Hình học Fractal

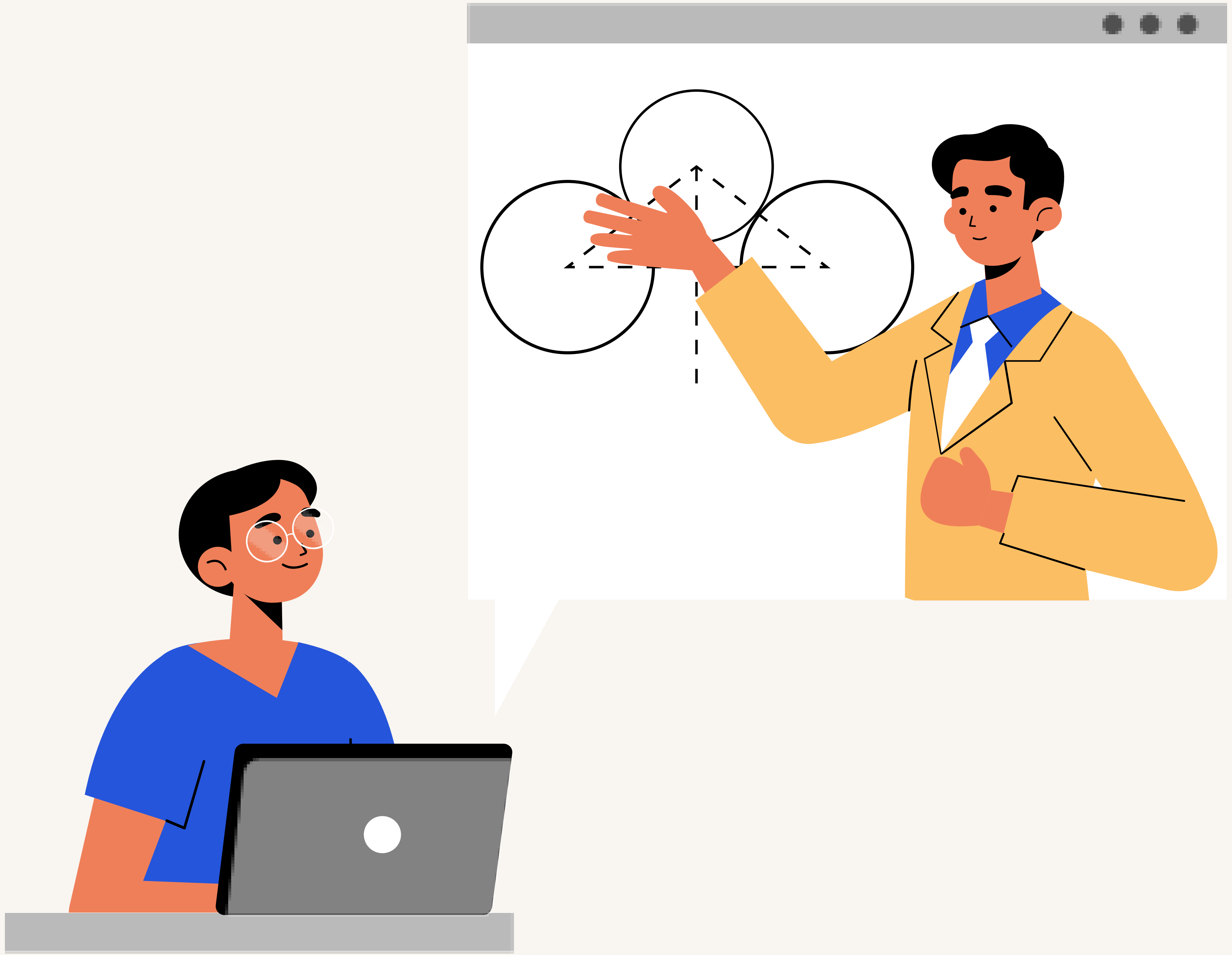
NỘI DUNG TRÌNH BÀY

01
Giới thiệu hình học fractal

02
Hình học fractal từ số thực

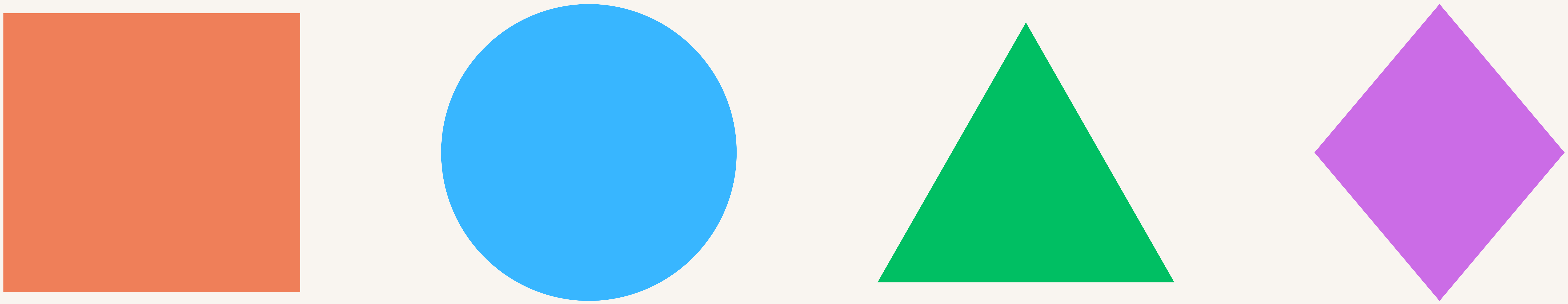
03
Hình học fractal từ số phức

I. GIỚI THIỆU HÌNH HỌC FRACTAL

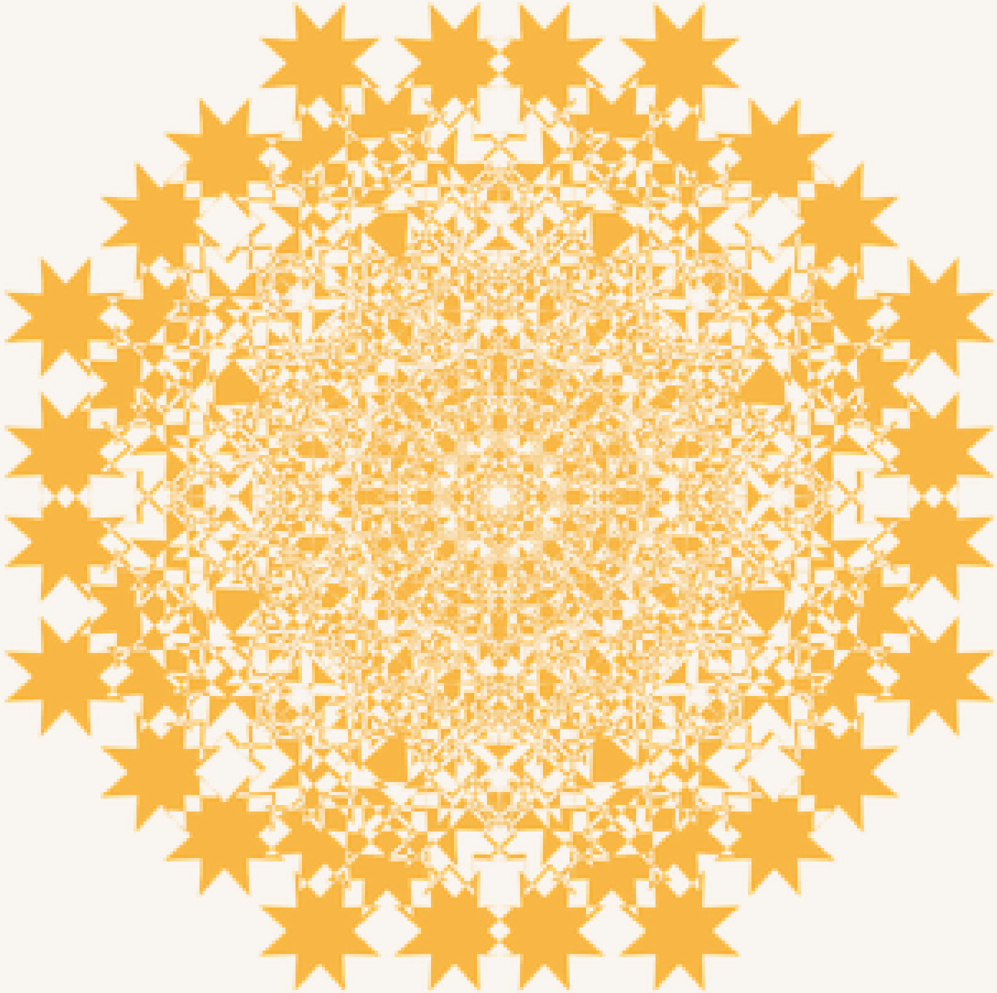


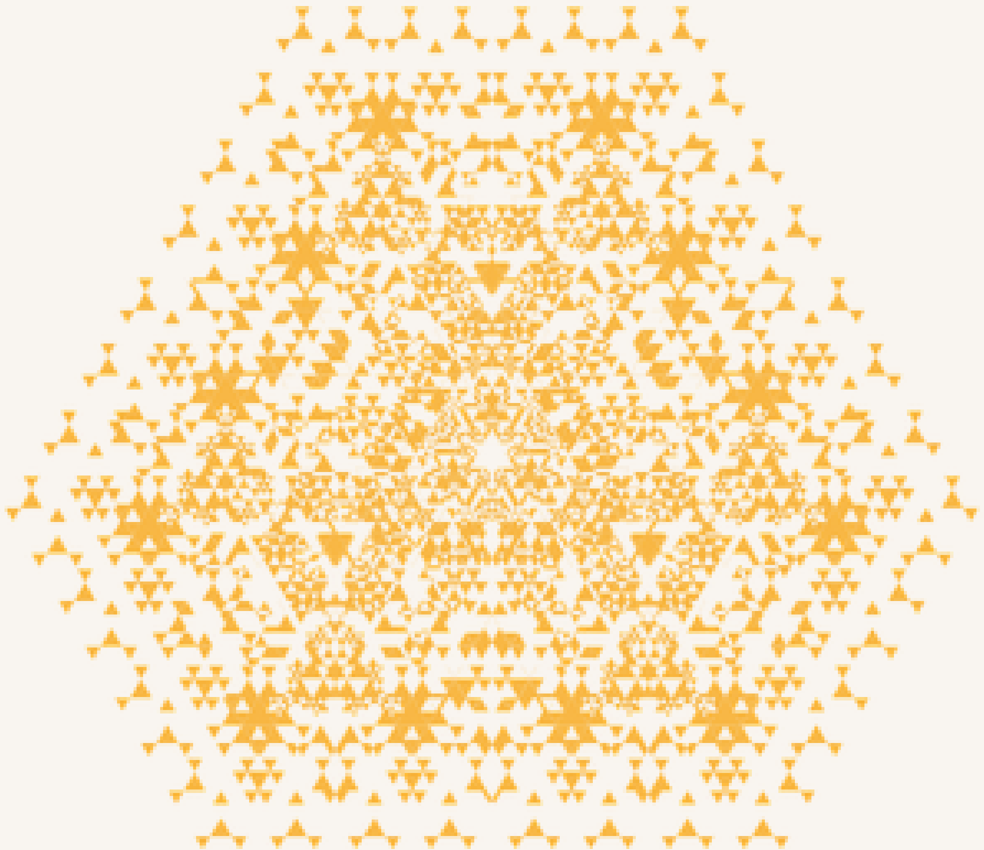
1. Đặt vấn đề

Chúng ta vẫn thường nghĩ hình học là những hình vẽ cứng nhắc, chỉ là những hình phẳng, hình khối.



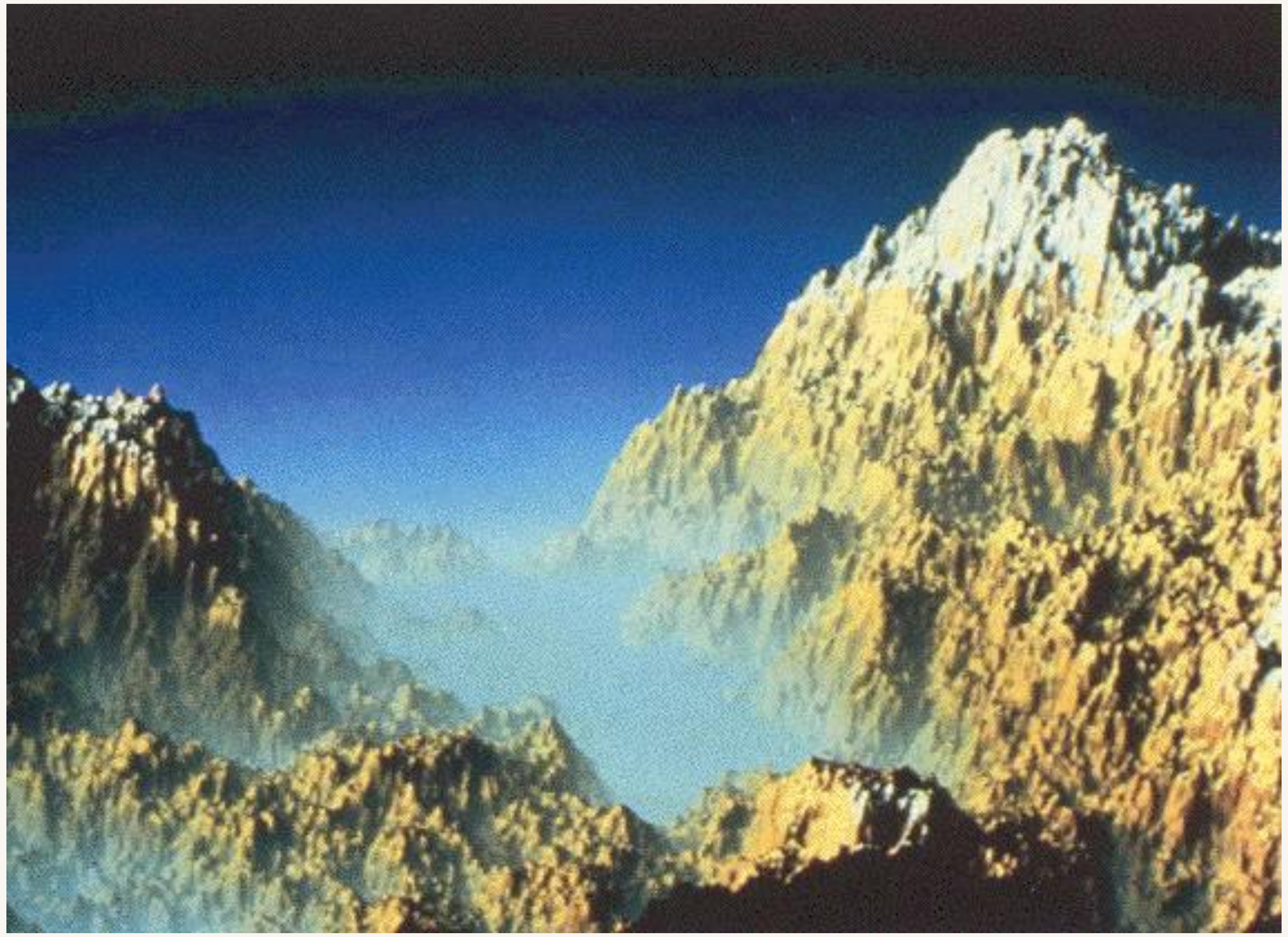
Tuy nhiên, vẫn có rất nhiều hình dạng độc đáo xuất hiện trong đời sống hằng ngày của chúng ta. Dường như chúng tạo thành bởi quy luật nào đó?



2. Lịch sử hình thành hình học fractal

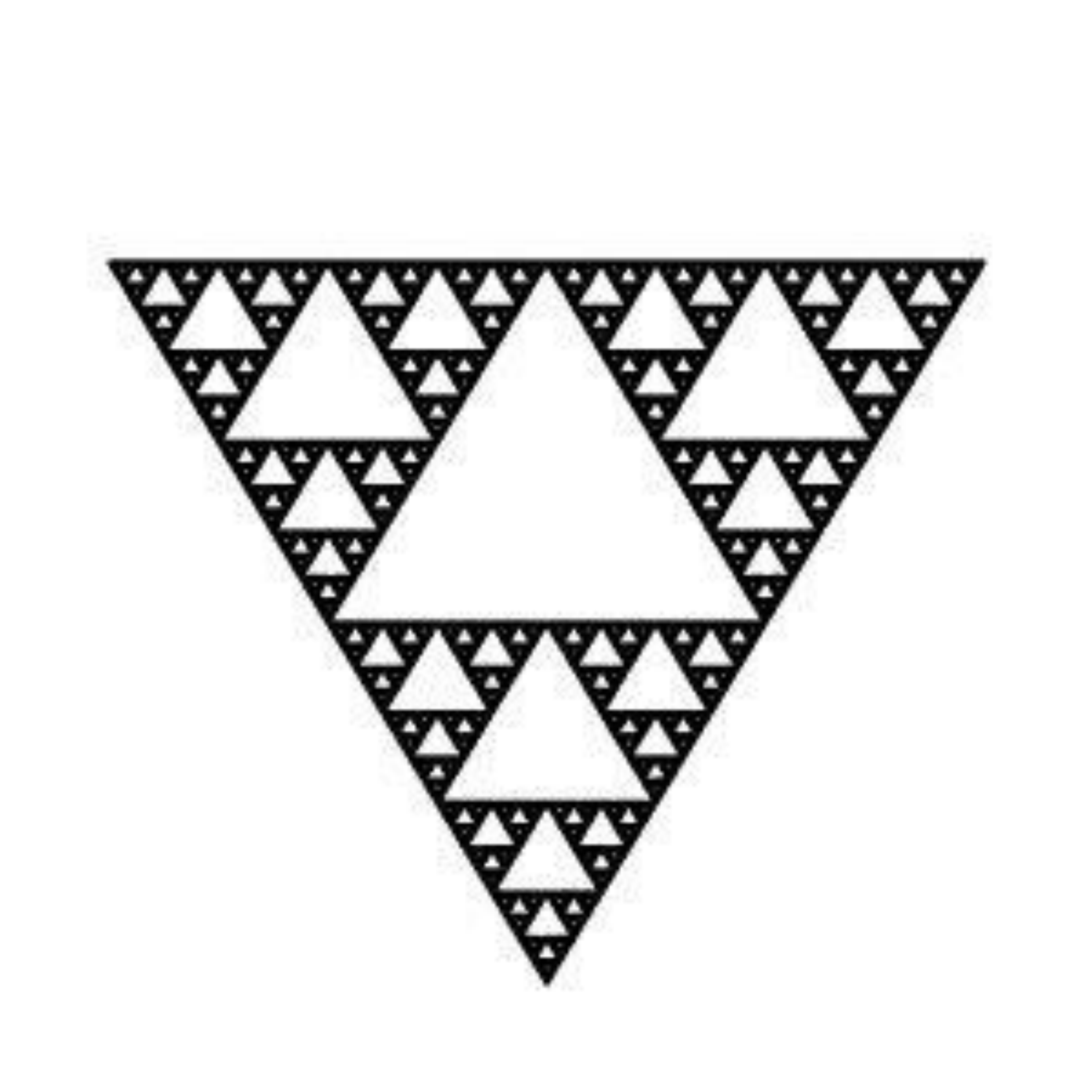
Hình học Euclid nhìn nhận mọi vật dưới dạng "đều đặn", "trơn nhẵn", không mô tả được thế giới tự nhiên, vốn xù xì, gồ ghề như ngọn núi, đường bờ biển



"Những đám mây không phải là những quả cầu, những ngọn núi không phải những hình nón, những bờ biển không phải là những đường tròn". Từ cảm nhận trực quan đó, nhà toán học thiên tài Mandelbrot nảy sinh ý tưởng về "Hình học của tự nhiên". Đó chính là hình học fractal (hình học phân dạng)

3. Khái niệm

Một hình phân dạng là một hình “gai góc lởm chởm” hoặc một hình có một dạng hình học nào đó, có thể được chia ra thành nhiều mảnh nhỏ, mỗi mảnh nhỏ - dù ở kích cỡ nào – đều đồng dạng (hoặc gần như đồng dạng) với toàn thể hình => tự đồng dạng

