ABSTRACT

The "**Airshow Simulation Project**" is a computer graphics and visualization project that aims to simulate an air show environment using various techniques and algorithms. The project involves creating a 3D virtual environment that includes a runway, aircraft, and various other objects. The aircrafts are modeled using 3D modeling software and are animated using keyframe animation techniques.

The environment is rendered using lighting and shading techniques to create a realistic look and feel. The goal of the project is to provide an immersive and interactive experience for the user, allowing them to feel as if they are part of an air show.

The project can be used for entertainment purposes, as well as for training and educational purposes. The project showcases the use of various computer graphics techniques and algorithms to create a realistic and interactive simulation of an air show. The project features various aircraft performing aerial stunts.

The main objective of the project is to create a realistic and immersive experience for the audience, while also demonstrating the advanced capabilities of computer graphics technology. The project is designed to be visually stunning and engaging and to provide a unique and memorable experience for viewers.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

The main theme of this project is to display the Aeroplane flying by emitting different coloured smoke successfully with the help computer graphics. The graphics package is based on the OpenGL library functions. The programming language used here is C using OpenGL libraries.

The project AIRSHOW SIMULATION is created to demonstrate OpenGL's concepts. It encompasses some of the skills learnt in our OpenGL classes such as pushmatrix(), translate(), popmatrix(), scale(). The scope is to use the basic primitives defined in OpenGL library creating complex objects. Expected Input will be the keyboard and mouse click events and expected output is the movement and rotational motion of the airplanes. The existing airshow simulation project is visually appealing computer graphics that can enhance the overall experience for the audience synchronized with the aerial performances to make the event more engaging and memorable.

1.2 Objectives

- **Realism:** The computer graphics should be highly realistic and visually appealing to create an immersive experience for the audience. The challenge lies in creating graphics that accurately depict aircraft, smoke trails, explosions, and other visual effects.
- **Synchronization:** The graphics should be synchronized with the aerial performances, meaning they should seamlessly integrate with the timing and movements of the aircraft. Achieving perfect synchronization between the live displays and the computer-generated graphics poses a significant challenge.
- **Performance:** The software must be able to render and display the graphics in real-time, ensuring smooth and uninterrupted playback throughout the event. This

requires optimizing the performance of the graphics engine and minimizing any potential delays or glitches.

- **Safety and Accuracy:** The graphics should not interfere with the pilots visibility or distract them during their performances.

1.3 Scope

The scope is to use the basic primitives defined in OpenGL library creating complex objects. Expected Input will be the keyboard and mouse click events and expected output is the movement and rotational motion of the airplanes by emitting smoke

CHAPTER 2

LITERATURE SURVEY

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

There are several ways in which the graphics generated by the program can be delivered.

- Frame- by- frame: A single frame can be drawn while the user waits.
- Frame-by-frame under control of the user: A sequence of frames can be drawn, as in a corporate power point presentation; the user presses to move on to the next slide, but otherwise has no way of interacting with the slides
- Animation: A sequence of frames proceeds at a particular rate while the user watches with delight.
- Interactive program: An interactive graphics presentation is watched, where the user controls the flow from one frame to another using an input device such as a mouse or keyboard, in a manner that was unpredictable at the time the program was written. This can delight the eye.

2.1 History

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

OpenGL is a standard specification that defines an API that is multi-language and multi-platform and that enables the codification of applications that output computerized graphics in 2D and 3D.

The interface consists in more than 250 different functions, which can be used to draw complex tridimensional scenes with simple primitives. It consists of many functions that help to create a real-world object and a particular existence for an object can be given.

2.2 Characteristics

- OpenGL is a better documented API.
- OpenGL is also a cleaner API and much easier to learn and program.
- OpenGL has the best demonstrated 3D performance for any API.
- Microsoft's Direct3D group is already planning a major API change called Direct Primitive that will leave any existing investment in learning Direct3D immediate mode largely obsolete.

2.3 Computer Graphics Library Organisation

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

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

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

Functions in main GL library name function names that begin with the letter ‘gl’.

- GLU library uses only GL functions but contains code for creating objects and simplify viewing.
- To interface with the window system and to get input from external devices GLUT library is used, which is a combination of three libraries GLX for X windows, ‘wgl’ for Windows and ‘agl’ for Macintosh.

- These libraries are included in the application program using pre-processor directives. E.g.: #include<GL/glut.h>
- The following figure shows the library organization in OpenGL.

