TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

HÌNH HỌC FRACTAL

ĐỒ ÁN II

Chuyên ngành : TOÁN TIN

Chuyên sâu: Tin học

Giảng viên hướng dẫn: TS. Vương Mai Phương

Sinh viên thực hiện: Nguyễn Thị Bích Ngọc

MSSV: 20185388

Lớp: Toán - Tin 01 - K63

Chữ ký của GVHD

HÀ NỘI, THÁNG 03/2022

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. **Mục đích và nội dung của đồ án:**

2. **Kết quả đạt được:**

3. **Ý thức làm việc của sinh viên:**

Hà Nội, ngày tháng năm
Giảng viên hướng dẫn
(Ký và ghi rõ họ tên)

MỤC LỤC

LỜI NÓI ĐẦU

Trong những năm gần đây, toán học và khoa học tự nhiên đã bước lên một bậc thềm mới, sự mở rộng và sáng tạo trong khoa học trở thành một cuộc thử nghiệm liên ngành. Cho đến nay nó đã đưa khoa học tiến những bước rất dài. Hình học phân hình đã được đông đảo mọi người chú ý và thích thú nghiên cứu. Với một người quan sát tình cờ màu sắc của các cấu trúc phân hình cơ sở và vẻ đẹp của chúng tạo nên một sự lôi cuốn hình thức hơn nhiều lần so với các đối tượng toán học đã từng được biết đến. Hình học phân hình đã cũng cấp cho các nhà khoa học một môi trường phong phú cho sự thám hiểm và mô hình hóa tính phức tạp của tự nhiên. Những nguyên nhân của sự lôi cuốn do hình học phân hình tạo ra là nó đã chỉnh sửa được khái niệm lỗi thời về thế giới thực thông qua tập hợp các bức tranh mạnh mẽ và duy nhất của nó.

Những thành công to lớn trong các lĩnh vực của khoa học tự nhiên và kỹ thuật đãn đến sự ảo tưởng về một thế giới hoạt động như một cơ chể đồng hồ vĩ đại, trong đó các quy luật của nó chỉ còn phải chờ đợi để giải mã từng bước một. Một khi các quy luật đã được biết, người ta tin rằng sự tiến hóa hoặc phát triển của các sự vật sẽ được dự đoán trước chính xác hơn nhiều, ít ra là về mặt nguyên tắc. Những bước phát triển ngoạn mục đầy lôi cuốn trong lĩnh vực kỹ thuật máy tính và sự hứa hẹn cho việc điều khiển thông tin nhiều hơn nữa của nó đã làm gia tăng hy vọng của nhiều người về máy móc hiện có và cả những máy móc ở tương lai. Nhưng ngày nay người ta đã biết chính xác dựa trên cốt lõi của khoa học hiện đại là khả năng xem xét tính chính xác các phát triển ở tương lai như thế sẽ không bao giờ đạt được. Một kết luận có thể thu được từ các lý thuyết mới còn rất non trẻ đó là: giữa sự xác định có tính nghiêm túc với sự phát triển có tính ngẫu nhiên không những không có sự loại trừ lẫn nhau mà chúng còn cùng tồn tại như một quy luật trong tự nhiên. Hình học phân hình và lý thuyết hỗn độn xác định kết luận này. Khi xét đến sự phát triển của một tiến trình trong một khoảng thời gian, chúng ta sử dụng các thuật ngữ của lý thuyết hỗn độn, còn khi quan tâm nhiều hơn đến các dạng có cấu trúc mà một tiến trình hỗn độn để lại trên đường đi của nó, chúng ta dùng các thuật ngữ của hình học phân hình là bộ mô hình học cho phép "sắp xếp thứ tự" sự hỗn độn. Trong ngữ cảnh nào đó hình học phân hình là ngôn ngữ đầu tiên để mô tả, mô hình hóa và phân tích các dạng phức tạp đã tìm thấy trong tự nhiên. Nhưng trong khi các phần tử của ngôn ngữ truyền thống (Hình học Euclide) là các dạng hiển thị cơ bản như đoạn thẳng,

đường tròn và hình cầu thì trong hình học phân hình đó là các thuật toán chỉ có thể biến đổi thành các dạng và cấu trúc nhờ máy tính.

Việc nghiên cứu ngôn ngữ hình học tự nhiên này mở ra nhiều hướng mới cho khoa học cơ bản và ứng dụng. Trong đề tài này chỉ mới thực hiện nghiên cứu một phần rất nhỏ về hình học phân hình và ứng dụng của nó. Nội dung của đề tài gồm có ba chương được trình bày như sau:

Chương 1: Trình bày các kiến thức tổng quan về lịch sử, ứng dụng hình học phân hình, về các kết quả của cơ sở lý thuyết

Chương 2: Trình bày các kỹ thuật hình học phân hình thông qua sự khảo sát các cấu trúc Fractal cơ sở

Chương 3: Kết quả cài đặt chương trình vẽ một số đường mặt fractal

Nhân đây, em cũng xin chân thành cảm ơn cô **TS. Vương Mai Phương** đã tận tình hướng dẫn, chỉ dạy giúp dỡ em trong suốt thời gian thực hiện đề tài nghiên cứu này.

Đề tài được thực hiện trong một thời gian tương đối ngắn, nên dù em đã hết sức cố gắng hoàn thành đề tài nhưng chắc chắn sẽ không thể tránh khỏi những thiếu sót nhất định. Rất mong nhận được sự thông cảm và đóng góp những ý kiến vô cùng quý báu của các Thầy/Cô, bạn bè để hoàn thiện đề tài này hơn nữa nhằm tạo tiền đề thuận lợi cho việc phát triển đề tài trong tương lai.

Sinh viên thực hiện

Nguyễn Thị Bích Ngọc

CHƯƠNG 1: TỔNG QUAN VỀ HÌNH HỌC FRACTAL

1.1. Đặt vấn đề

Fractal là một thuật ngữ do nhà Toán học Mandelbrot đưa ra khi ông khảo sát những hình hoặc những hiện tượng trong thiên nhiên không có đặc trưng về độ dài. Năm 1982, nhà toán học thiên tài Mandelbrot nảy sinh ra ý tưởng về sự tồn tại của một môn "Hình học của tự nhiên", Fractal Geometry. Vậy hình học Fractal là gì? Fractal là cấu trúc thể hiện sự gần giống nhau về hình dạng của các hình thể kích cỡ khác nhau [1]. Nếu bạn nghiền một củ khoai tây rán dòn bạn sẽ có vô số những mảnh vỏ lớn nhỏ, các mảnh này có thể gọi là Fractal. Fractal là những vật thể hình học có cấu trúc nhưng quá bất thường để có thể mô tả bằng hình học Euclide. Một fractal bao gồm nhiều phần nhưng mỗi phần lại là một hình ảnh copy thu nhỏ của vật thể đó.

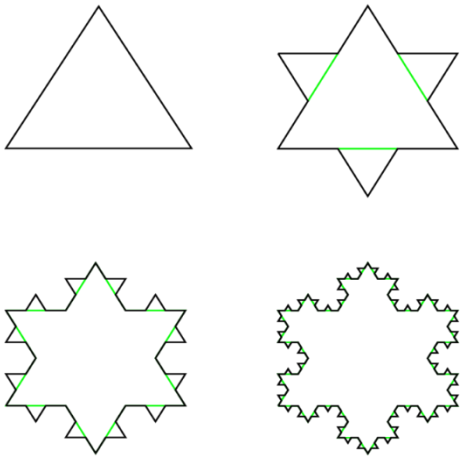
Lý thuyết hình học fractal được xây dựng dựa trên 2 vấn đề lớn được quan tâm ở những thập niên đầu thế kỷ 20. Các vấn đề đó bao gồm:

- Tính hỗn độn của các quá trình phát triển có quy luật trong tự nhiên.

- Sự mở rộng khái niệm số chiều và độ đo trong lý thuyết hình học Euclide cổ điển.

Hình học Euclid được giới thiệu ở trường trung học với việc khảo sát các hình đa giác, hình tròn, hình đa diện, hình cầu, hình nón…Hơn hai nghìn năm qua hình học Euclid đã có tác dụng to lớn đối với nền văn minh nhân loại, từ việc đo đạc ruộng đất đến vẽ đồ án xây dựng nhà cửa, chế tạo vật dụng và máy móc, từ việc mô tả quỹ đạo của các hành tinh trong hệ mặt trời đến mô tả cấu trúc của nguyên tử. Tuy nhiên, qua hình học Euclid ta nhìn mọi vật dưới dạng “đều đặn”, ”trơn nhẵn”. Với những hình dạng trong hình học Euclid ta không thể hình dung và mô tả được nhiều vật thể rất quen thuộc xung quanh như quả núi, bờ biển, đám mây, nhiều bộ phận trong cơ thể như mạch máu… là những vật cụ thể cực kỳ không đều đặn không trơn nhẵn mà rất xù xì, gồ ghề. Một ví dụ đơn giản: bờ biển quần đảo Trường Sa dài bao nhiêu? Ta không thể có được câu trả lời. Nếu dùng cách đo hình học quen thuộc dù thước đo có nhỏ bao nhiêu đi nữa ta cũng đã bỏ qua những lồi lõm giữa hai đầu của thước đo ấy, nhất là chỗ bờ đá nhấp nhô. Và với thước đo càng nhỏ ta có chiều dài càng lớn và có thể là vô cùng lớn.

Ví dụ 1: Bông tuyết Koch, được xây dựng từ tam giác đều Hình 1



Hình 1. Bông tuyết Koch

Ví dụ 2: Một số hình phân hình trong tự nhiên Hình 2

Hình 2. Một số hình ảnh về hình học fractal trong tự nhiên

1.2. Các ứng dụng của hình học fractal

- Ứng dụng [1] trong y học và sinh học:

Các nhà khoa học đã tìm ra các mối quan hệ giữa Fractal với hình thù của tế bào, quá trình trao đổi chất của cơ thể người, ADN, nhịp tim,... Trước đây, các nhà sinh học quan niệm lượng chất trao đổi phụ thuộc vào khối lượng cơ thể người, nghĩa là nó tỉ lệ bậc 3 khi xem xét con người là một đối tượng 3 chiều. Nhưng với góc nhìn từ hình học Fractal, người ta cho rằng sẽ chính xác hơn nếu xem con người là một mặt Fractal với số chiều xấp xỉ 2.5, như vậy tỉ lệ đó không nguyên nữa mà là một số hữu tỷ.

Việc chuẩn đoán bệnh áp dụng hình học Fractal đã có những tiến bộ rõ rệt. Bằng cách quan sát hình dạng của các tế bào theo quan điểm Fractal, người ta đã tìm ra các bệnh lý của con người, tuy nhiên những lĩnh vực này vẫn còn mới mẻ, cần phải được tiếp tục nghiên cứu.

- Ứng dụng trong hoá học:

Hình học Fractal được sử dụng trong việc khảo sát các hợp chất cao phân tử. Tính đa dạng về cấu trúc polyme thể hiện sự phong phú về các đặc tính của hợp chất cao phân tử chính là các Fractal. Hình dáng vô định hình, đường bẻ gảy, chuỗi, sự tiếp xúc của bề mặt polyme với không khí, sự chuyển tiếp của các sol-gel,... đều có liên quan đến các Fractal.

Sự chuyển động của các phân tử, nguyên tử trong hợp chất, dung dịch, các quá trình tương tác giữa các chất với nhau,...đều có thể xem như một hệ động lực hỗn độn (chaos).

- Ứng dụng trong vật lý:

Trong vật lý, khi nghiên cứu các hệ cơ học có năng lượng tiêu hao (chẳng hạn như có lực ma sát) người ta cũng nhận thấy trạng thái của các hệ đó khó xác định trước được và hình ảnh hình học của chúng là các đối tượng Fractal.

- Dự báo thời tiết:

[1] Viết cho những chuyển động của cuộc sống quanh tôi: Giới thiệu về hình học Fractal (buivanhuy1991.blogspot.com)

Hệ thống dự báo thời tiết được coi là một hệ động lực hỗn độn (chaos). Nó không có ý nghĩa dự đoán trong một thời gian dài (một tháng, một năm) do đó quy luật biến đổi của nó tuân theo qui luật Fractal.

- Thiên văn học:

Các nhà khoa học đã tiến hành xem xét lại các quỹ đạo của các hành tinh trong hệ mặt trời cũng như trong các hệ thiên hà khác. Một số kết quả cho thấy không phải các hành tinh này quay theo một quỹ đạo Ellipse như trong hình học Eulide mà nó chuyển động theo các đường Fractal. Quỹ đạo của nó được mô phỏng bằng những quỹ đạo trong các tập hút "lạ".

- Kinh tế:

Mô tả sự biến động của giá cả trên thị trường chứng khoán bằng các đồ hình Fractal sẽ cho phép chúng ta theo dõi sự biến động của giá cả. Trên cơ sở đó dự báo giá cả trên thị trường dựa theo các luật của hình học Fractal.

- Ứng dụng trong khoa học máy tính:

Hình học Fractal có thể giúp thiết kế các hình ảnh đẹp trên máy tính một cách đơn giản và trực quan. Đây là một trong những lĩnh vực được nhiều người quan tâm, nhất là đối với những người yêu mến nghệ thuật [1,7,9]. Bằng chứng là triển lãm tranh mang tên "Frontiers of Chaos: Images of Complex Dynamical System" của các nhà toán học người Đức H. Jurgens, H. O. Peitgen, M. Prufer, P. H. Richter và D. Saupe đã thu hút hơn 140.000 lượt khách tới tham dự. Từ năm 1985 đến nay triển lãm tranh đã đi qua hơn 150 thành phố của hơn 30 nước trên thế giới.

Cơ sở hình học Fractal cũng đã được ứng dụng trong công nghệ nén ảnh một cách hiệu quả thông qua các hệ hàm lặp (IFS), đây là một trong những lĩnh vực được các chuyên gia về khoa học máy tính đặc biệt quan tâm.

- Các lĩnh vực khác:

Fractal được ứng dụng trong việc đo chiều dài đường bờ biển chính xác hơn so với hình học Eulide bởi vì đường bờ biển là một hình Fractal. Fractal còn được sử dụng để mô tả các hình ảnh nhấp nhô của đồi núi, khảo sát các vết nứt chấn động địa chấn, các sự biến đổi trong lòng đất và dự báo sự biến động của địa chất.

Trong âm nhạc hình học Fractal cũng được đưa vào ứng dụng, các điểm hút, điểm đẩy là cơ sở cấu thành các nốt nhạc Fractal...

1.3. Độ đo fractal

a. Số chiều [2] tự đồng dạng (số chiều Hausdorff – Bescovitch):

Định nghĩa: Cho trước một cấu trúc tự đồng dạng được chia thành N phần, hệ số thu nhỏ của mỗi phần so với cấu trúc ban đầu là r. Ký hiệu D_s là đại lượng xác định bởi:

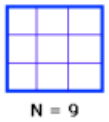
$$D_s = \frac{\log N}{\log 1/r}$$

Khi đó D_s được gọi là số chiều tự đồng dạng của cấu trúc đó.

Ví dụ:

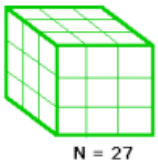
- Xét 1 hình vuông được chia thành 9 hình vuông nhỏ với tỷ lệ đồng dạng là 1/3. Khi đó số chiều tự đồng dạng của hình vuông ban đầu được xác định bởi:

$$D_s = \frac{\log 9}{\log \frac{1}{1/3}} = \frac{\log 3^2}{\log 3} = 2$$



- Xét 1 khối lập phương được chia thành 27 khối lập phương nhỏ với tỷ lệ đồng dạng là 1/3. Khi đó số chiều tự đồng dạng của hình khối lập phương ban đầu được xác định bởi:

$$D_s = \frac{\log 27}{\log \frac{1}{1/3}} = \frac{\log 3^3}{\log 3} = 3$$



b. Số chiều Compa:

Số chiều được xác định theo định nghĩa này được áp dụng cho các đường cong không phải là các đường cong tự đồng dạng hoàn toàn (như các đường bờ biển, các con sông,...), nhưng có thể sử dụng nhiều đơn vị khác nhau để xác định độ dài của chúng.

[2] Nghiên cứu về hình học fractal viết chương trình cài đặt một số đường và mặt fractal WORD slide (khotrithucso.com)

https://www.dam.brown.edu/math-coop/presentations/Fractals.pdf

Định nghĩa: Xét một đường cong không tự đồng dạng. Biểu diễn số đo của đường cong trên hệ tọa độ log/log với:

- Trục hoành: thể hện logarit của độ chính xác trong phép đo chiều dài đường cong. Độ chính xác được đặc tả bởi 1/s, với s là đơn vị đo độ dài. Ở đây giá trị s càng nhỏ thì độ chính xác của phép đo càng lớn.

