Montana Throne

Molly Gupta

Laura Brannan

Fractals: A Visual Display of Mathematics

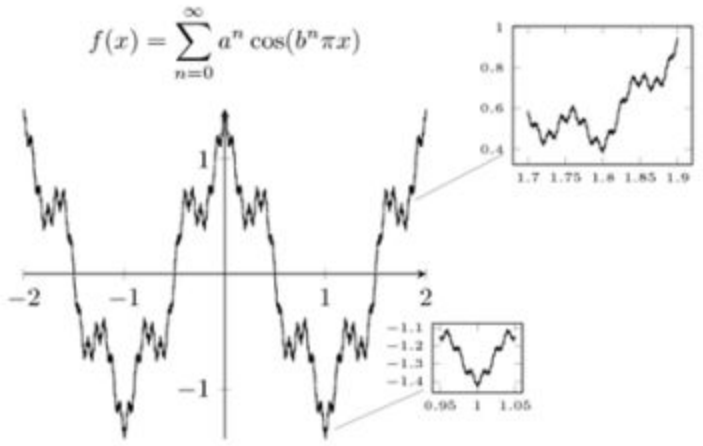
Linear Algebra - Math 2270

Introduction:

Fractals are infinite patterns that look similar at all levels of magnification and exist between the normal dimensions. With the advent of the computer, we can generate these complex structures to model natural structures around us such as blood vessels, heartbeat rhythms, trees, forests, and mountains, to name a few. We will explain the history of fractals, how they work, what types exists, and how humans have attempted to simulate these complex structures with computers.

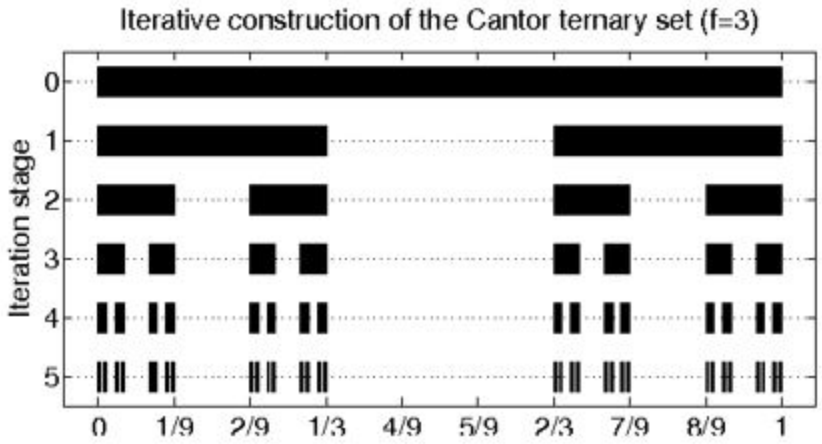
A Brief History:

An important property of fractals is the iteration or feedback loop. This concept first appears hundreds of years ago in the work of Leibniz. He studied self-similarity in the 17th century but didn't quite know how it fit in with the mathematics of the time. Two centuries passed before self-similarity appeared again in the Weierstrass Function. This function looks similar when the magnitude is increased, and it was the first function to show that it was possible for a function to not be differentiable and still be continuous.



