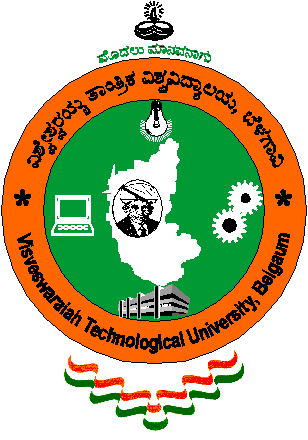
**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**BELAGAVI -590014**

****

**A MINI PROJECT REPORT ON**

**AIR TRAFFIC CONTROL**

**A Report Submitted in Partial Fulfillment of the Requirements**

**For the VI semester B.E**

**Under the Guidance of:**

**Prof. Basavaraj Neelagund**

**Prof. Ayisha Thasreefa P**

**Project Associate**

**Jasnitha Joy 4DM21CS021**

**Minal Fathima P 4DM21CS026**

****

**Department of Computer science & Engineering**

**YENEPOYA INSTITUTE OF TECHNOLOGY,**

Moodbidri**-**574225

2022-2023

****

Yenepoya Institute of Technology, Moodbidri-574225

***Certificate***

Department of Computer Science & Engineering – 2022-23

This is to certify that **Jasnitha Joy** and **Minal Fathima P** bearing the USN**:4DM21CS021** and **4DM21CS26** respectively, have satisfactorily completed the course of implementation of Mini Project entitled “**Air Traffic Control**”, prescribed by the Visveswaraya Technological University, Belagavi for the degree course in the laboratory of Computer Graphics And Image Processing Laboratory(21CSL66) in the semester 6th. This mini project report has been approved as it satisfies the academic requirements in the respect of project work.

………………. ………………….

Staff in-charge HOD

**(Prof. Ayisha Thasreefa P) (Prof.Pandu Naik)**

Examiners: 1) ………………………. 2) ……………………….

**ACKNOWLEDGEMENT**

Endeavors have borne fruit, and as we prepare to forge ahead, we stop for the moment in our tracks to acknowledge our sincere gratitude for the assistance, efforts and patronage we have received in the completion of my mini project to our respected teachers **Prof. Basavaral Neelagund** and **Prof. Ayisha ThasreefaP** of **CSE** Department, who inspired us as our guide with their experience and knowledge providing us with educative support. We whole-heartedly express our appreciation and gratitude.

We are thankful to our beloved **HOD of CSE Department. Prof.Pandu Naik** and to our respected principal **Dr.R.G.D’Souza** for their outstanding & continuous support throughout the completion of this mini project.

We sincerely thank all teaching & technical staffs, those who have helped directly or indirectly in completing this project. We also offer our deep gratitude to our parents and our seniors who have appreciated, encouraged and assisted us in our endeavors.

**Jasnitha Joy 4DM21CS021 Minal Fathima P 4DM21CS026**

**Abstract**

The main aim of this Mini Project is to illustrate the concepts and usage of Air Traffic Control in OpenGL. Air traffic control (ATC) is a service provided by ground-based controllers who direct aircraft on the ground and through controlled airspace. The primary purpose of ATC systems worldwide is to prevent collisions, organize and expedite the flow of traffic, and provide information and other support for pilots to prevent collisions, ATC enforces traffic separation rules, which ensure each aircraft maintains a minimum amount of empty space around it at all times. We have used input devices like the mouse and keyboard to interact with the program. We have used input devices like the mouse and keyboard to interact with the program. We have added menu which makes the program more interactive.

**CONTENTS**

**Topics page no.**

Chapter 1: Introduction to OpenGL 01-03

Chapter 2 : OpenGL Functionalities 04-06

Chapter 3: Air Traffic Control 07-07

Chapter 4: Source Code 08-17

Chapter 5: Output Snapshots 18-19

Conclusion 20-20

References 21-21

**Chapter 1**

**INTRODUCTION TO OPENGL**

**1.1 What is OpenGL?**

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that you use to specify the objects and operations needed to produce interactive three-dimensional applications.

Provides a set of commands to render a three dimensional scene. That means you provide the data in an OpenGL-useable form and OpenGL will show this data on the screen (render it). It is developed by many companies and it is free to use. You can develop OpenGL-applications without licensing.

OpenGL is a hardware- and system-independent interface. An OpenGL-application will work on every platform, as long as there is an installed implementation.

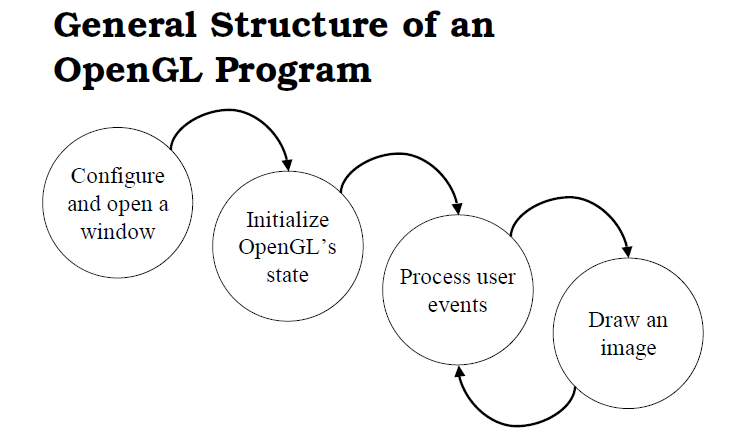
Because it is system independent, there are no functions to create windows etc., but there are helper functions for each platform. A very useful thing is GLUT.

