Koch-Like Fractal Images

Vincent J. Matsko
Department of Mathematics
University of San Francisco

vince.matsko@gmail.com

Abstract

The Koch snowflake is defined by a recursive sequence of forward moves and turns. By modifying the angles at which the turns are made, a surprising variety of images may be produced. Conversion of recursion to iteration allows for straightforward image and movie generation.

Introduction

The Koch curve, shown in Figure 1, is perhaps the simplest fractal which may be generated recursively. It

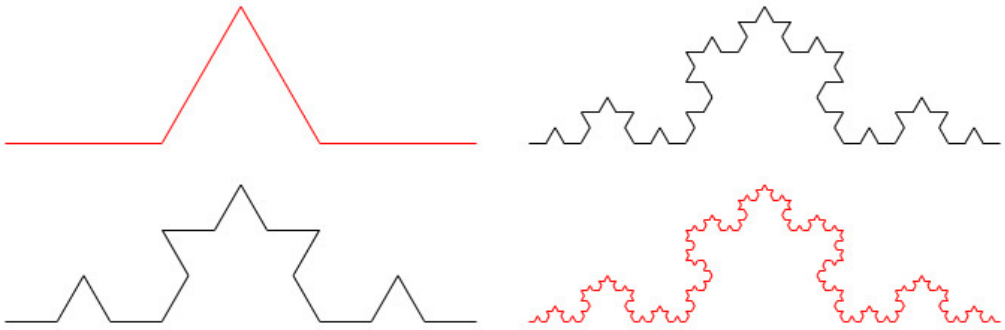


Figure 1 : *The Koch curve.*

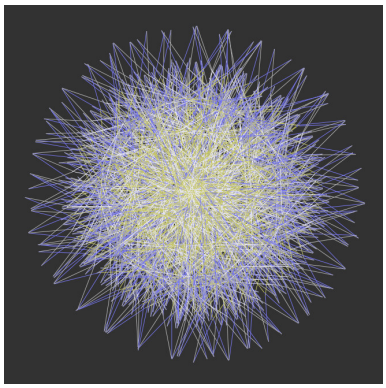
may be represented by the following rule:

$$\mathsf{F} \; + 60 \; \mathsf{F} \; + 240 \; \mathsf{F} \; + 60 \; \mathsf{F}. \tag{1}$$

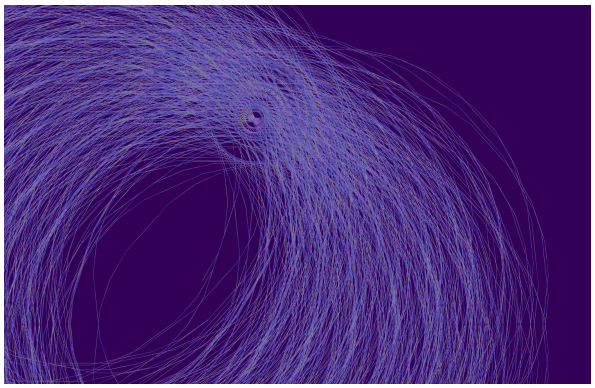
For the first iteration of the Koch curve, we go forward a given length, then turn counterclockwise $60°$, move forward again, turn counterclockwise $240°$ (equivalently, $120°$ clockwise), forward again, turn $60°$, and then move forward one last time. This is clearly seen in the first curve in Figure 1.

Next, recursively replace each of the segments just drawn with a copy of the first iteration, scaled by a factor of $1/3$. Then repeat as many times as desired. Practically, the resolution of the computer screen or printer will be reached – although abstractly, this process may be repeated infinitely many times.

One day, while explaining this to Thomas (one of my students), he asked what would happen if the same algorithm were used, but the angles were changed. A remarkably wide range of behaviors may be observed, two of which are shown in Figure 2. The path may close and repeat after just a few iterations, or after thousands. Most paths do not close and repeat, but exhibit chaotic behavior. This paper provides an introduction to an analysis of these behaviors.



