Computer Graphics (UCS505)

Project on

SKY DANCE
(AN AIR SHOW ANIMATION)

Submitted By:

Pulkit Arora 102103267

Sehajbir Singh Bains 102103290

Geet Inder Singh Sodhi 102103292

3CO10 (Group 11)

B.E. Third Year – COE

Submitted To:

KUNTAL CHOWDHURY

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

Patiala – 147001

<u>Table of Contents</u>

INTRODUCTION TO PROJECT

- Welcome to "Skydance"! Get ready to embark on an exciting journey into the world of aviation, where you'll witness jaw-dropping air shows filled with stunning planes and thrilling maneuvers. Our project is all about bringing the thrill of flying to life through amazing visuals and cool animations.

- In "Skydance," you'll see all sorts of airplanes, from sleek jets to classic biplanes and even helicopters. They're all designed to look just like the real thing, so you'll feel like you're right there in the cockpit. And the sounds? They're just as awesome, with roaring engines and spinning propellers making you feel like you're really at an air show.

- But "Skydance" isn't just about the planes—it's also a tribute to the amazing pilots who fly them. Watch as they pull off incredible stunts and mind-blowing tricks with perfect timing and skill. Whether they're flying solo or performing as part of a team, every pilot in "Skydance" is a true master of the sky.

- As you watch the planes zoom and loop through the air, you'll also learn all about the history of flight and the cool technology that makes it possible. From the very first airplanes built by the Wright brothers to the high-tech jets of today, "Skydance" will take you on a fascinating journey through the past, present, and future of aviation.

- Whether you're a big aviation fan, a curious learner, or just someone who loves awesome animations, "Skydance" has something for you. So buckle up, hold on tight, and get ready to soar through the clouds with "Skydance"! It's going to be an epic adventure you won't forget. Welcome aboard!

COMPUTER GRAPHICS CONCEPTS

In the creation of "Skydance," several fundamental computer graphics concepts were employed to bring the aviation world to life with stunning visuals and engaging animations. Here are some of the key concepts utilized in the project:

1) **3D Modeling:** "Skydance" utilizes 3D modeling to create lifelike representations of airplanes, helicopters, and other objects in the aviation environment. Each aircraft is meticulously designed with accurate proportions, textures, and details to enhance realism and immersion.

2) **Texturing:** Texturing is applied to the 3D models to enhance their appearance by adding surface details such as colors, patterns, and materials. In "Skydance," texturing is used to simulate the metallic sheen of airplane bodies, the glossy finish of cockpit canopies, and the intricate designs of aircraft liveries.

3) **Animation:** Animation plays a crucial role in bringing the aircraft and airshow performances to life in "Skydance." Through techniques such as keyframing and rigging, the planes perform dynamic maneuvers such as loops, rolls, and spins, while the pilots execute precise aerobatic routines with fluid motion and timing.

4) **Lighting and Shading:** Lighting and shading techniques are used to create realistic illumination and shadow effects in "Skydance." Dynamic lighting sources such as the sun and stadium floodlights cast shadows and highlights on the aircraft and environment, enhancing depth perception and visual fidelity.

5) **Rendering:** Rendering is the process of generating the final images or frames from the 3D scene. In "Skydance," rendering techniques such as ray tracing or rasterization are employed to produce high-quality visuals with realistic reflections, shadows, and atmospheric effects, ensuring an immersive viewing experience for the audience.

6) **Transformation:** Transformation concepts, including translation, rotation, and scaling, are integral to positioning and orienting objects within the 3D space of "Skydance." These transformations enable the choreography of aerial maneuvers, the movement of camera perspectives, and the placement of objects throughout the scene, enhancing the overall dynamism and realism of the virtual environment.

USER DEFINED FUNCTIONS

Here is a list of all the user-defined functions used in this project :

- **output(int x, int y, char *string,void *font) :** This function is used to render text on the screen at a specified position.
- **sphere(float r, float g, float b, float a) : Draws** a solid sphere with the specified color and transparency.
- **smoke(float size, float alpha, float R, float G, float B)** : Draws a smoke particle with the specified size, transparency, and color.
- **drawSmoke(float R, float G, float B, float x, float y, int reflect) :** Draws a collection of smoke particles with varying positions, sizes, and transparencies.
- **slab(float r, float g, float b) :** Draws a solid cube with the specified color.
- **bridge() :** Draws a bridge structure using cubes to simulate its components.
- **ground() :** Draws the ground with different colored polygons to represent terrain.
- **wing(int Colour) :** Draws an airplane wing with specified color and geometry.
- **fin(int Colour) :** Draws the fin of an airplane with specified color and geometry.
- **plane(int Colour) :** Draws an airplane with specified color and components.
- **cloud() :** Draws a cloud composed of multiple spheres to simulate its shape.
- **reshape(int width, int height) :** Reshapes the viewport and sets up the perspective projection.

- **display1() :** Renders the main scene of the animation including airplanes, smoke trails, ground, bridges, and clouds.
- **forward(int dir) :** Moves the viewpoint forward or backward based on the direction.
- **straight() :** Controls the straight motion of the viewpoint and updates the display.
- **up() :** Tilts the viewpoint upwards and updates the display.
- **down() :** Tilts the viewpoint downwards and updates the display.
- **mouse(int btn, int state, int x, int y) :** Handles mouse button events for initiating or stopping rotational translation.
- **front() :** Displays the introductory information and credits on the screen.
- **menu() :** Displays the main menu options on the screen.
- **help() :** Displays help information and controls for the animation.
- **intro() :** Displays an introduction to the airshow animation.
- **menuset() :** Sets up the menu display environment.
- **keyboardFunc(unsigned char key, int x, int y) :** Handles keyboard input for navigation within the application.
- **ground()** : generates the terrain surface for the "Skydance" project, providing a realistic foundation for aerial performances and incorporating visual details such as textures and collision detection.
- **cloud() :** creates clouds in the sky, adding an atmospheric element to the animation. The function positions and scales multiple spheres to represent cloud formations, enhancing the overall visual appeal of the scene.