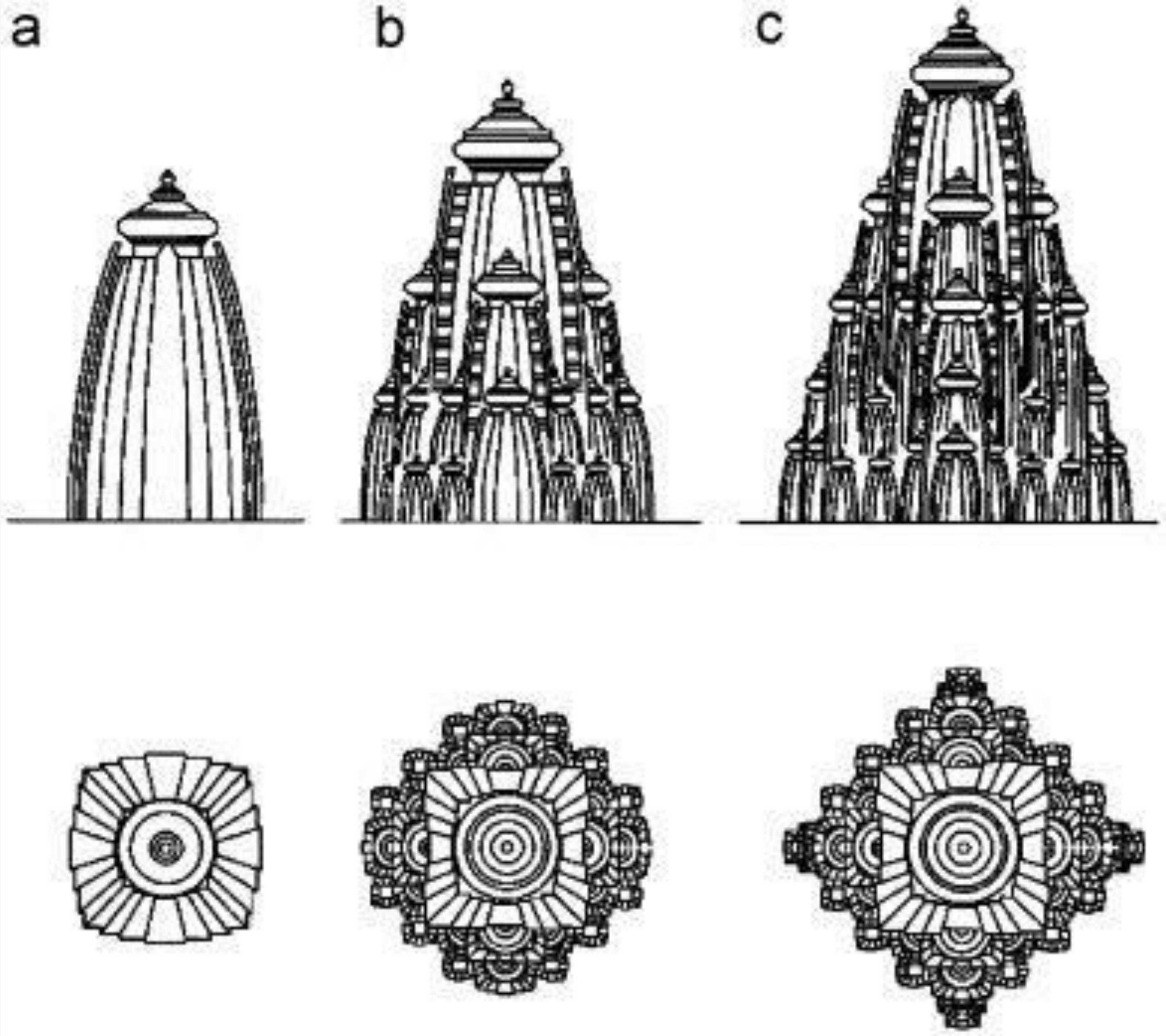


4. Tính chất

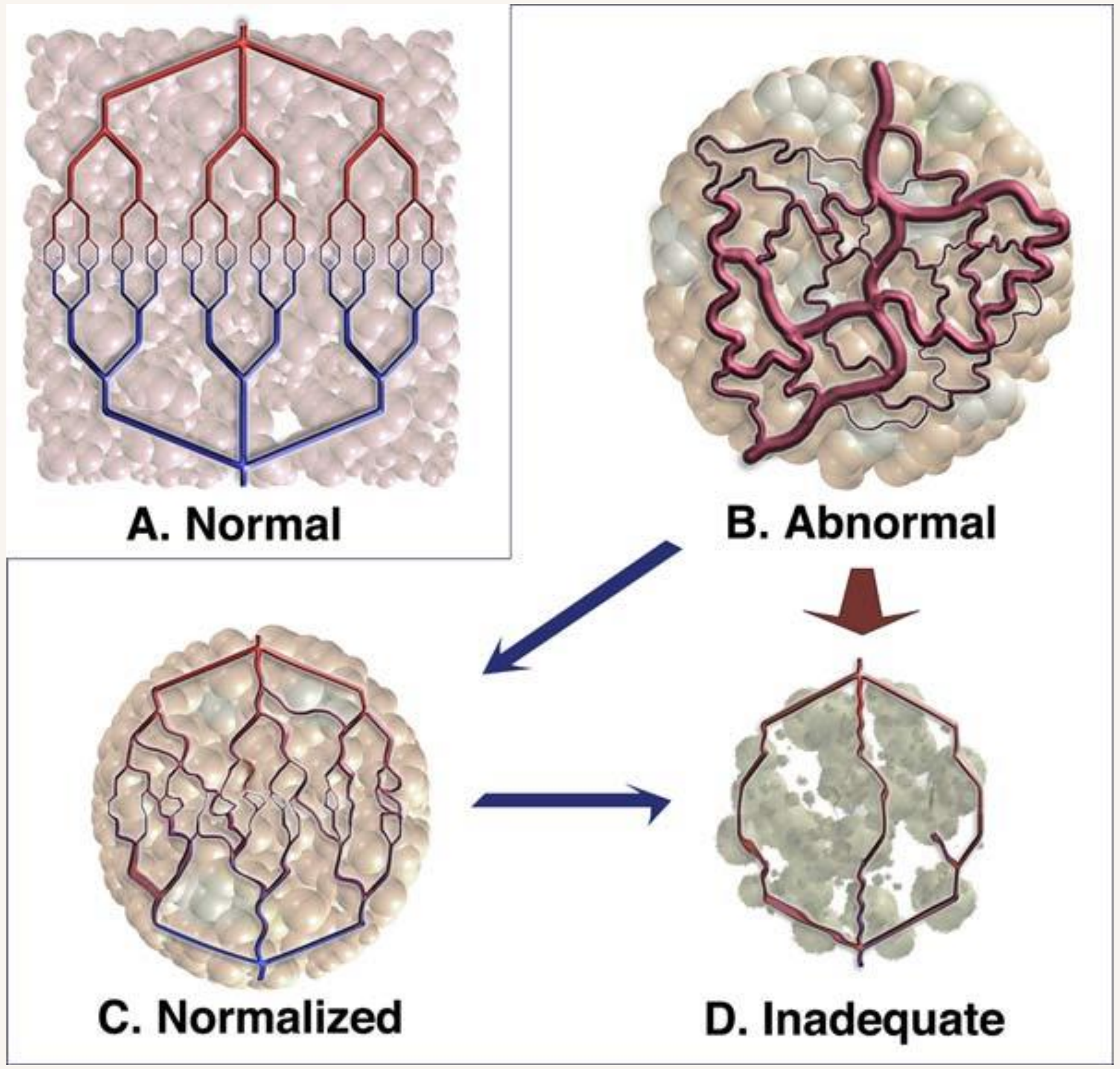
- Có cấu trúc tinh tế ở mọi kích cỡ. Mỗi thành phần đồng dạng với toàn thể
- Có hình dạng phức tạp
- Có thể tạo thành từ phép đệ quy của một hình đơn giản
- Có số chiều không phải là số nguyên

5. Ứng dụng

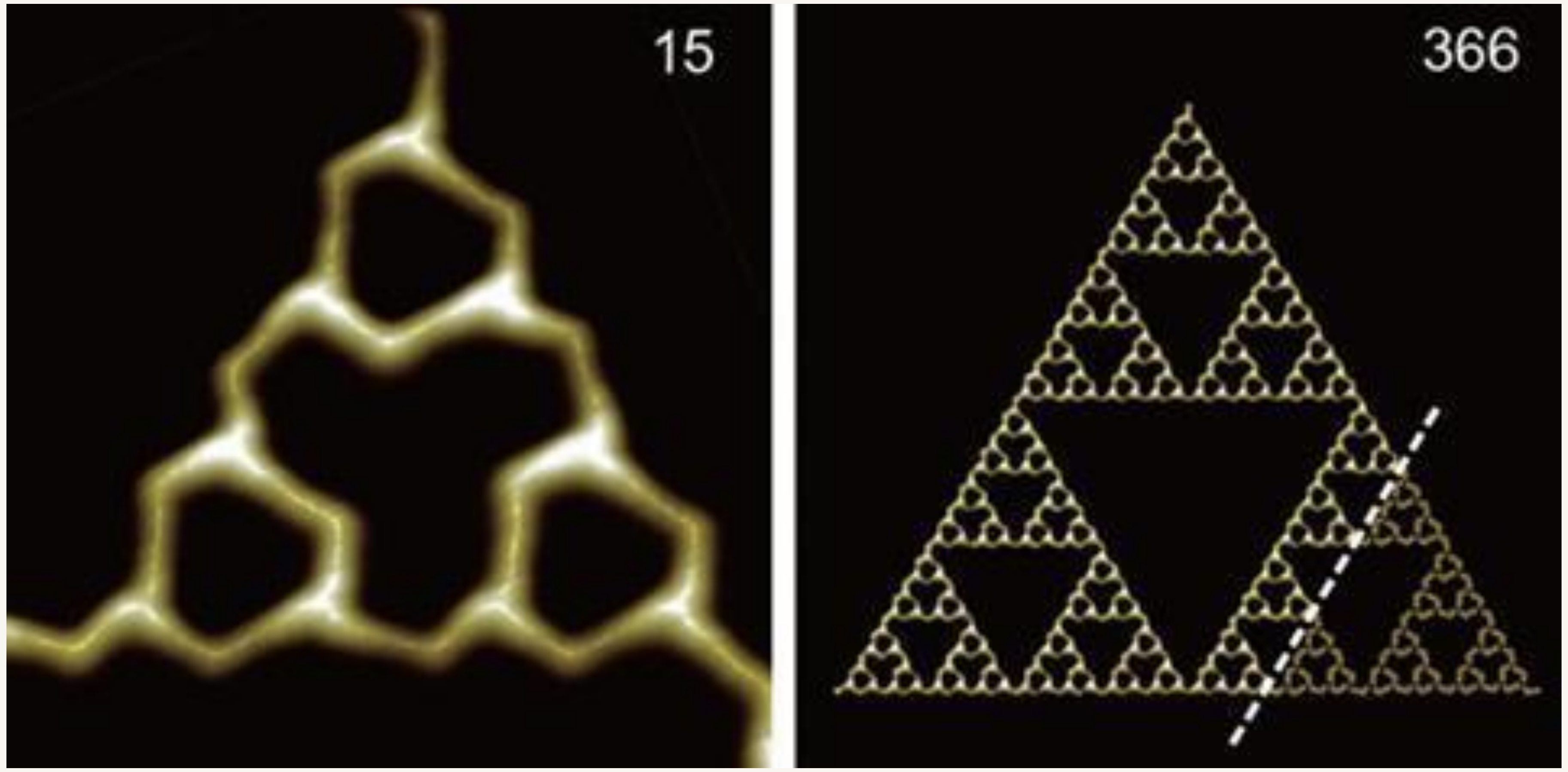
Nén ảnh



Tạo ảnh nghệ thuật



Sinh học

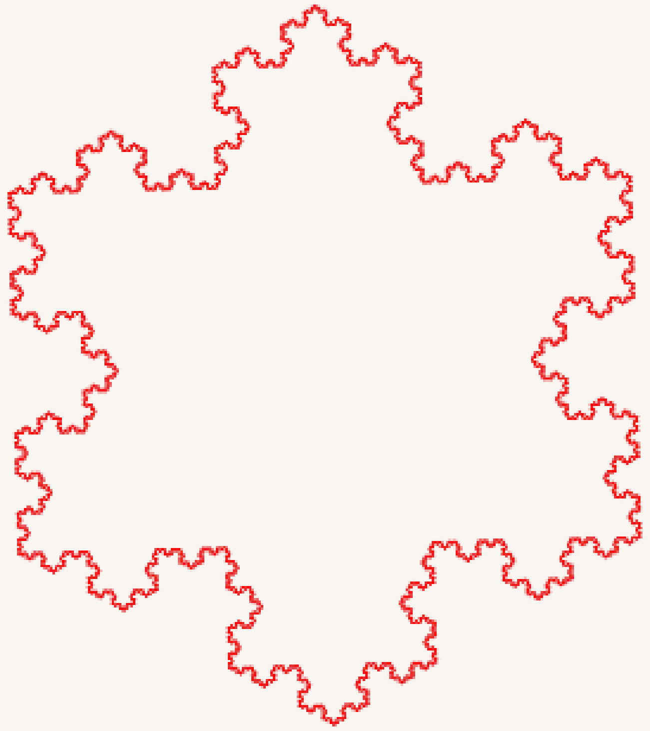
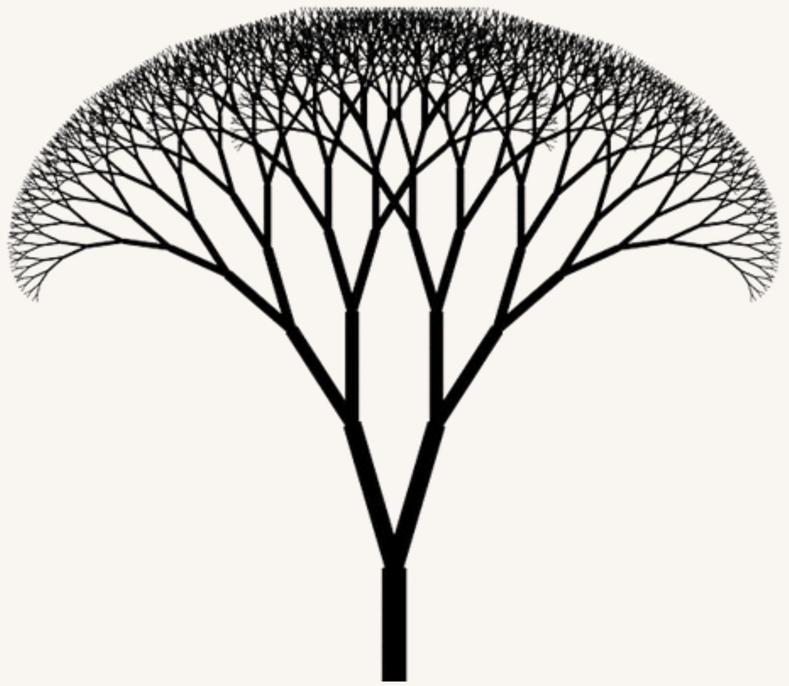
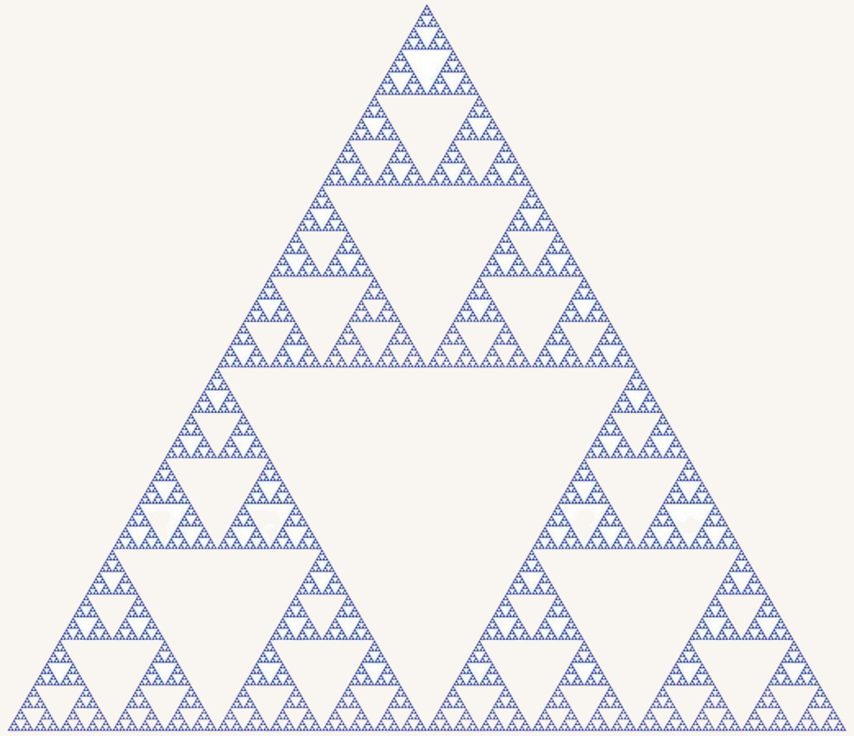
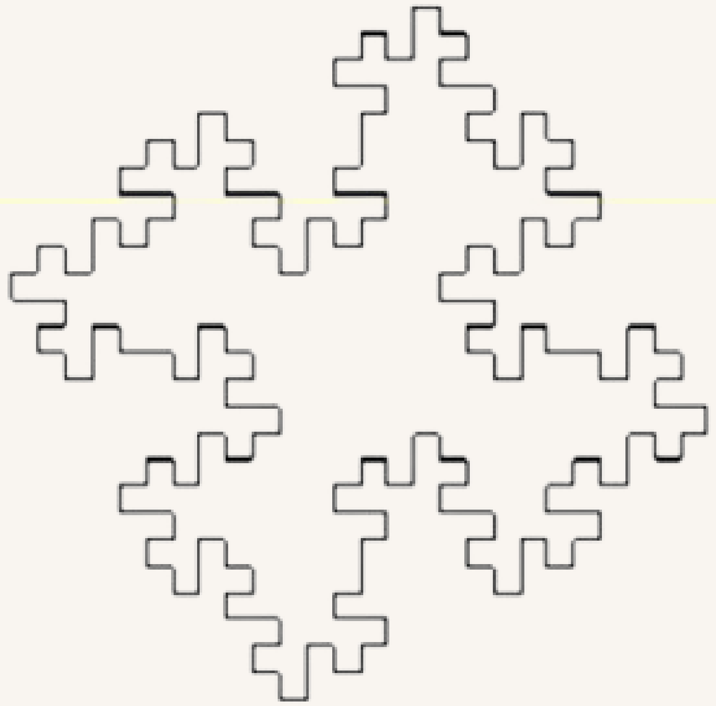


Hóa học

II. HÌNH HỌC FRACTAL TỪ SỐ THỰC

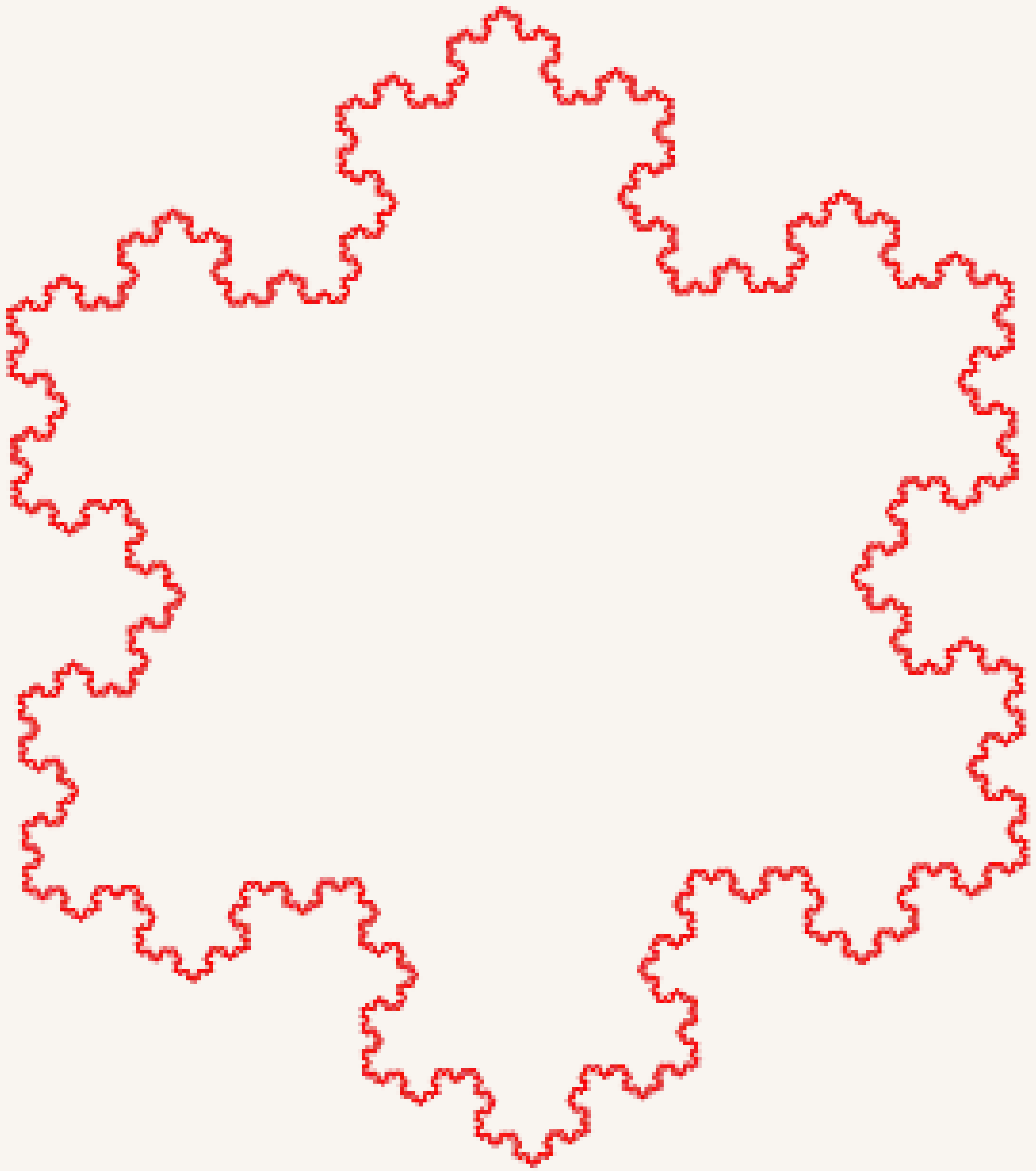
1. Khái niệm

- Dạng fractal từ số thực là dạng hình học fractal được định nghĩa bởi sự lặp lại của một số quy tắc đơn giản
- Fractal thuộc dạng xác định có cấu trúc rõ ràng và có thể được tính toán một cách chính xác
- Các ví dụ cơ bản của dạng này: bông tuyết Von Koch, tam giác Sierpinski, cây Canopy và hòn đảo Minskowki