Angles of $\alpha_0 = 11$ *and* $\alpha_1 = 191$.



Angles of $\alpha_0 = 0$ *and* $\alpha_1 = 12$.

Figure 2: *Sample results using a fractal algorithm.*

Recursion vs. Iteration

While the recursive generation of these images is fairly simple to implement, there are difficulties with this approach. For example, the image in Figure 3 would require 6 levels of recursion to create, resulting in $4^6 = 4096$ steps forward. Not only are some segments traversed more than once along the way, but just 1440 steps of the algorithm are necessary to actually produce the final image. To study this behavior, an iterative algorithm to generate these images is helpful.

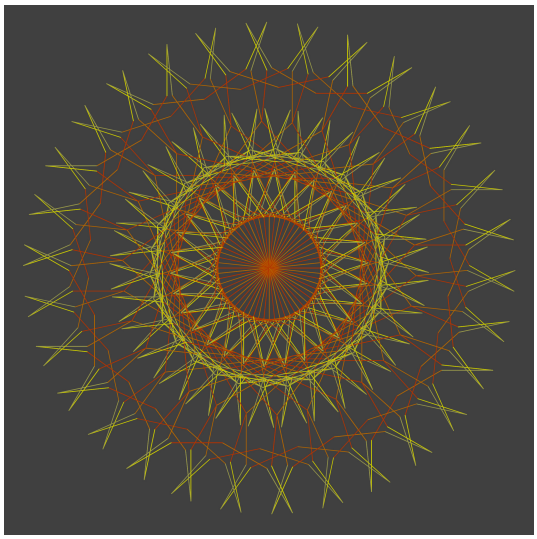


Figure 3: *Angles of* $\alpha_0 = 30$ *and* $\alpha_1 = 186$.

Suppose two angles α_0 and α_1 are given (the choice of subscripts will be explained soon), and consider the recursive procedure described by

$$\mathsf{F} \; +\alpha_0 \; \mathsf{F} \; +\alpha_1 \; \mathsf{F} \; +\alpha_0 \; \mathsf{F}.$$

The first level of recursion results in

$$(\mathsf{F}+\alpha_0\,\mathsf{F}+\alpha_1\,\mathsf{F}+\alpha_0\,\mathsf{F})\,\alpha_0\,(\mathsf{F}+\alpha_0\,\mathsf{F}+\alpha_1\,\mathsf{F}+\alpha_0\,\mathsf{F})\,\alpha_1\,(\mathsf{F}+\alpha_0\,\mathsf{F}+\alpha_1\,\mathsf{F}+\alpha_0\,\mathsf{F})\,\alpha_0\,(\mathsf{F}+\alpha_0\,\mathsf{F}+\alpha_1\,\mathsf{F}+\alpha_0\,\mathsf{F}),$$

and in general, higher levels of recursion results in a sequence of instructions

$$\mathsf{F}\ +\theta_1\ \mathsf{F}\ +\theta_2\ \mathsf{F}\ +\theta_3\ \mathsf{F}\ \ldots\ \mathsf{F}\ +\theta_k\ \mathsf{F}\ \ldots,$$

where each θ_k is an angle determined by the recursive process. To create an iterative procedure, we need to determine if each θ_k is α_0 or α_1.

To do this, it is useful to define $\tau(k)$ as the parity of the highest power of 2 which is a factor of k. For example, $\tau(k) = 0$ if k is odd, and $\tau(40) = 3$ since $40 = 2^3 \cdot 5$ and 3 is the highest power of 2 dividing 40. We then have the following result.

