National Institute of Technology, Manipur

COMPUTER GRAPHICS LAB RECORD(CS471)

SEMESTER:7TH

TABLE OF CONTENTS

4.	Write a C program to draw a cube using OpenGL.	14-09-21	8-11
5.	Write a C program to draw a line and show translation, rotation and scaling motion of the line using OpenGL.	12-10-21	12-15
6.	Write a C program to draw a cube and show translation, rotation and scaling motion of the line using OpenGL.	02-11-21	16-20
7.	Write a C program to draw a house and show rising and setting of sun in between mountains using OpenGL.	06-11-21	20-24
8.	Write a C program to draw a solar system showing rotation and revolution of sun, moon and earth using OpenGL.	06-11-21	24-30
9.	Write a C program to draw a ball and show its bouncing motion using OpenGL.	06-11-21	30-33

Page-1

Experiment no.1

Q1. Write a C program to draw a line using DDA algorithm.

AIM : To write a C program to draw a line using DDA algorithm

CODE:

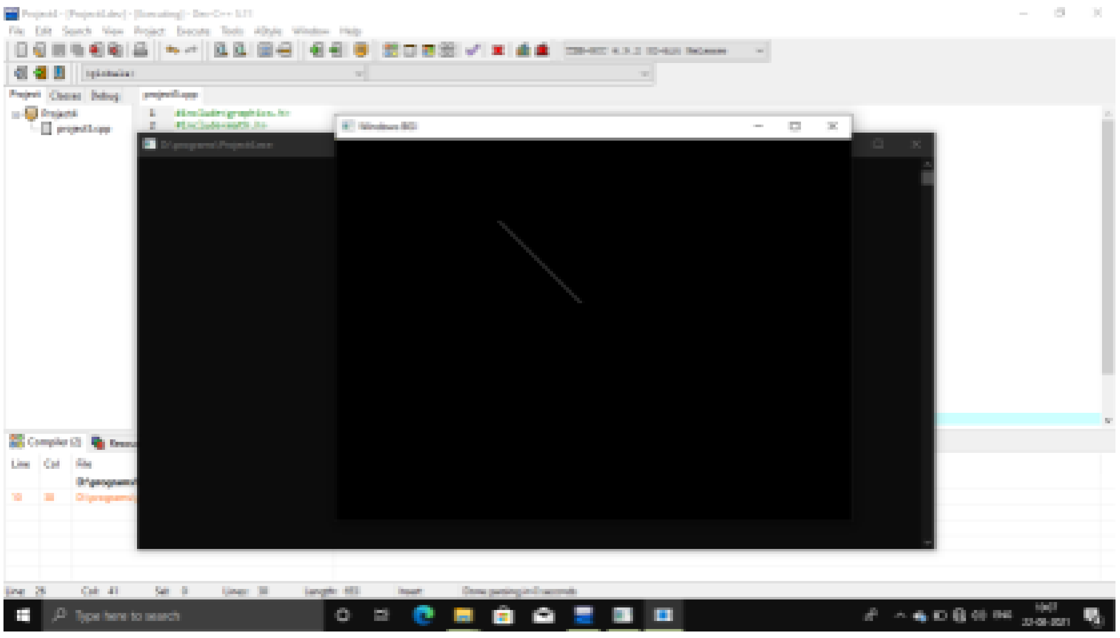
```
#include <math.h>
#include <GL/glut.h>
#include <stdlib.h>
//Initialize OpenGL
void init(void) {
glClearColor(0.0,0.0,0.0,0.0);
glMatrixMode(GL_PROJECTION);
gluOrtho2D(0.0,300.0,0.0,300.0);
}
void drawLines(void) {
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0,1.0,1.0);
glPointSize(1.0);
glBegin(GL_LINES);
glVertex2d(180, 15);
glVertex2d(10, 145);
glEnd();
glFlush();
}
int main(int argc, char**argv) {
glutInit(&argc, argv);
glutInitWindowPosition(10,10);
glutInitWindowSize(500,500);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutCreateWindow("OpenGL Line");
init();
```

```
glutDisplayFunc(drawLines);
glutMainLoop()
```

Page 2

```
}
```

OUTPUT:



Page 3

Experiment no.2

Q2.Write a C program to draw a line using OpenGL.