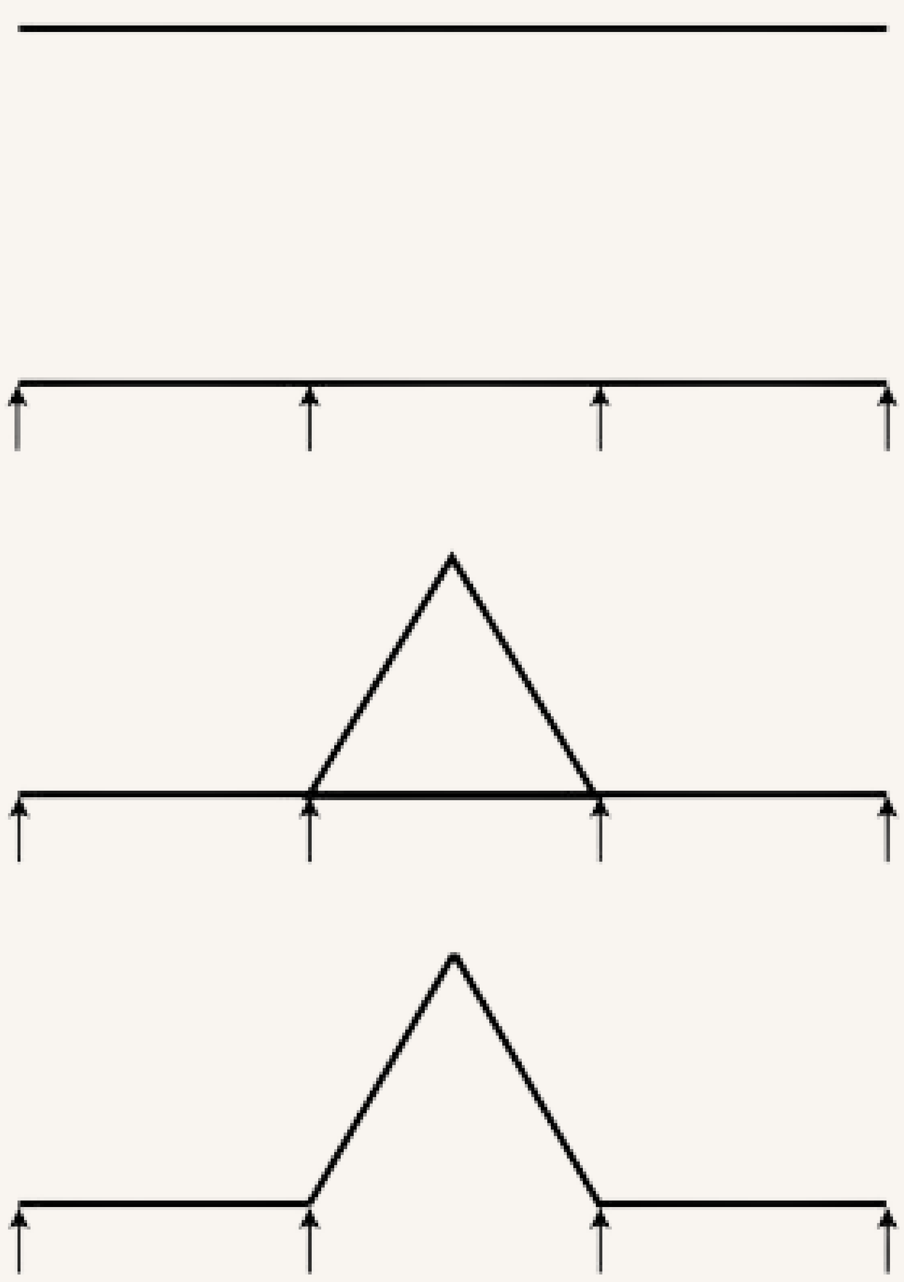
2. Bông tuyết Von Koch

- Bông tuyết Von Koch (đường cong Koch, ngôi sao Koch) là một trong nhiều phân dạng hình học fractal được phát hiện sớm nhất
- Được Helge von Koch đưa ra khái niệm vào năm 1904, trong một bài báo có tên “Trên một đường cong liên tục không có tiếp tuyến, thu được bằng cách xây dựng hình học cơ bản”



Đường cong von Koch

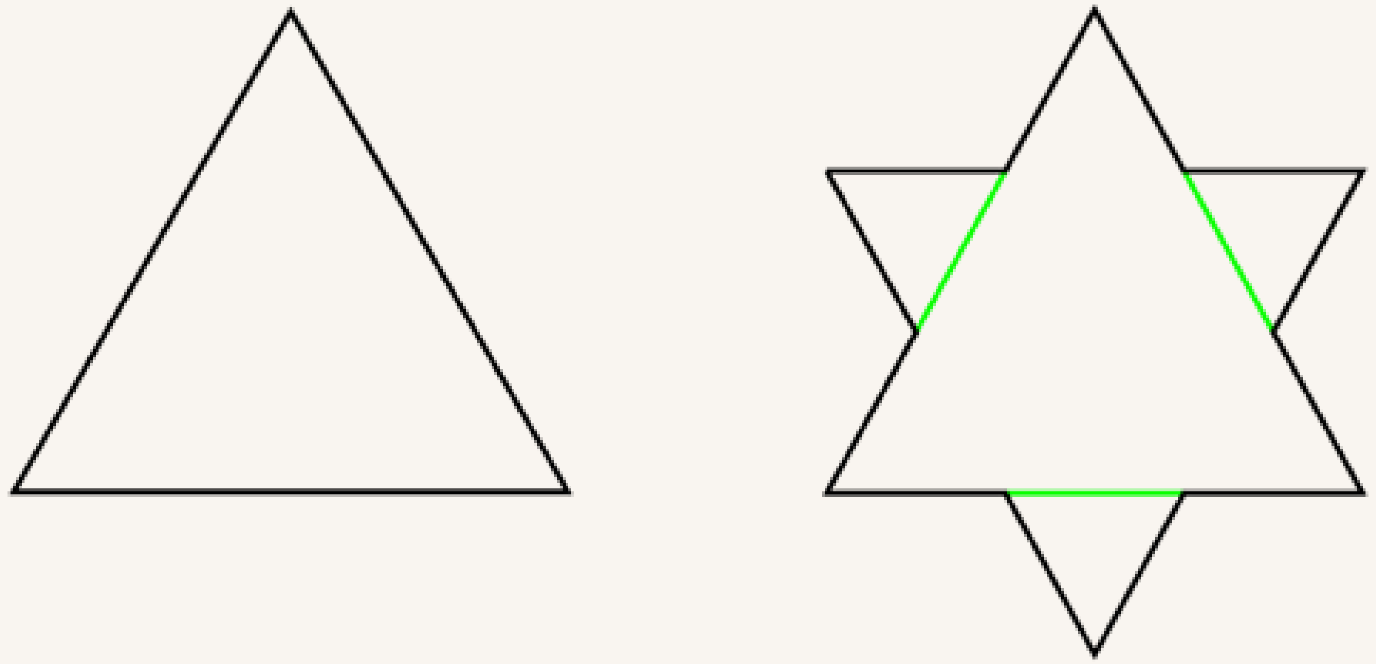
1. Khởi đầu bằng một đoạn thẳng
2. Chia đoạn thẳng thành ba phần bằng nhau. Thay đoạn ở giữa bằng 2 đoạn bằng nhau và bằng đoạn bị thay thế (2 đoạn đó sẽ tạo một góc 60 độ).
3. Lặp lại bước 2 cho bốn đoạn thẳng. Cứ thế tiếp tục mãi.



Bông tuyết von Koch

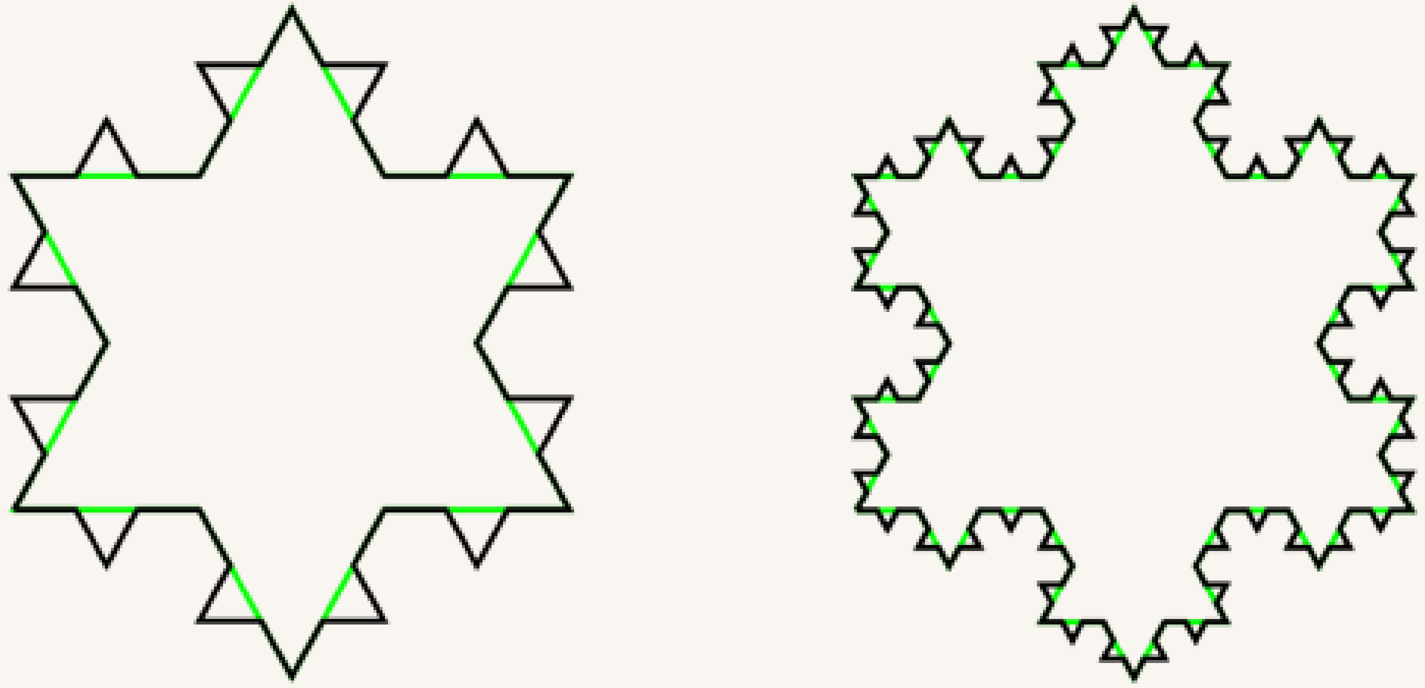
1. Khởi đầu bằng một tam giác đều

2. Chia mỗi cạnh của tam giác thành ba đoạn thẳng bằng nhau. Dựng tam giác đều trên đoạn ở giữa (ở bên ngoài tam giác đã cho) rồi xoá cạnh đáy của tam giác đều này thì được một đường gấp khúc kín.



Bông tuyết von Koch

3. Ở mỗi bước tiếp theo, chia mỗi đoạn của đường gấp khúc kín thành ba đoạn con bằng nhau, dựng tam giác đều trên đoạn con ở giữa (ở bên ngoài đường gấp khúc kín đó) rồi xoá cạnh đáy. Cứ như thế mãi thì được bông tuyết Von Koch.



Diện tích, chu vi của bông tuyết Von Koch

Gọi cạnh của tam giác đều ban đầu là a. Gọi hình thu được ở bước thứ n là Hn

Gọi Sn, pn lần lượt là diện tích và chu vi của hình Hn

Khi đó: $\lim_{n\to\infty} S_n, \lim_{n\to\infty} p_n$ lần lượt là diện tích và chu vi của bông tuyết Von Koch.

Chu vi của bông tuyết Von Koch

Ta sẽ tính chu vi của bông tuyết Von Koch.

Số cạnh của Hn là 3.4^n. Độ dài mỗi cạnh của Hn là $\frac{a}{3^n}$. Do đó, chu vi của Hn là $p_n = 3.4^n.\frac{a}{3^n} = 3a.\left(\frac{4}{3}\right)^n$

Vậy dãy số pn là một cấp số nhân và $\lim_{n\to\infty} p_n = +\infty$

Vậy chu vi của bông tuyết Von Koch là vô hạn.

Diện tích của bông tuyết Von Koch

Diện tích bông tuyết Von Koch được tính như sau:

Diện tích tam giác ABC cạnh a là S = $\frac{a^2\sqrt{3}}{4}$

$$S_1 - S = 3.\left(\frac{S}{9}\right) = \frac{S}{3}.$$

$$S_2 - S_1 = 4.3.\left(\frac{S}{9^2}\right) = \frac{S}{3}.\left(\frac{4}{9}\right).$$

$$S_3 - S_2 = 4^2.3.\left(\frac{S}{9^3}\right) = \frac{S}{3}.\left(\frac{4}{9}\right)^2.$$

Bằng phương pháp quy nạp, ta được:

$$S_n - S_{n-1} = 4^{n-1}.3.\left(\frac{S}{9^n}\right) = \frac{S}{3}\left(\frac{4}{9}\right)^{n-1}$$

Diện tích của bông tuyết Von Koch

Cộng từng vế n đẳng thức trên, ta được

$$S_n - S = \frac{S}{3} + \frac{S}{3}.\left(\frac{4}{9}\right) + \frac{S}{3}.\left(\frac{4}{9}\right)^2 \dots + \frac{S}{3}.\left(\frac{4}{9}\right)^{n-1}$$

Vế phải của đẳng thức trên là tổng của n số hạng đầu tiên của cấp số nhân lùi vô hạn có số hạng đầu là S/3 và công bội là 4/9. Tổng của cấp số nhân này là: $\frac{S}{3}.\frac{1}{1-\frac{4}{9}} = \frac{3S}{5}$.

Do đó: $\lim_{n\to\infty}(S_n - S) = \frac{3S}{5}$

Diện tích của bông tuyết Von Koch

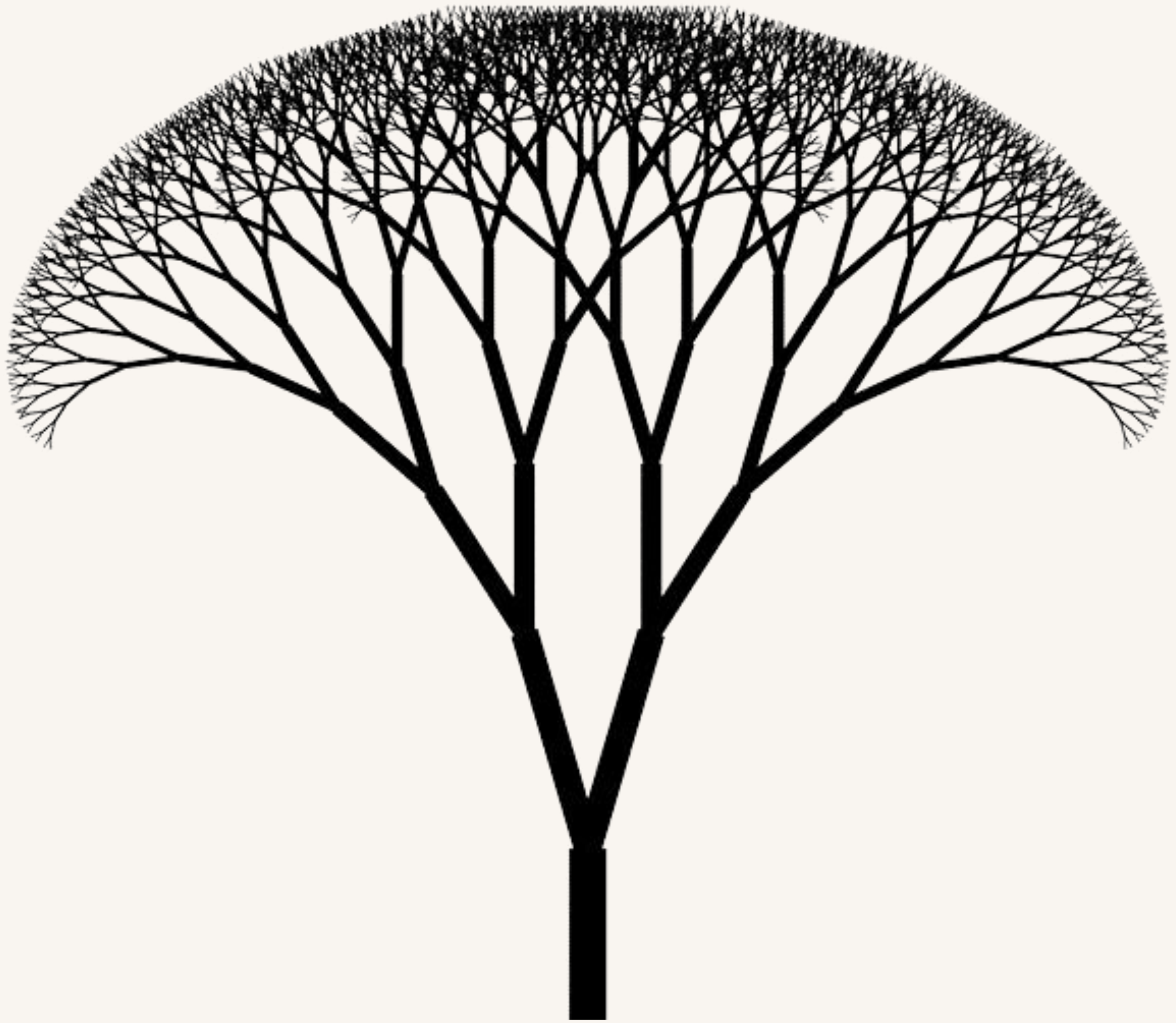
Suy ra: $\lim_{n\to\infty} S_n = \frac{3S}{5} + S = \frac{8S}{5} = \frac{8}{5} . \frac{a^2\sqrt{3}}{4}$

Vậy diện tích của bông tuyết Von Koch là hữu hạn. (nó chỉ gấp 1.6 lần diện tích tam giác đều ban đầu).