Proposition:

$$\theta_k = \alpha_{\tau(k)}. \tag{2}$$

We will prove this by induction on the level of recursion. The base case is immediate. Now suppose the formula for θ_k is valid for the first n levels of recursion – that is, the first $4^n - 1$ angle choices. We now carefully write out the angles choices for the first $n + 1$ levels of recursion:

$$\theta_1, \ldots, \theta_{4^n-1}, \alpha_0, \theta_{4^n+1}, \ldots, \theta_{4^n+(4^n-1)}, \alpha_1, \theta_{2\cdot4^n+1}, \ldots, \theta_{2\cdot4^n+(4^n-1)}, \alpha_0, \theta_{3\cdot4^n+1}, \ldots, \theta_{3\cdot4^n+(4^n-1)}.$$

Next, we look at all the angles after the first $4^n - 1$ and verify that (2) is valid. We break the analysis into six straightforward cases.

1. The first occurrence of α_0: This is θ_{4^n}, and we have $4^n = 2^{2n}$, so that $\tau(4^n) = 0$. Thus, (2) is valid.
2. $\theta_{4^n+1}, \ldots, \theta_{4^n+(4^n-1)}$: By the recursive algorithm, these are just the first $4^n - 1$ angles repeated. Thus, when $1 \le m \le 4^n - 1$, we must show that m and $m + 4^n$ have the same highest power of 2 as a factor. To this end, write $m = 2^p \cdot M$, where M is odd. Then

   $$m + 4^n = 2^p \cdot M + 2^{2n} = 2^p(M + 2^{2n-p}).$$

   Now since $m = 2^p \cdot M < 4^n = 2^{2n}$, we must have $p < 2n$, so that $2n - p > 0$ and 2^{2n-p} is even. Thus $M + 2^{2n-p}$ is odd, showing that p is also the highest power of 2 in $m + 4^n$. Again, (2) is valid.
3. The occurrence of α_1: This is $\theta_{2\cdot4^n}$. We have $2 \cdot 4^n = 2^{2n+1}$, so that $\tau(2 \cdot 4^n) = \tau(2^{2n+1}) = 1$, since $2n + 1$ is odd. So (2) is valid here as well.
4. $\theta_{2\cdot4^n+1}, \ldots, \theta_{2\cdot4^n+(4^n-1)}$: We argue just as in Case 2, since in this case we may write

   $$m + 2 \cdot 4^n = 2^p \cdot M + 2 \cdot 2^{2n} = 2^p(M + 2 \cdot 2^{2n-p}).$$

5. The second occurrence of α_0: This is $\theta_{3\cdot4^n}$. Since $3 \cdot 4^n = 3 \cdot 2^{2n}$, we have $\tau(3 \cdot 4^n) = \tau(2^{2n} \cdot 3) = 0$ since $2n$ is even, thus verifying (2) in this case.
6. $\theta_{3\cdot4^n+1}, \ldots, \theta_{3\cdot4^n+(4^n-1)}$: Again, we may argue as in Case 2. ■

We note that it is not strictly necessary to create an iterative procedure to optimize the number of steps drawn – a counter may be embedded in the recursive procedure, and the process may be stopped when the desired number of steps are drawn. However, as I made this procedure available on my mathematics blog [3], I wanted to find a way to make the coding as simple as possible – which meant creating a purely iterative procedure rather than embedding an iterative procedure within a recursive one. Finally, we note that the idea of looking at highest powers of two comes from creating an iterative algorithm to solve the Tower of Hanoi puzzle, where similar ideas are encountered.

Bounded vs. Unbounded

Once a length for the forward move F is given, the fractal algorithm may produce unbounded curves (such as the Koch curve), or bounded curves (such as in Figure 3). How may this be determined?

