AN ALGORITHM TO GENERATE MODELS OF SNOWFLAKES

PHILIP CHUNG, COLIN BLOOMFIELD

Abstract. In this paper we will describe our method of creating a computer algorithm to generate two-dimensional representations of snowflakes. The algorithm will make use of both transformation matrices from linear algebra and fractal geometry.

1. What are Fractals?

While there is a long history of mathematical work leading to and motivating fractals, the term was first coined by Benoît Mandelbrot in 1975. In 1967 Mandelbrot published a paper entitled "How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension," in which he poses the question of measuring the length of the British coastline which is particularly "jagged." The problem is that unlike in the measurement of a continuous curve with a straightedge, where reductions in the size of the ruler produces more accurate measurements of length converging to a single value, such a reduction in ruler length in measuring the coastline produces greater resultant estimates of length seemingly without bound. As stated by Mandelbrot, "When a bay or peninsula noticed on a map scaled to $1/100,000$ is reexamined on a map at $1/10,000$, subbays and subpeninsulas become visible. On a $1/1,000$ scale map, sub-subbays and sub-subpeninsulas appear, and so forth." Mandelbrot discovered this paradox through the work of Lewis Fry Richardson, who first noticed it while attempting to find a correlation between the size of two countries' shared border and armed conflict between them.[2] In referencing the encyclopedias of neighboring countries he observed a variance in measurement as great as 20%, with smaller countries often choosing more precise methods of measure than larger ones. Mandelbrot both recognizes that convention can resolve such discrepancies and stresses the importance of finding a solution to the general problem which does not rely on arbitrary agreements.

When considering the dimension of a mathematical space or object one may notice the relationship to its measure. Since a line segment is one-dimensional, using a one-foot ruler instead of a three-foot yardstick will produce a length measurement three times as great, i.e., 3^1. Similarly, for a two-dimensional square the ruler will give an area measurement of nine times as great or 3^2, and for a three-dimensional cube, $3^3 = 27$. It is this paradox, that we are considering the length of the coastline and yet upon further refinements it behaves as though its dimension is greater than 1, that motivates Mandelbrot to consider fractional dimension, assigning to natural coastlines a real-number dimension between one and two.

Date: March 26, 2015.

2. Are Snowflakes Fractals?

Snowflakes are formed as water droplets freeze on dust or other particulates in the atmosphere. From these initial seeds, more and more water molecules aggregate and an ice crystal is formed. One important difference between the freezing of water in an ice tray and the formation of a snowflake is that, in a snowflake, the freezing occurs from inside-out instead of from outside-in, producing an unstable boundary. [1] Thus as water molecules collect and freeze, small bumps form on the surface which then in turn accrue more water molecules than on neighboring regions. Thus branches begin to form, which then amass subtle surface variances, which in turn grow into sub-branches, and so on. In addition to the fractal-like self-similarity of the branches or dendrites, the hexagonal structure of snowflakes often exhibited at the macro scale is reflected at the atomic level in the arrangement of the frozen water molecules where the bonds between hydrogen and oxygen molecules are roughly at 120° angles from each other and ice solidifies in a structure of flat sheets with hexagonal tiling.

Some of the factors that are involved in determining the shape of the snowflake are the humidity and subtle differences in temperature during the descent of the snowflake. Heat diffusion as new water molecules freeze on and surface tension, while appearing inconsequential, are also said to play a significant role in determining the ultimate shape of the snowflake. [1] Thus, while there are a variety of factors which effect the resultant shape of the snowflake, the high degree of self-similiarity at various levels motivates the use of fractals in developing an algorithm for their representation.

3. Overview of Algorithm

3.1. **Linear Algebra.** The program uses matrices to transform line segments to generate new ones that are scaled, rotated and shifted. To scale a set of line segments expressed as column vectors in a $2 \times N$ matrix P, we multiply by a matrix T

$$TP = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{bmatrix} = P' \tag{1}$$

Where P' is the matrix of transformed line segments scaled by a factor of s. To shift the matrix P, called a translation by some value of x, x_0, and of y, y_0, we add a matrix T,

$$T + P = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{bmatrix} + \begin{bmatrix} x_0 & x_0 & \cdots & x_0 \\ y_0 & y_0 & \cdots & y_0 \end{bmatrix} = T' \tag{2}$$

