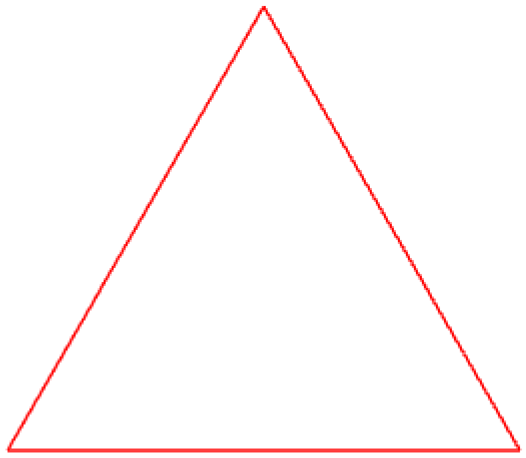
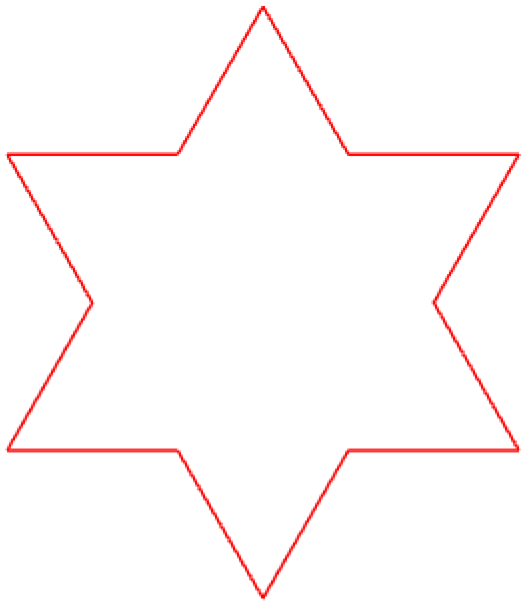
Philip Murray, CS 428

Below, 'N' represents the number of iterations performed.