3. Canopy Fractal

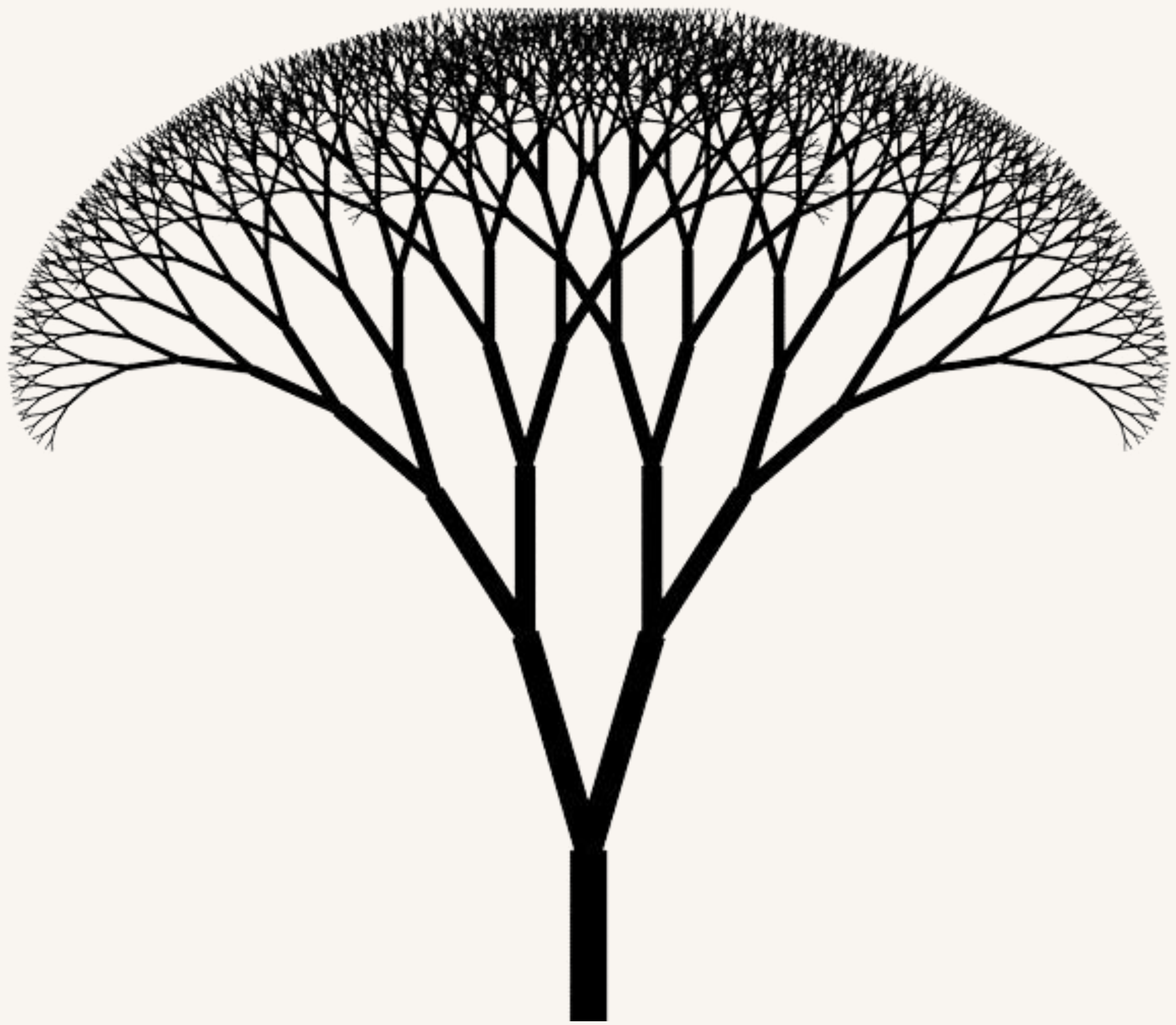
Là một loại fractal dùng để mô phỏng cấu trúc tán cây trong rừng

Phổ biến trong lĩnh vực khoa học môi trường và mô phỏng sinh học



Canopy Ftactal

Fractal Canopy được tạo ra bằng cách lặp lại một quá trình chia nhỏ các đoạn thẳng, tương tự như cách mà cấu trúc của nhánh cây được hình thành trong quá trình sinh trưởng.



Canopy Ftactal

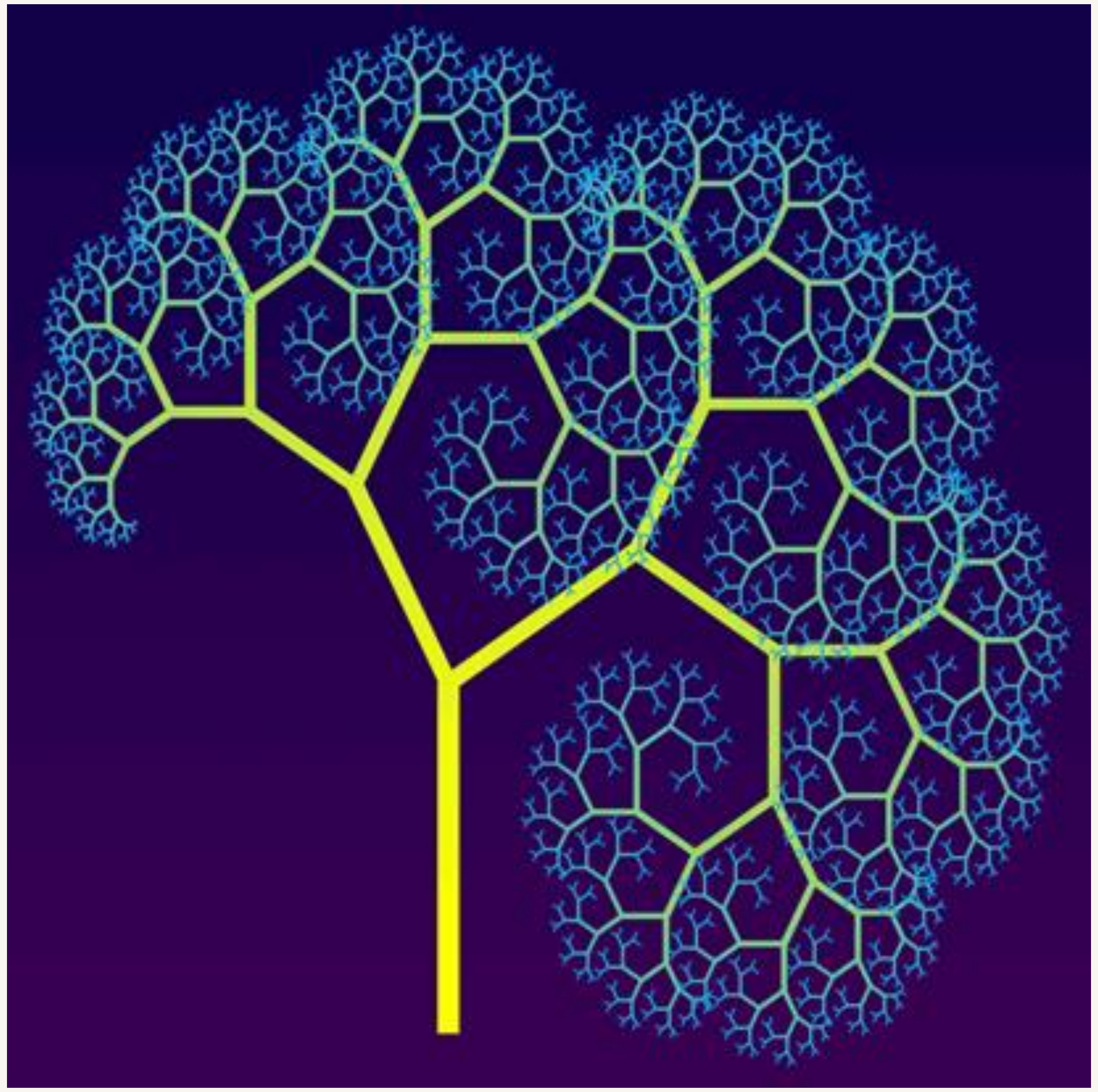
Cụ thể là:

1. Bắt đầu với một đoạn thẳng dài đại diện cho thân cây

2. Từ ngọn của thân cây, chia đoạn thẳng thành hai phần có độ dài bằng ½ đoạn thẳng ban đầu.

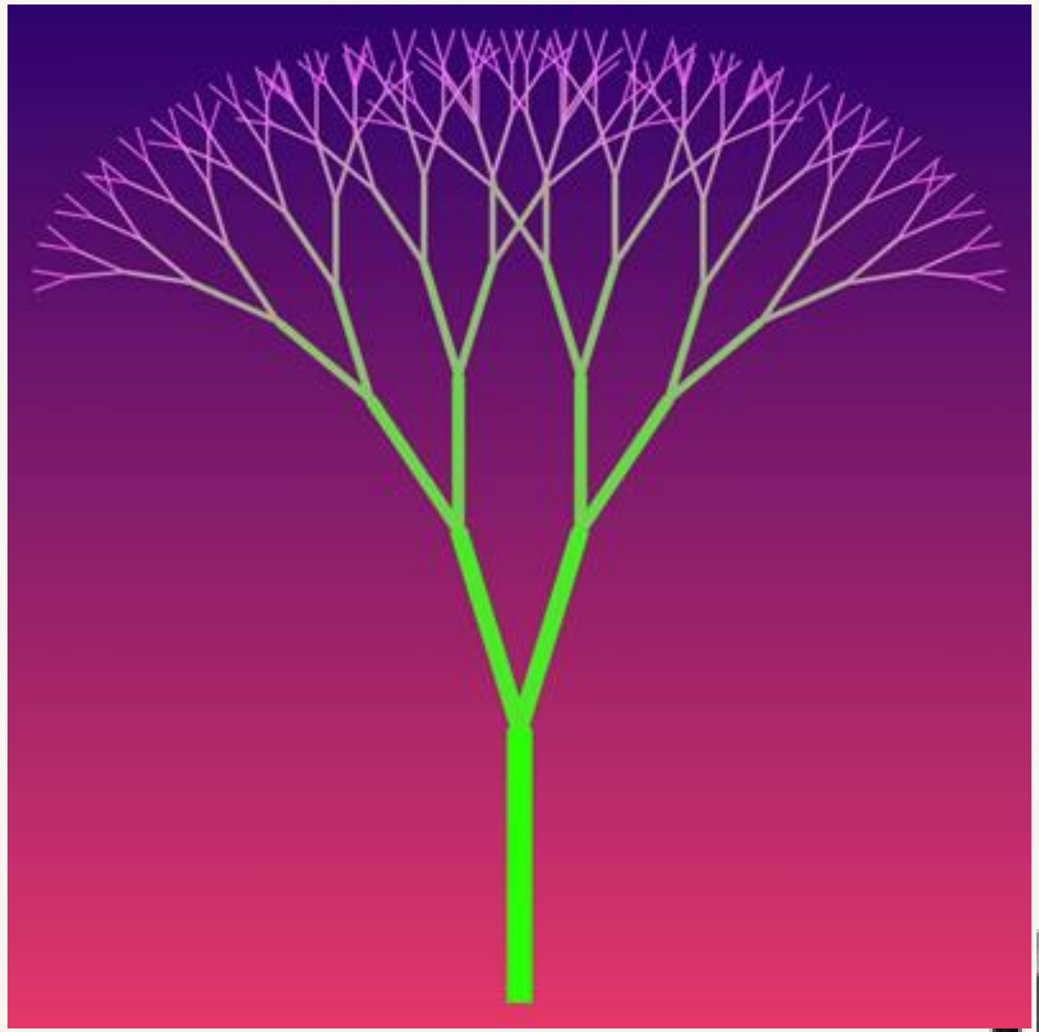
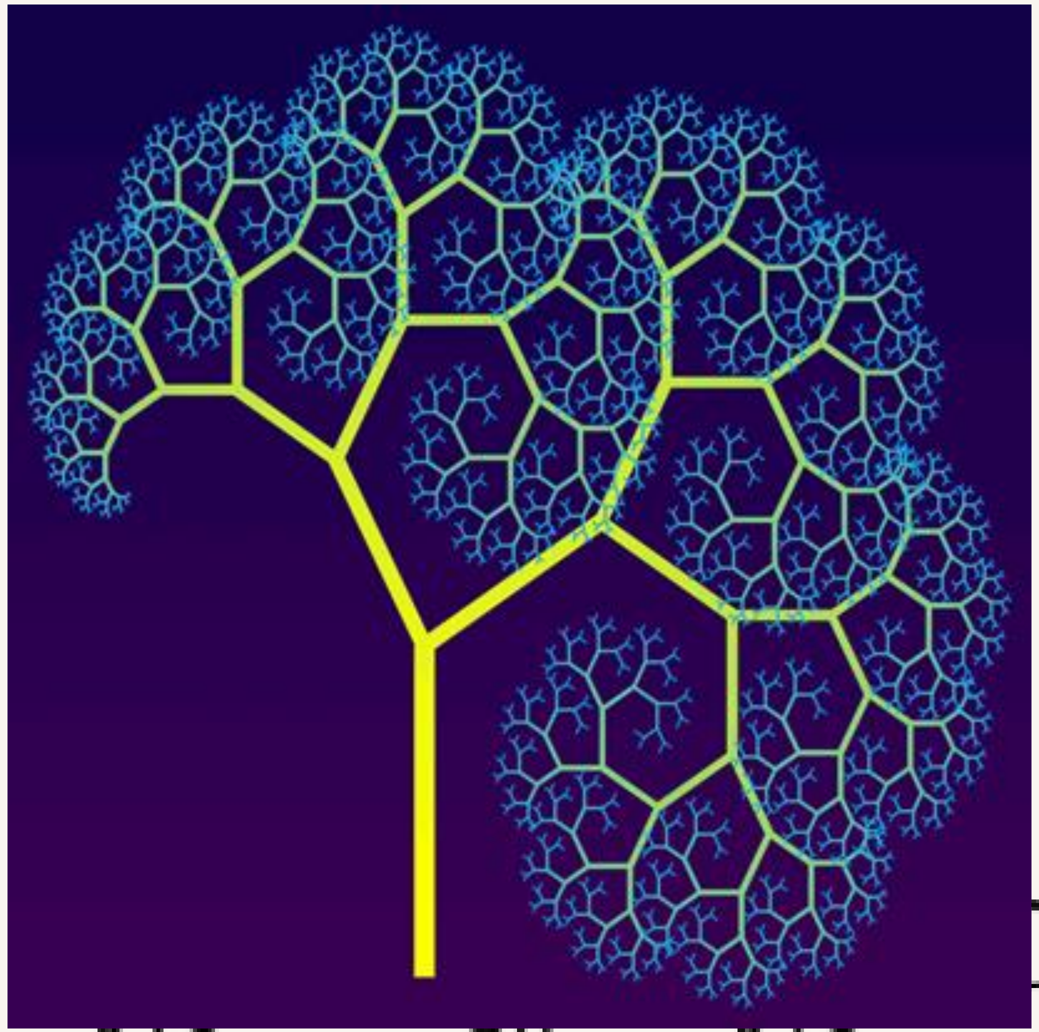
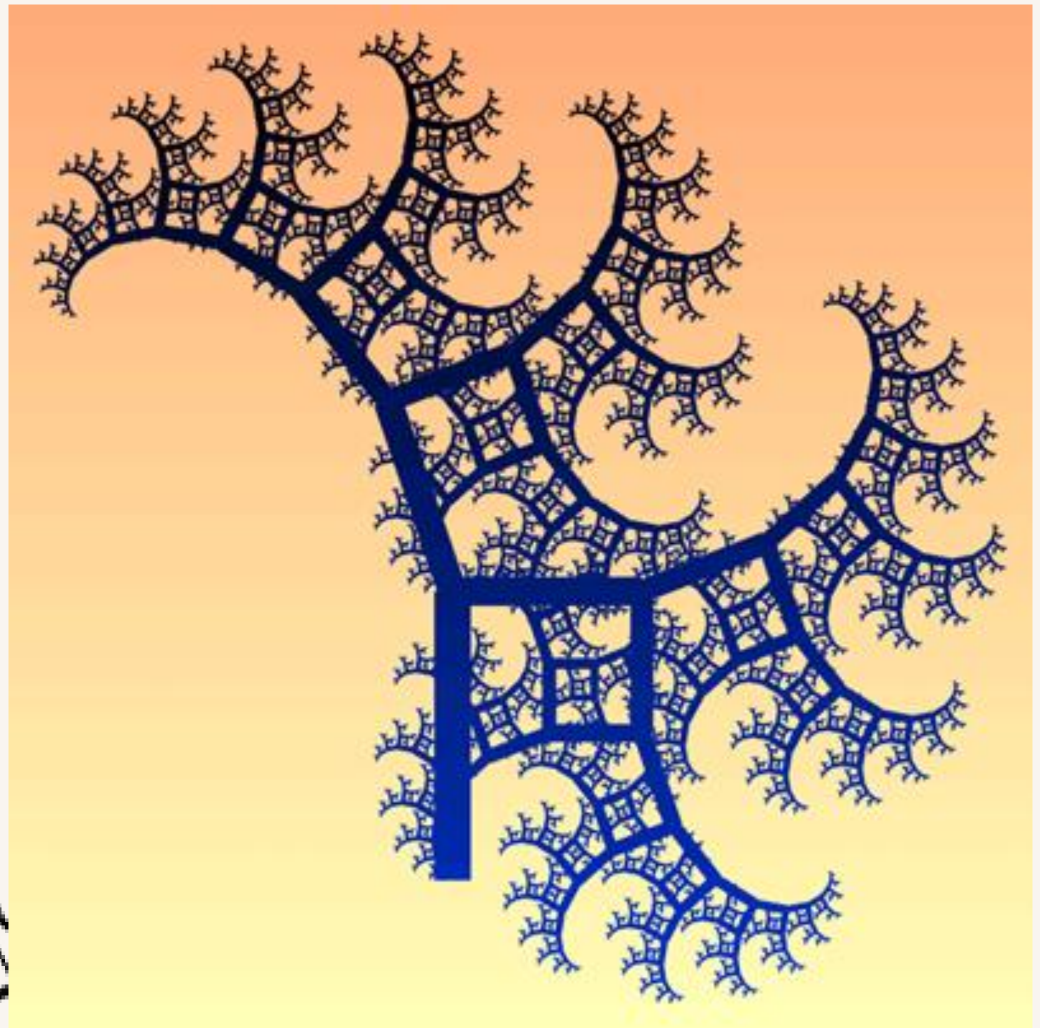
Canopy Ftactal

3. Lặp lại quá trình chia nhỏ đến khi kích thước của đoạn thẳng đạt đến một ngưỡng nhất định
4. Tăng tính ngẫu nhiên: thay đổi giá trị các tham số như góc quay, vị trí, khích thước và hình dáng của các đoạn thẳng