**1.2Graphics with OpenGL**

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 three-dimensional 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.

* 1. **Structure of OpenGL**



**1.4 OpenGL and related APIs**



**1.5How does OpenGL work?**

OpenGL bases on the state variables. There are many values, for example the color, that remain after being specified. That means, you can specify and draw several polygons, lines or what ever with this color then. There are no classes like in DirectX. However, it is logically structured. Before we come to the commands themselves, here is another thing:

To be hardware independent, OpenGL provides its own data types. They all begin with "GL". For example GLfloat, GLint and so on. There are also many symbolic constants, they all begin with "GL\_", like GL\_POINTS, GL\_POLYGON. Finally the commands have the prefix "gl" like glVertex3f (). There is a utility library called GLU, here the prefixes are "GLU\_" and "glu". GLUT commands begin with "glut", it is the same for every library. You want to know which libraries coexist with the ones called before? There are libraries for every system, Windows has the wgl\*-Functions, Unix systems glx\* and so on.

A very important thing is to know, that there are two important matrices, which affect the transformation from the 3d-world to the 2d-screen: The projection matrix and the modelview matrix. The projection matrix contains information, how a vertex – let's say a "point" in space shall be mapped to the screen. This contains, whether the projection shall be isometric or from a perspective, how wide the field of view is and so on. Into the other matrix you put information, how the objects are moved, where the viewer is and so on.

**Chapter 2**

**OPENGL FUNCTIONS**

The following OpenGL functions are used in this project:

* 1. **glRasterPos():**

Sets the current raster position. The x, y, z, and w arguments specify the coordinates of theraster position. If glRasterPos2f() is used, z is implicitly set to zero and w is implicitly set to onesimilarly, with glRasterPos3f(), w is set to one.

**2.2 glClearColor()**

The call to set a window’s background color in OpenGL is glClearColor(), it takes 4 arguments:the 1st three arguments indicates red, green, and blue, respectively. The fourth value in glClearColor() is named alpha. Generally, when you call glClearColor(), you want to set the alpha component to 1.0

**2.3 glViewport()**

Syntax: glViewport( x, y, width, height )

clips the vertex or raster position. For geometric primitives, a new vertex may be created. For raster primitives, the raster position is completely clipped.

**2.4 glOrtho()**

Syntax:glOrtho( left, right, bottom, top, zNear, zFar)

The viewing volume is shaped like a rectangular parallelepiped (a box). Vertices of an object are “projected” towards infinity, and as such, distancedoes not change the apparent size of an object, as happens under perspective projection. Orthographic projection is used for drafting, and design.

**2.5 glClearColor(0.0,0.0,0.0,0.0);**

It defines RGB color that is black because the first three components are set to 0.0 and is transparent. As alpha component is 0.0.

**2.6 glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);**

This function is used to make the window on the screen solid and white. Algorithm stores the information in depth buffer, we must clear the buffer to redraw the display.

**2.7 glLoadIdentity();**

This function sets a matrix initially to identitiy matrices.

**2.8 glMatrixMode(GL\_PROJECTION);**

**glMatrixMode(GL\_MODELVIEW);**

We select the matrix to which the operations apply by first setting the matrix mode. This can be applied to model view matrix and projection matrix.

**2.9glViewport(GLintx,GLinty,GLsizeiw,GLsizei h);**

Viewport is the rectangular area of the display window. It can be set to any smaller size in pixels by using above function where (x,y) is the lower-left corner of the viewport and w and h give the height and width respectively.

**2.10glutCreateWindow(char \*title);**

It specifies the title at the top of the window is given by the string title.

**2.11glutDisplayFunc(void (\*func) (void));**

In the above function named func will be whenever window needs to be redisplayed. This function is named through the GLUT function.

**2.12glutInit(int \*argc,char\*\* argv);**

when we open a window there must be interaction between windowing system and OpenGL. This interaction is initiated by above function call. The two arguments allow the user to pass command line arguments.

**2.13glutMainLoop();**

Above function execution will cause the program to begin an event processing loop.

**2.14glutReshapeFunc(reshape);**

This function is used whenever the window is to be resized.

**2.15 glRotate{fd}(TYPE angle, TYPE x, TYPE y, TYPE z);**

Multiplies the current matrix by a matrix that rotates an object (or the local coordinate system) in a counterclockwise direction about the ray from the origin through the point (x, y, z). The angle parameter specifies the angle of rotation in degrees

**2.16glTranslate{fd} (TYPEx, TYPE y, TYPEz);**

Multiplies the current matrix by a matrix that moves (translates) an object by the given x, y, and z values (or moves the local coordinate system by the same amounts).parameter specifies the angle of rotation in degrees

**Chapter 3**

**AIR TRAFFIC CONTROL**

**Introduction:**

The primary objective of this Mini Project is to demonstrate the principles and application of Air Traffic Control (ATC) using OpenGL. Air traffic control is an essential service provided by ground-based controllers who manage the safe and efficient movement of aircraft both on the ground and through controlled airspace. The main goals of ATC are to prevent collisions, organize and expedite air traffic flow, and offer critical support and information to pilots. To achieve these objectives, ATC systems enforce traffic separation rules to ensure that each aircraft maintains a safe distance from others.

In this project, OpenGL is utilized to create a simulation of ATC operations. Users interact with the simulation through keyboard and mouse inputs to manage and monitor air traffic effectively. Keyboard shortcuts streamline control: 'a' adds a new plane, 'r' removes a plane, 't' initiates takeoff, and 'q' exits the program. The inclusion of a menu system, accessible via mouse interactions, enhances user experience by providing a dynamic and intuitive interface. Right-clicking reveals a contextual menu, and left-clicking allows users to select options. This combination of input methods ensures a user-friendly and interactive simulation, providing valuable insights into air traffic management within an OpenGL framework.

