# Labset 4

Write a recursive program to construct a 3D Serpinski gasket with polygons.

#include<stdio.h>

#include<GL/glut.h>

typedef float point[3];

point v[]={{0.0,0.0,1.0},{0.0,0.942809,-0.333333},{-0.816497,0.471405,-0.333333}, {0.816497,-0.471405,-0.333333}};

int n;

void draw\_triangle(point a,point b,point c)

{

glBegin(GL\_POLYGON);

glVertex3fv(a);

glVertex3fv(b);

glVertex3fv(c);

glEnd();

}

void midpoint(point save,point a,point b)

{

save[0]=(a[0]+b[0])/2;

save[1]=(a[1]+b[1])/2;

save[2]=(a[2]+b[2])/2;

}

void divide\_tetrahedron(point a, point b, point c, point d, int m)

{

point ab, ac, ad, bc, bd, cd;

if(m>0)

{

midpoint(ab, a, b);

midpoint(ac, a, c);

midpoint(ad, a, d);

midpoint(bc, b, c);

midpoint(bd, b, d);

midpoint(cd, c, d);

divide\_tetrahedron(a, ab, ac, ad, m-1);

divide\_tetrahedron(ab, b, bc, bd, m-1);

divide\_tetrahedron(ac, bc, c, cd, m-1);

divide\_tetrahedron(ad, bd, cd, d, m-1);

}

else

{

glColor3f(1.0,0.0,0.0);

draw\_triangle(a,b,c);

glColor3f(0.0,0.0,1.0);

draw\_triangle(a,c,d);

glColor3f(0.0,0.0,0.0);

draw\_triangle(c,b,d);

glColor3f(0.0,1.0,0.0);

draw\_triangle(a,b,d);

}

}

void display(void)

{

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

glLoadIdentity();

divide\_tetrahedron(v[0],v[1],v[2],v[3],n);

glFlush();

}

void myReshape(int w,int h)

{

glViewport(0,0,w,h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if(w<=h)

glOrtho(-2.0, 2.0, -2.0\*(GLfloat)h/(GLfloat)w , 2.0\*(GLfloat)h/(GLfloat)w ,-10.0,10.0);

else

glOrtho(-2.0\*(GLfloat)w/(GLfloat)h, 2.0\*(GLfloat)w/(GLfloat)h, -2.0, 2.0, -10.0, 10.0);

glMatrixMode(GL\_MODELVIEW);

glutPostRedisplay();

}

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

{

printf("Enter the number of divisions: ");

scanf("%d",&n);

glutInit(&argc,argv);

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

glutInitWindowSize(500,500);

glutCreateWindow("3DGasket");

glutReshapeFunc(myReshape);

glutDisplayFunc(display);

glEnable(GL\_DEPTH\_TEST);

glClearColor(1.0,1.0,1.0,1.0);

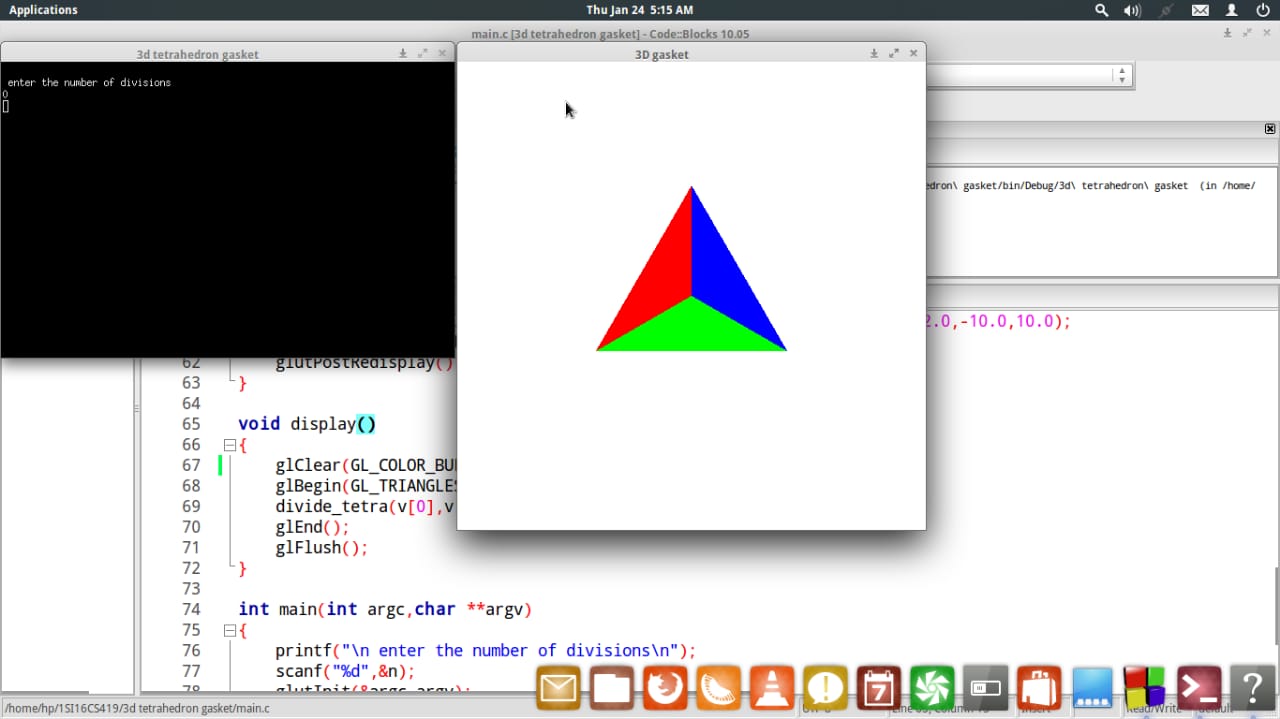
glutMainLoop();