A complete answer to this question has yet to be determined. For the purpose of this paper, we address one aspect of this question which is helpful in finding bounded curves. Specifically, if a curve is retraced over again in *exactly* the same way, it must eventually return to the origin in its original orientation. Thus, there must be some N for which

$$\sum_{k=1}^{N} \theta_k \tag{3}$$

is a multiple of $360°$. So, if $e(N)$ denotes the number of the θ_k which are α_0 and $o(N)$ denotes the number of the θ_k which are α_1, the sum in (3) is simply

$$e(N)\alpha_0 + o(N)\alpha_1. \tag{4}$$

To get a handle on $e(N)$ and $o(N)$, let $\nu_p(N)$ denote how many numbers k, where $1 \leq k \leq N$, are such that the highest power of 2 in k is p. Then we have

$$e(N) = \sum_{p \text{ even}} \nu_p(N), \quad o(N) = \sum_{p \text{ odd}} \nu_p(N). \tag{5}$$

Note that the sums may be taken over all p, since for p large enough, we have $\nu_p(N) = 0$. We now show that:

Proposition: For $p \geq 0$ and $N > 0$, we have

$$\nu_p(N) = \left\lfloor \frac{N + 2^p}{2^{p+1}} \right\rfloor. \tag{6}$$

The proof is fairly straightforward. We note that (6) is equivalent to $\nu_p(N)$ being that integer satisfying

$$\frac{N + 2^p}{2^{p+1}} - 1 < \nu_p(N) \leq \frac{N + 2^p}{2^{p+1}}.$$

Multiplying through by 2^{p+1} and rearranging terms results in

$$2^p(2\,\nu_p(N) - 1) \leq N < 2^p(2\,\nu_p(N) + 1). \tag{7}$$

The idea of the proof may be seen by looking at a specific example, say with $p = 2$ and $N = 42$. Note that $\nu_2(42) = 5$ counts the *odd* multiples of $2^2 = 4$ less than or equal to 42 (since any *even* multiple would increase the power of 2): 4, 12, 20, 28, and 36. Also note that these odd multiples are spaced $8 = 2^3$ apart. So the largest of these must not exceed $42 : 4 + (5 - 1) \cdot 8 \leq 42$. In the general case, we have

$$2^p + (\nu_p(N) - 1) \cdot 2^{p+1} \leq N,$$

which is equivalent to the left inequality in (7).

Finally, we note that 42 must be less than the *next* odd multiple of 4 greater than $36 : 42 < 4 + 5 \cdot 8 = 44$. Generalizing, we have

$$N < 2^p + \nu_p(N) \cdot 2^{p+1},$$

which is equivalent to the right inequality in (7). ■

Now it is straightforward to find fractal curves which have the potential to close up: use (4) and (5) to check, given values of α_0 and α_1, whether there is some N (less than some given maximum) such that (3) is a multiple of 360. I used *Mathematica* to do this, and found the parameters for the image in Figure 3 this way.

So even though the analysis of this algorithm is yet incomplete, the results of the last two sections provide mathematical tools which enable us to find many thousands of potential curves to work with.

Artisitic Considerations

In creating digital art, I frequently find that there is a tension between abstractness and aesthetics. A fractal curve (as discussed in this paper) is an abstract object – a polygonal curve with each segment being the same length. At its simplest, we might simply render such a curve as a series of thin, black line segments on a white background.

To qualify as "mathematical" art, I think some faithfulness to the abstract mathematical object should be maintained. To this end, I decided to work within the following self-imposed constraints:

1. Use only four colors for the line segments. This is faithful to the motif $\mathsf{F} + \alpha_0 \ \mathsf{F} + \alpha_1 \ \mathsf{F} + \alpha_0 \ \mathsf{F}$: each occurrence of F is assigned a given color, so that the algorithm "creates" a color texture by virtue of its recursive structure. Figure 3 is a good example of how a layering effect (concentric circles) may be created by judiciously choosing the four colors to be similar in pairs.
2. Use a single color for the background. It should be evident by looking at Figures 2 and 3 that the choice of background color substantially affects the final image.
3. Keep line widths constant. However, a slight liberty it taken, as seen in Figure 4. Here, the fluorescent effect is created by overlaying several copies of the same curve. The thickness of the lines in each copy is the same – but the darker blue copies are drawn with thicker lines, which get progressively thinner as the blue becomes brighter.

