CG LAB RECORD

NATIONAL INSTITUTE OF TECHNOLOGY MANIPUR

COMPUTER GRAPHICS LAB

7TH SEMESTER (JULY-NOV 2022)

SUBMITTED BY :-

ABHIJEET ANAND

ROLL : 19103001

SUBMITTED TO :-

MERINA MA'AM

CG LAB CSE Department

CONTENTS

EXPERIMENT – 1

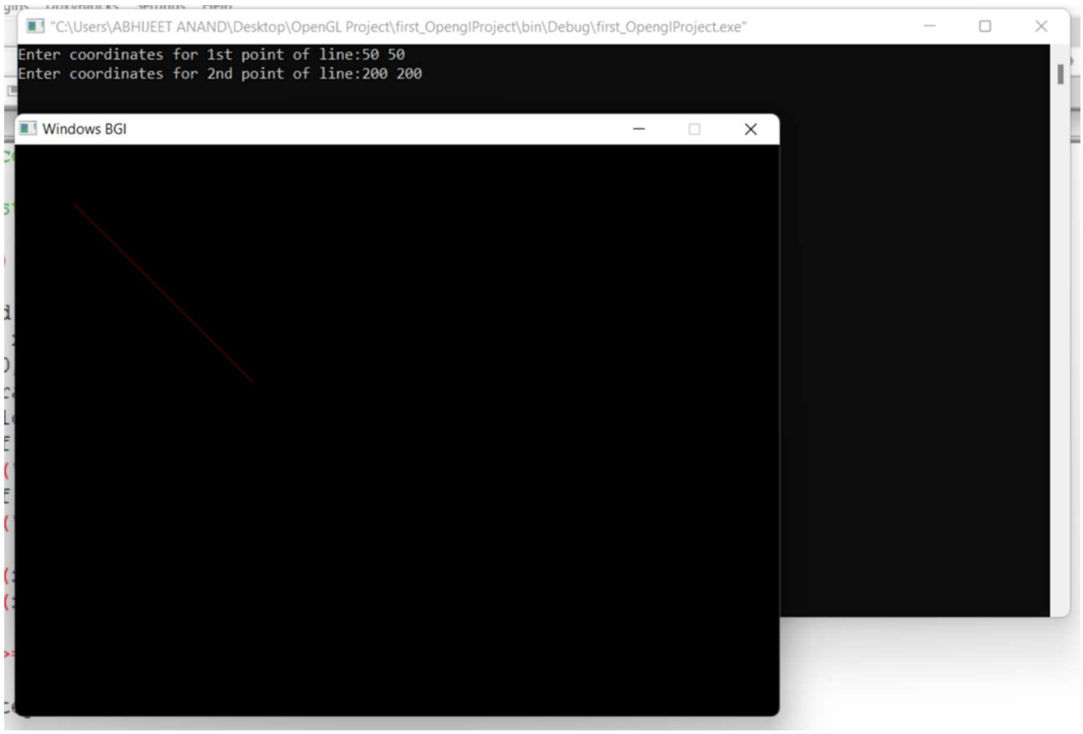
AIM :- Write a C program to draw a line using the DDA algorithm.

PROGRAM :-

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
int main() {
   int gd = DETECT ,gm, i;
   float x, y,dx,dy,steps;
   int x0, x1, y0, y1;
   initgraph(&gd, &gm, "C:\\TC\\BGI");
   setcolor(WHITE);
   printf("Enter coordinates for 1st point of line:");
   scanf("%d%d",&x0,&y0);
   printf("Enter coordinates for 2nd point of line:");
   scanf("%d%d",&x1,&y1);

   dx = (float)(x1 - x0);
   dy = (float)(y1 - y0);
   if(dx>=dy){
      steps = dx;
   } else {
      steps = dy;
   }

   dx = dx/steps;
   dy = dy/steps;
```

```
    x = x0;
    y = y0;
    i = 1;
    while(i<= steps) {
       putpixel(x, y, RED);
       x += dx;
       y += dy;
       i=i+1;
    }
    getch();
    closegraph();
}
```

OUTPUT :-



EXPERIMENT – 2

AIM :- Write a C program to draw a line using OpenGL.

PROGRAM :-

```
#include <GL/glut.h>
#include <stdlib.h>
static void init(void){
   glClearColor(0.0,0.0,0.0,0.0);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity() ;
   glOrtho(-50.0,50.0,-50.0,50.0, -1.0, 1.0);
}
static void display(void){
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(1.0,1.0,0.0);
   glBegin(GL_LINES);
   glLineWidth(2.5);
   glVertex3f(-45,30,0);
   glVertex3f(-30,30,0);
   glEnd();
   glFlush();
}
int main(int argc, char *argv[]){
   glutInit(&argc, argv);
   glutInitWindowSize(1400, 1000);
   glutInitWindowPosition(0,0);
   glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
   glutCreateWindow("GLUT Line");
```

