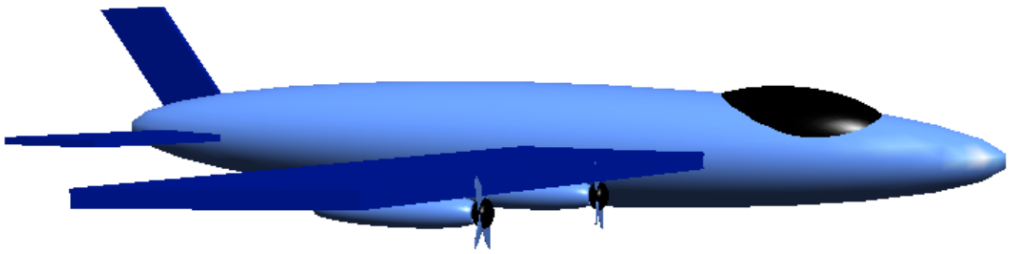
COMPUTER GRAPHICS

A User Interactive Game

PILOT YOUR PLANE!



GROUP MEMBERS

SHRUTI BHATT U18CO011

RIYA SINGHAL U18CO016

SEJAL TAJANE U18CO048

ANKITA KULKARNI U18CO067

Table of Contents

1. INTRODUCTION

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern recognition allow us to perceive and process pictorial data rapidly and efficiently. ***Interactive computer graphics*** *is the most important means of producing the pictures since the invention of photography and television. It has the added advantage that with the computer we can make pictures not only of concrete real world objects but also of abstract such as survey results.*

Our project ***"PILOT YOUT PLANE"*** *is similar to an arcade style game. The goal for the user is to go as far as possible without crash landing and try to score maximum points by passing through hoops that come along the way. With only a few controls at one's disposal, this is one of those "easy to learn, difficult to master" situations.*

Here the user is allowed to control the movement of the plane throughout a collision course. The plane's movement in the four principal directions can be controlled by the up, down, left and right arrow keys OR W,A,S,D keys. The user has to maneuver the plan using keyboard keys such that it passes through the hoops . For every hoop passed, the score for the player will be incremented by 1. The objective of the game is to maximise your score, and the game keeps track of your high score throughout. Moreover,the user must make sure that no part of the plane touches the buildings in the game environment as the game is programmed to end whenever you crash into the surroundings.

If you right click your mouse button, a drop down menu will appear. You can go to the HELP menu to get Familiar with the game or go to the PLANE menu to customise your plane before starting the game or while playing as well! The game allows you to exit mid-way whenever you wish to. The library glut.h is utilized in the implementation of the game as it provides a wide range of options to draw the necessary shapes.

2. Requirement Analysis

The requirement analysis phase of the project can be classified into:

- *Software Requirements*
- *Hardware Requirements*

2.1 SOFTWARE REQUIREMENTS:

Input Requirement	*Standard Input Device*	*:*	*Keyboard, Mouse*
Output Requirement	*Standard Output Device*	*:*	*Color Monitor(60Hz)*
Software Requirement	*Programming Language*	*:*	*C++*
	Compiler Used	*:*	*Codeblocks/ DevC++*
	Operating System	*:*	*Windows*

2.2 HARDWARE REQUIREMENTS:

The hardware requirements are very minimal and the software can run on most of the machines.

1.	*Main processor*	*:*	*Pentium 4*
2.	*Processor Speed*	*:*	*1000 MHz or More*
3.	*RAM Size*	*:*	*64 MB DDR or More*
4.	*Keyboard*	*:*	*Standard QWERTY serial or PS/2 keyboard*
5.	*Mouse*	*:*	*Standard serial or PS/2 mouse*
6.	*Compatibility*	*:*	*AT/T compatible*
7.	*Cache memory*	*:*	*512 KB*

3. openGL Functions

3.1 Basic Functions

1. glPushMatrix, glPopMatrix Function

The glPushMatrix and glPopMatrix functions push and pop the current matrix stack.

SYNTAX : void glPushMatrix();

void glPopMatrix(void);

PARAMETERS: This function has no parameters.

2. glColor3f Function

Sets the current color.

SYNTAX: void glColor3f(GLfloat red, GLfloat green, GLfloat blue);

3.2 Transformation Functions

1. glTranslate

*The **glTranslated** and **glTranslatef** functions multiply the current matrix by the translation matrix.*

void glTranslated(GLdouble x, GLdouble y, GLdouble z);

void glTranslatef(GLfloat x, GLfloat y, GLfloat z)

PARAMETERS

x, y, z : The x, y, and z coordinates of a translation vector.

2. glRotate

The glRotated and glRotatef functions multiply the current matrix by a rotation matrix.

void glRotate(Glfloat angle,Glfloat x,Glfloat y,Glfloat z);

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

PARAMETERS

angle : The angle of rotation, in degrees.

X: The x coordinate of a vector.

Y: The y coordinate of a vector.

Z: The z coordinate of a vector.

3. glScale

The glScaled and glScalef functions multiply the current matrix by a general scaling matrix

*void **glScaled**(GLdouble x, GLdouble y, GLdouble z);*

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

PARAMETERS

x, y, z : Scale factors along x, y and z axes respectively

3.3 Functions Used To Display

1. glClear Function

The glClear function clears buffers to preset values.

SYNTAX: glClear(GLbitfield mask);

PARAMETERS: mask

Bitwise OR operators of masks that indicate the buffers to be cleared. The four masks are:

GL_COLOR_BUFFER_BIT The buffers currently enabled for color writing.

GL_DEPTH_BUFFER_BIT The depth buffer.

glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

2. glMatrixMode Function

The glMatrixMode function specifies which matrix is the current matrix.

SYNTAX: void glMatrixMode(GLenum mode);

PARAMETERS: mode

The matrix stack that is the target for subsequent matrix operations. The mode parameter can assume one of three values:

GL_MODELVIEW : Applies subsequent matrix operations to the modelview matrix stack.

glMatrixMode(GL_MODELVIEW);

3. glLoadIdentity Function

The glLoadIdentity function replaces the current matrix with the identity matrix.

SYNTAX: void glLoadIdentity(void);

4. glutSwapBuffers

glutSwapBuffers swaps the buffers of the current window if double buffered.

SYNTAX: void glutSwapBuffers(void);

glutSwapBuffers();

3.4 Functions Used To Reshape

1. glutDisplayFunc Function

glutDisplayFunc sets the display callback for the current window.

