Fractals

Infinite Details from Simple Rules

Ashrafur Rahman

Adjunct Lecturer

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)

Definition

What are Fractals?

Definition

A **fractal** is a geometric shape containing detailed structure at arbitrarily small scales, usually having a **fractal dimension** strictly exceeding the **topological dimension**.

Key Characteristics:

- **Scale invariance**: Details at every level of magnification
- **Fractal dimension**: More than the *conventional* dimension
- **Infinite complexity**: Generated by simple, recursive rules
  - Like the infinite complexities of nature

What are Fractals?

Definition

A **fractal** is a geometric shape containing detailed structure at arbitrarily small scales, usually having a **fractal dimension** strictly exceeding the **topological dimension**.

Key Characteristics:

- **Scale invariance**: Details at every level of magnification
- **Fractal dimension**: More than the *conventional* dimension
- **Infinite complexity**: Generated by simple, recursive rules
  - Like the infinite complexities of nature

> *"Clouds are not spheres, mountains are not cones, coastlines are not circles..."*
> — **Benoit Mandelbrot**

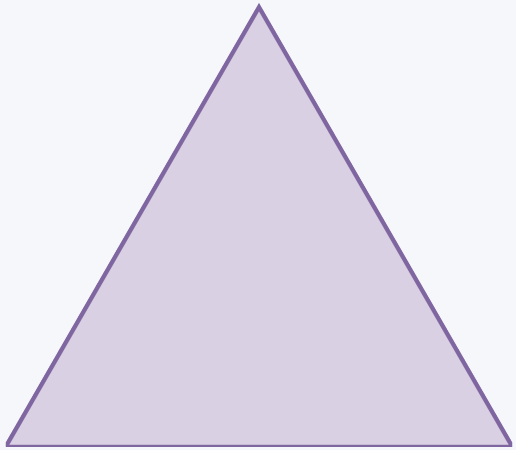
Some Fractals

The Sierpinski Triangle

Let's construct the Sierpinski Triangle:

Construction Process

1. Start with an equilateral triangle



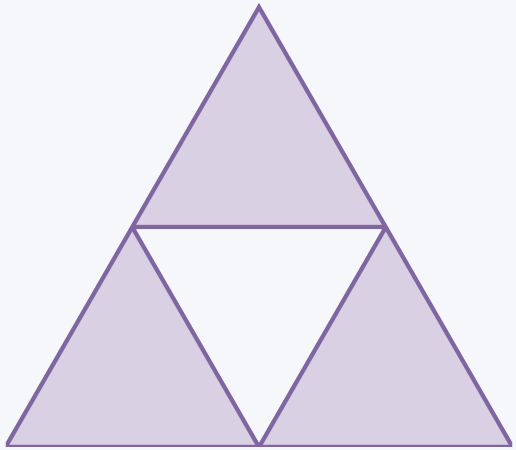
Iteration 0

The Sierpinski Triangle

Let's construct the Sierpinski Triangle:

Construction Process

1. Start with an equilateral triangle
2. Remove the central triangle (triangle connecting midpoints of the sides)



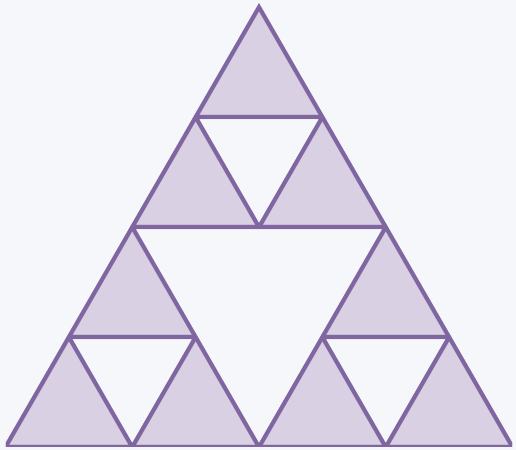
Iteration 1

The Sierpinski Triangle

Let's construct the Sierpinski Triangle:

Construction Process

1. Start with an equilateral triangle
2. Remove the central triangle (triangle connecting midpoints of the sides)
3. Repeat for each triangle



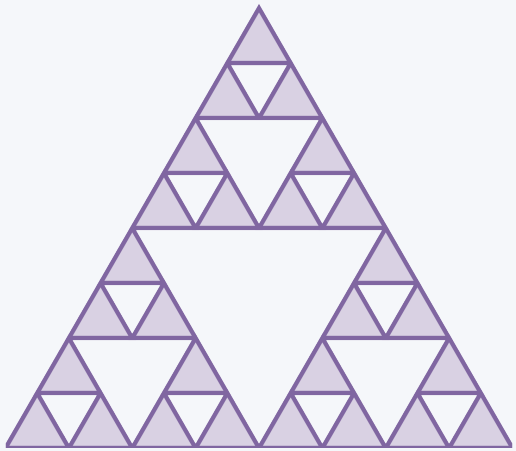
Iteration 2

The Sierpinski Triangle

Let's construct the Sierpinski Triangle:

Construction Process

1. Start with an equilateral triangle
2. Remove the central triangle (triangle connecting midpoints of the sides)
3. Repeat for each triangle
4. After infinite iterations, we get the Sierpinski Triangle



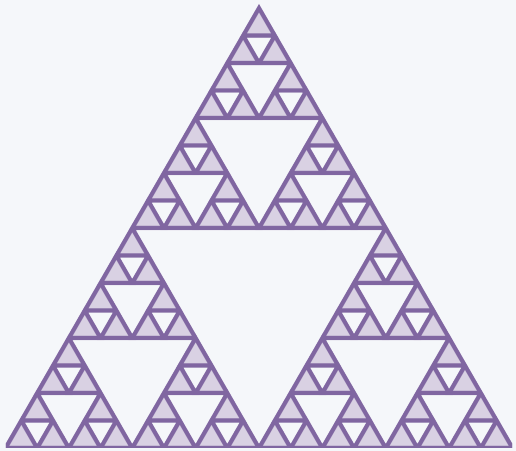
Iteration 3

The Sierpinski Triangle

Let's construct the Sierpinski Triangle:

Construction Process

1. Start with an equilateral triangle
2. Remove the central triangle (triangle connecting midpoints of the sides)
3. Repeat for each triangle
4. After infinite iterations, we get the Sierpinski Triangle



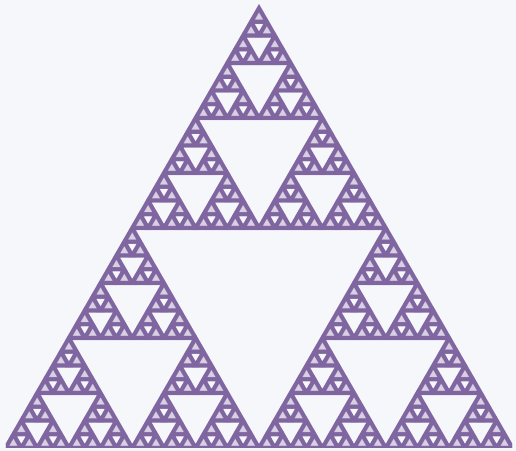
Iteration 4

The Sierpinski Triangle

Let's construct the Sierpinski Triangle:

Construction Process

1. Start with an equilateral triangle
2. Remove the central triangle (triangle connecting midpoints of the sides)
3. Repeat for each triangle
4. After infinite iterations, we get the Sierpinski Triangle



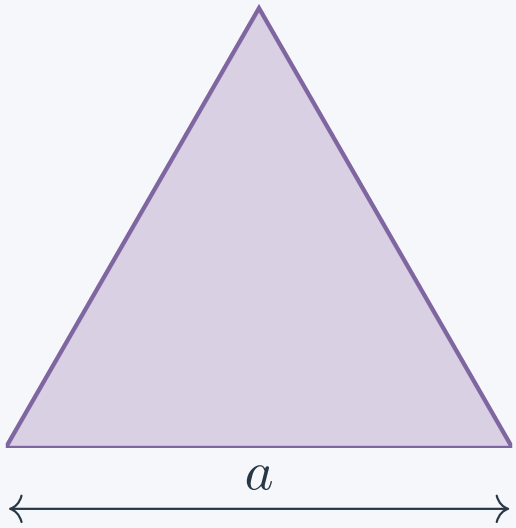
Iteration 5

The Sierpinski Triangle

Now let's find the area of triangle(s):

