****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**UNIVERSITY OF PETROLEUM & ENERGY STUDIES**

**College of Engineering Studies**

**Dehradun**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**LAB FILE**

Program : B. Tech. CSE with specialization OSS

Subject Name : Computer Graphics Lab

Session : Jan2017 **-** June2017

Batch : 2014-2018

Faculty : Ajay Poddar

**Submitted By: Submitted To:**

**Name: Anurag Viplaw Mr. Ajay Poddar**

**Roll No.: R100214016**

**SAP ID: 500038046**

# Programs:

**Q1. Write a c++ program to draw a line using GL library functions**

#include<GL/glut.h>

#include<GL/glu.h>

#include<GL/gl.h>

void display(void);

void init(void);

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutCreateWindow("500038046");

glutInitWindowPosition(10,10);

glutInitWindowSize(760,540);

glutCreateWindow("500038046");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

void init(void)

{

glClearColor(1.0,1.0,1.0,1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0,0.0,0.0);

glBegin(GL\_LINES);

glVertex2f(0,0.8);

glVertex2f(-0.4,0.4);

glVertex2f(-0.4,-0.4);

glVertex2f(0,-0.8);

glVertex2f(0.4,-0.4);

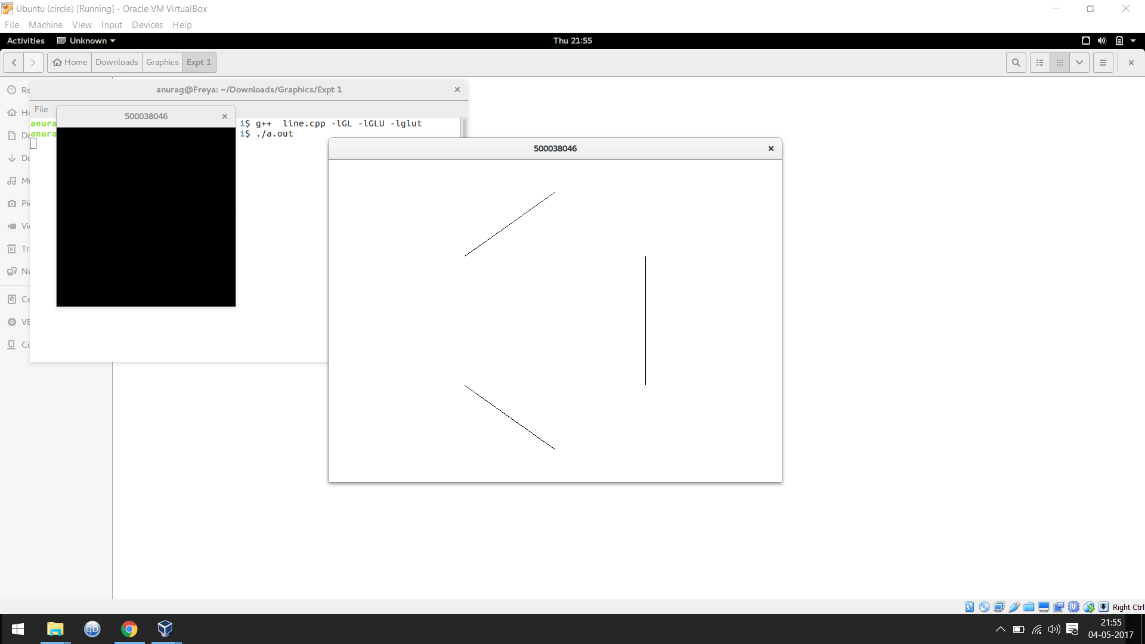
glVertex2f(0.4,0.4);

glEnd();

glFlush();

}

**Output :**

****

**Write a C++ program to draw a triangle using gl library functions**

#include<GL/glut.h>

#include<GL/glu.h>

#include<GL/gl.h>

void display(void);

void init(void);

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutCreateWindow("500038046");

glutInitWindowPosition(10,10);

glutInitWindowSize(760,540);

glutCreateWindow("500038046");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

void init(void)

{

glClearColor(1.0,1.0,1.0,1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0,0.0,0.0);

glBegin(GL\_TRIANGLES);

glVertex2f(0,0.8);

glVertex2f(-0.4,0.4);

glVertex2f(-0.4,-0.4);

glVertex2f(0,-0.8);

glVertex2f(0.4,-0.4);

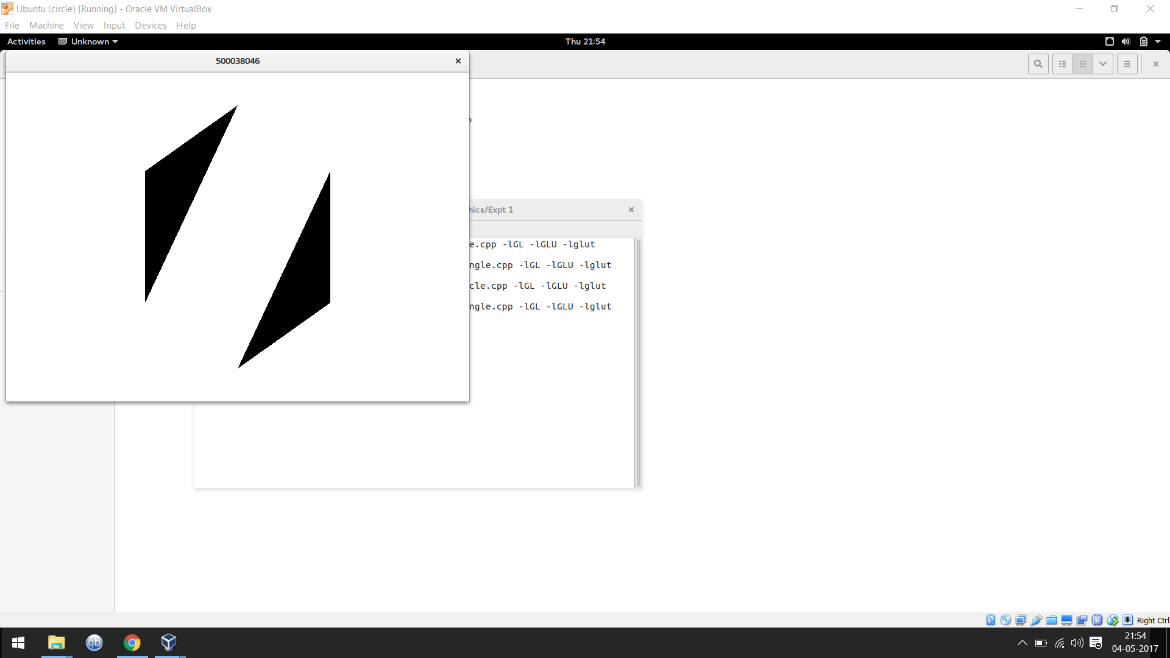
glVertex2f(0.4,0.4);

glEnd();

glFlush();

}

**Output:**



**Write a C++ program to draw a circle using gl library functions**

#include<GL/glut.h>

#include<GL/glu.h>

#include<GL/gl.h>

#include<math.h>

#define RADIUS 0.5

#define CLARITY 200

void display(void);

void init(void);

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowPosition(10,10);

glutInitWindowSize(760,540);

glutCreateWindow("500038046");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

void init(void)

{

glClearColor(1.0,1.0,1.0,1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

}

void display(void)

{

int Q=0;

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0,0.0,0.0);

glBegin(GL\_POINTS);

for(Q=0;Q<360\*CLARITY;Q++)

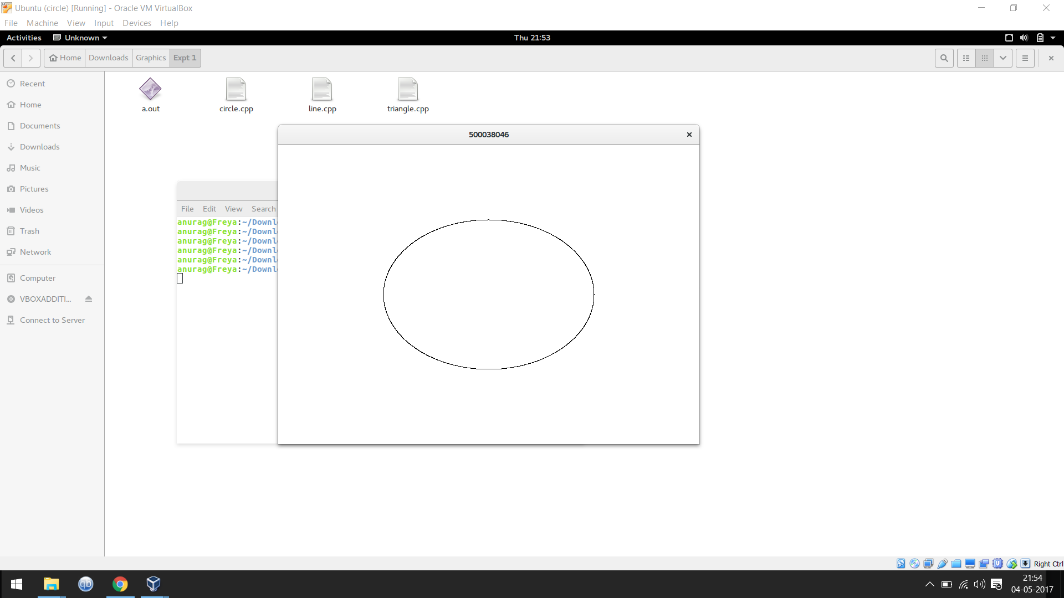
glVertex2f(RADIUS\*cos(Q),RADIUS\*sin(Q));

glEnd();

glFlush();