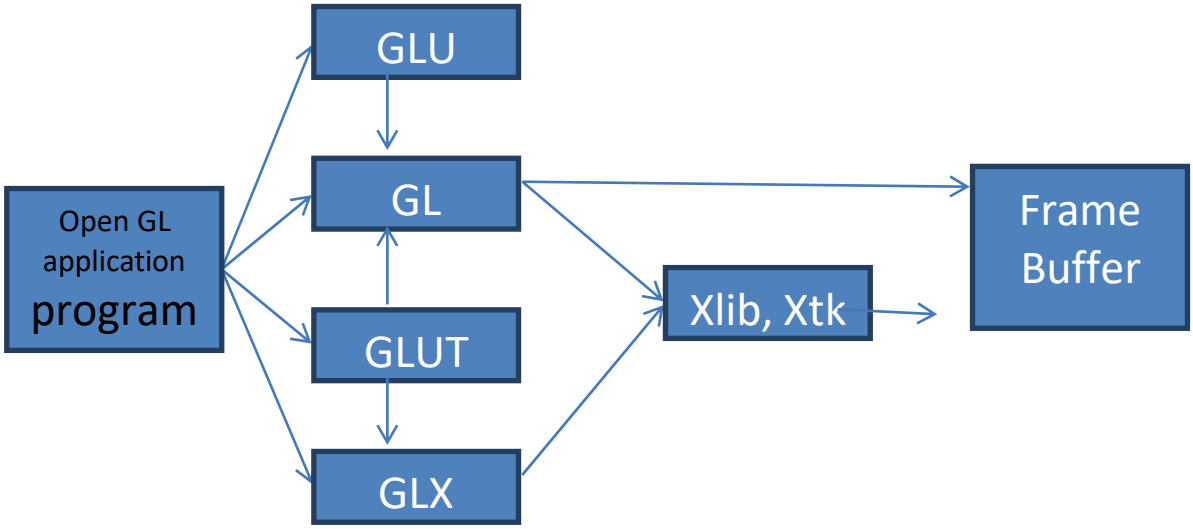

Fig 2.1: Library Organization

2.4 Graphics System and Functions

- Graphics system and functions can be considered as a black box, a term used to denote a system whose properties are only described by its inputs and output without knowing the internal working.
- Inputs to graphics system are functions calls from application program, measurements from input devices such as mouse and keyboard.
- Outputs are primarily the graphics sent to output devices.

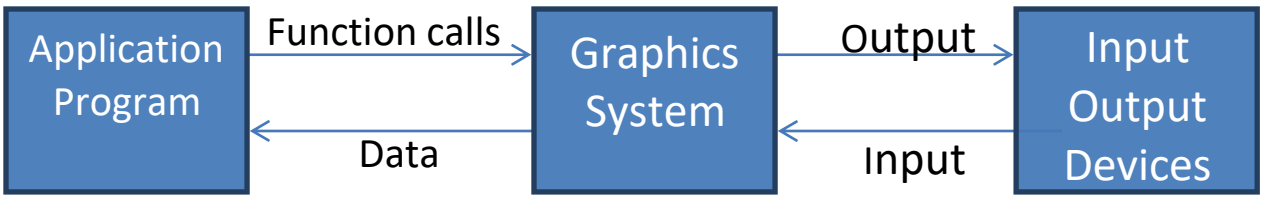

Fig 2.2: Graphics System as a Black Box

API's are described through functions in its library. These functions are divided into seven major groups.

1) Primitive Functions:

   Primitive functions define the low-level objects or atomic entities that a system can display, the primitives include line segments, polygons, pixels, points, text and various types of curves and surfaces.

2) Attribute Functions:

   Attribute Functions allow us to perform operations ranging from choosing the color to display a line segment, to packing a pattern to fill inside any solid figure.

3) Viewing Functions:

   Viewing functions allow us to specify various views.

4) Transformation Functions:

   Transformation functions allow us to carry out transformation of objects such as rotation, translation and scaling.

5) Input Functions:

   Input functions allow us to deal with the diverse forms of input that characterize modern graphics system. It deals with devices such as keyboard, mouse and data tablets.

6) Control Functions:

   Control Functions enable us to communicate with the window system, to initialize the programs, and to deal with any errors that occur during the execution of the program.

7) Query Functions:

   Query Functions provides information about the API.

CHAPTER 3

SYSTEM REQUIREMENTS

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

3.1 Hardware Requirement

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

3.2 Software Requirement

- Windows – XP/7/8/10
- Microsoft Visual Studio C/C++ 7.0 and above versions
- OpenGL Files
- DirectX 8.0 and above versions