```
    glutDisplayFunc(display);
    init();
    glutMainLoop();
    return EXIT_SUCCESS;
}
```

OUTPUT :-

EXPERIMENT – 3

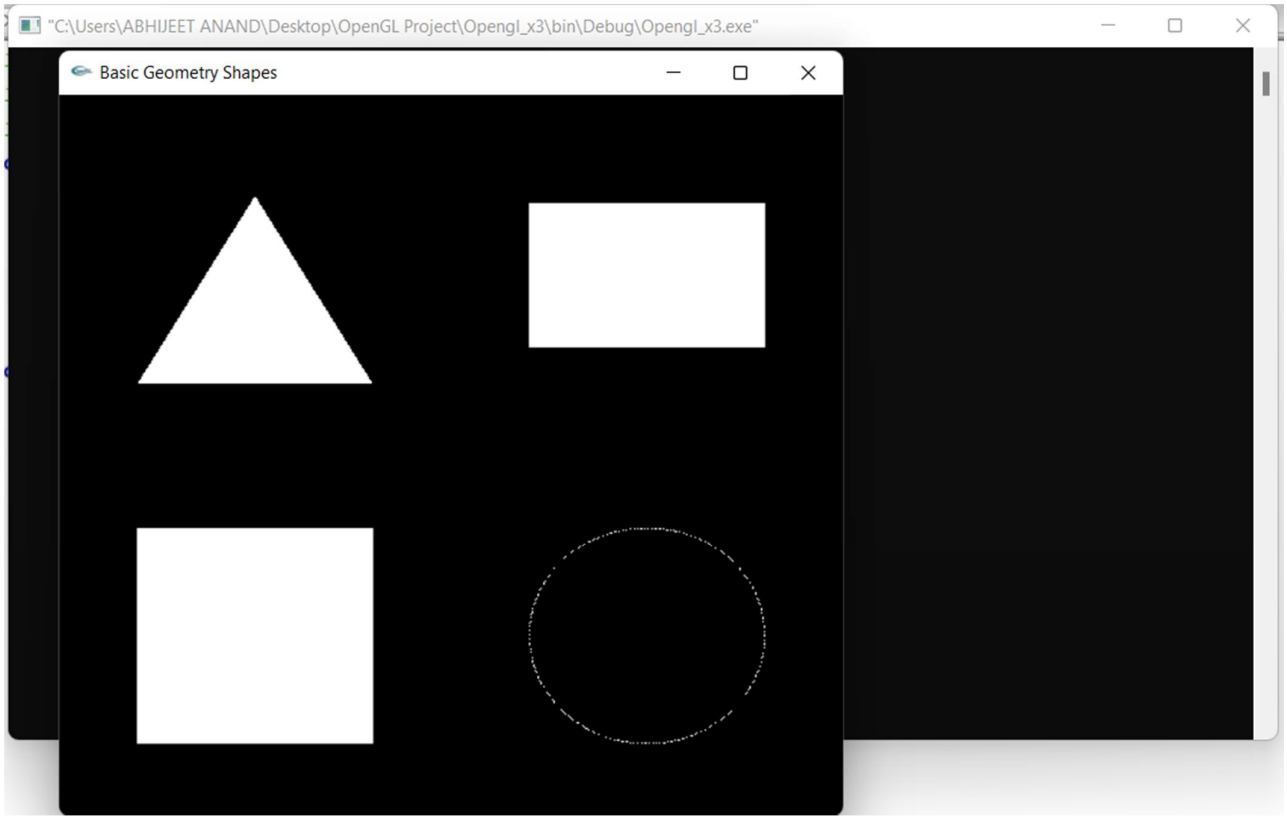
AIM : Write a C program to draw basic shapes of geometry using OpenGL.

PROGRAM :-

```
#include<Gl/glut.h>
#include<stdlib.h>
#include<math.h>
void init(void){
   glClearColor(0.0,0.0,0.0,0.0);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(0.0,1.0,0.0,1.0);
}
void display(void){
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(1.0,1.0,1.0);
   glBegin(GL_TRIANGLES); //triangle
   glVertex2f(0.1,0.6);
   glVertex2f(0.4,0.6);
   glVertex2f(0.25,0.86);
   glEnd();
   glBegin(GL_QUADS); //rectangle
   glVertex2f(0.6,0.85);
   glVertex2f(0.9,0.85);
   glVertex2f(0.9,0.65);
   glVertex2f(0.6,0.65);
   glEnd();
   glBegin(GL_QUADS); //square
   glVertex2f(0.1,0.1);
   glVertex2f(0.1,0.4);
   glVertex2f(0.4,0.4);
   glVertex2f(0.4,0.1);
   glEnd();
   float angle,x,y;
   glBegin(GL_LINES); //circle
```

```
    for (angle=0.0f; angle<=(2.0f*M_PI); angle+=0.01f){
        x = 0.15f * sin(angle);
        y = 0.15f * cos(angle);
        glVertex2f(0.75+x,0.25+y);
    }
    glEnd();
    glFlush();
}
int main(int argc, char **argv){
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500.0,500.0);
    glutCreateWindow("Basic Geometry Shapes");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

OUTPUT :-



EXPERIMENT – 4

AIM :- Write a C program to draw a cube using OpenGL.

