Computer Graphics (UCS505)

Project Report on

"Airplane Run"

Submitted By-

Ishit Choudhary 102003133

Gurleen Kaur 102003138

Group No. 5

B.E. Third Year - COE

Submitted to-

Dr. Harpreet Singh

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY

PATIALA - 147001

APRIL 2023

TABLE OF CONTENTS

DECLARATION

We, hereby declare that the project entitled Airplane Run submitted to TIET, Patiala is the record of original work done by us under the guidance of Dr. Harpreet Singh. This project has been submitted as a part of project evaluation in our course Computer Graphics (UCS505). We have worked with full determination for completion of this project. The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

Signature

Ishit Choudhary (102003133)

Gurleen Kaur (102003138)

INTRODUCTION TO PROJECT

Airplane Run is an exciting and challenging game built using OpenGL and C++. Get ready for a thrilling flying experience as you take control of an airplane soaring through the sky. The objective of the game is to avoid obstacles like buildings and clouds that come in your way while the airplane automatically moves forward and experiences the force of gravity pulling it down.

The game features a dynamic gameplay where the airplane moves forward at a constant speed, and you need to manoeuvre it skilfully to avoid collisions with the obstacles. You can make the airplane go up by pressing a button, providing a way to navigate through the challenging environments. But be careful, as the obstacles become increasingly difficult to avoid as you progress, requiring quick reflexes and strategic decision-making.

With immersive graphics powered by OpenGL, players will enjoy a visually stunning experience as they navigate through beautifully rendered environments, including cities, mountains, and clouds. The game also incorporates realistic physics, providing a sense of authenticity to the flight dynamics of the airplane.

As you progress, you can challenge yourself to achieve high scores, unlock achievements, and compete with friends or other players to see who can fly the furthest without crashing. With its addictive gameplay, stunning visuals, and challenging obstacles, Airborne Adventure is sure to provide an engaging gaming experience for players of all ages.

COMPUTER GRAPHICS CONCEPTS USED

The Airplane Run project uses various computer graphics concepts to create a visually appealing and interactive game. Some of these concepts include:

1. **OpenGL:** OpenGL is a popular graphics API that allows developers to create 2D and 3D applications, such as games, with high-performance graphics. Airplane Run uses OpenGL to render its immersive graphics.
2. **Texturing:** Texturing is the process of adding colors, patterns, or images to the surface of a 3D model. Airplane Run uses texturing to give its environments a more realistic look and feel.
3. **Transformations:** In computer graphics, transformations are used to manipulate the position, orientation, and scale of objects in a scene. Airplane Run uses transformations to move the airplane and obstacles, and to adjust the camera viewpoint. For example, when the player presses the button to make the airplane go up, **a translation transformation** is applied to move the airplane upward.
4. **Collision detection:** Collision detection is the process of detecting when two objects in a scene have collided. Airplane Run uses collision detection to detect when the airplane collides with obstacles, such as buildings and clouds.
5. **Animation:** Animation in computer graphics involves creating the illusion of motion by displaying a sequence of images or frames in rapid succession. In Airplane Run, animation is used to make the airplane and obstacles move smoothly through the environment and to provide visual feedback to the player. For example, when the player presses the button to make the airplane go up, the airplane moves upward smoothly over a period of time, creating a sense of motion and velocity. Similarly, when an obstacle is generated in the game, it moves toward the player's airplane, giving the player a sense of urgency and requiring them to make quick decisions to avoid a collision.

6. **3D Modelling**: The game features 3D models of the airplane, obstacles, and environment. 3D modelling is the process of creating three-dimensional objects using specialized software. In Airplane Run, 3D modelling is used to create realistic and detailed objects that can be rendered in real-time.
7. **Drawing algorithms**: Circle Drawing algorithms are used in computer graphics to generate circles and circular shapes on the screen. In Airplane Run, circle drawing algorithms may be used to create circular objects or elements in the game, such as clouds or circular obstacles.
8. **OpenGL Inbuilt Functions**: OpenGL is a graphics library that provides a set of built-in functions for creating and manipulating graphics primitives, such as points, lines, and polygons. Some of the inbuilt functions provided by OpenGL that may be used in the Airplane Run project include:
    - glBegin() and glEnd()
    - glPushMatrix() and glPopMatrix()
    - glTranslatef()
    - glColor3f()

    These functions can be used in Airplane Run to create a visually appealing and interactive game.

