**EXPERIMENT - 07**

**7. Program to recursively subdivide a tetrahedron to from 3D Sierpinski gasket. The number of recursive steps is to be specified by the user**

#include <stdlib.h>

#include <stdio.h>

#include <GL/glut.h>

typedef float point[3] **;**

point v[]=**{{0.0, 0.0, 0.0}, {0.0, 1.0, -1.0},{-1.0, -1.0, -1.0},**

**{1.0, -1.0, -1.0}};**

int n;

void triangle( point a, point b, point c)

{

glBegin(GL\_POLYGON);

glVertex3fv(a);

glVertex3fv(b);

glVertex3fv(c);

glEnd();

}

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

{

point v1, v2, v3;

int j;

if(m>0)

{

for(j=0; j<3; j++) v1[j]=(a[j]+b[j])/2;

for(j=0; j<3; j++) v2[j]=(a[j]+c[j])/2;

for(j=0; j<3; j++) v3[j]=(b[j]+c[j])/2;

divide\_triangle(a, v1, v2, m-1);

divide\_triangle(c, v2, v3, m-1);

divide\_triangle(b, v3, v1, m-1);

}

else(triangle(a,b,c)); /\* draw triangle at end of recursion \*/

}

void tetrahedron( int m)

{

glColor3f(1.0,0.0,0.0);

divide\_triangle(v[0], v[1], v[2], m);

glColor3f(0.0,1.0,0.0);

divide\_triangle(v[3], v[2], v[1], m);

glColor3f(0.0,0.0,1.0);

divide\_triangle(v[0], v[3], v[1], m);

glColor3f(0.0,0.0,0.0);

divide\_triangle(v[0], v[2], v[3], m);

}

void display(void)

{

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

glLoadIdentity();

tetrahedron(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);

}

void 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("3D Gasket");

glutReshapeFunc(myReshape);

glutDisplayFunc(display);

glEnable(GL\_DEPTH\_TEST);

glClearColor (1.0, 1.0, 1.0, 1.0);

glutMainLoop();

}

**OUTPUT :**