http://pgfplots.net/tikz/examples/weierstrass-function/

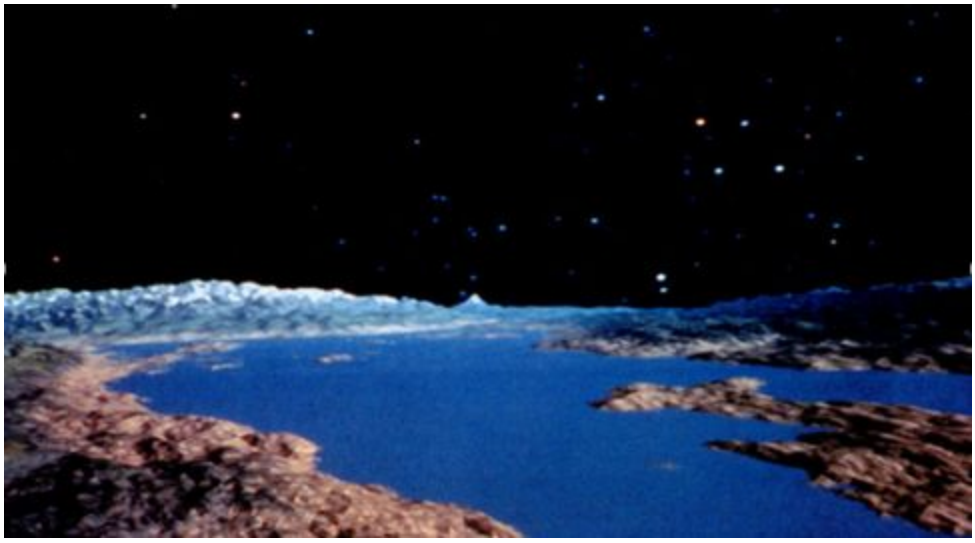
The Cantor set by George Cantor and Henri Poincare's work contain properties of self-similarity and were studied in the late 19th century. The Cantor Set is now recognized as a fractal.



https://www.semanticscholar.org/topic/Cantor-set/150629

Helge von Koch and Waclaw Sierpinski explored geometric construction that was self-similar and infinite and deviated from the normal. Felix Hausdorff's work on dimensions introduced the noninteger dimension that is also a property of fractals. These works inspired Benoit Mandelbrot who coined and defined the term *fractal*. The term stems from the Latin word "fractus" which means "to break" and "irregular."

With the help of computers in the 20th century, it was finally possible to visualize the theories and functions that relied upon infinite iterations. Fractals became more popular with Mendelbrot and Heinz-Otto Peitgen's published works. Inspired by Mendelbrot, Loren Carpenter used fractals to create computer generated graphics that were used in "Star Trek II: The Wrath of Khan."



http://www.ex-astris-scientia.org/treknology/treknology-g.htm

How They Work:

Fractals are irregular geometric objects that are infinitely complex. Each time a you enlarge one portion of the fractal image, more complexity is revealed. This is true for every portion that you enlarge on and on for infinity. This phenomenon is referred to as "self-similarity" Fractals exhibit this quality because they are created by repeating a simple process over and over in a feedback loop based on recursion. It's an Iterated function system—the process of repeatedly replacing shapes with other shapes. Endless repetition is the key to fractal geometry.

There are several transformations used in creating fractals. A transformation is a rule for mapping one space to another space:

$$T: \mathbb{R}^2 \to \mathbb{R}^2$$

$$T(\mathbf{x}) = A\,\mathbf{x}$$

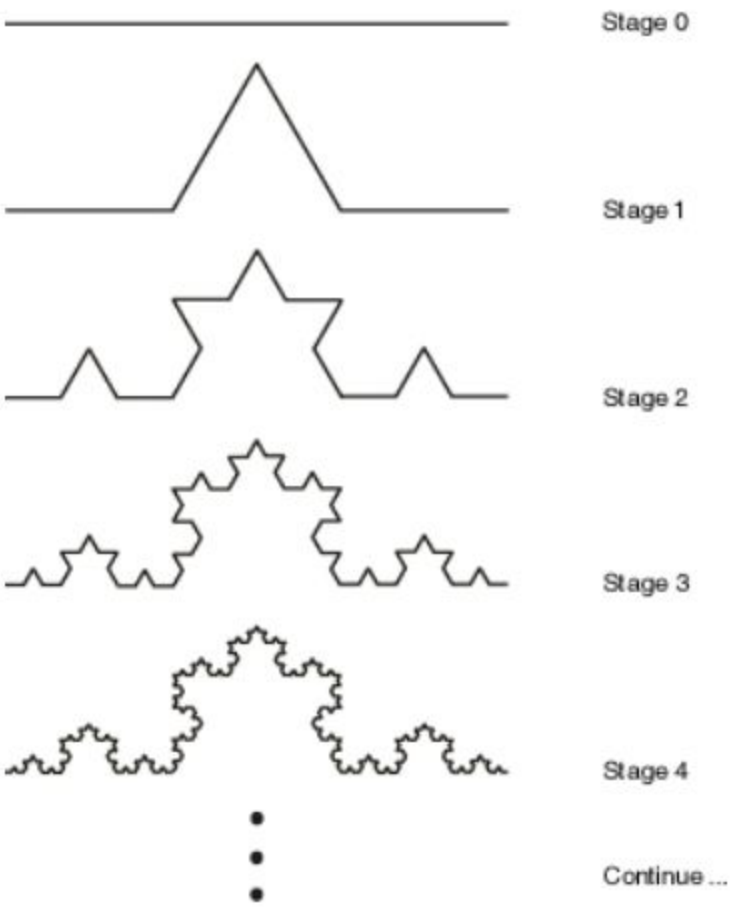
$$T\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Affine transformations are transformations that operate on individual points in the set. Examples of this type of transformations are translations, scalings, reflections, and rotations. These affine transformations and scaling maintain self-similarity. Affine transformations are computed the following way:

