VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgaum 590018, Karnataka

Department of Computer Science & Engineering

A Computer Graphics Mini Project Report on

"AIRPLANE SIMULATION"

In partial fulfilment of **COMPUTER GRAPHICS Laboratory (18CSL67)**

For the Academic Year **2021-2022**

SUBMITTED BY

MADHU SUDHAN N (1SK19CS022)

PAVAN KUMAR N (1SK19CS030)

UNDER THE GUIDANCE OF

Mrs. YAMUNA R

Assistant Professor

Department Of CSE

GOVT. S.K.S.J TECHNOLOGICAL INSTITUTE

(Affiliated to Visvesvaraya Technological University, Belgaum)
KR circle, Bangalore – 560001

GOVT. S.K.S.J TECHNOLOGICAL INSTITUTE

(Affiliated to Visvesvaraya Technological University, Belgaum)
KR circle, Bangalore – 560001

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE:

This is to certify that **MADHU SUDHAN N (1SK19CS022), PAVAN KUMAR N (1SK19CS030)** of **SIXTH Semester** students have successfully completed the mini project **"AIRPLANE SIMULATION"** in **COMPUTER GRAPHICS LABORATORY** as prescribed by the **VISVESVARAYA TECHNOLOGICAL UNIVERSITY** for the academic year **2021-2022.**

Signature of the Guide

Mrs. YAMUNA R
Asst. professor
Department of CSE

Signature of the HOD

Dr. PRADEEP KUMAR K
Assoc, Prof.& HOD of CSE

Name of the Examiners | **Signature with Date**

1) _____ 1) _____

2) _____ 2) _____

ACKNOWLEDGEMENT

A unique opportunity like this comes very rarely. It is indeed a pleasure for me to have worked on this project. The satisfaction that accompanies the successful completion of this project is incomplete without the mention of the people whose guidance and support are made it possible for me to complete this project.

We express my sincere gratitude to **Dr. K.G. CHANDRASHEKAR**, the principal Govt. SKSJTI for his valuable guidance, suggestion and consistent encouragement during the course of my mini project work and timely assistance for completion of the project.

We thank to entire faculty of the Computer Science and Engineering Department, especially **Dr. PRADEEP KUMAR K**, Head of the Department CSE, who has given me confidence to believe in myself and complete the project.

We grateful to **Mrs**. **YAMUNA R**, my internal guide, Computer Science and Engineering Department, for his support and encouragement throughout the process.

Last but not least I would also like to thank my parents and friends for their moral support.

ACKNOWLEDGEMENT

I express my sincere gratitude to Dr. K G CHANDRASHEKAR the principal Govt. SKSJTI for his valuable guidance, suggestion and consistent encouragement during the course of my mini project work and timely assistance for completion of the project. I thank to entire faculty of the Computer Science and Engineering Department, especially dr.Pradeep Kumar, Head of the Department CSE, who has given me confidence to believe in myself and complete the project. I'm grateful to Mrs.Yamuna Raju, B.E, M. TECH, my internal guide, Computer Science and Engineering Department, for his valuable guidance, suggestion and consistent encouragement during the course of my mini project Last but not least I would also like to thank my parents and friends for their moral support.

ABSTRACT

Our project is “AIRPLANE SIMULATION”. The game is created using OpenGL where the player tries to score. The objective of the game is to navigate through the score and levels , either user complete the game within a minute then he wins the game or he loses the game.

The aim of the mini project is to implement the path finding game.The Path finding game containing of any shape and size in which the controls are control by mouse buttons horizontal and vertical represent movement represent on clicking left button and releasing left button respectively.Path finding game can be made difficul to reasy,that is depend upon the score. Game can be implemented as in 2D, 3D or more higher dimensions. This game is very popular as in scoring. Hence it can be used as an effecting tool for logical reasoning and mental aptitude.

TABLE OF CONTENTS

1. INTRODUCTION

Graphics provides one of the most natural means of communicating with a computer ,since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and efficiently .Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surface and of data that have no inherent geometry ,such as survey results.

Using this editor you can draw and paint using the mouse. It can also perform a host of other functions like drawing lines ,circles ,polygons and so on .Interactive picture construction techniques such as basic positioning methods, rubber-band methods, dragging and drawing are used. Block operations like cut, copy and paste are supported to edit large areas of the workspace simultaneously. is user friendly and intuitivetouse.

OpenGL(open graphics library) is a standard specification defining a cross language crossplat form API for writing application that produce2D and 3Dcomputer graphics.Theinterface consists of over 250 different function calls which can be used to draw complex 3Dscenes from simple primitives. OpenGL was developed by silicon graphics Inc.(SGI) in 1992and is widely used in CAD ,virtual reality , scientific visualization , information visualization and flight simulation. It is also used in video games, where it competes with direct 3D on Microsoft Windows platforms. OpenGL is managed by the non-profit technology consortium, the group.

1.1 OpenGL serves two main purposes:

1.1.1 To hide the complexities of interfacing with different 3D accelerators, by presenting programmer with a single, uniform API

1.1.2 To hide the differing capabilities of hardware platforms, by requiring that all Implementations support the full openGL, feature set.

OpenGL has historically been influential on the development of 3D accelerator, promoting a base level of functionality that is now common in consumer level hardware:

- Rasterized points, lines and polygons are basic primitives.
- A transform and lighting pipeline.
- Z buffering.
- Texture Mapping.
- Alpha
- Blending..
- Support the library files

.

2. OBJECTIVES

The narrative mode (also known as the mode of narration) is the set of methods the author of a literary, theatrical, cinematic, or musical story uses to convey the plot to the audience. Narration, the process of presenting the narrative, occurs because of the narrative mode. It encompasses several overlapping areas of concern ,most importantly narrative point-of-view, which determines through whose perspective the story is viewed; narrative voice, which determines the manner through which the story is communicated to the author to be the same person. However, the narrator may be a fictive person devised by the author as a stand-alone entity, or even a character. The narrator is considered participant if an actual character in the story, and nonparticipant if only an implied character, or a sort of omniscient or semomniscie.
OpenGL(OpenGraphicsLibrary)isastandardspecificationdefiningacross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc.
OpenGL provides a powerful but primitive set of rendering command, and all higher-level drawing must be done in terms of these commands. There are several libraries that allow you to simplify your programming tasks, including the following:

OpenGLUtilityLibrary(GLU)contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections and rendering surfaces.

3. LITERATURE SURVEY

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube(CRT)screens so on after the introduction of computers.

Computer graphics today largely interactive, the user controls the contents, structure,and appearance of objects and of displayed images by using input devices, such as keyboard ,mouse, or touch-sensitive panel on the screen. Graphics based user interfaces allow millions of new users to control simple, low-cost application programs, such as spreadsheets, word processors, and drawing programs.

OpenGL(OpenGraphicsLibrary)isastandardspecificationdefiningacross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation. It is also used in videogames, where it competes with Direct3 Don Microsoft Windowsplatforms(seeDirect3Dvs.OpenGL).OpenGLismanagedbythenon-profittechnologyconsortium,theKhronosGroup.

In the 1980s, developing software that could function with a wide range of graphics hardware was a real challenge. By the early 1990s, Silicon Graphics (SGI) was a leader in 3Dgraphics for workstations. SGI's competitors (including Sun Microsystems, Hewlett-Packard and IBM) were also able. In addition, SGI had a large number of software customers; by changing to the OpenGL API they planned to keep their customers locked onto SGI (and IBM) hardware for a few years while market support for OpenGL matured to bring to market3D hardware, supported by extensions made to the PHIGS standard. In 1992, SGI led the creation of the OpenGL architectural review board (OpenGL ARB), the group of companies that would maintain and expand the

OpenGL specification took for years to come. On 17 December 1997, Microsoft and SGI initiated the Fahrenheit project, which was a joint effort with the goal of unifying the OpenGL and Direct3D interfaces (and adding a scene-graph API too). In 1998 Hewlett-Packard joined the project.[4] It initially showed some promise of bringing order to the world of interactive 3D computer graphics APIs, but on account of financial constraints at SGI,strategic reasons at Microsoft, and general lack of industry support, it was abandoned in1999[8].

Many opengl functions are used for rendering and transformation purposes.

Transformations functions like glRotate(), glTranslate(), glScaled() can be used.