Area Calculation

$$A_0 = \frac{1}{2} \cdot \sin(60°) \cdot a^2 = \frac{\sqrt{3}}{4} a^2$$



Iteration 0

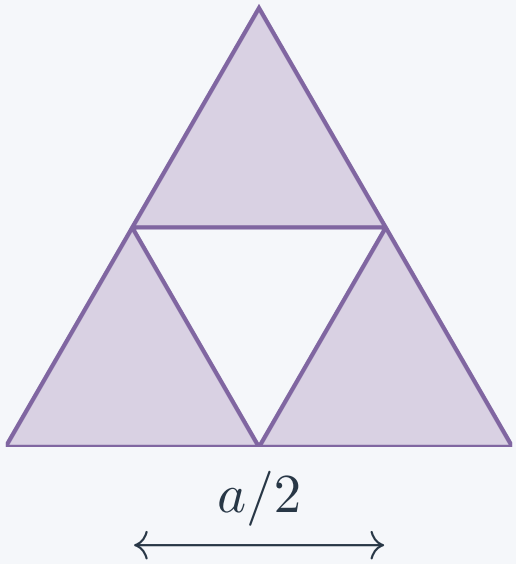
The Sierpinski Triangle

Now let's find the area of triangle(s):

Area Calculation

$$
\begin{aligned}
A_0 &= \frac{1}{2} \cdot \sin(60^\circ) \cdot a^2 = \frac{\sqrt{3}}{4} a^2 \\
A_1 &= A_0 - \frac{1}{2} \cdot \sin(60^\circ) \cdot \left(\frac{a}{2}\right)^2 \\
&= A_0 - \frac{\sqrt{3}}{16} a^2 = A_0 - \frac{1}{4} A_0 \\
&= \frac{3}{4} A_0
\end{aligned}
$$

We cut out one fourth of the area at each step.
So, naturally area is reduced by $\frac{3}{4}$ at each step.



Iteration 1

The Sierpinski Triangle

Now let's find the area of triangle(s):

Area Calculation

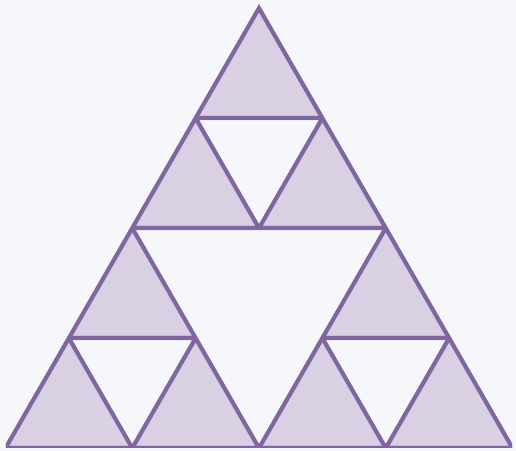
$$A_0 = \frac{1}{2} \cdot \sin(60^\circ) \cdot a^2 = \frac{\sqrt{3}}{4} a^2$$

$$A_1 = \left(1 - \frac{1}{4}\right) A_0 = \frac{3}{4} A_0$$

$$A_2 = \left(1 - \frac{1}{4}\right) A_1 = \left(\frac{3}{4}\right)^2 A_0$$

We cut out one fourth of the area at each step.

So, naturally area is reduced by $\frac{3}{4}$ at each step.



Iteration 2

The Sierpinski Triangle

Now let's find the area of triangle(s):

Area Calculation

$$A_0 = \frac{1}{2} \cdot \sin(60°) \cdot a^2 = \frac{\sqrt{3}}{4} a^2$$

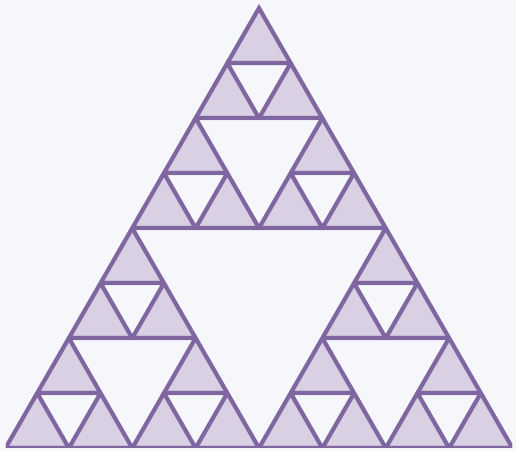
$$A_1 = \left(1 - \frac{1}{4}\right) A_0 = \frac{3}{4} A_0$$

$$A_2 = \left(1 - \frac{1}{4}\right) A_1 = \left(\frac{3}{4}\right)^2 A_0$$

$$\dots$$

$$A_n = \left(1 - \frac{1}{4}\right)^n A_0 = \left(\frac{3}{4}\right)^n A_0$$

$$\lim_{n \to \infty} A_n = 0$$



Iteration 3

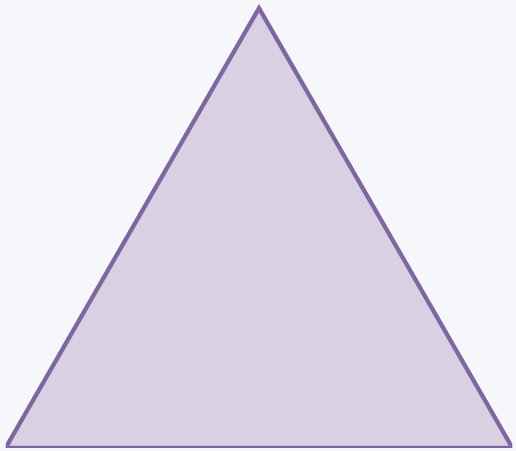
The area of the Sierpinski Triangle is zero even though it is a 2D shape.

The Sierpinski Triangle

Now let's find the perimeter:

Perimeter Calculation

$$P_0 = 3a$$



Iteration 0

The Sierpinski Triangle

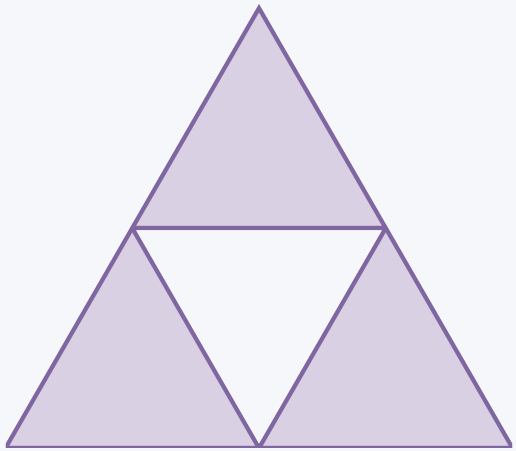
Now let's find the perimeter:

Perimeter Calculation

$$P_0 = 3a$$

$$P_1 = P_0 + 3 \cdot \frac{a}{2} = \frac{9}{2}a = \frac{3}{2}P_0$$

We add half of the previous perimeter at each step. This is because we add the same number of triangles as the last step, but each triangle is half the size of the previous one.
As a result, the perimeter grows by a factor of $\frac{3}{2}$ at each step.