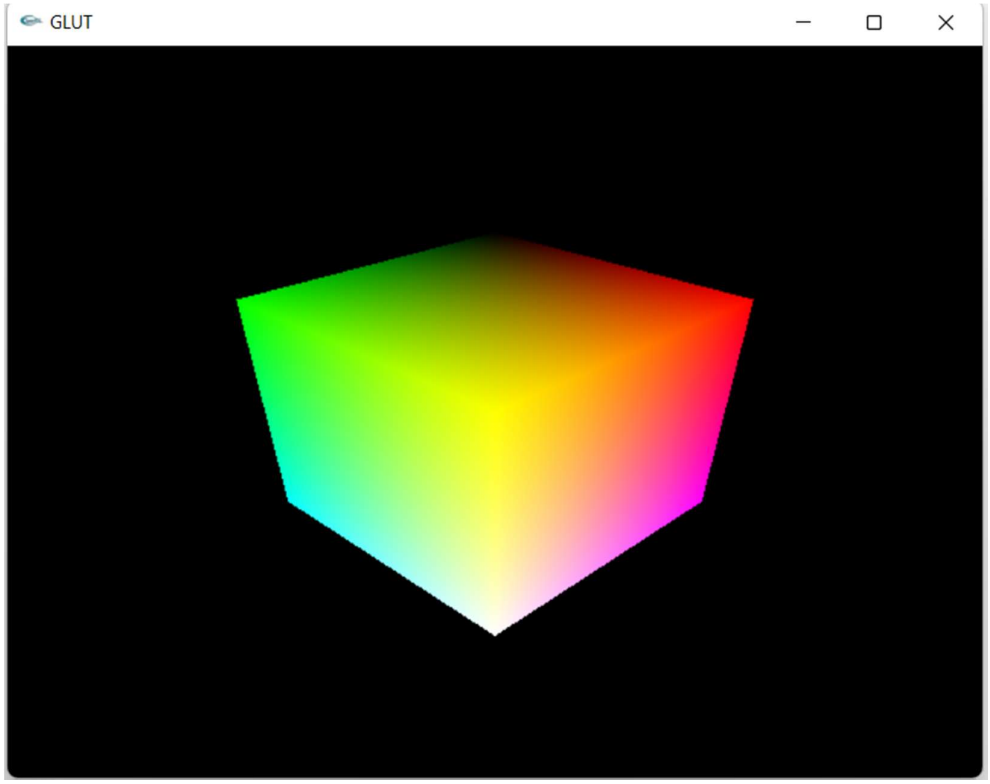
PROGRAM :-

```
#include <GL/glut.h>
float ver[8][3] =
{
   {-1.0,-1.0,1.0},
   {-1.0,1.0,1.0},
   {1.0,1.0,1.0},
   {1.0,-1.0,1.0},
   {-1.0,-1.0,-1.0},
   {-1.0,1.0,-1.0},
   {1.0,1.0,-1.0},
   {1.0,-1.0,-1.0},
};

GLfloat color[8][3] =
{
   {0.0,0.0,0.0},
   {1.0,0.0,0.0},
   {1.0,1.0,0.0},
   {0.0,1.0,0.0},
   {0.0,0.0,1.0},
   {1.0,0.0,1.0},
   {1.0,1.0,1.0},
   {0.0,1.0,1.0},
};

void quad(int a,int b,int c,int d) {
   glBegin(GL_QUADS);
   glColor3fv(color[a]);
   glVertex3fv(ver[a]);

   glColor3fv(color[b]);
   glVertex3fv(ver[b]);

   glColor3fv(color[c]);
```

```
    glVertex3fv(ver[c]);

    glColor3fv(color[d]);
    glVertex3fv(ver[d]);
    glEnd();
}
void colorcube() {
    quad(0,3,2,1);
    quad(2,3,7,6);
    quad(0,4,7,3);
    quad(1,2,6,5);
    quad(4,5,6,7);
    quad(0,1,5,4);
}
double rotate_y = 0;
double rotate_x = 0;
void specialKeys( int key, int x, int y ) {
    if (key == GLUT_KEY_RIGHT)
        rotate_y += 5;
    else if (key == GLUT_KEY_LEFT)
        rotate_y -= 5;
    else if (key == GLUT_KEY_UP)
        rotate_x += 5;
    else if (key == GLUT_KEY_DOWN)
        rotate_x -= 5;
    glutPostRedisplay();
}
void display() {
    glClearColor( 0, 0, 0, 1 );
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    int w = glutGet( GLUT_WINDOW_WIDTH );
    int h = glutGet( GLUT_WINDOW_HEIGHT );
    gluPerspective( 60, w / h, 0.1, 100 );

    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
    gluLookAt (
        3, 3, 3,
```

```
        0, 0, 0,
        0, 0, 1
        );
    glRotatef( rotate_x, 1.0, 0.0, 0.0 );
    glRotatef( rotate_y, 0.0, 1.0, 0.0 );
    colorcube();
    glutSwapBuffers();
}
int main( int argc, char **argv ){
    glutInit( &argc, argv );
    glutInitDisplayMode( GLUT_RGBA | GLUT_DEPTH | GLUT_DOUBLE );
    glutInitWindowSize( 640, 480 );
    glutCreateWindow( "GLUT" );
    glutDisplayFunc( display );
    glutSpecialFunc( specialKeys );
    glEnable( GL_DEPTH_TEST );
    glutMainLoop();
    return 0;
}
```