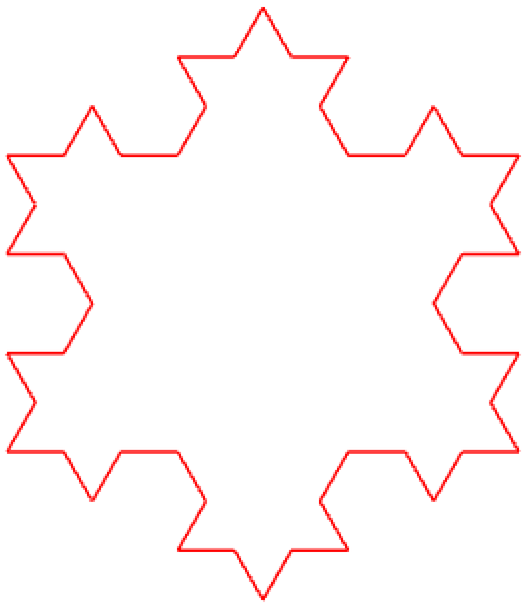
N = 0



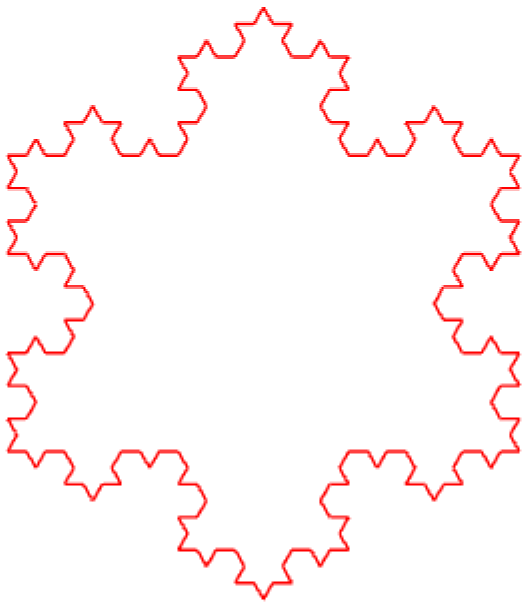
N = 1



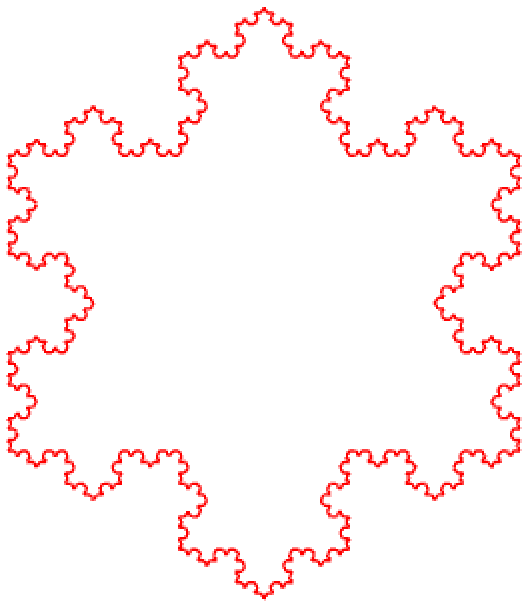
N = 2



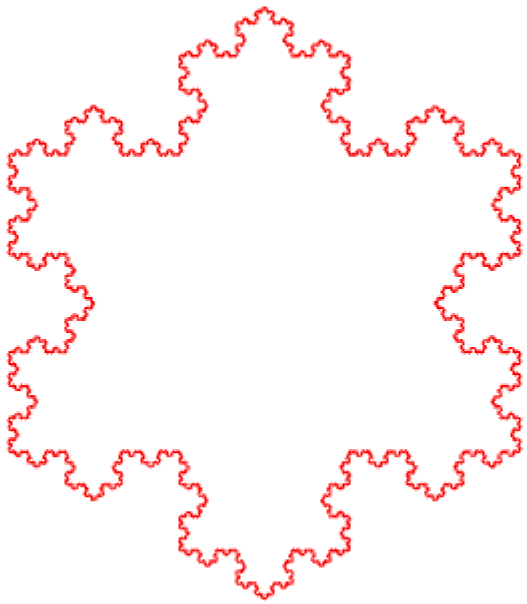
N = 3



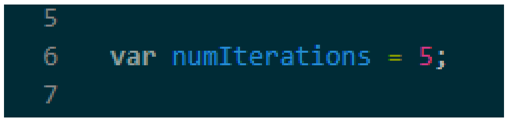
N = 4



N = 5



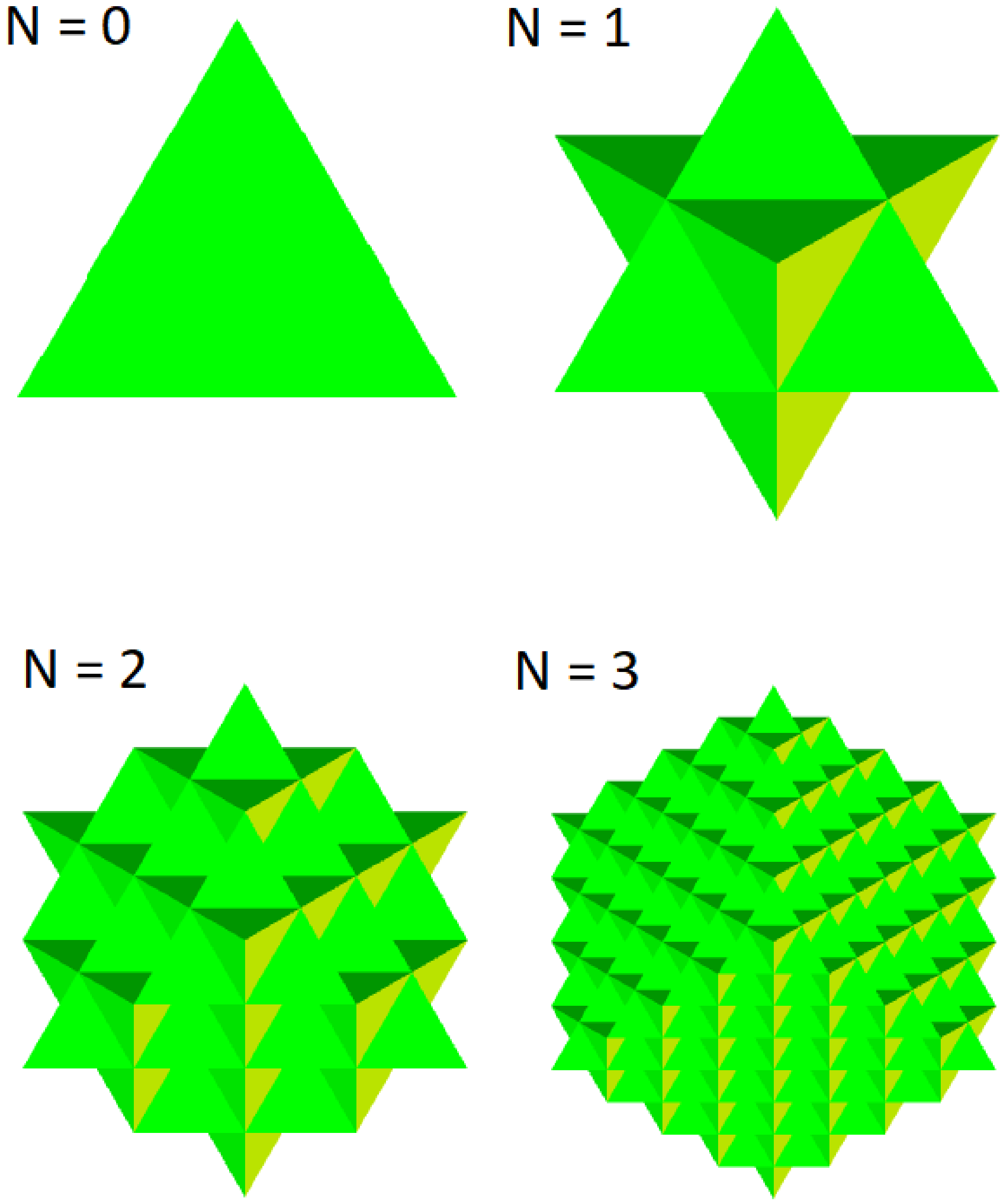
The number of iterations (times to divide) may be adjusted on line 6 of KochSnowflake.js (and line 10 of TetraFlake.js)



```
5
6   var numIterations = 5;
7
```