Header Files

- glut.h

Object File Libraries

- glut32.lib

DLL files

- glut32.dll

CHAPTER 4

SYSTEM DESIGN

4.1 Flow Chart

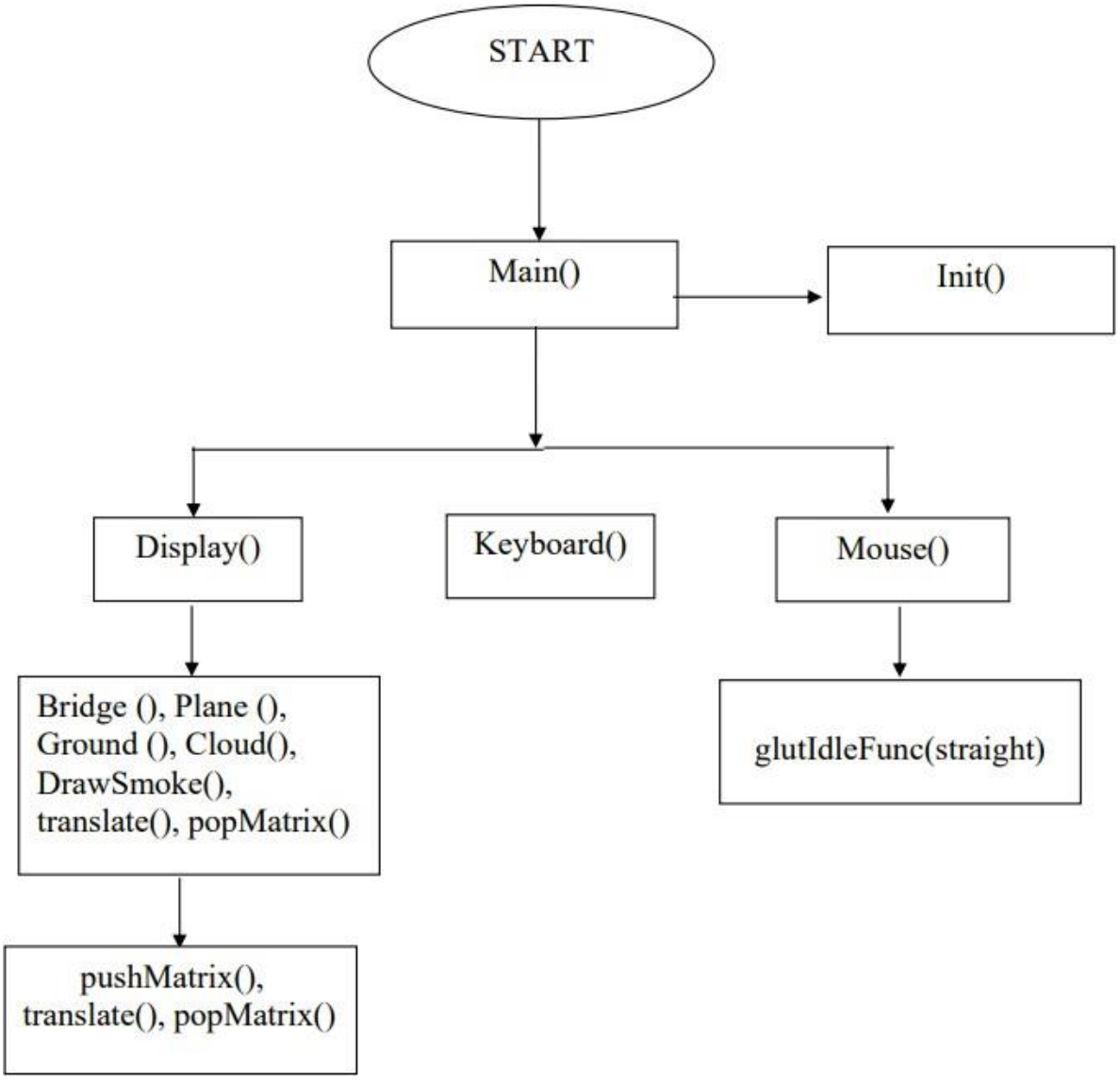

Fig 4.1 : Flow chart of the code

A flowchart is a picture of the separate steps of a process in sequential order. It is a generic tool that can be adapted for a wide variety of purposes, and can be used to describe various processes, such as a manufacturing process, an administrative or service process, or a project plan.

In OpenGL, an object is made up of geometric primitives such as triangle, quad, line segment and point. A primitive is made up of one or more vertices. OpenGL supports the following primitives: A geometric primitive is defined by specifying its vertices via glVertex function, enclosed within a pair glBegin and glEnd .

The flow of control in the flow chart is respected to the Texture Package. For any of the program flow chart is compulsory to understand the program. We consider the flow chart for the texture project in which the flow starts from start and proceeds to the main function after which it comes to the initialization of call back functions and further it proceeds to mouse and keyboard functions, input and calculation functions. Finally, it comes to quit, the end of flow chart.

This flow chart contains the flow of functions (inbuilt and user defined) which is required for the successful running of the airshow simulation program.

CHAPTER 5

IMPLEMENTATION

5.1 Header Files Used

- **#include<windows.h>**

  windows.h is a Windows-specific header file for the C and C++ programming languages which contains declarations for all of the functions in the Windows API, all the common macros used by Windows programmers, and all the data types used by the various functions and subsystems. It defines a very large number of Windows specific functions that can be used in C.

- **#include<GL/glut.h>**

  The OpenGL Utility Toolkit (GLUT) handles most of the system dependent actions required to display a window, put OpenGL graphics in it, and accept mouse and keyboard input. It is used to create the skeleton for all the assignments that hides Windows programming issues from you so you can concentrate on programming graphics. glut.h includes gl.h and glu.h, so they shouldn't be included if glut.h is included.

- **#include <math.h>**

  The math.h header defines various C mathematical functions and one macro. All the functions available in this library take double as an argument and return double as the result. Let us discuss some important C math functions one by one**.**

- **#include<stdio.h>**

  stdio.h is a header file which has the necessary information to include the input/output related functions in our program.

- **#include<string.h>**

  The string.h header defines one variable type, one macro, and various functions for manipulating arrays of characters. The string. h functions in C are not only used for string handling or manipulation but also used for various memory handling operations**.**

- **#include<stdlib.h>**

  The name "stdlib" stands for "standard library". It is the header of the general purpose standard library of C programming language which includes functions involving memory allocation, process control, conversions and others. the header of the general purpose standard library of C programming language which includes functions involving memory allocation, process control, conversions and others.