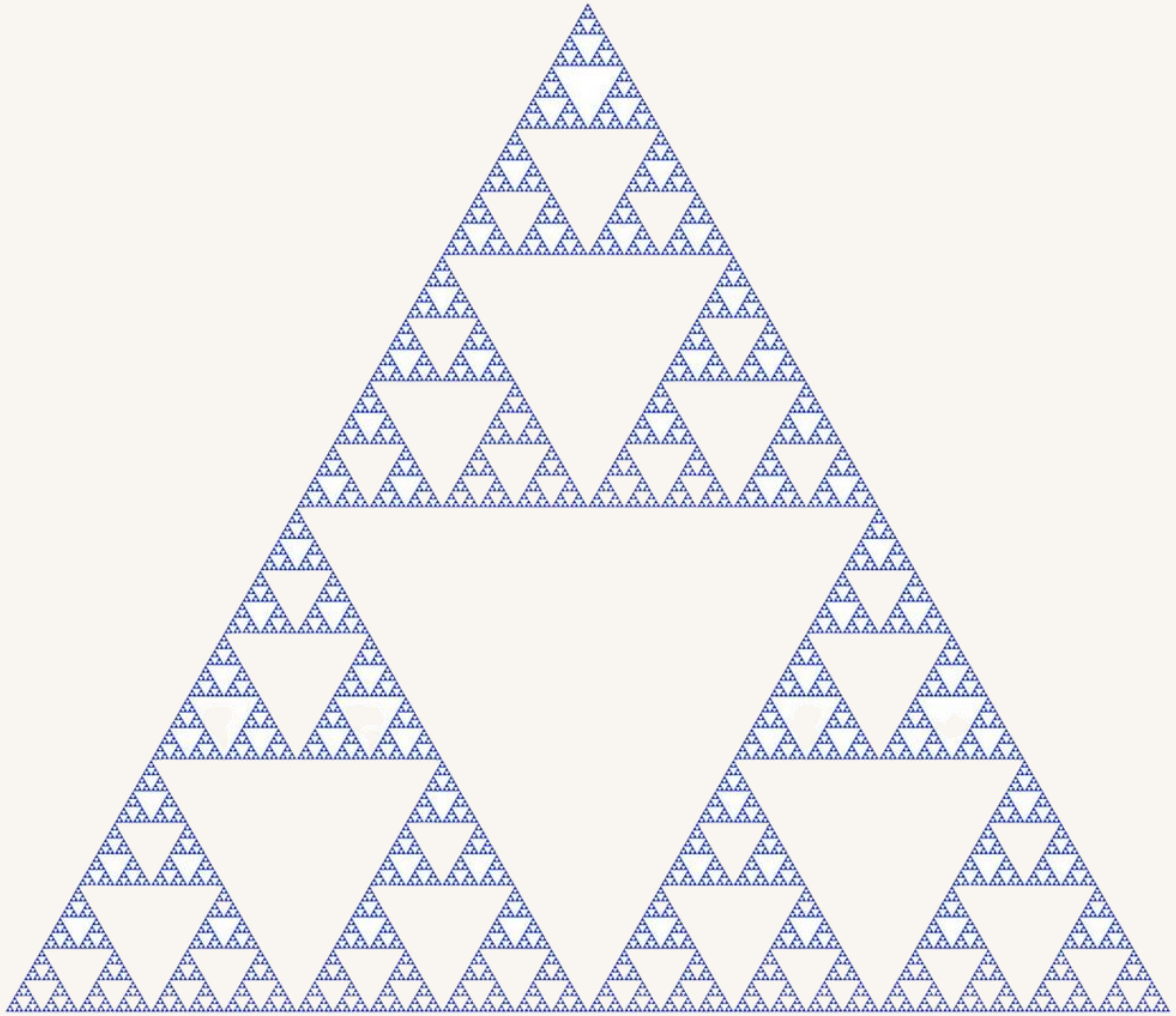
Canopy Ftactal

5. Áp dụng các màu sắc và hiệu ứng ánh sáng để tạo ra các hình ảnh sống động hơn

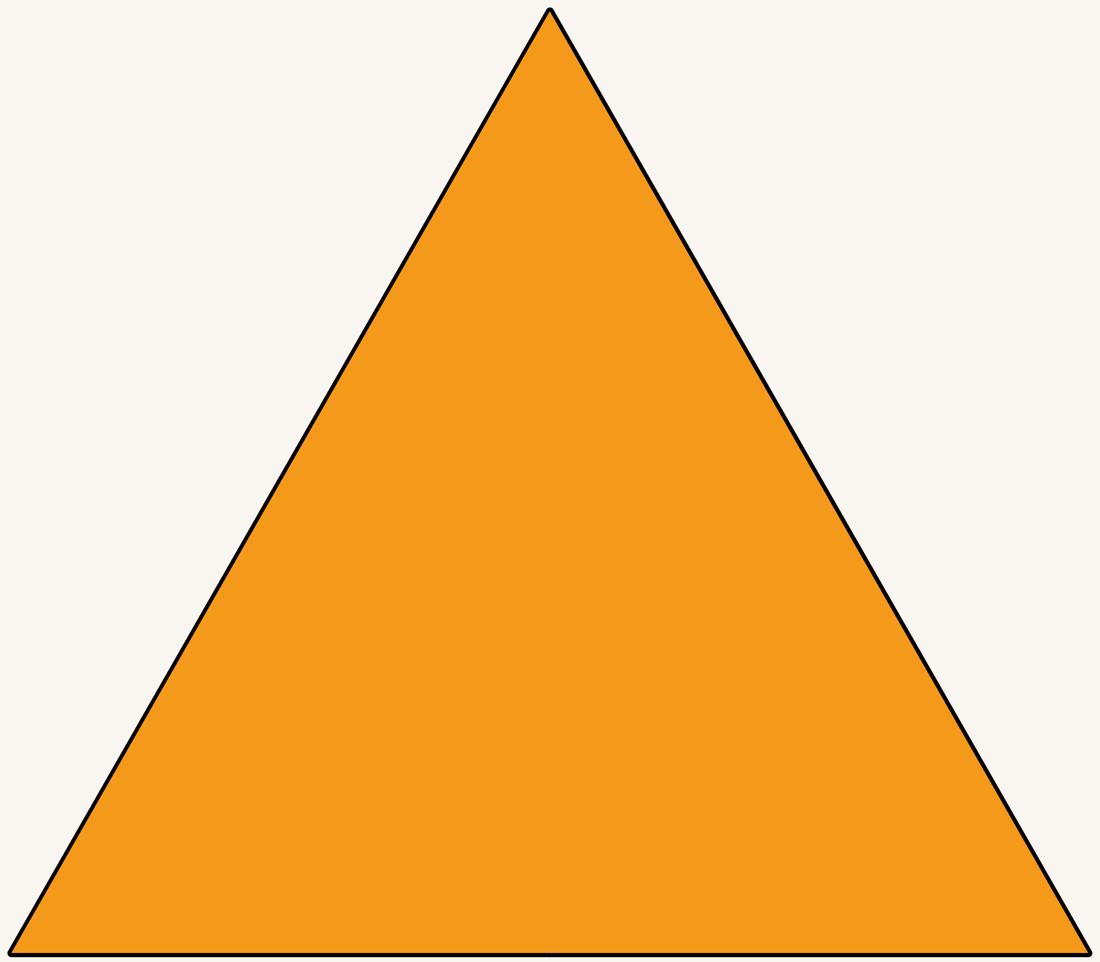
4. Tam giác Sierpinki

Tam giác Sierpinki, được đặt theo tên của nhà toán học Ba Lan Wacław Sierpiński. Nó có thể được tạo ra bằng cách bắt đầu với một hình tam giác lớn, bằng nhau, và sau đó liên tục cắt các hình tam giác nhỏ hơn ra khỏi tâm của nó.



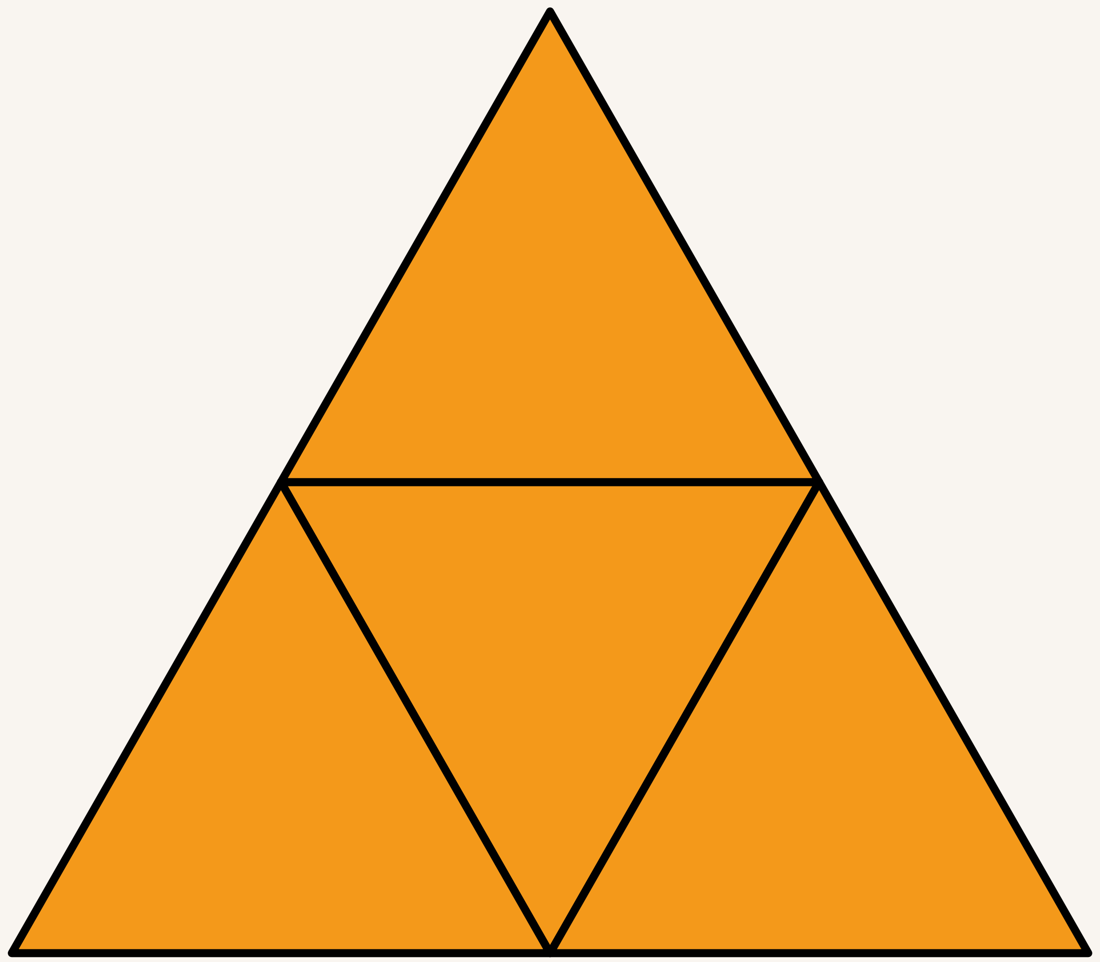
Thuật toán

1) Khởi đầu bằng một tam giác đều



Tam giác Sierpinki

2) Chia tam giác đó thành bốn tam giác đều bằng nhau (dựa vào trung điểm các cạnh). Sau đó bỏ tam giác ở giữa.

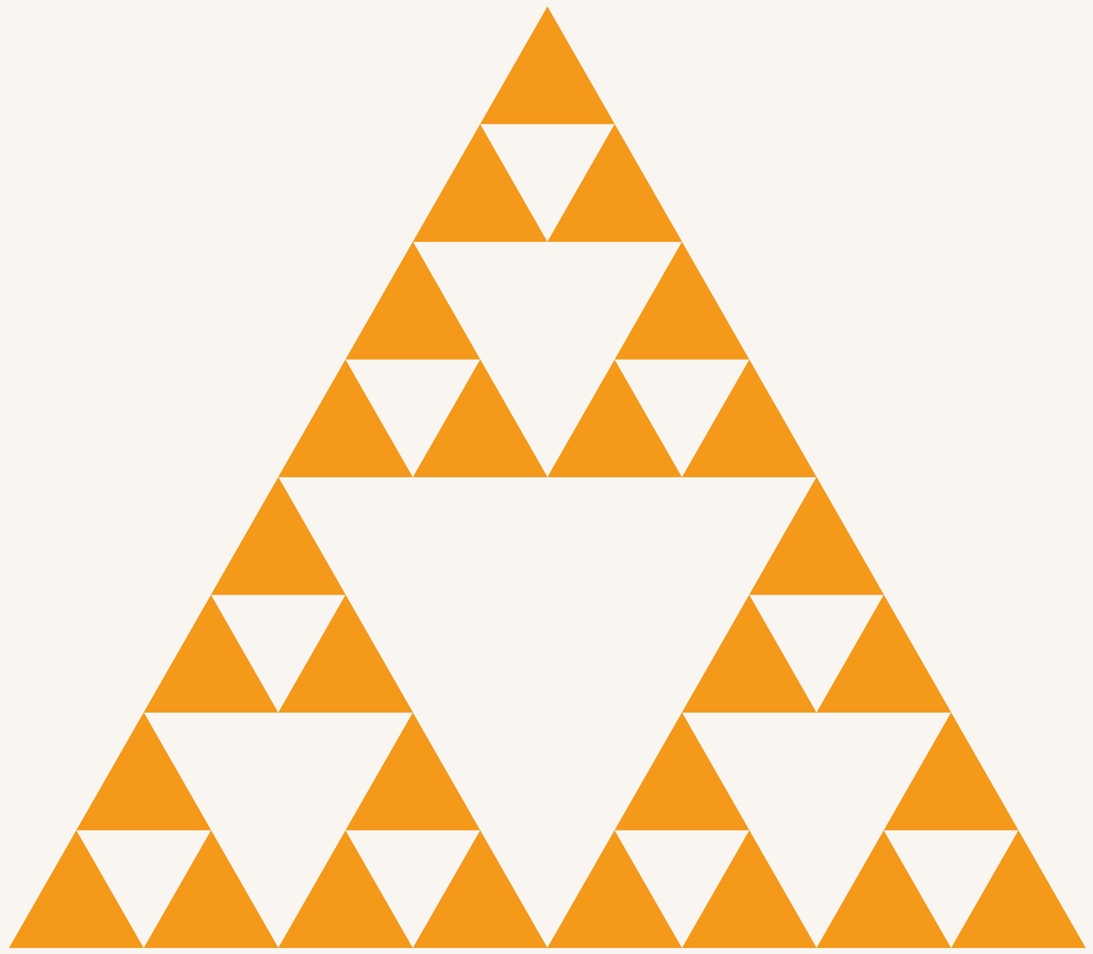


Tam giác Sierpinki

2) Chia tam giác đó thành bốn tam giác đều bằng nhau (dựa vào trung điểm các cạnh). Sau đó bỏ tam giác ở giữa.

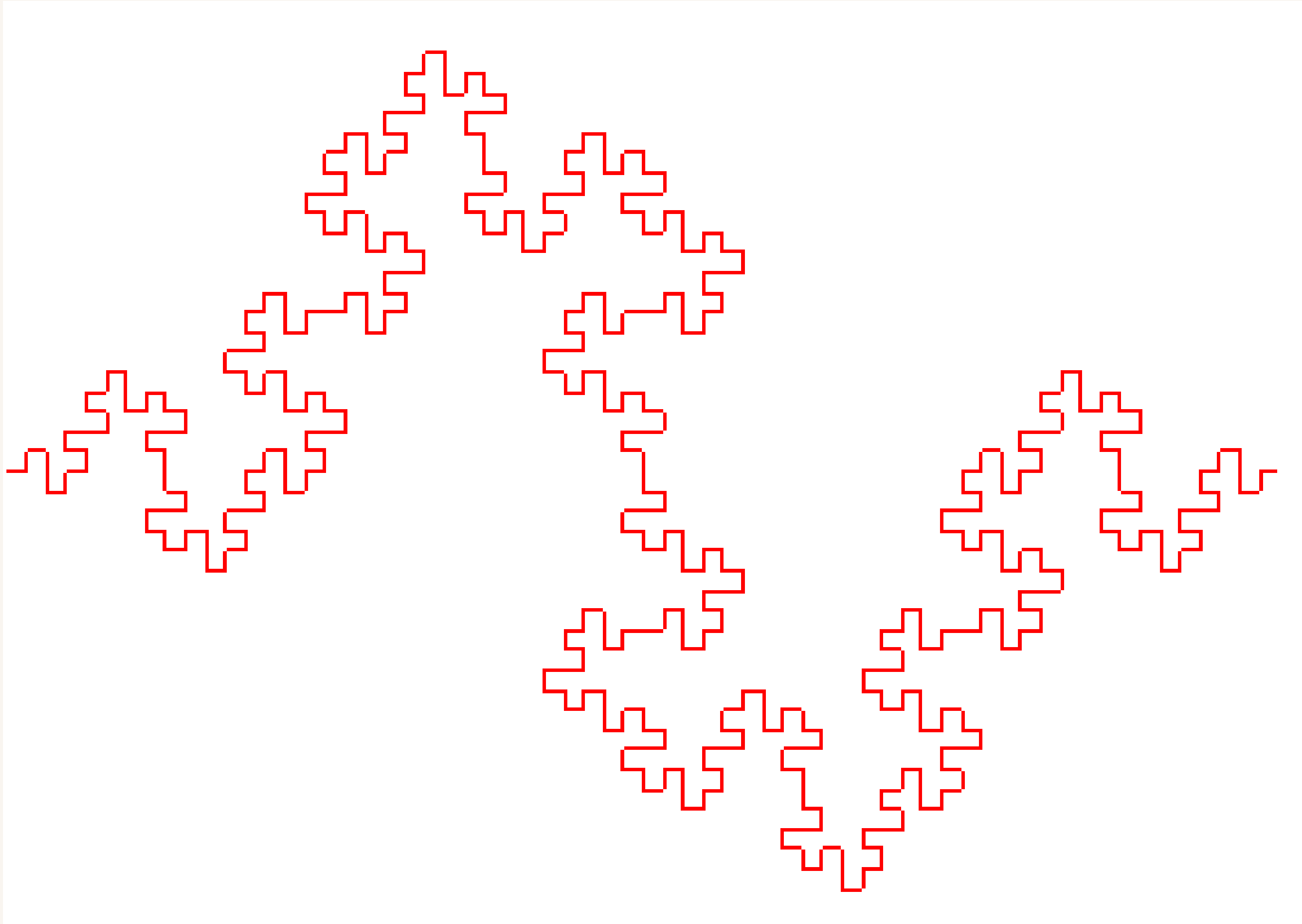
Tam giác Sierpinki

3) Lặp lại bước 2 cho các tam giác đều nhỏ còn lại. Cứ tiếp tục đến khi đạt mức độ mong muốn.



5. Đường cong Minskowki

- Được đặt theo tên của nhà toán học Hermann Minkowski.
- Được tạo ra bằng cách kết hợp hai đường cong hoặc tập hợp điểm trong không gian hai chiều, kết quả là một đường cong mới với các tính chất hợp nhất của hai đường cong ban đầu.



Đường cong Minskowki

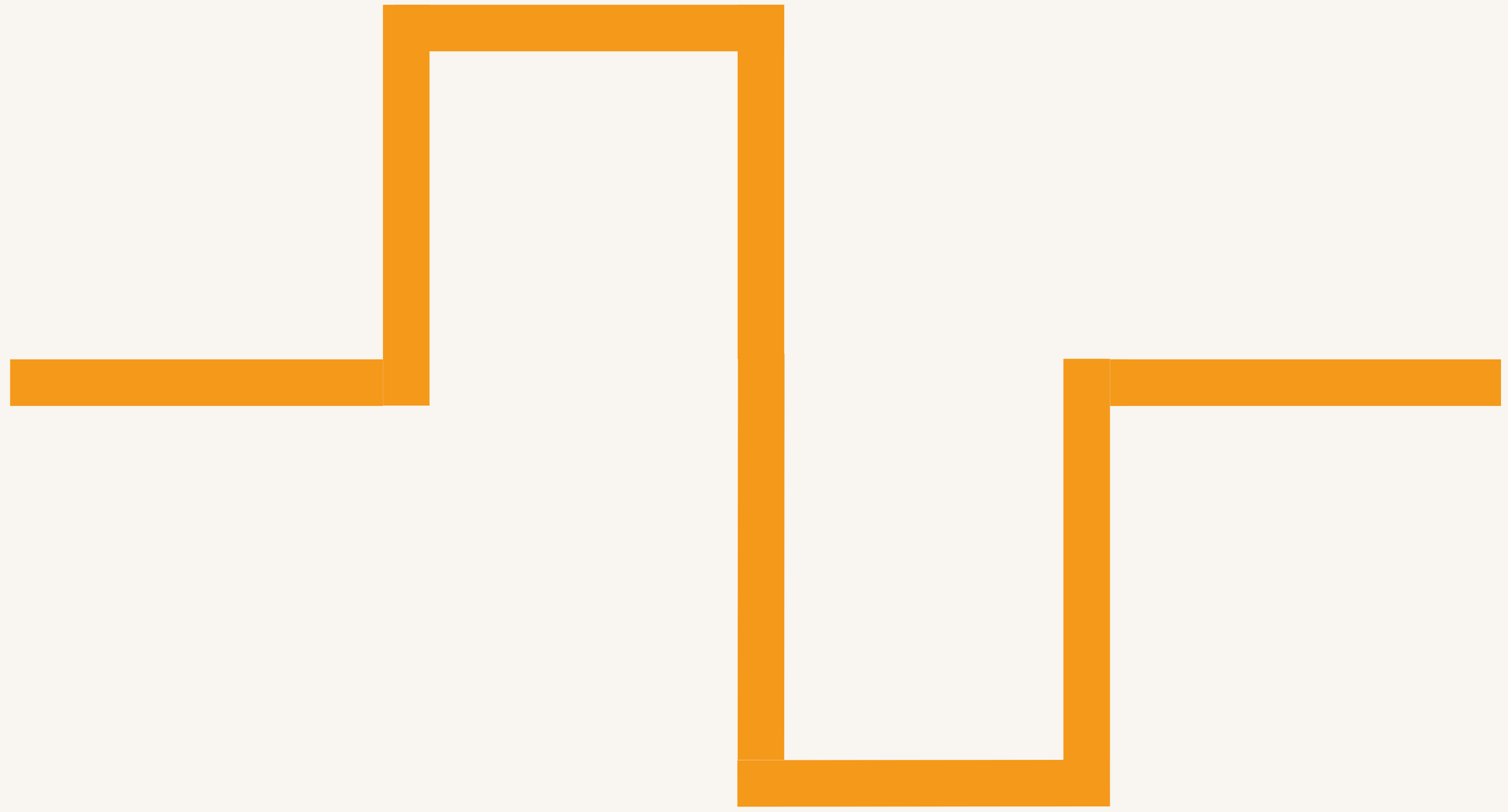
1) Khởi tạo 1 đoạn thẳng

Đường cong Minskowki

2) Chia đoạn thẳng thành 4 phần bằng nhau

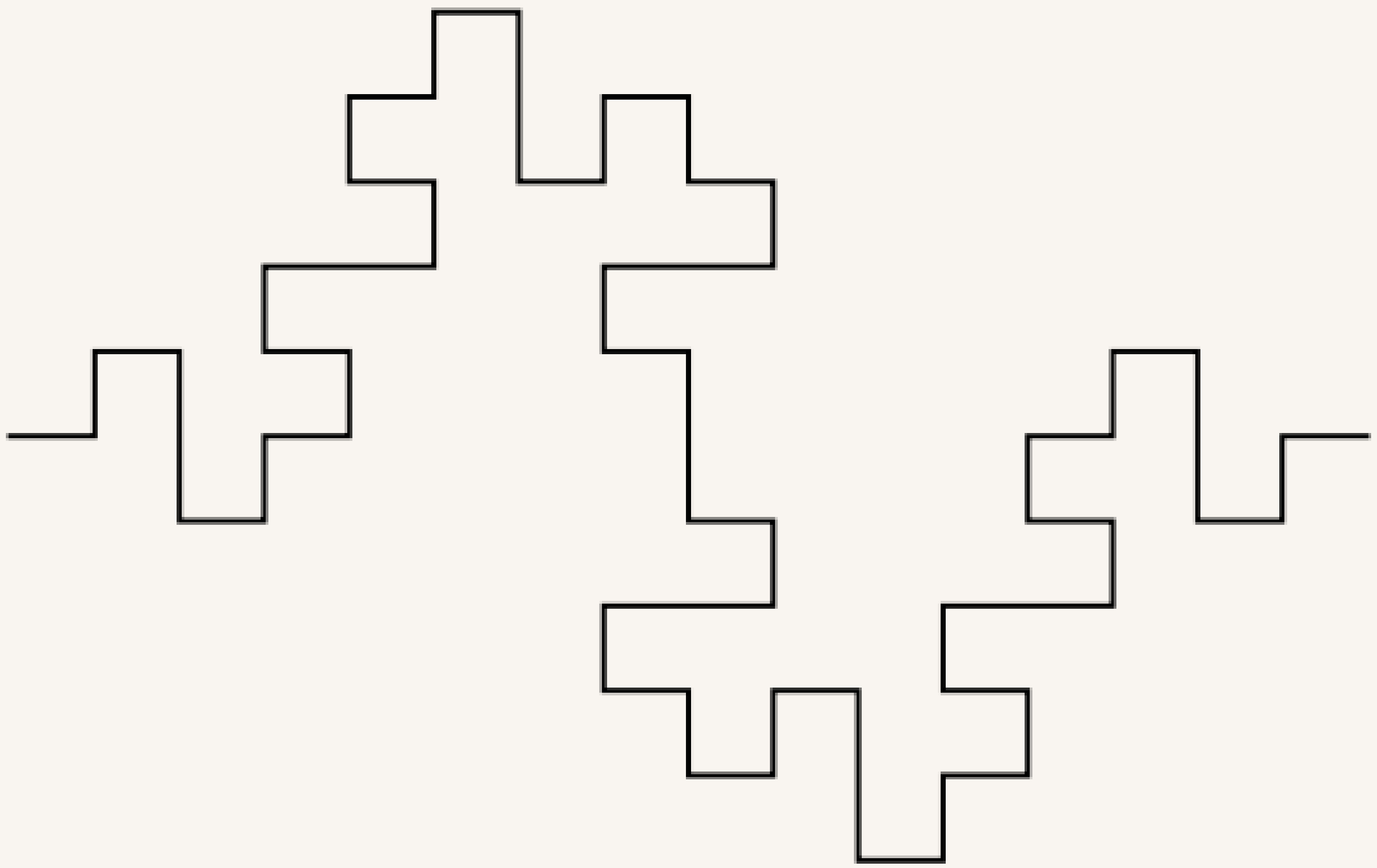
Đường cong Minskowki

3) Với 2 đoạn thẳng ở giữa, thay mỗi đoạn thành 3 đoạn thẳng sao cho nếu hợp với đoạn thẳng vừa thay thì được 1 hình vuông (với đoạn thẳng nằm ngang thì đoạn đầu tiên thay thì hình vuông hướng lên trên và đoạn thứ hai hướng xuống, tương tự với chiều dọc thì sẽ là trái và phải).