Tetrahedron

After making this PDF, I found a better way to color the faces of the fractal by mapping their outwards-facing direction vector to a color vector. This way, faces with the same orientation have the same color. I am going to leave this method up front as it is easier to observe, but my ‘interpretation’ of the results are afterwards on an example with semi-random color.



N = 0 is observing the bottom face of a tetrahedron in the z = 0 plane. Each face stores a direction vector, which is orthogonal to the plane made by the face, and is in the 'outwards' direction. If the face's direction vector is v, then the color vector for the face is <sine(v.x), cosine(v.y), v.z>. This way, same-shaded faces are pointing in the same direction.

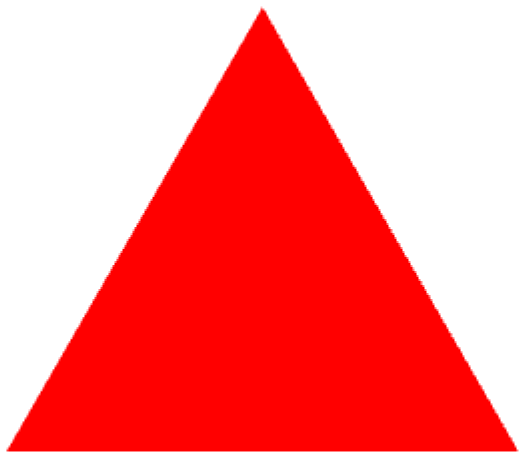
Algorithm with different colors for each new face:

For each iteration, a triangle is converted into 6 new triangles, 3 which generate a new tetrahedron, and 3 which are in the same plane as the original triangle. I've made it so that these same-planar triangles are colored the same as the original triangle. The 3 which construct a new tetrahedron are given the next three colors in the array past the color of the original triangle, where color selection repeats by the modulus operator.

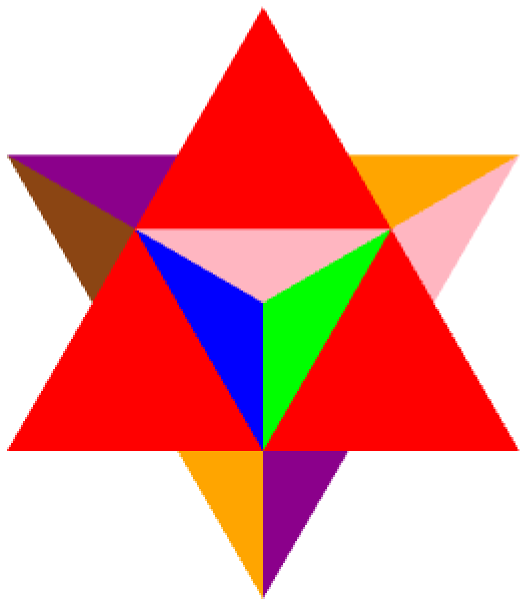
In the below figure, at N=0, the red triangle visible is the bottom face of the tetrahedron, viewing from the -Z direction (the bottom vertices all lie in Z = 0).