OpenGL provides a powerful but primitive set of rendering command, and all higher-level drawing must be done in terms of these commands.There are several libraries that allow you to simplify your programming tasks, including the following:

OpenGLUtilityLibrary(GLU)containsseveralroutinesthatuselower-levelOpenGLcommandstoperformsuchtasksassettingupmatricesforspecificviewingorientationsandprojectionsandrenderingsurfaces.

OpenGL Utility Toolkit (GLUT) is a window-system-independent toolkit, written by Mark Kill guard ,to hide the complexities of differing window APIs.

To achieve the objective of the project, information related to the light sources Is required with OpenGL we can manipulate the lighting and objects in a scene to create many different kinds of effects. It explains how to control the lighting in a scene, discusses the OpenGL conceptual model of lighting, and describes in detail how to set the numerousilluminationparameterstoachievecertaineffects.Thisconceptisbeingobtainedfrom.

To demonstrate the transformation and lightening, effects, different polygons have tobeused.Polygonsaretypicallydrawnbyfillinginallthepixelsenclosedwithintheboundary, but we can also draw them as outlined polygons or simply as points at the vertices.This conceptis obtained from.

The properties of a light source like its material, diffuse, emissive, has to mention in the

project.So to design the light source and the objects, programming guide of an OpenGL used.

4. SYSTEMR EQUIREMENTS

4.1 HARDWARE REQUIREMENTS

Minimum hardware specification

- Microprocessor: **1.0GHz**andaboveCPU based o n Processor Intel® Core TM i5-64 bit Microprocessor
- Mainmemory:**512MBRAM**
- HardDisk :**40GB**
- HarddiskspeedinRPM:**5400RPM**
- Keyboard: **QWERTY** Keyboard
- Mouse:**2or3**Button mouse
- Monitor:**1024x768**display resolution

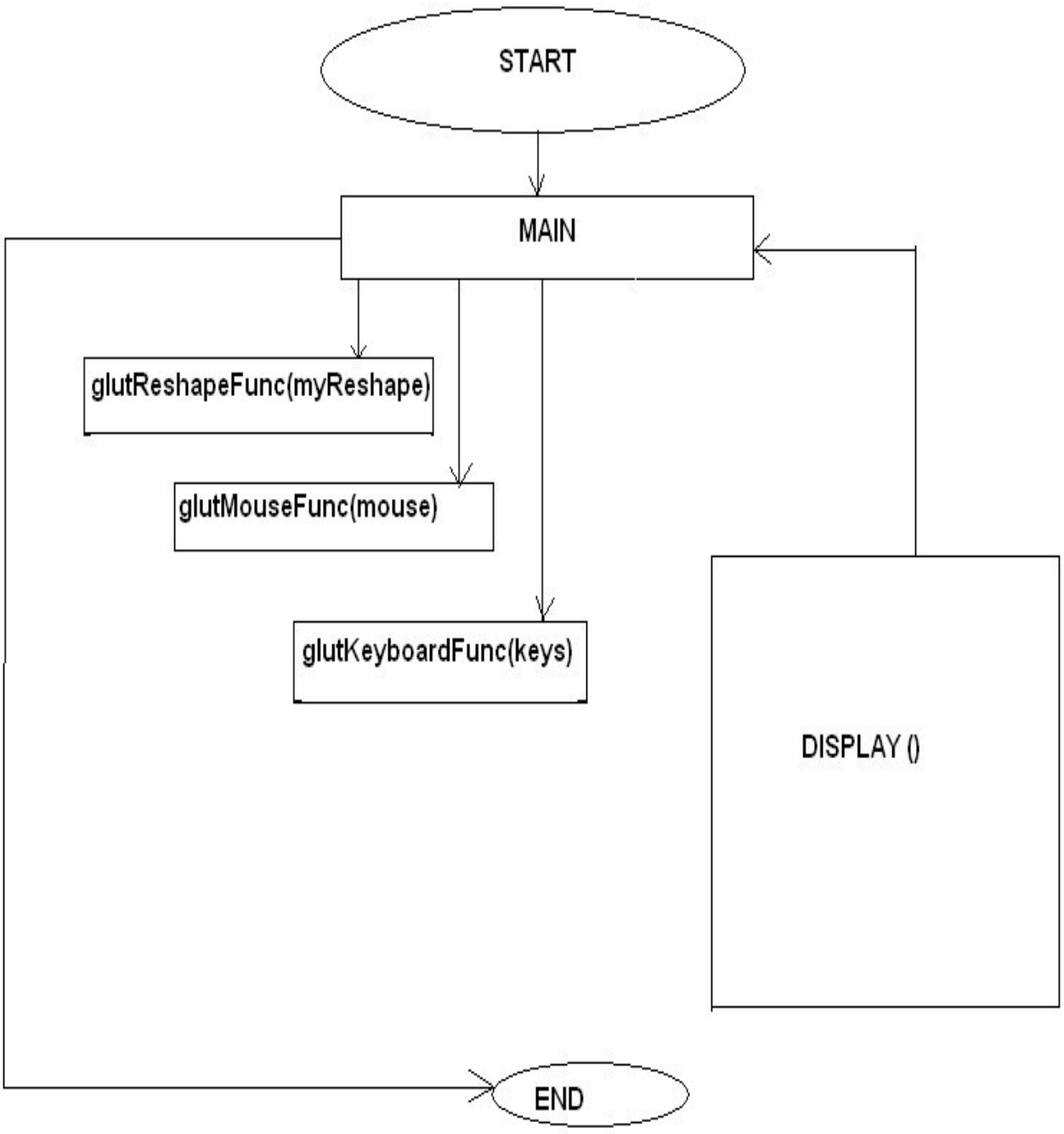
5 SOFTWARER EQUIREMENTS

Minimum software specification

Operatingsystem:WINDOWS.10

- ToolUsed:CODEBLOCKS
- OPENGL Library
- X86
- X64(WOW)
- Mouse Driver
- Graphics Driver
- CLanguage

6. SEQUENCE DIAGRAM



7. DESIGN

7.1 EXISTING SYSTEM

Existing system for a graphics is the TC++. This system will support only the 2D graphics. 2D graphics package being designed should be easy to use and understand. It should provide various options such as free hand drawing, line drawing, polygon drawing, filled polygons, flood fill, translation, rotation, scaling, clipping etc. Even though these properties were supported, it was difficult to render 2D graphics cannot be very difficult to get a 3 Dimensional object. Even the effects like lighting, shading cannot be provided. So we go for Microsoft Visual Studio software.

7.2 PROPOSED SYSTEM

To achieve three dimensional effects, open GL software is propos It is software which provides a graphical interface.

It is an interface between application program and graphics hardware.

The advantages are:

1. Open GL is designed as a streamlined.
2. It's a hardware independent interface i.e it can be implemented on many different hardware platforms.
3. With openGL we can draw a small set of geometric primitives such as points, lines and polygons etc.
4. It provides double buffering which is vital in providing transformations.
5. It is event driven software.
6. It provides call back function.

8. IMPLEMENTATION

8.1 Functions

The**glColor3f(float,float,float):-**This function will set the current drawing color

gluOrtho2D(GLdouble left, GL double right, GLdouble bottom, GLdouble top):- which defines a two dimensional viewing rectangle in the plane z=0.

glClear():- Takes a single argument that is the bitwise OR of several value indicating which buffer is to be cleared

glClearColor():-Specifies thered,green,blue,and alpha values used by **glClear** to clear the color buffers.

glMatrixMode(mode):-Sets the current matrixmode, mode can be GL_MODELVIEW,GL_PROJECTIONorGL_TEXTURE.

VoidglutInit(int*argc,charargv):-**InitializesGLUT,the arguments from main are passed in and can be used by the application.

Void glutInitDisplayMode (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.

Void glutInitWindowSize (int width, int height):- Specifies the initial position of the top-left corner the window in pixels

Int glutCreateWindow (char *title):-A window on the display. The string title can be usedto label the window. The return value provides references to the window that can be used when there are multiple windows.

VoidglutMouseFunc(void*f(intbutton,intstate,intx,inty):-Register the mouse callback function .The callback function returns the button ,the state of button after the event and the position of the mouse relative to the top-left corner of the window.

Void glutKeyboardFunc(void(*func) (void)):-This function is called every time when you press enter key to resume the game or when you press ‘b’ or ‘B’ key to go back to the initial screen or when you presses c key to exit from the application.

VoidglutDisplayFunc(void(*func)(void)):-Register the display functionfunc that is executed when the window needs to be redrawn.

