

## Supplement to Chapter 2 of *The Science of Digital Media* – Digital Image Representation

### Programming Exercise – Digital Imaging > Algorithmic Art > Koch Snowflakes<sup>1</sup>

---

#### Introduction:

A fractal is a geometric structure or image that is characterized by a pattern that repeats at descending levels of detail. The *Koch snowflake* (or *Koch star*), named for the Swedish mathematician who first described it, is an example of a fractal. You can picture a Koch snowflake as a six-pointed star where each point of the star is itself a six-pointed star. This description is inherently recursive in that a star is defined by other stars.

The figure below shows the stepwise creation of a Koch snowflake. The wireframe views are shown so you can have a clearer idea of the recursive nature of the construction, but usually the snowflakes are shown in solid color.

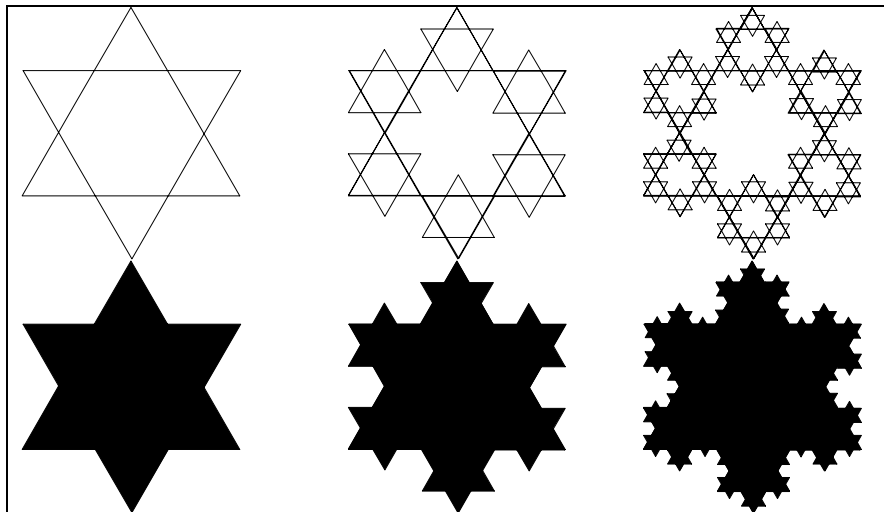


Figure 1 Construction of a Koch snowflake

#### The Assignment:

Write a recursive program to create a Koch snowflake. Your program should either (1) write a Koch snowflake bitmap to a permanent file or (2) the display the snowflake immediately to the screen.

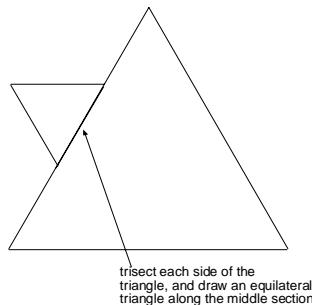
If you write your bitmap to a file, you can look at it later with Photoshop, MATLAB, or some other image viewing program.

---

<sup>1</sup>This material is based on work supported by the National Science Foundation under Grant No. DUE-0340969. This worksheet was written by Dr. Jennifer Burg, [burg@wfu.edu](mailto:burg@wfu.edu).

To implement this program, consider the recursive nature of the structure. There's more than one way to approach this. Here's one way to think about it.

```
draw_Koch_star(triangle1) {  
  /* triangle1 is an equilateral triangle. Assume you have access to its  
  vertices.*/  
  if the sides of triangle1 are too small to be further subdivided then  
    return  
  else {  
    for each side s of triangle1 {  
      divide s into three parts of equal length  
      draw an equilateral triangle whose base lies along the  
      middle section of trisected side s  
/*See the figure below. Note that each of these triangles will have sides of  
length 1/3 the length of the sides of s*/  
    }  
/*Find the 6 triangles that are the points of the star you just created and put  
these in a list called triangleList*/  
    for each triangle t in triangle_List  
      draw_Koch_star(t)  
  }  
}
```



