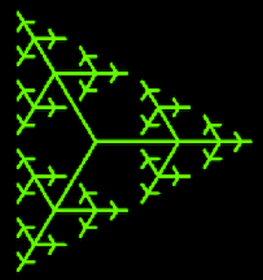
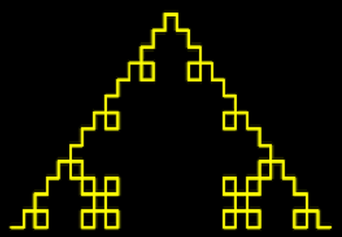
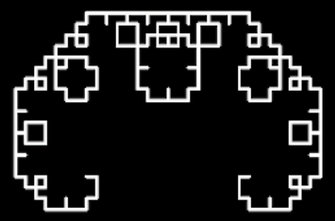
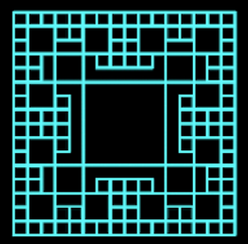
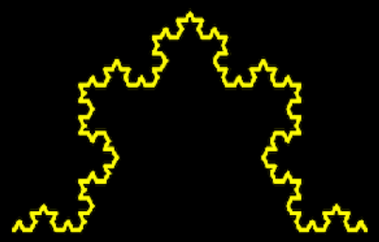
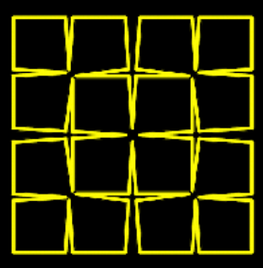
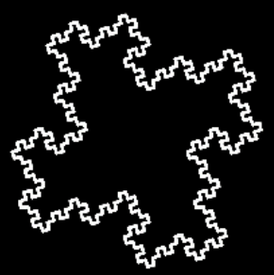
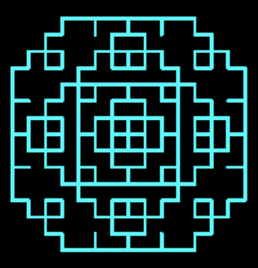
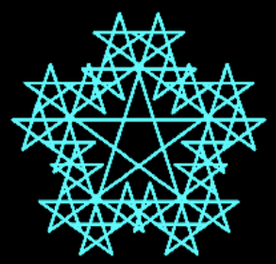
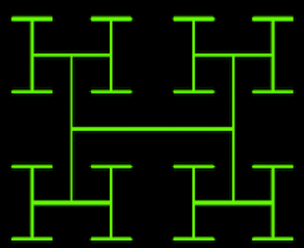
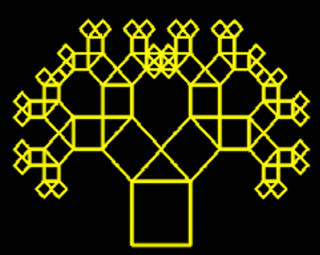
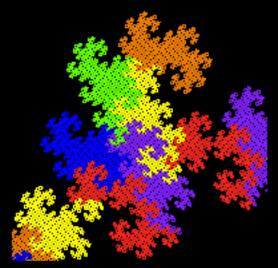
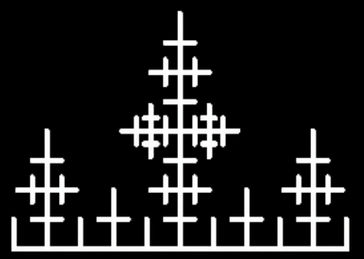
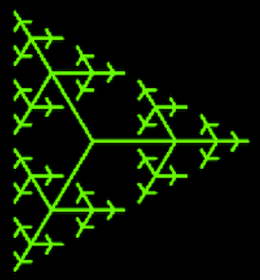
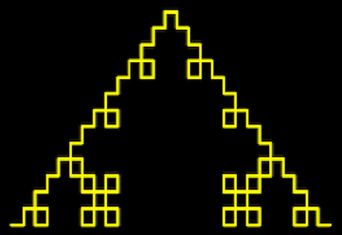
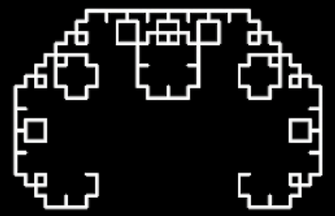
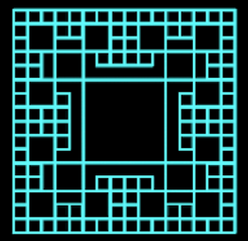
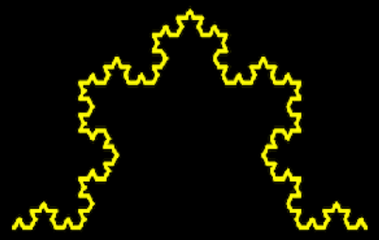






Fractal Picturebook

created by Thomas C Bretl

This book displays a variety of fractals that have been created using the author's own computer programs. Also included is information about how each fractal was created, in as simple terms as possible, without any programming details. Some use mod arithmetic, some use non-base 10 systems, and some use similarity transformations that paste smaller versions of the original pattern onto parts of itself.

The depths listed for each fractal indicate how much detail has been revealed. Theroretically the depth could be unlimited, but screen resolution and memory space make that impossible.

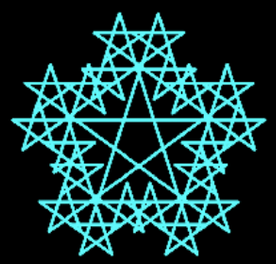
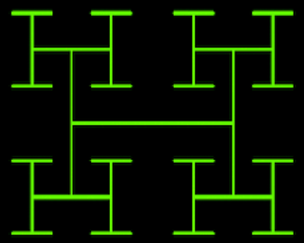
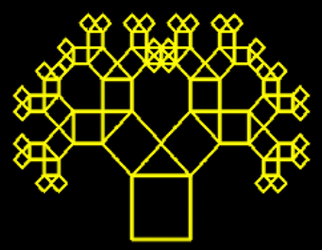
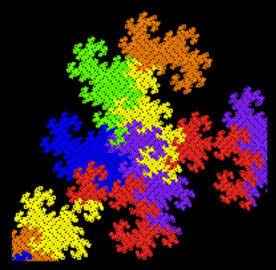
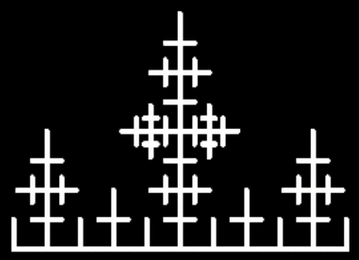
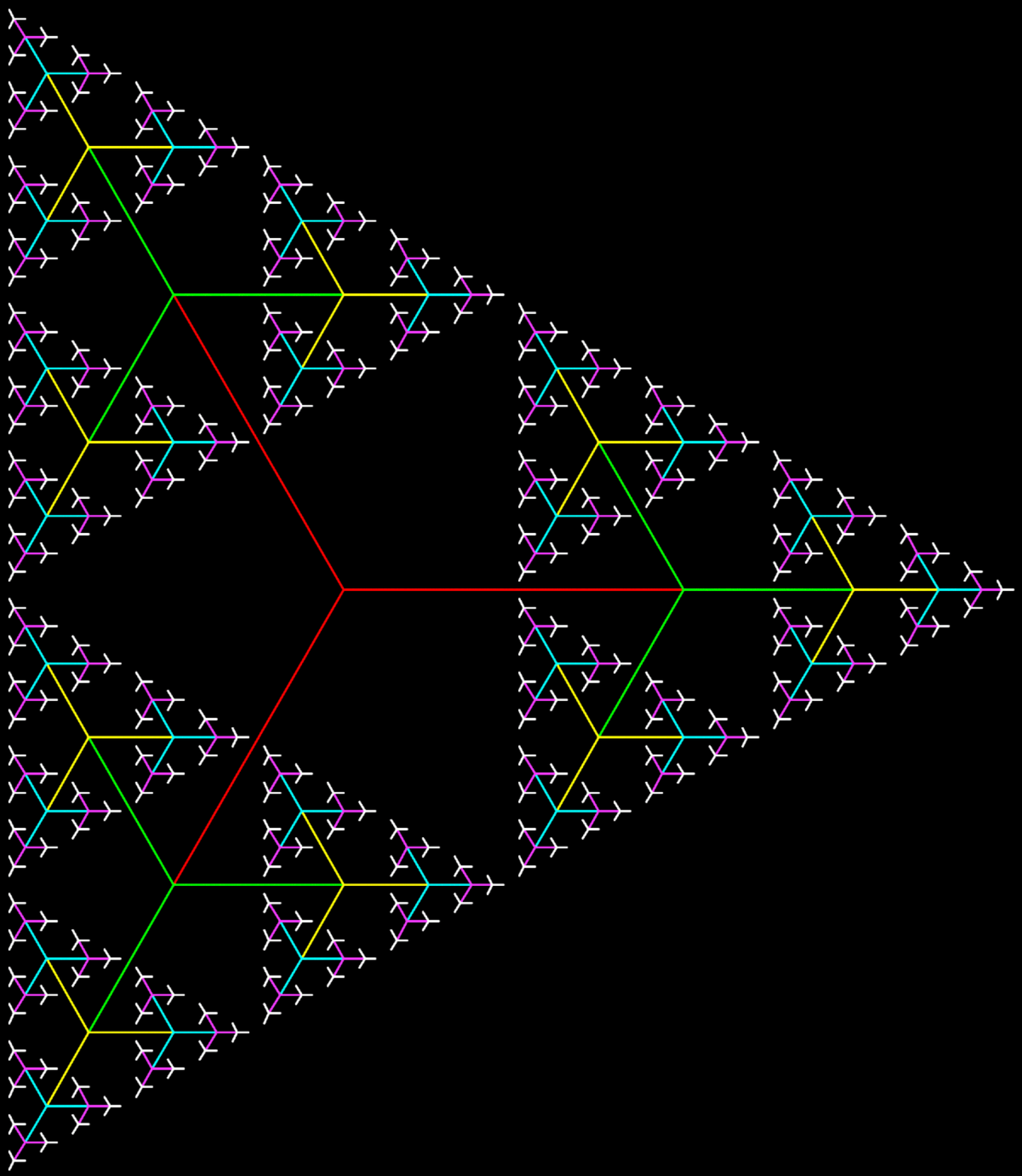
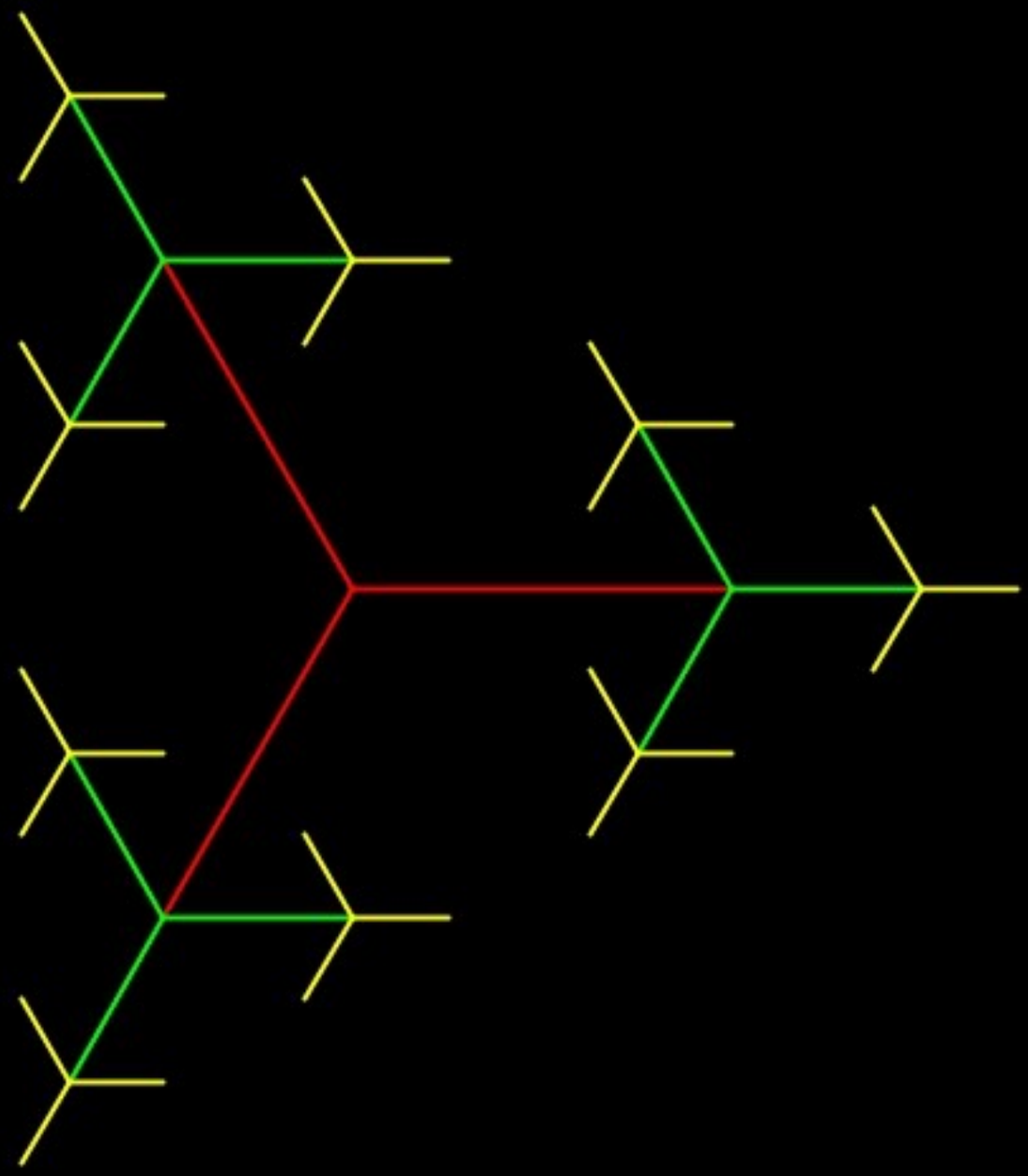






Table of Contents

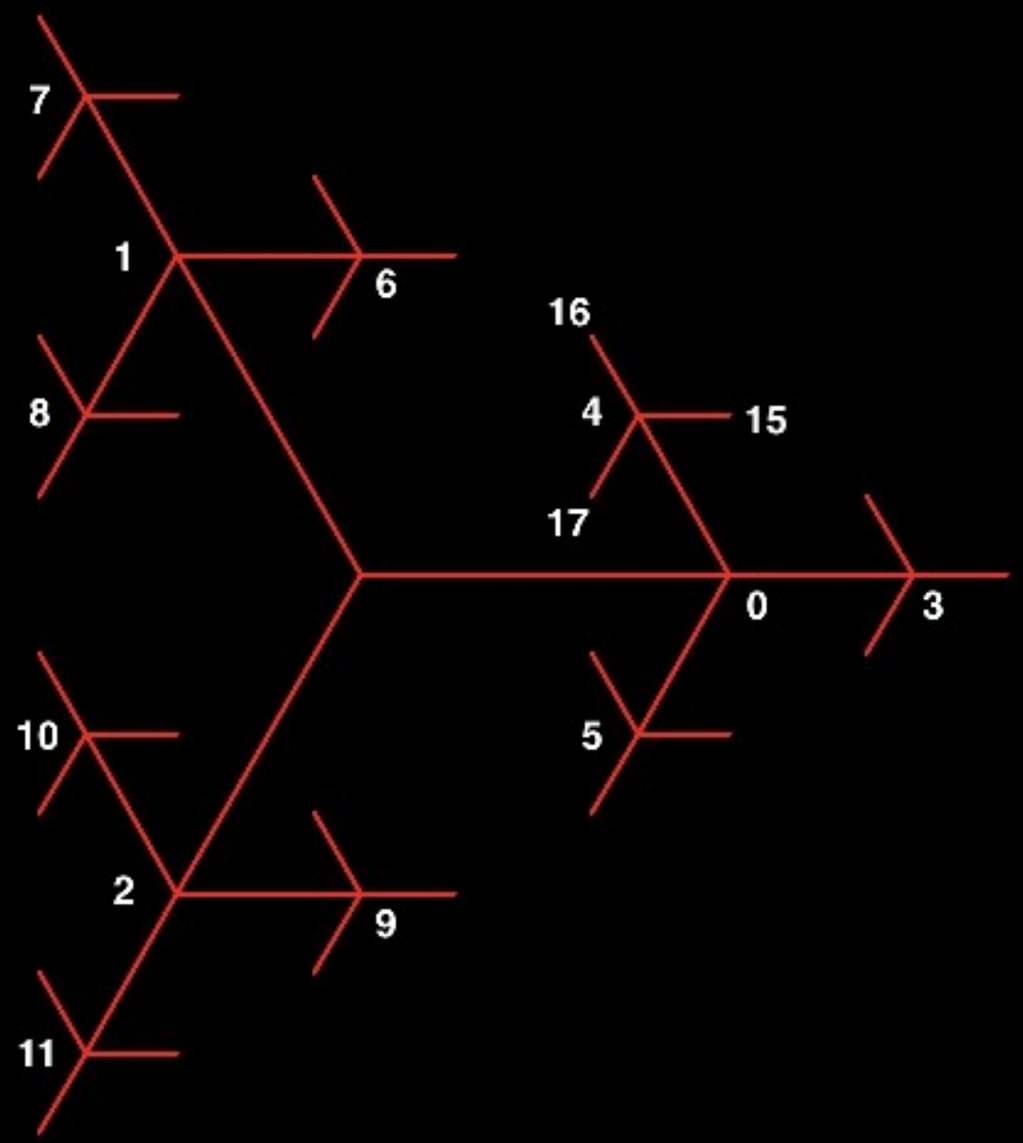
Ternary Fractal with Depth = 5



Ternary Fractal with Depth = 2



How the Ternary Fractal is Created when Depth = 2



The fractal grows from three initial segments of equal length, separated by 120°. Important points are numbered as shown in the diagram. When done in this way, those numbers determine both the location of the point and how it is connected to a previous point.

Points labeled 3, 4, and 5 surround **Point 0** because when you divide those numbers by 3, round down, and subtract 1, you get 0.

