

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Кваліфікаційна наукова
праця на правах рукопису

ЮНАК ОСТАП МИРОНОВИЧ

УДК 621.396

ДИСЕРТАЦІЯ

**МЕТОДИ ПОБУДОВИ ТА ОБРОБКИ ФРАКТАЛЬНИХ ЗОБРАЖЕНЬ З
ВИКОРИСТАННЯМ РАНДОМІЗОВАНОЇ СИСТЕМИ ІТЕРАЦІЙНИХ
ФУНКЦІЙ**

172 – Телекомунікації та радіотехніка
(шифр і назва спеціальності)

17 «Електроніка та телекомунікації»
(галузь знань)

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ / Юнак Остап Миронович /

Науковий керівник

Стрихалюк Богдан Михайлович д.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

ЛЬВІВ – 2023

АНОТАЦІЯ

Юнак О.М. Методи побудови та обробки фрактальних зображень з використанням рандомізованої системи ітераційних функцій. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 172 – Телекомунікації та радіотехніка. – Національний університет «Львівська політехніка» Міністерства освіти і науки України, Львів, 2023.

В дисертаційній роботі розв'язано науково-практичне завдання розробки удосконалених методів побудови фрактальних зображень на основі рандомізованої системи ітераційних функцій (РСІФ) та розвитку методів їх розпізнавання, захисту та шифрування із використанням нейронних мереж.

Метою представленої дисертаційної роботи є зменшення обчислювальної складності побудови та аналізу фрактальних структур шляхом використання рандомізованої системи ітераційних функцій та розвитку методів розпізнавання і обробки фрактальних зображень із застосуванням нейронних мереж.

Об'єктом дослідження є процес побудови та обробки фрактальних зображень методами ітераційних функцій.

Предметом дослідження є моделі, методи та алгоритми побудови фрактальних структур за допомогою рандомізованої системи ітераційних функцій та нейронних мереж.

В процесі досліджень використано методи математичного моделювання, алгоритмів, програмного моделювання та машинного навчання.

У вступі обґрунтовано актуальність теми дисертаційної роботи, зазначено зв'язок роботи з науковими програмами і темами, сформульовано мету і завдання дослідження, описано наукову новизну та практичне значення отриманих результатів. Також наведено інформацію про впровадження результатів роботи, апробацію, публікації та особистий внесок здобувача.

У першому розділі «Аналіз методів та моделей побудови фрактальних структур» проведено детальне дослідження основних типів фракталів та способів їх створення, зокрема геометричних (множина Кантора, крива Гільберта-Пеано, крива Коха, сніжинка Коха, "дракон" Хартера-Хейтуея, H-фрактал, Крива Мінковського, Крива Леві, квадрат Серпінського, трикутник Серпінського, T-фрактал, Дерево Піфагора), алгебраїчних (множини Жюліа, Множина Мандельброта) та стохастичних фракталів. Проаналізовано методи та алгоритми побудови фрактальних структур, такі як ітеративний метод (системи ітераційних функцій), метод рекурентних співвідношень і метод випадкових процесів. Розглянуті переваги та недоліки кожного методу, що дало змогу визначити найкращий метод для побудови фрактальних структур. В результаті, вибрано ітеративний метод (Система ітераційних функцій СІФ) і наведено його переваги порівняно з методом рекурентних співвідношень та методом випадкових процесів. На сьогоднішній день існують дві відомі системи ітераційних функцій (СІФ): детермінована система ітераційних функцій (ДСІФ) і рандомізована система ітераційних функцій (РСІФ). Однак, для побудови рандомізованої системи ітераційних функцій (РСІФ) і визначення параметрів ітераційних функцій для конкретного фрактального зображення потрібні глибокі знання в математиці та інформатиці. Пошук цих коефіцієнтів є складною задачею, і для кожного випадку потрібно використовувати різні підходи. Іноді визначення коефіцієнтів не можливо розрахувати аналітично. Крім того, необхідно визначити достатню кількість ітерацій для забезпечення потрібної якості зображення фракталу. РСІФ також допомагає розв'язувати складні задачі фрактального аналізу, зокрема пошук фрактальної розмірності, яка визначає складність фрактала - геометричної форми з нерегулярною структурою та складністю. Фрактальна розмірність застосовується для вимірювання складності та структури об'єктів, які не можуть бути описані звичайними геометричними формулами. РСІФ, як фрактальна графіка і фрактальна розмірність, має широке застосування в різних галузях, таких як

фізика, біологія, економіка, комп'ютерна графіка, криптографія, медична діагностика, матеріалознавство, екологія, соціальні науки та інформаційні технології. РСІФ дозволяє аналізувати та моделювати складні системи в цих галузях, що допомагає розуміти їх природу та будову.

У другому розділі «Удосконалення методу побудови фрактальних зображень з використанням РСІФ» доведено, що використання рандомізованої системи ітераційних функцій (РСІФ) для створення фрактальних зображень представляє собою важливий напрямок досліджень, оскільки ця технологія може мати значний вплив на покращення якості та ефективності обробки даних. Також наведено результати математичного моделювання, спрямованого на створення фрактальних зображень з використанням розширеної РСІФ. Зокрема, математично виведено зв'язок між координатою точки, яка вибирається за допомогою рандомізованого алгоритму, і координатою середини першого сегмента ітерації, який вона утворює. Формалізовано математичну модель для фрактального зображення, яке буде генеруватися за допомогою рандомізованої системи ітераційних функцій (РСІФ). Запропонована математична модель дозволить краще розуміти процес побудови фрактальних структур та визначити, яким чином можна їх покращити. Побудована математична модель є важливим інструментом для оптимізації та покращення процесу створення фрактальних зображень та відкриває нові можливості для використання цих зображень у різних галузях, включаючи візуалізацію, графічний дизайн, та наукові дослідження. Дана модель є основою для подальшого удосконалення методу побудови фрактальних зображень. Відповідно удосконалено метод побудови фрактальних структур, використовуючи систему РСІФ, що базується на аналізі центрів перших відрізків ітерації та обчисленні коефіцієнтів РСІФ. Основною відмінністю цього методу є можливість створення та генерації фрактальних структур в режимі реального часу. Дослідження показало, що розроблений метод виявився надзвичайно ефективним і дозволяє суттєво заощадити

обчислювальні операції пропорційно до роздільної здатності зображення порівняно з існуючими методами створення фрактальних структур.

У третьому розділі «**Методи та модель розпізнавання і шифрування фрактальних зображень на основі нейронних мереж**» розроблено метод розпізнавання фрактальних зображень, використовуючи нейронну мережу, яка була піддана навчанню на основі вдосконаленого методу створення фрактальних структур з використанням рандомізованої системи ітераційних функцій (РСІФ). Унікальністю цього підходу є те, що нейронна мережа самостійно генерує необхідний набір даних для навчання, що робить цей метод надзвичайно ефективним. Важливо також відзначити, що розпізнавання фрактальних зображень відбувається в реальному часі. Розроблено інноваційний метод захисту документів, використовуючи фрактальні структури, створені з використанням рандомізованої системи ітераційних функцій. Цей підхід відрізняється від існуючих алгоритмів розміщення водяних знаків, оскільки він дозволяє пов'язати номер документа з фрактальною структурою, що виступає в якості водяного знаку, нанесеного на сам документ. Застосування цього методу дозволяє забезпечити подвійну та потрібну перевірку документа. Імовірність шахрайства чи спотворення документа зменшується завдяки використанню цього методу, що також реалізується в реальному часі. Розроблено метод шифрування фрактальних зображень, який використовує матриці перетворень для запобігання можливості дешифрування за допомогою нейронних алгоритмів. У цьому методі велику роль відіграє використання рандомізованого алгоритму для створення матриць перетворень, що гарантує абсолютну неможливість розкриття інформації. Порівняно з існуючими підходами, цей метод виділяється застосуванням надлишковості та рандомізованих матриць переміщення пікселів, а також змінами в яскравості, виконанням інверсії та модифікацією кольорів зображення з метою надійного захисту від можливості дешифрування. Результатом такого підходу є гарантована конфіденційність та цілісність інформації, що передається, при

цьому забезпечуючи збереження прозорості при передачі і забезпечення високого рівня безпеки від можливого розкриття даних. Важливо відзначити, що цей метод є надзвичайно ефективним для шифрування будь-яких існуючих зображень і дозволяє проводити процес шифрування в режимі реального часу, використовуючи обчислювальні ресурси низької потужності.

У четвертому розділі **«Практична реалізація програмного забезпечення та бібліотек запропонованих методів у WEB-технологіях»** створено генератор фрактальних зображень у стилі "Cantor dust" на основі розробленого методу, який використовує покращену РСІФ. Генератор розроблений за допомогою веб-технологій, таких як HTML для структури сторінки, JavaScript для програмування та CSS для стилізації. Для написання коду генератора використані бібліотеки jQuery і Bootstrap. Генератор дозволяє додавати необмежену кількість ітераційних фігур, налаштовувати їх параметри, розмір, положення та обертати їх під певним кутом. Важливо зазначити, що генератор працює в реальному часі завдяки оптимізованій системі ітераційних функцій. Також в ньому є можливість зміни якості зображення, яка залежить від кількості операцій РСІФ, і це використовується для навчання нейронної мережі щодо розпізнавання та відтворення фрактальних об'єктів. Точність створення фрактальних структур виявилася надзвичайно високою, з похибкою менше 0,001%. Розроблено програмну систему для відтворення та спотворення фрактальних зображень у стилі "Cantor dust" на основі розробленого методу побудови фрактальних структур. Система створена з використанням веб-технологій, таких як HTML для структури сторінки, JavaScript для програмування та CSS для стилізації. Для написання коду системи використані бібліотеки jQuery, Bootstrap та Brain.js. Бібліотека Brain.js дозволяє створювати та навчати нейронну мережу для отримання параметрів удосконаленої РСІФ. Також програма дозволяє зберігати зображення зі спотвореннями, що допомагає навчати нейронну мережу розпізнавати фрактальні зображення з низькою роздільною здатністю. Крім того, програма дозволяє відновлювати

фрактальні зображення та виводити параметри удосконаленої РСІФ з точністю 0,01%. Розроблено бібліотеку для інтеграції запропонованого методу в WEB-технології. Реалізація бібліотеки досить проста, і для отримання доступу до її функціоналу в іншому проєкті достатньо встановити відповідні ідентифікатори в тегах інтегрованого проєкту. Бібліотека розміщена на приватному репозиторії Bitbucket і може бути встановлена за допомогою відповідної команди. Це рішення дає змогу скоротити час написання програмних продуктів для роботи з цими алгоритмами і інтегрувати їх в інші програмні засоби та продукти. Розроблені практичні рішення підкреслюють значення даної бібліотеки як інструмента для розв'язання різних завдань у веб-розробці, аналізі зображень та інших областях, де важливі фрактальні структури та їх обробка.

У дисертаційних висновках представлено узагальнені результати проведеного дослідження і надані рекомендації щодо практичного використання цих результатів. Зокрема, розроблені науково-прикладні рішення щодо методів побудови та обробки фрактальних зображень з використання рандомізованої системи ітераційних функцій, а саме з генерування фрактальних зображень, розпізнавання та відновлення фрактального зображення, визначення параметрів удосконаленої РСІФ за допомогою нейронних мереж, захисту документів та зображень.

Результати дисертаційної роботи використано при формуванні навчальних дисциплін, що викладаються студентам (за навчальним планом) у навчальному процесі в Коледжі інформаційних технологій Національного університету «Львівська політехніка», зокрема для студентів спеціальності 172 «Телекомунікації та радіотехніка» в курсі лекцій з дисципліни «Інженерна та комп'ютерна графіка», а також у навчальному процесі спеціальності 123 «Комп'ютерна інженерія» в курсі лекцій з дисципліни «Надійність, діагностика та експлуатація комп'ютерних систем та мереж».

Основні результати, отримані в ході дослідження в рамках дисертаційної роботи, використані та успішно впроваджені у Львівській філії АТ

«Укртелеком», ТЗОВ "АПДЕЙТ ТЕХНОЛОДЖІС СИСТЕМЗ" та ТЗОВ «АСТРА-ЛЬВІВ». Впровадження результатів документально підтверджено актами. Застосування цих результатів стосується оброблення даних, розміщення інформації на веб-вузлах та комп'ютерного програмування в організації.

Ключові слова: Фрактал, рандомізована система ітераційних функцій, захист документів, нейронна мережа, фрактальна розмірність.

Список публікацій здобувача:

Наукові праці, у яких опубліковані основні результати дисертації

1. Yunak O., Strykhaliuk B., Klymash M., Pyrih Y., Shpur O. A Neuron Network Learning Algorithm for the Recognition of Fractal Image and the Restoration of Their Quality // Lecture Notes in Electrical Engineering. – 2023. – Vol. 965 LNEE – P. 574–584. (SciVerse SCOPUS).

2. Юнак О. М., Климаш М. М., Шпур О. М., Мрак В. Б. Математична модель розпізнавання фрактальних структур з використанням технології нейронних мереж // Infocommunication Technologies and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія. – 2023. – Vol. 3, № 1. – P. 1–9.

3. Yunak O. M., Strykhaliuk B. M., Klymash M. M. Efficiency of a randomized iteration function system over a deterministic iteration function system in constructing fractal images with limited resolution // Measurement and Computing Techniques in Technological Processes. 2023. No. 1. P. 5–12.

4. Yunak O. Protection of documents with the help of fractal images formed by a randomized system of iterating functions // Infocommunication Technologies and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія. – 2022. – Vol. 2, № 2. – P. 50–57.

5. Юнак О. М., Стрихалюк Б. М., Климаш М. М., Шпур О. М. Побудова фрактального зображення типу “канторів пил”, з використанням рандомізованої системи ітераційних функцій // Infocommunication Technologies and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія. – 2022. – Vol. 2, № 1. – Р. 19–25.

6. Юнак О. М., Стрихалюк Б. М., Юнак О. П. Шифрування графічної інформації за допомогою матриць перетворень для захисту від дешифрування нейронними алгоритмами // Штучний інтелект. – 2020. – № 2 (88). – С. 15–20.

7. Yunak O., Shpur O., Strykhaliuk B., Klymash M. Algorithm forming randomized system of iterative functions by based cantor structure // Infocommunication Technologies and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія. – 2021. – Vol. 1, № 2. – Р. 71–80.

Наукові праці, які засвідчують апробацію матеріалів дисертації

8. Юнак О., Пелещак Б., Охремчук О., Метлевич Я. Перетворення зображення фрактальної структури типу «Фрактальний пил» (Множина Кантора) в рандомізовану систему ітераційних функцій // Последните постижения на европейската наука: за XII международна научна практична конференция, 7-25 юни 2016. София «БялГРАД-БГ», 2016. Том 10. Р. 27-29.

9. Щерба В., Юнак О., Юнак О. Віртуальний паспорт для входу в нову мережу TELEGRAM OPEN NETWORK // Вітчизняна наука на зламі епох: проблеми та перспективи розвитку: матеріали Всеукраїнської наук.-практ. інтернет- конф., 18 березня 2020р.: зб. наук. праць. Переяслав, 2020. Вип. 59. С. 100-102.

10. Юнак О., Цебенко О., Юнак О. TON BLOCKCHAIN – компонент мережі TELEGRAM OPEN NETWORK // Вітчизняна наука на зламі епох: проблеми та перспективи розвитку: матеріали Всеукраїнської наук.-практ. інтернет- конф., 18 березня 2020р.: зб. наук. праць. Переяслав, 2020. Вип. 59. С. 102-103.

11. Пелещак Б., Охремчук Н., Юнак О., Кащук В. Використання веб-сервісів для хостингу проектів (систем контролю версій) та їх спільної розробки як засобів інтерактивного навчання методом проектів // матеріали XXV Всеукраїнської наук.- практ. інтернет-конф., 16-17 верес. 2016 р.:зб. наук. праць. Переяслав-Хмельницький, 2016.Вип. 25. С. 147-150.

12. Пелещак Б., Юнак О., Охремчук О., Метлевич Я. Використання систем моделювання мереж зв'язку // Вітчизняна наука на зламі епох: проблеми та перспективи розвитку: матеріали XXIV Всеукраїнської наук.- практ. інтернет-конф., 25-26 червня 2016р.: зб.наук. праць. Переяслав-Хмельницький, 2016. Вип. 24. С.87-90.

ABSTRACT

Yunak O.M. Methods of construction and processing of fractal images using a randomized system of iterative functions. – Qualification research paper as a manuscript.

The thesis for the Doctor of Philosophy Degree in the specialty 172 – Telecommunications and Radio Engineering. – Lviv Polytechnic National University, Ministry of Education and Science of Ukraine, Lviv, 2023.

In the dissertation, a scientific and practical task has been solved, which involves the development of advanced methods for generating fractal images using a randomized system of iterative functions (RSIF), as well as the advancement of methods for their recognition, protection, and encryption using neural networks.

The aim of the presented work is to reduce the computational complexity of constructing and analyzing fractal structures by utilizing a randomized system of iterative functions and advancing methods for recognizing and processing fractal images using neural networks.

The object of the research is the process of generating and processing fractal images using iterative function methods.

The subject of the research is the models, methods, and algorithms for constructing fractal structures using a randomized system of iterative functions.

The research involved the use of methods of mathematical modeling, algorithms, software simulation, and machine learning.

The introduction provides a rationale for the relevance of the dissertation topic, establishes the connection between the work and scientific programs and themes, outlines the research's objectives, describes the scientific novelty and practical significance of the obtained results. Additionally, it includes information about the implementation of the research findings, their validation, publications, and the individual contribution of the researcher.

In the first chapter titled “**Analysis of methods and models for constructing fractal structures**”, a detailed examination of the main types of fractals and the methods of their creation is conducted. This includes geometric fractals (such as the Cantor set, Hilbert-Peano curve, Koch curve, Koch snowflake, Harter-Heighway dragon, N-fractal, Minkowski curve, Levy curve, Sierpinski square, Sierpinski triangle, T-fractal, Pythagoras tree), algebraic fractals (such as Julia sets, Mandelbrot set), and stochastic fractals. The methods and algorithms for constructing fractal structures are analyzed, including the iterative method (systems of iterative functions), the method of recursive relationships, and the method of random processes. The advantages and disadvantages of each method are examined, leading to the selection of the iterative method (specifically, the randomized system of iterated functions - RSIF) and a comparison with the recursive and random process methods. At present, there are two well-known systems of iterative functions (SIF): the deterministic system of iterated functions (DSIF) and the randomized system of iterated functions (RSIF). However, creating a randomized system of iterated functions (RSIF) and determining the parameters of the iterative functions for a specific fractal image requires a deep understanding of mathematics and computer science. Finding these coefficients can be a complex task, and different approaches may be needed for each case. In some instances, determining the coefficients may not be analytically calculable. Additionally, it is necessary to determine a sufficient number of iterations to ensure the desired image quality. RSIF also aids in addressing complex problems in fractal analysis, including the search for fractal dimension, which characterizes the complexity of a fractal - a geometric shape with irregular structure and complexity. Fractal dimension is used for measuring the complexity and structure of objects that cannot be described using ordinary geometric formulas. RSIF, as a tool for fractal graphics and fractal dimension, has broad applications in various fields, including physics, biology, economics, computer graphics, cryptography, medical diagnostics, materials science, ecology, social sciences, and

information technology. RSIF allows the analysis and modeling of complex systems in these fields, aiding in understanding their nature and structure.

In the second chapter titled “**Improvement of the fractal image generation method using RSIF**”, it is demonstrated that the utilization of the randomized system of iterated functions (RSIF) for creating fractal images is a significant research direction, as this technology can have a substantial impact on enhancing the quality and efficiency of data processing. The results of mathematical modeling focused on creating fractal images using an extended RSIF are also presented. Specifically, a mathematical relationship between the coordinates of a point selected through a randomized algorithm and the coordinates of the center of the first segment of the iteration it forms is derived. A mathematical model for generating fractal images using the randomized system of iterated functions (RSIF) is formalized. This proposed mathematical model provides a better understanding of the process of constructing fractal structures and how they can be improved. The constructed mathematical model serves as a crucial tool for optimizing and enhancing the process of creating fractal images and opens up new possibilities for utilizing these images in various fields, including visualization, graphic design, and scientific research. This model forms the basis for further refinement of the method for constructing fractal images. As a result, the method for constructing fractal structures using the RSIF, based on the analysis of the centers of the first iteration segments and the computation of RSIF coefficients, has been improved. The primary distinction of this method is its ability to create and generate fractal structures in real-time. Research has shown that the developed method is exceptionally efficient and allows for significant computational savings relative to the image resolution compared to existing methods for creating fractal structures.

In the third chapter titled “**Methods and model for recognition and encryption of fractal images based on neural networks**”, a method for recognizing fractal images using a neural network is developed. This neural network is trained using an improved method for generating fractal structures with the randomized

system of iterated functions (RSIF). The uniqueness of this approach lies in the fact that the neural network autonomously generates the necessary dataset for training, making this method highly effective. Importantly, recognition of fractal images occurs in real-time. An innovative method for document protection is also devised, using fractal structures created with RSIF. This approach differs from existing watermarking algorithms because it associates the document's number with a fractal structure acting as a watermark applied directly to the document. This method enables double and triple verification of the document, reducing the likelihood of fraud or document tampering. The application of this method is real-time, enhancing document security. A method for encrypting fractal images is also developed, which uses transformation matrices to prevent decryption using neural algorithms. In this method, the use of a randomized algorithm for creating transformation matrices plays a significant role in ensuring the absolute impossibility of information disclosure. Compared to existing approaches, this method stands out for its use of redundancy and randomized pixel displacement matrices, as well as changes in brightness, inversion, and color modification to ensure robust protection against decryption. This approach guarantees the confidentiality and integrity of the transmitted information while maintaining transparency during transmission and providing a high level of security against potential data disclosure. Importantly, this method is highly efficient for encrypting existing images and allows for real-time encryption using low computational resources.

In the fourth chapter titled **“Practical implementation of software and libraries for the proposed methods in WEB technologies”**, a generator of fractal images in the style of "Cantor dust" is created based on the developed method, which utilizes the improved RSIF. This generator is developed using web technologies such as HTML for page structure, JavaScript for programming, and CSS for styling. The jQuery and Bootstrap libraries are used for writing the generator's code. The generator allows for the addition of an unlimited number of iterative shapes, the configuration of their parameters, size, positioning, and rotation at a certain angle.

Importantly, the generator works in real-time due to an optimized system of iterative functions. It also provides the option to change image quality, which depends on the number of RSIF operations and is used for training a neural network for recognition and reproduction of fractal objects. The precision of creating fractal structures is exceptionally high, with an error of less than 0.001%. A software system for rendering and distorting fractal images in the style of "Cantor dust" based on the developed method for constructing fractal structures is also developed. The system is created using web technologies, including HTML for page structure, JavaScript for programming, and CSS for styling. The jQuery, Bootstrap, and Brain.js libraries are used for writing the system's code. The Brain.js library allows for the creation and training of a neural network to obtain parameters for the improved RSIF. The program also allows for saving images with distortions, aiding in training the neural network to recognize low-resolution fractal images. Additionally, the program can restore fractal images and output the parameters of the improved RSIF with an accuracy of 0.01%. A library for integrating the proposed method into web technologies is developed. The implementation of the library is straightforward, and to access its functionality in another project, it is sufficient to set the appropriate identifiers in the tags of the integrated project. The library is hosted on a private Bitbucket repository and can be installed using the relevant command. This solution reduces the time required to develop software products for working with these algorithms and integrates them into other software tools and products. The practical solutions developed underscore the importance of this library as a tool for addressing various tasks in web development, image analysis, and other areas where fractal structures and their processing are crucial.

In the dissertation conclusions, the generalized results of the conducted research are presented, and recommendations for the practical utilization of these results are provided. Specifically, scientific and applied solutions have been developed for methods of constructing and processing fractal images using the randomized system of iterated functions (RSIF). This includes the generation of fractal images,

recognition and restoration of fractal images, determining the parameters of the enhanced RSIF through the use of neural networks, and the protection of documents and images.

Scientific and practical results of the research performed are used in the educational process of the Colleges of Information technology of the Lviv Polytechnic National University, particularly for students of specialty 172 «Telecommunications and radio engineering» in the course of lectures on «Engineering and computer graphics», as well as in the educational process of specialty 123 «Computer engineering» in the course of lectures on the discipline «Reliability, diagnostics, and operation of computer systems and networks».

The main results obtained during the research within the framework of the dissertation work have been utilized and successfully implemented in the Lviv branch of the PJSC "Ukrtelecom," LLC "UPDATE TECHNOLOGIES SYSTEMS," and LLC "ASTRA-LVIV." The implementation of the results has been formally confirmed by official documents. The application of these findings pertains to data processing, information placement on web nodes, and computer programming within the organization.

Key words: Fractal, randomized iterative function system, document protection, neural network, fractal dimension.

The list of author's publications:

Proceedings where basic scientific results of thesis were published

1. Yunak O., Strykhaliuk B., Klymash M., Pyrih Y., Shpur O. A Neuron Network Learning Algorithm for the Recognition of Fractal Image and the Restoration of Their Quality // Lecture Notes in Electrical Engineering. – 2023. – Vol. 965 LNEE – P. 574–584. (SciVerse SCOPUS).
2. Ostap Yunak, Mykhailo Klymash, Olha Shpur, Vasyl Mrak, Mathematical model of fractal structures recognition using neural network technology //

Infocommunication Technologies and Electronic Engineering – 2023. – Vol. 3, № 1. – P. 1–9.

3. Ostap Yunak, Bohdan Strykhaliuk, Mykhajlo Klymash (2023). Efficiency of a randomized system of iterative functions over a determinist system of iterative functions in the construction of fractal images with limited resolution capacity. Measuring and computing devices in technological processes, (1), 5–12. <https://doi.org/10.31891/2219-9365-2023-73-1-1>.

4. Yunak O. Protection of documents with the help of fractal images formed by a randomized system of iterating functions // Infocommunication Technologies and Electronic Engineering. – 2022. – Vol. 2, № 2. – P. 50–57.

5. O. Yunak, B. Strykhaliuk, M. Klymash, O. Shpur. Construction of the fractal image of the “cantor dust” type, using a randomized system of iterating functions // Infocommunication Technologies and Electronic Engineering. – 2022. – Vol. 2, № 1. – P. 19–25.

6. O.M. Yunak, B.M. Strykhaliuk, O.P. Yunak. Encryption of graphic information by means of transformation matrixes for protection against decoding by neural algorithms // Artificial Intelligence. – 2020. – № 2 (88). – C. 15–20.

7. Yunak O., Shpur O., Strykhaliuk B., Klymash M. Algorithm forming randomized system of iterative functions by based cantor structure // Infocommunication Technologies and Electronic Engineering – 2021. – Vol. 1, № 2. – P. 71–80.

Proceedings that certify an approvement of thesis materials

8. Yunak O., Peleshchak B., Okhremchuk O., Metlevich Y. Transformation of Fractal Structure Image of "Fractal Dust" (Cantor Set) into Randomized Iterated Function System // Latest Achievements in European Science: Proceedings of the 12th International Scientific Practical Conference, June 7-25, 2016. Sofia "ByalGRAD-BG", 2016. Vol. 10. P. 27-29.

9. Shcherba V., Yunak O., Yunak O. Virtual Passport for Access to the New TELEGRAM OPEN NETWORK // Domestic science at the turn of an era: Problems and prospects of development: proceedings of the all-Ukrainian scientific-practical internet conference, March 18, 2020: collection of scientific works. Pereyaslav, 2020. Issue 59. P. 100-102.

10. Yunak O., Tsebenko O., Yunak O. TON BLOCKCHAIN - Component of the TELEGRAM OPEN NETWORK // Domestic science at the turn of an era: Problems and prospects of development: proceedings of the all-Ukrainian scientific-practical internet conference, March 18, 2020: Collection of Scientific Works. Pereyaslav, 2020. Vol. 59. P. 102-103.

11. Peleshchak B., Okhremchuk N., Yunak O., Kashchuk V. Utilizing Web services for hosting projects (version control systems) and their collaborative development as means of interactive project-based learning // Proceedings of the XXV all-Ukrainian scientific-practical internet conference, September 16-17, 2016: Collection of scientific works. Pereyaslav-Khmelnysky, 2016. Issue 25. P. 147-150.

12. Peleshchak B., Yunak O., Okhremchuk O., Metlevich Y. Utilization of Network Modeling Systems // Domestic science at the turn of an era: Problems and prospects of development: proceedings of the XXIV all-Ukrainian scientific-practical internet conference., June 25-26, 2016: Collection of scientific works. Pereyaslav-Khmelnyskyi, 2016. Vol. 24. P. 87-90.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	21
ВСТУП	22
РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ ТА МОДЕЛЕЙ ПОБУДОВИ ФРАКТАЛЬНИХ СТРУКТУР.....	30
1.1 Види фракталів та методи їх створення.....	30
1.2 Аналіз методів та алгоритмів побудови фрактальних структур	50
1.3 Переваги та недоліки існуючих методів побудови фрактальних структур.	61
1.4 Застосування фрактальних структур в телекомунікаціях та радіотехніці та їх значення у наукових дослідженнях.....	68
Висновки до розділу 1.....	77
РОЗДІЛ 2. УДОСКОНАЛЕННЯ МЕТОДУ ПОБУДОВИ ФРАКТАЛЬНИХ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ РСІФ	79
2.1 Математичне представлення та моделювання фрактального зображення за допомогою РСІФ	79
2.2 Удосконалення методу побудови фрактальних зображень з використанням РСІФ.	92
2.3 Оцінка обчислювальної складності методів побудови фрактального зображення з використанням РСІФ та ДСІФ	105
Висновки до 2-го розділу.....	111
РОЗДІЛ 3. МЕТОДИ ТА МОДЕЛЬ РОЗПІЗНАВАННЯ І ШИФРУВАННЯ ФРАКТАЛЬНИХ ЗОБРАЖЕНЬ НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ.....	113
3.1 Метод розпізнавання фрактальних структур на основі нейронних мереж	113
3.2 Захист документів за допомогою фрактальних зображень, сформованих рандомізованою системою ітераційних функцій (РСІФ).	131

3.3 Модель швидкого розрахунку фрактальної розмірності зображення з використанням нейронних мереж та удосконаленого методу побудови фрактальних структур.....	141
3.4 Метод шифрування фрактальних зображень за допомогою матриць перетворень для захисту від дешифрування нейронними алгоритмами	154
Висновки до 3-го розділу.....	164
РОЗДІЛ 4. ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА БІБЛІОТЕК ЗАПРОПОНОВАНИХ МЕТОДІВ У WEB-ТЕХНОЛОГІЯХ	167
4.1 Розробка програмної моделі генерації фрактальних зображень типу «Cantor dust» на основі удосконаленого методу побудови фрактальних структур	167
4.2 Програмна система відновлення (спотворення) та відтворення фрактальних зображень типу «Cantor dust».....	177
4.3 Розробка бібліотеки для інтеграції запропонованих рішень у WEB-технологіях.....	189
Висновки до 4-го розділу.....	198
ОСНОВНІ РЕЗУЛЬТАТИ ТА ВИСНОВКИ	201
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	206
Додаток А. Акти впровадження	220
Додаток Б. Список публікацій здобувача за темою дисертації та відомості про апробацію результатів дисертації	225

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

IFS – Iterated function system.

RSIF – Randomized system of iterative functions.

DSIF - Deterministic system of iterative functions

СІФ – Система ітераційних функцій.

РСІФ – Рандомізована система ітераційних функцій.

ДСІФ – Детермінована система ітераційних функцій.

HTML – Hyper Text Markup Language — мова розмітки гіпертексту

CSS - Cascading Style Sheets - Каскадні таблиці стилів

JS - JavaScript динамічна, об'єктно-орієнтована, прототипна мова програмування.

ШІ – Штучний інтелект.

JSON – JavaScript Object Notation, це текстовий формат обміну даними між комп'ютерами.

API – Application Programming Interface, прикладний програмний інтерфейс.

GIT - Розподілена система керування версіями файлів та спільної роботи.

NPM - (Node Package Manager) це менеджер пакунків для мови програмування JavaScript.

ВСТУП

Актуальність теми.

У сучасному світі все більше зростає потреба у точному розпізнаванні та аналізі складних структур, які присутні у фрактальних зображеннях. Фрактальні зображення мають широкий спектр застосувань у різних галузях, включаючи комп'ютерну графіку, медичну діагностику, геоінформаційні системи, антенні системи зв'язку та криптографію. Такі зображення можуть містити велику кількість деталей та нюансів, що вимагає високої обробки та обчислювальної потужності. Фрактальна розмірність є одним із ключових показників при аналізі фрактальних структур і має велике значення в різних областях науки і технологій. Швидке визначення фрактальної розмірності є важливим з практичної точки зору, оскільки дозволяє здійснювати швидку оцінку та аналіз фрактальних структур. Такий аналіз може використовуватись для оптимізації дизайну антен, визначення їх ефективних параметрів, а також для виявлення та класифікації фрактальних об'єктів у різних додатках, наприклад, в області медичного зображення чи аналізу геологічних даних.

У контексті шифрування, застосування фрактальних методів дозволяє забезпечити високий рівень захисту інформації. Фрактальне шифрування забезпечує складність дешифрування та стійкість до криптоаналізу, що є важливим у забезпеченні конфіденційності передаваної інформації.

Крім того, розпізнавання фракталів з використанням нейронних мереж відкриває нові можливості для автоматичного аналізу та інтерпретації фрактальних даних. Це стає особливо актуальним у сучасному світі, де обсяги інформації зростають експоненціально, і потрібні ефективні методи обробки та аналізу цих даних.

Системи ітераційних функцій (SIF) є потужними методами для побудови фрактальних структур. Одна з основних особливостей SIF полягає в їхній здатності генерувати складні та деталізовані зображення шляхом повторення простої функції. Незважаючи на свою ефективність і гнучкість, системи

ітераційних функцій також мають певні недоліки. Зокрема, вибір правильних параметрів та функцій для системи ітераційних функцій може бути нетривіальним завданням, і неправильний вибір може призвести до отримання некоректних або непередбачуваних результатів. Крім того, системи ітераційних функцій не завжди забезпечують повну самоподібність або деталізацію на всіх масштабах, що може обмежувати їхню універсальність та точність при моделюванні складних фрактальних структур. Сам процес побудови фрактальних структур може бути вимогливим щодо обчислювальних ресурсів, особливо при генерації складних та деталізованих зображень. Таким чином, при використанні систем ітераційних функцій необхідно уважно підходити до вибору параметрів та забезпечувати достатні ресурси для їх обчислення з метою досягнення бажаних результатів. Крім того, інтерпретація та використання фрактальної розмірності може бути складною, оскільки вона залежить від вибору методів обчислення та параметрів аналізу. Традиційні підходи щодо визначення фрактальної розмірності вимагають значної обчислювальної потужності та часу, особливо для складних структур та великих даних. Це може створювати обмеження для застосування фрактальної розмірності в реальному часі або великих масштабах.

Отже, можна зробити висновок, що фрактальний аналіз та методи побудови та обробки фрактальних зображень з використанням рандомізованої системи ітераційних функцій є актуальними та знаходять широке застосування в різних галузях науки та технології.

Основні недоліки існуючих систем побудови та обробки фрактальних зображення включають в себе складність налаштування параметрів, вимагають значних обчислювальних ресурсів, не можуть автоматизовано впоратися з визначенням параметрів фрактальних зображень, не можуть розпізнавати та відновлювати спотворені фрактальні зображення, не можуть обробляти великі об'єми даних, не відповідають сучасним вимогам швидкості, надійності та точності.

У зв'язку із тим, що системи ітераційних функцій знаходять широке застосування в різних галузях науки та технології, а традиційні методи побудови фрактальних структур та їх аналізу вже не відповідають сучасним вимогам швидкості, надійності та точності обробки даних, актуальним науково-практичним завданням є розробка удосконалених методів побудови фрактальних зображень на основі рандомізованої системи ітераційних функцій (РСІФ) та розвитку методів їх розпізнавання, захисту та шифрування із використанням нейронних мереж.

Зв'язок роботи з науковими програмами, планами, темами. Тематика дисертаційного дослідження виконувалась у відповідності до наукового напрямку кафедри радіоелектронних пристроїв та систем Національного університету «Львівська політехніка», в межах науково-дослідної роботи: «Стратегічні напрямки, методи і засоби цифровізації та інтелектуалізації енергетичних систем з використанням сучасних інформаційно-комунікаційних технологій» (№ держреєстрації 0123U101692, (2023-2025 рр.)).

Мета і завдання дослідження. Метою представленої дисертаційної роботи є зменшення обчислювальної складності побудови та аналізу фрактальних структур шляхом використання рандомізованої системи ітераційних функцій та розвитку методів розпізнавання і обробки фрактальних зображень із застосуванням нейронних мереж.

Досягнення поставленої мети здійснюється розв'язанням таких завдань:

1. Аналіз існуючих методів побудови та обробки фрактальних структур.
2. Удосконалення методу побудови фрактальних зображень з використанням РСІФ.
3. Розвинення методу розпізнавання фрактальних структур для розпізнавання та відновлення фрактальних зображень типу «Cantor dust» з використанням РСІФ та нейронних мереж.
4. Розроблення моделі швидкого визначення параметру фрактальної розмірності зображення.

5. Розроблення способу захисту документів з використанням побудови фрактальних структур за допомогою РСІФ.

6. Розвинути метод шифрування фрактальних зображень для забезпечення прозорості передачі даних, конфіденційності і цілісності інформації.

7. Розроблення програмної моделі генератора фрактальних зображень з використанням удосконаленої рандомізованої системи ітераційних функцій.

8. Практична реалізація автоматизованої системи розпізнавання фрактальних зображень різної якості з використанням нейронних мереж.

9. Розробка бібліотеки для інтеграції запропонованого методу у WEB-технологіях.

Об'єктом дослідження є процес побудови та обробки фрактальних зображень методами ітераційних функцій.

Предметом дослідження є моделі, методи та алгоритми побудови фрактальних структур за допомогою рандомізованої системи ітераційних функцій та нейронних мереж.

Методи дослідження. Під час досліджень використано методи теорії графів, алгоритмів, оптимізації, імітаційного моделювання, математичної статистики та машинного навчання.

Наукова новизна отриманих результатів.

1. Удосконалено метод побудови фрактальних структур, який, на відміну від відомих, використовує модифіковану рандомізовану систему ітераційних функцій, параметри яких визначаються за допомогою геометричних властивостей фігур перших ітерацій, що дало змогу без значних обчислювальних ресурсів в реальному часі представляти будь-яке зображення у вигляді фрактальних структур.

2. Розвинуто метод розпізнавання фрактальних структур, який, на відміну від відомих, використовує нейронні мережі для автоматичного визначення параметрів рандомізованої системи ітераційних функцій на базі самогенерованих фрактальних зображень, що дало змогу розпізнавати складні

фрактальні зображення із різною якістю, формуючи їх математичні ітераційні функції для подальшої обробки.

3. Запропоновано метод шифрування фрактальних зображень, який, на відмінну від відомих, використовує надлишковість та рандомізовані матриці переміщення пікселів, зміни яскравості, інвертації та зміни кольорів зображення для захисту від дешифрування, що дало змогу забезпечити прозорість передачі даних, зберігаючи при цьому конфіденційність і цілісність інформації.

Практичне значення одержаних результатів полягає в тому, що:

1. Розроблено програмну модель генерації фрактальних структур з використанням рандомізованої системи ітераційних функцій, на основі якої доведено, що розроблений метод забезпечує значну економію обчислювальних операцій пропорційно роздільній здатності зображення в порівнянні з існуючими методами фрактальних структур. Відносна похибка у формуванні фрактального зображення становить менше 0,001%, що свідчить про високу точність його побудови фрактальної структури.

2. Розроблено програмний модуль навчання нейронної мережі для автоматичного визначення параметрів рандомізованої системи ітераційних функцій із похибкою менше 0,01%. Даний модуль здатен генерувати результати в режимі реального часу, що забезпечує швидку і ефективну обробку даних для визначення оптимальних параметрів фрактальних структур.

3. Розвинуто спосіб захисту документів з використанням побудови фрактальних структур за допомогою рандомізованої системи ітераційних функцій, який, на відмінно від існуючих, робить прив'язку номера документу до фрактального зображення у вигляді водного знаку нанесеного на документ із подвійною та потрійною верифікацією, що дає змогу додатково перевірити документ на недійсність, забезпечуючи більш високий рівень захисту у запобіганні підробці.

4. Розроблено програмний модуль відновлення якості фрактального зображення. Проведені математичні розрахунки показали ефективність використання алгоритму. Програмна реалізація алгоритму дозволяє задавати коефіцієнт якості зображення для відновлення та спотворення відновлення та спотворення якості фрактального зображення.

5. Розроблено модель для швидкого автоматичного визначення фрактальної розмірності зображення, що робить його ідеальним для застосування в широкому спектрі областей, включаючи обробку зображень, комп'ютерну графіку та аналізу даних. Визначення фрактальної розмірності відбувається у реальному часі без значних затримок, що дає змогу швидко аналізувати та обробляти великі обсяги даних.

Розроблені науково-прикладні рішення використано у навчальних дисциплінах, що викладаються студентам (за навчальним планом) в Коледжі інформаційних технологій Національного університету «Львівська політехніка», зокрема для студентів спеціальності 172 «Телекомунікації та радіотехніка» в курсі лекцій з дисципліни «Інженерна та комп'ютерна графіка», а також у навчальному процесі спеціальності 123 «Комп'ютерна інженерія» в курсі лекцій з дисципліни «Надійність, діагностика та експлуатація комп'ютерних систем та мереж», а також у держбюджетній науково-дослідній роботі.

Основні результати, отримані в ході дослідження в рамках дисертаційної роботи, використані та успішно впроваджені в ТЗОВ «АПДЕЙТ ТЕХНОЛОДЖІС СИСТЕМЗ», ПАТ «Укртелеком» та ПП «АСТРА-НЕТ». Впровадження результатів документально підтверджено актами. Застосування цих результатів стосується оброблення даних, розміщення інформації на веб-вузлах та комп'ютерного програмування в організації.

Особистий внесок здобувача. Основні наукові результати дисертаційної роботи отримано автором самостійно. У працях, опублікованих у співавторстві, внесок Юнака О.М. є вирішальним, зокрема авторові належать (нумерація

згідно Додатку Б: у роботах [1] – розроблення алгоритму навчання нейронної мережі для розпізнавання фрактальних зображень та відновлення їх якості, [2] – розроблення автоматизованого алгоритму визначення параметрів фрактальних структур за допомогою нейронних мереж; [3] – визначення ефективності рандомізованою системою ітераційних функцій над детермінованою системою ітераційних функцій при побудові фрактальних зображень з обмеженою роздільною здатністю, [5] – алгоритм визначення кількості операцій в РСІФ; [6] – розроблення методу шифрування графічної інформації за допомогою матриць перетворень для захисту від дешифрування нейронними алгоритмами; [7] – алгоритм перетворення зображення фрактальної структури в рандомізовану систему ітераційних функцій.

Апробація результатів дисертації. Основні наукові результати та положення дисертації представлені, доповідались та обговорені на 4-х міжнародних науково-технічних конференціях та наукових семінарах: Вітчизняна наука на зламі епох: проблеми та перспективи розвитку: матеріали Всеукраїнської науково-практична інтернет-конференція (м. Переяслав, 2020р.); XII Міжнародна науково-практична конференція «Останні досягнення на європейській науці» (м. Софія 2016р.); XXV Всеукраїнської науково-практична інтернет-конференція «Вітчизняна наука на зламі епох: проблеми та перспективи розвитку» (м. Переяслав-Хмельницький 2016); XXIV Всеукраїнської науково-практична інтернет-конференція «Вітчизняна наука на зламі епох: проблеми та перспективи розвитку» (м. Переяслав-Хмельницький 2016). Крім цього, дисертаційна робота у повному обсязі представлена на наукових семінарах кафедри телекомунікацій Національного університету «Львівська політехніка».

Публікації. За результатами досліджень, які викладені у дисертаційній роботі, опубліковано 12 наукових праць, з них 5 статей у наукових фахових виданнях України, 1 стаття у науковому періодичному виданні інших держав, що входять до наукометричних баз Scopus/Web of Science, 1 стаття у

періодичному виданні України, 5 у збірниках матеріалів і тез доповідей міжнародних та всеукраїнських конференцій.

Структура та обсяг роботи. Робота складається з переліку умовних скорочень, вступу 4 розділів, висновків, списку використаних джерел і 2 додатків. Загальний обсяг роботи складає 226 сторінок друкарського тексту, із них 8 сторінок вступу, 204 сторінок основного тексту, 144 рисунків, 7 таблиці, список використаних джерел із 130 найменувань та 2 додатків. Додатки містять акти впровадження результатів дисертаційної роботи та список праць автора.

РОЗДІЛ 1. АНАЛІЗ МЕТОДІВ ТА МОДЕЛЕЙ ПОБУДОВИ ФРАКТАЛЬНИХ СТРУКТУР

1.1 Види фракталів та методи їх створення

Бенуа Мандельброт запропонував поняття фракталу, підкреслюючи його фрактальну природу та складність, яка перевищує простіші форми. Він заохочував досліджувати незвичайні форми, породжені фрактальною геометрією, як спосіб відображення різноманітних природних явищ і об'єктів. Мандельброт розбивав, здавалося, випадкові математичні форми на повторювані елементи, розкриваючи їхню фрактальну структуру. Його відкриття мало значний вплив на фізику, астрономію та біологію. Вчений Річард Фос сприяв розвитку фрактальної геометрії, створивши кольорові малюнки, що допомогли відобразити красу фракталів, оскільки попередні дослідники не мали сучасних засобів комп'ютерної графіки. Завдяки цим працям розпочався бурхливий розвиток фрактальної геометрії, яка знайшла застосування у багатьох природних явищах і процесах. Термін "фрактал" походить від латинського слова "fractus", що означає "розбитий" або "дроблений", що вказує на те, що фрактал складається з фрагментів, а кожна його частина містить інформацію про весь об'єкт. Фрактальна геометрія є способом відображення складних об'єктів, і вона знайшла застосування, наприклад, у фрактальному стисненні даних у комп'ютерній науці.

У 1984 році вчені Х.О. Пайтген і П.Х. Ріхтер з Бременського університету організували публічну виставку, яка отримала величезний успіх. На виставці були представлені комп'ютерно створені картини, слайди і відеофільми фракталів, які були виготовлені у їхній лабораторії комп'ютерної графіки. Щоб пояснити глядачам суть цих картин, вони видавали брошуру під назвою "Гармонія хаосу і порядку" і пізніше каталог, який був доступний німецькою та англійською мовами і розійшовся за кілька місяців. У 1986 році вийшла їхня книга "Краса фракталів" [1]. Фракталами також займався норвезький фізик Енс

Федер, який в своїй книзі "Фрактали" простим і зрозумілим способом пояснював математичні властивості фракталів і наводив приклади їх застосування у гідродинаміці, океанології, гідрології та інших галузях. Крім того, він розглядав методи комп'ютерної графіки [2].

Фрактал є складною геометричною фігурою, яка складається з послідовності частин, подібних до всієї фігури у цілому, і повторюється при збільшенні масштабу. У 1623 році Галілео Галілей висловив думку про те, що вся наука записана у великій книзі Всесвіту, і для розуміння її необхідно вивчити мову, на якій вона написана - мову математики з її геометричними фігурами, такими як трикутники, кола тощо [1].

Структура фрактала залишається нетривіальною на будь-якому масштабі. Навіть невеликий фрагмент регулярної фігури (наприклад, коло або еліпс) у великому масштабі нагадує фрагмент прямої. Фрактали зберігають свою форму при зміні масштабу. Фрактальна розмірність, яка вимірює "порізаність", "зламаність" чи "хвилястість" фрактала, дозволяє порівнювати їх між собою. Ця розмірність збільшується зі зростанням складності фрактала, тоді як топологічна розмірність не враховує всіх змін, що відбуваються з лінією або поверхнею. Особливістю фракталів є те, що їх розмірність є дробовим числом.

Фелікс Хаусдорф і Абрам Безікович внесли великий вклад у розвиток теорії фракталів. Фелікс Хаусдорф (1868-1942) був німецьким математиком і вважається одним з основоположників сучасної топології. Абрам Самойлович Безікович (1891-1970), британський математик, був обраний членом Королівського наукового товариства в 1934 році і отримав численні відзнаки за свою роботу.

Мандельброт, в своїй книзі "Фрактальна геометрія природи", вводить поняття фрактала як множини, розмірність Хаусдорфа-Безіковича якої строго більша за топологічну розмірність [3].

Фрактал визначається фрактальною розмірністю D . Фрактальна розмірність D , також відома як розмірність Хаусдорфа-Безіковича, вимірює

рівень розкладу об'єкта E на частини розміром r , підраховуючи кількість $N(r)$ частин, що покривають цей об'єкт [1].

$$D = \lim_{r \rightarrow \infty} \frac{\log N(r)}{\log \frac{1}{r}} \quad (1.1)$$

де N — кількість відрізків, r — довжина відрізка

Фрактал - це складна геометрична форма, яка має унікальну властивість самоподібності, що означає, що ціла фігура складається з багатьох менших частин, кожна з яких виглядає подібно до фігури в цілому. У більш широкому розумінні, фрактали можуть бути описані як безліч точок у тривимірному просторі, які мають дробову метричну розмірність, що відрізняється від звичайної топологічної розмірності. Зрештою, можна сказати, що фрактал - це фігура, що нескінченно самоподібна, тобто кожен її елемент повторюється в зменшеному масштабі [3].

Важливо зазначити, що термін "фрактал" не є математичним терміном і не має точного загальноприйнятого математичного визначення. Це поняття використовується для опису фігур, які володіють однією чи декількома з таких властивостей, що наведені нижче [3]:

- володіє складною не тривіальною структурою, яка виявляється на будь-якому масштабі, що відрізняє його від регулярних геометричних фігур (наприклад, кола або еліпса), де при збільшенні масштабу ми можемо помітити спрощення структури в деяких місцях, і можливо побачити лише прямі лінії. У випадку фрактала, збільшення масштабу не приводить до спрощення структури, і на будь-якому масштабі ми можемо побачити складну і однакову деталізовану картину [1];

- являє собою самоподібну або наближен до самоподібної структуру;

- має дробову метричну розмірність або метричну розмірність, яка перевершує топологічну.

Види фракталів. Фрактали бувають різних видів, розглянемо основні з них:

- геометричні;
- алгебраїчні;
- стохастичні;

Геометричні види фракталів. Фрактали геометричних форм є дуже інтуїтивними та легкими в розумінні, і кожна людина може побачити їх в дії. Більшість з них можна легко намалювати на аркуші паперу в клітинку. Трикутник Серпінського, Сніжинка Коха, H-фрактал, T-фрактал, Дракон, Крива Леві та Дерево Піфагора є добрими прикладами таких фракталів.

Фрактали цього типу відзначаються своєю виразністю. Вони формуються шляхом простих геометричних конструкцій. Наприклад, у двовимірному просторі вони отримуються шляхом застосування певного геометричного генератора до ламаної (або поверхні у тривимірному випадку). На кожному кроці алгоритму кожен відрізок (елемент ламаної) замінюється ламаною-генератором у відповідному масштабі. Шляхом безкінечного повторення цієї процедури отримується геометричний фрактал.

Перші ідеї фрактальної геометрії з'явилися ще у XIX столітті. Кантор, застосовуючи просту рекурсивну процедуру, перетворив лінію на набір незв'язаних точок, що називається "Пилом Кантора". Він видаляв центральну третину лінії, а потім повторював цю операцію з отриманими відрізками (рис 1.1) [4].

Множина Кантора є однією з класичних фрактальних структур і може бути використана в деяких антенних конструкціях для досягнення певних ефектів або властивостей (рис.1.2) [5-13].



Рис. 1.1. Формування множини Кантора.

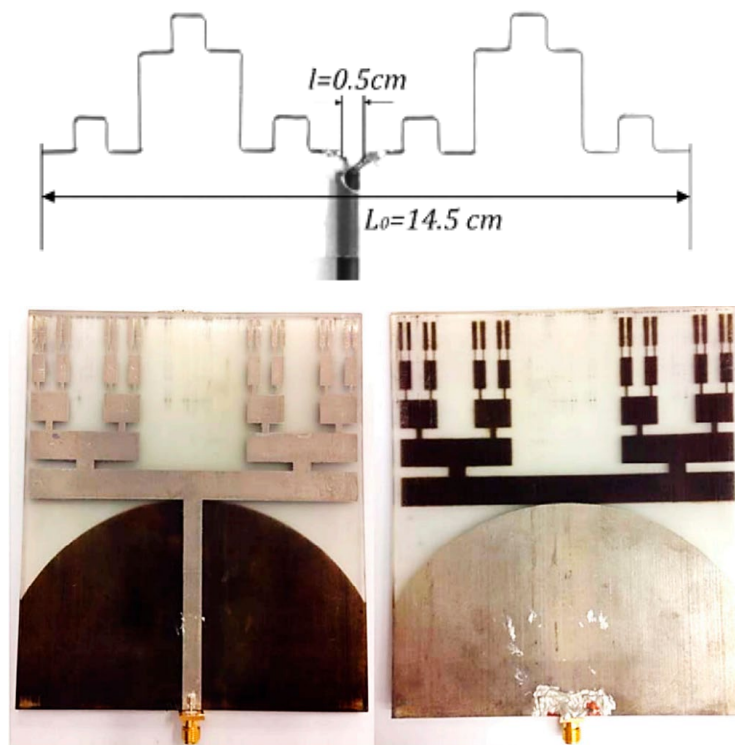


Рис 1.2 Приклади застосування множини Кантора у фрактальних антенах

Пеано створив особливий вид лінії, відомої як "Лінія Пеано" (рис. 1.3). Для її побудови італійський математик видалив нижню сторону квадрата. Так утворилась лінія Пеано першого порядку (рис. 1.3, а). Потім він зменшив квадрат вдвічі і створив 4 копії. Дві копії були розташовані паралельно одна одній, а інші дві були повернуті на чверть оберту в протилежних напрямках, і кінці ліній квадратів були з'єднані трьома однаковими відрізками, довжина яких дорівнювала стороні нового зменшеного квадрата. У результаті отримано лінію Пеано другого порядку (рис. 1.3, б). Цей процес можна продовжувати

нескінченно: крива другого порядку зменшується вдвічі, створюються чотири копії, з яких дві повертаються, і знову з'єднуються відрізками, які також зменшуються вдвічі (див. рис. 1.3, в-е). Цей алгоритм можна повторювати безкінечно [14]. Приклад застосування кривої Гільберта-Пеано в фрактальних антенах показано на рис.1.4 [15-25].

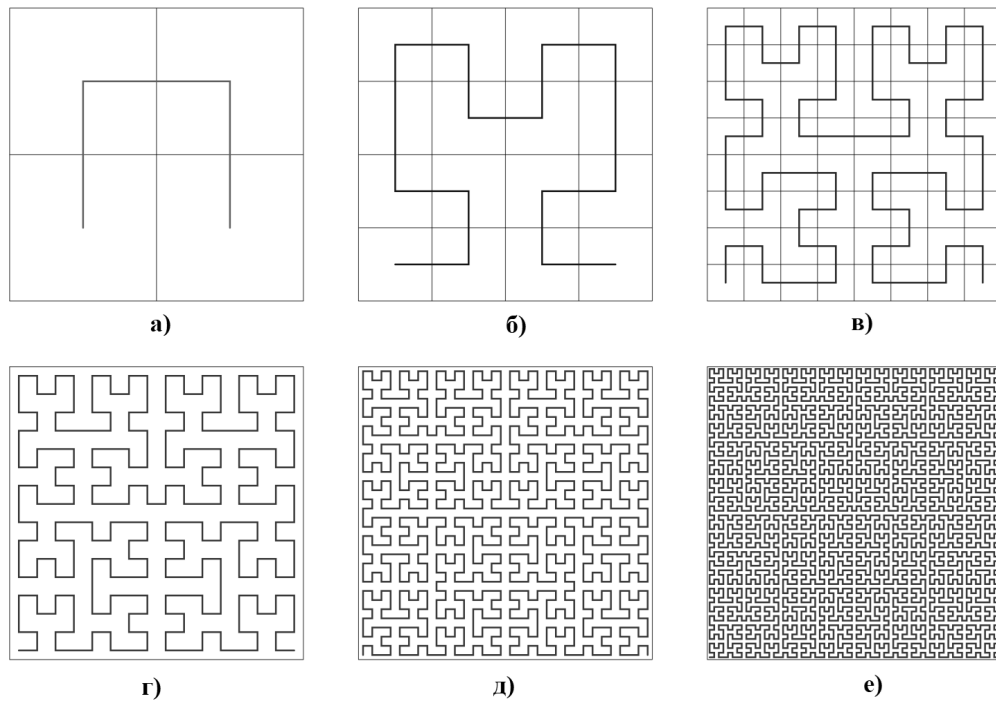


Рис. 1.3. Формування кривої Гільберта-Пеано

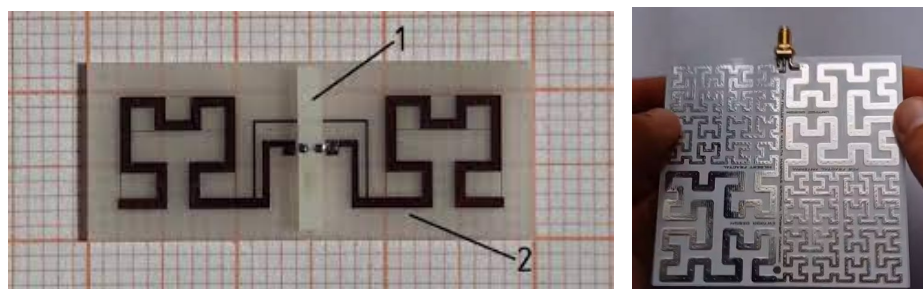


Рис. 1.4. Приклад застосування кривої Гільберта-Пеано в фрактальних антенах

Розглянемо фрактальний об'єкт - криву Коха, яка складається з тріадних елементів. Побудова цієї кривої розпочинається з одиничного відрізка (див. рис. 1.5, а), що відповідає нульовому поколінню кривої Коха. Далі кожний відрізок (у нульовому поколінні ми маємо лише один відрізок) замінюється на

спеціальний генеруючий елемент, як показано на рис. 1.5, б. Таким чином, отримуємо наступне покоління кривої Коха. У першому поколінні ця крива складається з чотирьох прямолінійних відрізків, кожен з яких має довжину $1/3$. Щоб отримати друге покоління, ми проводимо ті самі операції - кожен відрізок замінюється зменшеним генеруючим елементом. Отже, для отримання кожного наступного покоління всі відрізки попереднього покоління замінюються зменшеним генеруючим елементом. Крива n -го покоління, де n - скінченне число, називається передфракталом. Коли n наближається до нескінченності, крива Коха стає фрактальним об'єктом. Приклад застосування кривої Коха у фрактальній антені показано на рис.1.6 [26-33].

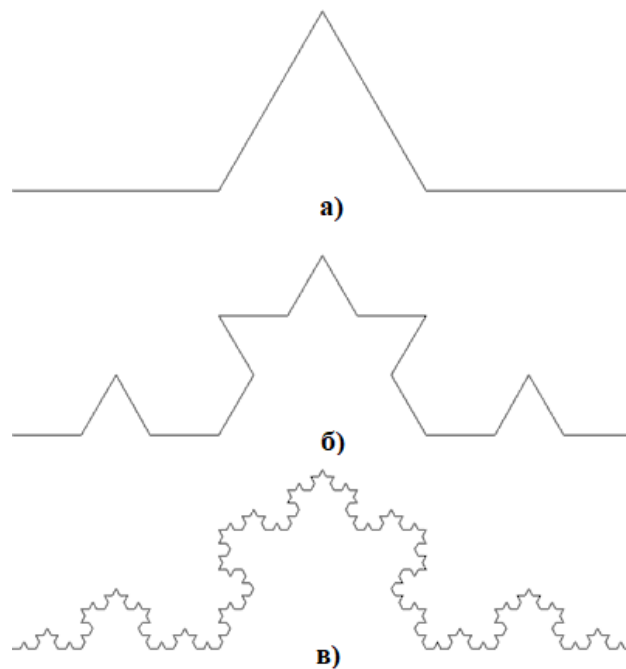


Рис. 1.5. Формування кривої Коха



Рис. 1.6. Приклад застосування кривої Коха у фрактальній антені

Сніжинка Коха - це дуже цікавий і відомий фрактал, що базується на рівносторонньому трикутнику. Кожна сторона трикутника замінюється на 4 менші сторони, кожна з яких має довжину $1/3$ від початкової. Цей процес повторюється нескінченну кількість разів, утворюючи сніжинку Коха нескінченної довжини. Цікаво, що ця нескінченна крива займає обмежену площу, як показано на рис. 1.7. Приклад застосування сніжинки Коха у фрактальній антені показано на рис.1.8 [34-40].