Iteration 1

The Sierpinski Triangle

Now let's find the perimeter:

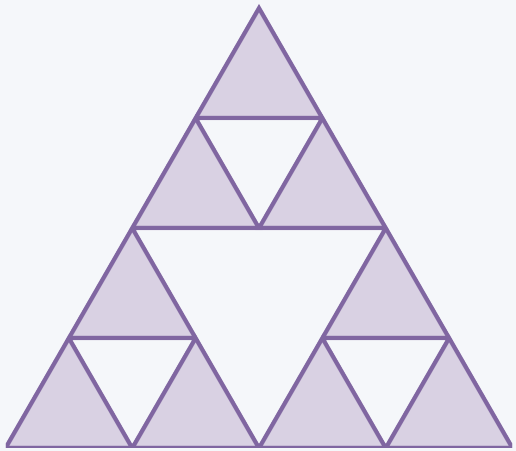
Perimeter Calculation

$$P_0 = 3a$$

$$P_1 = P_0 + 3 \cdot \frac{a}{2} = \frac{9}{2}a = \frac{3}{2}P_0$$

$$P_2 = P_1 + 3 \cdot 3 \cdot \frac{a}{4} = P_1 + \frac{1}{2}P_1 = \frac{3}{2}P_1$$

We add half of the previous perimeter at each step. This is because we add the same number of triangles as the last step, but each triangle is half the size of the previous one.
As a result, the perimeter grows by a factor of $\frac{3}{2}$ at each step.



Iteration 2

The Sierpinski Triangle

Now let's find the perimeter:

Perimeter Calculation

$$P_0 = 3a$$

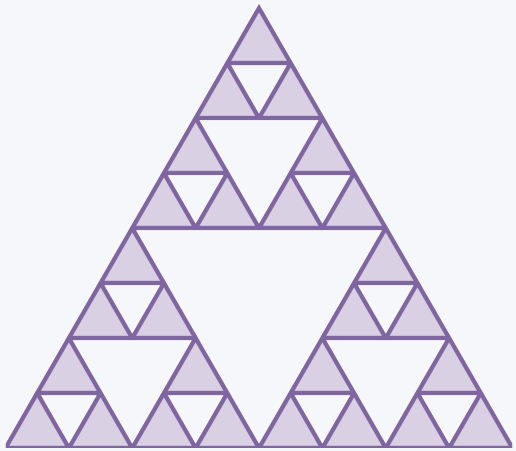
$$P_1 = P_0 + 3 \cdot \frac{a}{2} = \frac{9}{2}a = \frac{3}{2}P_0$$

$$P_2 = P_1 + 3 \cdot 3 \cdot \frac{a}{4} = P_1 + \frac{1}{2}P_1 = \frac{3}{2}P_1$$

$$\dots$$

$$P_n = \frac{3}{2}P_{n-1} = \left(\frac{3}{2}\right)^n P_0$$

$$\lim_{n \to \infty} P_n = \infty$$



Iteration 3

The Sierpinski Triangle has infinite perimeter but zero area.

The Koch Curve

Let's construct the Koch Curve:

Construction Process

1. Start with a line segment of length a

Iteration 0

The Koch Curve

Let's construct the Koch Curve:

Construction Process

1. Start with a line segment of length a
2. Divide it into three equal pieces
3. Remove the middle piece and replace it with two sides of an equilateral triangle (side length $a/3$)

Iteration 1

The Koch Curve

Let's construct the Koch Curve:

Construction Process

1. Start with a line segment of length a
2. Divide it into three equal pieces
3. Remove the middle piece and replace it with two sides of an equilateral triangle (side length $a/3$)
4. Repeat on every straight segment

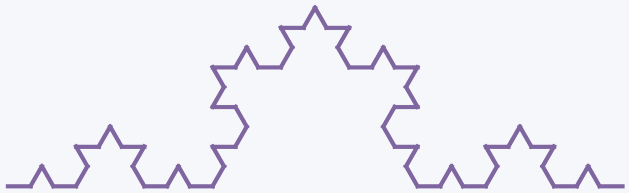
Iteration 2

The Koch Curve

Let's construct the Koch Curve:

Construction Process

1. Start with a line segment of length a
2. Divide it into three equal pieces
3. Remove the middle piece and replace it with two sides of an equilateral triangle (side length $a/3$)
4. Repeat on every straight segment
5. After infinite iterations, we get the Koch Curve



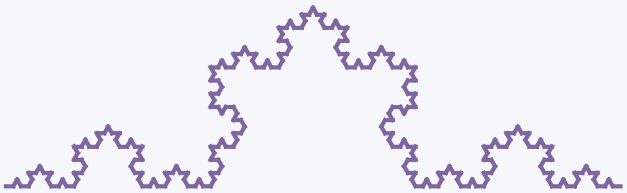
Iteration 3

The Koch Curve

Let's construct the Koch Curve:

Construction Process

1. Start with a line segment of length a
2. Divide it into three equal pieces
3. Remove the middle piece and replace it with two sides of an equilateral triangle (side length $a/3$)
4. Repeat on every straight segment
5. After infinite iterations, we get the Koch Curve



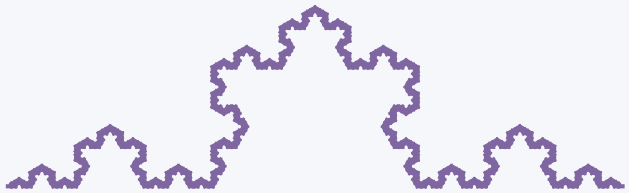
Iteration 4

The Koch Curve

Let's construct the Koch Curve:

Construction Process

1. Start with a line segment of length a
2. Divide it into three equal pieces
3. Remove the middle piece and replace it with two sides of an equilateral triangle (side length $a/3$)
4. Repeat on every straight segment
5. After infinite iterations, we get the Koch Curve



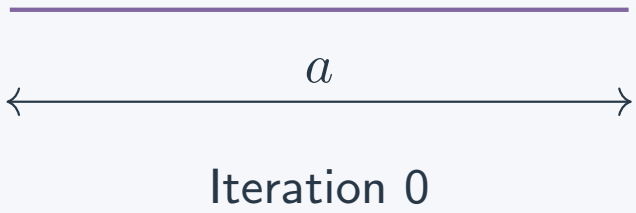
Iteration 5

The Koch Curve

Now let's look at how the length grows:

Length Calculation

$$L_0 = a$$



Iteration 0

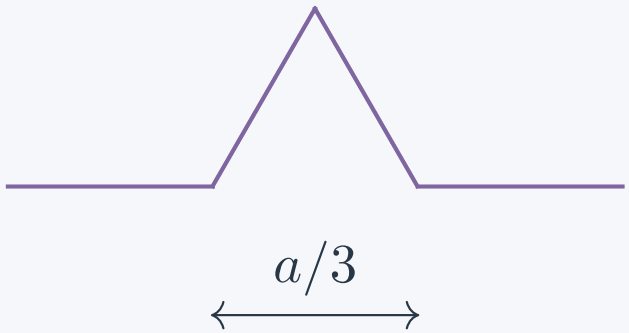
The Koch Curve

Now let's look at how the length grows:

Length Calculation

$$L_0 = a$$

$$L_1 = 4 \cdot \frac{a}{3} = \frac{4}{3} L_0$$



Iteration 1

Each step multiplies the number of segments by 4, each $\frac{1}{3}$ as long, so the total length scales by $\frac{4}{3}$ every iteration.

The Koch Curve

Now let's look at how the length grows:

Length Calculation

$$L_0 = a$$

$$L_1 = 4 \cdot \frac{a}{3} = \frac{4}{3} L_0$$

$$L_2 = 16 \cdot \frac{a}{9} = 4 \frac{L_1}{3} = \frac{4}{3} L_1$$

Iteration 2

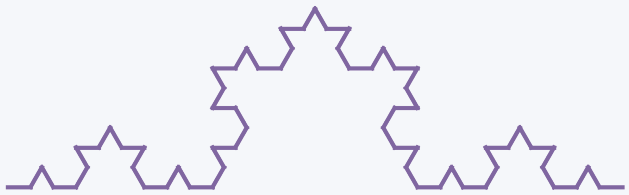
Each step multiplies the number of segments by 4, each $\frac{1}{3}$ as long, so the total length scales by $\frac{4}{3}$ every iteration.