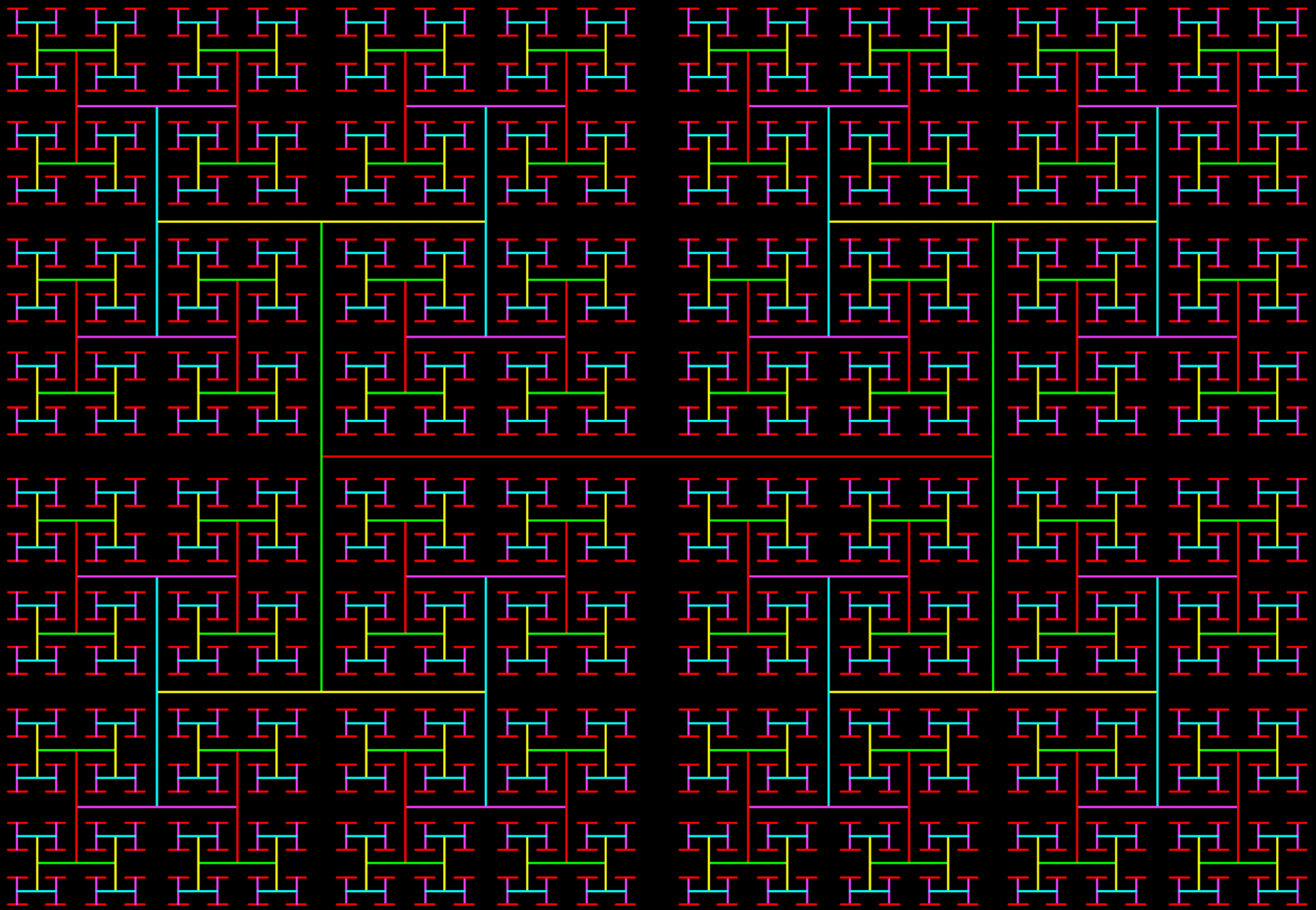
Points labeled 6, 7, and 8 surround **Point 1** because when you divide those numbers by 3, round down, and subtract 1, you get 1.

Points labeled 9. 10, and 11 surround **Point 2** because when you divide those numbers by 3, round down, and subtract 1, you get 2.

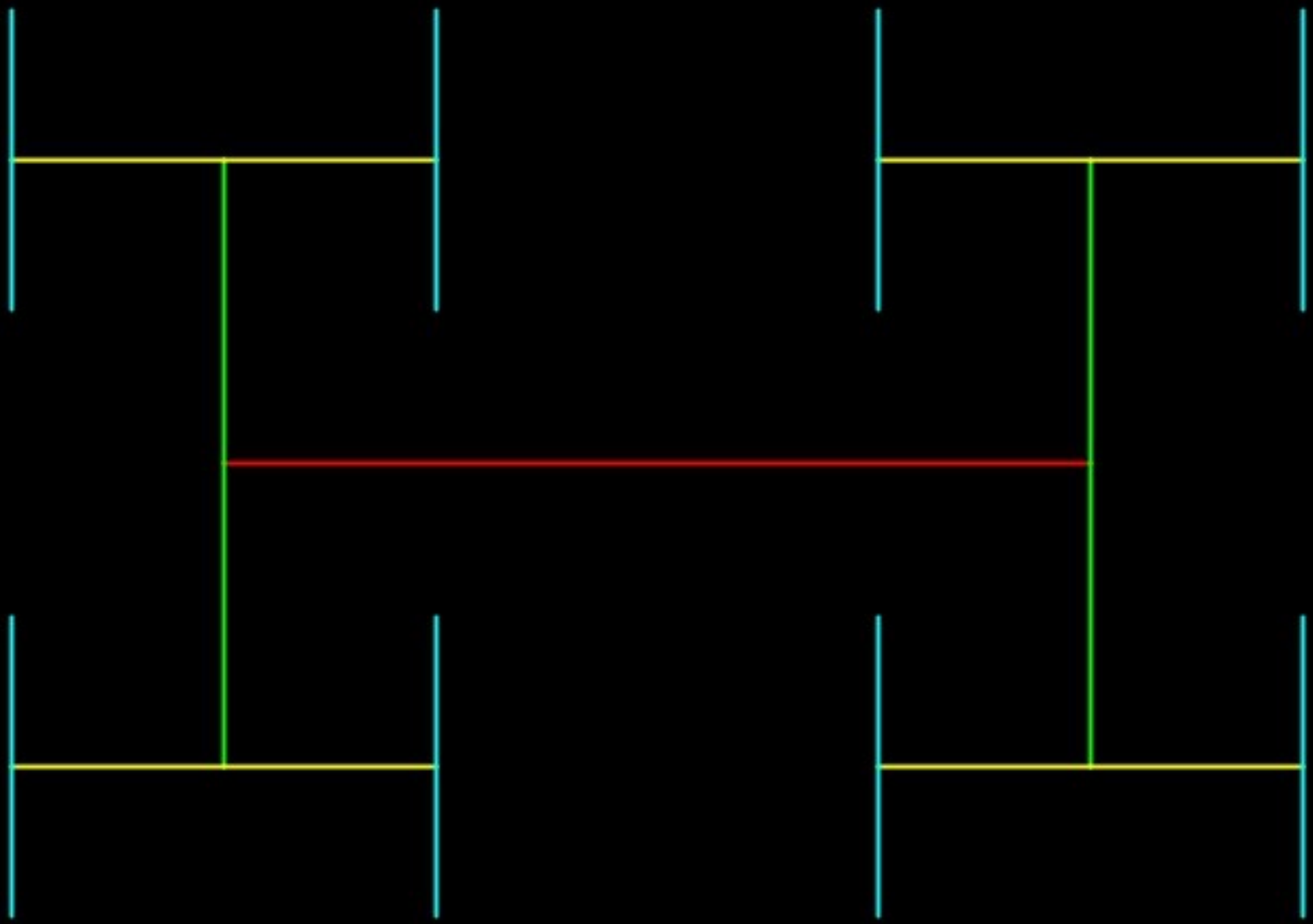
Mod arithmetic tells you at what angle you must go to get from the surrounded point to the surrounding points. Points with numbers which equal **0 mod 3** are located directly to the right (0°), points with numbers which equal **1 mod 3**, are located at an angle of 120°, and points with numbers which equal **2 mod 3** are located at an angle of -120°. The length of the segments equals 1/2 of the length of the original three segments.

Note that the same method works when you increase the depth: 15, 16, and 17 surround point 4, for example. Divide those numbers by 3, round down, and subtract 1. In each case, you get 4.

H Fractal with Depth = 10



H Fractal with Depth = 3



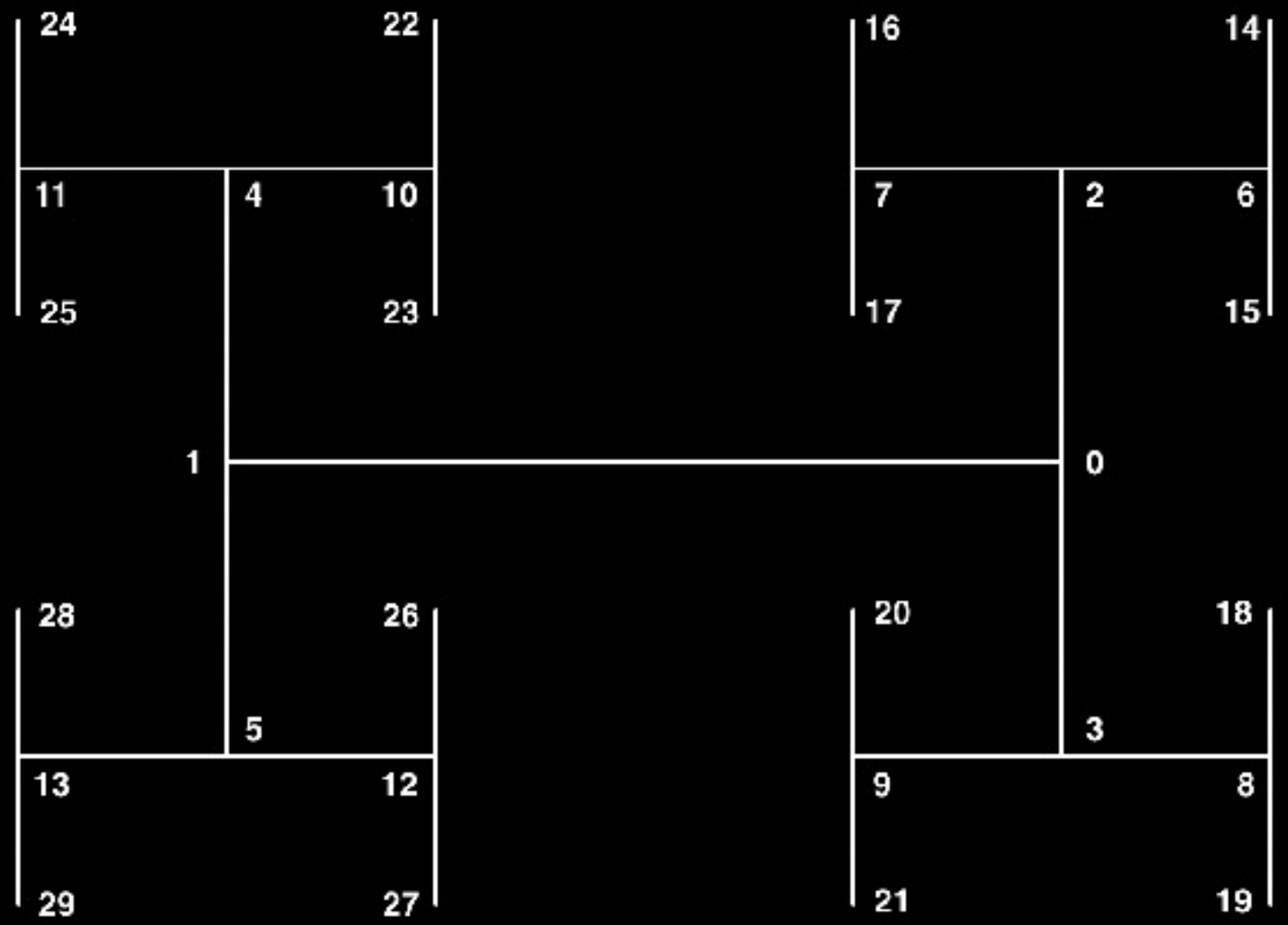
How it the H Fractal is Created when Depth = 3

Segments are added to this fractal by numbering the relevant points as shown here and connecting each of them to a previously defined point,

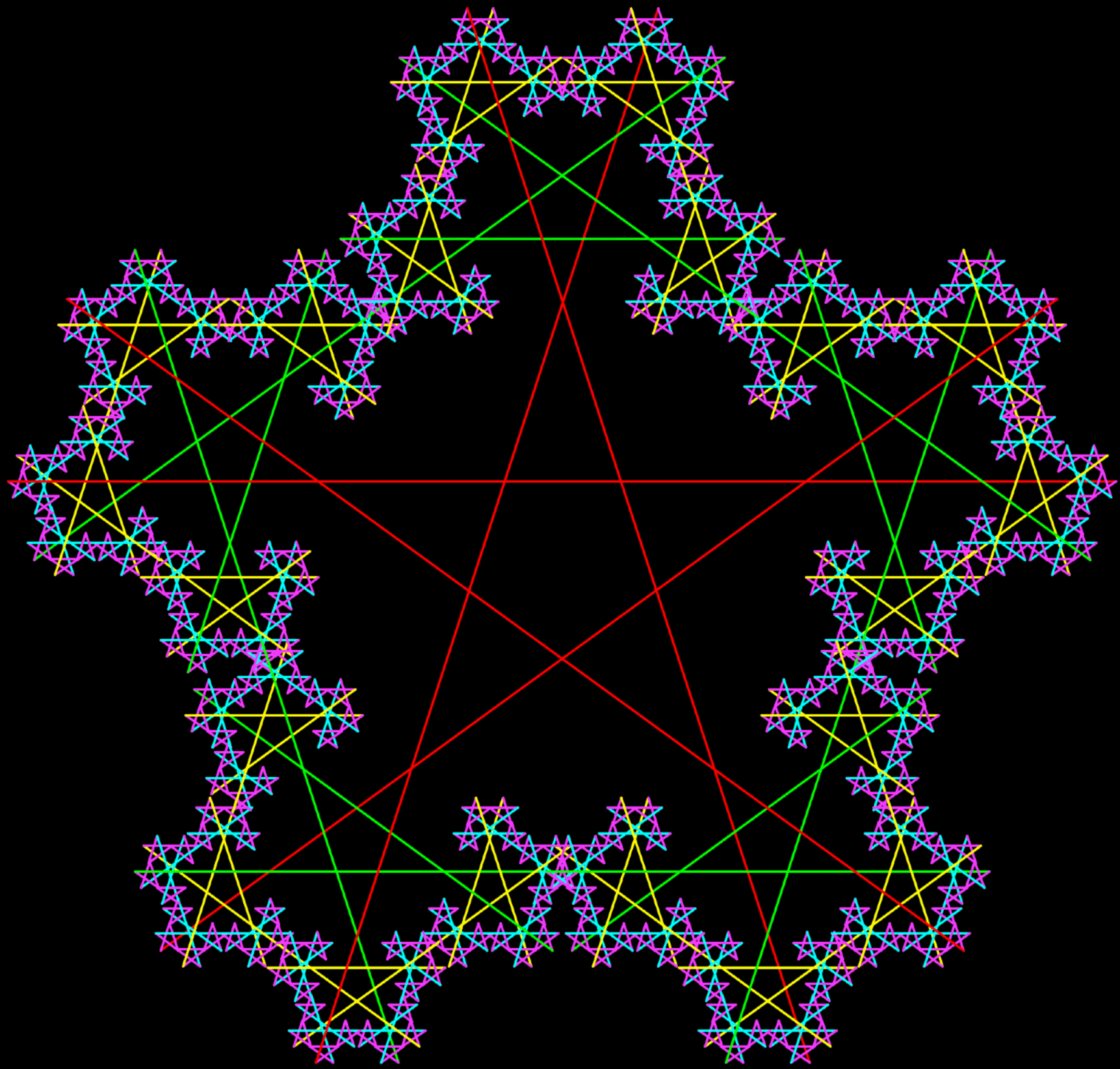
To determine where **Point N** belongs, divide N by 2, round down, and subtract 1. That gives you the number of the point to which **Point N** should be connected. If N mod 2 = 0 then **Point N** is either above or to the right of the previous point; if N mod 2 = 1, then **Point N** is below or to the left of the previous point.

Let D equal the depth of the points being added (in this case 3). If D mod 2 = 0, then the new **Point N** will be either to the left or the right. If D mod 2 = 1, then **Point N** will be above or below.

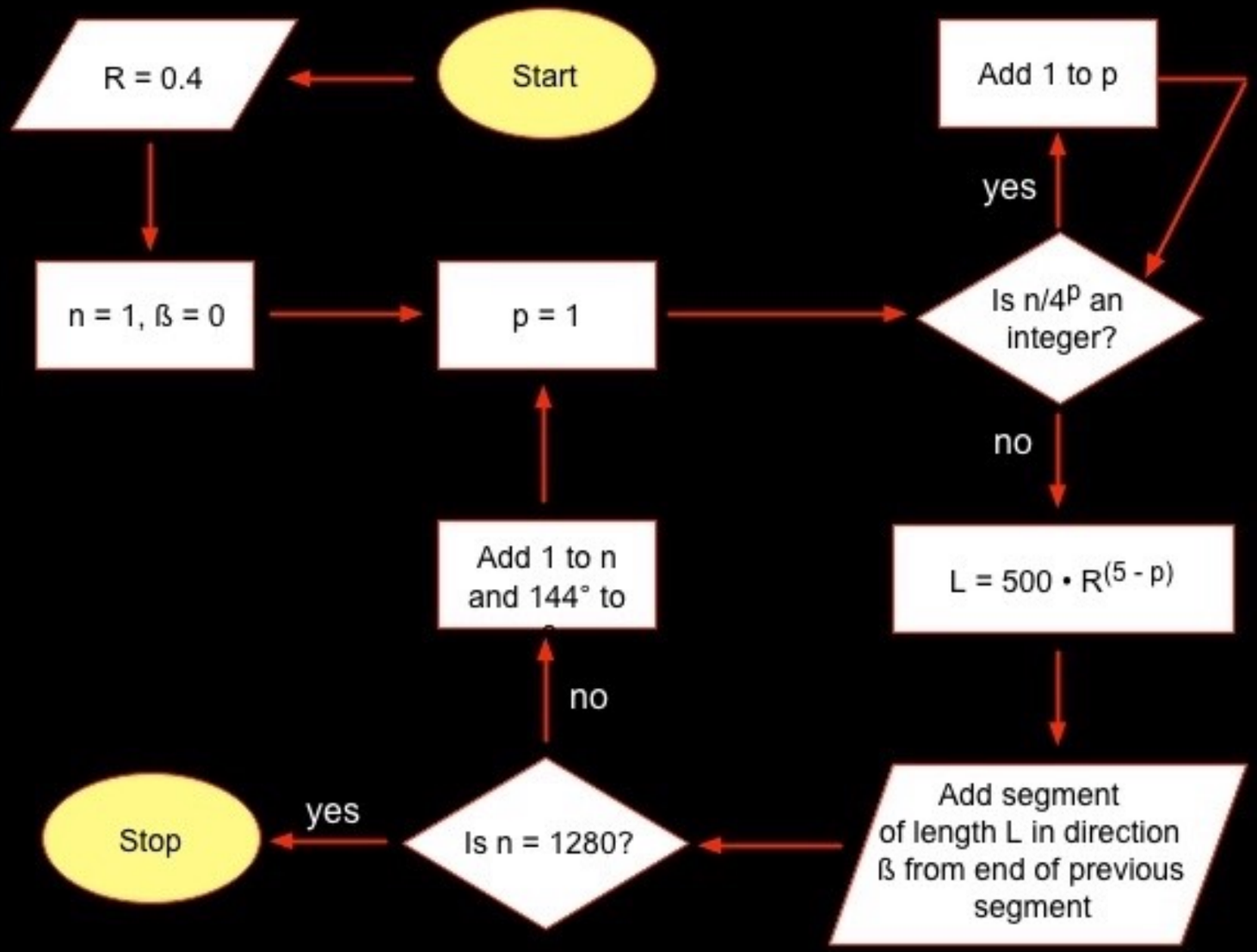
Example: 23/2 rounds down to 11. 11-1 = 10, so **Point 23** is connected to **Point 10.** 23 mod 2 = 1, so **Point 23** is either below or to the left of **Point 10**. Finally, D mod 2 = 1, so **Point 23** is below instead of the the left.



Star Fractal with 5 Segment Lengths and R = 0.40



Flowchart Showing how the Star Fractal is Created



R is the ratio between two successive segment lengths.