return 0;

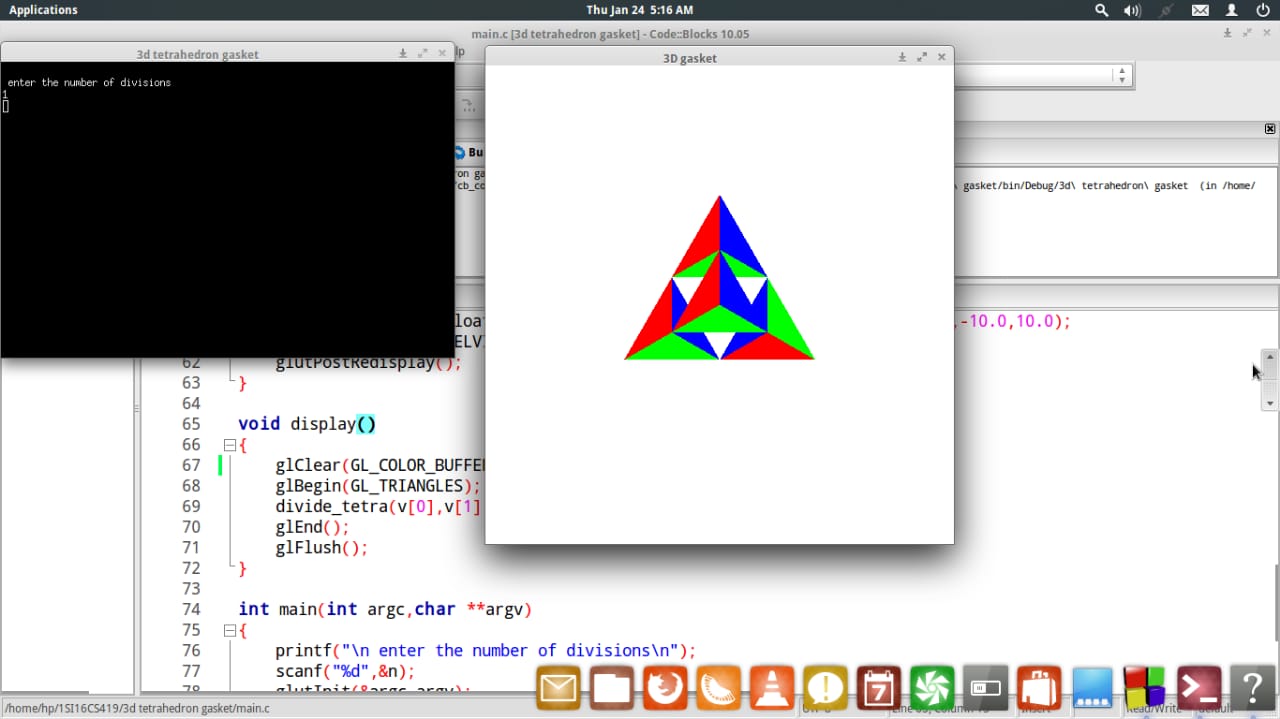
}

## Output:

1. Enter number of divisions: 0



1. Enter number of divisions: 1



1. Enter number of divisions: 2