}

**Output:**



**Q4. Write a C++ program to draw a moving circle using GL library functions**

#include<iostream>

#include<stdlib.h>

#ifdef \_\_APPLE\_\_

#include<openGL/openGL.h>

#include<GLUT/glut.h>

#else

#include<GL/glut.h>

#endif

using namespace std;

float ballX = -0.3f;

float ballY = 0.0f;

float ballZ = -1.0f;

/\*float ballX2 = 0.3f;

float ballY2 = 0.0f; float ballZ2 = -1.0f;

\*/

static int flag=1;

void drawBall(void)

{

glColor3f(0.0, 1.0, 0.0);

glTranslatef(ballX,ballY,ballZ);

glutSolidSphere (0.1, 10, 10);

}

void keyPress(int key, int x, int y)

{

if(key==GLUT\_KEY\_RIGHT)

ballX -= 0.05f;

if(key==GLUT\_KEY\_LEFT)

ballX += 0.05f;

glutPostRedisplay();

}

void initRendering()

{

glEnable(GL\_DEPTH\_TEST);

}

void handleResize(int w, int h) {

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity(); //Reset the camera

gluPerspective(45.0, (double)w / (double)h, 1.0, 200.0);

}

void drawScene()

{

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

drawBall();

glutSwapBuffers();

}

void update(int value) {

if(flag)

{

ballX += 0.001f;

if(ballX>0.3)

( flag=0; }

}

if (!flag)

{

ballX -= 0.001f;

if(ballX<-0.3)

{ flag=1; }

}

glutPostRedisplay();

glutTimerFunc(25, update, 0);

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB|GLUT\_DEPTH);

glutInitWindowSize(400,400);

glutCreateWindow("500038046");

initRendering();

glutDisplayFunc(drawScene);

glutSpecialFunc(keyPress);

glutReshapeFunc(handleResize);

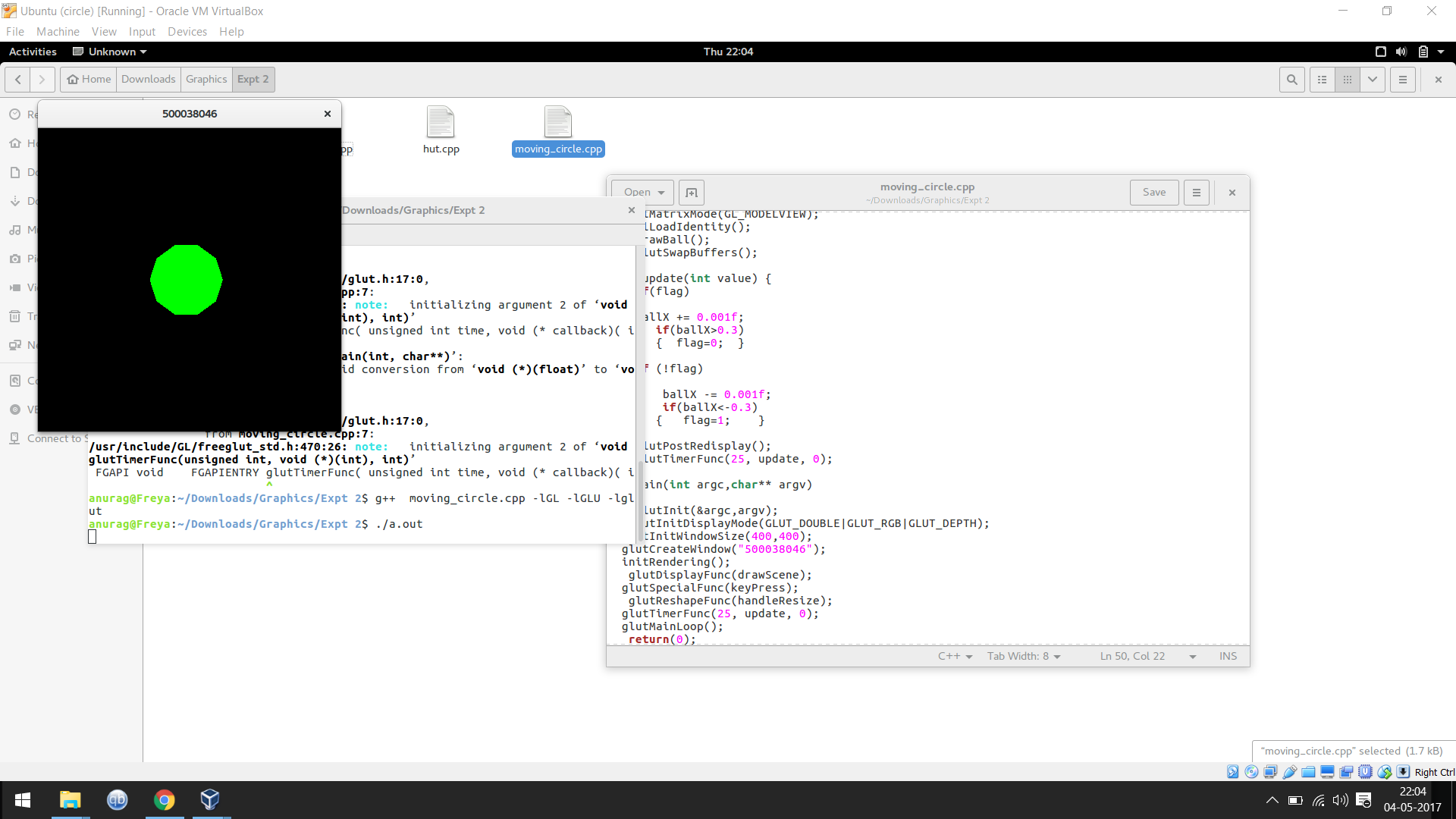
glutTimerFunc(25, update, 0);

glutMainLoop();

return(0);

}

**Output:**



**Q5. Write a c++ program to draw a hut using GL library functions**

#include<GL/glut.h>

#include<GL/glu.h>

#include<GL/gl.h>

#define K 0.015

void display(void);

void init(void);

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutCreateWindow("500038046");

glutInitWindowPosition(10,10);

glutInitWindowSize(1024,768);

glutCreateWindow("500038046");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

void init(void)

{

glClearColor(1.0,1.0,1.0,1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0,0.0,0.0);

glBegin(GL\_LINE\_LOOP);

glVertex3f(0,0,0);

glVertex3f(0.8,0,0);

glVertex3f(0.8,-0.6,0);

glVertex3f(0,-0.6,0);

glEnd();

//GRBox Out

glBegin(GL\_LINE\_LOOP);

glVertex3f(-0.05,0.05,0);

glVertex3f(0.85,0.05,0);

glVertex3f(0.85,-0.6,0);

glVertex3f(0.8,-0.6,0);

glVertex3f(0.8,0,0);

glVertex3f(0,0,0);

glVertex3f(0,-0.6,0);

glVertex3f(-0.05,-0.6,0);

glEnd();

//GRBox 16

for(float i=0;i<0.8;i+=0.2)

{

For(float j=0;j<0.6;j+=0.15)

{

glBegin(GL\_LINE\_LOOP);

glVertex3f(i+K,-j-K,0);

glVertex3f(i+0.2-K,-j-K,0);

glVertex3f(i+0.2-K,-j-0.15+K,0);

glVertex3f(i+K,-j-0.15+K,0);

glEnd();

}

}

//1RBox In

glBegin(GL\_LINE\_LOOP);

glVertex3f(-0.05,0.05,0);

glVertex3f(0.85,0.05,0);

glVertex3f(0.85,0.45,0);

glVertex3f(0.40,0.65,0);

glVertex3f(-0.05,0.45,0);

glEnd();

//1RBox Out

glBegin(GL\_LINE\_LOOP);

glVertex3f(0.85,0.5,0);

glVertex3f(0.85,0.45,0);

glVertex3f(0.40,0.65,0);

glVertex3f(-0.05,0.45,0);

glVertex3f(-0.05,0.5,0);