AIM: To write a C program to draw a line using OpenGL

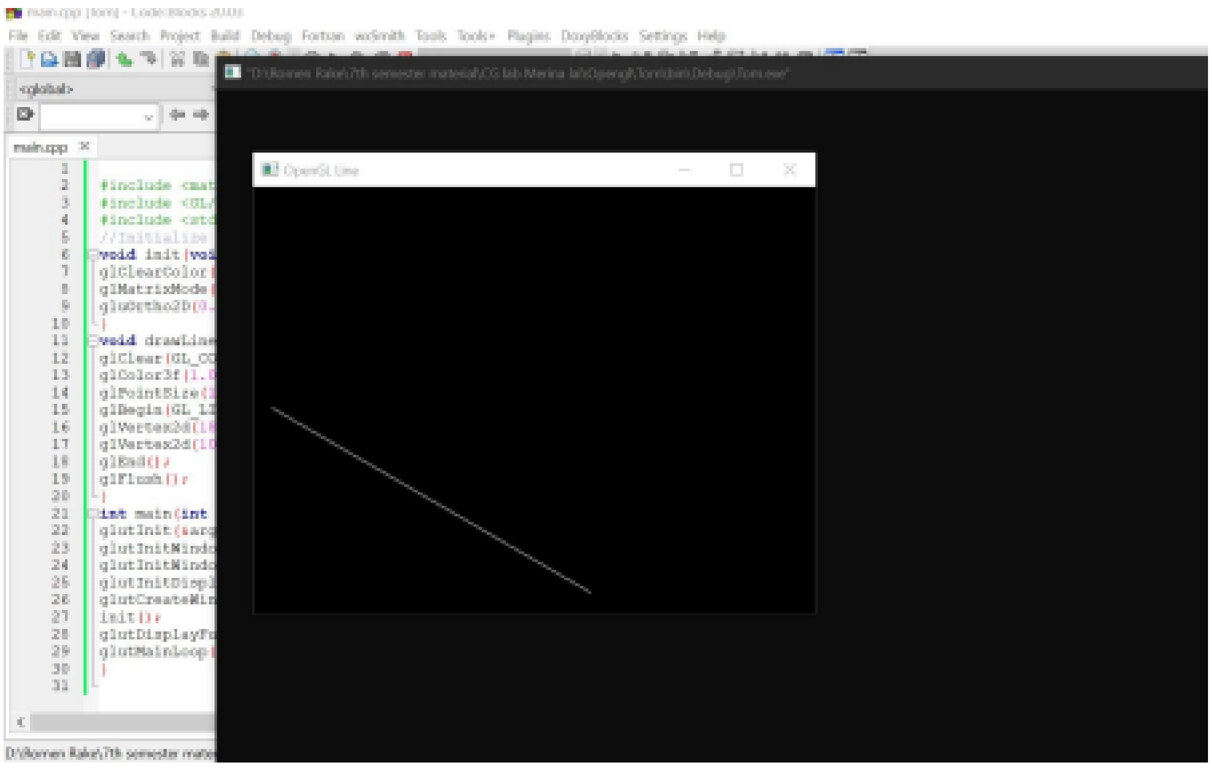
CODE:

```
#include <math.h>
#include <GL/glut.h>
#include <stdlib.h>
```

```
//Initialize OpenGL
void init(void) {
glClearColor(0.0,0.0,0.0,0.0);
glMatrixMode(GL_PROJECTION);
gluOrtho2D(0.0,300.0,0.0,300.0);
}
void drawLines(void) {
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0,1.0,1.0);
glPointSize(1.0);
glBegin(GL_LINES);
glVertex2d(180, 15);
glVertex2d(10, 145);
glEnd();
glFlush();
}
int main(int argc, char**argv) {
```

```
glutInit(&argc, argv);
glutInitWindowPosition(10,10);
glutInitWindowSize(500,500);
glutInitDisplayMode(GLUT_SINGLE |
GLUT_RGB); glutCreateWindow("OpenGL Line");
init();
glutDisplayFunc(drawLines);
glutMainLoop();
}
```

Output:


main.cpp [tom] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
main.cpp
"D:\Romen Raka\7th semester material\CG lab Merina la\OpenGL\Tom\bin\Debug\Tom.exe"
OpenGL Line

```
#include <mat
#include <GL/
#include <std
//Initialize
void init(voi
glClearColor(
glMatrixMode(
gluOrtho2D(0.
}
void drawLine
glClear(GL_CO
glColor3f(1.0
glPointSize(1
glBegin(GL_LI
glVertex2d(10
glVertex2d(10
glEnd();
glFlush();
}
int main(int
glutInit(&arg
glutInitWindo
glutInitWindo
glutInitDispl
glutCreateWin
init();
glutDisplayFu
glutMainLoop(
}
```