**Usage:**

In our air traffic control OpenGL project, interaction is facilitated through popular input devices such as the mouse and keyboard. Specific keys on the keyboard have designated functions to streamline the process: pressing 'a' adds a new plane, 'r' removes a plane, 't' initiates takeoff, and 'q' quits the program. These keyboard shortcuts provide quick and efficient control over the air traffic simulation. Additionally, mouse interactions enhance the user experience. Right-clicking the mouse triggers a contextual menu, and options within this menu can be selected with a left-click. This dual input method, combining keyboard shortcuts and mouse clicks, ensures a comprehensive and intuitive user interface for managing the air traffic control system within the OpenGL project.

**Chapter 4**

**SOURCE CODE**

#include <windows.h>

#include<string.h>

#include<stdarg.h>

#include<stdio.h>

#include <GL/glut.h>

#include<stdbool.h>

static double x[10]={0},x2=0.0,r1=0.0;

static double yaxis[10]={-15,-15,-15,-15,-15,-15,-15,-15,-15,-15};

static double max=0;

static bool takeOff=false;

void

stroke\_output(GLfloat x, GLfloat y, char \*format,...)

{

va\_list args;

char buffer[200], \*p;

va\_start(args, format);

vsprintf(buffer, format, args);

va\_end(args);

glPushMatrix();

glTranslatef(-2.5, y, 0);

glScaled(0.003, 0.005, 0.005);

for (p = buffer; \*p; p++)

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, \*p);

glPopMatrix();

}

//runway strip

void strip(float x1)

{

glPushMatrix();

glRotatef(-65,0,1,0);

glColor3f(1,1,1);

glTranslatef(x1,-3.5,7.8);

glScaled(1,0.15,0.1);

glutSolidCube(1);

glPopMatrix();

}

void drawPlane(float y1){

/\*\*\*\* PLANE CONSTRUCTION \*\*\*\*\*\*\*\*/

glPushMatrix();

// Main Body

glPushMatrix();

glScalef(.3,0.3,1.5);

if(y1<=15)

glColor3f(1.0, 0.0, 1.0); // Purple

if(y1>=15)

glColor3f(1.0, 0.0, 1.0); // Purple

glutSolidSphere(2.0,50,50);

glPopMatrix();

glPushMatrix();

glTranslatef(0.0,0.1,-1.8);

glScalef(1.0,1,1.5);

glColor3f(0,0,1);

glutSolidSphere(0.5,25,25);

glPopMatrix();

//Left Fin

glPushMatrix();

glTranslatef(-1.0,0,0);

glScalef(1.5,0.1,0.5);

glColor3f(0,0,0);

glutSolidSphere(1.0,50,50);

glPopMatrix();

// Right Fin

glPushMatrix();

glTranslatef(1.0,0,0);

glScalef(1.5,0.1,0.5);

glColor3f(0,0,0);

glutSolidSphere(1.0,50,50);

glPopMatrix();

//right Tail fin

glPushMatrix();

glTranslatef(0.8,0,2.4);

glScalef(1.2,0.1,0.5);

glColor3f(0.0,0,0);

glutSolidSphere(0.4,50,50);

glPopMatrix();

//left Tail fin

glPushMatrix();

glTranslatef(-0.8,0,2.4);

glScalef(1.2,0.1,0.5);

glColor3f(0.0,0,0);

glutSolidSphere(0.4,50,50);

glPopMatrix();

//Top tail fin

glPushMatrix();

glTranslatef(0,0.5,2.4);

glScalef(0.1,1.1,0.5);

glColor3f(0.0,0,0);

glutSolidSphere(0.4,50,50);

glPopMatrix();

// Blades

glPushMatrix();

glRotatef(x2,0.0,0.0,1.0);

glPushMatrix();

glTranslatef(0,0.0,-3.0);

glScalef(1.5,0.2,0.1);

glColor3f(0.0,0,0);

glutSolidSphere(0.3,50,50);

glPopMatrix();

//blades

glPushMatrix();

glRotatef(90,0.0,0.0,1.0);

glTranslatef(0,0.0,-3.0);

glScalef(1.5,0.2,0.1);

glColor3f(0.0,0,0);

glutSolidSphere(0.3,50,50);

glPopMatrix();

glPopMatrix();

/\* Blased End \*/

/\* Wheels \*/

//Front

glPushMatrix();

glTranslatef(0.0,-0.8,-1.5);

glRotatef(90,0.0,1,0);

glScaled(0.3,0.3,0.3);

glutSolidTorus(0.18,0.5,25,25);

glColor3f(1,1,1);

glutSolidTorus(0.2,0.1,25,25);

glPopMatrix();

glPushMatrix();

glTranslatef(0.0,-0.4,-1.5);

glRotatef(20,0.0,1,0);

glScaled(0.05,0.3,0.05);

glutSolidSphere(1.0,25,25);

glPopMatrix();

//Rear

glPushMatrix();

glTranslatef(0.3,-0.8,0.7);

glRotatef(90,0.0,1,0);

glScaled(0.3,0.3,0.3);

glColor3f(0,0,0);

glutSolidTorus(0.18,0.5,25,25);

glColor3f(1,1,1);

glutSolidTorus(0.2,0.1,25,25);

glPopMatrix();

glPushMatrix();

glTranslatef(0.3,-0.4,0.7);

glRotatef(20,0.0,1,0);

glScaled(0.05,0.3,0.05);

glutSolidSphere(1.0,25,25);

glPopMatrix();