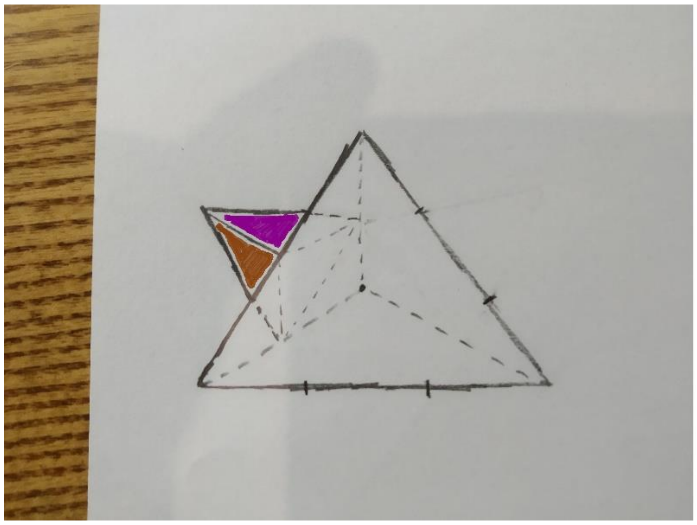
At N = 1, there is a new tetrahedron (blue-pink-green) sticking outwards from the bottom red face of the original tetrahedron.

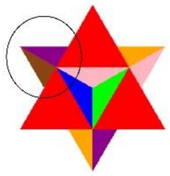
N = 0



N = 1

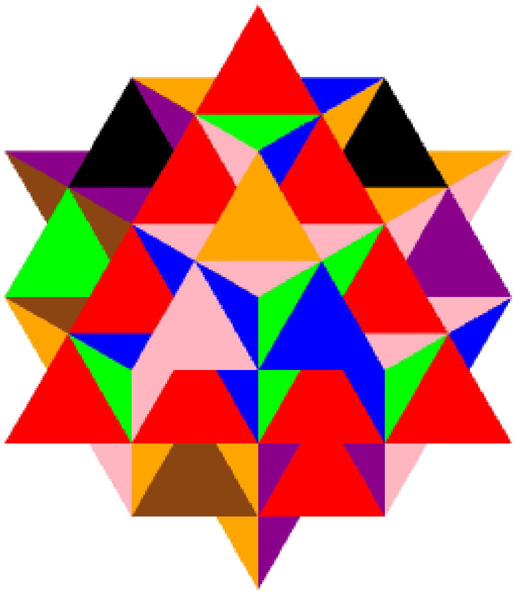




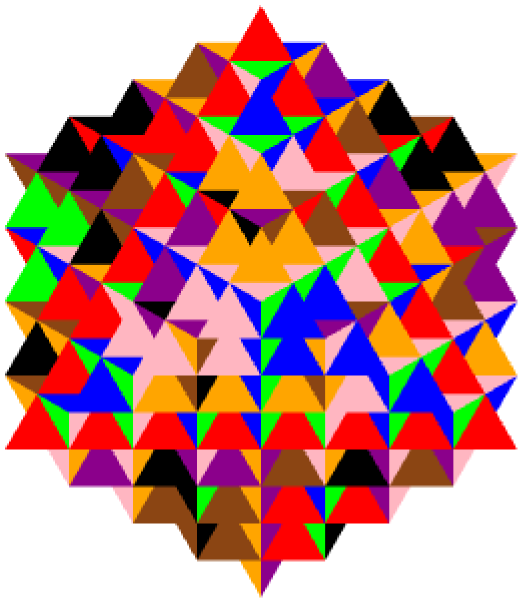


I drew the above picture to show why there appears to be three triangles (each composed of 2 triangles) sticking out of the red triangle. Behind the bottom-red face, the other three faces of the original tetrahedron each have a new tetrahedron sticking out, and only two of the faces can be seen by the viewing angle. In this case, the diagram shows the brown-purple faces of one of the sticking out tetrahedrons.