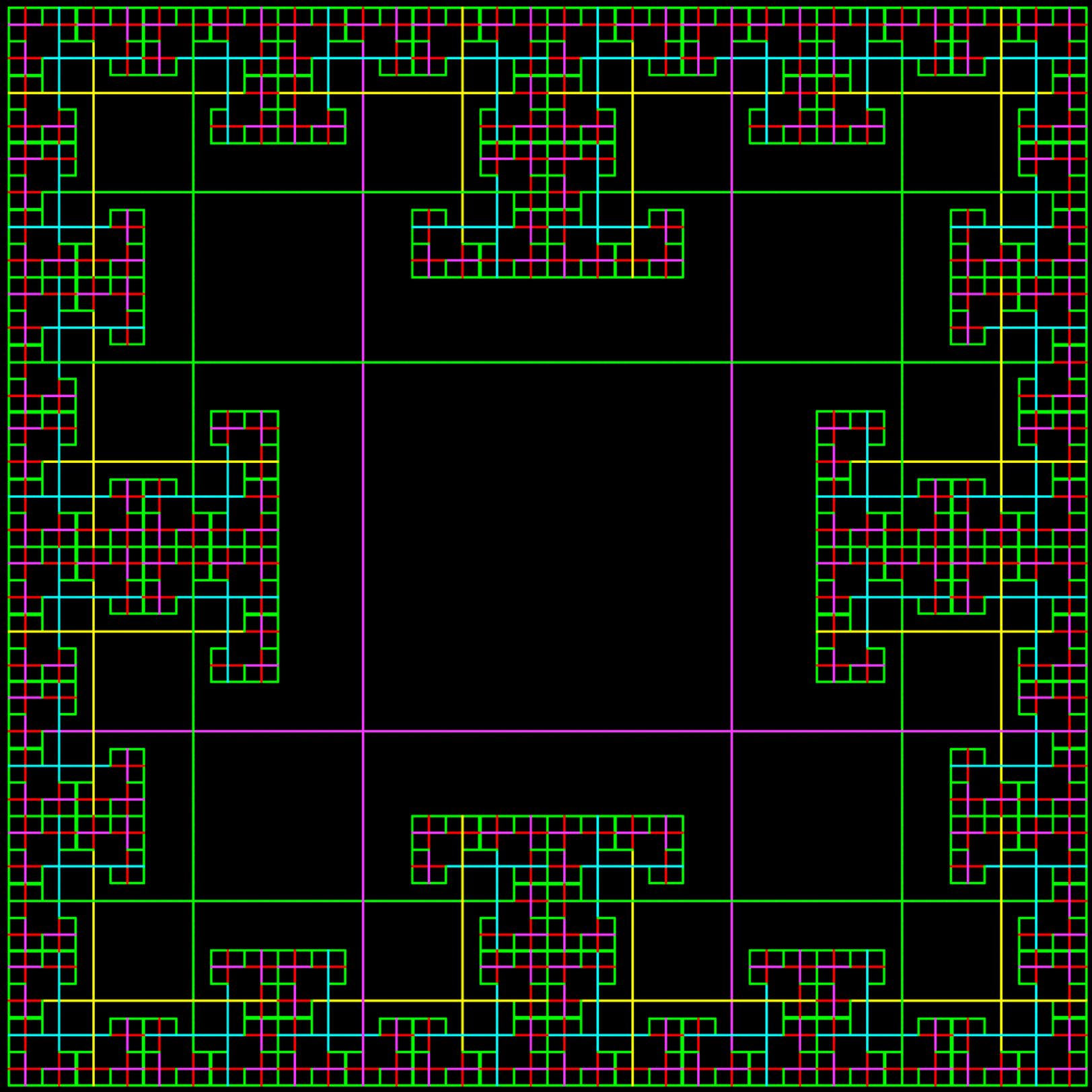
n is the number given to each seqment.

ß is the angle measured clockwise from the positive x-axis. It increases by 144° each time a segment is added to the pattern.

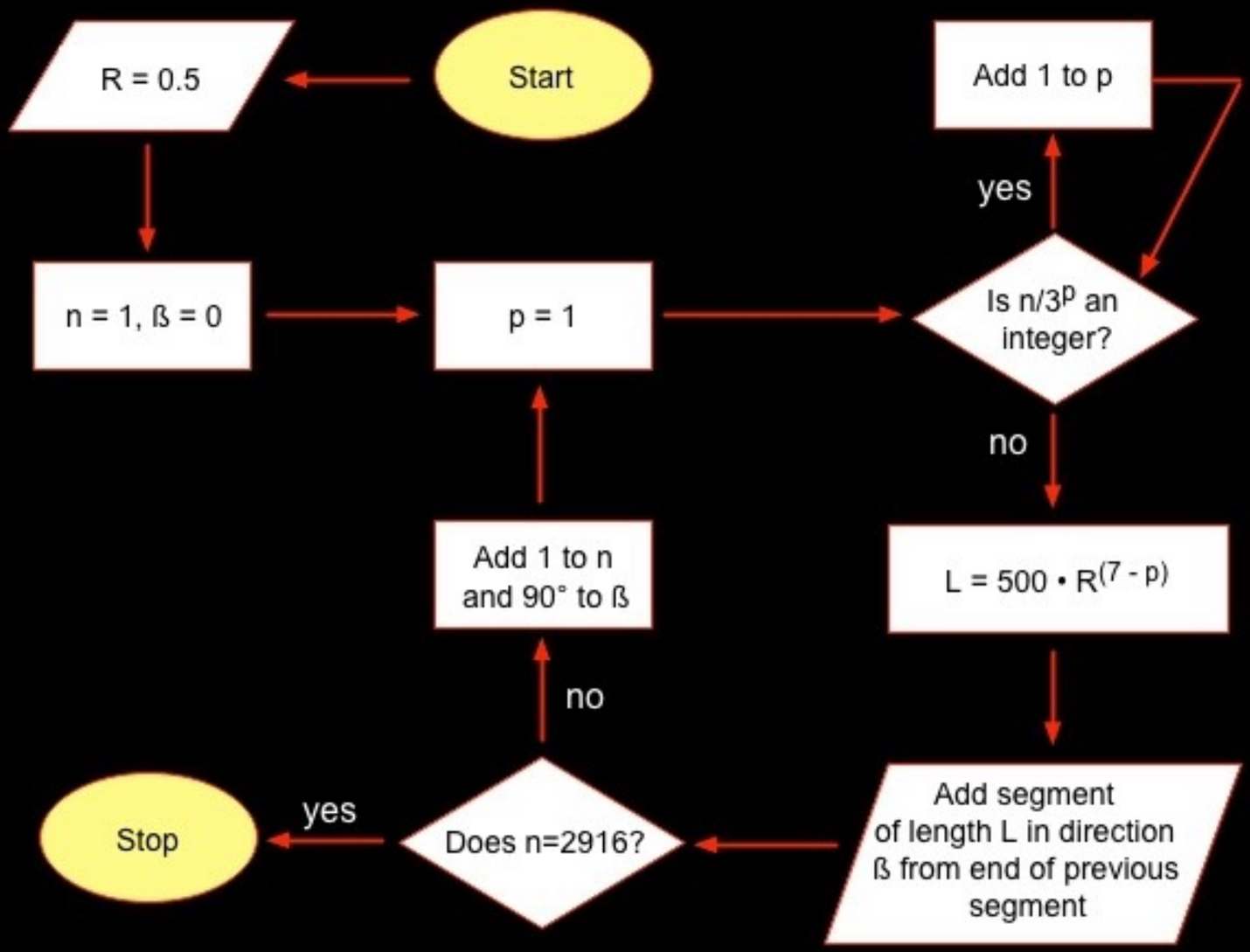
p increases until n is no longer evenly divisible by 4^p.

L is the segment length. The longest segments are 500; the shortest segments are 500 • R^4.

Square Fractal with 7 Segment Lengths and R = 0.5



Flowchart Showing how the Square Fractal is Created



This flowchart is almost the same as for the Star Fractal.

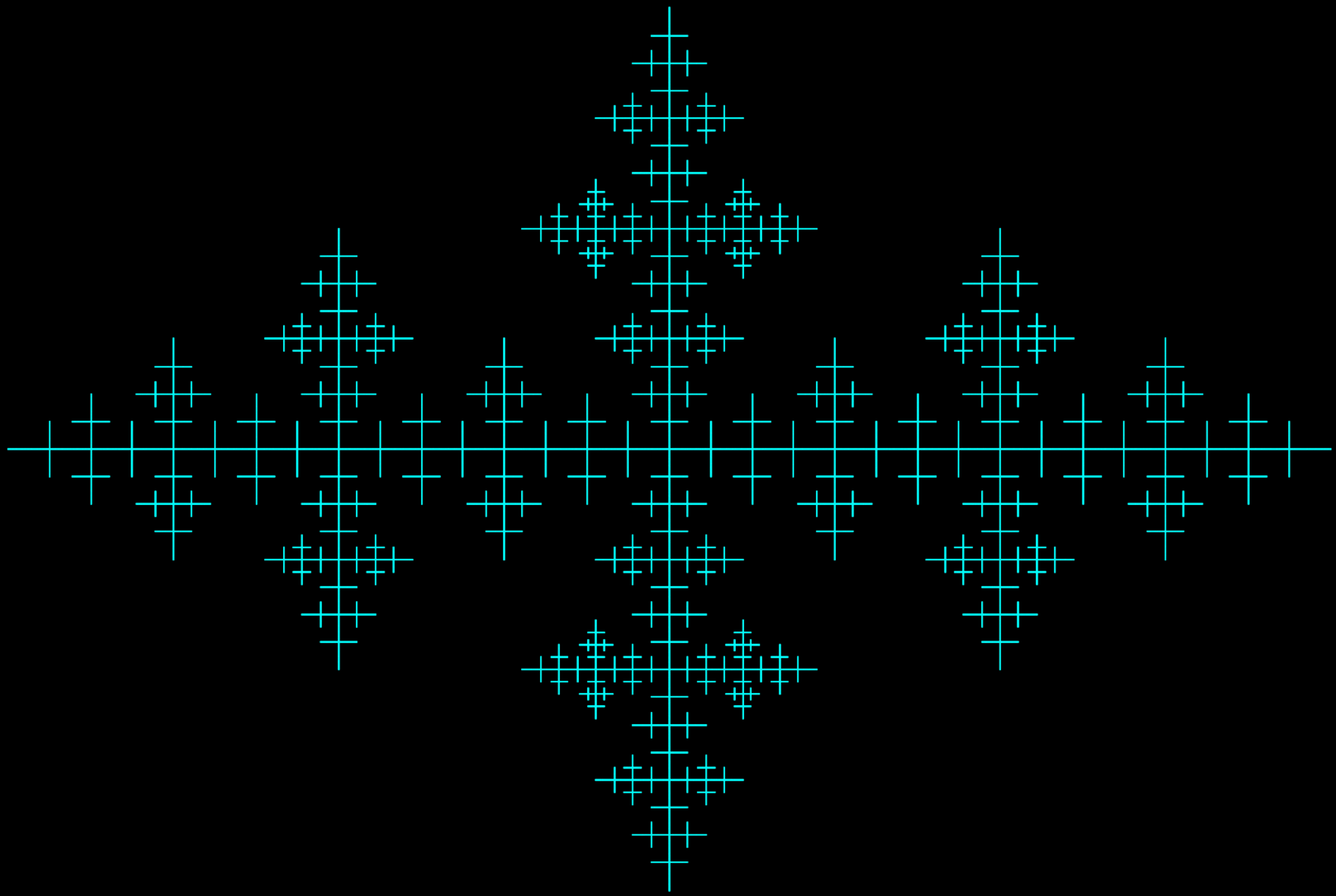
R is the ratio between two successive segment lengths, and n is the number given to each seqment.

ß is the angle measured clockwise from the positive x-axis, but it increases by 90° each time a segment is added to the pattern.

p increases until n is no longer evenly divisible by 3^p instead of 4^p

L is the segment length. The longest segments are 500; the shortest segments are $500 \cdot R^6$.

Two attached Ice Fractals with Depth = 5



Ice Fractal created using Similarity Transformations

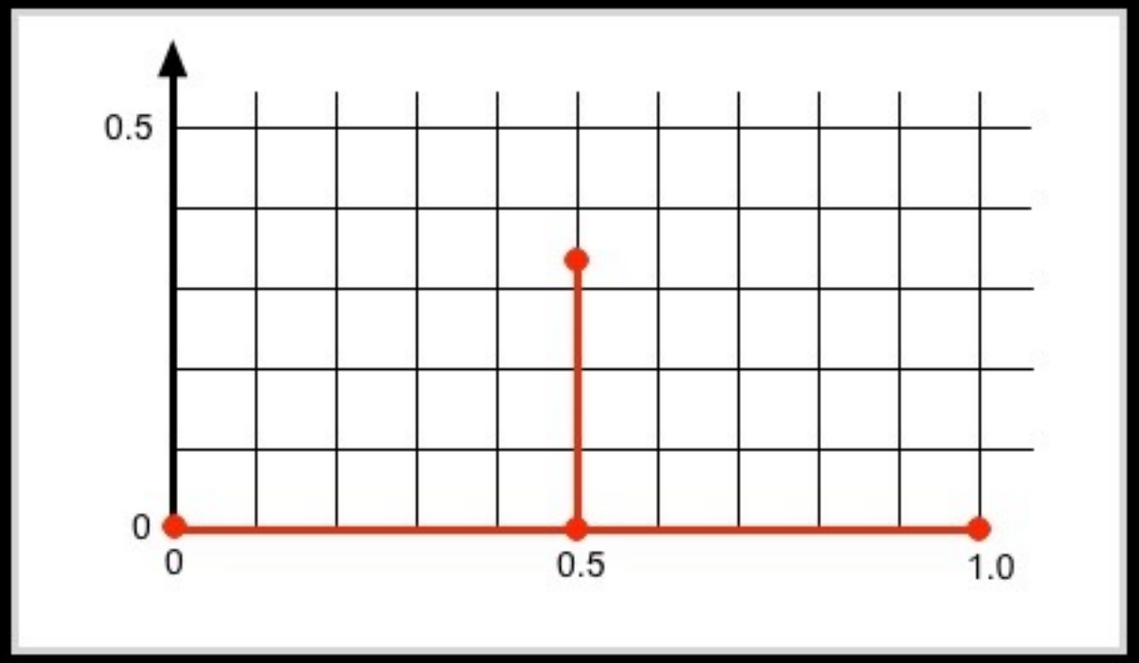
To construct the ice crystal fractal you start with the basic pattern shown here in red and make similarlity transformations of it onto its individual directed segment parts that go from (0,0) to (1/2,0), (1/2,0) to (1/2,1/3), (1/2,1/3) back to (1/2,0), and (1/2,0) to (1,0). The general formulas for doing this are:

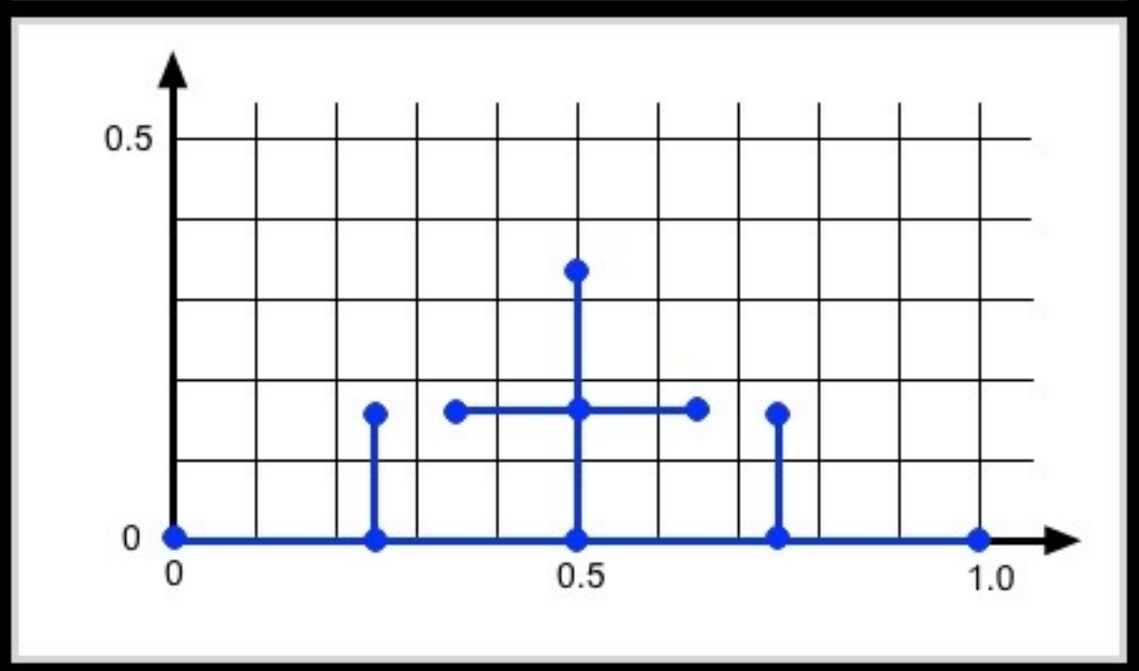
$x' = (x2 - x1)\cdot x - (y2-y1)\cdot y + x1$
$y' = (y2 - y1)\cdot x + (x2-x1)\cdot y + y1$

whiere (x1,y1) and (x2,y2) are the endpoints of the individual segments, (x,y) is any point on the basic pattern, and (x',y') is the transformed location of that point.

The transformation creates 16 shorter directed segments shown here in blue connecting (0,0) to (1/4,0), (1/4,0) to (1/4,1/6), (1/4,1/6) back to (1/4,0), (1/4,0) to (1/2,0), (1/2,0) to (1/2,1/6), (1/2,1/6) to (1/3,1/6), (1/3,1/6) back to (1/2,1/6), (1/2,1/6) to (1/2,1/3), (1/2,1/3) back to (1/2,1/6), (1/2,1/6) to (2/3,1/6), (2/3,1/6) back to (1/2,1/6), (1/2,1/6) back to (1/2,0), (1/2,0) to (3/4,0), (3/4,0) to (3/4,1/6), (3/4/1/6) back to (3/4.0), (3/4,0) to (1,0).