OUTPUT :-



EXPERIMENT – 5

AIM :- Write a C program to draw a line and show translation, rotation and scaling motion of the line using OpenGL.

Translation

PROGRAM :-

```
#include<bits/stdc++.h>
#include<graphics.h>
using namespace std;
void translateLine ( int P[][2], int T[]) {
        int gd = DETECT, gm, errorcode;
        initgraph (&gd, &gm, "c:\\tc\\bgi");

        // drawing original line using graphics functions
        setcolor (2);
        line(P[0][0], P[0][1], P[1][0], P[1][1]);

        // calculating translated coordinates
        P[0][0] = P[0][0] + T[0];
        P[0][1] = P[0][1] + T[1];
        P[1][0] = P[1][0] + T[0];
        P[1][1] = P[1][1] + T[1];

        // drawing translated line using graphics functions
        setcolor(3);
        line(P[0][0], P[0][1], P[1][0], P[1][1]);
        getch();
        closegraph();
}
// driver program
int main()
{
        int P[2][2] = {50, 80, 120, 180}; // coordinates of point
        int T[] = {20, 10}; // translation factor
        translateLine (P, T);
```

```
	//getch();
	return 0;
}
```

OUTPUT :-

Scaling :-

PROGRAM :-

```
#include<stdio.h>
#include<graphics.h>

void findNewCoordinate(int s[][2], int p[][1]) {
	int temp[2][1] = { 0 };
	for (int i = 0; i < 2; i++)
		for (int j = 0; j < 1; j++)
			for (int k = 0; k < 2; k++)
				temp[i][j] += (s[i][k] * p[k][j]);

	p[0][0] = temp[0][0];
	p[1][0] = temp[1][0];
}
```

```
// Scaling the Polygon
void scale(int x[], int y[], int sx, int sy) {
	// Line before Scaling
	line(x[0], y[0], x[1], y[1]);
	//line(x[1], y[1], x[2], y[2]);
	//line(x[2], y[2], x[0], y[0]);

	// Initializing the Scaling Matrix.
	int s[2][2] = { sx, 0, 0, sy };
	int p[2][1];

	// Scaling the Line
	for (int i = 0; i < 3; i++) {
		p[0][0] = x[i];
		p[1][0] = y[i];
		findNewCoordinate(s, p);
		x[i] = p[0][0];
		y[i] = p[1][0];
	}

	// LIne after Scaling
	line(x[0], y[0], x[1], y[1]);
	//line(x[1], y[1], x[2], y[2]);
	//line(x[2], y[2], x[0], y[0]);
}
int main()
{
	int x[] = { 100, 200, 300 };
	int y[] = { 200, 100, 200 };
	int sx = 2, sy = 2;
	int gd, gm;
	detectgraph(&gd, &gm);
	initgraph(&gd, &gm," ");
	scale(x, y, sx,sy);
	getch();
```

```
    return 0;
}
```

