RV COLLEGE OF ENGINEERING
BENGALURU – 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

"Stunt Plane – OpenGL Project"

COMPUTER GRAPHICS (16CS73)
ASSIGNMENT

OPEN ENDED EXPERIMENT REPORT

VII SEMESTER

2020-2021

Submitted by

GS SUNDAR – 1RV17CS190
R MAHESH – 1RV17CS191
Under the Guidance of

Prof. Mamatha T
Assistant Professor
Department of CSE, R.V.C.E., Bengaluru - 560059

RV COLLEGE OF ENGINEERING, BENGALURU - 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Open-Ended Experiment titled "Stunt Plane – OpenGL Project" has been carried out by **GS Sundar (1RV17CS190)** and **R Mahesh (1RV17CS191)**, bonafide students of R.V. College of Engineering, Bengaluru, have submitted in partial fulfillment for the Internal Assessment of Course: COMPUTER GRAPHICS (16CS73) Assignment during the year 2019-2020. It is certified that all corrections/suggestions indicated for the internal Assessment have been incorporated in the report.

Prof. Mamatha T
Faculty Incharge,
Department of CSE,
R.V.C.E., Bengaluru –59

Dr. Ramakanth Kumar P
Head of Department,
Department of CSE,
R.V.C.E., Bengaluru–59

RV COLLEGE OF ENGINEERING, BENGALURU - 560059
(Autonomous Institution Affiliated to VTU)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, **GS Sundar** and **R Mahesh** the students of Seventh Semester B.E., Computer Science and Engineering, R.V. College of Engineering, Bengaluru hereby declare that the mini-project titled "Stunt Plane – OpenGL Project" has been carried out by us and submitted in partial fulfilment for the Internal Assessment of Course:

COMPUTER GRAPHICS (16CS73) Assignment - Open-Ended Experiment

during the year 2019-2020. We do declare that matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

Place: Bengaluru

Date:

GS SUNDAR

R MAHESH

CONTENTS

Abstract

- Main aim of this Mini Project is to illustrate the concepts and usage of pre-built functions in OpenGL.
- Creating objects and games like StuntPlane using Opengl library.
- Functions like glutSolidSphere and glutSolidCube are used to create the plane.
- The rings were created using torus function provided in glut library.
- We have used input devices like mouse and key board to interact with program

System specifications

- SOFTWARE REQUIREMENTS :
  - MICROSOFT VISUAL C++
  - OPENGL
- HARDWARE REQUIREMENT :
  - GRAPHICS SYSTEM,
  - Pentium P4 with 256 of Ram(Min)

Implementation

This program is implemented using various openGL functions which are shown below.

Various functions used in this program.

- glutInit() : interaction between the windowing system and OPENGL is initiated
- glutInitDisplayMode() : used when double buffering is required and depth information is required
- glutCreateWindow() : this opens the OPENGL window and displays the title at top of the window
- glutInitWindowSize() : specifies the size of the window
- glutInitWindowPosition() : specifies the position of the window in screen co-ordinates
- glutKeyboardFunc() : handles normal ascii symbols
- glutSpecialFunc() : handles special keyboard keys
- glutReshapeFunc() : sets up the callback function for reshaping the window
- glutIdleFunc() : this handles the processing of the background
- glutDisplayFunc() : this handles redrawing of the window
- glutMainLoop() : this starts the main loop, it never returns
- glViewport() : used to set up the viewport

- glVertex3fv() : used to set up the points or vertices in three dimensions
- glColor3fv() : used to render color to faces
- glFlush() : used to flush the pipeline
- glutPostRedisplay() : used to trigger an automatic redrawal of the object
- glMatrixMode() : used to set up the required mode of the matrix
- glLoadIdentity() : used to load or initialize to the identity matrix
- glTranslatef() : used to translate or move the rotation centre from one point to another in three dimensions
- glRotatef() : used to rotate an object through a specified rotation angle

Interaction with program

 This program includes interaction through keyboard.

