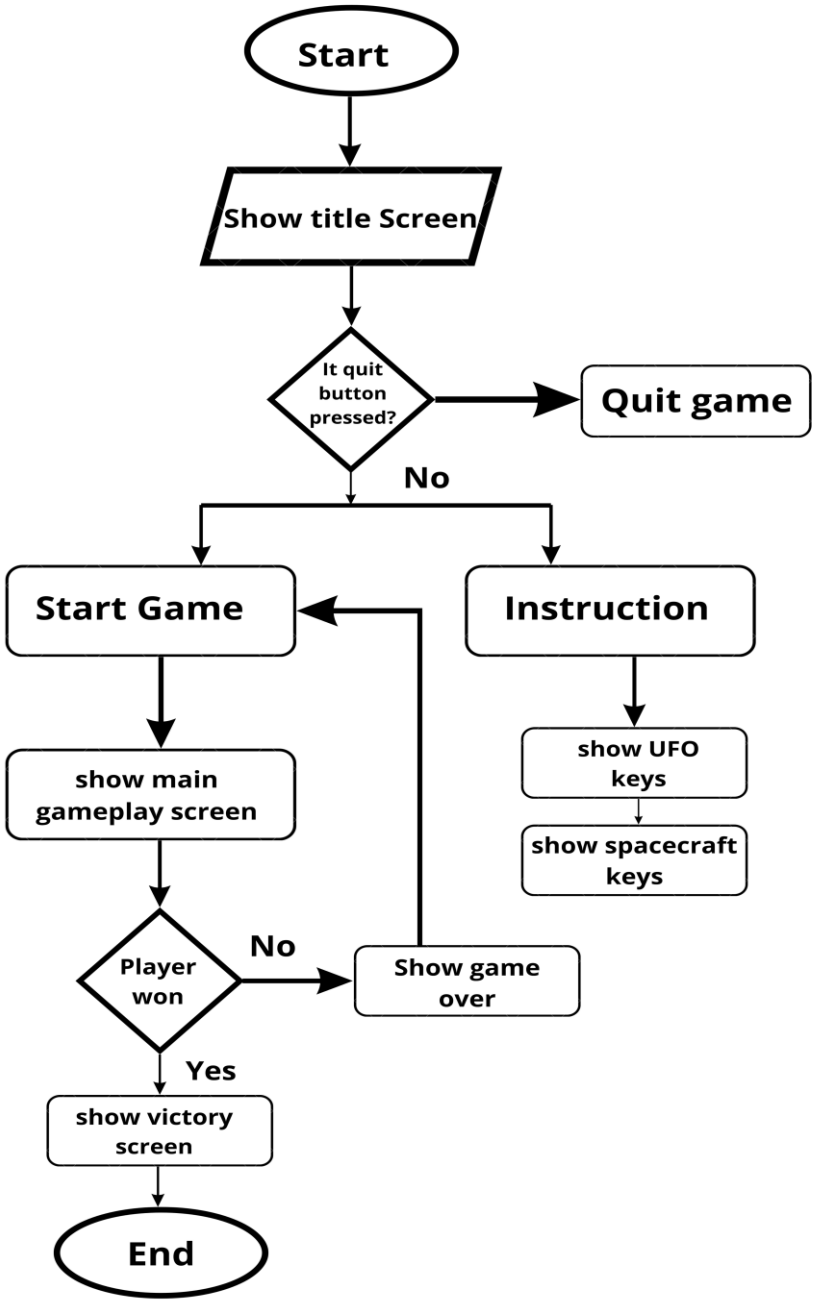

Fig 3.1 Flow Chart of UFO vs SPACECRAFT

3.2 Opengl Commands

glLoadIdentity: Replaces the current matrix with the identity matrix. It is semantically equivalent to calling glLoadMatrix with the identity matrix.

glEnable and glDisable: Enable and disable various capabilities.

glGet: To determine the current setting of any capability.

glBegin and glEnd: Delimit the vertices that define a primitive or a group of like primitives.

glBegin: Accepts a single argument that specifies in which of ten ways the vertices are interpreted.

glFlush: Different GL implementations buffer commands in several different locations,including network buffers and the graphics accelerator itself. glFlush empties all of these buffers, causing all issued commands to be executed as quickly as they are accepted by the actual rendering engine. Though this execution may not be completed in any particular time period, it does complete in finite time.

glColorMaterial: Specifies which material parameters track the current color. When GL_COLOR_MATERIAL is enabled,the material parameter or parameters specified by mode, of the material or materials specified by face, track the current color at all times.

glTranslate: To multiply the current matrix by a translation matrix. **glRotate:**

To multiply the current matrix by a rotation matrix. **glPushMatrix and**

glPopMatrix: To push and pop the current matrix stack. **glColor:** To set the

current color.

glutAddMenuEntry: To add a menu entry to the bottom of the current menu.