5.2 OpenGL API's Used

- **void glColor3f(float red, float green, float blue);**

  This function is used to mention the colour in which the pixel should appear. The number 3 specifies the number of arguments that the function would take. 'f ' gives the type that is float. The arguments are in the order RGB(Red, Green, Blue). The colour of the pixel can be specified as the combination of these 3 primary colours

- **void glClearColor(int red, int green, int blue, int alpha);**

  This function is used to clear the colour of the screen. The 4 values that are passed as arguments for this function are (RED, GREEN, BLUE, ALPHA) where the red, green, and blue components are taken to set the background colour and alpha is a value that specifies depth of the window. It is used for 3D images

- **void glutKeyboardFunc();**

  void glutKeyboardFunc(void (*func)(unsigned char key, int x, int y));

  where func is the new keyboard call-back function. glutKeyboardFunc sets the keyboard call-back for the current window. When a user types into the window, each

key press generating an ASCII character will generate a keyboard call-back. The key call-back parameter is the generated ASCII character. The x and y call-back parameters indicate the mouse location in window relative coordinates when the key was pressed. When a new window is created, no keyboard call back is initially registered, and ASCII keystrokes in the window are ignored. Passing NULL to glutKeyboardFunc disables the generation of keyboard call-backs.

- **void glutMouseFunc(void (*func)(intbutton, intstate , intx, inty));**

  glutMouseFunc () register an application callback function which GLUT will call when the user presses a mouse button within the window. The following values are passed to the callback function:

  button:- records which button was pressed, and can be -GLUT LEFT BUTTON -GLUT MIDDLE BUTTON -GLUT RIGHT BUTTON

  state:- records whether the event was generated by pressing the button (GLUT DOWM) or releasing it (GLUT UP).

  x,y:- give the current mouse position in pixels. When using OpenGL with X, the mouse y position is measured from the top of the window.

- **void glFlush();**

  Different GL implementations buffer commands in several different locations, including network buffers and the graphics accelerator itself. glFlush empties all 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 time, it does complete in finite time.

- **void glMatrixMode(GLenum mode);**

  where mode Specifies which matrix stack is the target for subsequent matrix operations. Three values are accepted: GL_MODELVIEW, GL_PROJECTION, and GL_TEXTURE. The initial value is GL_MODELVIEW. glMatrixMode sets the current matrix mode. Mode can assume one of three values.GL_MODELVIEW applies subsequent matrix operations to the model view matrixstack.

- **void glViewport(GLint x, GLint y, GLsizei width, GLsizei height);**

  where x, y Specify the lower left corner of the viewport rectangle, in pixels. The initial value is (0, 0). The width, height Specify the width and height of the viewport. When a GL context is first attached to a surface (e.g., window), width and height are set to the dimensions of that surface. GlViewport specifies the affine transformation of x and y from normalized device coordinates to window coordinates. Let (xnd, ynd) be normalized device coordinates. Then the window coordinates (xw, yw) are computed as follows:

  xw = (xnd + 1) width/2 + x

  yw = (ynd + 1) height/2 + y

- **void glutInit(int *argcp, char **argv);**

  glutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized. Examples of this situation include the failure to connect to the window system, the lack of window system support for OpenGL, and invalid command line options. glutInit also processes command line options, but the specific options parse are window system dependent.

- **void glutReshapeFunc(void (*func)(int width, int height));**

  glutReshapeFunc sets the reshape call-back for the current window. The reshape call-back is triggered when a window is reshaped. A reshape call-back is also triggered immediately before a window's first display call-back after a window is created or whenever an overlay for the window is established. The width and height parameters of the call-back specify the new window size in pixels. Before the call-back, the current window is set to the window that has been reshaped. If a reshape call-back is not registered for a window or NULL is passed to glutReshapeFunc (to deregister a previously registered call-back), the default reshape call-back is used. This default call-back will simply glOrtho()

Syntax: void glOrtho (GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far);

The function defines an orthographic viewing volume with all parameters measured from the centre of the projection plane.

- **glutPostRedisplay();**

  glutPostRedisplay, glutPostWindowRedisplay — marks the current or specified window as needing to be redisplayed.

- **void glutMainLoop(void);**

  glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never stop.

- **void glTranslatef (GLfloat x, GLfloat y, GLfloat z);**

  glTranslatef() creates a matrix M which performs a translation by (x,y,z) and then post-multiplies the current matrix by M.

- **void glScalef (GLfloat x,GLfloat y,GLfloat z);**

  glScalef() creates a matrix M which performs a scale by (x,y,z) and then post-multiplies the current matrix by M.

- **void glRotatef(GLfloat angle,GLfloat x,GLfloat y,GLfloat z);**

  glRotate() creates a matrix M which performs a counter-clockwise rotation of angle in degrees. The axis about which the rotation occurs in the vector from the origin (0, 0, 0) to the point (x,y,z),and then post-multiplies the current matrix by M

5.3 User Defined Functions

- **void plane(int Colour)**

  This function is defined to draw a plane with the respective colour which is given as input parameter. This function also calls other user defined functions to draw the

user desired plane**.**

```
void plane(int Colour)
{
        float rb = 0;
        float gb = 0;
        float bb = 0;
        if (Colour == 1)
                { rb = 0.65; }
        else if (Colour == 2)
                { gb = 1.0, rb = 1.0, bb = 1.0; }
        else if (Colour == 3)
                { bb = 0.65; }
        glPushMatrix(); //Body
        glTranslatef(0, 0, 4);
        glScaled(1, 0.8, 5);
        sphere(rb, gb, bb, 1);
        glPopMatrix();
        glPushMatrix();  //Windscreen
        glTranslatef(0, 2, 16);
        glScaled(0.5, 0.4, 0.75);
        sphere(0, 0, 0, 0.75);
        glPopMatrix();
        wing(Colour); //Left Wing
        glPushMatrix();
        glScalef(-1, -1, 1);
```

```
        wing(Colour);  //Right Wing
        glPopMatrix();
        glPushMatrix();
        glScalef(0.4, 0.4, 0.4);
        glTranslatef(0, 3, -25);
        wing(Colour);  //Left mini wing
        glPopMatrix();
        glPushMatrix();
        glScalef(-0.4, -0.4, 0.4);
        glTranslatef(0, -3, -25);
        wing(Colour); //Right mini wing
        glPopMatrix();
        fin(Colour);  //Fin
}
```