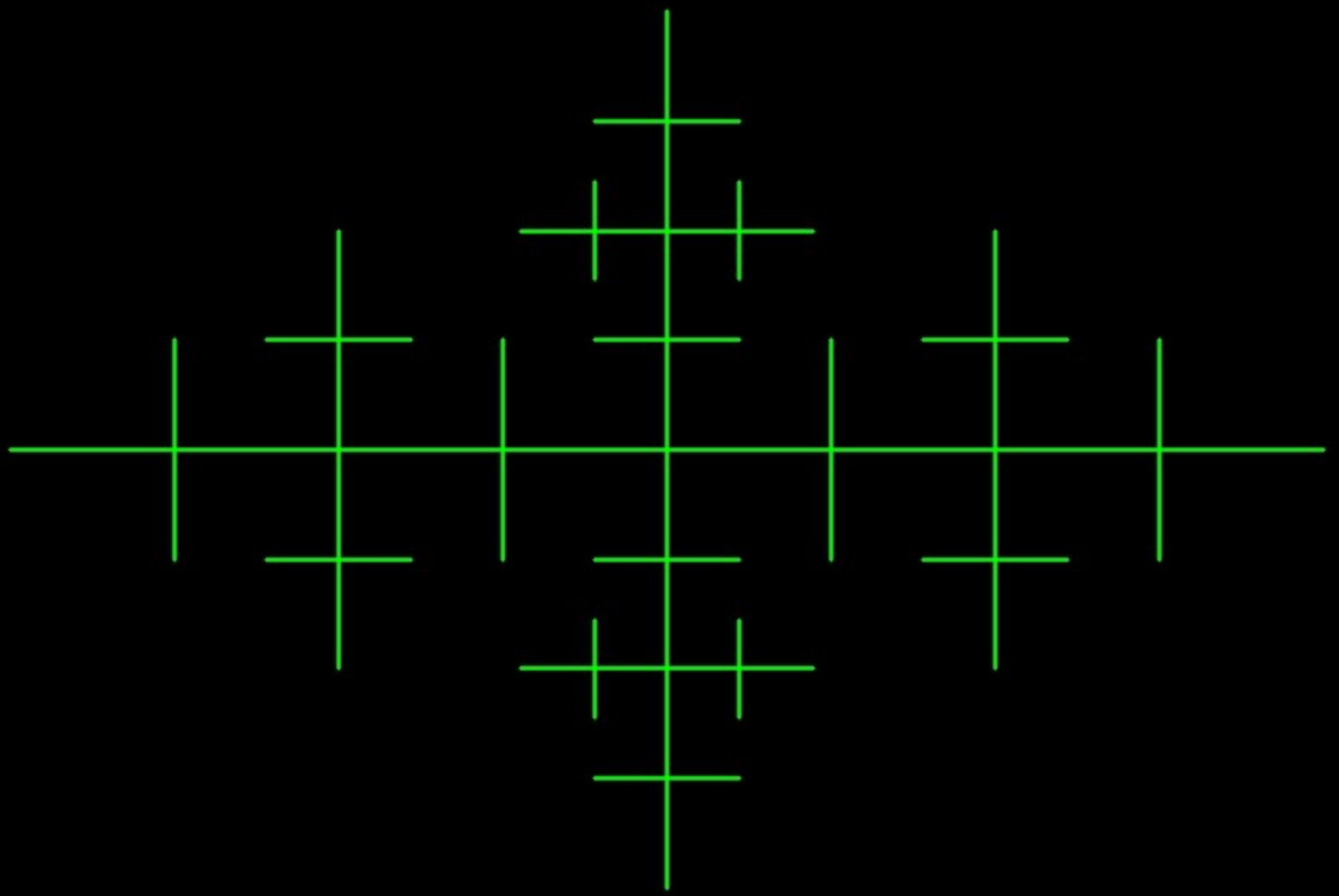
The same transformation can then applied to those shorter segments to go deeper into the fractal.



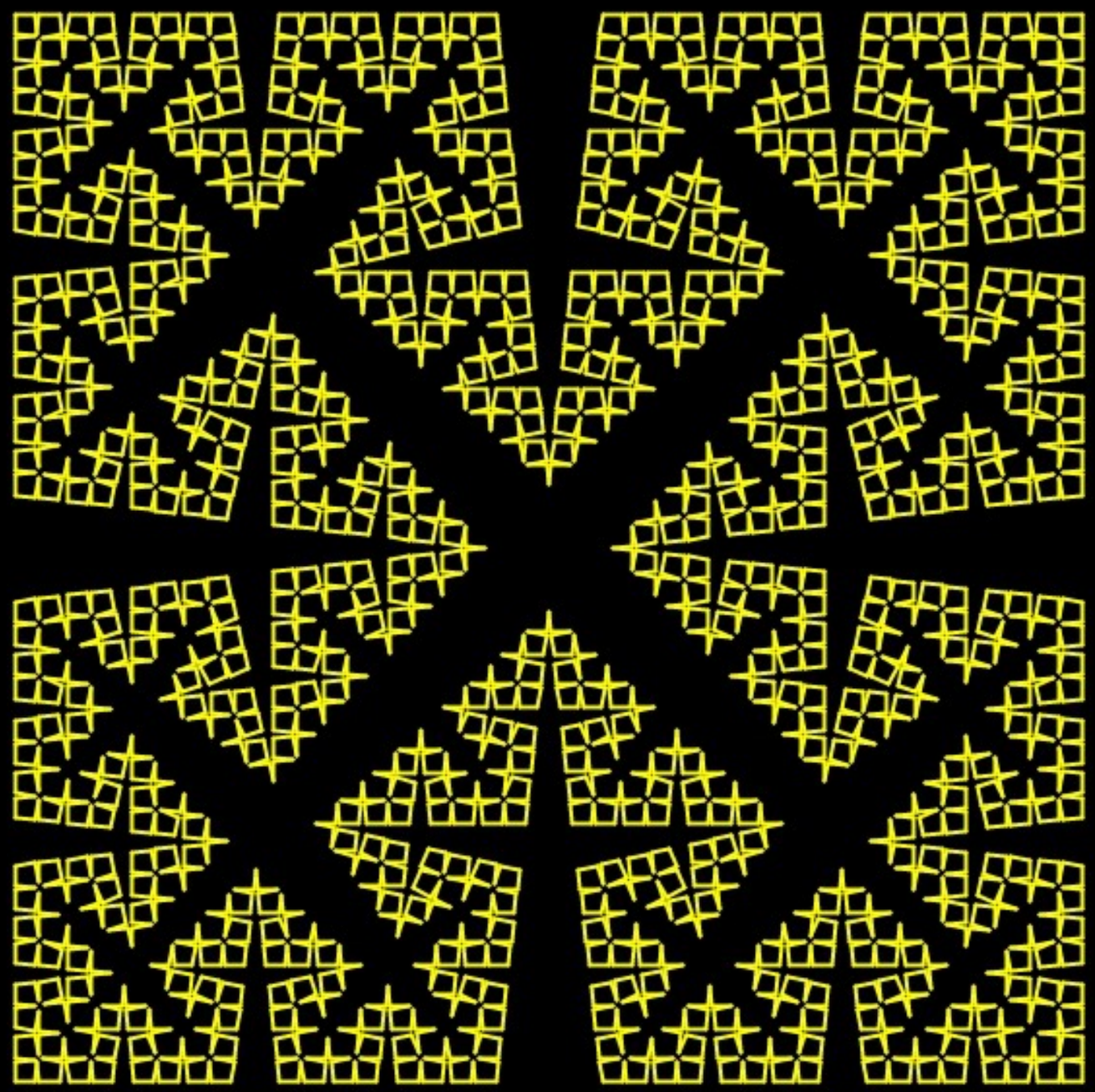


See next page to go one step deeper.

Two attached Ice Fractals with Depth = 3

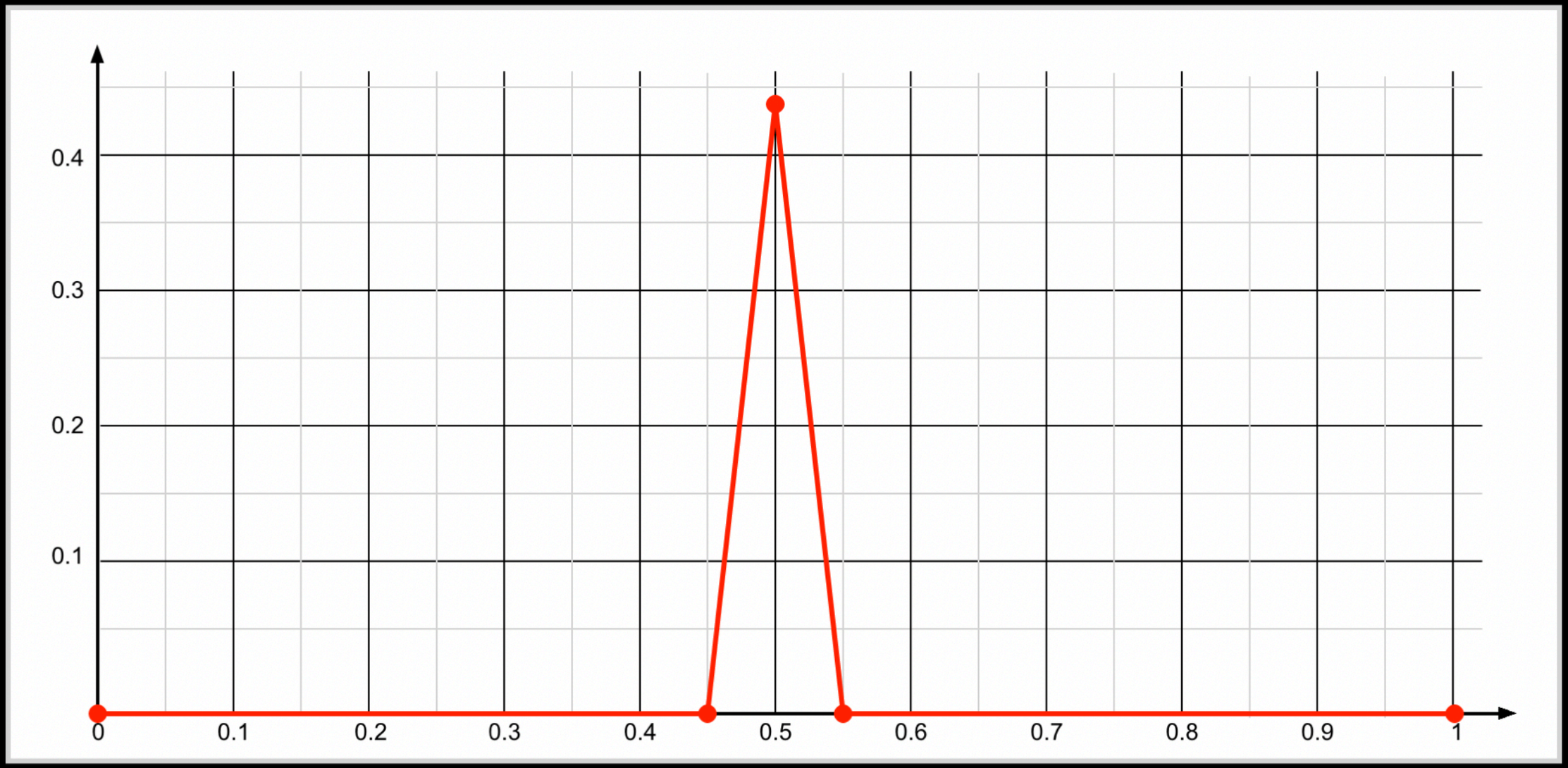


The Torn Square Fractal with Depth = 5

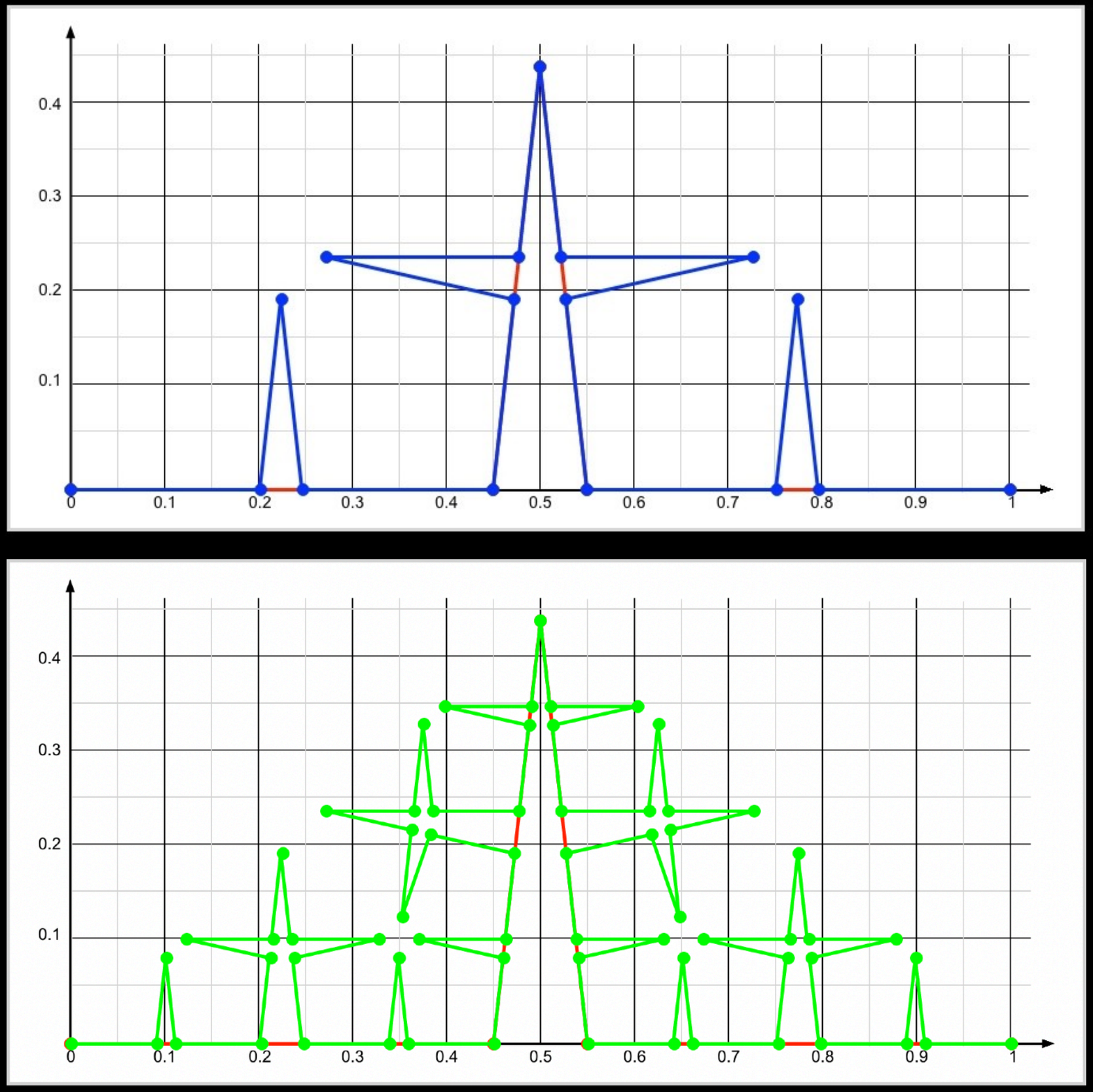


Torn Square Fractal created using Similarity Transformations

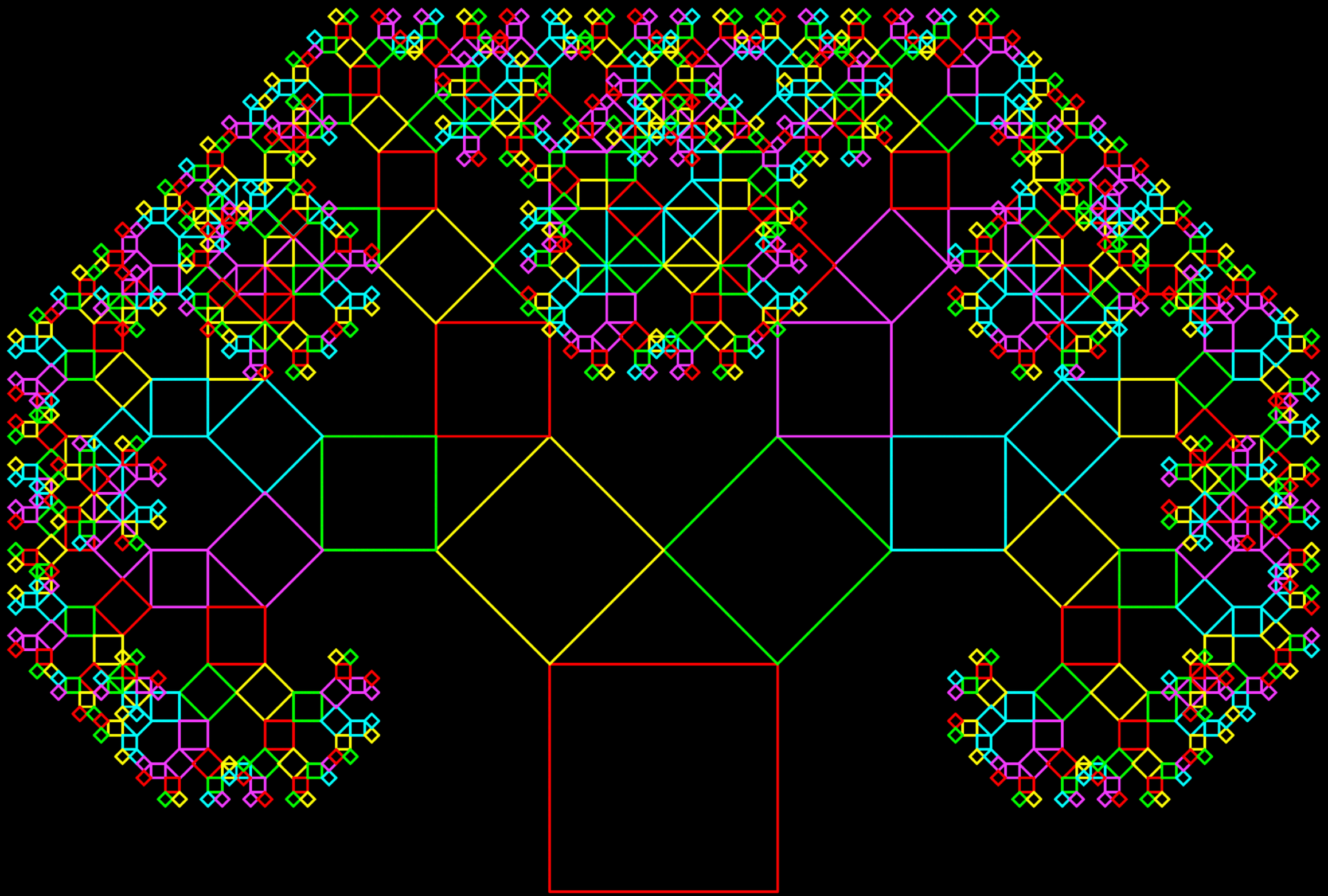
To create the Torn Square Fractal the red pattern shown below is transformed similarly onto each of the four segments that make it up: (0,0) to (.45,0), (.45,0) to (.5,.44), (.5,.44) to (.55,0), and (.55,0) to (1,0). That produces four smaller copies of the original pattern consisting of sixteen segments (shown in blue on the next page). Then the original pattern is transformed onto the blue segments. That produces sixteen copies of the original pattern consisting of 64 segments (shown in green on the next page). The next page illustrates just these first two steps, but the process can be continued indefinitely. The results are replicated and attached at right angles to each other to reveal the torn square effect.