- S Start the Project
- Use keys A,D,W,S to control the moment of Plane.
- Q-> Quit

Source Code

```
#include<string.h>
#include<stdarg.h>
#include<stdio.h>
#include<GL/glut.h>
#include<string>
static double x1=0.0;
static double x2=0.0;
static double a1=0.0;
static double r1=0.0;
static double r2=0.0;
static double r3=0.0;
static double r4=0.0;

static double move=0.0;
static double move_y=0.0;

static double z1=0.0;
static double speed=0.0;
static double interval=40.0;
static int final_score=0.0;
static int final_final_score=-1.0;
static int done = 0;
static double xp = 0.0;
static double yp = 0.0;

void stroke_output(GLfloat x, GLfloat y, char *format,...) {
        va_list args;
        char buffer[200], *p;
        va_start(args, format);
        vsprintf(buffer, format, args);
        va_end(args);
```

```
        glPushMatrix();
        glTranslatef(-2.5, y, 0);
        glScaled(0.003, 0.005, 0.005);
        for (p = buffer; *p; p++)
        glutStrokeCharacter(GLUT_STROKE_ROMAN, *p);
        glPopMatrix();
}

void Circle() {
        glColor3f(1.0,0.0,1.0);
        glutSolidTorus(0.4,3.5,50,50);
}


void drawPlane() {
        glPushMatrix();

        // Main Body
        glPushMatrix();
        glScalef(.3,0.3,1.5);
        glColor3f(1,0,0.3);
        glutSolidSphere(2.0,50,50);
        glPopMatrix();
        glPushMatrix();
        glTranslatef(0.0,0.1,-1.8);
        glScalef(1.0,1,1.5);
        glColor3f(1,1,0);
        glutSolidSphere(0.5,25,25);
        glPopMatrix();


        //Left Fin
        glPushMatrix();
        glTranslatef(-1.0,0,0);
        glScalef(1.5,0.1,0.5);
```

```
glColor3f(1,1,0);
glutSolidSphere(1.0,50,50);
glPopMatrix();

// Right Fin
glPushMatrix();
glTranslatef(1.0,0,0);
glScalef(1.5,0.1,0.5);
glColor3f(1,1,0);
glutSolidSphere(1.0,50,50);
glPopMatrix();

//right Tail fin
glPushMatrix();
glTranslatef(0.8,0,2.4);
glScalef(1.2,0.1,0.5);
glColor3f(0.0,0,1);
glutSolidSphere(0.4,50,50);
glPopMatrix();

//left Tail fin
glPushMatrix();
glTranslatef(-0.8,0,2.4);
glScalef(1.2,0.1,0.5);
glColor3f(0.0,0,1);
glutSolidSphere(0.4,50,50);
glPopMatrix();

//Top tail fin
glPushMatrix();
glTranslatef(0,0.5,2.4);
glScalef(0.1,1.1,0.5);
glColor3f(0.0,0,1);
glutSolidSphere(0.4,50,50);
glPopMatrix();
```

```
// Blades
glPushMatrix();
glRotatef(x2,0.0,0.0,1.0);
glPushMatrix();
glTranslatef(0,0.0,-3.0);
glScalef(1.5,0.2,0.1);
glColor3f(0.0,0,1);
glutSolidSphere(0.3,50,50);
glPopMatrix();

//Blades
glPushMatrix();
glRotatef(90,0.0,0.0,1.0);
glTranslatef(0,0.0,-3.0);
glScalef(1.5,0.2,0.1);
glColor3f(0.0,0,1);
glutSolidSphere(0.3,50,50);
glPopMatrix();

glPopMatrix();

//Front
glPushMatrix();
glTranslatef(0.0,-0.8,-1.5);
glRotatef(90,0.0,1,0);
glScaled(0.3,0.3,0.3);
glutSolidTorus(0.18,0.5,25,25);
glColor3f(1,1,0);
glutSolidTorus(0.2,0.1,25,25);
glPopMatrix();
glPushMatrix();
glTranslatef(0.0,-0.4,-1.5);
glRotatef(20,0.0,1,0);
glScaled(0.05,0.3,0.05);
```

```
glutSolidSphere(1.0,25,25);
glPopMatrix();

//Rear
glPushMatrix();
glTranslatef(0.3,-0.8,0.7);
glRotatef(90,0.0,1,0);
glScaled(0.3,0.3,0.3);
glColor3f(0,0,1);
glutSolidTorus(0.18,0.5,25,25);
glColor3f(1,1,0);
glutSolidTorus(0.2,0.1,25,25);
glPopMatrix();
glPushMatrix();
glTranslatef(0.3,-0.4,0.7);
glRotatef(20,0.0,1,0);
glScaled(0.05,0.3,0.05);
glutSolidSphere(1.0,25,25);
glPopMatrix();

//Rear 2
glPushMatrix();
glTranslatef(-0.3,-0.8,0.7);
glRotatef(90,0.0,1,0);
glScaled(0.3,0.3,0.3);
glColor3f(0,0,1);
glutSolidTorus(0.18,0.5,25,25);
glColor3f(1,1,0);
glutSolidTorus(0.2,0.1,25,25);
glPopMatrix();
glPushMatrix();
glTranslatef(-0.3,-0.4,0.7);
glRotatef(20,0.0,1,0);
glScaled(0.05,0.3,0.05);
glutSolidSphere(1.0,25,25);
```

```
        glPopMatrix();
        glPopMatrix();
}

void plane() {
        glClearColor(1,1,1,0.0);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();