- **void cloud()**

  This function is defined to draw clouds in the sky. The dimensions and size of the cloud that the user wants is defined within the function.

```
void cloud()
{
        glPushMatrix();
        glScalef(1.5, 1, 1.25);
        glTranslatef(0, 12, 0);
        sphere(1, 1, 1, 0.9);
        glPopMatrix();
        glPushMatrix();
        glScalef(1.5, 1, 1.25);
        glTranslatef(0, 5, 3);
        sphere(1, 1, 1, 0.9);
```

```
        glPopMatrix();
        glPushMatrix();
        glScalef(1.5, 1, 1.25);
        glTranslatef(4, 7, 0);
        sphere(1, 1, 1, 0.9);
        glPopMatrix();
        glPushMatrix();
        glScalef(1.5, 1, 1.25);
        glTranslatef(-4, 7, 0);
        sphere(1, 1, 1, 0.9);
        glPopMatrix();
        glPushMatrix();
        glScalef(2, 1.5, 2);
        glTranslatef(0, 5, 0);
        sphere(1, 1, 1, 0.5);
        glPopMatrix();
}
```

- **void bridge()**

  This user defined function is defined to draw a bridge ,which acts as a runway where the plane takes off.

```
void bridge()
{
        glPushMatrix();
        glScalef(20, 1, 10);
        slab(0.5, 0.5, 0.5):
        glPopMatrix();
        glPushMatrix();
        glScalef(20, 2, 1);
        glTranslatef(0, 0.75, 4.5);
```

```
        slab(0.3, 0.3, 0.3);
        glPopMatrix();
        glPushMatrix();
        glScalef(20, 2, 1);
        glTranslatef(0, 0.75, -4.5);
        slab(0.3, 0.3, 0.3);
        glPopMatrix();
        glPushMatrix();
        glScalef(1, 4, 1);
        glTranslatef(0, -0.65, 4.5);
        slab(0.3, 0.3, 0.3);
        glPopMatrix();
        glPushMatrix();
        glScalef(1, 4, 1);
        glTranslatef(0, -0.65, -4.5);
        slab(0.3, 0.3, 0.3);
        glPopMatrix();
}
```

- **void drawSmoke(float R, float G, float B, float x, float y, int reflect)**

  drawSmoke function is used to draw smoke which comes behind the aircrafts. Here R,G,B values represent the colour of smoke to be emitted. x,y values represent the direction of smoke .

```
void drawSmoke(float R, float G, float B, float x, float y, int reflect)
{
        // Calculate each position, size and transparency of smoke
        for (int xi = 0; xi < 150; xi++)
        {
        sa[xi] = sa[xi] - 0.0011;
```

```
                ss[xi] = ss[xi] + 0.005;
                glPushMatrix();
                glScalef(reflect * 0.5, reflect * 0.5, 0.5);
                glTranslatef((reflect * sx[xi]) - x, sy[xi] - y, sz[xi]);
                glRotatef(spinxs[xi], 1, 0, 0);
                glRotatef(reflect * spinys[xi], 0, 1, 0);
                glRotatef(reflect * spinzs[xi], 0, 0, 1);
                smoke(ss[xi], sa[xi], R, G, B);
                glPopMatrix();
            }
        }
```