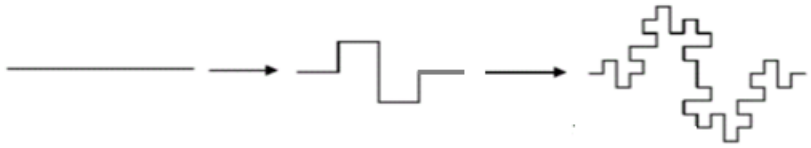
- Trục tung: thể hiện logarit của độ dài u đo được ứng với một đơn vị đo s.

- d: là hệ số góc của đường thẳng hồi quy dùng để xấp xỉ các giá trị đo u, đo được dựa trên phương pháp bình phương cực tiểu.

Ta có quan hệ: $\log u = d.\log\left(\frac{1}{s}+b\right)$, b là hệ số tự do.

Khi đó số chiều compa D_C được xác định bởi: $D_C = 1+d$

Ví dụ: Tìm số chiều của hình vẽ sau:



Biểu diễn các đại lượng có liên quan trên hệ tọa độ log/log đã được trình bày ở trên. Sau bước tạo sinh thứ k, đường cong gồm 8^k đoạn, mỗi đoạn có độ dài $s=1/4^k$ nên độ dài của đường cong sẽ là $8^k.1/4^k=2^k$

Khi đó giá trị trên trục hoành là: $\log_4 \frac{1}{1/4^k} = k$ ứng với giá trị trên trục tung là: $\log_4 2^k = k/2$. Từ đó ta xác định được d=0.5 do vậy: $D_C = 1+0.5 = 1.5$

c. Số chiều Box-Counting:

Số chiều xác định theo định nghĩa này được áp dụng cho các đường cong fractal không thể xác định số chiều theo 2 cách trên. Cách tính số chiều này có thể áp dụng cho mọi cấu trúc trong mặt phẳng và mở rộng cho cấu trúc trong không gian.

Định nghĩa: Xét một cấu trúc fractal bất kỳ. Lần lượt đặt cấu trúc này lên một dãy các lưới có kích thước ô lưới s giảm liên tiếp theo tỉ lệ ½. Gọi N(s) là các ô lưới có kích thước s có chứa một phần cấu trúc. Ta xây dựng hệ tọa độ log/log như sau:

- Trục hoành biểu thị giá trị của đại lượng $\log_2(1/s)$

- Trục tung biểu thị giá trị của đại lượng $\log_2(N(s))$

- D_B là hệ số góc của đường thẳng hồi quy đối với tập hữu hạn các điểm (s, N(s)) của hệ tọa độ.

Khi đó ta có: $D_B = \frac{\log_2 N(2^{-(k+1)}) - \log_2 N(2^{-k})}{\log_2 2^{k+1} - \log_2 2^k} = \log_2 \frac{N(2^{-(k+1)})}{N(2^{-k})}$

D_B xác định như vậy được gọi là số chiều box-counting của cấu trấu fractal đã cho.

CHƯƠNG 2: MỘT SỐ ĐƯỜNG FRACTAL CƠ BẢN

2.1. Họ đường Von Kock

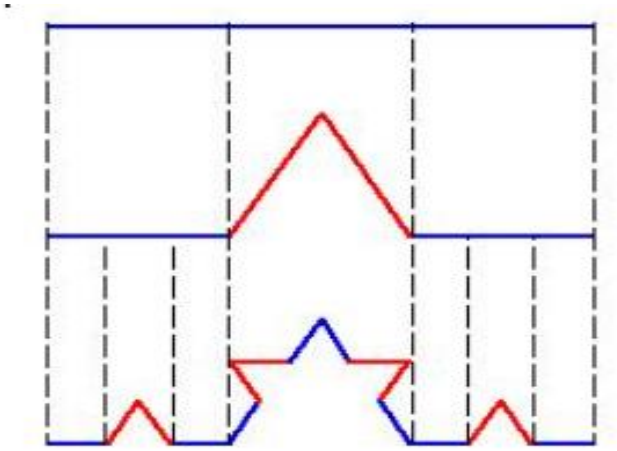
Trong phần này chúng ta sẽ cùng nhau thảo luận các hình học fractal được phát sinh bằng cách sử dụng đệ quy với kết quả là các hình tự đồng dạng hoàn toàn. Các hình này có số chiều được tính theo công thức của số chiều tự đồng dạng.

Chúng ta bắt đầu bằng một "hình khởi tạo", nó có thể là một đoạn thẳng hay một đa giác. Mỗi cạnh của hình khởi tạo được thay thế bởi một "hình phát sinh", mà là tập liên thông của các đoạn thẳng tạo nên bằng cách đi từ điểm bắt đầu đến điểm cuối của đường thay thế (thông thường các điểm của hình phát sinh là một lưới vuông hay một lưới tạo bởi các tam giác đều). Sau đó mỗi đoạn thẳng của hình mới được thay thế bởi phiên bản nhỏ hơn của hình phát sinh. Quá trình này tiếp tục không xác định được. Sau đây là một số đường Von Kock quan trọng:

2.1.1. Đường hoa tuyết Von Kock

Đường hoa tuyết được xây dựng bởi nhà toán học Thụy Điển Helge Von Kock vào năm 1904 [3]. Thuật toán lặp cho đường Von-Koch gồm những bước sau:

- Bắt đầu từ một đoạn thẳng cho trước rồi chia đoạn thẳng đó làm 3 phần bằng nhau
- Đoạn ở giữa được thay bởi 2 đoạn có chiều dài tương đương
- Mỗi đoạn trong số 4 đoạn này lại được thay bởi 4 đoạn mới có chiều dài bằng 1/3 đoạn trước
- Quá trình cứ thế lặp lại ta sẽ thu được đường hoa tuyết Von Kock dài vô hạn không có điểm dừng



Hình 3. Các bước tạo thành đường Von-Koch

Đặc điểm của đường Von-Koch:

[3] [PDF]Đồ họa máy tính - Chương 6: Hình học Fractal.pdf (tailieumienphi.vn)

• Đường cong này có tính tự tương tự, mỗi phần nhỏ khi được phóng to có thể tạo sinh lại giống như phần lớn hơn, nói cách khác nó bất biến với phép phóng to hình.

• Qua mỗi bước lặp, độ dài đường von-Koch tăng lên 4/3 lần

• Đường cong (sau vô hạn bước) có độ dài vô hạn mặc dù nó chỉ chiếm một phần diện tích hữu hạn của mặt phẳng

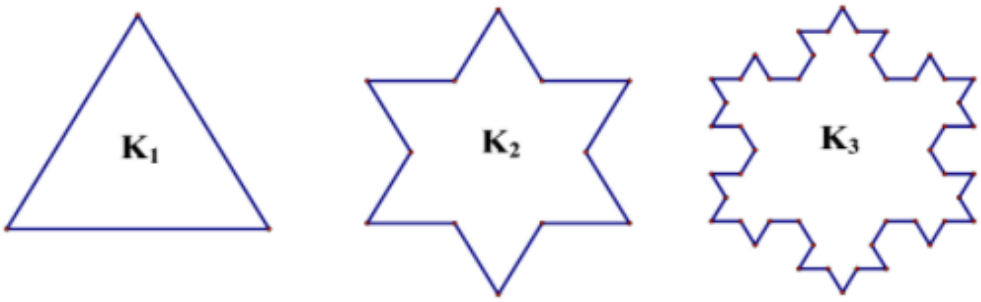
• Không tự cắt

• Thuật toán tạo sinh đường Von-Koch khá đơn giản, nhưng không có công thức đại số để xác định những điểm trên đường cong

• Số chiều của đường Von-Koch là vô tỷ: $D = \frac{\log N}{\log 1/r} = \frac{\log 4}{\log 3} \approx 1.2619$

2.1.2. Bông hoa tuyết Von Kock

Bông hoa tuyết Von-Koch [4] được xây dựng tương tự như đường hoa tuyết Von-Koch với khởi đầu là một tam giác đều K_1 có cạnh bằng a và mỗi cạnh là một đường hoa tuyết Von-Koch.



Hình 4. Các bước tạo thành bông hoa tuyết Von-Koch

Mỗi công đoạn cho ta một hình mới có số cạnh gấp 4 lần số cạnh ban đầu nên bông tuyết K_n có số cạnh là 3.4^{n-1}

Mỗi công đoạn lại làm độ dài mỗi cạnh giảm đi 3 lần nên bông tuyết K_n có độ dài cạnh là $a/3^{n-1}$

[4] https://documents.uow.edu.au/~mnelson/teaching.dir/pdf.dir/fractals.pdf

Koch snowflake - Wikipedia

nên độ dài (chu vi) của bông tuyết K_n sẽ là $p_n=3.4^{n-1}.a/3^{n-1}=3a(4/3)^{n-1}$

=> p_n là một cấp số nhân và lim $p_n=+\infty$

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

Gọi S_n là diện tích của hình K_n

Diện tích tam giác đều K_1 có cạnh bằng a là $S_1 = \frac{a^2\sqrt{3}}{4}$

Ta có: $S_2 - S_1 = 3.\left(\frac{S_1}{9}\right) = \frac{S_1}{3}$ (1)

$$S_3 - S_2 = 4.3.\left(\frac{S_1}{9^2}\right) = \frac{S_1}{3}.\left(\frac{4}{9}\right) \quad (2)$$

$$S_4 - S_3 = 4^2.3.\left(\frac{S_1}{9^3}\right) = \frac{S_1}{3}.\left(\frac{4}{9}\right)^2 \quad (3)$$

Bằng phương pháp quy nạp, ta được:

$$S_n - S_{n-1} = 4^{n-2}.3.\left(\frac{S_1}{9^{n-1}}\right) = \frac{S_1}{3}.\left(\frac{4}{9}\right)^{n-2} \quad \text{(n-1)}$$

Cộng từng vế n-1 đẳng thức trên, ta được:

$$S_n - S_1 = \frac{S_1}{3} + \frac{S_1}{3}.\left(\frac{4}{9}\right) + \frac{S_1}{3}.\left(\frac{4}{9}\right)^2 + ... + \frac{S_1}{3}.\left(\frac{4}{9}\right)^{n-2} \quad (*)$$

Vế phải của (*) là tổng của n-1 số hạng đầu tiên của cấp số nhân lùi vô hạn có số hạng đầu là $\frac{S_1}{3}$ và công bội là $\frac{4}{9}$. Tổng của cấp số nhân này là $\left(\frac{S_1}{3}\right).\frac{1}{1-\frac{4}{9}} = \frac{3S_1}{5}$

Do đó $\lim(S_n - S_1) = \frac{3S_1}{5}$, suy ra $\lim S_n = \frac{3S_1}{5} + S_1 = \frac{8S_1}{5} = \frac{8}{5}.\frac{a^2\sqrt{3}}{4} = \frac{2\sqrt{3}}{5}a^2$

Vậy diện tích của bông tuyết Von Koch là hữu hạn (nó chỉ gấp 1,6 lần diện tích tam giác đều ban đầu).

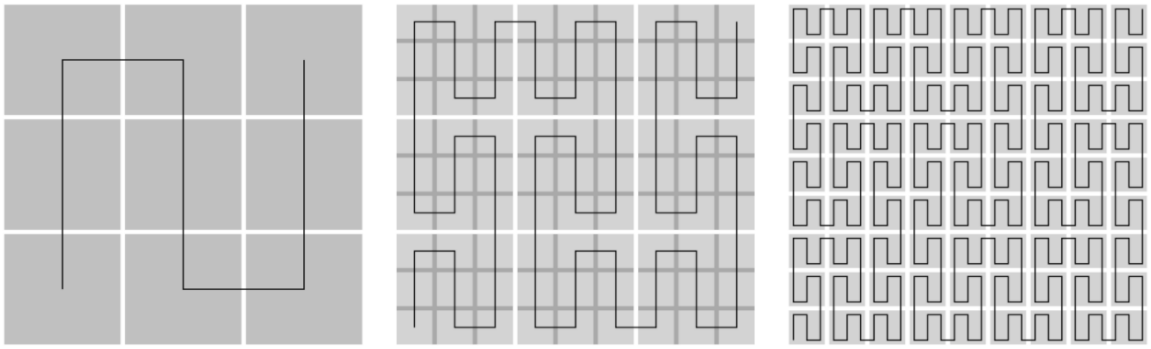
2.2. Đường cong Peano

Các đường này có số chiều bằng 2. Chúng được gọi là các đường cong Peano được phát hiện bởi Guiseppe Peano vào năm 1890 [5]. Do số chiều bằng 2 nên các đường này phải lấp đầy mặt phẳng. Cách xây dựng:

• Chia một hình vuông thành chín hình vuông nhỏ hơn bằng nhau, được cắt ngang theo đường cong S (đường cong S được hình thành bằng cách nhóm chín hình vuông nhỏ hơn thành ba cột, sắp xếp các tâm kề nhau trong mỗi cột, sau đó sắp xếp các cột từ cạnh này sang cạnh kia của hình vuông sao cho khoảng cách giữa mỗi cặp điểm liên tiếp có độ dài bằng cạnh của hình vuông nhỏ).

• Trong bước tiếp theo, mỗi hình vuông nhỏ ở bước trên lại được chia nhỏ thành chín hình vuông nhỏ hơn bằng nhau, được cắt ngang theo đường cong S và liên kết các đường cong S này lại với nhau để tạo thành một đường cong mới.

• Quá trình cứ thế lặp lại ta thu được tập hợp các đường cong S được nối với nhau trên cùng một mặt phẳng.

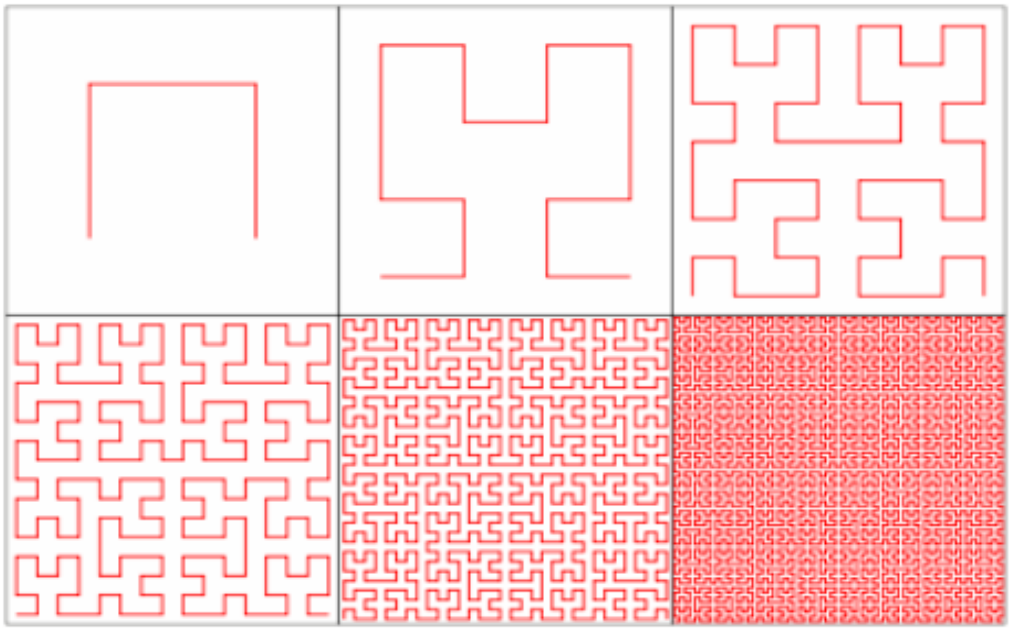


Hình 5. Các bước tạo thành đường cong Peano

Đường cong Hibert [6] là một biến thể đơn giản hơn, có cùng ý tưởng với đường cong Peano. Được xây dựng dựa trên việc chia nhỏ các hình vuông thành bốn hình vuông nhỏ hơn bằng nhau thay vì thành chín hình vuông nhỏ hơn bằng nhau. Đường cong Hilbert là đường cong lấp đầy không gian gãy liên tục được nhà toán học người Đức David Hilbert công bố năm 1891.

[5] Peano curve - Wikipedia

[6] Hilbert curve - Wikipedia



Hình 6. Các bước tạo thành đường cong Hibert

Độ dài của nó tại bước lặp thứ n là $L_n = 2^n - \frac{1}{2^n}$ tức là độ dài tăng theo hàm mũ đối với n và cũng có số chiều bằng 2.

2.3. Họ đường Sierpinski

Còn gọi là "**tam giác Sierpinski**"[7], bắt đầu từ một hình tam giác đều, đáy phẳng, nối liền trung điểm của ba cạnh thành một hình tam giác đều nhỏ rồi móc bỏ tam giác này. Cứ tiếp tục phân chia như vậy cho những tam giác nhỏ hơn cho đến khi không thể phân chia được nữa. Tam giác Sierpinski được đặt tên theo nhà toán học người Ba Lan Wacław Franciszek Sierpiński. Có thể áp dụng tương tự với khối hình vuông, hoặc khối hình hộp.