*SYNTAX: void glutDisplayFunc(void (*func)(void));*

PARAMETERS:

- *func*

  The new display callback function.

- *glutDisplayFunc(display);*

3.5 Main Functions

1. glutInitDisplayMode Function

glutInitDisplayMode sets the initial display mode.

SYNTAX: void glutInitDisplayMode(unsigned int mode);

PARAMETERS:

- *mode : Display mode, normally the bitwise OR-ing of GLUT display mode bit masks.*

  GLUT_RGB: An alias for GLUT_RGBA.

  GLUT_DOUBLE: Bit mask to select a double buffered window. This overrides GLUT_SINGLE if it is also specified.

  GLUT_DEPTH: Bit mask to select a window with a depth buffer.

- *glutInitDisplayMode(GLUT_RGB | GLUT_DEPTH | GLUT_DOUBLE);*

2. glutInitWindowPosition, glutInitWindowSize Functions

glutInitWindowPosition and glutInitWindowSize set the initial window position and size respectively.

SYNTAX : void glutInitWindowSize(int width, int height);

void glutInitWindowPosition(int x, int y);

PARAMETERS:

- *width : Width in pixels.*
- *height : Height in pixels.*
- *x : Window X location in pixels.*
- *y : Window Y location in pixels.*
- *glutInitWindowSize(300,300);*

3. glutMainLoop Function

glutMainLoop enters the GLUT event processing loop.

SYNTAX: void glutMainLoop(void);

- *glutMainLoop();*

3.6 Text Displaying Functions

1.glutStrokeCharacter

glutStrokeCharacter renders a stroke character using OpenGL.

*SYNTAX: void glutStrokeCharacter(void *font, int character);*

PARAMETERS:

font - Stroke font to use.

character- Character to render

The fonts used are : GLUT_STROKE_ROMAN and GLUT_STROKE_MONO_ROMAN

2.outtextxy()

outtextxy() displays the text or string at a specified point (x, y) on the screen.

*SYNTAX : void outtextxy(int x, int y, char *string);*

3.7 Interactive Functions

1.glutCreateMenu(), glutAddMenuEntry(), glutAttachMenu()

GLUT provides a simple mechanism for creating pop-up menus. Each menu is associated with a procedure (which you provide) when you create the menu. When the menu item is selected, this procedure is called and passed the index of the item selected. Submenus are also possible!

*SYNTAX : int glutCreateMenu(void (*func)(int value));*
*SYNTAX : glutAddMenuEntry(char *name, int value);*
SYNTAX : glutAttachMenu(int button);

2.glutSpecialFunc()

glutSpecialFunc sets the special keyboard callback (such as arrow keys, function keys , space bar etc) for the current window.
*SYNTAX : void glutSpecialFunc(void (*func)(int key, int x, int y));*

3.glutKeyboardFunc()

glutKeyboardFunc sets the keyboard callback for the current window.
*SYNTAX : void glutKeyboardFunc(void (*func)(unsigned char key, int x, int y));*

4.MessageBox

MessageBox class has an overloaded static Show method that displays a message box with a message and action buttons. The action buttons can be OK and Cancel, Yes and No etc. Here are some of the options that can be used in C# message box.

3.8 Color and Shading Functions

1.glLightfv, glMaterialfv

The glLightfv function returns light source parameter values.
The glMaterialfv function specifies material parameters for the lighting model.

*SYNTAX : glLightfv(GLenum light, GLenum pname, const GLfloat *params);*

*SYNTAX : glMaterialfv(GLenum face, GLenum pname,const GLfloat *params);*

Parameter:

GL_AMBIENT: The params parameter contains four floating-point values that specify the ambient RGBA intensity of the light.

GL_DIFFUSE: The params parameter contains four floating-point values that specify the diffuse RGBA intensity of the light.

GL_SPECULAR: The params parameter contains four floating-point values that specify the specular RGBA intensity of the light.

GL_POSITION: The params parameter contains four floating-point values that specify the position of the light in homogeneous object coordinates.

4. Implementation

4.1 Features

Customise your Plane :
The user can choose the colour of the plane out of the given options.

Instructions :
An instruction manual is provided to guide the user on how to play the game.

Environment :
The plane is surrounded by a beautiful environment consisting of colourful buildings of varied heights.

Plane Crash :
If the plane crashes into the surrounding buildings, the game will terminate and an appropriate message will be displayed. You can choose to play again.

Score :
The score will increase by one when the plane passes through any one of the rings.
A High Score will also be maintained that will be updated throughout the time you play our game.

Stabilise the Plane :
The rotation of the plane has boundaries in all directions, so that the plane does not become unstable.

Exit :
The user is provided with an option to exit the game at any time.

4.2 User Interactions

Both mouse and keyboard user instructions have been added to this OpenGL code. They are listed below -

Mouse interaction :

Right Mouse Button : Displays menu with following options:

- *Plane Color*
- *Help*
- *Exit*

Keyboard interactions :

1. *'A' or Left Arrow Key : To move to the left*
2. *'D' or Right Arrow Key : To move to the right*
3. *'S' or Down Arrow Key : To move down*
4. *'W' or Up Arrow Key : To move up*
5. *'m' : To go back to start menu*
6. *'z' : To zoom in*
7. *shift + 'z' : To zoom out*
8. *'g' : To start the game*
9. *'h' : To get instructions on how to play the game*

4.3 User Defined Modules

void drawStrokeText(char* str,int x,int y,int z) : *This function is called whenever some text is to be printed on screen*

void fan() : *This function is used to draw the fans on each side of the plane*

void plane() : *This function is used to draw the plane structure (central body of the plane, the front and rear wings and the propellers).*

void singleHouse(int R,int G,int B) : *This function is used to create the structure of a single building.*

void house(int n,int R,int G) : *This function calls the singleHouse function in a loop to create multiple buildings.*

void environment(int n) : *This function creates the entire environment including the ground, rings and the buildings on either side.*

void draw() : *All components including the plane, rings and environment are brought together in this function.*

static void display(void) : *Used to display the page containing all the components of the game. It can be either the start page, or the game page.*

void processSpecialKeys(int key, int x, int y) : *Determines the function of each arrow key used in the game.*

static void key(unsigned char key, int x, int y) : *Determines the function of each alphabet key used in the game.*