CHAPTER 6

SNAPSHOTS

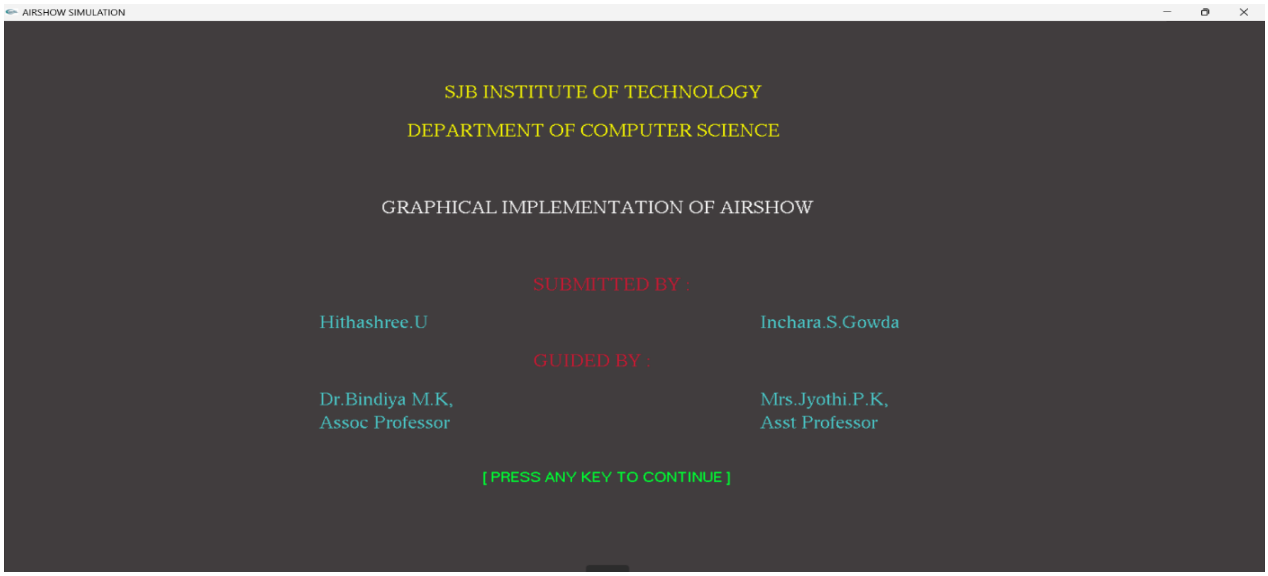

Fig 6.1: First display page

The above picture is the window that gets displayed as soon as we run the code. It contains the details of the creators of the projects and the guides. On clicking any key in the keyboard,it will get directed to another window.

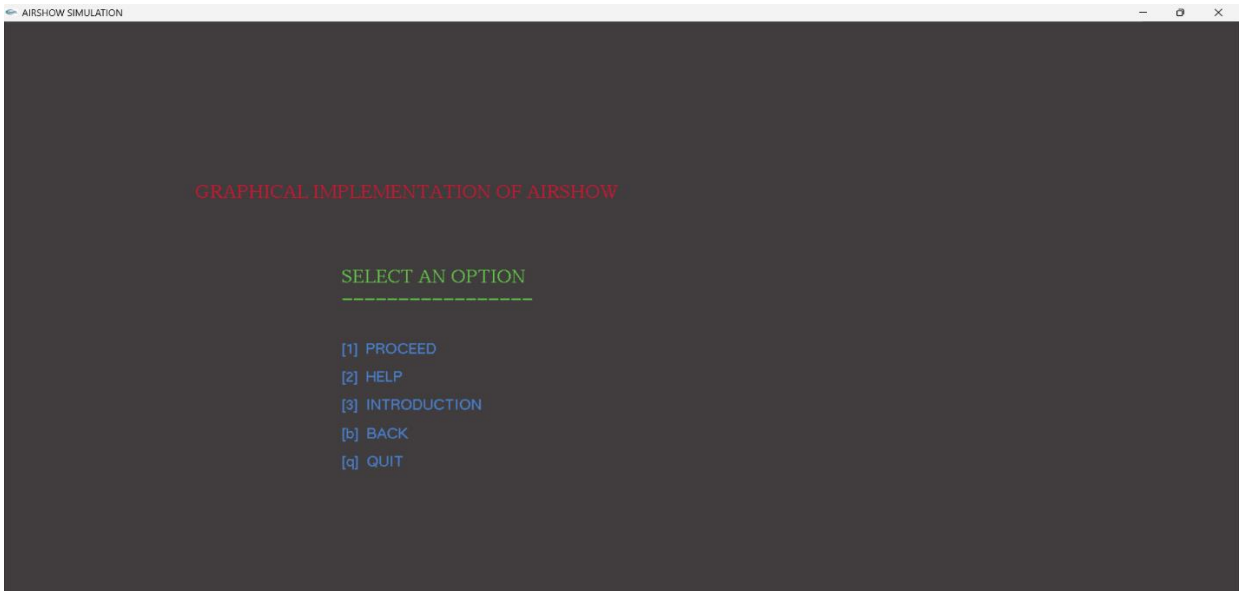

Fig 6.2: Menu page

This picture is the page that contains the actions that the user can perform in this project.