Lastly we need to rotate the line segments. Assume there is some matrix T such that $T\vec{v}$ is the vector $\vec{v}_0$ whose column vectors are rotated by some angle θ. Considering the $\vec{v}$ has some initial angle ϕ we may express the x and y components as follows,

$$T\vec{v} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} |\vec{v}| \cos\phi \\ |\vec{v}| \sin\phi \end{bmatrix} = \begin{bmatrix} |\vec{v}| \cos(\phi + \theta) \\ |\vec{v}| \sin(\phi + \theta) \end{bmatrix}$$

Multiplying the above expression and dividing both sides by the scalar $|\vec{v}|$ yields,

$$a\sin\phi + b\cos\phi = \sin(\phi+\theta) \qquad c\sin\phi + d\cos\phi = \cos(\phi+\theta)$$

Lastly, by applying the angle-sum identities, we obtain the transformation matrix T,

$$T\vec{v} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \vec{v}_0 \tag{3}$$

3.2. **Fractal Geometry.** In defining fractals, we will limit our discussion to self-similar fractals.

Definition 3.1 (Self-Similar Curve)**.** A curve S is self-similar if and only if it can be expresses as a set of curves $S_1, S_2, \cdots, S_k$, where each set S_i is a non-overlapping subset of S congruent to S scaled by the same factor s.[1]

Spaces that have non-integer or fractional dimensions are fractals. In order to consider such cases we need an expanded notion of dimension. As in linear algebra where dimension is generalized by vector bases to encompass n-dimensional Euclidean spaces, we will make use of Hausdorff dimension to encompass self-similar fractal curves.[2]

Definition 3.2 (Hausdorff Dimension)**.** The Hausdorff dimension of a self-similar set S, denoted $d_h(S)$ is defined as

$$d_h(S) \equiv \frac{\ln k}{\ln(1/s)} \tag{4}$$

where s is the scaling factor and k is the number of self-similar subsets.

Applying the change-of-base formula to the above equation,

$$\left(\frac{1}{s}\right)^{d_h(S)} = k \tag{5}$$

which we may use to consider examples. In the case of a square of side length one, there would be four self-similar subsets of side length $1/2$. Thus $s = 1/2$ and $k = 4$ and using the above formula, $d_h(1/2) = 2$ which is what we require. A fractal example is the Koch snowflake where each set has four self-similar subsets one-third the length of the prior and so $s = 1/3$ and $k = 4$. This relationship is best expressed by the iterative generation of the snowflake shown in Figure 1.

[1]The definition of self-similarity is taken from [4] p. 625

[2]Definition of Hausdorff dimension is also taken from [4] p. 627

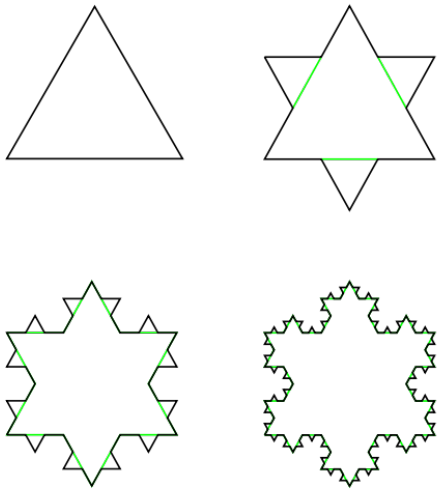

FIGURE 1. Koch snowflake [5]

Thus we may use the above equation to calculate the dimension which is approximately $d_h(S) = 1.26$, very close to the measured dimension of the coast of Britain, $d_h(S) = 1.25$ as calculated by Richardson.[3]

4. Detail of Algorithm

4.1. **Modifications to Matrix Arithmetic.** The actual implementation of the algorithm relies on some modifications to the arithmetic of transformations described in the previous section.

Translation involves adding a matrix, unlike the other two types of transformations, which involve multiplying by a matrix. This is problematic when attempting to transform an already-transformed line segment.

For example, a combination of transformations, represented by $T_1, \cdots, T_5$, on a line segment P resulting in P' may be represented in the "traditional" system as,

$$P' = T_4(T_2(T_1P) + T_3) + T_5 \tag{6}$$

If additional transformations need to be applied only after the existing transformations, it is conceivable to take a line segment from data storage, transform it, and replace the original line segment with the final result, in the same storage location. However, in some cases, a transformation T_0 must be applied *before* the others. If only the final result P' is stored, this requires the manipulation,

$$P'' = T_4(T_2(T_1T_0P) + T_3) + T_5 = F(P') \tag{7}$$

It is not clear what F should be to allow this. The only obvious way to accomplish this is to store a list of transformations applied to P and add this transformation to the beginning of the list. However, this is inefficient.

