5. Design, develop and implement recursively subdivide a tetrahedron to form 3D Sierpinski gasket. The number of recursive steps is to be specified by the user.

```
#include<stdio.h>
#include<GL/glut.h>
typedef float point[3];
point v[]={{0.0,0.0,1.0},{0.0,1.0,0.0},{-1.0,-0.5,0.0}, {1.0,-0.5,0.0}};
float colors[4][3]={{1.0,0.0,0.0},{0.0,1.0,0.0},{0.0,0.0,1.0},{0.0,0.0,0.0}};
int n;

void triangle(point a,point b,point c)
{
glBegin(GL_TRIANGLES);
glVertex3fv(a);
glVertex3fv(b);
glVertex3fv(c);
glEnd();
}

void divide_tetra(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;
 v2[j] = (a[j] + c[j]) / 2;
 v3[j] = (b[j] + c[j]) / 2;
 }
 divide_tetra(a,v1,v2,m-1);
 divide_tetra(c,v2,v3,m-1);
 divide_tetra(b,v3,v1,m-1);
 }
 else
 {
 triangle(a,b,c);
 }
}

void tetrahedron(int m)
{
```

Scanned with CamScanner

```
glColor3f(1.0,0.0,0.0);
divide_tetra(v[0],v[1],v[2],m);
glColor3f(0.0, 1.0, 0.0);
divide_tetra(v[3], v[2], v[1], m);
glColor3f(0.0, 0.0, 1.0);
divide_tetra(v[0], v[3], v[1], m);
glColor3f(0.0, 0.0, 0.0);
divide_tetra(v[0], v[2], v[3], m);
}

void display(void)
{
glClearColor(1.0,1.0,1.0,1.0);
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
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*(float)h/(float)w,2.0*(float)h/(float)w ,-10.0,10.0);
else
        glOrtho(-2.0*(float)w/(float)h,2.0*(float)w/(float)h,-2.0,2.0,-10.0,10.0);
glMatrixMode(GL_MODELVIEW);
glutPostRedisplay();
}
int main(int argc,char *argv[])
{
printf("enter the no of division ");
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);
glutMainLoop();
return 0;
}
```

Scanned with CamScanner