N = 2

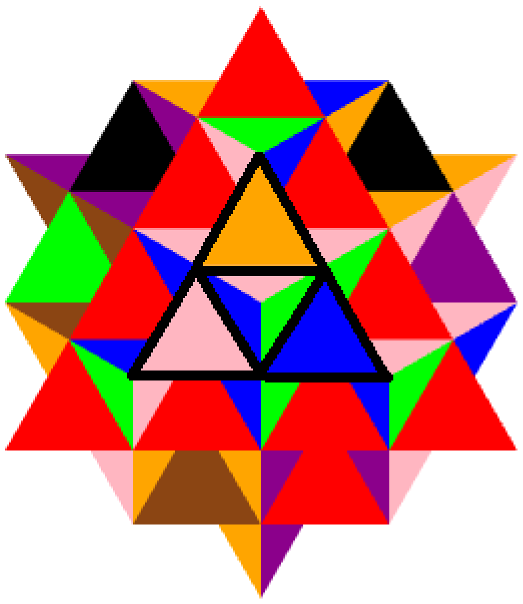


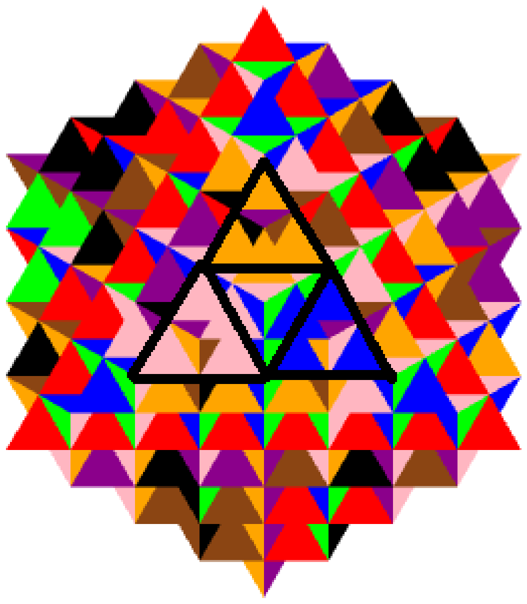
N = 3



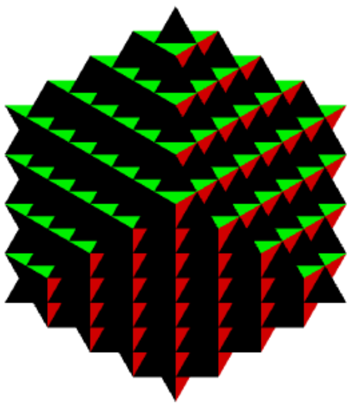
Below in a copy of the previous N = 2, I've highlighted three tetrahedrons in black (whose colors are orange, blue and pink) which are derived from the first tetrahedron which sticks out of the bottom red face in N = 1.

In the copy of N = 3, the same orange, blue, and pink tetrahedrons are visible. However, they are partially covered by new tetrahedrons that are derived from the tip of the tetrahedron that they came from. (It doesn't help that the these new covering-tetrahedrons are of the same color by the algorithm). But this view is more useful than the green shaded fractal because you can trace a subset of the tetrahedrons through a successive iteration. You can see that each of these labeled tetrahedrons has a new tetrahedron on top of them in the birds-eye view direction of the camera. Additionally, each of the labeled tetrahedrons has two more sticking out to the side, the orange one has two tetrahedrons that are both half red in color sticking out, the blue one has two with a brown color sticking out, and the pink one has two with a black color sticking out.



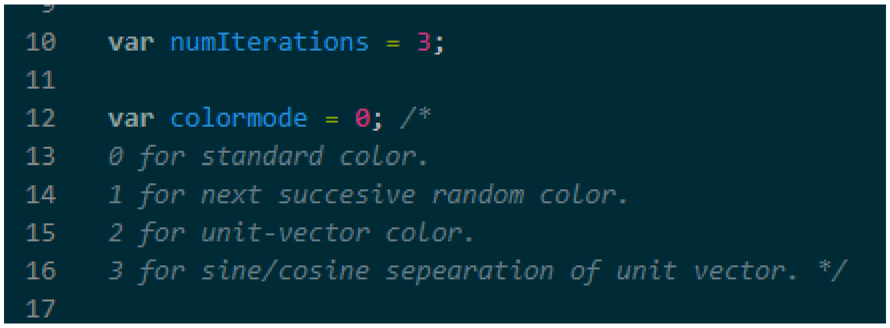


Below is an example of an N = 3 where the direction perpendicular/outwards to a face is mapped to a color. (Just plugging in a unit vector as the color3).



Only 3 colors are observed because two groups of unit vectors are close enough that they map to a shade of black. I tried applying sine and cosine to on the components to separate them into different colors, which is where I arrived at the green figure.

The number of iterations, and the coloring method can be adjusted in lines 10 and 12 of TetraFlake.js



```
10  var numIterations = 3;
11
12  var colormode = 0; /*
13  0 for standard color.
14  1 for next succesive random color.
15  2 for unit-vector color.
16  3 for sine/cosine sepearation of unit vector. */
17
```