With some modifications to the matrices, all transformations can be applied through matrix multiplication. Thus, the total transformation is simply multiplication by a set of transformation matrices,

$$P' = T_5 T_4 T_3 T_2 T_1 P \tag{8}$$

The transformation matrices can be multiplied to form $\prod T$ and only this product stored. Applying a transformation T_0 at the beginning can be represented as,

$$P'' = \left(\prod T\right) T_0 P \tag{9}$$

However, even if this representation is used, finding and storing the final result has the same problem when attempting to apply a transformation before the others. Resolving this requries another modification. Specifically, only the total transformation $\prod T$ is stored, and a common base is chosen for P.

Thus, applying a transformation at the beginning is implemented as, "Set $(\prod T)$ to $(\prod T)\, T_0$." Then, at the end, each total transformation is applied to the base, and the resulting line segment is drawn.

In our implementation, this base was chosen to be,

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \tag{10}$$

That is, the line segment between $(0, 0)$ and $(0, 1)$.

As suggested by [4] p. 602 (ex. 7) and as in the SVG (Scalable Vector Graphics) format for computer graphics [6], points are represented as column vectors of a matrix with an additional element with a value of one at the end. Thus, the $2 \times N$ matrix of line segments,

$$\begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{bmatrix}$$

becomes a $3 \times N$ matrix,

$$\begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{11}$$

In this representation, to translate by values x_0 in x and y_0 in y, we *multiply* by a matrix T,

$$TP = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \\ 1 & 1 & \cdots & 1 \end{bmatrix} = P' \tag{12}$$

To scale the line segments, we multiply by a different matrix,

$$TP = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \\ 1 & 1 & \cdots & 1 \end{bmatrix} = P' \tag{13}$$

Finally, to rotate,

$$T\vec{v} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \vec{v}_0 \tag{14}$$

4.2. **Nonrandom Version.** To explore self-similar fractal snowflakes, a nonrandom algorithm was devised. It involves replacing each line segment in the current set with seven transformed line segments, as illustrated in Figure 2.

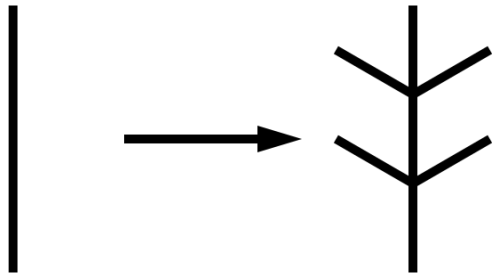

FIGURE 2. Transformations applied to a line segment, resulting in seven line segments, each a third the length of the original. The original line segment seems to reappear in the transformed set, but it is really three smaller line segments.

Assuming that the original line segment is the base segment (described in the previous subsection), the transformations are,

(1) Scale by 1/3.
(2) Scale by 1/3, then translate by $(0, 1/3)$.
(3) Scale by 1/3, then translate by $(0, 2/3)$.
(4) Scale by 1/3, rotate counterclockwise by $\pi/3$, then translate by $(0, 1/3)$.
(5) Scale by 1/3, rotate clockwise by $\pi/3$ (or counterclockwise by $-\pi/3$), then translate by $(0, 1/3)$.
(6) Scale by 1/3, rotate counterclockwise by $\pi/3$, then translate by $(0, 1/3)$.
(7) Scale by 1/3, rotate clockwise by $\pi/3$, then translate by $(0, 2/3)$.

To avoid confusion, these will be referred to as the "essential transformations."

The essential transformations can be applied only when the original line segment is the base. In order to apply them to a line segment that is not the base, the existing transformation for the original line segment is applied after applying the essential transformations to the base. Mathematically:

$$\left(\prod T\right) P \to T_i B \to \left(\prod T\right) T_i P \tag{15}$$

where $\prod T$ is the product of transformations from the base segment B to the original line segment and T_i is one of the essential transformations. As described in the previous subsection, this is achieved by multiplying $\prod T$ by T_i and storing this as the new $\prod T$.