   glTranslatef(0.0f,0.0f,-13.0f);
        stroke_output(-2.5,2, "Press s to start.");

        glPushMatrix();
        glRotatef(x1,0.0,1.0,0.0);
        drawPlane();
        glPopMatrix();

        glFlush();
        glutSwapBuffers();
}

void gameOver(){
        glClearColor(1.0,1.0,1.0,0.0);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();
   glTranslatef(0.0f,0.0f,-13.0f);
        stroke_output(-2.5,2, "The plane crashed.");
        std::string str1 = "Score: ";
        str1.append(std::__cxx11::to_string(final_final_score));
        int n = str1.length();
        char temp[n+1];
        strcpy(temp, str1.c_str());
        stroke_output(-2.5,1, temp);

        glBegin(GL_POLYGON);
                glColor3f(0.0,1.0,0.0);
```

```
                glVertex3f(-100,-5,100);
                glVertex3f(100,-5,100);
                glVertex3f(100,-5,-100);
                glVertex3f(-100,-5,-100);
        glEnd();

        glFlush();

}

static double obstacles[6][2] = {{0,1},{-2,0},{2,0},{1,0},{2,0},{-2,0}};

double distance(double x1,double y1,double x2,double y2) {
        return (x1-x2)*(x1-x2)+(y1-y2)*(y1-y2);
}

void fighterPlane()
{
        if (int(z1/30.0)>done) {
                double dist = distance(xp, -1+yp, obstacles[6][done%6], -
1+obstacles[6][done%6]);
                if (dist<3.4*3.4) {
                        final_score++;
                }
                done++;
        }

        glClearColor(0.53,0.81,0.92,0.0);
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();

        glTranslatef(0.0f,0.0f,-13.0f);
                std::string str1 = "Score: ";
        str1.append(std::__cxx11::to_string(final_score));
        int n = str1.length();
```

```
char temp[n+1];
strcpy(temp, str1.c_str());

char *p;
glPushMatrix();
glTranslatef(-4.5, 3, 0);
glScaled(0.003, 0.005, 0.005);
for (p = temp; *p; p++)
glutStrokeCharacter(GLUT_STROKE_ROMAN, *p);
glPopMatrix();

str1 = "Distance: ";
str1.append(std::__cxx11::to_string(z1));
n = str1.length();
strcpy(temp, str1.c_str());
glPushMatrix();
glTranslatef(-4.5, 2, 0);
glScaled(0.003, 0.005, 0.005);
for (p = temp; *p; p++)
glutStrokeCharacter(GLUT_STROKE_ROMAN, *p);
glPopMatrix();

if(move_y<=-4) {
        if (final_final_score == -1.0)
        final_final_score = final_score;
        gameOver();
} else {
        // Floor
        glBegin(GL_POLYGON);
                glColor3f(0.0,1.0,0.0);
                glVertex3f(-100,-5,100);
                glVertex3f(100,-5,100);
                glVertex3f(100,-5,-100);
                glVertex3f(-100,-5,-100);
        glEnd();
```

```
int i = int(z1/180);
        // draw the obstacles
        glPushMatrix();
        glTranslatef(0,1,-180*i-30+z1);
        Circle();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(-2,0,-180*i-60+z1);
        Circle();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(2,0,-180*i-90+z1);
        Circle();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(1,0,-180*i-120+z1);
        Circle();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(2,0,-180*i-150+z1);
        Circle();
        glPopMatrix();

        glPushMatrix();
        glTranslatef(-2,0,-180*i-180+z1);
        Circle();
        glPopMatrix();

// Call drawPlane
glPushMatrix();
```

```
                glTranslatef(move,-1.0+move_y,0);
                glRotatef(r1,0.0,0.0,1.0);
                glRotatef(r2,1.0,0.0,0.0);
                drawPlane();
                glPopMatrix();
        }
        glFlush();
   glutSwapBuffers();
}

void s() {
        x1+=0.3;
        fighterPlane();
}

void start() {
        x1+=0.3;
        x2+=5.0;
        z1+=speed;
        fighterPlane();
}

void p1() {
        x2+=10.0;
        plane();
}


void doInit()
{

        /* Background and foreground color */
   glClearColor(0.0,0.0,0.0,0.0);
   glColor3f(.0,1.0,1.0);
```