CODE

```
#include<stdlib.h>
#include <GL/glut.h>
#include <math.h>
#include<string.h>
static double id = 0;
int s=0;
//Camera position
static double cx = 100;
static double cy = 50;
static double cz = 100;
//Rotation
static int spinx = 0;
static int spiny = 0;
static int spinz = 0;
//Position
static double x = 0.0;
static double y = 0.0;
static double z = 0.0;
//Smoke particle rotation
static int spinxs[150];
static int spinys[150];
static int spinzs[150];
//smoke  particle position
float sx[150];
float sy[150];
float sz[150];
float sa[150];
float ss[150];
//Count
int i = 0,f=0;
void output(int x, int y, char *string,void *font)
{
  int len, i;
  glRasterPos2f(x, y);
  len = (int) strlen(string);
  for (i = 0; i < len; i++) {
    glutBitmapCharacter(font, string[i]);
  }
}
//Sphere
void sphere(float r, float g, float b, float a)
{
  glColor4f(r,g,b,a);
  glutSolidSphere(4,32,32);
}
void smoke(float size, float alpha, float R, float G, float B)
```

```
{
  glPushMatrix();
  //Colour and transparency of Smoke
  glColor4f(R,G,B,alpha);
  glTranslatef(0,0,-15);
  glutSolidSphere((1 + size),16,16);
  glPopMatrix();
}
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();
  }
}
void slab(float r, float g, float b)
{
  glColor3f(r,g,b);
  glutSolidCube(1);
}
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 ground()
{
  glBegin(GL_POLYGON);
  glColor3f(0.0,0.8,0.0);
  glVertex3f(-100,0,100);
  glVertex3f(0,0,100);
  glVertex3f(-30,0,50);
  glVertex3f(0,0,0);
  glVertex3f(-50,0,-100);
  glVertex3f(-100,0,-100);
  glEnd();
  glBegin(GL_POLYGON);
  glColor3f(0,0.8,0.0);
  glVertex3f(10,0,0);
  glVertex3f(-20,0,50);
  glVertex3f(10,0,100);
  glVertex3f(100,0,100);
  glVertex3f(100,0,-100);
  glVertex3f(-40,0,-100);
  glEnd();
  glBegin(GL_POLYGON);
  glColor3f(0.0,0.128,0.0);
  glVertex3f(0,-10,100);
  glVertex3f(-30,-10,50);
  glVertex3f(-30,0,50);
  glVertex3f(0,0,100);
  glEnd();

  glBegin(GL_POLYGON);
  glColor3f(0.0,0.128,0.0);
  glVertex3f(-30,-10,50);
  glVertex3f(0,-10,0);
  glVertex3f(0,0,0);
  glVertex3f(-30,0,50);
  glEnd();
  glBegin(GL_POLYGON);
  glColor3f(0.4,0.2,0.16);
  glVertex3f(0,-10,0);
  glVertex3f(-50,-10,-100);
  glVertex3f(-50,0,-100);
  glVertex3f(0,0,0);
  glEnd();
  glBegin(GL_POLYGON);
  glColor3f(0.4,0.25,0.16);
  glVertex3f(-40,-10,-100);
  glVertex3f(10,-10,0);
  glVertex3f(10,0,0);
  glVertex3f(-40,0,-100);
  glEnd();
  glBegin(GL_POLYGON);
  glColor3f(0.4,0.25,0.16);
  glVertex3f(10,-10,0);
  glVertex3f(-20,-10,50);
  glVertex3f(-20,0,50);
  glVertex3f(10,0,0);
  glEnd();
  glBegin(GL_POLYGON);
  glColor3f(0.4,0.25,0.16);
  glVertex3f(-20,-10,50);
  glVertex3f(10,-10,100);
  glVertex3f(10,0,100);
  glVertex3f(-20,0,50);
  glEnd();
//Water
  glBegin(GL_POLYGON);
  glColor3f(0.0, 0.4, 0.6);
  glVertex3f(-100,-2,100);
  glVertex3f(100,-2,100);
  glVertex3f(100,-2,-100);
  glVertex3f(-100,-2,-100);
  glEnd();
}
//Wings
void wing(int Colour)
{
  float rs=1; //Side red
  float re=0;     //Edge red
  float bs=0;     //Side blue
  float be=0;     //Edge blue
  float gs=0.5;   //Side green
  float ge=0;     //Edge green
  if (Colour == 1){
          rs = 0.75;
          re = 0.5;
  } else if (Colour == 2){
          bs = 0.75;
          be = 0.5;
  } else if (Colour == 3){
          bs = 0.75;
          be = 0.5;
  }
//Front
  glBegin(GL_POLYGON);
          glColor3f(re,ge,be);
          glVertex3f(1.5,-1,10);
          glVertex3f(25,-0.25,0);
          glVertex3f(25,0.25,0);
          glVertex3f(1.5,1,10);
  glEnd();
//Back
  glBegin(GL_POLYGON);
          glColor3f(re,ge,be);
          glVertex3f(1.5,1,-1);
          glVertex3f(25,0.25,-7);
          glVertex3f(25,-0.25,-7);
          glVertex3f(1.5,-1,-1);
  glEnd();
//Top
  glBegin(GL_POLYGON);
          glColor3f(rs,gs,bs);
          glVertex3f(1.5,1,10);
          glVertex3f(25,0.25,0);
          glVertex3f(25,0.25,-7);
          glVertex3f(1.5,1,-1);
  glEnd();
//Bottom
  glBegin(GL_POLYGON);
          glColor3f(rs,gs,bs);
          glVertex3f(1.5,-1,-1);
          glVertex3f(25,-0.25,-7);
          glVertex3f(25,-0.25,0);