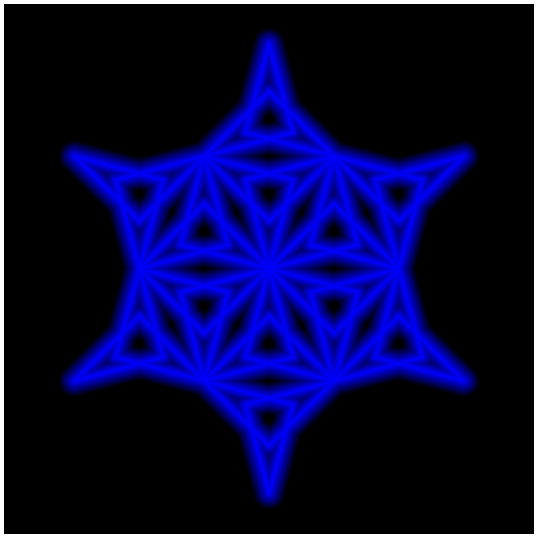


Figure 4: *Angles of* $\alpha_0 = 90$ *and* $\alpha_1 = 150$.

Color choices are critical, and are sometimes suggested by the image itself. For example, the image in Figure 5 reminded me of pine needles, and so my color palette reflected this. Also note that this is an

excerpt from a larger image. Selecting a region to use for the final images involves considerations of figure and ground.

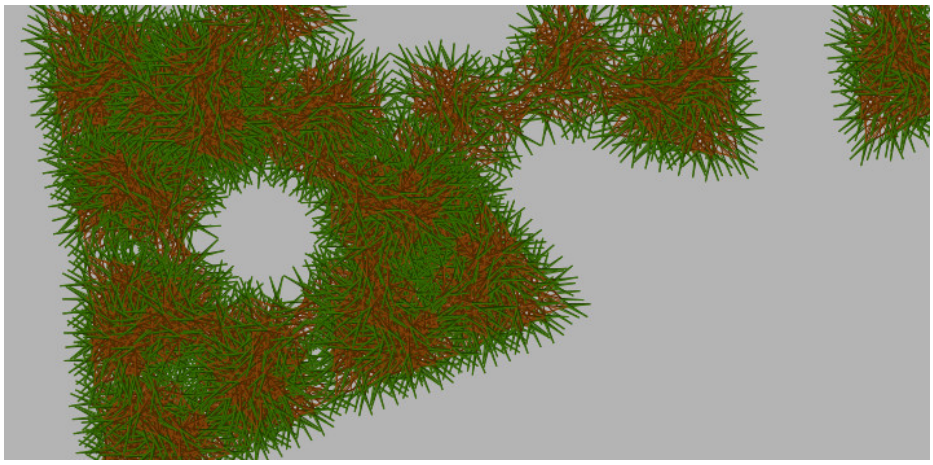


Figure 5 : *Angles of* $\alpha_0 = 11$ *and* $\alpha_1 = 185$.

L-Systems

The algorithm described in [2] is an example of general class of algorithms called L-*systems*, after theoretical biologist Aristid Lindenmayer, who first described them in 1968 [4]. I first became interested in L-systems while encountering the wonderful book *The Algorithmic Beauty of Plants* [2].

The Koch curve is a relatively simple example of an L-system. Not only is the recursive description fairly brief, we have also assumed that each occurrence of F draws a segment of the same length, and that the first and third of the three angles required are equal. We cannot offer a full discussion here, but will present a few examples of what happens with more complex systems.