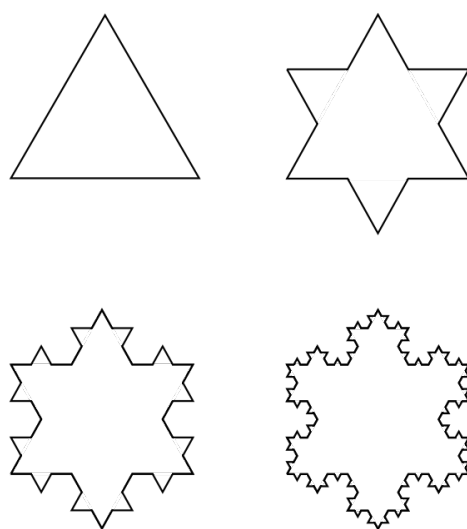


Рис. 1.7. Формування сніжинки Коха

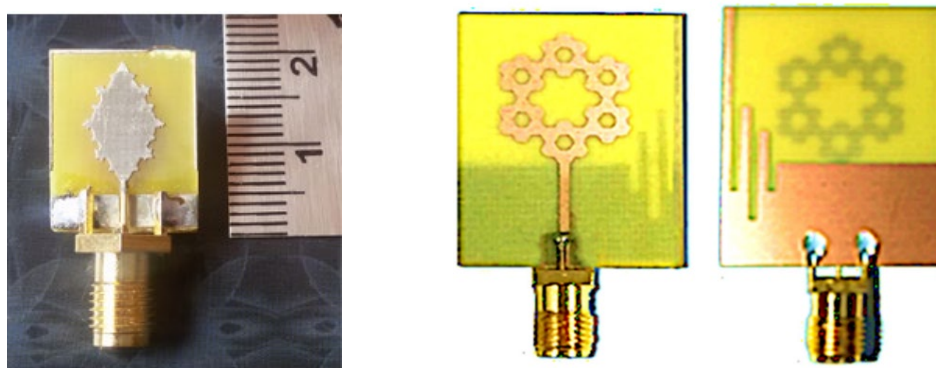


Рис. 1.8. Приклад застосування сніжинки Коха у фрактальній антені [34]

Н-Фрактал - це фігура, яка будується у формі літери "Н" (рис. 1.9), де вертикальні і горизонтальні відрізки мають однакову довжину. Далі, кожній з

чотирьох вершин фігури призначається зменшена в два рази копія. Потім до кожного з 16 кінців (які вже є) призначаються копії Н-фрактала, зменшені в 4 рази. Цей процес повторюється безкінечно. Зі збільшенням кількості кроків фрактал стає все більш схожим на заповнення певного квадрата. Н-Фрактал щільно заповнює цей квадрат, означаючи, що в будь-якій області навколо будь-якої точки квадрата можна знайти точки фрактала [3]. Приклади застосування Н-фракталу у фрактальних антенах показано на рис. 1.10 [41-46].

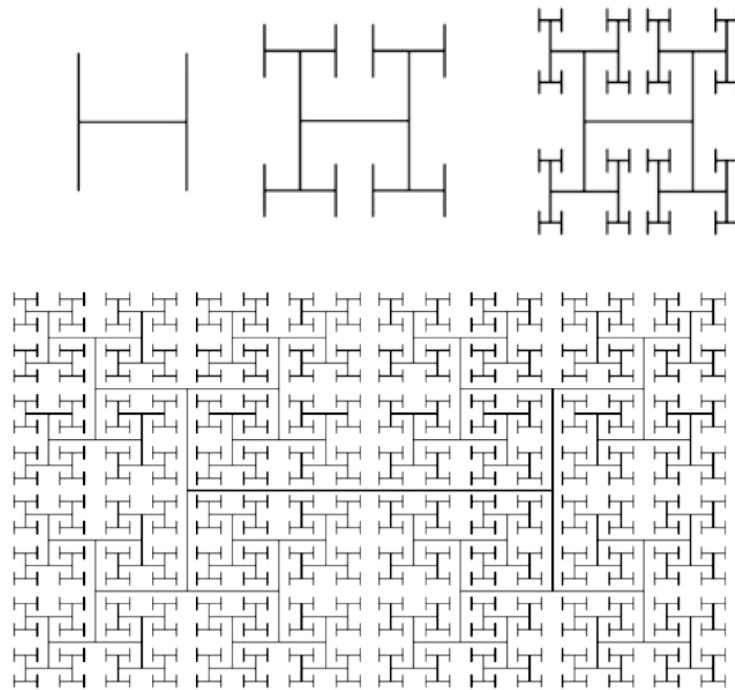


Рис. 1.9. Формування Н-фракталу

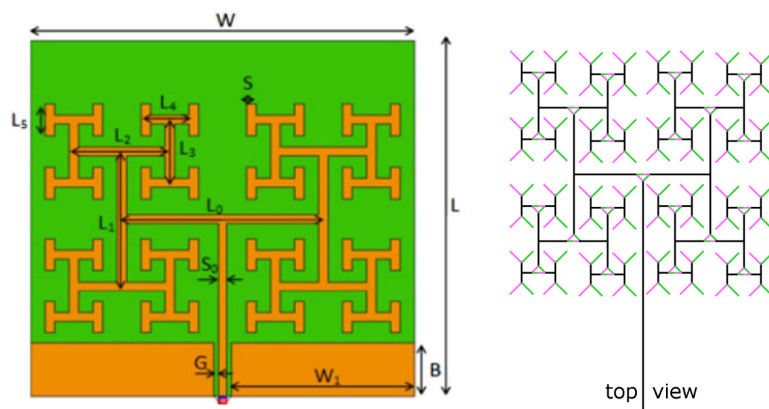


Рис. 1.10. Приклади застосування Н-фракталу у фрактальних антенах

Крива Мінковського (фрактал, запропонований Германом Мінковським) є геометричним фракталом, що формується за допомогою ініціатора, яким є відрізок (рис. 1.11, а), та генератора, яким є ламана, що складається з восьми ланок, де дві рівні ланки продовжують одна одну (рис. 1.11, б) [3]. Приклади застосування Кривої Мінковського у фрактальних антенах показано на рис.1.12 [47-58].

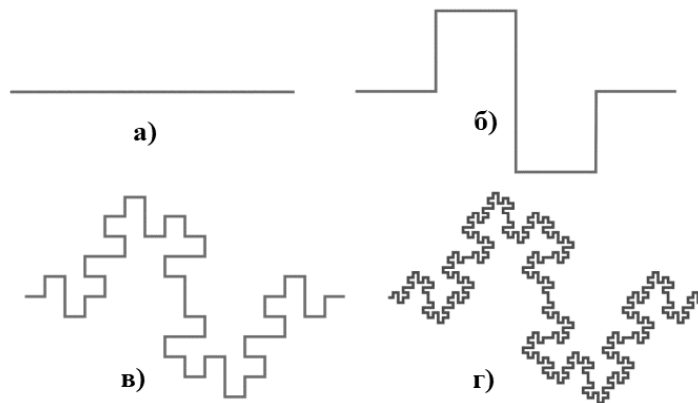


Рис. 1.11. Формування Кривої Мінковського

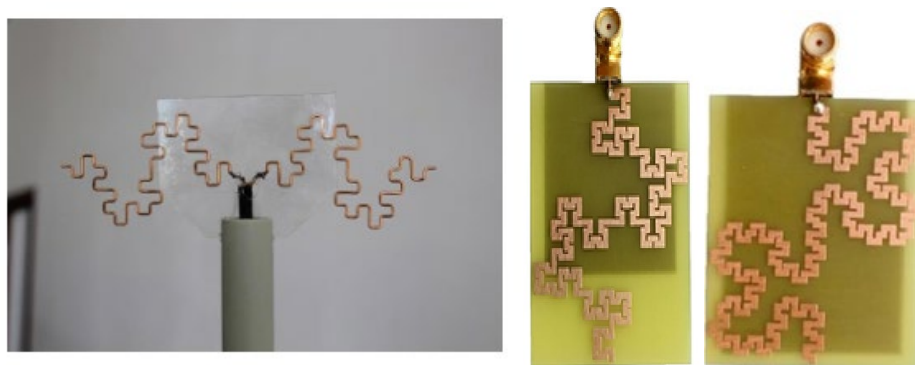


Рис. 1.12. Приклади застосування Кривої Мінковського у фрактальних антенах

Фрактал, відомий як квадрат Серпінського (рис. 1.13), був запропонований польським математиком Верацлавом Серпінським. Для його конструкції використовується початковий суцільний квадрат, який поділяється на 9 рівних квадратів, і при цьому видаляється середина центрального квадрата. На наступному кроці з решти 8 квадратів видаляються 8 центральних квадратів, і

так далі. Після нескінченного повторення цієї процедури, від початкового суцільного квадрата залишається замкнута підмножина, яку ми знаємо як килим Серпінського [14]. Приклади застосування квадрата Серпінського у фрактальних антенах показано на рис.1.14 [59-70].

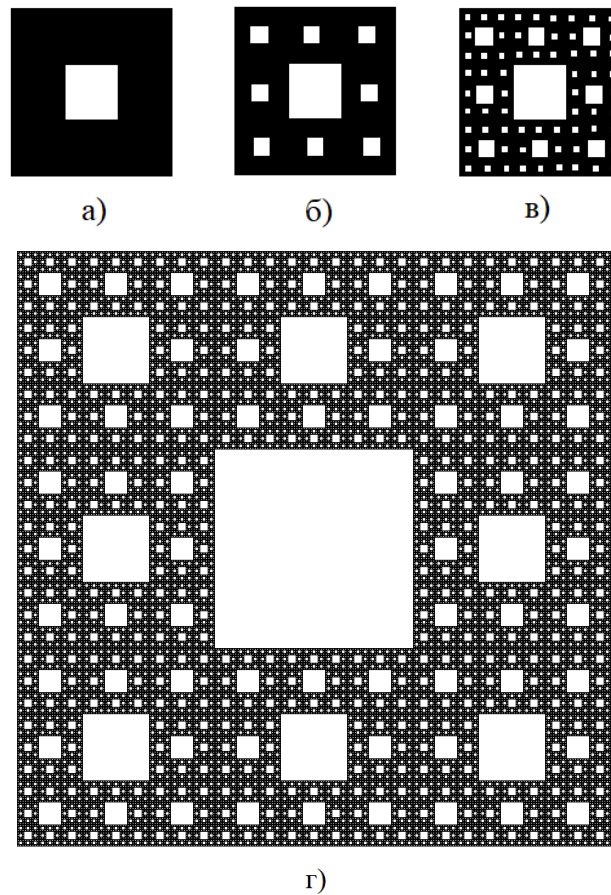


Рис. 1.13. Формування квадрата Серпінського

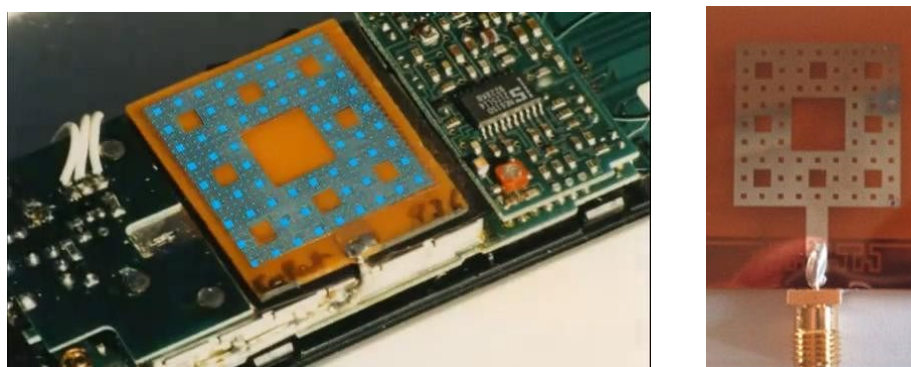


Рис. 1.14. Приклади застосування квадрата Серпінського у фрактальних антенах

У 1915 році Верацлав Серпінський дослідив інший фрактал, відомий як трикутник Серпінського (рис. 1.15). Цей фрактал також називають "серветкою" або "решіткою" Серпінського. Для побудови цього фрактала береться рівносторонній трикутник (рис. 1.15, а). На першому кроці видаляється трикутник, який має вершини в середині сторін початкового трикутника (рис. 1.15, б). На наступних кроках видаляються аналогічні трикутники з трьох менших трикутників, які залишилися після першого кроку, і так далі. Після нескінченного повторення цієї процедури від початкового рівностороннього трикутника залишається підмножина, яку ми знаємо як трикутник Серпінського [14]. Приклади застосування трикутника Серпінського у фрактальних антенах показано на рис.1.16 [71-75].

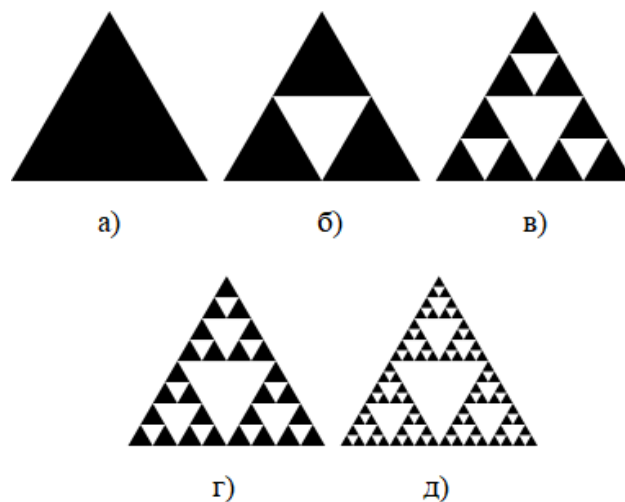


Рис. 1.15. Формування трикутника Серпінського

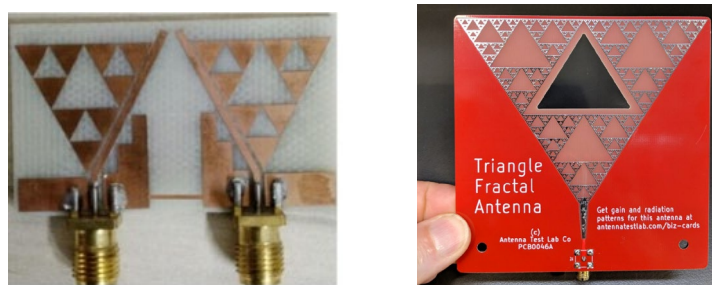


Рис. 1.16. Приклади застосування трикутника Серпінського у фрактальних антенах

T-фрактал, ймовірно, отримав свою назву через схожість з рейсшиною, на якій є перпендикулярна планка у формі букви Т. В англійському варіанті цей інструмент називається T-square.

Побудова T-фракталу починається з одиничного квадрата. На першому кроці потрібно зафарбувати білим кольором квадрат зі стороною $1/2$, розташований у центрі. Потім квадрат розділяється на 4 однакових квадрати, і в центрі кожного з них зафарбовується квадрат зі стороною $1/4$. Кожен з цих 4 квадратів знову розділяється на 4 частини, отримуючи загалом 16 квадратиків, і з кожним з них проводиться аналогічна процедура. Цей процес повторюється нескінченну кількість разів [3]. Приклади застосування T-фракталу у фрактальних антенах показано на рис.1.18 [77-82].

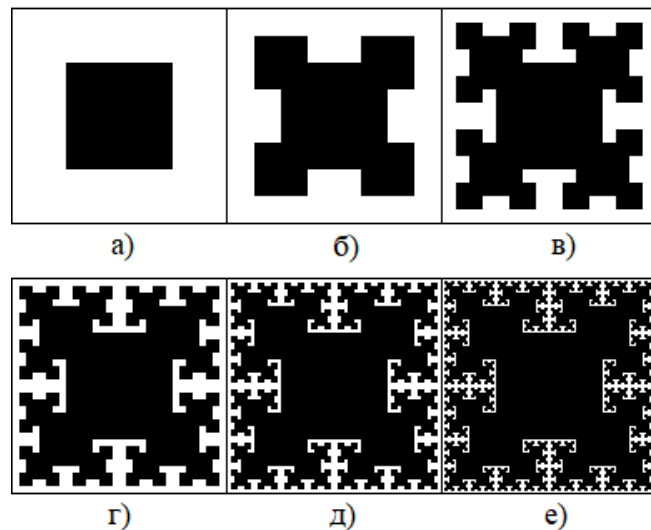


Рис. 1.17. Формування T-фракталу

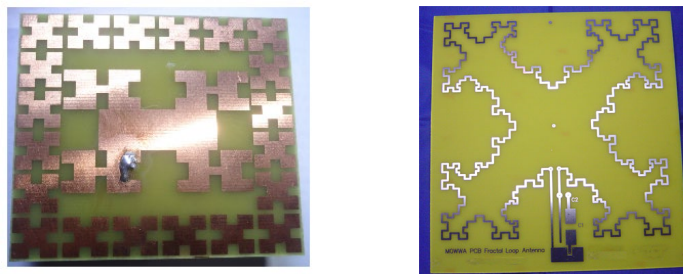


Рис. 1.18. Приклади застосування T-фракталу у фрактальних антенах

Дерево Піфагора названо так через те, що воно складається з трійок попарно дотичних квадратів, які утворюють прямокутні трикутники. Цей фрактал часто ілюструє теорему Піфагора: "Площі квадратів, побудованих на катетах прямокутного трикутника, дорівнюють площі квадрату, побудованого на гіпотенузі".

Дерево Піфагора обмежене в межах прямокутника розміром 6×4 , якщо найбільший квадрат має сторону довжиною 1 одиницю. Тому його площа не перевищує 24. Але з кожним наступним кроком додається в два рази більше трійок квадратиків, а їх розміри стають у $\sqrt{2}$ рази меншими. Таким чином, на кожному кроці додається та сама площа, яка дорівнює площі початкової конфігурації, тобто 2. Здається, що площа дерева повинна бути нескінченною. Однак, через перекриття квадратиків площа збільшується не так швидко, і вона все ж буде скінченною. Точне значення площі досі невідоме, і це є відкритою проблемою [14]. Приклади застосування дерева Піфагора у фрактальних антенах показано на рис.1.20 [83-89].

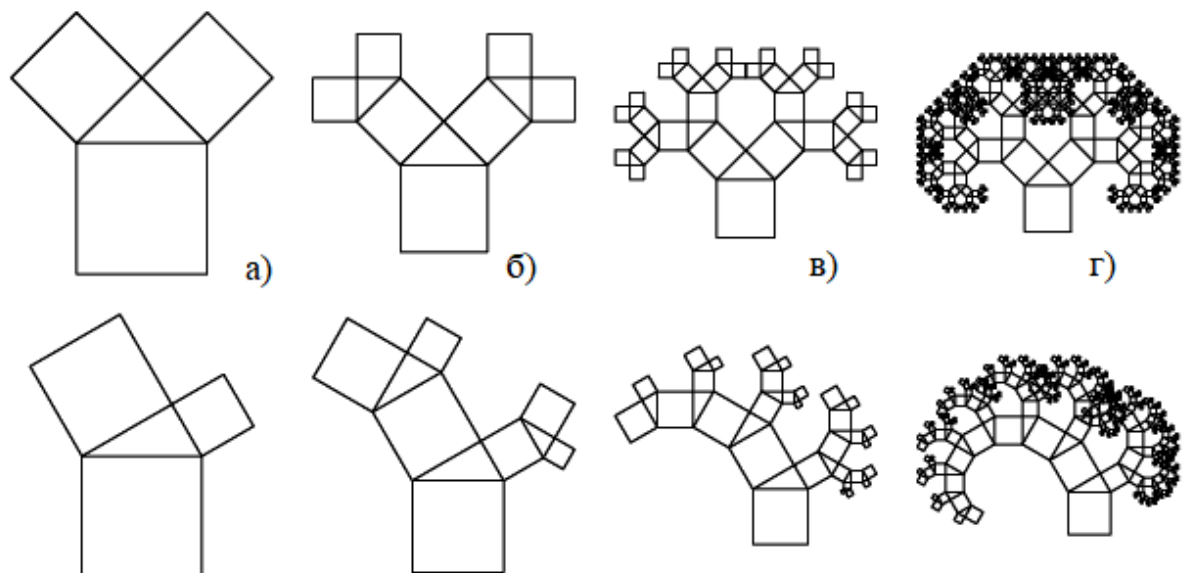


Рис. 1.19. Формування дерева Піфагора

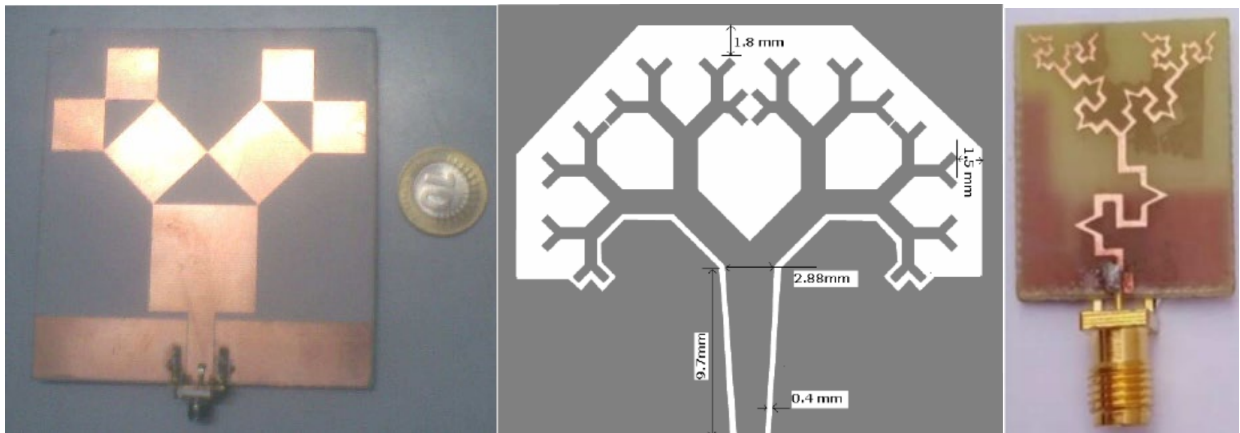


Рис. 1.20. Приклади застосування дерева Піфагора у фрактальних антенах

У сфері комп'ютерної графіки геометричні фрактали, такі як дерево Піфагора, використовуються для створення зображень дерев, кущів, берегових ліній. Двовимірні геометричні фрактали також застосовуються для створення текстур у тривимірних моделях об'єктів.

Алгебраїчні фрактали. Вони створюються шляхом застосування нелінійних процесів у n -вимірних просторах. Найбільше досліджені двовимірні процеси. Якщо ітераційний процес нелінійної системи інтерпретується як дискретна динамічна система, то можна користуватися термінологією теорії таких систем, як фазовий портрет, стійкий процес, аттрактор і т.д. [90]

Відомо, що нелінійні динамічні системи можуть мати кілька стійких станів. Конкретний стан, в якому опиниться система після деякої кількості ітерацій, залежить від її початкового стану. Тому кожен стійкий стан (або аттрактор) має визначену область початкових станів, з яких система обов'язково потрапить у дані кінцеві стани. Таким чином, фазовий простір системи розбивається на області, в яких відбувається притягання аттракторів. Якщо фазовий простір є двовимірним, то, зафарбовуючи області притягання різними кольорами, можна отримати кольоровий фазовий портрет цієї системи (ітераційного процесу). Змінюючи алгоритм вибору кольорів, можна отримати складні фрактальні зображення з химерними багатокольоровими візерунками.

Математикам стало несподіванкою те, що за допомогою простих алгоритмів можна породжувати дуже складні нетривіальні структури.

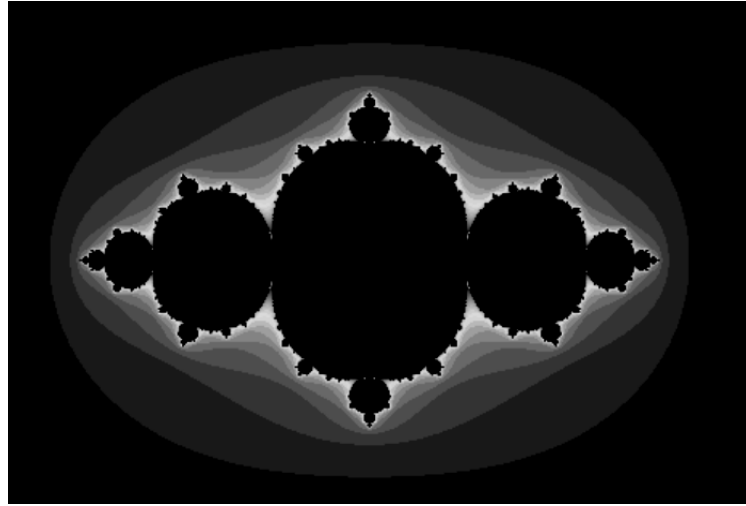
Множина Жюліа представляє собою динамічну систему, що визначена на комплексній площині. Вона складається з множини точок z , які збігаються до нескінченності при ітерації функції $f(z) = z^2 + C$, де C - певне комплексне число, а z_0 - комплексне число, що відповідає координатам відповідної точки на площині.

Побудова множин Жюліа виконується таким чином:

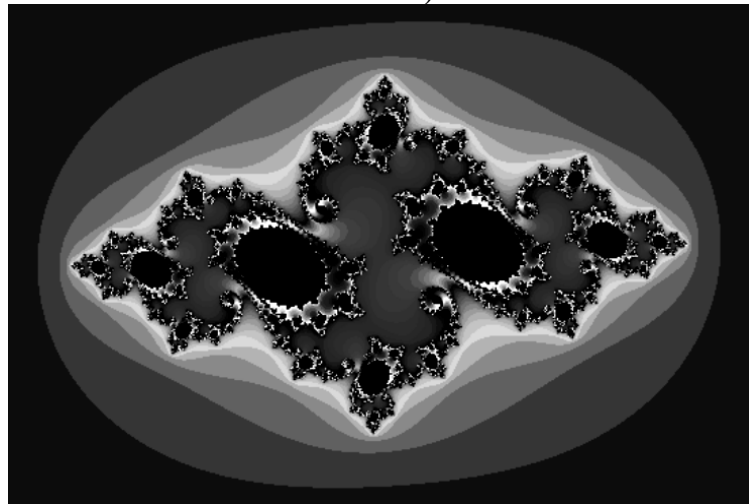
- вибирається комплексне число C та кількість ітерацій m (ідеальною є нескінченна кількість ітерацій);

- для кожної точки u в просторі обчислюється значення $f(z_m)$ після виконання заданої кількості ітерацій. Якщо значення точки протягом усіх ітерацій не збільшується до нескінченності, а залишається в обмеженій області, то ця точка входить до множини Жюліа. Якщо $|f(z_a)| > 2$, де a - номер ітерації від 0 до m , то це число однозначно збільшується до нескінченності і не входить до множини Жюліа.

Нижче показані приклади множин Жюліа з різними параметрами на рис. 1.21 а,б. Вони мають привабливий вигляд завдяки використанню кольору. У цих зображеннях кольори точок залежать від кількості ітерацій, необхідних для визначення того, чи вони входять до множини Жюліа. На колірній схемі чим яскравіший колір, тим більше ітерацій було зроблено для визначення належності точки до множини Жюліа. Чорні області відповідають точкам, які входять до множини Жюліа [3].



а)



б)

Рис. 1.21. Приклади множин Жюліа з різними параметрами

На наступному зображенні (рис 1.22) представлено еволюцію множини Жюліа. Видно, що зі збільшенням значення степені z , множина поступово перетворюється у кругову форму [14].

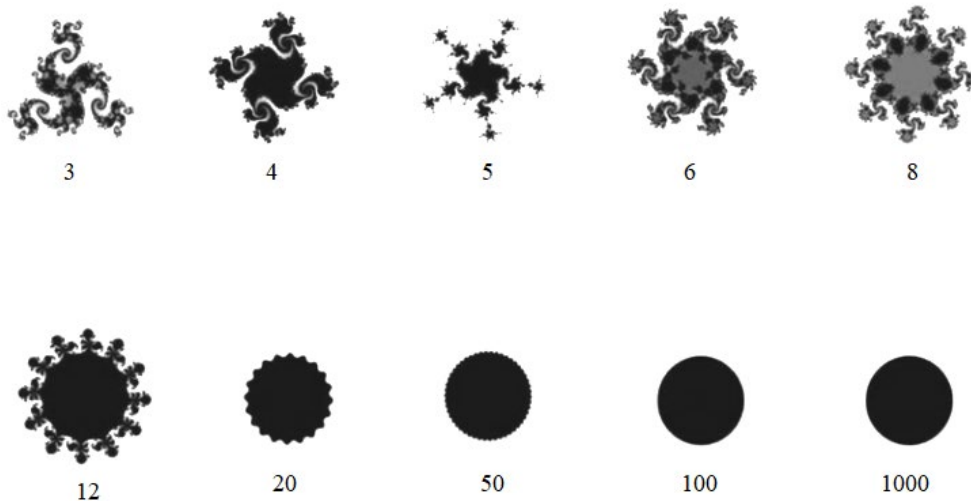


Рис. 1.22. Приклади множин Жюліа зі зміною ступеня

Множина Мандельброта представляє собою динамічну систему, що визначена на комплексній площині. Множина Мандельброта (рис 1.23) складається з точок c , для яких ітерація функції $f_c(z) = z^2 + c$ не розходиться. Іншими словами, якщо значення точки протягом усіх ітерацій не збільшується до нескінченності, а залишається в обмеженій області. Зображення множини Мандельброта створюється за допомогою кольорів, подібно до множини Жюліа. Якщо залишити c незмінним у формулі, а початкове значення z зробити змінним, то отримаємо відповідну множину Жюліа для точки c [3]. Приклади застосування Множина Мандельброта у фрактальних антенах показано на рис.1.24 [91-92].

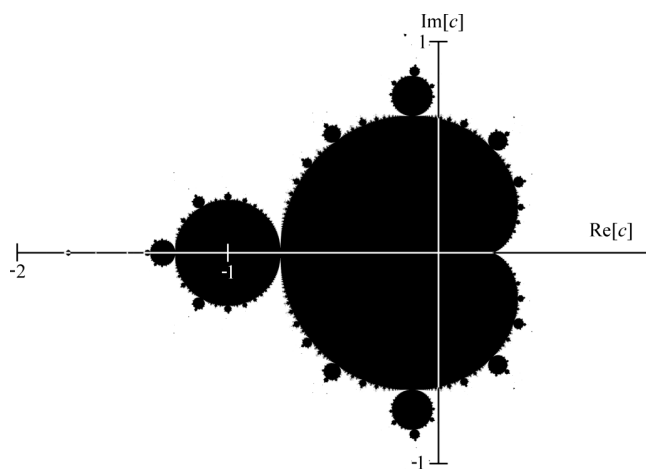


Рис. 1.23. Множина Мандельброта

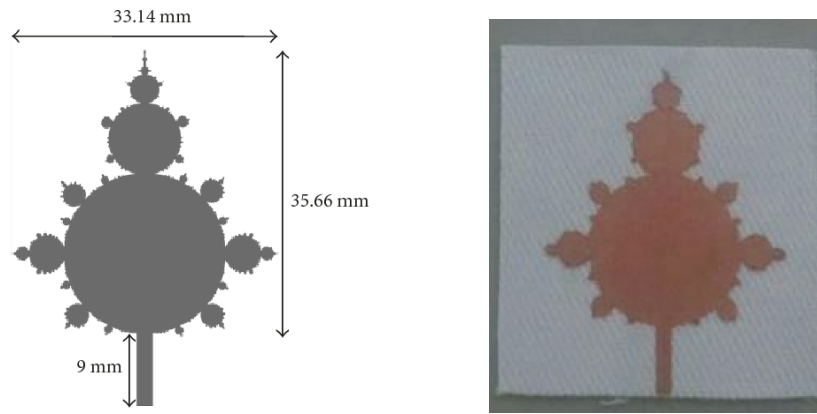


Рис. 1.24. Приклади застосування Множини Мандельброта у фрактальних антенах

На наступних зображеннях (рис. 1.25) представлено еволюцію множини Мандельброта. Зауважується, що при збільшенні значення степені z , множина Мандельброта веде себе аналогічно до множини Жюліа і поступово набуває форми кола [14]. Приклади застосування Множин Мандельброта у фрактальних антенах показано на рис.1.26 [1, 93].

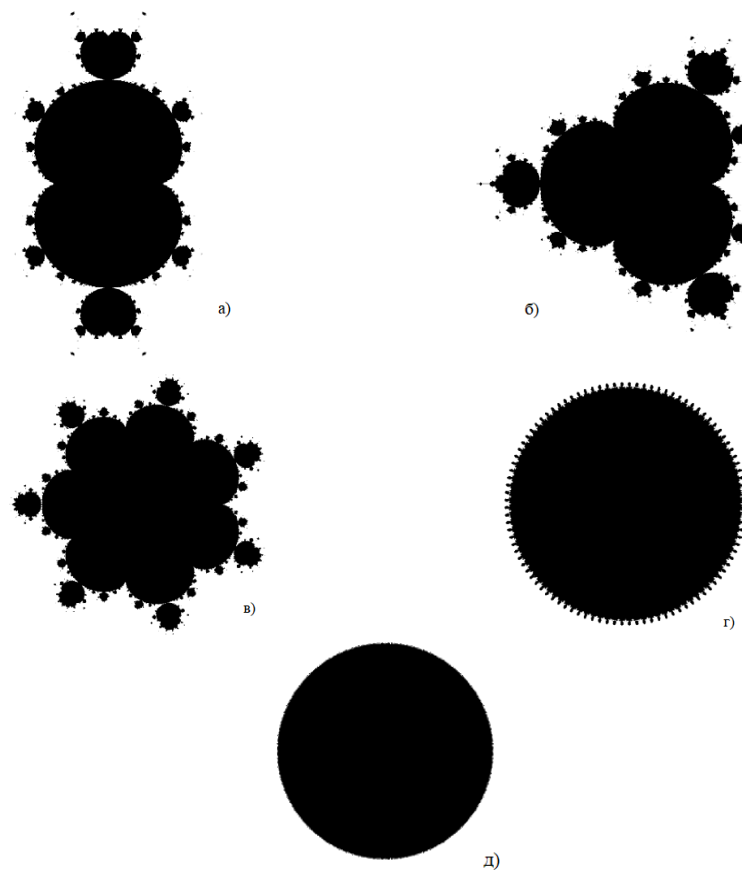


Рис. 1.25. Множина Мандельброта в різних степенях

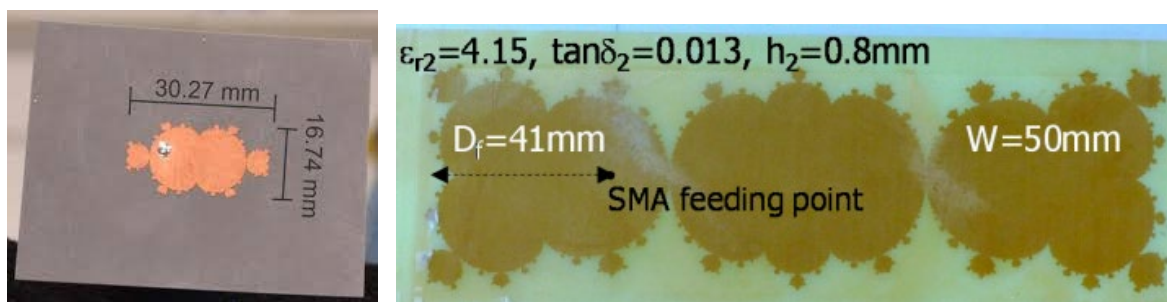


Рис. 1.26. Приклади застосування множин Мандельброта у фрактальних антенах

Стохастичні фрактали. Іншим відомим класом фракталів є стохастичні фрактали, які виникають, коли в ітераційному процесі випадковим чином змінюються його параметри. Це призводить до формування об'єктів, які нагадують природні утворення, такі як нерегулярні дерева або зубчасті берегові лінії. Двовимірні стохастичні фрактали широко використовуються для моделювання рельєфу ландшафту і поверхні океану. Тому їх часто використовують при створенні моделей різних природних об'єктів, таких як рельєф ландшафту, поверхня океану та інші (див. рис. 1.27) [94]. Приклади застосування Стохастичні фракталів у фрактальних антенах показано на рис.1.28. Приклади застосування стохастичні фракталів у фрактальних антенах показано на рис.1.28 [86, 95].

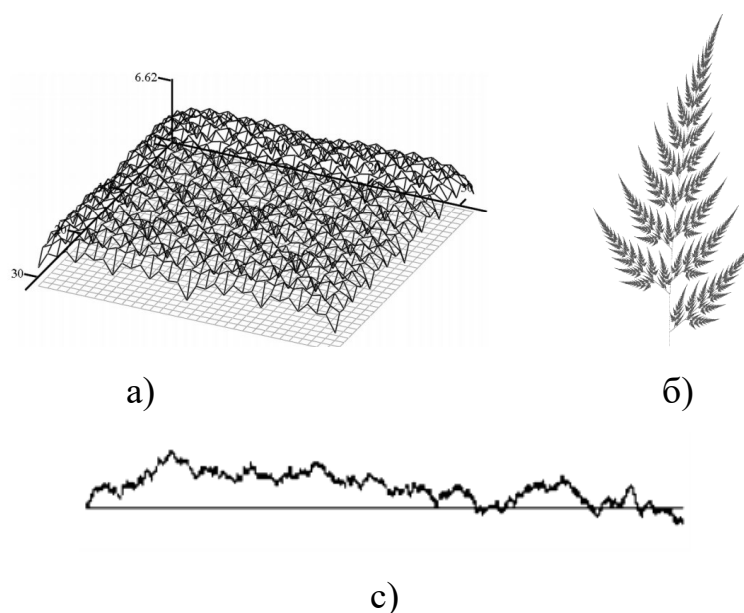


Рис. 1.27. Стохастичні фрактали

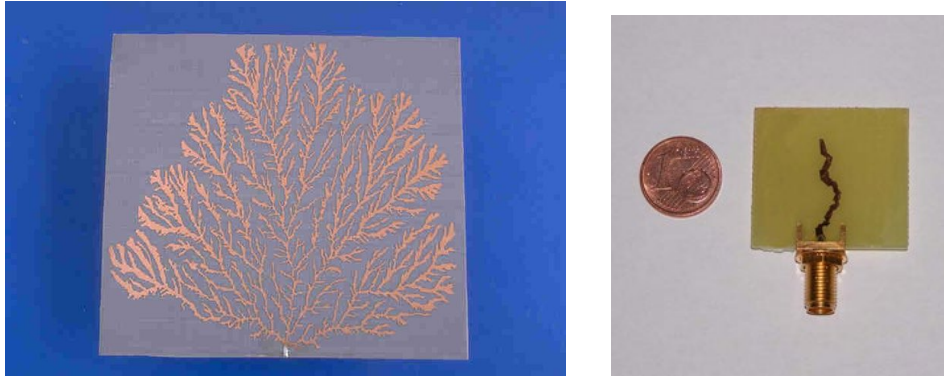


Рис. 1.28. Приклади застосування стохастичні фракталів у фрактальних антенах

1.2 Аналіз методів та алгоритмів побудови фрактальних структур

Існує три методи, що широко використовуються для створення (генерування) фракталів:

1. Ітеративний метод (Система ітераційних функцій). Цей метод ґрунтується на ітеративному процесі, де початкові значення пікселів або точок систематично змінюються за допомогою математичних правил або функцій. Кожна ітерація додає нову інформацію до фракталу, що дозволяє йому поступово розкривати деталі та складні структури.
2. Рекурентні співвідношення - це математичний підхід, за яким фрактали формуються шляхом застосування спеціальних співвідношень до кожної точки простору, такого як комплексна площина. Ці фрактали, як правило, мають квазісамоподібні властивості і часто називаються фракталами часу втечі або орбітальними. Деякі приклади таких фракталів включають множину Мандельброта, множину Жюліа, фрактал Ньютона і фрактал Ляпунова. Отримані фрактали таким методом відносяться до алгебраїчних.
3. Випадкові процеси. Фрактали, які формуються за допомогою стохастичних, а не детермінованих процесів, відомі як випадкові

процеси. Прикладами таких процесів є фрактальні ландшафти, траєкторія Леві та броунівське дерево. Останнє викликає утворення кластерів дифузійних концентратів та реакційних концентратів.

Система ітераційних функцій (Iterated Functions System). У середині 80-х років був розроблений метод Iterated Functions System (IFS) як простий спосіб створення фрактальних структур [94].

IFS представляє собою систему фіксованого класу функцій, які перетворюють одну багатовимірну множину на іншу. Простий IFS складається з афінних перетворень площини.

У 1988 році американські дослідники Барнслі і Слоан використали ідеї з теорії динамічних систем для стиснення і збереження графічної інформації. Вони назвали свій метод "методом фрактального стиснення інформації", оскільки геометричні фігури, що виникають у цьому методі, часто мають фрактальну природу, подібну до Мандельброта (1.2-1.3) [94].

$$\begin{aligned} X' &= AX + BY + C \\ Y' &= DX + EY + F \end{aligned} \quad (1.2)$$

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = T \left(\begin{bmatrix} X \\ Y \end{bmatrix} \right) = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} E \\ F \end{bmatrix} \quad (1.3)$$

На даний момент існує дві відомі системи ітераційних функцій (СІФ) детермінована система ітераційних функцій (ДСІФ) та рандомізована система ітераційних функцій (РСІФ).

Детермінована система ітераційних функцій (DIFS) є одним з методів побудови фрактальних зображень. Вона використовується для створення складних, деталізованих і повторюваних фрактальних структур шляхом ітераційної апроксимації.

Принцип роботи DIFS базується на понятті самоподібності, що означає, що фрактальні структури мають подібну форму на різних масштабах. В основі

DIFS лежить набір математичних функцій, які використовуються для генерації фрактальних зображень.

Для побудови фрактального зображення за допомогою DIFS використовуються наступні кроки:

1. Вибір базового об'єкта: спочатку обирається початковий об'єкт, який служить основою для створення фракталу. Це може бути проста геометрична фігура, наприклад, трикутник або квадрат.
2. Визначення ітераційної функції: для кожного елемента базового об'єкта визначається ітераційна функція. Ця функція використовується для перетворення кожного елемента в більш складну структуру, що додає деталі і складність до фракталу. Ітераційна функція може містити математичні операції, такі як зміщення, масштабування, обертання, зміна кольору тощо.
3. Повторення ітераційної функції: ітераційна функція застосовується до кожного елемента базового об'єкта, що створює нові елементи або підрозділи. Цей процес повторюється декілька разів, і кожен новий елемент стає базовим для наступної ітерації.
4. Зупинка ітераційного процесу: ітераційний процес може бути зупинений, коли досягнуто заданої кількості ітерацій або коли фрактал досягнув бажаного рівня деталізації.
5. Візуалізація фрактального зображення: на останньому кроці створене фрактальне зображення може бути візуалізоване на екрані або виведене у форматі зображення.

Принцип детермінованої системи ітераційних функцій дозволяє створювати складні і красиві фрактальні зображення шляхом повторення простих операцій над базовим об'єктом. Цей процес дозволяє досягти самоподібності і деталізації на різних рівнях масштабу, створюючи захоплюючі інтерактивні візуальні ефекти.

Один із прикладів використання DIFS - побудова фрактальних структур, таких як "дракон" Хартера-Хейтуея і крива Коха. Ми можемо ідентифікувати схожі частини в цих структурах і для кожної з них визначити коефіцієнти афінного перетворення. У фінальному афінному колажі будуть використані стільки афінних перетворень, скільки існує подібних частин, що утворюють ціле зображення [94].

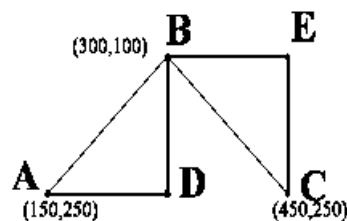


Рис. 1.29. Зображена заготовка для побудови IFS "дракона" Хартера-Хейтуея

Для створення цього фрактала розташовуємо перше покоління на сітці координат. Позначимо точки ламаної, яку отримали, як A, B, C. За правилами побудови цього фрактала на рис. 1.29 видно, що є дві частини, які подібні до цілого - ламані ADB і BEC. Знаючи координати кінців цих відрізків, ми можемо обчислити коефіцієнти для двох афінних перетворень, які перетворюють ламану ABC в ADB і BEC за допомогою формули (1.4).

$$\begin{aligned}
 X' &= -0.5X - 0.5Y + 490; \\
 Y' &= 0.5X - 0.5Y + 120; \\
 X' &= 0.5X - 0.5Y + 340; \\
 Y' &= 0.5X + 0.5Y - 110.
 \end{aligned}
 \tag{1.4}$$

Починаючи зі стартової точки (наприклад, $X=0$, $Y=0$) і застосовуючи ітеративно IFS до неї, після десятої ітерації ми отримуємо фрактальну структуру, зображену на рисунку 1.30, яка представляє собою "дракона" Хартера-Хейтуея. Кодом цього фрактала (стислим описом) є набір коефіцієнтів для двох афінних перетворень [94].

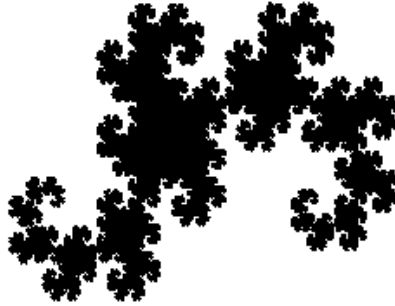


Рис. 1.30. Зображений "дракон" Хартера-Хейтуея, побудований за допомогою IFS

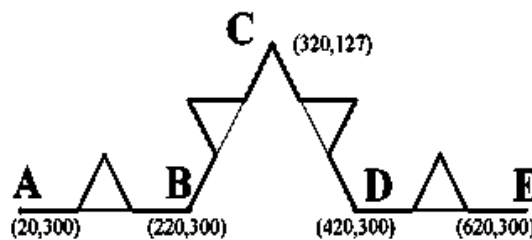


Рис. 1.31. Зображена заготовка для побудови IFS для кривої Коха

Також можна побудувати IFS для кривої Коха за аналогічним принципом. Легко помітити, що ця крива складається з чотирьох частин, які подібні до цілої кривої (рис. 1.31). Для побудови IFS розмістимо перше покоління цього фрактала на сітці координат [94].

Для побудови кривої Коха нам знадобиться набір афінних перетворень, який складається з чотирьох перетворень, як показано у формулі (1.5) [94].

$$\begin{aligned}
 X' &= 0.333X + 13.333; \\
 Y' &= 0.333Y + 200; \\
 X' &= 0.333X + 413.333; \\
 Y' &= 0.333Y + 200; \\
 X' &= 0.167X + 0.289Y + 130; \\
 Y' &= -0.289X + 0.167Y + 256; \\
 X' &= 0.167X - 0.289Y + 403; \\
 Y' &= 0.289X + 0.167Y + 71.
 \end{aligned}
 \tag{1.5}$$

Результат застосування цього афінного колажу після десятої ітерації можна побачити на рис. 1.32.



Рис. 1.32. Зображення кривої Коха, побудованої за допомогою IFS

Рандомізована система ітераційних функцій (RIFS) є іншим методом побудови фрактальних зображень. У порівнянні з детермінованою системою ітераційних функцій (DIFS), RIFS використовує випадковий вибір ітераційних функцій для генерації фракталу. Це призводить до випадкових та унікальних фрактальних структур.

Принцип роботи RIFS включає наступні кроки:

1. Вибір базових об'єктів: початкові об'єкти обираються як базові елементи для побудови фракталу. Це можуть бути прості геометричні форми, такі як точки, лінії або квадрати.
2. Визначення набору ітераційних функцій: для кожного базового об'єкта визначається набір ітераційних функцій. Цей набір складається з різних трансформацій, які можуть зміщувати, масштабувати, обертати або інакше змінювати базовий об'єкт.
3. Випадковий вибір ітераційних функцій: на кожній ітерації випадковим чином обирається одна з ітераційних функцій з набору. Ця випадковість призводить до різноманітності та непередбачуваності фрактальної структури.
4. Повторення ітераційного процесу: вибрана випадковим чином ітераційна функція застосовується до кожного базового об'єкта, що створює нові елементи або підрозділи. Цей процес повторюється

декілька разів з випадковим вибором ітераційних функцій на кожній ітерації.

5. Зупинка ітераційного процесу: ітераційний процес може бути зупинений після досягнення заданої кількості ітерацій або після досягнення бажаного рівня деталізації та вигляду фракталу.
6. Візуалізація фрактального зображення: останній крок полягає у візуалізації отриманого фрактального зображення, що може бути показане на екрані або збережене у форматі зображення.

Один із прикладів використання RIFS – побудова фрактальної структури трикутника Серпінського за допомогою гри «Хаос».

Основна ідея "гри" полягає в наступному: на площині фіксується правильний трикутник $A_1 A_2 A_3$. Вибирається довільна початкова точка B_0 . Потім на випадковій основі обирається одна з трьох вершин трикутника, і ми відзначаємо точку B_1 - середину відрізка між цією вершиною та B_0 (на малюнку нижче ми випадково вибрали вершину A_1). Цей процес повторюється з точкою B_1 , щоб отримати B_2 . Потім ми отримуємо точки B_3, B_4 і так далі (Рис. 1.33). Важливо, щоб точка "стрибала" випадковим чином, тобто кожен раз вершина трикутника обирається випадковим чином, незалежно від попередньо обраних точок. Цікаво, що якщо ми відзначимо точки з послідовності B_i , незабаром на площині з'явиться трикутник Серпінського. На наведених нижче рисунках показано результат після відзначення 1000, 10000 та 100000 точок (Рис 1.34).

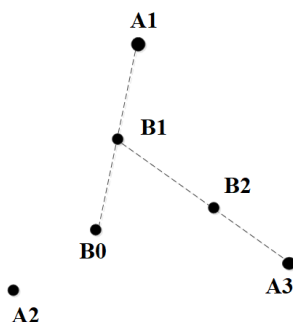


Рис. 1.33. Побудова фрактальної структури трикутник Серпінського за допомогою гри «Хаос»

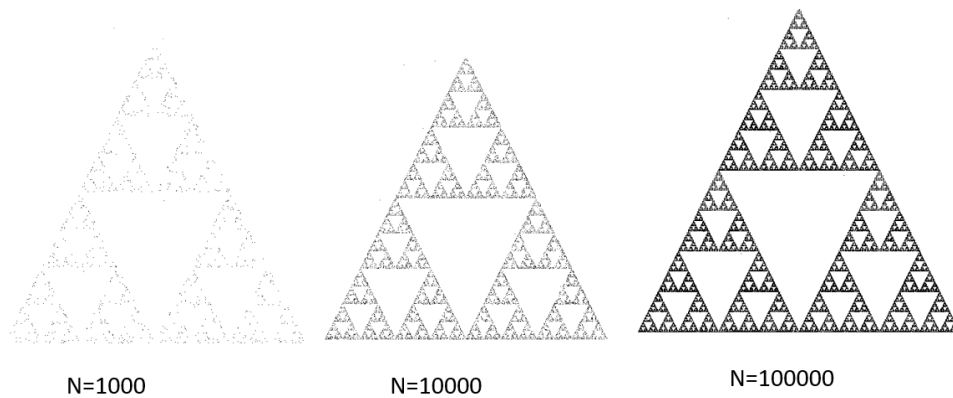


Рис. 1.34. Побудова фрактальної структури трикутника Серпінського за допомогою гри «Хаос»

Даний процес описується рандомізованою системою ітераційних функцій:

$$\begin{aligned}
 T_1\left(\begin{bmatrix} X \\ Y \end{bmatrix}\right) &= \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}, p_1 = 1/3 \\
 T_2\left(\begin{bmatrix} X \\ Y \end{bmatrix}\right) &= \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}, p_2 = 1/3 \\
 T_3\left(\begin{bmatrix} X \\ Y \end{bmatrix}\right) &= \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} 0.25 \\ \sqrt{3}/4 \end{bmatrix}, p_3 = 1/3
 \end{aligned} \tag{1.6}$$

Принцип рандомізованої системи ітераційних функцій (RIFS) надає можливість створювати унікальні та непередбачувані фрактальні структури. Випадковий вибір ітераційних функцій додає елемент несподіваності та креативності до процесу генерації фрактальних зображень [3].

Рекурентні співвідношення. Принцип роботи рекурентних співвідношень для побудови фрактальних зображень полягає в застосуванні ітеративного процесу до кожної точки простору для визначення її поведінки в межах певного математичного співвідношення. Цей процес повторюється для кожної точки, що веде до формування фрактальної структури.

Основні кроки використання рекурентних співвідношень для побудови фрактальних зображень:

1. Визначення математичного співвідношення: обирається математичне співвідношення, яке описує залежність між поточною точкою і

наступною точкою в ітераційному процесі. Це може бути складна формула або рекурентний вираз, що використовує попередні значення для обчислення нових значень.

2. Визначення початкових умов: встановлюються початкові значення для точок простору, з яких починається ітеративний процес. Це можуть бути координати на площині або векторні значення.
3. Ітеративний процес: для кожної точки виконується ітеративний процес, в якому застосовується математичне співвідношення для обчислення нових значень. Отримані значення використовуються як вхідні дані для наступної ітерації. Цей процес повторюється протягом заданої кількості ітерацій або досягнення певної умови зупинки.
4. Візуалізація фрактального зображення: отримані значення використовуються для візуалізації фрактального зображення. Залежно від специфікацій задачі, значення можуть бути маповані на кольори, яскравості або інші візуальні атрибути для створення фрактальної структури.
5. Ключовою ідеєю рекурентних співвідношень є повторення певного математичного процесу для кожної точки, що призводить до деталізації та формування фрактальної структури. Різні співвідношення можуть давати різні типи фракталів з унікальними властивостями.

Фрактали в нелінійних динамічних системах стосуються систем, що описуються ітераціями многочлена або голоморфної функції комплексної змінної на площині.

Давайте розглянемо послідовність, яка виникає за наступним правилом: нехай $F(z)$ - многочлен, z_0 - комплексне число. Починаючи з z_0 , ми обчислюємо $z_1 = F(z_0)$, $z_2 = F(z_1) = F(F(z_0))$, $z_3 = F(z_2) = F(F(F(z_0)))$, і так далі [3].

Ми цікавимося поведінкою цієї послідовності, коли n наближається до нескінченності. Така послідовність може проявляти наступні види поведінки:

- збіжність до скінченної границі;
- розбіжність до нескінченності;
- циклічну поведінку;
- хаотичну поведінку, коли не проявляється жоден з вищезазначених типів.

Множини значень z_0 , для яких послідовність проявляє певний тип поведінки, а також множини точок біфуркації між різними типами поведінки, часто виявляють фрактальні властивості.

Точка біфуркації - це критичний стан системи, при якому вона стає нестійкою до збурень і виникає невизначеність: система може стати хаотичною або перейти на вищий рівень впорядкованості.

Множина Мандельброта складається з комплексних точок c , для яких функція $f_c(z) = z^2 + c$ не розбігається під час ітерації, починаючи з $z_0 = 0$. Іншими словами, це набір значень комплексних чисел c , для яких орбіта числа 0 під дією квадратичного полінома $z_n = z^2 + c$ залишається обмеженою. Це означає, що для будь-якого натурального числа $n > 0$, абсолютне значення z_n залишається обмеженим.

Орбіта точки - це послідовність значень, отриманих шляхом повторного застосування функції до даного початкового елемента. Наприклад, орбіта точки z під дією відображення $f: \mathbb{C} \rightarrow \mathbb{C}$ - це послідовність $z, f(z), f(f(z)), \dots$

Ми можемо розкрити вищезазначену послідовність для кожної точки c шляхом ітераційної формули [14].

Якщо представити значення z_n як комплексні координати (x, y) , тобто $z_n = x_n + iy_n$, то ми можемо використати наступні формули для обчислення координат точки в наступній ітерації:

$$x_{n+1} = x_n^2 - y_n^2 + x_0, \quad (1.7)$$

$$y_{n+1} = 2x_n y_n + y_0 \quad (1.8)$$

Хоча зазвичай під множиною Мандельброта мається на увазі множина, описана вище, будь-яка функція комплексної змінної може мати свою відповідну множину Мандельброта. Наприклад, замість квадратичної функції, можна використовувати кубічну функцію.

Випадкові процеси. Принцип роботи випадкових процесів для побудови фрактальних зображень полягає у використанні стохастичних або випадкових елементів для визначення форми та структури фракталу. Ці процеси використовують випадковість і незалежність для генерації деталей фракталу, що веде до утворення складних і непередбачуваних структур.

Основні кроки використання випадкових процесів для побудови фрактальних зображень:

1. Визначення початкових умов: встановлюються початкові умови або точки, з яких починається процес генерації фракталу. Це можуть бути координати або параметри, що визначають початкову точку фрактальної структури.
2. Випадковий вибір: застосовуються випадкові елементи, такі як випадкові числа або випадкові генератори, для вибору напрямку, масштабу, форми або інших властивостей фрактального елемента. Випадковість дозволяє отримати різноманітність і непередбачуваність у фрактальній структурі.
3. Ітераційний процес: повторюється ітеративний процес, в якому випадкові елементи використовуються для кожної наступної ітерації. Кожна ітерація може включати зміну позиції, розміру, форми або інших характеристик фрактального елемента з використанням випадкових факторів.
4. Акумуляція результату: отримані значення під час ітераційного процесу акумулюються, щоб створити фрактальне зображення. Це

може включати накладання кольорів, текстур або інших візуальних атрибутів, щоб підкреслити структуру та деталі фракталу.

Випадкові процеси дозволяють створювати фрактальні зображення зі складними і непередбачуваними структурами. Вони надають можливість експериментувати з випадковими факторами, що призводить до унікальних варіацій фракталів з кожним запуском алгоритму. Однак, через випадковість, результати можуть бути менш детермінованими та складними для контролю.

1.3 Переваги та недоліки існуючих методів побудови фрактальних структур

Детермінована система ітераційних функцій (DIFS). Детермінована система ітераційних функцій (DIFS) має свої переваги і недоліки, які варто враховувати.

Переваги DIFS:

1. Гнучкість: DIFS дозволяє створювати різноманітні фрактальні зображення шляхом зміни ітераційних функцій та параметрів. Це дає великий простір для творчості та експериментів.
2. Деталізація: за допомогою повторюваних ітерацій DIFS дозволяє досягти високого рівня деталізації фрактальних зображень. Кожна ітерація додає нові деталі та складність до фракталу, створюючи захоплюючі візуальні ефекти.
3. Швидкість генерації: DIFS може бути ефективним з точки зору обчислень, оскільки базується на ітераційних функціях. Це дозволяє швидко створювати фрактальні зображення, особливо для простих фракталів.

Недоліки DIFS:

1. Великі обчислювальні вимоги: деякі складні фрактальні структури можуть вимагати значних обчислювальних ресурсів, особливо при

великій кількості ітерацій. Генерація деталізованих фрактальних зображень може зайняти багато часу.

2. Потребує експертизи: використання DIFS вимагає розуміння математичних принципів та функцій, що використовуються для генерації фрактальних зображень. Це може бути складно для новачків, які не мають досвіду з математикою або програмуванням. Є необхідність довгого та складного розрахунку коефіцієнтів ітераційних функцій.
3. Обмежена варіативність: в деяких випадках DIFS може бути обмеженою у створенні деяких типів фракталів або складних форм. Інші методи, такі як фрактальна геометрія або фрактальне моделювання, можуть бути більш підходящими для певних фрактальних структур.

Узагалі, DIFS є потужним інструментом для створення фрактальних зображень зі складними деталями та самоподібністю. Враховуючи його переваги і недоліки, важливо вибрати підхід, що найкраще відповідає конкретним потребам та обмеженням.

Рандомізована система ітераційних функцій (RIFS). Рандомізована система ітераційних функцій (RIFS) має свої переваги і недоліки, які варто враховувати.

Переваги RIFS:

1. Унікальність та різноманітність: використання випадкового вибору ітераційних функцій дозволяє створювати фрактальні зображення, які є унікальними та відрізняються від інших. Це надає більшу креативність та різноманітність в процесі генерації фракталів.
2. Швидкість генерації: RIFS може бути ефективним з точки зору обчислень, оскільки базується на ітераційних функціях. Це дозволяє швидко створювати фрактальні зображення.

3. **Непередбачуваність:** випадковий вибір ітераційних функцій додає елемент несподіваності та непередбачуваності до фрактальних зображень. Кожне виконання RIFS може призвести до нових та цікавих візуальних ефектів, що збагачує процес творчості.
4. **Ефективність:** RIFS може бути більш ефективною з точки зору обчислень у порівнянні з деякими іншими методами побудови фрактальних зображень. Випадковий вибір ітераційних функцій може дозволити швидше збільшення складності та деталізації фракталу.