Đường cong Minskowki

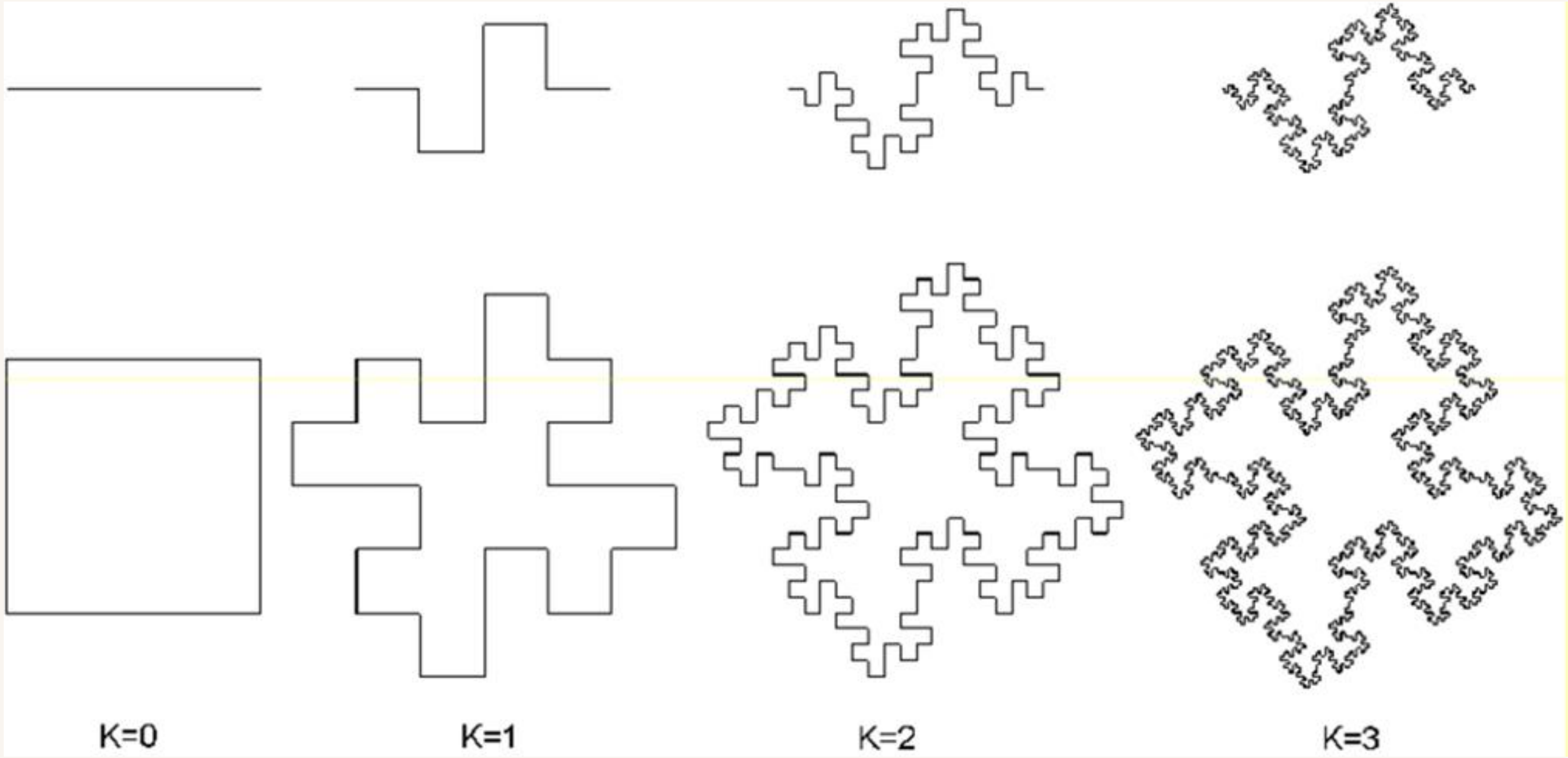
4) Lặp lại từ bước 2, đến khi đạt đủ lần lặp



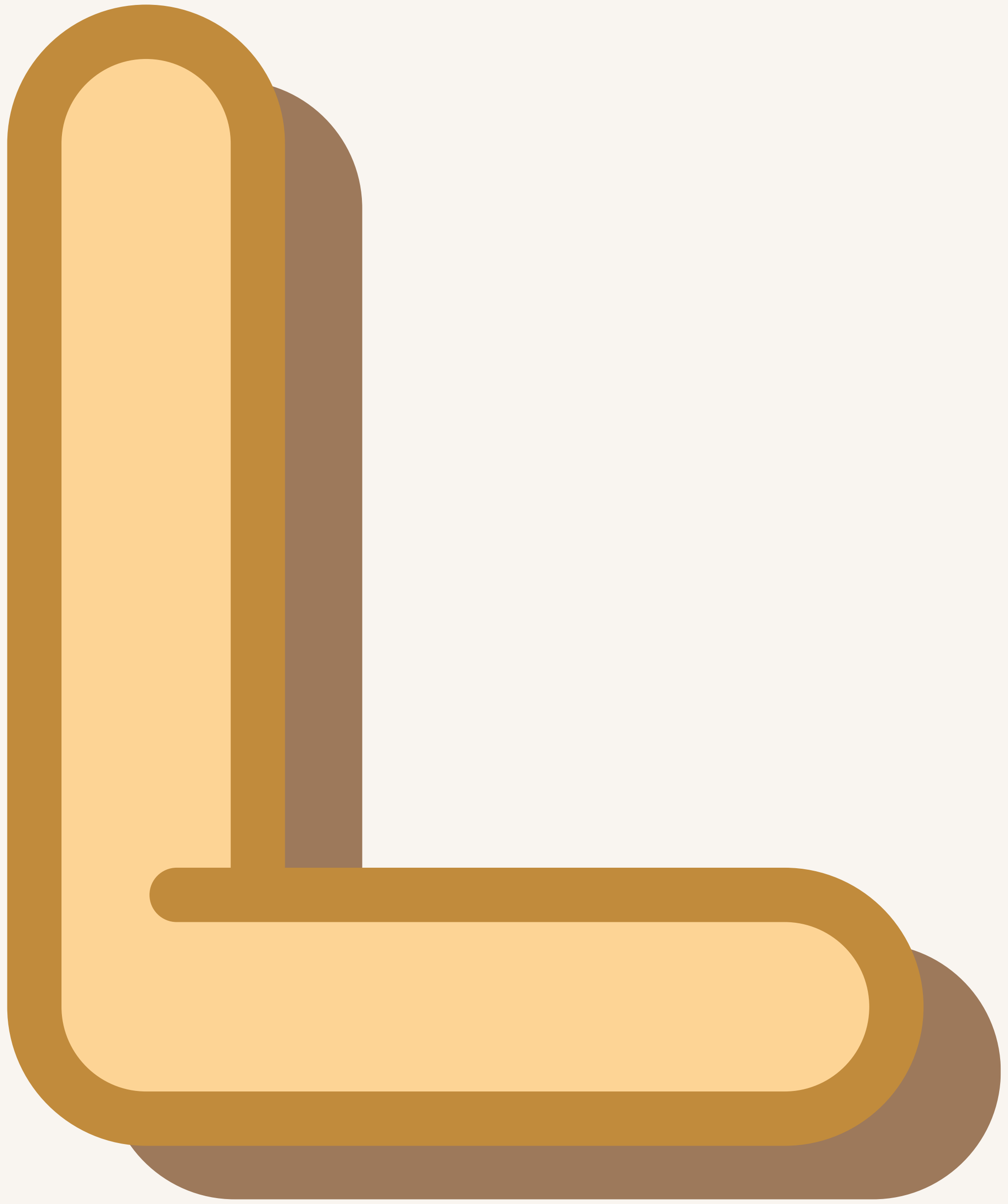
Hòn đảo Minskowki

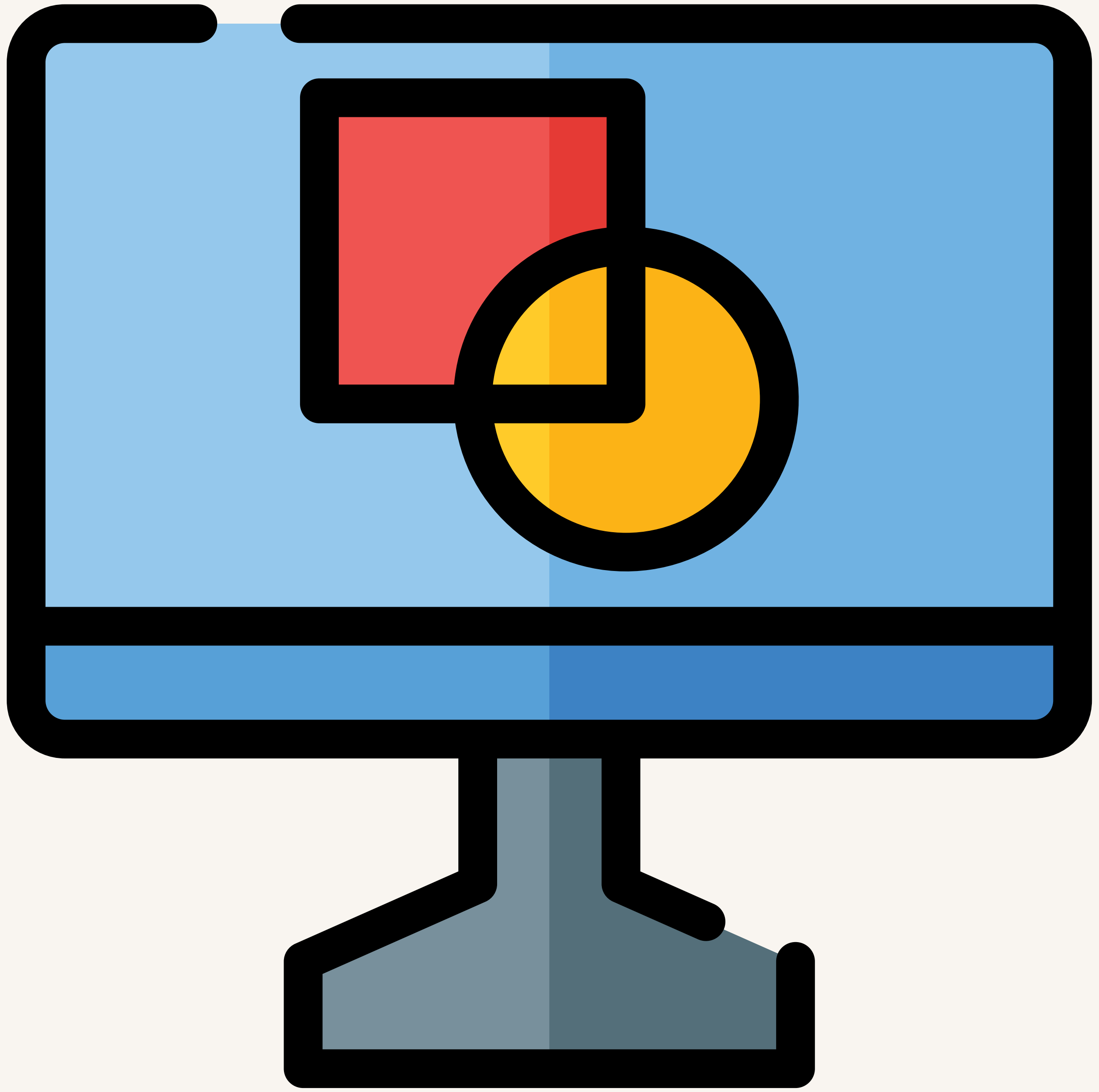
1) Khởi tạo một vuông

2) Biến đổi các cạnh của hình vuông thành đường cong Minskowki



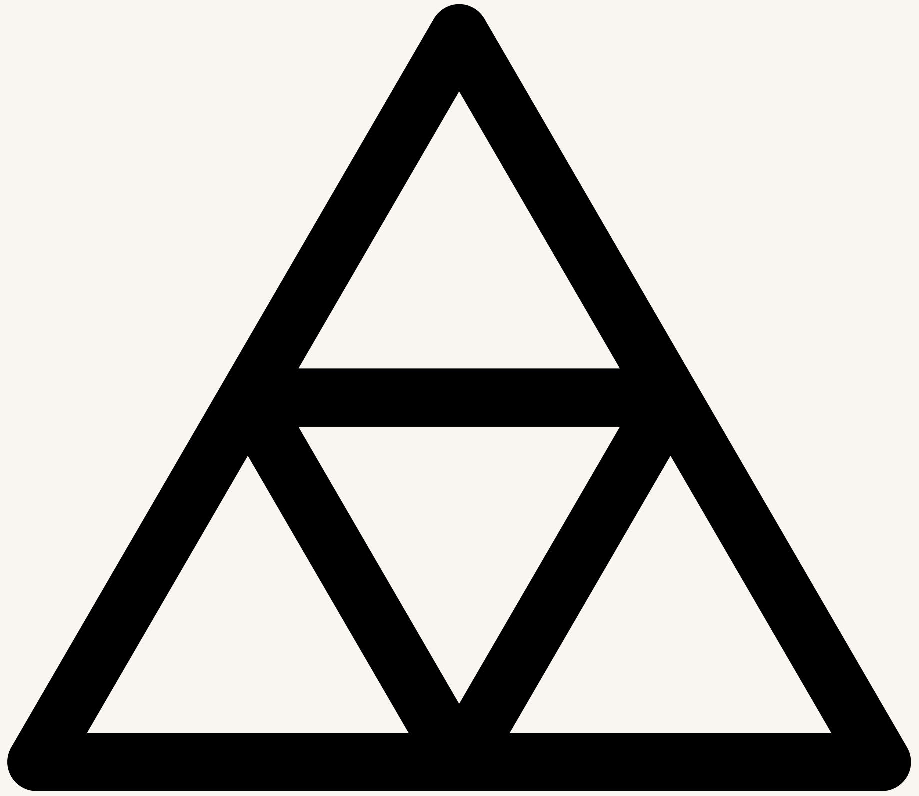
6. Vẽ fractal số thực với L-System





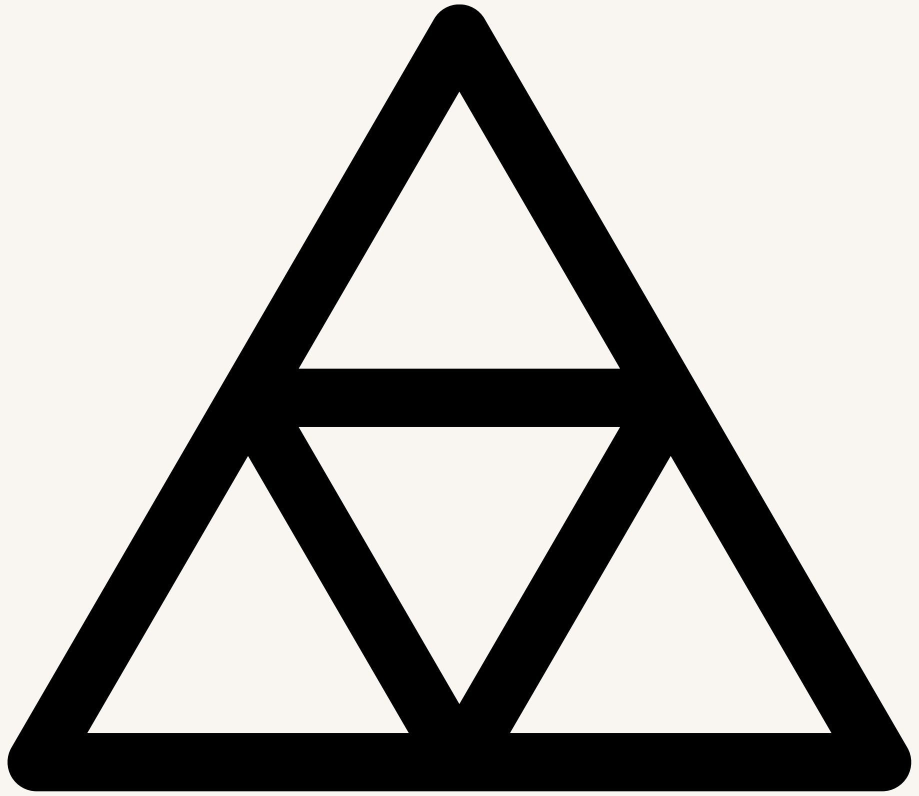
Đồ họa Turtle

- Đồ thị Turtle là một dạng đồ thị phổ biến trong đồ họa máy tính sử dụng con trỏ tương đối trên Hệ tọa độ Descartes.
- Nó cho phép vẽ một nét không nhấc bút và tiến đến một đoạn cùng với thao tác quay một góc so với hướng hiện hành.



Đồ họa Turtle

- Người khai sinh: Seymour Papert và Wally Feurzeig
- Turtle Graphics là một phần của ngôn ngữ lập trình Logo
- Trong Turtle Graphics, một con rùa ảo được sử dụng để vẽ các hình học trên màn hình.



Đồ họa Turtle trong Python

- Turtle Graphics là một module đồ họa trong Python cho phép vẽ đồ họa 2D bằng cách sử dụng các lệnh điều khiển con rùa ảo.

- Module này sử dụng thư viện đồ họa TKinter để hiển thị các hình vẽ trên cửa sổ đồ họa.

Đồ họa Turtle trong Python

- import turtle
- stack = []

Thuộc tính (vị trí p, hướng α, △d, △α)		
Lệnh trong python	Lệnh vẽ	Nội dung
turtle.forward(△d)	F	Vẽ thẳng tới trước 1 đoạn △d
turtle.left(△α)	+	Quay bút qua trái một góc △α
turtle.right(△α)	-	Quay bút qua phải một góc △α
stack.append((turtle.position(), turtle.heading()))	[	Lưu lại ví trí và hướng của bút vẽ
position, heading = stack.pop() turtle.penup() turtle.goto(position) turtle.setheading(heading) turtle.pendown()	]	Phục hồi lại vị trí và hướng của bút vẽ

Hệ L-System

- Hệ thống L-System được sáng lập bởi Lindenmayer.
- Nó bao gồm các thành phần sau:
  - Tập ký hiệu {F, +, -, [,],...}.
  - Tiên đề s0.
  - Tập luật sinh {f1, f2, ..., fN}.
- Quá trình sinh sản được thực hiện bằng cách thay thế các ký hiệu trong chuỗi s0 bằng vế phải của tập luật sinh, lần lượt cho đến khi nhận được chuỗi sn.