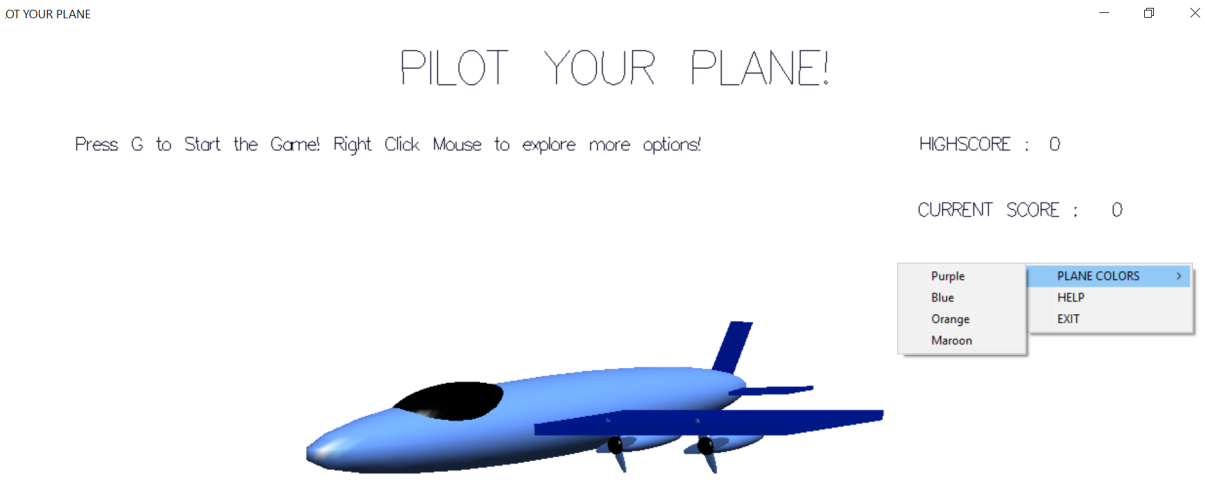
void GoMenu (int value) : *This function contains the different colour options available for the plane.*

5. OUTPUT

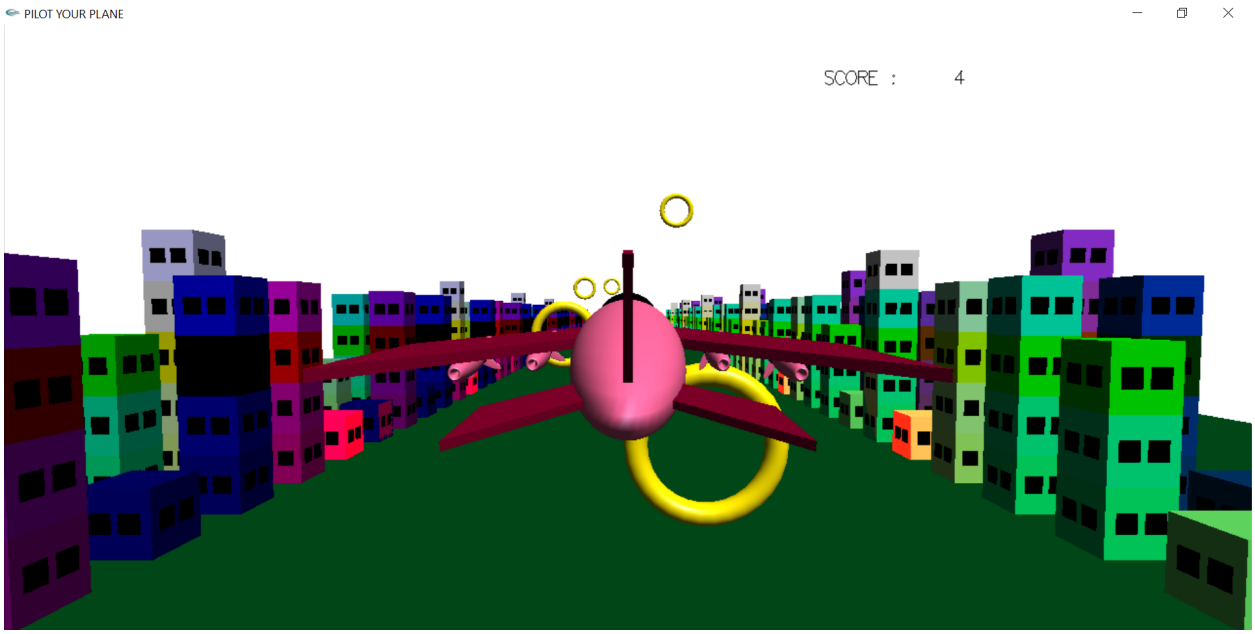
The following are snapshots of our output demonstrating all the features and the whole working of this project.

1. ***This image shows the home/start page of the game :***

2. ***This image shows the functions obtained after right clicking on the screen :***