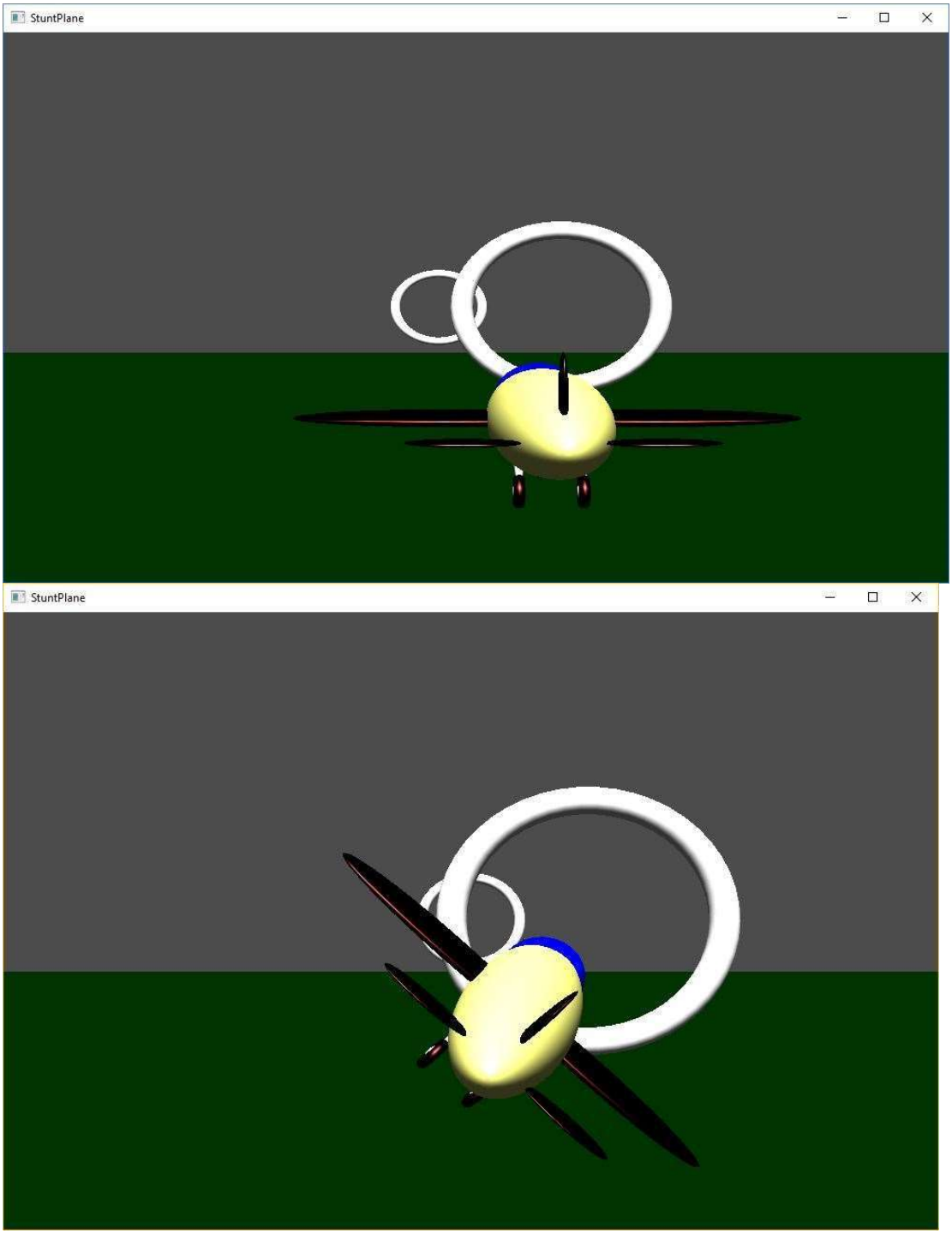
```
        /* Select the projection matrix and reset it then
     setup our view perspective */
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(30.0f,(GLfloat)640/(GLfloat)480,0.1f, 50.0f);
    /* Select the modelview matrix, which we alter with rotatef() */
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glClearDepth(2.0f);
        glEnable(GL_COLOR_MATERIAL);
    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LEQUAL);
}

void doDisplay()
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
        glTranslatef(0.0f,0.0f,-13.0f);
        glPushMatrix();
                glScaled(0.7,0.7,0.7);
                stroke_output(-2.0, 1.7, "Easy, Medium or Hard?");
                stroke_output(-2.0, 0.7, "Press E, M or H.");
        glPopMatrix();
        GLfloat mat_ambient[]={0.0f,1.0f,2.0f,1.0f};
        GLfloat mat_diffuse[]={0.0f,1.5f,.5f,1.0f};
        GLfloat mat_specular[]={5.0f,1.0f,1.0f,1.0f};
        GLfloat mat_shininess[]={50.0f};
        glMaterialfv(GL_FRONT,GL_AMBIENT,mat_ambient);
        glMaterialfv(GL_FRONT,GL_DIFFUSE,mat_diffuse);
        glMaterialfv(GL_FRONT,GL_SPECULAR,mat_specular);
        glMaterialfv(GL_FRONT,GL_SHININESS,mat_shininess);

        GLfloat lightIntensity[]={1.7f,1.7f,1.7f,1.0f};
        GLfloat light_position3[]={0.0f,5.0f,5.0f,0.0f};
```

```
        glLightfv(GL_LIGHT0,GL_POSITION,light_position3);
        glLightfv(GL_LIGHT0,GL_DIFFUSE,lightIntensity);


        glFlush();
        glutSwapBuffers();

}



void mykey(unsigned char key,int x,int y)
{
        if(key=='s')
        {
                glutIdleFunc(start);
        }

        if(key=='S')
        {
                glutIdleFunc(s);
        }

        if(key=='e'|| key=='E')
        {
                x1+=1.3;
                speed=0.3;
                glutIdleFunc(p1);
        }

        if(key=='m'|| key=='M')
        {
                x1+=1.3;
                speed=0.8;
                glutIdleFunc(p1);
```

```
        }

        if(key=='h'|| key=='H')
        {
                x1+=1.3;
                speed=1.5;
                glutIdleFunc(p1);
        }