***Tam giác Sierpinski**:

Khởi đầu với một tam giác đều, ta thực hiện các bước sau:

[7] Sierpiński triangle - Wikipedia

https://documents.uow.edu.au/~mnelson/teaching.dir/pdf.dir/fractals.pdf

FRACTALS AND THE SIERPINSKI TRIANGLE (wpi.edu)

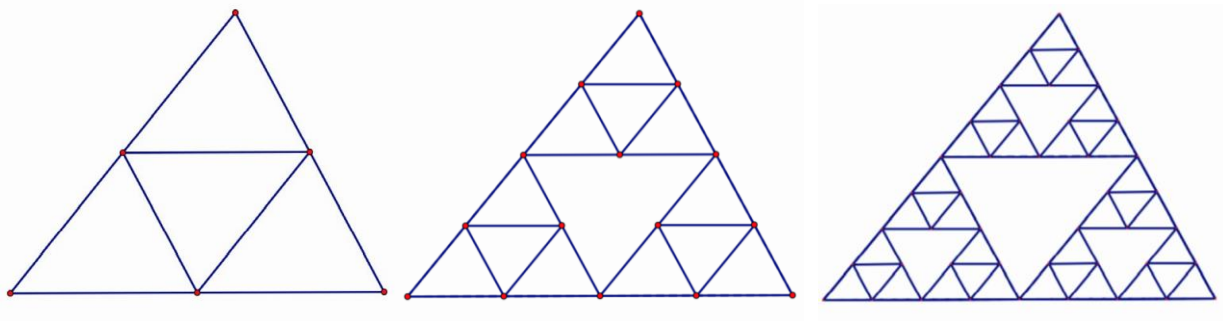
• Chia nhỏ tam giác này thành 4 tam giác đều bằng nhau, sau đó ta loại bỏ tam giác ở chính giữa (trung tâm).

• Lặp lại bước trên cho các tam giác vừa được chia ra (còn lại).

Hình 7. Các bước tạo thành tam giác Sierpinski

Số chiều: $D = \frac{\log N}{\log 1/r} = \frac{\log 3}{\log 2} \approx 1.585$

Xét trường hợp tam giác cho ban đầu không phải là tam giác đều thì tam giác Sierpinski được tạo thành như sau:



• Tìm các điểm giữa (trung điểm) của mỗi cạnh của tam giác và kết nối điểm

• Lặp lại quy trình với ba hình tam giác "bên ngoài" của hình. Tìm trung điểm của các cạnh của mỗi tam giác nhỏ hơn. Để trống tam giác ở giữa, nối các điểm giữa

• Quá trình cứ thế lặp lại

***Tấm thảm Sierpinski** [8]:

Tấm thảm Sierpinski được tạo ra như sau:

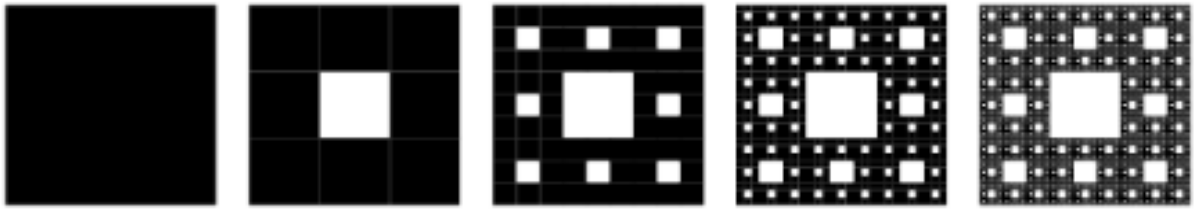
[8] Tấm thảm Sierpinski – Wikipedia tiếng Việt

• Đầu tiên cho một hình vuông.

• Ta sẽ chia hình vuông này thành 9 phần bằng nhau, rồi xóa phần chính giữa.

• Tiếp theo, ta lại chia 8 phần còn lại, mỗi phần được chia thành 9 phần, rồi xóa các phần chính giữa (nhỏ hơn) của chúng.

• Cứ như thế, ta được phần còn lại là tấm thảm Sierpinski.

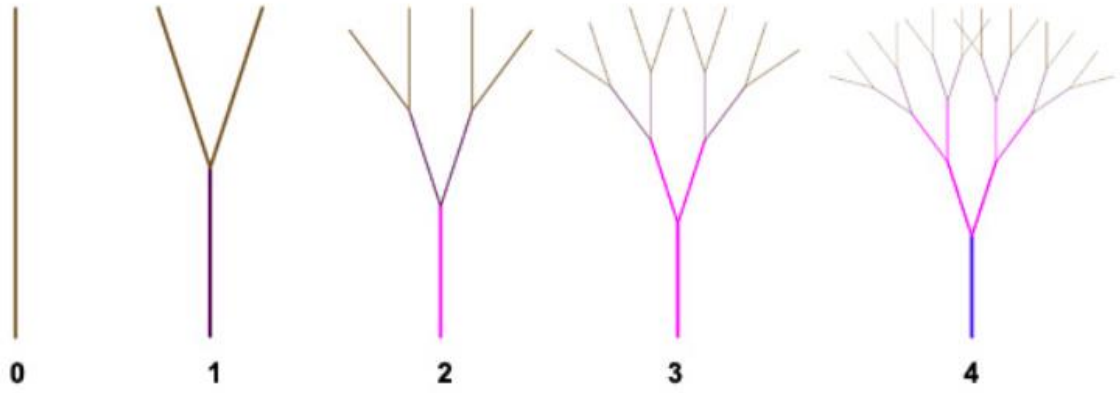


Hình 8. Các bước tạo thành tấm thảm Sierpinski

Số chiều: $D = \frac{\log N}{\log 1/r} = \frac{\log 8}{\log 3} \approx 1.893$

2.4. Cây fractal

Mỗi tán cây fractal được tạo ra bằng cách tách một đoạn thẳng thành hai đoạn thẳng nhỏ hơn (cây nhị phân đối xứng). Sau n lần tách, cây bao gồm $2^n + 1$ đoạn thẳng (bao gồm cả thân cây)



Hình 9. Các bước tạo thành cây fractal

Một tán cây fractal phải có ba đặc tính sau [9]:

- Góc giữa hai đoạn thẳng lân cận bất kỳ là như nhau trong suốt fractal.

- Tỷ số độ dài của hai đoạn thẳng liên tiếp bất kỳ là không đổi.

- Các điểm ở cuối các đoạn thẳng nhỏ nhất được nối với nhau. Có nghĩa là toàn bộ hình là một đồ thị liên thông.

2.5. Tập Mandelbrot

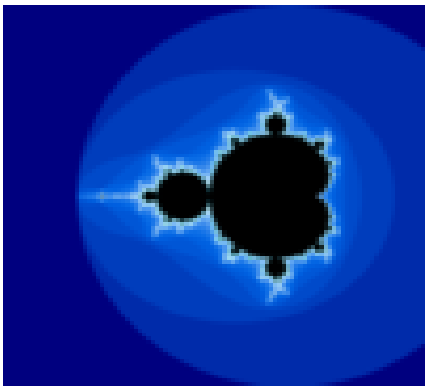
Tập Mandelbrot [10] được đặt tên theo nhà toán học Benoît Mandelbrot, người đầu tiên đã nghiên cứu và phát triển nó. Một số có hình dạng giống vật thiên nhiên (ví dụ lá cây, não, vỏ ốc, vi khuẩn, sấm sét, tia sáng, tuyết, sao biển...) nên hình học fractal cũng được gọi là hình học thiên nhiên.

Tập Mandelbrot là một tập hợp các số phức được xác định dựa trên công thức đệ quy sau:

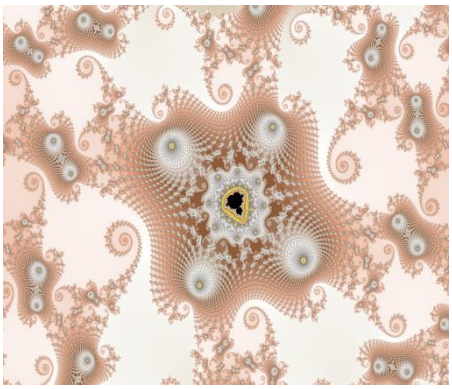
$$z_{n+1} = z_n^2 + c, \qquad z_0 = 0 \quad \forall c \in \mathbb{C}, |c| \le 2$$

và được phân chia theo bậc:

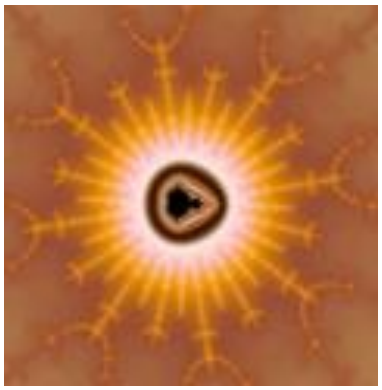
***Tập hợp Mandelbrot bậc hai:**



Tập hợp Mandelbrot bậc hai



Vòng sóng



Virus Corona

[9] Fractal canopy - Wikipedia

[10] Tập hợp Mandelbrot – Wikipedia tiếng Việt

http://www.3dfractals.com/docs/3DFractals.pdf

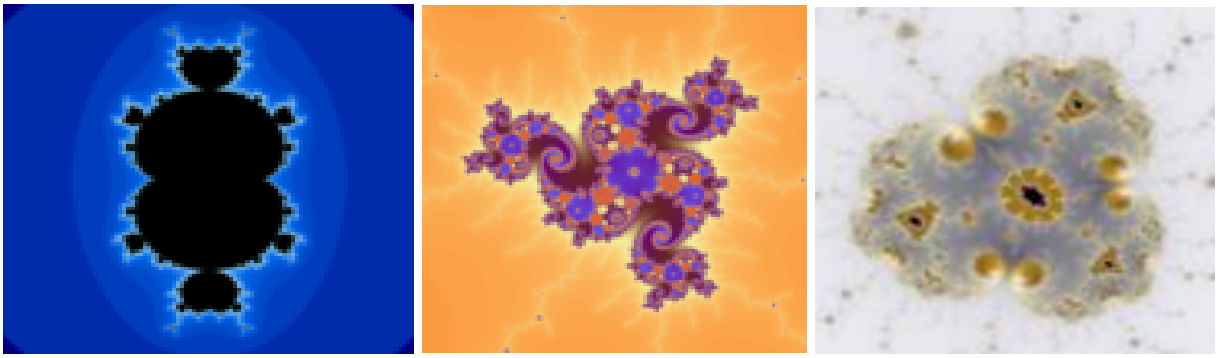
http://www.opentextbookstore.com/mathinsociety/2.5/Fractals.pdf

Ổ xoắn Bông tuyết Hào quang mặt trời

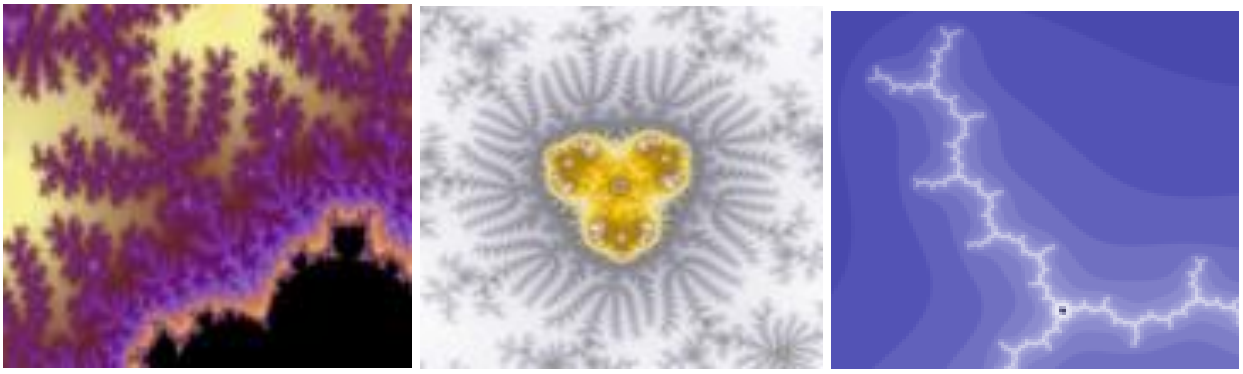
Hình 10. Một số hình ảnh về tập hợp Mandelbrot bậc hai

Tập hợp Mandelbrot bậc hai được tính bằng công thức đệ quy bậc 2 sau: $z_{n+1} = z_n^2 + c, \quad z_0 = 0 \quad \forall c \in \mathbb{C}, |c| \le 2$ và có một trục đối xứng là trục 0x.

***Tập hợp Mandelbrot bậc ba**



Tập hợp Mandelbrot bậc ba Vi khuẩn Mây bậc ba

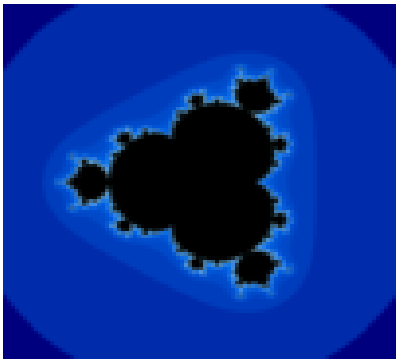


Dương xỉ cổ sinh Rừng rong biển Sấm sét

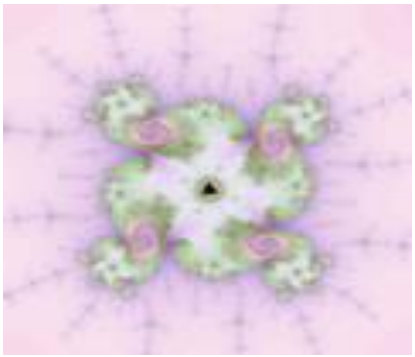
Hình 11. Một số hình ảnh về tập hợp Mandelbrot bậc ba

Tập hợp Mandelbrot bậc ba được tính bằng công thức đệ quy bậc 3 sau: $z_{n+1} = z_n^3 + c, \quad z_0 = 0 \quad \forall c \in \mathbb{C}, |c| \leq 2$ và có hai trục đối xứng cách nhau 90° (π/2 radian) là 2 trục 0x, 0y.

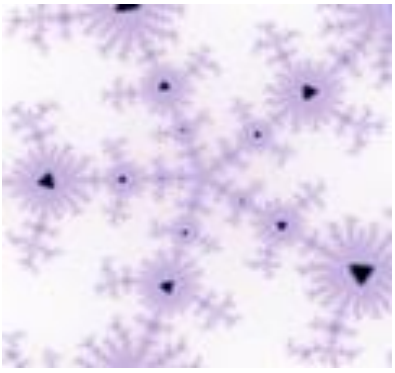
***Tập hợp Mandelbrot bậc bốn**



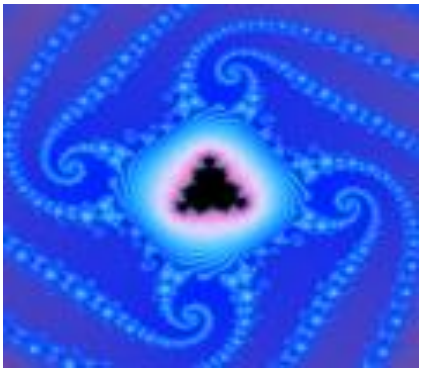
Tập hợp Mandelbrot bậc bốn



Vi khuẩn



Bão tuyết xoay

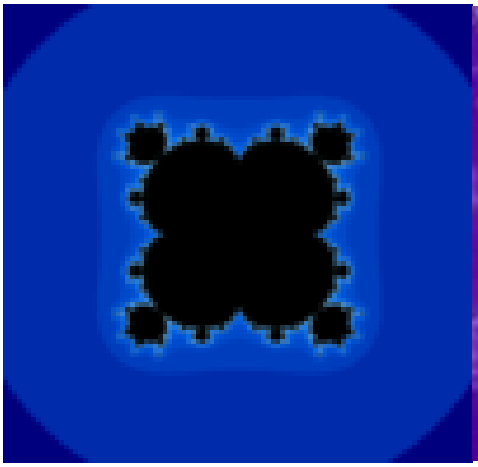


Bốn ốc xoay

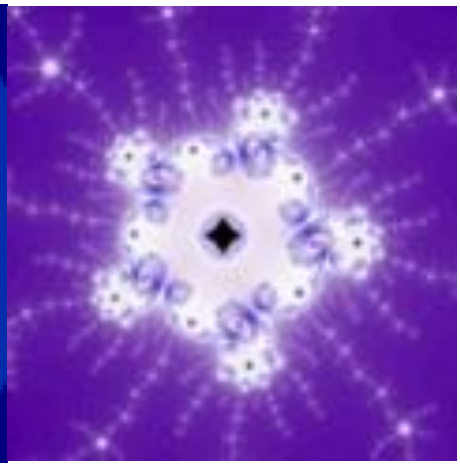
Hình 12. Một số hình ảnh về tập hợp Mandelbrot bậc bốn

Tập hợp Mandelbrot bậc bốn được tính bằng công thức đệ quy bậc 4 sau: $z_{n+1} = z_n^4 + c, \quad z_0 = 0 \quad \forall c \in \mathbb{C}, |c| \leq 2$ và có hình dạng tam giác, ba trục đối xứng cách nhau 120° (hay 2π/3 radian).