          glVertex3f(1.5,-1,10);
  glEnd();
//End
  glBegin(GL_POLYGON);
          glColor3f(re,ge,be);
          glVertex3f(25,-0.25,0);
          glVertex3f(25,-0.25,-7);
          glVertex3f(25,0.25,-7);
          glVertex3f(25,0.25,0);
  glEnd();
}
//Fin
void fin(int Colour)
{
  float rs=1;
  float re=0;
  float bs=0;
  float be=0;
  float gs=0.5;
  float ge=0;

  if (Colour == 1){
          rs = 0.75;
          re = 0.5;
  } else if (Colour == 2){
          bs = 0.75;
          be = 0.5;
  } else if (Colour == 3){
          bs = 0.75;
          be = 0.5;
  }
//Front
  glBegin(GL_POLYGON);
          glColor3f(re,ge,be);
          glVertex3f(-0.5,2,-7);
          glVertex3f(0.5,2,-7);
          glVertex3f(0,9.5,-10);
  glEnd();
//Back
  glBegin(GL_POLYGON);
          glColor3f(re,ge,be);
          glVertex3f(0,9.5,-12);
          glVertex3f(0.5,2,-11);
          glVertex3f(-0.5,2,-11);
  glEnd();
//Side A
  glBegin(GL_POLYGON);
          glColor3f(rs,gs,bs);
          glVertex3f(0.5,2,-7);
          glVertex3f(0.5,2,-11);
          glVertex3f(0,9.5,-12);
          glVertex3f(0,9.5,-10);
  glEnd();
//Side B
  glBegin(GL_POLYGON);
          glColor3f(rs,gs,bs);
          glVertex3f(0,9.5,-10);
          glVertex3f(0,9.5,-12);
          glVertex3f(-0.5,2,-11);
          glVertex3f(-0.5,2,-7);
  glEnd();
}
void drawSun() {
   // Draw the sun
   glColor3f(1.0, 1.0, 0.0); // Yellow color for the sun
   glPushMatrix();
   glTranslatef(50, 50, -150); // Adjust position of the sun
   glutSolidSphere(10, 20, 20); // Draw a simple solid sphere as the sun
   glPopMatrix();
}
void plane(int Colour)
{
  float rb=0.5;
  float gb=0.5;
  float bb=0.5;
  //Selects which color plane to display
  if (Colour == 1){
            rb = 0.5;
  } else if (Colour == 2){
            gb = 1.0,rb=1.0,bb=1.0;
  } else if (Colour == 3){
            bb = 0.65;
  }
  //Body
  glPushMatrix();
  glTranslatef(0,0,4);
  glScaled(1,0.8,5);
  sphere(rb,gb,bb,1);
  glPopMatrix();
  //Windscreen
  glPushMatrix();
  glTranslatef(0,2,16);
  glScaled(0.5,0.4,0.75);
  sphere(0,0,0,0.75);
  glPopMatrix();
  //Left Wing
  wing(Colour);
  //Right Wing
  glPushMatrix();
  glScalef(-1,-1,1);
  wing(Colour);
  glPopMatrix();
  //Left mini wing
  glPushMatrix();
  glScalef(0.4,0.4,0.4);
  glTranslatef(0,3,-25);
  wing(Colour);
  glPopMatrix();
  //Right mini wing
  glPushMatrix();
  glScalef(-0.4,-0.4,0.4);
  glTranslatef(0,-3,-25);
  wing(Colour);
  glPopMatrix();
  //Fin
  fin(Colour);
```

```
}
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();
}
/* reshape callback function
   executed when window is moved or resized */
void reshape(int width, int height)
{
   glViewport(0, 0, width, height);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(50.0,1.0,15.0,600.0);  //Perspective
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}
/* display routine this where the drawing takes place  */
void display1(void)
{
   glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);   /* clear window */
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();

  gluLookAt(cx, cy, cz, x/2, y/2, z/2, 0.0, 1.0, 0.0);  //position of the eye
//Calculate position and rotation of current smoke particle
```

```
sx[i] = x;
sy[i] = y;
sz[i] = z;
spinxs[i] = spinx;
spinys[i] = spiny;
spinzs[i] = spinz;
sa[i] = 1;
ss[i] = 0;
  //Bridge
for (int b = 0; b < 600; b=b+20)
{
glPushMatrix();
glTranslatef(-300+b,-47,0);
bridge();
glPopMatrix();
}
//Bridge Reflection
glPushMatrix();
glScalef(-1,-1,1);
glTranslatef(0,56,0);
bridge();
glPopMatrix();
glPushMatrix();
glScalef(-1,-1,1);
glTranslatef(-20,56,0);
bridge();
glPopMatrix();
//Ground
glPushMatrix();
glScalef(3,1,3);
glTranslatef(0,-50,0);
ground();
glPopMatrix();
//yellow plane
glPushMatrix();
glScalef(0.5,0.5,0.5);
glTranslatef(x-120,y,z);
glRotatef(spinx,1,0,0);
glRotatef(spiny,0,1,0);
glRotatef(spinz,0,0,1);
plane(1);
glPopMatrix();
glPushMatrix();
glScalef(0.5,0.5,0.5);
glTranslatef(x+120,y,z);
glRotatef(spinx,1,0,0);
glRotatef(spiny,0,1,0);
glRotatef(spinz,0,0,1);
plane(1);
```