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

A 2x2 matrix altered by a vector. In the above equation, a, b, c, d, e and f are all scalar quantities.

The Koch Curve is a great example of how this repetitive loop process works. At each iteration, the length of the curve gets longer, and each segment of the curve becomes more complex and more jagged. With the Koch curve we begin with an initial shape of straight line. The line is split into three segments, and the middle segment is replaced with an equilateral triangle (with the bottom line removed). This is repeated over and over, making the curve more complex for infinity.



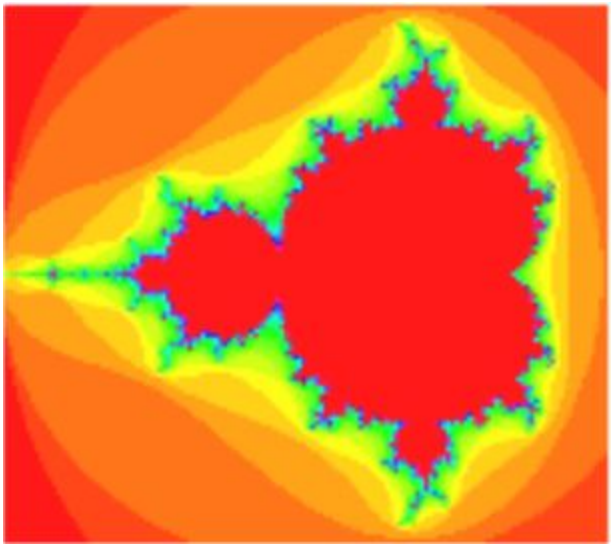
The algorithm applied is simple, yet it results in an incredibly complex shape. The fractal nature of the curve is such that it become bounded with infinite length. It is increased with each iteration, increasing with the pattern 3 + ¾ + ¾ + ¾ +.... until infinity. This shape can only be described using fractal geometry, not conventional mathematics. This process is easily computed using a programming technique called recursion. Recursion is when a programming algorithm performs some action, then calls itself on the product, performing the action over and over again. For example, the mandelbrot set is based on the following recursive formula:

$$Z_n = Z_{n-1}^2 + C$$

Where C is a number on the complex plane (of the form C = a + bi). We begin with Z_0, and whatever the result is is then utilized in place of Z_{-1} to calculate Z_1, and so on, to gather all of

the points in the mandelbrot set. This is the simple maple code to produce the infinitely complex mandelbrot set:

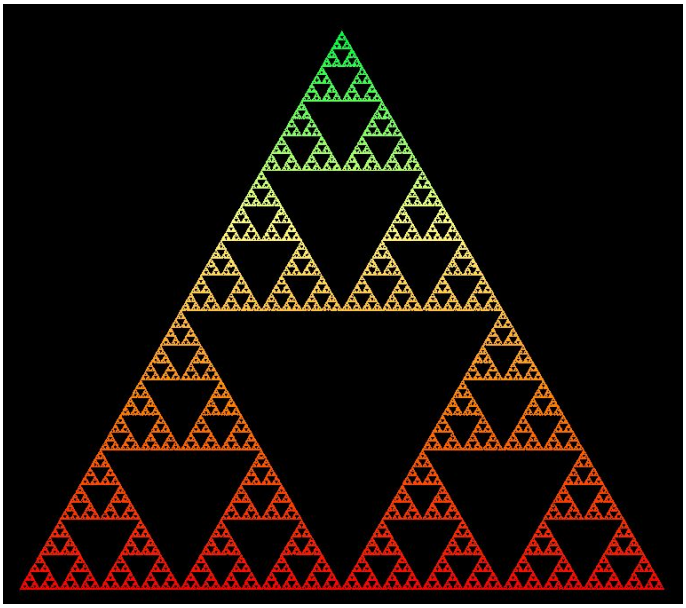
```
mandelbrot := proc(x, y)
local c, z, m;
c := evalf(x+y*I); z := c;
for m to 50 while abs(z) < 2 do z := z^2+c od;
m end;

plots[densityplot](mandelbrot,-2 .. 0.7, -1.35 .. 1.35,
s_tyle=patchnogrid,colorstyle=HUE,numpoints=62500,axes=none);
```



https://youtu.be/0jGaio87u3A

Types of Fractals:

Sierpinski Triangle was named after Waclaw Sierpinski, who is a famous mathematician based in Poland. The basic idea of creating this type of triangle is through the use of recursion. The process is recursively dividing a triangle into smaller divisions of triangles creating an image composed of triangles within each other and layered next to each other. However, an important idea of this fractal is the main and largest triangle is upward and the rest of the triangles within the largest triangle are upside down. All of the triangles drawn within the largest one must have corners touching the midpoint of the triangles surrounding them.



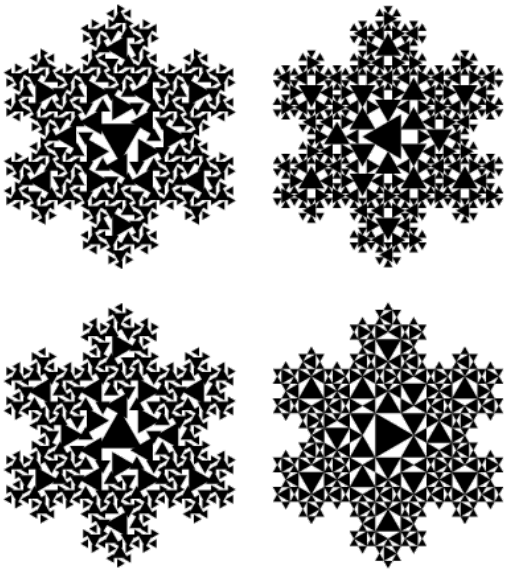
The picture of the sierpinski triangle is provided from the link:

http://ecademy.agnesscott.edu/~lriddle/ifs/siertri/siertri.htm

The code for this fractal involves recursive methods. Below, the sample code for sierpinski triangle is provided from the website, http://lodev.org/cgtutor/sierpinski.html. As shown by the code, there are multiple recursions occurring through each method.

```
//Declaration of the drawSierpinski function. The coordinates are the 3 outer corners of the Sierpinski Triangle.
void drawSierpinski(float x1, float y1, float x2, float y2, float x3, float y3);
//Declaration of the subTriangle function, the coordinates are the 3 corners, and n is the number of recursions.
void subTriangle(int n, float x1, float y1, float x2, float y2, float x3, float y3);

//depth is the number of recursive steps
int depth = 7;

//The main function sets up the screen and then calls the drawSierpinski function
int main(int argc, char *argv[])
{
  screen(640, 480, 0, "Sierpinski Triangle");
  cls(RGB_White); //Make the background white
  drawSierpinski(10, h - 10, w - 10, h - 10, w / 2, 10); //Call the sierpinski function (works with any corners inside the s
  //After drawing the whole thing, redraw the screen and wait until the any key is pressed
  redraw();
  sleep();
  return(0);
}

//This function will draw only one triangle, the outer triangle (the only not upside down one), and then start the recursive
void drawSierpinski(float x1, float y1, float x2, float y2, float x3, float y3)
{
    //Draw the 3 sides of the triangle as black lines
    drawLine(int(x1), int(y1), int(x2), int(y2), RGB_Black);
    drawLine(int(x1), int(y1), int(x3), int(y3), RGB_Black);
    drawLine(int(x2), int(y2), int(x3), int(y3), RGB_Black);

    //Call the recursive function that'll draw all the rest. The 3 corners of it are always the centers of sides, so they're
    subTriangle
    (
      1, //This represents the first recursion
      (x1 + x2) / 2, //x coordinate of first corner
      (y1 + y2) / 2, //y coordinate of first corner
      (x1 + x3) / 2, //x coordinate of second corner
      (y1 + y3) / 2, //y coordinate of second corner
      (x2 + x3) / 2, //x coordinate of third corner
      (y2 + y3) / 2  //y coordinate of third corner
    );
}

//The recursive function that'll draw all the upside down triangles
void subTriangle(int n, float x1, float y1, float x2, float y2, float x3, float y3)
{
  //Draw the 3 sides as black lines
  drawLine(int(x1), int(y1), int(x2), int(y2), RGB_Black);
  drawLine(int(x1), int(y1), int(x3), int(y3), RGB_Black);
  drawLine(int(x2), int(y2), int(x3), int(y3), RGB_Black);

  //Calls itself 3 times with new corners, but only if the current number of recursions is smaller than the maximum depth
  if(n < depth)
  {
    //Smaller triangle 1
    subTriangle
    (
      n+1, //Number of recursions for the next call increased with 1
      (x1 + x2) / 2 + (x2 - x3) / 2, //x coordinate of first corner
      (y1 + y2) / 2 + (y2 - y3) / 2, //y coordinate of first corner
      (x1 + x2) / 2 + (x1 - x3) / 2, //x coordinate of second corner
      (y1 + y2) / 2 + (y1 - y3) / 2, //y coordinate of second corner
      (x1 + x2) / 2, //x coordinate of third corner
      (y1 + y2) / 2  //y coordinate of third corner
    );
    //Smaller triangle 2
    subTriangle
    (
      n+1, //Number of recursions for the next call increased with 1
      (x3 + x2) / 2 + (x2 - x1) / 2, //x coordinate of first corner
      (y3 + y2) / 2 + (y2 - y1) / 2, //y coordinate of first corner
      (x3 + x2) / 2 + (x3 - x1) / 2, //x coordinate of second corner
      (y3 + y2) / 2 + (y3 - y1) / 2, //y coordinate of second corner
      (x3 + x2) / 2, //x coordinate of third corner
      (y3 + y2) / 2  //y coordinate of third corner
    );
    //Smaller triangle 3
    subTriangle
    (
      n+1, //Number of recursions for the next call increased with 1
      (x1 + x3) / 2 + (x3 - x2) / 2, //x coordinate of first corner
      (y1 + y3) / 2 + (y3 - y2) / 2, //y coordinate of first corner
      (x1 + x3) / 2 + (x1 - x2) / 2, //x coordinate of second corner
      (y1 + y3) / 2 + (y1 - y2) / 2, //y coordinate of second corner
      (x1 + x3) / 2, //x coordinate of third corner
      (y1 + y3) / 2  //y coordinate of third corner
    );
  }
}
```

Koch was originally visualized by Helge von Koch. Recursion curve with triangle like formations branching off from one another. The inside of the shapes tend to stay hollow with only the outer layer showing. There is a specific formation pattern for this fractal. In order to create this fractal, an equilateral triangle must become the starting point. Every side of the triangle is split into thirds, and the middle third is replaced with another equilateral triangle. The splitting and replacing becomes recursive. Every equilateral triangle will go through this effect until the recursion loop stops. The example below specifically involves the shape of a snowflake koch found on the website: http://mathworld.wolfram.com/KochSnowflake.html. The snowflake pattern involves using triangles in a different pattern described above but the same recursive method and use of triangles.