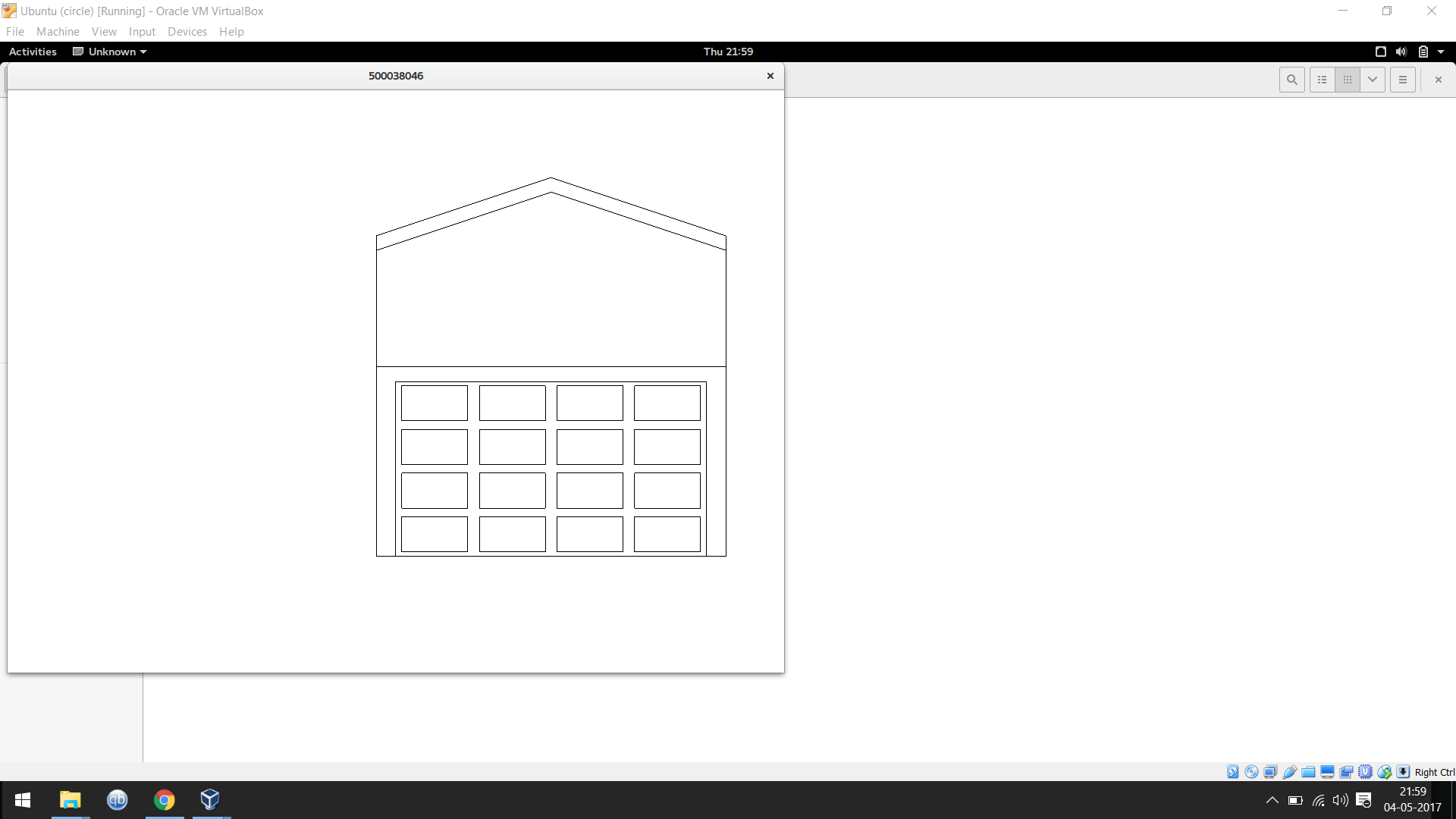
glVertex3f(0.40,0.7,0);

glEnd();

glFlush();

}

**Output:**



**Q6. Write a c++ program to draw a Solid Sphere using GL library functions**

#include <GL/glut.h>

#include <GL/glu.h>

#include <GL/gl.h>

GLfloat xRotated, yRotated, zRotated;

GLdouble radius=1;

void display(void);

void reshape(int x, int y);

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

{

glutInit(&argc, argv);

glutInitWindowSize(350,350);

glutCreateWindow("500038046");

xRotated = yRotated = zRotated = 30.0;

xRotated=43;

yRotated=50;

glutDisplayFunc(display);

glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}

void display(void)

{

glMatrixMode(GL\_MODELVIEW);

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();

glTranslatef(0.0,0.0,-5.0);

glColor3f(0.9, 0.3, 0.2);

glRotatef(xRotated,1.0,0.0,0.0);

glRotatef(yRotated,0.0,1.0,0.0);

glRotatef(zRotated,0.0,0.0,1.0);

glScalef(1.0,1.0,1.0);

glutSolidSphere(radius,20,20);

glFlush();

}

void reshape(int x, int y)

{

if (y == 0 || x == 0) return;

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

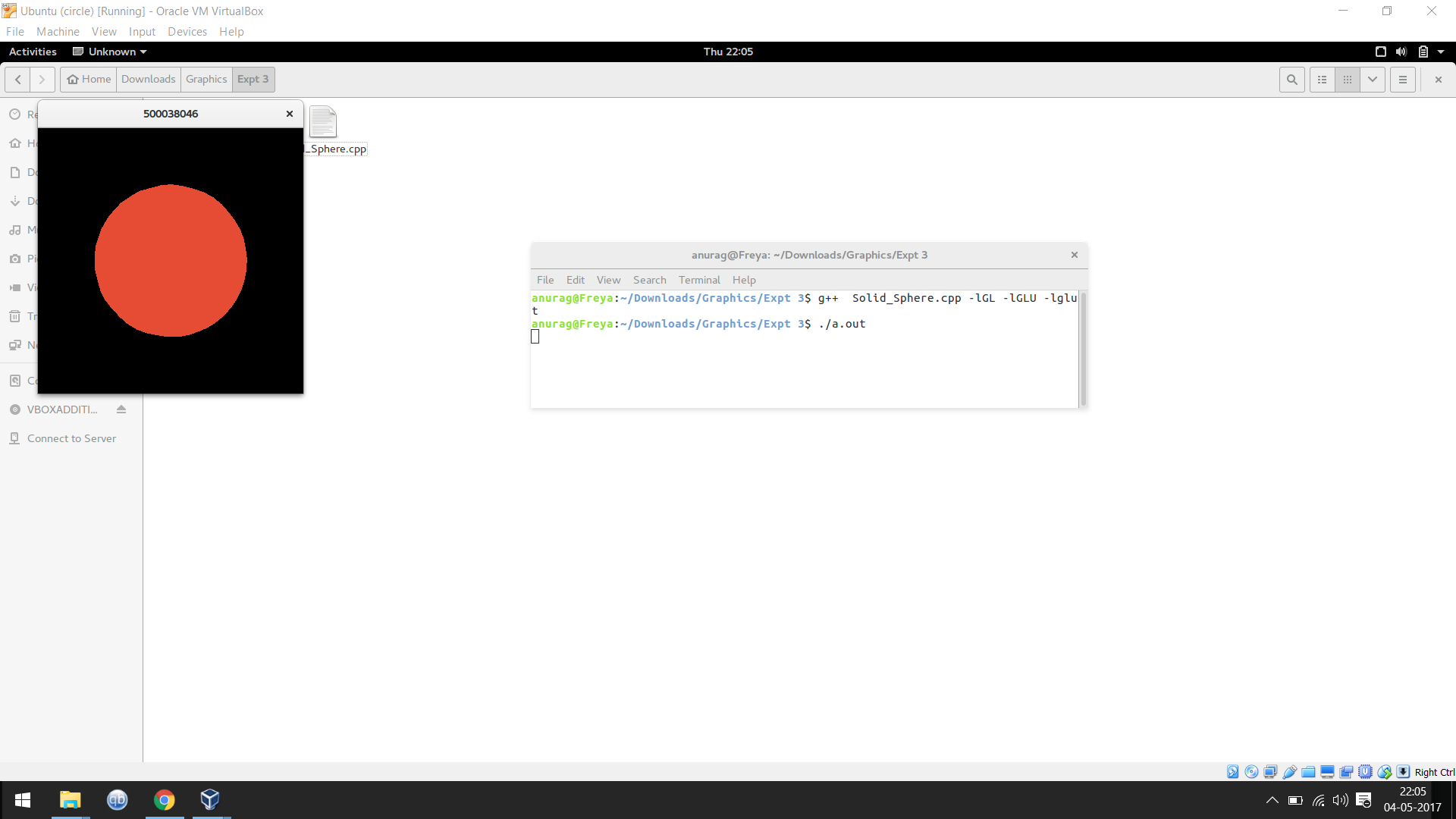
gluPerspective(39.0,(GLdouble)x/(GLdouble)y,0.6,21.0);

glMatrixMode(GL\_MODELVIEW);

glViewport(0,0,x,y);

}

**Output:**



**Q7. Write a c++ program to draw a Ellipse using GL library functions**

#include <GL/glut.h>

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

int h,k;

float a,b;

void display(void)

{

double I,J;

int i,j;

glClear (GL\_COLOR\_BUFFER\_BIT);

glColor3f (1.0, 0.0, 0.0);

glBegin(GL\_POINTS);

glVertex2s(h,k);

for( i=0 ; i<=a ; i+=1)

{

J=sqrt(1 - (i\*i)/(a\*a))\*b;

j=(int)(J);

glVertex2s(h+i,k+j);

glVertex2s(h-i,k+j);

glVertex2s(h-i,k-j);

glVertex2s(h+i,k-j);