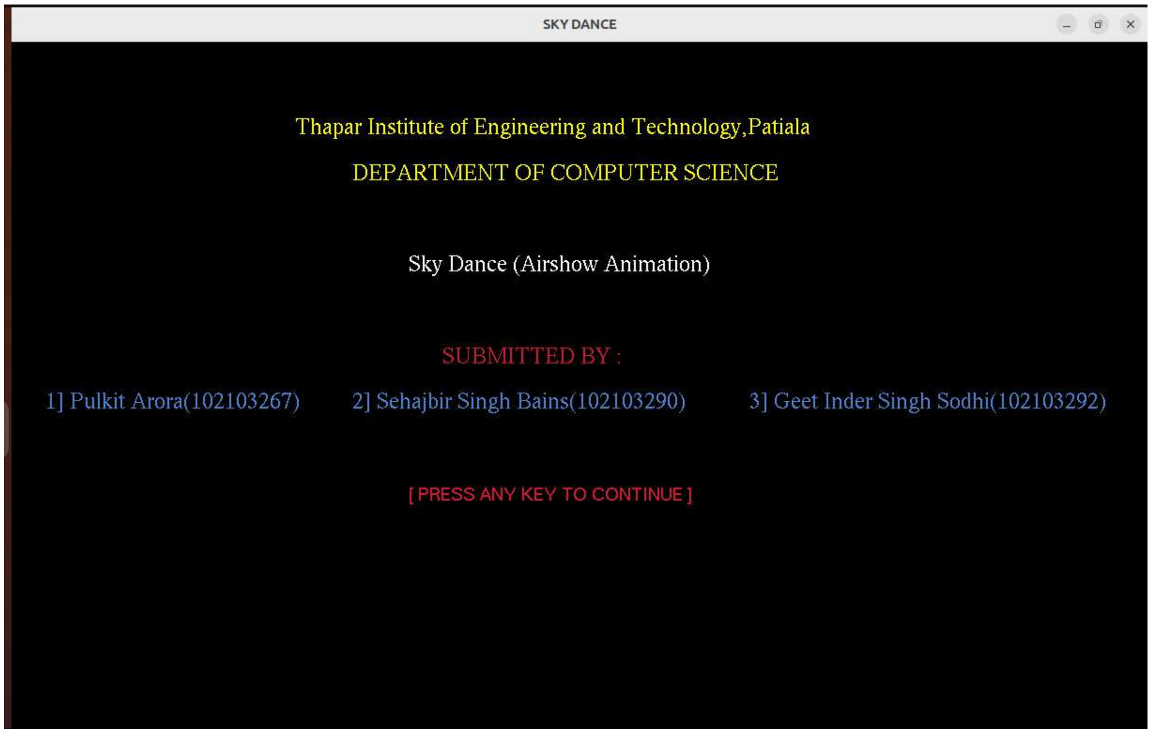
```
  glPopMatrix();
  //Red plane
  glPushMatrix();
  glScalef(0.5,0.5,0.5);
  glTranslatef(x-60,y,z);
  glRotatef(spinx,1,0,0);
  glRotatef(spiny,0,1,0);
  glRotatef(spinz,0,0,1);
  plane(1);
  glPopMatrix();
//Green plane
  glPushMatrix();
  glScalef(0.5,0.5,0.5);
  glTranslatef(x,y,z);
  glRotatef(spinx,1,0,0);
  glRotatef(spiny,0,1,0);
  glRotatef(spinz,0,0,1);
  plane(1);
  glPopMatrix();
//Blue Plane
  glPushMatrix();
  glScalef(0.5,0.5,0.5);
  glTranslatef(x+60,y,z);
  glRotatef(spinx,1,0,0);
  glRotatef(spiny,0,1,0);
  glRotatef(spinz,0,0,1);
  plane(1);
  glPopMatrix();
//Smoke trails for planes
  drawSmoke(1,0.5,0.2,120,0,1);
  drawSmoke(1,0.5,0.2,60,0,1);
  drawSmoke(1,1,1,0,0,1);
  drawSmoke(0,0.64,0,-60,0,1);
  drawSmoke(0,0.64,0,-120,0,1);
//Increase count
		i++;
//Reset count
   if (i > 149) i = 0;
//Clouds
  glPushMatrix();
  glTranslatef(-40,40,0);
  glScalef(2,1.5,2);
  cloud();
  glPopMatrix();
  glPushMatrix();
  glTranslatef(20,30,0);
  glScalef(1.5,1,1.5);
  cloud();
  glPopMatrix();
  glPushMatrix();
  glTranslatef(0,50,-70);
  glRotatef(45,0,1,0);
  glScalef(1,1,1);
  cloud();
  glPopMatrix();
  glPushMatrix();
  glTranslatef(30,50,70);
  glRotatef(45,1,0,0);
  glScalef(0.5,0.5,0.5);
  cloud();
  glPopMatrix();
  glPushMatrix();
  glTranslatef(-70,50,70);
  glRotatef(35,0,1,0);
  glScalef(2,1,2);
  cloud();
  glPopMatrix();
  glPushMatrix();
  glTranslatef(-70,10,70); // Adjust position of the sun
  drawSun();
  glPopMatrix();

  glutSwapBuffers();
  glFlush();
}
/* graphics initialisation */
void init(void)
{
   glClearColor(0.0,0.0,0.0,0);   /* window will be cleared to sky blue
*/
   glEnable(GL_DEPTH_TEST);
   //Enable Alpha channel
   glEnable(GL_BLEND);
   glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
   glAlphaFunc(GL_GREATER, 0./255.);
   glEnable(GL_CULL_FACE);  // Enable back  culling
   glCullFace(GL_BACK);     // Cull back faces
}
void forward(int dir)
{
   float xrad; float yrad; float zrad;
            xrad = (spinx * 3.141592654)/180;
            yrad = (spiny * 3.141592654)/180;
            zrad = (spinz * 3.141592654)/180;
            if (spinx <90)
            {
                    z = z +(dir*(cos(yrad)));
            }
            else if (spinx == 90 || spinx == 270)
            {
                    z = z;
            }
            else if(spinx <270)
            {
                    z = z - (dir*(cos(yrad)));
            }
            else
            {
                    z = z + (dir*(cos(yrad)));
            }
            x = x + (dir*(sin(yrad)));
            if (spiny <90)
            {
                    y = y - (dir*(sin(xrad)));
            } else if (spiny == 90 || spiny == 270)
            {
                    y = y;
            }else if(spiny <270)
            {
                    y = y + (dir*(sin(xrad)));
            }else
            {
                    y = y - (dir*(sin(xrad)));
            }
}
void straight()
{
  if (spinz >= 360)
  {
          spinz = 0;
  }
  if(id==1)
        spinz = (spinz + 10) % 360;
  if(s==1)
  forward(4);
  glutPostRedisplay();
}
void up()
{
  if (spinx >= 360)
     {
          spinx = 0;
     }
        spinx = (spinx -1) % 360;
  forward(4);
  glutPostRedisplay();
}
```

```
void down()

{
  if (spinx >= 360) {
            spinx = 0;
     }
        spinx = (spinx +1) % 360;
  forward(4);
  glutPostRedisplay();
}
void mouse(int btn, int state, int x, int y)
{
  switch(btn)
   {
  case GLUT_LEFT_BUTTON:
           if (state == GLUT_DOWN)
           {        id=1;
                    glutIdleFunc(straight);
                    break;
           }

  case GLUT_RIGHT_BUTTON:
           if (state == GLUT_DOWN)
           {        id=0;
                    glutIdleFunc(straight);
                    break;
           }
  }
}
void *fonts[]=
{
GLUT_BITMAP_9_BY_15,
GLUT_BITMAP_TIMES_ROMAN_10,
GLUT_BITMAP_TIMES_ROMAN_24,
GLUT_BITMAP_HELVETICA_18,
GLUT_BITMAP_HELVETICA_12
};
void front()
{
  glColor3f(1.0,1.0,0.0);
  output(250,650,"Thapar Institute of Engineering and Technology,Patiala",fonts[2]);
  glColor3f(1.0,1.0,0.0);
  output(300,600,"DEPARTMENT OF COMPUTER SCIENCE",fonts[2]);
  //glColor3f(0.8,0.1,0.2); red
  glColor3f(1.0,1.0,1.0);
  output(350,500,"Sky Dance (Airshow Animation)",fonts[2]);
  glColor3f(0.8,0.1,0.2);
  output(380,400,"SUBMITTED BY :",fonts[2]);
  glColor3f(0.3,0.5,0.8);
  output(30,350,"1] Pulkit Arora(102103267)",fonts[2]);
  output(300,350,"2] Sehajbir Singh Bains(102103290)",fonts[2]);
  output(650,350,"3] Geet Inder Singh Sodhi(102103292)",fonts[2]);
  glColor3f(0.8, 0.1, 0.2); // Red color
//          glColor3f(0.6,0.25,0.0);
  output(350,250,"[ PRESS ANY KEY TO CONTINUE ]",fonts[3]);
}
void menu()
{
  glColor3f(0.8,0.1,0.2);
  output(200,520,"GRAPHICAL IMPLEMENTATION OF AIRSHOW",fonts[2]);
  glColor3f(0.8,0.1,0.2);
  output(250,480,"Sky Dance (Airshow Animation)",fonts[2]);
  glColor3f(0.0,0.6,0.3);
  output(300,400,"SELECT AN OPTION",fonts[2]);
  output(300,380,"-----------------",fonts[2]);
  glColor3f(0.3,0.5,0.8);
  output(300,340,"[1] PROCEED",fonts[3]);
  output(300,300,"[2] HELP",fonts[3]);
  output(300,260,"[3] INTRODUCTION",fonts[3]);
  output(300,220,"[b] BACK",fonts[3]);