Galaxies are widely speculated to be a fractal like pattern. There is no factual evidence that proves this theory, but observations and education hypothesis create very convincing arguments supporting this theory. The clusters within the sky from stars, meteorites, and the mysteries of space bring a fractal like formation to the galaxy. Our specific galaxy has the sun and planets revolving around it. There are galaxies similar to our galaxy out there in the universe. The similarities in the galaxies is evidence of the fractal patterns. The picture below is provided by the website: https://www.newscientist.com/article/dn14200-galaxy-map-hints-at-fractal-universe/.

Fractals in Nature:

We may not realize, but many objects in nature have fractal-like structures. A repeated simple branching process is an integral part of the natural world around us. Just as a fractal is

formed by a simple pattern repeated in a loop, many objects in nature are formed by a similar repetitive and combining expansion. It wasn't until research and the development of fractals took off that mathematicians and scientist realized that almost all of nature can be represented or described using mathematics. With Euclidean mathematics, this never would have been possible. The development of fractal geometry opened up a whole new world in regards to describing and representing the world around us.

Examples:

Geographic features: mountains, coastlines

Mountains are created by tectonic forces pushing the crust upward and erosion tearing some of that crust down, resulting in a fractal pattern.

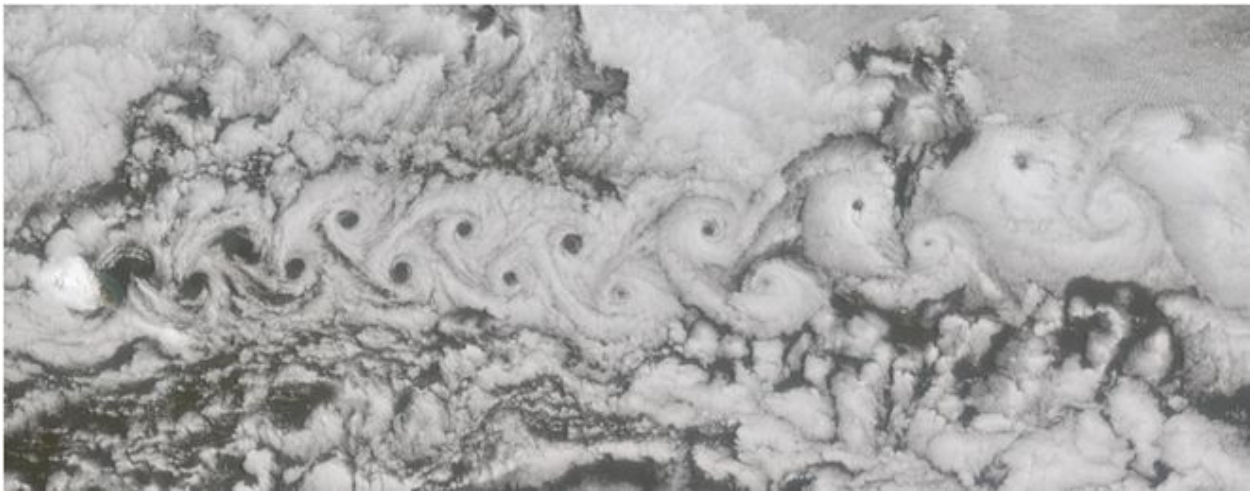
(aerial view of the himalayas)

Romanescu

This vegetable exhibits a pattern that is a natural representation of the Fibonacci or golden spiral, a logarithmic spiral where every quarter turn is farther from the origin by a factor of phi, the golden ratio.

Lightning, clouds, hurricanes

A fractal cloud pattern is interrupted by a series of diagonal grooves.