glutSwapBuffers: To swap the buffers of the current window if double buffered**.**

glutSolidSphere: To render a solid or wireframe sphere respectively.

glutMainLoop: To enters the GLUT event processing loop.

glutInitDisplayMode: To sets the initial display mode.

glutInit: To initialize the glut library.

glMatrixMode: To specify which matrix is the current matrix.

glClearDepth: To specify the clear value for the depth buffer.

glutKeyboardFunc(myKey): Refers to the keyboard callback function.Here keyboard interface is given to quit, the user can quit by pressing *q' and to see next example of the implementation, the user should press 'n'.

glutInit(int argc, char*arg): Initializes GLUT< the arguments from main are passed in and can by the application.

glutCreate Window(char*title): Creates a window on the display. The string title can be used to label the window. The return value provides a reference to the window that can be used when there are multiple windows.

glutinit Display mode(unsigned int mode): Requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model (GLUT_RGB<GLUT_INDEX) and buffering (GLUT_SINGLE<GLUT DOUBLE).

glutinit Window Size(int width, int heights): Specifies the initial height and width of the window in pixels.

glutinit Window Position(int x,inty): Specifies the initial position of the top-left corner of the window in pixels.

glViewport(int x,int y GLsizei width.GLsizei height): Specifies a width * height viewport in pixels whose lower-left corner is at (x,y) measured from the origin of the window.

glut Main Loop(): Cause the program to ente an event processing loop.it should be the statement in main.

glutPostRedisplay: Requests that the display callback be executed after the current callback.

glcallList(): glCallLists causes each display list in the list of names passed as lists to be executed.As a result, the commands saved in each display list are executed in order, just as if they were called without using a display list. Names of display lists that have not been defined are ignored.

glEnable() and glDisable(): glEnable and glDisable enable and disable various capabilities. Use glIsEnabled or glGet to determine the current setting of any capability.