In pseudocode, the algorithm may be expressed as,

```
Create the set of transformations S
Add the transformation of a single line segment to S
Create a temporary set of transformations T
For each iteration
    For each transformation in S
        For each essential transformation
            Multiply the current transformation from S
              by the current essential transformation
            Add this result to T
        End
    End
    Copy T to S
    Clear T
End
For each transformation in S
    Apply the current transformaton to the base
    Draw the result
End
```

Starting with a single line segment, however, does not yield a realistic figure. Instead, the algorithm first generates the transformations for six radial line segments, as shown in Figure 3, and begins from those.

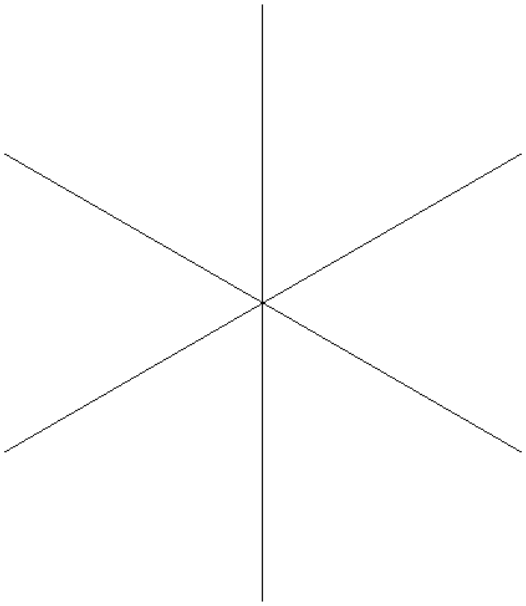

FIGURE 3. Initial line segments for the nonrandom algorithm

Successive iterations generate ever more-detailed snowflakes, as shown in Figure 4.

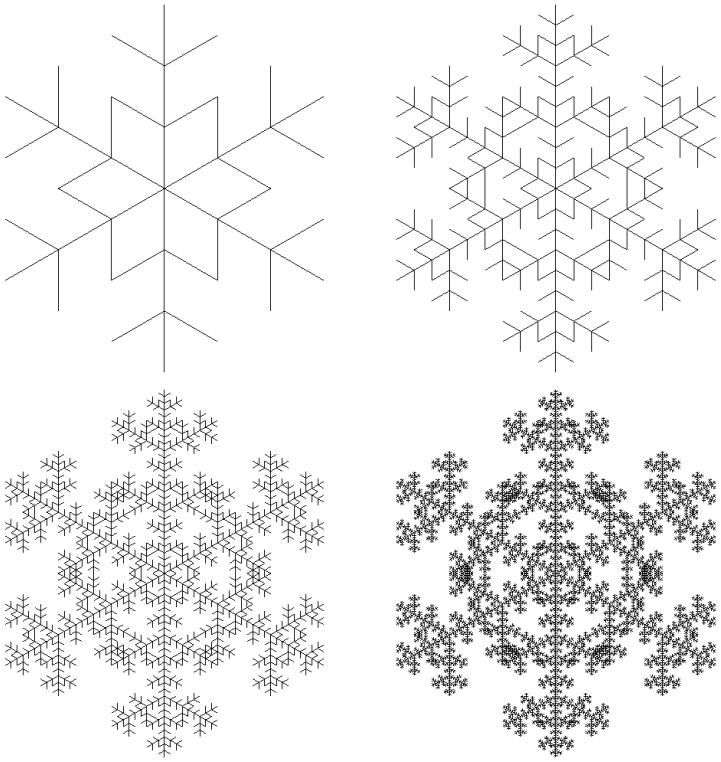

FIGURE 4. Results of the first four iterations of the nonrandom algorithm

4.3. **Randomized Version.** The nonrandom version does not capture the immense variety found in real snowflakes that leads to the saying, "No two snowflakes are alike." In order to generate different snowflakes, some randomness must be introduced. This is done by varying the scaling and translation in transformations 4–7 of the nonrandom algorithm, as shown in Figure 5. Each pair of "branches" is scaled and translated by a different, random amount.