OUTPUT :-

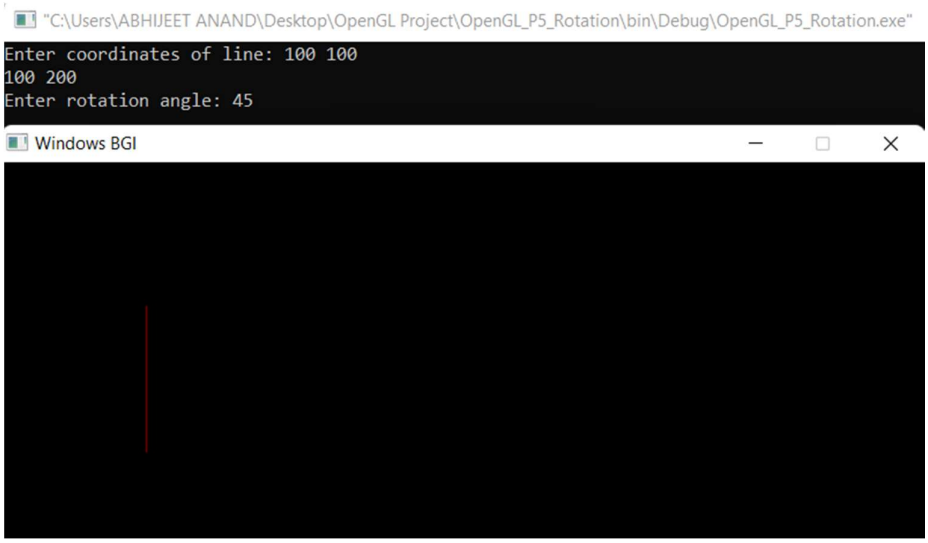
Rotation

PROGRAM :-

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
int main()
{
   int gd=0,gm,x1,y1,x2,y2;
   double s,c, angle;
   initgraph(&gd, &gm, "C:\\TC\\BGI");
   setcolor(RED);
   printf("Enter coordinates of line: ");
   scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
   cleardevice();
   setbkcolor(WHITE);
   line(x1,y1,x2,y2);
   //getch();
   setbkcolor(BLACK);
```

```
    printf("Enter rotation angle: ");
    scanf("%lf", &angle);
    //setbkcolor(WHITE);
    c = cos(angle *3.14/180);
    s = sin(angle *3.14/180);
    x1 = floor(x1 * c + y1 * s);
    y1 = floor(-x1 * s + y1 * c);
    x2 = floor(x2 * c + y2 * s);
    y2 = floor(-x2 * s + y2 * c);
    cleardevice();
    line(x1, y1 ,x2, y2);
    getch();
    closegraph();
return 0;
}
```

OUTPUT :-



EXPERIMENT – 6

AIM :- Write a C program to draw a cube and show translation, rotation and scaling motion of the line using OpenGL.

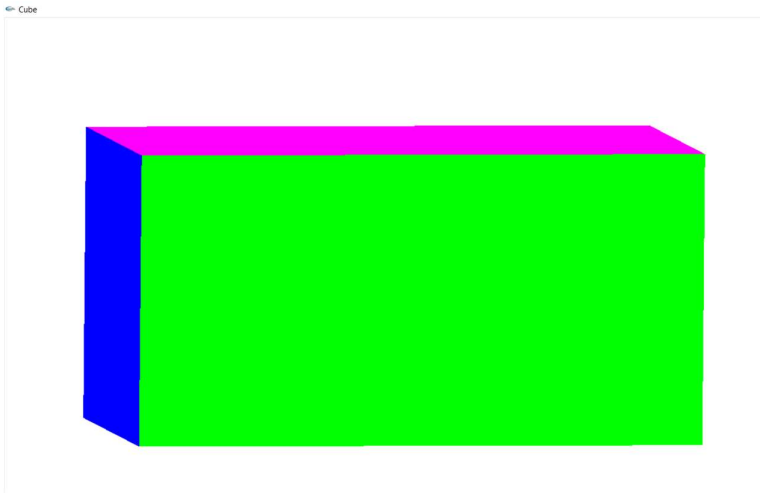
PROGRAM :-

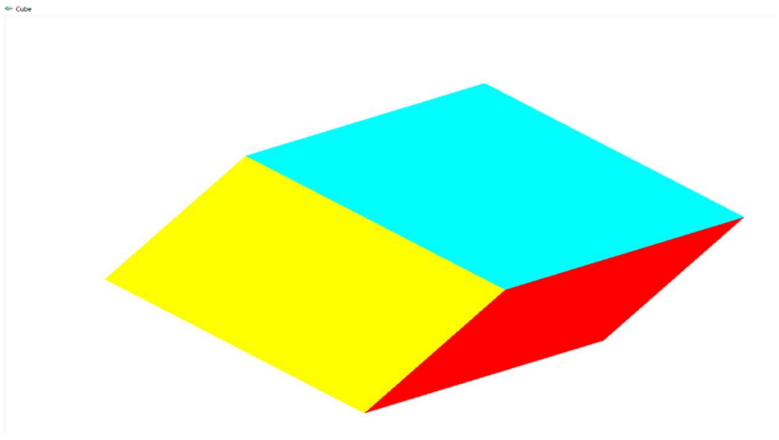
```
#include <GL/glu.h>
#include <GL/glut.h>
#include <GL/gl.h>
GLfloat T = 45;
GLfloat D = -1;
GLfloat Z = 0.01;
void MyInit() {
 glClearColor(1, 1, 1, 1);
 glColor3f(1, 0, 0);
 glEnable(GL_DEPTH_TEST);
}
void spin() {
 T = T + 0;
 if(T > 360) T = 0;
 glutPostRedisplay();
}
void translate() {
 D = D + 0.01;
 if(D > 0) D = -1;
 glutPostRedisplay();
}
void scale() {
 Z = Z + 0.01;
 if(Z > 1.1) Z = 0.01;
 glutPostRedisplay();
}
void allinone() {
 T = T + 1;
 if(T > 360) T = 0;
 D = D + 0.01;
```

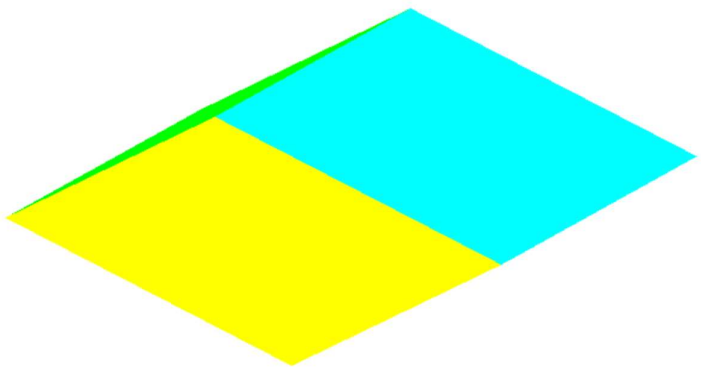
```
 if(D > 0) D = -1;
 Z = Z + 0.01;
 if(Z > 1.1) Z = 0.01;
 glutPostRedisplay();
}
void face(GLfloat a[], GLfloat b[], GLfloat c[], GLfloat d[]) {
 glBegin(GL_POLYGON);
 glVertex3fv(a);
 glVertex3fv(b);
 glVertex3fv(c);
 glVertex3fv(d);
 glEnd();
}
void cube(GLfloat v0[], GLfloat v1[], GLfloat v2[], GLfloat v3[], GLfloat v4[], GLfloat
v5[], GLfloat v6[], GLfloat v7[]) {
 glColor3f(1, 0, 0);
 face(v0, v1, v2, v3);
 glColor3f(0, 1, 0);
 face(v4, v5, v6, v7);
 glColor3f(0, 0, 1);
 face(v0, v3, v7, v4);
 glColor3f(0, 1, 1);
 face(v1, v2, v6, v5);
 glColor3f(1, 0, 1);
 face(v0, v1, v5, v4);
 glColor3f(1, 1, 0);
 face(v3, v2, v6, v7);
}
void display() {
 GLfloat v[8][3] = {
 {-0.5, 0.5, 0.5},
 {0.5, 0.5, 0.5},
 {0.5, -0.5, 0.5},
 {-0.5, -0.5, 0.5},
 {-0.5, 0.5, -0.5},
 {0.5, 0.5, -0.5},
```