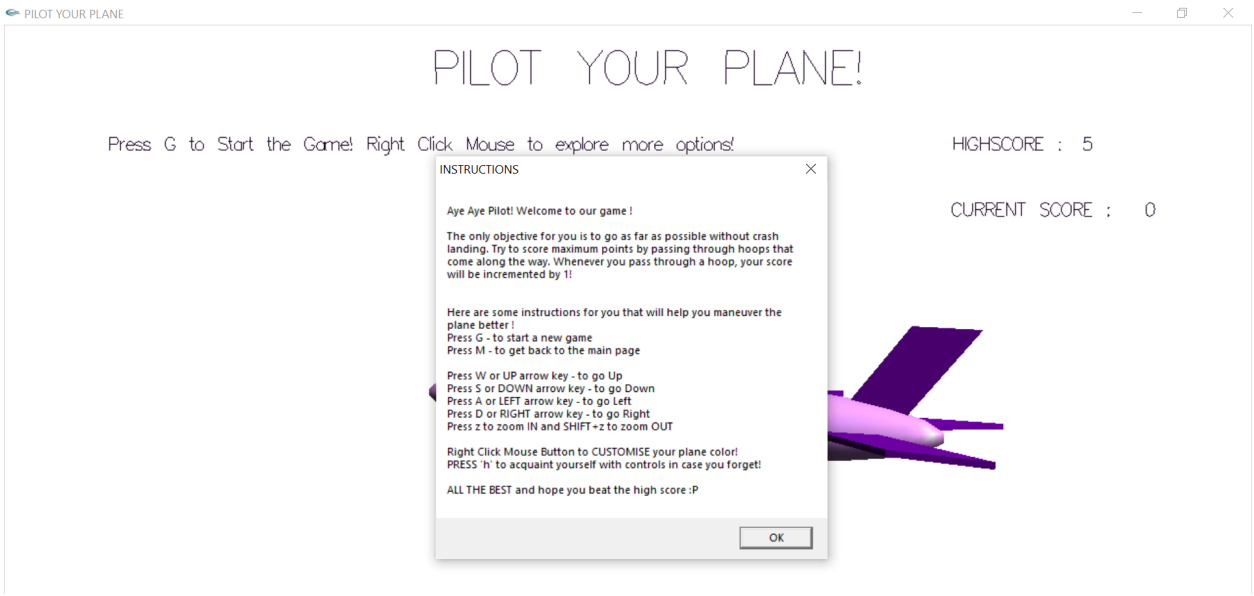


3. ***This image shows the plane in the middle of the game :***

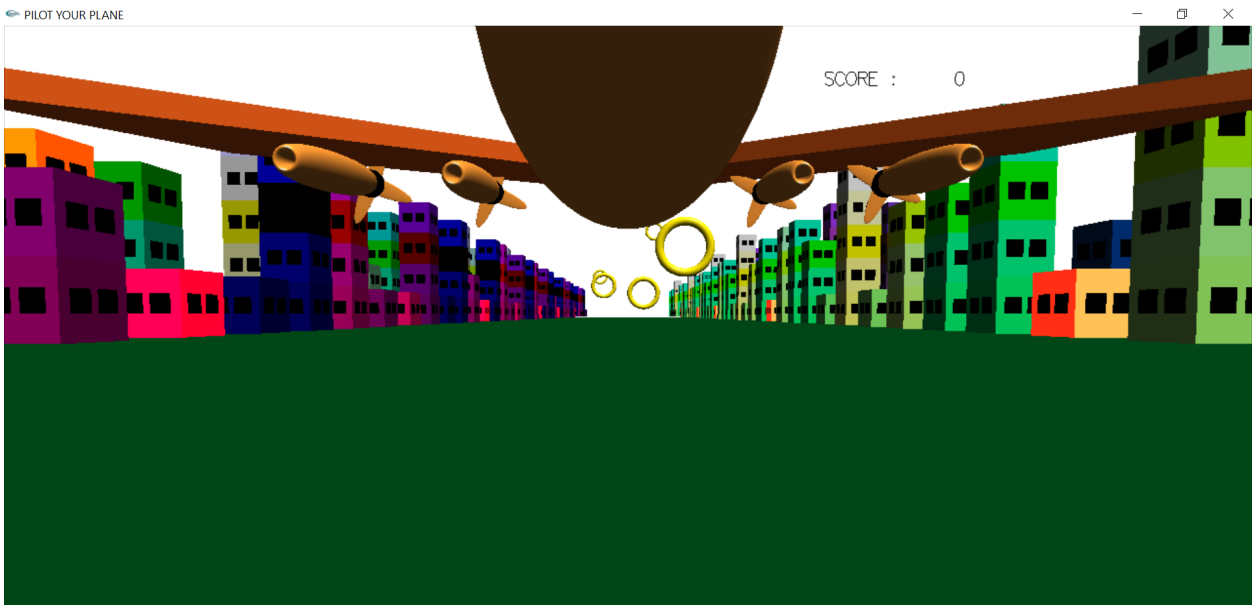


4. ***This image shows the home page after the plane has crashed, it also shows the current score as well as the high score :***

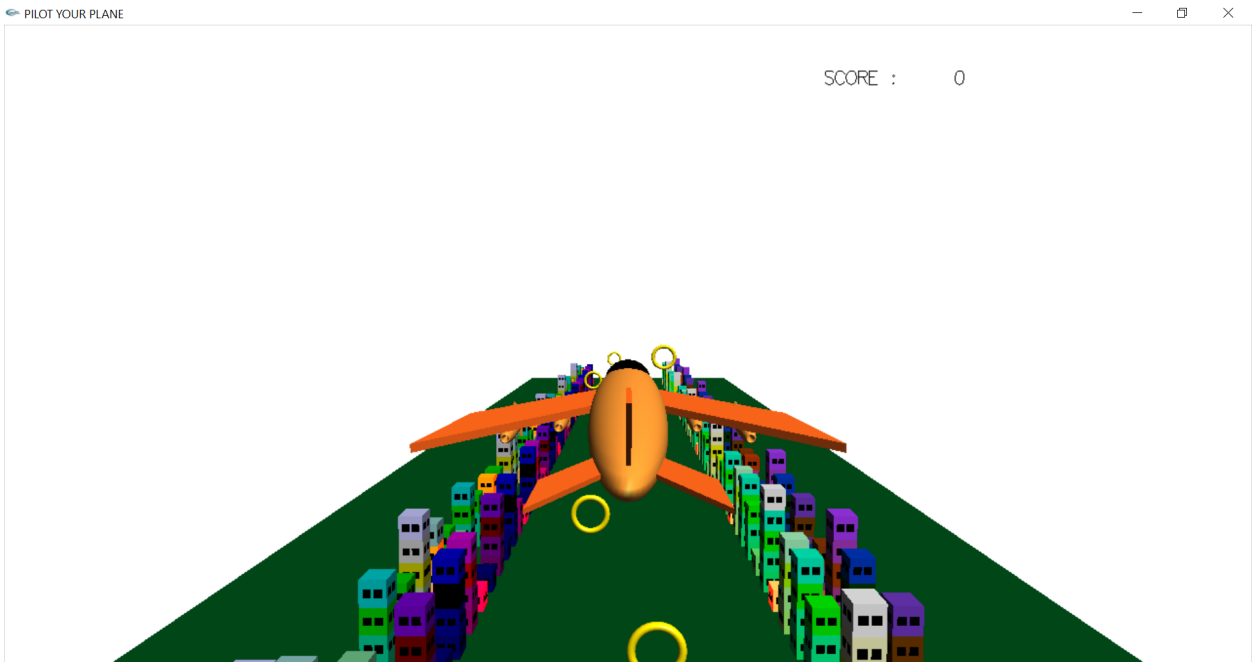
5. ***This image shows the help page :***