USER DEFINED FUNCTIONS

void keyPressed(unsigned char, int, int): This function is designed to handle keyboard presses. It looks out for the 'p' key. On pressing the 'p' key, the user can pause/un-pause the game. First character argument is for key pressed while the two integer arguments give the position of the mouse during the key press.

void mouse(int button, int state, int x, int y): This function's purpose is to read mouse input for clicking on buttons and controlling the airplane. The argument button differentiates between the left and right click. State tells us if that particular mouse button is pressed or not. The last two arguments tells us the position of the mouse in x and y coordinates.

void printString(float x, float y, float z, void* font, char* string): This function is used to print strings throughout the game. It is called many times and every time we need to print something on the screen, this function is called. The string is passed as the last argument while the required font is passed as the second last argument.

void buildingBlock(): This function builds a building. It initialises the building structure and defines the building height and colour

.

void CloudBlock(): Similar to buildingBlock, this function initialises the cloud structure.

void init(): Initial function to start the game. Decides which obstacle comes first, building or cloud based on a random decision.

void drawJet(): This function draws the jet. The jet is majorly drawn with the help of polygons. It was drawn in a modular fashion with separate modules for each of its component.

void gameEnd(): Displays the game-end screen which comes once the airplane has hit either the buildings or the clouds or has gone out-of-bounds (hitting the ground or going too-far-up).

void drawBg(): This function draws the background of the entire game. For example it sets the ground colour and gives the screen its sky-blue colour.

void welcome(): Displays the welcome screen and all the buttons, namely Play, Instructions, About, Exit.

void drawBuilding(): This function draws the entire building on the screen. It draws the building base, the building structure and the windows on the building. Everything is drawn with the help of polygons or rectangles.

void drawCloud(): This function draws the cloud onto the screen with the help of solid spheres.

bool cloudHit(): This function has a series of if-else statements to check if at any point the plane is touching/overlapping/hitting the Clouds. If it is, then the function returns True, else it returns False.

bool buildingHit(): This function has a series of if-else statements to check if at any point the plane is touching/overlapping/hitting the building. If it is, then the function returns True, else it returns False.

void printScore(): This function calculates and prints the score at the bottom of the screen when a player is playing the game.

void display(): One of the main function that binds all the other functions together into a common game logic. Calls other function as and when required by the game.

void moveJetU(): This function makes the jet translate in the positive y-axis by a fixed amount so as to give the impression that the jet is flying upwards.

void moveJetD(): This function makes the jet translate in the negative y-axis by a fixed amount so as to give the impression that the jet is falling downwards under the effect of gravity.

CODE

```
#include<stdlib.h>
#include<gl/glut.h>
#include<time.h>
#include<stdio.h>
#include<math.h>
#pragma GCC diagnostic ignored "-Wwrite-strings"
#define BLOCKSPEED 0.025
#define BOOSTER_MAX 10

int SCREENH = 600, SCREENW = 800;

//---------------- 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;
char score_Str[20], slevel[20];   //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(0.6, 0.6, 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(0.6, 0.6, 0.6);
   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(0.5, 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, 0.5, 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(0.6, 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(0.0, 0.0, 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,const char* string)
{
   const 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, "COMPUTER GRAPHICS
PROJECT ");
   drawString(35.5, 80, 0, GLUT_BITMAP_TIMES_ROMAN_24, "AIRPLANE RUN ");
   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

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

    if (booster > 0 && boost)
        score += 0.03;//SCORE with booster
    else
        score += 0.005;//SCORE without booster

    drawString(38, 1.5, 0, GLUT_BITMAP_TIMES_ROMAN_10, "Distance");
    sprintf_s(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.");
        drawString(23, 45, 0, GLUT_BITMAP_HELVETICA_18, "- Also the meter at the bottom shows the distance ");
```

```
    drawString(23, 41, 0, GLUT_BITMAP_HELVETICA_18, "    travelled,NITROS left,Atitude and the LEVEL.");
    drawString(23, 37, 0, GLUT_BITMAP_HELVETICA_18, "- As you reach distance multples 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, "ABOUT");
    drawString(21, 61, 0, GLUT_BITMAP_HELVETICA_18, "          COMPUTER GRAPHICS PROJECT ");
    drawString(23, 53, 0, GLUT_BITMAP_HELVETICA_18, "          Gurleen | Ishit ");

    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 += 0.01;
   }
   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 += 0.04;

    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 - 0.25;//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 - 0.25;
       }
       else
       {
          s.block_x -= bspd;
       }
    }
    if (b.state == true)
    {
       glTranslatef(b.block_x, 0.0, 0.0);
       drawBuilding();
    }
    else if (s.state == true)
    {
```