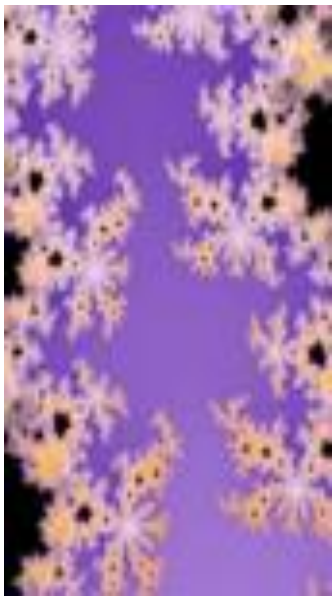
***Tập hợp Mandelbrot bậc năm**



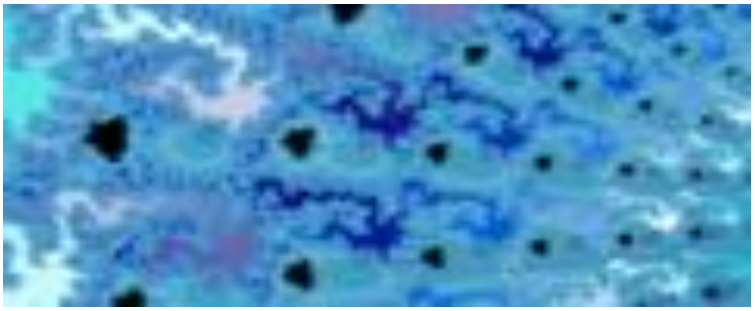
Tập hợp Mandelbrot bậc năm



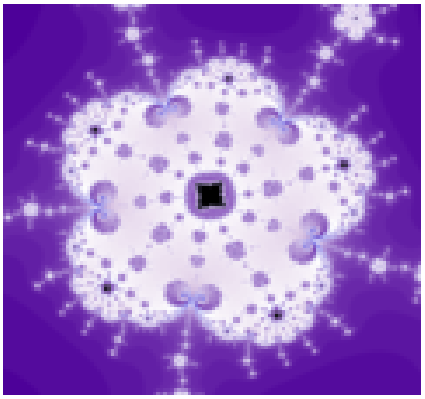
Vi khuẩn



San hô



Mảng rong biển

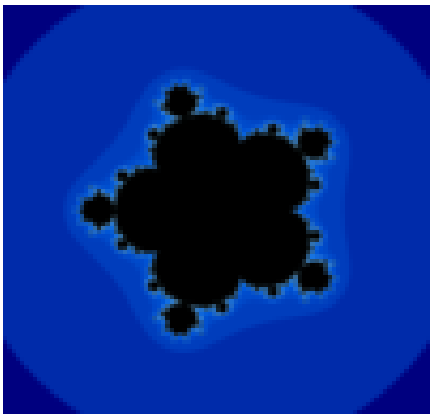


Mây ngũ giác

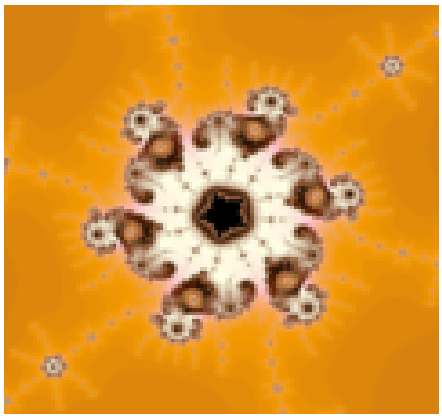
Hình 13. Một số hình ảnh về tập hợp Mandelbrot bậc năm

Tập hợp Mandelbrot bậc năm được tính bằng công thức đệ quy bậc 5 sau: $z_{n+1} = z_n^5 + c, \quad z_0 = 0 \quad \forall c \in \mathbb{C}, |c| \leq 2$ và có dạng hình vuông, bốn trục đối xứng cách nhau 90° (hay π/2 radian).

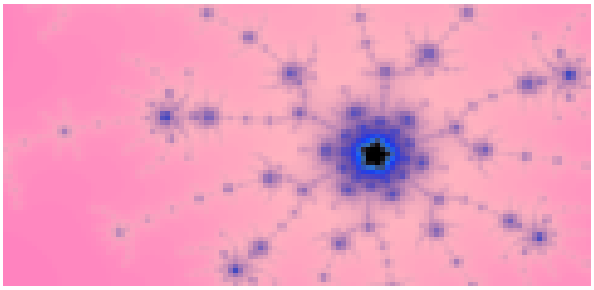
***Tập hợp Mandelbrot bậc sáu**



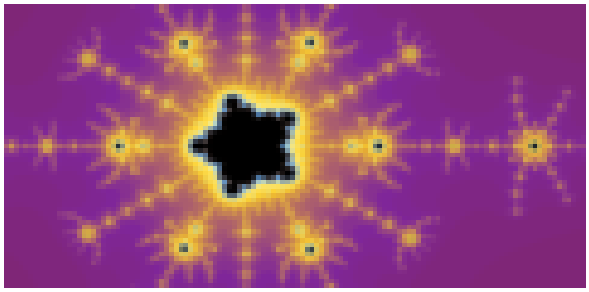
Tập hợp Mandelbrot bậc sáu



Vi khuẩn



Ngôi sao tia Hyperbol

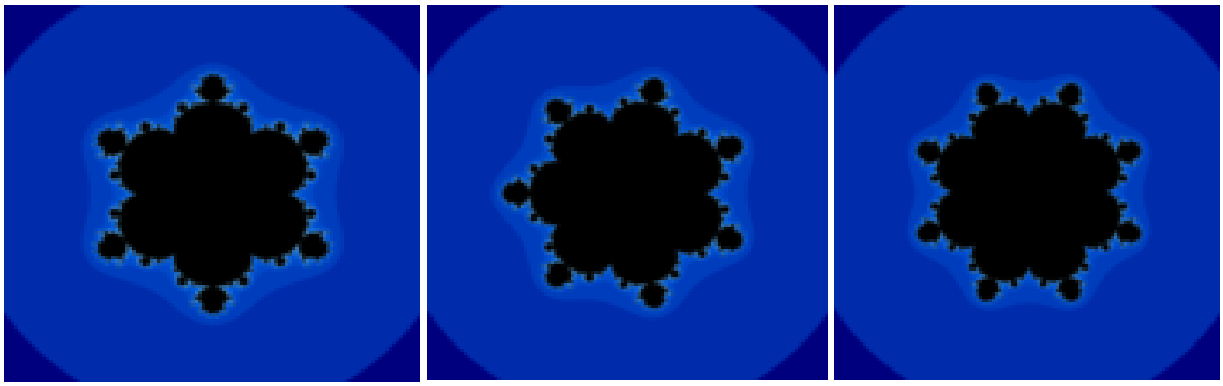


Pháo Bông

Hình 14. Một số hình ảnh về tập hợp Mandelbrot bậc sáu

Tập hợp Mandelbrot bậc sáu được tính bằng công thức đệ quy bậc 6 sau: $z_{n+1} = z_n^6 + c, \quad z_0 = 0 \quad \forall c \in \mathbb{C}, |c| \le 2$ và có hình dạng ngũ giác (giống ngôi sao năm cánh), năm trục đối xứng cách nhau 72° (hay 2π/5 radian).

***Tập hợp Mandelbrot bậc b>3**



Hình 15. Tập hợp Mandelbrot bậc bảy, tám, chín

Tập hợp Mandelbrot bậc b được tính bằng công thức đệ quy bậc b sau: $z_{n+1} = z_n^b + c, \quad z_0 = 0 \quad \forall c \in \mathbb{C}, |c| \le 2$ và b là số nguyên lớn hơn 3, có (b-1) trục đối xứng cách nhau 360°/(b-1) hay π/(b-1) radian và có cấu trúc đa giác tương tự: bậc bốn là tam giác, bậc năm là hình vuông, bậc sáu là ngũ giác,...Tất cả các tập hợp Mandelbrot đều có một trục đối xứng chung là trục 0x.

2.6. Lá cây dương xỉ Barnsley (Barnsley fern)

Lá cây dương xỉ Barnsley [11] là một fractal được đặt tên theo nhà toán học người Anh Michael Barnsley, người lần đầu tiên mô tả nó trong cuốn sách Fractals Everywhere của ông. Điểm đầu tiên được vẽ là tại điểm gốc (x_0=0, y_0=0) và sau đó các điểm mới được tính toán lặp lại bằng cách áp dụng ngẫu nhiên một trong bốn phép biến đổi tọa độ sau:

Hình 15. Lá cây dương xỉ Barnsley

• Tạo cành của lá cây với hệ số xác suất (độ chính xác) p = 0.01

$$f_1(x,y) = \begin{bmatrix} 0.00 & 0.00 \\ 0.00 & 0.16 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \Leftrightarrow \begin{cases} x_{n+1} = 0 \\ y_{n+1} = 0.16 y_n \end{cases}$$

• Tạo ra các lá nhỏ dần liên tiếp với hệ số xác suất p =0.85

$$f_2(x,y) = \begin{bmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.00 \\ 1.60 \end{bmatrix} \Leftrightarrow \begin{cases} x_{n+1} = 0.85 x_n + 0.04 y_n \\ y_{n+1} = -0.04 x_n + 0.85 y_n + 1.6 \end{cases}$$

• Tạo các lá bên tay trái với hệ số xác suất p =0.07

$$f_3(x,y) = \begin{bmatrix} 0.20 & -0.26 \\ 0.23 & 0.22 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.00 \\ 1.60 \end{bmatrix} \Leftrightarrow \begin{cases} x_{n+1} = 0.2 x_n - 0.26 y_n \\ y_{n+1} = 0.23 x_n + 0.22 y_n + 1.6 \end{cases}$$

• Tạo các lá bên tay phải với hệ số xác suất p =0.07

$$f_4(x,y) = \begin{bmatrix} -0.15 & 0.28 \\ 0.26 & 0.24 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.00 \\ 0.44 \end{bmatrix} \Leftrightarrow \begin{cases} x_{n+1} = -0.15 x_n + 0.28 y_n \\ y_{n+1} = 0.26 x_n + 0.24 y_n + 0.44 \end{cases}$$

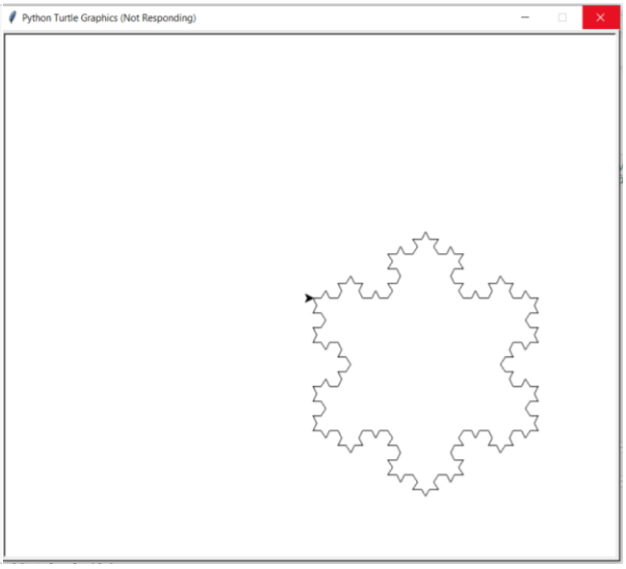
[11] Barnsley fern - Wikipedia

CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH

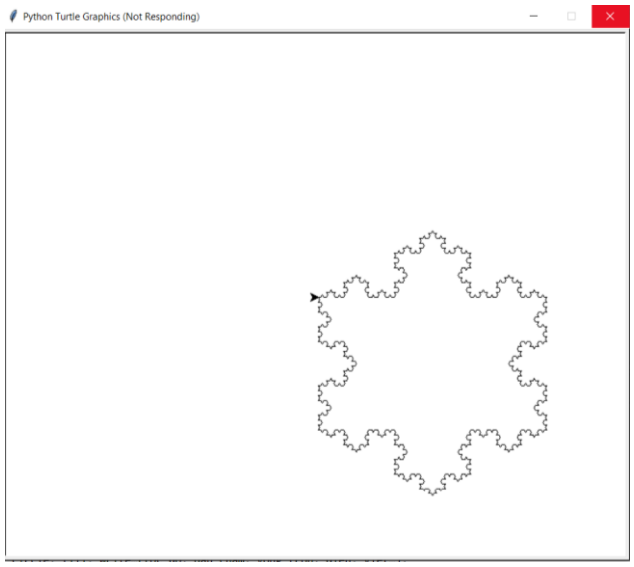
Các chương trình sau đây vẽ một số đường mặt fractal [2] được viết bằng ngôn ngữ python 3.0 (jupyter notebook), sử dụng package/module turtle...để vẽ.

3.1. Bông hoa tuyết Von Kock
*Koch snowflake

```
In [1]: import turtle
def courbe_von_koch (longueur, etape):
    if etape == 0:
        turtle.forward (longueur)
    else:
        courbe_von_koch (longueur/3, etape-1)
        turtle.left (60)
        courbe_von_koch (longueur/3, etape-1)
        turtle.right (120)
        courbe_von_koch (longueur/3, etape-1)
        turtle.left (60)
        courbe_von_koch (longueur/3, etape-1)
def flocon_von_koch (longueur, etape):
    for i in range (3):
        courbe_von_koch (longueur, etape)
        turtle.right (120)

flocon_von_koch (280, 3)
```



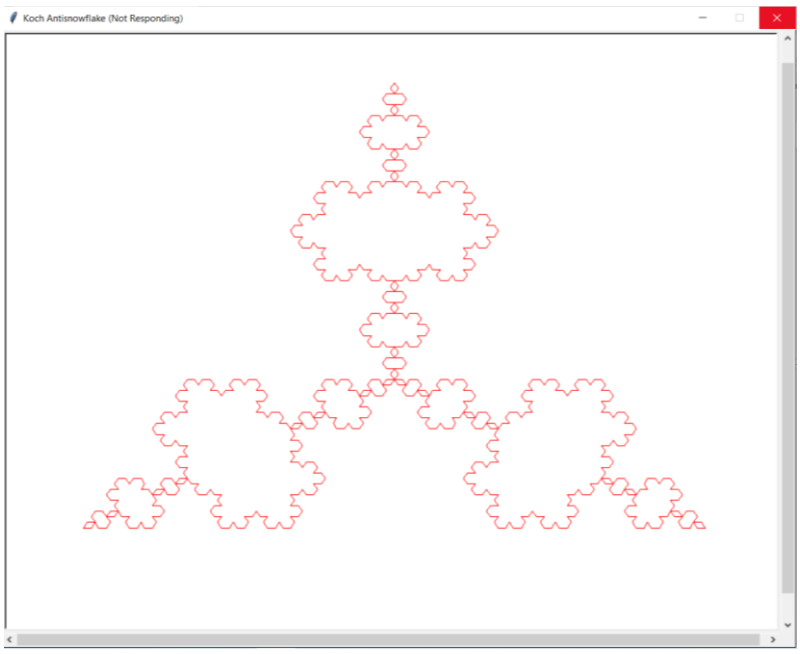
Sau khi tăng thêm 1 lần lặp nữa (với số lần lặp etape=4) ta được hình sau:



*Koch antisnowflake [12]

In [2]:
```
import turtle
import math

screen = turtle.Screen()
screen.title('Koch Antisnowflake')
screen.setup(1000,1000) #kích thước màn hình
screen.setworldcoordinates(-1000,-1000,1000,1000) #đặt tọa độ cho màn hình
turtle.speed(0)  #tốc độ vẽ hình
turtle.hideturtle() #ẩn nét vẽ
turtle.color("red") #màu nét vẽ

def koch_anti(x1,y1,x2,y2):
    dist = ((x2-x1)**2+(y2-y1)**2)**0.5 #tính khoảng giữa 2 điểm
    if dist<50:
        turtle.goto(x2,y2) #di chuyển đến tọa độ
        return
    direction = math.atan2(y2-y1,x2-x1) #tính actan(y/x), giá trị radian
    px1, py1 = x1+dist/3*math.cos(direction), y1+dist/3*math.sin(direction)
    px2, py2 = px1+dist/3*math.cos(direction-math.radians(60)), py1+dist/3*math.sin(direction-math.radians(60))
    px3, py3 = x1+2*dist/3*math.cos(direction), y1+2*dist/3*math.sin(direction)

    koch_anti(x1,y1,px1,py1)
    koch_anti(px1,py1,px2,py2)
    koch_anti(px2,py2,px3,py3)
    koch_anti(px3,py3,x2,y2)

turtle.up() #dừng vẽ
turtle.goto(800,-600) #di chuyển đến tọa độ
turtle.down() #tiếp tục vẽ
koch_anti(800,-600,-800,-600)
koch_anti(-800,-600,0,-600+1600*3**0.5/2)
koch_anti(0,-600+1600*3**0.5/2,800,-600)
```

Chạy đoạn mã trên cho chúng ta kết quả sau:

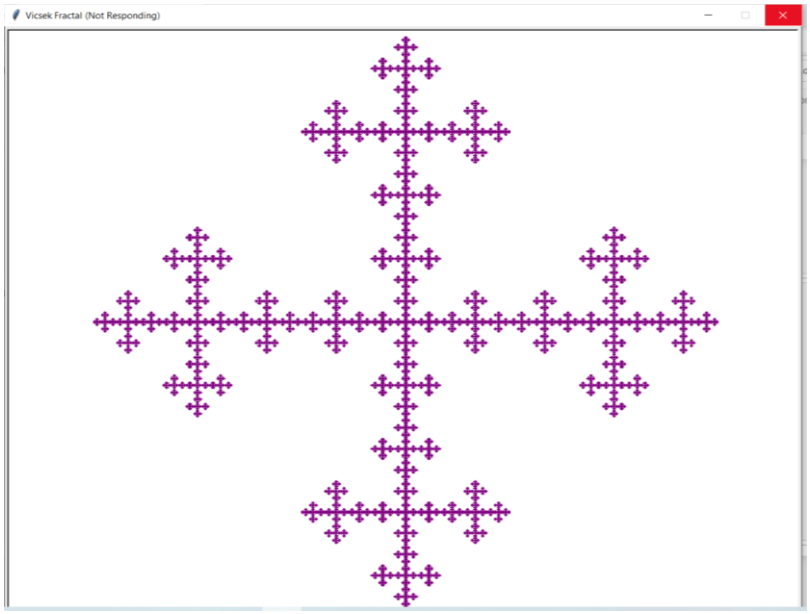


[12] Koch Anti-Snowflake (agnesscott.org)

*Vicsek snowflake [13]

In [1]:
```
import turtle
screen = turtle.Screen()
screen.title('Vicsek Fractal')
screen.setup(1000,800) #kích thước màn hình
screen.setworldcoordinates(-1000,-800,1000,800) #đặt tọa độ cho màn hình
screen.tracer(0,0)
turtle.speed(0) #tốc độ vẽ hình
turtle.hideturtle() #ẩn nét vẽ
turtle.color('purple') #màu nét vẽ

def draw_cross(x,y,length):
    turtle.up() #dừng vẽ
    turtle.goto(x-length/2,y-length/6) #đến vị trí có tọa độ
    turtle.down()   #tiếp tục vẽ
    turtle.seth(0)  #tạo vòng lặp
    turtle.begin_fill()
    for _ in range(4):
        turtle.fd(length/3)
        turtle.right(90)
        turtle.fd(length/3)
        turtle.left(90)
        turtle.fd(length/3)
        turtle.left(90)
    turtle.end_fill()

def vicsek(x,y,length,n):
    if n==0:
        draw_cross(x,y,length)
        return
    vicsek(x,y,length/3,n-1)
    vicsek(x+length/3,y,length/3,n-1)
    vicsek(x-length/3,y,length/3,n-1)
    vicsek(x,y+length/3,length/3,n-1)
    vicsek(x,y-length/3,length/3,n-1)

vicsek(0,0,1600,5)
screen.update()
```

Chạy đoạn mã trên cho chúng ta kết quả sau:

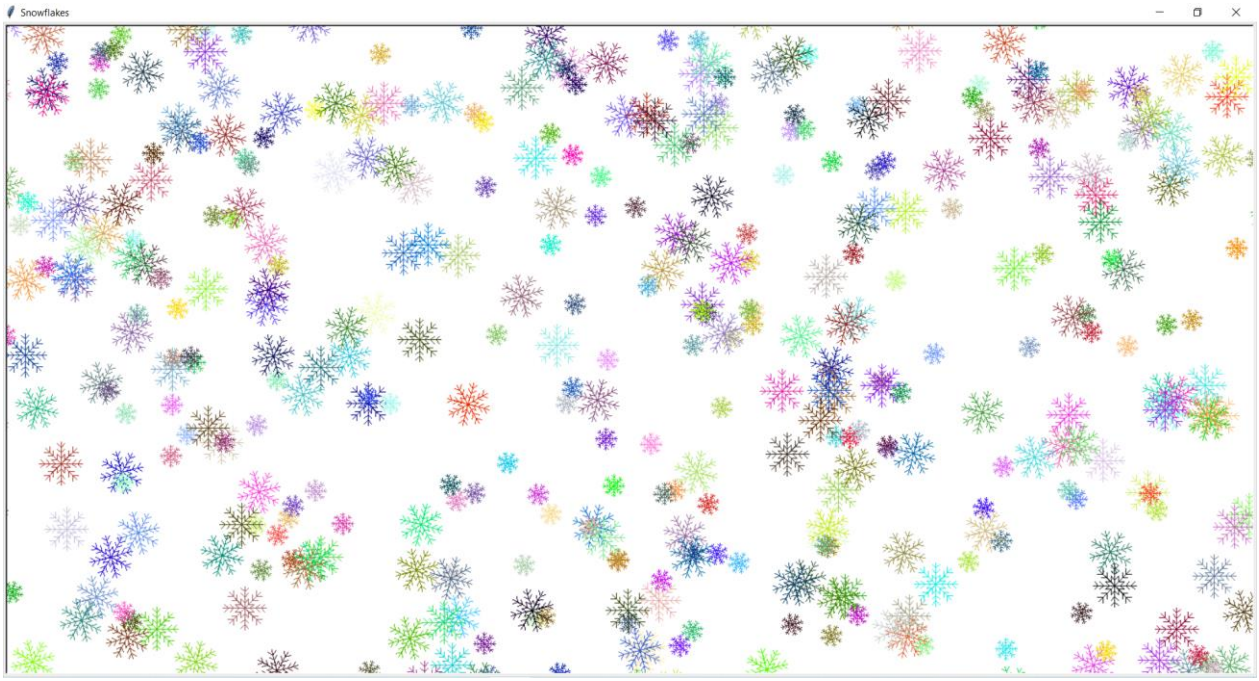


[13] Vicsek fractal - Wikipedia

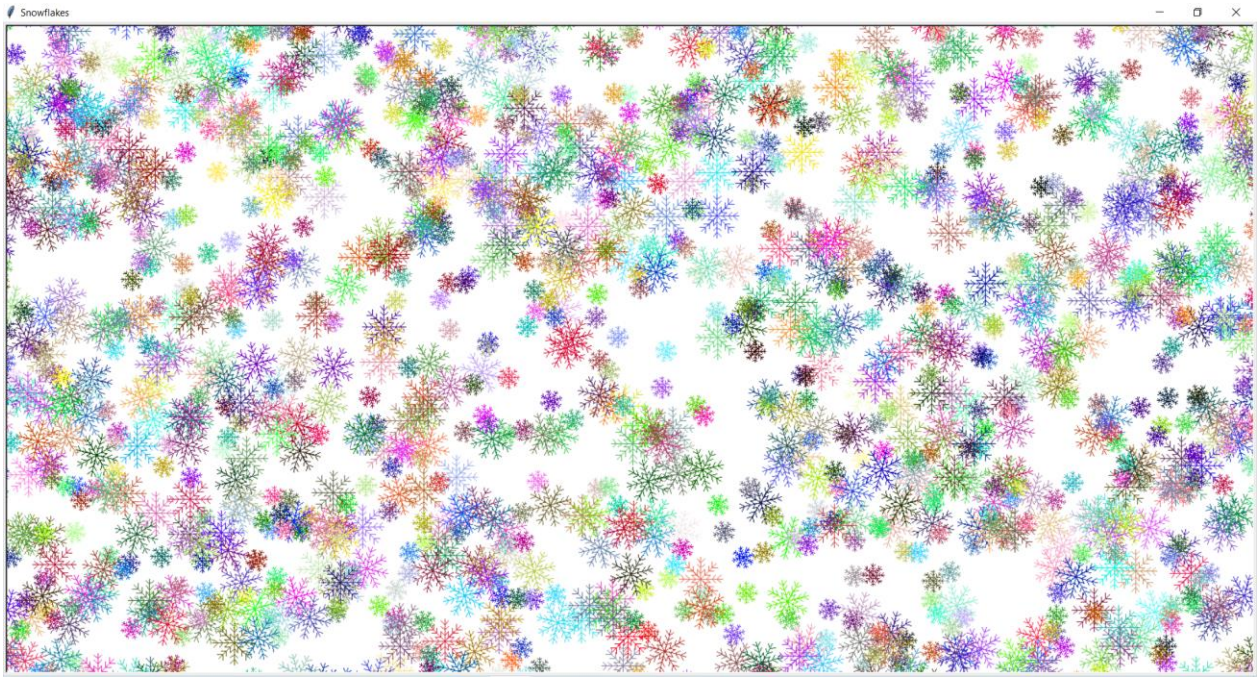
In [15]:
```
import turtle
from random import randint

def generate_random_colour():
    #Tạo các giá trị R, G, B ngẫu nhiên trong phạm vi 0 đến 255
    r = randint(0, 255)
    g = randint(0, 255)
    b = randint(0, 255)
    return r, g, b

def setup_screen(title, background = 'white'):
    print('Set up Screen')
    turtle.title(title)
    turtle.setup(640, 600)
    turtle.hideturtle()
    turtle.penup()
    turtle.tracer(200)
    #Đặt màu nền của màn hình
    turtle.bgcolor(background)

def draw_snowflake(size):
    #Vẽ hình bông tuyết
    turtle.penup()
    turtle.forward(10 * size)
    turtle.left(45)
    turtle.pendown()
    turtle.color(generate_random_colour())

    #Vẽ nhánh 8 lần để tạo thành bông tuyết
    for _ in range(8):
        draw_branch(size)
        turtle.forward(size)
        turtle.left(45)

    turtle.penup()

def draw_branch(size):
    #Vẽ một nhánh riêng lẻ trên một bông tuyết
    side_branch_size = size / 3
    for _ in range(3):
        for i in range(3):
            turtle.forward(side_branch_size)
            turtle.backward(side_branch_size)
            turtle.right(45)
        turtle.left(90)
        turtle.backward(side_branch_size)
        turtle.left(45)
    turtle.right(90)

setup_screen('Snowflakes')
#Cho biết các số RGB sẽ nằm trong phạm vi từ 0 đến 255
turtle.colormode(255)

print("Drawing snowflakes at random locations")
for _ in range(3000):
    x = randint(-1000, 1000)
    y = randint(-800, 800)
    snowflake_size = randint(1, 2) * 10
    turtle.goto(x, y)
    turtle.right(15)
    draw_snowflake(snowflake_size)

turtle.update()
print('Done')
turtle.done()
```

Chạy đoạn mã trên cho chúng ta kết quả sau:

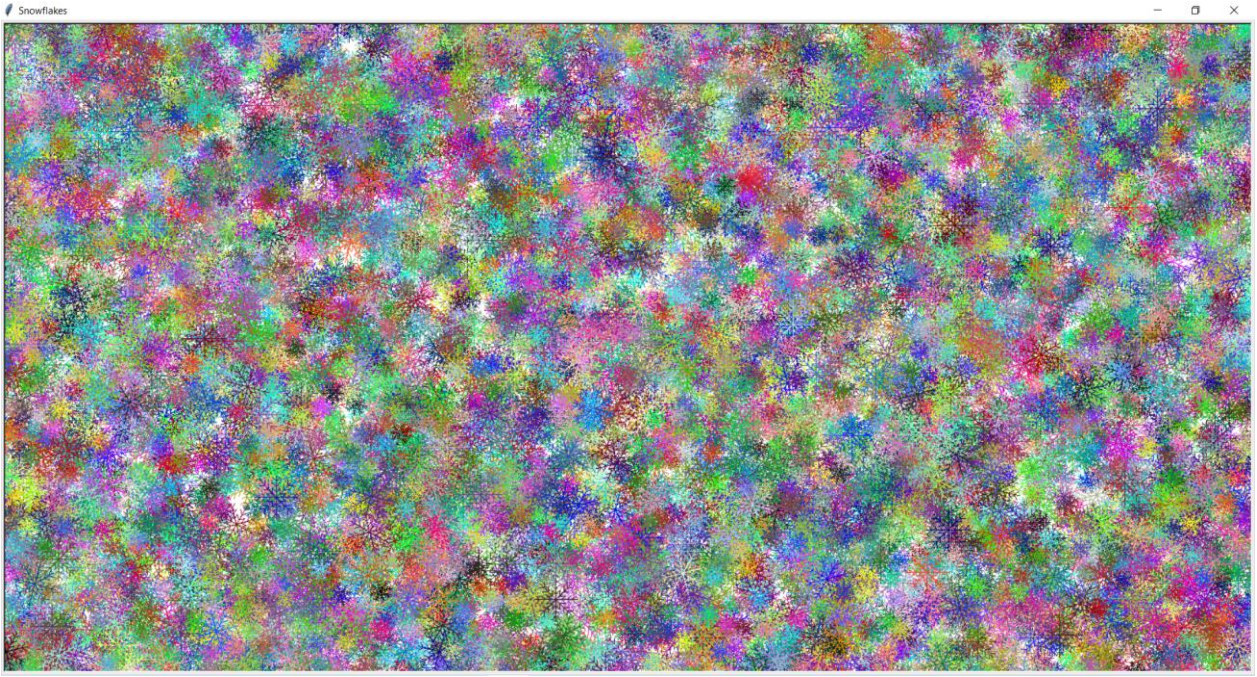
với 1000 bông tuyết:



với 3000 bông tuyết:



Mất khoảng **10 giây, 40 giây** để vẽ lần lượt **1 nghìn, 3 nghìn bông tuyết** bất kỳ để thu được hình ảnh tương ứng như trên. Khi tăng số bông tuyết lên tới **1 triệu bông** thì sau khoảng **2 giờ** thu được hình ảnh như sau (vẫn còn đang vẽ):

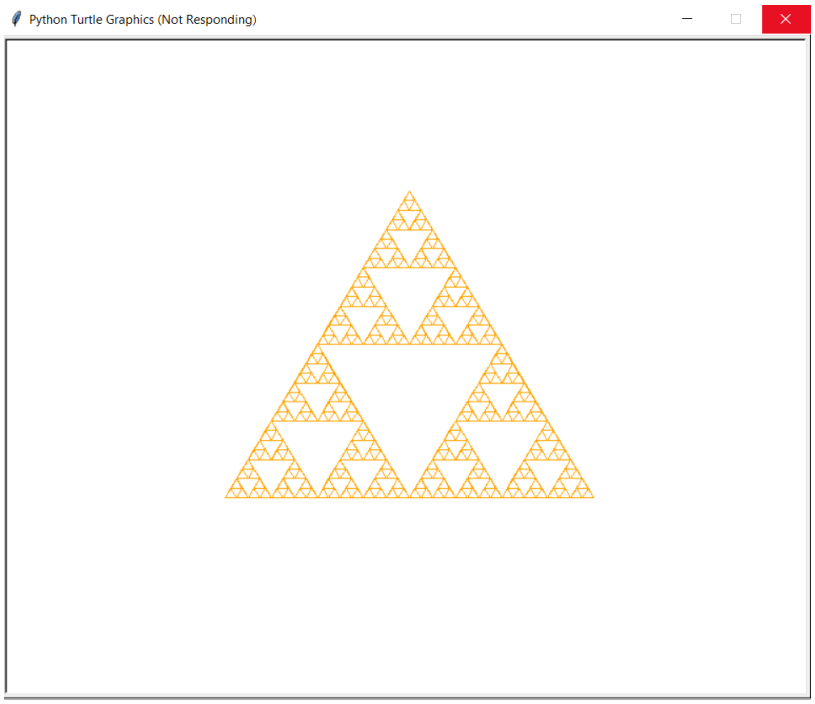


3.2. Họ đường Sierpinski

*Tam giác Sierpinski

In [3]:
```
import turtle
PROGNAME = 'Tam giác Sierpinski'
myPen = turtle.Turtle()
myPen.ht()
myPen.speed(5)
myPen.pencolor('orange')

points = [[-175,-125],[0,175],[175,-125]] #Kích thước của hình tam giác cho ban đầu

def getMid(p1,p2):
    return ( (p1[0]+p2[0]) / 2, (p1[1] + p2[1]) / 2) #Tìm điểm giữa (trung điểm)

def triangle(points,depth):

    myPen.up()
    myPen.goto(points[0][0],points[0][1])
    myPen.down()
    myPen.goto(points[1][0],points[1][1])
    myPen.goto(points[2][0],points[2][1])
    myPen.goto(points[0][0],points[0][1])

    if depth>0:
        triangle([points[0], getMid(points[0], points[1]), getMid(points[0], points[2])], depth-1)

        triangle([points[1], getMid(points[0], points[1]), getMid(points[1], points[2])], depth-1)

        triangle([points[2], getMid(points[2], points[1]), getMid(points[0], points[2])], depth-1)


triangle(points,5)
```