Недоліки RIFS:

1. **Важкість управління:** випадковий вибір ітераційних функцій може зробити процес створення фракталів складним і важким у керуванні. Контроль за виглядом та характеристиками фракталу може бути викликом, оскільки рішення приймається на основі випадкового вибору.
2. **Обчислювальні вимоги:** в залежності від складності фракталу та кількості ітерацій, RIFS може вимагати значних обчислювальних ресурсів. Обробка великих фрактальних зображень або використання складних ітераційних функцій може зайняти багато часу та потребувати потужного обчислювального обладнання.
3. **Відсутність детермінізму:** RIFS використовує випадковий вибір, тому немає гарантії, що два запуски RIFS з однаковими параметрами будуть мати однакові результати. Це може бути недоліком, якщо ви хочете досягти повторюваності або точно визначити фрактальну структуру.

Враховуючи переваги та недоліки, RIFS є потужним інструментом для створення унікальних та цікавих фрактальних зображень. Однак, важливо врахувати вимоги до обчислювальних ресурсів, управління процесом та можливість неповторюваності при використанні цієї системи.

Рекурентні співвідношення. Рекурентні співвідношення для побудови фрактальних зображень мають свої переваги і недоліки.

Переваги:

1. Гнучкість: рекурентні співвідношення дозволяють створювати широкий спектр фракталів з різноманітними формами та деталізацією. Вони дають можливість експериментувати з математичними співвідношеннями та налаштуваннями, щоб отримати бажаний вигляд фракталу.
2. Квазісамоподібність: багато фракталів, побудованих з використанням рекурентних співвідношень, мають квазісамоподібні властивості. Це означає, що їх структура залишається подібною на різних масштабах, що додає їм естетичну привабливість та глибину.
3. Математична основа: використання рекурентних співвідношень забезпечує математичну основу для створення фракталів. Це дає можливість досліджувати та аналізувати їх математичні властивості, що має значення в наукових та дослідницьких доменах.

Недоліки:

1. Обчислювальні вимоги: побудова фракталів з використанням рекурентних співвідношень може бути обчислювально витратною задачею. Чим складніші та більш деталізовані співвідношення, тим більше часу і ресурсів потрібно для обчислення фрактального зображення.
2. Залежність від співвідношення: фрактальна структура, отримана за допомогою рекурентних співвідношень, залежить від вибраного математичного співвідношення. Неправильний вибір або недостатня налаштування можуть призвести до небажаних результатів.
3. Складність налаштування: налаштування параметрів рекурентних співвідношень може бути складним процесом, особливо для

складних фракталів. Вимагається розуміння математичних властивостей та їх впливу на фрактальну структуру.

Загалом, рекурентні співвідношення є потужним інструментом для побудови фрактальних зображень з унікальними властивостями. Проте, їх використання вимагає обчислювальних ресурсів, дослідницького підходу та належної настройки для досягнення бажаного результату.

Випадкові процеси. Випадкові процеси для побудови фрактальних зображень мають свої переваги і недоліки.

Переваги:

1. **Непередбачуваність:** випадкові процеси дозволяють створювати фрактальні зображення з унікальними та непередбачуваними структурами. Кожне запускання процесу може приводити до нових і цікавих варіацій фракталу.
2. **Більш природній вигляд:** випадкові процеси можуть допомогти створити фрактальні зображення, які наближаються до природних форм і структур. Наприклад, фрактальні ландшафти можуть мати схожість з рельєфом поверхностей у природі.
3. **Велика різноманітність:** випадковість дозволяє отримати широкий спектр різних фрактальних структур. Ви можете експериментувати з різними параметрами та випадковими елементами, щоб отримати різноманітність форм, деталей і текстур у фрактальному зображенні.

Недоліки:

1. **Втрата контролю:** випадковість може унеможливити повний контроль над формою та властивостями фракталу. Результати можуть бути менш передбачуваними, що може бути проблемою, коли ви маєте конкретне бачення або очікування від фрактального зображення.
2. **Обчислювальні вимоги:** використання випадковості може збільшити обчислювальні вимоги для генерації фрактальних зображень. Більш

складні та деталізовані фрактальні структури можуть вимагати більше часу та ресурсів для їх обчислення.

3. Вибір випадкових параметрів: вибір випадкових параметрів може бути викликом, особливо якщо вам потрібно досягти певних характеристик або ефектів у фрактальному зображенні. Вимагається дослідницький підхід та експериментування для досягнення бажаних результатів.

Загалом, випадкові процеси мають свої переваги та недоліки. Вони надають можливість створювати унікальні та природні фрактальні зображення, але можуть вимагати більше часу та зусиль для контролю та досягнення бажаного результату.

Переваги СІФ над рекурентними співвідношеннями. СІФ для побудови фрактальних структур має переваги, які роблять її кращою та ефективнішою в порівнянні з методами рекурентних співвідношень:

1. Простота реалізації: система ітераційних функцій зазвичай є більш простою для реалізації. Вона вимагає меншої кількості математичних обчислень та складних формул, що полегшує програмування та реалізацію фрактальних алгоритмів.
2. Гнучкість та контроль: за допомогою ітераційних функцій можна більш точно контролювати форму та деталі фракталу. Можливість налаштування параметрів, які впливають на розмір, положення, масштаб та інші характеристики фракталу, дає більший рівень гнучкості і творчого контролю.
3. Обчислювальна ефективність: ітераційні функції можуть бути обчислювально ефективними, особливо при використанні оптимізованих алгоритмів та структур даних. Це особливо важливо при генерації складних та деталізованих фракталів, де обчислювальні вимоги можуть бути високими.

4. Можливість комбінування з іншими методами: систему ітераційних функцій можна легко поєднати з іншими методами і техніками для створення більш складних та унікальних фракталів. Наприклад, можна використовувати ітераційні функції разом з текстурами, освітленням, колірними алгоритмами та іншими ефектами для досягнення бажаного візуального враження.

Переваги СІФ над випадковими процесами. Система ітераційних функцій для побудови фрактальних структур має переваги над методом побудови випадковими процесами.

1. Контрольованість: в системі ітераційних функцій ви маєте більший контроль над виглядом та властивостями фракталу. Ви можете визначити точні формули та параметри, які впливають на структуру фракталу, що дозволяє досягти бажаного результату. За допомогою настройки параметрів ви можете контролювати деталі, масштаб, колірну палітру та інші аспекти фракталу.
2. Повторюваність: система ітераційних функцій забезпечує повторюваність результатів. Якщо ви використовуєте однакові формули та параметри, ви будете отримувати однакові або дуже схожі фрактальні зображення кожного разу, коли запускаєте алгоритм. Це дозволяє вам точно відтворювати та контролювати результати вашої роботи.
3. Обчислювальна ефективність: у порівнянні з методом побудови випадкових процесів, система ітераційних функцій може бути більш обчислювально ефективною. Ітераційні функції зазвичай використовують прості математичні операції, що дозволяє швидше обчислювати фрактальні зображення, особливо при великій кількості ітерацій.
4. Більш точний контроль деталей: за допомогою системи ітераційних функцій ви можете досягти більш точного контролю над деталями

фракталу. Ви можете змінювати параметри, щоб відобразити певні властивості або структури, які ви хочете включити у фрактал. Це особливо корисно, якщо ви працюєте над конкретним дизайном або хочете досягти певного візуального ефекту.

1.4 Застосування фрактальних структур в телекомунікаціях та радіотехніці та їх значення у наукових дослідженнях

Застосування фрактальних структур у телекомунікаціях та радіотехніці виявляється у багатьох аспектах, включаючи антенні системи, передачу даних, компресію сигналів та моделювання каналів зв'язку.

Фрактальні антени. Одним з головних застосувань фракталів у телекомунікаціях є розробка фрактальних антенних систем. Фрактальні антени мають складну геометрію, яка дозволяє їм мати багато рівнів самоподібності на різних масштабах. Це дозволяє досягти більш широкосмугового зондування, кращого покриття та кращої ефективності в порівнянні з традиційними антенами. Фрактальні антени також можуть бути більш компактними, легкими у виготовленні та мають низьку ціну виробництва. Вони використовуються у бездротових мережах, супутниковому зв'язку, радіофізиці, радарях та інших телекомунікаційних системах.

Фрактальна антена - це тип антени, у якій використовуються фрактальні або квазіфрактальні структури з метою збільшення ефективної довжини або периметру антени. Ці структури дозволяють використовувати матеріали, які можуть ефективно приймати або випромінювати електромагнітні хвилі в межах обмеженої площі поверхні або об'єму.

Американський інженер Натан Коен, проживаючи у центрі Бостона, вперше використав фрактальну геометрію у проектуванні антенних пристроїв. Заборона на зовнішні антени на будинках не зупинила його - він створив фігуру у формі кривої Коха, вирізану з алюмінієвої фольги, яку наклеїв на аркуш

паперу й підключив до приймача. Виявилось, що така антена працює не гірше за звичайну [96].

Заснувавши власну компанію, Коен розпочав масове виробництво антен з використанням фрактальної геометрії. Це був початок інтенсивного розвитку таких антен.

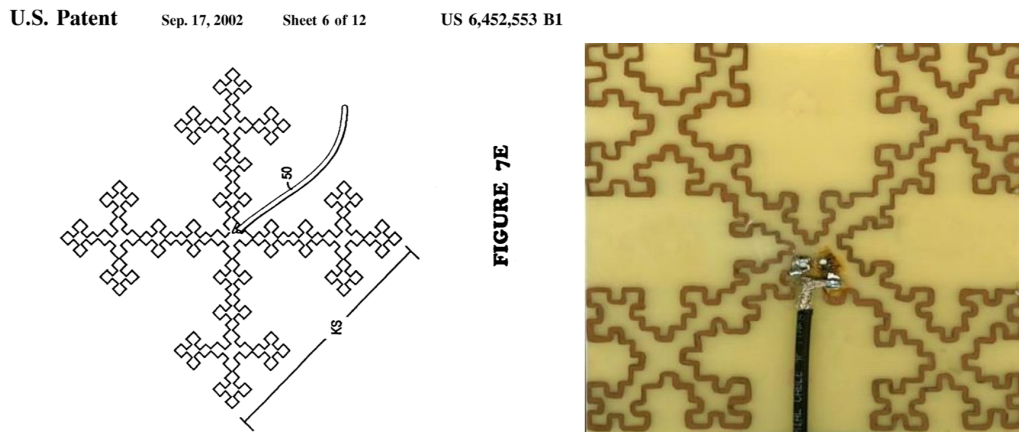


Рис. 1.35 Приклад фрактальної антени на основі кривої Мінковського

На Рис. 1.35 описано приклад фрактальної антени яка описана в патенті N. Cohen. U.S. Patent № 6140975A. H01Q 1/48. Fractal. Antenna Ground Counterpoise, Ground Planes and Loading. Elements. - Oct. 31, 2000 [97].

У 1999 році компанія Fractus, заснована Карлесом Пуенте Баліарда з університету Барселони, налагодила серійне виробництво фрактальних антен для мобільних телефонів у Барселоні.

Компанія Fractal Antenna Systems отримала патент на розроблену її спеціалістами технологію "плоского об'єктива". Заявляється, що ця технологія відкриває нові можливості для отримання зображень та керування потоком мікрохвиль, інфрачервоного та видимого випромінювання.

Розробка стала результатом понад двох десятиліть досліджень. Ключовим елементом є метаматеріал, створений за допомогою фрактальних структур, які виконують роль антен або резонаторів для електромагнітних хвиль. Фрактальна структура забезпечує роботу не на одній частоті, а в діапазоні хвиль.

Такий метаматеріал дозволяє керувати поширенням хвиль. Зокрема, можна створити "невидиму" поверхню. У 2012 році була продемонстрована можливість сховати людину від спостереження в мікрохвильовому діапазоні за допомогою нового матеріалу. Технологію "плоского об'єктива" показана на рис.1.36.

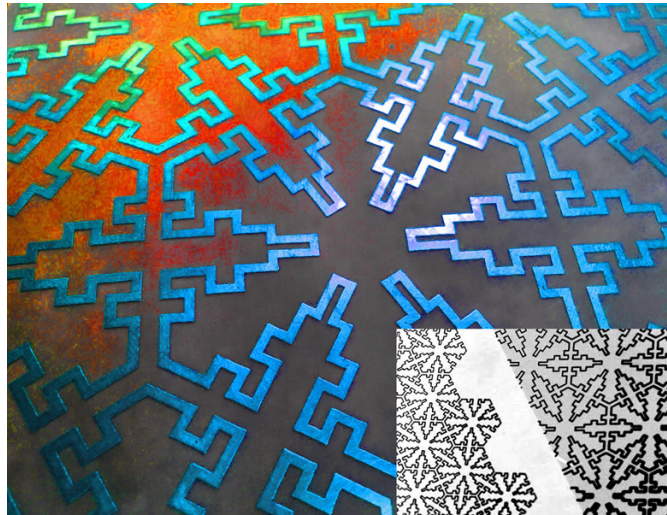


Рис. 1.36 Технологію "плоского об'єктива"

Ця розробка є розвитком досліджень у галузі невидимості. Фрактальні метаматеріали використовуються як множники та хвильоводи, дозволяючи отримати надтонкі лінзи та лінзи, що повторюють форму поверхні, а також лінзи, які працюють, будучи повернутими боком до джерела світла.

За словами Fractal Antenna Systems, новий патент є першим у серії патентів, пов'язаних з описаною технологією.

Технологія доступна для ліцензування. Хоча наразі вона працює лише в радіочастотному та мікрохвильовому діапазонах, перші виробники обладнання уже виявили свій інтерес до її використання.

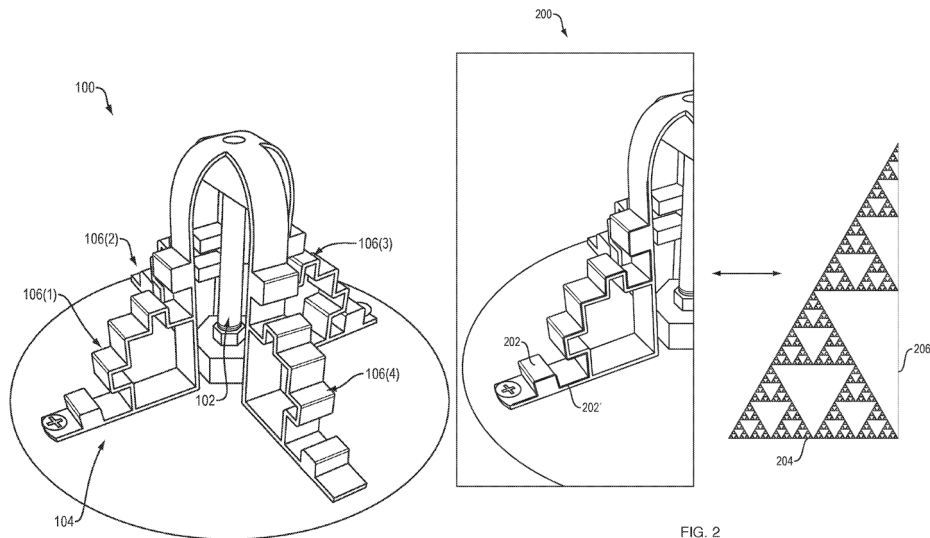


Рис 1.37 Фрактальні метаматеріальні каркасні антени

На Рис. 1.37 наведено приклад патенту США № 10 594 038 [номер заявки 15/528 397] було видано патентним відомством 2020-03-17 для фрактальних метаматеріальних антен у клітці. Наразі цей патент надано Fractal Antenna Systems, Inc.. Грантоотримувачем цього патенту є Fractal Antenna Systems, Inc.. Авторами винаходу є Натан Коен, Деніел Ерл, Джастін Мітчелл. Описуються каркасні антени та відповідні компоненти. Такі каркасні антени включають укорочений антенний елемент, такий як монополь (наприклад, висотою приблизно $1/8$ хвилі бажаної робочої довжини хвилі), який можна розмістити на вкороченій площині заземлення (наприклад, розміром приблизно в чверть хвилі). Схожий на клітку ансамбль (наприклад, клітка) може бути розміщений поверх елемента антени, але не торкаючись його. Кліткова структура може мати фрактальну, складену та/або гофровану структуру, серед іншого. Цю каркасну структуру можна виготовити за допомогою різноманітних засобів, включаючи, але не обмежуючись, 3-D друк з використанням провідних матеріалів або матеріалів з індуктивним кодуванням [98].

В статті [99] описують інноваційну фрактальну монопольну антену МІМО для сучасних систем зв'язку 5G (Рис 1.38).

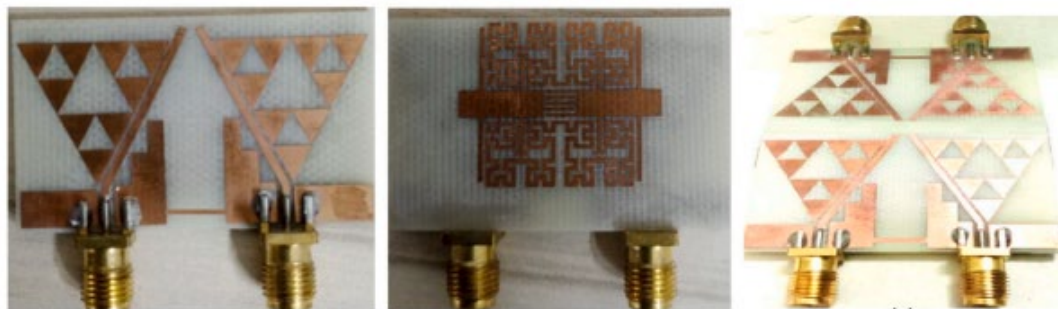
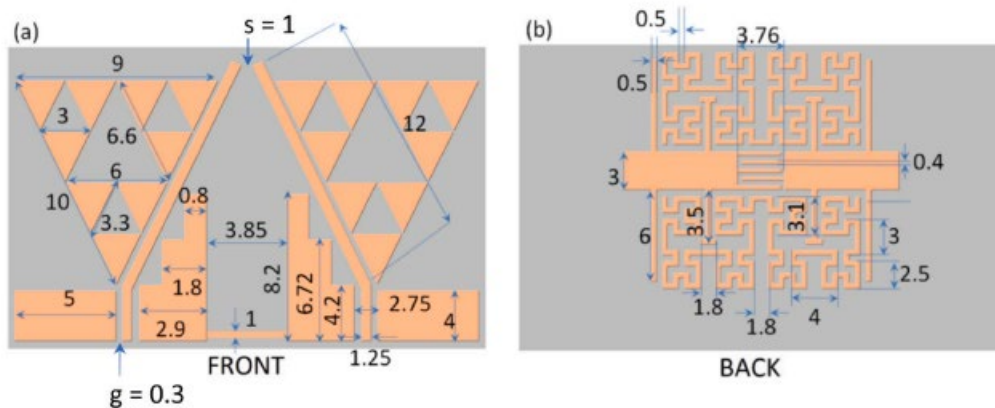


Рис. 1.38 Інноваційна фрактальна монопольна антена МІМО для сучасних додатків 5G

У цій роботі пропонується конструкція 2D антенної решітки для бездротових систем МІМО на частоті нижче 6 ГГц. Індивідуальні монопольні патч-антени в решітці засновані на фрактальній геометрії Серпінського. Мета використання фрактальних кривих полягала в тому, щоб отримати масив, який демонструє збільшення коефіцієнта підсилення та пропускну здатності. У решітку включено функцію фільтрації, яка забезпечується двовимірною композитною право-ліворонною структурою (CRLH), яка розташована безпосередньо під антенною решіткою. Структура CRLH заснована на фрактальній геометрії Гільберта та розроблена для пом'якшення поширення поверхневих хвиль, щоб зменшити взаємний зв'язок між елементами антени в решітці. Прототип конструкції антенної решітки випромінює енергію в діапазонах частот 2-3 ГГц, 3,4-3,9 ГГц і 4,4-5,2 ГГц. Антенна решітка має середнє посилення 5,5 дБі у вказаних діапазонах частот. Розмір загальної

антенної решітки $40 \times 30 \times 1,67$ мм підходить для мобільних безпроводних пристроїв [99].

Властивості фрактальних антен залежать від їх фрактальної розмірності. Фрактальна розмірність визначається структурою антени і впливає на її фізичні і електромагнітні характеристики. Ось як ці залежності можуть виявлятися:

- Ефективна довжина: фрактальні антени з вищою фрактальною розмірністю можуть мати більшу ефективну довжину. Це означає, що при заданих фізичних розмірах антени вона може мати більшу ефективну довжину порівняно зі звичайними антенами. Це дозволяє досягти кращого зв'язку та збільшити дальність передачі сигналу.
- Ширина смуги: фрактальні антени зазвичай виявляють більшу ширину смуги порівняно з традиційними антенами. Це означає, що вони можуть працювати на більш широкому діапазоні частот, що забезпечує більшу гнучкість в їх застосуванні.
- Діаграма напрямленості: фрактальні антени можуть мати складніші діаграми напрямленості, що означає, що вони можуть керувати розподілом енергії в просторі в більш складний спосіб. Це дозволяє покращити точність направлення сигналу та зменшити взаємний вплив між різними антенами в багатоантенних системах.
- Розмір та компактність: фрактальні антени можуть бути більш компактними порівняно зі звичайними антенами завдяки їх фрактальній структурі. Це забезпечує зручність в установці і застосуванні, особливо в обмежених просторових умовах.

У кінцевому підсумку, властивості фрактальних антен значно залежать від фрактальної розмірності, яка визначає їхні електромагнітні характеристики, ефективну довжину, ширину смуги, діаграму напрямленості та розмір. Ці властивості роблять фрактальні антени привабливими для застосування в телекомунікаціях та радіотехніці.

Компресії даних. Ще одним важливим застосуванням фракталів є їх використання у компресії даних. Фрактальне стиснення даних використовує самоподібність фракталів для ефективної передачі та збереження інформації. Замість передачі всіх пікселів або деталей зображення, фрактальне стиснення дозволяє передати лише початкове зображення та набір інструкцій для його відтворення на стороні отримувача. Це дозволяє економити пропускну здатність каналу та зберігати високу якість зображення при стисненні.

В статті [100] автори зазначили, що його застосування має свої обмеження. Одне з них - це тривалий процес стиснення, що не дозволяє його ефективно використовувати в додатках, де важливо миттєво передавати стиснені зображення, наприклад, під час прямої трансляції відео через мережу або відеоконференцій.

Крім того, через складність алгоритму та непередбачуваність результату, фрактальне стиснення зображень рідко використовується на практиці порівняно з іншими методами стиснення, такими як JPEG [101].

Загалом, фрактальне стиснення зображень має обмеження, що ускладнюють його широке застосування, але воно може мати свої використання в конкретних випадках, де важливі його унікальні властивості [101-104].

Розпізнавання фрактальних зображень за допомогою нейронних мереж та перетворення їх в систему ітераційних функцій надасть додаткові інструменти до стиснення зображень та відтворення їх.

Моделювання процесів в каналах зв'язку. Фрактали також знаходять застосування в моделюванні каналів зв'язку. Вони можуть бути використані для створення реалістичних моделей розповсюдження радіохвиль у складних середовищах, таких як міська зона або лісиста місцевість. Фрактальні моделі каналу дозволяють досліджувати вплив шумів, втрат сигналу та інших ефектів на якість зв'язку. Це допомагає вдосконалювати проектування та оптимізацію систем зв'язку.

На даний момент існує значна кількість наукових публікацій, що присвячені аналізу фрактальних властивостей трафіку. Виявлення самоподібних характеристик інформаційних потоків стало можливим у локальних та глобальних мережах, таких як Ethernet, ATM, TCP, IP, VoIP. Вчені, такі як K. Park, W. Willinger, P. Abry, MS Taqqu, I. Norros, Потапов А.А., Цибаков Б.С., Шелухін О.І. та інші, зробили значний внесок у розвиток теорії самоподібних процесів, дослідження фрактальних властивостей телетрафіку і створення моделей фрактального трафіку [105].

У останні десятиліття активно проводяться дослідження методів управління фрактальним трафіком з метою покращення якості обслуговування мережі. Це включає вибір та застосування методів і протоколів маршрутизації. Однак, незважаючи на зростаючу кількість робіт у цьому напрямку, є кілька невирішених питань. До них відносяться дослідження механізмів покращення якості обслуговування і методів управління трафіком у мультисервісних мережах, зокрема, у мережах MPLS, що працюють з самоподібним і мультифрактальним трафіком [105].

Отже, розробка методів управління інформаційними потоками та ресурсами мультисервісних комп'ютерних мереж, які враховують фрактальні властивості трафіку і забезпечують високий рівень якості обслуговування, є актуальною і важливою науково-технічною задачею [105].

На сьогоднішній день поняття фракталу виросло в нову математичну модель, котра дає єдиний опис властивостей, що притаманні багатьом природним явищам. Аналіз побудованих моделей є базою для дослідження реальних хвильових полів, різних за своєю природою [106-109].

Значення фрактальних структур у наукових роботах полягає в розширенні нашого розуміння телекомунікаційних та радіотехнічних систем. Вони дозволяють розвивати нові методи та технології, які покращують ефективність, якість та надійність зв'язку. Використання фрактальних структур може привести до зниження вартості обладнання та підвищення продуктивності

комунікаційних систем. Крім того, фрактальна геометрія надає новий підхід до моделювання та аналізу складних систем, допомагаючи розуміти їх поведінку та взаємодію з оточенням.

У підсумку, фрактальні структури відіграють важливу роль у телекомунікаціях та радіотехніці. Вони знайшли застосування у розробці антенних систем, стисненні та передачі даних, моделюванні каналів зв'язку та багатьох інших аспектах. Їх значення у наукових роботах полягає в пошуку нових інноваційних рішень, поліпшенні ефективності та якості зв'язку, а також у розширенні нашого розуміння та моделюванні телекомунікаційних систем.

РСІФ як фрактальна графіка та фрактальна розмірність може бути застосована у різних галузях, наприклад:

5. Фізика: використовується для опису складності геометричних структур, таких як гірські породи, реконструкції пульсарів, а також для вивчення фрактальної природи об'єктів в астрономії та космічних дослідженнях.
6. Біологія: може бути використана для аналізу складної геометрії біологічних структур, таких як судини, рослини, листя та корені. Вона також застосовується для вивчення складних систем, таких як розгалужені системи кровоносних судин та легені.
7. Комп'ютерна графіка: використовується для генерації складних геометричних фігур у комп'ютерних програмах та відеоіграх.
8. Криптографія: може бути використана для захисту інформації в криптографії. Зокрема, можна використовувати фрактали для генерації випадкових ключів шифрування та створення складних шифрів.
9. Медична діагностика: може бути використана для оцінки складності структур в медичних зображеннях, наприклад, для діагностики хвороб легень, серця та мозку.

- 10.Матеріалознавство: може бути використана для аналізу структури матеріалів, таких як метали, кераміка та полімери. Вона може допомогти у покращенні властивостей матеріалів та розробці нових матеріалів зі збільшеною міцністю та еластичністю.
- 11.Екологія: може бути використана для аналізу складної структури екосистем та вивчення екологічних процесів, таких як поширення рослинних видів у природних умовах.
- 12.Інформаційні технології: застосовується в обробці зображень і сигналів для визначення ступеня складності структур в розподілених мережах та інше.

У зв'язку з широким використанням систем ітераційних функцій в різних галузях науки та технології, та враховуючи, що традиційні методи створення та аналізу фрактальних структур не задовольняють поточним вимогам щодо швидкості, надійності та точності обробки даних, сьогодні важливим науково-практичним завданням є розробка нових підходів до створення фрактальних зображень з використанням рандомізованої системи ітераційних функцій (РСІФ), а також розвиток методів їх розпізнавання, захисту та шифрування з використанням нейронних мереж.

Висновки до розділу 1

В першому розділі розглянуто поняття фракталів, які вони мають основні властивості. Детально розглянуто основні види фракталів та способи їх створення, а саме геометричні фрактали (множина Кантора, крива Гільберта-Пеано, крива Коха, сніжинка Коха, "дракона" Хартера-Хейтуея, H-фрактал, Крива Мінковського, Крива Леві, квадрат Серпінського, трикутник Серпінського, T-фрактал, Дерево Піфагора), алгебраїчні фрактали (множини Жюліа, Множина Мандельброта) та стохастичні фрактали. Був проведений аналіз методів та алгоритмів побудови фрактальних структур, а саме ітеративний метод (системи ітераційних функцій), метод рекурентних

співвідношень, та метод випадкових процесів. Наведено переваги та недоліки існуючих методів побудови фрактальних структур, що дало можливість, в свою чергу, визначити, який з методів найкраще підходить для побудови фрактальних структур. Вибір зупинився на Ітеративному методі (Система ітераційних функцій СІФ), наведено переваги даного методу над методом рекурентних співвідношень та методом випадкових процесів. На даний момент існує дві відомі системи ітераційних функцій (СІФ): детермінована система ітераційних функцій (ДСІФ) та рандомізована система ітераційних функцій (РСІФ). Для порівняння цих двох методів був проведений математичний аналіз ефективності використання РСІФ над ДСІФ, що наочно показав перевагу першого методу над другим.

Для побудови рандомізована система ітераційних функцій (РСІФ), для заданого фрактального зображення необхідно визначити параметри ітераційних функцій, що є дуже складною математичною задачею, і потребує багато знань в області математики та інформатики, для кожного окремого випадку доводиться використовувати різні підходи для пошуку цих коефіцієнтів. Пошук коефіцієнтів не завжди можна визначити розрахунковим шляхом. Також потрібно визначити достатню кількість ітерацій руху точки, для того, щоб забезпечити потрібну якість зображення фракталу. РСІФ також допоможе вирішити одну з основних складних задач фрактального аналізу - це пошук фрактальної розмірності, який описує складність фрактала, тобто геометричної форми, що має нерегулярну структуру та складність. Основне застосування фрактальної розмірності полягає в тому, що вона дає змогу вимірювати складність форми та структури об'єктів, які не можна описати звичайними геометричними формулами. Узагальнюючи, РСІФ може бути використана для аналізу та моделювання складних систем у різних галузях, допомагаючи розуміти їх природу та побудову.

РОЗДІЛ 2. УДОСКОНАЛЕННЯ МЕТОДУ ПОБУДОВИ ФРАКТАЛЬНИХ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ РСІФ

2.1 Математичне представлення та моделювання фрактального зображення за допомогою РСІФ

Застосування РСІФ для створення фрактальних зображень є актуальним напрямком досліджень, оскільки ця технологія може мати великий потенціал у покращенні якості та ефективності обробки даних. У цьому розділі будуть представлені результати математичного моделювання для побудови фрактальних зображень з використанням модифікованої РСІФ.

Експеримент 1. Для першого експерименту розглянемо гру «Хаос» (яка описана в 1.2), для трикутника Серпінського (рис 1.35), зі системою ітераційних функцій (1.8), оскільки існуюче математичне представлення системи не дає змогу визначити необхідні параметри (фрактальну розмірність, розмір та кут повороту фігур першої ітерації) для аналізу та побудови фрактального зображення, то у роботі пропонується представити систему ітераційних функцій з врахуванням коефіцієнтів відстані між точками з координатами на котру має переміститися «блукаюча» точка. Такий підхід дасть змогу визначити залежність центру фігури першої ітерації фракталу від координати точки гри «Хаосу» для побудови та відтворення фрактального зображення за допомогою РСІФ. Відповідно формулу 1.8. предсталено у вигляді системи (2.1)

$$\begin{cases} x_n = x_A - \frac{x_A - x_{n-1}}{k} \\ y_n = y_A - \frac{y_A - y_{n-1}}{k} \end{cases}, p = P(A) \quad (2.1)$$

де x_A, y_A – координати точки, яка випала при рандомізованому алгоритмі розподілу; x_{n-1}, y_{n-1} і x_n, y_n – попередні і наступні координати точки, яка «блукає» та створює фрактальне зображення; k – коефіцієнт, який розділяє відстань між

точками з координатами (x_A, y_A) та (x_{n-1}, y_{n-1}) , на котру має підійти «блукаюча» точка; p - ймовірність вибору точки $A P(A)$.

Формули (2.1) за допомогою своїх параметрів спростить написання системи ітераційних функцій для побудови фрактальних зображень. Для прикладу формалізуємо систему для трикутника Серпінського (2.2):

$$\left[\begin{cases} \begin{cases} x_n = x_{A1} - \frac{x_{A1} - x_{n-1}}{2} \\ y_n = y_{A1} - \frac{y_{A1} - y_{n-1}}{2} \end{cases}, p = 1/3 \\ \begin{cases} x_n = x_{A2} - \frac{x_{A2} - x_{n-1}}{2} \\ y_n = y_{A2} - \frac{y_{A2} - y_{n-1}}{2} \end{cases}, p = 1/3 \\ \begin{cases} x_n = x_{A3} - \frac{x_{A3} - x_{n-1}}{2} \\ y_n = y_{A3} - \frac{y_{A3} - y_{n-1}}{2} \end{cases}, p = 1/3 \end{cases} \right. \quad (2.2)$$

Як видно з формули (2.2), коефіцієнт на котрий потрібно розділити відрізок $k = 2$, дана формула тепер є загальною для всіх трикутників Серпінського (прямокутного, гострокутного, тупокутного). Для експерименту візьмемо координатну площину 500px на 500px та введемо в формулу (2.2) координати точок $A1(500;0)$, $A2(0;250)$, $A3(500;500)$ (2.3):

$$\left[\begin{cases} \begin{cases} x_n = 500 - \frac{500 - x_{n-1}}{2} \\ y_n = 0 - \frac{0 - y_{n-1}}{2} \end{cases}, p = 1/3 \\ \begin{cases} x_n = 0 - \frac{0 - x_{n-1}}{2} \\ y_n = 250 - \frac{250 - y_{n-1}}{2} \end{cases}, p = 1/3 \\ \begin{cases} x_n = 500 - \frac{500 - x_{n-1}}{2} \\ y_n = 500 - \frac{500 - y_{n-1}}{2} \end{cases}, p = 1/3 \end{cases} \right. \quad (2.3)$$

Результати роботи програми, яка виконує (2.3) представлені на рис. 2.1 після «блукань» (кількості операцій) $N = 1000, 10000$ та 100000 .

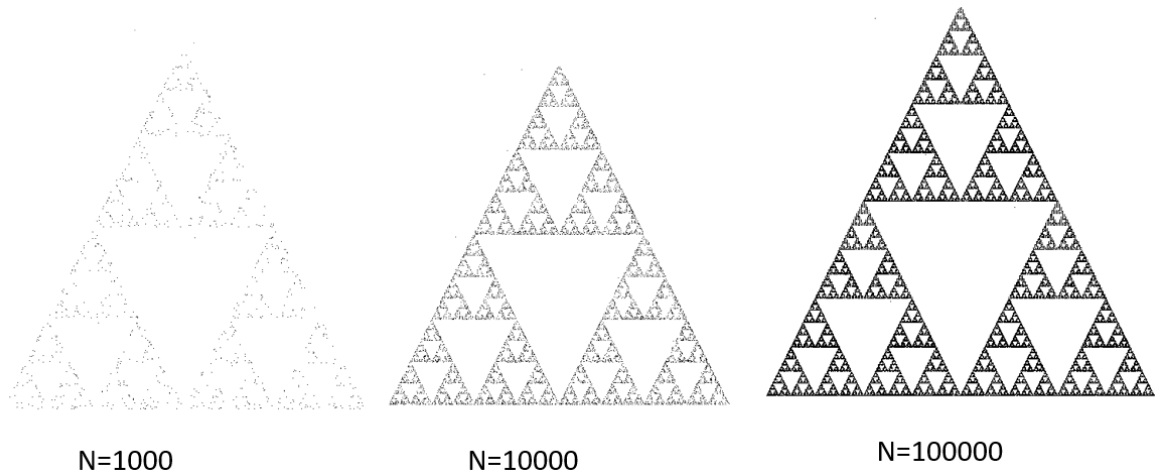


Рис. 2.1. Побудова фрактальної структури трикутник Серпінського за допомогою формули (2.3)

Експеримент 2. Для другого експерименту в грі «Хаос» збільшимо кількість точок до чотирьох і розставимо їх по вершинах чотирикутника, а «блукаюча» точка буде рухатись до точки, яка випала при рандомізованому алгоритмі розподілу, $2/3$ від їхньої довжини, тобто $k = 3$ рис 2.2:

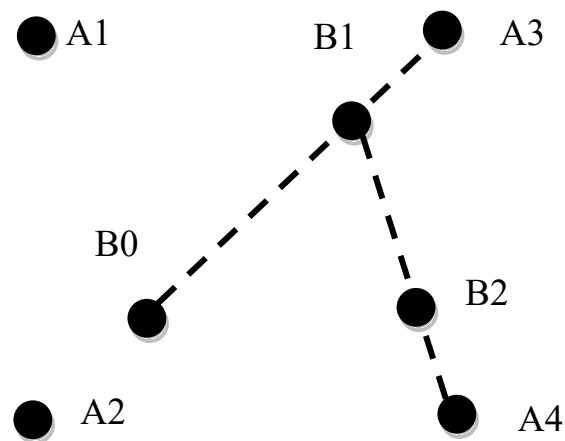


Рис. 2.2. Побудова фрактальної структури за допомогою гри «Хаос»

Відповідно загальна модифікована система ітераційних функцій представляється виразом (2.4):

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} x_n = x_{A1} - \frac{x_{A1} - x_{n-1}}{3} \\ y_n = y_{A1} - \frac{y_{A1} - y_{n-1}}{3} \end{array} \right. , p = 1/4 \\ \left\{ \begin{array}{l} x_n = x_{A2} - \frac{x_{A2} - x_{n-1}}{3} \\ y_n = y_{A2} - \frac{y_{A2} - y_{n-1}}{3} \end{array} \right. , p = 1/4 \\ \left\{ \begin{array}{l} x_n = x_{A3} - \frac{x_{A3} - x_{n-1}}{3} \\ y_n = y_{A3} - \frac{y_{A3} - y_{n-1}}{3} \end{array} \right. , p = 1/4 \\ \left\{ \begin{array}{l} x_n = x_{A4} - \frac{x_{A4} - x_{n-1}}{3} \\ y_n = y_{A4} - \frac{y_{A4} - y_{n-1}}{3} \end{array} \right. , p = 1/4 \end{array} \right. \quad (2.4)$$

Дана формула тепер є загальною для всіх типів чотирикутників. Для експерименту візьмемо координатну площину 500px на 500px та введемо в формулу (2.4) координати точок A1(0;0), A2(0;500), A3(500;500) та , A4(500;0) (2.5):

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} x_n = 0 - \frac{0 - x_{n-1}}{3} \\ y_n = 0 - \frac{0 - y_{n-1}}{3} \end{array} \right. , p = 1/4 \\ \left\{ \begin{array}{l} x_n = 0 - \frac{0 - x_{n-1}}{3} \\ y_n = 500 - \frac{500 - y_{n-1}}{3} \end{array} \right. , p = 1/4 \\ \left\{ \begin{array}{l} x_n = 500 - \frac{500 - x_{n-1}}{3} \\ y_n = 500 - \frac{500 - y_{n-1}}{3} \end{array} \right. , p = 1/4 \\ \left\{ \begin{array}{l} x_n = 500 - \frac{500 - x_{n-1}}{3} \\ y_n = 0 - \frac{0 - y_{n-1}}{3} \end{array} \right. , p = 1/4 \end{array} \right. \quad (2.5)$$

Результати роботи, яка виконує обрахунок (2.5), представленні на рис. 2.3 після «блукань» (кількості операцій) $N = 10000$.

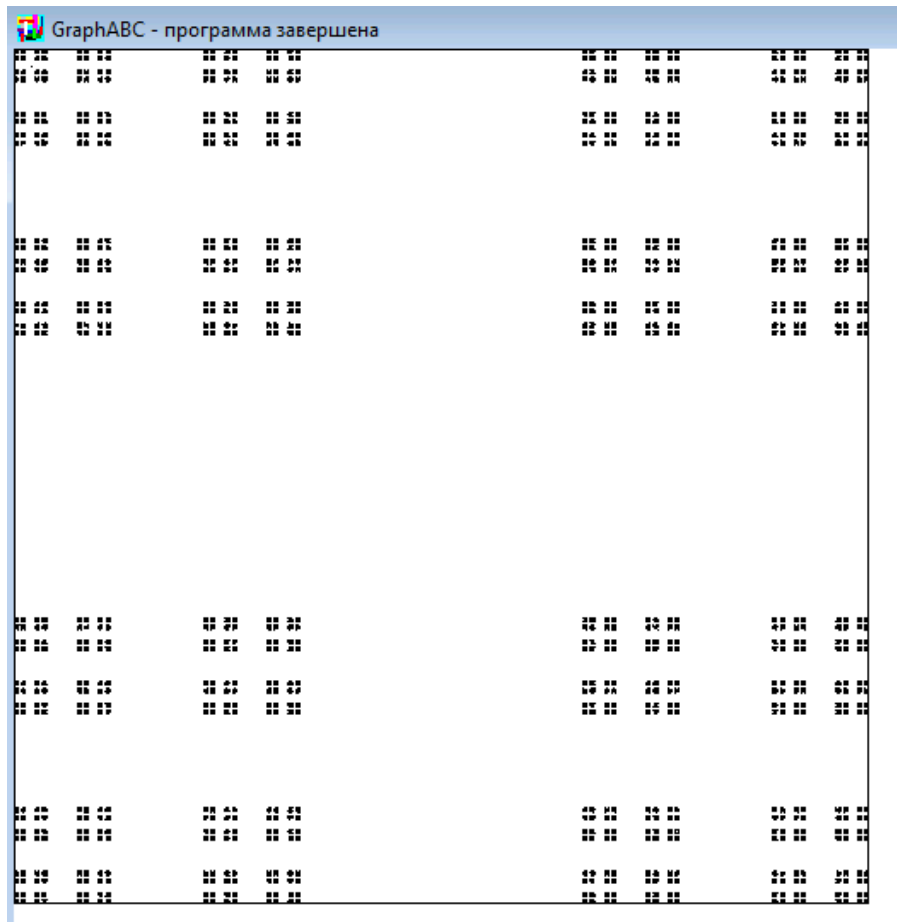


Рис. 2.3. Побудова фрактальної структури за допомогою формули (2.5)

Експеримент 3. Для третього експерименту в грі «Хаос» збільшимо кількість точок до п'яти і розставимо їх по вершинах п'ятикутника, а «блускаюча» точка буде рухатись до точки, яка випала при рандомізованому алгоритмі розподілу, $3/4$ від їхньої довжини, тобто $k = 4$, і давайте напишемо загальну модифіковану систему ітераційних функцій (2.6); дана формула тепер є загальною для всіх типів п'ятикутників. Для експерименту візьмемо координатну площину 500×500 та введемо в формулу (2.6) координати точок $A_1(0;250)$, $A_2(250;0)$, $A_3(500;250)$, $A_4(333;500)$ та $A_5(167;500)$ (2.7):

$$\left[\begin{array}{l}
\left\{ \begin{array}{l} x_n = x_{A1} - \frac{x_{A1} - x_{n-1}}{4} \\ y_n = y_{A1} - \frac{y_{A1} - y_{n-1}}{4} \end{array} \right. , p = 1/5 \\
\left\{ \begin{array}{l} x_n = x_{A2} - \frac{x_{A2} - x_{n-1}}{4} \\ y_n = y_{A2} - \frac{y_{A2} - y_{n-1}}{4} \end{array} \right. , p = 1/5 \\
\left\{ \begin{array}{l} x_n = x_{A3} - \frac{x_{A3} - x_{n-1}}{4} \\ y_n = y_{A3} - \frac{y_{A3} - y_{n-1}}{4} \end{array} \right. , p = 1/5 \\
\left\{ \begin{array}{l} x_n = x_{A4} - \frac{x_{A4} - x_{n-1}}{4} \\ y_n = y_{A4} - \frac{y_{A4} - y_{n-1}}{4} \end{array} \right. , p = 1/5 \\
\left\{ \begin{array}{l} x_n = x_{A5} - \frac{x_{A5} - x_{n-1}}{4} \\ y_n = y_{A5} - \frac{y_{A5} - y_{n-1}}{4} \end{array} \right. , p = 1/5
\end{array} \right. \quad (2.6)$$

$$\left[\begin{array}{l}
\left\{ \begin{array}{l} x_n = 0 - \frac{0 - x_{n-1}}{4} \\ y_n = 250 - \frac{250 - y_{n-1}}{4} \end{array} \right. , p = 1/5 \\
\left\{ \begin{array}{l} x_n = 250 - \frac{250 - x_{n-1}}{4} \\ y_n = 0 - \frac{0 - y_{n-1}}{4} \end{array} \right. , p = 1/5 \\
\left\{ \begin{array}{l} x_n = 500 - \frac{500 - x_{n-1}}{4} \\ y_n = 250 - \frac{250 - y_{n-1}}{4} \end{array} \right. , p = 1/5 \\
\left\{ \begin{array}{l} x_n = 333 - \frac{333 - x_{n-1}}{4} \\ y_n = 500 - \frac{500 - y_{n-1}}{4} \end{array} \right. , p = 1/5 \\
\left\{ \begin{array}{l} x_n = 167 - \frac{167 - x_{n-1}}{4} \\ y_n = 500 - \frac{500 - y_{n-1}}{4} \end{array} \right. , p = 1/5
\end{array} \right. \quad (2.7)$$

Результати обрахунку системи (2.7), представленні на рис. 2.4 після «блукань» (кількості операцій) $N = 10000$.

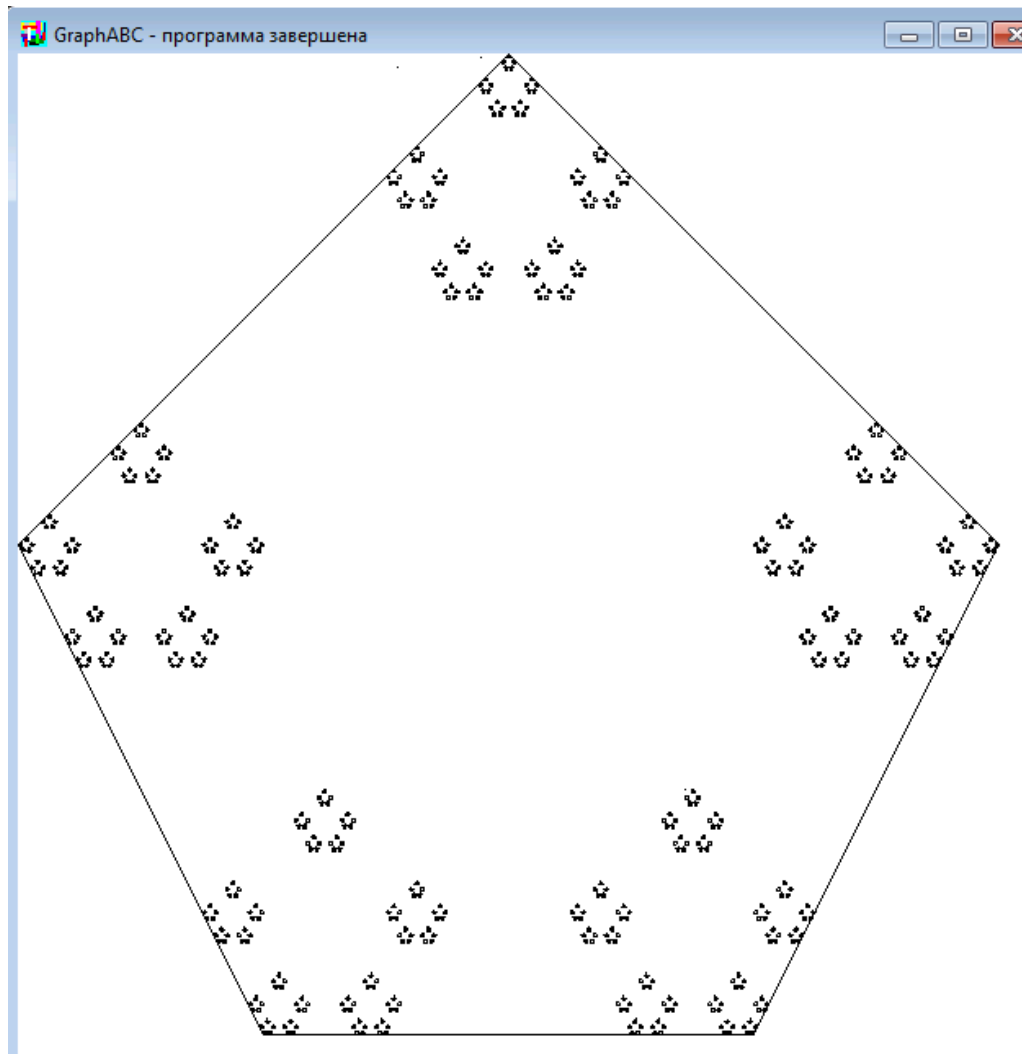


Рис. 2.4. Побудова фрактальної структури за результатами Експерименту 3

Експеримент 4. Для четвертого експерименту в грі «Хаос» збільшимо кількість точок до шести і розставимо їх по вершинах шестикутника, а «блукуюча» точка буде рухатись до точки, яка випала при рандомізованому алгоритмі розподілу, $4/5$ від їхньої довжини, тобто $k = 5$, і після написання загальної модифікованої системи ітераційних функцій та вибирання координатної площини 500×500 з введенням в систему координати точок ми отримаємо наступний результат рис. 2.5:

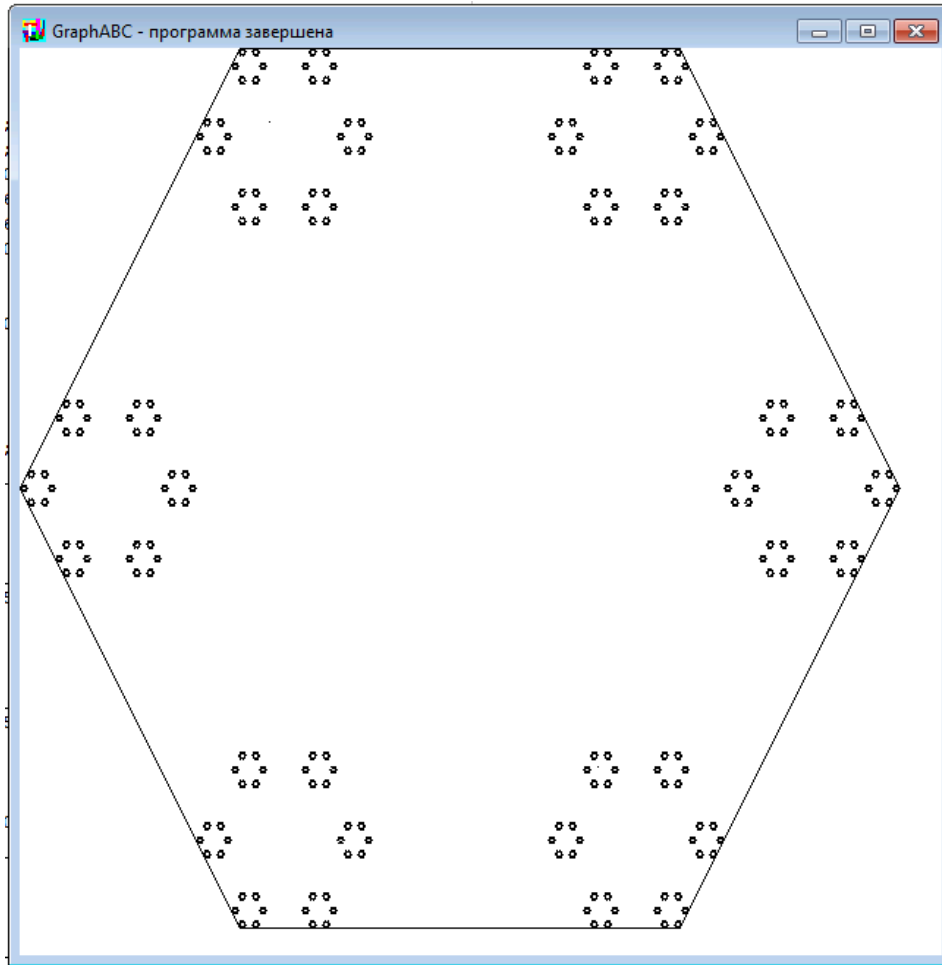


Рис. 2.5. Побудова фрактальної структури за результатами Експерименту 4

Експеримент 5. Дані для експерименту 5 візьмемо з експерименту 4 та змінимо параметр $k = 3$. Після написання загальної модифікованої системи ітераційних функцій та вибирання координатної площини 500px на 500px з введенням в систему координати точок ми отримаємо наступний результат рис. 2.6:

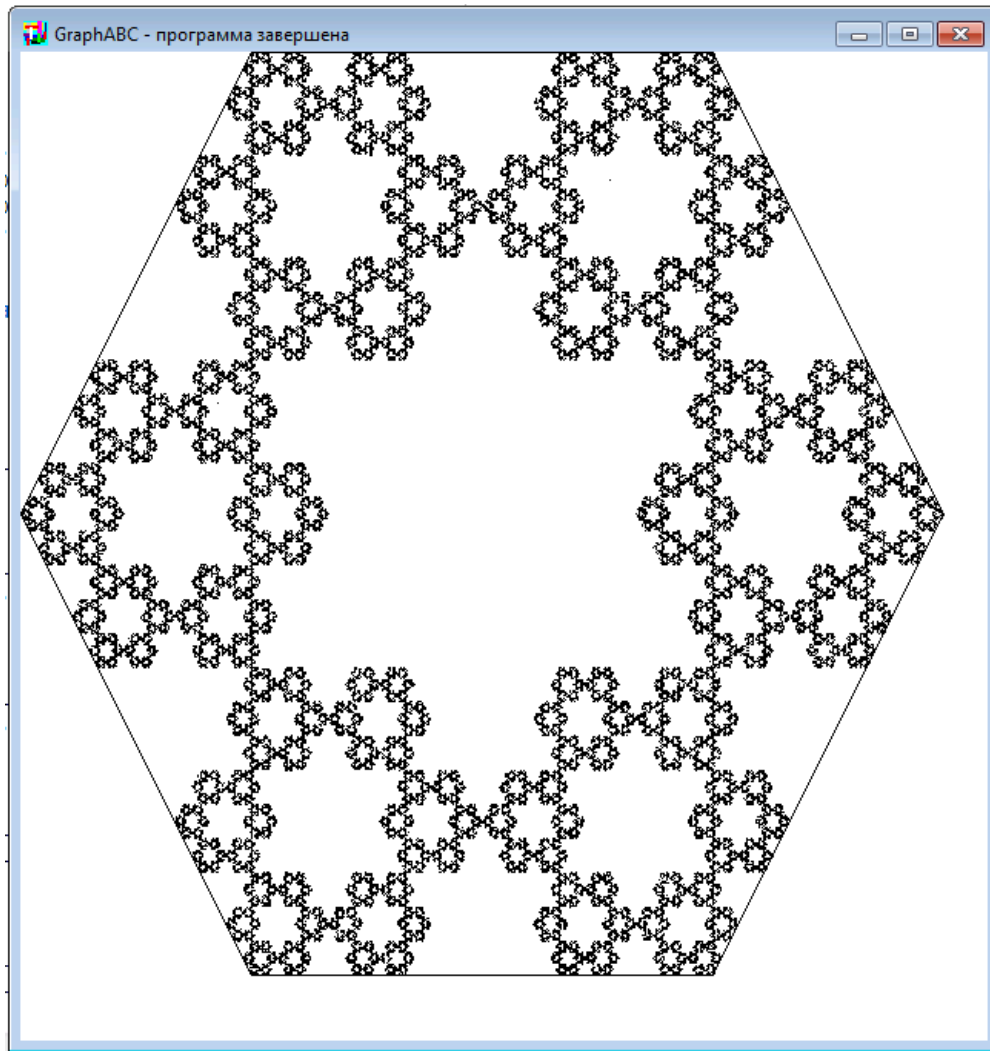


Рис. 2.6. Побудова сніжинки Коха за результатами Експерименту 5

Експеримент 6. Аналогічно можна побудувати Квадрат Серпінського, розставивши точки, як показано на рис. 2.7, та побудувавши з параметр $k = 3$ модифіковану систему ітераційних функцій, з вибиранням координатної площини 500×500 з введенням в систему координати точок ми отримаємо наступний результат рис. 2.8:

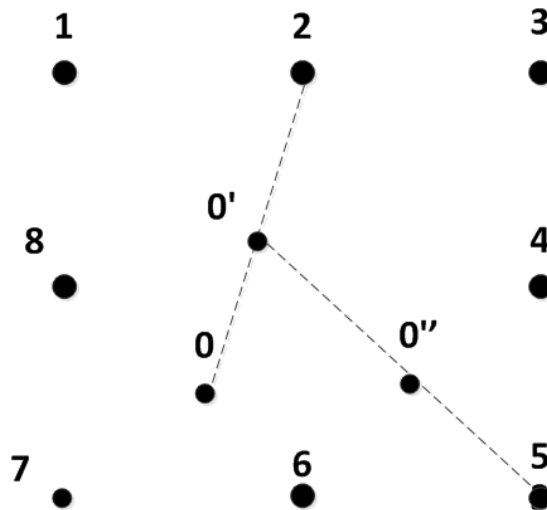


Рис. 2.7. Побудова фрактальної структури за допомогою гри «Хаос»

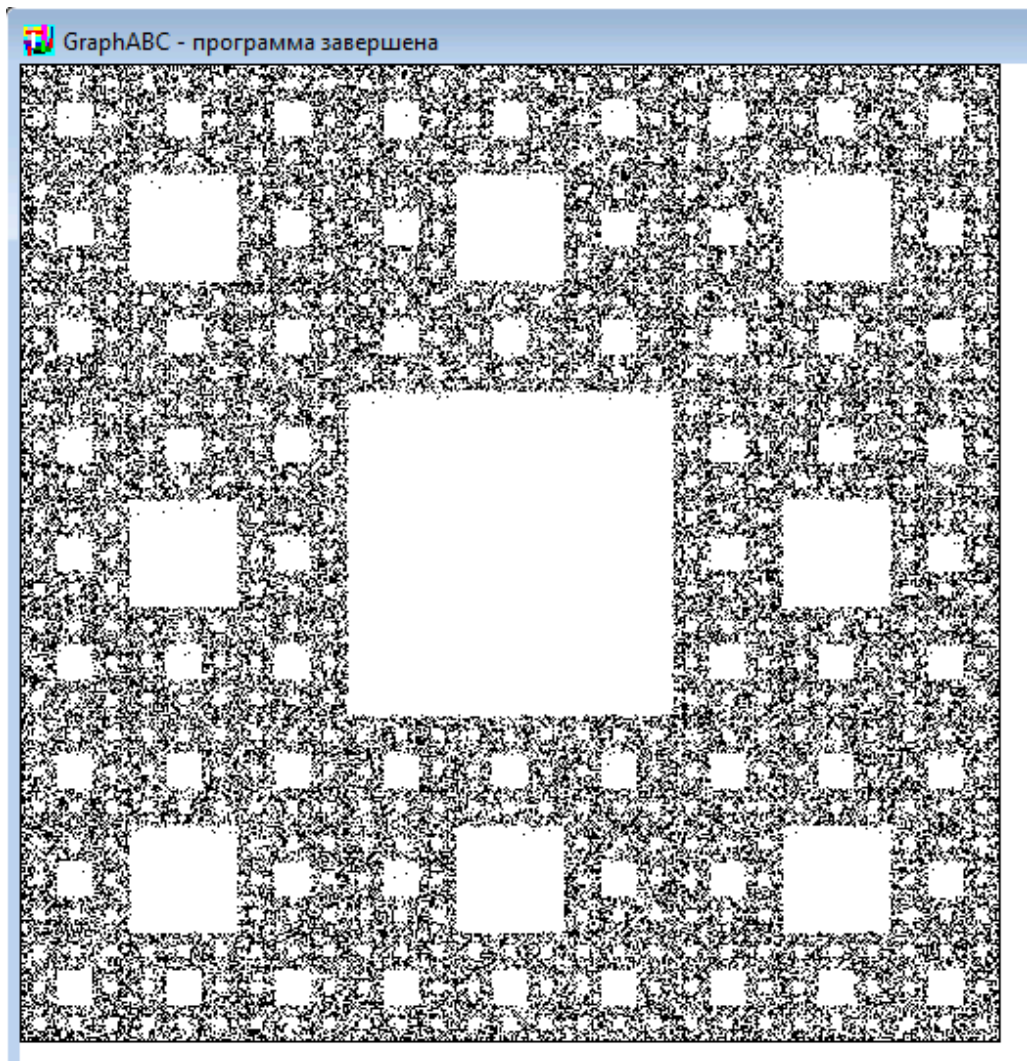


Рис. 2.8. Побудова квадрату Серпінського за результатами Експерименту 6

Після проведення шести експериментів вдалося виявити декілька особливостей:

- по-перше, запропоноване математичне представлення системи ітераційних функцій дозволяє наглядно зрозуміти гру «Хаос»;
- по-друге, координати точок до яких рухається «блукаюча» точка, не співпадають з центрами першої ітерації фрактального зображення, яке вони створюють, хоча розташування явно залежить від неї;
- по-третє, кількість фігур першої ітерації прямо пропорційна кількості ітераційних функцій;
- по-четверте, коефіцієнт пропорційності K (відношення довжини чи ширини фрактального зображення до довжини чи ширини фігури першої ітерації) повністю дорівнює параметру функції k (2.8) [110].

$$K = \frac{L}{l} = k \quad (2.8)$$

де K - коефіцієнт пропорційності; L – довжина або ширина фрактального зображення; l - довжина або ширина фрактального зображення (при умові що фрактальне зображення правильної форми, якщо ні, то цей параметр обраховується окремо для довжини і ширини).