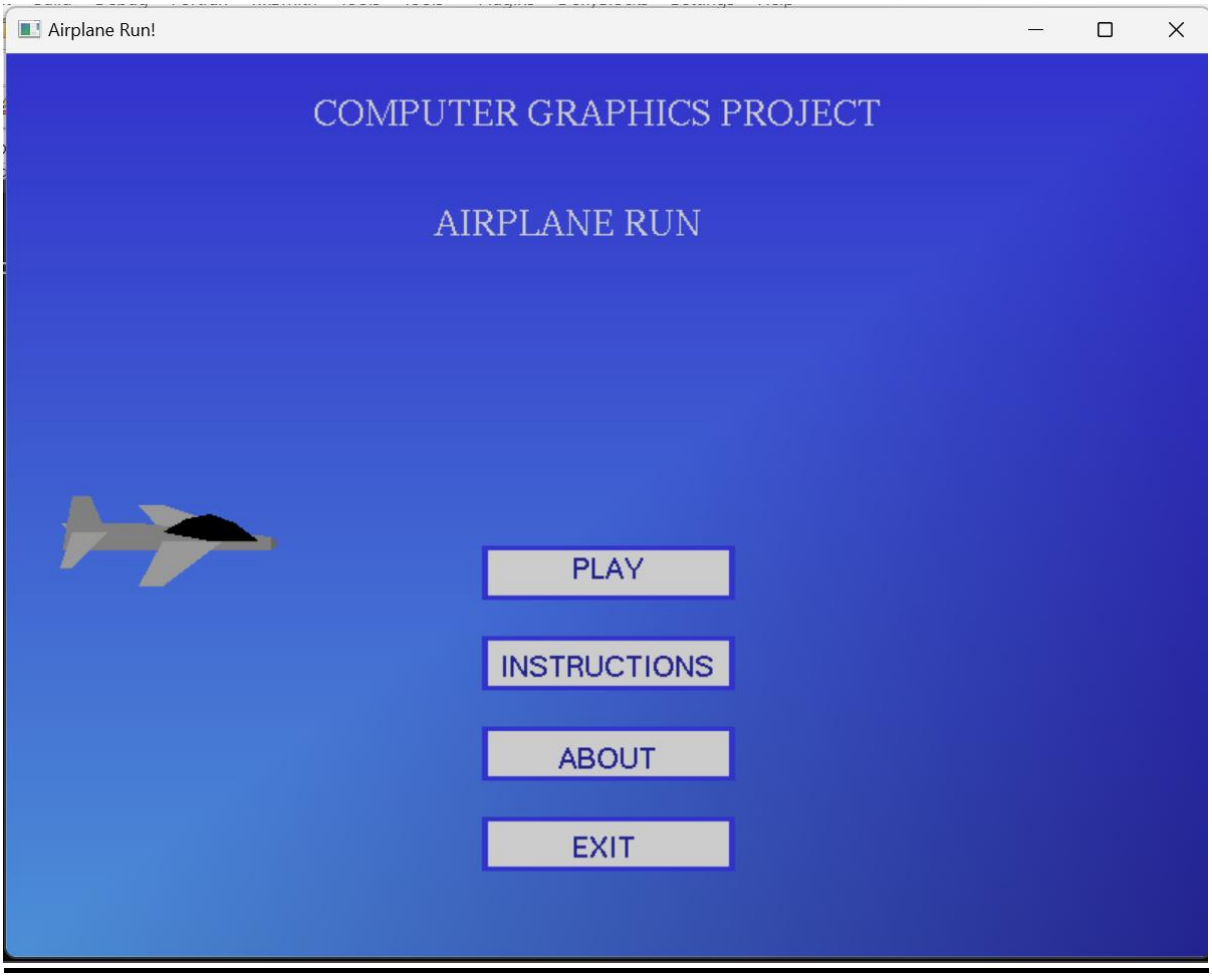
```
            glTranslatef(s.block_x, 0.0, 0.0);
            drawCloud();
        }
        glPopMatrix();

        printScore();
    }
    glFlush();
    glutSwapBuffers();
}

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

        plane_mvmt += 0.03;
        glutPostRedisplay();
    }
}

void moveJetD()          // jet moving down
{
    if (start == false)
        glutPostRedisplay();
    else if (pause == false)
    {
        //alti_ang+=0.15;
        plane_mvmt -= 0.025;
        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)
        {
```