Page-5

Q3.Write a C program to draw basic shapes of geometry using OpenGL

AIM: To write a C program to draw basic shapes of geometry using OpenGL

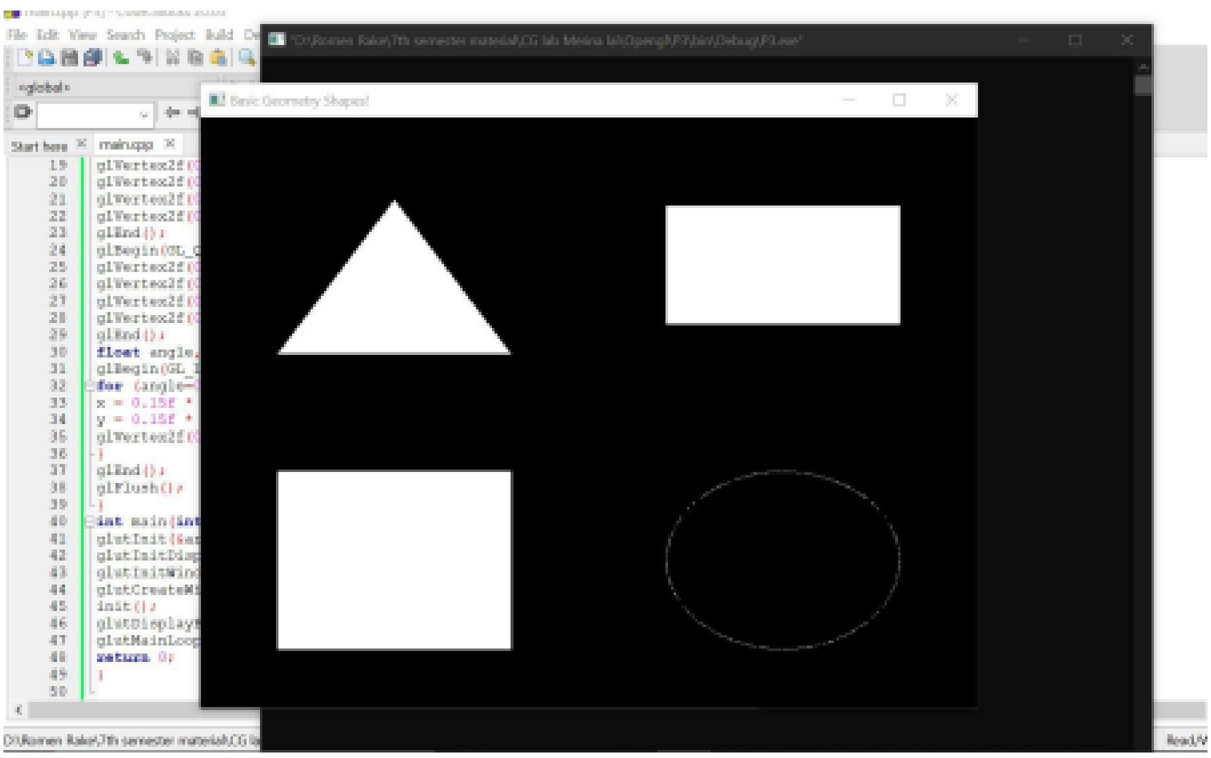
CODE:
```
#include<Gl/glut.h>
#include<stdlib.h>
#include<math.h>
void init(void){
```

```
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:



```
glVertex2f(
glVertex2f(
glVertex2f(
glVertex2f(
glEnd();
glBegin(GL_
glVertex2f(
glVertex2f(
glVertex2f(
glVertex2f(
glEnd();
float angle,
glBegin(GL_
for (angle=
x = 0.15f *
y = 0.15f *
glVertex2f(
}
glEnd();
glFlush();
}
int main(int
glutInit(&a
glutInitDis
glutInitWind
glutCreateWi
init();
glutDisplay
glutMainLoop
return 0;
}
```

Q4.Write a C program to draw a cube using OpenGL.

AIM : To write a C program to draw a cube using OpenGL.

CODE:

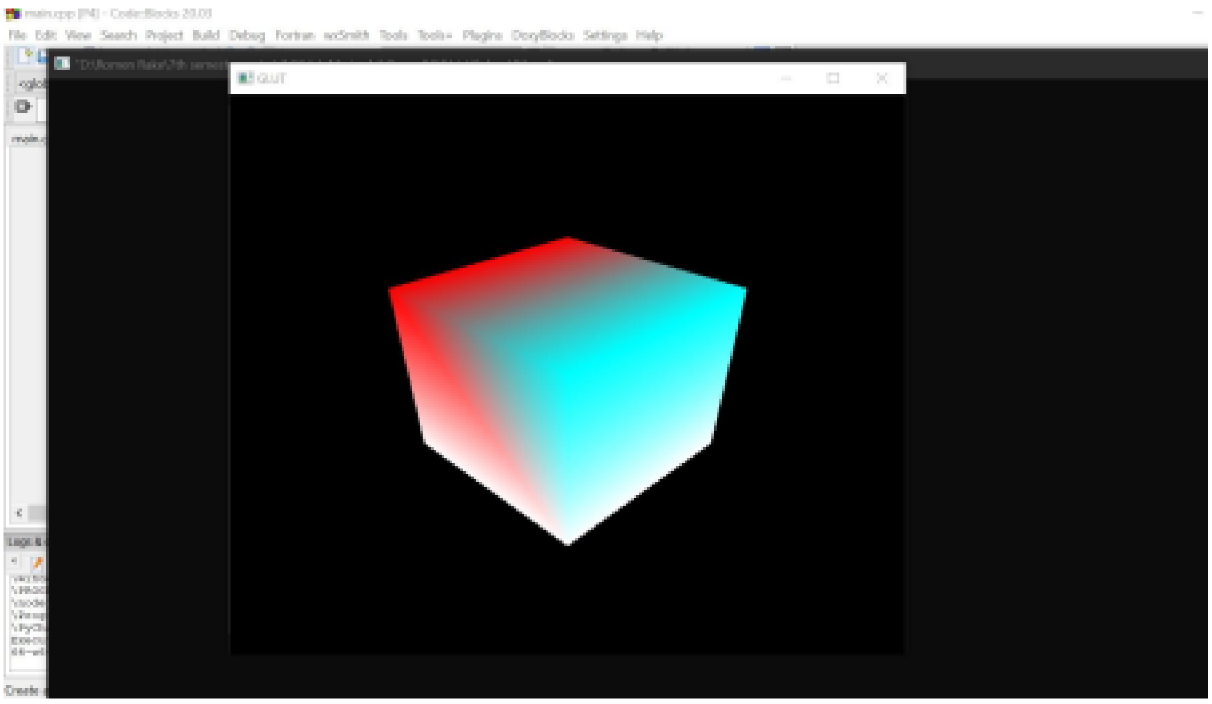
```
#include <GL/glu.h>
#include <GL/glut.h>
#include <GL/gl.h>
#include <math.h>