Chapter 4

IMPLEMENTATION

4.1 Code Snippets

```
 #ifdef _WIN32
#include<windows.h>
#endif
#include<stdio.h>
#include<stdlib.h>
#include<GL/glut.h>
#include<math.h>
#define GL_SILENCE_DEPRECATION

#define XMAX 1200
#define YMAX 700
#define SPACESHIP_SPEED 20
#define TOP 0
#define RIGHT 1
#define BOTTOM 2
#define LEFT 3


GLint m_viewport[4];
bool mButtonPressed = false;
float mouseX, mouseY;
enum view {INTRO, MENU, INSTRUCTIONS, GAME, GAMEOVER};
view viewPage = INTRO; // initial value
bool keyStates[256] = {false};
bool direction[4] = {false};
bool laser1Dir[2] = {false};
bool laser2Dir[2] = {false};

int alienLife1 = 100;
int alienLife2 = 100;
bool gameOver = false;
float xOne = 500, yOne = 0;
float xTwo = 500, yTwo = 0;
bool laser1 = false, laser2 = false;
GLint CI=0;
GLfloat a[][2]={0,-50, 70,-50, 70,70, -70,70};
GLfloat LightColor[][3]={1,1,0,   0,1,1,   0,1,0};
GLfloat AlienBody[][2]={{-4,9}, {-6,0}, {0,0}, {0.5,9}, {0.15,12}, {-14,18}, {-19,10}, {-
20,0},{-6,0}};
GLfloat AlienCollar[][2]={{-9,10.5}, {-6,11}, {-5,12}, {6,18}, {10,20}, {13,23}, {16,30},
{19,39}, {16,38},
                                                {10,37}, {-13,39}, {-18,41}, {-20,43}, {-20.5,42}, {-
21,30}, {-19.5,23}, {-19,20},
```

```
                                        {-14,16}, {-15,17},{-13,13},  {-9,10.5}};
GLfloat ALienFace[][2]={{-6,11}, {-4.5,18}, {0.5,20}, {0.,20.5}, {0.1,19.5}, {1.8,19}, {5,20},
{7,23}, {9,29},
                                        {6,29.5}, {5,28}, {7,30}, {10,38},{11,38}, {11,40},
{11.5,48}, {10,50.5},{8.5,51}, {6,52},
                                        {1,51}, {-3,50},{-1,51}, {-3,52}, {-5,52.5}, {-6,52},
{-9,51}, {-10.5,50}, {-12,49}, {-12.5,47},
                                        {-12,43}, {-13,40}, {-12,38.5}, {-13.5,33},{-15,38},{-
14.5,32},  {-14,28}, {-13.5,33}, {-14,28},
                                        {-13.8,24}, {-13,20}, {-11,19}, {-10.5,12}, {-6,11} } ;
GLfloat ALienBeak[][2]={{-6,21.5}, {-6.5,22}, {-9,21}, {-11,20.5}, {-20,20}, {-14,23}, {-
9.5,28}, {-7,27}, {-6,26.5},
                                        {-4.5,23}, {-4,21}, {-6,19.5}, {-8.5,19}, {-10,19.5}, {-
11,20.5} };


void displayRasterText(float x ,float y ,float z ,char *stringToDisplay) {
      glRasterPos3f(x, y, z);
      for(char* c = stringToDisplay; *c != '\0'; c++){
            glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24 , *c);
      }
}

void init()
{
      glClearColor(0.0,0.0,0.0,0);
      glColor3f(1.0,0.0,0.0);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();

   gluOrtho2D(-1200,1200,-700,700);              //<-----CHANGE THIS TO GET EXTRA
SPACE
//  gluOrtho2D(-200,200,-200,200);
      glMatrixMode(GL_MODELVIEW);
}

void introScreen()
{
      glClear(GL_COLOR_BUFFER_BIT);

            glColor3f(1.0, 0.0, 0.0);
      displayRasterText(-425, 490, 0.0,"NMAM INSTITUTE OF TECHNOLOGY");
            glColor3f(1.0, 1.0, 1.0);
      displayRasterText(-700, 385, 0.0,"DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING");
            glColor3f(0.0, 0.0, 1.0);
      displayRasterText(-225, 300, 0.0,"A MINI PROJECT ON ");
            glColor3f(1.0, 0.0, 1.0);
      displayRasterText(-125, 225, 0.0,"Space Shooter");
            glColor3f(1.0, 0.7, 0.8);
      displayRasterText(-100, 150, 0.0,"created by");
            glColor3f(1.0, 1.0, 1.0);
```

```
        displayRasterText(-130, 80, 0.0,"SHOOTERS");
                glColor3f(1.0, 0.0, 0.0);
         displayRasterText(-800, -100, 0.0," STUDENT NAMES");
                glColor3f(1.0, 1.0, 1.0);
        displayRasterText(-800, -200, 0.0," Saurav N Shetty");
        displayRasterText(-800, -285, 0.0," Rajath R Pai");
                glColor3f(1.0, 0.0, 0.0);
        displayRasterText(500, -100, 0.0,"Under the Guidance of");
                glColor3f(1.0, 1.0, 1.0);
        displayRasterText(500, -200, 0.0,"Prof X");