}

glColor3f (1.0, 1.0, 1.0);

for(int i=-100 ; i<=100 ; i++)

{

glVertex2s(i,0);

glVertex2s(0,i);

}

for(int i=-2; i<=2 ; i++)

{

glVertex2s(95+i,4+i);

glVertex2s(95-i,4+i);

}

for(int i=0; i<=2 ; i++)

{

glVertex2s(4+i,95+i);

glVertex2s(4-i,95+i);

glVertex2s(4,95-i);

}

glEnd();

glFlush();

}

void init(void)

{

glClearColor (0.0, 0.0, 0.0, 0.0);

glOrtho(-100.0, 100.0, -100.0, 100.0, -1.0, 1.0);

}

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

{

printf("Enter the center of ellipse:\n");

scanf("%d %d",&h,&k);

printf("Enter the parameters a & b:\n");

scanf("%f %f",&a,&b);

glutInit(&argc, argv);

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize (500, 500);

glutInitWindowPosition (100, 100);

glutCreateWindow ("500038046 ");

init ();

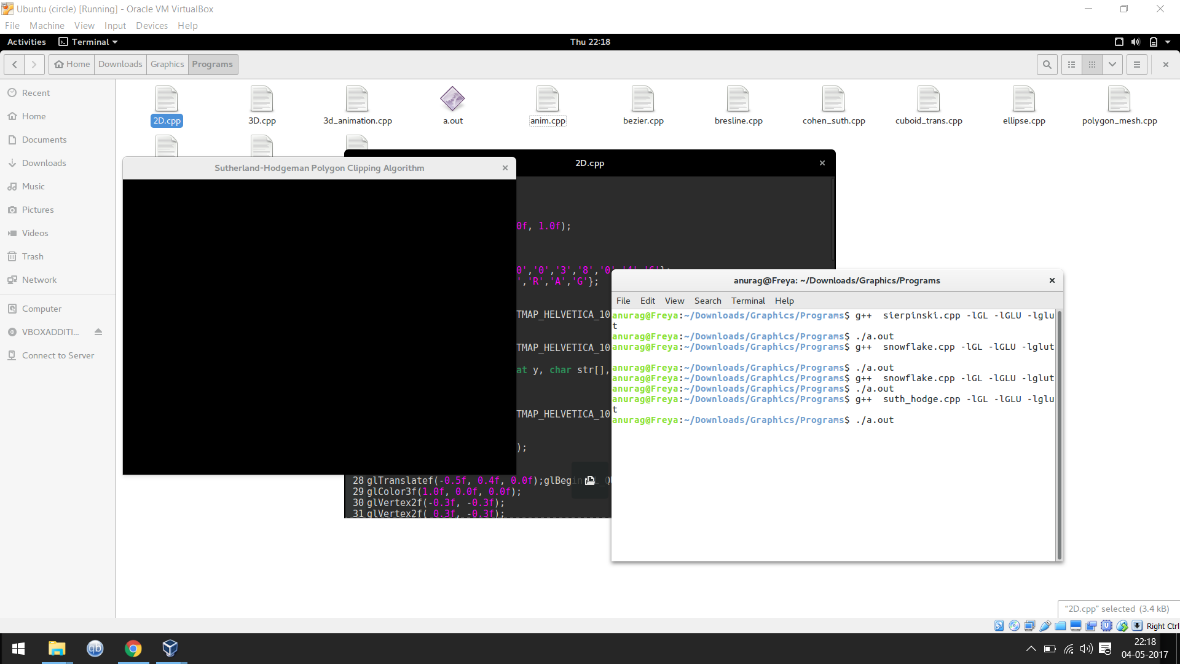
glutDisplayFunc(display);

glutMainLoop();

return 0;}

return 0;}

**Output:**



**Q8. Write a c++ program for line clipping using GL library functions**

#include <GL/glut.h>

#include <GL/glu.h>

#include <GL/gl.h>

#include<math.h>

#include<stdio.h>

#define outcode int

void myinit(void);

double xmin=50,ymin=50,xmax=100,ymax=100; // Windows boundaries

double xvmin=200,yvmin =200, xvmax=300,yvmax=300; // Viewport boundaries

const int RIGHT= 8; //bit codes for the right

const int LEFT =2; //bit codes for the left

const int TOP=4; //bit codes for the top

const int BOTTOM=1; //bit codes for the bottom

outcode ComputeOutCode(double x,double y); // used to compute bit codes of a point

// Cohen -Sutherland clipping algorithm clips a line from

// p0=(x0,y0) to p1 =(x1,y1) against a rectangle with.

// diagonal from (xmin,ymin)to (xmax,ymax)

void CohenSutherlandLineClipAnddraw(double x0,double y0,double x1,double y1)

{

// OutCodes for P0 ,P1 and Whatever point(among P0 & P1) lies outside the

// clip rectangle

outcode outcode0,outcode1,outcodeOut;

int accept =0,done =0;// These are two bits to indicate trivial accept and/or

// done with clipping

//compute outcodes

outcode0= ComputeOutCode(x0,y0);

outcode1= ComputeOutCode(x1,y1);

do

{

if(!(outcode0|outcode1)) // logical or is 0 trivially accept and exit

{ accept=1;

done=1;

}

else

if(outcode0 & outcode1) // logical and is 0 trivially reject and exit

done=1;

else

{

double x,y;

// at least one endpoint is outside the clip rectangle ; pick it.

outcodeOut= outcode0?outcode0:outcode1;

if(outcodeOut & TOP) //point is above the clip rectangle

{

x= x0+(x1-x0)\*(ymax-y0)/(y1-y0);

y=ymax;

}

else

if(outcodeOut & BOTTOM) //point is below the clip rectangle

{

x= x0+(x1-x0)\*(ymin-y0)/(y1-y0);

y=ymin;

}

else

if(outcodeOut & RIGHT) //point is to the right of clip rectangle

{

y= y0+(y1-y0)\*(xmax-x0)/(x1-x0);

x=xmax;

}

else //point is to the left of the clip rectangle

{

y= y0+(y1-y0)\*(xmin-x0)/(x1-x0);

x=xmin;

}

// now we move outside point to intersection point to clip

// and get ready for next pass.

if(outcodeOut == outcode0) // If the outside point was p0 update x0,y0 to x,y

{ x0=x; // so x,y become the new x0,y0

y0=y;

outcode0 = ComputeOutCode(x0,y0);

//compute outcode of new endpoint

}

else // If the outside point was p1 update x1,y1 to x,y

{ // so x,y becomes the new x1,y1

x1=x;

y1=y;

outcode1 = ComputeOutCode(x1,y1);

// compute outcode of new endpoint

}

}

}while(!done);

if(accept) //If line was trivial reject no need to draw viewport

{

// window to viewport mapping

double sx=(xvmax-xvmin)/(xmax-xmin);// scale parameter in x direction

double sy=(yvmax-yvmin)/(ymax-ymin);// scale parameter in y direction

double vx0 = xvmin+(x0-xmin)\*sx;

double vy0 = yvmin+(y0-ymin)\*sy;

double vx1 = xvmin+(x1-xmin)\*sx;

double vy1 = yvmin+(y1-ymin)\*sy;

//draw a red color viewport

glColor3f(1.0,0.0,0.0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(xvmin,yvmin);

glVertex2f(xvmax,yvmin);

glVertex2f(xvmax,yvmax);

glVertex2f(xvmin,yvmax);

glEnd();

glColor3f(0.0,0.0,1.0);

glBegin(GL\_LINES);

glVertex2d(vx0,vy0);

glVertex2d(vx1,vy1);

glEnd();

}

}

// compute the bit code for a point (x,y) using the clip rectangle

// bounded diagonally by (xmin,ymin) and (xmax,ymax)

outcode ComputeOutCode(double x,double y)

{

outcode code =0;

if(y>ymax) //above the clip window

code |=TOP;

if(y<ymin) //below the clip window

code |=BOTTOM;

if(x>xmax) //to the right of the clip window

code |=RIGHT;

if(x<xmin) //to the left of the clip window

code |=LEFT;

return code;

}

void display()

{

double x0,y0,x1,y1,n;

int i;

scanf("%lf",&n);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0,0.0,1.0); // draw a blue colored window

glBegin(GL\_LINE\_LOOP);

glVertex2f(xmin,ymin);

glVertex2f(xmax,ymin);

glVertex2f(xmax,ymax);

glVertex2f(xmin,ymax);

glEnd();

for(i=0;i<n;i++)

{

scanf("%lf %lf %lf %lf",&x0,&y0,&x1,&y1);

glColor3f(1.0,0.0,0.0); // draw red color lines

glBegin(GL\_LINES);

glVertex2d(x0,y0);

glVertex2d(x1,y1);

//glVertex2d(60,20);

//glVertex2d(80,120);

glEnd();

CohenSutherlandLineClipAnddraw(x0,y0,x1,y1);

//CohenSutherlandLineClipAnddraw(60,20,80,120);

printf("line entered, next coordinates \n");

}

glColor3f(1.0,1.0,0.3);

glFlush();