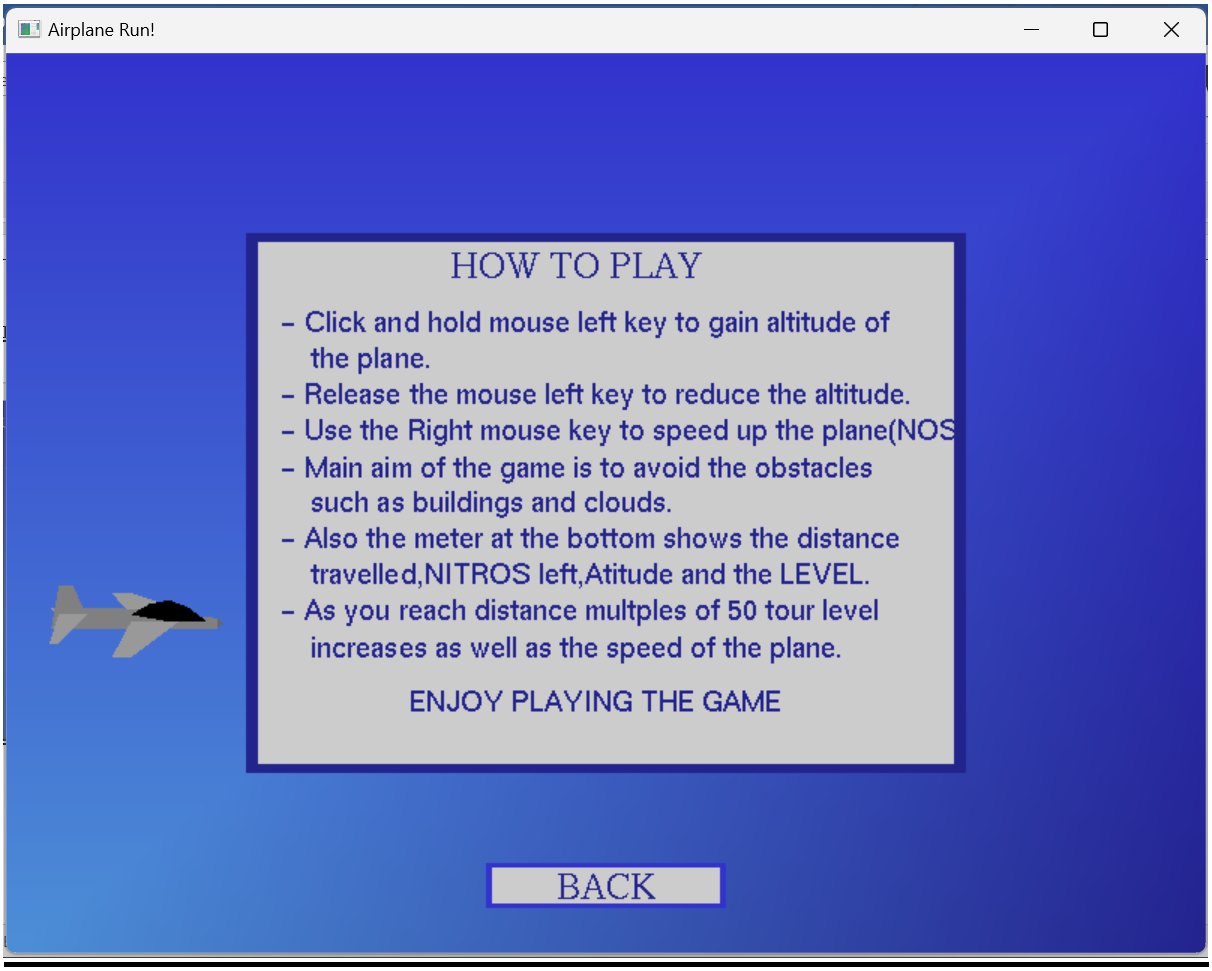
```
        if (my > 5 && my < 10)
        {
            wflag = true;
            if (instflag)
                instflag = false;
            else if (abtflag)
                abtflag = false;
            if (gameEndStatus)
            {
                wflag = true;
                gameEndStatus = 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);
        }
    }
}
```