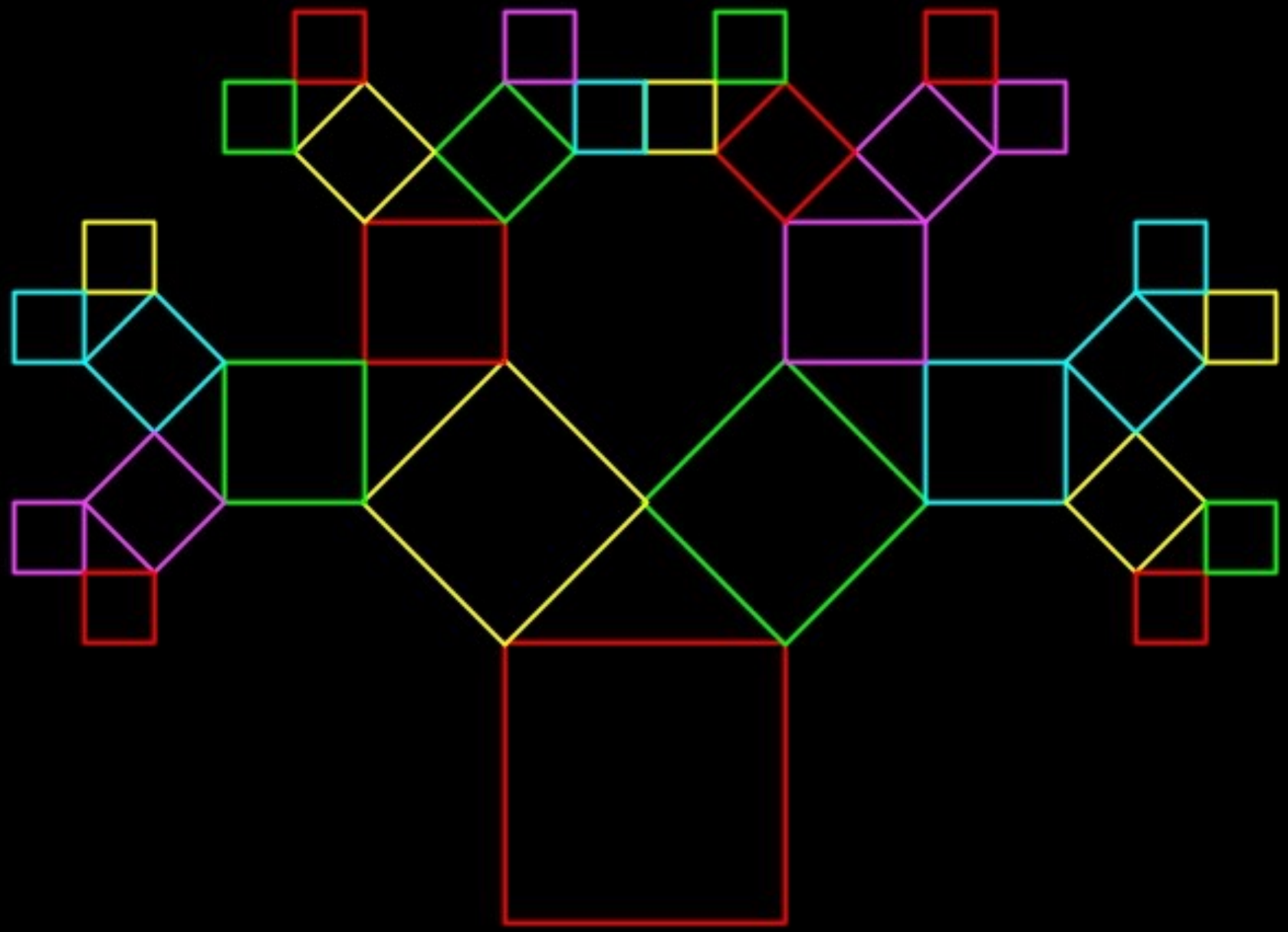
First two Similarity Transformations



Tree Fractal with Depth = 9



Tree Fractal with Depth = 4



How it was Created with Depth = 4

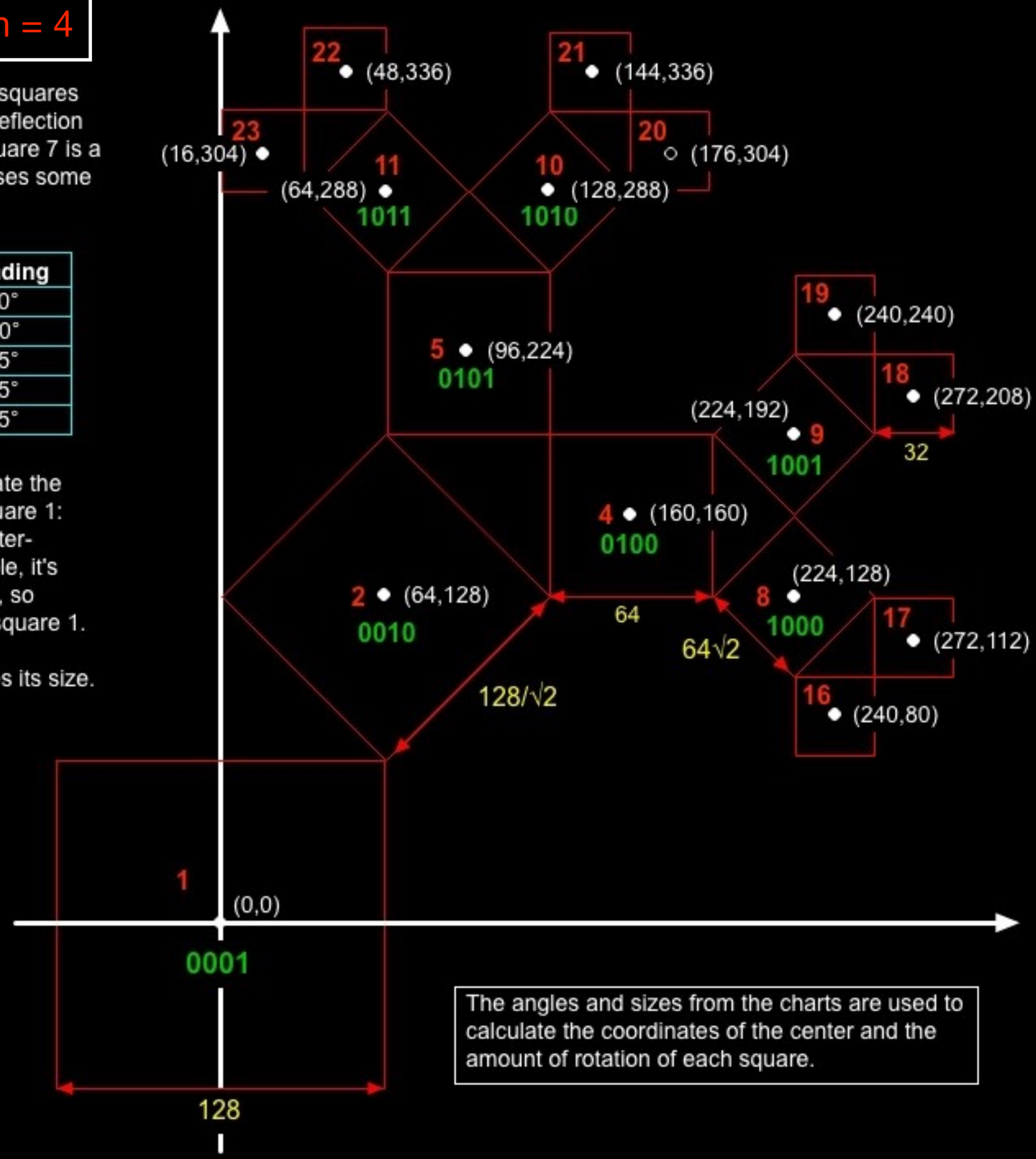
The squares are numbered as shown. The omitted squares are their reflections over the y-axis. (Square 3 is a reflection of square 2, square 6 is a reflection of square 5, square 7 is a reflection of square 4, etc.) The chart below expresses some of those numbers in binary form.

	Binary	Heading		Binary	Heading
1	0001	0°	6	0110	0°
2	0010	45°	7	0111	- 90°
3	0011	- 45°	8	1000	135°
4	0100	90°	9	1001	45°
5	0101	0°	10	1010	45°

The binary digits to the right of the left-most 1 indicate the rotations involved as you go to that square from square 1: 0 means rotate clockwise 45°; 1 means rotate counter-clockwise 45°. To get to square 9 (1001), for example, it's clockwise 45°, clockwise 45°, counterclockwise 45°, so square 9 ends up rotated 45° clockwise relative to square 1.

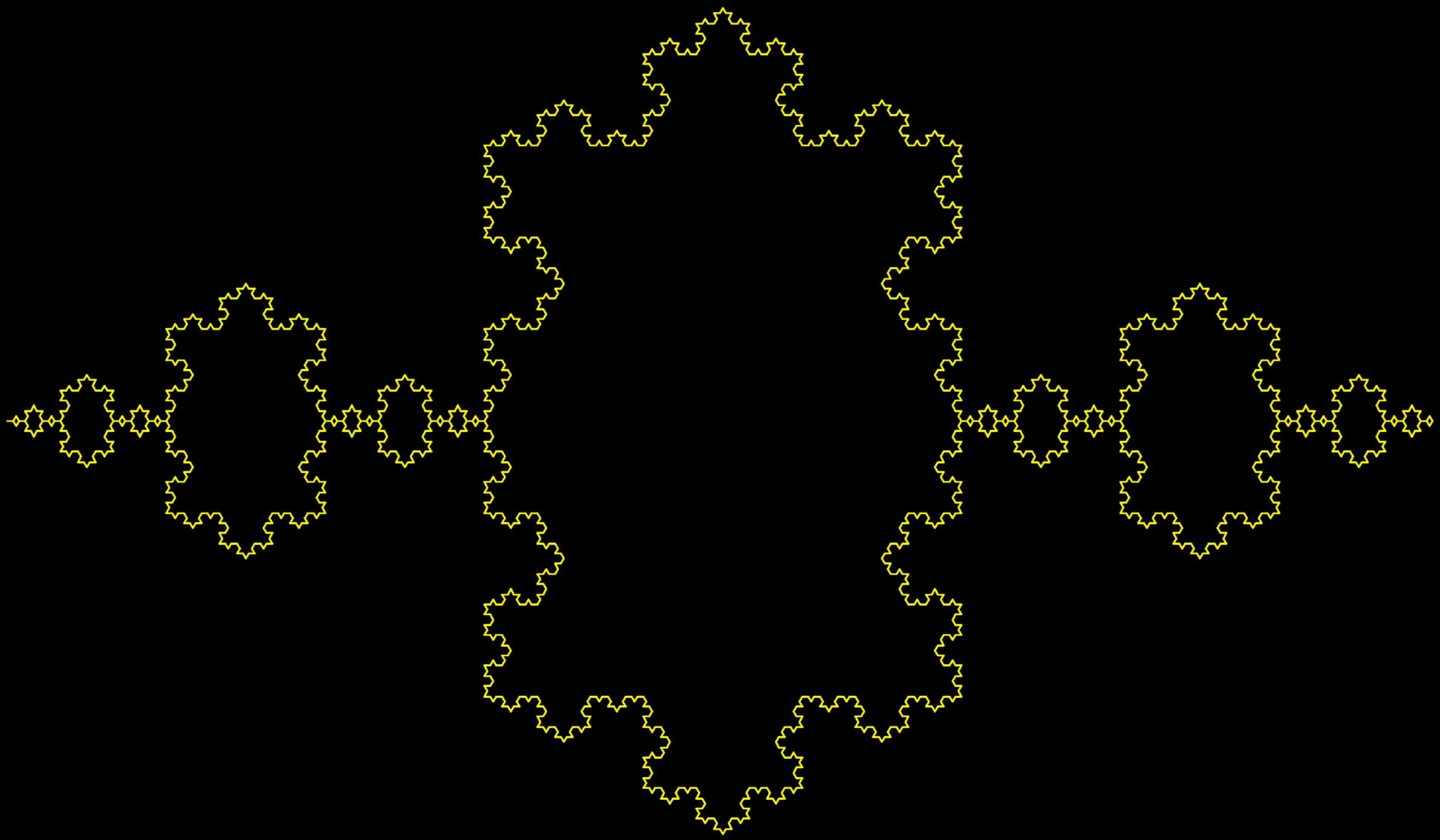
The digits of a square's binary number also indicates its size. Look for the left-most digit which is a 1.

	Binary	Digit	Size
1	0001	1's	128
2	0010	2's	128/√2
3	0011	2's	128/√2
4	0100	4's	64
5	0101	4's	64
6	0110	4's	64
7	0111	4's	64
8	1000	8's	64/√2
9	1001	8's	64/√2
10	1010	8's	64/√2

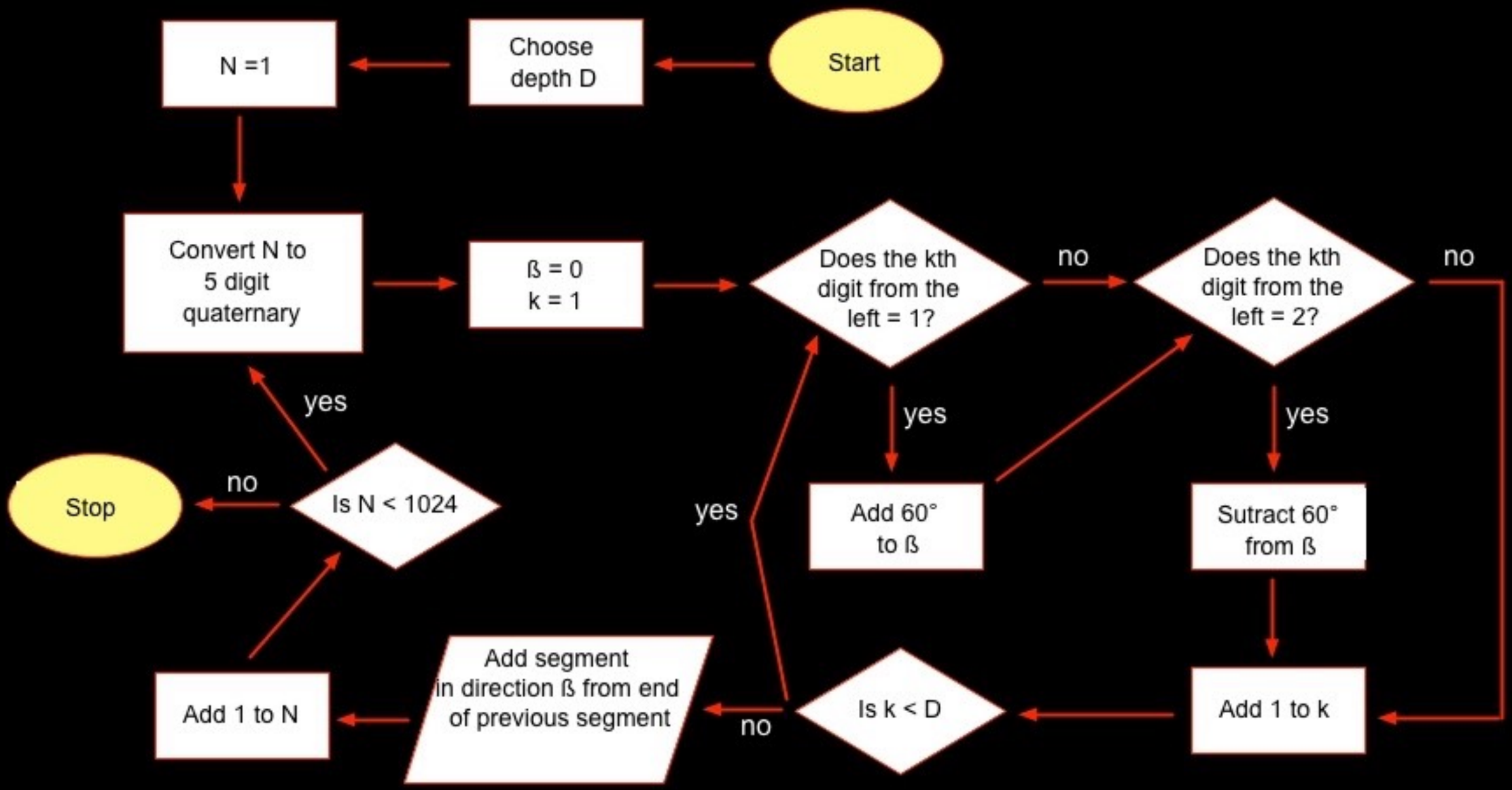


The angles and sizes from the charts are used to calculate the coordinates of the center and the amount of rotation of each square.