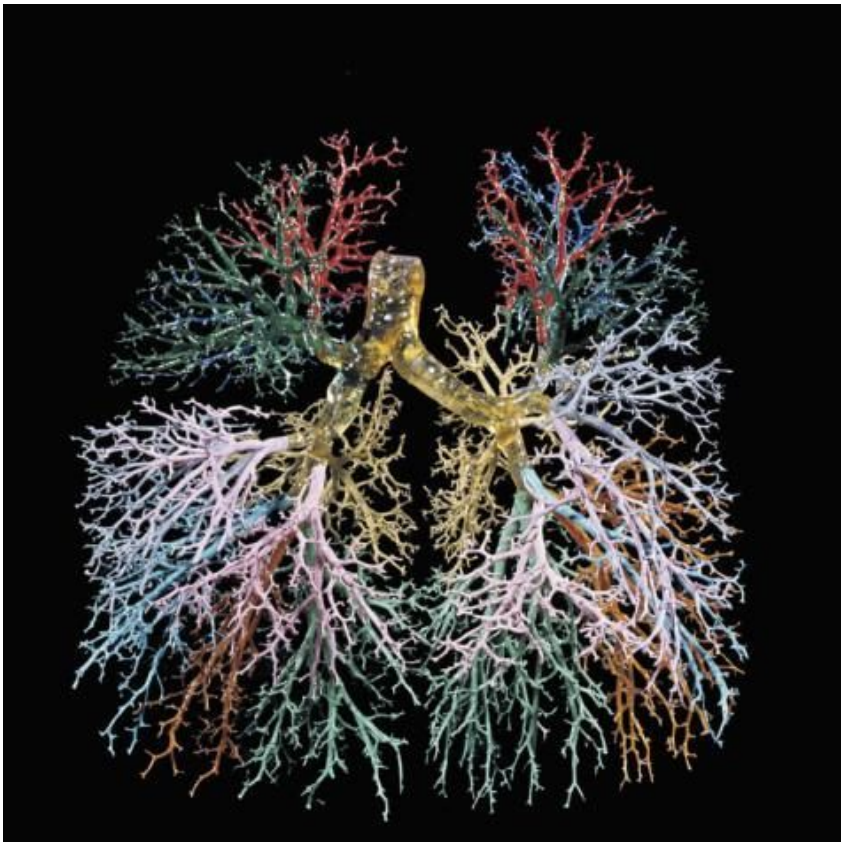
Lightning takes a step by step path to the ground, following a fractal trend with each branch.

Trees/ferns

Ferns are a common example of a self-similar set, meaning that their pattern can be mathematically generated and reproduced at any magnification or reduction. The mathematical formula that describes ferns, named after Michael Barnsley, was one of the first to show that random numbers generated over and over using Barnsley's Fern formula ultimately produce a unique fern-shaped object.

The human lung and brain

The bronchial tubes in the human lung have one fractal dimension for the first seven generations of branching, and a different fractal dimension from there on in.



Every cell in the body must be close to a blood vessel in order to receive oxygen and nutrients. The only way this is possible is through a fractal branching network where blood vessels branch and branch until they are incredibly small.

Bibliography

https://prezi.com/9nox7unaacwf/using-linear-algebra-techniques-to-generate-fractals/

http://www.math.utah.edu/~korevaar/fractals/

https://www.newscientist.com/article/dn14200-galaxy-map-hints-at-fractal-universe

https://georgemdallas.wordpress.com/2014/05/02/what-are-fractals-and-why-should-i-care/

http://www.fractal.org/Bewustzijns-Besturings-Model/Fractals-Useful-Beauty.htm

https://www.wired.com/2010/09/fractal-patterns-in-nature/

Peitgen, Heinz-Otto. *Beauty of Fractals*. Springer-Verlag.

Mandelbrot, Benoit. *The fractal geometry of nature: (formerly: Fractals).* New York: Freeman, 1983.

Peitgen, Heinz-Otto, et al. *Chaos and Fractals: New Frontiers of Science.* Springer, 2004.