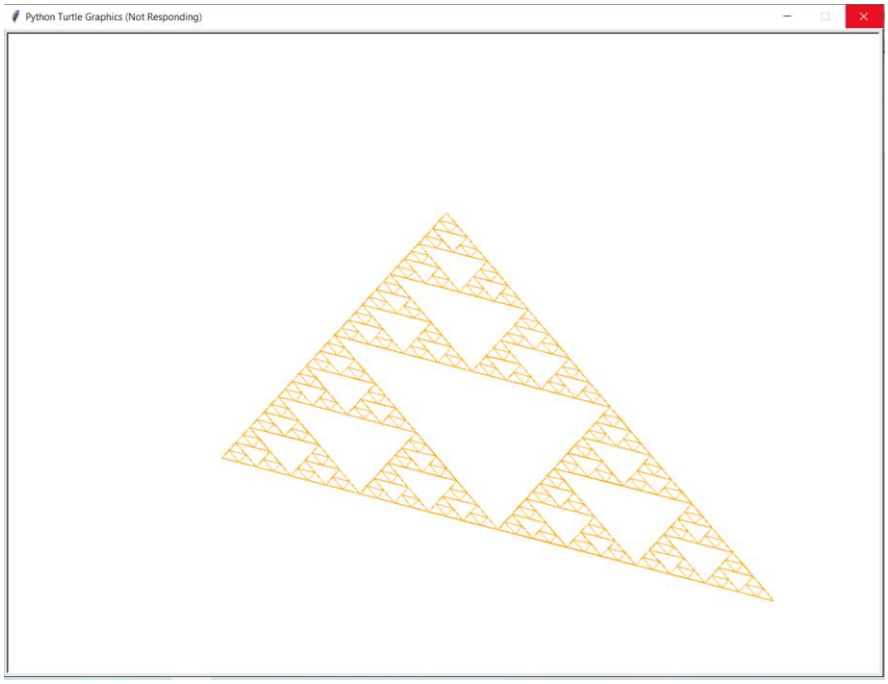
Chạy đoạn mã trên cho chúng ta kết quả sau (với 5 lần lặp):



Cũng với đoạn code trên khi ta thay đổi tọa độ 3 đỉnh của tam giác ban đầu như sau:

```
points = [[-275,-125],[0,175],[400,-300]]
```

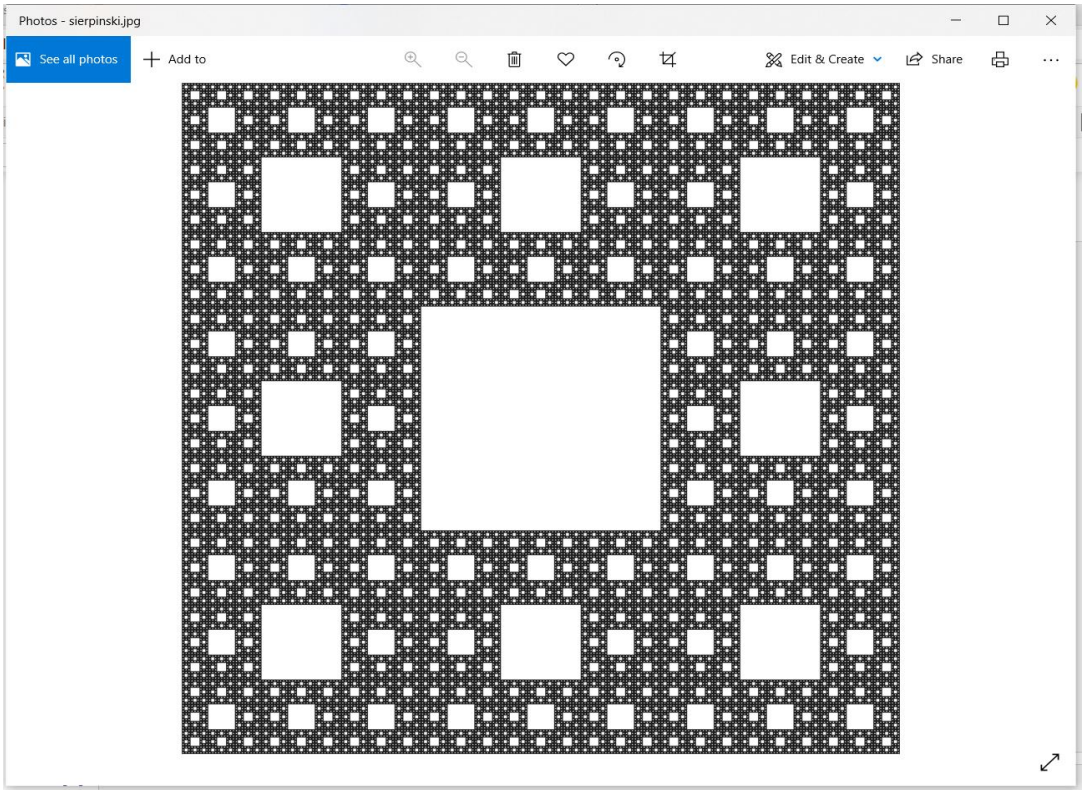
và kết quả chạy chương trình cho ta hình vẽ sau:



***Tấm thảm Sierpinski**

In [6]:
```
import numpy as np
from PIL import Image

total = 7 #Số lần quy trình sẽ được lặp lại
size = 3**total #Kích thước của hình ảnh

# tạo hình ảnh
square = np.empty([size, size, 3], dtype = np.uint8)
color = np.array([255, 255, 255], dtype = np.uint8)

# tô màu đen
square.fill(0)

for i in range(0, total + 1):
    stepdown = 3**(total - i)
    for x in range(0, 3**i):
        # kiểm tra ô vuông ở giữa
        if x % 3 == 1:
            for y in range(0, 3**i):
                if y % 3 == 1:
                    # thay đổi màu của nó
                    square[y * stepdown:(y + 1)*stepdown, x * stepdown:(x + 1)*stepdown] = color

# lưu hình ảnh đã tạo
save_file = "sierpinski.jpg"
Image.fromarray(square).save(save_file)

# hiển thị nó trong bảng điều khiển
i = Image.open("sierpinski.jpg")
i.show()
```

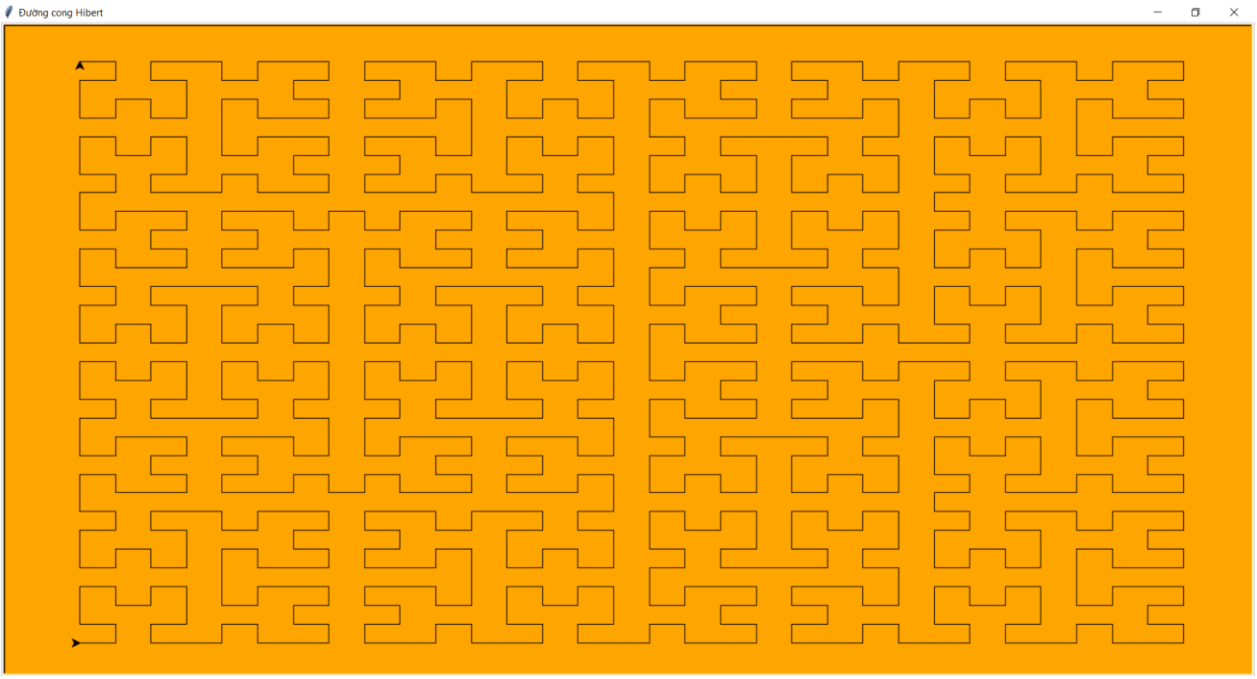
Chạy đoạn mã trên cho chúng ta kết quả sau (với 7 lần lặp):



3.3. Đường cong Hibert

In [5]:
```
import turtle as tt
import inspect
import pylab as pl

def Hibert(iterations=1):
    ivan = tt.Turtle(shape = "classic", visible = True)
    screen = tt.Screen()
    screen.title("Đường cong Hibert") # tên hình
    screen.bgcolor("Orange") # màu nền
    screen.delay(0) # Tốc độ vẽ
    screen.setup(width=0.95, height=0.9) #chiều rộng, chiều dài khuôn khổ vẽ
    walk = 1 # kích thước mỗi đoạn thẳng
    def screenlength(k):
        if k != 0:
            length = screenlength(k-1)
            return 2*length + 1
        else:
            return 0
    kj = screenlength(iterations)
    screen.setworldcoordinates(-1, -1, kj + 1, kj + 1)
    ivan.color("Black")
    def step1(k):
        if k != 0:
            ivan.left(90)
            step2(k - 1)
            ivan.forward(walk)
            ivan.right(90)
            step1(k - 1)
            ivan.forward(walk)
            step1(k - 1)
            ivan.right(90)
            ivan.forward(walk)
            step2(k - 1)
            ivan.left(90)
    def step2(k):
        if k != 0:
            ivan.right(90)
            step1(k - 1)
            ivan.forward(walk)
            ivan.left(90)
            step2(k - 1)
            ivan.forward(walk)
            step2(k - 1)
            ivan.left(90)
            ivan.forward(walk)
            step1(k - 1)
            ivan.right(90)
    ivan.left(90)
    step2(iterations)
    tt.done() # hoàn thành việc vẽ

Hibert(5)
pl.show()
```

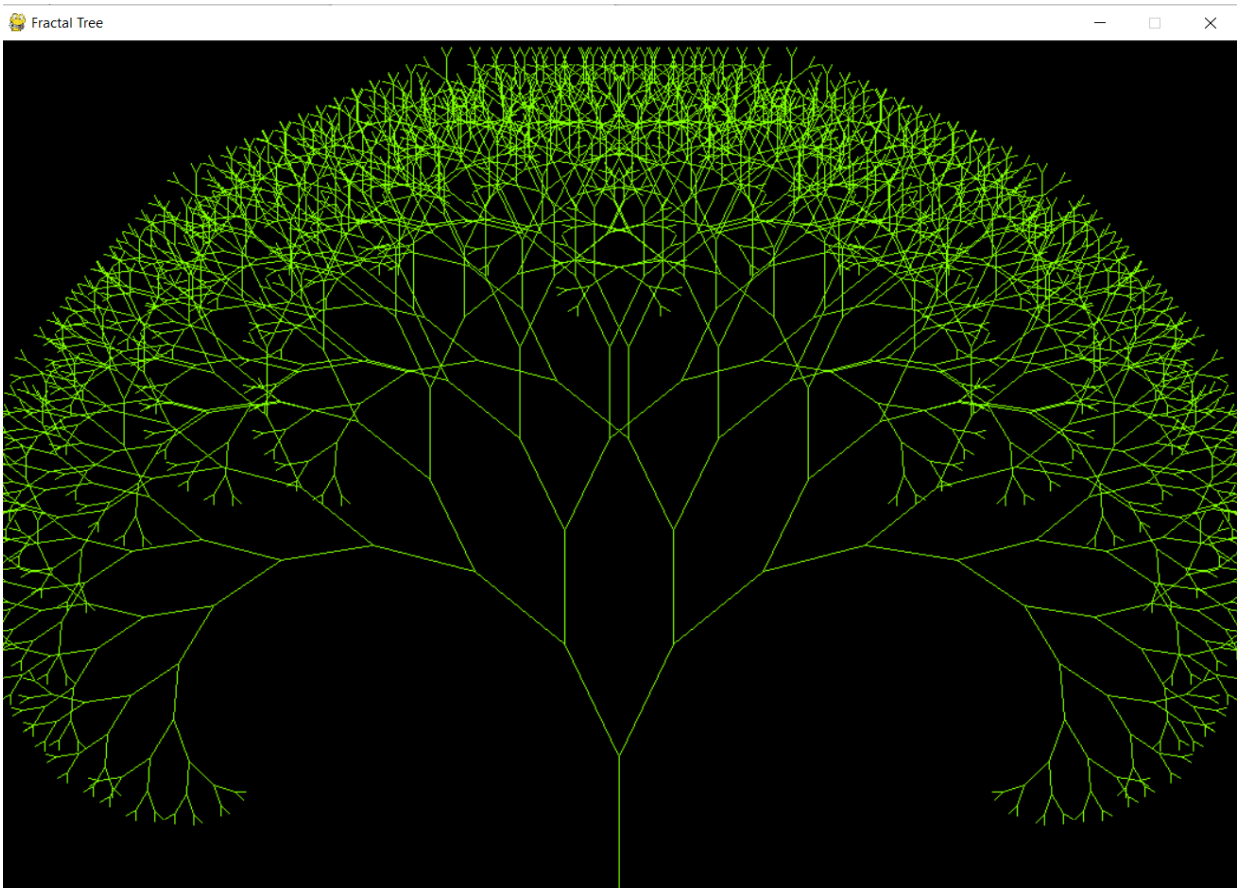
Chạy đoạn mã trên cho chúng ta kết quả sau:



3.4. Cây fractal

In [ ]:
```
import pygame, math

pygame.init()
screen = pygame.display.set_mode((1050, 750)) #kích thước bố cục hình
pygame.display.set_caption("Fractal Tree")
display = pygame.display.get_surface()

def drawTree(a, b, pos, deepness):
    if deepness:
        c = a + int(math.cos(math.radians(pos)) * deepness * 10)
        d = b + int(math.sin(math.radians(pos)) * deepness * 10)
        pygame.draw.line(display, (127,255,0), (a, b), (c, d), 1)
        drawTree(c, d, pos - 25, deepness - 1)
        drawTree(c, d, pos + 25, deepness- 1)

def process(event):
    if event.type == pygame.QUIT:
        exit(0)

drawTree(525, 750, -90, 12)
pygame.display.flip()
while True:
    process(pygame.event.wait())
```

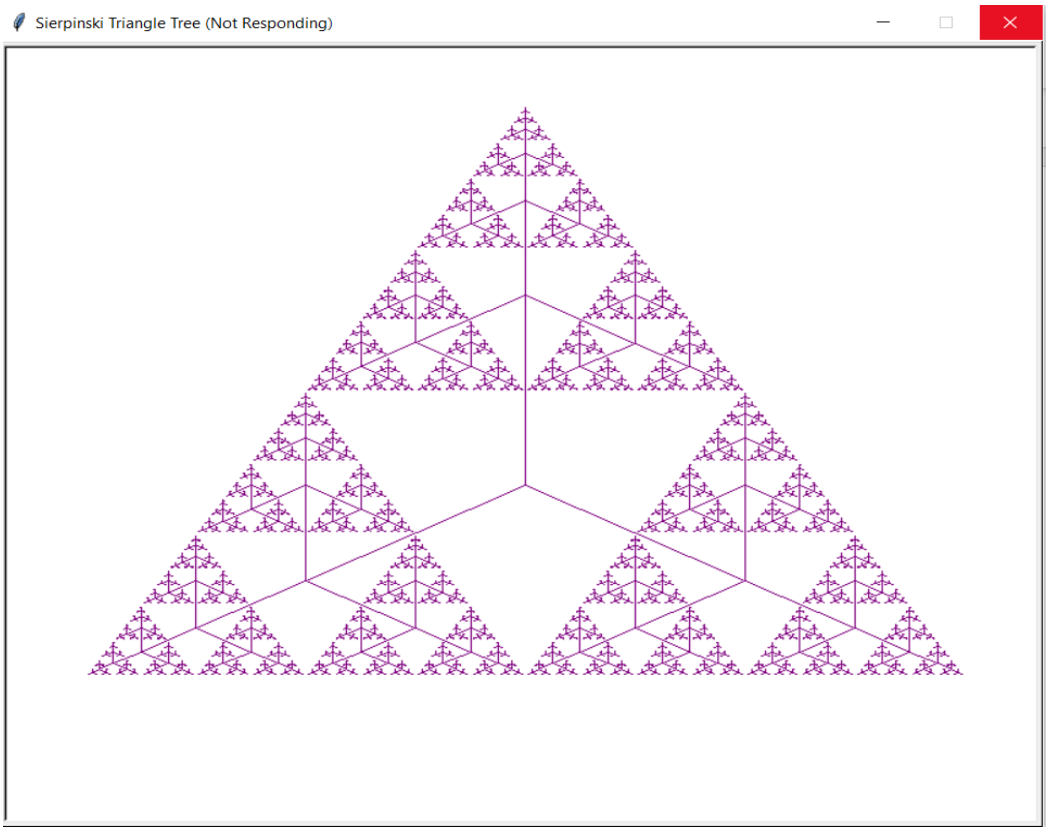
Chạy đoạn mã trên cho chúng ta kết quả sau (với 11 lần phân tách):