                glColor3f(1.0, 0.0, 0.0);
        displayRasterText(-250, -400, 0.0,"Academic Year 2020-2021");
        glColor3f(1.0, 1.0, 1.0);
        displayRasterText(-300, -550, 0.0,"Press ENTER to start the game");
        glFlush();
        glutSwapBuffers();
}

void startScreenDisplay()
{
        glLineWidth(10);
        //SetDisplayMode(MENU_SCREEN);

        glColor3f(1,0,0);
        glBegin(GL_LINE_LOOP);              //Border
                glVertex2f(-750 ,-500);
                glVertex2f(-750 ,550);
                glVertex2f(750 ,550);
                glVertex2f(750 ,-500);
        glEnd();

        glLineWidth(1);

        glColor3f(1, 1, 0);
        glBegin(GL_POLYGON);                            //START GAME PLOYGON
                glVertex2f(-200 ,300);
                glVertex2f(-200 ,400);
                glVertex2f(200 ,400);
                glVertex2f(200 ,300);
        glEnd();

        glBegin(GL_POLYGON);                            //INSTRUCTIONS POLYGON
                glVertex2f(-200, 50);
                glVertex2f(-200 ,150);
                glVertex2f(200 ,150);
                glVertex2f(200 ,50);
        glEnd();

        glBegin(GL_POLYGON);                            //QUIT POLYGON
                glVertex2f(-200 ,-200);
                glVertex2f(-200 ,-100);
                glVertex2f(200, -100);
```

```
            glVertex2f(200, -200);
        glEnd();

        if(mouseX>=-100 && mouseX<=100 && mouseY>=150 && mouseY<=200){
            glColor3f(0 ,0 ,1) ;
            if(mButtonPressed){
                alienLife1 = alienLife2 = 100;
                viewPage = GAME;
                mButtonPressed = false;
            }
        } else
            glColor3f(0 , 0, 0);

        displayRasterText(-100 ,340 ,0.4 ,"Start Game");

        if(mouseX>=-100 && mouseX<=100 && mouseY>=30 && mouseY<=80) {
            glColor3f(0 ,0 ,1);
            if(mButtonPressed){
                viewPage = INSTRUCTIONS;
                printf("button pressed bitch\n");
                mButtonPressed = false;
            }
        } else
            glColor3f(0 , 0, 0);
        displayRasterText(-120 ,80 ,0.4 ,"Instructions");

        if(mouseX>=-100 && mouseX<=100 && mouseY>=-90 && mouseY<=-40){
            glColor3f(0 ,0 ,1);
            if(mButtonPressed){
                mButtonPressed = false;
                exit(0);
            }
        }
        else
            glColor3f(0 , 0, 0);
        displayRasterText(-100 ,-170 ,0.4 ,"    Quit");
        glutPostRedisplay();
}

void backButton() {
        if(mouseX <= -450 && mouseX >= -500 && mouseY >= -275 && mouseY <= -250){
                glColor3f(0, 0, 1);
                if(mButtonPressed) {
                        viewPage = MENU;
                        mButtonPressed = false;
                        //instructionsGame = false;
                        glutPostRedisplay();
                }
        }
        else glColor3f(1, 0, 0);
        displayRasterText(-1000 ,-550 ,0, "Back");
}
```

```
void instructionsScreenDisplay()
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	//SetDisplayMode(MENU_SCREEN);
	//colorBackground();
	glColor3f(1, 0, 0);
	displayRasterText(-900 ,550 ,0.4 ,"INSTRUCTIONS");
   glColor3f(1, 0, 0);
	displayRasterText(-1000 ,400 ,0.4 ,"PLAYER 1");
	displayRasterText(200 ,400 ,0.4 ,"PLAYER 2");
	glColor3f(1, 1, 1);
	displayRasterText(-1100 ,300 ,0.4 ,"Key 'w' to move up.");
	displayRasterText(-1100 ,200 ,0.4 ,"Key 's' to move down.");
	displayRasterText(-1100 ,100 ,0.4 ,"Key 'd' to move right.");
	displayRasterText(-1100 ,0 ,0.4 ,"Key 'a' to move left.");
	displayRasterText(100 ,300 ,0.4 ,"Key 'i' to move up.");
   displayRasterText(100 ,200 ,0.4 ,"Key 'k' to move down.");
   displayRasterText(100 ,100 ,0.4 ,"Key 'j' to move right.");
   displayRasterText(100 ,0 ,0.4 ,"Key 'l' to move left.");
