CSE471 Introduction to Computer Graphics (Spring 2015)
Instructor: Prof. Won-Ki Jeong
Start date: May 1, 2015 Due date: May 17, 2015, 11:59 pm.

Assignment 3: Geometry Shader

In this assignment, you will use geometry shader to subdivide input geometry.

1. Flat shading using Geometry shader (20 points)

In your geometry shader, you can calculate per-face normal and assign it to three vertices of a triangle. For example, the skeleton code draws a unit cube without assigning per-vertex normal – therefore, the rendered image is constant color (Fig 1 left). You need to compute per-face normal and assign it to three vertices so that the image should look like Fig 1 right.

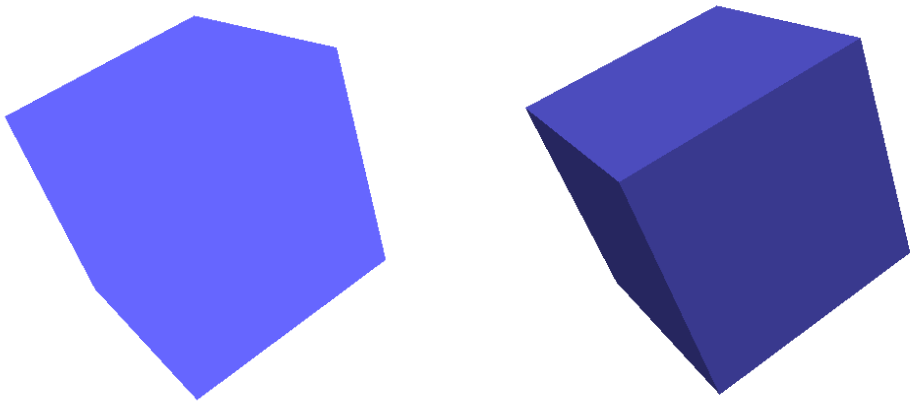

Figure 1. Example of flat shading using geometry shader. Left : no shading due to missing normals, Right : Flat shading

2. Fractal shader using Geometry shader (70 points)

In this shader, you will use a geometry shader to make fractal-like holes in each triangle. This is called "Sierpinski Triangle" (Fig 2). The main idea is to recursively split a triangle into four but do not create the center triangle. That means, you only create triangles for black regions in Figure 2. Since you cannot use recursive functions in GLSL, you need to implement this in a single pass.

Figure 2. Sierpinski Triangle. From Left to Right: Input triangle, Level 1, 2, 3, and 4, respectively.

Note that, for each level, black triangles are all in a same shape and size. That means, you can split the input triangle into same-sized triangles but discard some of them (which belong to white region). In fact, you can borrow the idea from the subdivision geometry shader we discussed in the class. For a given level, you first subdivide the input triangle into small triangles. If a triangle belongs to a black region, create the primitive. If not, discard it. To check where the triangle belongs to, you may want to use Pascal's triangle sequence (http://en.wikipedia.org/wiki/Pascal's_triangle).

To make this assignment easier, you are required to implement Sierpinski fractal only up to level 3 (in Figure 2). Use the key 0 to 3 to select the subdivision level, and make this shader working with your mesh viewer so that you can render input triangle meshes with holes.

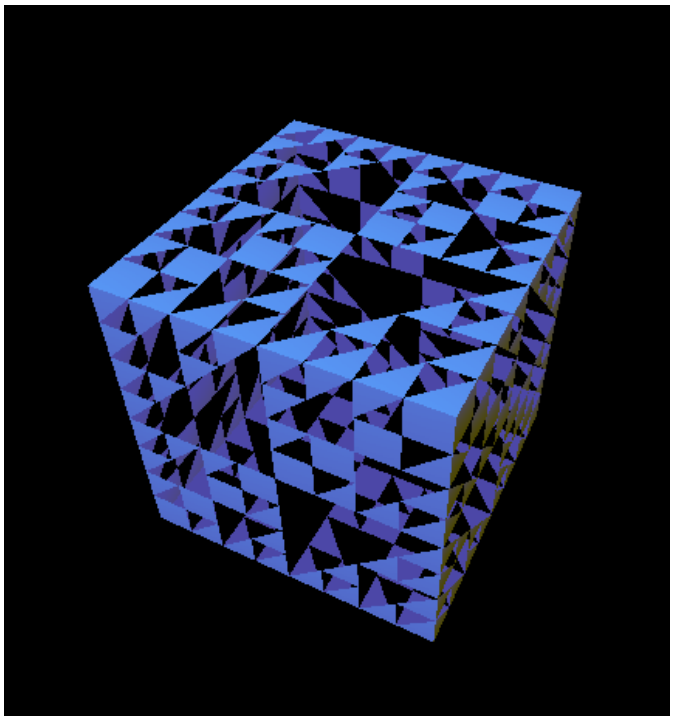

Figure 3. Expected result of Sierpenski geometry shader

Summary: functions you are required to implement are:

- Geometry shader that can make Sierpenski triangle holes on the mesh surface correctly (40 pts)
- Keyboard callbacks to change subdivision levels up to 3 (20 pts)
- Mesh viewer with virtual trackball working (10 pts)

3. Report (10 pts)

Submit a report describing your code, implementation details, and the results.