//rear 2

glPushMatrix();

glTranslatef(-0.3,-0.8,0.7);

glRotatef(90,0.0,1,0);

glScaled(0.3,0.3,0.3);

glColor3f(0,0,0);

glutSolidTorus(0.18,0.5,25,25);

glColor3f(1,1,1);

glutSolidTorus(0.2,0.1,25,25);

glPopMatrix();

glPushMatrix();

glTranslatef(-0.3,-0.4,0.7);

glRotatef(20,0.0,1,0);

glScaled(0.05,0.3,0.05);

glutSolidSphere(1.0,25,25);

glPopMatrix();

glPopMatrix();

}

void animate(float y1,float x1){

// Plane Transition

glPushMatrix();

//Move the Plane towards rotating zone

if(y1<=-2){

glTranslatef(5.5+y1,3,0);

glRotatef(-90,0,1,0);

}

// Move the Plane towards 2nd runway

if(takeOff)

if(y1>=15){

glRotatef(140,0,1,0);

if(y1>=15 && y1<=20)

glTranslatef(2+15-y1,-3,-3);

if(y1>=20)

glTranslatef(2+15-y1,-3-20+y1,-3);

}

// keep rotating the plane

if(y1>=-2 && y1<=2){

glTranslatef(3.0,3.0,0.0);

}

//Start descending the plane

if(y1>=2 && y1<=6.5)

{

glTranslatef(3,3-y1+2,0);

}

// move towards runway

if(y1>=6.5 && y1<=8.2)

{

glTranslatef(3-y1+6.5,3-y1+2,0);

}

// landing only change the x-axis

if(y1>=8.2 && y1<=15)

{

glTranslatef(3-y1+6.5,3-8.2+2,0);

}

// Rotate the plane about its own axis w.r.t y-axis.

if(y1>=-2)

glRotatef(x1,0,1,0);

glPushMatrix();

//Move the plane away from its axis

glTranslatef(1,0,0);

glScaled(0.3,0.3,0.15);

//tilt the plane until its being rotated

if(y1<=8.2)

if(yaxis[0]>=-2)

glRotatef(15,0,0,1);

if(y1<=15){

drawPlane(y1);

}

if(y1>=15 && takeOff){

drawPlane(y1);

}

glPopMatrix();

glPopMatrix();

}

void airport(){

//Floor

glColor3f(0,1,0);

glBegin(GL\_POLYGON);

glVertex3f(-19,-3.5,19);

glVertex3f(-19,-3.5,-19);

glVertex3f(19,-3.5,-19);

glVertex3f(19,-3.5,19);

glEnd();

glPushMatrix();

// runway landing

glPushMatrix();

glColor3f(1,1,1);

glTranslatef(0,-3.5,-1);

glScaled(17,0.1,1);

glutSolidCube(1);

glPopMatrix();

// runway takeoff

glPushMatrix();

glColor3f(1,1,1);

glTranslatef(-0.5,-3.5,4);

glRotatef(-60,0,1,0);

glScaled(11,0.1,1);

glutSolidCube(1);

glPopMatrix();

// runway + parking

glPushMatrix();

glRotatef(-65,0,1,0);

glColor3f(0.1,0.1,0.1);

glTranslatef(3,-3.5,7.8);

glScaled(15.5,0.1,1);

glutSolidCube(1);

glPopMatrix();

//parking place 1

glPushMatrix();

glRotatef(-65,0,1,0);

glColor3f(0.1,0.1,0.1);

glTranslatef(-1,-3.5,7);

glScaled(2.5,0.1,1.5);

glutSolidCube(1);

glPopMatrix();

//parking plane 1

glPushMatrix();

glRotatef(-65,0,1,0);

glTranslatef(3,-2.7,7.4);

glScaled(0.15,0.3,0.15);

drawPlane(16);

glPopMatrix();

//parking place 2

glPushMatrix();

glRotatef(-65,0,1,0);

glColor3f(0.1,0.1,0.1);

glTranslatef(3,-3.5,7);

glScaled(2.5,0.1,1.5);

glutSolidCube(1);

glPopMatrix();

//parking plane 2

glPushMatrix();

glRotatef(-65,0,1,0);

glTranslatef(6.5,-2.7,7.4);

glScaled(0.15,0.3,0.15);

drawPlane(16);

glPopMatrix();

//parking place 3

glPushMatrix();

glRotatef(-65,0,1,0);

glColor3f(0.1,0.1,0.1);

glTranslatef(7,-3.5,7);

glScaled(2,0.1,1.5);

glutSolidCube(1);

glPopMatrix();

//parking plane 3

glPushMatrix();

glRotatef(-65,0,1,0);

glTranslatef(10,-2.7,7.4);

glScaled(0.15,0.3,0.15);

drawPlane(16);

glPopMatrix();

// parking building

glPushMatrix();

glRotatef(-65,0,1,0);

glColor3f(0,0.5,0.5);

glTranslatef(4,-3.5,5.5);

glScaled(14,2.2,1);

glutSolidCube(1);

glPushMatrix();

//glRotatef(15,0,1,0);

glTranslatef(0,0.3,0);

glScaled(0.9,0.3,1);

glColor3f(0.1,0.1,0.1);

glutSolidCube(1);

glPopMatrix();

glPopMatrix();

strip(-2);

strip(0);

strip(2);

strip(4);

strip(6);

strip(8);

strip(10);