  output(300,180,"[q] QUIT",fonts[3]);
  glColor3f(0.5,0.2,0.6);
}
void help()
{
  glColor3f(0.8,0.1,0.2);
  output(170,600,"GRAPHICAL IMPLEMENTATION OF AIRSHOW",fonts[2]);
  glColor3f(0.0,0.6,0.3);
  output(180,560,"=> RIGHT CLICK MOUSE TO STOP THE ROTATIONAL TRANSLATION
<=",fonts[3]);
  output(180,520,"=> LEFT CLICK MOUSE FOR ROTATION TRANSLATION <=",fonts[3]);
  output(180,480,"=> [U] FOR MOVEUP <=",fonts[3]);
  output(180,440,"=> [D] FOR MOVEDOWN <=",fonts[3]);
  output(180,400,"=> [P] FOR PAUSE <=",fonts[3]);
  output(180,360,"=> [S] FOR START <=",fonts[3]);
  output(180,320,"=> [R] FOR RESET <=",fonts[3]);
  glColor3f(0.3,0.5,0.8);
  output(400,280,"SELECT AN OPTION",fonts[2]);
  output(400,265,"-----------------",fonts[2]);
  output(400,230,"[h] HOME",fonts[3]);
  output(400,190,"[b] BACK",fonts[3]);
  output(400,150,"[q] QUIT",fonts[3]);
  glColor3f(0.5,0.2,0.6);
  output(600,60,"[ Dept. of CSE, TIET ]",fonts[0]);
}
void intro()
{
  glColor3f(1.0, 0.5, 0.0); // Orange color (higher red and green, lower blue)
  output(170,480,"GRAPHICAL IMPLEMENTATION OF AIRSHOW",fonts[2]);
  glColor3f(0.0,0.6,0.3);
  output(160,430,"IN THIS AIRSHOW A GROUP OF AIRPLANE EMITS COLORFUL
SMOKE",fonts[0]);

  output(160,400,"    WHICH PAINTS THE SKY IN TRICOLOR(INDIAN FLAG).",fonts[0]);
  glColor3f(0.3,0.5,0.8);
  output(400,300,"SELECT AN OPTION",fonts[2]);
  output(400,270,"-----------------",fonts[2]);
  output(400,230,"[h] HOME",fonts[3]);
  output(400,190,"[b] BACK",fonts[3]);
  output(400,150,"[q] QUIT",fonts[3]);

  glColor3f(0.5,0.2,0.6);
  output(600,60,"[ COMPUTER SCIENCE DEPARTMENT]",fonts[0]);
}
void menuset()
{
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0, 1000, 0.0, 750,-2000,1500);
  glMatrixMode(GL_MODELVIEW);
  glClear( GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);
}
void display()
{
  if(f==0)
  {
          menuset();
          front();

          glutSwapBuffers();
  }
  else if(f==1)
  {
     menuset();
          menu();
          glutSwapBuffers();
  }
  else if(f==3)
  {
          menuset();
          help();
          glutSwapBuffers();
  }
  else if(f==4)
  {
          menuset();
```

```
                intro();
            glutSwapBuffers();
        }
        else
        {
            glClearColor(0.45,0.8,0.88,0.0);
                display1();
        }
}
void keyboardFunc( unsigned char key, int x, int y )
{
        if(f==0)
                f=1;
        else if(f==1)
        {
                switch(key)
                {
                case '1':f=2;break;
                case '2':f=3;break;
                case '3':f=4;break;
                case 'b':
                case 'B':f=0;break;
                case 'q':
                case 'Q':exit(0);
                }
        }
        else if(f==2)
        {
                switch(key)
                {
                case 'q':
                case 'Q':exit(0);break;
                case 'b':
                case 'B':f=1;break;
                case 'h':
                case 'H':f=0;break;
                case 'r':
                case 'R':
            spinx=0.0;
                        spiny=0.0;
                        spinz=0.0;
            x = 0.0;
                y = 0.0;
                        z = 0.0;
                        for (i = 0; i<150; i++)
                        {
                                sx[i] = x;
                                sy[i] = y;
                                sz[i] = z;
```

```
                              spinxs[i] = spinx;
                              spinys[i] = spiny;
                              spinzs[i] = spinz;
                              sa[i] = 1;
                              ss[i] = 0;
                    }
                    glutIdleFunc(NULL);
           glutPostRedisplay();
      break;
                    case 'p':
                    case 'P':
                    s=0;
                 spinx=0.0;
                    spiny=0.0;
                    spinz=0.0;
           glutIdleFunc(NULL);
           glutPostRedisplay();
      break;
           case'u':
                    case 'U':
        up();
                              break;
                    case 'D':
                    case 'd':
                    down();
                    break;
                    case 'S':
                    case 's':s=1;
                              glutIdleFunc(straight);
                              break;            }
}
else if(f==3)
{
          switch(key)
          {
          case 'b':
          case 'B':f=1;break;
          case 'h':
          case 'H':f=0;break;
          case 'q':
          case 'Q':exit(0);
          }
}
else
{         switch(key)
          {
          case 'b':
          case 'B':f=1;break;
          case 'h':
            case 'H':f=0;break;
            case 'q':
            case 'Q':exit(0);
            }
  }
  reshape( 1400,700 );
   glutPostRedisplay( );
}
int main(int argc, char** argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
  glEnable(GL_DEPTH_TEST);
  glutInitWindowSize (1400, 700);
  glutInitWindowPosition (0, 0);
  glutCreateWindow ("SKY DANCE");
  init();
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutMouseFunc(mouse);
  glutKeyboardFunc(keyboardFunc);
  glutMainLoop();
  return 0;
}
```