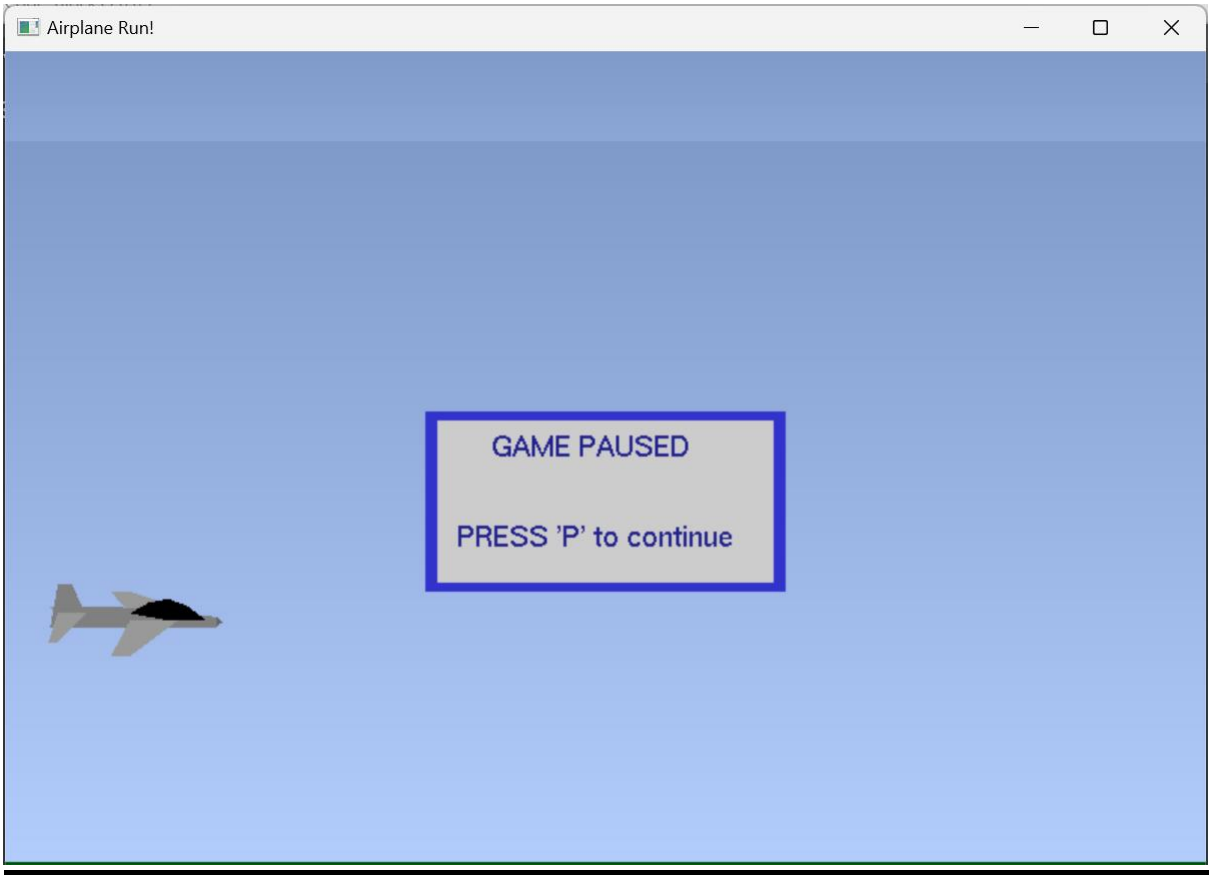
```
    else
    {

        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 = 0.05;
                }
            }
            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 == 27)
    {
        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);
```