// runway Lights

for(float j=-1.3;j<=-0.5;j+=0.8)

{

for(float i=-4.5;i<=1.8;i+=0.9){

glPushMatrix();

glColor3f(1,0,0);

glTranslatef(i,-3.4,j);

//glScaled(6,0.2,1);

glutSolidSphere(0.05,10,10);

glPopMatrix();

}}

glTranslatef(7,0,-4);

// Building

glPushMatrix();

glTranslatef(-2,-3,-2);

glutSolidCube(1);

glPopMatrix();

//Rotating Dish

glPushMatrix();

glColor3f(0,0,1);

glTranslatef(-2,-2.0,-2);

glRotatef(x2/15,0,1,0);

glScaled(0.1,0.3,1);

glutSolidCube(1);

glPopMatrix();

//dish connector

glPushMatrix();

glColor3f(0,1,0);

glTranslatef(-2,-2.5,-2);

glScaled(0.1,0.9,0.1);

glutSolidCube(1);

glPopMatrix();

glPopMatrix();

//Mountain

glPushMatrix();

glColor3f(0,1,0);

glTranslatef(-3,-3,-15);

glScaled(10,4,1);

glutSolidDodecahedron();

glPopMatrix();

}

// Start your Drawing ---Draw pyramid

void controller()

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

glTranslatef(0.0, 0.0, -25.0);

for (int i = 0; i < max; i++) {

if (yaxis[i] >= -5) {

animate(yaxis[i + 1], x[i + 1]);

if (yaxis[i + 1] >= -2 && yaxis[i + 1] <= 6.7) {

x[i + 1] += 0.5; // Further reduced rotation speed

}

if (yaxis[i + 1] <= 0) {

yaxis[i + 1] += 0.01; // Further reduced speed of adding planes

} else if (yaxis[i + 1] >= 0 && yaxis[i + 1] <= 6.7) {

yaxis[i + 1] += 0.005; // Further reduced speed of adding planes

} else if (yaxis[i + 1] >= 6.7 && yaxis[i + 1] <= 15) {

yaxis[i + 1] += 0.01; // Further reduced speed of adding planes

} else if (takeOff && yaxis[i + 1] <= 30) {

yaxis[i + 1] += 0.01; // Further reduced speed of taking off

}

}

}

// Update the first plane's animation logic as well

if(yaxis[0] >= -2 && yaxis[0] <= 6.7) {

x[0] += 0.5; // Further reduced rotation speed

}

if(yaxis[0] <= 0) {

yaxis[0] += 0.01; // Further reduced speed of adding planes

} else if(yaxis[0] >= 0 && yaxis[0] <= 6.7) {

yaxis[0] += 0.005; // Further reduced speed of adding planes

} else if(yaxis[0] >= 6.7) {

yaxis[0] += 0.01; // Further reduced speed of adding planes

}

airport();

x2 += 7.5; // Reduced rotation speed of plane blades

// Increments of the first plane

if(yaxis[0] >= -2 && yaxis[0] <= 6.7) {

x[0] += 1.75; // Reduced rotation speed

}

// Translate the plane

// Conditions to increase or decrease the speed of the first plane

if(yaxis[0] <= 0) {

yaxis[0] += 0.075; // Reduced speed of adding planes

} else if(yaxis[0] >= 0 && yaxis[0] <= 6.7) {

yaxis[0] += 0.03; // Reduced speed of adding planes

} else if(yaxis[0] >= 6.7) {

yaxis[0] += 0.05; // Reduced speed of adding planes

}

glFlush();

glutSwapBuffers();

}

void doInit()

{

/\* Background and foreground color \*/

glClearColor(0.529, 0.808, 0.922, 1.0);

/\* Select the projection matrix and reset it then

setup our view perspective \*/

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(30.0f,(GLfloat)640/(GLfloat)480,0.1f,200.0f);

/\* Select the model view matrix, which we alter with rotatef() \*/

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glClearDepth(2.0f);

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LEQUAL);

glEnable(GL\_COLOR\_MATERIAL);

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

glTranslatef(0.0f,0.0f,-13.0f);

stroke\_output(-2.0, 1.7, "p--> Pyramid Clockwise");

stroke\_output(-2.0, 1.0, "P--> Pyramid Anti Clockwise");

stroke\_output(-2.0, 0.3, "h--> House Clockwise");

stroke\_output(-2.0, -0.4, "H--> House Anti-Clockwise");

stroke\_output(-2.0, -1.1, "q--> quit");

GLfloat mat\_ambient[]={0.0f,1.0f,2.0f,1.0f};

GLfloat mat\_diffuse[]={0.0f,1.5f,.5f,1.0f};

GLfloat mat\_specular[]={5.0f,1.0f,1.0f,1.0f};

GLfloat mat\_shininess[]={50.0f};

glMaterialfv(GL\_FRONT,GL\_AMBIENT,mat\_ambient);

glMaterialfv(GL\_FRONT,GL\_DIFFUSE,mat\_diffuse);

glMaterialfv(GL\_FRONT,GL\_SPECULAR,mat\_specular);

glMaterialfv(GL\_FRONT,GL\_SHININESS,mat\_shininess);

GLfloat lightIntensity[]={3.7f,0.7f,0.7f,1.0f};

GLfloat light\_position[]={0.0f,3.0f,2.0f,0.0f};

glLightfv(GL\_LIGHT0,GL\_POSITION,light\_position);

glLightfv(GL\_LIGHT0,GL\_DIFFUSE,lightIntensity);

glutIdleFunc(controller);

glFlush();

glutSwapBuffers();