Для того, щоб із зображення перейти в систему ітераційних функцій, тобто виконати зворотну задачу, потрібно зв'язати, координати центрів фігур першої ітерації з координатами точок, до яких рухається «блукаюча точка».

Експеримент 7. Побудуємо фрактал множини кантора використовуючи гру «Хаос» та запропоновану систему ітераційних функцій (2.8), а «блукаюча» точка буде рухатись до точки, яка випала при рандомізованому алгоритмі розподілу, $2/3$ від їхньої довжини, тобто $k = 3$. Для експерименту візьмемо координатну пряму Ox довжиною $500px$, координати точок $A_1(0)$, $A_2(500)$ (2.9) [110]:

$$\left[\begin{array}{l} x_n = x_{A1} - \frac{x_{A1} - x_{n-1}}{3}, p = 1/2 \\ x_n = x_{A2} - \frac{x_{A2} - x_{n-1}}{3}, p = 1/2 \end{array} \right. \quad (2.8)$$

$$\left[\begin{array}{l} x_n = 0 - \frac{0 - x_{n-1}}{3}, p = 1/2 \\ x_n = 500 - \frac{500 - x_{n-1}}{3}, p = 1/2 \end{array} \right. \quad (2.9)$$

Результати обчислення (2.9), представлені на рис. 2.9 після «блукань» (кількості операцій) N 10000 [110].

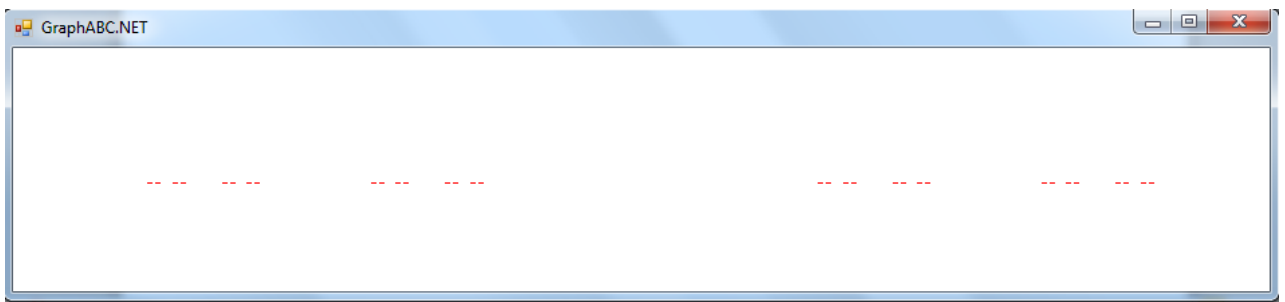


Рис. 2.9. Побудова множини Кантора за результатами Експерименту 7.

Експеримент 8. Проведемо ще експерименти відповідно до параметрів експерименту 7, змінюючи коефіцієнт k (Рис. 2.10) [110]:

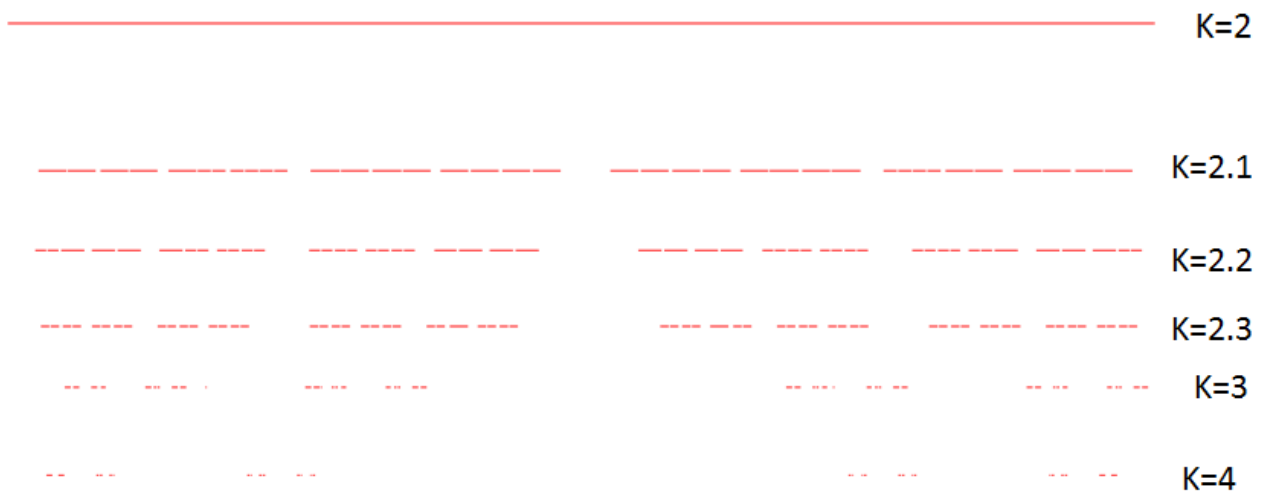


Рис. 2.10. Результатами Експерименту 8

Експеримент 9. Проведемо експеримент відповідно до параметрів експерименту 8 з умовою, якщо зустрічається цифра А1, то позначається червоним кольором, а у випадку цифри А2 – синім кольором (Рис. 2.11) [110]:

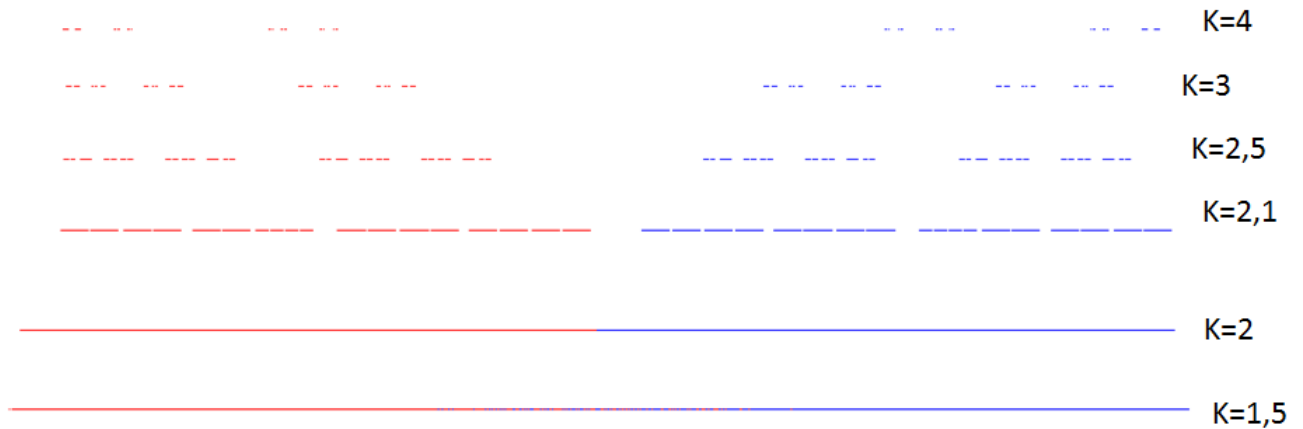


Рис. 2.11. Результатами Експерименту 9

Експеримент 10. Побудуємо фрактал множину Кантора, використовуючи гру «Хаос» та модифіковану систему ітераційних функцій (2.10), а «блукаюча» точка буде рухатись до точки, яка випала при рандомізованому алгоритмі розподілу, $3/4$ від їхньої довжини, тобто $k = 4$. Для експерименту візьмемо координатну пряму Ох довжиною 500рх, координати точок А1(0), А2(250), А2(500) (2.11):

$$\left[\begin{array}{l} x_n = x_{A1} - \frac{x_{A1} - x_{n-1}}{4}, p = 1/3 \\ x_n = x_{A2} - \frac{x_{A2} - x_{n-1}}{4}, p = 1/3 \\ x_n = x_{A3} - \frac{x_{A3} - x_{n-1}}{4}, p = 1/3 \end{array} \right. \quad (2.10)$$

$$\left[\begin{array}{l} x_n = 0 - \frac{0 - x_{n-1}}{4}, p = 1/3 \\ x_n = 250 - \frac{250 - x_{n-1}}{4}, p = 1/3 \\ x_n = 500 - \frac{500 - x_{n-1}}{4}, p = 1/3 \end{array} \right. \quad (2.11)$$

Результати обрахунку (2.11), представленні на рис. 2.12 після «блукань» (кількості операцій) $N = 10000$ [110].

K=4

Рис. 2.12. Побудова множини Кантора за результатами Експерименту 10

Експеримент 11. Побудуємо фрактал множини Кантора, використовуючи параметри експерименту 10, змінимо розташування точки A_2 та подивимося, як зміниться вигляд фракталу рис. 2.13 [110]:

Рис. 2.13. Побудова множини Кантора за результатами Експерименту 11

Провівши ряд математичних експериментів та розрахунків, доведено, що:

1) обрана за допомогою рандомізованого підходу точка утворює власний сегмент першої ітерації фрактала [110];

2) довжина першого відрізка ітерації не залежить від розташування точки.

Виходячи з цих двох результатів, можна зробити висновок про наявність зв'язку між координатою точки, яка вибирається за допомогою рандомізованого методу, і координатою середини першого сегмента ітерації, який вона утворює [110], що дасть змогу з фрактального зображення визначити координати точок гри «Хаосу» для представлення у вигляді РСІФ.

2.2 Удосконалення методу побудови фрактальних зображень з використанням РСІФ

Для того, щоб удосконалити метод побудови фрактальних зображень необхідно сформулювати математичну модель фрактального зображення, створеного за допомогою РСІФ. Для цього у роботі виведено зв'язок між координатою точки, яка вибирається за допомогою рандомізованого алгоритму,

і координатою середини першого сегмента ітерації, який вона утворює [110-111].

Розглянемо відрізок на координатній прямій Ox , який має довжину L і починається в точці 0 . Відрізок $X_{\Pi}X_{\kappa}$ відповідає першій ітерації і утворюється точкою з координатою X_2 , X_1 є серединою відрізка $X_{\Pi}X_{\kappa}$, а L/k є його довжиною рис. 2.14 [110-111]:

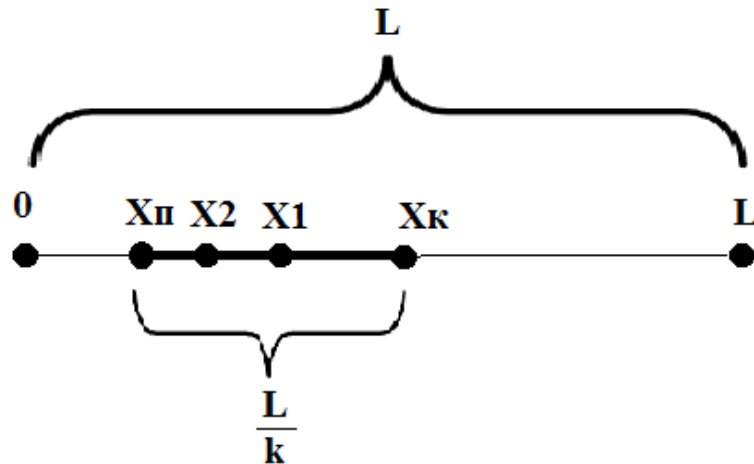


Рис. 2.14. Сегмент першої ітерації

Виходячи з наведеного рис. 2.14, можна зробити висновок про наступні закономірності [110-111]:

$$X_{\kappa} - X_{\Pi} = \frac{L}{k} \quad (2.12)$$

$$X_1 - X_{\Pi} = \frac{L}{2k} \Rightarrow X_{\Pi} = X_1 - \frac{L}{2k} \quad (2.13)$$

$$X_{\kappa} - X_1 = \frac{L}{2k} \Rightarrow X_{\kappa} = X_1 + \frac{L}{2k} \quad (2.14)$$

Оскільки точка X_2 повинна зберігати пропорційність відносно відрізка $0L$ і відрізка $X_{\Pi}X_{\kappa}$, то виникає така закономірність [110-111]:

$$\frac{X_2 - X_{\Pi}}{X_2} = \frac{X_{\kappa} - X_2}{L - X_2} \quad (2.15)$$

$$X_2(L - (X_{\kappa} - X_{\Pi})) = L \cdot X_{\Pi} \quad (2.16)$$

Підставляємо (2.12) і (2.13) в (2.16) і отримуємо:

$$X2 \left(L - \frac{L}{k} \right) = L \left(X1 - \frac{L}{2k} \right) \quad (2.17)$$

Таким чином, зв'язок між координатою X2, яку вибираємо за допомогою рандомізованого алгоритму, і координатою середини першої ітерації відрізка X1 [110-111]:

$$X2 = \frac{kX1 - \frac{L}{2}}{k-1} \quad (2.18)$$

Представимо математичну модель фракталу з урахування коефіцієнтів пропорційності по кожній фігурі першої ітерації (2.19) [110]:

$$\left[\begin{array}{l} \left\{ \begin{array}{l} x_n = x_{a_1} - \frac{x_{a_1} - x_{n-1}}{k_{X_{a_1}}} \\ y_n = y_{a_1} - \frac{y_{a_1} - y_{n-1}}{k_{Y_{a_1}}} \end{array} \right. \\ \dots \\ \left\{ \begin{array}{l} x_n = x_{a_j} - \frac{x_{a_j} - x_{n-1}}{k_{X_{a_j}}} \\ y_n = y_{a_j} - \frac{y_{a_j} - y_{n-1}}{k_{Y_{a_j}}} \end{array} \right. \end{array} \right. \quad (2.19)$$

Знаючи залежність між серединою першої ітерації та координатою точки, вибраною за допомогою рандомізованого алгоритму, можна записати загальну математичну модель фрактального зображення (2.20) з урахуванням закономірності (2.18) [112]:

$$\left\{ \begin{array}{l} x_n = x_{a_1} - \frac{x_{a_1} - x_{n-1}}{k_{X_{a_1}}} \\ y_n = y_{a_1} - \frac{y_{a_1} - y_{n-1}}{k_{Y_{a_1}}} \end{array} \right., p = P(a_1)$$

...

$$\left\{ \begin{array}{l} x_n = x_{a_j} - \frac{x_{a_j} - x_{n-1}}{k_{X_{a_j}}} \\ y_n = y_{a_j} - \frac{y_{a_j} - y_{n-1}}{k_{Y_{a_j}}} \end{array} \right., p = P(a_j)$$
(2.20)

де (x_n, y_n) - поточна координати точки, яка покроково будує фрактал; $\{(x_{a_1}, y_{a_1}), \dots, (x_{a_j}, y_{a_j})\}$, $j \in \mathbb{N}$ - координати точок, що утворюють перші ітерації фракталу; $\{k_{X_{a_1}}, k_{Y_{a_1}}, \dots, k_{X_{a_j}}, k_{Y_{a_j}}\}$, $j \in \mathbb{N}$ - коефіцієнти пропорційності відповідних фігур першої ітерації.

Для визначення координати точок що утворюють перші ітерації фракталу (x_{a_j}, y_{a_j}) , використаємо центри фігур першої ітерації X_{a_j}, Y_{a_j} та вдосконалимо формулу (2.18) в (2.21) [110]:

$$x_{a_j} = \frac{k_{X_{a_j}} X_{a_j} - \frac{L_x}{2}}{k_{X_{a_j}} - 1}; \quad y_{a_j} = \frac{k_{Y_{a_j}} Y_{a_j} - \frac{L_y}{2}}{k_{Y_{a_j}} - 1}$$
(2.21)

Запропонована система ітераційних функцій РСІФ (2.20) обмежена у здатності описати фрактал з поворотними елементами перших ітерацій. Для вирішення цієї проблеми розглянемо пару ітераційних функцій (2.22) [112]:

$$\begin{cases} x_n = x_{a_j} - \frac{x_{a_j} - x_{n-1}}{k_{Xa_j}} \\ y_n = y_{a_j} - \frac{y_{a_j} - y_{n-1}}{k_{Ya_j}} \end{cases}, \quad (2.22)$$

Перемістимо центр фігури першої ітерації паралельним перенесенням так, щоб він знаходився на початку системи координат, і представимо систему пари ітераційних функцій таким чином (2.23) [112]:

$$\begin{cases} x_n = x_{a_j} - \frac{x_{a_j} - x_{n-1}}{k_{Xa_j}} - X_{a_j} \\ y_n = y_{a_j} - \frac{y_{a_j} - y_{n-1}}{k_{Ya_j}} - Y_{a_j} \end{cases}, \quad (2.23)$$

де X_{a_j}, Y_{a_j} - центри фігур першої ітерації j фігури;

Тепер введемо афінні перетворення повороту на кут φ_j для формули (2.24) [112]:

$$\begin{cases} x_n = \left(x_{a_j} - \frac{x_{a_j} - x_{n-1}}{k_{Xa_j}} - X_{a_j} \right) \cdot \cos(\varphi_j) - \left(y_{a_j} - \frac{y_{a_j} - y_{n-1}}{k_{Ya_j}} - Y_{a_j} \right) \cdot \sin(\varphi_j) \\ y_n = \left(x_{a_j} - \frac{x_{a_j} - x_{n-1}}{k_{Xa_j}} - X_{a_j} \right) \cdot \sin(\varphi_j) + \left(y_{a_j} - \frac{y_{a_j} - y_{n-1}}{k_{Ya_j}} - Y_{a_j} \right) \cdot \cos(\varphi_j) \end{cases}, \quad (2.24)$$

У наступному кроці здійснимо паралельне перенесення центру фігури першої ітерації на його попереднє положення (2.25)[30]:

$$\begin{cases} x_n = \left(x_{a_j} - \frac{x_{a_j} - x_{n-1}}{k_{Xa_j}} - X_{a_j} \right) \cdot \cos(\varphi_j) - \left(y_{a_j} - \frac{y_{a_j} - y_{n-1}}{k_{Ya_j}} - Y_{a_j} \right) \cdot \sin(\varphi_j) + X_{a_j} \\ y_n = \left(x_{a_j} - \frac{x_{a_j} - x_{n-1}}{k_{Xa_j}} - X_{a_j} \right) \cdot \sin(\varphi_j) + \left(y_{a_j} - \frac{y_{a_j} - y_{n-1}}{k_{Ya_j}} - Y_{a_j} \right) \cdot \cos(\varphi_j) + Y_{a_j} \end{cases}, \quad (2.25)$$

Відповідно із формули (2.25), загальна математична модель фрактального зображення, створеного за допомогою системи ітераційних функцій, буде виглядати наступним чином (2.26) [112]:

$$\left[\begin{array}{l} \left\{ \begin{array}{l} x_n = \left(x_{a_1} - \frac{x_{a_1} - x_{n-1}}{k_{X_{a_1}}} - X_{a_1} \right) \cdot \cos(\varphi_1) - \left(y_{a_1} - \frac{y_{a_1} - y_{n-1}}{k_{Y_{a_1}}} - Y_{a_1} \right) \cdot \sin(\varphi_1) + X_{a_1} \\ y_n = \left(x_{a_1} - \frac{x_{a_1} - x_{n-1}}{k_{X_{a_1}}} - X_{a_1} \right) \cdot \sin(\varphi_1) + \left(y_{a_1} - \frac{y_{a_1} - y_{n-1}}{k_{Y_{a_1}}} - Y_{a_1} \right) \cdot \cos(\varphi_1) + Y_{a_1} \end{array} \right. \\ \dots \\ \left\{ \begin{array}{l} x = \left(x_{a_j} - \frac{x_{a_j} - x_{n-1}}{k_{X_{a_j}}} - X_{a_j} \right) \cdot \cos(\varphi_j) - \left(y_{a_j} - \frac{y_{a_j} - y_{n-1}}{k_{Y_{a_j}}} - Y_{a_j} \right) \cdot \sin(\varphi_j) + X_{a_j} \\ y_n = \left(x_{a_j} - \frac{x_{a_j} - x_{n-1}}{k_{X_{a_j}}} - X_{a_j} \right) \cdot \sin(\varphi_j) + \left(y_{a_j} - \frac{y_{a_j} - y_{n-1}}{k_{Y_{a_j}}} - Y_{a_j} \right) \cdot \cos(\varphi_j) + Y_{a_j} \end{array} \right. \end{array} \right. , p = P(a_1) , p = P(a_j) \quad , (2.26)$$

За допомогою пропонованої системи ітераційних функцій (2.26) побудуємо фрактал трикутника Серпінського і відобразимо його результат на рис.2.15 [112]:

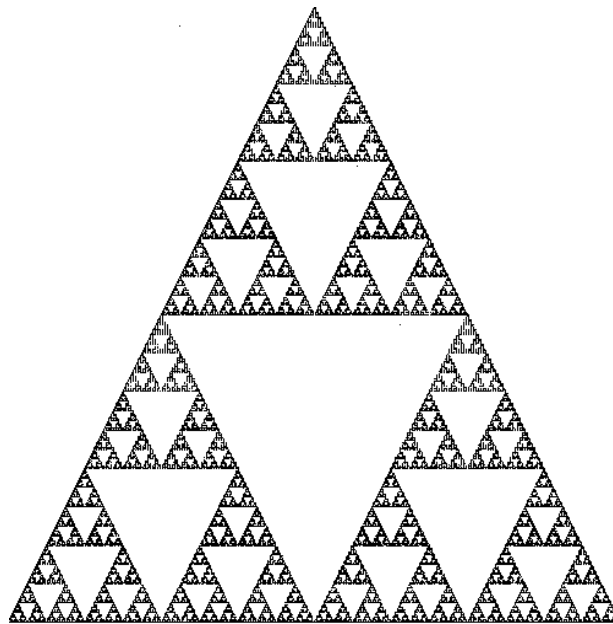


Рис. 2.15. Трикутник Серпінського, сформований за допомогою РСІФ (2.26)

Змінивши кут φ для фігури першої ітерації на 180 градусів для рис. 2.15, ми скористаємося вдосконаленою системою ітераційних функцій (2.26), щоб побудувати фрактал, і відобразимо його результат на рис.2.16 [112]:

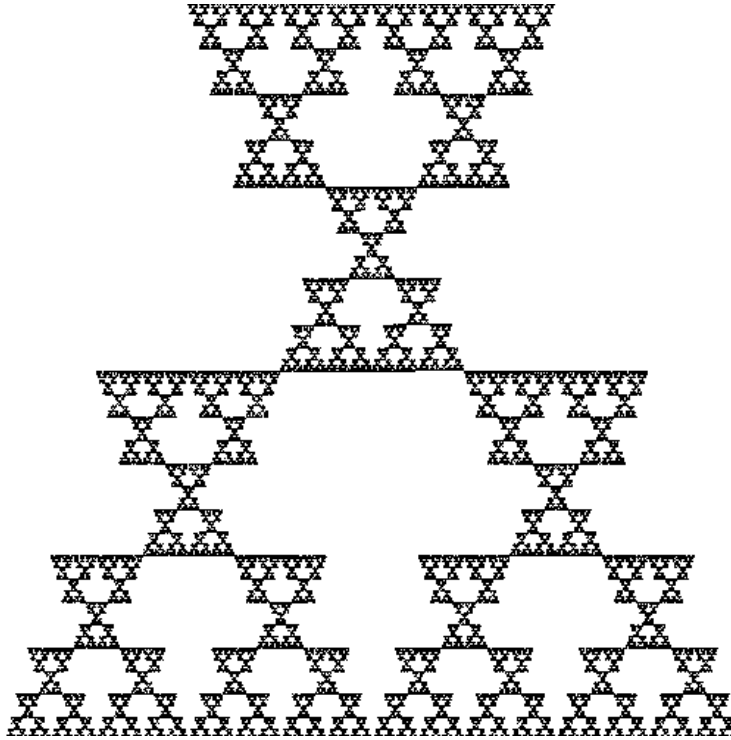


Рис. 2.16. Трикутник Серпінського створений за допомогою системи ітераційних функцій РСІФ (2.26) з поворотом фігури першої ітерації на 180 градусів

У цьому пункті опишемо процес запису РСІФ для побудови фракталів і виведення РСІФ з готових фракталів за формулою (2.26).

На прикладі спробуємо перетворити побудований фрактал в систему ітераційних функцій, за допомогою якої його можна буде потім відтворити.

Для наочного прикладу візьмемо фрактал правильної форми рис. 2.17 (фрактал «Усмішка») [110-111]

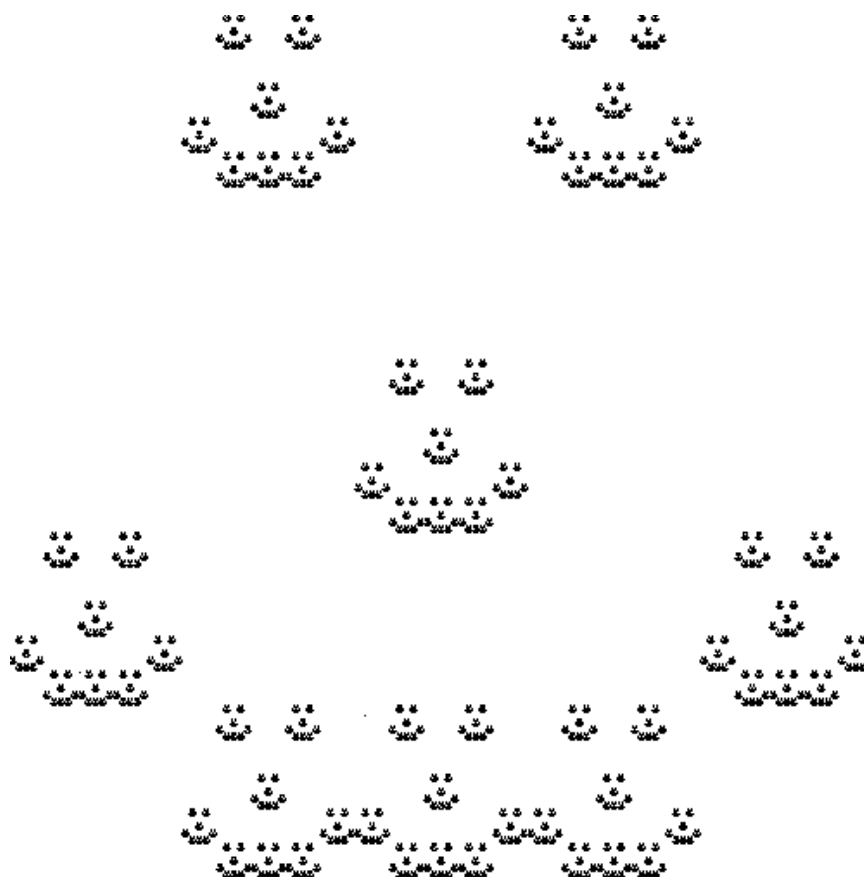


Рис. 2.17. Фрактал «Усмішка»

Крок 1. Фігури першої ітерації поміщаємо в рівні прямокутники (рис. 2.18)

Крок 2. Розраховуємо k згідно формули (2.8) $k = L / l = 500 / 100 = 5$.

Крок 3. Розраховуємо координати центрів фігур першої ітерації X_{a_j}, Y_{a_j}

Крок 4. Використовуючи формулу (2.21), визначаємо координати точки, які вибираються за допомогою рандомізованого алгоритму.

Крок 5. Складаємо удосконалену РСІФ згідно формули (2.26).

Крок 6. Програмуємо удосконалену РСІФ та виводимо результат.

Блок схема удосконаленого методу побудови фрактальних зображень з використанням РСІФ показано на рис. 2.19 [110-111].

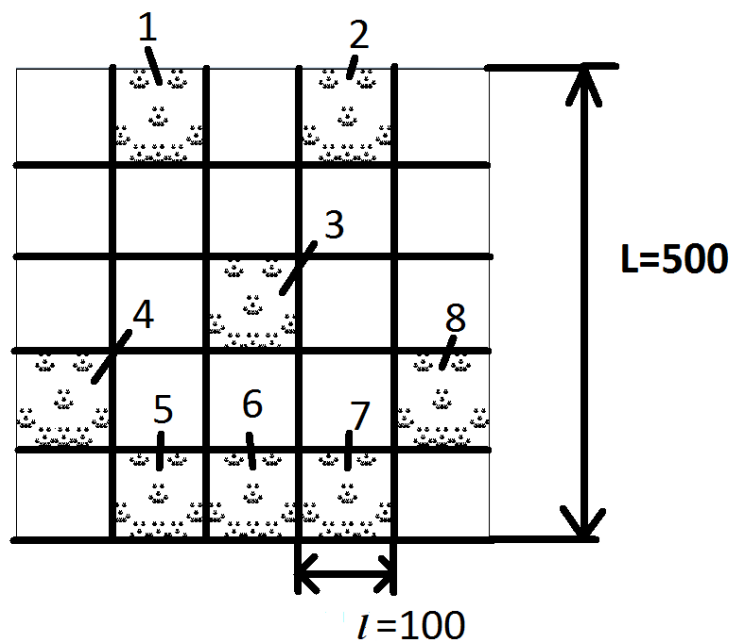


Рис. 2.18. Процес визначення центрів та розмірів фігур першої ітерації

Таблиця 2.1

Результати виконання кроків 2.1

Номер фігури першої ітерації	центри фігур першої ітерації	координати точки які вибирається за допомогою рандомізованого алгоритму	РСІФ
1	(150;0)	(125;0)	$x = 125 - (125 - x)/5$ $y = 0 - (0 - y)/5$
2	(350;0)	(375;0)	$x = 375 - (375 - x)/5$ $y = 0 - (0 - y)/5$
3	(250;250)	(250;250)	$x = 250 - (250 - x)/5$ $y = 250 - (250 - y)/5$
4	(50;350)	(0;375)	$x = 0 - (0 - x)/5$ $y = 375 - (375 - y)/5$
5	(150;500)	(125;500)	$x = 125 - (125 - x)/5$ $y = 500 - (500 - y)/5$
6	(250;500)	(250;500)	$x = 250 - (250 - x)/5$ $y = 500 - (500 - y)/5$
7	(350;500)	(375;500)	$x = 375 - (375 - x)/5$ $y = 500 - (500 - y)/5$
8	(450;350)	(500;375)	$x = 500 - (500 - x)/5$ $y = 375 - (375 - y)/5$

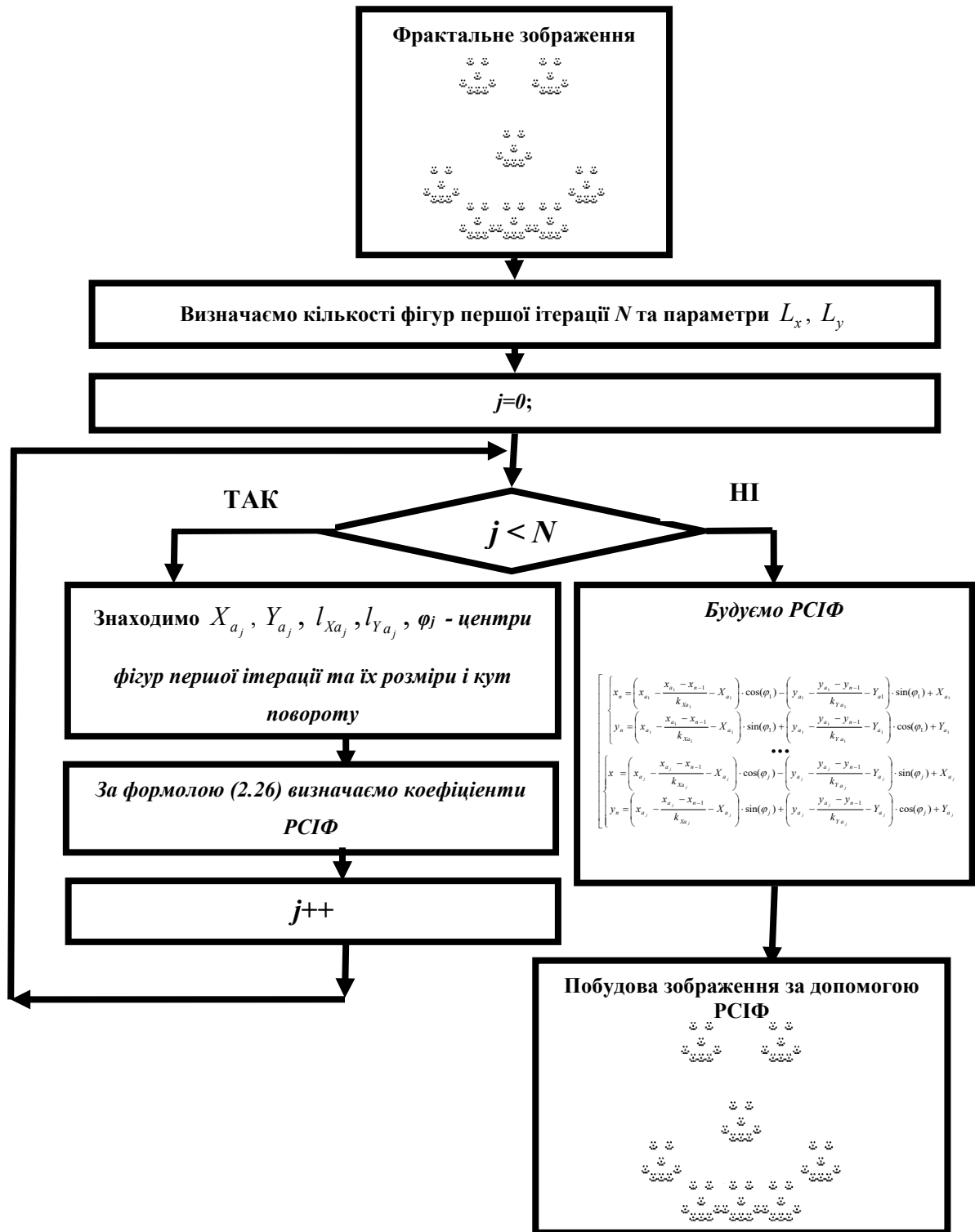


Рис. 2.19. Блок схема удосконаленого методу побудови фрактальних зображень з використанням модифікованої системи РСІФ

Подібним чином застосуємо метод побудови фрактальних зображень з використанням модифікованої системи РСІФ для перетворення зображення

(об'єкт) у фрактал. Для цього візьмемо, для прикладу, зображення у вигляді буки А рис. 2.20 [110-111]:

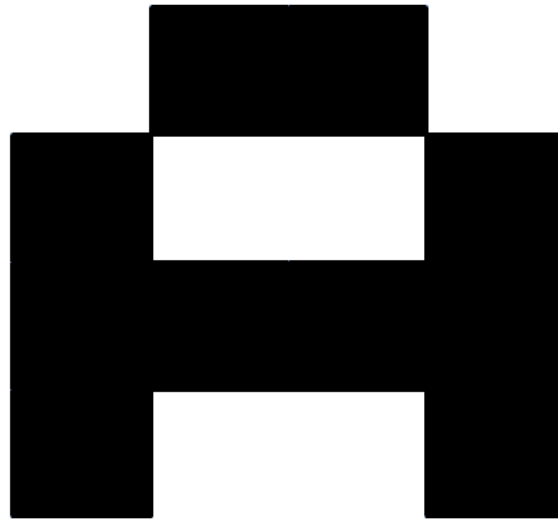


Рис. 2.20. Зображення для перетворення у фрактал

Виконуємо крок 1, нашого методу рис.2.22:

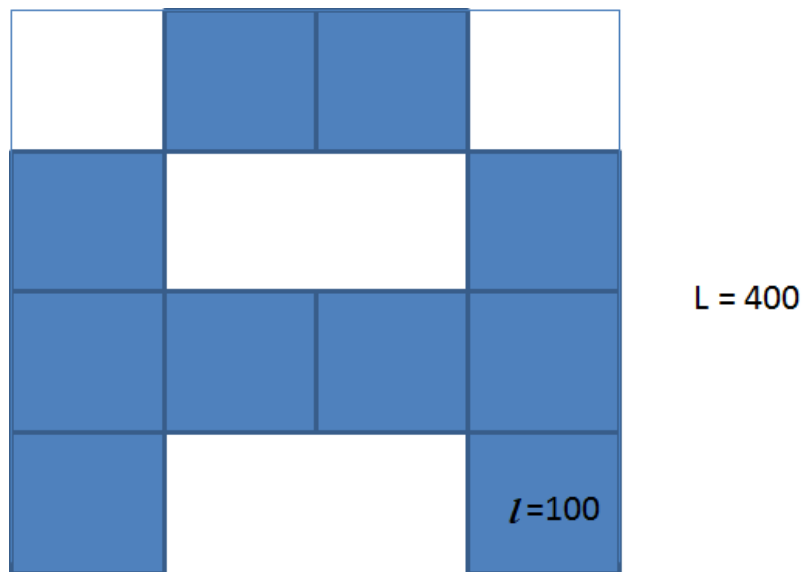


Рис. 2.21. Розбивання зображення на ітерації

Після виконання кроків 1-6 запропонованого методу побудови фрактальних зображень з використанням удосконаленої системи РСІФ, отримаємо наступний результат Рис. 2.22 [110-111]:

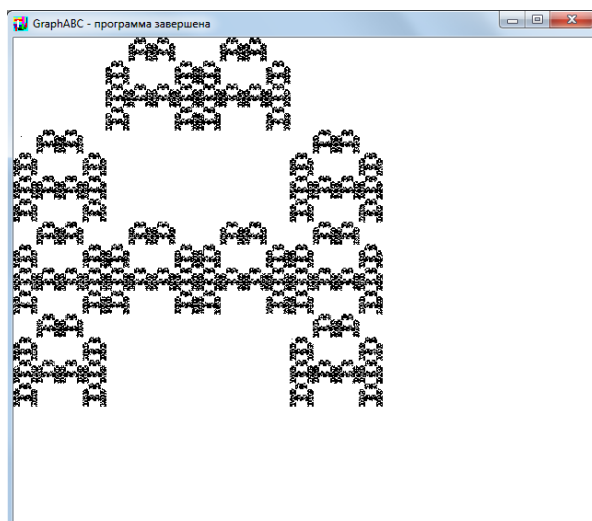


Рис. 2.22. Результат перетворення зображення за допомогою запропонованого методу побудови фрактальних зображень з використанням удосконаленої системи РСІФ

Приклади фрактальних зображень, побудованих за допомогою запропонованого методу побудови фрактальних зображень з використанням удосконаленої системи РСІФ, зображені на рис 2.23-2.26 [110-111]:

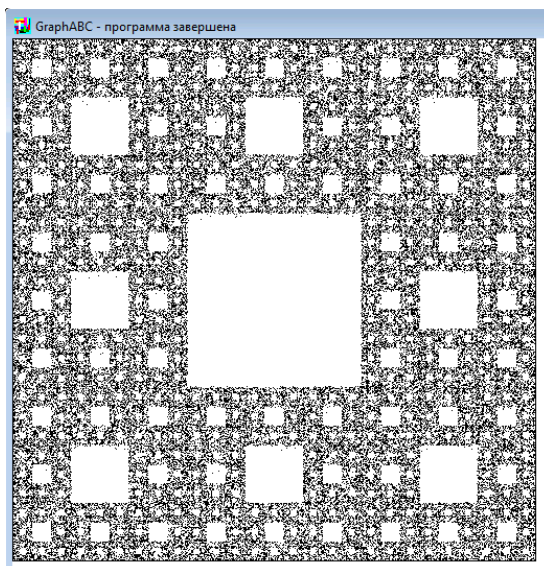


Рис. 2.23. Квадрат Серпінського (результат роботи алгоритму)

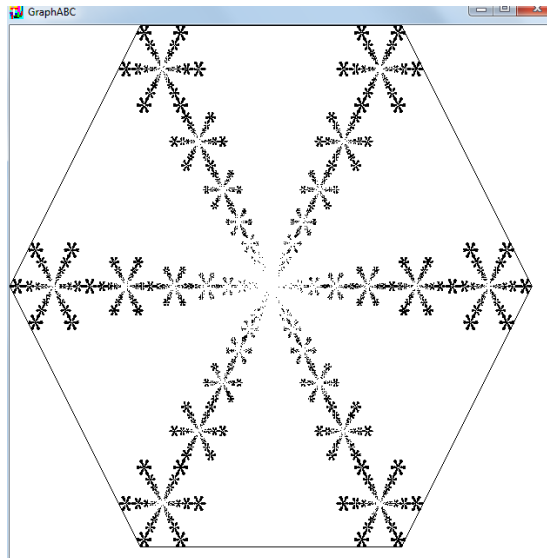


Рис. 2.24. Сніжинка у нескінченність (результат роботи алгоритму)

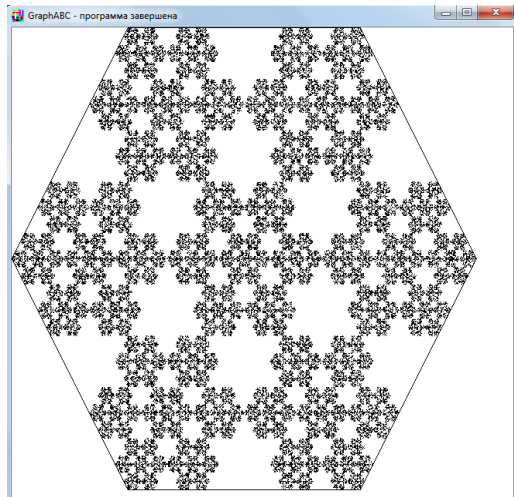


Рис. 2.25. Сніжинка Коха (результат роботи алгоритму)

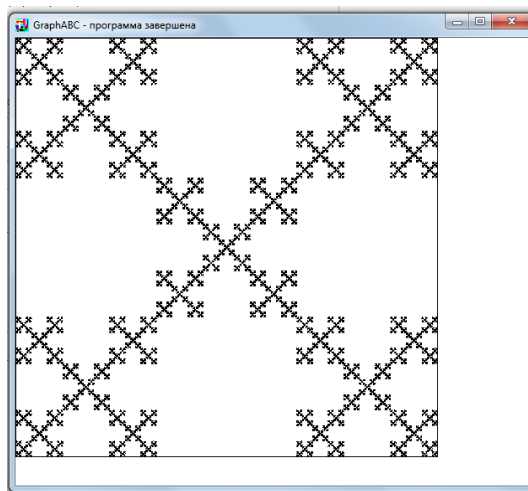


Рис. 2.26. Фрактал Хрест (результат роботи алгоритму)

2.3 Оцінка обчислювальної складності методів побудови фрактального зображення з використанням РСІФ та ДСІФ

Детермінована система ітераційних функцій застосовується для створення фрактальних зображень шляхом послідовної обробки початкового набору значень, який може представляти собою простий геометричний об'єкт, такий як квадрат. Застосування системи ітераційних функцій до цього початкового об'єкта генерує нове зображення на першій ітерації. Цей процес повторюється кілька разів на наступних етапах ітерації, що дозволяє фракталу набирати більш складні форми. Однак важливо зазначити, що цей метод вимагає обробки кожного пікселя зображення математичними операціями на кожному етапі ітерації для кожної ітераційної функції, що може бути досить обчислювально витратним процесом.

У свою чергу, рандомізована система ітераційних функцій забезпечує побудову фрактального зображення піксель за пікселем. Цей метод не вимагає великих обсягів пам'яті і не потребує значних обчислень. За допомогою розрахунків в роботі [113] оцінено ефективність використання рандомізованої системи ітераційних функцій (РСІФ) порівняно з детермінованою системою ітераційних функцій (ДСІФ).

Зокрема, для оцінки ефективності створення фрактальних зображень проведено порівняння кількості операцій (обчислювальної складності), необхідних для генерації одного фрактального зображення з обмеженою роздільною здатністю. Ця метрика дала змогу визначити, наскільки швидше РСІФ може побудувати фрактал у порівнянні з ДСІФ:

$$E = \frac{N_D}{N_R} \quad (2.20)$$

де E - ефективність побудови фрактального зображення; N_D - кількість операцій для побудови фрактального зображення з використанням ДСІФ; N_R - кількість операцій для побудови фрактального зображення з використанням РСІФ.

Для побудови фрактального зображення з використанням детермінованої системи ітераційних функцій використано афінні перетворення площини. Починаємо з початкової фігури і застосовуємо до неї перетворення (ітерацію), щоб отримати нову фігуру. Потім цей процес повторюється, доки не досягнуто необхідну кількість ітерацій фракталу (Рис.2.27).

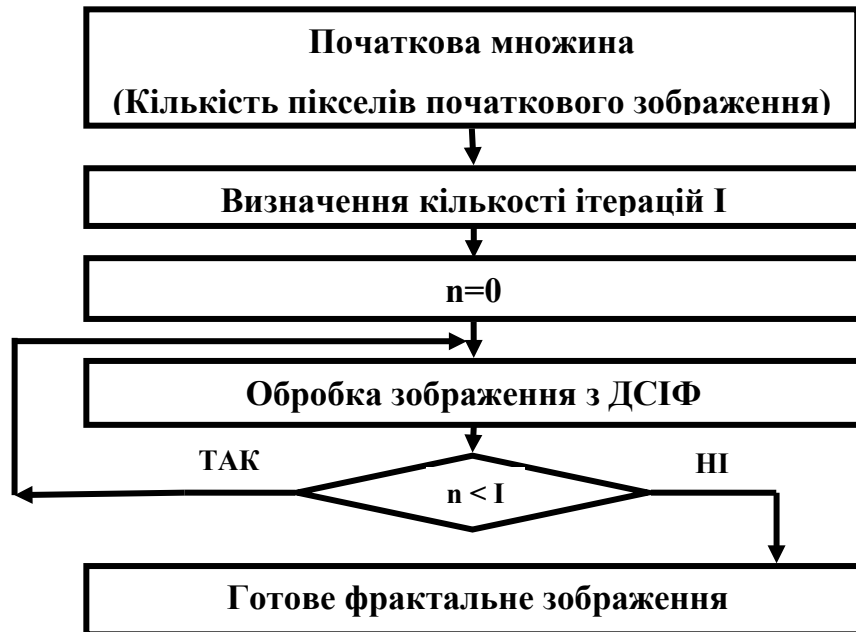


Рис. 2.27. Побудова фракталу з використанням ДСІФ

Результати першої і другої ітерації, що виникають при роботі алгоритму побудови детермінованої системи ітераційних функцій, зображено на рис. 2.28 та рис. 2.29:

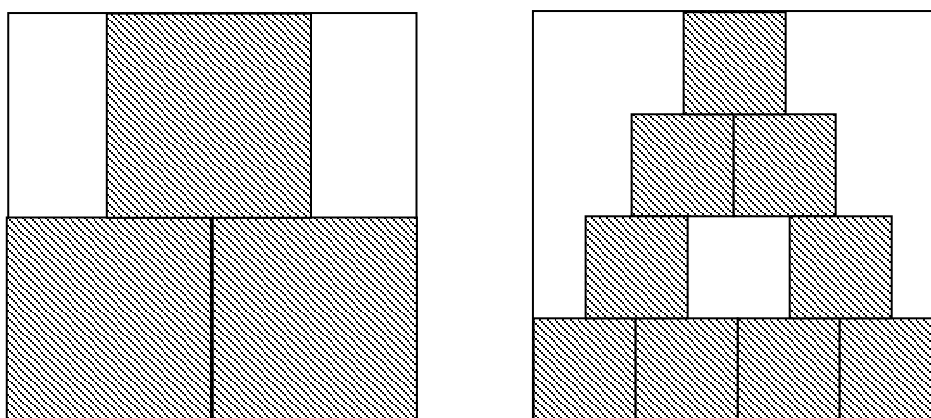


Рис. 2.28. Першої та другої ітерації побудови фракталу трикутник Серпінського

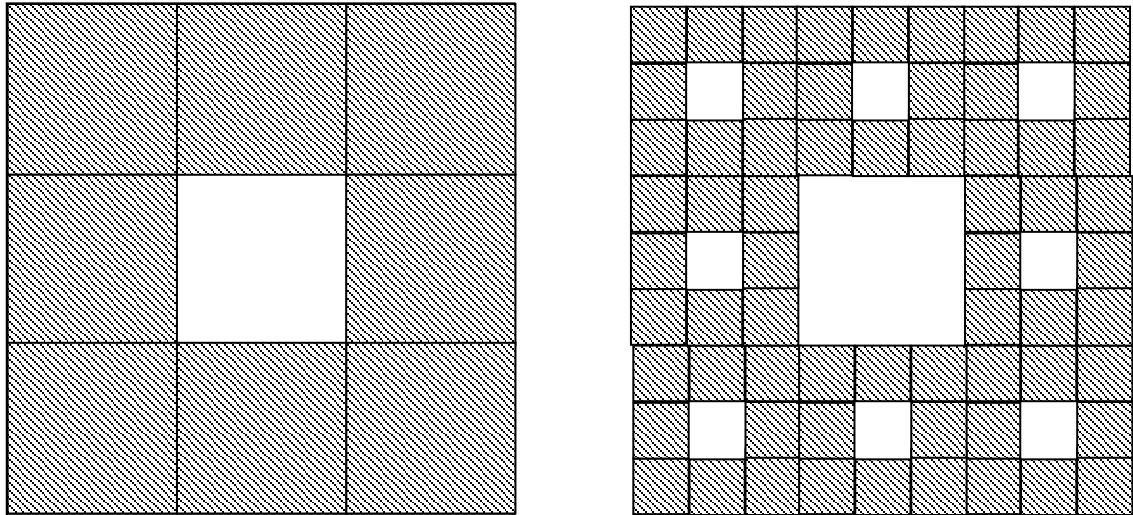


Рис. 2.29. Першої та другої ітерації побудови фракталу квадрату Серпінського

Для аналізу даного алгоритму роботи потрібно визначити кількість ітерацій I . Для цього ми задаємо початкову довжину і ширину зображення L , а також довжину і ширину елемента першої ітерації l . Потім ми знаходимо коефіцієнт пропорційності K за формулою (2.21):

$$K = \frac{L}{l} \quad (2.21)$$

де K - коефіцієнт пропорційності.

Щоб визначити кількість ітерацій, нам потрібно знати, скільки разів будуть застосовуватися афінні перетворення, поки остання ітерація не дасть 1 піксель. Для цього ми використовуємо формулу (2.22):

$$\frac{L}{K^I} = 1 \quad (2.22)$$

де I - кількість ітерацій.

Тепер визначимо кількість ітерацій I (2.23-2.24):

$$L = K^I \quad (2.23)$$

$$I = \log_K L \quad (2.24)$$

Далі ми визначаємо кількість операцій N_{DI} , необхідну для побудови першої ітерації фрактального зображення з використанням ДСІФ з урахуванням кількості самоподібних фігур першої ітерації n (2.25):

$$N_{D1} = L^2 \cdot n \quad (2.25)$$

де n - кількість самоподібних фігур першої ітерації.

Отже, для побудови фрактального зображення з I кількістю ітерацій ми використовуємо наступний вираз (2.26):

$$N_D = L^2 \cdot n^I \quad (2.26)$$

Підставляючи формулу (2.24) у формулу (2.26), ми отримуємо кількість операцій, необхідних для побудови фрактального зображення з обмеженою роздільною здатністю з використанням ДСІФ (2.27):

$$N_D = L^2 \cdot n^{\log_K L} \quad (2.27)$$

де N_D - кількість операцій для побудови фрактального зображення з використанням ДСІФ; K - коефіцієнт пропорційності; n - кількість самоподібних фігур першої ітерації; L - початкова довжина і ширина зображення.

Для створення фрактального зображення за допомогою рандомізованої системи ітераційних функцій (РСІФ), яка виводиться в розділі 3 даної дисертації, ми використовуємо наступний вираз:

$$N_I = L^2 \cdot \left(\frac{n}{K^2} \right)^{\log_K L} \quad (2.28)$$

де N_D - кількість операцій для побудови фрактального зображення з використанням РСІФ; n - параметр; L - довжина зображення; K - коефіцієнт пропорційності.

Нижче наведені графіки, що показують залежність кількості операцій від довжини зображення для трикутника та квадрата Серпінського на Рис. 2.30 і Рис. 2.31.

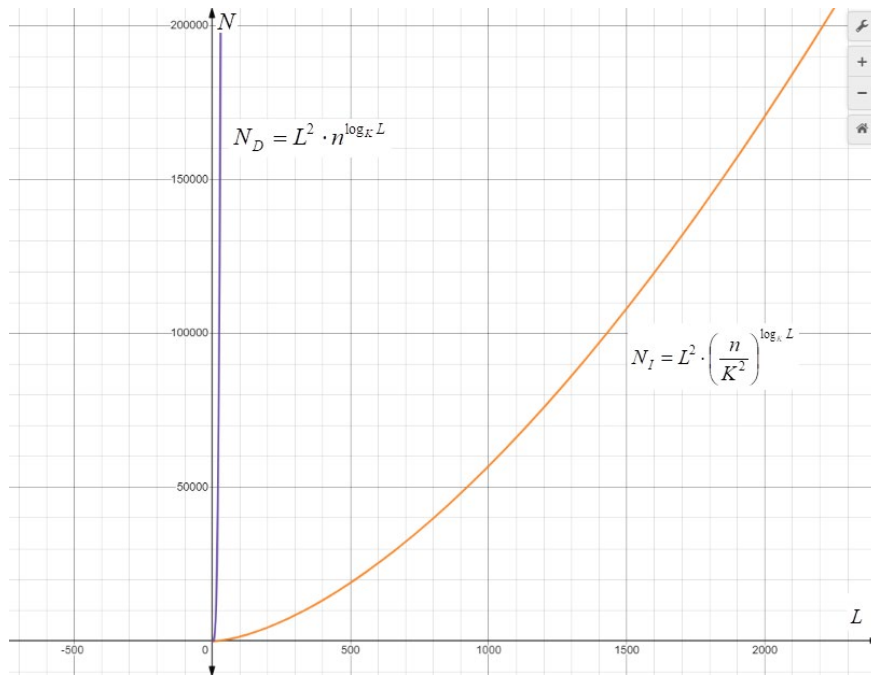


Рис. 2.30. Графіки залежності кількостей операцій (обчислювальної складності) ДСІФ та РСІФ від довжини зображення для трикутника Серпінського з параметрами $n=3$, $K=2$

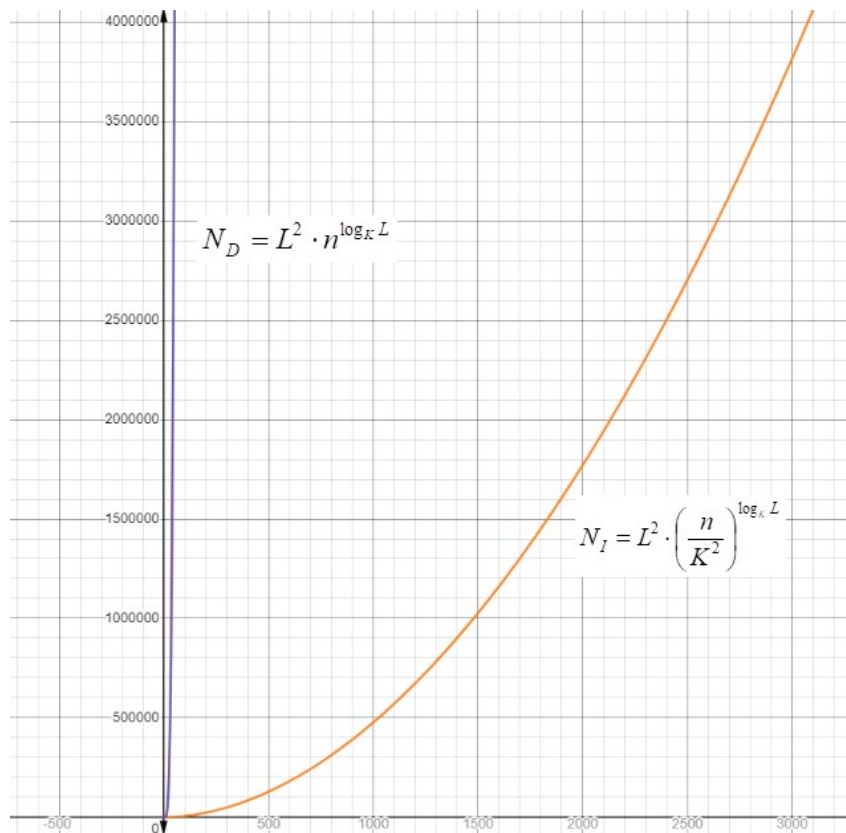


Рис. 2.31. Графіки залежності кількості операцій ДСІФ та РСІФ від довжини зображення для квадрату Серпінського з параметрами $n=8$, $K=3$

Залежно від розмірів фрактального зображення (довжина і ширина), побудова фрактальних зображень за допомогою детермінованої системи ітераційних функцій (ДСІФ) вимагає значних обчислень та обчислювальних ресурсів, у порівнянні з побудовою за допомогою рандомізованої системи ітераційних функцій (РСІФ), як показано на графіках на Рис. 2.30 і Рис.2.31.

Для визначення ефективності використання РСІФ порівняно з ДСІФ, можемо використати формулу (2.20) з підстановкою формул (2.27-2.28):

$$E = \frac{N_D}{N_R} = \frac{L^2 \cdot n^{\log_K L}}{L^2 \cdot \left(\frac{n}{K^2}\right)^{\log_K L}} = \frac{n^{\log_K L} \cdot K^{2\log_K L}}{n^{\log_K L}} = K^{2\log_K L} = K^{\log_K L^2} = L^2$$

$$E = L^2 = S \quad (2.29)$$

де E - ефективність побудови фрактальних зображень з використанням РСІФ над ДСІФ; N_D - кількість операцій для побудови фрактального зображення з використанням ДСІФ (згідно формули (2.27)); N_R - кількість операцій для побудови фрактального зображення з використанням РСІФ (згідно формули (2.28)).

Якщо довжина та ширина зображення різняться, аналогічними розрахунками ми можемо отримати наступний вираз (2.30):

$$E = L_X \cdot L_Y = S \quad (2.30)$$

де L_X та L_Y - довжина та ширина фрактального зображення; S - роздільна здатність (у пікселях) фрактального зображення.

Таким чином, формули (2.29) і (2.30) дозволяють визначити ефективність використання РСІФ порівняно з ДСІФ з урахуванням розмірів та роздільної здатності фрактального зображення.

Провівши розрахунки кількості операцій для побудови фрактального зображення обмеженого роздільною здатністю та побудувавши графіки залежності стає очевидним перевагу використання рандомізованої системи

ітераційних функцій РСІФ над детермінованою системою ітераційних функцій ДСІФ у S разів. Для розробки систем генерування фрактальних зображень, явно переважає РСІФ, так як не потребує значних обчислень та великого об'єму пам'яті, що в свою чергу дасть змогу підготувати матеріали для машинного навчання, для розпізнавання графічних об'єктів та відновлення та відтворення (спотворення), стискання графічних фрактальних зображень. Швидкість роботи РСІФ дозволить виконати поставленні задачі у реальному часі, що неможливо б було зробити при ДСІТ.

Висновки до 2-го розділу

Сьогодні фрактальні структури дуже поширені в різних галузях науки і техніки, але інструментів для моделювання фрактальних структур дуже мало. У більшості випадків використовуються детерміновані системи ітераційних функцій, які потребують значних математичних розрахунків. Алгоритми для RSIF спрощують це завдання, але розрахунок коефіцієнтів для функцій РСІФ досить складний і невизначений. Тому запропоновано подання РСІФ в удосконаленій математичній формі (2.16), що дає змогу подавати РСІФ як геометричні параметри перших ітерацій фрактального зображення. У роботі розглянуто і вдосконалено метод побудови фрактальних структур, використовуючи систему РСІФ, що базується на аналізі центрів перших відрізків ітерації та обчисленні коефіцієнтів РСІФ. Однією з основних переваг цього підходу є можливість виконувати як прямі, так і зворотні перетворення без потреби у додаткових програмних або апаратних ресурсах. Це робить його особливо привабливим для застосування у різних областях, де важлива точність обробки фрактальних зображень. Використання прямого та зворотного перетворень дозволить у майбутньому сформувати вихідний набір даних для нейронних мереж, які ляжуть в основу розпізнавання об'єктів. В роботі проведено 11 експериментів симуляційного моделювання, результати яких дали можливість визначити залежність центрів перших відрізків ітерації з точками,

які вибираються рандомізовано (2.18), це в свою чергу дало змогу описати математичну модель фрактального зображення створеного за допомогою удосконаленої РСІФ. В результаті дослідження запропонований метод та симуляційне моделювання дали змогу відтворити фрактальні структури з точністю 99,99%, відтворені наступні фрактальні структури: множина Кантора, трикутник і квадрат Серпінського (рис. 2.15 та рис. 2.23), Сніжинка Коха (рис. 2.25), фрактальний хрест (рис. 2.26), фрактал «Усмішка» (рис. 2.17), Сніжинка у нескінченість (рис. 2.24).