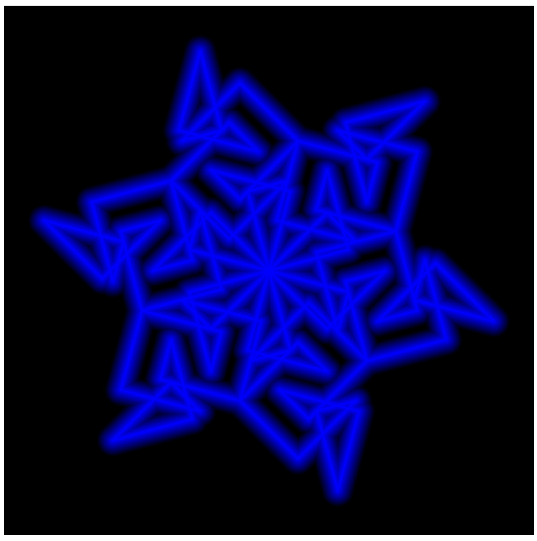


Figure 6 : *Lengths in the ratios* $2 : 2 : 1 : 2$.

If we alter just one length in the image in Figure 4 – making the third occurrence of F half as long as the others – we obtain Figure 6. While there are certainly similarities between the two figures, it is not obvious that they were created by the same algorithm except for the one change specified.

L-systems may also involve mutually recursive functions, such as one for describing the Sierpinksi triangle [4]:

$$\mathsf{F} \to +60\ \mathsf{G} + 300\ \mathsf{F} + 300\ \mathsf{G} + 60, \qquad \mathsf{G} \to +300\ \mathsf{F} + 60\ \mathsf{G} + 60\ \mathsf{F} + 300.$$

Here, we begin with F, which mutually recursively calls G.

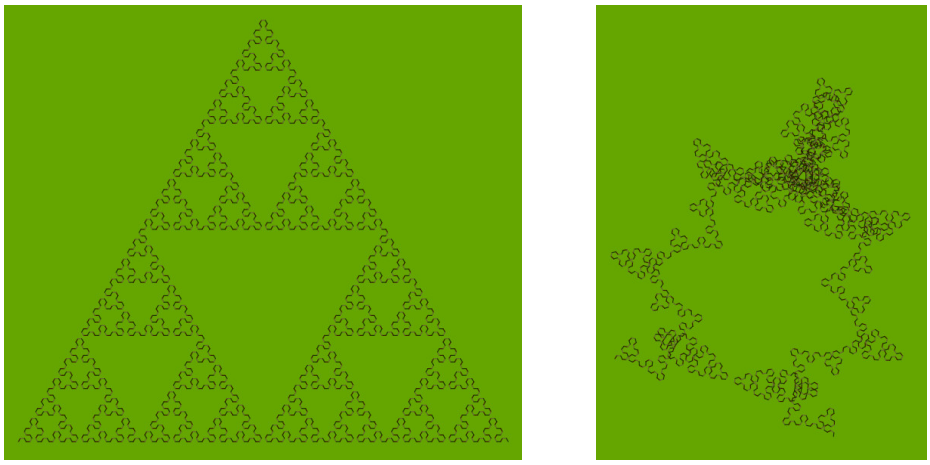


Figure 7 : *Sierpinski triangle and variation.*

In Figure 7, we see seven levels of this recursion creating the Sierpinksi triangle on the left. By modifying the angles slightly – changing the 300° to 298.994° – we obtain the figure on the right of Figure 7, which resembles a beetle. Still seven levels deep, there is much more interaction between parts of the curve with a change of only about one-third of a percent in one of the angles. As the L-systems become more complex, there is virtually infinite variation in what can be created.

Making Movies

In addition to creating individual images, movies may easily be created using the Processing platform. If two fractal images have the same number of vertices, intermediate images may be created by linearly interpolating between the vertices of the initial and the final images. Additional effects may be created by also performing a linear interpolation between background colors or colors of segments in the image.

However, interpolation is also possible if two fractal images do not have the same number of vertices. In Figure 8 below, the final image has three times as many vertices as the initial image. In the first few screen shots, the triplication of the initial image is easily seen. As the movie progresses, the final image takes shape.