```
 {0.5, -0.5, -0.5},
 {-0.5, -0.5, -0.5}
 };
 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
 glLoadIdentity();
 glRotatef(T, 1, 1, 0);
 glTranslatef(D, 0, 0);
 glScalef(Z, Z, Z);
 cube(v[0], v[1], v[2], v[3], v[4], v[5], v[6], v[7]);
 glutSwapBuffers();
}
int main(int argc, char *argv[]) {
 glutInit(&argc, argv);
 glutInitWindowPosition(100, 100);
 glutInitWindowSize(200, 200);
 glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
 if (!glutGet(GLUT_DISPLAY_MODE_POSSIBLE))
 {
 exit(1);
 }
 glutCreateWindow("Cube");
 MyInit();
 glutDisplayFunc(display);
 glutIdleFunc(allinone);
 glutMainLoop();
 return 0;
}
```

OUTPUT :-







EXPERIMENT – 7

AIM :- Write a C program to draw a house and show rising and setting of sun in between mountains using OpenGL.

PROGRAM :-

```
#include <GL/glu.h>
#include <GL/glut.h>
#include <GL/gl.h>
#include <math.h>
float D = 0;
float diff = -0.0001;
void MyInit() {
glClearColor(1, 1, 1,0);
glColor3f(1, 0, 0);
}
void rise() {
if(D < 0) diff = 0.0001;
if(D > 1) diff =-0.0001;
D += diff;
glutPostRedisplay();
}
void mountain() {
glColor3f(0.64, 0.16,0.16);
glBegin(GL_POLYGON);
glVertex2f(-1, 0);
glVertex2f(-0.5,0.5);
glVertex2f(0,0.25);
glVertex2f(0, 0);
glEnd();
glBegin(GL_POLYGON);
glVertex2f(1,0);
glVertex2f(0, 0.25);
glVertex2f(0, 0);
glEnd();
}
void garden()
{
glColor3f(0.49, 0.98,0);
glBegin(GL_POLYGON);
```

