Programming with OpenGL
Part 6: Three Dimensions

Ed Angel

Professor Emeritus of Computer Science

University of New Mexico

Objectives

- Develop a more sophisticated three-dimensional example
  - Sierpinski gasket: a fractal
- Introduce hidden-surface removal

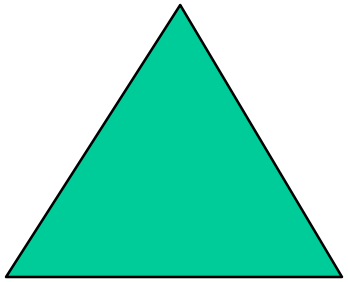
Three-dimensional Applications

- In OpenGL, two-dimensional applications are a special case of three-dimensional graphics
- Going to 3D
  - Not much changes
  - Use **`vec3, glUniform3f`**
  - Have to worry about the order in which primitives are rendered or use hidden-surface removal

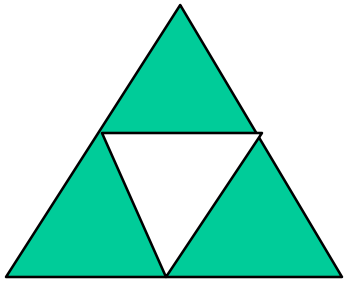
The University of New Mexico

Sierpinski Gasket (2D)

- Start with a triangle



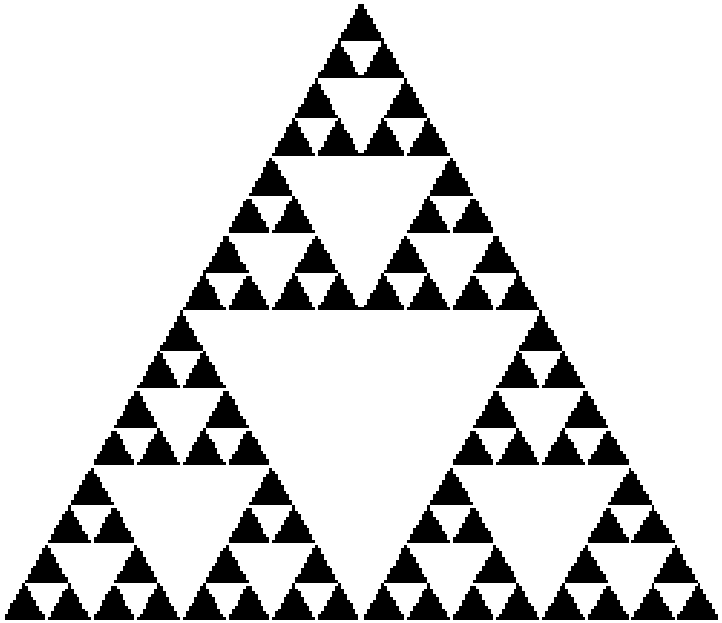
- Connect bisectors of sides and remove central triangle



- Repeat

Example

- Five subdivisions



The gasket as a fractal

- Consider the filled area (black) and the perimeter (the length of all the lines around the filled triangles)
- As we continue subdividing
  - the area goes to zero
  - but the perimeter goes to infinity
- This is not an ordinary geometric object
  - It is neither two- nor three-dimensional
- It is a *fractal* (fractional dimension) object

Gasket Program

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

/* initial triangle */

point2 v[3] ={point2(-1.0, -0.58),
              point2(1.0, -0.58),
              point2 (0.0, 1.15)};

int n; /* number of recursive steps */
```

Draw one triangle

```
void triangle( point2 a, point2 b, point2 c)

/* display one triangle  */
{
      static int i =0;

      points[i] = a;
      points[i+1] = b;
      points[i+2] = c;
      i += 3;
}
```

Triangle Subdivision

```
void divide_triangle(point2 a, point2 b, point2 c, int m)
{
/* triangle subdivision using vertex numbers */
    point2 ab, ac, bc;
    if(m>0)
    {
        ab = (a + b )/2;
        ac = (a + c)/2;
        bc = (b + c)/2;
        divide_triangle(a, ab, ac, m-1);
        divide_triangle(c, ac, bc, m-1);
        divide_triangle(b, bc, ab, m-1);
    }
    else(triangle(a,b,c));
 /* draw triangle at end of recursion */
}
```

display and init Functions

```
void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glDrawArrays(GL_TRIANGLES, 0, NumVertices);
    glFlush();
}

void myinit()
{
    vec2 v[3] = {point2(……
    .
    .
    divide_triangles(v[0], v[1], v[2], n);
    .
    .
}
```

main Function