3.5. Cây tam giác Sierpinski (Sierpinski Triangle Tree)

In [4]:
```
import turtle
turtle.title('Sierpinski Triangle Tree')
turtle.setworldcoordinates(-2000,-2000,2000,2000)
screen = turtle.Screen()
screen.tracer(0,0)
turtle.hideturtle()
turtle.pencolor('purple')

def sierpinski_tree(x,y,length,tilt,n):
    if n==0:
        return
    turtle.up()     #dừng vẽ
    turtle.goto(x,y)     #đưa đến vị trí có tọa độ (x,y)
    turtle.seth(tilt)    #độ (góc) nghiêng
    turtle.down()   #tiếp tục vẽ
    turtle.fd(length)   #chiều dài
    sierpinski_tree(turtle.xcor(),turtle.ycor(),length/2,turtle.heading(),n-1)

    turtle.up()
    turtle.goto(x,y)
    turtle.seth(tilt+120)
    turtle.down()
    turtle.fd(length)
    sierpinski_tree(turtle.xcor(),turtle.ycor(),length/2,turtle.heading(),n-1)

    turtle.up()
    turtle.goto(x,y)
    turtle.seth(tilt-120)
    turtle.down()
    turtle.fd(length)
    sierpinski_tree(turtle.xcor(),turtle.ycor(),length/2,turtle.heading(),n-1)

sierpinski_tree(0,-250,1000,90,8)
screen.update()
```

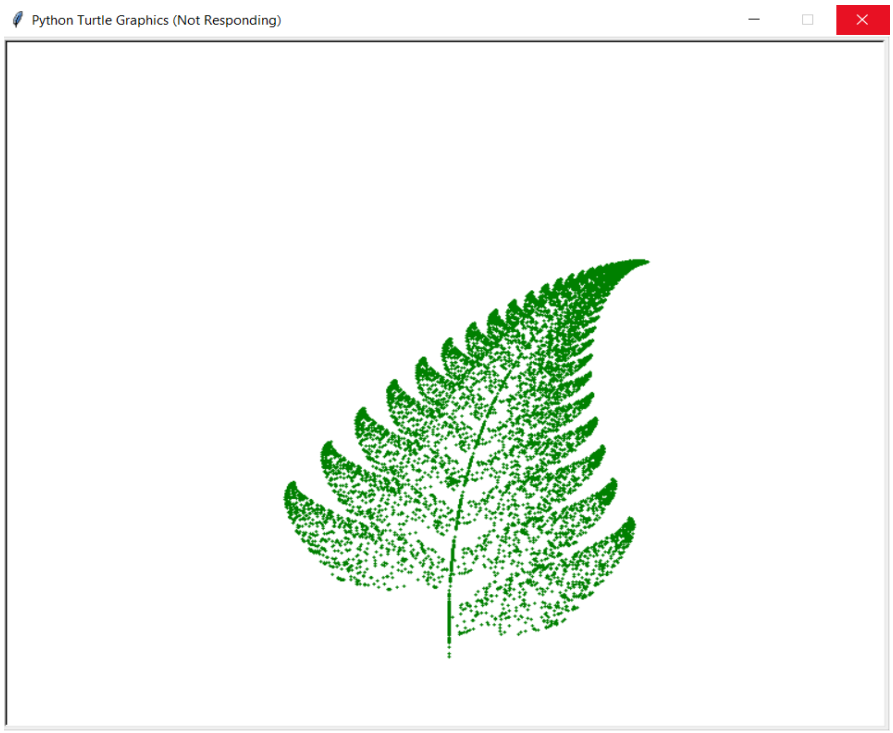
Chạy đoạn mã trên cho chúng ta kết quả sau (với 8 lần lặp):



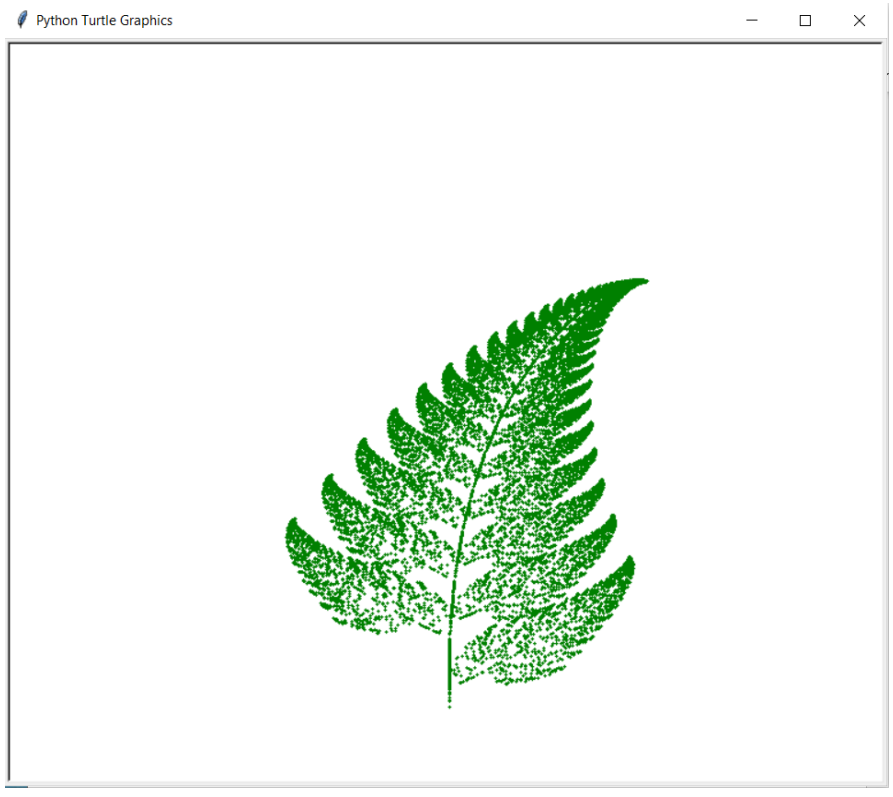
3.6. Lá cây đươn xỉ Barnsley

In [14]:
```
import turtle
import random

pen = turtle.Turtle()
pen.speed(0)
pen.color("green")
pen.penup()

x = 0
y = 0
for n in range(11000):
    pen.goto(65 * x, 37 * y - 252) #vị trí tọa độ
    pen.pendown()
    pen.dot(3)                     #chấm các điểm
    pen.penup()
    r = random.random()
    if r < 0.01:
        x, y =  0.00 * x + 0.00 * y,  0.00 * x + 0.16 * y + 0.00
    elif r < 0.86:
        x, y =  0.85 * x + 0.04 * y, -0.04 * x + 0.85 * y + 1.60
    elif r < 0.93:
        x, y =  0.20 * x - 0.26 * y,  0.23 * x + 0.22 * y + 1.60
    else:
        x, y = -0.15 * x + 0.28 * y,  0.26 * x + 0.24 * y + 0.44
```

Chạy đoạn mã trên cho chúng ta kết quả sau:



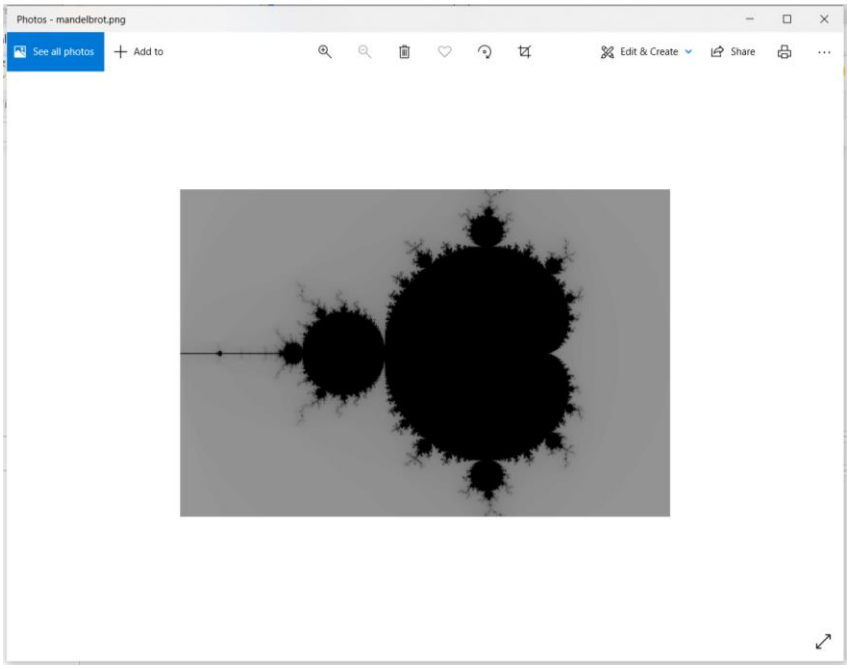
Khi chạy với n = **11000 chấm**, em đã mất khoảng **gần 1 giờ** để có được hình ảnh như trên. Sau khi tăng n = **100000000 = 10^8 chấm**, em đã chờ đến khoảng **2 giờ** để có được hình ảnh dưới đây (với tốc độ vẽ là speed(1000000 = 10^6) thì hình vẽ vẫn chưa vẽ xong):



3.7. Tập hợp Mandelbrot

In [29]:
```
from PIL import Image, ImageDraw
MAX_ITER = 80

def mandelbrot(c):
    z = 0
    n = 0
    while abs(z) <= 2 and n < MAX_ITER:
        z = z*z + c
        n += 1
    return n

#Kích thước khung hình ảnh (pixel)
WIDTH = 600     #chiều rộng
HEIGHT = 400    #chiều cao

#Hình vẽ
RE_START = -2    #giá trị phần thực bắt đầu
RE_END = 1       #giá trị phần thực kết thúc
IM_START = -1    #giá trị phần ảo bắt đầu
IM_END = 1       #giá trị phần ảo kết thúc

im = Image.new('RGB', (WIDTH, HEIGHT), (0, 0, 0))
draw = ImageDraw.Draw(im)

for x in range(0, WIDTH):
    for y in range(0, HEIGHT):
        #Chuyển đổi tọa độ pixel thành số phức
        c = complex(RE_START + (x / WIDTH) * (RE_END - RE_START),
                    IM_START + (y / HEIGHT) * (IM_END - IM_START))
        #Tính số lần lặp lại
        m = mandelbrot(c)
        #Màu sắc phụ thuộc vào số lần lặp lại
        color = 150 - int(m * 150 / MAX_ITER)
        #Phác thảo điểm
        draw.point([x, y], (color, color, color))

im.save('mandelbrot.png', 'PNG')
```

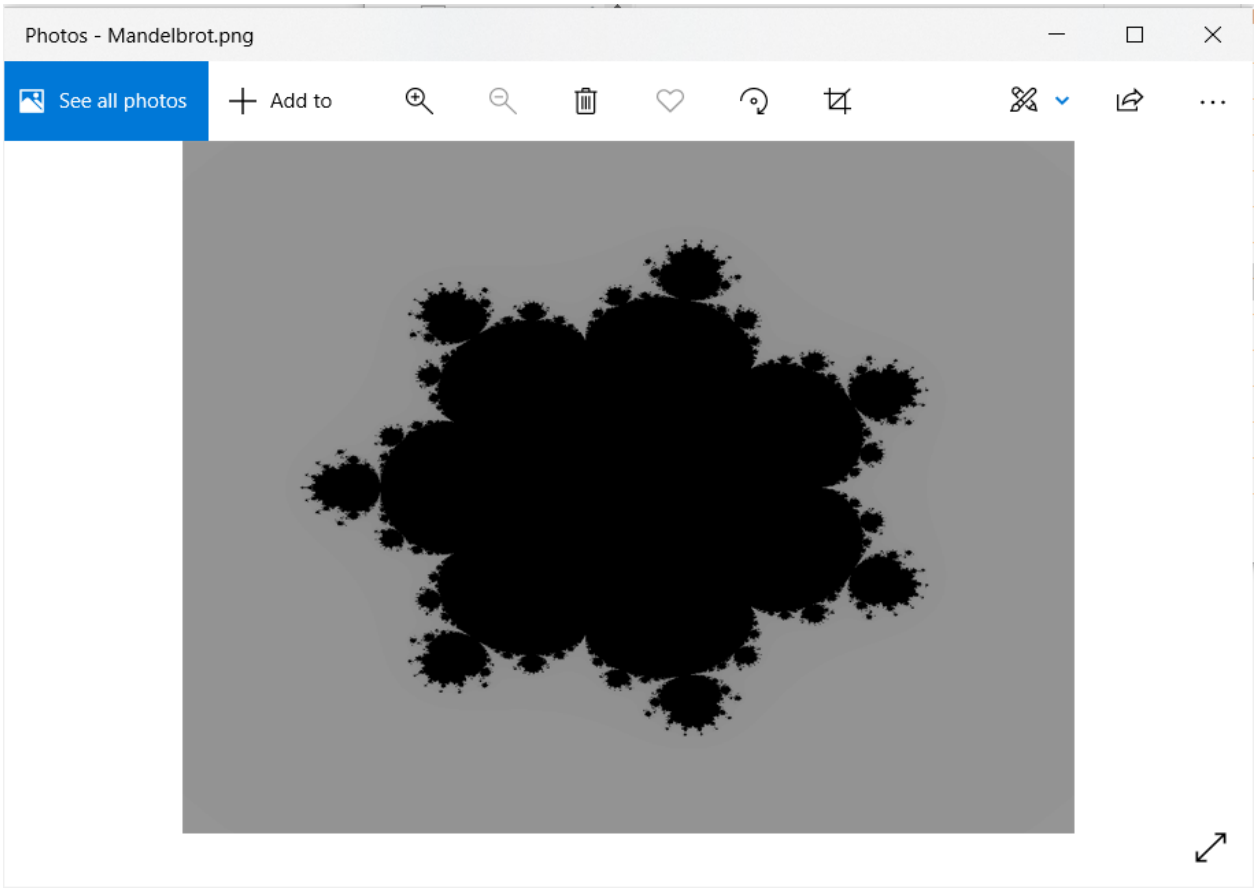
Chạy đoạn mã trên cho hình ảnh về tập hợp Mandelbrot bậc hai:



Tiếp tục với đoạn code trên, khi ta thay đổi bậc của tập hợp Mandelbrot và điều chỉnh tọa độ hình vẽ như sau:

```
def mandelbrot(c):
    z = 0
    n = 0
    while abs(z) <= 2 and n < MAX_ITER:
        z = z**8 + c
        n += 1
    return n

#Kích thước khung hình ảnh (pixel)
WIDTH = 1000     #chiều rộng
HEIGHT = 800    #chiều cao

#Hình vẽ
RE_START = -1.5    #giá trị phần thực bắt đầu
RE_END = 1.5       #giá trị phần thực kết thúc
IM_START = -1.5    #giá trị phần ảo bắt đầu
IM_END = 1.5       #giá trị phần ảo kết thúc
```