        if(key=='q'|| key=='Q'){
        exit(0);

        }

}

static void specialKey(int key,int x,int y) {

        if(key==GLUT_KEY_DOWN){
        r1=0;
        r2=-10;
         //lower the nose of plane
                move_y-=0.5;
                yp = yp-0.5;

        }

        if(key==GLUT_KEY_UP){
                yp = yp+0.5;
                move_y+=0.5;
                r1=0;
                r2=10;
                // raise the nose of plane
                }
        if(key==GLUT_KEY_LEFT){
```

```
                xp = xp-0.1;
                r1=45;
                        r2=10;
                        move-=0.1;

                }
        if(key==GLUT_KEY_RIGHT){
                xp = xp+0.1;
                r1=-45;
                        r2=10;
                        move+=0.1;
                }
}


int main(int argc, char *argv[])
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
   glutInitWindowSize(640,480);
   glutInitWindowPosition(30,0);
   glutCreateWindow("StuntPlane");
   glutDisplayFunc(doDisplay);
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glShadeModel(GL_SMOOTH);
        glEnable(GL_DEPTH_TEST);
        glEnable(GL_NORMALIZE);
        glutKeyboardFunc(mykey);
        glutSpecialFunc(specialKey);
        doInit();
   glutMainLoop();
        return 0;
}
```

OUTPUT OF THE PROGRAM




StuntPlane
you crashed my plane

Conclusion

The project “Stunt Plane” clearly demonstrates the usage of inbuilt functions of OpenGL library.
Finally, we conclude that this program clearly illustrates the concept of computer graphics using openGL and has been completed successfully and is ready to be demonstrated.

Bibliography

WE HAVE OBTAINED INFORMATION FROM MANY RESOURCES TO DESIGN AND IMPLEMENT OUR PROJECT SUCCESSIVELY. WE HAVE ACQUIRED MOST OF THE KNOWLEDGE FROM RELATED WEBSITES. THE FOLLOWING ARE SOME OF THE RESOURCES :

- TEXT BOOKS :
  INTERACTIVE COMPUTER GRAPHICS A TOP-DOWN APPROACH
  -By Edward Angel.

- COMPUTER GRAPHICS,PRINCIPLES & PRACTICES
  - Foley van dam
  - Feiner hughes

- WEB REFERENCES:
  http://jerome.jouvie.free.fr/OpenGl/Lessons/Lesson3.php
  http://google.com
  http://opengl.org