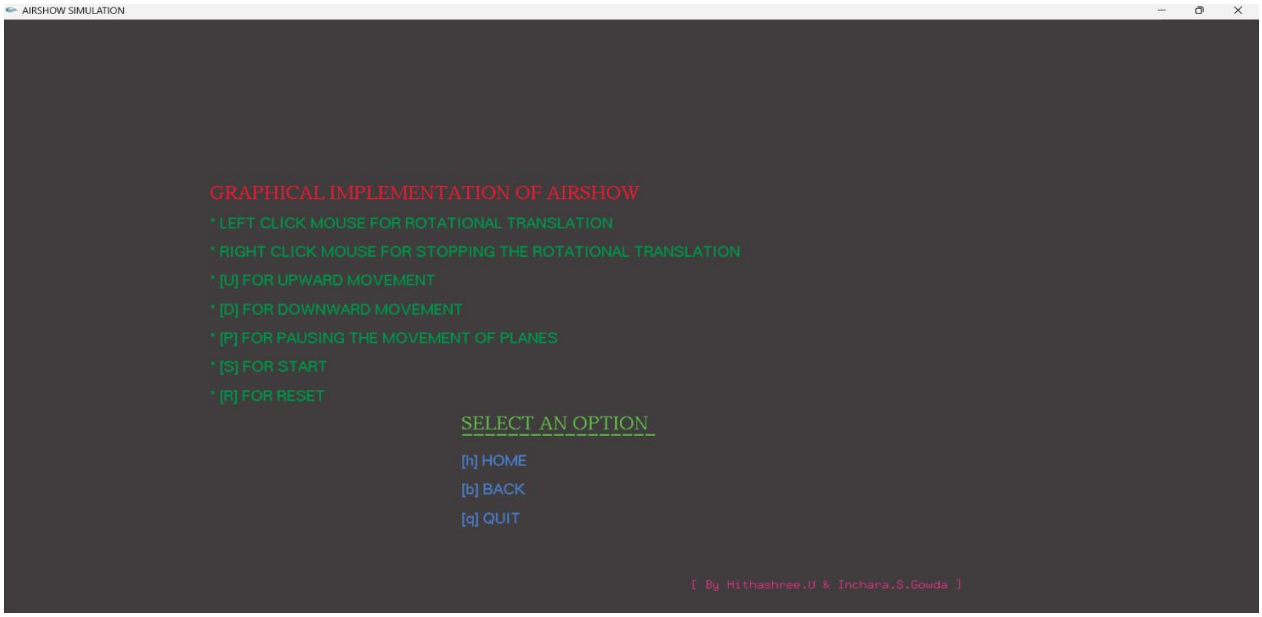

Fig 6.3: Help page

The picture above contains the instructions of all the actions that the user can perform on the plane like starting the aircraft, pausing it , it's upward and downward movement and rotation

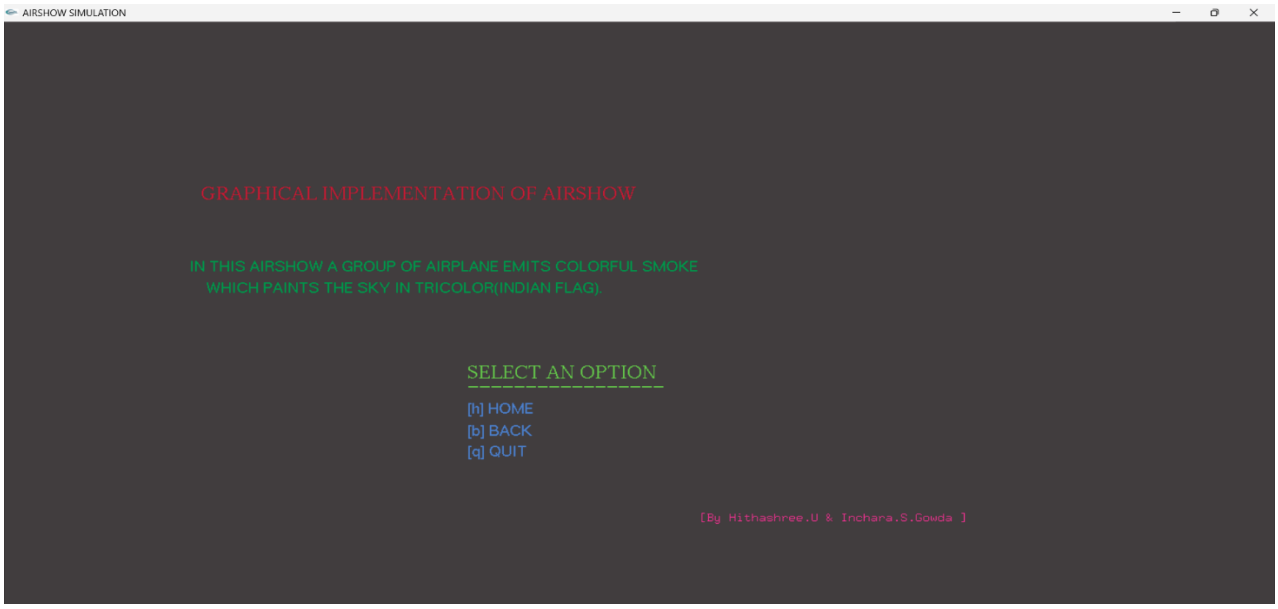

Fig 6.4: Introduction page

The above picture is a page that contains the introduction about the project, it says that "IN THIS AIRSHOW A GROUP OF AIRPLANE EMITS COLORFUL SMOKE WHICH PAINTSTHE SKY IN TRICOLOR(INDIAN FLAG)."