Аналізуючи кількість операцій, необхідних для побудови фрактального зображення з обмеженою роздільною здатністю і аналізуючи графіки залежності, стає очевидним, що використання рандомізованої системи ітераційних функцій (РСІФ) на S разів виявляється більш ефективним в порівнянні з детермінованою системою ітераційних функцій (ДСІФ).

РОЗДІЛ 3. МЕТОДИ ТА МОДЕЛЬ РОЗПІЗНАВАННЯ І ШИФРУВАННЯ ФРАКТАЛЬНИХ ЗОБРАЖЕНЬ НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ

3.1 Метод розпізнавання фрактальних структур на основі нейронних мереж

Для побудови та розпізнавання фракталів за допомогою РСІФ потрібно обчислити координати кожної точки на кожному кроці, щоб вивести їх на екран. Цей обчислювальний процес залежить від кількості пікселів у фрактальному зображенні і вимагає розрахунку великої кількості операцій. Використання формул для знаходження кількості операцій дозволить визначити кількість випадкових подій (стохастичний рух точки) і відкриє можливість створення бібліотеки JavaScript для стиснення та відтворення зображень типу "канторова пила" на основі рандомізованої системи ітераційних функцій [114].

Розглянемо ситуацію, коли коефіцієнт подібності однаковий для осей X і Y , а також для всіх фігур першої ітерації. Почнемо з обмеження зображення квадратом довжиною L . Спробуємо відобразити першу та другу ітерації трикутника Серпінського, а також квадрат Серпінського (Рис. 3.1, Рис. 3.2) [114]:

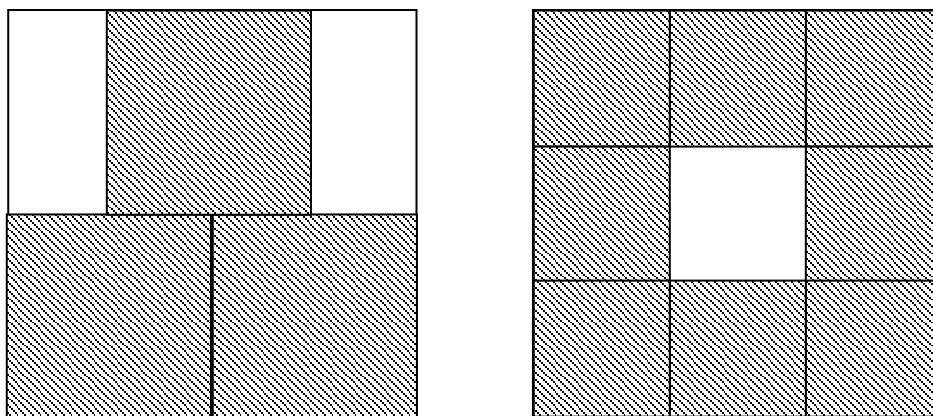


Рис. 3.1. Зображення першої ітерації трикутника та квадрату Серпінського

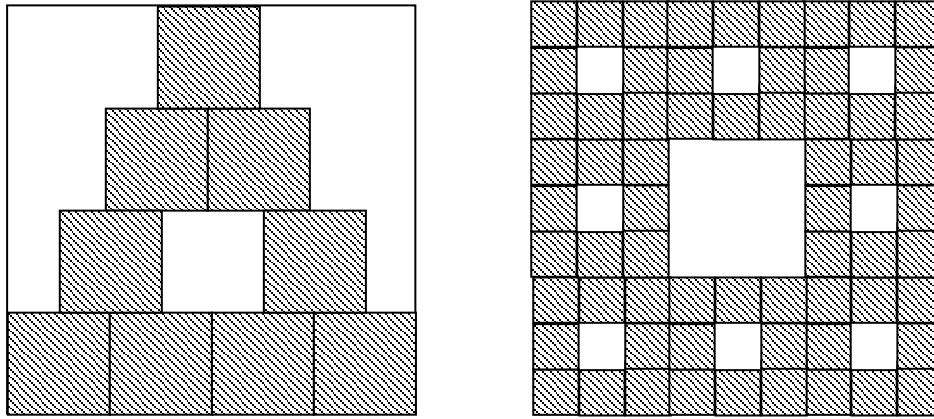


Рис. 3.2. Зображення другої ітерації трикутника та квадрату Серпінського

Припустимо, що сторона квадрата першої ітерації має довжину l . Тепер ми розрахуємо коефіцієнт пропорційності K :

$$K = \frac{L}{l}, \quad (3.1)$$

Спробуємо встановити відношення між кількістю пікселів фігур першої ітерації і кількістю точок квадрата, який обмежує їх [114]:

$$\frac{N_{\phi_1}}{N} = \frac{n \cdot N_0}{N}, \quad (3.2)$$

де N_{ϕ_1} - кількість пікселів першої ітерації фрактала; N - кількість пікселів в квадраті, який обмежує; n - кількість фігур першої ітерації; N_0 - кількість пікселів у фігурі першої ітерації.

Визначимо кількість пікселів в нашому квадраті:

$$N = L^2 \quad (3.3)$$

Визначимо кількість пікселів у фігурі першої ітерації:

$$N_0 = l^2 \quad (3.4)$$

Підставимо значення виразів (3.3) і (3.4) у формулу (3.2), отримаємо:

$$\frac{N_{\phi_1}}{N} = \frac{n \cdot N_0}{N} = \frac{n \cdot l^2}{L^2} \quad (3.5)$$

Підставимо значення виразу (3.1) у формулу (3.5), отримаємо:

$$\frac{N_{\phi_1}}{N} = \frac{n \cdot l^2}{L^2} = \frac{n}{\frac{L^2}{l^2}} = \frac{n}{\left(\frac{L}{l}\right)^2} = \frac{n}{K^2} \quad (3.6)$$

Спробуємо встановити відношення між кількістю пікселів у фігурах з кількістю ітерацій і кількістю пікселів у квадраті, який обмежує їх [114]:

$$\frac{N_{\phi}}{N} = \left(\frac{n}{K^2}\right)^I \quad (3.7)$$

де I - кількість ітерацій фрактала; N_{ϕ} - кількість пікселів фрактала.

Як видно з виразу (3.7), кількість пікселів фрактала визначається (3.8) [114]:

$$N_{\phi} = N \cdot \left(\frac{n}{K^2}\right)^I = L^2 \cdot \left(\frac{n}{K^2}\right)^I \quad (3.8)$$

Для визначення кількості ітерацій можна виконати такі кроки: візьмемо довжину відрізка L і поділимо її на коефіцієнт пропорційності. Потім візьмемо один з отриманих відрізків і знову поділіть його на коефіцієнт пропорційності. Продовжуємо цей процес, ділячи відрізки, поки довжина отриманого відрізка не стане рівною 1 точці (пікселю). Кількість поділень, яку ми виконали, відповідатиме кількості ітерацій [114]:

$$\frac{L}{K^I} = 1 \quad (3.9)$$

$$L = K^I \quad (3.10)$$

$$I = \log_K L \quad (3.11)$$

Підставимо формулу (3.11) у формулу (3.8) та знайдемо кількість пікселів фрактала (3.12) [114]:

$$N_{\phi} = L^2 \cdot \left(\frac{n}{K^2}\right)^{\log_K L} \quad (3.12)$$

На Рис. 3.3 показано алгоритм визначення кількості пікселів при однакових коефіцієнтах подібності [114]:

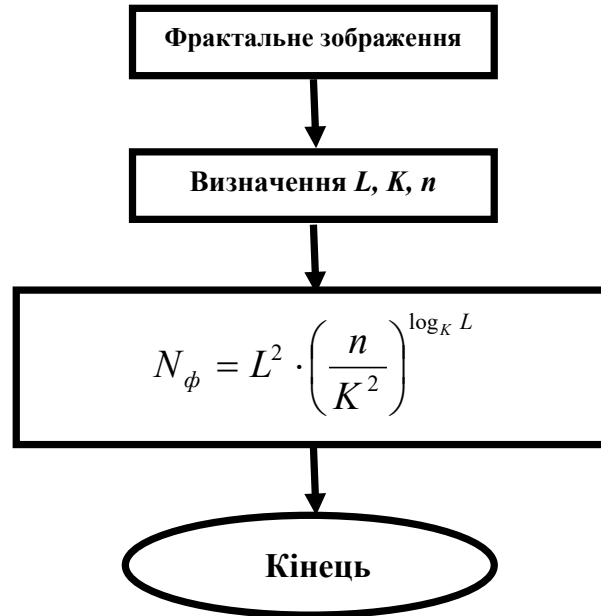


Рис. 3.3 Алгоритм знаходження кількості пікселів при рівних коефіцієнтах подібності

Розглянемо ситуацію, коли коефіцієнт подібності відрізняється для осей X і Y, а також для всіх фігур першої ітерації.

На початку обмежимо зображення прямокутником зі сторонами L_X і L_Y .

Припустимо, що сторони n-прямокутника першої ітерації мають значення l_{Xn} і l_{Yn} (ці значення різні для кожного прямокутника першої ітерації). Тепер розрахуємо коефіцієнт пропорційності (3.13-3.14) [114]:

$$K_{Xn} = \frac{L_X}{l_{Xn}} \quad (3.13)$$

$$K_{Yn} = \frac{L_Y}{l_{Yn}} \quad (3.14)$$

де K_{Xn} , K_{Yn} - коефіцієнт пропорційності по осях X та Y n-прямокутника першої ітерації.

Давайте спробуємо встановити залежність між кількістю пікселів у фігурах першої ітерації і кількістю пікселів у прямокутнику, що їх обмежує (3.15):

$$\frac{N_{\phi_1}}{N} = \frac{\sum_{i=1}^n N_i}{N} \quad (3.15)$$

де N_i - кількість пікселів у фігурі першої ітерації.

Визначимо кількість пікселів у прямокутнику, що обмежує (3.16):

$$N = L_X * L_Y \quad (3.16)$$

Визначимо кількість пікселів у фігурі першої ітерації (3.17):

$$N_i = l_{X_i} * l_{Y_i} \quad (3.17)$$

Підставимо формули (3.16) і (3.17) у формулу (3.15), отримаємо (3.18) [114]:

$$\frac{N_{\phi_1}}{N} = \frac{\sum_{i=1}^n N_i}{N} = \frac{\sum_{i=1}^n l_{X_i} \cdot l_{Y_i}}{L_{X_i} \cdot L_{Y_i}} \quad (3.18)$$

Підставимо формули (3.13) і (3.14) у формулу (3.18), отримаємо (3.18):

$$\frac{N_{\phi_1}}{N} = \frac{\sum_{i=1}^n l_{X_i} \cdot l_{Y_i}}{L_{X_i} \cdot L_{Y_i}} = \sum_{i=1}^n \left(\frac{1}{K_{X_i} \cdot K_{Y_i}} \right) \quad (3.19)$$

Спробуємо встановити відношення між кількістю пікселів у фігурах з кількістю ітерацій і кількістю пікселів у прямокутнику, який їх обмежує (3.20):

$$\frac{N_{\phi}}{N} = \left(\sum_{i=1}^n \left(\frac{1}{K_{X_i} \cdot K_{Y_i}} \right) \right)^I \quad (3.20)$$

Як видно з формули (3.20), кількість пікселів фрактала буде рівна [114]:

$$N_{\phi} = N \cdot \left(\sum_{i=1}^n \left(\frac{1}{K_{X_i} \cdot K_{Y_i}} \right) \right)^I = L_X \cdot L_Y \cdot \left(\sum_{i=1}^n \left(\frac{1}{K_{X_i} \cdot K_{Y_i}} \right) \right)^I \quad (3.21)$$

Тепер визначимо кількість ітерацій (3.22-3.24) [114]:

$$\frac{L_X \cdot L_Y}{\left(\frac{\sum_{i=1}^n (K_{X_i})}{n} \cdot \frac{\sum_{i=1}^n (K_{Y_i})}{n} \right)^I} = 1 \quad (3.22)$$

$$L_X \cdot L_Y = \left(\frac{\sum_{i=1}^n (K_{X_i}) \cdot \sum_{i=1}^n (K_{Y_i})}{n^2} \right)^I \quad (3.23)$$

$$I = \log \left(\frac{\sum_{i=1}^n (K_{X_i}) \cdot \sum_{i=1}^n (K_{Y_i})}{n^2} \right) L_X \cdot L_Y \quad (3.24)$$

Підставимо формулу (3.24) у формулу (3.21) і знайдемо кількість пікселів фрактала (3.25) [114]:

$$N_{\phi} = L_X \cdot L_Y \cdot \left(\sum_{i=1}^n \left(\frac{1}{K_{X_i} \cdot K_{Y_i}} \right) \right)^{\log \left(\frac{\sum_{i=1}^n (K_{X_i}) \cdot \sum_{i=1}^n (K_{Y_i})}{n^2} \right) L_X \cdot L_Y} \quad (3.25)$$

На Рис. 3.4 наведено алгоритм, який дозволяє визначити кількість пікселів при різних коефіцієнтах подібності [114]:

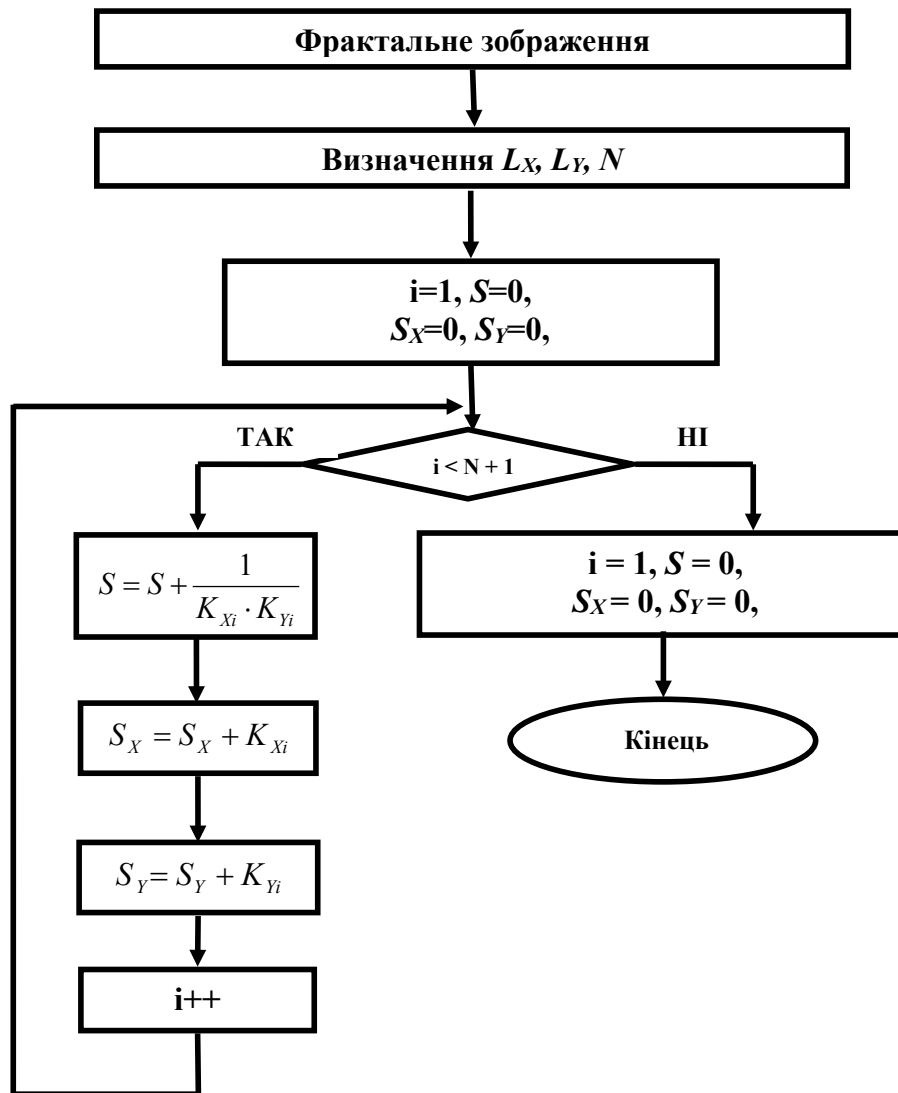


Рис. 3.4. Алгоритм визначення кількості пікселів при різних коефіцієнтах подібності

На сьогоднішній день фрактальні структури широко застосовуються в різних галузях науки і техніки, але інструментів для моделювання таких структур є недостатньо. Розробка інструменту для побудови фрактальних структур вимагає складних математичних обчислень. Алгоритм визначення пікселів фрактального зображення типу «Cantor dust» дозволить спростити процес побудови фрактальних зображень. Цей алгоритм обчислення не використовує вкладеного циклу та рекурсивних функцій, і він достатньо оптимізований, що дозволяє ефективно використовувати обчислювальні

ресурси. Застосування подальших формул дозволить визначити кількість випадкових подій, що забезпечить якість відтворення зображень за допомогою РСІФ. Крім того, це дозволить створити автоматизовану систему стиснення та відтворення зображень, а також надасть можливість формувати набір даних для навчання нейронних мереж, які можуть використовуватись для розпізнавання фрактальних об'єктів.

Щоб реалізувати алгоритм відновлення та відтворення фрактального зображення з низькою якістю потрібно навчити нейронну мережу розпізнавати фрактали з низькою якістю. Складність задачі пов'язана з тим, що генератор фракталів генерує фрактали з високою якістю, тому спробуємо вирішити дану задачу.

Для зменшення якості зображення, треба зменшити кількість пікселів фрактального зображення. Розглянемо ймовірність події $P(A)$, що точно не попадає в задане місце на фракталі, при m випробувань можна розрахувати за формулою (3.26) [115]:

$$P(A) = \left(\frac{N_\phi - 1}{N_\phi} \right)^m \quad (3.26)$$

де N_ϕ - кількість точок фрактала; m - кількість випробувань.

Якість зображення Q являє собою протилежну подію до події $P(A)$ (3.27):

$$Q = 1 - P(A) \quad (3.27)$$

Визначимо кількість випробувань m для забезпечення якості зображення Q (3.28-3.29) [115]:

$$Q = 1 - \left(\frac{N_\phi - 1}{N_\phi} \right)^m \quad (3.28)$$

$$m = \log_{1 - \frac{1}{N_\phi}} 1 - Q \quad (3.29)$$

Підставивши формулу (3.12), визначимо кількість випробувань m для фрактального зображення правильної форми (3.30) [115]:

$$m = \log_{1 - \frac{1}{L^2 \cdot \left(\frac{n}{K^2}\right)^{\log_K L}}} 1 - Q \quad (3.30)$$

Підставивши формулу (3.25), визначимо кількість випробувань m для прямокутного фрактального зображення (3.31) [115]:

$$m = \log_{1 - \frac{1}{L_X \cdot L_Y \cdot \left(\sum_{i=1}^n \left(\frac{1}{K_{X_i} \cdot K_{Y_i}} \right) \right)^{\log \left(\frac{\sum_{i=1}^n (K_{X_i}) \cdot \sum_{i=1}^n (K_{Y_i})}{n^2} \right)^{L_X \cdot L_Y}}} 1 - Q \quad (3.31)$$

Використання нейронних мереж для розпізнавання об'єктів є актуальною задачею в сфері інформаційних технологій. Розпізнавання об'єктів є складною задачею, тому що потребує великого збору вхідних та вихідних даних. Розпізнавання фрактальних зображень - ще більш складніша задача, тому що побудова фракталів потребує великих обчислюваних потужностей. У разі використання РСІФ, задача дещо спрощується, немає необхідності зберігати великі масиви даних у пам'яті. Тому цей алгоритм зручно використовувати на комп'ютерах з обмеженими ресурсами. Побудова фракталів за допомогою РСІФ включає в себе розрахунок на кожному кроці координати однієї точки для виведення її на екран, для цього потрібно розрахувати кількість операцій, яка на пряму залежить від кількості пікселів фрактального зображення. РСІФ дає можливість змінювати якість зображення, що в свою чергу дає можливість навчати нейронну мережу розпізнавати фрактал з різною якістю та відновлювати його.

Для навчання нейронної моделі, яка б змогла розпізнавати фрактальні зображення і перетворювати їх в систему ітераційних функцій РСІФ, потрібно вхідні і вихідні дані. Вхідними даними будуть виступати фрактальні зображення (пікселі зображення), вихідними даними будуть виступати РСІФ (Рис.3.5) [115].

За допомогою РСІФ (2.26) та алгоритму, представленого в рис. 2.19 , ми можемо побудувати любий фрактал (Рис. 3.7-3.8), [115]:

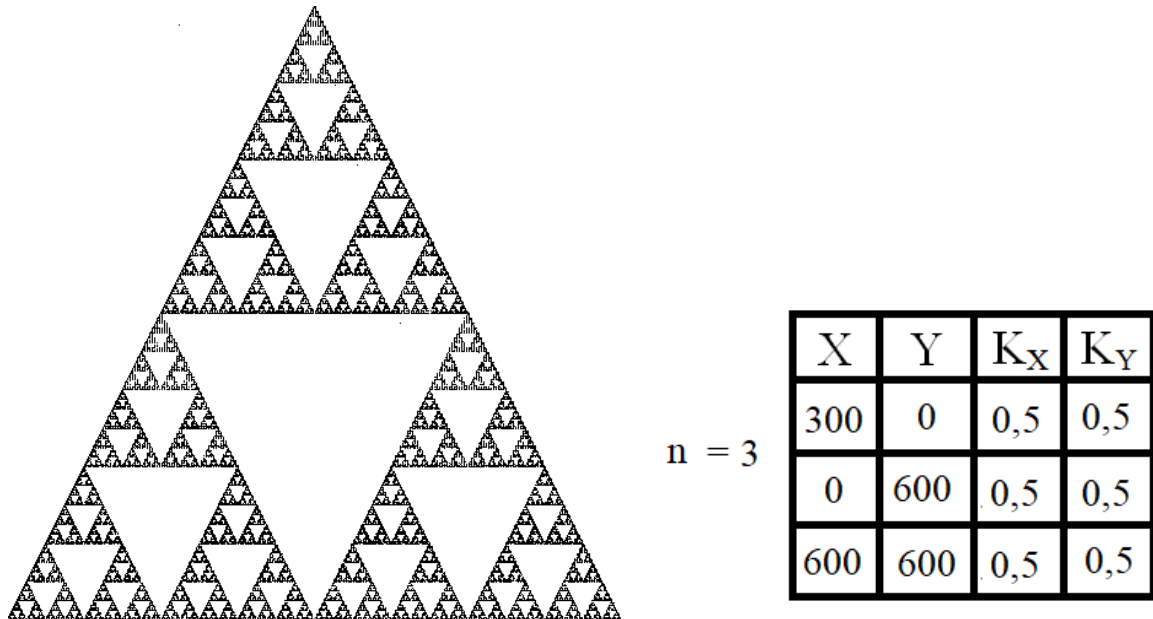


Рис. 3.7. Трикутник Серпінського (результат роботи алгоритму)

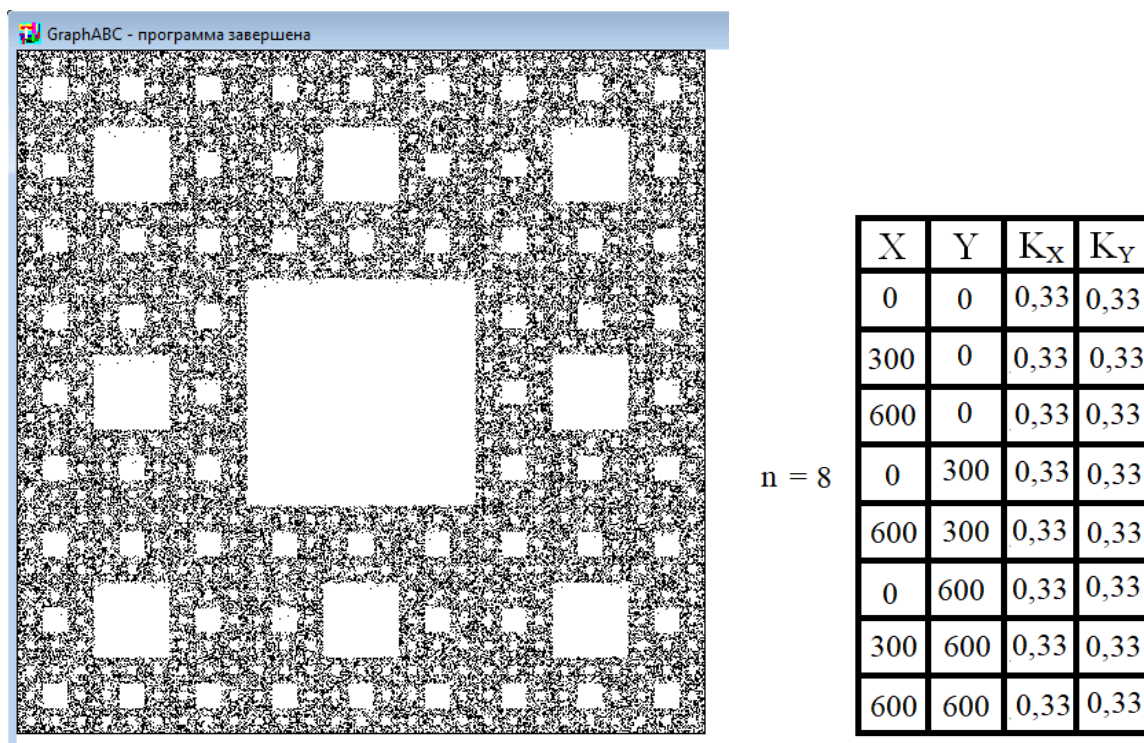


Рис. 3.8. Квадрат Серпінського (результат роботи алгоритму)

Як бачимо з попередніх результатів (Рис. 3.7, Рис. 3.8) матриця РСІФ на пряму залежить від роздільної здатності зображення. З'являється необхідність

задавати (генерувати) розміри зображення L_X, L_Y . Так як наша нейронна модель може довчатися, для початку задамо обмежений діапазон роздільної здатності до 1024px на 1024px (3.32) [115]:

$$1 < L_X < 1024, L_X = \text{Random}(0..1024); \quad (3.32)$$

$$1 < L_Y < 1024, L_Y = \text{Random}(0..1024).$$

Наступним кроком з'являється необхідність задавати (генерувати) кількість фігур першої ітерації. Так як наша нейронна модель може довчатися, для початку задаємо обмежений діапазон генерації фігур першої ітерації до 100 (3.33) [115]:

$$1 < n < 100, n = \text{Random}(0..100) \quad (3.33)$$

Наступним кроком буде визначення параметрів ітераційних функцій $X_i, Y_i, K_{X_i}, K_{Y_i}$ для побудови матриці РСІФ за допомогою рандомізованого алгоритму (3.34-3.37) [115]:

$$X_i = \text{Random}(0..L_X) \quad (3.34)$$

$$Y_i = \text{Random}(0..L_Y) \quad (3.35)$$

$$K_{X_i} = \text{Random}(0..1) \quad (3.36)$$

$$K_{Y_i} = \text{Random}(0..1) \quad (3.37)$$

Наступним кроком потрібно визначити кількість пікселів зображення N . Для цього використаємо наступну формулу (3.25).

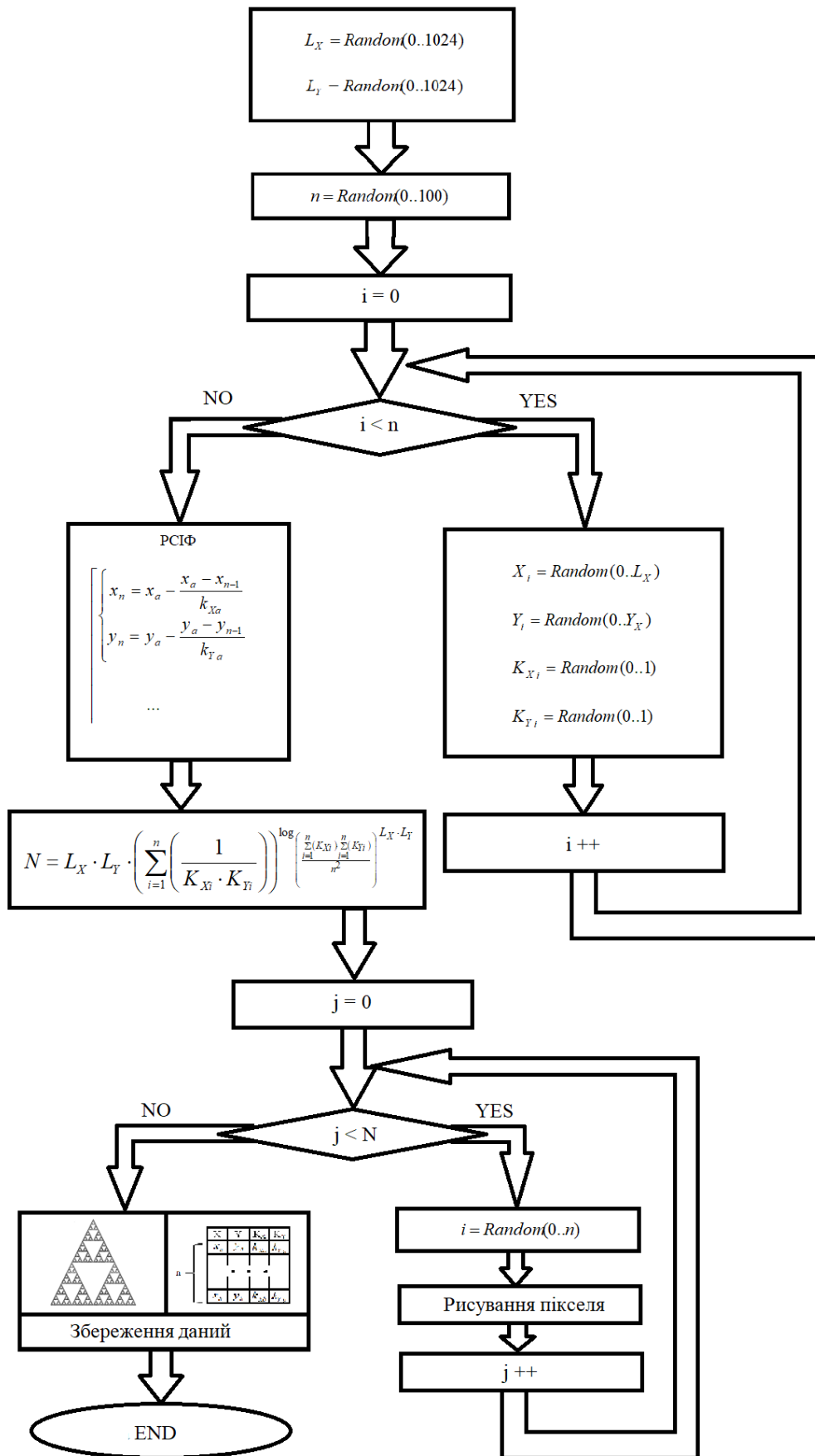


Рис. 3.9. Алгоритм роботи генератора фракталів на удосконаленій РСІФ

Всі параметри для побудови РСІФ є визначені за допомогою попередніх формул, схема алгоритму генератора матриць РСІФ та побудованих відповідних фракталів наведена на Рис. 3.9. Наведений алгоритм побудови не використовує рекурсивних функцій та входження циклу в цикл, що дозволяє не затрачати великі обчислювальні потужності, і є досить оптимізованим. Тому цей алгоритм зручно використовувати на комп'ютерах з обмеженими ресурсами. Побудова фракталів за допомогою РСІФ включає в себе розрахунок на кожному кроці координати однієї точки для виведення її на екран. Кількість операцій роботи алгоритму на пряму залежить від кількості пікселів фрактального зображення (3.25) [115].

Дані алгоритму дають можливість навчання нейронної моделі, для розпізнавання фрактальних зображень типу «Фрактальний пил» (набір Кантора). Даний алгоритм (Рис. 3.9) не дозволяє розпізнавати фрактальне зображення з низькою якістю, це пов'язано з тим що генератор фракталів генерує фрактали з високою якістю. Попробуємо вирішити дану задачу [115].

Для зменшення якості зображення, треба зменшити кількість пікселів фрактального зображення. Цю задачу ми вирішуємо за допомогою формули (3.31).

Для визначення якості зображення Q використаємо рандомізований алгоритм (3.38)

$$Q = \text{Random}(0..1) \quad (3.38)$$

Тепер алгоритм генератора фрактальних зображень на базі удосконаленої РСІФ буде мати наступний вигляд Рис. 3.10 [115]:

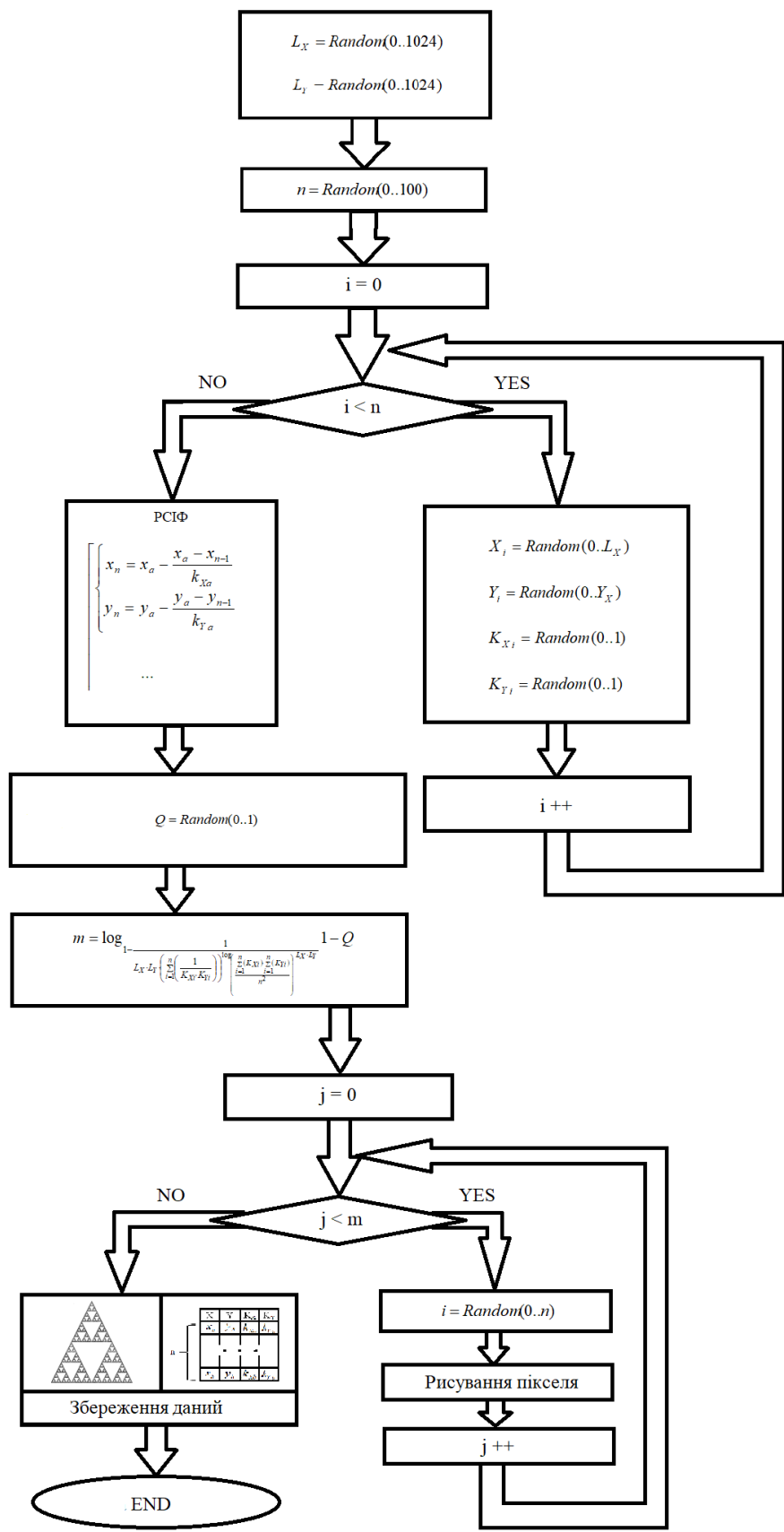


Рис. 3.10 Алгоритм роботи генератора фракталів та РСІФ з різною якістю зображення

Програмна реалізація алгоритму. Для навчання нейронної мережі використаємо наступну схему Рис. 3.11 [115]:

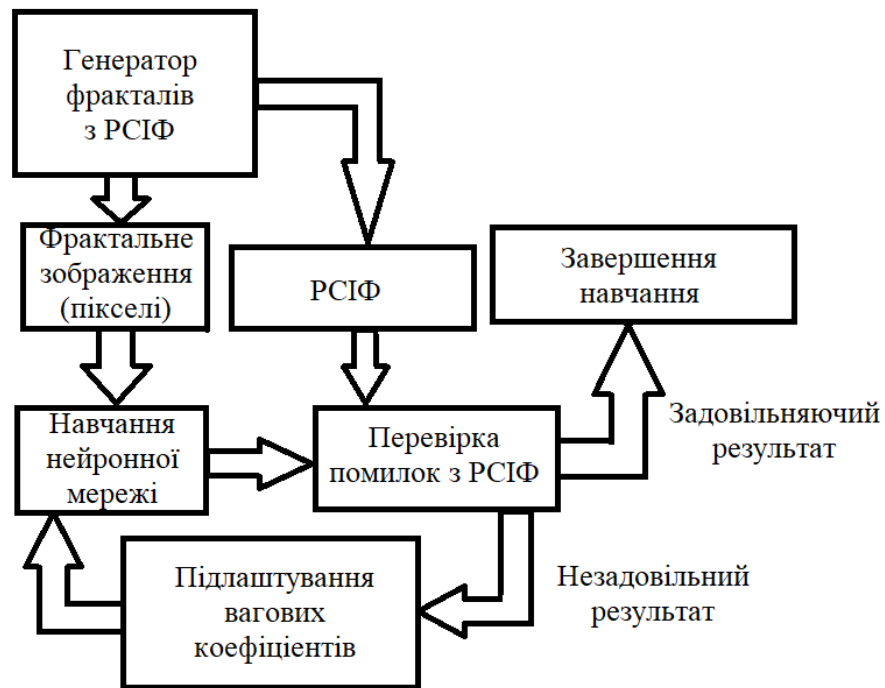


Рис.3.11 Алгоритм навчання нейронної мережі з використанням генератора фрактальних зображень з РСІФ

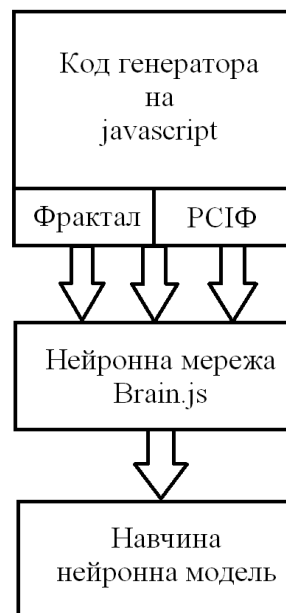


Рис.3.12 Програмна реалізація нейронної мережі на базі бібліотеки Brain.js

Для програмної реалізації нейронної мережі будемо використовувати WEB технології, а саме для написання коду і візуалізації використаємо мову програмування javascript, та javascript бібліотеки brain.js, схема програмної реалізації зображена на Рис. 3.12 [115].

Результати практичної реалізації. Після написання коду програми і навчання нейронної мережі перейдемо до перевірки результатів роботи. Для перевірки візьмемо фрактальні зображення з низькою якістю ($Q < 0.15$) (Рис. 3.13) та пропустимо це зображення через нашу навчену нейронну мережу (Рис. 3.14), отримаємо РСІФ, після чого побудуємо наш фрактал (Рис 3.15) [115].

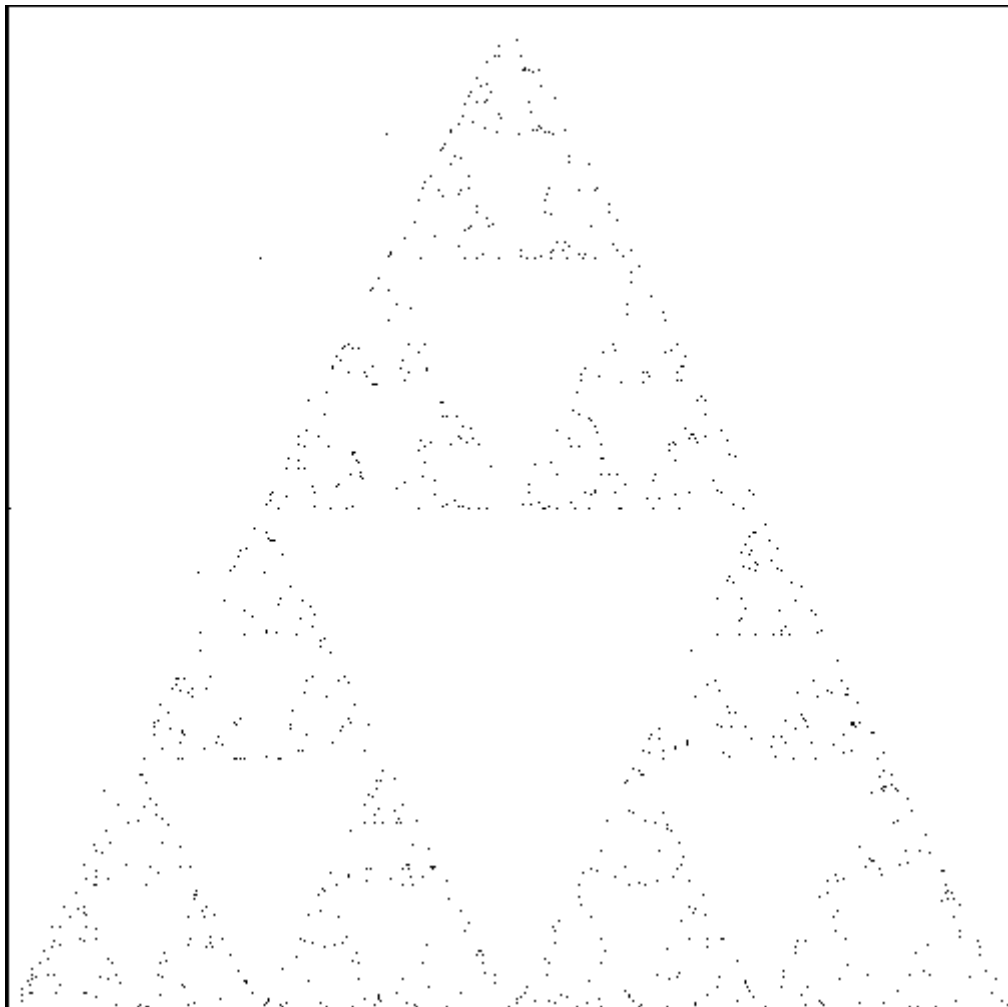


Рис. 3.13 Фрактальне зображення (трикутник Серпінського), з якістю $Q < 0.15$

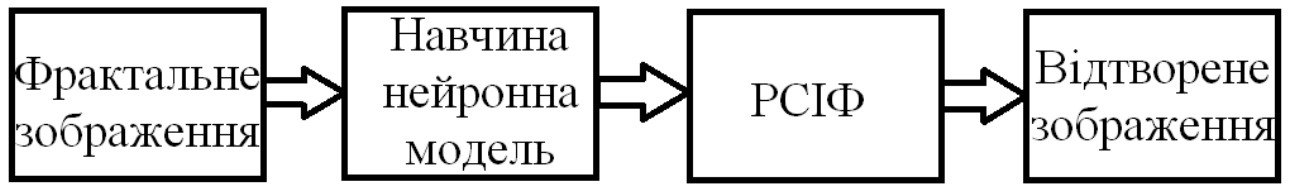


Рис.3.14 Схема роботи навченої нейронної мережі

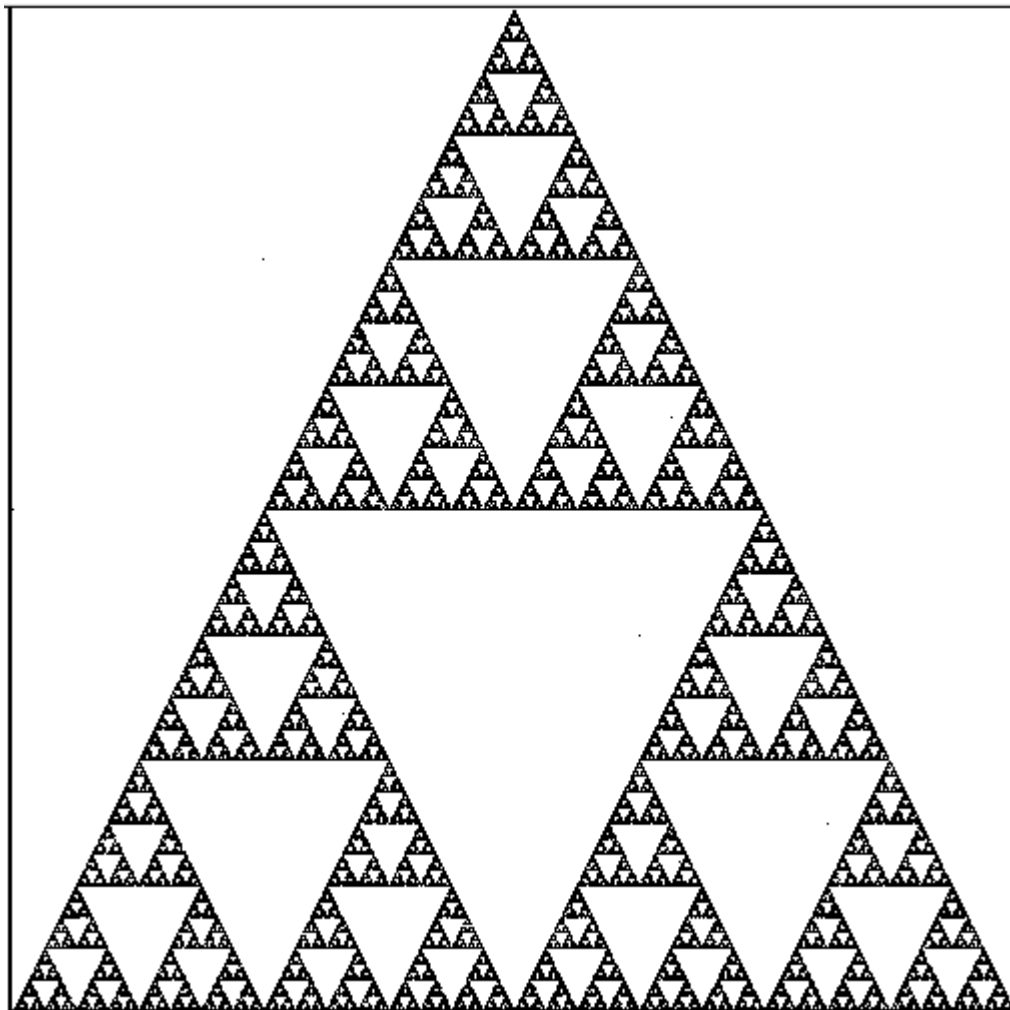


Рис. 3.15 Результати роботи навченої нейронної мережі (трикутник Серпінського з якістю $Q=0.95$)

Розпізнавання об'єктів є однією з актуальних задач в галузі інформаційних технологій. Розпізнавання самоподібних об'єктів (фрактальних структур) є однією з складних задач, це зумовлено тим, що зібрати (сформувати) дані (фрактали) дуже складно, оскільки побудова фрактальних структур потребує

значних інформаційних обчислень, ресурсів. Для вирішення цієї задачі використано РСІФ, так як немає необхідності зберігати великі масиви даних у пам'яті. Наведений алгоритм побудови не використовує рекурсивних функцій та входження циклу в цикл, що дозволяє не затрачати великих обчислювальних потужностей, і є досить оптимізованим. Тому цей алгоритм зручно використовувати на комп'ютерах з обмеженими ресурсами. Побудова фракталів за допомогою РСІФ включає в себе розрахунок на кожному кроці координати однієї точки для виведення її на екран. Кількість операцій роботи алгоритму напряму залежить від кількості пікселів фрактального зображення. Результати роботи алгоритму показали, що навчена нейронна мережа якісно розпізнає фрактальні зображення і перетворює їх в РСІФ. Отримана матриця РСІФ дозволяє покращувати якість фрактального зображення, перетворює растрову графіку у векторну, що свою чергу дозволить змінювати розміри зображення без втрати якості, також цей алгоритм зменшує розмір зображення до набору коефіцієнтів.

В майбутньому цю навчену нейронну мережу, як бібліотеку, можна буде інтегрувати в системи розпізнавання образів, в графічні редактори.

3.2 Захист документів за допомогою фрактальних зображень, сформованих рандомізованою системою ітераційних функцій (РСІФ)

Захист документів у сучасному цифровому світі досить складна та непроста задача, яка потребує все нових і нових складних елементів захисту. Побудова фрактальних зображень являє собою непросту задачу, так як потребує необхідності зберігати великі масиви даних у пам'яті та складних математичних обчислень. Перевірити, чи фігура є фракталом, не складає проблем, для цього достатньо порівняти першу і другу ітерації фрактального зображення. А якщо фрактальне зображення подати у вигляді шифру цифр, це дасть можливість здійснювати подвійну перевірку номера документа. У разі використання РСІФ немає необхідності зберігати великі масиви даних у

пам'яті, початковий набір становить лише одну точку. Алгоритм побудови не потребує значних обчислень, на кожному кроці розраховуються координати однієї точки для виведення її на екран. Перевірка документа включає в себе поділ документа на частини, що не є складною задачею, та порівняння частин з шаблонами. Даний алгоритм дозволяє перевіряти першу і другу ітерації фракталу з шаблоном, це дасть можливість верифікувати номер документа [116].

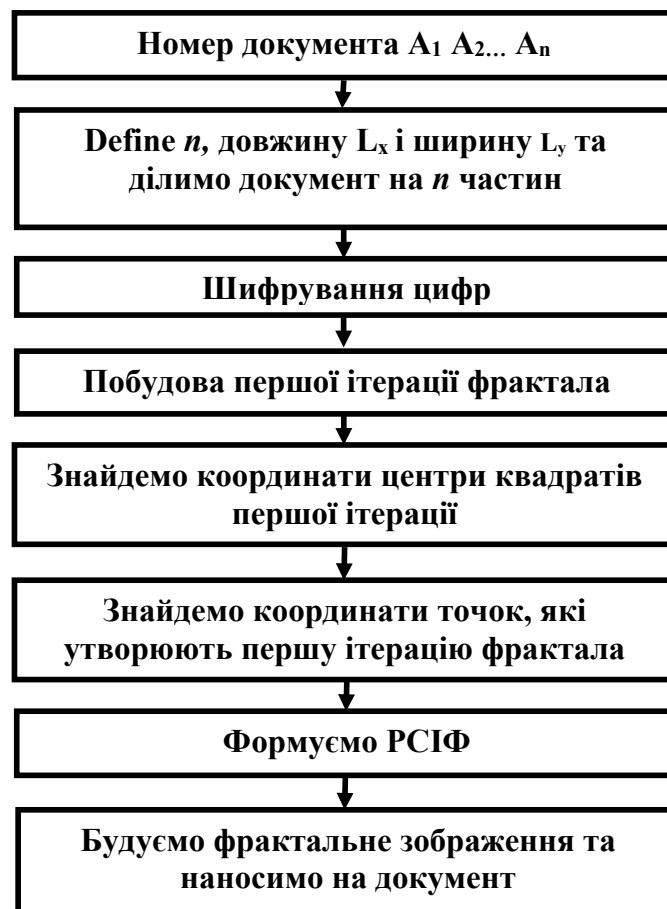


Рис.3.16. Алгоритм побудови фрактального зображення за допомогою РСІФ по заданому номеру документа

Алгоритм побудови фрактального зображення за допомогою РСІФ по заданому номеру документа (Рис. 3.16).

Для побудови фрактального зображення будемо використовувати серію документа, для нашого прикладу використаємо 9-значні цифрові номери (кількість цифр може бути різною).

Виділимо на документі прямокутну область, в яку помістимо стільки квадратів, скільки цифр у серії документа, так, щоб вона була повністю заповнена (Рис 3.17.) [116]:

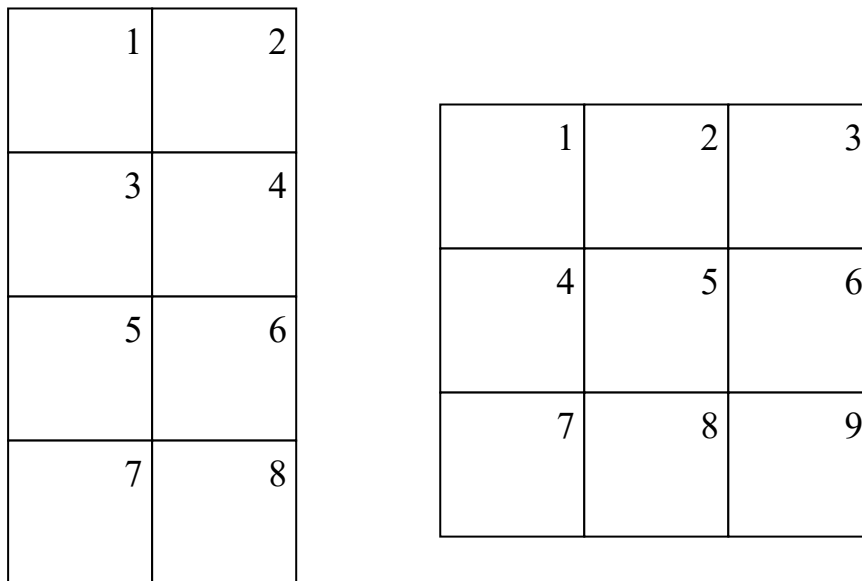


Рис.3.17 Приклади виділеної прямокутної області, розбитої на квадрати (восьмизначний та дев'ятизначний номери документа)

На наступному кроці розділимо кожний квадрат на 9 однакових квадратів (Рис. 3.18):

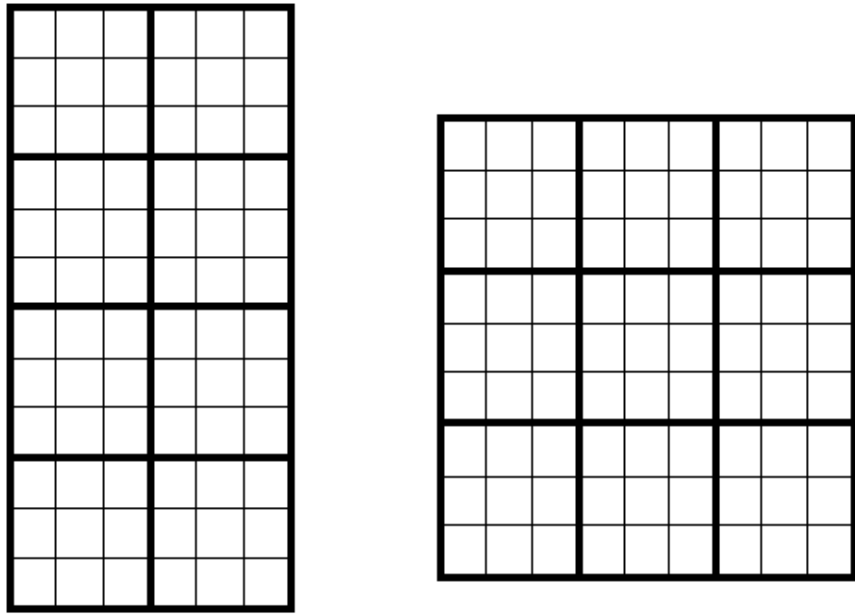


Рис. 3.18. Поділ квадратів на 9 частин

Кожну цифру від 1 до 9 представимо у вигляді замальованого квадрата (таблиця 3.1), відповідно до схеми (рис 3.19).

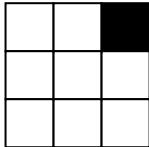
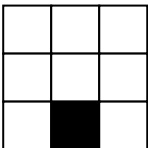
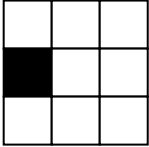
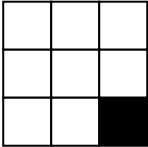
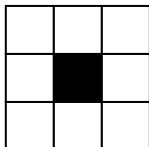
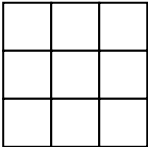
1	2	3
4	5	6
7	8	9

Рис. 3.19 Схема відповідності цифри до квадрата

Таблиця 3.1

Відображення цифр

Відображення цифри	Цифра	Відображення цифри	Цифра
	1		6
	2		7

	3		8
	4		9
	5		0

Відображення цифр можна буде змінити по бажанню, наприклад, для різних типів документів.

Для побудови фрактала використаємо удосконалену рандомізовану систему ітераційних функцій РСІФ (2.26), для кожного замальованого квадрата, який являтиме собою першу ітерацію фрактала.