```
glVertex2f(-1, 0);
glVertex2f(-1, -1);
glVertex2f(1, -1);
glVertex2f(1, 0);
glEnd();
}
void house()
{
glColor3f(0, 0, 1);
glBegin(GL_POLYGON);
glVertex2d(-0.85,-0.5);
glVertex2d(-0.5,-0.25);
glVertex2d(-0.15, -0.5);
glEnd();
glColor3f(1, 0,0);
glBegin(GL_POLYGON);
glVertex2d(-0.75, -0.5);
glVertex2d(-0.75, -0.85);
glVertex2d(-0.25, -0.85);
glVertex2d(-0.25, -0.5);
glEnd();
}
void sun()
{
glColor3f(0, 0, 0);
glBegin(GL_POLYGON);
glVertex2f(-1, 0);
glVertex2f(-1, 1);
glVertex2f(-1, 1);
glVertex2f(1, 0);
glEnd();
float x1,y1,x2,y2;
float angle;
double radius=0.25;
x1 = 0,y1 = D;
glColor3f(1.0,1.0,0.6);
glBegin(GL_TRIANGLE_FAN);
glVertex2f(x1,y1);
for(angle=1.0f;angle<361.0f;angle+=0.2) {
   x2 = x1+sin(angle)*radius;
   y2 = y1+cos(angle)*radius;
```

```
    glVertex2f(x2,y2); }glEnd();
}
void display() {
glLoadIdentity();
sun();
mountain();
garden();
house();
glFlush();
}
int main(int argc, char *argv[])
{
glutInit(&argc, argv);
glutInitWindowPosition(100, 100);
glutInitWindowSize(250, 250);
glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE); glutCreateWindow("Scene");
MyInit();
glutDisplayFunc(display);
glutIdleFunc(rise);
glutMainLoop();
return 0;
}
```

OUTPUT :-

EXPERIMENT – 8

AIM :- Write a C program to draw a solar system showing rotation and revolution of sun, moon and earth using OpenGL.

PROGRAM :-

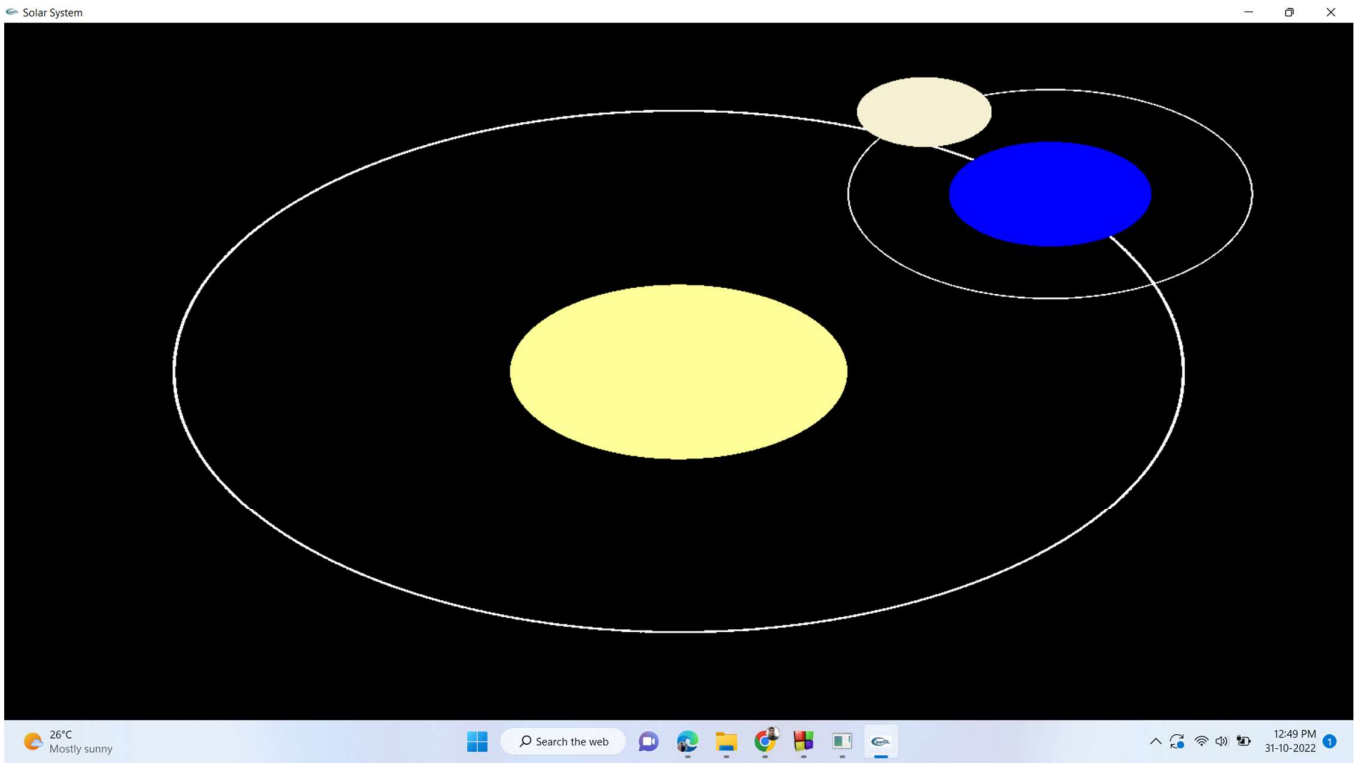
```
#include <GL/glu.h>
#include <GL/glut.h>
#include <GL/gl.h>
#include <math.h>
float diff = -0.0001;
float Er = 0.75, Eangle = 0.0;
float Mr = 0.3, Mangle =
0.0;
void MyInit() {
        glClearColor(1, 1, 1, 0);
        glColor3f(1, 0, 0);
}
void moonRevolve() {
        Mangle += 0.003;
        if(Mangle > 360) Mangle =0.0;
}
void earthRevolve() {
        Eangle += 0.001;
        if(Eangle > 360) Eangle = 0.0;
        moonRevolve();
        glutPostRedisplay();
}
void moon(float x, float y) {
        float x1,y1,x2,y2;
        float angle;
        double radius= Mr;
        x1 = x, y1 = y;
        glColor3f(1,1,1);
        glBegin(GL_LINE_LOOP);
```