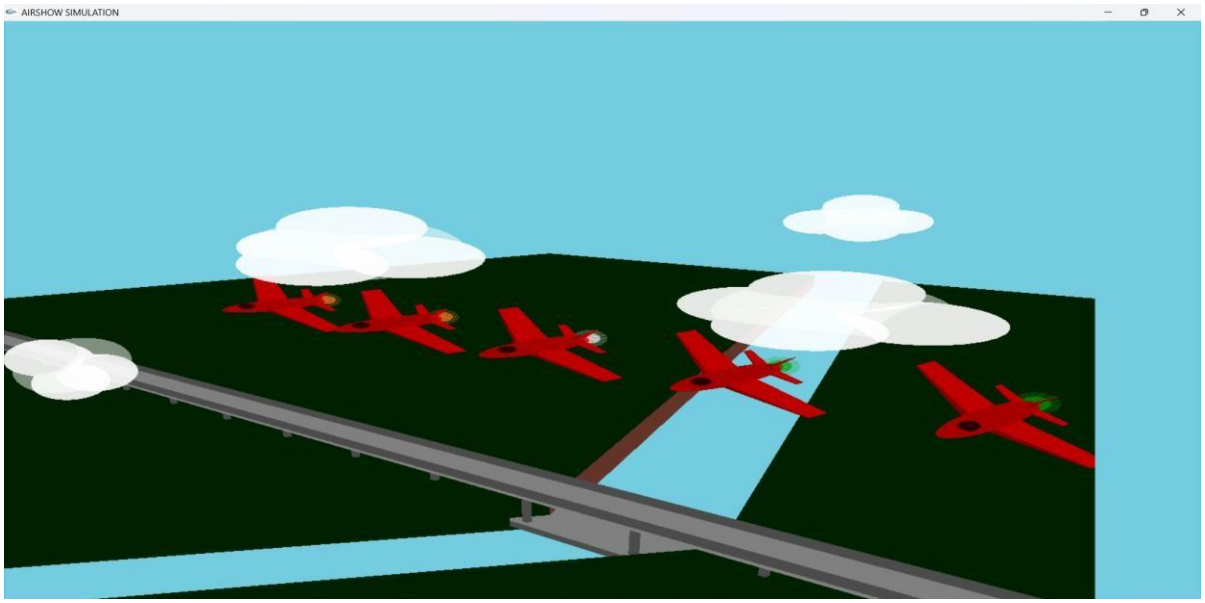

Fig 5.5: Starting position of the planes

This picture is the view of the starting position of the aircrafts it contains 5 planes, bridge andclouds. Once the planes start moving they emit their respective assigned colours.

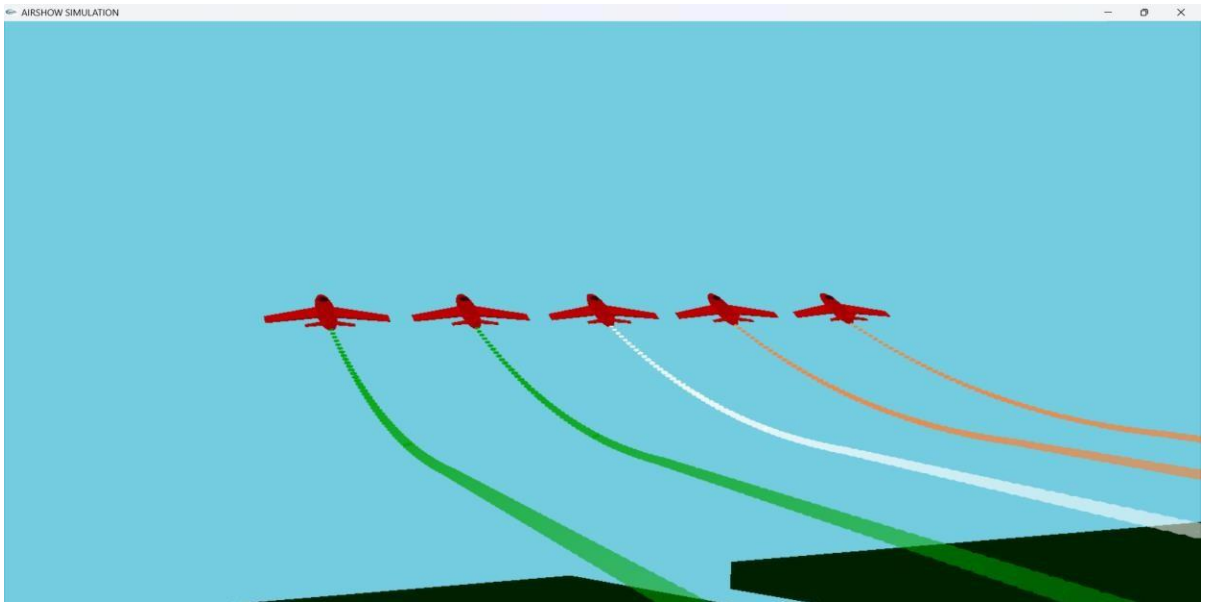

Fig 5.6: Planes flying in the upward direction

The above picture appears once we click the ‘u’ key in the keyboard the planes start moving in the upward motion by emitting different coloured smokes.

CONCLUSION

After the completion of this project “AIRSHOW SIMULATION” we came to know how we can implement a project using an Open source OpenGL tool kit. By implementing a project using Open GL we came to know how to use the functions like menu, rotation, translation and scaling. With the completion of this project we have achieved a sense of happiness and we want to thank all those who helped us directly or indirectly to make this idea come true.

This project can be used for simulation of air shows during National festivals like INDEPENDENCE DAY and REPUBLIC DAY to avoid accidents happening during training by scaling the airplanes in a screen of 1366x768 and using a multiplying factor to check the pattern of air show in real life.

Some **future enhancements** can be:

- trying the flag colours of different countries.
- trying different views of the planes across different angles.
- detailed view of take-off and landing of planes.
- flying the planes on different terrestrial locations.

REFERENCES

[1] Donald Hearn and Pauline Baker, Computer Graphics - OpenGL Version –2nd Edition, Pearson Education, 2003.

[2] Edward Angel, Interactive Computer Graphics A Top-Down Approach with OpenGL, 5th Edition, eddison-Wesley, 2008.

[3] James D Foley, Andries Van Dam, Steven K Feiner, John F Hughes, Computer Graphics –Addison-Wesley 1997.

[4] Xiang, Plastock : Computer Graphics , sham‟s outline series, 2nd edition, TMG.

[5] Kelvin Sung, Peter Shirley, steven Baer : Interactive Computer Graphics, concepts and applications, Cengage Learning.

[6] M M Raikar & Shreedhara K S Computer Graphics using OpenGL,Cengage publication.

[7] https://www.opengl.org/

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