The challenge in creating animations is selecting from the large number of special effects it is possible to introduce. For the movie illustrated in Figure 8, special effort was made to make the development of the images interesting from a geometrical point of view, and having this perspective dominate other concerns.

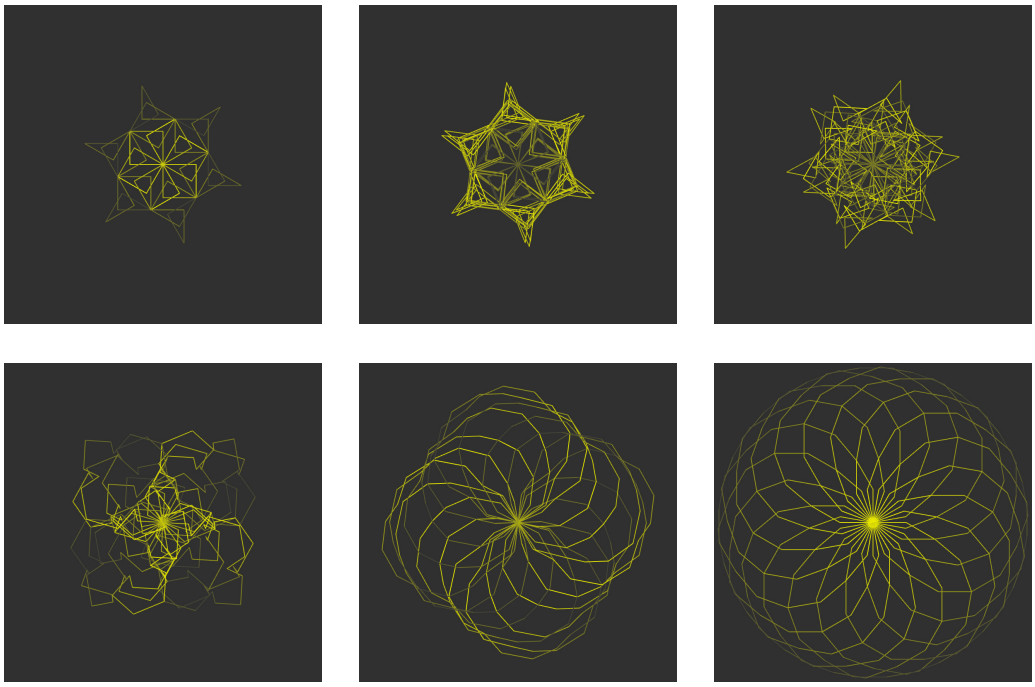


Figure 8 : *Screen shots from Processing.*

Concluding Remark

It is hoped that these few examples illustrate the virtually infinite world of recursively generated fractal images. The large number of parameters which may be modified – including the creation of entirely new recursive algorithms – allows for an extensive range of images which may be produced. Rendering these images in a way which emphasizes the inherent geometry requires careful use of computer-generated effects.

Acknowledgments

I would like to thank Thomas Biba for asking the right question, and Matthieu Pluntz for discovering the image in Figure 4 and engaging in a lively email conversation about generating fractal images.

References

[1] Ibrahim, M., and Krawczyk, R., *Exploring the Effect of Direction on Vector-Based Fractals,* in R. Sarhangi, ed., *Proceedings of Bridges 2002: Mathematical Connections in Art, Music, and Science,* Bridges Conference, pp. 213–219.

[2] Lindenmayer, A., and Prusinkiewicz, P., *The Algorithmic Beauty of Plants,* Springer-Verlag, New York, 1990.

[3] Matsko, Vincent J., *Creativity and Mathematics,* 2016. `http://www.cre8math.com` (as of Jan. 31, 2016).

[4] Wikipedia: The Free Encyclopedia, *L-system.* `https://en.wikipedia.org/wiki/L-system` (as of Jan. 31, 2016).