Experiment No: 10

Experiment Name:

Drawing 3D Sierpinski Gasket using Sub-Division of Tetrahedron

---

Objective:

The objective of this experiment is to generate and display a **3D Sierpinski Gasket** by repeatedly **sub-dividing a tetrahedron** using OpenGL, in order to understand **fractal geometry** and recursive algorithms.

Theory:

A Sierpinski Gasket is a type of fractal that shows self-similar patterns at different levels of detail. In 3D, it is formed using a tetrahedron (a solid with four triangular faces).

The gasket is generated by:

1. Starting with a single tetrahedron
2. Dividing it into smaller tetrahedrons
3. Removing the central tetrahedron
4. Repeating the process recursively

This recursive subdivision produces a complex 3D fractal structure.

In OpenGL:

- Triangles are used to draw tetrahedron faces
- Recursion is used for subdivision
- Different colors help visualize depth

Requirements:

- Code::Blocks
- C++
- OpenGL, GLUT Library

Procedure / Description of Code:

1. Initialize GLUT and create an OpenGL window.
2. Define four vertices of a tetrahedron.
3. Create a recursive function to subdivide the tetrahedron.
4. At each recursion level, divide the tetrahedron into four smaller ones.
5. Stop subdivision at the base level and draw the faces.
6. Use `GL_TRIANGLES` to draw tetrahedron faces.
7. Enable depth testing for proper 3D visualization.
8. Display the generated 3D Sierpinski Gasket on the screen.

SOURCE CODE:

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

// Depth of subdivision

int depth = 4;

/* Draw a triangle */

void drawTriangle(float *a, float *b, float *c)

{

   glBegin(GL_TRIANGLES);

      glVertex3fv(a);

      glVertex3fv(b);

      glVertex3fv(c);
```

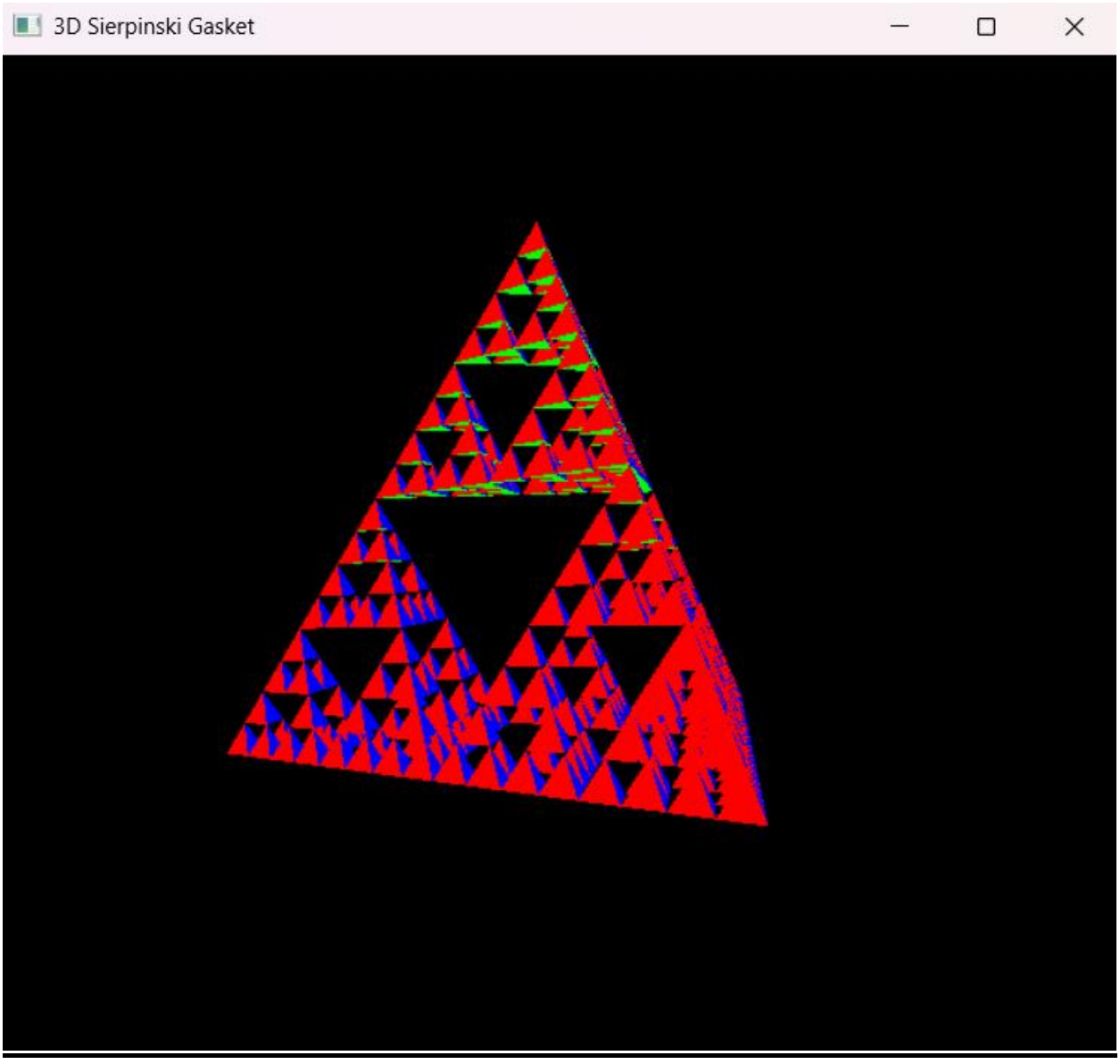
```
    glEnd();
}
/* Draw a tetrahedron */
void tetrahedron(float *a, float *b, float *c, float *d)
{
    glColor3f(1, 0, 0);
    drawTriangle(a, b, c);