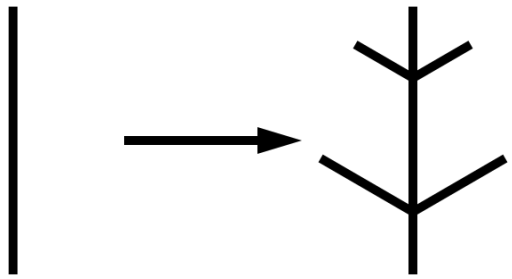

FIGURE 5. Sample transformation of a line segment with random variation

In the nonrandom version, the original line segment "reappears" in the form of three smaller line segments, which are subject to the same transformations. This results in shorter and shorter branches on the same apparent line segment. To prevent this in the randomized version, the branches are simply added to the original line segment, and the original line segment is marked as "completed" to ensure that no additional branches are added to that particular segment.

Snowflakes generally exhibit radial symmetry. So, to ensure that each radial base line has the same branches, subbranches, etc., the randomized version does not begin with the six initial radial line segments; rather, the transformations are applied to a single line segment, and the resulting line segments are all rotated to produce the remaining radii.

Experimentation has resulted in the following restrictions on the randomness that produce reasonably realistic snowflakes,

- The scale of the branches is between 1/4 and 1/3.
- The first pair of branches is translated a distance in the first half of the line segment, and the second pair is translated a distance in the second half of the line segment. More generally, for n pairs of branches, the i-th pair is translated a distance between $\frac{i-1}{n}$ and $\frac{i}{n}$ times the length of the original line segment.

The algorithm run for four iterations produces the snowflakes in Figure 6.

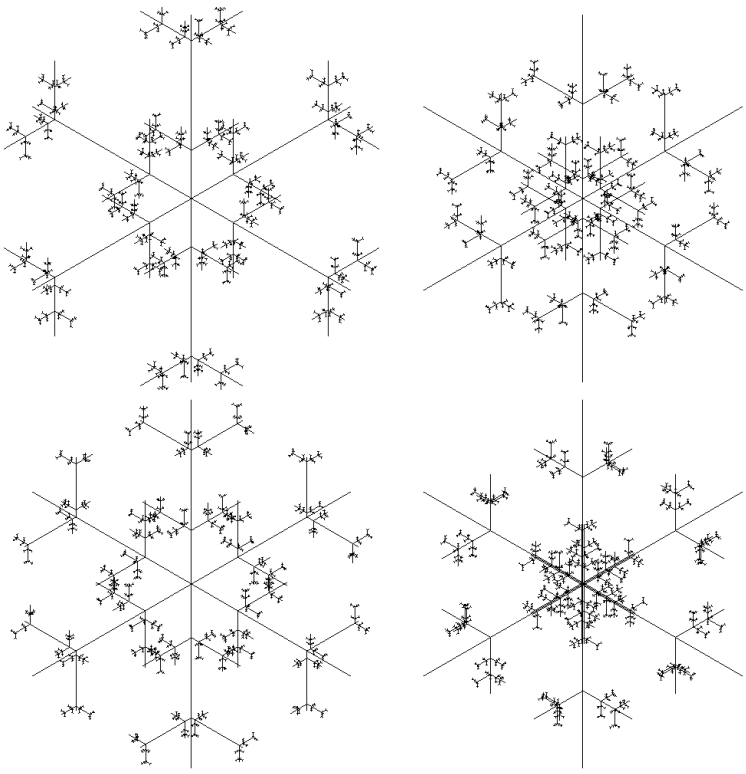

FIGURE 6. Snowflakes produced with the algorithm run for four iterations

Other implementations may wish to experiment with other variations, such as,

- The scale is between 0 and $1 - d$, where d is the proportion of the distance translated to the length of the original line segment.
- The scale is between different set values.
- The branches of each pair are scaled differently.
- The branches do not appear in pairs that are translated by the same amount.

4.4. **Pentagonal Drawing.** The algorithm works, but the resulting drawings lack "substance." This is because the line segments, having no real thickness, are drawn as thinly as possible. To make the line segments more substantial, each is transformed into a pentagon, as shown in Figure 7.

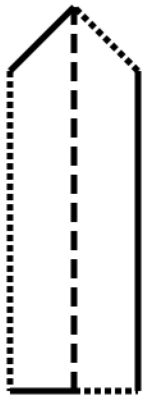

FIGURE 7. Transformation from a line segment into a pentagon. The original line segment appears in the center (dashed).

The line segment is transformed into three different line segments (shown solid in Figure 7), and the vertices of the figure are determined from the endpoints of these line segments. (The remaining edges to complete the pentagon are shown dashed in Figure 7.) To simplify the transformations, there is an additional vertex in the bottom edge of the pentagon, coinciding with one endpoint of the original line segment.