```
        for (angle=0.0f;angle<270.0f;angle+=0.2) {
                x2 = x1+cos(angle)*radius;
                y2 = y1+sin(angle)*radius;
                glVertex2f(x2,y2);
        }
        glEnd();
        glColor3f(0.96, 0.94, 0.83);
        radius=0.10;
        x1 = cos(Mangle)*(Mr) + x,y1 = sin(Mangle)*(Mr) +y;
        glBegin(GL_TRIANGLE_FAN);
        glVertex2f(x1,y1);
        for (angle=1.0f;angle<361.0f;angle+=0.2){
                x2 = x1+cos(angle)*radius;
                y2 = y1+sin(angle)*radius;
                glVertex2f(x2,y2);
        }
        glEnd();
}

void sun() {
        float x1,y1,x2,y2;
        float angle;
        double radius=0.25;
        x1 = 0,y1 = 0;
        glColor3f(1.0,1.0,0.6);
        glBegin(GL_TRIANGLE_FAN);
        glVertex2f(x1,y1);
        for (angle=1.0f;angle<361.0f;angle+=0.2) {
                x2 = x1+cos(angle)*radius;
                y2 = y1+sin(angle)*radius;
                glVertex2f(x2,y2);
        }
        glEnd();
}
void earth() {
        glColor3f(0, 0, 0);
```

```
        glBegin(GL_POLYGON);
        glVertex2f(-1, -1);
        glVertex2f(-1, 1);
        glVertex2f(1, 1);
        glVertex2f(1, -1);
        glEnd();
        float x1,y1,x2,y2;
        float angle;
        double radius=Er;
        x1 = 0,y1 = 0;
        glColor3f(1,1,1);
        glBegin(GL_LINE_LOOP);
        for (angle=0.0f;angle<270.0f;angle+=0.2) {
                x2 = x1+cos(angle)*radius;
                y2 = y1+sin(angle)*radius;
                glVertex2f(x2,y2);
        }
        glEnd();
        glColor3f(0, 0, 1);
        radius=0.15;
        x1 = cos(Eangle)*Er,y1 =
        sin(Eangle)*Er;
        glBegin(GL_TRIANGLE_FAN);
        glVertex2f(x1,y1);
        for (angle=1.0f;angle<361.0f;angle+=0.2) {
                x2 = x1+cos(angle)*radius;
                y2 = y1+sin(angle)*radius;
                glVertex2f(x2,y2);
        }
        glEnd();
        moon(x1, y1);
}
void display() {
        glLoadIdentity();
        earth();
        sun();
```

```
        glFlush();
}
int main(int argc, char *argv[]) {
        glutInit(&argc, argv);
        glutInitWindowPosition(100, 100);
        glutInitWindowSize(250, 250);
        glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
        glutCreateWindow("Solar System");
        MyInit();
        glutDisplayFunc(display);
        glutIdleFunc(earthRevolve);
        glutMainLoop();
        return 0;
}
```

OUTPUT :-



THANKS