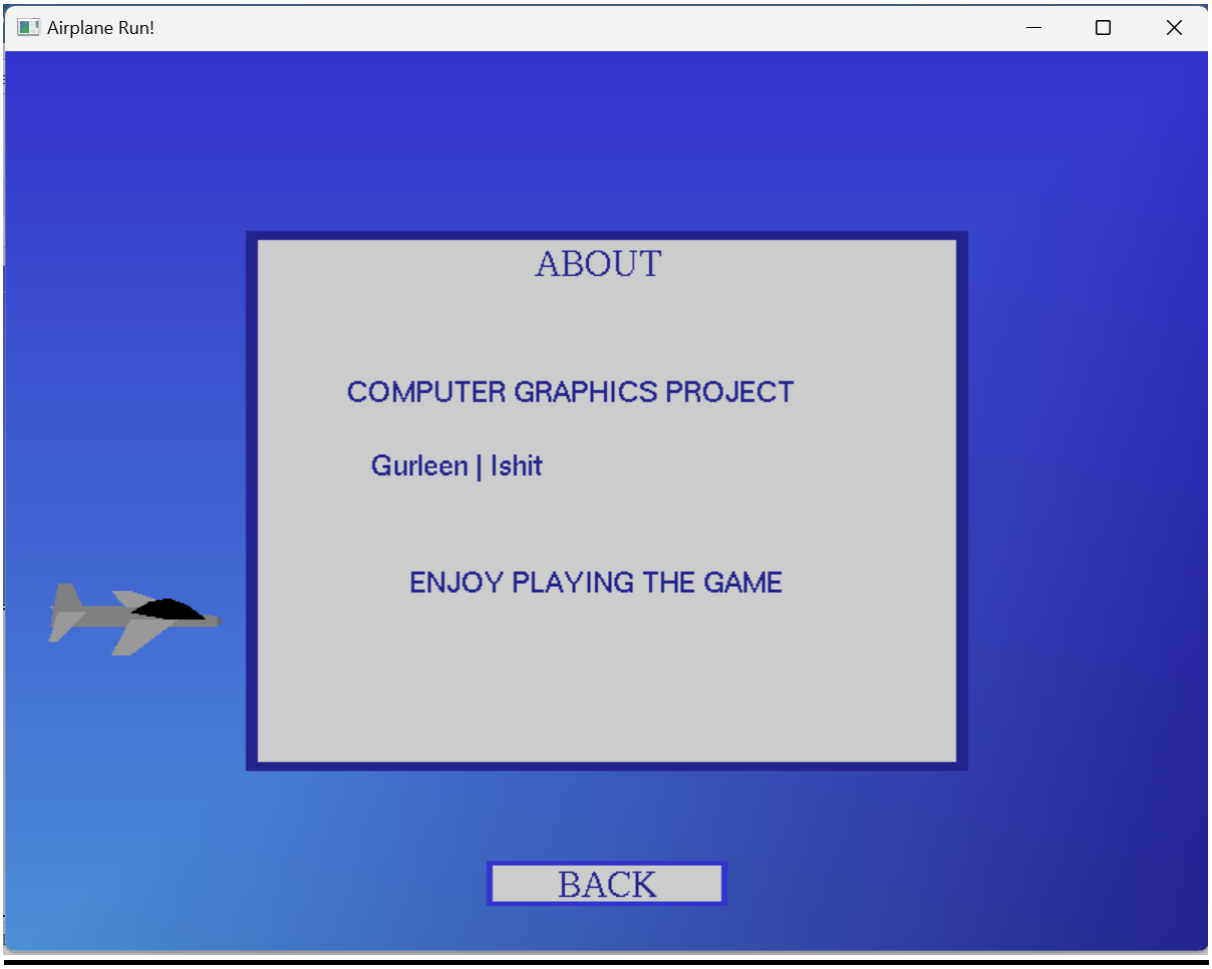
```
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 100.0, 0.0, 100.0, -5.0, 10.0);
    glMatrixMode(GL_MODELVIEW);
}

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

int main(int argc, char** argv)
{

    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(SCREENW, SCREENH);
    glutInitWindowPosition(50, 50);
    glutCreateWindow("Airplane Run!");
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(myReshape);
    glutMouseFunc(mouse);
    glutKeyboardFunc(keyPressed);
    glutMainLoop();
    return 0;
}
```