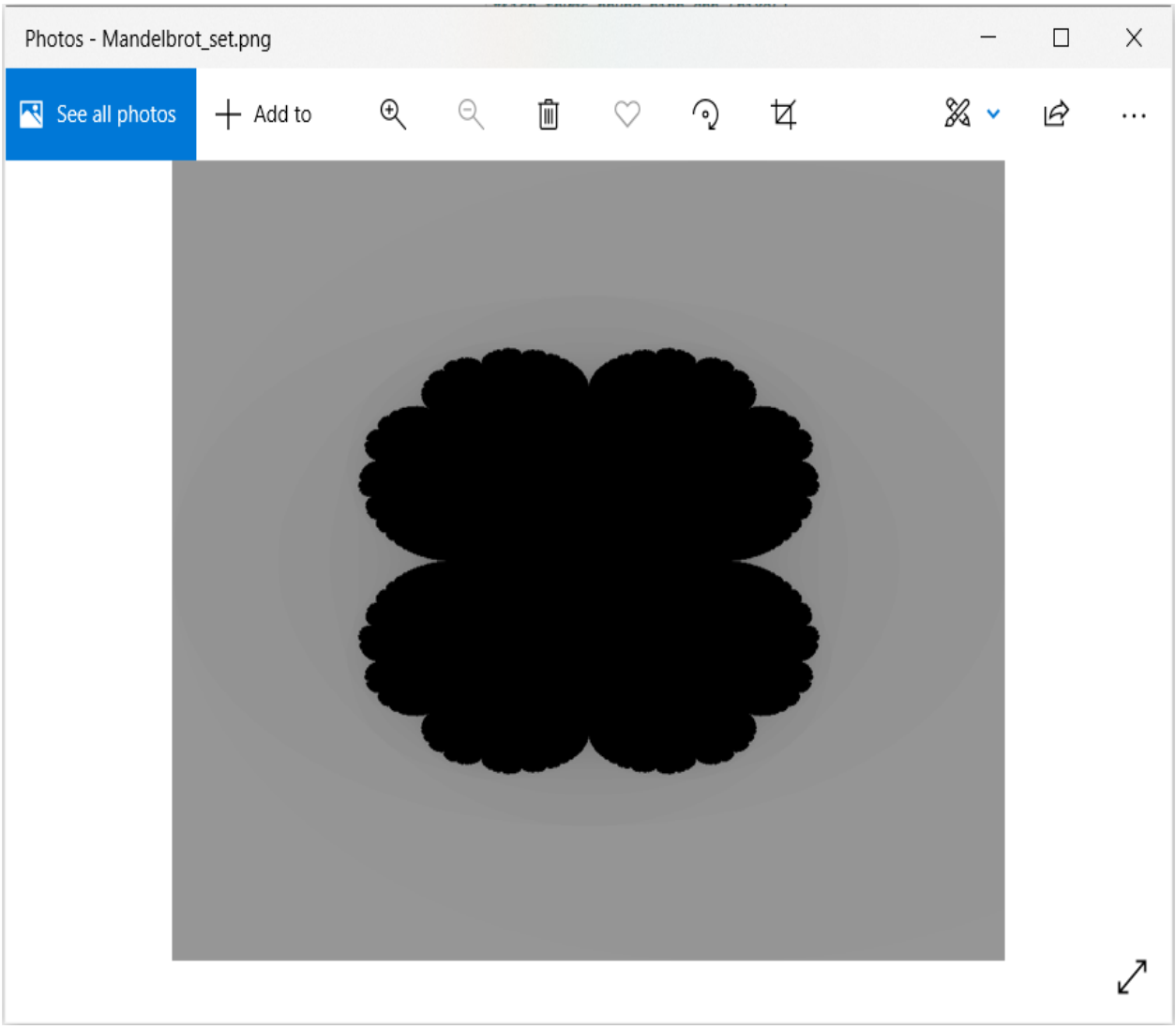
sau khi chạy ta thu được hình ảnh về tập hợp Mandelbrot bậc 8 như sau:



Vẫn tiếp tục với đoạn code ban đầu, khi ta thay đổi công thức lặp của tập hợp Mandelbrot và điều chỉnh tọa độ hình vẽ như sau:

```python
def mandelbrot(c):
    n = 0
    while abs(c) <= 2 and n < MAX_ITER:
        c = c**2 + c
        n += 1
    return n

#Kích thước khung hình ảnh (pixel)
WIDTH = 1000     #chiều rộng
HEIGHT = 800    #chiều cao

#Hình vẽ
RE_START = -2    #giá trị phần thực bắt đầu
RE_END = 1       #giá trị phần thực kết thúc
IM_START = -2    #giá trị phần ảo bắt đầu
IM_END = 2      #giá trị phần ảo kết thúc
```

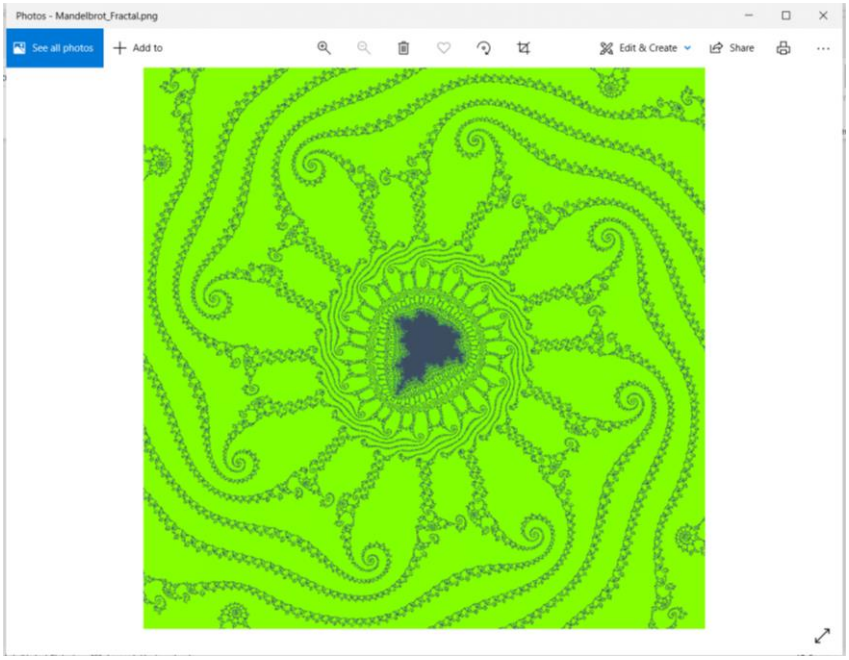
sau khi chạy ta thu được hình ảnh như sau:



In [26]:
```
from PIL import Image
from numpy import complex, array
import colorsys
import math
import datetime

H = 1024
xmin=-0.7746806106269039-0.000000000001506/2.0
xmax=-0.7746806106269039+0.000000000001506/2.0
ymin=-0.1374168856037867-0.000000000001506/2.0
ymax=-0.1374168856037867+0.000000000001506/2.0

maxiter=2000
k=math.exp(-1.0/100)
kp=1.0-k
R=1000000000.0**2
mandel_color = tuple([61,77,97])
mandel_color_hsv = colorsys.rgb_to_hsv(mandel_color[0]/255.0,
                                       mandel_color[1]/255.0,mandel_color[2]/255.0)
x0=0.0
y0=0.0
# biến cho đạo hàm được sử dụng cho công cụ ước tính khoảng cách
derx=1.0
dery=0.0
derx2=0.0
dery2=0.0
derx0=0.0
dery0=0.0
# biến cho phái sinh được sử dụng để phát hiện bên trong
derintx=0.0
derinty=0.0
derintx2=0.0
derinty2=0.0
derintx0=0.0
derinty0=0.0

zx=0.0
zy=0.0
zx2=0.0
zy2=0.0
r2=0.0
rd2=0.0
d2=0.0
t=0.0
reason=False
thick=0.02
eps = 0.01
n0=0
n1=0
n2=0

def rgb_conv(i,t):
    color = 255 * array(colorsys.hsv_to_rgb((1-k**i)/4*t+mandel_color_hsv[0]*(1-t),
                                            t+mandel_color_hsv[1]*(1-t), t+mandel_color_hsv[2]*(1-t)))
    return tuple(color.astype(int))

# tạo hình ảnh mới ở chế độ RGB
DX=xmax-xmin
DY=ymax-ymin
HY = int(H*DY/DX)
DX2=DX/H
DY2=DY/HY

seuil = 2*(thick*DX2)**2
img = Image.new('RGB', (H,HY) )
pixels = img.load()
```

```
for x in range(H):
    if x % 64==0:      # hiển thị tiến trình
        print(str(x) + ' - ' + str(datetime.datetime.now())+' - n0='+str(n0))
        n1+=n0
        n0=0
    for y in range(HY):
        x0=xmin+x*DX2
        y0=ymin+y*DY2
        zx=x0
        zy=y0
        zx2=zx**2
        zy2=zy**2
        derx=1.0
        dery=0.0
        derintx=1.0
        derinty=0.0
        reason=False
        for i in range(1, maxiter):
            if zx2+zy2 > R:
                reason=True
                break
            if abs(derintx)+abs(derinty) < eps:
                n0+=1
                break
            derx0 = (zx*derx - zy*dery)
            derx0 = derx0 + derx0 +1.0
            dery = (zx*dery+zy*derx)
            dery = dery + dery
            derx = derx0
            derintx0 = (zx*derintx - zy*derinty)
            derintx0 = derintx0 + derintx0
            derinty = (zx*derinty+zy*derintx)
            derinty = derinty + derinty
            derintx = derintx0
            zy = zx*zy
            zy = zy+zy + y0
            zx = zx2-zy2 + x0
            zy2 = zy*zy
            zx2 = zx*zx

        if reason== False:
            pixels[x, y]=mandel_color #Mandelbrot set
        else:
            r2 = zx2+zy2
            rd2 = derx**2+dery**2
            d2 = r2*math.log(r2)/rd2
            t = d2/seuil
            if t>1.0:
                t=1.0
                n2+=1
            if t<0.0:
                t=0.0
            pixels[x, y]=rgb_conv(i,t)

# hiển thị Fractal đã tạo sau và hoàn thành số lần lặp đã cho
print(' End computing at ' + str(datetime.datetime.now())+' - n1='+str(n1)+' - n2='+str(n2))
img.show()
img.save("Mandelbrot_Fractal.png", "PNG")
```

Chạy đoạn mã trên cho chúng ta kết quả sau:

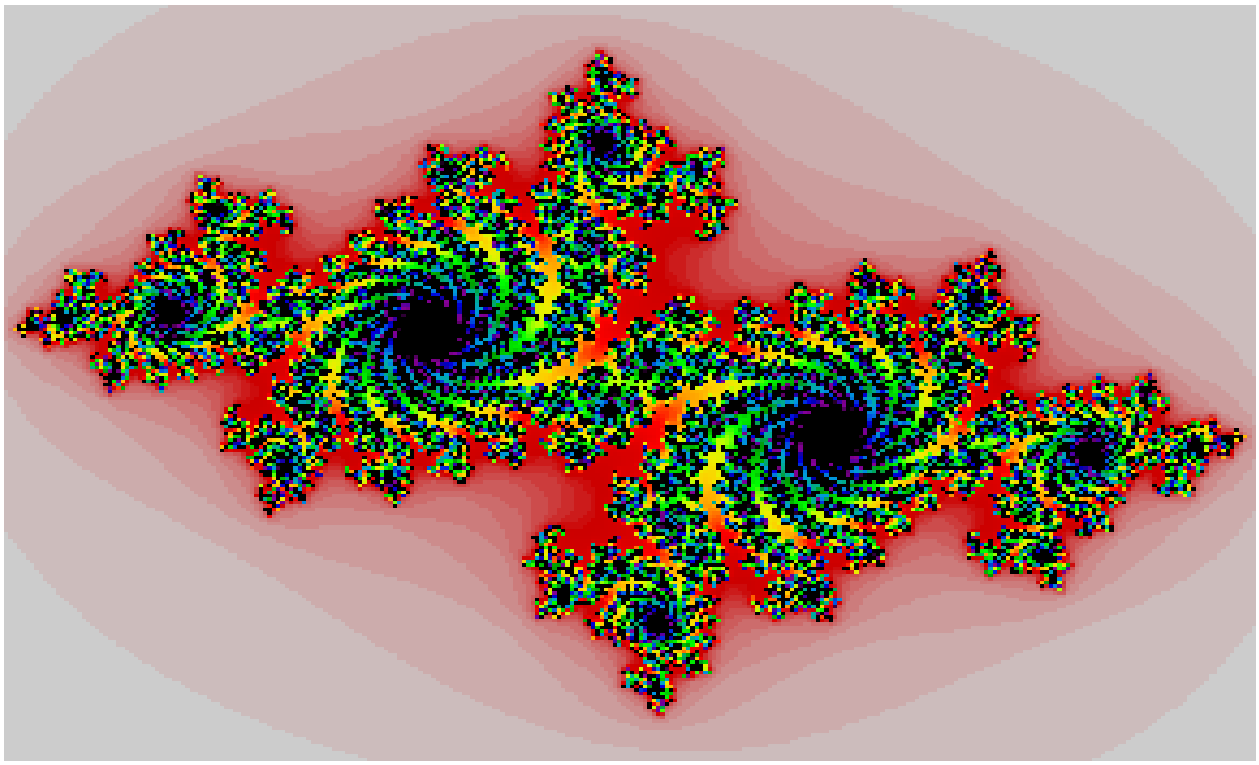


3.8. Julia set [14]

In [10]:
```
import itertools
from functools import partial
from numbers import Complex
from typing import Callable
import matplotlib.pyplot as plt
import numpy as np

def douady_hubbard_polynomial(z: Complex, * ,c: Complex):
    return z ** 2 + c

def julia_set(*,
              mapping: Callable[[Complex], Complex],   #hàm xác định bộ Julia
              min_coordinate: Complex,                 #tọa độ mặt phẳng phức dưới cùng bên trái
              max_coordinate: Complex,                 #tọa độ mặt phẳng phức phía trên bên phải
              width: int,                              #pixels trong trục tung
              height: int,                             #pixels trong trục hoành
              iterations_count: int = 256,             #Số lần lặp lại
              threshold: float = 2.) -> np.ndarray:
    imaginary_axis, real_axis = np.ogrid[
                        min_coordinate.imag: max_coordinate.imag: height * 1j,
                        min_coordinate.real: max_coordinate.real: width * 1j]
    complex_plane = real_axis + 1j * imaginary_axis
    result = np.ones(complex_plane.shape)
    for _ in itertools.repeat(None, iterations_count):
        mask = np.abs(complex_plane) <= threshold
        if not mask.any():
            break
        complex_plane[mask] = mapping(complex_plane[mask])
        result[~mask] += 1
    return result

if __name__ == '__main__':
    mapping = partial(douady_hubbard_polynomial, c=-0.7 + 0.27015j)
    image = julia_set(mapping=mapping, min_coordinate=-1.5 - 1j, max_coordinate=1.5 + 1j, width=800, height=600)
    plt.axis('off')
    plt.imshow(image, cmap='nipy_spectral', origin='lower')
    plt.show()
```

[14] Julia set - Wikipedia

Chạy đoạn mã trên cho chúng ta kết quả sau:



KẾT LUẬN

Sau một thời gian tìm hiểu và nghiên cứu về hình học fractal để có được bài báo cáo đồ án này, đã giúp em hiểu và có thêm một số kiến thức sau:

- Fractal là một thuật ngữ do nhà Toán học Mandelbrot đưa ra khi ông khảo sát những hình hoặc những hiện tượng trong thiên nhiên không có đặc trưng về độ dài.
- Hình học Fractal có nhiều ứng dụng trong cuộc sống và mở ra nhiều hướng nghiên cứu mới trong nhiều lĩnh vực như sinh học, y học, thiên văn, kinh tế, công nghệ thông tin,...
- Đặc điểm chung của các hình học fractal là tính tự đồng dạng, biểu hiện ở chỗ chúng có thể phân tích thành bộ phận nhỏ tùy ý mà mỗi bộ phận ấy lặp lại y hệt cấu trúc toàn thể. Tính tự đồng dạng ấy thể hiện rõ ở đường Peano, tam giác Spierpinki, tấm thảm Spierpinki,...
- Số chiều của hình học fractal dùng để đo kích thước của hình đó và thông thường số chiều của các hình học này đều không phải là một số tự nhiên.
- Tìm hiểu các bước tạo ra hay đặc điểm của một số đường fractal cơ bản như đường hoa tuyết Von Kock, bông hoa tuyết Von Kock, đường cong Peano, đường cong Hibert, tam giác Spierpinki, tấm thảm Spierpinki, cây fractal,...
- Sử dụng ngôn ngữ python để xây dựng chương trình vẽ một số đường fractal cơ bản nhưng những hình ảnh fractal thu được đều là những hình ảnh trong không gian 2D...

Bên cạnh những kết quả đã đạt được, em vẫn còn một vấn đề mà thời điểm này vẫn chưa giải quyết được đó là: chưa xây dựng được chương trình vẽ một số đường fractal trong không gian 3D.

TÀI LIỆU THAM KHẢO

[1]. Publisher W. H. Freeman and Co Publication, "The Fractal Geometry of Nature By Benoit B. Mandelbrot", *Country United States*, 1982.

[2]. Ph. D. Stan Blank, "Python Programming in OpenGL: A Graphical Approach to Programming", *Wayne City High School, Wayne Illinois*, 2009.

http://new.math.uiuc.edu/public198/ipython/stanblank/PyOpenGL.pdf

[3]. Martin Churchill, "Introduction to Fractal Geometry", 2004.

http://mdc.nfshost.com/fractals.pdf

[4]. John Wiley & Sons, "Fractal Geometry: Mathematical Foundations and Application", *Second Edition Kenneth Falconer*, 2003.

https://onlinelibrary.wiley.com/doi/pdf/10.1002/0470013850.fmatter

[5]. Yuval Fisher & Michael McGuire, "The Science of Fractal Images", 1988.

https://link.springer.com/content/pdf/bfm%3A978-1-4612-3784-6/1

[6]. Ritu Ahuja, "An Introduction to Fractals Geometry", *International Journal of Mathematics Trends and Technology (IJMTT)* - Volume 52 Number 10 December 2017.

http://www.ijmttjournal.org/2017/Volume-52/number-10/IJMTT-V52P593.pdf

[7]. Ankit Garg, Akshat Agrawal & Ashish Negi, "A Review on Natural Phenomenon of Fractal Geometry", *International Journal of Computer Applications (0975 – 8887)* - Volume 86 – No 4, January 2014.

https://research.ijcaonline.org/volume86/number4/pxc3893157.pdf