	displayRasterText(-1100 ,-100 ,0.4 ,"Key 'c' to shoot, Use 'w' and 's' to change direction.");
	displayRasterText(100 ,-100 ,0.4 ,"Key 'm' to shoot, Use 'i' and 'k' to change direction.");
	//displayRasterText(-1100 ,-100 ,0.4 ,"The packet can be placed only when 's' is pressed
before.");
	displayRasterText(-1100, -300,0.4,"The Objective is to kill your opponent.");
	displayRasterText(-1100 ,-370 ,0.4 ,"Each time a player gets shot, LIFE decreases by 5
points.");
	backButton();
	//if(previousScreen)
	//	nextScreen = false ,previousScreen = false; //as set by backButton()
}

void DrawAlienBody(bool isPlayer1)
{
	if(isPlayer1)
		glColor3f(0,1,0);
	else
		glColor3f(1,1,0);		//BODY color
	glBegin(GL_POLYGON);
	for(int i=0;i<=8;i++)
		glVertex2fv(AlienBody[i]);
	glEnd();

	glColor3f(0,0,0);			//BODY Outline
	glLineWidth(1);
	glBegin(GL_LINE_STRIP);
	for(int i=0;i<=8;i++)
		glVertex2fv(AlienBody[i]);
	glEnd();

	glBegin(GL_LINES);		//BODY effect
		glVertex2f(-13,11);
```

```
            glVertex2f(-15,9);
        glEnd();
}
void DrawAlienCollar()
{
        glColor3f(1,0,0);                       //COLLAR
        glBegin(GL_POLYGON);
        for(int i=0;i<=20 ;i++)
                glVertex2fv(AlienCollar[i]);
        glEnd();

        glColor3f(0,0,0);                       //COLLAR outline
        glBegin(GL_LINE_STRIP);
        for(int i=0;i<=20 ;i++)
                glVertex2fv(AlienCollar[i]);
        glEnd();
}
void DrawAlienFace(bool isPlayer1)
{
        //glColor3f(0.6,0.0,0.286);                     //FACE
        //glColor3f(0.8,0.2,0.1);
        //glColor3f(0,0.5,1);
        //if(isPlayer1)
        glColor3f(0,0,1);
        // else
        //      glColor3f(0,1,0);

        glBegin(GL_POLYGON);
        for(int i=0;i<=42 ;i++)
                glVertex2fv(ALienFace[i]);
        glEnd();

        glColor3f(0,0,0);                       //FACE outline
        glBegin(GL_LINE_STRIP);
        for(int i=0;i<=42 ;i++)
                glVertex2fv(ALienFace[i]);
        glEnd();

        glBegin(GL_LINE_STRIP);     //EAR effect
                glVertex2f(3.3,22);
                glVertex2f(4.4,23.5);
                glVertex2f(6.3,26);
        glEnd();
}
void DrawAlienBeak()
{
        glColor3f(1,1,0);                       //BEAK color
        glBegin(GL_POLYGON);
        for(int i=0;i<=14 ;i++)
                glVertex2fv(ALienBeak[i]);
        glEnd();
```

```
	glColor3f(0,0,0);				//BEAK outline
	glBegin(GL_LINE_STRIP);
	for(int i=0;i<=14 ;i++)
		glVertex2fv(ALienBeak[i]);
	glEnd();
}
void DrawAlienEyes(bool isPlayer1)
{
	// if(isPlayer1)
	glColor3f(0,1,1);
	// else
	//	glColor3f(0,0,0);

	glPushMatrix();
	glRotated(-10,0,0,1);
	glTranslated(-6,32.5,0);	//Left eye
	glScalef(2.5,4,0);
	glutSolidSphere(1,20,30);
	glPopMatrix();

	glPushMatrix();
	glRotated(-1,0,0,1);
	glTranslated(-8,36,0);						//Right eye
	glScalef(2.5,4,0);
	glutSolidSphere(1,100,100);
	glPopMatrix();
}
void DrawAlien(bool isPlayer1)
{
	DrawAlienBody(isPlayer1);
	DrawAlienCollar();
	DrawAlienFace(isPlayer1);
	DrawAlienBeak();
	DrawAlienEyes(isPlayer1);
}
void DrawSpaceshipBody(bool isPlayer1)
{
	if(isPlayer1)
		glColor3f(1, 0, 0);			//BASE
	else
		glColor3f(0.5, 0, 0.5);

	glPushMatrix();
	glScalef(70,20,1);
	glutSolidSphere(1,50,50);
	glPopMatrix();

	glPushMatrix();						//LIGHTS
	glScalef(3,3,1);
	glTranslated(-20,0,0);			//1
	glColor3fv(LightColor[(CI+0)%3]);
	glutSolidSphere(1,1000,1000);
```

```
	glTranslated(5,0,0);                    //2
	glColor3fv(LightColor[(CI+1)%3]);
	glutSolidSphere(1,1000,1000);
	glTranslated(5,0,0);                    //3
	glColor3fv(LightColor[(CI+2)%3]);
	glutSolidSphere(1,1000,1000);
	glTranslated(5,0,0);                    //4
	glColor3fv(LightColor[(CI+0)%3]);
	glutSolidSphere(1,1000,1000);
	glTranslated(5,0,0);                    //5
	glColor3fv(LightColor[(CI+1)%3]);
	glutSolidSphere(1,1000,1000);
	glTranslated(5,0,0);                    //6