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("500038046");

printf("Enter no of lines you want and the coordinates \n");

glutDisplayFunc(display);

myinit();

glutMainLoop();

return 0;

}

void myinit(void)

{

glClearColor(1.0,1.0,1.0,1.0);

glColor3f(1.0,0.0,0.0);

glPointSize(1.0);

glMatrixMode(GL\_PROJECTION);

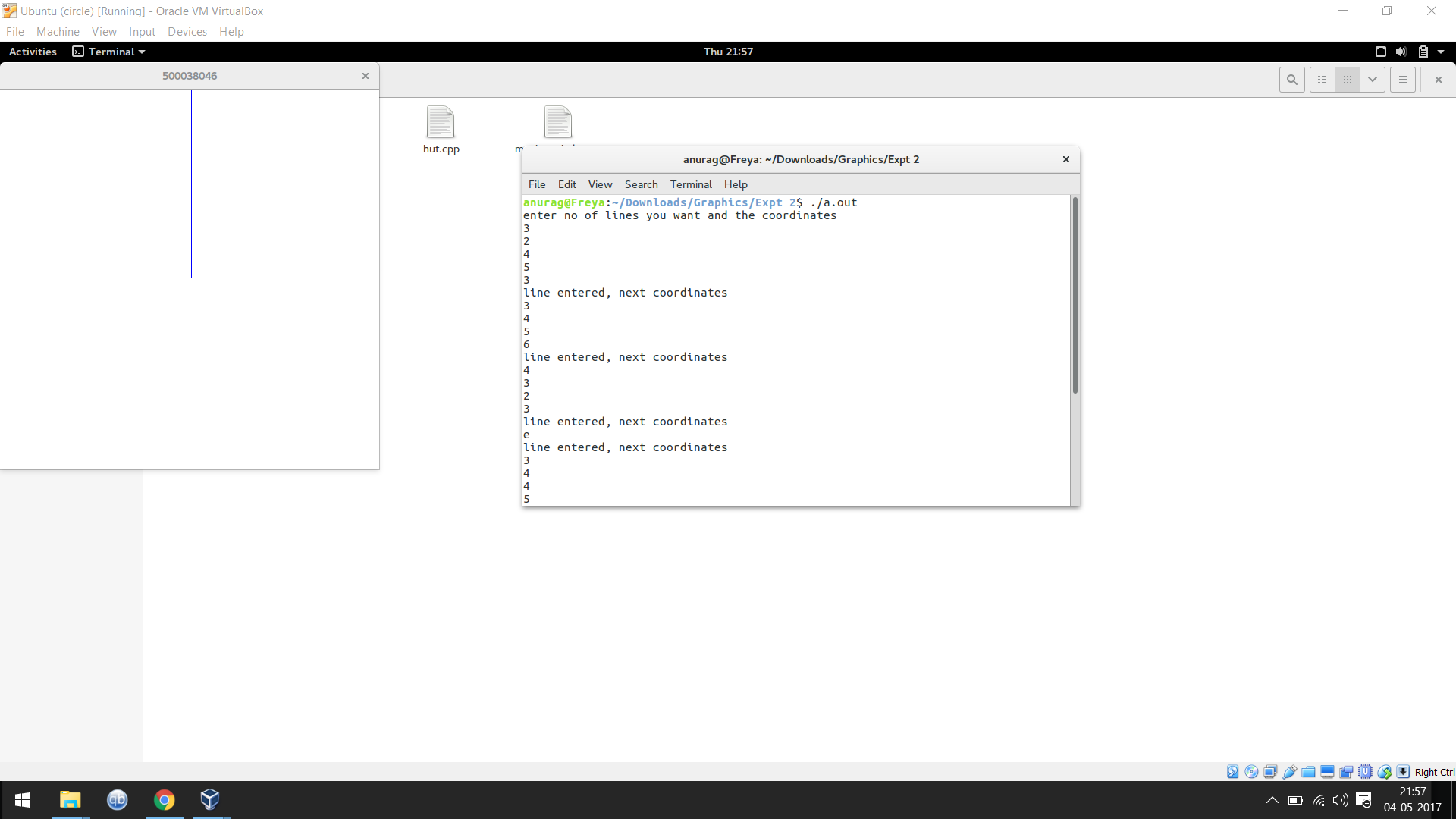
glLoadIdentity();

gluOrtho2D(0.0,99.0,0.0,99.0);

return;

}

**Output:**



**Q9. Write a c++ program for 2D Transformation using GL library functions**

#include <GL/glut.h>

#include <GL/glu.h>

#include <GL/gl.h>

void initGL() {

glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

}

void printhead()

{

char sap[10] = {'5','0','0','0','3','8','0','4','6'};

char name[] = {'A','N','U','R','A','G'};

glRasterPos2f(0.0,0.4);

for (int i=0;i<10;i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10,sap[i]);

glRasterPos2f(0.0,0.5);

for (int i=0;i<8;i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10,name[i]);

}

void printname(float x, float y, char str[], int s)

{

glRasterPos2f(x,y);

for (int i=0;i<s;i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10,str[i]);

}

void display() {

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(-0.5f, 0.4f, 0.0f);glBegin(GL\_QUADS);

glColor3f(1.0f, 0.0f, 0.0f);

glVertex2f(-0.3f, -0.3f);

glVertex2f( 0.3f, -0.3f);

glVertex2f( 0.3f, 0.3f);

glVertex2f(-0.3f, 0.3f);

glEnd();

glTranslatef(0.1f, -0.7f, 0.0f); // Translate right and down

glBegin(GL\_QUADS);

// Each set of 4 vertices form a quad

glColor3f(0.0f, 1.0f, 0.0f); // Green

glVertex2f(-0.3f, -0.3f);

glVertex2f( 0.3f, -0.3f);

glVertex2f( 0.3f, 0.3f);

glVertex2f(-0.3f, 0.3f);

glEnd();

glTranslatef(-0.3f, -0.2f, 0.0f); // Translate left and down

glBegin(GL\_QUADS);

// Each set of 4 vertices form a quad

glColor3f(0.2f, 0.2f, 0.2f); // Dark Gray

glVertex2f(-0.2f, -0.2f);

glColor3f(1.0f, 1.0f, 1.0f); // White

glVertex2f( 0.2f, -0.2f);

glColor3f(0.2f, 0.2f, 0.2f); // Dark Gray

glVertex2f( 0.2f,

0.2f);

glColor3f(1.0f, 1.0f, 1.0f); // White

glVertex2f(-0.2f,

0.2f);

glEnd();

glTranslatef(1.1f, 0.2f, 0.0f); // Translate right and up

glBegin(GL\_TRIANGLES);

// Each set of 3 vertices form a triangle

glColor3f(0.0f, 0.0f, 1.0f); // Blue

glVertex2f(-0.3f, -0.2f);

glVertex2f( 0.3f, -0.2f);

glVertex2f( 0.0f,

0.3f);glEnd();

glTranslatef(0.2f, -0.3f, 0.0f);

// Translate right and down

glRotatef(180.0f, 0.0f, 0.0f, 1.0f); // Rotate 180 degree

glBegin(GL\_TRIANGLES);

// Each set of 3 vertices form a triangle

glColor3f(1.0f, 0.0f, 0.0f); // Red

glVertex2f(-0.3f, -0.2f);

glColor3f(0.0f, 1.0f, 0.0f); // Green

glVertex2f( 0.3f, -0.2f);

glColor3f(0.0f, 0.0f, 1.0f); // Blue

glVertex2f( 0.0f,

0.3f);

glEnd();

glRotatef(-180.0f, 0.0f, 0.0f, 1.0f); // Undo previous rotate

glTranslatef(-0.1f, 1.0f, 0.0f); // Translate right and down

glBegin(GL\_POLYGON); // The vertices form one closed polygon

glColor3f(1.0f, 1.0f, 0.0f); // Yellow

glVertex2f(-0.1f, -0.2f);

glVertex2f( 0.1f, -0.2f);

glVertex2f( 0.2f, 0.0f);

glVertex2f( 0.1f, 0.2f);

glVertex2f(-0.1f, 0.2f);

glVertex2f(-0.2f, 0.0f);

glEnd();

glColor3f(1.0,0.0,0.0);

char name[15]={'2','-','D',' ','T','R','A','N','S','F','O','R','M'};

printname(0.3,0.1,name,15);

glBegin(GL\_POLYGON);

glVertex2f(-0.3,0.3);

glVertex2f(0.3,0.3);

glVertex2f(0.3,0.7);

glVertex2f(-0.3,0.7);

glEnd();

glColor3f(1.0,1.0,1.0);

printhead();glFlush();

}

void reshape(GLsizei width, GLsizei height) {

if (height == 0) height = 1;

GLfloat aspect = (GLfloat)width / (GLfloat)height;

glViewport(0, 0, width, height);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if (width >= height) {

gluOrtho2D(-1.0 \* aspect, 1.0 \* aspect, -1.0, 1.0);

} else {

gluOrtho2D(-1.0, 1.0, -1.0 / aspect, 1.0 / aspect);