6. ***This image shows the zoomed in view while the plane is flying :***



7. This image shows the zoomed out view while the plane is flying :



6. CODE

```
///display highscore - done!!
///arrow keys - done!!
///plane has crashed - done!!
///check for the coordinates - speed constant - done!!
///customize plane - done!!
///instructions - done!!
///sound track
///check rubrik - clipping? - done!!
///q button to exit while playing - done!!


#include<windows.h>
#include<graphics.h>
#include <GL/glut.h>
#include<bits/stdc++.h>
#include <stdlib.h>
#include<conio.h>
#include<sstream>
#include<string>
#define rad (3.1416/180)
#define EN_SIZE 20

using namespace std;

int highscore = 0;
int score = 0;
int fuel=0;
int crash = 0;
int f,f1,f2,f3,f4,f5,f6;
float zoom=4;
int houseArr[5000][5000];
float tX=0,tY=0,tZ=-8,rX=0,rY=0,rZ=4;
float tZ1=-20,tZ2=-40,tZ3=-60,tZ4=-80,tZ5=-100,tZ6=-120;
float rotX=0,rotY=0,rotZ=0;
float angle=0;
float xEye=0.0f,yEye=5.0f,zEye=30.0f;
float cenX=0,cenY=0,cenZ=0,roll=0;
float radius=0;
float theta=0,slope=0;
float speed = 0.3;
float angleBackFrac = 0.2;
float r[] = {0.1,0.4,0.0,0.9,0.2,0.5,0.0,0.7,0.5,0.0};
float g[] = {0.2,0.0,0.4,0.5,0.2,0.0,0.3,0.9,0.0,0.2};
float b[] = {0.4,0.5,0.0,0.7,0.9,0.0,0.1,0.2,0.5,0.0};
int TIME=0;
bool START = false;
```

```
float torusPosX[7] = {1,-2,3,-4,-2,0,2};
float torusPosY[7] = {2,3,10,6,7,4,3};
char * buf;
bool rot = false;


float R=0.40; float G=0.6; float B=1; //plane
float Rw=0.0; float Gw=0.1; float Bw=0.6; //wings


static void resize(int width, int height)
{
   const float ar = (float) width / (float) height;

   glViewport(0, 0, width, height);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glFrustum(-ar, ar, -1.0, 1.0, 2.0, 1000.0);

   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}

void drawStrokeChar(char c,float x,float y,float z)
{
        glPushMatrix();
      glTranslatef(x, y+8,z);
      glScalef(0.002f,0.002f,z);
      glutStrokeCharacter(GLUT_STROKE_ROMAN , c);
        glPopMatrix();
}

void drawStrokeText(char* str,int x,int y,int z)
{
        char *c;
        glPushMatrix();
        glTranslatef(x, y+8,z);
        glScalef(0.002f,0.002f,z);

        for (c=str; *c != '\0'; c++)
        {
                glutStrokeCharacter(GLUT_STROKE_ROMAN , *c);
        }
        glPopMatrix();
}

void drawStrokeText2(char* str,int x,int y,int z)
{
        char *c;
        glPushMatrix();
        glTranslatef(x, y+8,z);
```

```
		glScalef(0.005f,0.005f,z);

		for (c=str; *c != '\0'; c++)
		{
			glutStrokeCharacter(GLUT_STROKE_ROMAN , *c);
		}
		glPopMatrix();
}


void IntToChar(int number, char * buf)
{
   ostringstream msg;
   msg << number;
   strcpy(buf, msg.str().c_str());
}


void fan(){
   ///center
   glColor3d(0,0,0);
   glPushMatrix();
      glTranslated(0,0,0);
      glScaled(1,1,0.7);
      glutSolidSphere(0.8,30,30);
   glPopMatrix();

   ///blades
   glColor3d(R,G,B);
   glPushMatrix();
      glTranslated(0,0,0);
      glRotated(5,0,1,0);
      glScaled(0.5,2.5,0.05);
      glutSolidSphere(1,30,30);
   glPopMatrix();

   glColor3d(R,G,B);
   glPushMatrix();
      glTranslated(0,0,0);
      glRotated(-5,0,1,0);
      glRotated(90,0,0,1);
      glScaled(0.5,2.5,0.05);
      glutSolidSphere(1,30,30);
   glPopMatrix();

}

//menu
void plane(){
   const double t = glutGet(GLUT_ELAPSED_TIME) / 1000.0;
   double a = t*90.0;
```

```
/// Main body
glColor3d(R,G,B); //color - cylinder
glPushMatrix();
   glTranslated(0,0,0);
   glScaled(3,0.4,0.5);
   glutSolidSphere(1,30,30);
glPopMatrix();

glColor3d(0,0,0); //color - cockpit
glPushMatrix();
   glTranslated(1.7,0.1,0);
   glScaled(1.5,0.7,0.8);
   glRotated(40,0,1,0);
   glutSolidSphere(0.45,30,30);
glPopMatrix();

///front wings

///Right wing
glColor3d(Rw,Gw,Bw); //color
glPushMatrix();
   glTranslated(0,0,1.2);
   glRotated(-50,0,1,0);
   glScaled(0.7,0.1,3);
   glRotated(25,0,1,0);
   glutSolidCube(1);
glPopMatrix();

///propeller1
glColor3d(Rw,Gw,Bw); //color
glPushMatrix();
   glTranslated(-0.3,-0.15,1.5);
   glRotated(90,0,1,0);
   /// FAN
   glPushMatrix();
      glTranslated(0,0,0.5);
      glRotated(10*a,0,0,1);
      glScaled(0.1,0.1,0.1);
      fan();
  glPopMatrix();
   glScaled(0.1,0.1,0.9);
   glutSolidTorus(0.5,0.5,50,50);
glPopMatrix();

///propeller2
glColor3d(Rw,Bw,Gw); //color
glPushMatrix();
   glTranslated(0.2,-0.15,0.9);
   glRotated(90,0,1,0);
   /// FAN
```

```
    glPushMatrix();
        glTranslated(0,0,0.5);
        glRotated(10*a,0,0,1);
        glScaled(0.1,0.1,0.1);
        fan();
    glPopMatrix();
    glScaled(0.1,0.1,0.9);
    glutSolidTorus(0.5,0.5,50,50);
glPopMatrix();

///Left wing
glColor3d(Rw,Gw,Bw); //color
glPushMatrix();
    glTranslated(0,0,-1.2);
    glRotated(50,0,1,0);
    glScaled(0.7,0.1,3);
    glRotated(-25,0,1,0);
    glutSolidCube(1);
glPopMatrix();

///propeller1
glColor3d(Rw,Gw,Bw); //color
glPushMatrix();
    glTranslated(-0.3,-0.15,-1.5);
    glRotated(90,0,1,0);
    /// FAN
    glPushMatrix();
        glTranslated(0,0,0.5);
        glRotated(10*a,0,0,1);