	glColor3fv(LightColor[(CI+2)%3]);
	glutSolidSphere(1,1000,1000);
	glTranslated(5,0,0);                    //7
	glColor3fv(LightColor[(CI+0)%3]);
	glutSolidSphere(1,1000,1000);
	glTranslated(5,0,0);                    //8
	glColor3fv(LightColor[(CI+1)%3]);
	glutSolidSphere(1,1000,1000);
	glTranslated(5,0,0);                    //9
	glColor3fv(LightColor[(CI+2)%3]);
	glutSolidSphere(1,1000,1000);

	glPopMatrix();
}
void DrawSteeringWheel()
{
	glPushMatrix();
	glLineWidth(3);
	glColor3f(0.20,0.,0.20);
	glScalef(7,4,1);
	glTranslated(-1.9,5.5,0);
	glutWireSphere(1,8,8);
	glPopMatrix();

}
void DrawSpaceshipDoom()
{
	glColor4f(0.7,1,1,0.0011);
	glPushMatrix();
	glTranslated(0,30,0);
	glScalef(35,50,1);
	glutSolidSphere(1,50,50);
	glPopMatrix();
}

void DrawLaser(int x, int y, bool dir[]) {
	//glPushMatrix();
	int xend = -XMAX, yend = y;
	if(dir[0])
```

```
            yend = YMAX;
        else if(dir[1])
            yend = -YMAX;
        glLineWidth(5);
        glColor3f(1, 0, 0);
        glBegin(GL_LINES);
            glVertex2f(x, y);
            glVertex2f(xend, yend);
        glEnd();
        //glPopMatrix();
}

void SpaceshipCreate(int x, int y, bool isPlayer1){
        glPushMatrix();
        glTranslated(x,y,0);
        // if(!checkIfSpaceShipIsSafe() && alienLife1 ){
        //      alienLife1-=10;
        //      xStart -= 23;
        // }
        DrawSpaceshipDoom();
        glPushMatrix();
        glTranslated(4,19,0);
        DrawAlien(isPlayer1);
        glPopMatrix();
        DrawSteeringWheel();
        DrawSpaceshipBody(isPlayer1);
        // DrawSpaceShipLazer();
        // if(mButtonPressed) {
        //      DrawLazerBeam();
        // }
        glEnd();
        glPopMatrix();
}

void DisplayHealthBar1() {
        char temp1[40];
        glColor3f(1 ,1 ,1);
        sprintf(temp1,"  LIFE = %d",alienLife1);
        displayRasterText(-1100 ,600 ,0.4 ,temp1);
        glColor3f(1 ,0 ,0);
}

void DisplayHealthBar2() {
        char temp2[40];
        glColor3f(1 ,1 ,1);
        sprintf(temp2,"  LIFE = %d",alienLife2);
        displayRasterText(800 ,600 ,0.4 ,temp2);
        glColor3f(1 ,0 ,0);
}

void checkLaserContact(int x, int y, bool dir[], int xp, int yp, bool player1) {
        int xend = -XMAX, yend = y;
```

```
	xp += 8; yp += 8; // moving circle slightly up to fix laser issue
	if(dir[0])
		yend = YMAX;
	else if(dir[1])
		yend = -YMAX;

	// Here we find out if the laser(line) intersects with spaceship(circle)
	// by solving the equations for the same and finding the discriminant of the
	// quadratic equation obtained
	float m = (float)(yend - y) / (float)(xend - x);
	float k = y - m * x ;
	int r = 50; // approx radius of the spaceship

	//calculating value of b, a, and c needed to find discriminant
	float b = 2 * xp - 2 * m * (k - yp);
	float a = 1 + m * m;
	float c = xp * xp + (k - yp) * (k - yp) - r * r;

	float d = (b * b - 4 * a * c); // discriminant for the equation
	printf("\nDisc: %f x: %d, y: %d, xp: %d, yp: %d", d, x, y, xp, yp);
	if(d >= 0) {
		if(player1)
			alienLife1 -= 5;
		else
			alienLife2 -= 5;

		printf("%d %d\n", alienLife1, alienLife2);
	}
}

void gameScreenDisplay()
{
	DisplayHealthBar1();
	DisplayHealthBar2();
	glScalef(2, 2 ,0);

	if(alienLife1 > 0){
		SpaceshipCreate(xOne, yOne, true);
		if(laser1) {
			DrawLaser(xOne, yOne, laser1Dir);
			checkLaserContact(xOne, yOne, laser1Dir, -xTwo, yTwo, true);
		}
	}
	else {
		viewPage = GAMEOVER;
	}

	if(alienLife2 > 0) {
		glPushMatrix();
		glScalef(-1, 1, 1);
		SpaceshipCreate(xTwo, yTwo, false);
		if(laser2) {
```

```
                DrawLaser(xTwo, yTwo, laser2Dir);
                checkLaserContact(xTwo, yTwo, laser2Dir, -xOne, yOne, false);
            }
            glPopMatrix();
        }
        else {
            viewPage = GAMEOVER;
        }

        if(viewPage == GAMEOVER) {
            xOne = xTwo = 500;