The pentagons should be equally wide, regardless of the length of the original line segment. Therefore, the scaling of the line segments should vary.

For convenience, we define a "normalizing scale" $n = \frac{w}{2l}$, where w is the desired width of the pentagon and l is the length of the line segment. Scaling the line segment by this amount results in a line segment as long as half the desired width.

Again assuming the base line segment, the transformations for the three line segments are:

(1) Scale by n, then rotate counterclockwise by $\pi/2$.
(2) Scale by $1 - n$, then translate by $(n, 0)$.
(3) Scale by $n\sqrt{2}$, rotate counterclockwise by $3\pi/4$, then translate by $(0, 1)$.

Applying this pentagonal drawing results in the snowflakes in Figure 8.

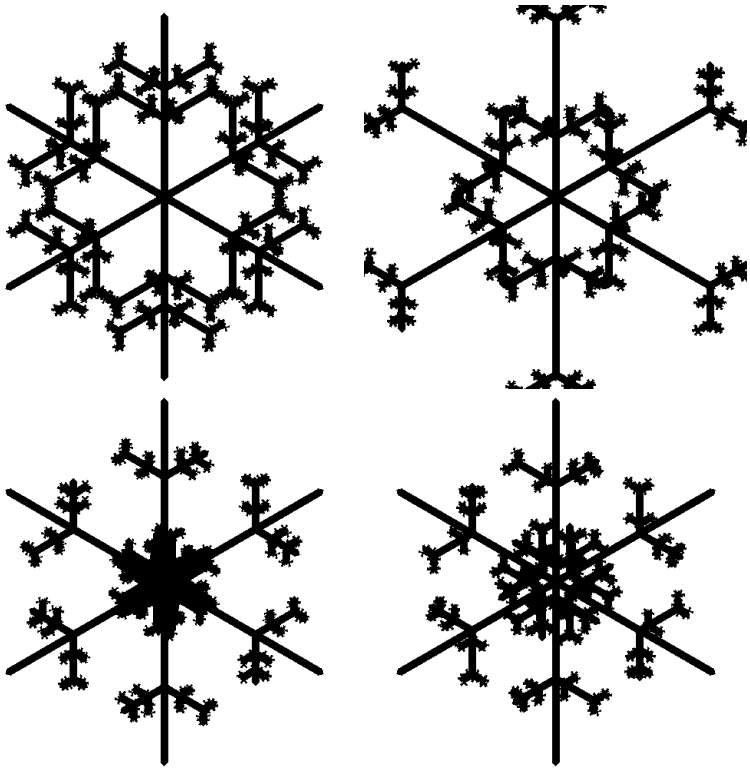

FIGURE 8. Snowflakes produced with the algorithm run for four iterations

5. Conclusions

In our discussion and model we made use of self-similarity to define a fractal which makes the Hausdorff dimension simpler but adds additional constraints on the allowable shapes of our model.[2] A definition allowing for a "statistical self-similarity"[3] which Mandelbrot uses in considering natural coastlines might allow for greater variance as is seen in the variety of forms of snowflakes found in nature.

References

[1] Gleick, J. *Chaos* New York, NY, Penguin Books (1987), pp. 127-128 2

[2] Mandelbrot, B. *How Long is the Coast of Britian?* From *The Fractal Geometry of Nature* Copyright © 1983 by Benoît B. Mandelbrot 1, 5

[3] Mandelbrot, B. *How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension.* Science, New Series, Vol. 156, No. 3775 (May 5, 1967), pp. 636-638, American Association for the Advancement of Science `http://www.jstor.org/stable/1721427` 3.2, 5

[4] Anton, H. *Elementary Linear Algebra : Applications Version 11th Edition*, Quad Graphics/Versailles, Copyright © 2014 by Anton Textbooks, Inc. 1, 2, 4.1

[5] Wxs. *KochFlake.* Licensed under CC BY-SA 3.0 `http://creativecommons.org/licenses/by-sa/3.0/` via Wikimedia Commons - `http://commons.wikimedia.org/wiki/File:KochFlake.svg` 1

[6] The World Wide Web Consortium. *Scalable Vector Graphics (SVG) 1.1 (Second Edition)*. (August 16, 2011), sec. 7.4 `http://www.w3.org/TR/2011/REC-SVG11-20110816/coords.html#EstablishingANewUserSpace` 4.1