OUTPUT / SCREENSHOTS


SKY DANCE

Thapar Institute of Engineering and Technology,Patiala

DEPARTMENT OF COMPUTER SCIENCE

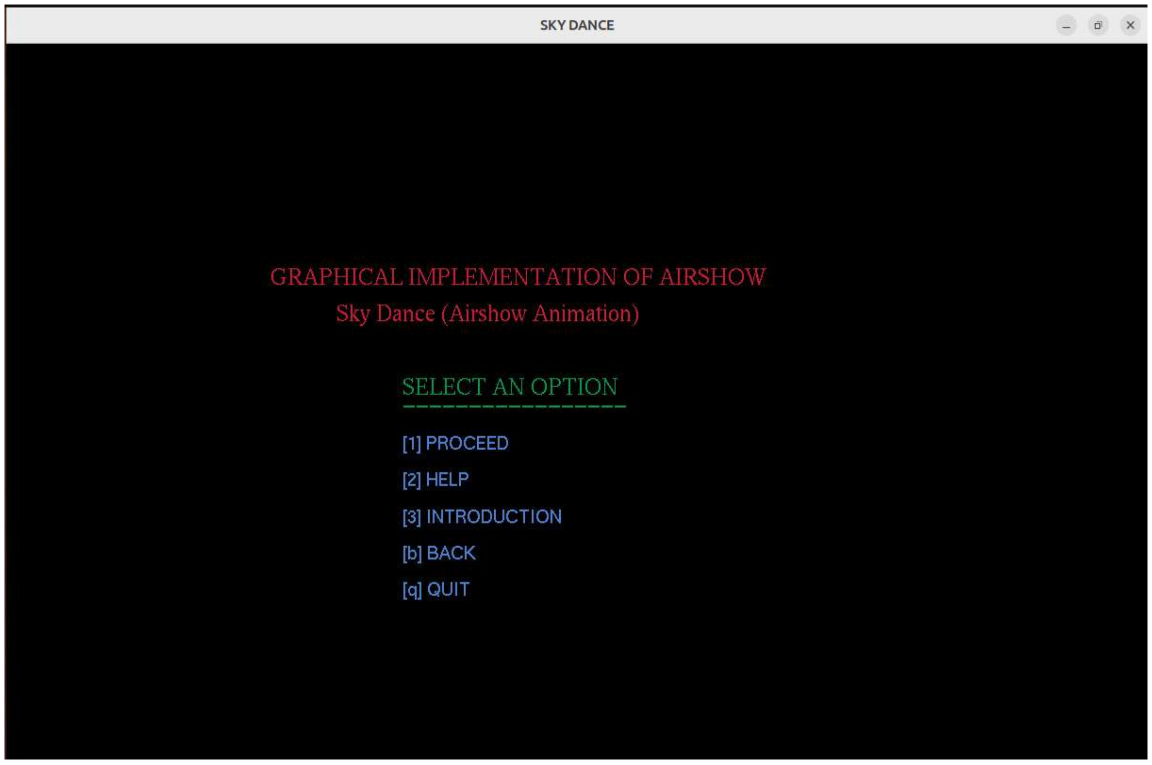
Sky Dance (Airshow Animation)

SUBMITTED BY :

1] Pulkit Arora(102103267)   2] Sehajbir Singh Bains(102103290)   3] Geet Inder Singh Sodhi(102103292)

[PRESS ANY KEY TO CONTINUE]

Fig 1


SKY DANCE

GRAPHICAL IMPLEMENTATION OF AIRSHOW

Sky Dance (Airshow Animation)