VoidglutSpecialFunc(void(*func)(void)):-Thisfunctioniscalledwhenyoupressthespecial keys in the keyboard like arrow keys, function keys etc. In our program, the func isinvokedwhentheuparrowordownarrowkeyispressedforselectingtheoptionsinthemain menu and when the left or right arrow key is pressed for moving the object(car)accordingly.

glutPostReDisplay ( ):-which requests that the display callback be executed after the current callback returns.

Void MouseFunc (void (*func) void)):-This function is invoked when mouse keys are pressed. This function is used as an alternative to the previous function i.e., it is used to move the object(car)to right or left in our program by clicking left and right button respectively.

9. RESULTS & SNAPSHOTS

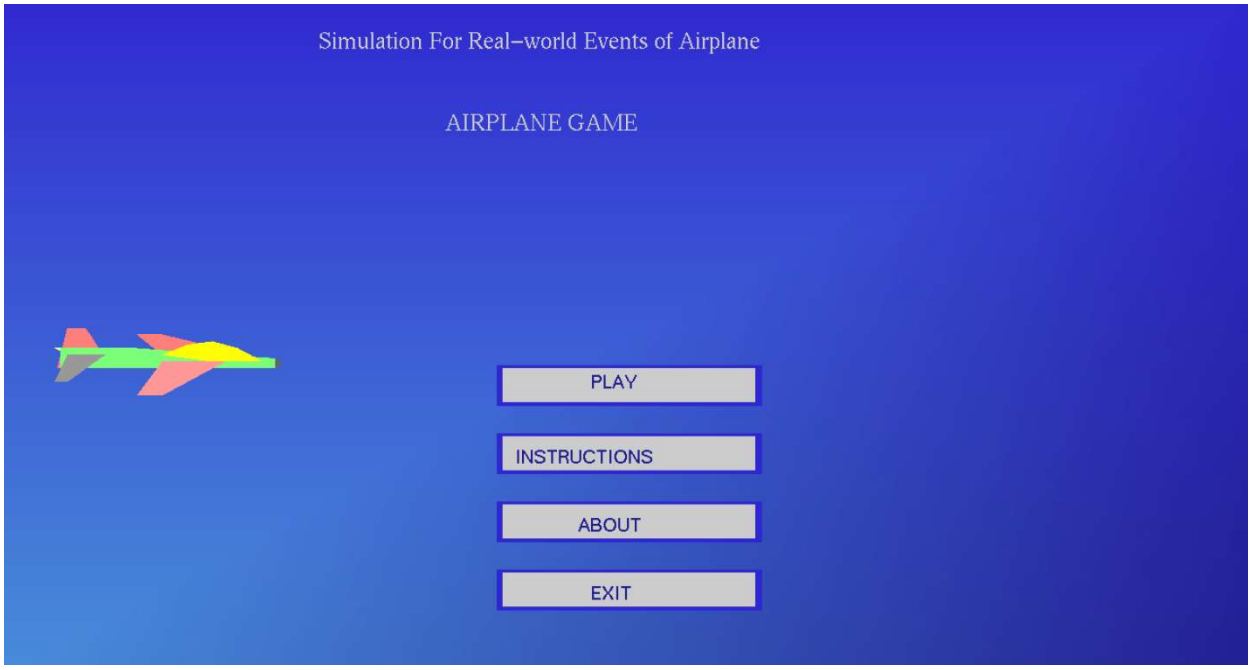


Figure6.1 after runing code we get menu

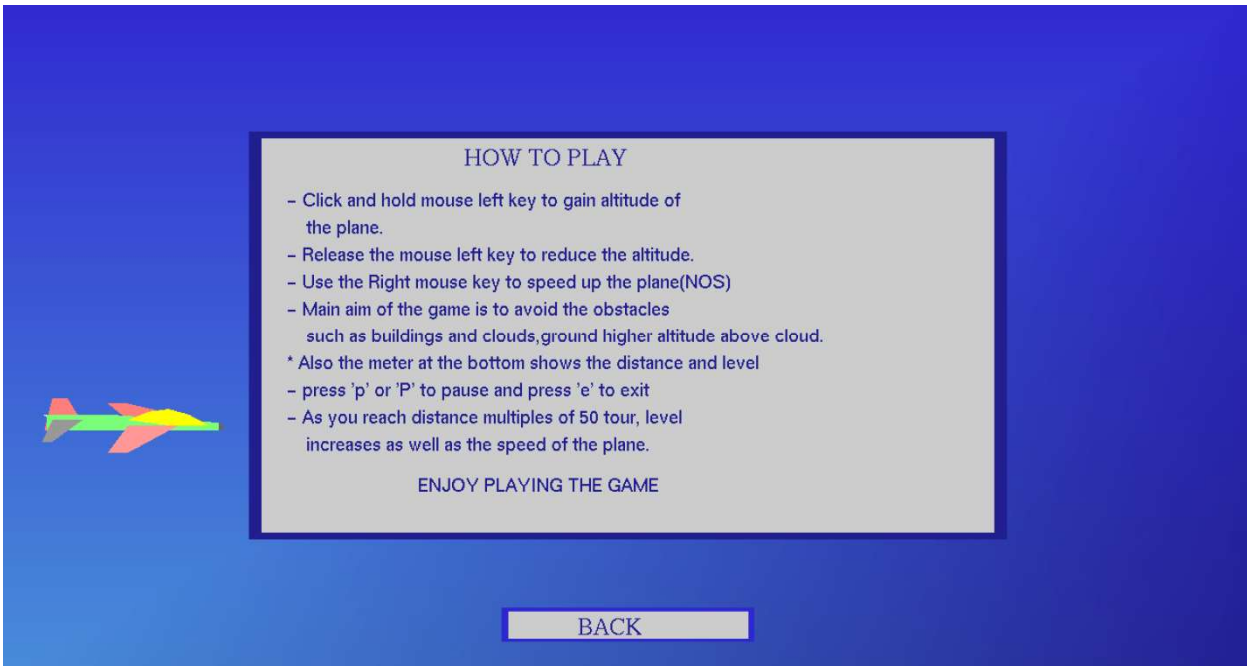


Figure6.2 On click **INSTRUCTIONS**

Figure6.3On click ABOUT

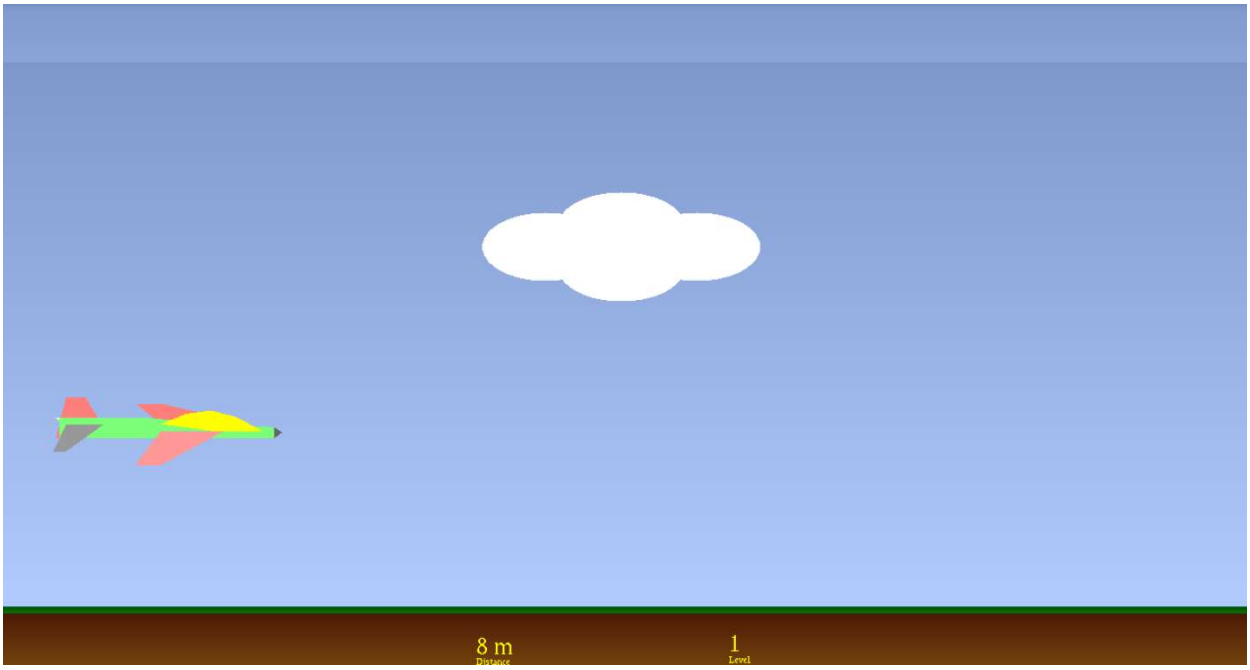


Figure6.4 On click PLAY

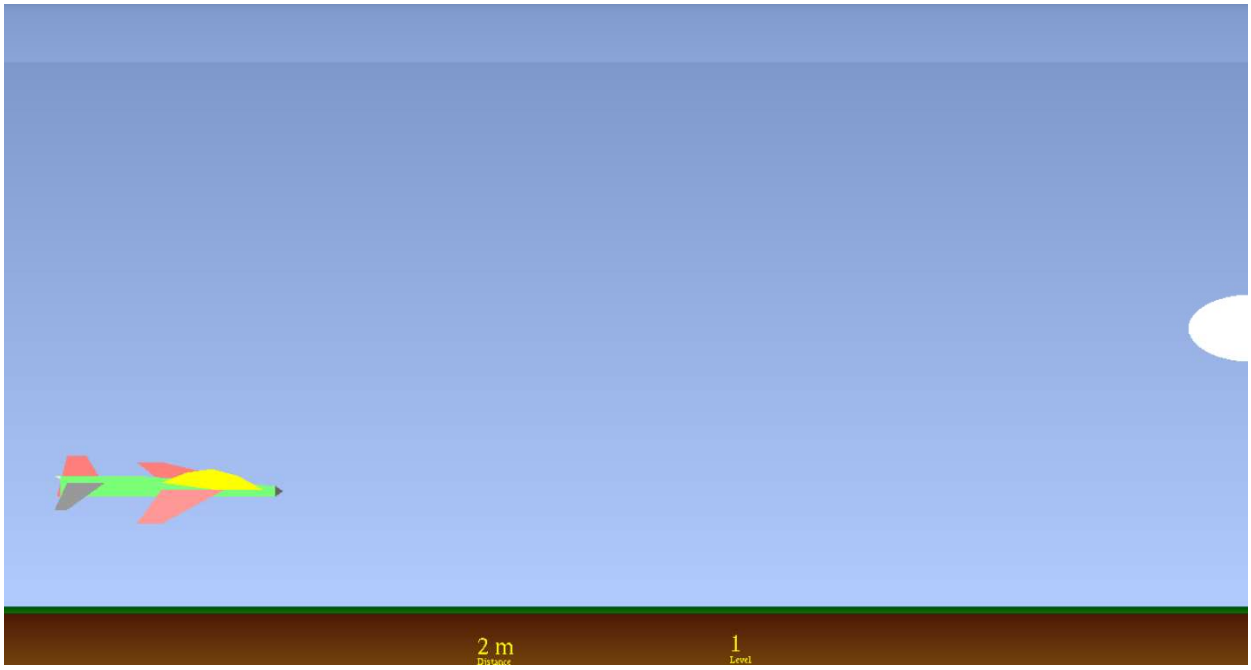


Figure6.5 plane moved with customized speed

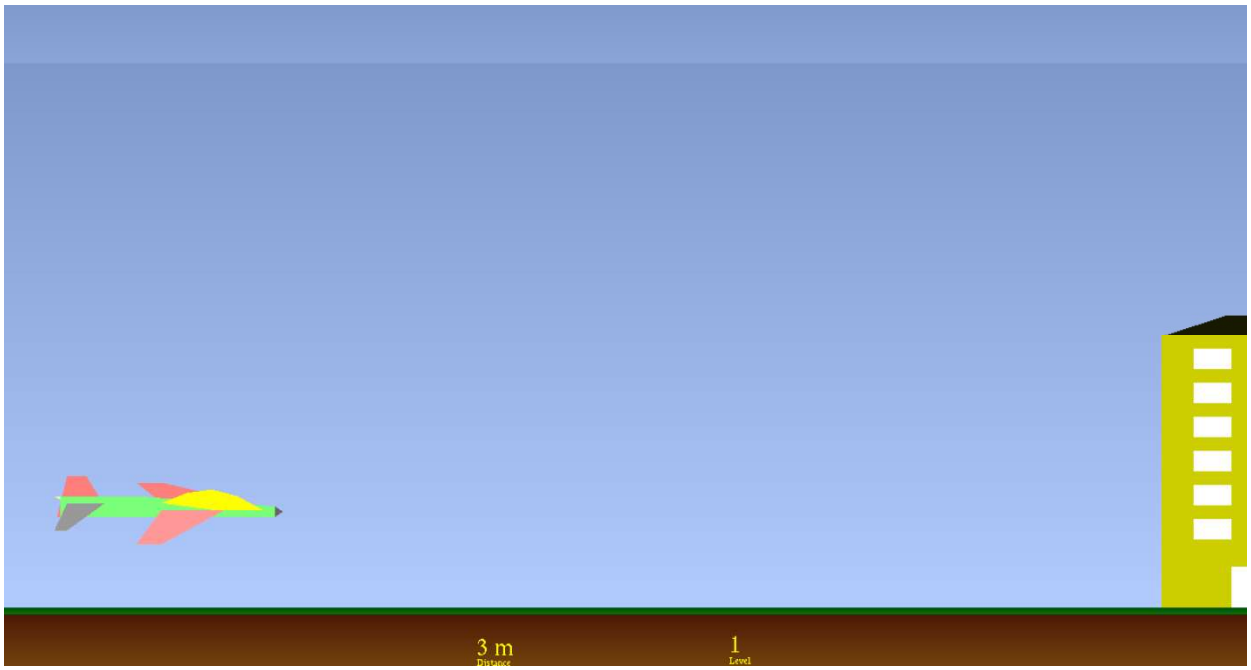


Figure6.6 moving towards building

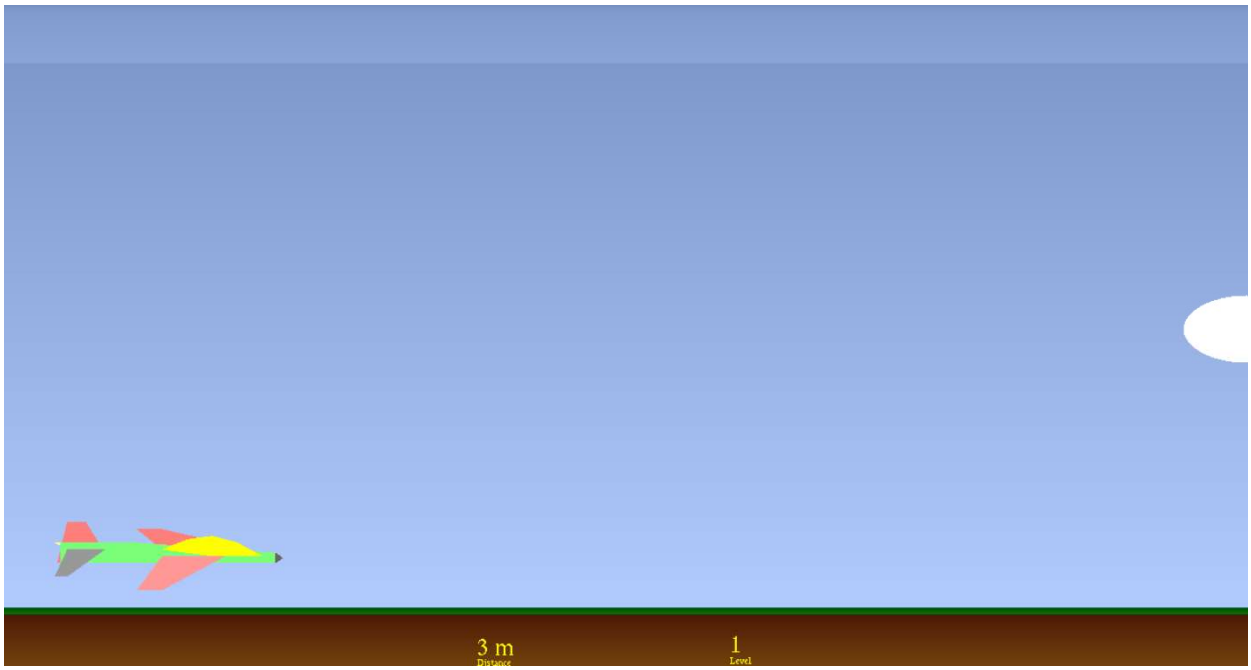


Figure6.7 Plane moving down

Figure6.8 End of game and you can Restart the gam

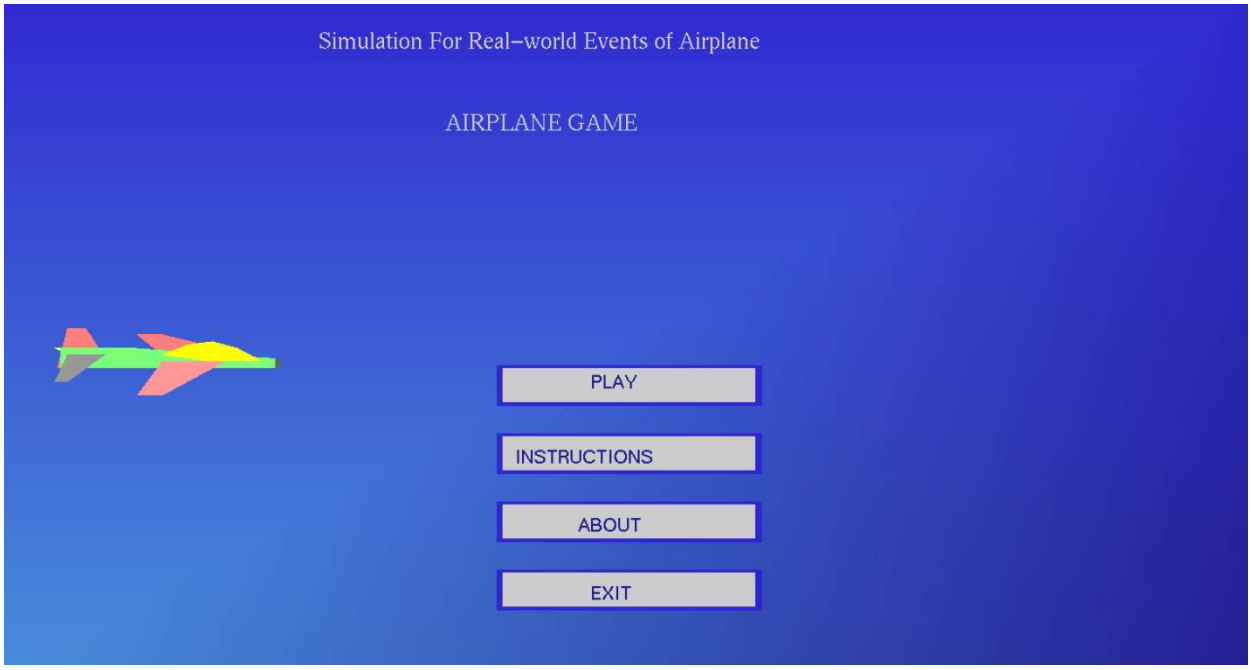


Figure6.6 Play again

10. SOURCE CODE

```
#include<windows.h>
#include<stdlib.h>
#include<gl/glut.h>
#include<time.h>
#include<stdio.h>
#include<math.h>

#define BLOCKSPEED 0.001
#define BOOSTER_MAX 50

int SCREENH=700,SCREENW=1300;
//---------------- Obstacles declaration----------
typedef struct building
{
	float block_x,block_y;
	bool state;
	int no_floors;
}building;
typedef struct Cloud
{
	float block_x,block_y;
	bool state;
}Cloud;
//-------------------declarations---------------
float bspd=BLOCKSPEED;  // block speed
bool pause=false,lflag = true,wflag =
true,gameEndStatus=false,instflag=false,abtflag=false,start=false;  //flags
float plane_mvmt=0.0;//jet movement up or down
float score=1.0;
char score_Str[20],slevel[10];   //score string and levelstring
int level=1,buildColor;     // initial level=1
building b;  // building struct
Cloud s;     // cloud struct
float booster=BOOSTER_MAX,boost=0;
//plane bounds
///-------------function prototypes------------------
void keyPressed(unsigned char,int,int);
void mouse(int button, int state, int x, int y);
void printString(float x,float y,float z,void *font,char *string);//what does this do??
void buildingBlock();
void CloudBlock();
```

```
void init();
void drawJet();
void gameEnd();
void drawBg();
void welcome();
void drawBuilding();
void drawCloud();
bool cloudHit();
bool buildingHit();
void printScore();
void display();
void moveJetU();
void moveJetD();

void buildingBlock()
{
	b.block_x=50.0;
	srand(time(0));
	b.no_floors = rand()%3+4;
	buildColor = rand()%3;
	b.block_y=b.no_floors*10 +15;   // generate block y cordinate depending on no of floors
	b.state=true;
	s.state=false;
}
void CloudBlock()
{
	s.block_x=50.0;
	srand(time(0));
	s.block_y=(rand()%30)+50;   //randomly generate block y cordinate
	s.state=true;
	b.state=false;
}

void semiCircle(float p1,float q1,float radius)
{
	float p,q;
	float angle;
	glBegin(GL_POINTS);

	for (angle=1.0f;angle<360.0f;angle++)
	{
		p = p1+sin(angle)*radius;
		q = q1+cos(angle)*radius;
		if(q>=100)
		glVertex2f(p,q);
	}
	glEnd();
```

```
}


void Circle(float x1,float y1,float radius)
{
	float x2,y2;
	float angle;
	glBegin(GL_POINTS);

	for (angle=1.0f;angle<360.0f;angle++)
	{
		x2 = x1+sin(angle)*radius;
		y2 = y1+cos(angle)*radius;
		glVertex2f(x2,y2);
	}
	glEnd();

}

void drawJet()
{
	//left tail wing

	glColor3f(1.0,1.0,0.6);
	glBegin(GL_POLYGON);
	glVertex2f(5.5,47.0);
	glVertex2f(8.5,47.0);
	glVertex2f(5.5,48.0);
	glVertex2f(4.5,48.0);
	glEnd();


	//left front wing

	glColor3f(1.0,0.5,0.5);
	glBegin(GL_POLYGON);
	glVertex2f(13.0,47.0);
	glVertex2f(20.0,47.0);
	glVertex2f(13.0,50.0);
	glVertex2f(11.0,50.0);
	glEnd();

	//tail
	glColor3f(1.0,0.5,0.5);
	glBegin(GL_POLYGON);
	glVertex2f(4.7,45.0);
	glVertex2f(5.5,51.0);
	glVertex2f(7.0,51.0);
```

```
glVertex2f(9.0,45.0);
glEnd();


//body
glColor3f(0.5,1.0,0.5);
glBegin(GL_POLYGON);
glVertex2f(5.0,48.0);
glVertex2f(11.0,48.0);
glVertex2f(22.0,46.5);
glVertex2f(22.0,45.0);
glVertex2f(5.0,45.0);
glEnd();


//right front wing
glColor3f(1.0,0.6,0.6);
glBegin(GL_POLYGON);
glVertex2f(13.0,46.0);
glVertex2f(18.0,46.0);
glVertex2f(13.0,41.0);
glVertex2f(11.0,41.0);
glEnd();


//dome
glColor3f(1.0,1.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(13.0,47.0);
glVertex2f(15.0,48.5);
glVertex2f(17.0,49.0);
glVertex2f(19.0,48.0);
glVertex2f(21.0,46.0);
glVertex2f(17.0,46.0);
glVertex2f(15.0,47.5);
glVertex2f(13.0,47.0);
glEnd();


//right tail wing