The Koch Curve

Now let's look at how the length grows:

Length Calculation

$$
\begin{aligned}
L_0 &= a \\
L_1 &= 4 \cdot \frac{a}{3} = \frac{4}{3} L_0 \\
L_2 &= 16 \cdot \frac{a}{9} = 4 \frac{L_1}{3} = \frac{4}{3} L_1 \\
&\dots \\
L_n &= \left(\tfrac{4}{3}\right)^n L_0 \\
\lim_{n \to \infty} L_n &= \infty
\end{aligned}
$$



Iteration 3

The Koch curve has infinite length, despite being bounded in a small area.

Fractal Dimension

Fractal Strangeness

Fractals are not like regular geometric shapes.

Interestingly we saw:

- We saw a 2D fractal (Sierpinski) with **zero area**.
- We saw a 1D fractal (Koch) with **infinite length**.
- No matter how many times we zoom in, we always find more detail.

Regular shapes on the other hand:

- A 2D shape has a **finite area**.
- A 1D shape has a **finite length**.
- Zooming in eventually reveals no new details.

Fractal Strangeness

Fractals are not like regular geometric shapes.

Interestingly we saw:

- We saw a 2D fractal (Sierpinski) with **zero area**.
- We saw a 1D fractal (Koch) with **infinite length**.
- No matter how many times we zoom in, we always find more detail.

Regular shapes on the other hand:

- A 2D shape has a **finite area**.
- A 1D shape has a **finite length**.
- Zooming in eventually reveals no new details.

This suggests that fractals have a dimension in between the traditional dimensions.

Dividing Shapes

Normal Shapes

If we divide a normal shape into smaller pieces, it scales with the dimension.

Dividing Shapes

Normal Shapes

If we divide a normal shape into smaller pieces, it scales with the dimension.

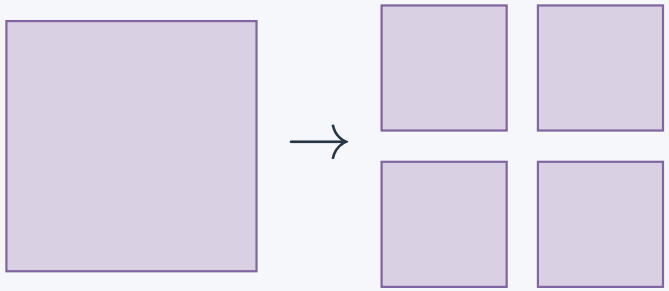
- A **line** divides into 2 pieces, each piece is still a line of scaled by $\frac{1}{2}$.

Dividing Shapes

Normal Shapes

If we divide a normal shape into smaller pieces, it scales with the dimension.

- A **square** divides into $4 = 2^2$ pieces, each piece is still a square scaled by $\frac{1}{2}$.

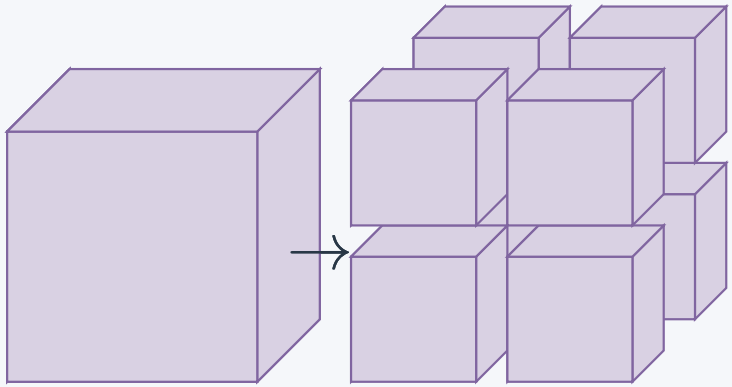


Dividing Shapes

Normal Shapes

If we divide a normal shape into smaller pieces, it scales with the dimension.

- A **cube** divides into $8 = 2^3$ pieces, each piece is still a cube scaled by $\frac{1}{2}$.



Dividing Shapes

From these observations, we can define **dimension** as:

Self-Similarity Dimension

Let,

$$N = \text{number of pieces}$$
$$r = \text{scaling factor}$$
$$D = \text{dimension}$$

Then,

$$N = \left(\frac{1}{r}\right)^D$$
$$\log N = D \cdot \log\left(\frac{1}{r}\right)$$
$$D = \frac{\log N}{\log\left(\frac{1}{r}\right)}$$

Dividing Shapes

Fractals

If we divide a fractal into smaller pieces, it doesn't scale with the dimension.

Dividing Shapes

Fractals

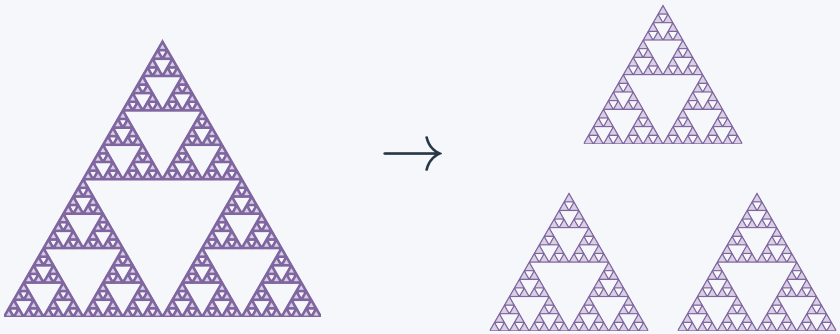
If we divide a fractal into smaller pieces, it doesn't scale with the dimension.

- A **Sierpinski triangle** divides into 3 pieces, each piece is a Sierpinski triangle scaled by $\frac{1}{2}$.

$$N = 3$$

$$r = \frac{1}{2}$$

$$D = \frac{\log 3}{\log 2} \approx 1.585$$



Dividing Shapes

Fractals

If we divide a fractal into smaller pieces, it doesn't scale with the dimension.

- A **Koch curve** divides into 4 pieces, each piece is a Koch curve scaled by $\frac{1}{3}$.

$$N = 4$$

$$r = \frac{1}{3}$$

$$D = \frac{\log 4}{\log 3} \approx 1.262$$

Fractal Dimension Theory

The **Self-Similarity Dimension** works for self-similar fractals. A more general definition is the **Box-Counting Dimension**.

Box-Counting Dimension

The fractal dimension D is defined as:

$$D = \lim_{\varepsilon \to 0} \frac{\log N(\varepsilon)}{\log(1/\varepsilon)}$$

where $N(\varepsilon)$ is the number of boxes of size ε needed to cover the fractal.

Please check this 3blue1brown video for an excellent explanation of fractal dimension.

Nature's Fractal Dimensions

Fractal dimension measures the **complexity** of a shape. We usually assume that when zoomed in things become smooth. Natural objects like fractals, exhibit complexity even at small scales.

Natural Fractals

- Coastline of Britain: $D \approx 1.25$
- Clouds: $D \approx 2.35$
- Lightning: $D \approx 1.7$
- Lung bronchi: $D \approx 2.97$

Coastline of Britain

More Fractals

The Dragon Curve

Let's construct the Dragon Curve:

Construction Process

1. Start with a line segment

Iteration 0

The Dragon Curve

Let's construct the Dragon Curve:

Construction Process

1. Start with a line segment
2. Fold the line in half at a 90° angle

Iteration 1

The Dragon Curve

Let's construct the Dragon Curve:

Construction Process

1. Start with a line segment
2. Fold the line in half at a 90° angle
3. Repeat the folding process on the entire curve

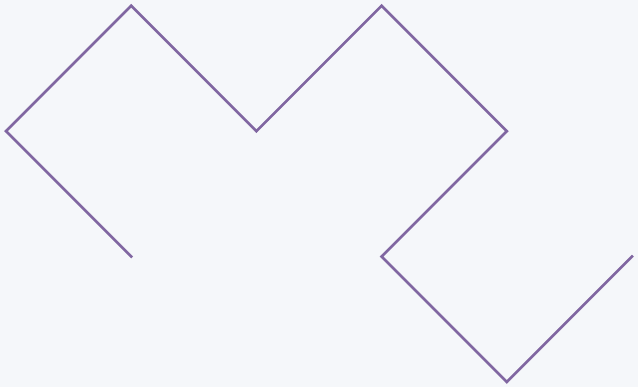
Iteration 2

The Dragon Curve

Let's construct the Dragon Curve:

Construction Process

1. Start with a line segment
2. Fold the line in half at a 90° angle
3. Repeat the folding process on the entire curve
4. After infinite iterations, we get the Dragon Curve



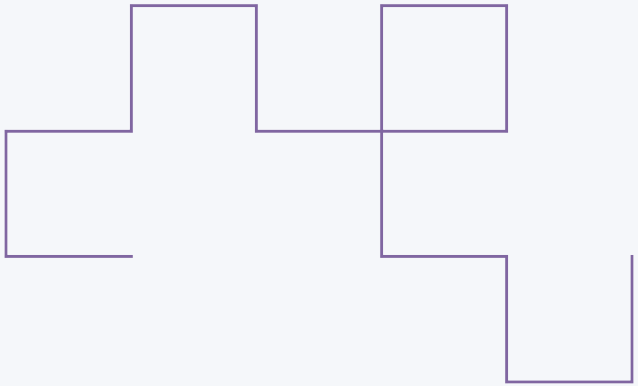
Iteration 3

The Dragon Curve

Let's construct the Dragon Curve:

Construction Process

1. Start with a line segment
2. Fold the line in half at a 90° angle
3. Repeat the folding process on the entire curve
4. After infinite iterations, we get the Dragon Curve



Iteration 4

The Dragon Curve

Let's construct the Dragon Curve:

Construction Process

1. Start with a line segment
2. Fold the line in half at a 90° angle
3. Repeat the folding process on the entire curve
4. After infinite iterations, we get the Dragon Curve

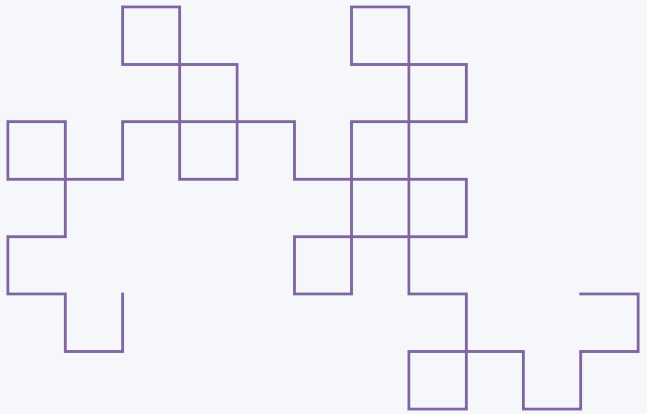
Iteration 5

The Dragon Curve

Let's construct the Dragon Curve:

Construction Process

1. Start with a line segment
2. Fold the line in half at a 90° angle
3. Repeat the folding process on the entire curve
4. After infinite iterations, we get the Dragon Curve



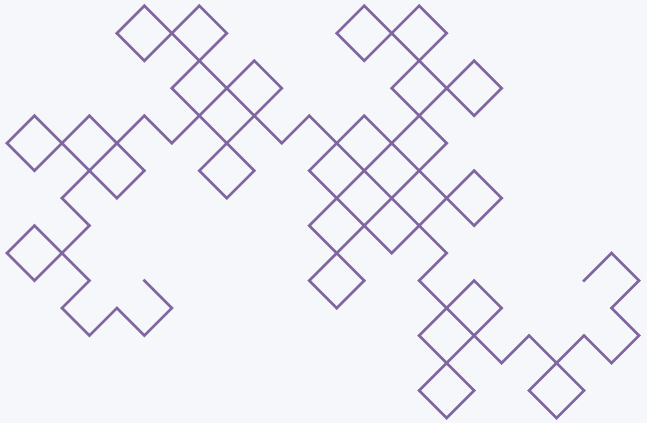
Iteration 6

The Dragon Curve

Let's construct the Dragon Curve:

Construction Process

1. Start with a line segment
2. Fold the line in half at a 90° angle
3. Repeat the folding process on the entire curve
4. After infinite iterations, we get the Dragon Curve



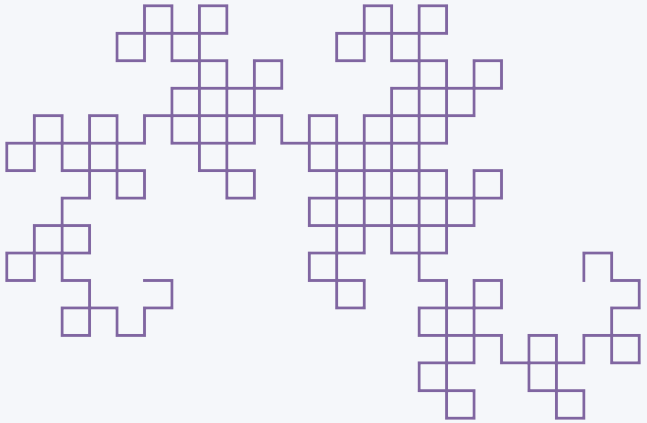
Iteration 7

The Dragon Curve

Let's construct the Dragon Curve:

Construction Process

1. Start with a line segment
2. Fold the line in half at a 90° angle
3. Repeat the folding process on the entire curve
4. After infinite iterations, we get the Dragon Curve



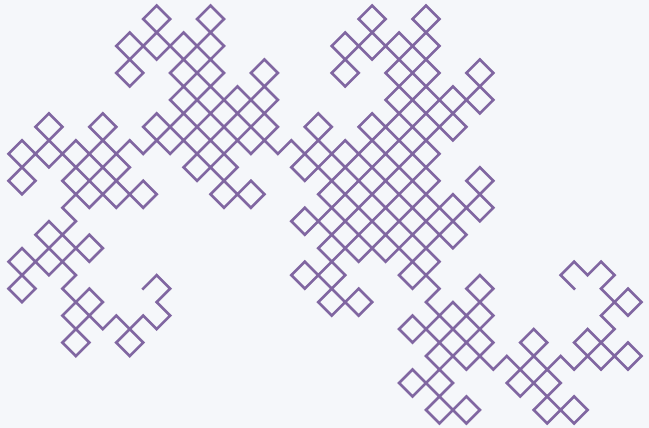
Iteration 8

The Dragon Curve

Let's construct the Dragon Curve:

Construction Process

1. Start with a line segment
2. Fold the line in half at a 90° angle
3. Repeat the folding process on the entire curve
4. After infinite iterations, we get the Dragon Curve



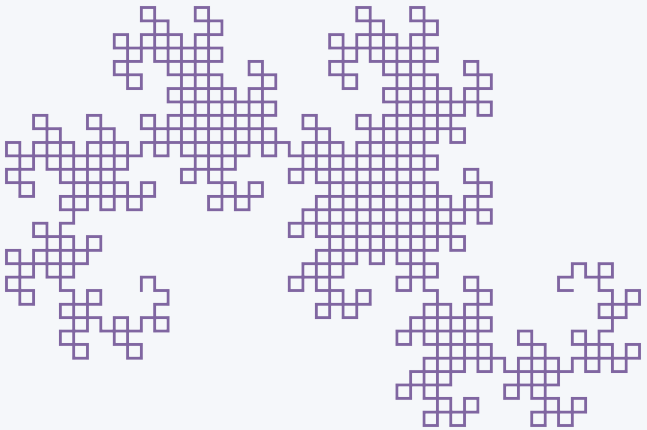
Iteration 9

The Dragon Curve

Let's construct the Dragon Curve:

Construction Process

1. Start with a line segment
2. Fold the line in half at a 90° angle
3. Repeat the folding process on the entire curve
4. After infinite iterations, we get the Dragon Curve



Iteration 10

The Dragon Curve

Now let's find the fractal dimension:

Dimension Calculation

The Dragon Curve can be divided into 2 copies of itself, each scaled by $\frac{1}{\sqrt{2}}$.