        glScaled(0.1,0.1,0.1);
        fan();
    glPopMatrix();
    glScaled(0.1,0.1,0.9);
    glutSolidTorus(0.5,0.5,50,50);
glPopMatrix();

///propeller2
glColor3d(Rw,Gw,Bw); //color
glPushMatrix();
    glTranslated(0.2,-0.15,-0.9);
    glRotated(90,0,1,0);
    /// FAN
    glPushMatrix();
        glTranslated(0,0,0.5);
        glRotated(10*a,0,0,1);
        glScaled(0.1,0.1,0.1);
        fan();
    glPopMatrix();
    glScaled(0.1,0.1,0.9);
    glutSolidTorus(0.5,0.5,50,50);
glPopMatrix();
```

```
    ///rear wings
    glPushMatrix();
      glTranslated(-2.8,0,0);
      glScaled(0.8,0.5,0.3);

      ///Right
      glColor3d(Rw,Gw,Bw); //color
      glPushMatrix();
        glTranslated(0.4,0,1.5);
        glRotated(-30,0,1,0);
        glScaled(0.7,0.1,3);
        glRotated(10,0,1,0);
        glutSolidCube(1);
      glPopMatrix();

      ///left
      glColor3d(Rw,Gw,Bw);  //color
      glPushMatrix();
        glTranslated(0.4,0,-1.5);
        glRotated(30,0,1,0);
        glScaled(0.7,0.1,3);
        glRotated(-10,0,1,0);
        glutSolidCube(1);
      glPopMatrix();
    glPopMatrix();

    ///vertical
      glColor3d(Rw,Gw,Bw); //color
      glPushMatrix();
        glTranslated(-2.7,0.5,0);
        glRotated(45,0,0,1);
        glScaled(0.8,2,0.1);
        glRotated(-20,0,0,1);
        glutSolidCube(0.5);
    glPopMatrix();

}


void singleHouse(int R,int G,int B){
    glColor3d(r[R%11],g[G%11],b[B%11]);
    glPushMatrix();
      glTranslated(0,0,0);
      glutSolidCube(1);
    glPopMatrix();

    glColor3d(0,0,0);
    glPushMatrix();
      glTranslated(0.2,0,0);
```

```
        glScaled(0.3,0.3,1.001);
        glutSolidCube(1);
    glPopMatrix();

    glColor3d(0,0,0);
    glPushMatrix();
        glTranslated(-0.2,0,0);
        glScaled(0.3,0.3,1.001);
        glutSolidCube(1);
    glPopMatrix();

    glColor3d(0,0,0);
    glPushMatrix();
        glTranslated(0,0,0.2);
        glScaled(1.001,0.3,0.3);
        glutSolidCube(1);
    glPopMatrix();

    glColor3d(0,0,0);
    glPushMatrix();
        glTranslated(0,0,-0.2);
        glScaled(1.001,0.3,0.3);
        glutSolidCube(1);
    glPopMatrix();

}


void house(int n,int R,int G){
    for(int i=0;i<n;i++){
        glPushMatrix();
            glTranslated(0,0.8+i,0);
            singleHouse(G,R,i);
        glPopMatrix();
    }
}

void environment(int n){

    /// Ground
    glColor3d(0.0,0.3,0.1);
    glPushMatrix();
        glTranslated(0,0,0);
        glScaled(EN_SIZE*2,0.3,EN_SIZE*2);
        glutSolidCube(1);
    glPopMatrix();

    ///rings
    glColor3d(1,0.9,0.00);
    glPushMatrix();
        glTranslated(torusPosX[n],torusPosY[n],0);
```

```
      glScaled(0.3,0.3,0.3);
      glutSolidTorus(0.5,3,30,30);
   glPopMatrix();

      for(int i=-(EN_SIZE/2)+1;i<(EN_SIZE/2);i+=2){
         for(int j=-(EN_SIZE/2)+1;j<(EN_SIZE/2);j+=2){
            if(houseArr[i+(EN_SIZE/2)+1][j+(EN_SIZE/2)+1]!=0){
               glPushMatrix();
                  glTranslated(i,0,j);
                  house(houseArr[i+(EN_SIZE/2)+1][j+(EN_SIZE/2)+1],i,j);
               glPopMatrix();
            }else if(i>=-5&&i<=5){}
            else{
               houseArr[i+(EN_SIZE/2)+1][j+(EN_SIZE/2)+1]=(rand()%5)+1;
               glPushMatrix();
                  glTranslated(i,0,j);
                  house(houseArr[i+(EN_SIZE/2)+1][j+(EN_SIZE/2)+1],i,j);
               glPopMatrix();
            }
         }
      }
}


void draw(){

   f=0;f1=0;f2=0;f3=0;f4=0;f5=0;f6=0;

   ///Plane
   //rotation tilt
   if(rotX>10)rotX=10;
   if(rotX<-10)rotX=-10;
   if(rotZ>10)rotZ=10;
   if(rotZ<-10)rotZ=-10;

   glPushMatrix();
      //initial position
      glTranslated(0,1,0);
      glRotated(90,0,1,0);
      glRotated(5,0,0,1);

      glRotated(rotX,1,0,0);
      glRotated(rotY,0,1,0);
      glRotated(rotZ,0,0,1);
      glScaled(0.4,0.4,0.4);
      plane();
   glPopMatrix();


   ///Environment
   if(tY>0.1)tY= 0.1; //ground
```

```
if(tY<-15)tY= -15; //sky

if(tX>=6 && tY>=-4){
   crash=1;
   START = false;
}if(tX<=-6 && tY>=-4){
   crash=1;
   START = false;
}

///rings/hoops
//cout<<tX<<" "<<tY<<" "<<tZ<<endl;
f=0;f1=0;f2=0;f3=0;f4=0;f5=0;f6=0;

///-1,-1
if(tX<=-0.5 && tX>=-1.5 && tY<=-0.5 && tY>=-1.5 && tZ>=-8.5 && tZ<=-7.5){
   f=1;
   cout<<"hello ring0"<<endl;
}
if(f==1){score=score+1;}


///2.1, -1.8
if(tX>=1.6 && tX<=2.6 && tY>=-2.3 && tY<=-1.3 && tZ1>=-8.5 && tZ1<=-7.5){
   f1=1;
   //cout<<"hello ring1"<<endl;
}
if(f1==1){score=score+1;}


///-3, -9
if(tX>=-3.5 && tX<=-2.5 && tY>=-9.5 && tY<=-8.5 && tZ2>=-8.5 && tZ2<=-7.5){
   f2=1;
   //cout<<"hello ring2"<<endl;
}
if(f2==1){score=score+1;}


///4,-5
if(tX<=4.5 && tX>=3.5 && tY<=-4.5 && tY>=-5.5 && tZ3>=-8.5 && tZ3<=-7.5){
   f3=1;
   //cout<<"hello ring3"<<endl;
}
if(f3==1){score=score+1;}


///1.8, -6
if(tX>=1.3 && tX<=2.3 && tY>=-6.5 && tY<=-5.5 && tZ4>=-8.5 && tZ4<=-7.5){
   f4=1;
   //cout<<"hello ring4"<<endl;
}
```

```
if(f4==1){score=score+1;}


///0,-3
if(tX<=0.5 && tX>=-0.5 && tY<=-2.5 && tY>=-3.5 && tZ5>=-8.5 && tZ5<=-7.5){
   f5=1;
   //cout<<"hello ring5"<<endl;