glColor3f(0.6,0.6,0.6);
glBegin(GL_POLYGON);
glVertex2f(5.5,47.0);
glVertex2f(8.5,47.0);
glVertex2f(5.5,43.0);
glVertex2f(4.5,43.0);
glEnd();
```

```
        // front tip
        glColor3f(0.4,0.4,0.4);
        glBegin(GL_POLYGON);
        glVertex2f(22.0,45.0);
        glVertex2f(22.3,45.375);
        glVertex2f(22.6,45.75);
        glVertex2f(22.3,46.125);
        glVertex2f(22.0,46.5);
        glEnd();
}


void drawString(float x,float y,float z,void *font,char *string)
{
        char *c;
        glRasterPos3f(x, y,z);

        for (c=string; *c != '\0'; c++)
        {
                glutBitmapCharacter(font, *c);
        }
}

void gameEnd()
{
        gameEndStatus = true;
        glColor3f(0.3,0.56,0.84);   //game end background screen
        glBegin(GL_POLYGON);
        glVertex3f(0.0,0.0,0.0);
        glColor3f(0.137,0.137,0.556);
        glVertex3f(100.0,0.0,0.0);
        glColor3f(0.196,0.196,0.8);
        glVertex3f(100.0,100.0,0.0);
        glVertex3f(0.0,100.0,0.0);
        glEnd();
        glPushMatrix();
        glScalef(0.8,0.8,0);
        drawJet();
        glPopMatrix();

        glColor3f(0.196,0.196,0.8);  // disp box
        glRectf(20.0,20.0,80.0,80.0);
        glColor3f(0.8,0.8,0.8);
        glRectf(21.0,21.0,79.0,79.0);

        glColor3f(0.196,0.196,0.8);   //restart button
        glRectf(40,5,60,10);
```

```
        glColor3f(0.8,0.8,0.8);
        glRectf(40.5,5.5,59.5,9.5);
        glColor3f(0.137,0.137,0.556);

        drawString(43,6,0,GLUT_BITMAP_TIMES_ROMAN_24,"RESTART");
        drawString(41,71,0,GLUT_BITMAP_TIMES_ROMAN_24,"GAME OVER!!!");
        drawString(23,61,0,GLUT_BITMAP_HELVETICA_18,"DISTANCE :");
        drawString(40,61,0,GLUT_BITMAP_TIMES_ROMAN_24,score_Str);
                printf("m\n");
        printf("\n");
        drawString(23,56,0,GLUT_BITMAP_HELVETICA_18,"LEVEL        :");
        drawString(40,56,0,GLUT_BITMAP_TIMES_ROMAN_24,slevel);



        drawString(33,30,0,GLUT_BITMAP_HELVETICA_18," ENJOY PLAYING THE
GAME");

        glutPostRedisplay();

}

void drawBg()
{
        glPushMatrix();

        glColor3f(0.0,0.48,0.047);              // green floor

        glBegin(GL_POLYGON);
        glVertex3f(0.0,9.0,0.0);
        glVertex3f(100.0,9.0,0.0);
        glColor3f(0.0,0.3,0.03);
        glVertex3f(100.0,10.0,0.0);
        glVertex3f(0.0,10.0,0.0);
        glVertex3f(0.0,9.0,0.0);
        glEnd();

        glColor3f(0.474,0.298,0.074);  // brown ground
        glBegin(GL_POLYGON);
        glVertex3f(0.0,0.0,0.0);
        glVertex3f(100.0,0.0,0.0);
        glColor3f(0.3,0.1,0.03);
        glVertex3f(100.0,9.0,0.0);
        glVertex3f(0.0,9.0,0.0);
        glEnd();

        glColor3f(0.5,0.6,0.79);
        glBegin(GL_POLYGON);                            //ceiling
```

```
        glVertex3f(0.0,100.0,0.0);
        glVertex3f(100.0,100.0,0.0);
        glColor3f(0.6,0.7,0.89);
        glVertex3f(100.0,80.0,0.0);
        glVertex3f(0.0,80.0,0.0);
        glEnd();

        glColor3f(0.5,0.6,0.79);// sky blue
        glBegin(GL_POLYGON);   //background screen
        glVertex3f(0.0,90.0,5.0);
        glVertex3f(100.0,90.0,5.0);
        glColor3f(0.7,0.8,0.99);//sky
        glVertex3f(100.0,10.0,5.0);
        glVertex3f(0.0,10.0,5.0);
        glEnd();

        glPopMatrix();
}

void welcome()
{
        glColor3f(0.3,0.56,0.84);   //welcome background
        glBegin(GL_POLYGON);
        glVertex3f(0.0,0.0,0.0);
        glColor3f(0.137,0.137,0.556);
        glVertex3f(100.0,0.0,0.0);
        glColor3f(0.196,0.196,0.8);
        glVertex3f(100.0,100.0,0.0);
        glVertex3f(0.0,100.0,0.0);
        glEnd();
        drawJet();

        // button 1 .. PLAY
        glColor3f(0.196,0.196,0.8);
        glRectf(39.5,39.5,60.5,45.5);

        glColor3f(0.8,0.8,0.8);
        glRectf(40,40,60,45);
        glColor3f(0.137,0.137,0.556);
        drawString(47,42,0,GLUT_BITMAP_HELVETICA_18,"PLAY");

        // button 2 .. instructions
        glColor3f(0.196,0.196,0.8);
        glRectf(39.5,29.5,60.5,35.5);

        glColor3f(0.8,0.8,0.8);
        glRectf(40,30,60,35);
        glColor3f(0.137,0.137,0.556);
```

```
        drawString(41,31,0,GLUT_BITMAP_HELVETICA_18,"INSTRUCTIONS");

        // button 3 .. ABOUT
        glColor3f(0.196,0.196,0.8);
        glRectf(39.5,19.5,60.5,25.5);

        glColor3f(0.8,0.8,0.8);
        glRectf(40,20,60,25);
        glColor3f(0.137,0.137,0.556);
        drawString(46,21,0,GLUT_BITMAP_HELVETICA_18,"ABOUT");

        // button 4 .. exit
        glColor3f(0.196,0.196,0.8);
        glRectf(39.5,9.5,60.5,15.5);

        glColor3f(0.8,0.8,0.8);
        glRectf(40,10,60,15);
        glColor3f(0.137,0.137,0.556);
        drawString(47,11,0,GLUT_BITMAP_HELVETICA_18,"EXIT");


        glPushMatrix();

        glColor3f(0.8,0.8,0.8);
