A fractal is an object with the two properties:

- there is infinite detail at every point
- there is self-similarity between the object parts and the overall object

Alternatively, we can say that a fractal is an object with fractional dimension. (We’ll get to what that is soon.)
We typically describe fractals with procedures that get repeated to produce the detail in the object subparts. Theoretically, these procedures repeat an infinite number of times. Pragmatically, we stop after the detail gets too small for us to worry about.

Here’s a simple example, the Koch snowflake. To produce the detail, we take every straight line segment in the snowflake and replace it with a pattern of 4 segments, each one 1/3 of the length of the original.

The fractal object does this transformation infinitely. So whenever we think we have a straight line segment of it pinned down, it goes and subdivides again.

Here’s an image of the first three subdivisions of the Koch curve, starting with a triangle initiator.
What we work with in computer graphics are often called **pseudofractals**, because we don’t keep subdividing an infinite number of times. But then, neither does nature.

**Fractal Dimension**

The **fractal dimension** is a number that characterizes the amount of variation in the structure of a fractal object. It is defined analogously to the dimension of an ordinary object.
If we divide an ordinary object up into pieces that are scaled by $\frac{1}{2}$, we get 2 pieces in 1 dimension, 4 pieces in two dimensions, and 8 pieces in 3 dimensions. The relationship is that $ns^D = 1$, where $D$ is the dimension, $s$ is the scale factor, and $n$ is the number of pieces we get.

We can solve this equation for $D$:

$$D = \frac{\log n}{\log(1/s)}$$
If we apply this formula to our Koch snowflake curve, we have \( n = 4 \) pieces, at a scale factor of \( s = 1/3 \). So its fractal dimension is

\[
D = \frac{\log 4}{\log(1/(1/3))} = \frac{\log 4}{\log 3} \approx 1.2619
\]

If we have a fractal where the subparts use different scaling factors \( s_1, s_2, \text{ etc.} \) then the formula is the more-difficult to solve

\[
\sum_{k=1}^{n} S_k^D = 1
\]

which becomes our simple \( ns^D = 1 \) when all scale factors are the same.

For example, suppose the generator we used for the Koch curve is replaced with:

![Diagram of Koch snowflake generator](image)

which has 1 segment of length 1/2 and 4 segments of length 1/4.
\[
\sum_{k=1}^{n} S_k^D = \left(\frac{1}{2}\right)^D + \left(\frac{1}{4}\right)^D + \left(\frac{1}{4}\right)^D + \left(\frac{1}{4}\right)^D + \left(\frac{1}{4}\right)^D = 1 \\
\left(\frac{1}{2}\right)^D + 4\left(\frac{1}{4}\right)^D = 1
\]

or D is approximately 1.357.

For more general object shapes (ones with curves and nonplanar faces, for example) we can use the notion of box covering to obtain a fractional dimension. Basically, we count the number of boxes necessary to cover the object at some scale factor of boxes.

And then use

\[
D = \frac{\log n}{\log(1/s)}
\]

My guess for this example is \( n = 92 \) with a scale factor of 1/15, so we get

\[
D = \frac{\log 92}{\log 15} \approx 1.67
\]

for the box counting dimension of the object.
For a fractal curve that lies in the plane, the fractal dimension $D$ is greater than 1. The closer it is to 1, the smoother the curve. If $D=2$, we have a 2D-space-filling Peano curve. It will completely fill some finite region of 2D. If $D > 2$, the curve covers some finite region, self-intersects, and may cover the region an infinite number of times!

**Deterministic self-similar fractals** can be constructed in the way we constructed the Koch curve, starting with an **initiator** (the triangle) and subdividing using a **generator** (the 4-segment replacement curve).

Note that curves with fractal dimension $> 1$ that are constructed this way have infinite length. For example, at each level of subdivision, the Koch curve increases its length by a factor of $4/3$. Since there are infinite levels of subdivision, there’s infinite length.

There’s no rule saying that generators have to be connected. For instance, consider the generator:

```
  ——  ——
```

Which generates a “curve” with two parts of scale factor $1/3$. 
The fractal dimension of this object is

\[ D = \frac{\log n}{\log(1/s)} = \frac{\log 2}{\log 3} \approx 0.631 \]

Here the length keeps getting smaller by a factor of 2/3 at each level of subdivision, so the total length of it is 0! This is called the **Cantor set**.