Two Koch Fractals with Depth = 5

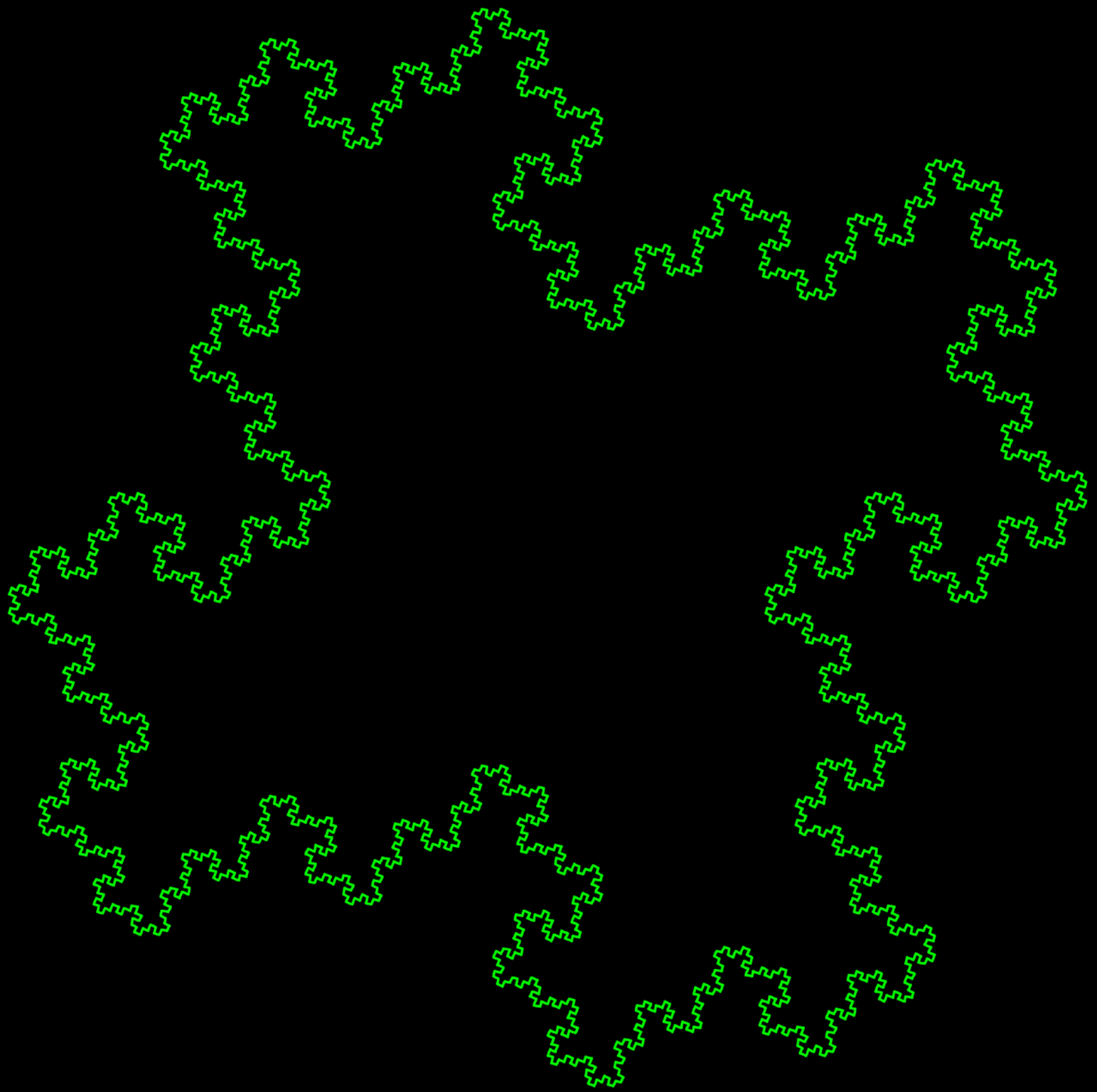


Flowchart Showing how the Koch Fractal is Created



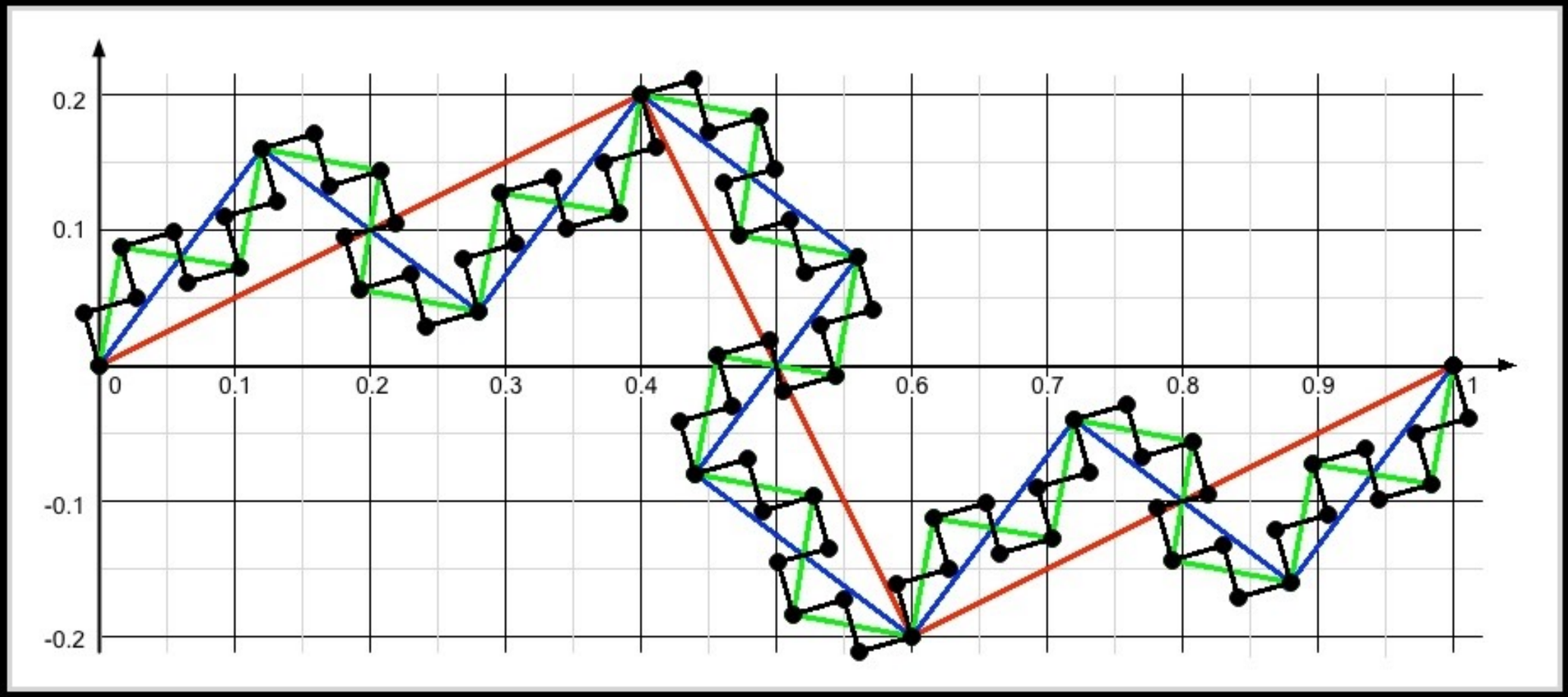
1024 very short segments are combined to form this meandering fractal. The greater the depth D, the more changes in direction are inserted between those segments. The location of the turns are determined by the segment number N after it has been converted to the quaternary (base 4) number system. The first D digits from the left are examined. If a digit = 1, then ß is increased by 60°; if a digit = 2, then ß is decreased by 60°. The value of ß determines the diirection of the added segment. The angles are measured counter-clockwise from the positive x-axis.

Island Fractal with Depth = 6

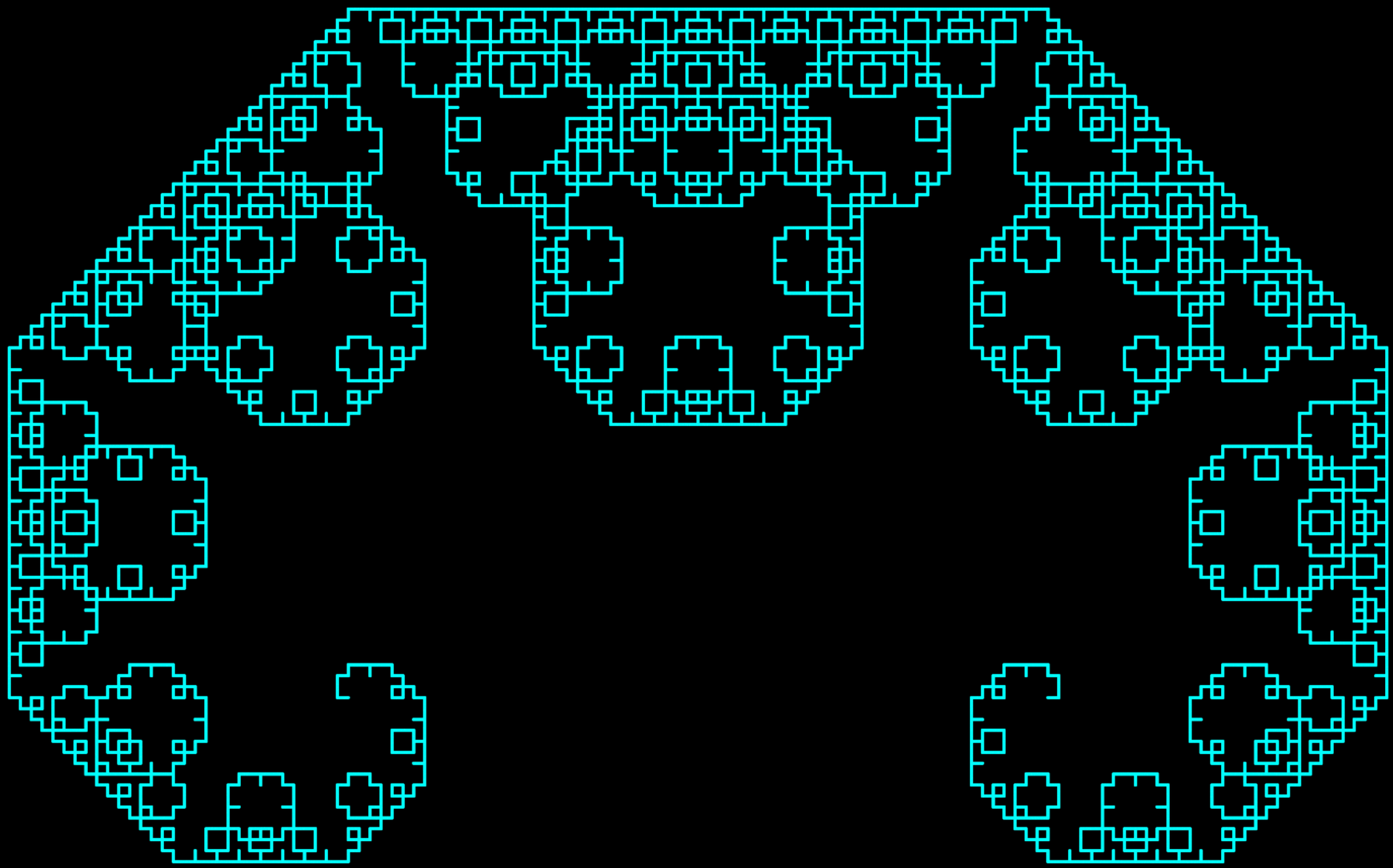


Island Fractal created using Similarity Transformations

The island is created by first transforming the red pattern shown below similarly onto each of the three red segments that make it up: (0,0) to (.4,.2), (.4,.2) to (.6,-.2), and (.6,-.2) to (1,0). This produces nine smaller blue segments onto which the pattern can again be transformed, producing 27 green segments. Finally the red pattern can be transformed onto the green segments, producing 81 black segments. This process can be continued indefinitely. The results are then replicated and attached at right angles to each other to form the island.



Levy's Fractal with Depth = 12

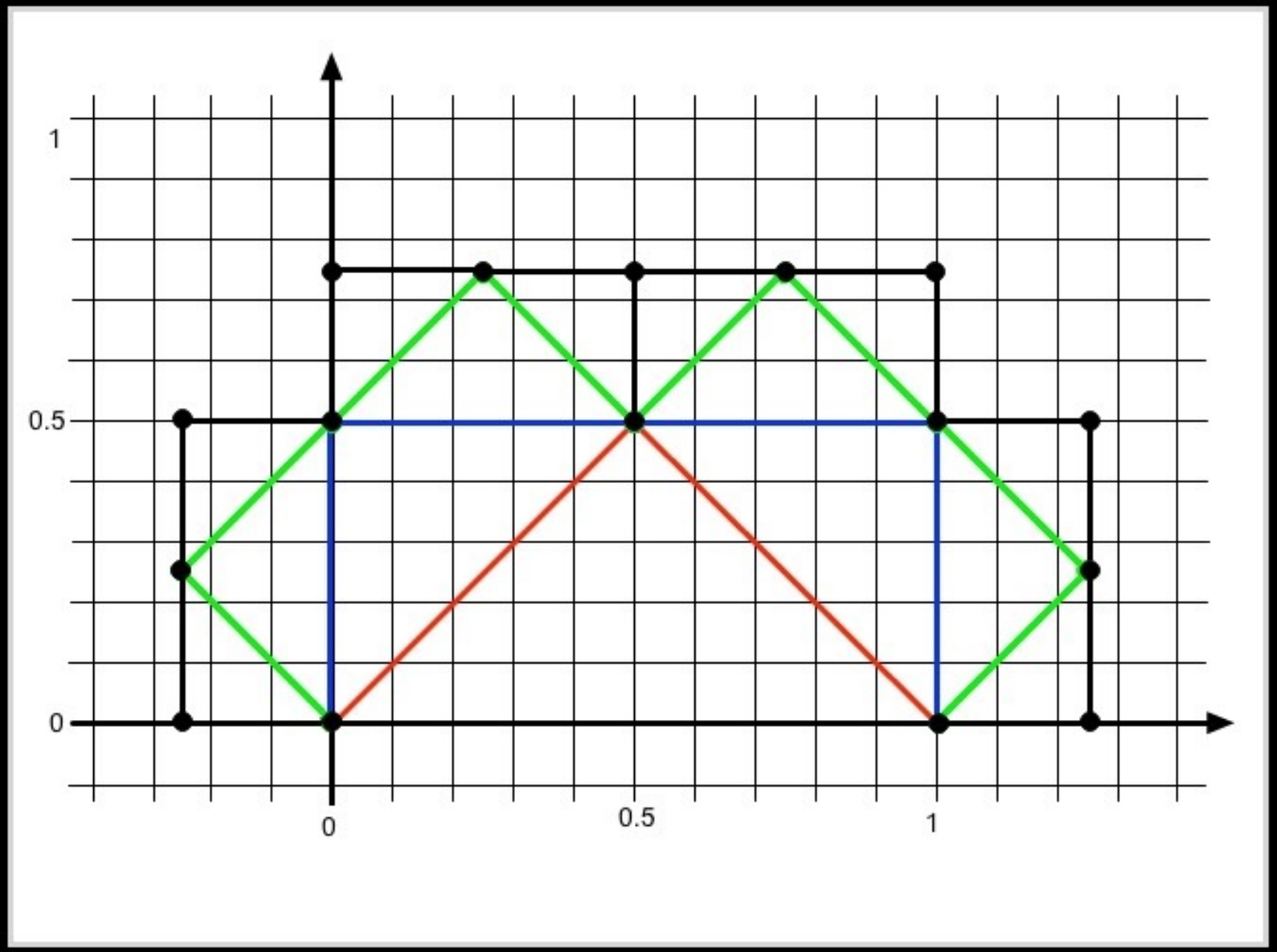


Levy's Fractal created using Similarity Transformations

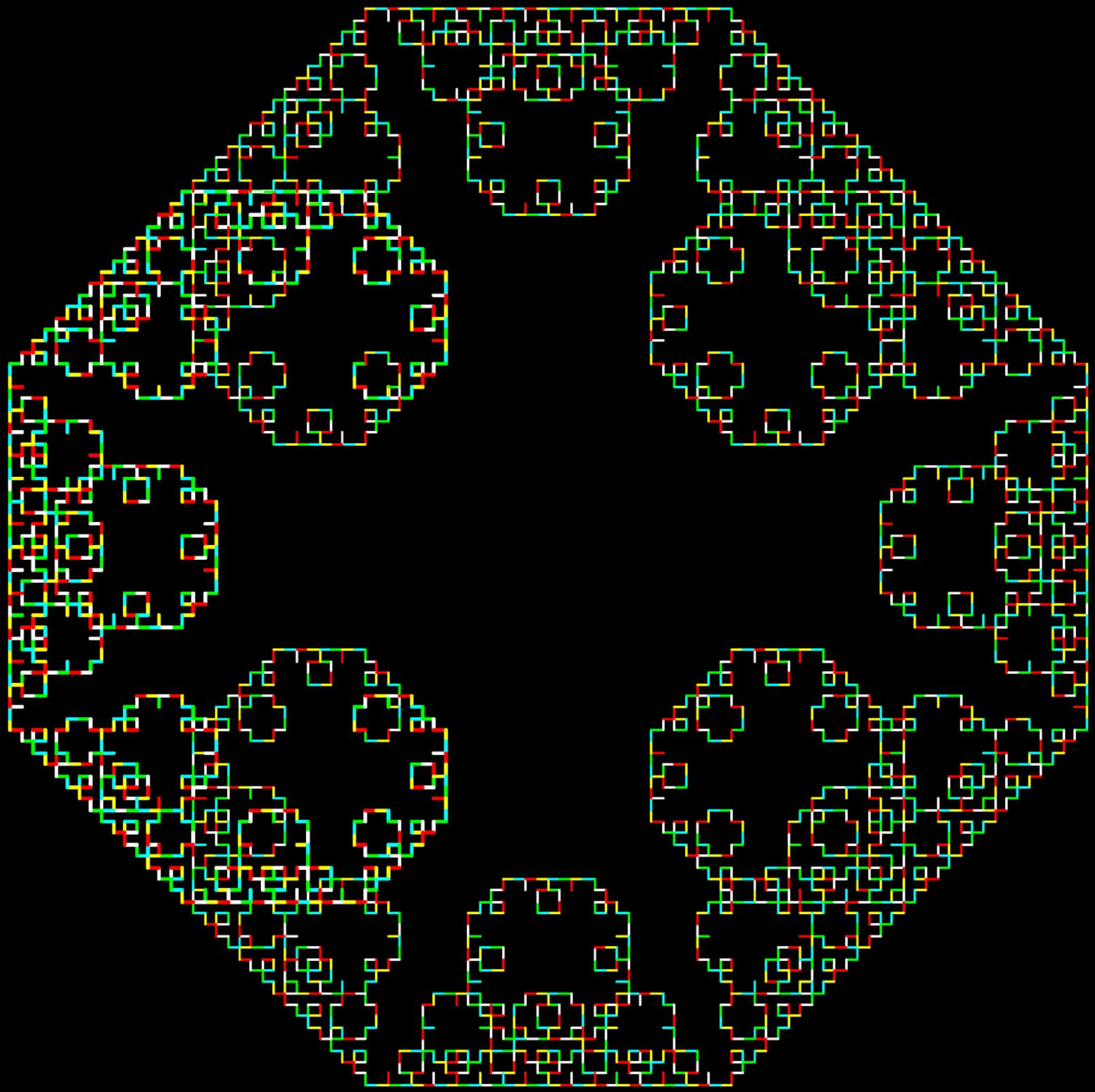
You can construct Levy's fractal by making similarity transformations of the simple inverted V shape, shown in red, that connects (0,0) to (0.5,0) and (0.5,0 to (1,0). When you first apply the transformation, you get the blue segments. When you apply the transfomation to the endpoints of the blue segments, you get the green segments, and when you apply it to the endpoints of the green segments, you get the black segmeents.

The math:
x=(x2-x1)•oldx - (y2-y1)•oldy+x1
y=(y2-y1)•oldx + (x2-x1)•oldy+y1

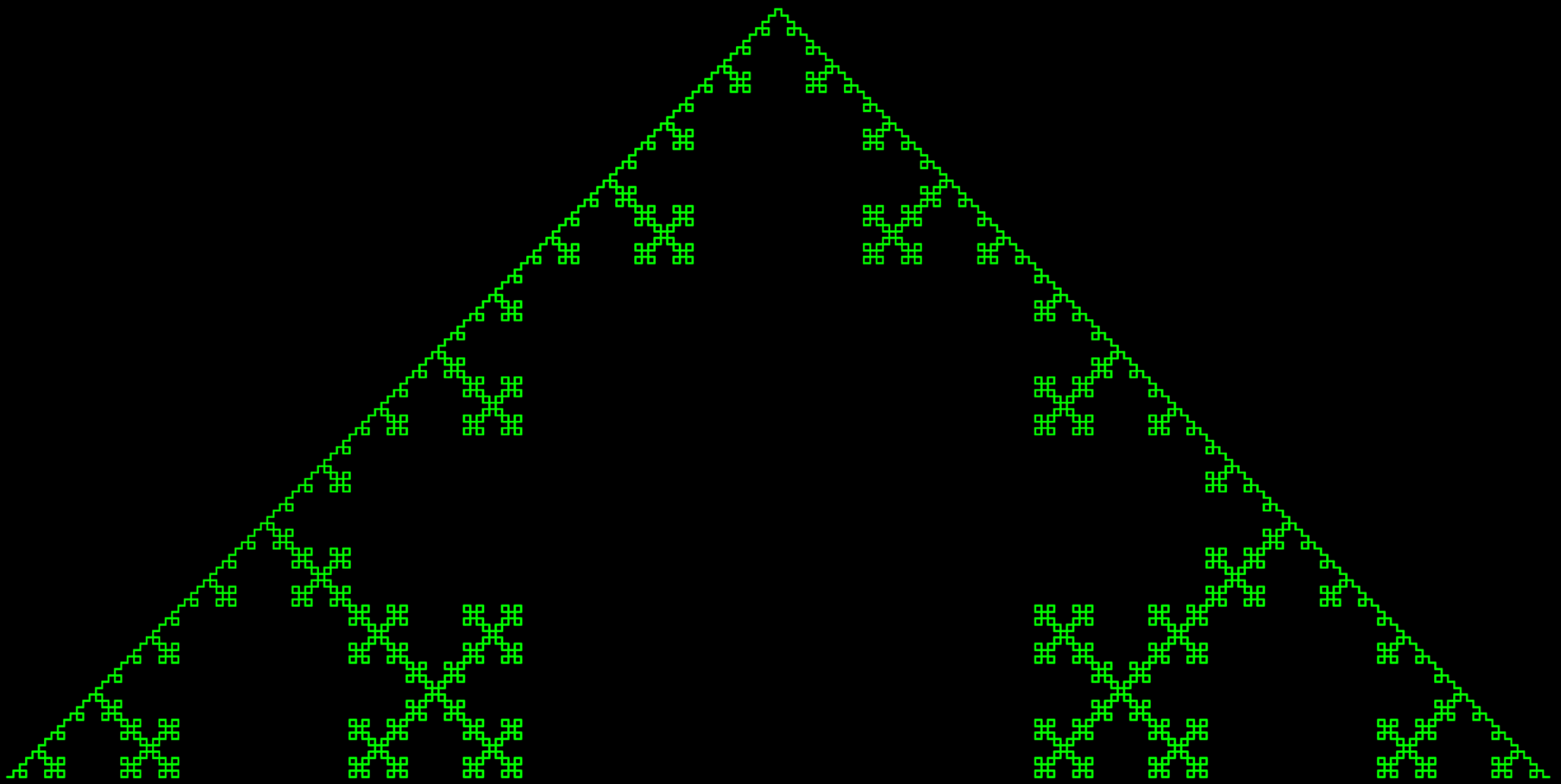
where (x1,y1) and (x2,y2) are the endpoints of the segment to which you are attaching the new shape, and (x,y) is the location to which each (oldx,oldy) is transformed.