float radius = 0.05, h = radius;
float g = 9.8, v = 6, u = v;
float max_h = sqrt((v * v) / (2 * g));
float t = 0;

void bounce() {
    t += 0.00025;
    h = u * t - (g * t * t) / 2;
    if(h <= 0) t = 0;
    glutPostRedisplay();
}
```

Page 9

```
void MyInit() {
    glClearColor(1, 1, 1, 0);
    glColor3f(1, 0, 0);
}

void ball() {
    glColor3f(0, 0, 0);
    glBegin(GL_POLYGON);
        glVertex2f(-1, -1);
        glVertex2f(-1, 1);
        glVertex2f(1, 1);
        glVertex2f(1, -1);
    glEnd();

    glColor3f(0, 0, 1);
    float x1 = 0,y1 = h - 1;
    glBegin(GL_TRIANGLE_FAN);
        glVertex2f(x1,y1);
        for (float angle=1.0f;angle<361.0f;angle+=0.2)
        {
            float x2 = x1+cos(angle)*radius;
            float y2 = y1+sin(angle)*radius;
            glVertex2f(x2,y2);
        }
    glEnd();
```

```
}

void display() {
    glLoadIdentity();

    ball();

    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(bounce);
    glutMainLoop();
    return 0;
```
Page-11
```
}
```

Output:



Page-12

Q5.Write a C program to draw a line and show translation, rotation and scaling motion of the line using OpenGL

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

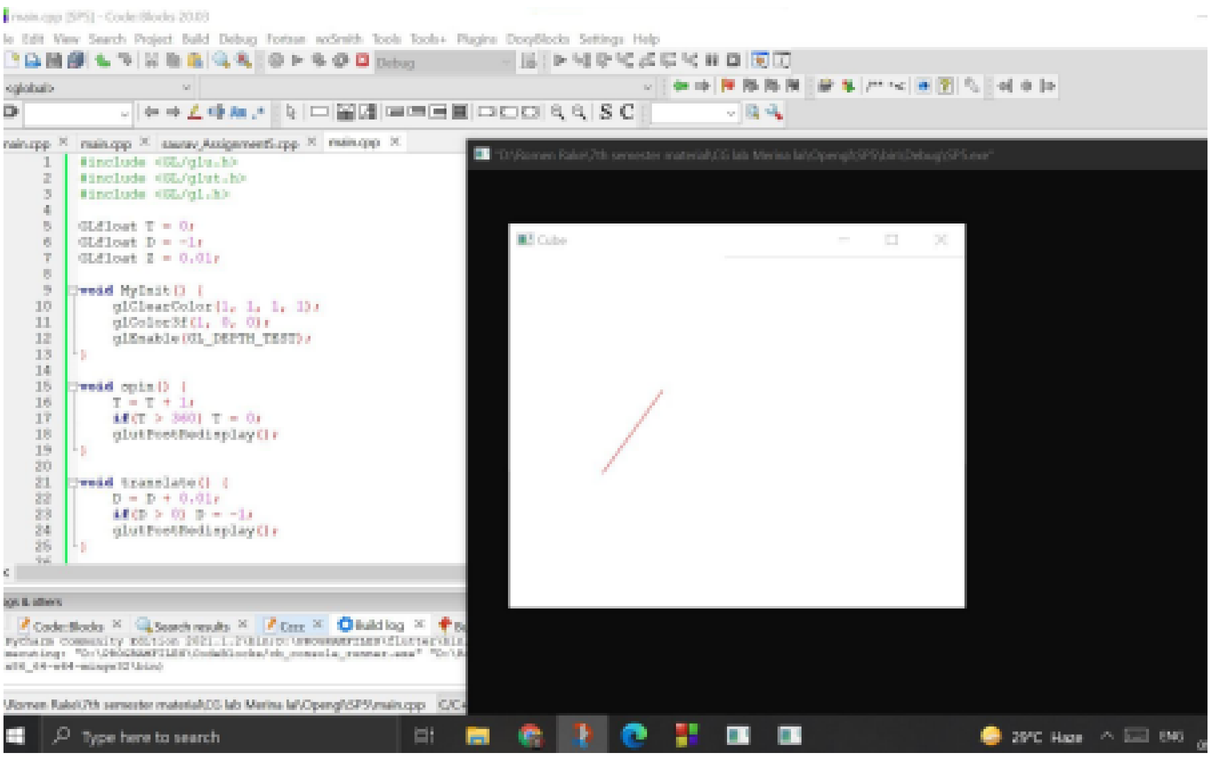
CODE:

```
#include <GL/glu.h>
#include <GL/glut.h>
#include <GL/gl.h>

GLfloat T = 0;
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 + 1;
   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 line() {
    glBegin(GL_LINES);
        glVertex3f(0.5, 0.5, 0.0);
        glVertex3f(-0.5, -0.5, 0.0);
    glEnd();
}

void display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glLoadIdentity();
    glRotatef(T, 0, 1, 0);
    glTranslatef(D, 0, 0);
glScalef(Z, Z, Z);

    line();


    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:



```
#include <GL/glu.h>
#include <GL/glut.h>
#include <GL/gl.h>

GLfloat T = 0;
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 + 1;
    if(T > 360) T = 0;
    glutPostRedisplay();
}

void translate() {
    D = D + 0.01;
    if(D > 0) D = -1;
    glutPostRedisplay();
}
```

Page-16
Experiment no.6

Q6.Write a C program to draw a cube and show translation, rotation and scaling motion of the line using OpenGL.

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

Code:#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);
```

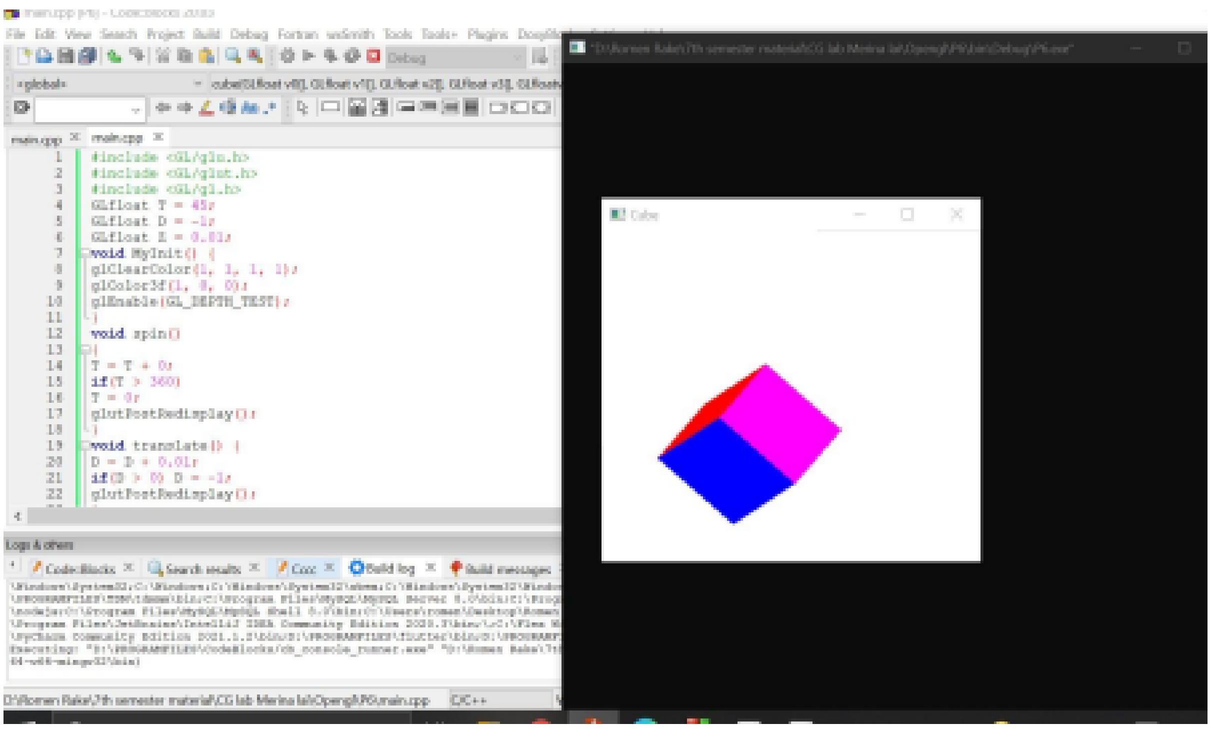
```
face(v4, v5, v6, v7);
glColor3f(0, 0, 1);
face(v0, v3, v7, v4);
glColor3f(0, 1, 1);
face(v1, v2, v6, v5);
glColor3f(1, 0, 1);
face(v0, v1, v5, v4);
glColor3f(1, 1, 0);
face(v3, v2, v6, v7);
}
void display() {
GLfloat v[8][3] = {
{-0.5, 0.5, 0.5},
{0.5, 0.5, 0.5},
{0.5, -0.5, 0.5},
{-0.5, -0.5, 0.5},
{-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;
}