Розглянемо на прикладі, як реалізувати даний спосіб захисту документів [116]:

1) Для цього випадковим способом виберемо дев'ятизначний номер документа:

157702869

2) Наступним кроком побудуємо першу ітерацію фракталу відповідно до таблиці 3.1, та серії документа 157702869, (0 – не відображається квадратом):

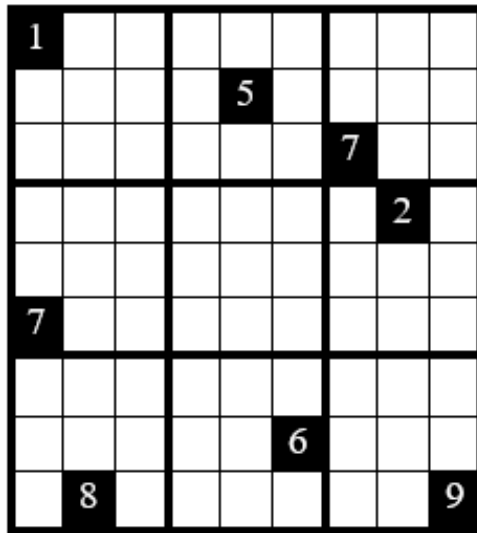


Рис.3.20. Відображення першої ітерації фракталу відповідно до серії 157702869

3) Виберемо довжину L_x і ширину L_y квадрата області по **500** пікселів, координата x та y визначатимуться відповідно до пікселів зображення

4) Знайдемо координати центрів квадратів першої ітерації (таблиця 3.2):

Таблиця 3.2

Центри квадратів першої ітерації відповідно до серії 157702869

Цифра серії	1	5	7	7	0	2	8	6	9
Координати центру	(28;28)	(83;250)	(139;361)	(306;28)	-	(194;417)	(472;83)	(417;306)	(472;472)

5) Координати, які вибираються рандомізованим алгоритмом відповідно до серії 157702869 результати зведемо в таблицю 3.3:

Таблиця 3.3

Координати які вибираються рандомізованим алгоритмом відповідно до серії 157702869

Цифра серії	1	5	7	7	0	2	8	6	9
Координати, які вибираються рандомізованим алгоритмом	(0;0)	(55;222)	(111;333)	(278;0)	-	(166;389)	(444;55)	(389;278)	(444;444)

б) Формуємо таблицю ітераційних функцій, використовуючи формулу (2.26), результати зведемо в таблицю 3.4:

Таблиця 3.4

РСІФ для побудови фракталу відповідно до серії 157702869

Цифра серії	РСІФ
1	$x = 0 - (0 - x)/9$ $y = 0 - (0 - y)/9$
5	$x = 55 - (55 - x)/9$ $y = 222 - (222 - y)/9$
7	$x = 111 - (111 - x)/9$ $y = 333 - (333 - y)/9$
7	$x = 278 - (278 - x)/9$ $y = 0 - (0 - y)/9$
0	-
2	$x = 166 - (166 - x)/9$ $y = 389 - (389 - y)/9$
8	$x = 444 - (444 - x)/9$ $y = 55 - (55 - y)/9$
6	$x = 389 - (389 - x)/9$ $y = 278 - (278 - y)/9$
9	$x = 444 - (444 - x)/9$ $y = 444 - (444 - y)/9$

7) Виконуємо РСІФ та отримуємо результат (Рис. 3.21) [116]:

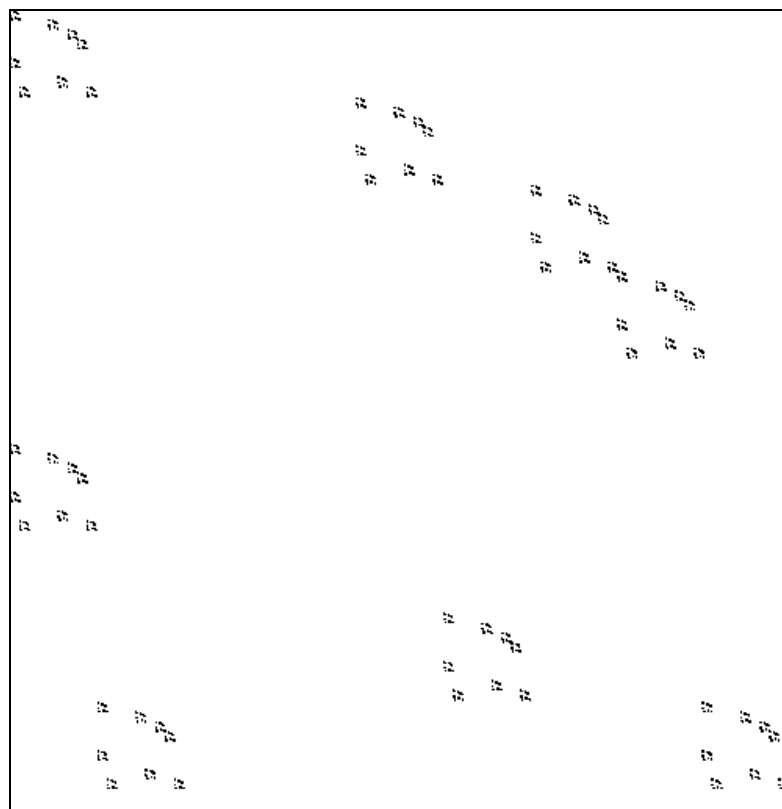


Рис. 3.21 Побудований фрактал відповідно до серії 157702869

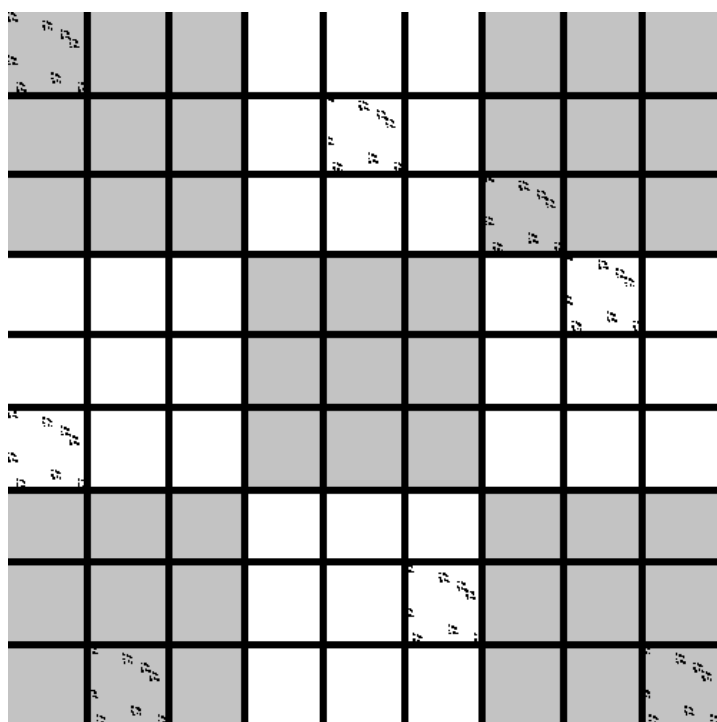
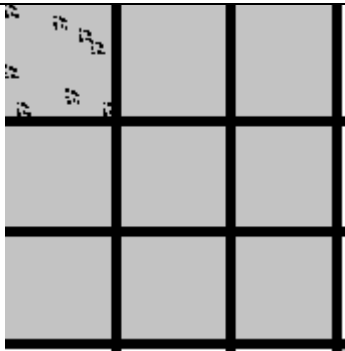


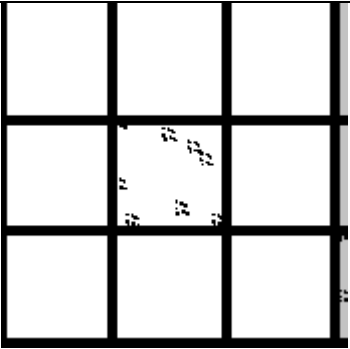
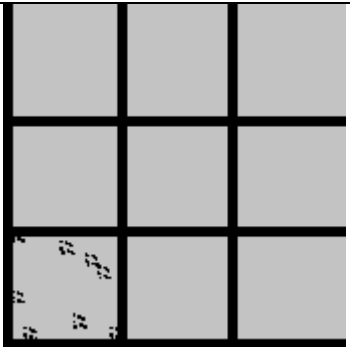
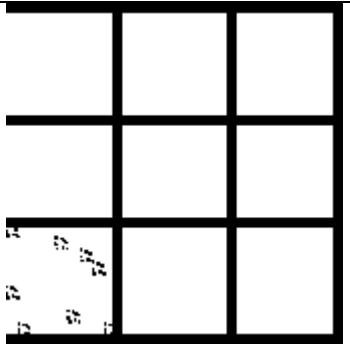
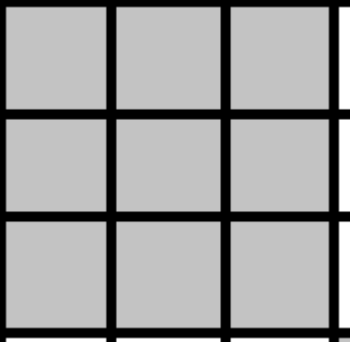
Рис. 3.22 Поділ фрактального зображення на частини

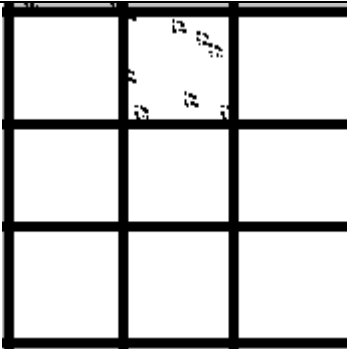
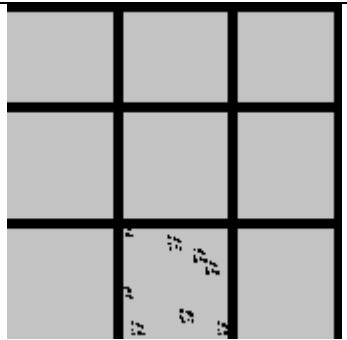
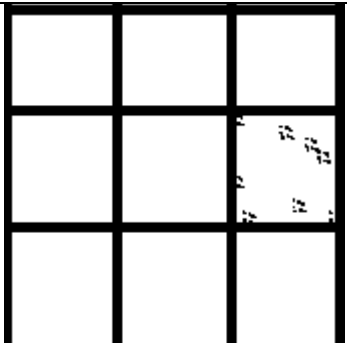
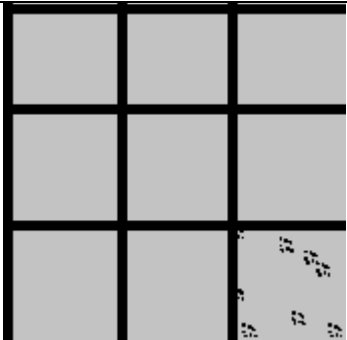
Щоб розшифрувати зображення та визначити його серії, необхідно виконати поділ зображення на 81 частину відповідно до Рис. 3.22, в результаті ми можемо переділити серію документа, використовуючи таблицю 3.1 (Рис. 8), результат зведемо в таблицю 3.5 [116]:

Таблиця 3.5

Відповідність одиниці серії до зображення

Зображення	Відповідна одиниця серії
	1

Зображення	Відповідна одиниця серії
	5
	7
	7
	0

Зображення	Відповідна одиниця серії
	2
	8
	6
	9

Як ми бачимо з Рис. 3.22, розшифроване значення відповідає початковому значенню **157702869**.

Аналогічно проведений алгоритм по другій ітерації повинен дати такий самий результат Рис. 3.23:

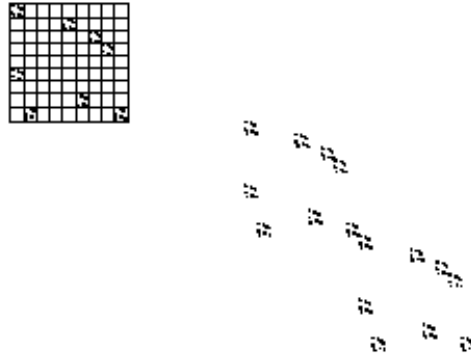


Рис. 3.23 Поділ сегмента другої ітерації фрактального зображення на частини

Як ми бачимо з Рис. 3.23, розшифроване значення відповідає початковому значенню **157702869**, таким чином відбувається подвійна верифікація серії документа.

Графічні побудови у вигляді фрактальних зображень підвищують ефективність захисту документів. Вони дозволяють поєднати серію документа із зображенням на ньому. Наведений алгоритм є гнучким, що дозволяє йому видозмінюватись відповідно до розмірів та типів документа. Подвійна верифікація та простий спосіб аналізу ускладнюють можливість підробки документу. Даний підхід практичний для виготовлення великої кількості бланків, так як швидкість формування зображення за допомогою системи РСІФ є досить значна відносно детермінованих систем ітераційних функцій побудови фрактальних зображень. При збільшенні роздільної здатності є можливість проводити потрібну верифікацію документа по третій ітерації.

3.3 Модель швидкого розрахунку фрактальної розмірності зображення з використанням нейронних мереж та удосконаленого методу побудови фрактальних структур.

Фрактальна розмірність - це показник, який описує складність фрактала, тобто геометричної форми, яка має нерегулярну структуру та складність.

Основне застосування фрактальної розмірності полягає в тому, що вона дає змогу вимірювати складність форми та структури об'єктів, які не можна описати звичайними геометричними формулами. Цей показник може бути застосований у різних галузях.

Розрахунок фрактальної розмірності є складним процесом через декілька причин:

1. Складність самої концепції фракталів: фрактали - це геометричні структури, які мають багато рівнів деталізації та самоподібності на кожному рівні. Ця складність потребує спеціальних методів та технік для розуміння та аналізу фрактальних структур.
2. Нестандартність форм: фрактали мають нестандартну форму та складну геометрію. Традиційні методи вимірювання розміру не можуть бути застосовані для фрактальних структур, що потребує розробки спеціальних методів та алгоритмів.
3. Велика кількість даних: для розрахунку фрактальної розмірності необхідно зібрати значну кількість даних, що може бути дуже часомірним та затратним процесом.
4. Обробка даних: після збору даних необхідно їх обробити та аналізувати, що також може бути складним та затратним процесом.
5. Залежність від контексту: фрактальна розмірність залежить від контексту та параметрів аналізу, таких як роздільна здатність, масштабування та рівень деталізації. Це може призводити до різних значень фрактальної розмірності для однієї і тієї ж структури залежно від використаних параметрів.
6. Обмеження обчислювальної потужності: деякі фрактальні структури дуже великі та складні, що може вимагати значних обчислювальних ресурсів та потужності. Це може бути проблемою для багатьох дослідників та вимагати використання спеціальних методів обробки даних та високопродуктивних обчислювальних систем.

Усі ці проблеми та виклики роблять розрахунок фрактальної розмірності складним та потребують спеціальних знань та навичок в області фракталів та обробки даних. Отже визначення фрактальної розмірності являє собою дуже складний процес. Запропонована методика розрахунку з використанням нейронних мереж суттєво пришвидшує процес розрахунку фрактальної розмірності.

Фрактальна розмірність визначається за формулою (3.39):

$$D = \log_K N, \quad (3.39)$$

де D – фрактальна розмірність; N – кількість самоподібних фігур першої ітерації фракталу; K – коефіцієнт пропорційності;

Для трикутника та квадрата Серпінського D є рівним (Рис 3.24):

$$D = \log_2 3 \text{ та } D = \log_3 8$$

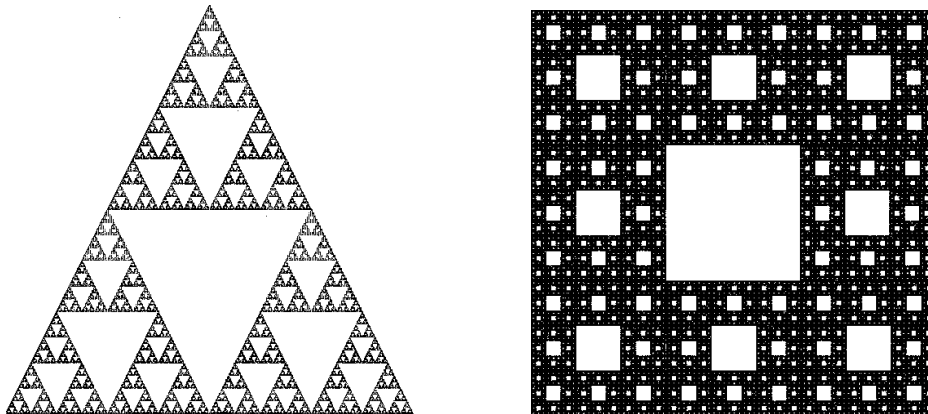


Рис. 3.24. Трикутник та квадрат Серпінського

Формула (3.39) застосовується тільки тоді, коли фігури першої ітерації є однаковими і пропорційними K . Спробуємо розрахувати фрактальну розмірність при різних розмірах фігур першої ітерації фракталу. Для цього подаємо параметр N як суму співвідношень площ фігур відносно еталонної площі фігури (найбільшої площі фігури першої ітерації) S_K (3.40):

$$S_K = \frac{S}{K^2}, \quad (3.40)$$

де S_K – площа еталонної фігури; S – площа фрактального зображення; K – коефіцієнт пропорційності;

Використовуючи формулу (3.40), N можна подати наступним чином (3.41):

$$N = \frac{S_1}{S_K} + \frac{S_2}{S_K} + \frac{S_3}{S_K} + \dots + \frac{S_n}{S_K} = \frac{1}{S_K} \cdot \sum_{i=1}^n S_i, \quad (3.41)$$

де S_K – площа еталонної фігури; n – кількість фігур першої ітерації;

Виведемо формулу розмірності фрактального зображення, використовуючи формули (3.39) та (3.41), отримаємо (3.42):

$$D = \log_K \frac{1}{S_K} \cdot \sum_{i=1}^n S_i = \log_K \frac{K^2}{S} \cdot \sum_{i=1}^n S_i = \log_K K^2 \cdot \frac{S_{f1}}{S}, \quad (3.42)$$

де D – фрактальна розмірність; K – коефіцієнт пропорційності; S – площа фрактального зображення; S_{f1} – сума площ фігур першої ітерації;

Алгоритм розрахунку фрактальної розмірності зображення представлено на Рис. 3.25:

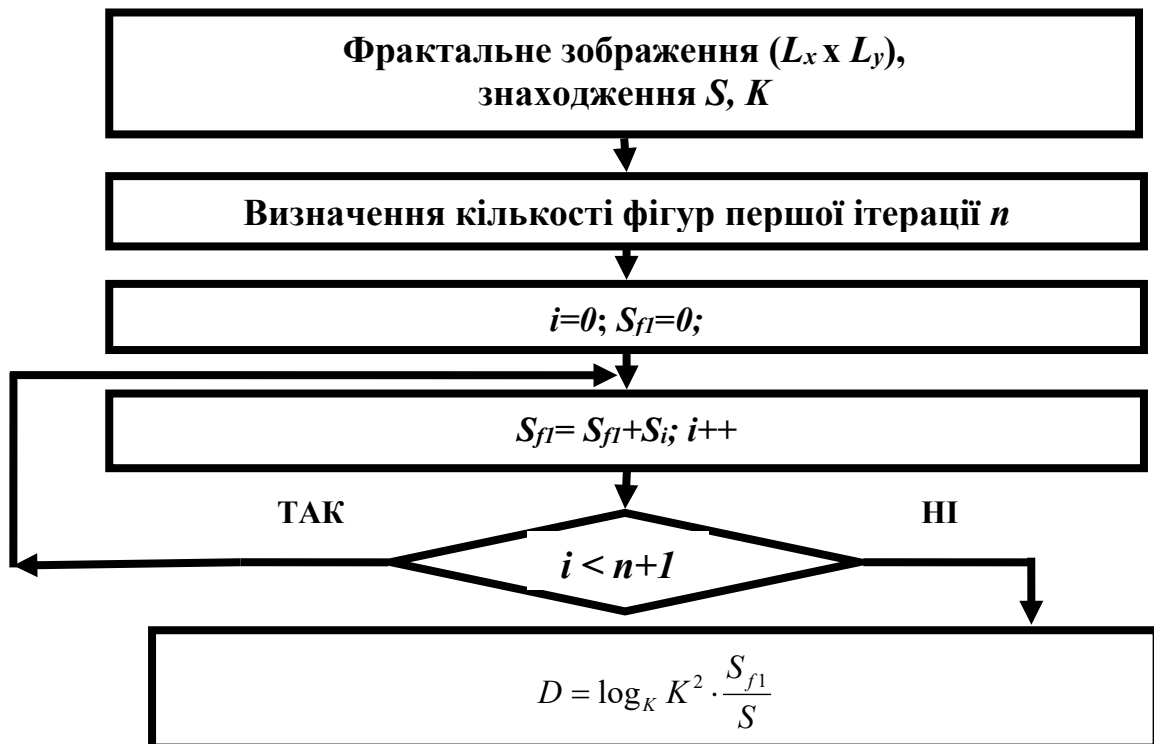


Рис. 3.25. Алгоритм знаходження фрактальної розмірності зображення

Формула (3.42) не дає можливості точно визначити фрактальну розмірність зображення, якщо фігури першої ітерації будуть накладатися одна на одну, дана формула буде давати завищений коефіцієнт, так як площі фігур будуть перетинатися. Для того, щоб уникнути цієї неточності, внесемо поправку у суму площ фігур першої ітерації S_{f1} (3.43):

$$S_{f1} = \sum_{i=1}^n S_i - S_p, \quad (3.43)$$

де S_p – сума площ перетених зображень першої ітерації;

Тепер наша задача визначення фрактальної розмірності зводиться до пошуку сум площ перетених зображень першої ітерації S_p . Щоб вирішити цю задачу, нам потрібно мати математичну модель фрактального зображення, це можна зробити за допомогою удосконаленої систем ітераційних функцій РСІФ. РСІФ дозволять нам побудувати фрактальні зображення і дасть нам параметри функцій, за допомогою яких ми зможемо обчислювати S_p

Для початку опишемо відомі методи пошуку коефіцієнтів РСІФ, які використовуються для побудови фрактальних зображень:

1. Експериментальний підхід: цей метод полягає в емпіричному визначенні коефіцієнтів на основі візуальної оцінки результатів побудови фрактальних зображень. Дослідник виконує ітерації з різними значеннями коефіцієнтів і оцінює отримані зображення, підбираючи коефіцієнти, які забезпечують потрібний рівень деталізації та складності зображення.
2. Метод апроксимації: цей метод використовується для визначення коефіцієнтів, які найкращим чином відповідають конкретному фракталу, відображаючи його побудову на основі апроксимації точок. Дослідник виконує ітерації з різними значеннями коефіцієнтів та порівнює отримані зображення з оригінальним фракталом. Коефіцієнти,

що найбільш точно апроксимують оригінальний фрактал, вважаються оптимальними.

3. Аналітичний підхід: цей метод використовується для точного визначення коефіцієнтів, які забезпечують потрібний рівень деталізації та складності зображення. Для цього дослідник використовує математичний аналіз та теорію фракталів для розрахунку оптимальних значень коефіцієнтів.
4. Генетичний алгоритм: цей метод використовує ідеї еволюції для визначення оптимальних коефіцієнтів. Дослідник створює популяцію різних значень коефіцієнтів та виконує ітерації з кожним з них. Кожен ітераційний процес оцінюється за допомогою певної функції придатності. Найбільш успішні ітерації використовуються для створення нової популяції коефіцієнтів. Цей процес повторюється до тих пір, поки не буде досягнуто оптимальних значень коефіцієнтів.

Кожний з цих методів є складним у пошуку коефіцієнтів РСІФ, потребує складних наближених обчислень, для різних фракталів не дає загального алгоритму пошуку коефіцієнтів. Результатом виконання цих методик є СІФ (Наприклад, побудови трикутника чи квадрата Серпінського) у вигляді:

$$\left[\begin{array}{l} \left\{ \begin{array}{l} x_n = a_1 + b_1 x_{n-1} \\ y_n = d_1 + e_1 x_{n-1} \end{array} \right. \\ \dots \\ \left\{ \begin{array}{l} x_n = a_2 + b_2 x_{n-1} \\ y_n = d_2 + e_2 x_{n-1} \end{array} \right. \end{array} \right. , \quad (3.44)$$

де $(a_1, b_1, d_1, e_1, \dots, a_2, b_2, d_2, e_2)$ – коефіцієнти ітераційних функцій, які визначають вигляд фігур першої ітерації.

Дана система ітераційних функцій, є мало інформаційною, так як не дозволяє визначити центри фігур першої ітерації фракталу, ширину і довжину,

коефіцієнти пропорційностей, що, в свою чергу, не дає можливості визначити площу перетину фігур першої ітерації S_p

Для вирішення цієї проблеми пропонується удосконалена РСІФ:

$$\left\{ \begin{array}{l} x_n = x_a - \frac{x_a - x_{n-1}}{k_{x_a}} \\ y_n = y_a - \frac{y_a - y_{n-1}}{k_{y_a}} \\ \dots \\ x_n = x_b - \frac{x_b - x_{n-1}}{k_{x_b}} \\ y_n = y_b - \frac{y_b - y_{n-1}}{k_{y_b}} \end{array} \right. , \quad (3.45)$$

де (x_n, y_n) - поточні координати точки, яка покроково будує фрактал; $(x_a, y_a) \dots (x_b, y_b)$ - координати точок, що утворюють перші ітерації фракталу; $k_{x_a}, k_{y_a} \dots k_{x_b}, k_{y_b}$ - коефіцієнти пропорційності відповідних фігур першої ітерації.

Для визначення коефіцієнтів пропорційності відповідних фігур першої ітерації $k_{x_a}, k_{y_a} \dots k_{x_b}, k_{y_b}$ використаємо наступну формулу:

$$k = \frac{L}{l}, \quad (3.46)$$

де k - коефіцієнт пропорційності відповідної фігури першої ітерації фракталу; L - початкова довжина або ширина фрактального зображення; l - початкова довжина або ширина зображення відповідної фігури першої ітерації фракталу.

Для визначення координати точок що утворюють перші ітерації фракталу, (x_a, y_a) , використаємо центри фігур першої ітерації X_a, Y_a :

$$x_a = \frac{k_{x_a} X_a - \frac{L}{2}}{k_{x_a} - 1}; \quad y_a = \frac{k_{y_a} Y_a - \frac{L}{2}}{k_{y_a} - 1} \quad (3.47)$$

Для визначення кількості операцій для побудови фрактального зображення за допомогою удосконаленої рандомізованої системи ітераційних функцій РСІФ скористаємося наступною формулою:

$$N_I = L^2 \cdot \left(\frac{n}{K^2} \right)^{\log_K L} \quad (3.48)$$

де N_I – кількість операцій для побудови фрактального зображення за допомогою удосконаленої РСІФ; K - коефіцієнт пропорційності; n - кількість самоподібних фігур першої ітерації; L - початкова довжина і ширина зображення.

Аналіз систем ітераційних функцій показав ефективність використання удосконаленої рандомізованої системи ітераційних функцій РСІФ над детермінованою системою ітераційних функцій ДСІФ. Удосконалена РСІФ дозволить сформулювати систему ітераційних функцій та за допомогою неї побудувати фрактальне зображення (3.45).

Для знаходження площі перетину двох фігур першої ітерації розглянемо рис.3.26:

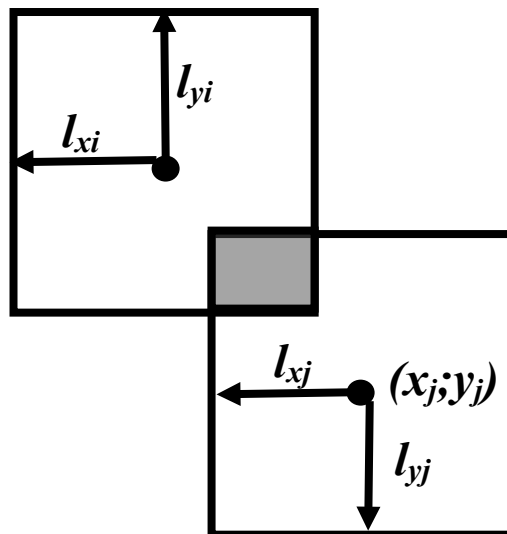


Рис.3.26 Перетин фігур першої ітерації фрактального зображення

На рис. 3.26 $(x_i; y_i)$ та $(x_j; y_j)$ - координати центрів фігур першої ітерації, l_{xi} та l_{yi} – половини сторін фігур першої ітерації, які можна визначити за формулами (3.49-3.50):

$$l_{xi} = \frac{L_x}{2K_{xi}}, \quad (3.49)$$

$$l_{yi} = \frac{L_y}{2K_{yi}}, \quad (3.50)$$

де l_{xi} та l_{yi} – половини сторін фігур першої ітерації; L_x та L_y - довжина і ширина фрактального зображення; K_{xi} та K_{yi} – коефіцієнти пропорційності відповідної ітерації.

Ширину і довжину площі перетину двох фігур першої ітерації можна знайти за наступними математичними виразами (3.51) та (3.52):

$$l_{x_{i-j}} = |l_{xi} + l_{xj}| - |x_i - x_j|, \quad \text{за умови } l_{x_{i-j}} > 0 \quad (3.51)$$

$$l_{y_{i-j}} = |l_{yi} + l_{yj}| - |y_i - y_j|, \quad \text{за умови } l_{y_{i-j}} > 0 \quad (3.52)$$

де l_{xi} та l_{yi} – половини сторін фігур першої ітерації; $(x_i; y_i)$ та $(x_j; y_j)$ - координати центрів фігур першої ітерації.

Якщо дві умови (3.51) і (3.52) виконуються, можна визначити площу перетину двох фігур першої ітерації (3.53):

$$S_{p_{i-j}} = l_{x_{i-j}} \cdot l_{y_{i-j}}, \quad (3.53)$$

Запишемо суму площ фігур першої ітерації $S_{f1} S_p$ (3.54):

$$S_{f1} = \sum_{i=1}^n S_i - \sum_{k=1}^n S_{p_k}, \quad (3.54)$$

У формулу обрахунку розмірності фракталу (3.42) підставимо вираз (3.54) і отримаємо наступний вираз (3.55):

$$D = \log_K K^2 \cdot \frac{\sum_{i=1}^n S_i - \sum_{k=1}^n S_{p_k}}{S}, \quad (3.55)$$

Алгоритм розрахунку фрактальної розмірності представлений на рис. 3.27:

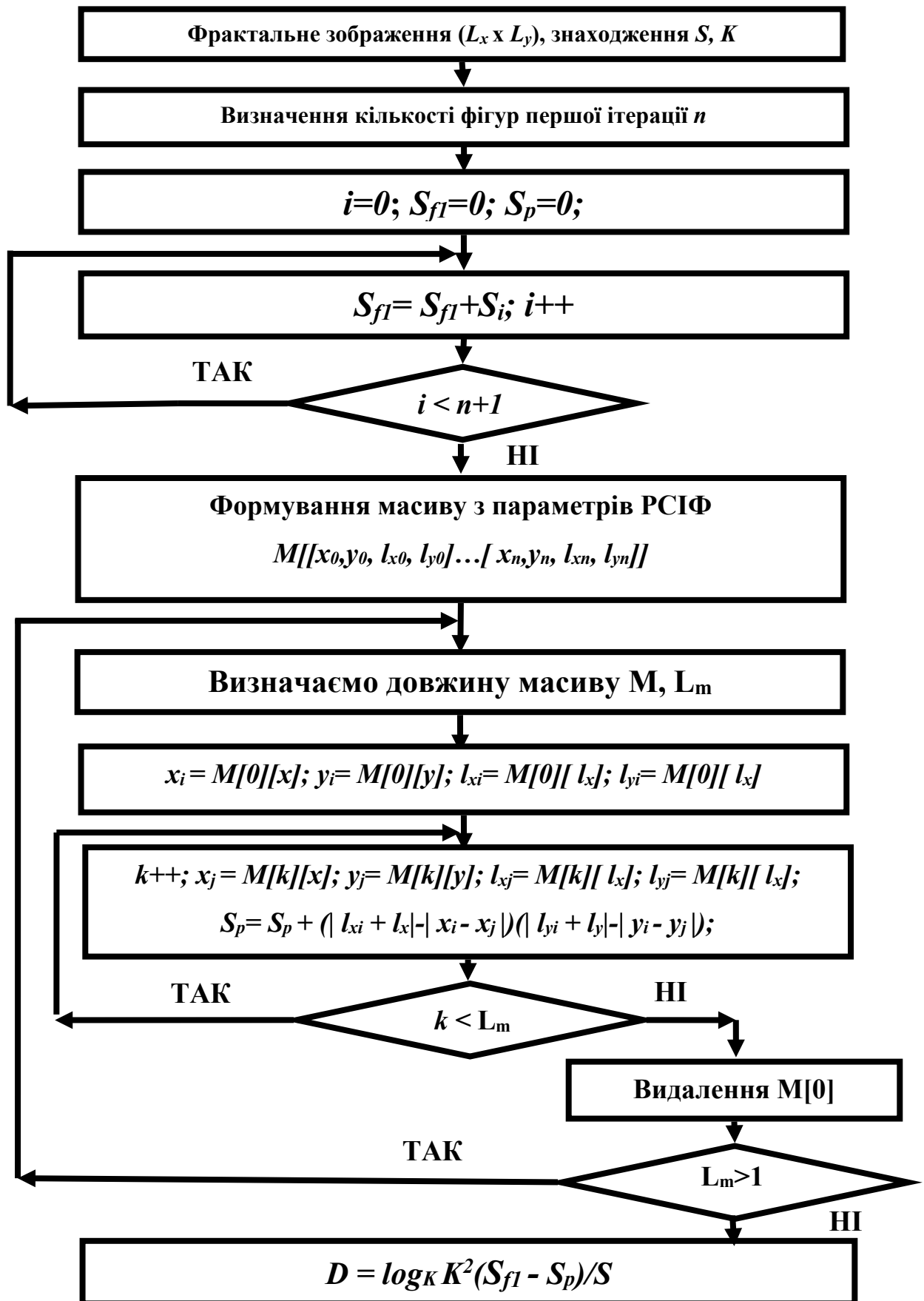


Рис. 3.27. Алгоритм знаходження фрактальної розмірності зображення

Програмна реалізація алгоритму. Для навчання нейронної мережі визначати розмірність фрактального зображення необхідно згенерувати фрактал із заданою удосконаленою РСІФ. Вхідними даними будуть виступати фрактальні зображення (пікселі зображення), вихідними даним будуть виступати параметри удосконаленої РСІФ по яких будемо здійснювати обчислення фрактальної розмірності (3.55) (Рис.3.28).

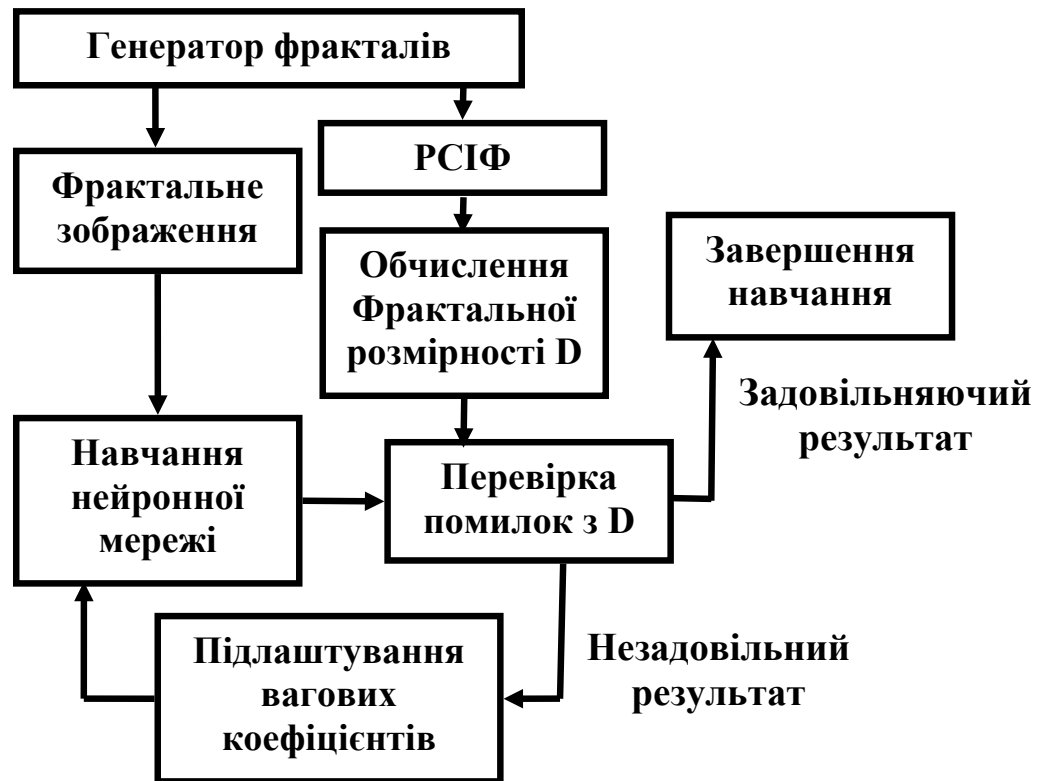


Рис.3.28 Алгоритм навчання нейронної мережі для визначення фрактальної розмірності

Для програмної реалізації нейронної мережі будемо використовувати мову програмування JavaScript та технології WEB-програмування для візуалізації, для JavaScript є готові бібліотеки для навчання нейронної мережі brain.js, схема програмної реалізації зображена на Рис. 3.29:

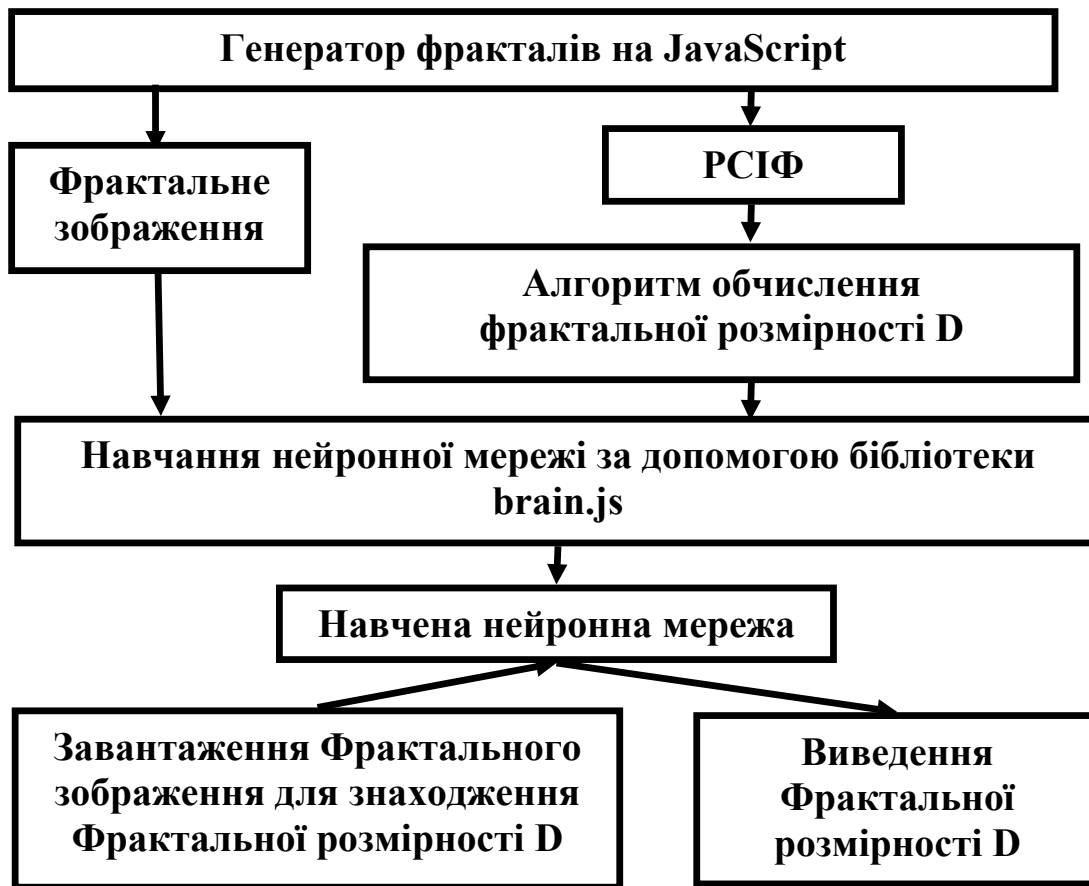


Рис.3.29 Програмна реалізація нейронної мережі на базі бібліотеки Brain.js

Результати практичної реалізації. Після написання коду програми алгоритму знаходження фрактальної розмірності зображення і навчання нейронної мережі перевіримо результати роботи.

Після завантаження в програму квадрату Серпінського рис. 3.24 нейронна мережа видала нам результат **1,893155286**, порівняємо його з результатом розрахунку $\log_3 8 = 1,89278926$, похибка обчислення нейронною мережею складає менше 1%.

Для більш наочного оцінювання роботи алгоритму завантажимо модифікований квадрат Серпінського, у якого одна з фігур першої ітерації має інший розмір з іншими коефіцієнтами пропорційності рис. 3.30, а фігури першої ітерації фракталу наведені на рис. 3.31:

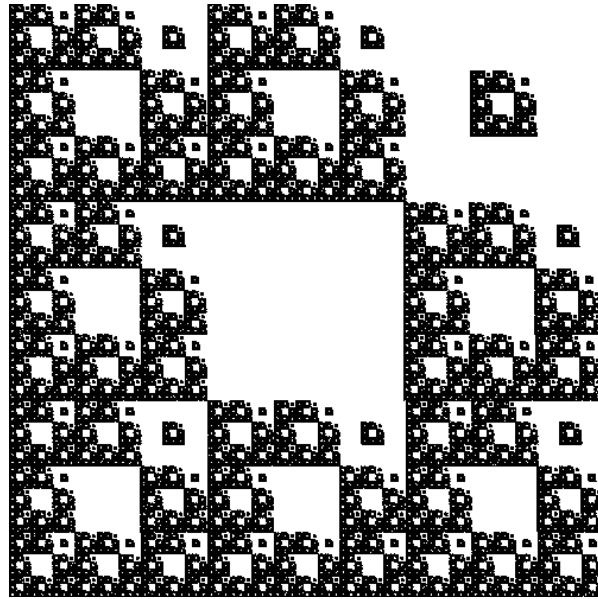


Рис. 3.30. Модифікований квадрат Серпінського

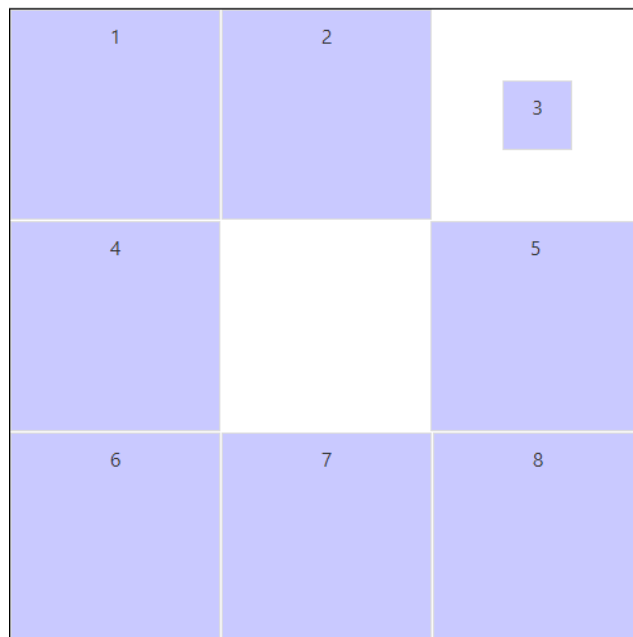


Рис. 3.31. Фігури першої ітерації модифікованого квадрату Серпінського

Нейронна мережа після завантаження модифікованого квадрату Серпінського дала нам результат **1,098525197**, порівняємо його з результатом розрахунку $\log_3(7+1/9) = \mathbf{1,098612288}$. Похибка обчислення нейронною мережею складає менше 1%, що показує досить високу точність визначення

фрактальної розмірності зображення та правильного вибору СІФ для навчання нейронної мережі.

Визначення фрактальної розмірності є складною математичною задачею, навчена нейронна мережа дозволяє нам виконувати цю задачу. Запропонований алгоритм розрахунку фрактальної розмірності дає можливість автоматизувати навчання нейронної мережі за допомогою генерації фракталів. Генерація фрактальних зображень є дуже складним процесом, так як потребує багато математичних обчислень. Аналіз систем ітераційних функцій показав, що запропонована удосконалена РСІФ є найбільш ефективною СІФ, так як її коефіцієнти ітераційних функцій дозволяють знаходити ширину, довжину, коефіцієнти пропорційності по ширині і висоті, та координати центрів фігур першої ітерації. Знаходження цих параметрів дозволяє точно визначити площі перетинів фігур, що, в свою чергу, дозволяє здійснити корекцію формули розрахунку фрактальної розмірності. Удосконалена РСІФ дозволяє сформувати масив даних, який включає в себе зображення фракталів та значення фрактальних розмірностей. Цей сформований масив даних дасть можливість сформувати автоматизовану систему для навчання нейронної мережі визначати по заданому зображенню фрактальну розмірність. Швидкість роботи удосконаленої РСІФ дозволить швидко навчити нейронну мережу, що неможливо було б зробити при ДСІФ. Навчена нейронна мережа миттєво видає значення та показує точні результати знаходження фрактальної розмірності, похибка при цьому складає не більше 1%.

3.4 Метод шифрування фрактальних зображень за допомогою матриць перетворень для захисту від дешифрування нейронними алгоритмами

На сьогоднішній день не існує повної гарантії, що зловмисники не зможуть отримати доступ до сучасних систем передачі графічної інформації. Але можна забезпечити надійний захист даних шляхом використання надійного

шифрування, яке унеможливить зловживання інформацією. Зараз більшість систем шифрування використовують афінні перетворення, функції (ітераційні функції), та системи функцій, що призводить до того, що сучасні нейронні алгоритми навчаються розшифровувати інформацію швидко. Однак, можна розробити алгоритм шифрування, який використовує ключ у вигляді набору матриць перетворень, формованих за допомогою випадкових процесів. Це унеможливить дешифрування за допомогою нейронних мереж.

Наш алгоритм шифрування включає наступні кроки, які можна виконати з зображенням [117]:

1. Зміна розміру зображення.
2. Додавання надлишкових пікселів, включаючи інші зображення.
3. Зміна розташування пікселів.
4. Зміна яскравості пікселів зображення.
5. Інвертування пікселів.
6. Зміна кольорів пікселів зображень.

Кожна з цих дій виконується як окремий крок у нашому шифрувальному алгоритмі.

Для зміни розміру зображення ми використовуємо базове зображення зі спочатку визначеними розмірами $N_{x0} \times N_{y0}$. Ми поміщаємо це зображення на білий фон з розмірами $N_x \times N_y$, з такою умовою, що N_{x0} менше ніж половина N_x , і N_{y0} менше ніж половина N_y . Крім того, на граничних лініях y_0 та x_0 ми забезпечуємо, що принаймні один піксель має не білий колір (рис. 3.32) [117].

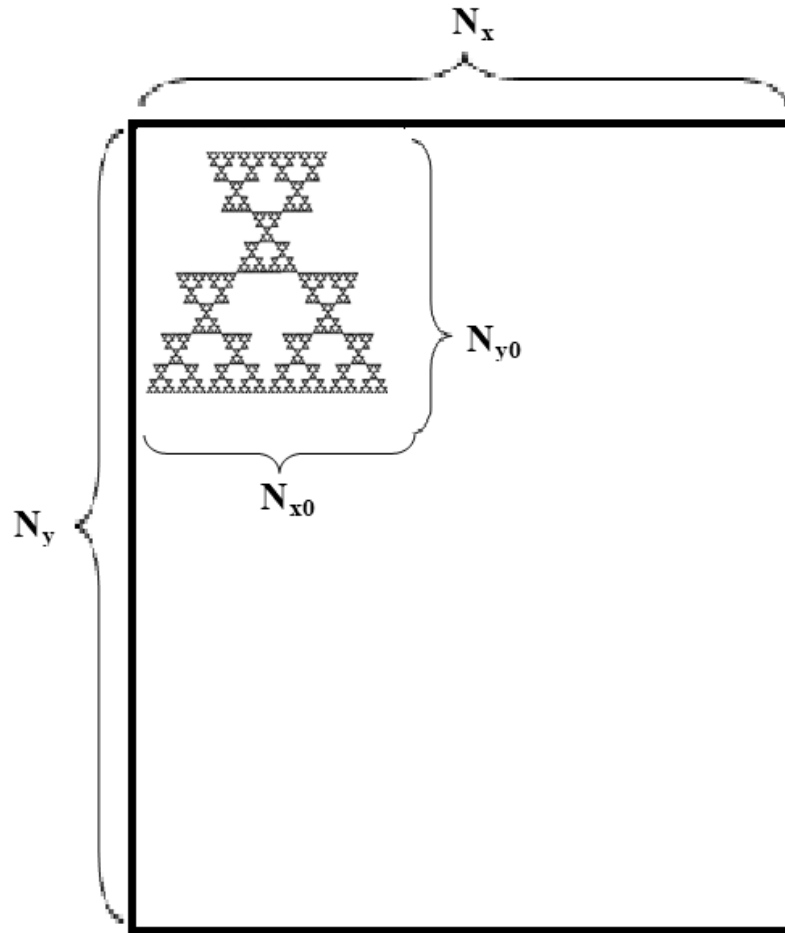


Рис. 3.32. Зміна розміру зображення

Додаємо додаткові надлишкові зображення, для того ми формуємо базу випадкових зображень з розмірами $M_x \times M_y$. При цьому, M_x дорівнює половині значення N_x , а M_y дорівнює половині значення N_y . Ми розташовуємо ці додаткові зображення таким чином, щоб вони торкалися протилежних кутів інших зображень (Рис. 3.33) [117].

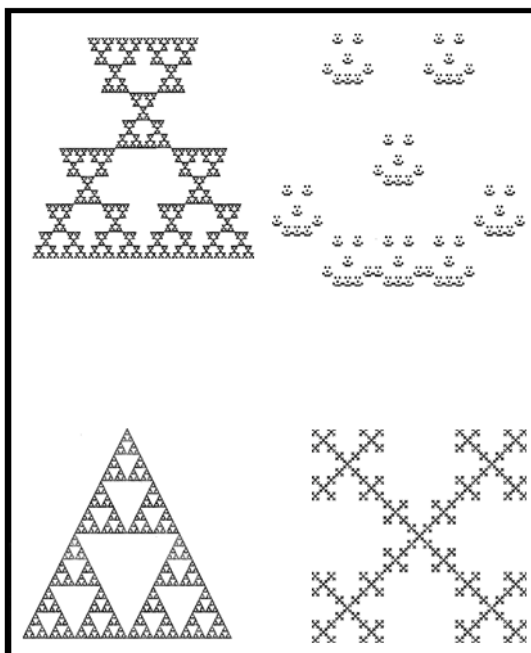


Рис. 3.33. Додавання надлишкових зображень

Змінюємо розташування пікселів шляхом створення двовимірної матриці розміром $[i, j]$, де $i = N_x - 1$, а $j = N_y - 1$. Ця матриця містить числа від 1 до $N_x \cdot N_y$, розташовані у випадковому порядку без повторень. Потім послідовно, зліва направо та зверху вниз, переносимо кожен пункт нашого зображення на нове місце, використовуючи наступне правило: $x_n = i' + 1$, $y_n = j' + 1$, де i' та j' є координатами цифри, яка відповідає порядковому номеру пікселя n (Рис. 3.34-3.35) [117].

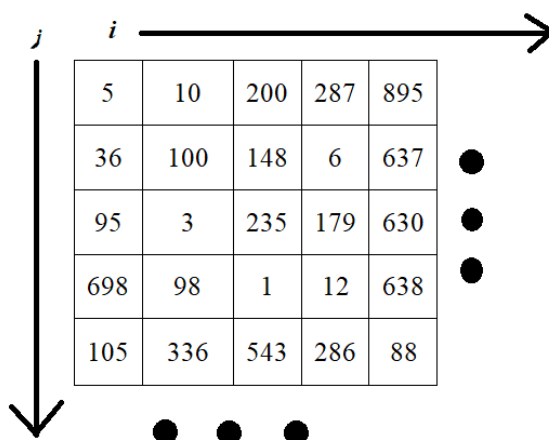


Рис. 3.34. Матриця переміщень пікселів для формування нового зображення

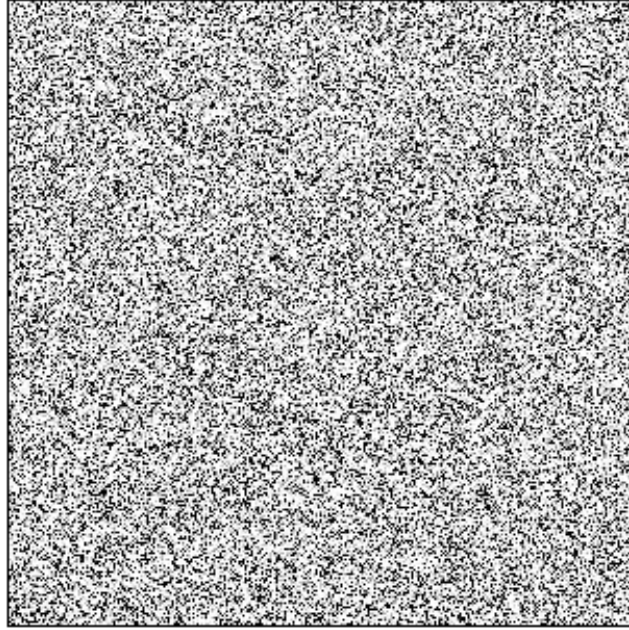


Рис. 3.35. Результати переміщення пікселів (зашифроване зображення)

Для **зміни яскравості пікселів** створюємо двовимірну матрицю розміром $[i, j]$, де $i = N_x - 1$, а $j = N_y - 1$. У цій матриці числа від 0 до 7 розташовані у випадковому порядку без повторень (рис. 3.36). Кожне число відповідає певній зміні яскравості і має таке значення [117]:

- 0 - не змінювати яскравість;
- 1 - зменшити на половину яскравість червоного кольору (R) в колірній моделі RGB;
- 2 - зменшити наполовину яскравість зеленого кольору (G);
- 3 - зменшити наполовину яскравість синього кольору (B);
- 4 - зменшити наполовину яскравість червоного і зеленого кольорів;
- 5 - зменшити наполовину яскравість червоного і синього кольорів;
- 6 - зменшити наполовину яскравість зеленого і синього кольорів;
- 7 - зменшити наполовину яскравість всіх кольорів.

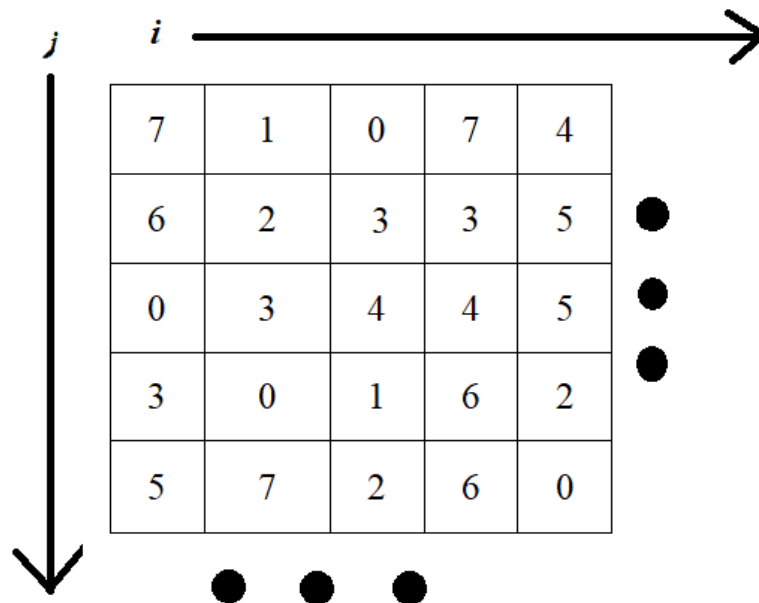


Рис. 3.36. Матриця змін яскравості зображення

Для проведення інвертації пікселів створюємо двовимірну матрицю розміром $[i, j]$, де $i = N_x - 1$, а $j = N_y - 1$. У цій матриці числа від 0 до 7 розташовані у випадковому порядку без повторень (Рис. 3.37). Кожне число відповідає певній інверсії пікселів і має таке значення [117]:

- 0 - не змінювати піксель;
- 1 - здійснити інверсію червоного кольору пікселя (R) в колірній моделі RGB;
- 2 - здійснити інверсію зеленого кольору (G);
- 3 - здійснити інверсію синього кольору (B);
- 4 - здійснити інверсію червоного і зеленого кольорів;
- 5 - здійснити інверсію червоного і синього кольорів;
- 6 - здійснити інверсію зеленого і синього кольорів;
- 7 - здійснити інверсію всіх кольорів.

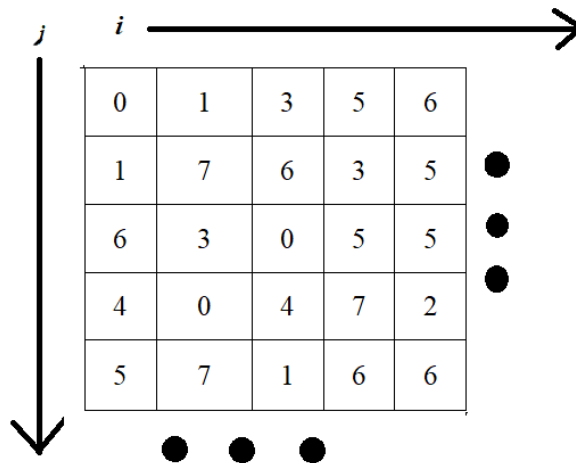


Рис. 3.37. Матриця змін інвертацій зображення

Для **зміни кольорів пікселів** також створюємо двовимірну матрицю розміром $[i, j]$, де $i = N_x - 1$, а $j = N_y - 1$. У цій матриці числа від 0 до 5 розташовані у випадковому порядку без повторень (Рис. 3.38). Кожне число відповідає певній зміні кольорів пікселів і має таке значення [117]:

0 - без змін;

1 - червоний колір змінюється на значення синього кольору в колірній моделі RGB;

2 - червоний колір змінюється на значення зеленого кольору;

3 - синій колір змінюється на значення зеленого кольору в колірній моделі RGB;

4 - червоний колір ставиться на місце синього кольору, синій колір - на місце зеленого, а зелений колір - на місце червоного;

5 - червоний колір ставиться на місце зеленого кольору, зелений колір - на місце синього, а синій - на місце червоного.

j	i	→					
↓		5	3	1	5	4	
		0	2	4	3	5	●
		4	3	0	5	5	●
		2	1	4	4	2	●
		0	2	1	0	1	
		●	●	●			

Рис. 3.38. Матриця змін кольорів зображення

Після застосування шифрування з матрицями перетворень, що створені за допомогою випадкових процесів, отримуємо результат шифрування (Рис. 3.39) [117]:

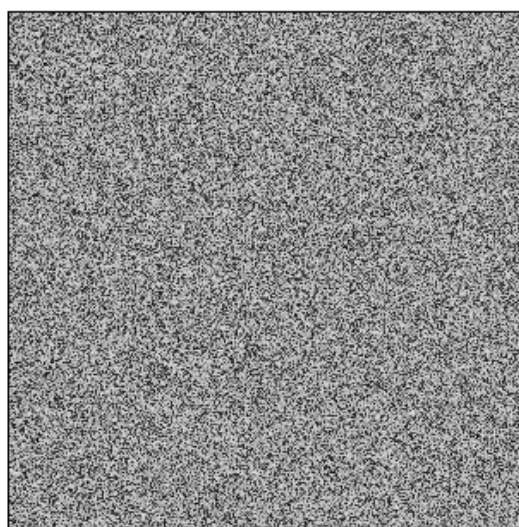


Рис. 3.39. Результат шифрування зображення

Аналіз можливих комбінацій. Оскільки всі матриці формувалися випадковим чином і мають надлишковість $3/4$, нейронна мережа повинна проходити через всі можливі комбінації [117].

Матриця зміни розташування пікселів дає нам $(N_x \cdot N_y)!$ комбінацій.

Матриця зміни яскравості пікселів дає нам $8^*(N_x \cdot N_y)!$ комбінацій.

Матриця інвертації пікселів дає нам $8 \cdot (N_x \cdot N_y)!$ комбінацій.

Матриця зміни кольору пікселів дає нам $6 \cdot (N_x \cdot N_y)!$ комбінацій.

Отже, загальна кількість комбінацій N може бути обчислена як сума цих чотирьох значень [117].

$$N = 384 \cdot ((N_x \cdot N_y)!)^4, \quad (3.56)$$

де $N_x \cdot N_y$ - кількість пікселів зображення з надлишковістю, при цьому виконуються умови $N_{x0} < N_x/2$ і $N_{y0} < N_y/2$. Таким чином, кількість комбінацій N відповідає вимогам [117]:

$$N \approx 384 \cdot (4! \cdot (N_{x0} \cdot N_{y0})!)^4 = 128065536 \cdot ((N_{x0} \cdot N_{y0})!)^4, \quad (3.57)$$

Давайте обчислимо кількість комбінацій для різної кількості пікселів на зображенні $N_{x0} \cdot N_{y0}$: 1, 10, 100, 1000. Отримані результати представимо у вигляді Таблиці 3.6 [117].

Таблиця 3.6

Кількість комбінацій

$N_{x0} \cdot N_{y0}$	N
1	128065536
10	$4,6 \cdot 10^{14}$
100	$1,19 \cdot 10^{166}$
1000	$5,1 \cdot 10^{2576}$

Декодування зображення. Спробуємо відтворити початкове зображення, застосовуючи протилежні дії, використовуючи зворотні матриці, які були сформовані (рис. 3.40) [117].

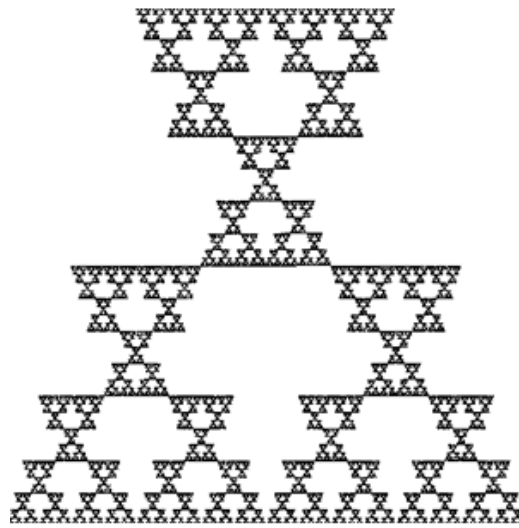


Рис. 3.40. Результат дешифрування фрактального зображення

Даний шифрувальний алгоритм має просту реалізацію і дозволяє створювати шифрувальні ключі, що є необхідними для розшифрування зображення. Алгоритм маскує розмір зображення, що забезпечує додатковий рівень захисту. Використання надлишкових елементів перешкоджає нейронним мережам у співставленні пікселів. Зміна кольорів, яскравості та інверсія з використанням випадкових процесів ускладнюють завдання по підбору функції розшифрування. Враховуючи те, що матриці формуються випадковим чином, нейронні мережі не можуть встановити шифрувальну функцію. Для розшифрування зображення з 1000 пікселів потрібно перебрати надзвичайно велику кількість комбінацій (більше $5.1 \cdot 10^{2576}$), що на сьогоднішній день не є реалістичним технічним рішенням. Такий алгоритм можна успішно застосовувати в областях військових та розвідувальних справ, а також в сферах захисту інформації. Великою перевагою даного алгоритму є можливість передавати зашифровані зображення по відкритим каналам.