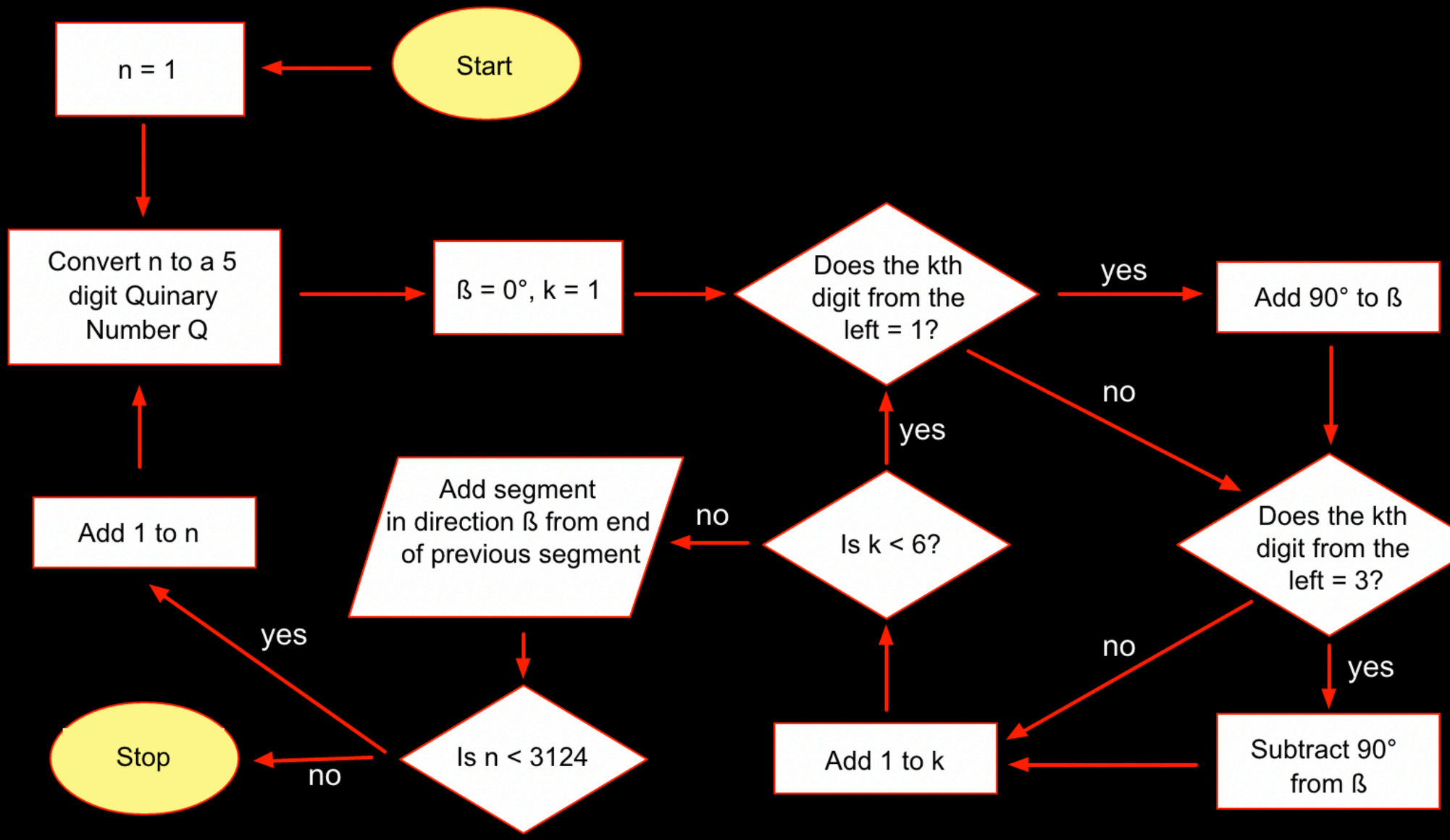
4 Levy Fractals connected at 90° Angles with Depth=10



Cross Fractal with Depth = 5

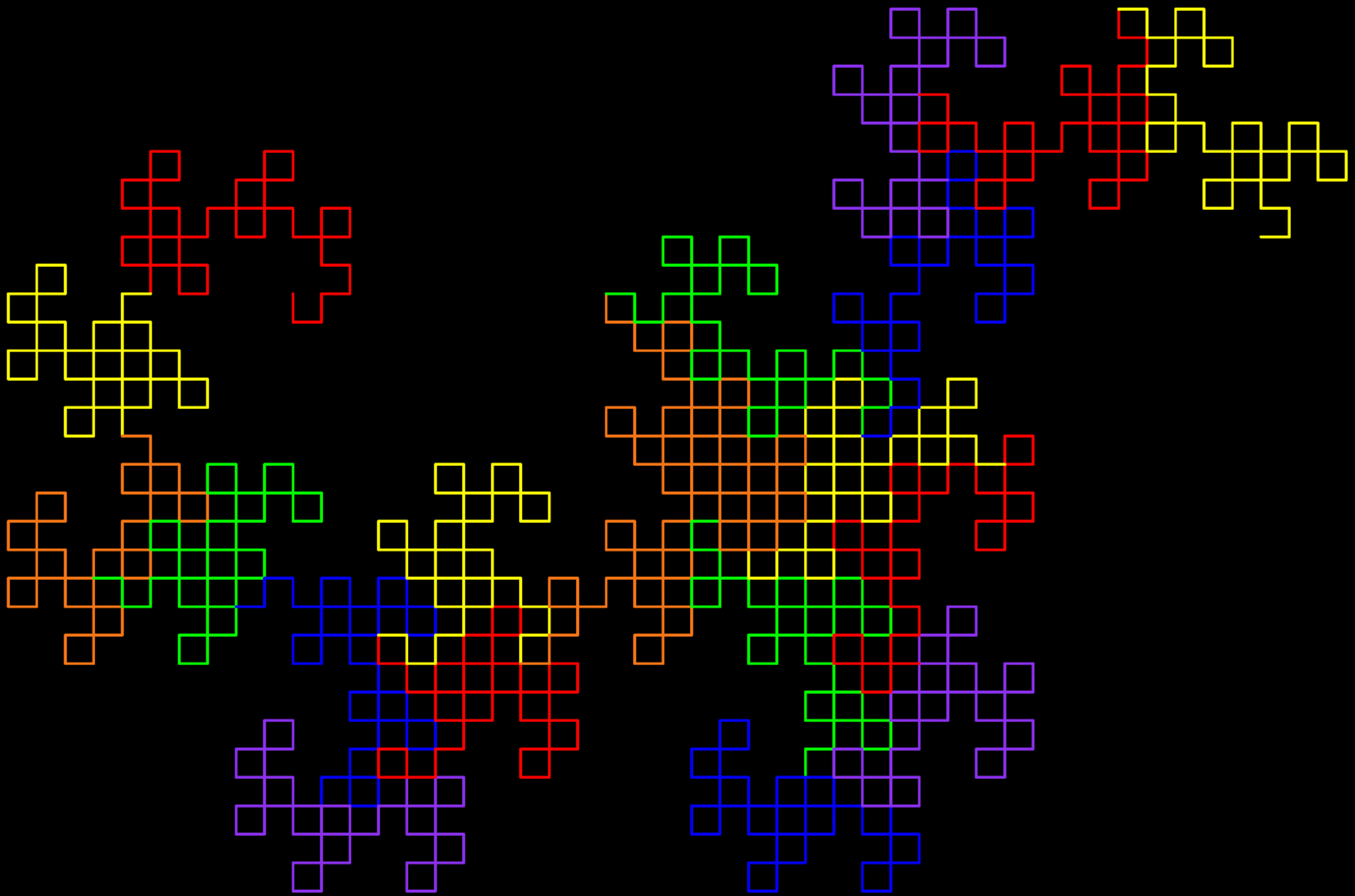


Flowchart Showing how the Cross Fractal is Created

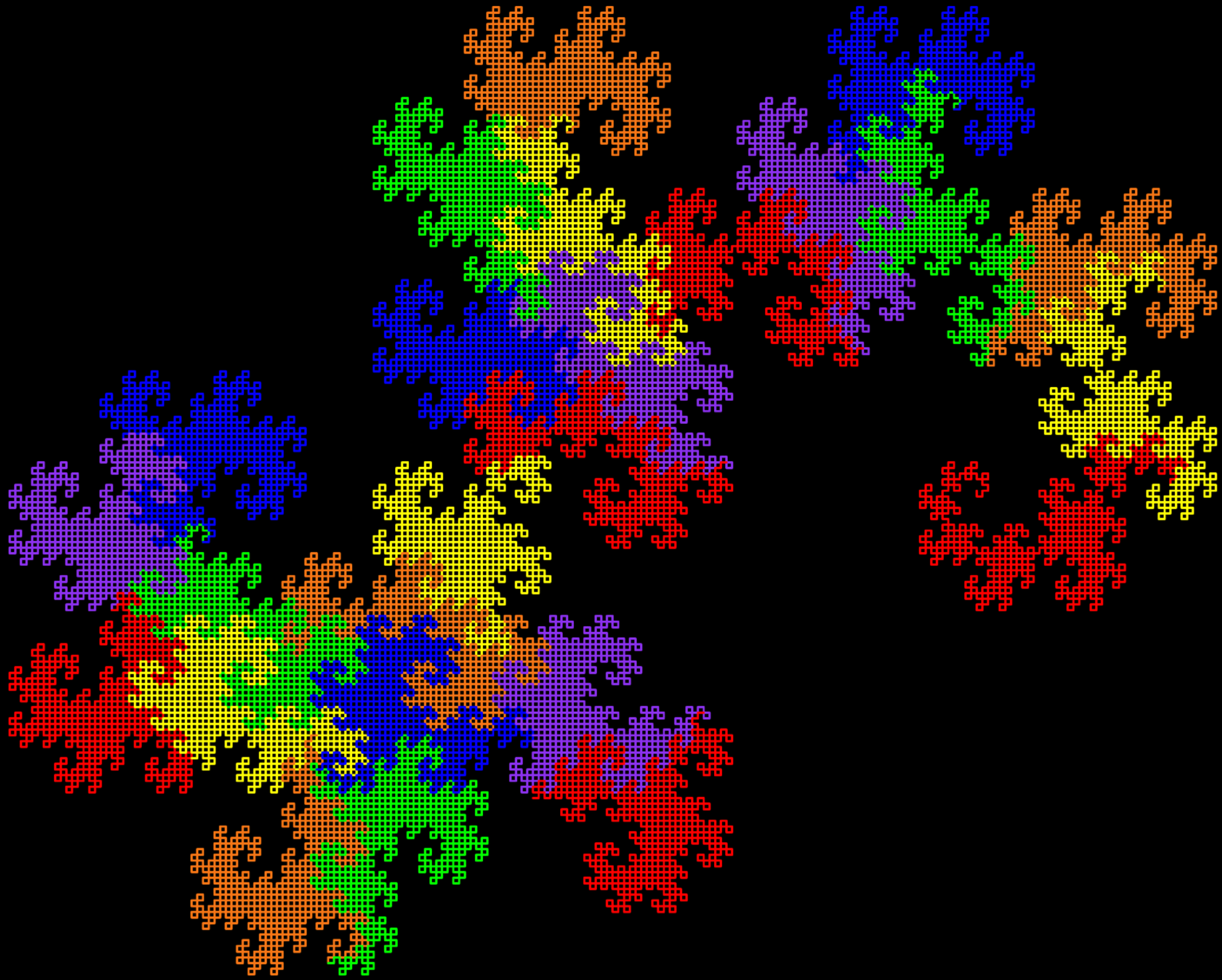


3124 very short segments are combined to form this fractal. The greater the depth, the more changes in direction are inserted between those segments. The location of the turns are determined by the segment number (n) after it has been converted to quinary (Q). When the depth is 5, all five digits of the number Q are examined. If a digit = 1, then ß is increased by 90°; if a digit = 3, then ß is decreased by 90°. Other digits are ignored. The value of ß determines the diirection of the added segment. The angles are measured counter-clockwise from the positive x-axis.

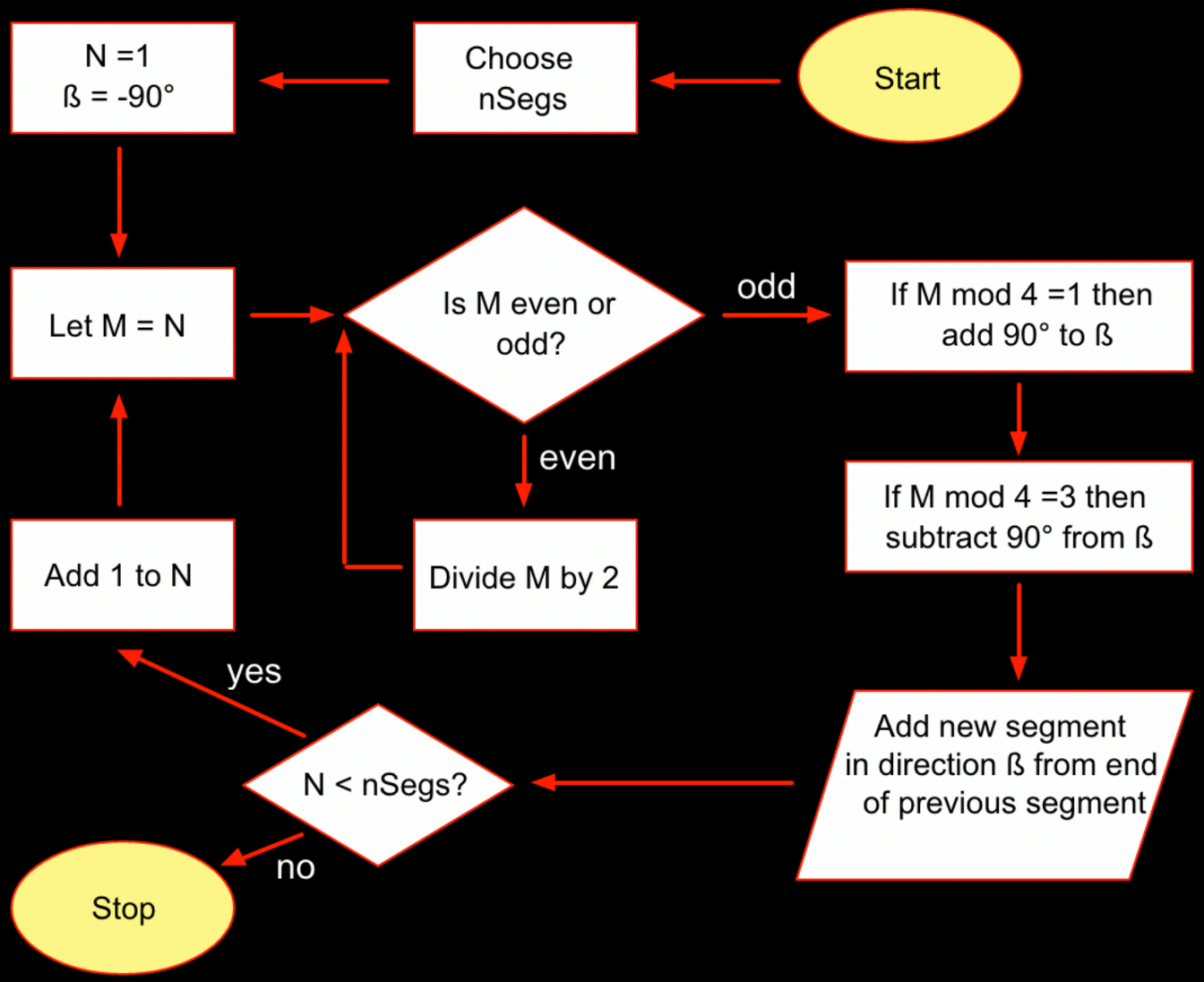
Dragon Curve with 1000 Segments



Dragon Curve with 25000 Segments



Flowchart Showing how the Dragon Curve is Created



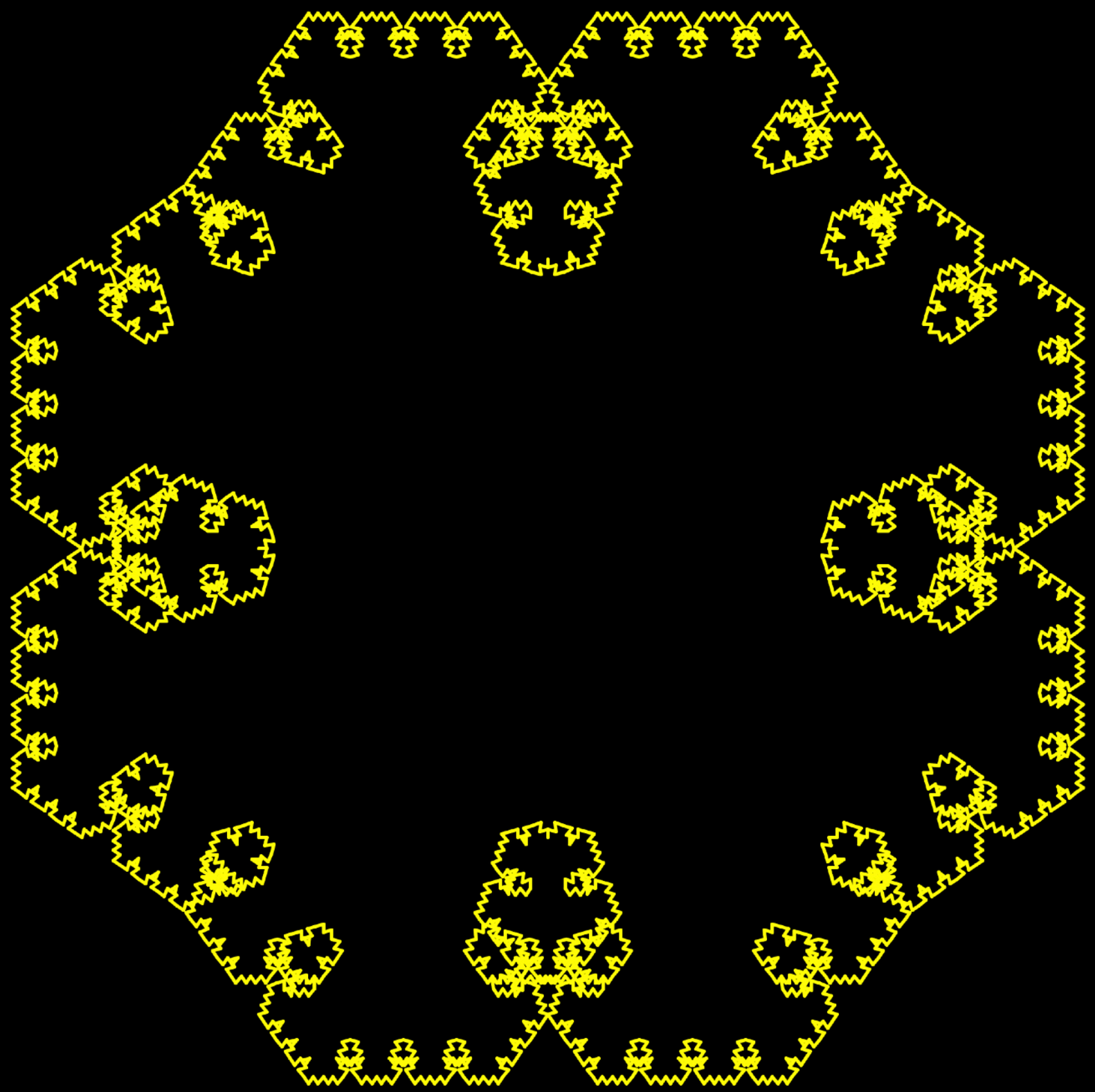
The **Dragon Curve** is one long meandering line composed of short individual segments. Changes in the direction of those segments are based on mod 4 arithmetic. The curve gets increasingly complex as you decrease the lengths of the segments and increase the number of them.

nSegs is the number of segments. A choice of 1000 shows a close-up view of what's happening; a choice of 25000 reveals the amazing complexity of the curve.