}

void menu(int id)

{

switch(id)

{

case 1:max+=1;

break;

case 2:max-=1;

break;

case 3:takeOff=!takeOff;

break;

case 4:exit(0);

break;

}

glFlush();

glutSwapBuffers();

glutPostRedisplay();

}

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

{

if(key=='p')

{

glutIdleFunc(controller);

}

if(key=='f')

{

takeOff=!takeOff;

}

if(key=='a')

{

max+=1;

}

if(key=='r')

{

max-=1;

}

if(key=='q'|| key=='Q')

{

exit(0);

}

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB);

glutInitWindowSize(1000,480);

glutInitWindowPosition(100,100);

glutCreateWindow("Glut Plane");

glutDisplayFunc(display);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glShadeModel(GL\_SMOOTH);

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_NORMALIZE);

glutKeyboardFunc(mykey);

glutCreateMenu(menu);

glutAddMenuEntry("Add Plane 'a'",1);

glutAddMenuEntry("Remove 'r'",2);

glutAddMenuEntry("Takeoff 'f'",3);

glutAddMenuEntry("Quit 'q'",4);

glutAttachMenu(GLUT\_LEFT\_BUTTON);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

doInit();

glutMainLoop();

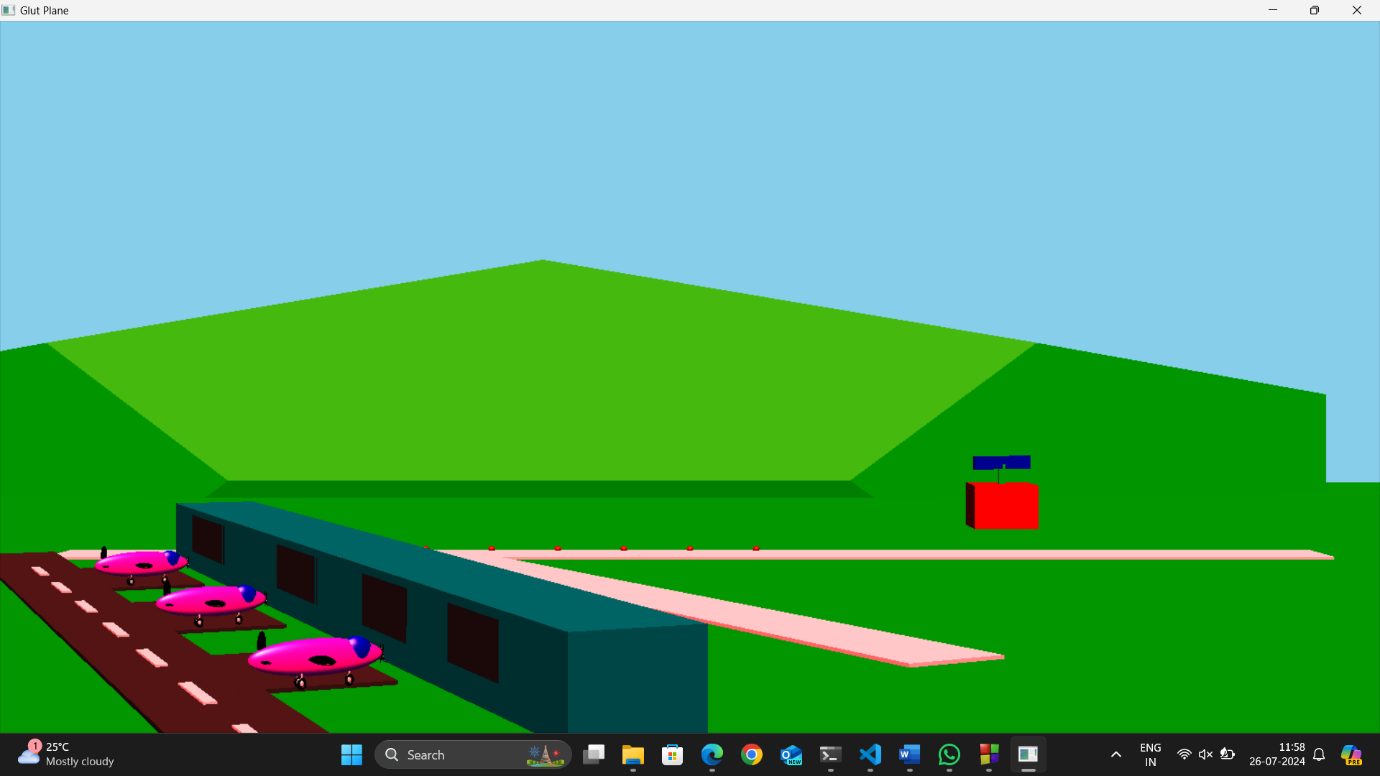
return 0;