}
if(f5==1){score=score+1;}


///-1.95, -1.95
if(tX>=-2.45 && tX<=-1.45 && tY>=-2.45 && tY<=-1.45 && tZ6>=-8.5 && tZ6<=-7.5){
   f6=1;
   //cout<<"hello ring6"<<endl;
}
if(f6==1){score=score+1;}

//cout<<f<<" "<<f1<<" "<<f2<<" "<<f3<<" "<<f4<<" "<<f5<<" "<<f6<<endl;


glPushMatrix();
   glTranslated(tX,tY,tZ);
   environment(0);
glPopMatrix();

glPushMatrix();
  glTranslated(tX,tY,tZ1);
   environment(1);
glPopMatrix();

glPushMatrix();
   glTranslated(tX,tY,tZ2);
   environment(2);
glPopMatrix();

glPushMatrix();
   glTranslated(tX,tY,tZ3);
   environment(3);
glPopMatrix();

glPushMatrix();
   glTranslated(tX,tY,tZ4);
   environment(4);
glPopMatrix();

glPushMatrix();
   glTranslated(tX,tY,tZ5);
   environment(5);
glPopMatrix();
```

```
    glPushMatrix();
       glTranslated(tX,tY,tZ6);
       environment(6);
    glPopMatrix();



    tZ+=speed;
    tZ1+=speed;
    tZ2+=speed;
    tZ3+=speed;
    tZ4+=speed;
    tZ5+=speed;
    tZ6+=speed;

   /*cout<<"TZ0 "<<tZ<<" "<<"TZ1 "<<tZ1<<" "<<"TZ2 "<<tZ2<<" "<<"TZ3 "<<tZ3<<" "
   <<"TZ4 "<<tZ4<<" "<<"TZ5 "<<tZ5<<" "<<"TZ6 "<<tZ6<<" "<<endl;*/

    if(tZ>=20)tZ=-110;
    if(tZ1>=20)tZ1=-110;
    if(tZ2>=20)tZ2=-110;
    if(tZ3>=20)tZ3=-110;
    if(tZ4>=20)tZ4=-110;
    if(tZ5>=20)tZ5=-110;
    if(tZ6>=20)tZ6=-110;


    ///plane gets pivoted back to original position
    if(rotX>0)rotX-=angleBackFrac;
    if(rotX<0)rotX+=angleBackFrac;
    if(rotY>0)rotY-=angleBackFrac;
    if(rotY<0)rotY+=angleBackFrac;
    if(rotZ>0)rotZ-=angleBackFrac;
    if(rotZ<0)rotZ+=angleBackFrac;
    //cout<<rotX<<" "<<rotY<<" "<<rotZ<<endl;


    ///speed variables

    speed = 0.6; //constant speed
    //cout<<speed<<endl;
    //cout<<score<<endl;

}


static void display(void)
{

    //glutGet(GLUT_ELAPSED_TIME) - no of milliseconds elapsed when glut init is called
    const double t = glutGet(GLUT_ELAPSED_TIME) / 1000.0; //seconds
```

```
double a = t*90.0;
double aa=a;

if(!rot){ //if rot == false
   a=0;
}

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

glLoadIdentity();

gluLookAt(       0.0, 4.5, 10.0,
          0, 4, 0,
          0, 1.0f, 0.0f);

if(START){

   glPushMatrix();
      glTranslated(0,0,0);
      glScaled(zoom,zoom,zoom); //current matrix is multiplied by a scaling matrix
      glRotated(a,0,1,0); //a - angle
      draw();
   glPopMatrix();

   drawStrokeText("SCORE : ",3,0,0);
   char str[3];
   int x=score;
   itoa(x,str,10);
   drawStrokeText(str,5,0,0);
}
else{

   glPushMatrix();
      glTranslated(0,3,0);
      glRotated(aa,0,1,0);
      glScaled(1.5,1.5,1.5);
      plane();
   glPopMatrix();

   drawStrokeText("Press G to Start the Game! Right Click Mouse to explore more options!",-8,-1,0);
   drawStrokeText2("PILOT YOUR PLANE!",-3,0,0);
   drawStrokeText("HIGHSCORE :",5,-1,0);

   drawStrokeText("CURRENT SCORE :",5,-2,0);
   char str1[3];
   int x=score;
   itoa(x,str1,10);
   drawStrokeText(str1,8,-2,0);

   char str2[3];
   if(highscore<x){
```

```
            highscore=x;
        }
        itoa(highscore,str2,10);
        drawStrokeText(str2,7,-1,0);

        if(crash==1){
        drawStrokeText("YOUR PLANE CRASHED :(",-2,-2,0);
        }

    }

    glutSwapBuffers();
}


void processSpecialKeys(int key, int x, int y)
{
    float frac = 0.15;
    float rotFrac = 1;
            switch(key) {
                    case GLUT_KEY_UP:
            tY-=frac;
            rotZ+=rotFrac;
            break;
        case GLUT_KEY_DOWN:
            tY+=frac;
            rotZ-=rotFrac;
            break;
        case GLUT_KEY_LEFT:
            tX+=frac;
            rotX-=rotFrac*3;
            rotY+=rotFrac/2;
            break;
        case GLUT_KEY_RIGHT:
            tX-=frac;
            rotX+=rotFrac*3;
            rotY-=rotFrac/2;
            break;
              }
}

static void key(unsigned char key, int x, int y)
{
    float frac = 0.15;
    float rotFrac = 1;
    switch (key)
    {
        case 'g':
            START=true;
            crash=0;
            score=0;
```

```
        tX=0,tY=0;
        break;
    case 'm':
        START=false;
        break;

    case 'z':
        zoom+=0.05;
        break;
    case 'Z':
        zoom-=0.05;

    case 'w':
        tY-=frac;
        rotZ+=rotFrac;
        break;
    case 's':
        tY+=frac;
        rotZ-=rotFrac;
        break;
    case 'a':
        tX+=frac;
        rotX-=rotFrac*3;
        rotY+=rotFrac/2;
        break;
    case 'd':
        tX-=frac;
        rotX+=rotFrac*3;
        rotY-=rotFrac/2;
        break;
    case 'q':
        exit(0);
        break;
    case 'h' :
        MessageBox(NULL,
                    "Press G - to start a new game\n"
                    "Press M - to get back to the main page\n\n"

                    "Press W or UP arrow key - to go Up\n"
                    "Press S or DOWN arrow key - to go Down\n"
                    "Press A or LEFT arrow key - to go Left\n"
                    "Press D or RIGHT arrow key - to go Right\n"
                    "Press z to zoom IN and SHIFT+z to zoom OUT\n\n"