Another example of a non-connected generator is

where the segment length is 1/8. This has a fractal dimension of

\[ \frac{\log 16}{\log 8} = \frac{4}{3} \approx 1.333 \]

Here’s the second and third levels for that generator:
The generators we’ve looked at so far are mainly one-dimensional (i.e. curves). This means we can easily encode them as strings. Suppose we give the following characters meanings:

- **S**: a segment of unit length
- **l**: a left turn of 60 degrees
- **r**: a right turn of 60 degrees
Then we can encode the Koch generator as

\[ S \rightarrow SISrrSIS \]

And the Koch initiator (the triangle) by

\[ SrrSrrS \]

except that we want the scale factor on the segments to go down by a factor of 1/3 each time we use the generator. So let’s suppose we redefine \( S \) to do that

\[ S: \text{a length } (1/3)^k \text{ segment, where } k \text{ is the number of subdivisions done.} \]

So let’s start with the initiator \( S \) (it’s simpler than \( SrrSrrS \)) and do a couple of levels of subdivision. So 0 levels of subdivision gives us just

\[ S \]

To get to the next level of subdivision, we apply the generator to every \( S \) in the previous level. So for level 1 we get:

\[ SISrrSIS \]
Again we apply the generator to every $S$ to get the second level of subdivision:

$$S\text{SrrSIS} | S\text{SrrSIS} \text{ rr } S\text{SrrSIS} | S\text{SrrSIS}$$

And again for the third level:

$$S\text{SrrSIS} | S\text{SrrSIS} \text{ rr } S\text{SrrSIS} | S\text{SrrSIS} \text{  rr } S\text{SrrSIS} | S\text{SrrSIS} | S\text{SrrSIS}$$

These are essentially driving directions for someone who wants to drive a Koch curve.

For other curves, we may need a letter that stands for moving a distance without drawing a segment. Consider the Cantor set generator:

$$\_\_\_\_\_\_\_\_\_$$

We can model this as $S \Rightarrow STS$, $T \Rightarrow TTT$ where $S$ is a segment of length $(1/3)^k$ as before, and $T$ is a non-drawing movement of length $(1/3)^k$. We can get the first few levels of subdivision as follows:
Those of you who have had compilers or programming languages may recognize that we are doing something very much like a context-free grammar with the strings here. Our system has a set of nonterminal symbols such as $S$ and $T$, a set of terminal symbols such as $l$ and $r$, a start string, and a set of productions, each of which look like:

$$N \Rightarrow \text{some string of terminals and nonterminals}$$

The difference with grammars is that at each stage of a derivation, we replace all of the nonterminals rather than just one. This rule switch makes this a Lindenmeyer system (or L-system for short).

In an L-system, as with grammars, we can have more than one production for a given nonterminal. For instance, we could make a variation on the Koch curve with the productions

$$S \Rightarrow STS \Rightarrow STS TTT STS \Rightarrow STSTTTSTS TTTTTTTTT STSTTTSTS$$
$S \Rightarrow S1SrrS1S$

$S \Rightarrow SrrS1rS$

At each step of the derivation, for each $S$, the L-system would choose one of these two productions to use to replace $S$ with.
With this addition, there’s no longer a single (deterministic) fractal curve being described; we’re describing a whole family of stochastic fractal curves.

Stochastic means random, and indeed we’ve introduced the random choice of which of the two productions to use for S.

Yet one more thing we can do with L-systems helps when we have a branching generator.
It’s a technique called **bracketing**. Basically, you include a set (or sets) of brackets as terminals; these brackets are always introduced in pairs, and open bracket is like a OpenGL **pushmatrix** and close bracket is like **popMatrix**.

The production for this generator is

\[
S \Rightarrow S\{lS\}S\{rS\}S
\]

And here’s the fourth subdivision using that generator.
Here’s another branching generator:

\[ S \Rightarrow S\{IS\}{rS}S\{IS\}{rS}S\{IS\}{rS}S \]

And its third subdivision:

\[ S \Rightarrow U\{IS\}{rS}U\{IS\}{rS}S \]

\[ U \Rightarrow UUU \]
I’ve made the $U$ lines thicker so you can see them. At the next level, all the $S$ segments subdivide, but the $U$ segments don’t. (A $U$ technically subdivides into 3 segments in a straight line, but the net effect of that is to leave it unchanged.)

Even adding a little randomness can make more natural-looking objects:

- $S \Rightarrow U\{lS\}{rS}S$
- $S \Rightarrow US$
- $U \Rightarrow UU$

where $l$ and $r$ rotate 30 degrees each.
S => Ur{rS}S
S => Ul{rS}S
S => UrS
S => UIS
U => U

l and r rotate randomly from 0 to 20 degrees.
We have mainly been looking at 1D (line, curve) generators in the 2D plane. (The exception is the Cantor set, which is a 1D generator in 1D.) We can also look at 1D generators in 3-space.

We could look at 2D generators in 2-space.

That fractal is known as the **Sierpinski gasket**. It has a well-known relationship to Pascal’s triangle:
Or we could look at 2D generators in 3-space. Consider the following generator:

![Diagram of fractal surface generation](image)

where we replace an equilateral triangle by 6 equilateral triangles scaled by a factor of 1/2. That gives this fractal surface a fractal dimension of

\[
D = \frac{\log n}{\log(1/s)} = \frac{\log 6}{\log 2} \approx 2.585
\]

The surface that is generated is the 3D equivalent of the 2D Koch curve.

And it doesn’t stop there. We can look at k-dimensional generators in any space of at least k dimensions.