Hệ L-System

Ví dụ, giả sử chúng ta có một L-system đơn giản với các quy tắc sau:

- A → AB
- B → A

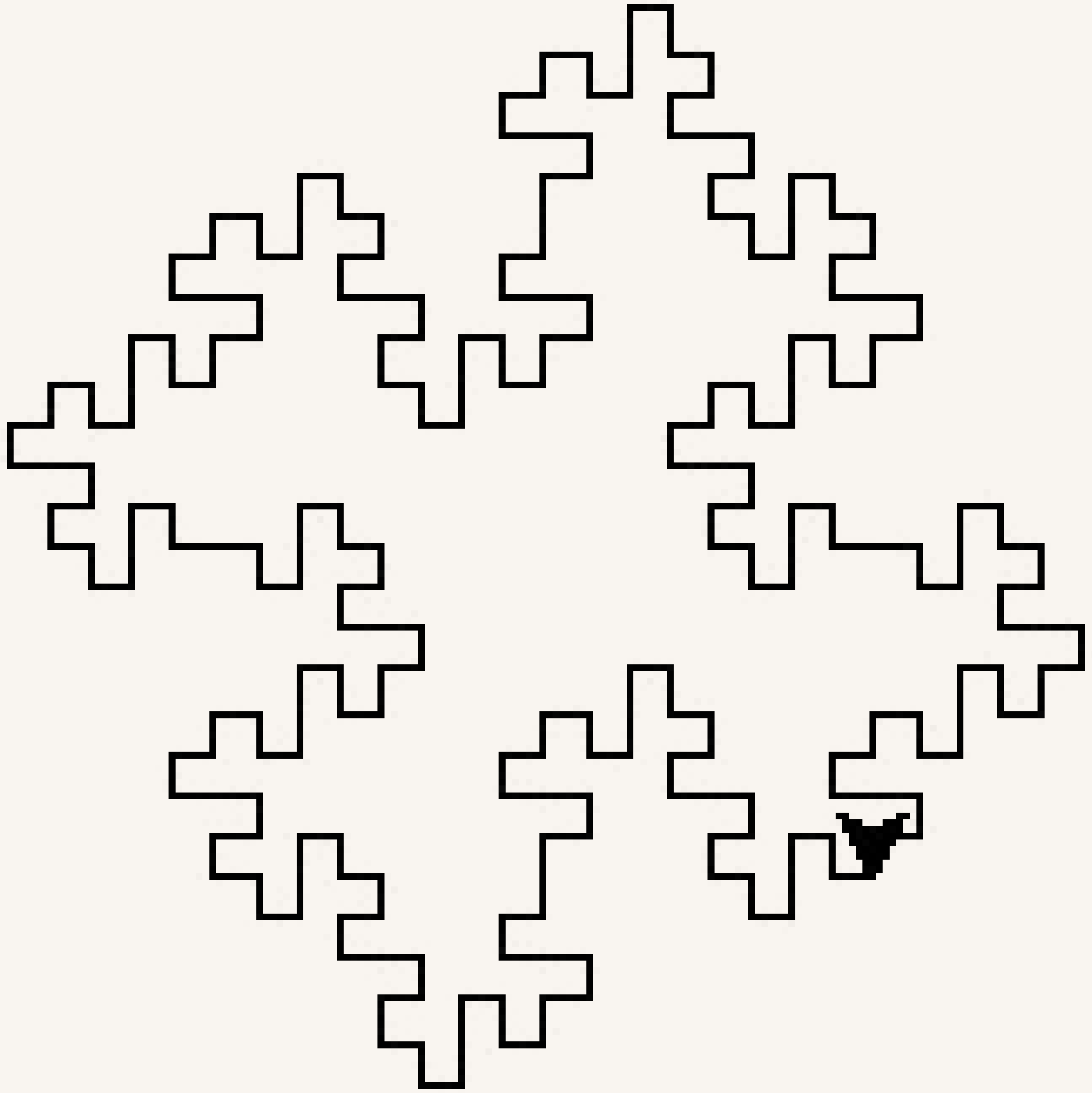
Với "axiom" ban đầu là "A", chúng ta có thể áp dụng quy tắc và lặp lại để tạo ra các chuỗi ký tự mới như sau:

- Lặp 0: A
- Lặp 1: AB
- Lặp 2: ABA
- Lặp 3: ABAAB
- Lặp 4: ABAABABA
- ...

Từ chuỗi kết quả có được ta có thể thực hiện vẽ đồ họa với Turtle

Ví dụ

- Tập ký hiệu: {F,+,-}
- Tiền đề S0 = F-F-F-F
- Tập sinh:
  - F = F-F+F+FF-F-F+F
- △α: 90
- Iteration: 2



Ví dụ

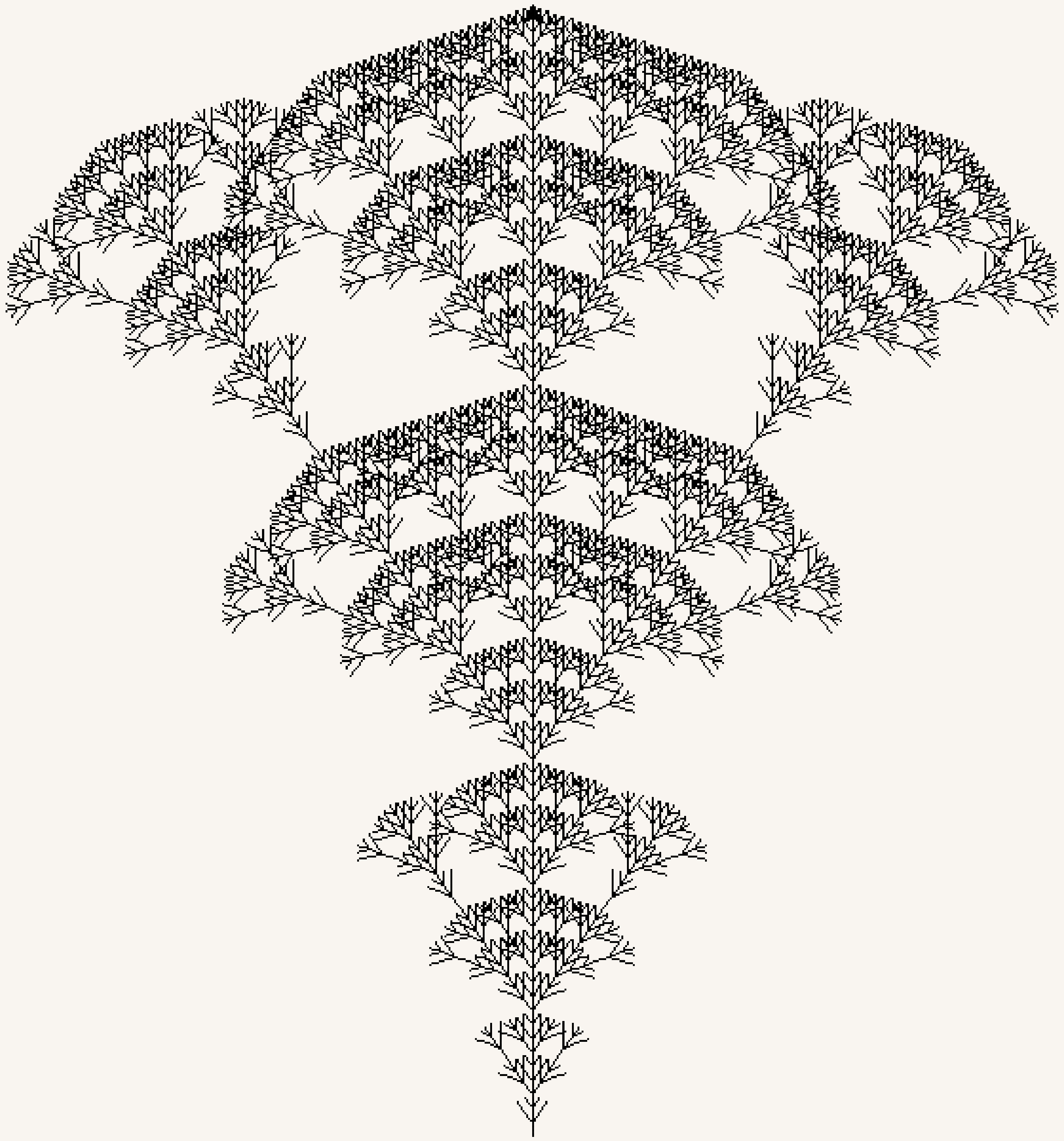
- Tập ký hiệu: {F,+,-,[,]}
- Tiền đề S0 = F
- Tập sinh:
  - F = FF+[+F-F-F]-[-F+F+F]
- △α: 25
- Iteration: 5

Ví dụ

- Tập ký hiệu: {F,+,-,[,],X}
- Tiền đề S0 = X
- Tập sinh:
  - F = FF
  - X = F-[[X]+X]+F[+FX]-X
- △α: 22.5
- Iteration: 5

Ví dụ

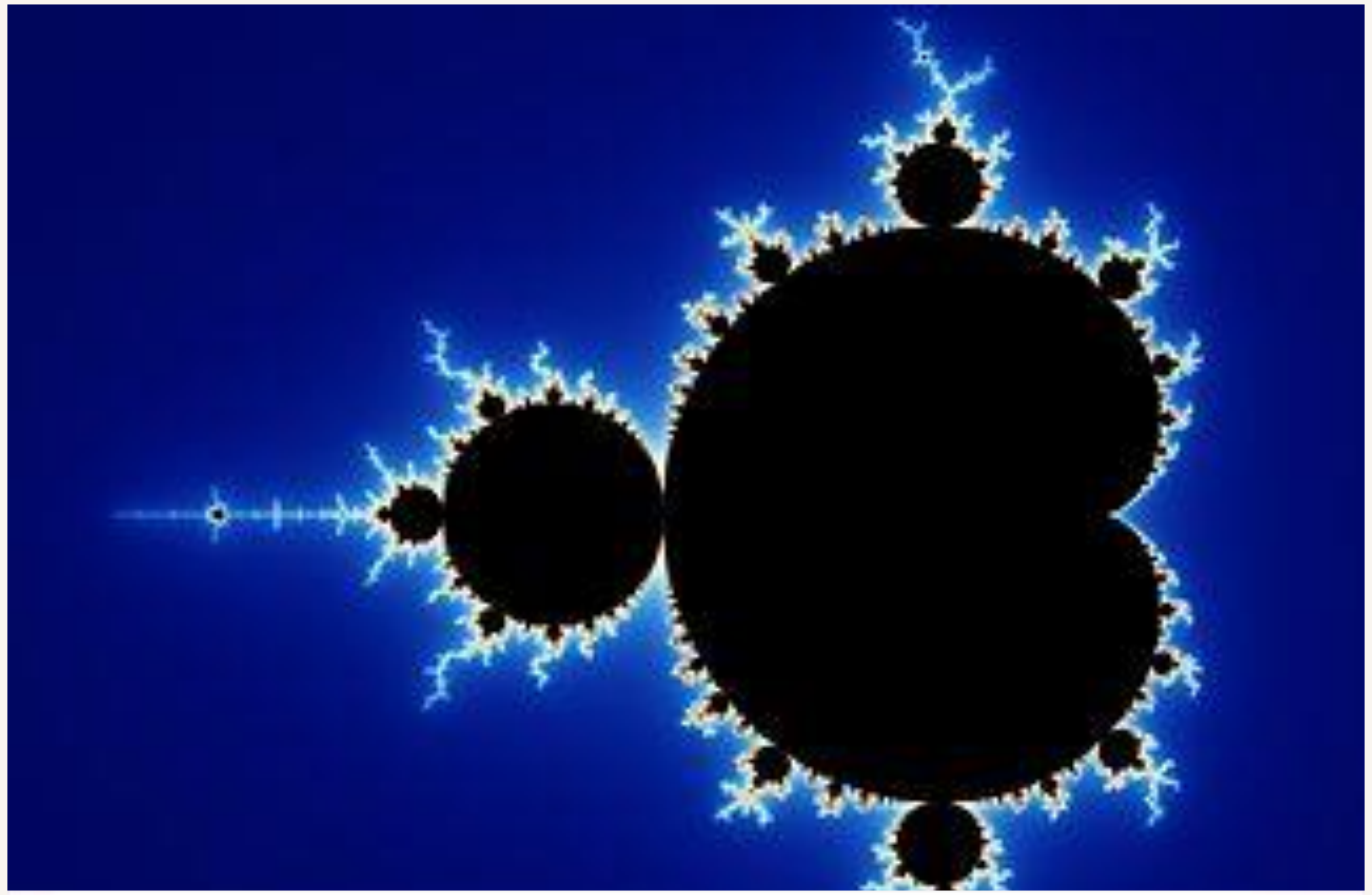
- Tập ký hiệu: {F,+,-,[,],}
- Tiền đề S0 = F
- Tập sinh:
  - F = F[+FF][-FF]F[-F][+F]F
- △α: 35
- Iteration: 4

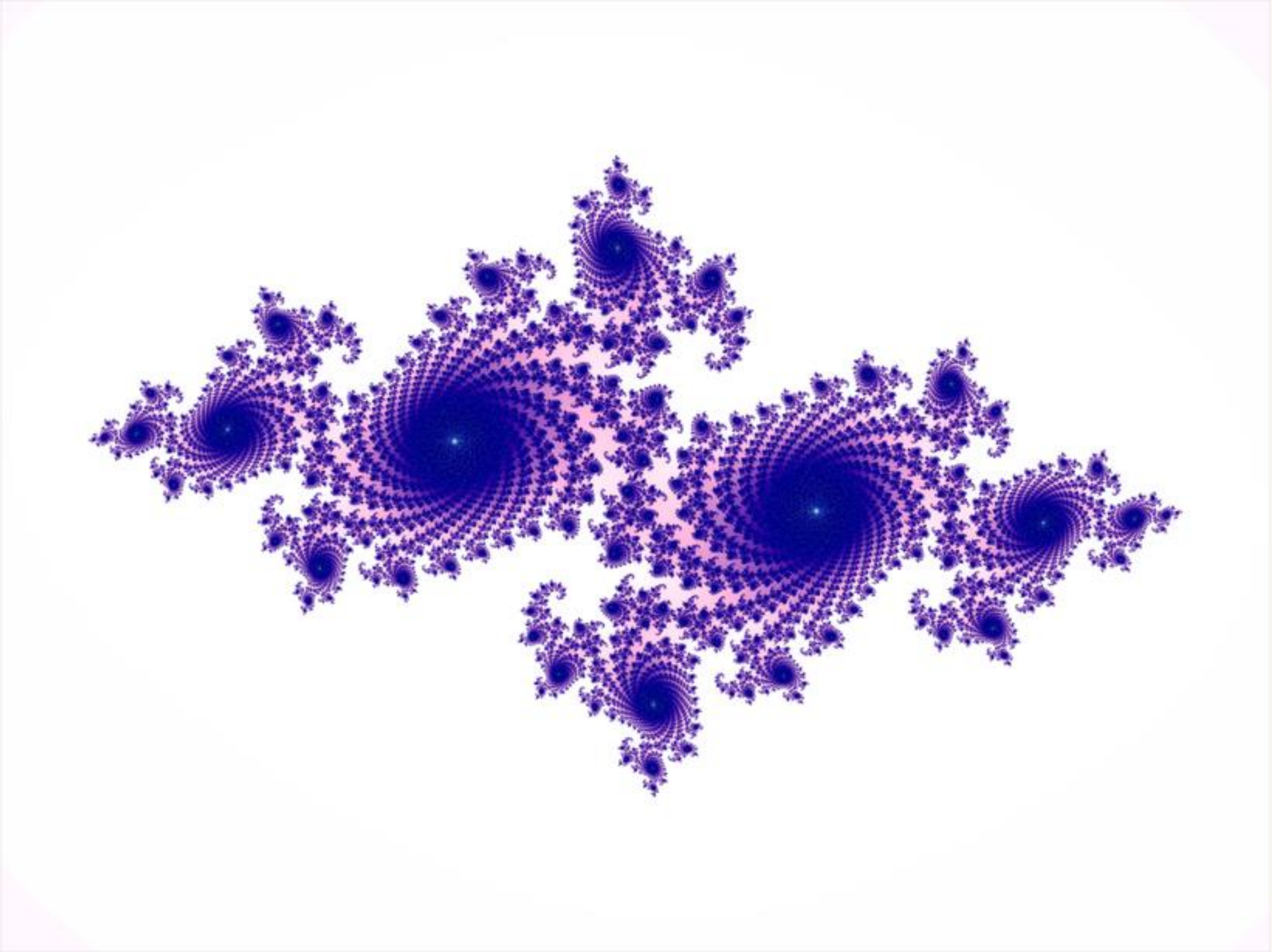


III. HÌNH HỌC FRACTAL TỪ SỐ PHỨC

1. Khái niệm

- Fractal từ số phức là dạng fractal được tạo ra bằng các phép toán đặc biệt và phức tạp, có thể tự phát sinh
- Có cấu trúc phức tạp, không thể tính toán một cách chính xác
- Hai ví dụ phổ biến cho dạng này là tập hợp Mandelbrot và Julia





2. Tập hợp Mandelbrot

Là tập hợp các giá trị của số phức c để phép lặp sau bị chặn

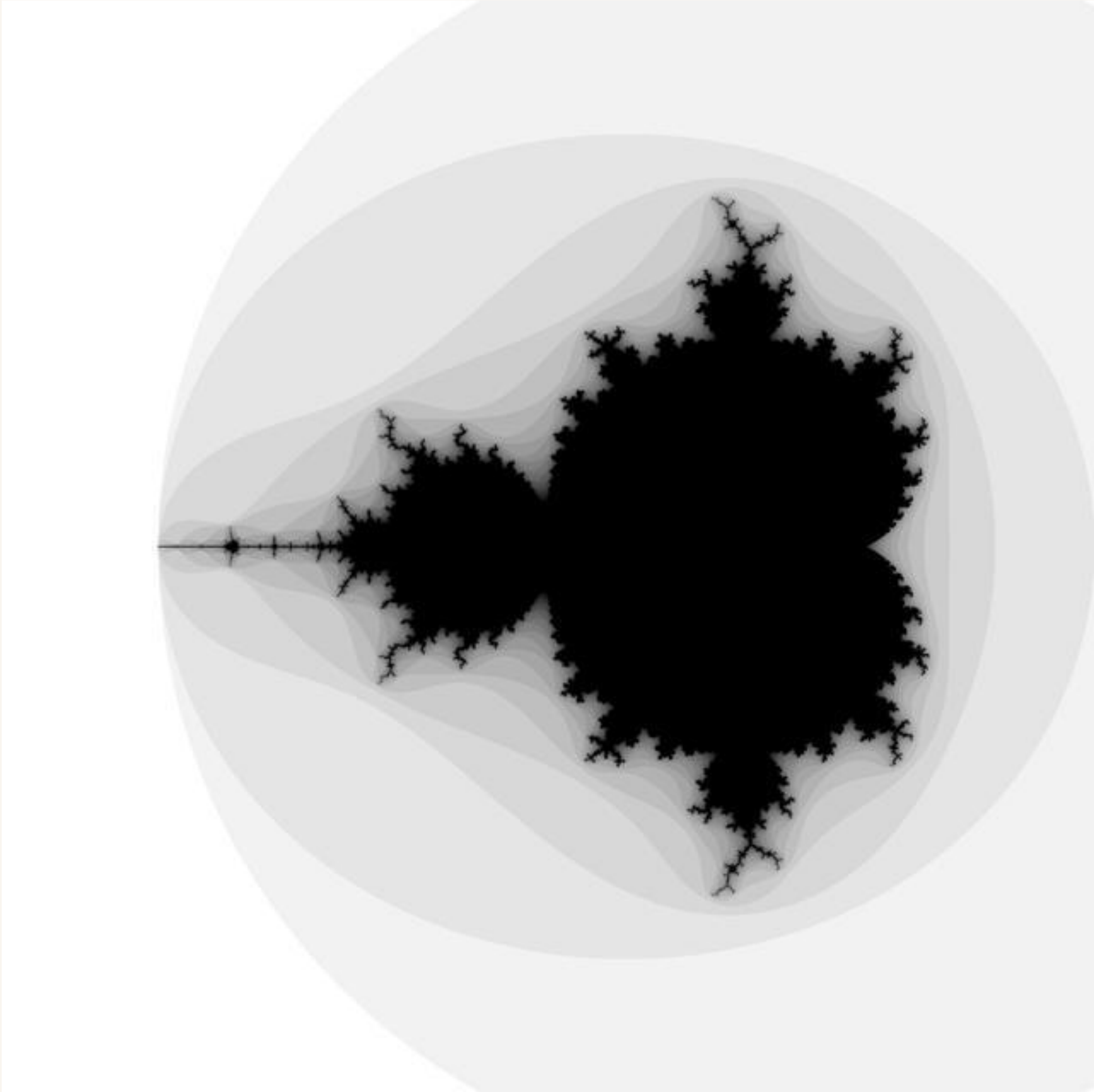
$$\begin{cases} z_0 = 0 \\ z_{n+1} = z_n^2 + c \end{cases}$$