}

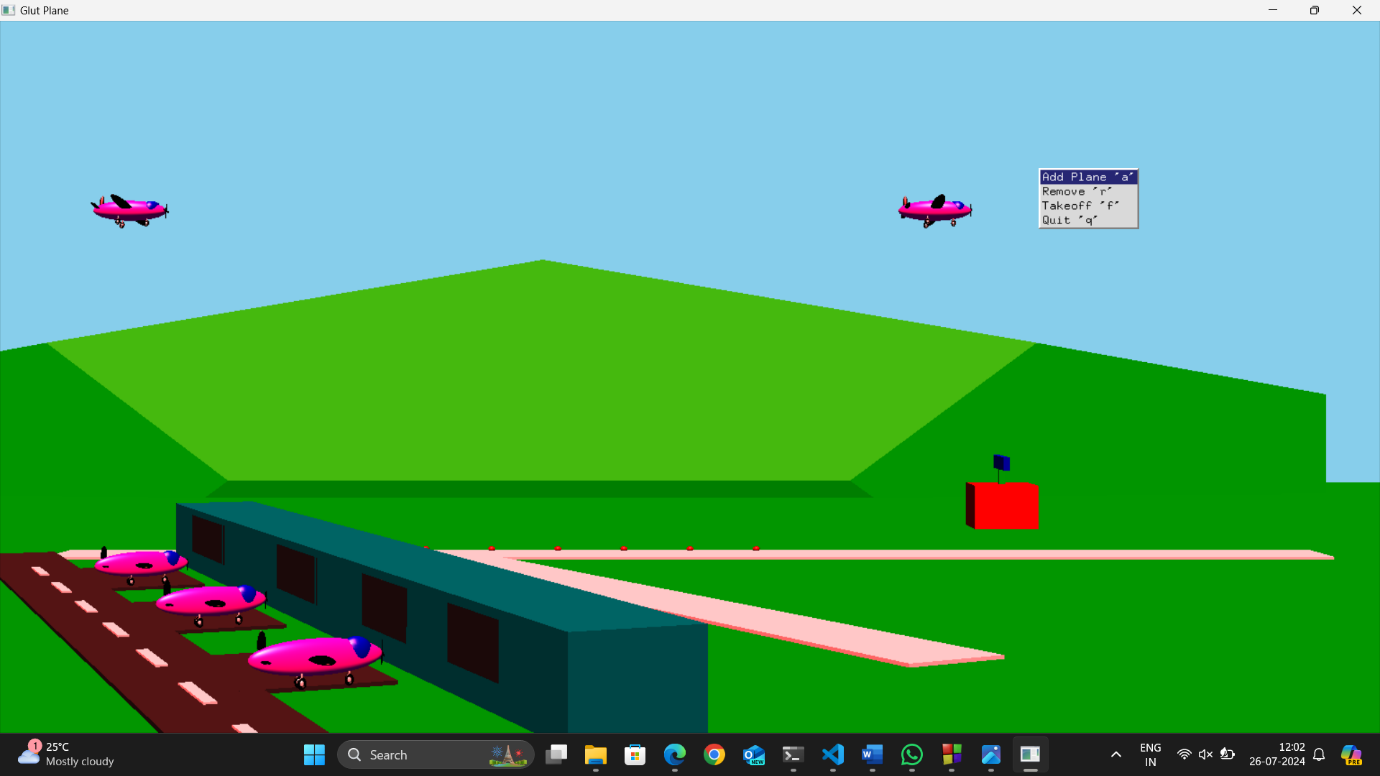
**Chapter 5**

**OUTPUT SNAPSHOTS**

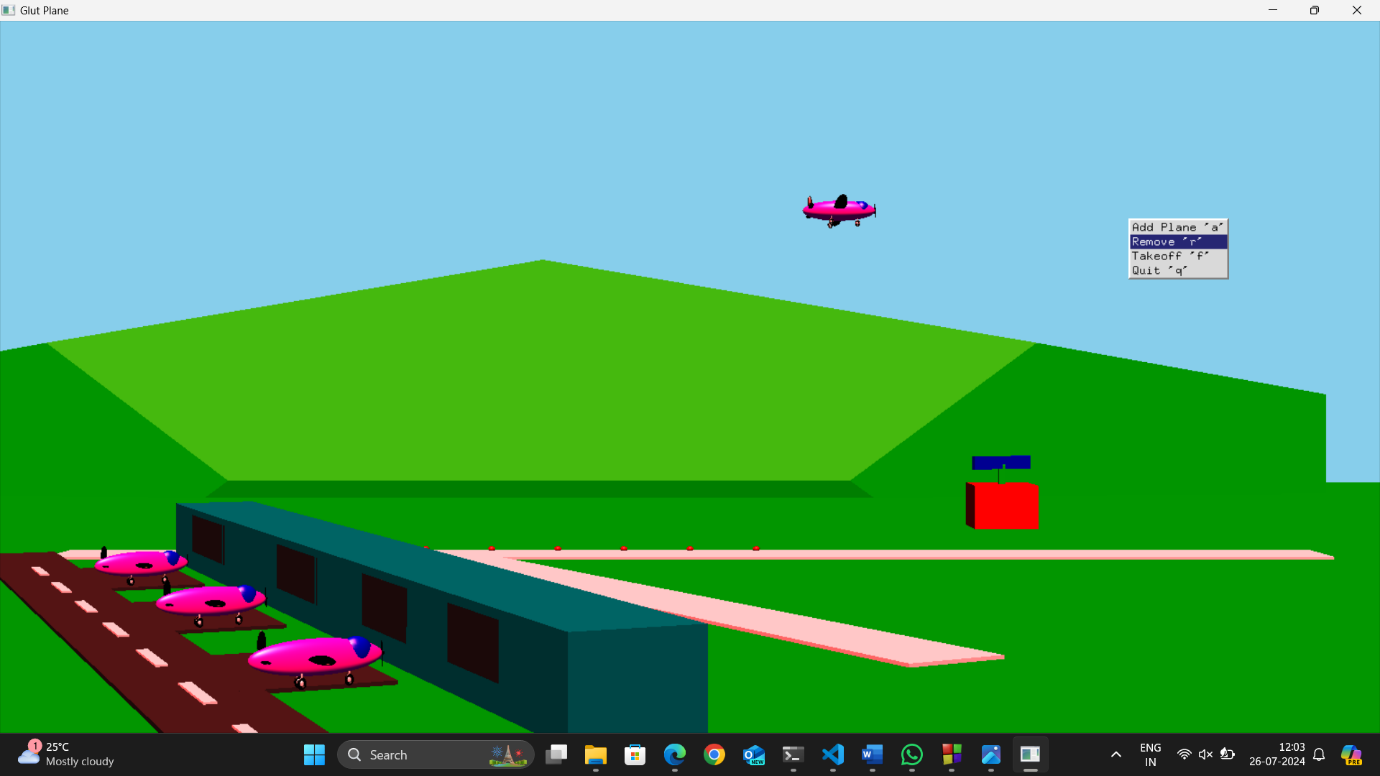
**5.1 Airport View**



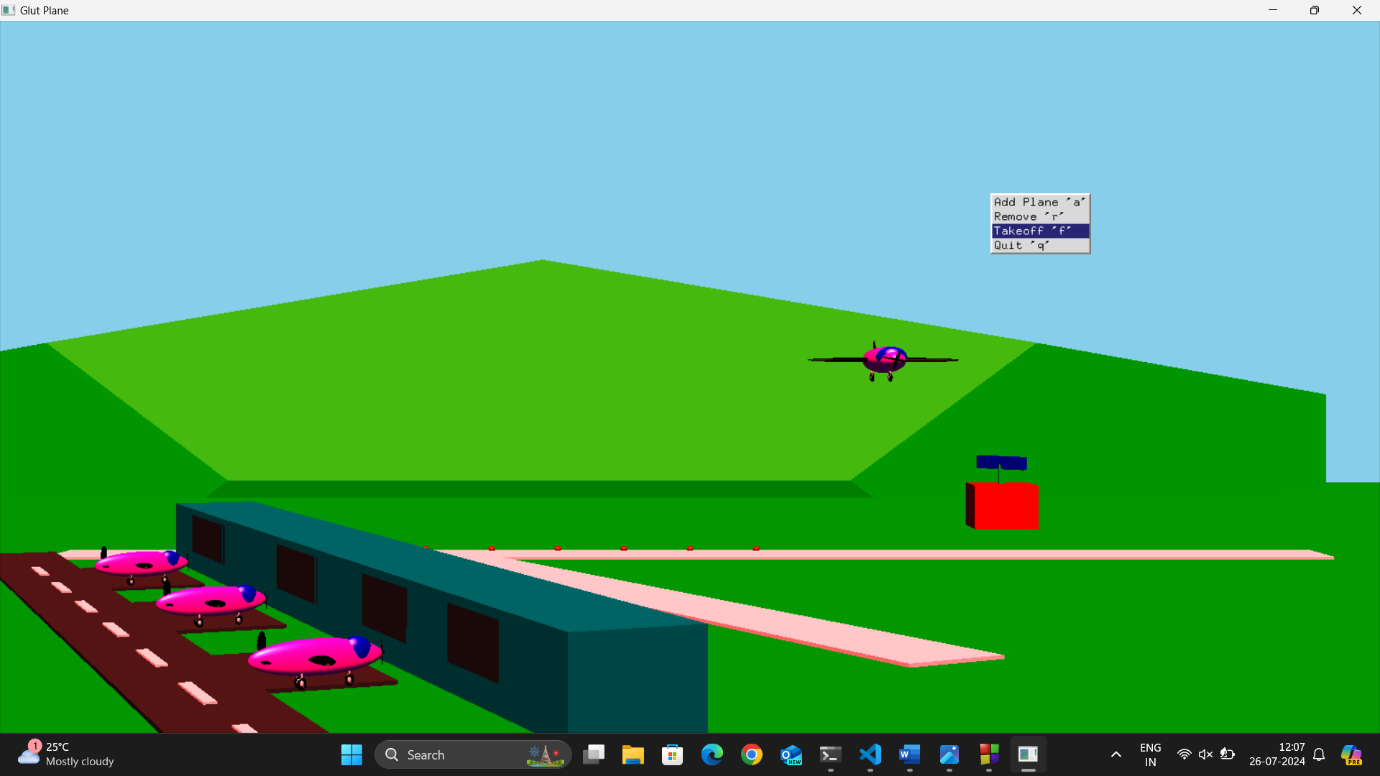
**5.2 Adding of Plane**



**5.3 Removing of Plane**



**5.4 Takeoff of Plane**



**CONCLUSION**

In conclusion, this Mini Project successfully illustrates the concepts and usage of Air Traffic Control (ATC) within an OpenGL environment. By implementing ATC principles, including collision prevention, traffic organization, and support for pilots, the project demonstrates how ground-based controllers direct aircraft both on the ground and through controlled airspace. The interactive nature of the project is enhanced through the use of popular input devices such as the mouse and keyboard, which allow users to efficiently manage air traffic operations. The inclusion of a menu system further enriches the user experience, making the program more engaging and user-friendly. Overall, this project effectively showcases the critical functions of ATC systems and provides a practical, hands-on understanding of air traffic management principles within a simulated environment.

**REFERENCES**

1. <http://google.com>
2. [Free Air Traffic Control CG Project (getsetproject.com)](https://getsetproject.com/info-project.php?id=330&name=Free%20Air%20Traffic%20Control%20CG%20Project)
3. <https://youtu.be/wN8PRvSJ5sk?si=LhObMOww8-ZsE_28>