Iteration 1

The Dragon Curve

Now let's find the fractal dimension:

Dimension Calculation

The Dragon Curve can be divided into 2 copies of itself, each scaled by $\frac{1}{\sqrt{2}}$.

$$N = 2$$

$$r = \frac{1}{\sqrt{2}}$$

Iteration 2

The Dragon Curve

Now let's find the fractal dimension:

Dimension Calculation

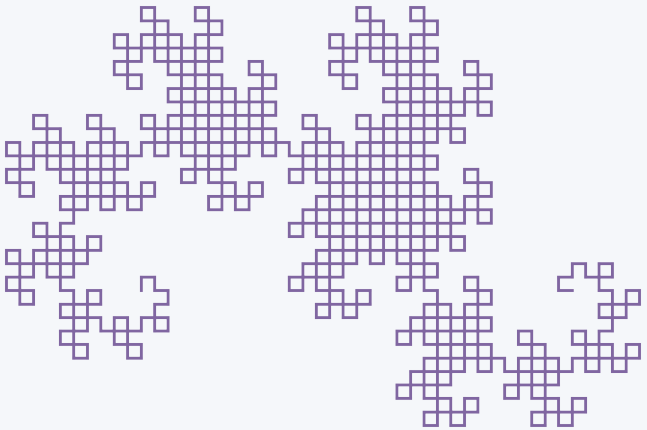
The Dragon Curve can be divided into 2 copies of itself, each scaled by $\frac{1}{\sqrt{2}}$.

$$N = 2$$

$$r = \frac{1}{\sqrt{2}}$$

$$D = \frac{\log N}{\log(1/r)} = \frac{\log 2}{\log \sqrt{2}}$$

$$= \frac{\log 2}{\frac{1}{2}\log 2} = 2$$



Iteration 10

The Dragon Curve has a fractal dimension of exactly 2. Check the animation.

The Hilbert Curve

Let's construct the Hilbert Curve:

Construction Process

1. Start with a U-shaped curve

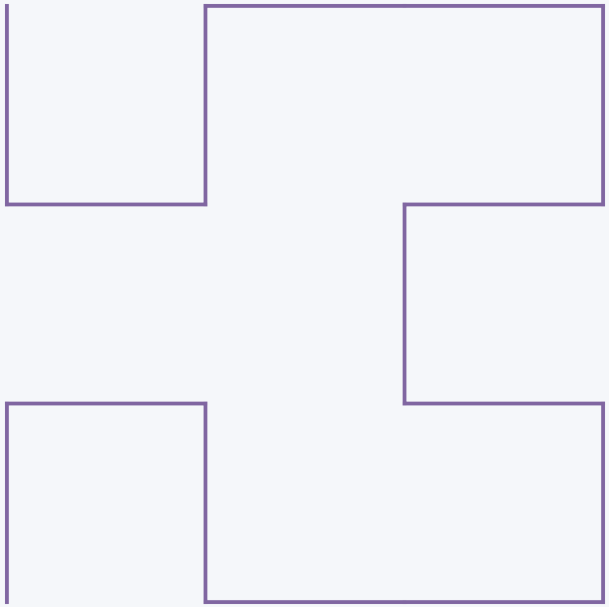
Order 1

The Hilbert Curve

Let's construct the Hilbert Curve:

Construction Process

1. Start with a U-shaped curve
2. Replace each *corner* with a smaller copy of the entire curve



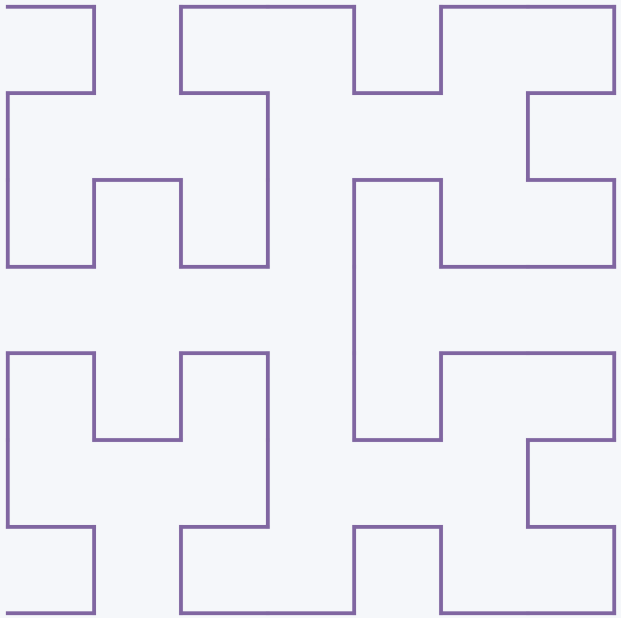
Order 2

The Hilbert Curve

Let's construct the Hilbert Curve:

Construction Process

1. Start with a U-shaped curve
2. Replace each *corner* with a smaller copy of the entire curve
3. Connect the curves appropriately



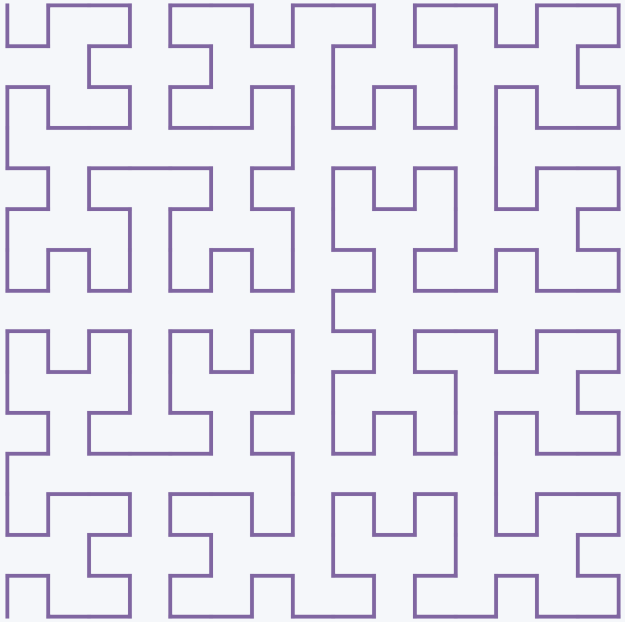
Order 3

The Hilbert Curve

Let's construct the Hilbert Curve:

Construction Process

1. Start with a U-shaped curve
2. Replace each *corner* with a smaller copy of the entire curve
3. Connect the curves appropriately
4. After infinite iterations, we get the Hilbert Curve



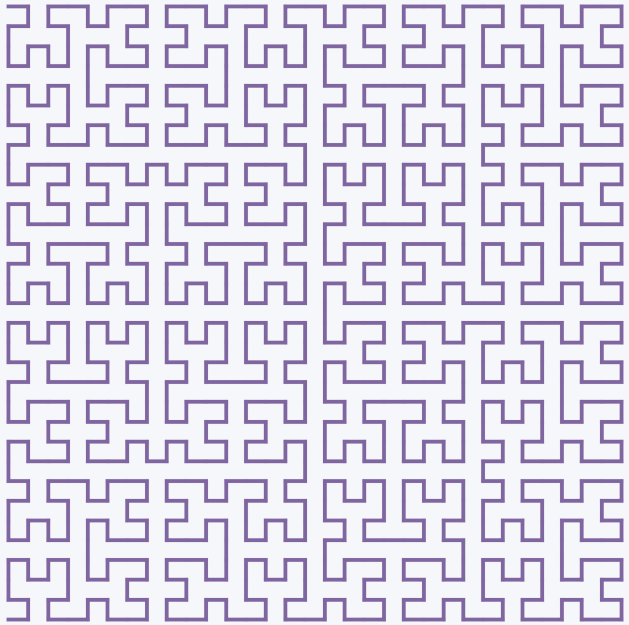
Order 4

The Hilbert Curve

Let's construct the Hilbert Curve:

Construction Process

1. Start with a U-shaped curve
2. Replace each *corner* with a smaller copy of the entire curve
3. Connect the curves appropriately
4. After infinite iterations, we get the Hilbert Curve



Order 5

The Hilbert Curve

Now let's find the fractal dimension:

Dimension Calculation

The Hilbert Curve can be divided into 4 copies of itself, each scaled by $\frac{1}{2}$.