        drawString(25.5,92,0,GLUT_BITMAP_TIMES_ROMAN_24,"Simulation For Real-
world Events of Airplane");
        drawString(35.5,80,0,GLUT_BITMAP_TIMES_ROMAN_24,"AIRPLANE GAME");
        glPopMatrix();
        glColor3f(0.137,0.137,0.556);

}

void drawBuilding()
{
        glPushMatrix();                                         // 3D part
        if(buildColor==0)
                glColor3f(0.1,0.0,0.0);
        else if (buildColor ==1)
                glColor3f(0.1,0.1,0.0);
        else
                glColor3f(0.0,0.1,0.1);

        glTranslatef(b.block_x,b.no_floors*10.0+10,0.0);
        glBegin(GL_POLYGON);
        glVertex3f(0.0,0.0,0.0);
        glVertex3f(5.0,3.0,0.0);
        glVertex3f(20.0,3.0,0.0);
        glVertex3f(20.0,-b.no_floors*10.0,0.0);
```

```
glVertex3f(0.0,-b.no_floors*10.0,0.0);
glEnd();
glPopMatrix();

for(int i=1;i<=b.no_floors;i++)
{
        glPushMatrix();

        if(buildColor==0)
                glColor3f(0.8,0.0,0.0);
        else if (buildColor ==1)
                glColor3f(0.8,0.8,0.0);
        else
                glColor3f(0.0,0.8,0.8);

        glTranslatef(b.block_x,10.0*i,0.0);   //base
        glBegin(GL_POLYGON);
        glVertex3f(0.0,0.0,0.0);
        glVertex3f(15.0,0.0,0.0);
        glVertex3f(15.0,10.0,0.0);
        glVertex3f(0.0,10.0,0.0);
        glEnd();
        glColor3f(1.0,1.0,1.0);     // left window
        glBegin(GL_POLYGON);
        glVertex3f(2.5,5.0,0.0);
        glVertex3f(5.5,5.0,0.0);
        glVertex3f(5.5,8.0,0.0);
        glVertex3f(2.5,8.0,0.0);
        glEnd();
        glColor3f(1.0,1.0,1.0);     // left window
        glBegin(GL_POLYGON);
        glVertex3f(2.5,0.0,0.0);
        glVertex3f(5.5,0.0,0.0);
        glVertex3f(5.5,3.0,0.0);
        glVertex3f(2.5,3.0,0.0);
        glEnd();
        glColor3f(1.0,1.0,1.0);     // right window
        glBegin(GL_POLYGON);
        glVertex3f(12.5,5.0,0.0);
        glVertex3f(9.5,5.0,0.0);
        glVertex3f(9.5,8.0,0.0);
        glVertex3f(12.5,8.0,0.0);
        glEnd();
        glColor3f(1.0,1.0,1.0);     // right window
        glBegin(GL_POLYGON);
        glVertex3f(12.5,.0,0.0);
        glVertex3f(9.5,0.0,0.0);
        glVertex3f(9.5,3.0,0.0);
```

```
                glVertex3f(12.5,3.0,0.0);
                glEnd();
                glPopMatrix();
        }
        glPushMatrix();

        if(buildColor==0)
                glColor3f(0.8,0.0,0.0);
        else if (buildColor ==1)
                glColor3f(0.8,0.8,0.0);
        else
                glColor3f(0.0,0.8,0.8);

        glTranslatef(b.block_x,10.0,0.0);   //base
        glBegin(GL_POLYGON);
        glVertex3f(0.0,0.0,0.0);
        glVertex3f(15.0,0.0,0.0);
        glVertex3f(15.0,10.0,0.0);
        glVertex3f(0.0,10.0,0.0);
        glEnd();
        glColor3f(1.0,1.0,1.0);     // door
        glBegin(GL_POLYGON);
        glVertex3f(5.5,0.0,0.0);
        glVertex3f(9.5,0.0,0.0);
        glVertex3f(9.5,6.0,0.0);
        glVertex3f(5.5,6.0,0.0);
        glEnd();
        glPopMatrix();
}

void drawCloud()
{
        glColor3f(1.0,1.0,1.0);
        glTranslatef(s.block_x,s.block_y,0.0);
        glutSolidSphere(5,100,10);
        glTranslatef(6,-3.0,0.0);
        glutSolidSphere(5,100,10);
        glTranslatef(0,6.0,0.0);
        glutSolidSphere(5,100,10);
        glTranslatef(6,-3.0,0.0);
        glutSolidSphere(5,100,10);

}

bool cloudHit()
{
        if(s.block_x < 13 && s.block_x> -5)
                if(plane_mvmt-3+50 >s.block_y-3 && plane_mvmt-3+50 < s.block_y+3)   //
```

```
plane front to cloud mid box1
                        return true;

        if(s.block_x < 12 && s.block_x> -4)
                if(plane_mvmt-3+50 >s.block_y-5 && plane_mvmt-3+50 < s.block_y+5)   //
plane front to cloud mib box2
                        return true;

        if(s.block_x < 10 && s.block_x> -1)
                if(plane_mvmt-3+50 >s.block_y-6 && plane_mvmt-3+50 < s.block_y-2)
                        return true;

        //for top wing and bottom wing
        if(s.block_x < 5 && s.block_x> -3)
                if(plane_mvmt-3+50 >s.block_y-11 && plane_mvmt-3+50 < s.block_y+13)
                        return true;

        return false;
}


bool buildingHit()
{
        if (((int)b.block_x<=8 &&(int)b.block_x>=-7 && ((int)plane_mvmt+50)-
b.block_y<=3))   //buildin back  body to tail
                return true;
        else if (((int)b.block_x<=10 &&(int)b.block_x>=-5 && ((int)plane_mvmt+50)-
b.block_y<=0))   //body to tail
                return true;
        else if(((int)b.block_x<=6 &&(int)b.block_x>=-3 && ((int)plane_mvmt+47)-
b.block_y<=0))  //right wing
                return true;
        else if(((int)b.block_x<=4 &&(int)b.block_x>=-4 && ((int)plane_mvmt+47)-
b.block_y<=3))  //   building back right wing
                return true;