            yOne = yTwo = 0;
        }
}

void displayGameOverMessage() {
        glColor3f(1, 1, 0);
        char* message;
        if(alienLife1 > 0)
            message = "Game Over! Player 1 won the game";
        else
            message = "Game Over! Player 2 won the game";

        displayRasterText(-350 ,600 ,0.4 , message);
}

void keyOperations() {
        if(keyStates[13] == true && viewPage == INTRO) {
            viewPage = MENU;
            printf("view value changed to %d", viewPage);
            printf("enter key pressed\n");
        }
        if(viewPage == GAME) {
            laser1Dir[0] = laser1Dir[1] = false;
            laser2Dir[0] = laser2Dir[1] = false;
            if(keyStates['c'] == true) {
                laser2 = true;
                if(keyStates['w'] == true)      laser2Dir[0] = true;
                if(keyStates['s'] == true)      laser2Dir[1] = true;
            }
            else {
                laser2 = false;
                if(keyStates['d'] == true) xTwo-=SPACESHIP_SPEED;
                if(keyStates['a'] == true) xTwo+=SPACESHIP_SPEED;
                if(keyStates['w'] == true) yTwo+=SPACESHIP_SPEED;
                if(keyStates['s'] == true) yTwo-=SPACESHIP_SPEED;
            }

            if(keyStates['m'] == true) {
                laser1 = true;
                if(keyStates['i'] == true) laser1Dir[0] = true;
                if(keyStates['k'] == true) laser1Dir[1] = true;
```

```
		}
		else {
			laser1 = false;
			if(keyStates['l'] == true) xOne+=SPACESHIP_SPEED;
			if(keyStates['j'] == true) xOne-=SPACESHIP_SPEED;
			if(keyStates['i'] == true) yOne+=SPACESHIP_SPEED;
			if(keyStates['k'] == true) yOne-=SPACESHIP_SPEED;
		}
	}
}

void display()
{
	//glClearColor(, 0 , 0, 1);
	keyOperations();
	glClear(GL_COLOR_BUFFER_BIT);

	switch (viewPage)
	{
		case INTRO:
			introScreen();
			break;
		case MENU:
			startScreenDisplay();
			break;
		case INSTRUCTIONS:
			instructionsScreenDisplay();
			break;
		case GAME:
			gameScreenDisplay();
			//reset scaling values
			glScalef(1/2 ,1/2 ,0);
			break;
		case GAMEOVER:
			displayGameOverMessage();
			startScreenDisplay();
			break;
	}

	glFlush();
	glLoadIdentity();
	glutSwapBuffers();
}

// void reshape(GLint w, GLint h)
// {
//	glViewport(0, 0, w, h);
//	glMatrixMode(GL_PROJECTION);
//	glLoadIdentity();
//	if(h>w)
//	{
//		gluOrtho2D(0, 500, ((float)h/(float)w)*(0), ((float)h/(float)w)*500);
```

```
//	}
//	else
//	{
//		gluOrtho2D(((float)w/(float)h)(0), ((float)w/(float)h)(500), 0, 500);
//	}
//	glMatrixMode(GL_MODELVIEW);
//	glutPostRedisplay();
// }

void passiveMotionFunc(int x,int y) {

	//when mouse not clicked
	mouseX = float(x)/(m_viewport[2]/1200.0)-600.0;  //converting screen resolution to ortho 2d spec
	mouseY = -(float(y)/(m_viewport[3]/700.0)-350.0);

	//Do calculations to find value of LaserAngle
	//somethingMovedRecalculateLaserAngle();
	glutPostRedisplay();
}

void mouseClick(int buttonPressed ,int state ,int x, int y) {

	if(buttonPressed == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
		mButtonPressed = true;
	else
		mButtonPressed = false;
	glutPostRedisplay();
}

void keyPressed(unsigned char key, int x, int y)
{
	keyStates[key] = true;
	glutPostRedisplay();
}

void refresh() {
	glutPostRedisplay();
}

void keyReleased(unsigned char key, int x, int y) {
	keyStates[key] = false;
}

int main(int argc, char **argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
   glutInitWindowPosition(0, 0);
   glutInitWindowSize(1200, 600);
   glutCreateWindow("Space Shooter");
   init();
```

```
    //glutReshapeFunc(reshape);
       glutIdleFunc(refresh);
    glutKeyboardFunc(keyPressed);
       glutKeyboardUpFunc(keyReleased);
       glutMouseFunc(mouseClick);
       glutPassiveMotionFunc(passiveMotionFunc);
       glGetIntegerv(GL_VIEWPORT ,m_viewport);
    glutDisplayFunc(display);
    glutMainLoop();
}
```