}

}

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

glutInit(&argc, argv);

glutInitWindowSize(640, 480);

glutInitWindowPosition(50, 50);

glutCreateWindow("2D Transformation");

glutDisplayFunc(display);

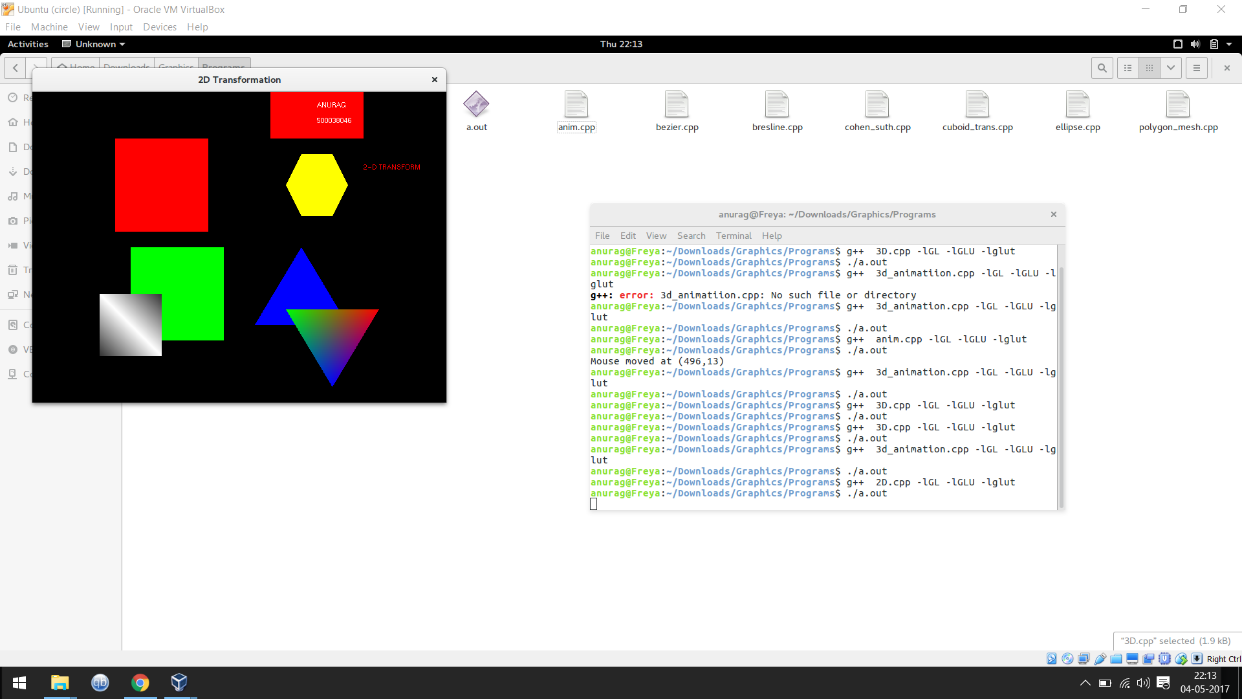
glutReshapeFunc(reshape);

initGL();

glutMainLoop();

return 0;}

**Output:**



**Q10. Write a c++ program for 3D Transformation using GL library functions**

#include <GL/glut.h>

GLfloat xRotated, yRotated, zRotated;

void printhead()

{

char sap[10] = {'5','0','0','0','3','8','0','4','6'};

char name[8] = {'A','N','U','R','A','G'};

glRasterPos2f(-0.2,0.1);

for (int i=0;i<10;i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10,sap[i]);

glRasterPos2f(-0.2,0.2);

for (int i=0;i<8;i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10,name[i]);

}

void printname(float x, float y, char str[], int s)

{

glRasterPos2f(x,y);

for (int i=0;i<s;i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10,str[i]);

}

void redisplayFunc(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();

glTranslatef(0.0,0.0,-5.0);

glColor3f(0.9, 0.0, 0.0);

glRotatef(xRotated,1.0,0.0,0.0);

glRotatef(yRotated,0.0,1.0,0.0);

glRotatef(zRotated,0.0,0.0,1.0);

glScalef(1.0,1.0,1.0);

glutWireCube(1.0);

glColor3f(1.0,0.0,0.0);

char name[15]={'3','-','D',' ','T','R','A','N','S','F','O','R','M'};

printname(0.5,0.8,name,15);glBegin(GL\_POLYGON);

glVertex2f(-0.3,0);

glVertex2f(0.4,0);

glVertex2f(0.4,0.3);

glVertex2f(-0.3,0.3);

glEnd();

glColor3f(1.0,1.0,1.0);

printhead();

glFlush();

glutSwapBuffers();

}

void reshapeFunc(int x, int y)

{

if (y == 0 || x == 0) return;

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(40.0,(GLdouble)x/(GLdouble)y,0.5,20.0);

glMatrixMode(GL\_MODELVIEW);

glViewport(0,0,x,y);

}

void idleFunc(void)

{

// rotation around x axis

xRotated += 0.03;

redisplayFunc();

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(350,350);

// create the window

glutCreateWindow("3D Transformation By Vinayak Sarawagi");

glPolygonMode(GL\_FRONT\_AND\_BACK,GL\_LINE);

xRotated = yRotated = zRotated = 0.0;

glClearColor(0.0,0.0,0.0,0.0);//Assign

//the function used in events

glutDisplayFunc(redisplayFunc);

glutReshapeFunc(reshapeFunc);

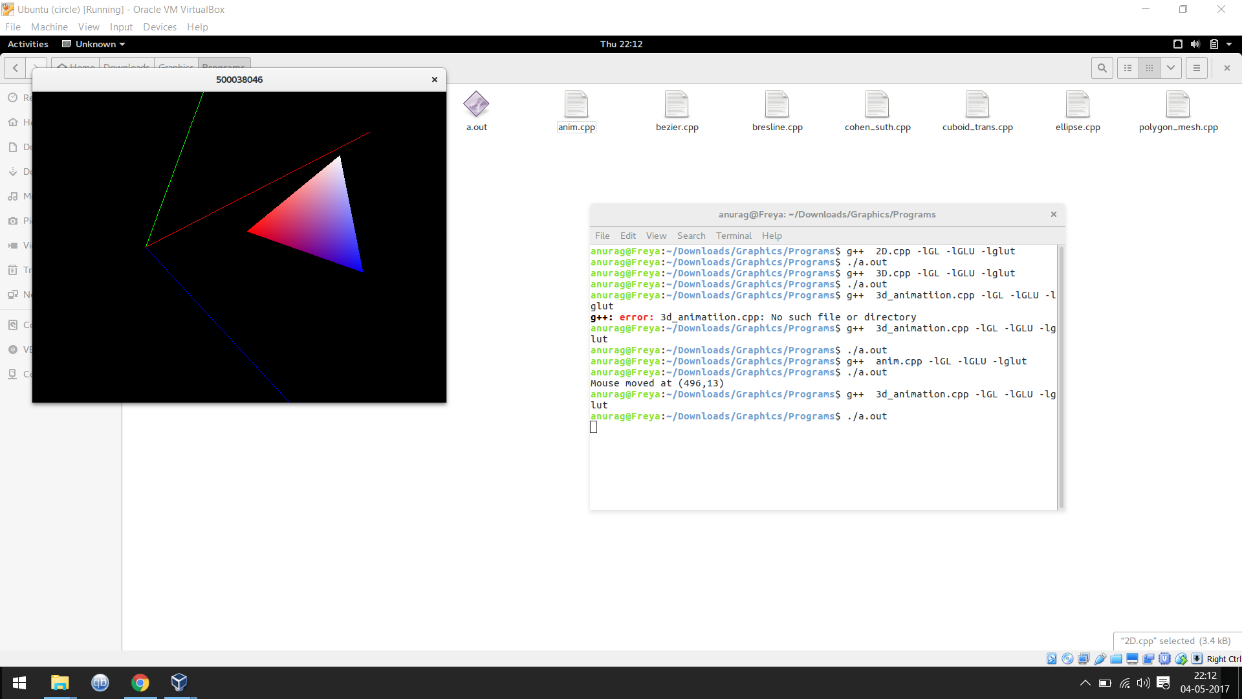
glutIdleFunc(idleFunc);

glutMainLoop();

return 0;

}

**Output:**



**Q11. Write a c++ program to draw Biezer Curve using GL library functions**

#include <GL/gl.h>

#include <GL/glu.h>

#include <stdlib.h>

#include <GL/glut.h>

GLfloat ctrlpoints[4][3] = {

{ -4.0, -4.0, 0.0}, { -2.0, 4.0, 0.0},

{2.0, -4.0, 0.0}, {4.0, 4.0, 0.0}};

void init(void)

{

glClearColor(0.0, 0.0, 0.0, 0.0);

glShadeModel(GL\_FLAT);

glMap1f(GL\_MAP1\_VERTEX\_3, 0.0, 1.0, 3, 4, &ctrlpoints[0][0]);

glEnable(GL\_MAP1\_VERTEX\_3);

}

void display(void)

{

int i;

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 1.0, 1.0);

glBegin(GL\_LINE\_STRIP);

for (i = 0; i <= 30; i++)

glEvalCoord1f((GLfloat) i/30.0);

glEnd();