SELECT AN OPTION

[1] PROCEED

[2] HELP

[3] INTRODUCTION

[b] BACK

[q] QUIT

Fig 2

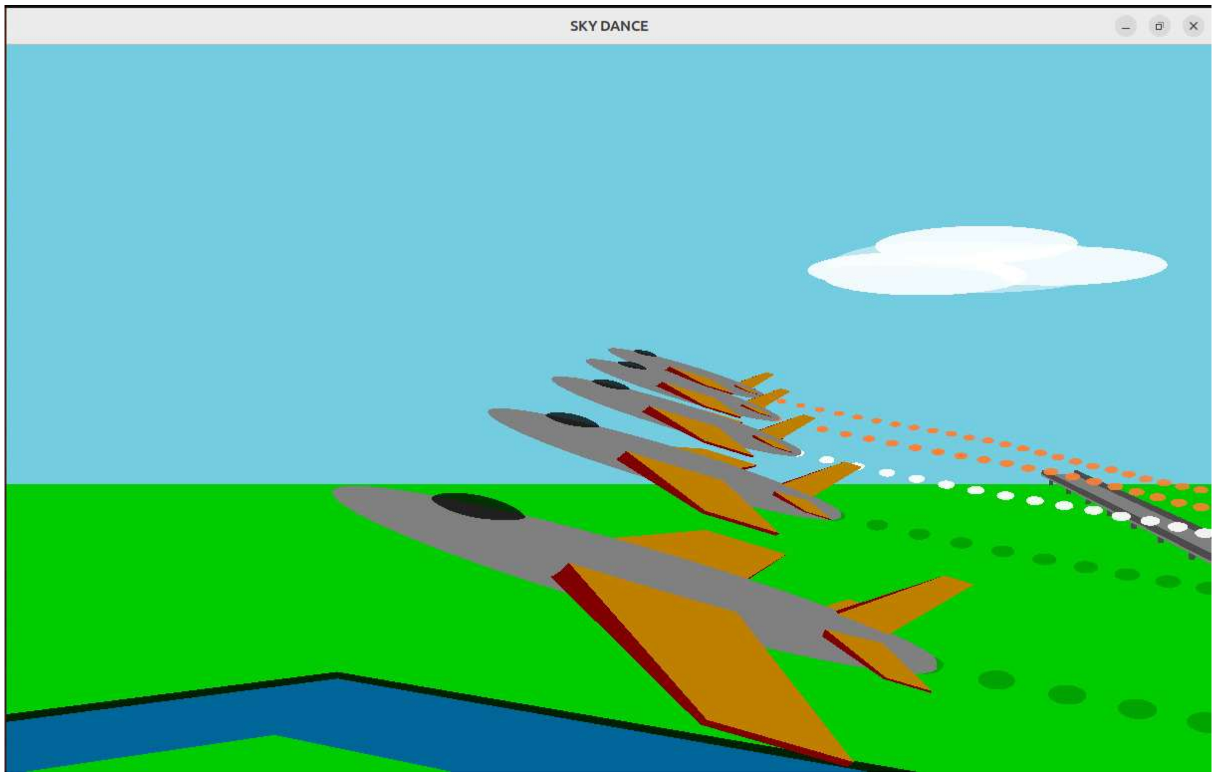


Fig 3

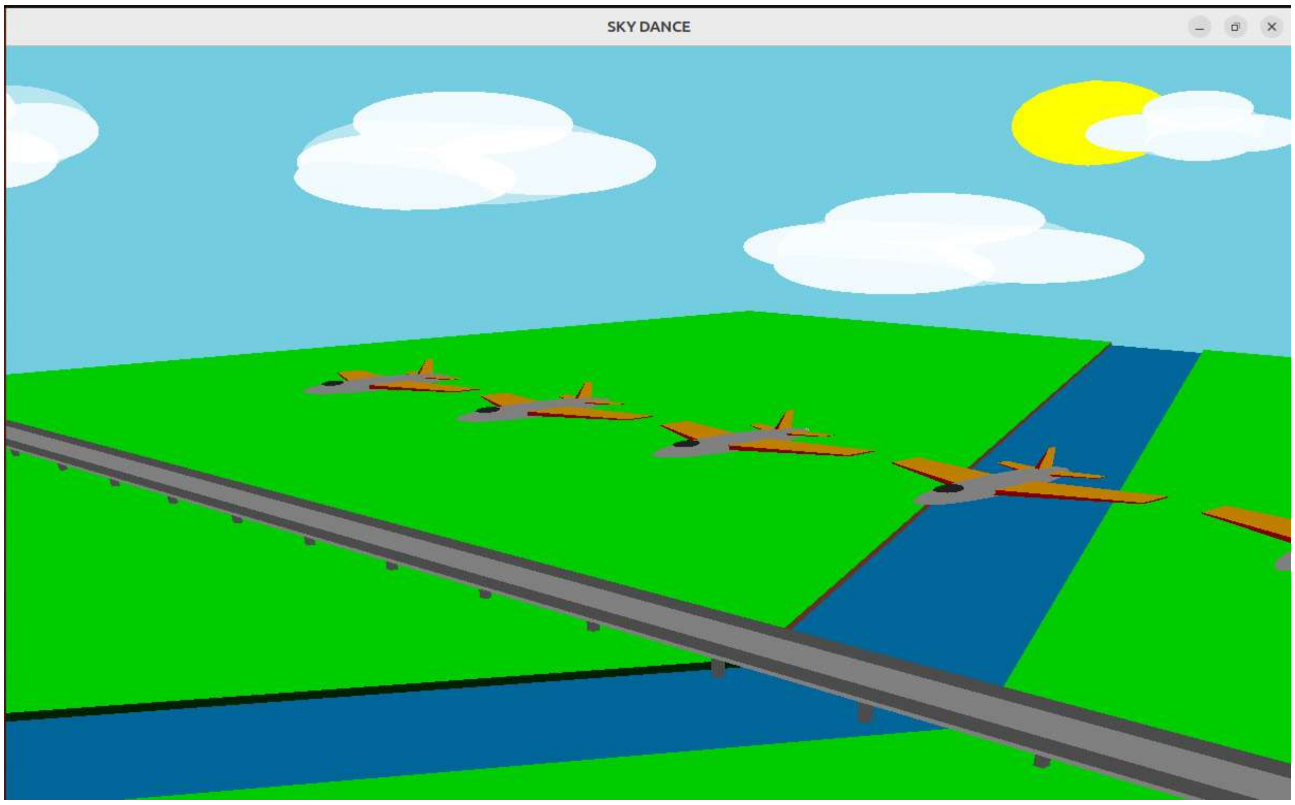