                    "Right Click Mouse Button to CUSTOMISE your plane color!\n"

                    "\n\nALL THE BEST!"

                  ,"CONTROLS", MB_OK);
}
```

```
   glutPostRedisplay();
}

static void idle(void)
{
   glutPostRedisplay();
}

const GLfloat light_ambient[]  = { 0.0f, 0.0f, 0.0f, 1.0f };
const GLfloat light_diffuse[]  = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat light_specular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat light_position[] = { 2.0f, 5.0f, 5.0f, 0.0f };

const GLfloat mat_ambient[]    = { 0.7f, 0.7f, 0.7f, 1.0f };
const GLfloat mat_diffuse[]    = { 0.8f, 0.8f, 0.8f, 1.0f };
const GLfloat mat_specular[]   = { 1.0f, 1.0f, 1.0f, 1.0f };
const GLfloat high_shininess[] = { 100.0f };


void GoMenu (int value)
{
    switch(value)
    {
        case 1://purple
            R=0.9; G=0.6; B=1;
            Rw=0.47;Gw=0.00;Bw=0.70;
            break;

        case 2://blue
            Rw=0.0; Gw=0.1; Bw=0.6;
            R=0.40;G=0.60;B=1;
            break;

        case 3://orange
            Rw=1.00; Gw=0.40; Bw=0.10;
            R=1;G=0.6;B=0.20;
            break;

        case 4://maroon
            R=1; G=0.4; B=0.6;
            Rw=0.50;Gw=0.00;Bw=0.16;
            break;

        case 5:
           MessageBox(NULL,
                    "Aye Aye Pilot! Welcome to our game ! \n\n"
                    "The only objective for you is to go as far as possible without crash landing. "
                    "Try to score maximum points by passing through hoops that come along the way. "
                    "Whenever you pass through a hoop, your score will be incremented by 1! \n\n\n"
```

```
                        "Here are some instructions for you that will help you maneuver the plane better !
\n"
                        "Press G - to start a new game\n"
                        "Press M - to get back to the main page\n\n"

                        "Press W or UP arrow key - to go Up\n"
                        "Press S or DOWN arrow key - to go Down\n"
                        "Press A or LEFT arrow key - to go Left\n"
                        "Press D or RIGHT arrow key - to go Right\n"
                        "Press z to zoom IN and SHIFT+z to zoom OUT\n\n"

                        "Right Click Mouse Button to CUSTOMISE your plane color!\n"

                        "PRESS 'h' to acquaint yourself with controls in case you forget!"

                        "\n\nALL THE BEST and hope you beat the high score :P"

                    ,"INSTRUCTIONS", MB_OK);
            break;

          case 6:
            exit(0);
            break;

     }
       glutPostRedisplay();

}

int main(int argc, char *argv[])
{
   glutInit(&argc, argv);
   glutInitWindowPosition(0,0);
           glutInitWindowSize(1366,720);
   glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGBA);

   glutCreateWindow("PILOT YOUR PLANE");


   ///menu
   int sub1 = glutCreateMenu(GoMenu);

   glutAddMenuEntry("Purple",1);
   glutAddMenuEntry("Blue",2);
   glutAddMenuEntry("Orange",3);
   glutAddMenuEntry("Maroon",4);
   glutCreateMenu(GoMenu);
   glutAddSubMenu("PLANE COLORS", sub1);

   glutAddMenuEntry("HELP",5);
   glutAddMenuEntry("EXIT",6);
```

```
    glutAttachMenu(GLUT_RIGHT_BUTTON);


    glutReshapeFunc(resize);
    glutDisplayFunc(display);
    glutKeyboardFunc(key);
    glutSpecialFunc(processSpecialKeys);
    glutIdleFunc(idle);

    glClearColor(1,1,1,1);
    glEnable(GL_CULL_FACE);
    glCullFace(GL_BACK);

    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LESS);

    glEnable(GL_LIGHT0);
    glEnable(GL_NORMALIZE);
    glEnable(GL_COLOR_MATERIAL);
    glEnable(GL_LIGHTING);

    glLightfv(GL_LIGHT0, GL_AMBIENT,  light_ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE,  light_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);

    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, high_shininess);

    glutMainLoop();

    return EXIT_SUCCESS;
}
```

6. CONCLUSION

The project 'Pilot Your Plane' implemented on Windows platform using C++ was completed successfully. It involves various translating, scaling and rotating effects effectively bringing out a simple animation. The utilities and functions provided by the OpenGL have been used in an optimal way to achieve a completely working project. The work done during the process of completion of this project has helped us understand the various functionalities available in a better and a more practical environment and we realize that the scope of OpenGL platform has a premier game developing launch pad. Hence it has indeed been useful in developing many games. OpenGL in its own right is good for low cost and simple game development. It serves as an important stepping stone for venturing into other fields of Computer Graphics design and applications. 3D animation, modeling, gaming etc. are the next big things, and working on this project has helped us learn the basics required for it. We have implemented this project for entertainment purposes and this game is very easy to play.