/\* The following code displays the control points as dots. \*/

glPointSize(5.0);

glColor3f(1.0, 1.0, 0.0);

glBegin(GL\_POINTS);

for (i = 0; i < 4; i++)

glVertex3fv(&ctrlpoints[i][0]);

glEnd();

glFlush();

}

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei) w, (GLsizei) h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if (w <= h)

glOrtho(-5.0, 5.0, -5.0\*(GLfloat)h/(GLfloat)w,

5.0\*(GLfloat)h/(GLfloat)w, -5.0, 5.0);

else

glOrtho(-5.0\*(GLfloat)w/(GLfloat)h,

5.0\*(GLfloat)w/(GLfloat)h, -5.0, 5.0, -5.0, 5.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowSize (500, 500);

glutInitWindowPosition (100, 100);

glutCreateWindow (argv[0]);

init ();

glutDisplayFunc(display);

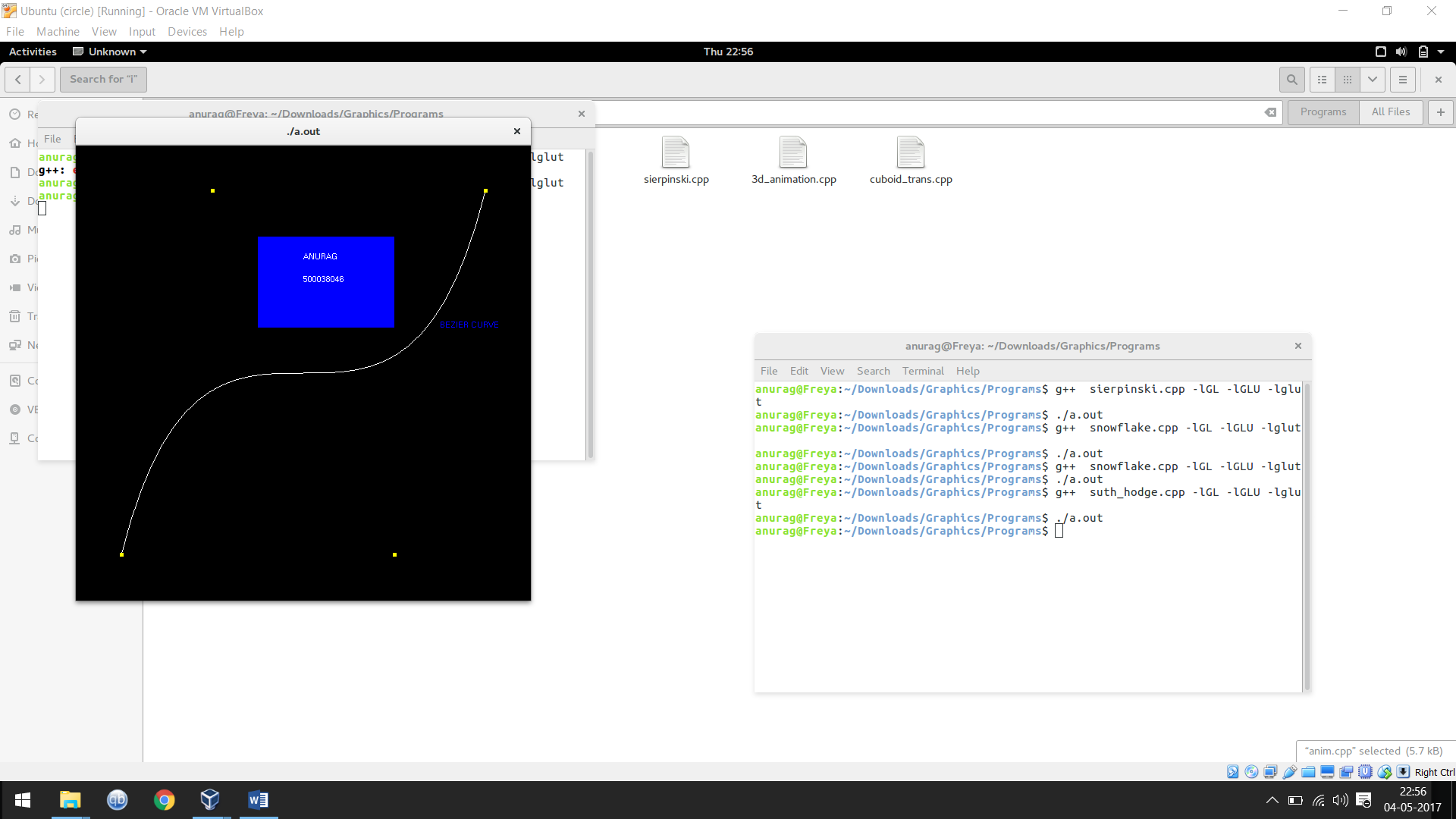
glutReshapeFunc(reshape);

glutMainLoop();

return 0;

}

**Output:**



**Q12. Write a c++ program to show Fractals using GL library functions**

# include <stdlib>

# include <GL/glut.h>

# include <GL/glut.h>

# include <GL/freeglut.h>

using namespace std;

int main ( int argc, char \*argv[] );

void display ( );

void myinit ( );

typedef GLfloat point2[2];

int main ( int argc, char \*argv[] )

{

glutInit ( &argc, argv );

glutInitDisplayMode ( GLUT\_SINGLE | GLUT\_RGB );

glutInitWindowSize ( 400, 600 );

glutInitWindowPosition ( 0, 0 );

glutCreateWindow ( "500038046");

glutDisplayFunc ( display );

myinit ( );

glutMainLoop ( );

return 0;

}

void display ( )

{

int i;

point2 p;

int point\_num = 500000;

double prob[4] = { 0.85, 0.92, 0.99, 1.00 };

double r;

double x;

double y;

glClear ( GL\_COLOR\_BUFFER\_BIT );

p[0] = drand48 ( );

p[1] = drand48 ( );

for ( i = 0; i < point\_num; i++ )

{

r = drand48 ( );

if ( r < prob[0] )

{

x = 0.85 \* p[0] + 0.04 \* p[1] + 0.0;

y = - 0.04 \* p[0] + 0.85 \* p[1] + 1.6;

}

else if ( r < prob[1] )

{

x = 0.20 \* p[0] - 0.26 \* p[1] + 0.0;

y = 0.23 \* p[0] + 0.22 \* p[1] + 1.6;

}

else if ( r < prob[2] )

{

x = - 0.15 \* p[0] + 0.28 \* p[1] + 0.0;

y = 0.26 \* p[0] + 0.24 \* p[1] + 0.44;

}

else

{

x = 0.00 \* p[0] + 0.00 \* p[1] + 0.0;

y = 0.00 \* p[0] + 0.16 \* p[1] + 0.0;

}

p[0] = x;

p[1] = y;

glBegin ( GL\_POINTS );

glVertex2fv ( p );

glEnd ( );

}

glFlush ( );

return;

}

void myinit ( )

{

glClearColor ( 1.0, 1.0, 1.0, 1.0 );

glColor3f ( 0.133, 0.545, 0.133 );

glMatrixMode ( GL\_PROJECTION );

glLoadIdentity ( );