Page-20

Output:



```
#include <GL/glu.h>
#include <GL/glut.h>
#include <GL/gl.h>
GLfloat T = 45;
GLfloat D = -1;
GLfloat I = 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();
```

```
[illegible]
Executing: "D:\PROGRAMFILES\CodeBlocks/cb_console_runner.exe" [illegible]
```

main.cpp [P6] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins
Debug
<global>
main.cpp
Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
D:\Romen Rake\7th semester material\CG lab Merina lab\Opengl\P6\main.cpp
C/C++
Cube

Experiment no.7

Q7.Write a C program to draw a house and show rising and setting of sun in between mountains using OpenGL.

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

```
#include <GL/glu.h>
#include <GL/glut.h>
#include <GL/gl.h>
#include <math.h>
```

Page-21

```
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)
```

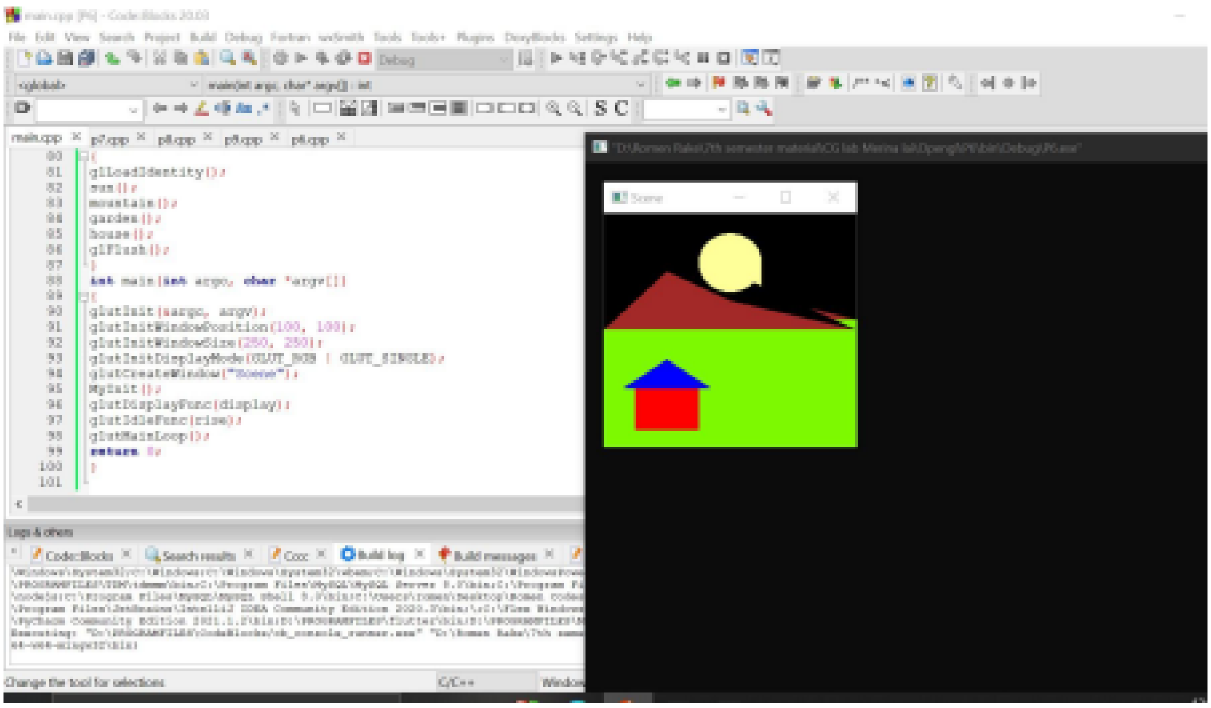
```
; glVertex2f(-1, 0);
glVertex2f(-1, 1);
glVertex2f(-1, 1);
glVertex2f(1, 0);
glEnd();
float x1,y1,x2,y2;
float angle;
```

```
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);
Page-24

glutMainLoop();
return 0;
}

Output:



```
{
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;
}
```

```
[illegible]
```

Experiment no.8

Q8.Write a C program to draw a solar system showing rotation and revolution of sun, moon and earth using OpenGL.

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

Page-25

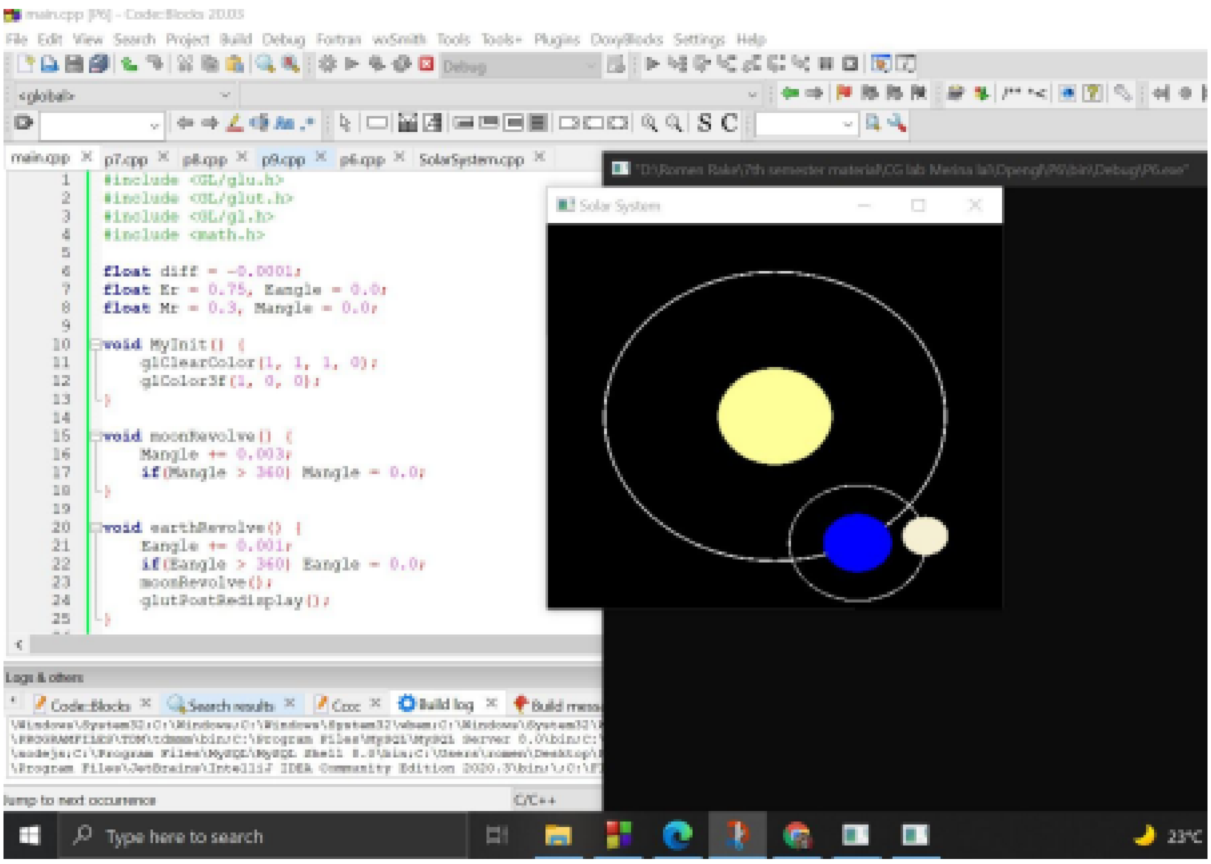
Code :

```
#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;
   Page-26

  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();

}
```
Page-27

```
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;
    Page-28

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 :



```
#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();
}
```

```
\Windows\System32;C:\Windows;C:\Windows\System32\wbem;C:\Windows\System32\
\PROGRAMFILES\TDM\tdmgm\bin;C:\Program Files\MySQL\MySQL Server 8.0\bin;C:\
\nodejs;C:\Program Files\MySQL\MySQL Shell 8.0\bin;C:\Users\romen\Desktop\
\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.3\bin;\;C:\F
```

main.cpp [P6] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
main.cpp p7.cpp p8.cpp p9.cpp p6.cpp SolarSystem.cpp
"D:\Romen Rake\7th semester material\CG lab Meina la\Opengl\P6\bin\Debug\P6.exe"
Solar System
Logs & others
Code::Blocks Search results Cccc Build log Build messages
Jump to next occurrence
C/C++
Type here to search
23°C

Experiment no.9

Q9.Write a C program to draw a ball and show its bouncing motion using OpenGL.

AIM : To write a C program to draw a ball and show its bouncing motion using OpenGL

CODE:

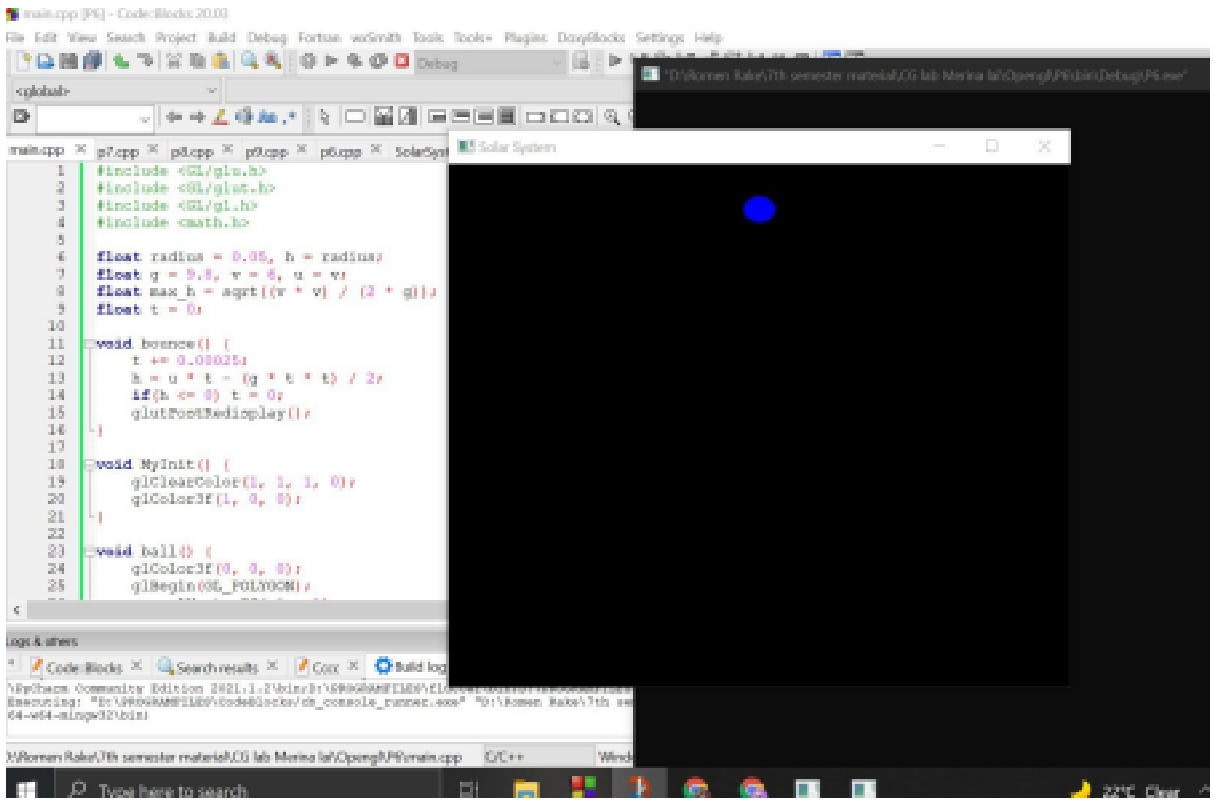
```
#include <GL/glu.h>
#include <GL/glut.h>
#include <GL/gl.h>
#include <math.h>

float radius = 0.05, h =
radius; float g = 9.8, v = 6, u =
v;
float max_h = sqrt((v * v) / (2 *
g)); float t = 0;

void bounce() {
   t += 0.00025;
   h = u * t - (g * t * t) / 2;
   if(h <= 0) t = 0;
   glutPostRedisplay();
}

void MyInit() {
   glClearColor(1, 1, 1, 0);
   glColor3f(1, 0, 0);
}

void ball() {
   glColor3f(0, 0, 0);
   glBegin(GL_POLYGON);
      glVertex2f(-1, -1);
      glVertex2f(-1, 1);
      glVertex2f(1, 1);
      glVertex2f(1, -1);
```

```
    glEnd();
    glColor3f(0, 0, 1);
    float x1 = 0,y1 = h - 1;
    glBegin(GL_TRIANGLE_FAN);
        glVertex2f(x1,y1);
        for (float angle=1.0f;angle<361.0f;angle+=0.2)
        {
            float x2 = x1+cos(angle)*radius;
            float y2 = y1+sin(angle)*radius;
            glVertex2f(x2,y2);
        }
    glEnd();
}

void display() {
    glLoadIdentity();

    ball();

    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();
```

Page-33

```
    glutDisplayFunc(display);
    glutIdleFunc(bounce);
    glutMainLoop();
    return 0;
}
```

Output:



```
#include <GL/glu.h>
#include <GL/glut.h>
#include <GL/gl.h>
#include <math.h>

float radius = 0.05, h = radius;
float g = 9.8, v = 6, u = v;
float max_h = sqrt((v * v) / (2 * g));
float t = 0;

void bounce() {
    t += 0.00025;
    h = u * t - (g * t * t) / 2;
    if(h <= 0) t = 0;
    glutPostRedisplay();
}

void MyInit() {
    glClearColor(1, 1, 1, 0);
    glColor3f(1, 0, 0);
}

void ball() {
    glColor3f(0, 0, 0);
    glBegin(GL_POLYGON);
```

```
\PyCharm Community Edition 2021.1.2\bin;D:\PROGRAMFILES\f
Executing: "D:\PROGRAMFILES\CodeBlocks/cb_console_runner.exe" "D:\Romen Rake\7th se
X4-w64-mingw32\bin)
```

main.cpp [P6] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
<global>
main.cpp
p7.cpp
p8.cpp
p9.cpp
p6.cpp
SolarSyst
"D:\Romen Rake\7th semester material\CG lab Merina la\Opengl\P6\bin\Debug\P6.exe"
Solar System
Logs & others
Code::Blocks
Search results
Cccc
Build log
D:\Romen Rake\7th semester material\CG lab Merina la\Opengl\P6\main.cpp
C/C++
Type here to search
22°C Clear

------------------------X----------------------

-