        else
                return false;
}

bool boundHit()
{
        if(plane_mvmt+50>=100||plane_mvmt+50 <=18)   // top and bottom boundary
                return true;
        else
                return false;
}
void printScore()
{
```

```
        glColor3f(1.0,1.0,0.0);//score

        sprintf(slevel, "%d", (int)level );
        drawString(58,1.8,0,GLUT_BITMAP_TIMES_ROMAN_10,"Level");
        drawString(58,3.5,0,GLUT_BITMAP_TIMES_ROMAN_24,slevel);

        if(booster >0 && boost)
                score+=0.04;//SCORE with booster
        else
                score+=0.005;//SCORE without booster

        drawString(38,1.5,0,GLUT_BITMAP_TIMES_ROMAN_10,"Distance");
        sprintf(score_Str, "%d m",(int)score );
        drawString(38,3,0,GLUT_BITMAP_TIMES_ROMAN_24, score_Str);


}



void display()
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        //GameOver Checking
        if(gameEndStatus == true)
        {
                gameEnd();
        }
        else if(wflag==true)//Welcome Screen
        {
                welcome();
        }
        else if (instflag == true)
        {
                glColor3f(0.3,0.56,0.84);   // background
                glBegin(GL_POLYGON);
                glVertex3f(0.0,0.0,0.0);
                glColor3f(0.137,0.137,0.556);
                glVertex3f(100.0,0.0,0.0);
                glColor3f(0.196,0.196,0.8);
                glVertex3f(100.0,100.0,0.0);
                glVertex3f(0.0,100.0,0.0);
                glEnd();
                glPushMatrix();
                glScalef(0.8,0.8,0);
                drawJet();
                glPopMatrix();
                glColor3f(0.137,0.137,0.556);
```

```
                glRectf(20.0,20.0,80.0,80.0);
                glColor3f(0.8,0.8,0.8);
                glRectf(21.0,21.0,79.0,79.0);


                glColor3f(0.196,0.196,0.8);
                glRectf(40,5,60,10);
                glColor3f(0.8,0.8,0.8);
                glRectf(40.5,5.5,59.5,9.5);

                glColor3f(0.137,0.137,0.556);
                drawString(46,6,0,GLUT_BITMAP_TIMES_ROMAN_24,"BACK");

                glColor3f(0.137,0.137,0.556);
                drawString(37,75,0,GLUT_BITMAP_TIMES_ROMAN_24,"HOW TO PLAY");
                drawString(23,69,0,GLUT_BITMAP_HELVETICA_18,"- Click and hold mouse
left key to gain altitude of ");
                drawString(23,65,0,GLUT_BITMAP_HELVETICA_18,"    the plane.");
                drawString(23,61,0,GLUT_BITMAP_HELVETICA_18,"- Release the mouse
left key to reduce the altitude.");
                drawString(23,57,0,GLUT_BITMAP_HELVETICA_18,"- Use the Right mouse
key to speed up the plane(NOS)");
                drawString(23,53,0,GLUT_BITMAP_HELVETICA_18,"- Main aim of the game
is to avoid the obstacles ");
                drawString(23,49,0,GLUT_BITMAP_HELVETICA_18,"    such as buildings and
clouds,ground higher altitude above cloud.");
                drawString(23,45,0,GLUT_BITMAP_HELVETICA_18,"* Also the meter at the
bottom shows the distance and level ");
                drawString(23,41,0,GLUT_BITMAP_HELVETICA_18,"- press 'p' or 'P' to pause
and press 'e' to exit");
                drawString(23,37,0,GLUT_BITMAP_HELVETICA_18,"- As you reach distance
multiples of 50 tour, level ");
                drawString(23,33,0,GLUT_BITMAP_HELVETICA_18,"    increases as well as
the speed of the plane.");
                drawString(33,27,0,GLUT_BITMAP_HELVETICA_18," ENJOY PLAYING
THE GAME");

                glutPostRedisplay();

        }
        else if (abtflag == true)
        {
                glColor3f(0.3,0.56,0.84);   // background
                glBegin(GL_POLYGON);
                glVertex3f(0.0,0.0,0.0);
                glColor3f(0.137,0.137,0.556);
                glVertex3f(100.0,0.0,0.0);
                glColor3f(0.196,0.196,0.8);
```

```
                glVertex3f(100.0,100.0,0.0);
                glVertex3f(0.0,100.0,0.0);
                glEnd();
                glPushMatrix();
                glScalef(0.8,0.8,0);
                drawJet();
                glPopMatrix();
                glColor3f(0.137,0.137,0.556);
                glRectf(20.0,20.0,80.0,80.0);
                glColor3f(0.8,0.8,0.8);
                glRectf(21.0,21.0,79.0,79.0);


                glColor3f(0.196,0.196,0.8);
                glRectf(40,5,60,10);
                glColor3f(0.8,0.8,0.8);
                glRectf(40.5,5.5,59.5,9.5);
                glColor3f(0.137,0.137,0.556);
                drawString(46,6,0,GLUT_BITMAP_TIMES_ROMAN_24,"BACK");

                glColor3f(0.137,0.137,0.556);
                drawString(44,75,0,GLUT_BITMAP_TIMES_ROMAN_24,"Developer info");
                drawString(21,61,0,GLUT_BITMAP_HELVETICA_18,"          introducing by:
");
                drawString(23,53,0,GLUT_BITMAP_HELVETICA_18,"          Madhu sudhan