**Figure 2 Trisecting the sides of a triangle and constructing another triangle**

Another way to think of this algorithm is as follows:

```
draw_Koch_star2(triangle1) {  
  /* triangle1 is an equilateral triangle. Assume you have access to its  
  vertices.*/  
  if triangle1 is too small to be further subdivided then  
    return  
  else {  
/*Find the center of triangle1. See the figure below. */  
    center = center(triangle1)
```

```
/*Draw a second triangle the same size as the first. Align the centers of the
two triangles, and make one side of triangle1 parallel to one side of
triangle2. See the figure below*/
```

```
    triangle2 = flip(triangle1)
```

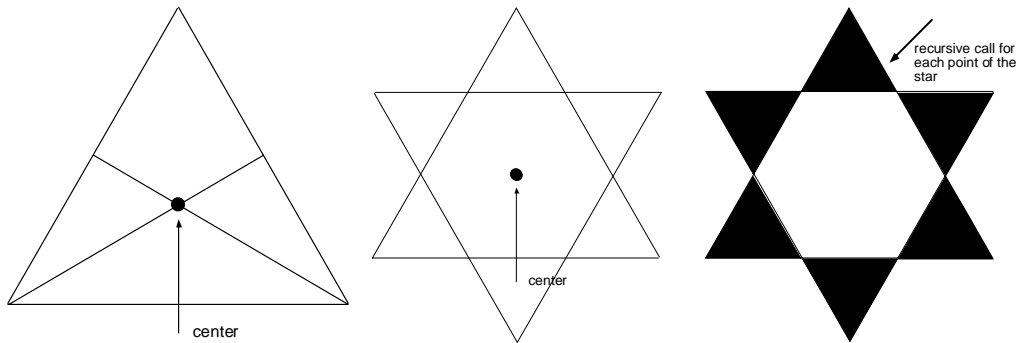
```
/* Triangle1 and triangle2 together form a 6-pointed star*/
```

```
/*Find the 6 triangles that are the points of the star and put these in a list
called triangleList*/
```

```
    for each triangle t in triangle_List
```

```
    }
```

```
}
```



**Figure 3 An alternative method for constructing a Koch snowflake**

Another way this problem has been solved is by means of "turtle graphics" programs. In these programs, the movement of a drawing mechanism is defined recursively in terms of forward movements,  $60^\circ$  turns to the left, and  $60^\circ$  turns to the right (or two moves together, for  $120^\circ$ ). The "turtle" simply draws lines as it moves along these paths. The algorithm works like this:

```
draw_Koch_star3(size, n) {
```

```
/*size gives the size of the smallest line segment in the figure. n is the level
of recursion to which to descend*/
```

```
    draw_Koch_shape(scale, n)
```

```
    turn  $-120^\circ$ 
```

```
    draw_Koch_shape(scale, n)
```

```
    turn  $-120^\circ$ 
```

```
    draw_Koch_shape(scale, n)
```

```
}
```

```
draw_Koch_shape(size, n) {
```

```
    if n = 0 then
```

```
        draw_line(size)
```

```
/*Line is drawn in the direction in which the "turtle" is moving*/
```

```
    else {
```

```
        draw_Koch_shape(size, n-1)
```

```
        turn  $60^\circ$ 
```

```
        draw_Koch_shape(size, n-1)
```

```
        turn  $-120^\circ$ 
```

```
        draw_Koch_shape(size, n-1)
```

```
        turn 60°  
        draw_Koch_shape(size, n-1)  
    }  
}
```

These algorithms give you the basic concept. It's up to you to choose an efficient way to implement the program.

### **Creative Exploration**

Koch stars are just one example of a recursive fractal structure. Other interesting structures include Gosper's flowsnake, Sierpinski's gasket, and a whole host of fractals similar to the Koch snowflake but varying in how the sides are divided. The Koch snowflake actually can be generalized in a number of ways. You can use any regular polygon for your initial shape, and you can divide the sides into different numbers of sections. You can even apply the approach to 3-D figures.

See if you can design an interesting fractal structure of your own and implement it in a recursive program.