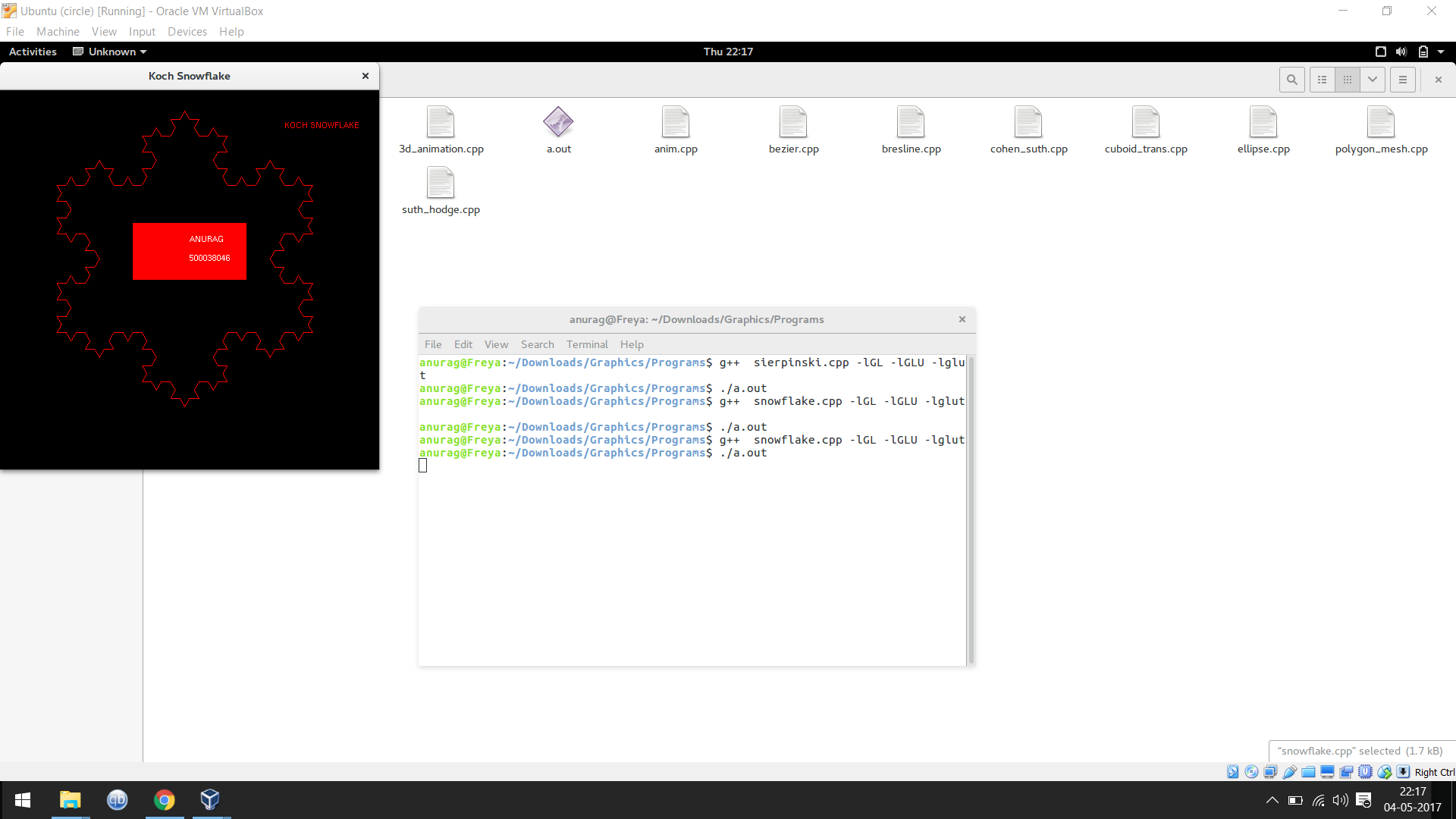
gluOrtho2D ( -4.0, 4.0, -1.0, 11.0 );

glMatrixMode ( GL\_MODELVIEW );

return;

}

**Output :**



**Q13. Write a c++ program for Polygon mesh**

**#include<GL/glut.h>**

**GLfloat xRotated,yRotated,zRotated;**

**GLdouble radius=1;**

**void printhead()**

**{**

**char sap[10] = {'5','0','0','0','3','8','0','4','6'};**

**char name[] = {'A','N','U','R','A','G'};**

**glRasterPos2f(0.0,0.1);**

**for (int i=0;i<10;i++)**

**glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10,sap[i]);**

**glRasterPos2f(0.0,0.2);**

**for (int i=0;i<8;i++)**

**glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10,name[i]);**

**}**

**void printname(float x, float y, char str[], int s)**

**{**

**glRasterPos2f(x,y);**

**for (int i=0;i<s;i++)**

**glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_10,str[i]);**

**}**

**void redisplayFunc72(void){**

**glClearColor(1.0,1.0,1.0,0.0);**

**gluOrtho2D(0.,300.0,0.0,300.0);**

**glMatrixMode(GL\_MODELVIEW);**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**glLoadIdentity();**

**glTranslatef(0.0,0.0,-4.5);**

**glColor3f(0.8,0.2,0.1);**

**glRotatef(xRotated,1.0,0.0,0.0);**

**glRotatef(yRotated,0.0,1.0,0.0);**

**glRotatef(zRotated,0.0,0.0,1.0);**

**glScalef(1.0,1.0,1.0);**

**glutSolidSphere(radius,20,20);**

**glColor3f(0.0,1.0,0.0);**

**char name[15]={'S','P','H','E','R','I','C','A','L',' ','M','E','S','H'};**

**printname(0.5,1.0,name,15);**

**glBegin(GL\_POLYGON);**

**glVertex2f(-0.3,0);**

**glVertex2f(0.3,0);**

**glVertex2f(0.3,0.3);glVertex2f(-0.3,0.3);**

**glEnd();**

**glColor3f(0.0,0.0,0.0);**

**printhead();**

**glFlush();**

**glutSwapBuffers();**

**}**

**void reshapeFunc72(int x,int y){**

**if(y==0 || x==0)**

**return;**

**glMatrixMode(GL\_PROJECTION);**

**glLoadIdentity();**

**gluPerspective(40.0,(GLdouble)x/(GLdouble)y,0.5,20.0);**

**glMatrixMode(GL\_MODELVIEW);**

**glViewport(0,0,x,y);**

**}**

**void idleFunc(void){**

**yRotated+=0.01;**

**redisplayFunc72();**

**}**

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

**glutInit(&argc,argv);**

**glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);**

**glutInitWindowSize(400,350);**

**glutCreateWindow("Spherical Mesh");**

**glPolygonMode(GL\_FRONT\_AND\_BACK,GL\_LINE);**

**xRotated=yRotated=zRotated=30.0;**

**xRotated=33;**

**yRotated=40;**

**glClearColor(0.0,0.0,0.0,0.0);**

**glutDisplayFunc(redisplayFunc72);**

**glutReshapeFunc(reshapeFunc72);**

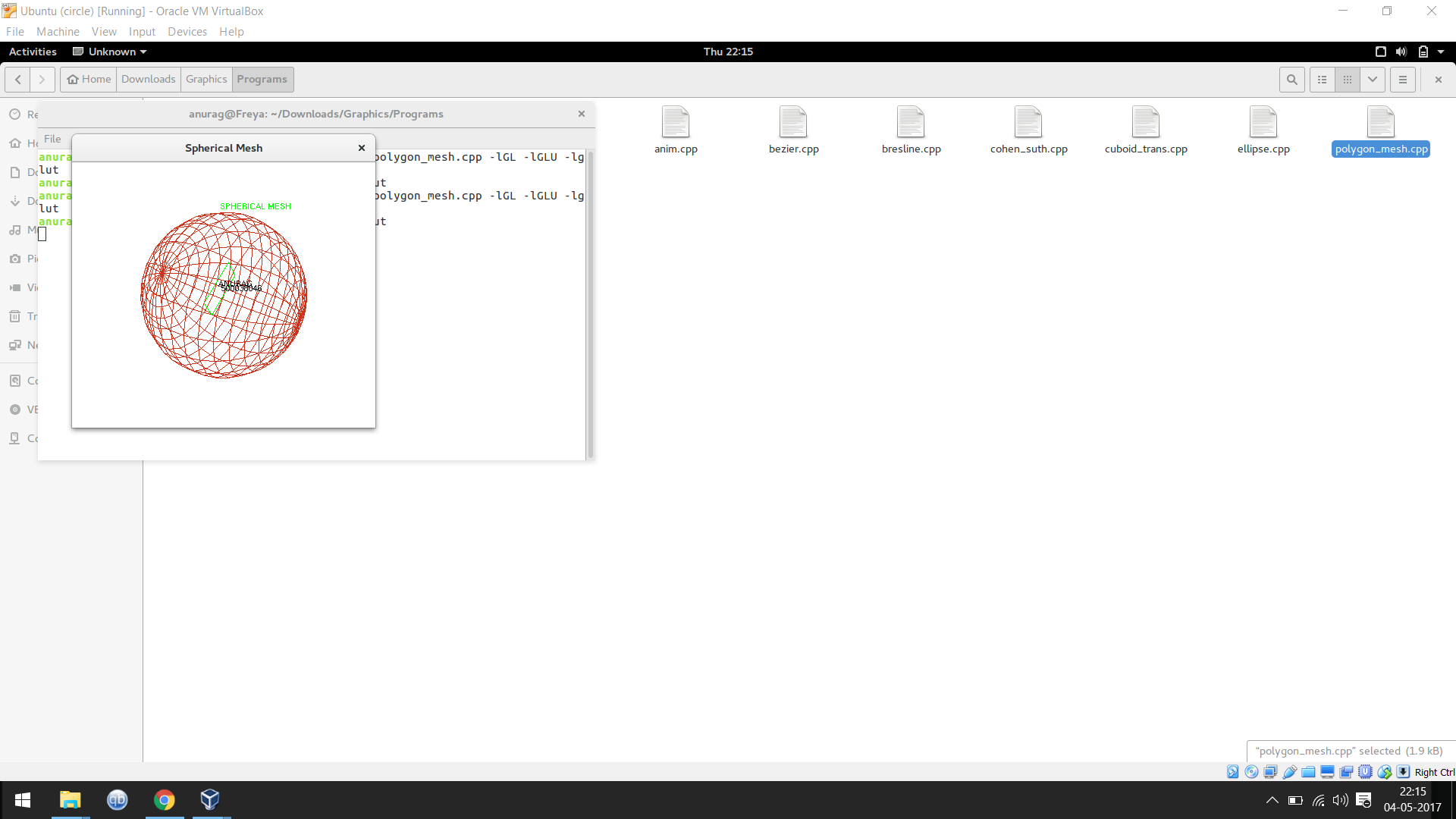
**glutIdleFunc(idleFunc);**

**glutMainLoop();**

**return 0;**

**}**

**Output:**

****