Trong đó:

- **z0 = 0** là ***giá trị khởi tạo*** **của dãy z**
- **c** là số phức đại diện cho ***điểm cần xác định*** **thuộc tập Mandelbrot hay không**

Element	c = -1	c = 0	c = 1
z_0	0	0	0
z_1	-1	0	1
z_2	0	0	2
z_3	-1	0	5
z_4	0	0	26
z_5	-1	0	677
z_6	0	0	458,330
z_7	-1	0	210,066,388,901

- Mỗi điểm trên tập hợp Mandelbrot tương ứng một số phức c
- Ngưỡng giá trị để xác định chặn thường là 2
- Màu sắc của một điểm thể hiện số lần lặp lại phép toán trước khi giá trị z vượt quá ngưỡng



3. Tập hợp Julia

Là tập hợp các giá trị của số phức z0 để phép lặp sau bị chặn

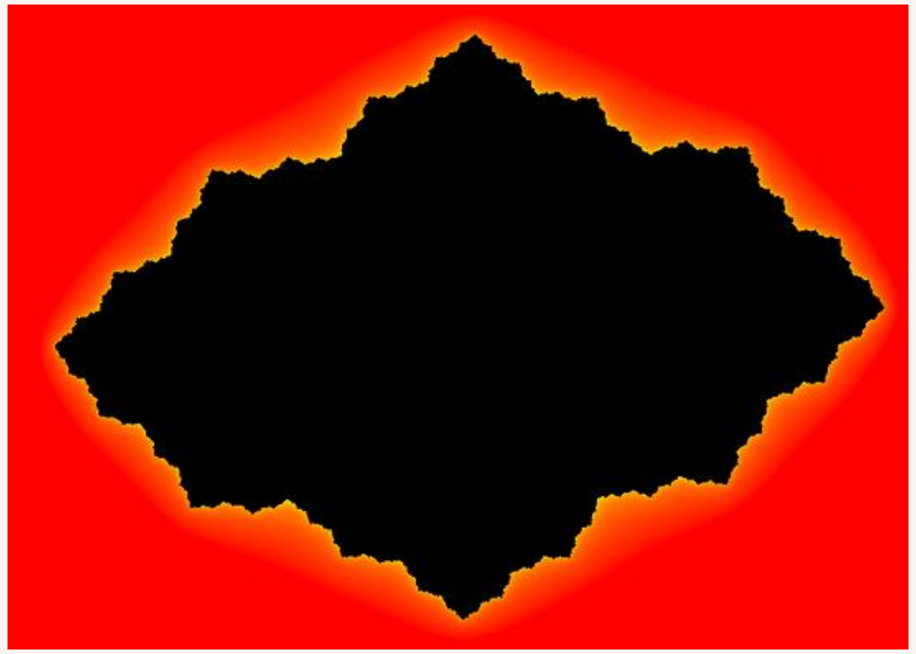
$$z_{n+1} = z_n^2 + c$$

Trong đó:

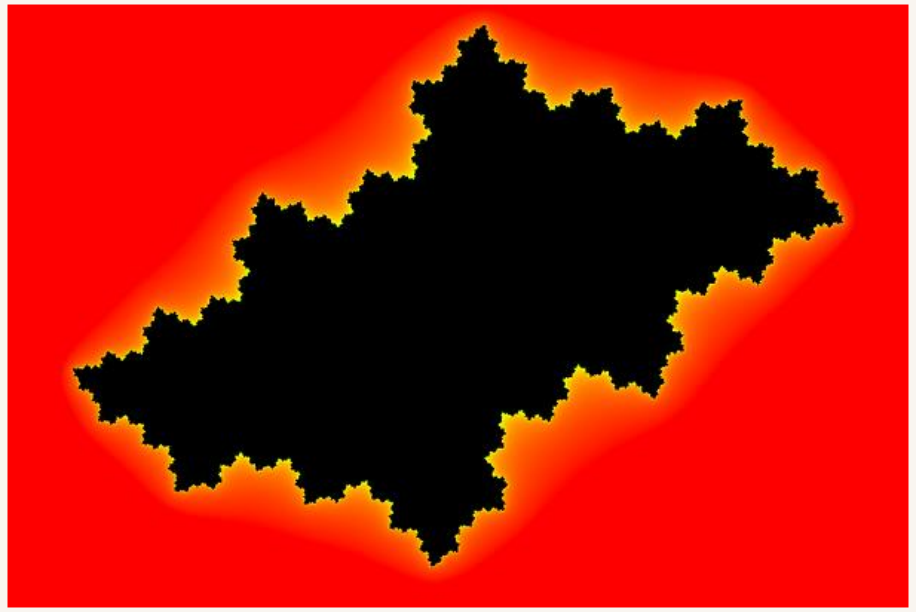
- **zn là số phức**
- **z0 là giá trị khởi tạo của dãy z, được gán bằng giá trị *số phức cần kiểm tra* thuộc tập Julia không**
- **c là *hằng số* được cung cấp sẵn**

Quá trình xác định tập hợp Julia được thực hiện bằng cách lặp lại công thức trên cho mỗi điểm trên mặt phẳng phức và kiểm tra xem giá trị của z có hội tụ hay không.

- Nếu giá trị của z vượt quá một ngưỡng nhất định sau một số lần lặp , điểm đó được coi là không thuộc tập hợp Julia.
- Ngược lại, nếu giá trị của z không vượt quá ngưỡng, điểm đó thuộc tập hợp Julia và có thể được tô màu tương ứng



c = -0.45 + 0.1i



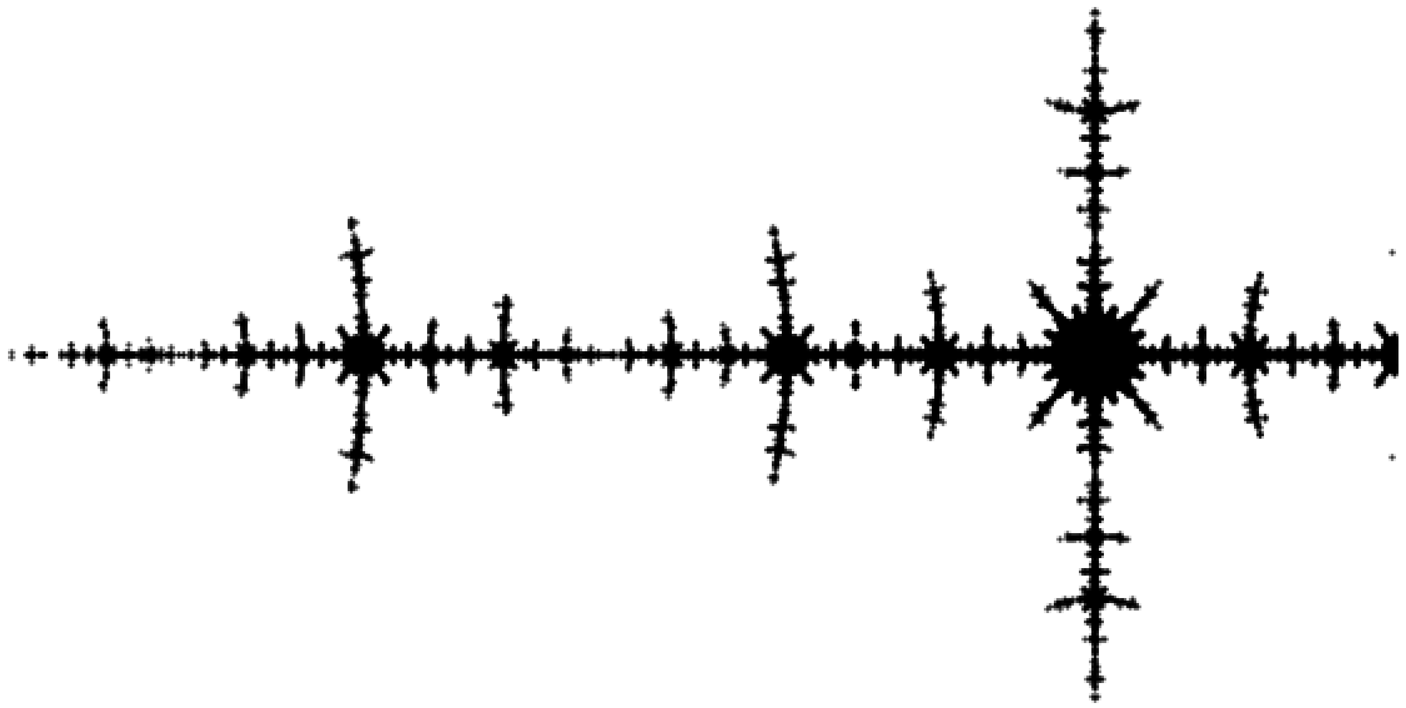
c = -0.45 + 0.45i

Công thức lặp của tập Mandelbrot và Julia là giống nhau. Tuy nhiên:

- Mandelbrot: khởi tạo z0=0 và gán giá trị cần kiểm tra cho c. Mỗi pixel dùng c khác nhau
- Julia: khởi tạo z0 bằng giá trị cần kiểm tra, cung cấp giá trị sẵn cho c. Gía trị c cố định cho mọi pixel

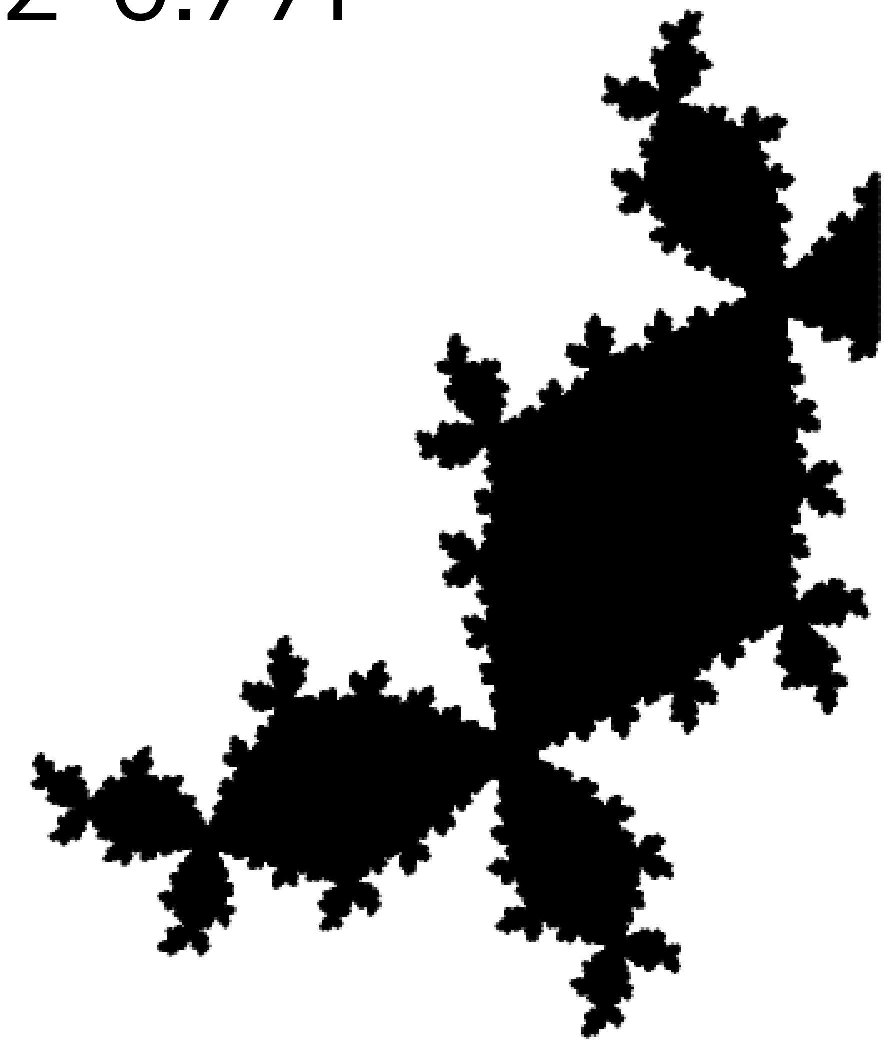
Gía trị	**Mandelbrot**	**Julia**
z0	0	Gía trị ứng viên
c	Gía trị ứng viên	Hằng số

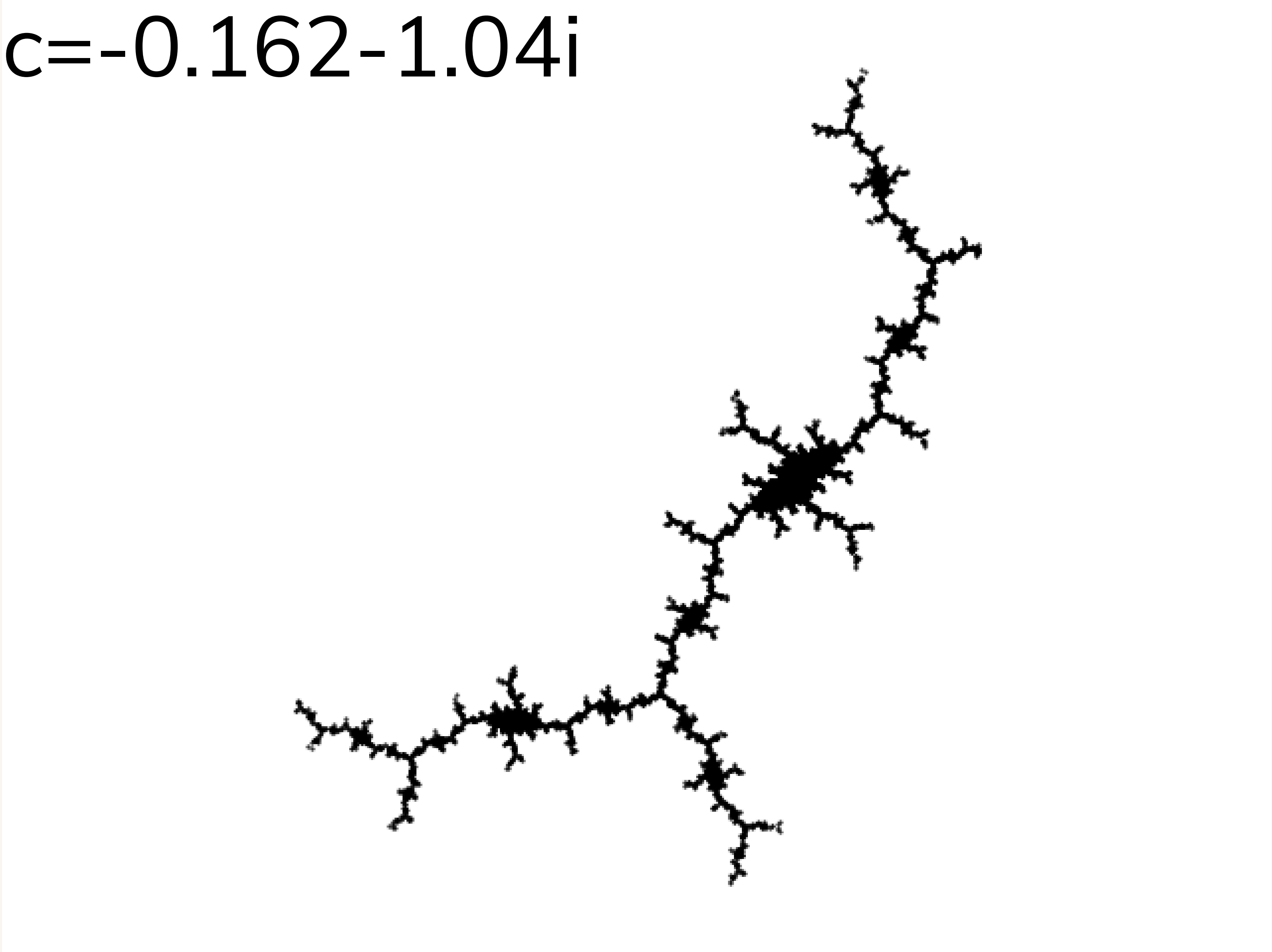
- Trong tập hợp Julia, c được xem như một tham số xác định hình dạng của tập hợp Julia.
- Mỗi số phức c sẽ tạo ra một tập hợp Julia khác nhau.



c=-1.476

c= -0.12-0.77i



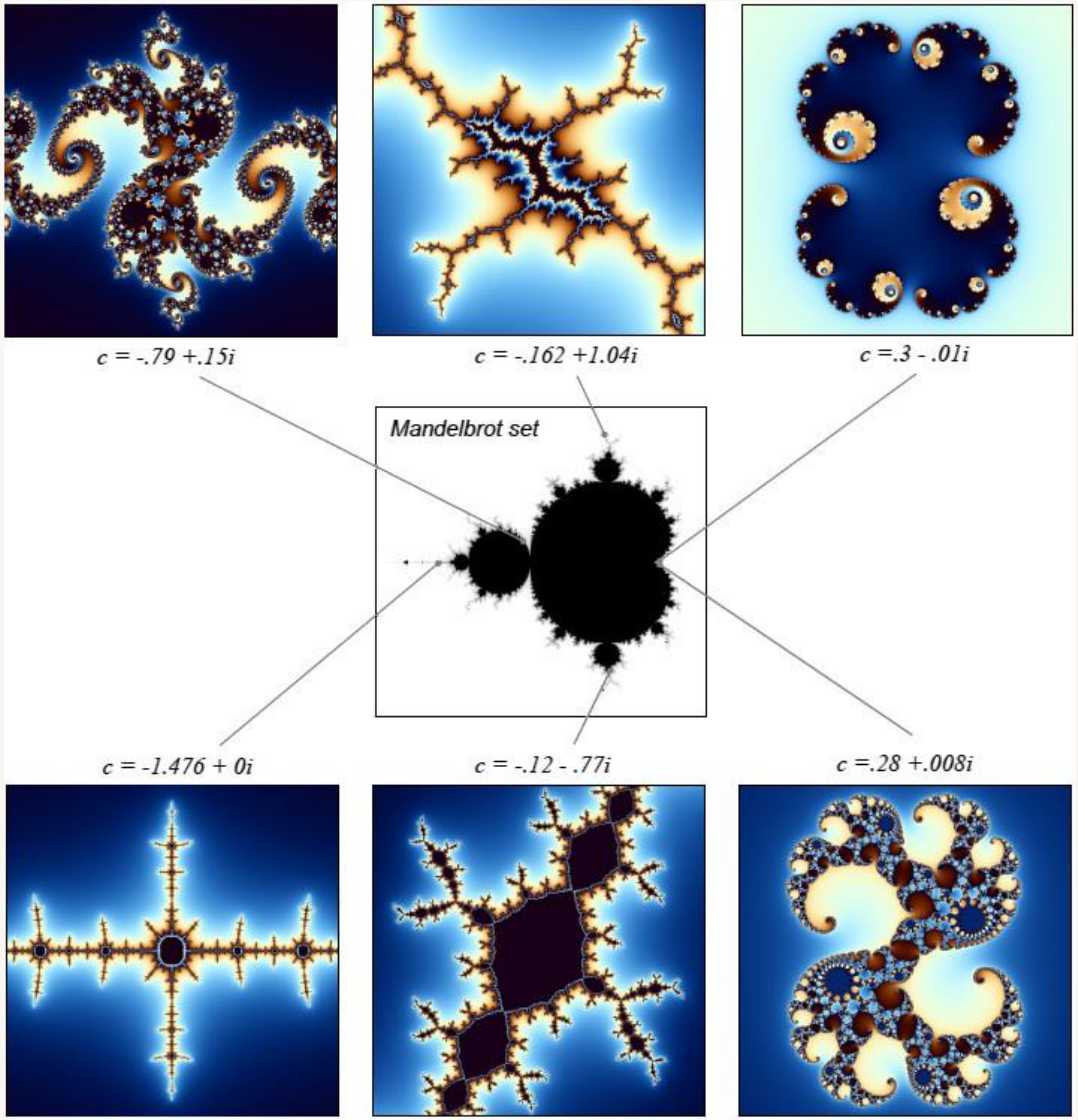

c=-0.162-1.04i


c=-0.79-0.01i

4. Mối liên hệ giữa Mandelbrot và Julia

Thực tế, tập hợp Julia có thể được sử dụng để tạo ra các thành phần trong ảnh của tập hợp Mandelbrot. Để vẽ tập hợp Mandelbrot thì cần xác định các điểm trên mặt phẳng phức thỏa mãn yêu cầu bị chặn. Ta sẽ sử dụng tập hợp Julia để xác định các tham số c phù hợp bằng cách:

- Chọn một điểm trên mặt phẳng phức làm giá trị khởi tạo của z và dùng nó làm tham số c cho Julia
- Nếu điểm đó thuộc tập hợp Julia, điểm đó sẽ thuộc tập hợp Mandelbrot



Các thành phần trong một tập Mandelbrot được tạo thành từ các tập Julia với tham số c khác nhau