Fig 4

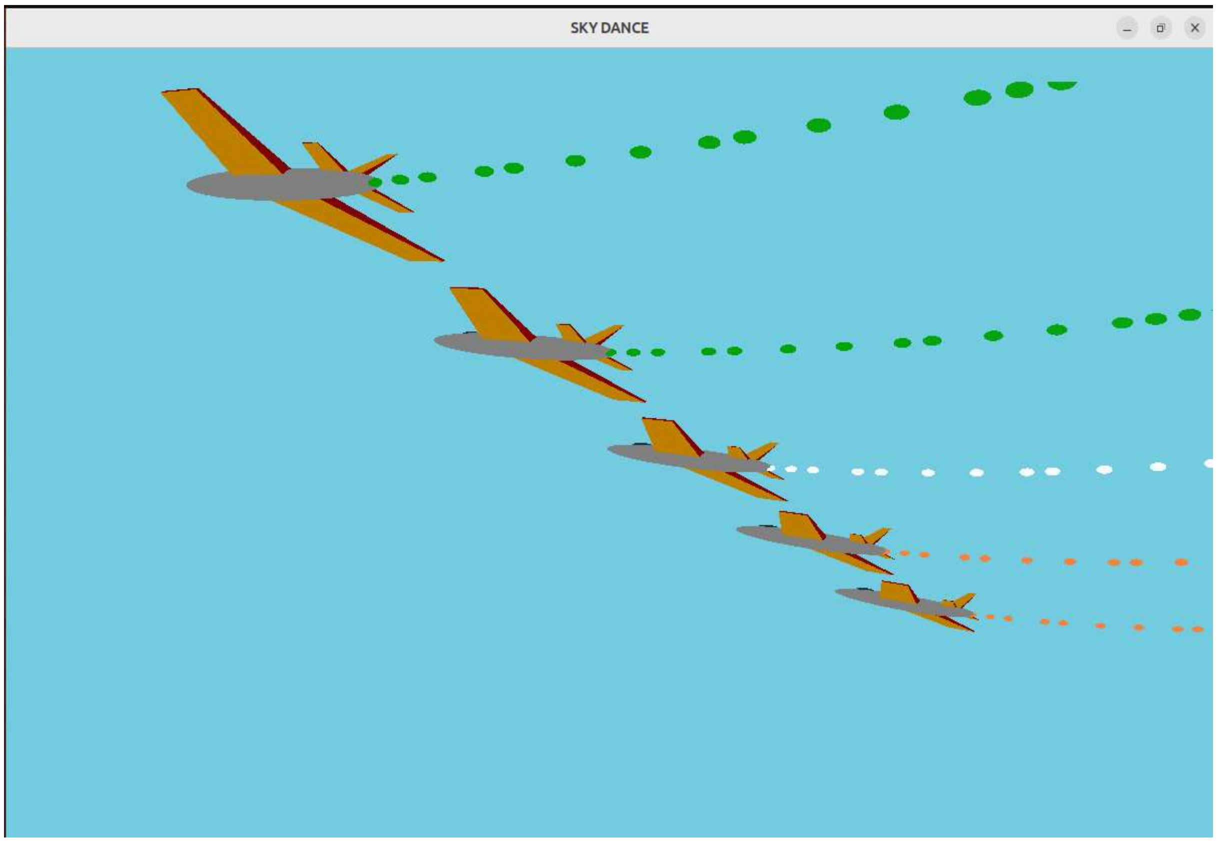


Fig 5

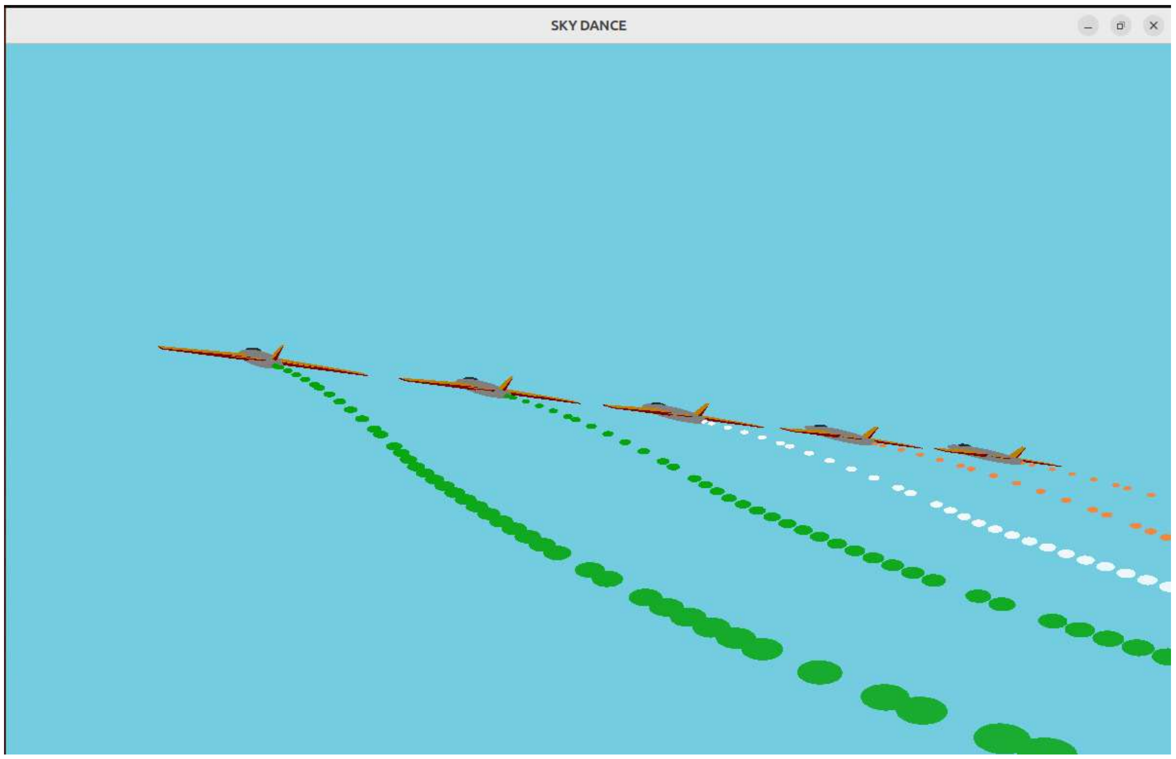


Fig 6

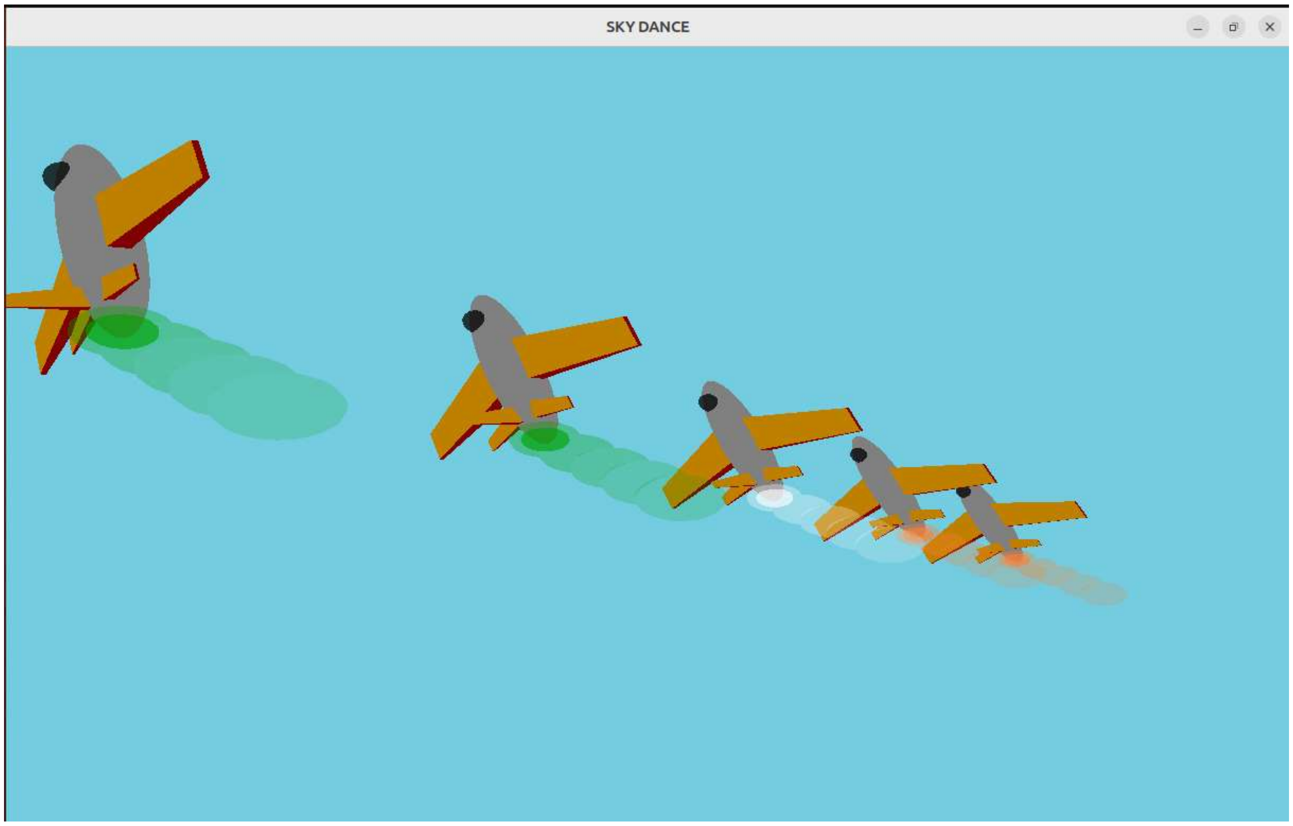


Fig 7