Chapter 5

RESULT AND DISCUSSIONS

5.1 Introduction Page

Fig 5.1: Introduction Page

5.2 Start Position

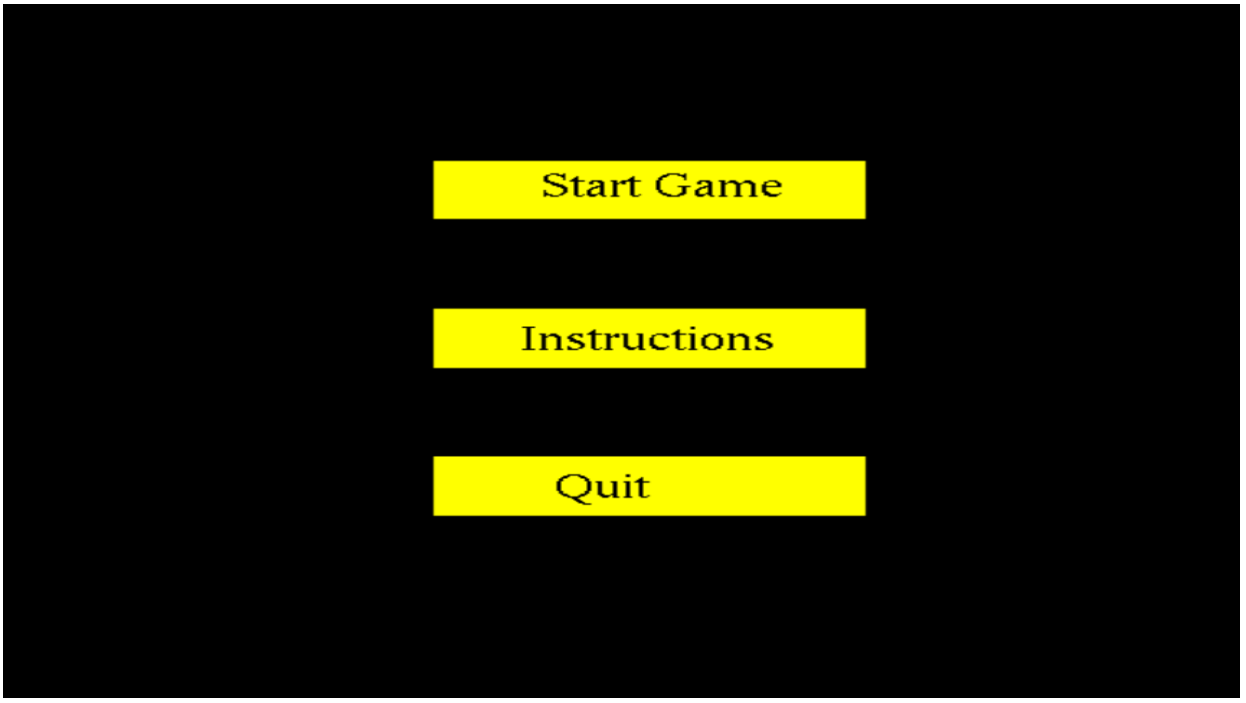

Fig 5.2: Start Position

5.3 Instructions Page

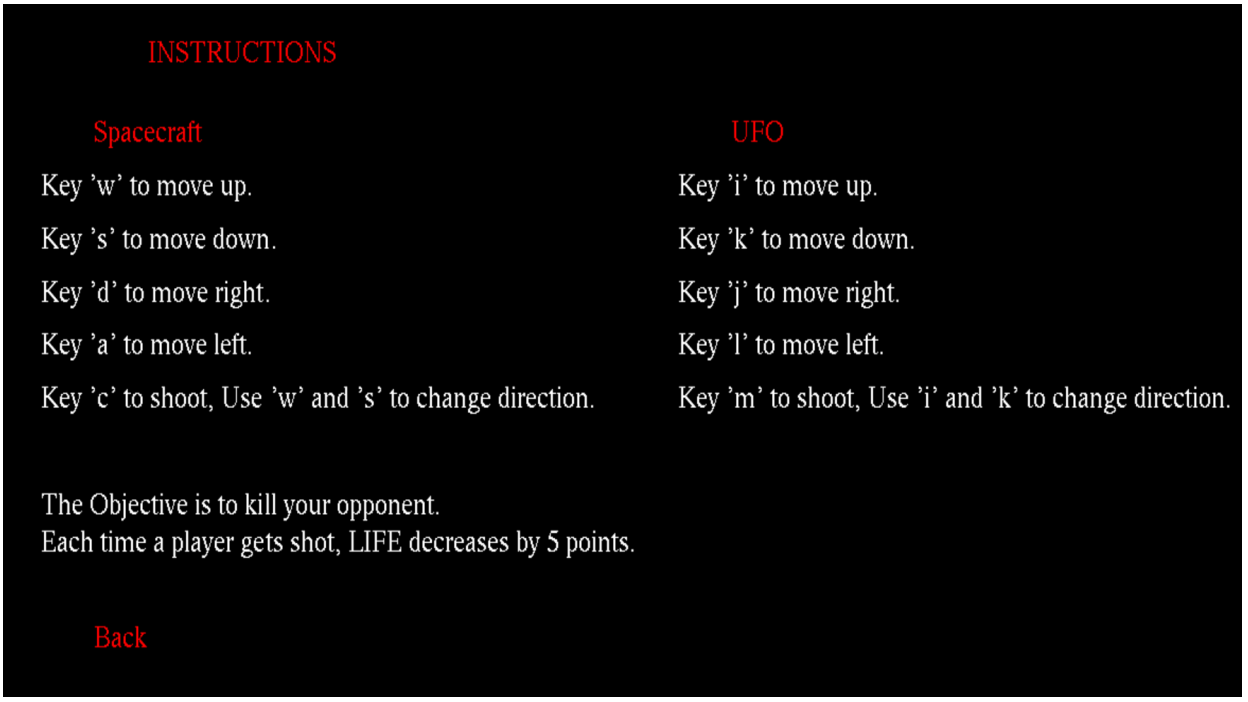

Fig 5.3: Instruction Page

5.4 Gaming Screen

Fig 5.4: Gaming Screen

Chapter 6

CONCLUSION AND FUTURE ENHANCEMENTS

6.1 CONCLUSION

An attempt has been made to develop an OpenGL package which meets necessary requirements of the user successfully. Since it is user friendly it enables the user to interact efficiently and easily. The development of mini project has given us a good exposure to OpenGL by which we have learnt some of the techniques which help in the development of animation picture and gaming. Hence it is helpful for us even to take this field as our career too and develop some other features in OpenGL and provide as token of contribution to the graphics world.

6.2 FUTURE ENHANCEMENT

The mini project developed has a scope for future enhancement too as follows.

- It can be used to give a demo on UFO vs SPACECRAFT.
- It can give an overview of the surroundings to the pilots.
- It can also be used in arcade games.

BIBLIOGRAPHY

REFERENCE BOOKS

[1] Interactive Computer Graphics A Top-Down Approach with OpenGL – EdwardAngel, 5 th Edition, Addison-Wesley, 2008

[2] Xiang, Plastock: computer Graphics,sham's outline series, TMG.

[3] Computer graphics - OpenGL Version-Donald Hearn and Pauline Baker, 2nd Edition, Pearson Education, 2003.

WEBSITE LINKS

[1] https://learnopengl.com/Getting-started/OpenGL

[2] http://www.opengl-tutorial.org