```
int main(int argc, char **argv)
{
    n=4;
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("2D Gasket");
    glutDisplayFunc(display);
    myinit();
    glutMainLoop();
}
```

Moving to 3D

- We can easily make the program three-dimensional by using

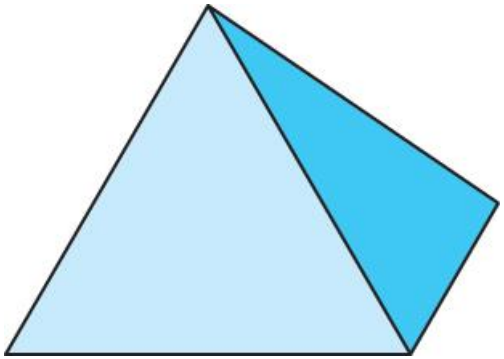
```
point3 v[3]
```

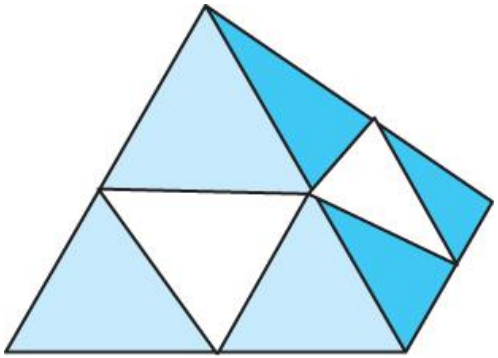
and we start with a tetrahedron

The University of New Mexico

3D Gasket

- We can subdivide each of the four faces



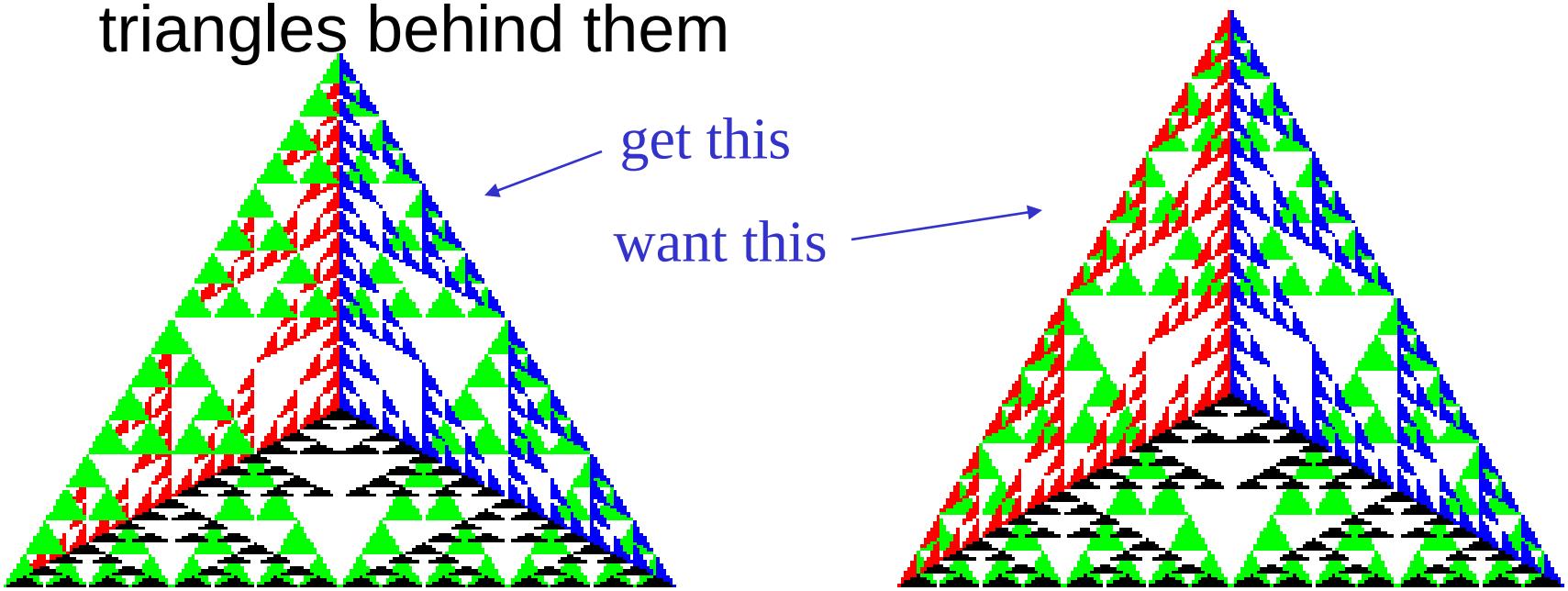


- Appears as if we remove a solid tetrahedron from the center leaving four smaller tetrahedra
- Code almost identical to 2D example

The University of New Mexico

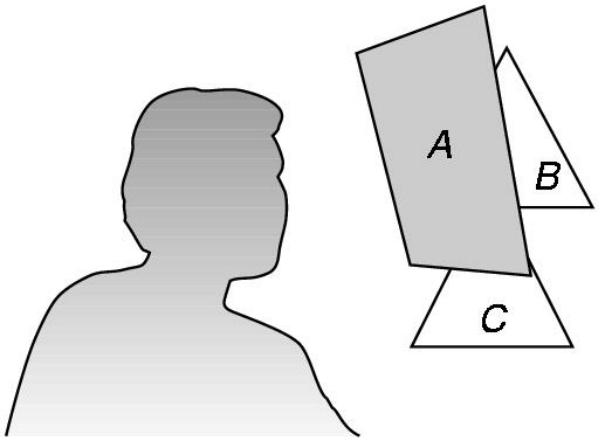
Almost Correct

- Because the triangles are drawn in the order they are specified in the program, the front triangles are not always rendered in front of triangles behind them



Hidden-Surface Removal

- We want to see only those surfaces in front of other surfaces
- OpenGL uses a *hidden-surface* method called the *z*-buffer algorithm that saves depth information as objects are rendered so that only the front objects appear in the image



Using the *z*-buffer algorithm

- The algorithm uses an extra buffer, the z-buffer, to store depth information as geometry travels down the pipeline
- It must be
  - Requested in **`main.c`**
    - **`glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH)`**
  - Enabled in **`init.c`**
    - **`glEnable(GL_DEPTH_TEST)`**
  - Cleared in the display callback
    - **`glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)`**

Surface vs Volume Subdvision

- In our example, we divided the surface of each face
- We could also divide the volume using the same midpoints
- The midpoints define four smaller tetrahedrons, one for each vertex
- Keeping only these tetrahedrons removes a *volume* in the middle
- See text for code

Volume Subdivision