N and Pavan kumar N");

                drawString(33,40,0,GLUT_BITMAP_HELVETICA_18," ENJOY PLAYING
THE GAME");
                glutPostRedisplay();

        }
        else if( pause == true)
        {
                drawBg();
                glPushMatrix();
                glScalef(0.8,0.8,0);
                drawJet();
                glPopMatrix(); glPushMatrix();
                glColor3f(0.196,0.196,0.8);
                glRectf(35.0,40.0,65.0,60.0);
                glColor3f(0.8,0.8,0.8);
                glRectf(36.0,41.0,64.0,59.0);
                glPopMatrix();
                glColor3f(0.137,0.137,0.556);
           //drawString(40,55,0,GLUT_BITMAP_HELVETICA_18," GAME PAUSED");
                //drawString(37,45,0,GLUT_BITMAP_HELVETICA_18," PRESS 'P' to
continue");
```

```
                glutPostRedisplay();

        }
        else if((b.state == true && buildingHit() == true)|| boundHit()== true)
        {
                gameEndStatus = true;
                gameEnd();
        }
        else if(s.state == true && cloudHit() == true)
        {

                gameEndStatus = true;
                gameEnd();
        }
        else
        {

                if((int)score%50==0 && lflag==true)// l-level
                {
                        lflag=false;
                        level++;
                        bspd+=0.02;
                }
                else if((int)score%50!=0 && lflag==false)
                {
                        lflag=true;
                }

                glPushMatrix();
                drawBg();


                glPushMatrix();
                glTranslatef(0.0,plane_mvmt,0.0);
                drawJet();                      //code for jet
                glPopMatrix();


                if( booster <= BOOSTER_MAX && !boost)    // booster charging
                        booster+=0.005;



                if( (b.state == true && b.block_x < -10) || (s.state == true && s.block_x < -10))
  //for new building //building has gone outside the screen- state=true
                {
                        srand(time(NULL));
                        int random = rand()%2;//for random building or cloud
```

```
                        if( random == 0)
                        {
                                buildingBlock();
                        }
                        else
                        {
                                CloudBlock();
                        }
                }

                else if(b.state == true)
                {
                        if(booster >0 && boost)
                        {
                                b.block_x-=bspd+boost;
                                booster = booster-0.02;//reduce to normal speed after leaving
boost key
                        }
                        else
                                b.block_x-=bspd;
                }
                else if( s.state == true)
                {
                        if(booster >0 && boost)
                        {
                                s.block_x-=bspd+boost;
                                booster = booster-0.02;

                        }
                        else
                        {
                                s.block_x-=bspd;
                        }
                }
                if(b.state == true)
                {
                        glTranslatef(b.block_x,0.0,0.0);
                        drawBuilding();
                }
                else if( s.state == true)
                {
                        glTranslatef(s.block_x,0.0,0.0);
                        drawCloud();
                }
                glPopMatrix();

                printScore();
        }
```

```
	glFlush();
	glutSwapBuffers();
}


void moveJetU()	// jet moving up
{
	if(start == false)
		glutPostRedisplay();
	else if(pause == false)
	{
		//alti_ang-=0.15;

		plane_mvmt+=0.5;
		glutPostRedisplay();
	}
}


void moveJetD()		// jet moving down
{
	if(start == false)
		glutPostRedisplay();
	else if(pause == false )
	{
		//alti_ang+=0.15;
		plane_mvmt-=0.05;
		glutPostRedisplay();
	}
}



void mouse(int button, int state, int x, int y)		// takes input from mouse
{
	int mx=x*100/SCREENW,my=(SCREENH-y)*100/SCREENH;		// m = mouse
cordinate to graphics

	/*		mouse calculation//converting to screen coordinates-ortho values
	SCREENSIZE  ----> ORTHO
	x(reqd val) ----> ???
	*/
	if(instflag || abtflag || gameEndStatus)
	{
		if(mx>40 && mx<60)
		{
			if(my>5 && my<10)
			{
```

```
                        wflag = true;
                        if(instflag)
                                instflag = false;
                        else if (abtflag)
                                abtflag = false;
                        if(gameEndStatus)