OUTPUT SCREENSHOTS

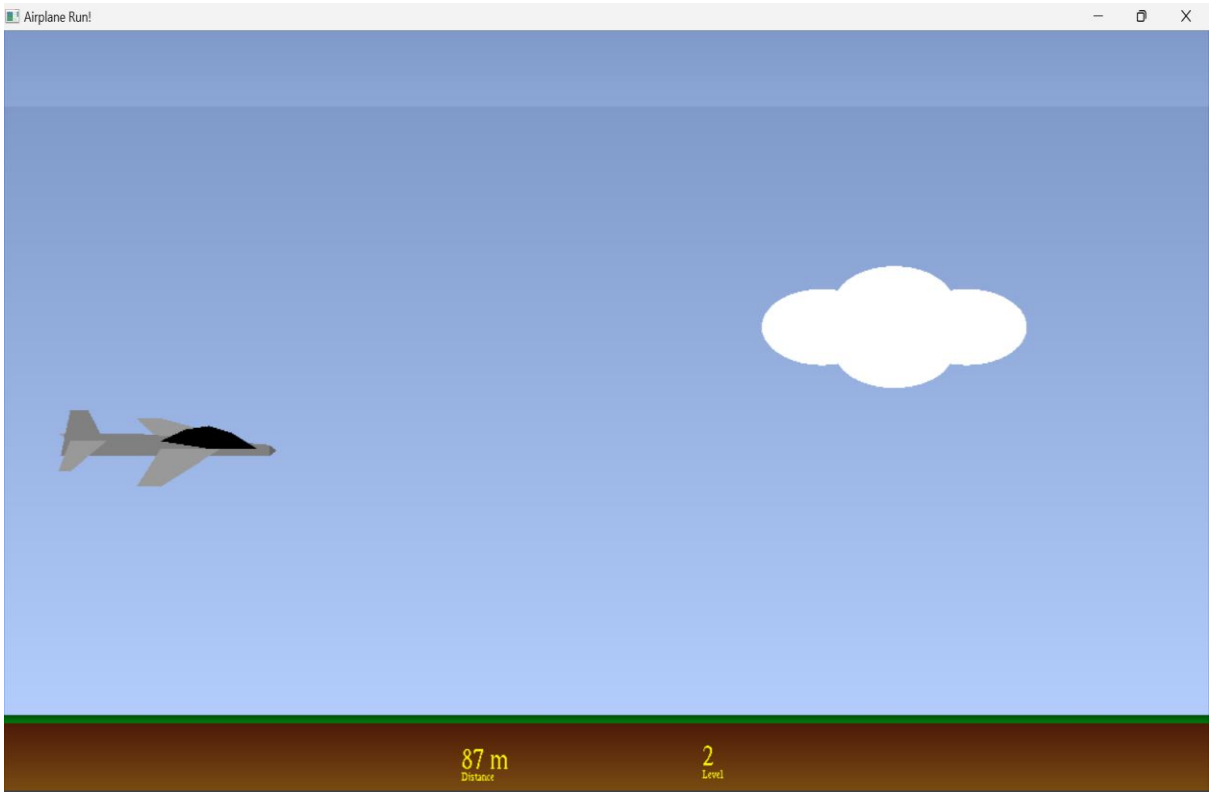


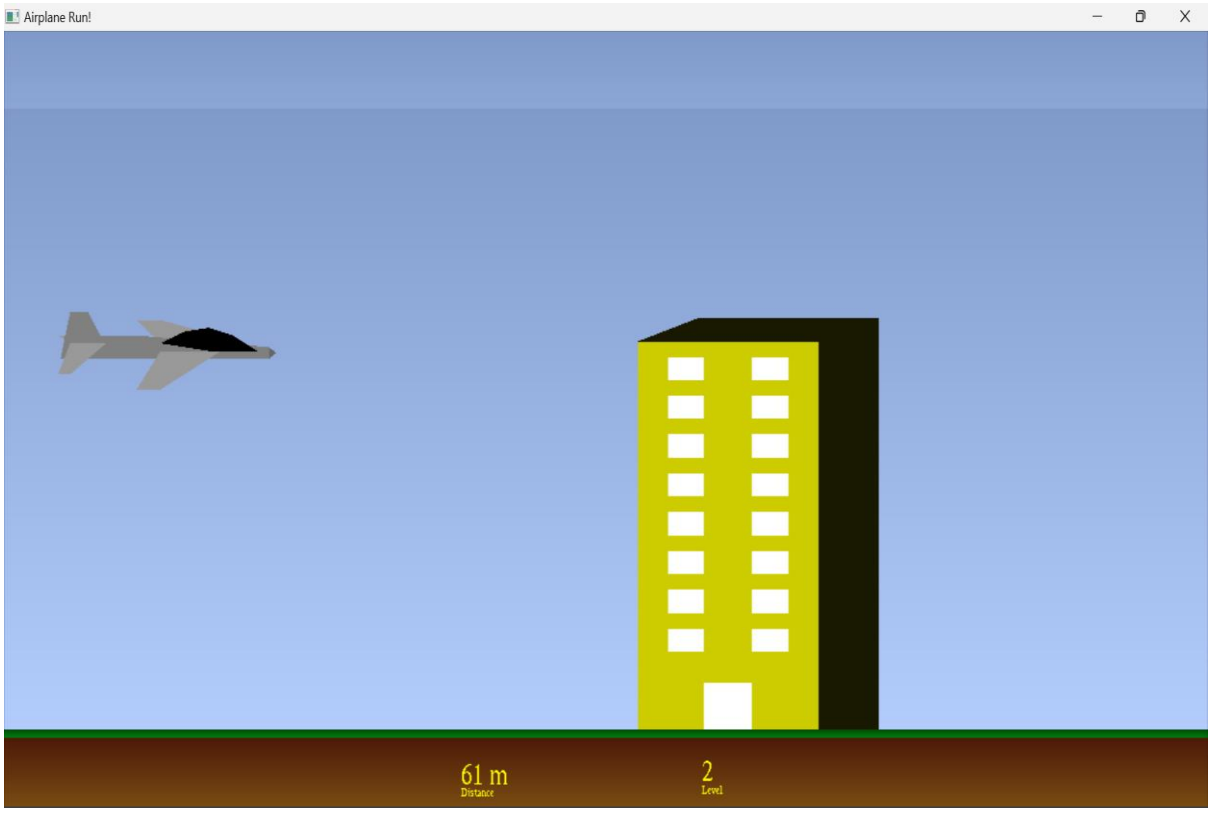

Airplane Run!
HOW TO PLAY
– Click and hold mouse left key to gain altitude of the plane.
– Release the mouse left key to reduce the altitude.
– Use the Right mouse key to speed up the plane(NOS
– Main aim of the game is to avoid the obstacles such as buildings and clouds.
– Also the meter at the bottom shows the distance travelled,NITROS left,Atitude and the LEVEL.
– As you reach distance multples of 50 tour level increases as well as the speed of the plane.
ENJOY PLAYING THE GAME
BACK


Airplane Run!
GAME PAUSED
PRESS 'P' to continue


Airplane Run!
ABOUT
COMPUTER GRAPHICS PROJECT
Gurleen | Ishit
ENJOY PLAYING THE GAME
BACK


Airplane Run!
GAME OVER!!!
DISTANCE : 8 m
LEVEL : 1
ENJOY PLAYING THE GAME
RESTART


Airplane Run!
87 m
Distance
2
Level


Airplane Run!
61 m
Distance
2
Level

REMARKS