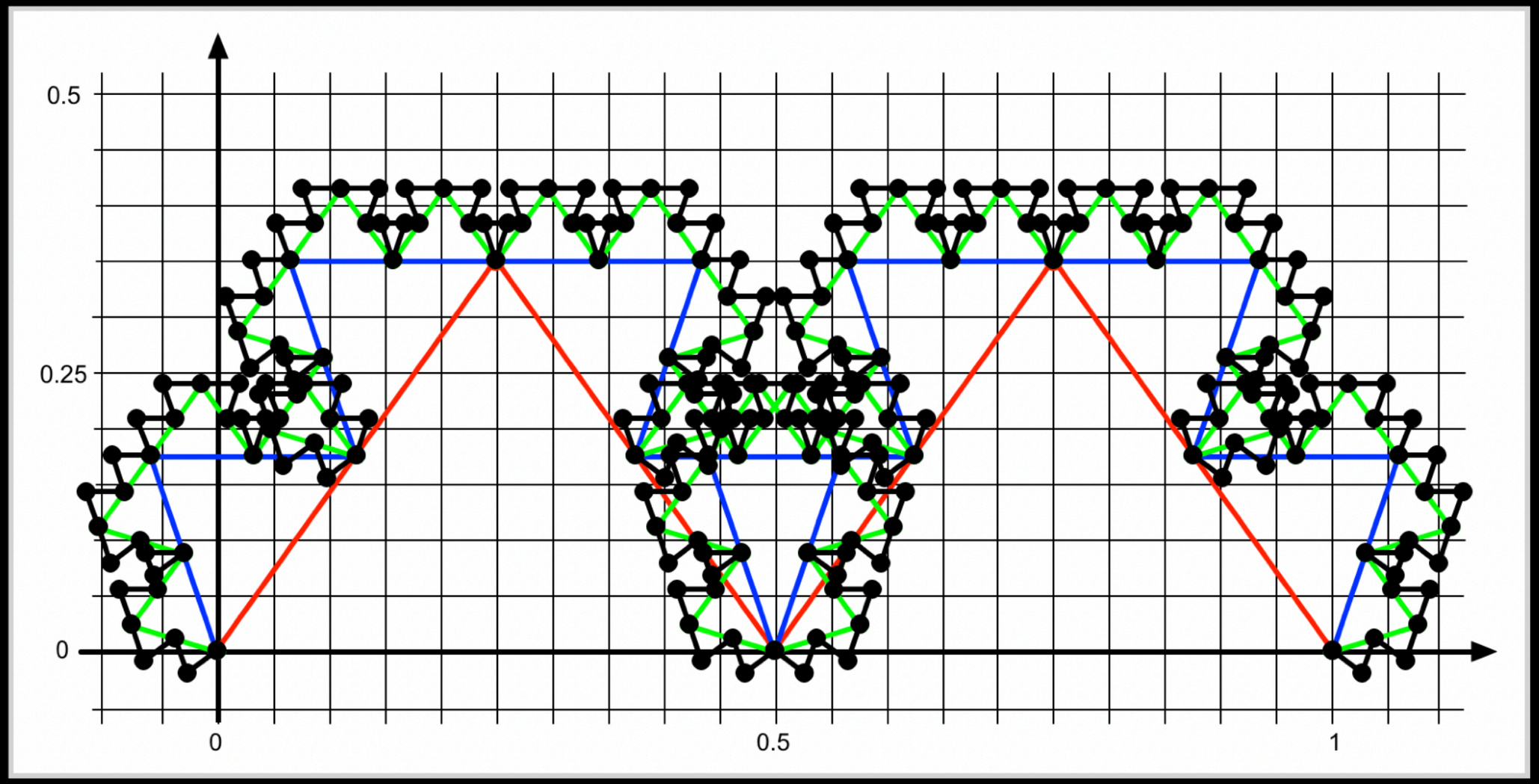
ß is the direction angle of each segment, measured counter-clockwise from the x-axis. The direction of the initial segment is arbitrarily set to -90° so that more of the curve fits on the screen. Notice that the line never retraces any of its previous parts.

The segment colors have no special meaning. When nSegs is 1000 the color changes after every 50 segments. When nSegs is 250000, the color changes after every 1000 segments.

Four connected Mountain Fractals with Depth = 5

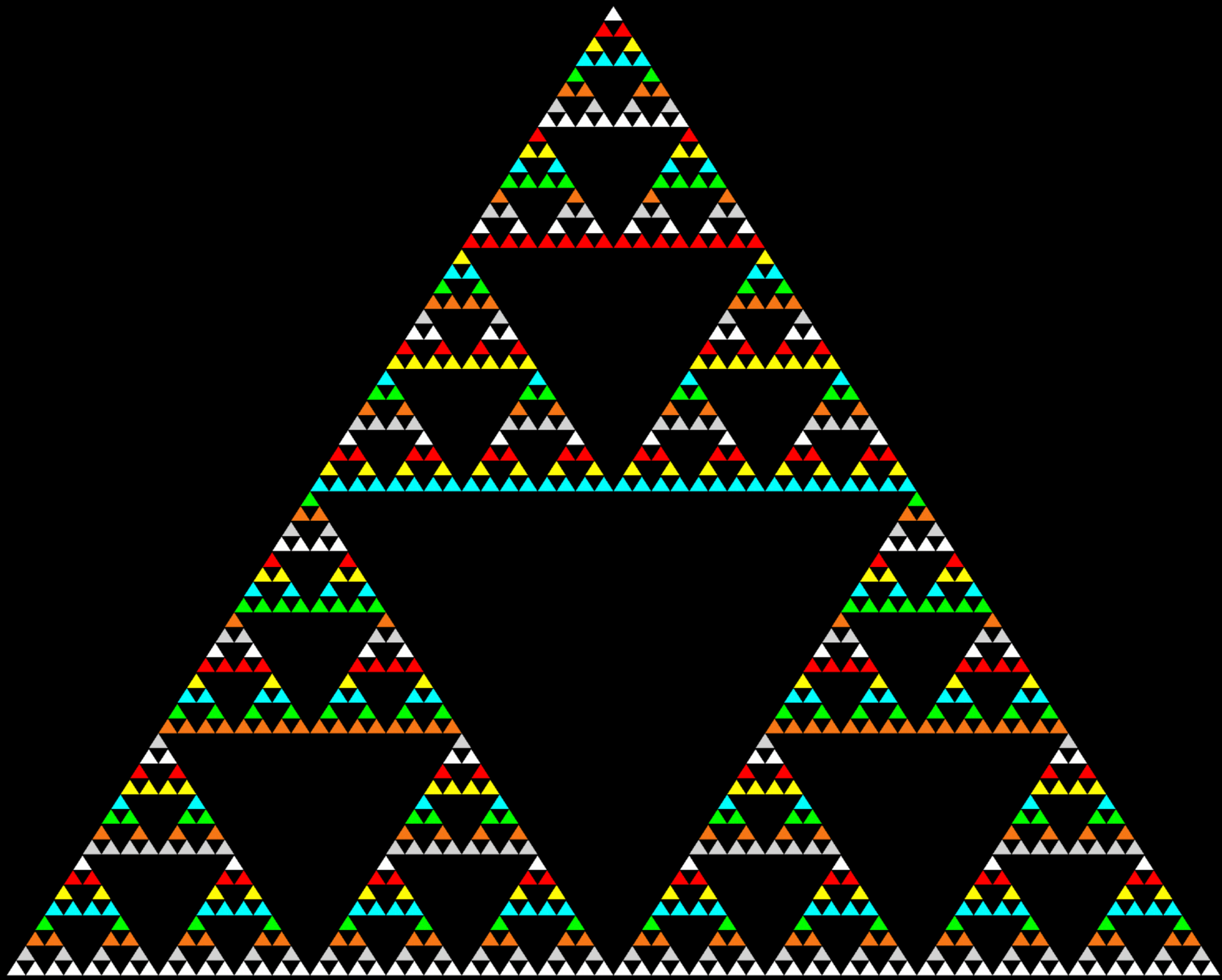


Transformation used to create the Mountain Fractal



The Mountain Fractal is constructed by making similarity transforrmations of the simple M shape that connects (0,0) to (.25,.35), (.25,.35) to (.5,0), (.5,0) to (.75,.35) and (.75,.35) to (1,0). The transformation attaches a smaller copy of the original red shape onto each of its segments, producing 16 blue segments. Then the original M shape is transformed onto each of the blue segments, producing 64 green segments. Then the original M shape is transformed onto each of the green segments, producing 256 black segments. This process can be continued indefinitely. Four copies of the resulting fractal can then be combined, attaching the start of each one to the end of the previous one after being rotated by 90°.

Sierpinski's Triangle



Sierpinski’s Triangle created using the Game of Life

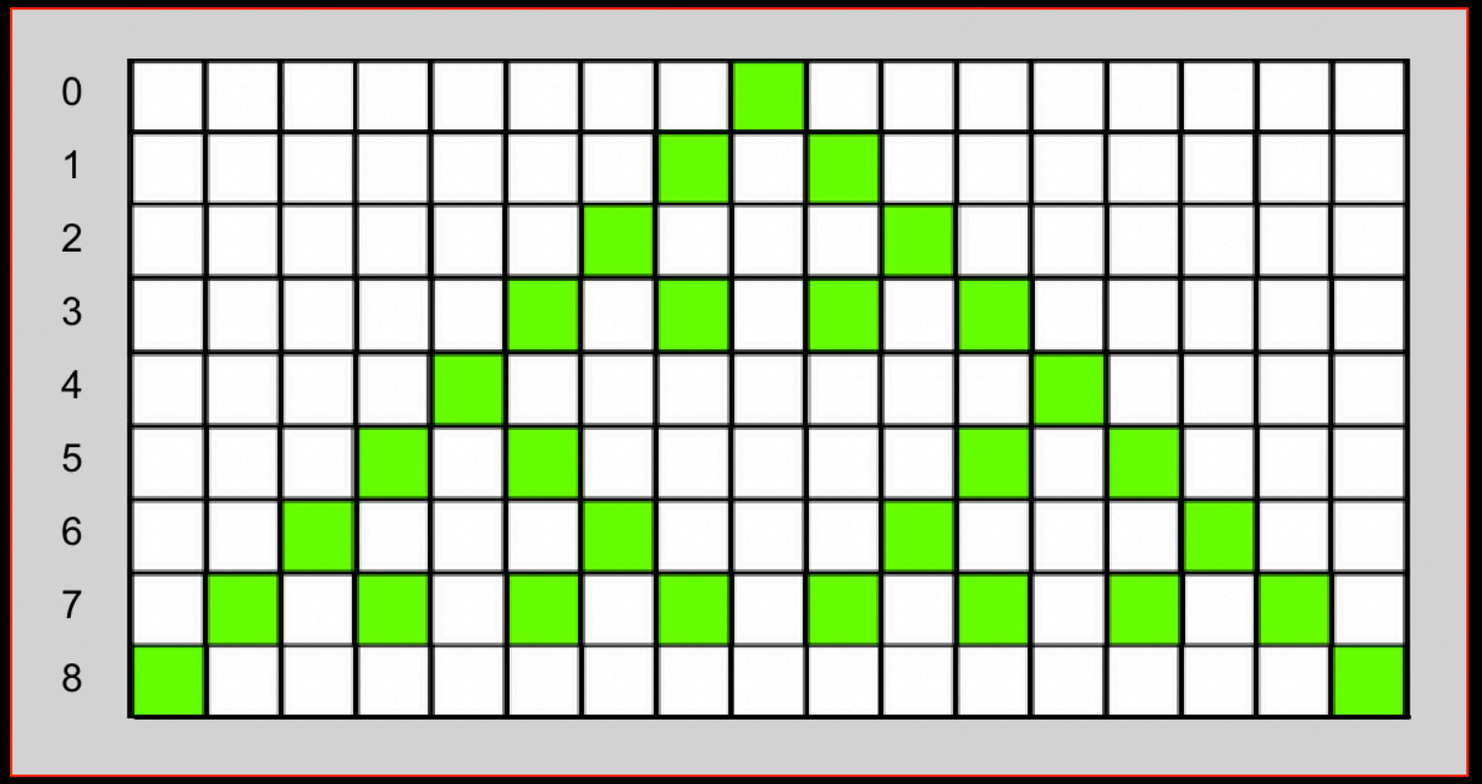
One way to create Sierpinski's Trriangle is by playing a one-dimensional game of life.

Step 0: Start with a single "living" green cell.

Step 1: Any white cell that touches that green cell becomes green, but the original green cell becomes white. The result is also pasted below step 0.

Step 2: Any white cell that touches only 1 green cell becomes green, but the two green cells from step 1 become white. Press the right arrow to see the result. The result is pasted below step 1.

Repeat the process over and over again. Any white cell from the previous step that touches just one white cell becomes green. Any green cell from the previous step becomes white. Steps 0 through 8 are shown here. The display on the previous page goes through step 54, using small triangles instead of squares. This process can be continued indefinitely, revealing more and more detail.



Similarity Transformation Example

The equations for doing a **similarity transformation** of a pattern of line segments connecting (0,0) to (1,0) are:

$$x' = (x2-x1)\cdot x - (y2-y1)\cdot y + x1$$
$$y' = (y2-y1)\cdot x + (x2-x1)\cdot y + y1$$

where (x1,y1) and (x2,y2) are the chosen endpoints for the transformed pattern of segments, (x,y) is any point on the original pattern, and (x',y') is its transformed point.

In the example shown here, the objective is to transform the red pattern of segments to the black pattern of segments.

x1=1/2, y1 =1, x2=1, and y2=3/2, so the equations become:

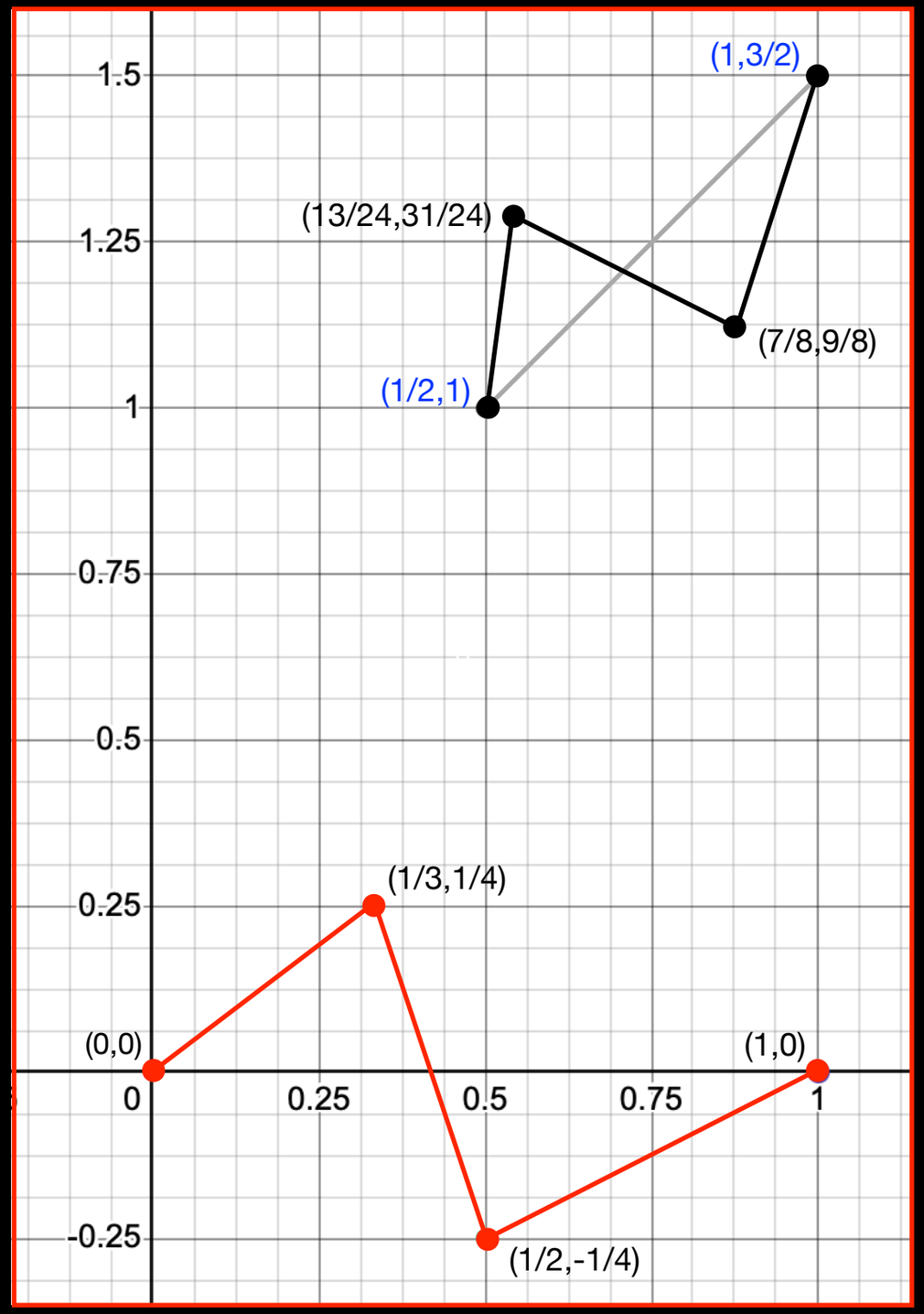
$$x' = (1/2)\cdot x - (1/2)\cdot y + 1/2$$
$$y' = (1/2)\cdot x + (1/2)\cdot y + 1$$

If (x,y) = (0,0), then x' = 0 - 0 +1/2 =1/2 and
y' = 0 + 0 +1 = 1, so (0,0) ⟶ (1/2,1).

If (x,y) = (1,0), then x' = 1/2 - 0 +1/2 =1 and
y' = 1/2 + 0 + 1 = 3/2, so (1,0) ⟶ (1,3/2).

If (x,y) = (1/3,1/4), then x' = 1/6 - 1/8 + 1/2 = 13/24 and
y' = 1/6 + 1/8 + 1 = 31/24, so (1/3,1/4) ⟶ (13/24,31/24).

If (x,y) = (1/2,-1/4) then x' = 1/4 + 1/8 + 1/2 = 7/8 and
y' = 1/4 - 1/8 + 1 = 9/8, so (1/2,-1/4) ⟶ (7/8,9/8)



Acknowledgements

Thanks to LiveCode for the app creation platform that I used to write my fractal programs.

Thanks to Hans Lauwerier for his book “Fractals” that helped me understand why the programs worked.