                        {
                                wflag = true;
                                gameEndStatus = false;
                                plane_mvmt = 0;
                                start = false;
                                init();
                                bspd = BLOCKSPEED;//restarting the game
                                booster=BOOSTER_MAX;
                                score=1;
                                level=1;
                                glutPostRedisplay();
                        }

                }
        }
}
if(wflag == true)
{
        if(mx>40 && mx<60)
        {
                if(my>40 && my<45)
                {
                        start = true;
                        wflag=false;
                }
                else if(my>30 && my<35)
                {
                        instflag = true;
                        wflag = false;
                }
                else if(my>20 && my<25)
                {
                        abtflag = true;
                        wflag = false;
                }
                else if(my>10 && my<15)
                {
                        exit(0);
                }

        }
}
```

```
	else
	{

		if(button == GLUT_LEFT_BUTTON)
		{
			if (state == GLUT_DOWN )
				glutIdleFunc(moveJetU);

			else if (state == GLUT_UP )
				glutIdleFunc(moveJetD);
		}
		if(button == GLUT_RIGHT_BUTTON)
		{

			if(state == GLUT_DOWN)
			{
				if(booster>0)
				{
					boost = 1;
				}
			}
			if(state == GLUT_UP)
			{
				boost = 0;
			}
		}
	}
}
void keyPressed(unsigned char key,int x, int y) // int x and y are mouse pos at time of press
{
	if(key == 'e')
	{
		exit(0);
	}
	else if(key == 'p' || key == 'P')
	{
		if(pause == true)
			pause = false;

		else
			pause = true;
	}
	glutPostRedisplay();
}
void myReshape(int w, int h)
{
	SCREENH=h,SCREENW=w;
	printf("width = %d\theight= %d",w,h);
```

```
        glViewport(0,0,w,h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(0.0, 100.0, 0.0, 100.0,          -5.0 , 10.0);
        glMatrixMode(GL_MODELVIEW);
}

void init()
{
        srand(time(0));      int random = rand()%2;
        if( random == 0)
        {
                buildingBlock();
        }
        else
        {
                CloudBlock();
        }
}

int main(int argc, char** argv)
{
       printf("\nHow To Play");
                printf("Click and hold mouse left key to gain altitude of the plane\n");
                printf("Release the mouse left key to reduce the altitude\n");
                printf("Use the Right mouse key to speed up the plane(NOS)\n");
                printf("\n");
                printf("-->The main aim of the game is to avoid the obstacles: buildings and
clouds\n");
                printf("-->Also the meter at the bottom shows the distance travelled and
LEVEL.\n");
                printf("\n");
                printf("-->As you reach distance multiples of 50 tour level increases as well as
the speed of the plane.\n");
                printf("==>ENJOY PLAYING THE GAME\n");
        glutInit(&argc, argv);
        glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        glutInitWindowSize (SCREENW,SCREENH);
        glutInitWindowPosition (50,50);
        glutCreateWindow ("Airplane Simulation Game");
        init();  glutDisplayFunc(display);
        glutReshapeFunc(myReshape);
        glutMouseFunc(mouse);
        glutKeyboardFunc(keyPressed);
        glutMainLoop();
        return 0;
}
```

11. CONCLUSION

Using simple flight dynamics, restricted terrain, and objects, this research successfully created a virtual reality flight sim- ulator to imitate aeroplane flight. It provides a breathtaking view of flying in mid-air. Because the actual aeroplane flight dynamics were not incorporated into the created programme, the virtual reality flight simulator was not a complete and powerful flight simulator.

To put it another way, a virtual reality flight simulator, which is a relatively new technology for virtual environments, allows users to interact with the computer-generated world. A downside of this virtual reality flight simulator is the multiple motions required for a typical flight (joystick, pedal, button, etc.); it still lacks a better facility to deliver an experience that is comparable to that of a genuine flight.

To deliver a better experience, it should be a semi-portable device. The next stage will contain the plane's complicated flight dynamics, as well as richer terrains and more realistic plane objects.

FUTURE ENHANCEMENTS

These are the features that are planned to be supported in the future

* Support for multiple canvases
* Support for pattern filling SS
* Support for 3d transformations
* Support for

transparency of layers