Висновки до 3-го розділу

На сьогоднішній день використання фрактальних структур знаходить широке застосування в різних галузях науки і техніки. Однак наявність ефективних інструментів для моделювання таких структур є обмеженою. Розробка спеціального інструменту для побудови фрактальних структур вимагає складних математичних обчислень. Використання алгоритму визначення пікселів фрактального зображення, такого як "Cantor dust", спростить процес побудови фрактальних зображень. Цей алгоритм обчислення оптимізований і не вимагає вкладених циклів або рекурсивних функцій, що дозволяє ефективно використовувати обчислювальні ресурси. Додаткове застосування відповідних формул дозволить точно визначити кількість випадкових подій, що забезпечить високу якість відтворення зображень з використанням розширеної системи ітераційних функцій (РСІФ).

Розпізнавання фрактальних об'єктів є важливою задачею в галузі інформаційних технологій. Однак розпізнавання самоподібних об'єктів є складною задачею через складність у зборі та формуванні фракталів. Побудова фрактальних структур вимагає значних обчислювальних ресурсів. Для вирішення цієї проблеми використано удосконалену методику розширеної системи ітераційних функцій (РСІФ), яка не вимагає збереження великих масивів даних у пам'яті. Наведений алгоритм побудови фракталів не використовує рекурсивних функцій та вкладеного циклу, що дозволяє ефективно використовувати обчислювальні ресурси. Цей алгоритм є оптимізованим і зручно використовується на комп'ютерах з обмеженими ресурсами для побудови фрактальних зображень з використанням РСІФ. Він розраховує координати кожної точки на екрані на кожному кроці, уникаючи складних обчислень. Кількість операцій алгоритму прямо залежить від розміру фрактального зображення. Експерименти з алгоритмом показали, що навчена нейронна мережа здатна якісно розпізнавати фрактальні зображення і перетворювати їх у РСІФ. Отримана матриця РСІФ дозволяє покращити якість

фрактального зображення та перетворити його з растрової графіки на векторну, що дозволяє змінювати розміри зображення без втрати якості. Крім того, цей алгоритм дозволяє зменшити розмір зображення до набору коефіцієнтів.

В майбутньому цю навчену нейронну мережу як бібліотеку можна буде інтегрувати в системи розпізнавання образів, в графічні редактори.

Використання фрактальних зображень у графічних побудовах може значно підвищити ефективність захисту документів. Це дозволить поєднати серію документів з унікальними фрактальними зображеннями, що мають високу невідтворюваність. Наведений алгоритм побудови фрактальних зображень є гнучким і може адаптуватись до різних розмірів і типів документів. Використання подвійної верифікації та простого методу аналізу ускладнює можливість підробки документа. Цей підхід є практичним для масового виготовлення бланків, оскільки швидкість генерації фрактальних зображень за допомогою системи РСІФ є значною порівняно з детермінованими системами ітераційних функцій побудови фракталів. При збільшенні роздільної здатності можна навіть використовувати потрібну верифікацію документа, що базується на третій ітерації алгоритму.

Визначення фрактальної розмірності є складним математичним завданням, але навчена нейронна мережа може виконати це завдання. Запропонований алгоритм розрахунку фрактальної розмірності дозволяє автоматизувати процес навчання нейронної мережі шляхом генерації фракталів. Генерація фрактальних зображень є складним процесом, оскільки вимагає багато математичних обчислень. Аналіз систем ітераційних функцій показав, що запропонована удосконалена РСІФ є найефективнішою, оскільки її коефіцієнти дозволяють визначати ширину, довжину, пропорційні коефіцієнти за шириною та висотою, а також координати центрів фігур першої ітерації. Визначення цих параметрів дозволяє точно визначити площі перетинів фігур, що, в свою чергу, допомагає скоригувати формулу розрахунку фрактальної розмірності. Удосконалена РСІФ дозволяє створити масив даних, який містить зображення фракталів та

відповідні значення фрактальної розмірності. Цей масив даних забезпечить автоматизовану систему для навчання нейронної мережі визначати фрактальну розмірність для заданого зображення. Швидкість роботи удосконаленої РСІФ дозволяє швидко навчити нейронну мережу, що неможливо було б зробити за допомогою детермінованої СІФ. Навчена нейронна мережа миттєво надає значення та точні результати визначення фрактальної розмірності, з похибкою не більше 1%.

Цей шифрувальний алгоритм є легким у реалізації і дозволяє створювати шифрувальні ключі, необхідні для розшифрування зображень. Він також маскує розмір зображення, що забезпечує додатковий рівень захисту. Використання додаткових елементів ускладнює співставлення пікселів для нейронних мереж. Зміна кольорів, яскравості та інверсія з використанням випадкових процесів ускладнюють завдання підбору функції розшифрування. Оскільки матриці формуються випадковим чином, нейронні мережі не можуть встановити шифрувальну функцію. Для розшифрування зображення з 1000 пікселів потрібно перебрати надзвичайно велику кількість комбінацій (понад $5.1 \cdot 10^{2576}$), що на сьогоднішній день не є реалістичним технічним рішенням. Такий алгоритм може бути успішно застосований у військових, розвідувальних та сферах захисту інформації. Великою перевагою цього алгоритму є можливість передавати зашифровані зображення через відкриті канали.

РОЗДІЛ 4. ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА БІБЛІОТЕК ЗАПРОПОНОВАНИХ МЕТОДІВ У WEB-ТЕХНОЛОГІЯХ

4.1 Розробка програмної моделі генерації фрактальних зображень типу «Cantor dust» на основі удосконаленого методу побудови фрактальних структур

Для побудови генератора фрактальних зображень типу «Cantor dust» на базі розробленого алгоритму будемо використовувати Web-технології, а саме мову розмітки гіпертексту HTML [118-119], мову програмування JavaScript [120-121], та каскадні таблиці стилів CSS. Вибір цих інструментів дозволить створити Web-бібліотеку.

Також для роботи будуть використані дві бібліотеки jQuery [122-123] та Bootstrap, які працюють на основі JavaScript. HTML, CSS та Bootstrap відповідатимуть за дизайн генератора. JavaScript та jQuery відповідатимуть за бізнес логіку. Базова структура компонентів генератора (Рис. 4.1)

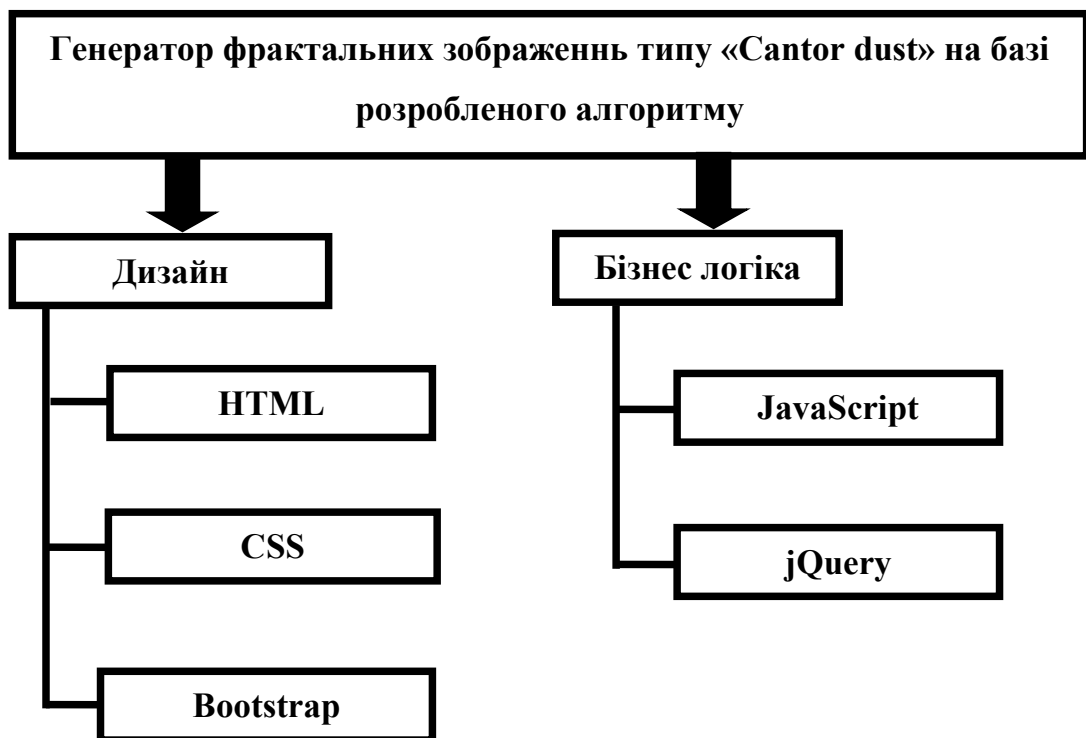


Рис. 4.1. Структура компонентів програмної моделі генератора

Спочатку створимо папку **Fractal generator**:



Рис. 4.2. Папку Fractal generator

В цій папці створимо файл `index.html` та завантажимо бібліотеки jQuery та Bootstrap (Рис. 4.3):

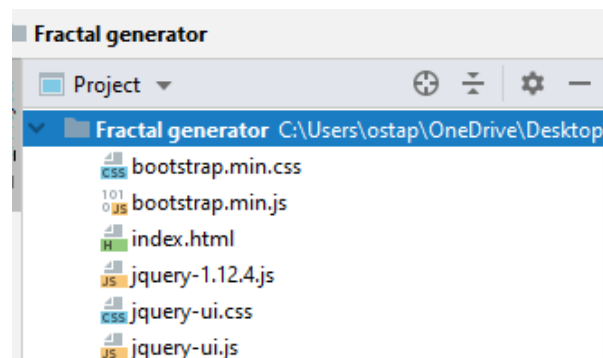


Рис. 4.3. Структура компонентів генератора

У файлі `index.html` підключаємо завантажені бібліотеки Рис4.4:

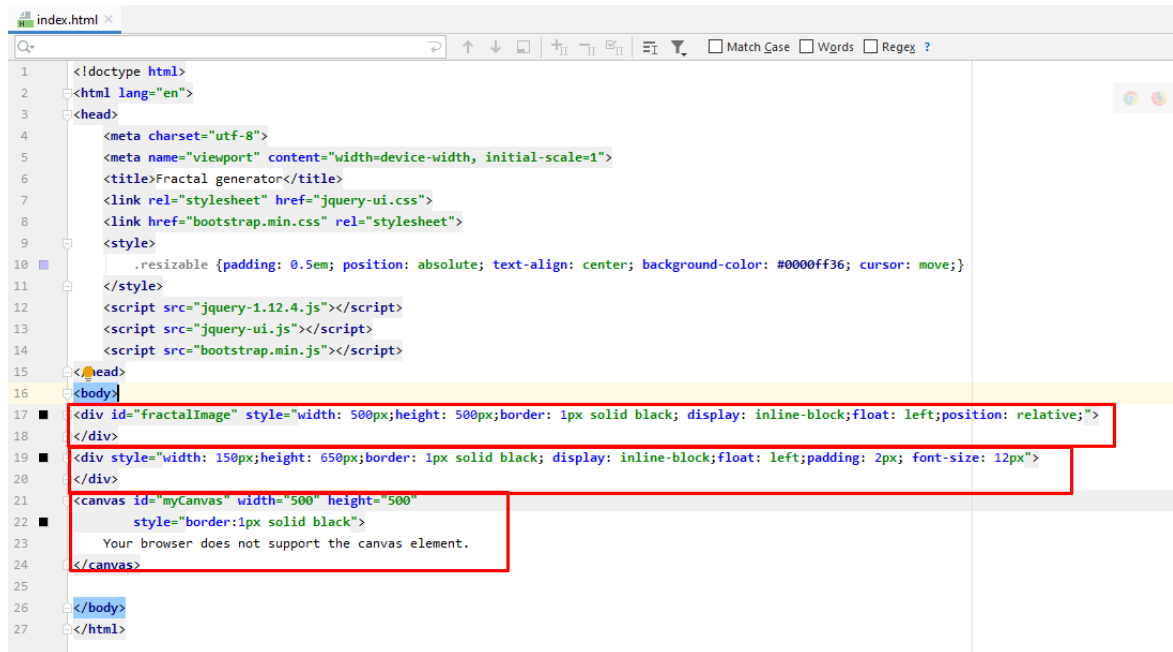


Рис. 4.4. Підключення бібліотек

Добавимо стилі для компонента нашого генератора:

```
<style>
  .resizable {padding: 0.5em; position: absolute; text-align: center; background-
  color: #0000ff36; cursor: move;}
</style>
```

Сформуємо з HTML компонентів три віконечка, перше відповідатиме за конструювання фракталу, друге відповідатиме за елементи керування, третє – за відтворення фрактального зображення (Рис. 4.5-4.6):



```
1 <!doctype html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1">
6 <title>Fractal generator</title>
7 <link rel="stylesheet" href="jquery-ui.css">
8 <link href="bootstrap.min.css" rel="stylesheet">
9 <style>
10   .resizable {padding: 0.5em; position: absolute; text-align: center; background-color: #0000ff36; cursor: move;}
11 </style>
12 <script src="jquery-1.12.4.js"></script>
13 <script src="jquery-ui.js"></script>
14 <script src="bootstrap.min.js"></script>
15 </head>
16 <body>
17 <div id="fractalImage" style="width: 500px; height: 500px; border: 1px solid black; display: inline-block; float: left; position: relative;">
18 </div>
19 <div style="width: 150px; height: 650px; border: 1px solid black; display: inline-block; float: left; padding: 2px; font-size: 12px">
20 </div>
21 <canvas id="myCanvas" width="500" height="500" style="border: 1px solid black">
22   Your browser does not support the canvas element.
23 </canvas>
24 </body>
25 </html>
```

Рис. 4.5. Формування елементів генератора

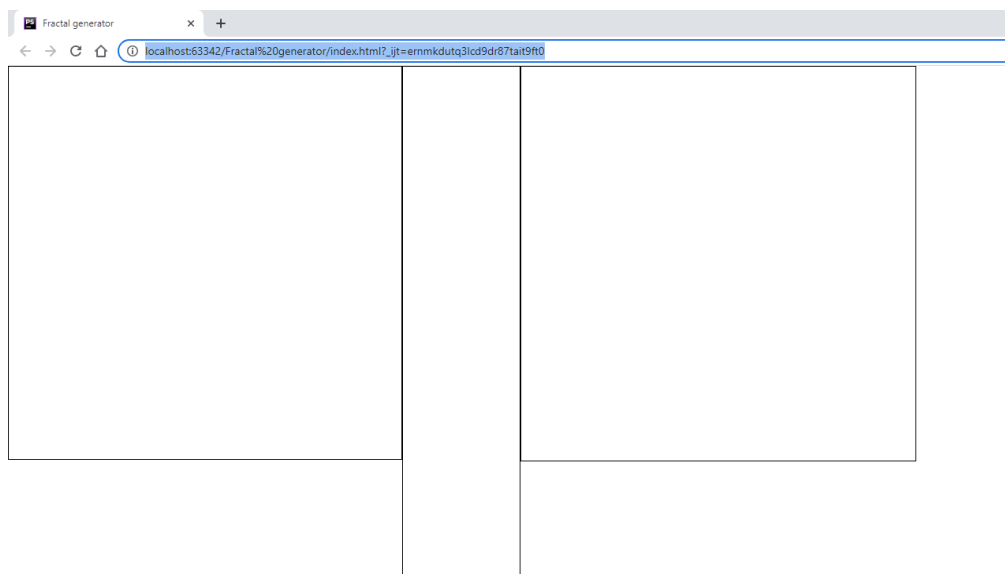


Рис. 4.6. Формування елементів генератора

Наступним кроком добавимо елементи керування у другий блок, наступний код:

```

<div class="form-group" style="text-align: center">
  <input type="text" class="form-control" id="addWidth" placeholder="Width" value="100">
  <input type="text" class="form-control" id="addHeight" placeholder="Height" value="100">
  <input type="text" class="form-control" id="addX" placeholder="Position X" value="0">
  <input type="text" class="form-control" id="addY" placeholder="Position Y" value="0">
  <input type="text" class="form-control" id="addFi" placeholder="Fi" value="0">
  <button id="addNew" type="button" class="btn btn-primary">Add New</button>
</div>
<div id="updateEl" class="form-group" style="text-align: center; display: none">
  <label id="upLabel" for="upWidth">#</label>
  <input type="text" class="form-control" id="upWidth" placeholder="Width">
  <input type="text" class="form-control" id="upHeight" placeholder="Height">
  <input type="text" class="form-control" id="upX" placeholder="Position X">
  <input type="text" class="form-control" id="upY" placeholder="Position Y">
  <input type="text" class="form-control" id="upFi" placeholder="Fi">
  <button id="updateFr" data-number="0" type="button" class="btn btn-primary">Update</button>
  <button id="deleteFr" data-number="0" type="button" class="btn btn-warning">Delete</button>
</div>
<div class="form-group" style="text-align: center;">
  <input type="text" class="form-control" id="Quality" placeholder="Quality" value="10000">
</div>
<div class="form-group" style="text-align: center;">
  <button id="createFr" type="button" class="btn btn-primary">Fractal</button>
</div>
<div class="form-group" style="text-align: center;">
  <a id="btn-Convert-Html2Image" href="#" style="display: none">Download</a>
</div>

```

Вигляд нашого генератора зміниться відповідно до рис. 4.7



Рис. 4.7. Вигляд елементів керування

Ми добавили три групи компонентів керування:

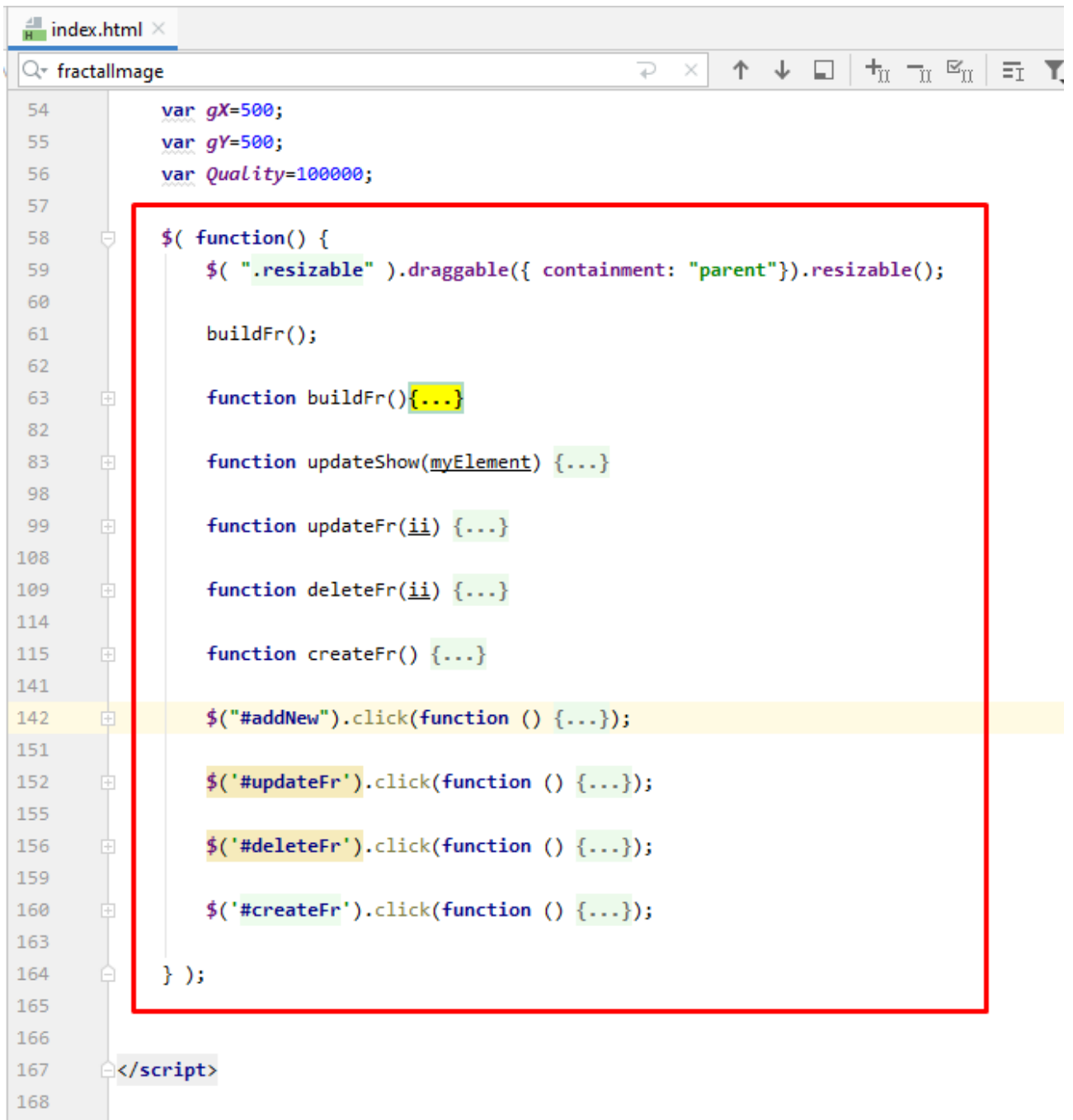
- Група, яка відповідає за додавання компонента фрактала, до її складу входять наступні поля: `addWidth` – поле, котре відповідає за ширину елемента; `addHeight` – поле, котре відповідає за висоту елемента; `addX` – поле, котре відповідає за позицію елемента по осі X; `addY` – поле, котре відповідає за позицію елемента по осі Y; `addFi` – поле, котре відповідає за кут повороту елемента; `addNew` – кнопка, котра відповідає за додавання елемента.
- Група, яка відповідає за редагування компонента фрактала, до її складу входять наступні поля: `upWidth` – поле, котре відповідає за ширину елемента; `upHeight` – поле, котре відповідає за висоту елемента; `upX` – поле, котре відповідає за позицію елемента по осі X; `upY` – поле, котре відповідає за позицію елемента по осі Y; `upFi` – поле, котре відповідає за кут повороту елемента; `updateFr` – кнопка, котра відповідає за внесення змін до елемента; `deleteFr` – кнопка, котра відповідає за видалення елемента.
- Група, яка відповідає за формування фракталу до її складу входять наступні поля: `Quality` - поле, котре відповідає за кількість операцій РСІФ; `createFr` – кнопка, котра відповідає за створення фрактала; `btn-Convert-Html2Image` – кнопка, котра відповідає за збереження фрактального зображення.

Наступним кроком добавимо власний JavaScript:

```
<script>
  var elements=[{width:100,height:100,x:0,y:0,fi:0}];
  var gX=500;
  var gY=500;
  var Quality=100000;
</script>
```

Де змінна `elements` – це масив даних параметрів фігур перших ітерацій фрактального зображення, `gX` - ширина поля фрактального зображення, `gY` – довжина поля фрактального зображення, `Quality` – кількість операцій РСІФ.

Наступним етапом є додавання функцій до нашого генератора Рис.4.8:



```
54     var gX=500;
55     var gY=500;
56     var Quality=100000;
57
58     $( function() {
59         $( ".resizable" ).draggable({ containment: "parent"}).resizable();
60
61         buildFr();
62
63         function buildFr(){...}
82
83         function updateShow(myElement) {...}
98
99         function updateFr(ii) {...}
108
109         function deleteFr(ii) {...}
114
115         function createFr() {...}
141
142         $("#addNew").click(function () {...});
151
152         $('#updateFr').click(function () {...});
155
156         $('#deleteFr').click(function () {...});
159
160         $('#createFr').click(function () {...});
163
164     } );
165
166
167 </script>
168
```

Рис. 4.8. Додавання функцій до генератора

Функція buildFr() ініціалізується із завантаженням сторінки, дана функція має наступний код:

```
function buildFr(){
    var fractalImage=$('#fractalImage');
    fractalImage.empty();
    for(var ii=0;ii<elements.length;ii++){
        var
```

```

        newElement=$(`<div data-number="'+ii+'" id="resizable_'+ii+'" class="ui-
widget-content resizable">'+(ii+1)+'</div>').draggable({
    containment: "parent").resizable({ containment:
"parent"}).height(elements[ii].height).width(elements[ii].width).css({
    marginLeft: 0, marginTop: 0,
    top: elements[ii].y+'px', left: elements[ii].x+'px', position:
"absolute"
}).mouseup(function() {
    var myElement=$(this);
    updateShow(myElement);
}).resize(function() {
    var myElement=$(this);
    updateShow(myElement);
});
    fractalImage.append(newElement);
}
}
}

```

При запуску цієї функції очищається конструктор (перша частина генератора), після чого з масиву `elements` в циклі будується елементи першої ітерації масиву, по замовчуванні при загрузці сторінки знаходиться перший елемент, на кожен з елементів масиву вішаються дві події `mouseup` та `resize`, ці події в свою чергу будуть викликати функцію `updateShow()`, з передачею на неї компонента зі змінами параметрів.

Розглянемо функцію `updateShow()`:

```

function updateShow(myElement) {
    var elementPadding=2*parseInt(myElement.css("padding"), 10);
    var elementBorder=2*parseInt(myElement.css("border"), 10);
    elements[parseFloat(myElement.data('number'))]={width:parseInt(myElement.width(),
10)+elementPadding+elementBorder,height:parseInt(myElement.height(),
10)+elementPadding+elementBorder,x:parseInt(myElement.position().left,10),y:parseInt(my
Element.position().top,10), fi:elements[parseFloat(myElement.data('number'))].fi};
    $('#updateEl').show();
    $('#upLabel').text('#'+(parseFloat(myElement.data('number'))+1));
    $('#updateFr').data('number',myElement.data('number'));
    $('#deleteFr').data('number',myElement.data('number'));
    $('#upWidth').val(elements[parseFloat(myElement.data('number'))].width);
    $('#upHeight').val(elements[parseFloat(myElement.data('number'))].height);
    $('#upX').val(elements[parseFloat(myElement.data('number'))].x);
    $('#upY').val(elements[parseFloat(myElement.data('number'))].y);
    $('#upFi').val(elements[parseFloat(myElement.data('number'))].fi);
}

```

Функція `updateShow()` апдейтить параметри елемента в масиві `elements` та передає їх на Групу, яка відповідає за редагування компонента фрактала.

Натиснувши кнопку `updateFr`, виникає подія `click`, на яку підписана кнопка `updateFr`, що в свою чергу запускає функцію `updateFr()`:

```
$('#updateFr').click(function () {
    updateFr($(this).data('number'));
});
```

Розглянемо функцію updateFr():

```
function updateFr(ii) {
    var elementWidth=parseFloat($('#upWidth').val());
    var elementHeight=parseFloat($('#upHeight').val());
    var elementX=parseFloat($('#upX').val());
    var elementY=parseFloat($('#upY').val());
    var elementFi=parseFloat($('#upFi').val());

    elements[ii]={width:elementWidth,height:elementHeight,x:elementX,y:elementY,fi:elementFi};
    buildFr();
}
```

Функція updateFr() оновлює параметри елемента в масиві elements та перезапускає функцію buildFr().

Натиснувши кнопку deleteFr, виникає подія click, на яку підписана кнопка deleteFr, що в свою чергу запускає функцію deleteFr ():

```
$('#deleteFr').click(function () {
    deleteFr($(this).data('number'));
});
```

Розглянемо функцію deleteFr():

```
function deleteFr(ii) {
    elements.splice(ii, 1);
    buildFr();
    $('#updateE1').hide();
}
```

Функція deleteFr() видаляє елемент в масиві elements та перезапускає функцію buildFr().

Натиснувши кнопку addNew, виникає подія click, на яку підписана кнопка addNew, що в свою чергу додає новий елемент в масив elements з параметрами, які були внесені в Групу, яка відповідає за додавання компонента фрактала, та перезапускає функцію buildFr():

```

$("#addNew").click(function () {
    var elementWidth=parseFloat($('#addWidth').val());
    var elementHeight=parseFloat($('#addHeight').val());
    var elementX=parseFloat($('#addX').val());
    var elementY=parseFloat($('#addY').val());
    var elementFi=parseFloat($('#addFi').val());

    elements.push({width:elementWidth,height:elementHeight,x:elementX,y:elementY,fi:element
    Fi});
    buildFr();
});

```

Натиснувши кнопку createFr, виникає подія click, на яку підписана кнопка createFr, що в свою чергу запускає функцію createFr ():

```

$('#createFr').click(function () {
    createFr();
});

```

Розглянемо функцію createFr ():

```

function createFr() {
    Quality=parseFloat($('#Quality').val());
    var canvas = document.getElementById("myCanvas");
    var ctx = canvas.getContext("2d");
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    var x2,y2;
    var xx=0,yy=0;
    var x3,y3;
    for(var i=0;i<Quality;i++){
        var nn=Math.floor(Math.random() * elements.length);
        var x1=elements[nn].x+ Math.floor(elements[nn].width/2);
        var xK=Math.floor(gX/elements[nn].width);
        var y1=elements[nn].y+ Math.floor(elements[nn].height/2);
        var yK=Math.floor(gY/elements[nn].height);
        x2=Math.floor((xK*x1-gX/2)/(xK-1));
        y2=Math.floor((yK*y1-gY/2)/(yK-1));
        x3=x2-Math.floor((x2-xx)/xK);
        y3=y2-Math.floor((y2-yy)/yK);
        xx=(x3-x1)*Math.cos(elements[nn].fi*Math.PI/180) - (y3-
        y1)*Math.sin(elements[nn].fi*Math.PI/180) + x1;
        yy=(x3-x1)*Math.sin(elements[nn].fi*Math.PI/180) + (y3-
        y1)*Math.cos(elements[nn].fi*Math.PI/180) + y1;
        ctx.fillRect(xx, yy, 1, 1);
    }
    var imgageData = canvas.toDataURL("image/png");
    var newData = imgageData.replace(/^data:image\/png/, "data:application/octet-
    stream");
    $("#btn-Convert-Html2Image").show().attr("download", "fractal.png").attr("href",
    newData);
}

```

Функція createFr() формує з масиву elements удосконалену РСІФ та буде фрактал за допомогою елемента canvas, після чого формується посилання для скачування фрактального зображення та вставляється в кнопку btn-Convert-Html2Image.

Принцип та результат роботи фрактального генератора зображено на Рис.4.9

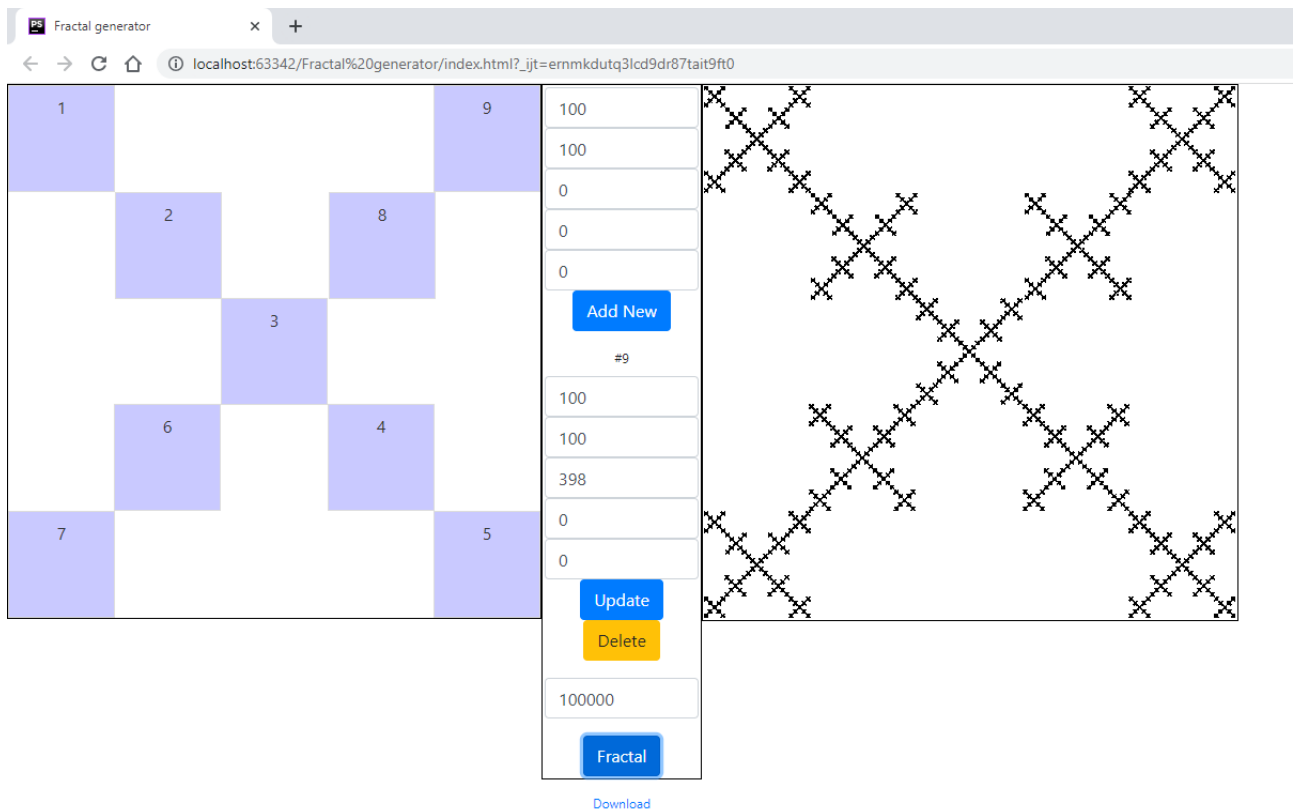


Рис.4.9 Принцип та результат роботи фрактального генератора на базі удосконаленої РСІФ

4.2 Програмна система відновлення (спотворення) та відтворення фрактальних зображень типу «Cantor dust».

Для написання системи відновлення (спотворення) та відтворення фрактальних зображень типу «Cantor dust» на базі розробленого алгоритму будемо використовувати Web-технології, а саме мову розмітки гіпертексту HTML, мову програмування JavaScript, та каскадні таблиці стилів CSS, вибір цих інструментів дозволить створити Web-бібліотеку.

Також для роботи будуть використані три бібліотеки jQuery, Bootstrap та Brain.js [126-127], які працюють на основі JavaScript. HTML, CSS та Bootstrap відповідатимуть за дизайн генератора. JavaScript, jQuery та Brain.js відповідатимуть за бізнес логіку. Базова структура компонентів генератора (Рис. 4.10)

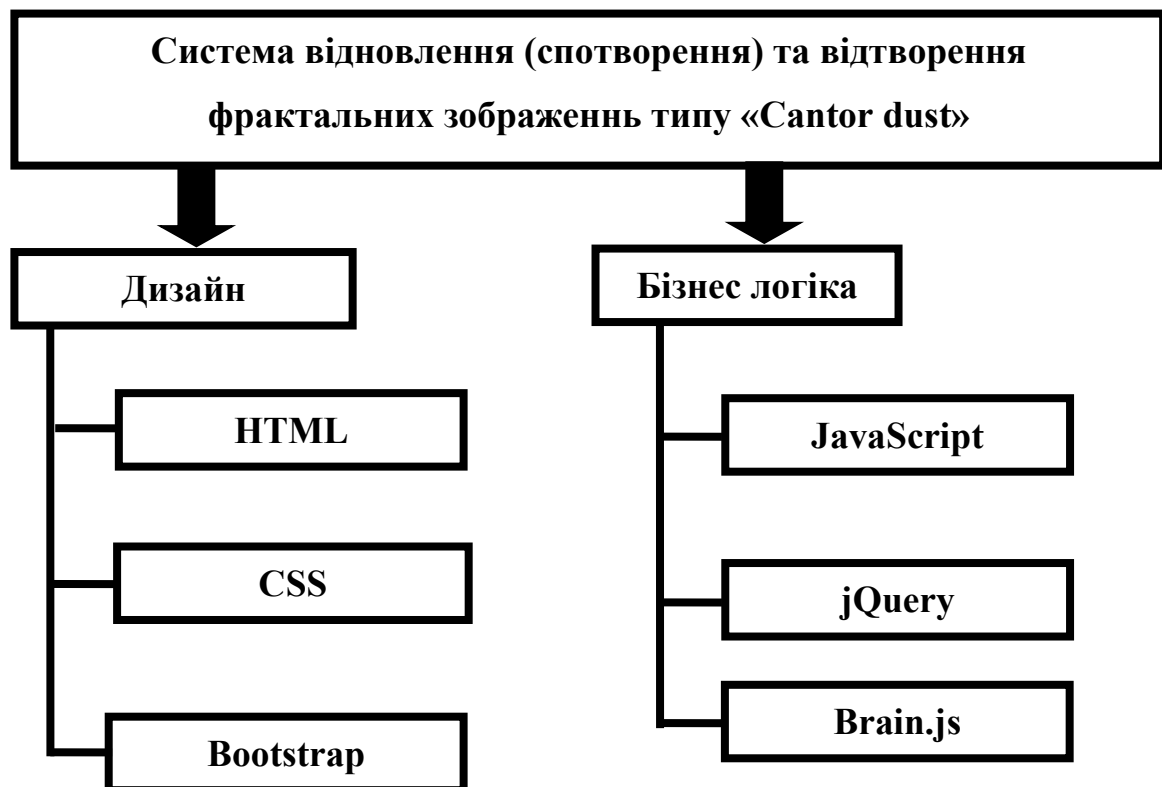


Рис. 4.10. Структура компонентів системи відновлення (спотворення) та відтворення фрактальних зображень типу «Cantor dust».

Спочатку створимо папку **Fractal recovery**:

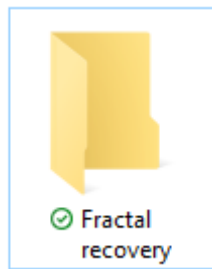


Рис. 4.11. Папку Fractal recovery

В цій папці створимо файл `index.html` та завантажимо бібліотеки jQuery та Bootstrap (Рис. 4.12):

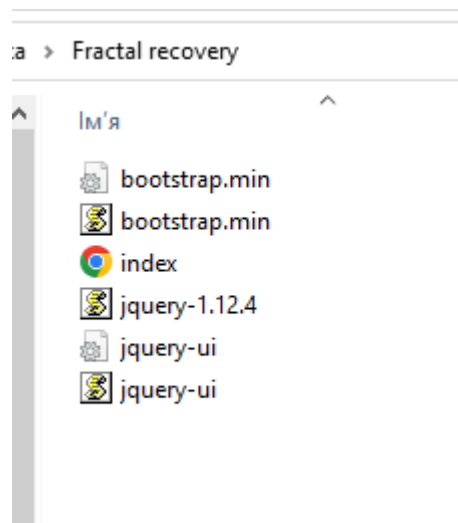


Рис. 4.12. Структура компонентів Fractal recovery

Наступним кроком встановлюємо бібліотеку `brain.js` з сайту `npmjs.com` (Рис. 4.13):

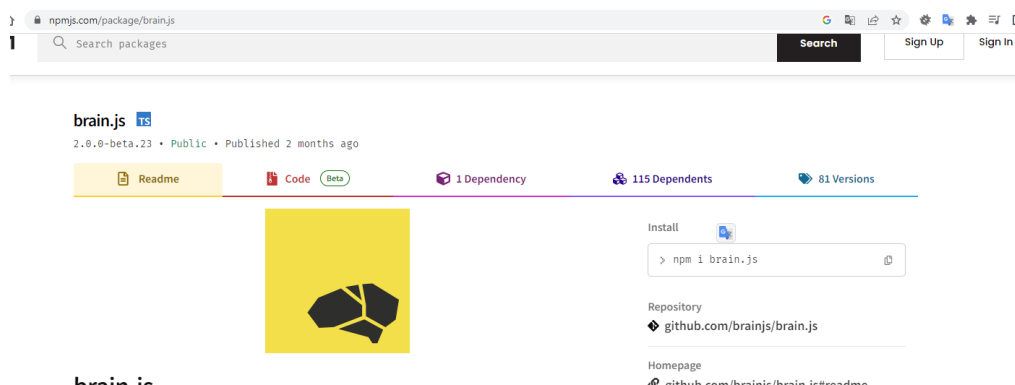


Рис. 4.13. Встановлення Brain.js

У файлі index.html підключаємо завантажені бібліотеки Рис.4.14:

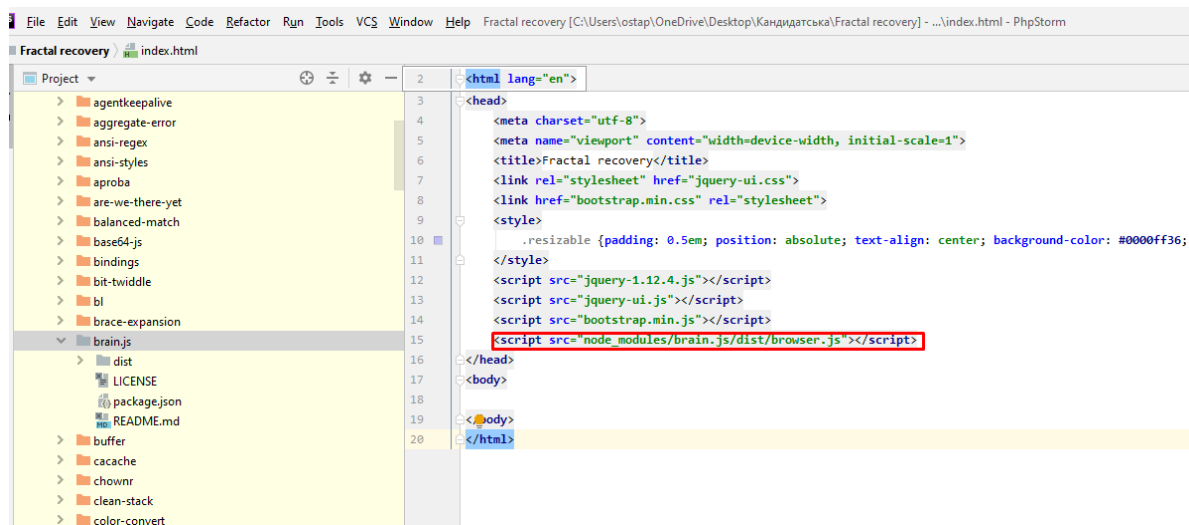


Рис. 4.14. Підключення бібліотек

Добавимо стилі для компонента нашої програми та сформуємо з HTML компонентів п'ять віконечок, перше відповідатиме за керування, друге відповідатиме за вивід генерованого фрактального зображення або завантаженого фрактального зображення, третє за виведення відтвореного фрактального зображення за відтворення фрактального зображення, четверте та п'яте за виведення параметрів для удосконаленої РСІФ, які генерувались і відтворювалися (Рис. 4.15-4.16):

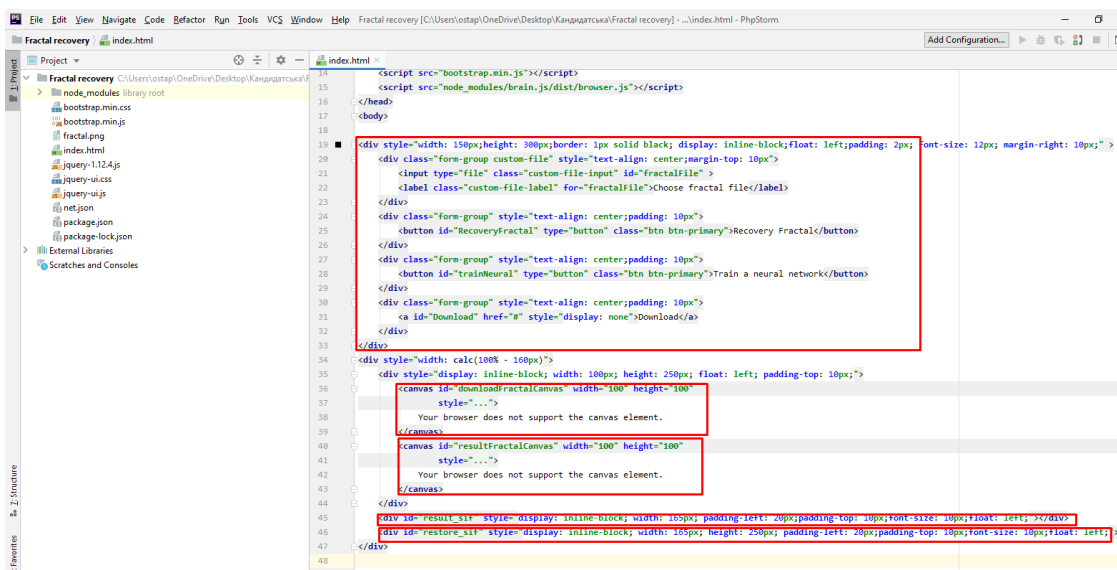


Рис. 4.15. Формування елементів програми

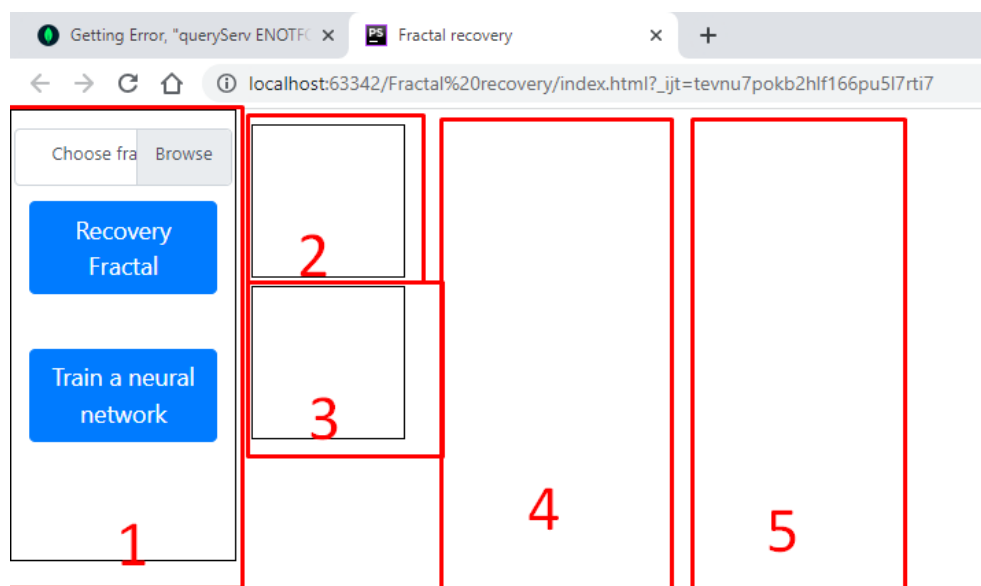


Рис. 4.16. Формування елементів програми

Ми добавили чотири компоненти керування (Рис. 4.17):

- Перший компонент відповідає за додавання фрактального зображення, котре потрібно відновити, `fractalFile` – кнопка, котра відповідає за додавання фрактального зображення.
- Другий компонент відповідає за відновлення (покращення якості) фрактального зображення, котре було завантажено, `RecoveryFractal` – кнопка, котра відповідає за відновлення фрактального зображення.
- Третій компонент відповідає за навчання нейронної мережі і виведення генерованого фрактального зображення спотвореної якості та виведення зображення удосконаленої РСІФ (оригінальне зображення), `trainNeural` – кнопка, котра запускає процес навчання нейронної мережі `brain.js`.
- Четвертий компонент відповідає за збереження відновленого або генерованого (спотвореного) фрактального зображення, елемент `Download` відповідає за збереження зображення.

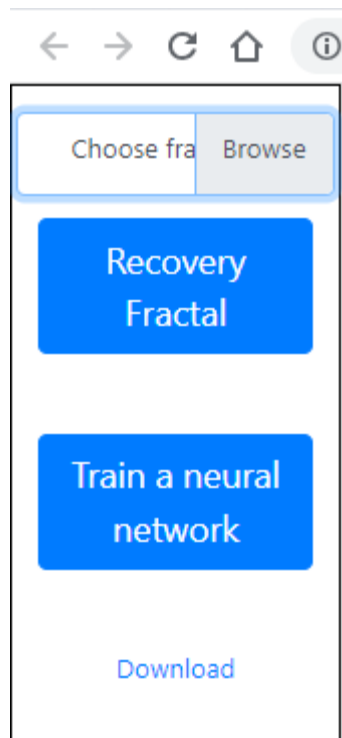


Рис. 4.17. Вигляд елементів керування

Наступним кроком добавимо власний JavaScript (Рис.4.18) та за допомогою змінної `net = new brain.NeuralNetwork()` глобально створюємо нейронну мережу `net`, за допомогою котрої будемо проводити навчання нейронної мережі та відновлення якості фрактального зображення.

```
<script>  
  
    var gX=100;  
    var gY=100;  
    var canvas = document.getElementById('downloadFractalCanvas');  
    var ctx = canvas.getContext('2d');  
    var net = new brain.NeuralNetwork();  
  
    $( function() {...} );  
</script>
```

Рис. 4.18. Додавання глобальних змінних

Де змінні `canvas` та `ctx` – відповідають за виведення інформації у віконечко 2 (Рис. 4.17) та отримання інформації з нього, `gX` - ширина поля фрактального зображення, `gY` – довжина поля фрактального зображення.

Наступним етапом є додавання функцій до нашої програми Рис.4.19:

```
var gX=100;
var gY=100;
var canvas = document.getElementById('downloadFractalCanvas');
var ctx = canvas.getContext('2d');
var net = new brain.NeuralNetwork();

$( function() {

    $("#trainNeural").click(function () {...});

    $("#RecoveryFractal").click(function () {...});

    async function trainNeural() {...}

    getTrainNeuralJson = function () {...};

    var saveBlob = (function () {...})();

    var imageLoader = document.getElementById('fractalFile');
    imageLoader.addEventListener('change', handleImage, false);

    function handleImage(e){...}

    async function RecoveryFractal(){...}

});
</script>
</body>
</html>
```

Рис. 4.19. Додавання функцій до програми

Натиснувши кнопку `trainNeural`, виникає подія `click`, на яку підписана кнопка `updateFr`, що в свою чергу запускає функцію `trainNeural()`:

```
$("#trainNeural").click(function () {
    trainNeural();
});
```

Розглянемо функцію `trainNeural()` (в коментарях описано кожний компонент роботи функції):

```

async function trainNeural() {

    let trainNeuralSteps=10000; // Кількість згенерованих фракталів для навчання
    let globalTrainData =[];

    var canvas2 = document.getElementById("resultFractalCanvas");
    var ctx2 = canvas2.getContext("2d"); //Поле виведення згенерованого фракталу

    for(let s=0; s<trainNeuralSteps; s++){
        ctx2.clearRect(0, 0, canvas2.width, canvas2.height); //Очищення полів
        ctx.clearRect(0, 0, canvas.width, canvas.height);
        let N = 2000; //кількість операцій удосконаленої РСІФ
        let elementsNumber = Math.floor(Math.random()*9) + 2; //сило фігур першої
        //ітерації фракталу

        let elements = [];
        let elements2 = [];

        elements2 = {};

        for(let j=0; j<elementsNumber; j++){ //генерація параметрів фігур
            let startX = Math.floor(Math.random()*0.8*gX);
            let startY = Math.floor(Math.random()*0.8*gY);
            let width = Math.floor(Math.random()*gX/3)+gX/5;
            let height = Math.floor(Math.random()*gY/3)+gY/5;
            width = width + startX < gX-2 ? width : (width + startX)/2;
            height = height + startY < gY-2 ? height : (height + startY)/2;
            let Xa =Math.floor((width/2 + startX));
            let Ya =Math.floor((height/2 + startY));
            let kx = gX/width;
            let ky = gY/height;
            let Fi = 0; // Math.random()*180;
            elements.push([kx,ky,Xa,Ya,Fi]); // Формування вихідних даних для навчання
            elements2['sif_'+j*5] = kx/10;
            elements2['sif_'+(j*5+1)] = ky/10;
            elements2['sif_'+(j*5+2)] = Xa/100;
            elements2['sif_'+(j*5+3)] = Ya/100;
            elements2['sif_'+(j*5+4)] = Fi;
        }

        let x2,y2;
        let xx=0,yy=0;
        let x3,y3;
        for(let i=0;i<N;i++){ // формування удосконаленої РСІФ
            let nn=Math.floor(Math.random() * elements.length);

            let kx=elements2['sif_'+nn*5]*10;
            let ky=elements2['sif_'+(nn*5+1)]*10;
            let Xa=elements2['sif_'+(nn*5+2)]*100;
            let Ya=elements2['sif_'+(nn*5+3)]*100;
            let Fi=0;
        }
    }
}

```

```

x2=Math.floor((kx*Xa-gX/2)/(kx-1));

y2=Math.floor((ky*Ya-gY/2)/(ky-1));
x3=x2-Math.floor((x2-xx)/kx);
y3=y2-Math.floor((y2-yy)/ky);
xx=Math.floor((x3-Xa)*Math.cos(Fi*Math.PI/180) - (y3-Ya)*Math.sin(
Fi*Math.PI/180) + Xa);
yy=Math.floor((x3-Xa)*Math.sin( Fi*Math.PI/180) + (y3-Ya)*Math.cos(
Fi*Math.PI/180) + Ya);
if(xx>-1 && yy>-1 && i < 100){
    ctx2.fillRect(xx, yy, 1, 1);//Побудова спотвореного фрактала
}
ctx.fillRect(xx, yy, 1, 1); // Побудова оригінального фрактала
}
let imgData={};
let s=0;
for(var i=0; i<gX; i++){ // Формування вхідних даних для навчання
    for(var j=0; j<gY; j++){
        var color =ctx2.getImageData(i-1, j-1, 1, 1).data;
        if(color[0]||color[1]||color[2]||color[3]){
            imgData[s]=i;
            s++;
            imgData[s]=j;
            s++;
        }
    }
}
globalTrainData.push({ input: imgData, output: elements2 });
var imgageData = canvas2.toDataURL("image/png");
var newData = imgageData.replace(/^data:image\/png/, "data:application/octet-
stream");
//Формування посилання для збереження спотвореного фрактального зображення

$("#Download").show().attr("download", "fractal.png").attr("href", newData);
let html='<div>Create ISF</div>';
for(let el of elements){ // Виведення параметрів удосконаленої РСІФ
    html+='<br><div>Kx = '+el[0].toFixed(2)+' Ky = '+el[1].toFixed(2)+' Xa =
'+el[2].toFixed(2)+' Ya = '+el[3].toFixed(2)+' Fi = '+el[4].toFixed(2)+'</div>';
}
$("#result_sif").html(html);
}
net.train(globalTrainData,{ // Запуск процесу навчання нейронної мережі
    iterations: 20000,
    log: true,
    errorThresh: 0.0005,
    momentum: 0.1,
    learningRate: 0.3,
});
}

```

Результат роботи функції навчання нейронної мережі (Рис.4.20):

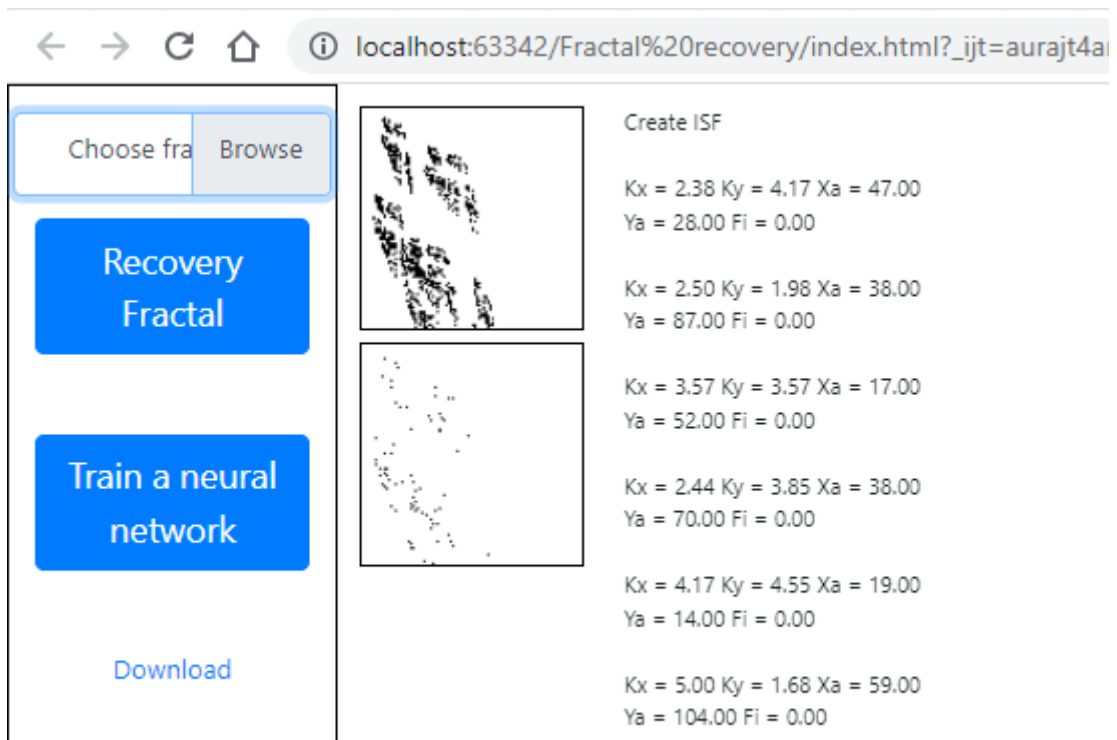


Рис.4.20. Результат роботи функції trainNeural()

Після натискання посилання «Download», виконується збереження спотвореного зображення (рис 4.21):

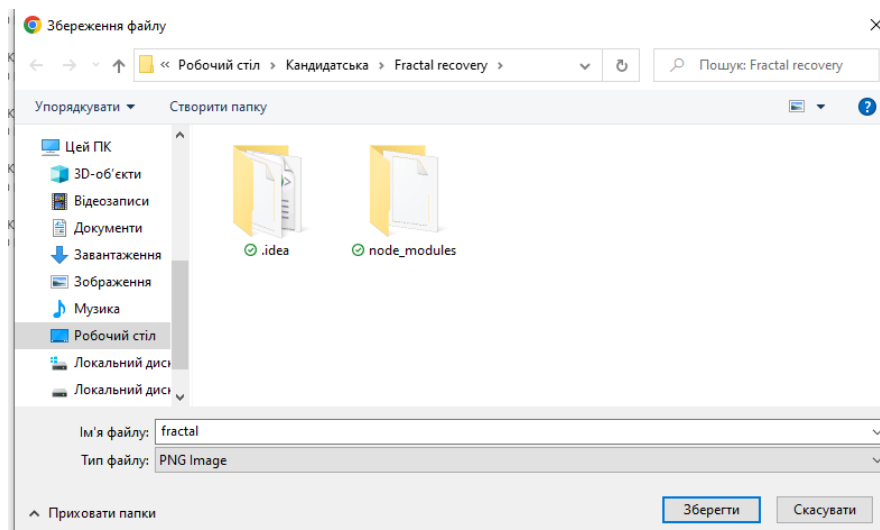


Рис.4.21. Збереження спотвореного фрактального зображення

Натиснувши кнопку fractalFile, відкривається вікно завантаження зображення (Рис.4.22), в ньому вибираємо попередньо збережений спотворений фрактал:

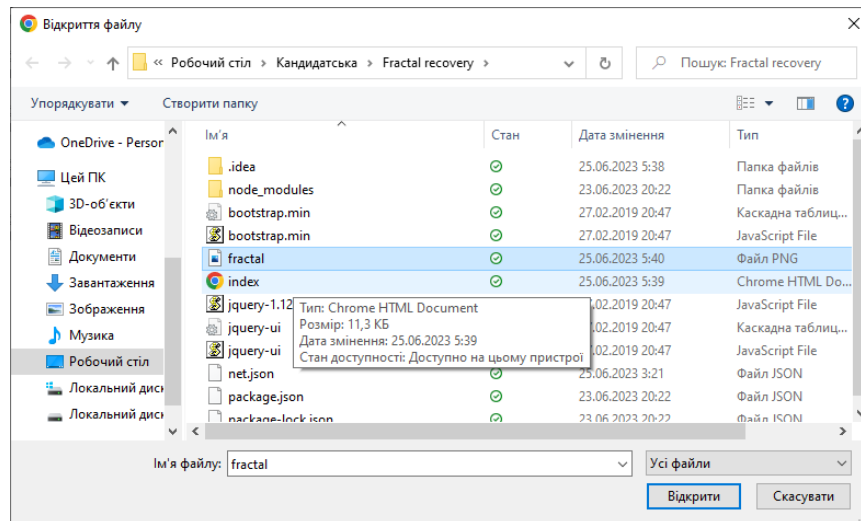


Рис.4.22. Завантаження спотвореного фрактального зображення в програму

Після чого виникає подія change, на яку підписана кнопка fractalFile, що в свою чергу запускає функцію handleImage():

```
var imageLoader = document.getElementById('fractalFile');
imageLoader.addEventListener('change', handleImage, false);
```

Функція handleImage() має наступний вигляд:

```
function handleImage(e){
    var reader = new FileReader();
    reader.onload = function(event){
        var img = new Image();
        img.onload = function(){
            canvas.width = img.width;
            canvas.height = img.height;
            ctx.drawImage(img,0,0); //виведення завантаженого зображення у програму
        };
        img.src = event.target.result;
    };
    reader.readAsDataURL(e.target.files[0]);
}
```