Order 1

The Hilbert Curve

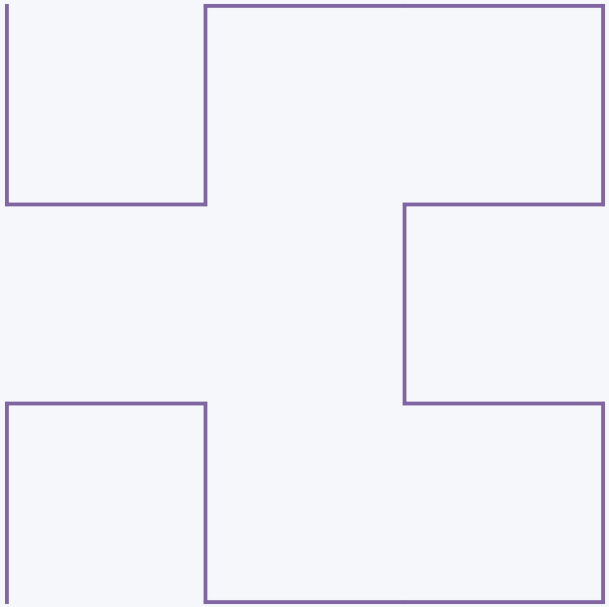
Now let's find the fractal dimension:

Dimension Calculation

The Hilbert Curve can be divided into 4 copies of itself, each scaled by $\frac{1}{2}$.

$$N = 4$$

$$r = \frac{1}{2}$$



Order 2

The Hilbert Curve

Now let's find the fractal dimension:

Dimension Calculation

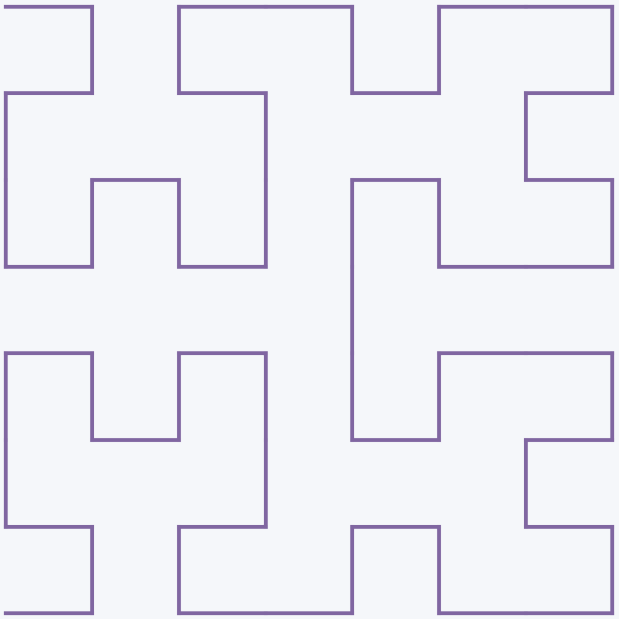
The Hilbert Curve can be divided into 4 copies of itself, each scaled by $\frac{1}{2}$.

$$N = 4$$

$$r = \frac{1}{2}$$

$$D = \frac{\log N}{\log(1/r)} = \frac{\log 4}{\log 2}$$

$$= \frac{2\log 2}{\log 2} = 2$$



Order 3

The Hilbert Curve has a fractal dimension of exactly 2.

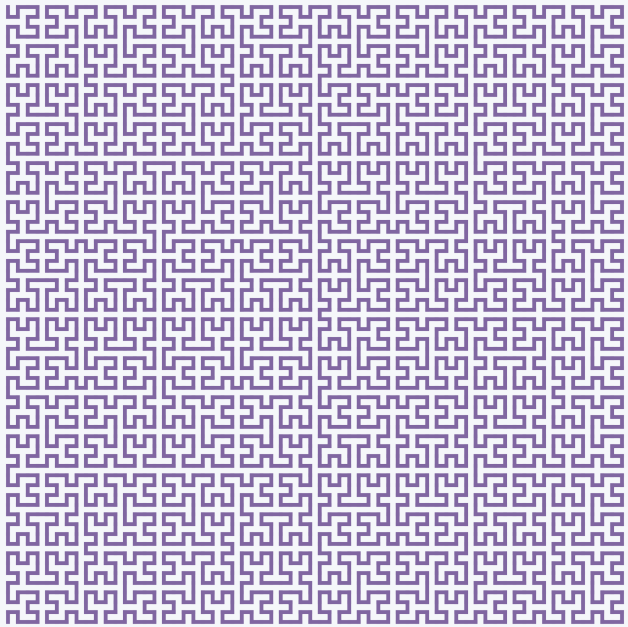
Space-Filling Curves

Space-Filling Curves

A **space-filling curve** is a curve whose range contains the entire 2D area (or higher dimensional analogue).

Key Properties:

- They are **continuous** curves that pass through every point in an area
- They have **fractal dimension 2** - they completely fill 2D space
- Examples: Hilbert curve, Dragon curve
- They provide a **mapping** from 1D to 2D space



Hilbert Curve filling a square

L-Systems

Introduction to L-Systems

L-Systems, or **Lindenmayer Systems** is a way to generate fractals using **string rewriting**.

Each character in the generated string represents a **drawing command**. To generate a higher depth fractal, we apply a set of **rules** to the string repeatedly.

Lindenmayer Systems (L-Systems)

Components:

- **Axiom**: Starting string (initial state)
- **Rules**: String replacement rules applied simultaneously
- **Interpretation**: How to draw/interpret the string

L-Systems

Basic Symbols

Movement Commands:

- **X**: Move forward
- $+$: Turn right by angle δ
- $-$: Turn left by angle δ

State Commands:

- **[**: Push current state
- **]**: Pop and restore state

Try out the L-System Renderer to create your own L-Systems!

L-Systems

Basic Symbols

Movement Commands:

- **X**: Move forward
- **+**: Turn right by angle δ
- **−**: Turn left by angle δ

State Commands:

- **[**: Push current state
- **]**: Pop and restore state

Try out the L-System Renderer to create your own L-Systems!

Example 1

Axiom: F

Rule: $F \rightarrow F + F - -F + F$

Angle: $\delta = 60°$

Generations:

- $n = 0 : F$

L-Systems

Basic Symbols

Movement Commands:

- **X**: Move forward
- **+**: Turn right by angle δ
- **−**: Turn left by angle δ

State Commands:

- **[**: Push current state
- **]**: Pop and restore state

Try out the L-System Renderer to create your own L-Systems!

Example 1

Axiom: F

Rule: $F \to F + F - -F + F$

Angle: $\delta = 60°$

Generations:

- $n = 1 : F + F - -F + F$

L-Systems

Basic Symbols

Movement Commands:

- **X**: Move forward
- $+$: Turn right by angle δ
- $-$: Turn left by angle δ

State Commands:

- **[**: Push current state
- **]**: Pop and restore state

Try out the L-System Renderer to create your own L-Systems!

Example 1

Axiom: F

Rule: $F \rightarrow F + F - - F + F$

Angle: $\delta = 60°$

Generations:

- $n = 2 : F+F--F+F+$
  $F+F--F+F--F+F-$
  $-F+F+F+F--F+F$

This produces the Koch curve!

L-Systems

Basic Symbols

Movement Commands:

- **X**: Move forward
- $+$: Turn right by angle δ
- $-$: Turn left by angle δ

State Commands:

- **[**: Push current state
- **]**: Pop and restore state

Try out the L-System Renderer to create your own L-Systems!

Example 2

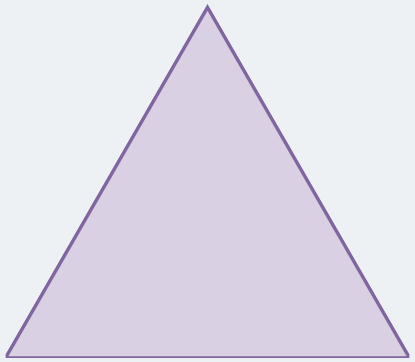
Axiom: $F - G - G$

Rules: $F \to F-G+F+G-F$, $G \to GG$

Angle: $\delta = 120^\circ$

Generations:

- $n = 0 : F - G - G$



L-Systems

Basic Symbols

Movement Commands:

- **X**: Move forward
- $+$: Turn right by angle δ
- $-$: Turn left by angle δ

State Commands:

- **[**: Push current state
- **]**: Pop and restore state

Try out the L-System Renderer to create your own L-Systems!