    glColor3f(0, 1, 0);
    drawTriangle(a, c, d);

    glColor3f(0, 0, 1);
    drawTriangle(a, d, b);

    glColor3f(1, 1, 0);
    drawTriangle(b, d, c);
}
/* Recursive subdivision */
void divideTetra(float *a, float *b, float *c, float *d, int n)
{
```

```
float ab[3], ac[3], ad[3], bc[3], bd[3], cd[3];


if (n > 0)
{
    for (int i = 0; i < 3; i++)
    {
        ab[i] = (a[i] + b[i]) / 2;
        ac[i] = (a[i] + c[i]) / 2;
        ad[i] = (a[i] + d[i]) / 2;
        bc[i] = (b[i] + c[i]) / 2;
        bd[i] = (b[i] + d[i]) / 2;
        cd[i] = (c[i] + d[i]) / 2;
    }


    divideTetra(a, ab, ac, ad, n - 1);
    divideTetra(ab, b, bc, bd, n - 1);
    divideTetra(ac, bc, c, cd, n - 1);
    divideTetra(ad, bd, cd, d, n - 1);
}
else
```

```
    {
        tetrahedron(a, b, c, d);
    }
}
void display()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();

    glTranslatef(0.0, 0.0, -2.5);
    glRotatef(30, 1, 1, 0);

    float v[4][3] =
    {
        { 0.0, 0.0, 1.0 },
        { 0.0, 0.9428, -0.3333 },
        { -0.8165, -0.4714, -0.3333 },
        { 0.8165, -0.4714, -0.3333 }
    };
```

```
    divideTetra(v[0], v[1], v[2], v[3], depth);
    glFlush();
}
void init()
{
    glClearColor(0, 0, 0, 1);
    glEnable(GL_DEPTH_TEST);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60, 1, 1, 10);

    glMatrixMode(GL_MODELVIEW);
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB |
GLUT_DEPTH);
    glutInitWindowSize(600, 600);
```

```
    glutCreateWindow("3D Sierpinski Gasket");
    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```



Conclusion / Discussion:

This experiment demonstrates the use of **recursion** and **fractal geometry** in computer graphics. The Sierpinski Gasket shows how complex 3D structures can be created from simple geometric shapes. This concept is widely used in modeling natural objects and procedural graphics.