Example 2

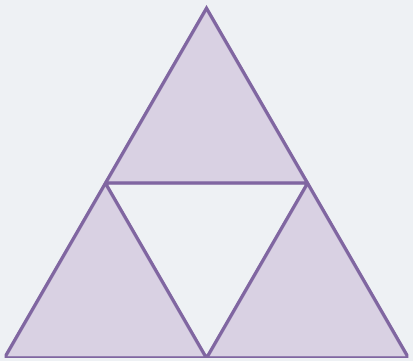
Axiom: $F - G - G$

Rules: $F \to F - G + F + G - F$, $G \to GG$

Angle: $\delta = 120°$

Generations:

- $n = 1 : F - G + F + G - F - GG - GG$



L-Systems

Basic Symbols

Movement Commands:

- **X**: Move forward
- **+**: Turn right by angle δ
- **−**: Turn left by angle δ

State Commands:

- **[**: Push current state
- **]**: Pop and restore state

Try out the L-System Renderer to create your own L-Systems!

Example 2

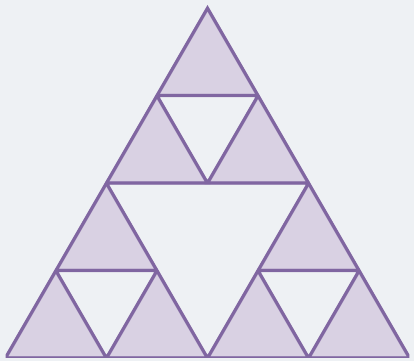
Axiom: $F - G - G$

Rules: $F \to F - G + F + G - F$, $G \to GG$

Angle: $\delta = 120°$

Generations:

- $n = 2 : F - G + F + G - F - GG + F - G + F + G - F + GG - F - G + F + G - F - GGGG - GGGG$



L-Systems

Basic Symbols

Movement Commands:

- **X**: Move forward
- **+**: Turn right by angle δ
- **−**: Turn left by angle δ

State Commands:

- **[**: Push current state
- **]**: Pop and restore state

Try out the L-System Renderer to create your own L-Systems!

Example 3

Axiom: F

Rules: $F \to T[-F][+F]$, $T \to TT$

Angle: $\delta = 30°$

Generations:

- $n = 0 : F$

L-Systems

Basic Symbols

Movement Commands:

- **X**: Move forward
- **+**: Turn right by angle δ
- **−**: Turn left by angle δ

State Commands:

- **[**: Push current state
- **]**: Pop and restore state

Try out the L-System Renderer to create your own L-Systems!

Example 3

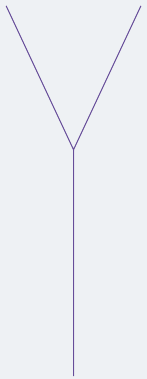
Axiom: F

Rules: $F \to T[-F][+F]$, $T \to TT$

Angle: $\delta = 30°$

Generations:

- $n = 1 : T[-F][+F]$



L-Systems

Basic Symbols

Movement Commands:

- **X**: Move forward
- $+$: Turn right by angle δ
- $-$: Turn left by angle δ

State Commands:

- **[**: Push current state
- **]**: Pop and restore state

Try out the L-System Renderer to create your own L-Systems!

Example 3

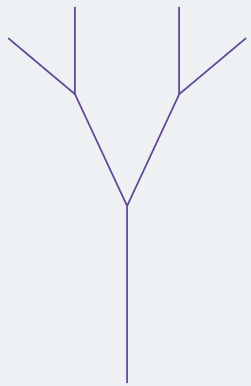
Axiom: F

Rules: $F \to T[-F][+F]$, $T \to TT$

Angle: $\delta = 30°$

Generations:

- $n = 2$: $TT[-T[-F][+F]][+T[-F][+F]]$



L-Systems

Basic Symbols

Movement Commands:

- **X**: Move forward
- **+**: Turn right by angle δ
- **−**: Turn left by angle δ

State Commands:

- **[**: Push current state
- **]**: Pop and restore state

Try out the L-System Renderer to create your own L-Systems!

Example 3

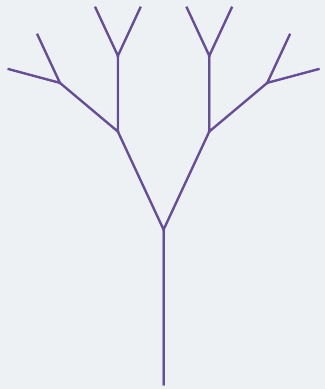
Axiom: F

Rules: $F \to T[-F][+F]$, $T \to TT$

Angle: $\delta = 30°$

Generations:

- $n = 3 : TTTT[\ldots]$



This produces a binary tree!

Fractals in Nature

Fractals and Nature

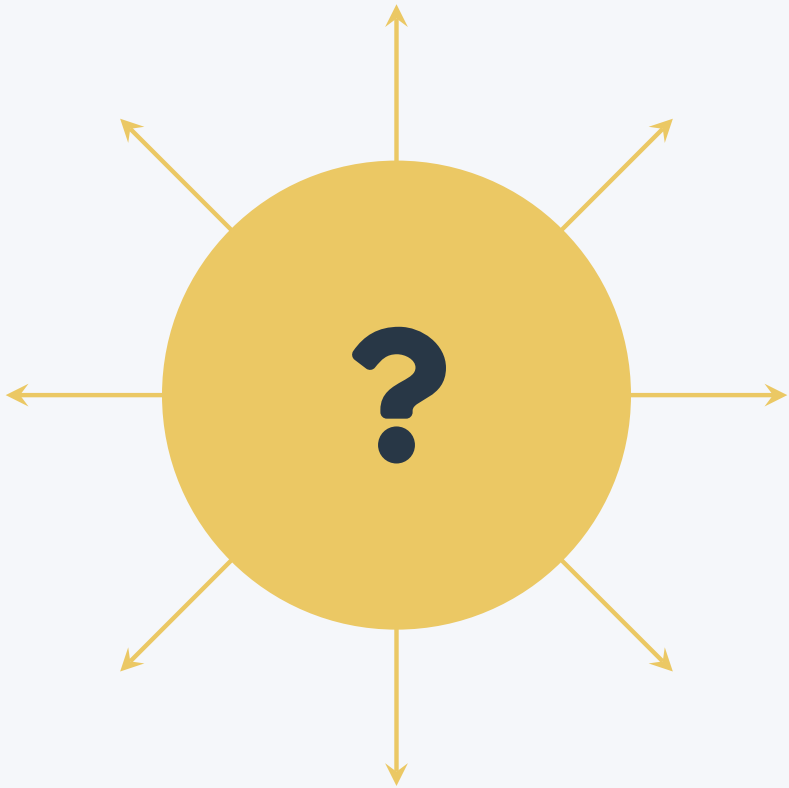
Fractals occur naturally in many forms, from the branching of trees to the structure of snowflakes. Therefore, fractals are used to model natural phenomena, like the branching of trees or the formation of clouds.

Fractals in Nature

Trees generated by L-Systems

Questions & Discussion

Questions?



References & Further Reading

- Peter Shirley and Steve Marschner et al. *Fundamentals of Computer Graphics (4th Edition)*. CRC Press, 2016.
  Available as PDF
- Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation (4th Edition)*. Morgan Kaufmann, 2023.
  Availabe online
- Peter Shirley. *Ray Tracing in One Weekend*. Self-published, 2016–2020.
  Project Website
- MIT OpenCourseWare: 6.837 Computer Graphics.
  ocw.mit.edu/6-837
- Scratchapixel: Learn Computer Graphics Programming.
  scratchapixel.com