Після виконання функції handleImage() (Рис 4.23) два зображення будуть ідентичні:

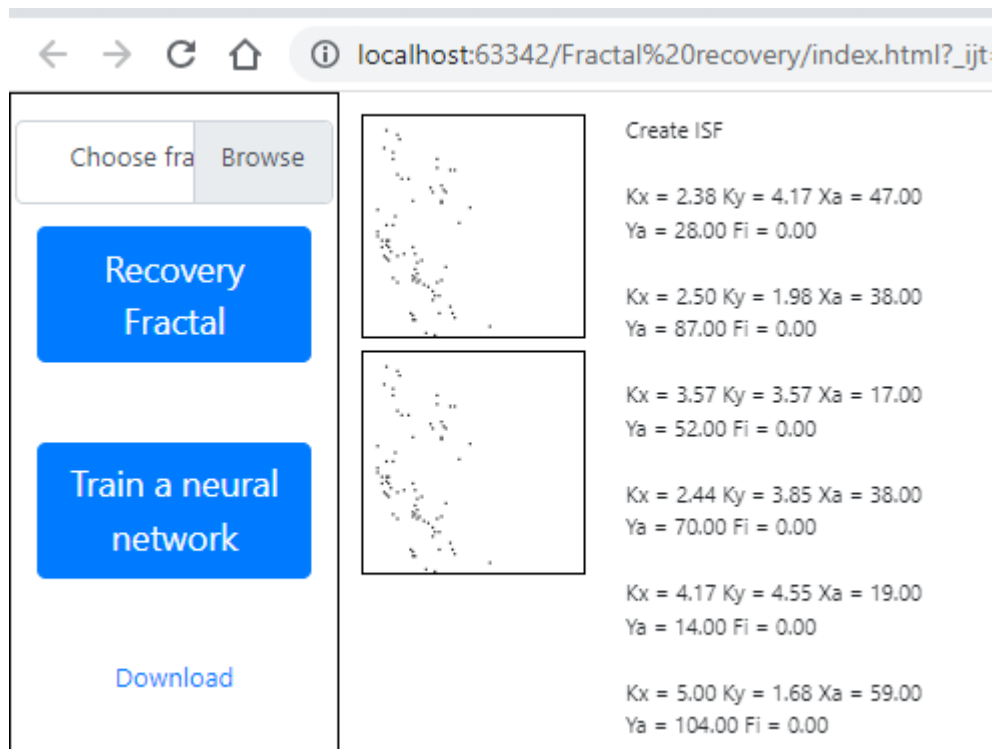


Рис.4.23. Завантаження спотвореного фрактального зображення в програму

Натиснувши на кнопку RecoveryFractal, виникає подія click, на яку підписана кнопка RecoveryFractal, що в свою чергу запускає функцію RecoveryFractal():

```
$("#RecoveryFractal").click(function () {
    RecoveryFractal();
});
```

Розглянемо функцію RecoveryFractal() (в коментарях вказано основні частини роботи функції):

```
async function RecoveryFractal(){
    let imgData={};
    let s=0;
    for(var i=0; i<gX; i++){ //формування даних для в нейронну мережу
        for(var j=0; j<gY; j++){
            var color =ctx.getImageData(i-1, j-1, 1, 1).data;
            if(color[0]||color[1]||color[2]||color[3]){
                imgData[s]=i;
                s++;
                imgData[s]=j;
                s++;
            }
        }
    }
}
```

```

}
let sif = net.run(imgData); //отримання результатів з нейронної мережі
let elements=[];
let ss=0;
for(let i=0;i<Object.keys(sif).length;i+=5){
elements[ss]=[sif['sif_'+i],sif['sif_'+(i+1)],sif['sif_'+(i+2)],sif['sif_'+(i+3)],sif[
sif_'+(i+4)]];|
    ss++
}
var canvas2 = document.getElementById("resultFractalCanvas");
var ctx2 = canvas2.getContext("2d");
ctx2.clearRect(0, 0, canvas2.width, canvas2.height);
let N = 1000;
let x2,y2;
let xx=0,yy=0;
let x3,y3;
for(let i=0;i<N;i++){//формування удосконаленої РСІФ по результатам отриманих від
//нейронної мережі
    let nn=Math.floor(Math.random() * elements.length);

    let kx=elements[nn][0]*10;
    let ky=elements[nn][1]*10;
    let Xa=Math.floor(elements[nn][2]*100);
    let Ya=Math.floor(elements[nn][3]*100);
    let Fi=0;

    x2=Math.floor((kx*Xa-gX/2)/(kx-1));
    y2=Math.floor((ky*Ya-gY/2)/(ky-1));
    x3=x2-Math.floor((x2-xx)/kx);
    y3=y2-Math.floor((y2-yy)/ky);
    xx=Math.floor((x3-Xa)*Math.cos(Fi*Math.PI/180) - (y3-Ya)*Math.sin(
Fi*Math.PI/180) + Xa);
    yy=Math.floor((x3-Xa)*Math.sin( Fi*Math.PI/180) + (y3-Ya)*Math.cos(
Fi*Math.PI/180) + Ya);
    if(xx>-1 && yy>-1){
        ctx2.fillRect(xx, yy, 1, 1);//Виведення відновленого фракталу
    }

}
let html='<div>Restore ISF</div>';
for(let el of elements){ //Виведення параметрів РСІФ по результатам отриманих від
//нейронної мережі
    html+='<br><div>Kx = '+el[0]*10).toFixed(2)+' Ky = '+el[1]*10).toFixed(2)+'
Xa = '+el[2]*100).toFixed(2)+' Ya = '+el[3]*100).toFixed(2)+' Fi =
'+el[4].toFixed(2)+'</div>';
}
$("#restore_sif").html(html);
//Формування посилання для збереження відновленого фрактального зображення
var imgageData = canvas2.toDataURL("image/png");
var newData = imgageData.replace(/^data:image\/png/, "data:application/octet-
stream");
$("#Download").show().attr("download", "fractal.png").attr("href", newData);
}

```

Результат роботи функції RecoveryFractal() відновлення фрактального зображення (Рис.4.24):

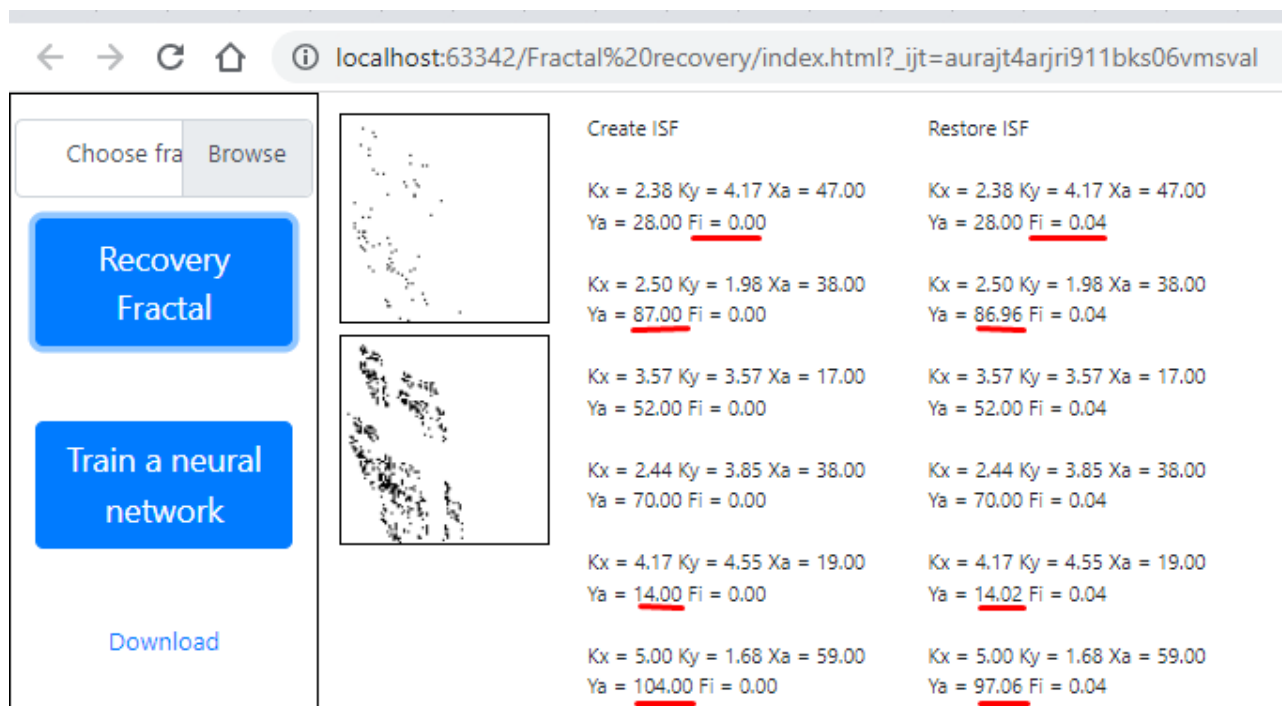


Рис.4.24. Результат роботи функції RecoveryFractal()

Як видно з результатів відновлення фрактального зображення Рис.4.24, відновлене фрактальне зображення майже ідентичне оригіналу Рис.4.20, має незначні відхилення від оригіналу (менше 4%), це пов'язано з тим, що спотворене зображення несе в собі мало інформації, та й навчання нейронної мережі потребує багато часу і багато згенерованих елементів для навчання.

4.3 Розробка бібліотеки для інтеграції запропонованих рішень у WEB-технологіях.

Занесення розроблених програм у Web-бібліотеки, дозволить нам:

- скоротити час написання програмних продуктів для роботи з даними алгоритмами;
- інтегрувати їх в інші програмні засоби та продукти;
- популяризувати наше програмне забезпечення.

Для початку створимо папочку web-fractal (Рис.4.25):

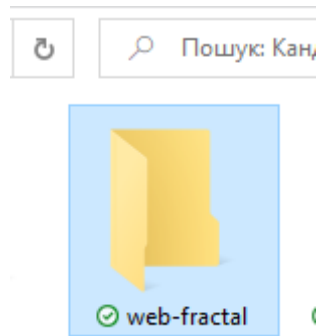


Рис. 4.25. Папка web-fractal

Наступним кроком нам необхідно весь javascript код з двох програм («Генератор фрактальних зображень типу «Cantor dust» на базі розробленого алгоритму» та «Система відновлення (спотворення) та відтворення фрактальних зображень типу «Cantor dust») винести у файл з назвою index.js (Рис. 4.26-4.27):

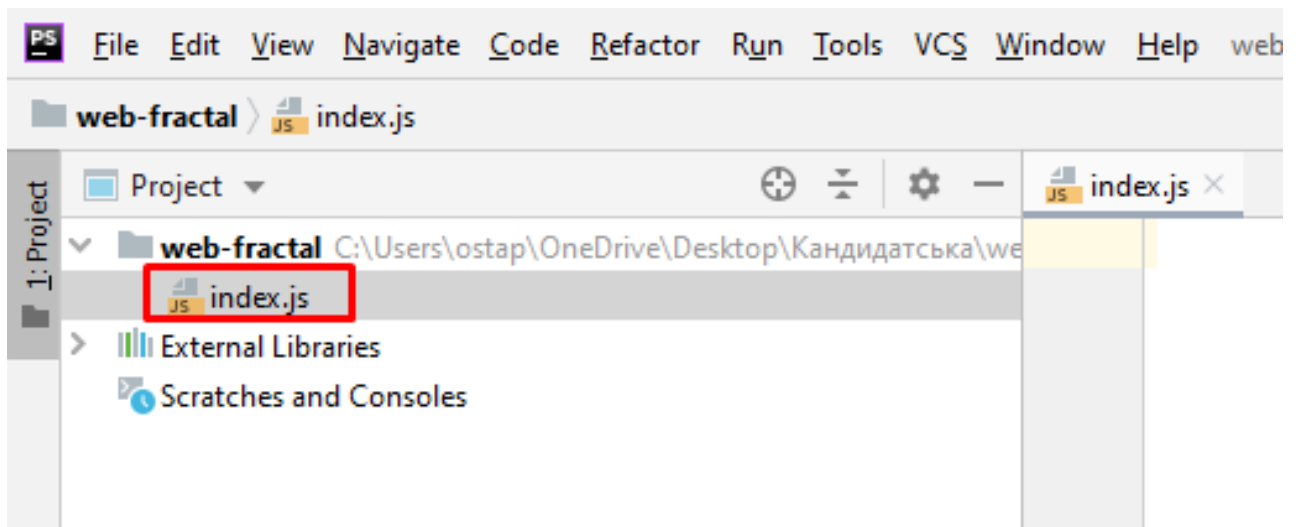
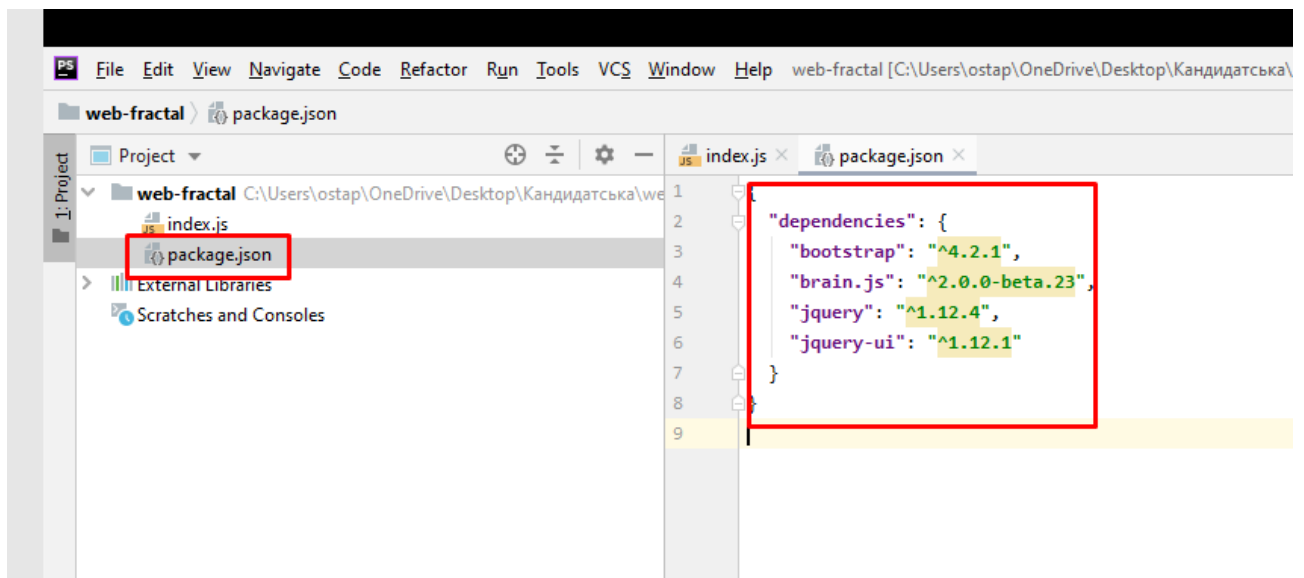


Рис. 4.26. Файл index.js

```
index.js ×
1  var elements=[{width:100,height:100,x:0,y:0,fi:0}];
2  var Quality=100000;
3
4  var gX=100;
5  var gY=100;
6  var canvas = document.getElementById( elementId: 'downloadFractalCanvas');
7  var ctx = canvas.getContext( contextId: '2d');
8  var net = new brain.NeuralNetwork();
9
10 //Fractal generator
11 $( function() {...} );
118
119 //Fractal recovery
120 $( function() {...} );
```

Рис. 4.27. Файл index.js

Перенесення коду не має складати проблем, так як програми писалися на javascript. Наступним кроком є вказання залежних бібліотек для нашої бібліотеки, а саме Brain.js, jQuery та Bootstrap, для цього створимо файл package.json та опишемо ці залежності в dependencies (Рис.4.28):



The screenshot shows an IDE window for a project named 'web-fractal'. The file explorer on the left shows 'package.json' selected. The main editor displays the content of 'package.json', which includes a 'dependencies' object with the following entries: 'bootstrap': '^4.2.1', 'brain.js': '^2.0.0-beta.23', 'jquery': '^1.12.4', and 'jquery-ui': '^1.12.1'. A red box highlights the 'dependencies' section in the code editor.

```
web-fractal > package.json
Project
└─ web-fractal C:\Users\ostap\OneDrive\Desktop\Кандидатська\we
   └─ index.js
      └─ package.json
         External Libraries
         Scratches and Consoles
1  {
2  "dependencies": {
3  "bootstrap": "^4.2.1",
4  "brain.js": "^2.0.0-beta.23",
5  "jquery": "^1.12.4",
6  "jquery-ui": "^1.12.1"
7  }
8  }
9  }
```

Рис. 4.28. Файл package.json

Наступним етапом написання web-бібліотеки є написання інструкції користування бібліотекою за допомогою файла README.md (Рис 4.29):

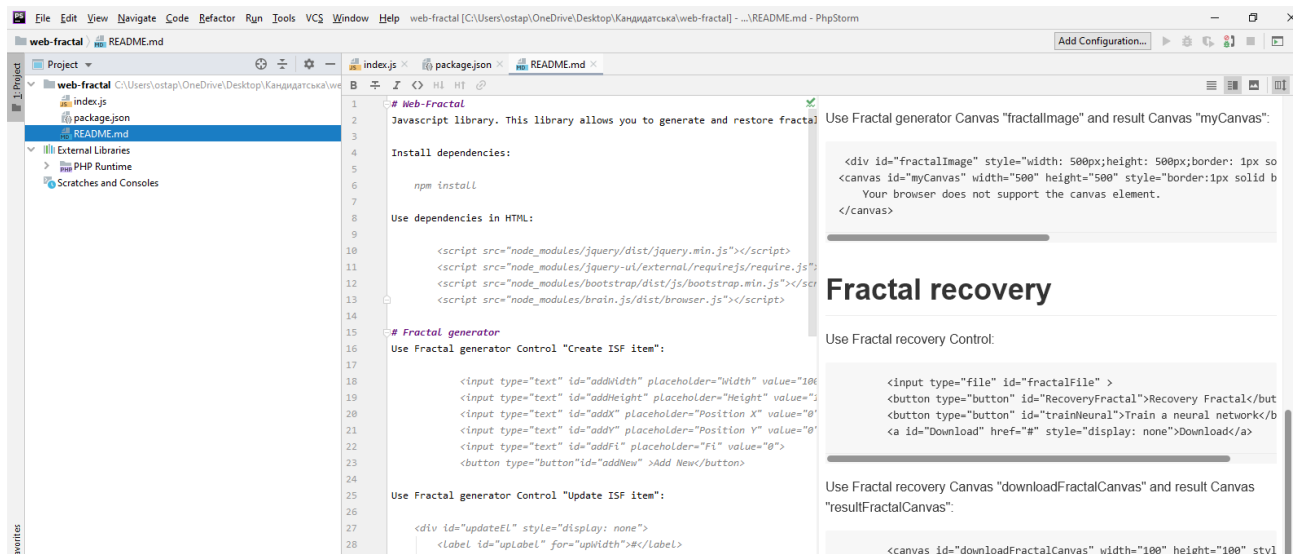


Рис. 4.29. README.md

Текс вмісту файла README.md:

Web-Fractal

Javascript library. This library allows you to generate and restore fractal images. It also allows machine learning to recognize fractal images and transform them into RSIF

Install dependencies:

```
npm install
```

Use dependencies in HTML:

```
<script src="node_modules/jquery/dist/jquery.min.js"></script>
<script src="node_modules/jquery-ui/external/requirejs/require.js"></script>
<script src="node_modules/bootstrap/dist/js/bootstrap.min.js"></script>
<script src="node_modules/brain.js/dist/browser.js"></script>
```

Fractal generator

Use Fractal generator Control "Create ISF item":

```
<input type="text" id="addWidth" placeholder="Width" value="100">
<input type="text" id="addHeight" placeholder="Height" value="100">
<input type="text" id="addX" placeholder="Position X" value="0">
<input type="text" id="addY" placeholder="Position Y" value="0">
<input type="text" id="addFi" placeholder="Fi" value="0">
<button type="button" id="addNew" >Add New</button>
```

Use Fractal generator Control "Update ISF item":

```
<div id="updateEl" style="display: none">
  <label id="upLabel" for="upWidth">#</label>
  <input type="text" id="upWidth" placeholder="Width">
  <input type="text" id="upHeight" placeholder="Height">
```



```




<button type="button" id="updateFr" data-number="0">Update</button>
<button type="button" id="deleteFr" data-number="0">Delete</button>
</div>

```

Use Fractal generator Control "Fractal":

```


<button type="button" id="createFr">Fractal</button>
<a id="btn-Convert-Html2Image" href="#" style="display: none">Download</a>

```

Use Fractal generator Canvas "fractalImage" and result Canvas "myCanvas":

```

<div id="fractalImage" style="width: 500px; height: 500px; border: 1px solid black;
display: inline-block; float: left; position: relative;"></div>
<canvas id="myCanvas" width="500" height="500" style="border: 1px solid black">
  Your browser does not support the canvas element.
</canvas>

```

Fractal recovery

Use Fractal recovery Control:

```



```

Use Fractal recovery Canvas "downloadFractalCanvas" and result Canvas "resultFractalCanvas":

```

<canvas id="downloadFractalCanvas" width="100" height="100"
style="border: 1px solid black">
  Your browser does not support the canvas element.
</canvas>
<canvas id="resultFractalCanvas" width="100" height="100" style="border: 1px
solid black">
  Your browser does not support the canvas element.
</canvas>

```

Use Fractal recovery results RISFs:

```

<div id="result_sif"></div>
<div id="restore_sif"></div>

```

=====

brain.js project

Тепер потрібно обрати репозиторій для завантаження нашої бібліотеки. Ми використаємо Git-сервіс Bitbucket [28-30].

Bitbucket є продуктом компанії Atlassian, яка відома своїми інструментами для спільної розробки, такими як JIRA і Confluence. Він підтримує Git і Mercurial VCS (не SVN) та написаний на Python з використанням веб-

фреймворку Django. Bitbucket доступний для Mac, Windows і Android через додаток і має відповідність стандарту безпеки SOC 2 Type II.

Bitbucket має кілька особливостей, включаючи вбудовані обговорення, pull-запити, вікі та можливість порівняння гілок і перегляд історії комітів.

Переваги використання Bitbucket такі:

Ціна. Bitbucket надає можливість мати необмежену кількість приватних репозиторіїв з до 5 співавторами.

Гнучкість. Bitbucket може імпортувати з репозиторіїв Git, CodePlex, Google Code, HG, SourceForge і SVN, тоді як GitHub обмежений Git, SVN, HG і TFS.

Пошук. Пошук у Bitbucket сканує синтаксис, щоб знайти відповідні визначення, а не лише імена змінних.

Інтеграції. Bitbucket має інтеграцію з такими інструментами, як HipChat, Trello, Bamboo, JIRA, Slack, Zapier, Bitium і Flowdock.

Деякі недоліки Bitbucket включають не дуже інтуїтивний інтерфейс порівняно з GitHub.

В цілому, Bitbucket є відмінним інструментом для контролю версій і відладки, який дозволяє інтегрувати різні інструменти і співпрацювати з командами. Він є чудовою альтернативою, яку варто розглянути.

Деякі компанії, які використовують Bitbucket, включають BBC Worldwide, Alibaba, AVG, Avast, Blackberry та інші.

Заходимо на сайт Bitbucket та створюємо новий приватний репозиторій (Рис. 4.30):

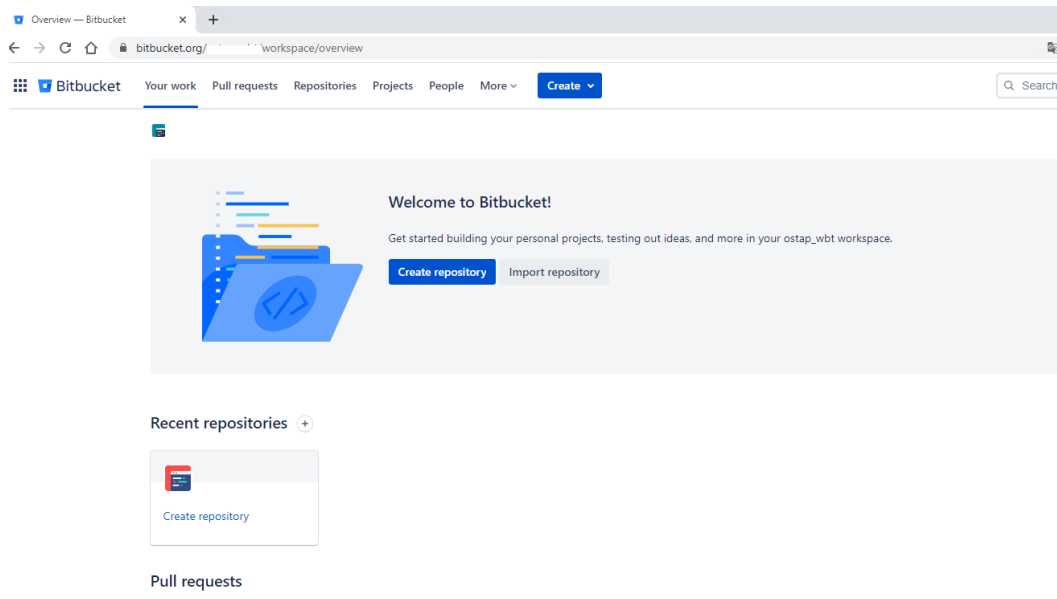


Рис. 4.30. Сайт Bitbucket

Наступним кроком створюємо приватний репозиторій web-fractal (Рис.4.31)

Рис. 4.31. Створення приватного репозиторію web-fractal

Переходимо на створений нами репозиторій web-fractal (Рис.4.32):

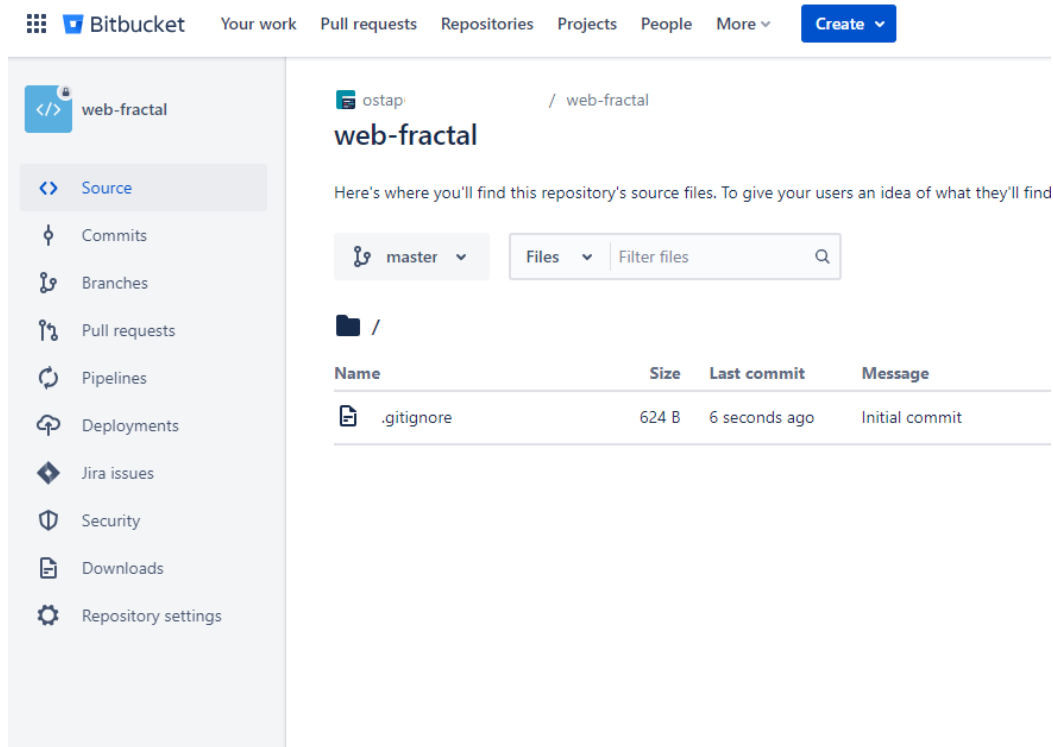


Рис. 4.32. Вікно репозиторію web-fractal

Наступним кроком додаємо файли у наш репозиторій web-fractal (Рис.4.32):

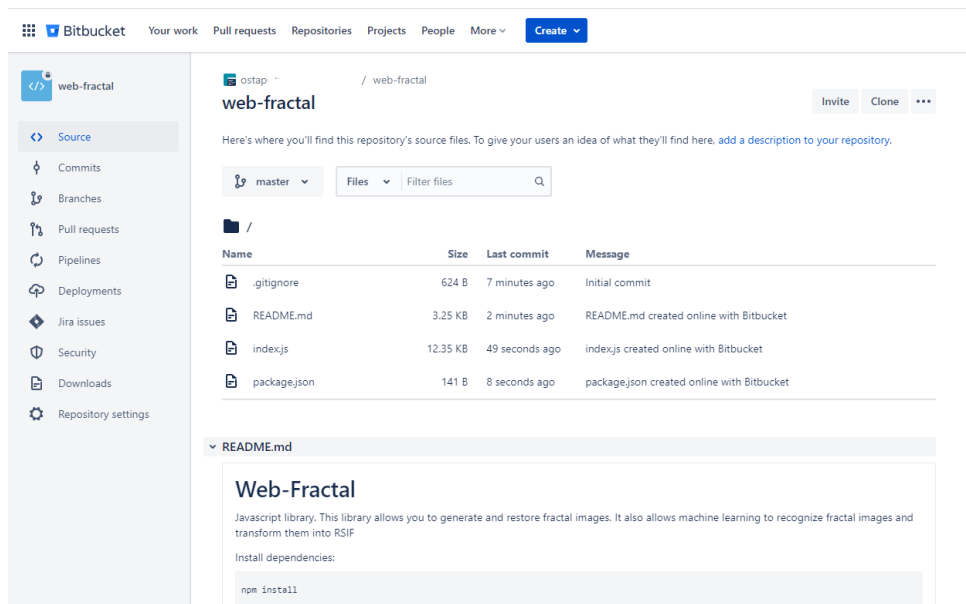


Рис. 4.33. Додавання бібліотеки до репозиторію web-fractal

Бібліотека готова до використання! Тепер спробуємо її встановити в себе локально на комп'ютер, відкриваємо вікно (Рис.4.33) та копіюємо команду для завантаження бібліотеки:

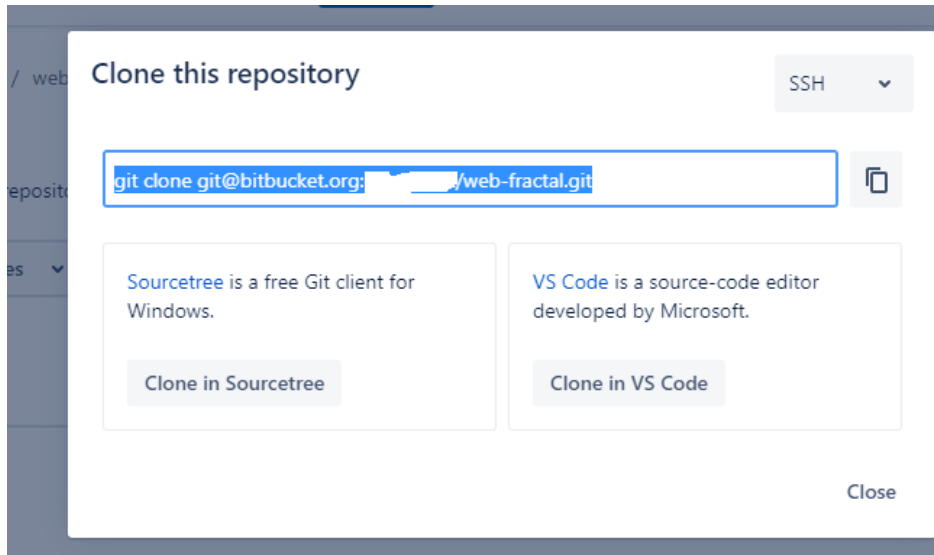


Рис. 4.33. Команда для завантаження бібліотеки з репозиторію web-fractal

Запускаємо команду в консолі (Рис 4.33):

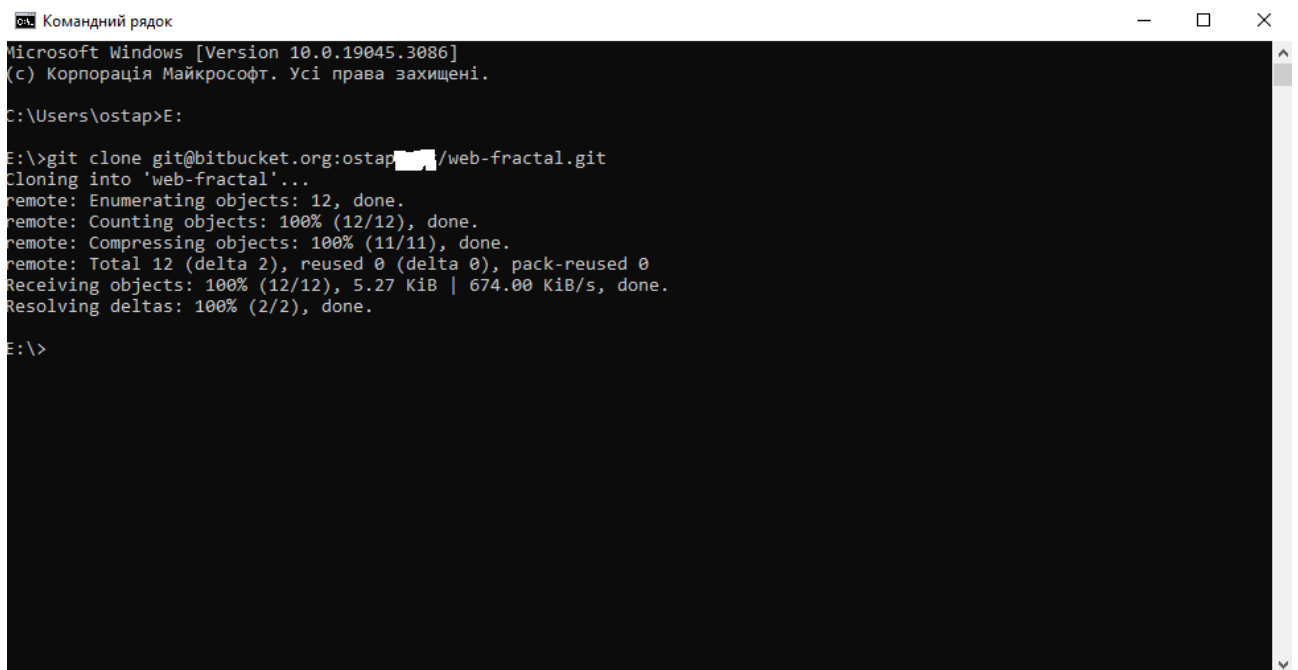


Рис. 4.34. Встановлення бібліотеки з репозиторію web-fractal

Перевіряємо результат установки бібліотеки з репозиторію web-fractal (Рис. 4.35)

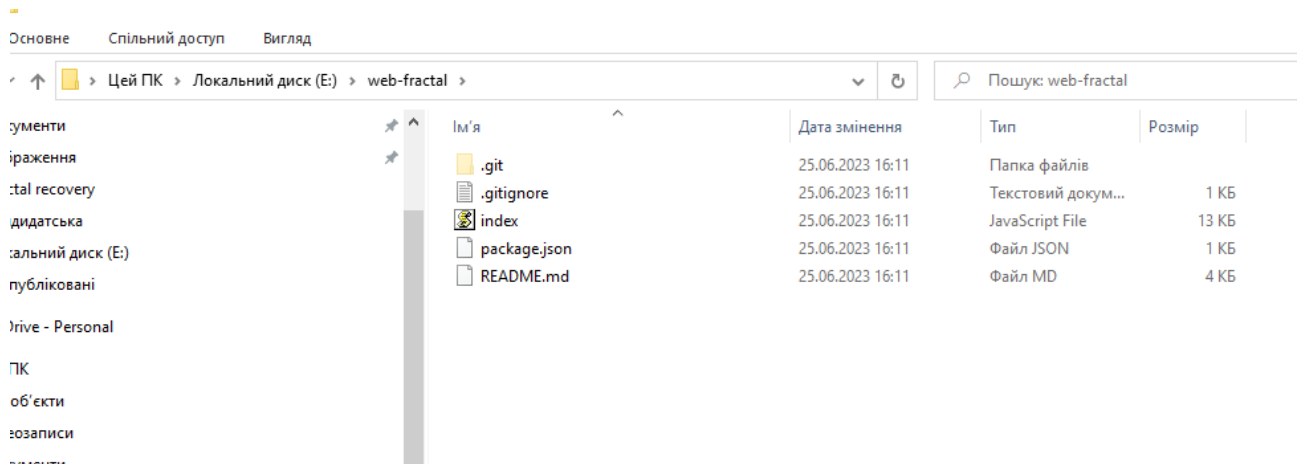


Рис. 4.35. Завантажена бібліотека з репозиторію web-fractal

Як бачимо, бібліотека успішно завантажена і готова до інтеграцій в інших проектах.

Висновки до 4-го розділу

У роботі здійснено реалізацію практичну реалізацію програмного забезпечення та бібліотек із використанням запропонованого методу у Web-технологіях.

В першій частині розділу реалізовано генератор фрактальних зображень типу «Cantor dust» на базі розробленого алгоритму з використанням удосконаленої РСІФ. Генератор реалізований за допомогою web-технологій, а саме мови розмітки гіпертексту HTML, мови програмування JavaScript, та каскадних таблиць стилів CSS. Для написання коду роботи генератора використані дві бібліотеки jQuery та Bootstrap. Розроблено простий та зрозумілий дизайн та функціонал генератора. Даний генератор дозволяє добавляти необмежену кількість ітераційних фігур, дозволяє змінювати їх параметри, довжину, ширину, місце розташування та повертати їх на певний кут. Генератор працює в реальному часі, так як в його основі лежить

удосконалена система ітераційних функцій. Також в генераторі реалізована функція зміни якості зображення, яка напряду залежить від кількості операцій РСІФ, що в свою чергу буде використано для навчання нейронної мережі розпізнавати та відновлювати фрактальні об'єкти. Генератор також дозволяє зберігати згенеровані зображення. Розташування фігур першої ітерації та їх параметрів можна виконувати як мишкою так і введенням їх в меню управління.

В другій частині розділу реалізовано систему відновлення (спотворення) та відтворення фрактальних зображень типу «Cantor dust» на базі розробленого алгоритму з використанням удосконаленої РСІФ. Програма реалізована за допомогою web-технологій, а саме мови розмітки гіпертексту HTML, мови програмування JavaScript, та каскадних таблиць стилів CSS. Для написання коду роботи системи використані три бібліотеки jQuery, Bootstrap та Brain.js. Бібліотека Brain.js дає можливість створити нейронну мережу, яку можна навчати і отримувати від неї розрахункові результати, такі як параметри удосконаленої РСІФ. Розроблено простий та зрозумілий дизайн та функціонал програми. Дана програма працює в реальному часі, так як в її основі лежить удосконалена РСІФ. Програма дозволяє генерувати випадкові фрактальні зображення і навчати нейронну мережу, подаючи на вхід зображення, а на вихід параметри удосконаленої РСІФ. Також дозволяє зберегти зображення, яке згенеровано з внесенням спотворення, що дозволяє навчати нейронну мережу розпізнавати фрактальні зображення з низькою роздільною здатністю. Також програма дозволяє завантажити фрактальне зображення для відновлення його якості, з подальшим виведенням параметрів удосконаленої РСІФ та збереженням відновленого фракталу.

В третій частині розділу розроблено бібліотеку для інтеграції запропонованого методу у WEB-технологіях, дві попередні програми, а саме їх функціональні можливості об'єднані в одну javascript бібліотеку, дана бібліотека містить інструкцію та пакетний файл package.json для підключення

бібліотек jQuery, Bootstrap і Brain.js. Реалізація бібліотеки досить проста, для отримання доступу до функціоналу в іншому продукті достатньо виставити відповідні id в тегах інтегруючого проекту. Бібліотека розміщена на приватному репозиторію Bitbucket, і має можливість встановлюватися за допомогою команди, дане рішення дає можливість скоротити час написання програмних продуктів для роботи з даними алгоритмами та інтегрувати їх в інші програмні засоби та продукти.

ОСНОВНІ РЕЗУЛЬТАТИ ТА ВИСНОВКИ

Сукупність наукових положень, сформульованих та обґрунтованих в дисертаційній роботі, становить розв'язок науково-практичного завдання розробки удосконалених методів побудови фрактальних зображень на основі рандомізованої системи ітераційних функцій (РСІФ) та розвитку методів їх розпізнавання, захисту та шифрування із використанням нейронних мереж.

Основні результати роботи полягають у наступному:

1. Проведено дослідження основних типів фракталів і способів їх створення. Серед них розглянуті геометричні фрактали, такі як множина Кантора, крива Гільберта-Пеано, крива Коха, сніжинка Коха, "дракон" Хартера-Хейтуея, H-фрактал, Крива Мінковського, Крива Леві, квадрат Серпінського, трикутник Серпінського, T-фрактал і Дерево Піфагора. Також розглянуті алгебраїчні фрактали, зокрема множина Жюліа і Множина Мандельброта, а також стохастичні фрактали. Проаналізовані різні методи та алгоритми побудови фрактальних структур, такі як ітеративний метод (системи ітераційних функцій), метод рекурентних співвідношень і метод випадкових процесів. Розглянуті переваги і недоліки кожного методу, що дозволило визначити найкращий метод для побудови фрактальних структур. У результаті обрано ітеративний метод, зокрема рандомізовану систему ітераційних функцій (РСІФ), і наведені його переваги порівняно з методом рекурентних співвідношень і методом випадкових процесів.

2. Удосконалено метод побудови фрактальних структур за допомогою рандомізованої системи ітераційних функцій. Цей метод відрізняється від відомих, можливістю відтворення та генерації фрактальних структур в реальному часі. Важливою особливістю цього методу є його оптимізованість, що є ключовим аспектом для зменшення обчислювальної складності. На основі проведених розрахунків кількості операцій, необхідних для побудови фрактальних зображень з обмеженою роздільною здатністю, встановлено, що використання рандомізованої системи ітераційних функцій (РСІФ) переважає

детерміновану систему ітераційних функцій (ДСІФ) в обчислювальних операціях пропорційно до роздільної здатності. Це вказує на те, що РСІФ є більш ефективним та оптимізованим методом для створення фрактальних структур в умовах обмеженої роздільної здатності. Ця перевага РСІФ відкриває нові можливості для генерації фрактальних зображень в реальному часі та спрощує обчислювальні вимоги, що робить його важливим інструментом у візуалізації та обробці графічних даних.

3. Розвинуто метод розпізнавання фрактального зображення за допомогою нейронної мережі, яка навчена на базі розробленого методу побудови фрактальних структур з використанням рандомізованої системи ітераційних функцій. Особливістю цього методу є те, що нейронна мережа самостійно генерує необхідний набір даних для свого навчання. Даний метод виявляється дуже ефективним, оскільки відновлення або спотворення фрактального зображення відбувається в реальному часі.

4. Розвинуто спосіб захисту документів, використовуючи побудову фрактальних структур з використанням рандомізованої системи ітераційних функцій. Цей метод відрізняється від існуючих алгоритмів побудови водяних знаків, оскільки він прив'язує номер документа до фракталу, який у вигляді водяного знаку наноситься на сам документ. Застосування цього методу дозволяє подвійну та потрійну перевірку документа. Цей метод виявляється дуже ефективним, оскільки побудова фрактального зображення відбувається в реальному часі.

5. Розроблено модель швидкого автоматичного визначення фрактальної розмірності зображення із використанням нейронних мереж, яка навчена на основі удосконаленої рандомізованої системи ітераційних функцій. Важливою особливістю пропонованої моделі є самостійність генерування набору даних для навчання нейронної мережі. Визначення фрактальної розмірності здійснюється у реальному часі. Це відкриває можливості для застосування моделі в областях, де важливо отримувати оперативну інформацію про

фрактальну розмірність зображень, такі як медична діагностика, обробка зображень, визначення параметрів телекомунікаційних фрактальних антен (розмір антени, форма, товщина та розташування) .

6. Розроблено метод шифрування фрактальних зображень, використовуючи матриці перетворень, що запобігають дешифруванню з використанням нейронних алгоритмів. Ефективність цього методу полягає у використанні рандомізованого алгоритму побудови матриць перетворень, що гарантує неможливість розшифрування інформації. Запропонований метод шифрування фрактальних зображень відрізняється від існуючих підходів тим, що використовує надлишковість та рандомізовані матриці переміщення пікселів, впроваджує зміни в яскравість, виконує інверсію та модифікує кольори зображення з метою захисту від можливості дешифрування. Цей підхід дозволяє забезпечити конфіденційність та цілісність інформації під час передачі даних, при цьому зберігаючи прозорість у передачі і забезпечуючи безпеку від розкриття інформації. Цей метод є ефективним для будь-яких існуючих зображень і дозволяє проводити шифрування в реальному часі з використанням обчислювальних ресурсів низької потужності.

7. Реалізовано генератор фрактальних зображень типу «Cantor dust» на базі розробленого алгоритму з використанням удосконаленої РСІФ. Генератор реалізований за допомогою web-технологій, а саме мови розмітки гіпертексту HTML, мови програмування JavaScript, та каскадних таблиць стилів CSS. Для написання коду роботи генератора використані дві бібліотеки: jQuery та Bootstrap. Розроблено простий та зрозумілий дизайн та функціонал генератора. Даний генератор дозволяє добавляти необмежену кількість ітераційних фігур, дозволяє змінювати їх параметри, довжину, ширину, місце розташування та повертати їх на певний кут. Генератор працює в реальному часі, так як в його основі лежить удосконалена система ітераційних функцій. Також в генераторі реалізована функція зміни якості зображення, яка напряму залежить від кількості операцій РСІФ, що в свою чергу використано для навчання нейронної

мережі розпізнавати та відновлювати фрактальні об'єкти. Важливо відзначити, що точність створення фрактальних структур виявилася надзвичайно високою, і відносна похибка складає менше 0,001%. Генератор також дозволяє зберігати згенеровані зображення. Розташування фігур першої ітерації та їх параметрів можна виконувати як мишкою так і введення їх в меню управління.

8. Реалізовано програмну систему відновлення (спотворення) та відтворення фрактальних зображень типу «Cantor dust» на базі розробленого методу побудови фрактальних структур. Систему реалізовано за допомогою web-технологій, а саме мови розмітки гіпертексту HTML, мови програмування JavaScript, та каскадних таблиць стилів CSS. Для написання коду роботи системи використані три бібліотеки: jQuery, Bootstrap та Brain.js. Бібліотека Brain.js дає можливість створити нейронну мережу, яку можна навчати і отримувати від неї розрахункові результати, такі як параметри удосконаленої РСІФ. Розроблено простий та зрозумілий дизайн та функціонал програми. Дана програма працює в реальному часі, так як в її основі лежить удосконалена РСІФ. Програма дозволяє генерувати випадкові фрактальні зображення і навчати нейронну мережу, подаючи на вхід зображення, а на вихід параметри удосконаленої РСІФ. Також дозволяє зберегти зображення, яке згенеровано з внесенням спотворення, що дозволяє навчати нейронну мережу розпізнавати фрактальні зображення з низькою роздільною здатністю. Також програма дозволяє завантажити фрактальне зображення для відновлення його якості з подальшим виведенням параметрів удосконаленої РСІФ та збереженням відновленого фракталу з точністю 0,01%.

9. Розроблена і впроваджена бібліотека для інтеграції запропонованого методу в WEB-технології. Програмна система відновлення фрактальних структур та модель генерації фракталів, інтегровані в одну JavaScript бібліотеку. Ця бібліотека включає в себе інструкцію та файл package.json для підключення бібліотек jQuery, Bootstrap і Brain.js. Реалізація бібліотеки досить проста, і для отримання доступу до її функціоналу в іншому проєкті достатньо встановити

відповідні ідентифікатори в тегах інтегрованого проєкту. Бібліотека розміщена на приватному репозиторії Bitbucket і може бути встановлена за допомогою відповідної команди. Це рішення дозволяє скоротити час написання програмних продуктів для роботи з цими алгоритмами і інтегрувати їх в інші програмні засоби та продукти. Розроблені практичні рішення підкреслюють значення даної бібліотеки як інструмента для розв'язання різних завдань у веб-розробці, аналізі зображень та інших областях, де важливі фрактальні структури та їх обробка.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Н. П. Щекань, “Геометричні та алгебраїчні фрактальні методи в інформаційних технологіях обробки і аналізу потоків даних”, *Моделювання та інформ. системи в економіці: зб. наук. праць /відп. ред. В. К. Галіцин*, no. 95, pp. 206-219, 2018.
2. Н. В. Новікова, О. В. Сагай, “Використання фракталів в математичному моделюванні складних природних систем”, *Науковий журнал «ЛОГОΣ. Мистецтво наукової думки»*, no. 1, pp. 147-148, 2018,
3. Б. Мандельброт, “Фрактальна геометрія природи / Бенуа Мандельброт”, *Інститут комп'ютерних досліджень*, 2002.
4. О. В. Капустян, В. В. Пічкур, В. В. Собчук, “Теорія динамічних систем. Навчальний посібник”, *Вежа-Друк, Луцьк*, 2020.
5. Anguera *et al.*, “Fractal antennas: An historical perspective,” *Fractal Fract.*, vol. 4, no. 1, p. 3, 2020.
6. A. Bakytbekov, A. R. Maza, M. Nafe, and A. Shamim, “Fully inkjet printed wide band cantor fractal antenna for RF energy harvesting application,” in *2017 11th European Conference on Antennas and Propagation (EUCAP)*, 2017.
7. S. K. Terlapu, P. S. R. Chowdary, C. Jaya, V. V. S. S. Sameer Chakravarthy, and S. C. Satpathy, “On the design of fractal UWB wide-slot antenna with notch band characteristics,” in *Lecture Notes in Electrical Engineering*, Singapore: Springer Singapore, 2018, pp. 907–912.
8. A. Reha, A. El Amri, O. Benhammouch, A. Oulad Said, A. El Ouadih, and M. Bouchouirbat, “CPW-fed slotted CANTOR Set fractal antenna for WiMAX and WLAN applications,” *Int. J. Microw. Wirel. Technol.*, vol. 9, no. 4, pp. 851–857, 2017.
9. Y.-S. Li, X.-D. Yang, C.-Y. Liu, and T. Jiang, “Analysis and investigation of a cantor set fractal uwb antenna with a notch-band characteristic,” *Prog. Electromagn. Res. B Pier B*, vol. 33, pp. 99–114, 2011.

10. J. A. Bossard, L. Lin, and D. H. Werner, “Evolving random fractal Cantor superlattices for the infrared using a genetic algorithm,” *J. R. Soc. Interface*, vol. 13, no. 114, p. 20150975, 2016.
11. É. C. Braz and A. L. P. de S. Campos, “Multiband frequency selective surfaces with a modified multifractal cantor geometry,” *J. Microw. Optoelectron. Electromagn. Appl.*, vol. 13, no. 2, pp. 111–121, 2014.
12. M. Kumar and V. Nath, “Multiband CPW-fed Circular Microstrip Antenna with Modified Cantor Fractal Slot for DCS/GPS/WiMAX/WLAN/HiPERLAN2 Applications,” in *2018 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, 2018.
13. A. R. Maza, “Inkjet-printed Ultra wide band fractal antennas.” *KAUST Research Repository*, 2012.
14. B.B. Mandelbrot, “The Fractal Geometry of Nature”, *W.H. Freeman & Company: New York, NY, USA*, 1999
15. A. A. Zahed, A. H. El-Hag, N. Qaddoumi, R. Hussein, and K. B. Shaban, “Comparison of different fourth order Hilbert fractal antennas for partial discharge measurement,” *IEEE Trans. Dielectr. Electr. Insul.*, vol. 24, no. 1, pp. 175–182, 2017.
16. T. Dong *et al.*, “Design of electrically small Hilbert fractal NFRP magnetic monopole antennas,” *J. Electromagn. Waves Appl.*, vol. 33, no. 4, pp. 454–464, 2019.
17. D.-O. Kim, C.-Y. Kim, and D.-G. Yang, “Flexible Hilbert-curve loop antenna having a triple-band and omnidirectional pattern for WLAN/WiMAX applications,” *Int. J. Antennas Propag.*, vol. 2012, pp. 1–9, 2012.
18. J. Li, T. Jiang, C. Cheng, and C. Wang, “Hilbert fractal antenna for UHF detection of partial discharges in transformers,” *IEEE Trans. Dielectr. Electr. Insul.*, vol. 20, no. 6, pp. 2017–2025, 2013.
19. Y. Fan, H. Liu, X. Liu, Y. Cao, Z. Li, and M. M. Tentzeris, “Novel coated differentially fed dual-band fractal antenna for implantable medical devices,” *IET Microw. Antennas Propag.*, vol. 14, no. 2, pp. 199–208, 2020.

20. S. Suganthi, S. Raghavan, D. Kumar, and J. Arputha Vijaya Selvi, "Study of compact Hilbert curve fractal antennas for implantable medical applications," 2012.
21. T. Li, J. Zhang, B. Cheng, X. Lei, Z. Xu, and J. Gao, "Compact wideband dual-frequency antenna based on a simplified composite right/left-handed transmission line with Hilbert curve loading," *Int. J. Antennas Propag.*, vol. 2019, pp. 1–8, 2019.
22. J. K. Ali, "Microstrip-fed printed slot antennas based on Hilbert-type space-filling curves for wireless communication systems," 2013.
23. Y. Wang, Z. Wang, and J. Li, "UHF Moore fractal antennas for online GIS PD detection," *IEEE Antennas Wirel. Propag. Lett.*, vol. 16, pp. 852–855, 2017.
24. J. A. Russer, "Printed self-complementary Hilbert curve (SCHC) fractal broad-band antenna," in *2018 Baltic URSI Symposium (URSI)*, 2018.
25. R. S. Daniel, "Asymmetric coplanar strip-fed with Hilbert curve fractal antenna for multiband operations," *Wirel. Pers. Commun.*, vol. 116, no. 1, pp. 791–803, 2021.
26. C. Elavarasi and T. Shanmuganatham, "Multiband SRR loaded Koch star fractal antenna," *Alex. Eng. J.*, vol. 57, no. 3, pp. 1549–1555, 2018.
27. N. Sharma and V. Sharma, "A design of Microstrip Patch Antenna using hybrid fractal slot for wideband applications," *Ain Shams Eng. J.*, vol. 9, no. 4, pp. 2491–2497, 2018.
28. M.-V. Nichita, M.-A. Paun, V.-A. Paun, and V.-P. Paun, "On the 5G communications: Fractal-shaped antennas for PPDR applications," *Complexity*, vol. 2021, pp. 1–12, 2021.
29. A. Jamil, M. Z. Yusoff, and N. Yahya, "Analysis of a hybrid fractal curve antenna using the segmentation method," *Eng. Rep.*, vol. 2, no. 11, 2020.
30. A. Jamil, M. Rauf, A. Sami, A. Ansari, and M. Dawood Idrees, "A wideband hybrid fractal ring antenna for WLAN applications," *Int. J. Antennas Propag.*, vol. 2022, pp. 1–8, 2022.

31. R. Colburn, "Why mobile phones can do so many things: The invention of the fractal antenna," *IEEE-USA InSight*, 04-Aug-2021.
32. D. Tumakov, D. Chikrin, and P. Kokunin, "Miniaturization of a Koch-type fractal antenna for WI-Fi applications," *Fractal Fract.*, vol. 4, no. 2, p. 25, 2020.
33. C. P. Baliarda, J. Romeu and A. Cardama, "The Koch monopole: a small fractal antenna," in *IEEE Transactions on Antennas and Propagation*, vol. 48, no. 11, pp. 1773-1781, Nov. 2000, doi: 10.1109/8.900236.
34. O. Benkhadda *et al.*, "A miniaturized Tri-wideband Sierpinski hexagonal-shaped fractal antenna for wireless communication applications," *Fractal Fract.*, vol. 7, no. 2, p. 115, 2023.
35. O. F. Gonzales Palacios, R. E. Diaz Vargas, J. A. Heraud Perez, and S. B. Correa Erazo, "S-band koch snowflake fractal antenna for cubesats," in *2016 IEEE ANDESCON*, 2016.
36. C. Ben Nsir, J. M. Ribero, C. Boussetta, and A. Gharsallah, "Design of a 1×2 CPW fractal antenna array on Plexiglas substrate with defected ground plane for telecommunication applications," *Eng. Technol. Appl. Sci. Res.*, vol. 11, no. 6, pp. 7897–7903, 2021.
37. Z. Yu, J. Yu, C. Zhu, and Z. Yang, "An improved Koch snowflake fractal broadband antenna for wireless applications," in *2017 IEEE 5th International Symposium on Electromagnetic Compatibility (EMC-Beijing)*, 2017.
38. S. Sivasundarapandian and C. D. Suriyakala, "A planar multiband Koch snowflake fractal antenna for cognitive radio," *Int. J. Microw. Wirel. Technol.*, vol. 9, no. 2, pp. 335–339, 2017.
39. Y. K. Choukiker and S. K. Behera, "Wideband frequency reconfigurable Koch snowflake fractal antenna," *IET Microw. Antennas Propag.*, vol. 11, no. 2, pp. 203–208, 2017.
40. M. Manohar, "Miniaturised low-profile super-wideband Koch snowflake fractal monopole slot antenna with improved BW and stabilised radiation pattern," *IET Microw. Antennas Propag.*, vol. 13, no. 11, pp. 1948–1954, 2019.

41. W.-C. Weng and C.-L. Hung, “An H-Fractal Antenna for Multiband Applications,” *IEEE Antennas Wirel. Propag. Lett.*, vol. 13, pp. 1705–1708, 2014.
42. W.-C. Weng and C.-L. Hung, “An H-Fractal Antenna for Multiband Applications,” *IEEE Antennas Wirel. Propag. Lett.*, vol. 13, pp. 1705–1708, 2014.
43. R. Yadav and L. Malviya, “UWB antenna and MIMO antennas with bandwidth, band-notched, and isolation properties for high-speed data rate wireless communication: A review,” *Int. J. RF Microw. Comput-Aid. Eng.*, vol. 30, no. 2, 2020.
44. A. P. Singh and M. Aggarwal, “Development of an inverted-h shaped fractal microstrip patch antenna for cognitive radio,” *J. Inst. Eng. (India) Ser. B*, vol. 102, no. 1, pp. 49–57, 2021.
45. Dhivyabharathi and K. Ramprakash, “FRACTAL ANTENNA FOR MULTIBAND APPLICATIONS,” 2016.
46. F. Miyamaru *et al.*, “Terahertz electric response of fractal metamaterial structures,” *Phys. Rev. B Condens. Matter Mater. Phys.*, vol. 77, no. 4, 2008.
47. N. Sharma and S. S. Bhatia, “Comparative analysis of hybrid fractal antennas: A review,” *Int. J. RF Microw. Comput-Aid. Eng.*, vol. 31, no. 9, 2021.
48. I. S. Bangi and J. S. Sivia, “Minkowski and Hilbert curves based hybrid fractal antenna for wireless applications,” *Int. J. Electron. Commun.*, vol. 85, pp. 159–168, 2018.
49. I. S. Bangi and J. S. Sivia, “Moore, minkowski and Koch curves based hybrid fractal antenna for multiband applications,” *Wirel. Pers. Commun.*, vol. 108, no. 4, pp. 2435–2448, 2019.
50. M.-A. Paun, M.-V. Nichita, V.-A. Paun, and V.-P. Paun, “Minkowski’s loop fractal antenna dedicated to sixth generation (6G) communication,” *Fractal Fract.*, vol. 6, no. 7, p. 402, 2022.
51. A. K. Vallappil, B. A. Khawaja, M. K. A. Rahim, M. Uzair, M. Jamil, and Q. Awais, “Minkowski–Sierpinski fractal structure-inspired 2×2 antenna array for use in next-generation wireless systems,” *Fractal Fract.*, vol. 7, no. 2, p. 158, 2023.

52. G. Bharti and J. S. Sivia, "A design of compact wideband antenna based on hybridization of Minkowski fractal curves on hexagonal patch and partial ground plane with truncated corners," *Wirel. Pers. Commun.*, vol. 124, no. 2, pp. 1609–1621, 2022.
53. S. Manafi and H. Deng, "Design of a small modified Minkowski fractal antenna for passive deep brain stimulation implants," *Int. J. Antennas Propag.*, vol. 2014, pp. 1–9, 2014.
54. I. S. Bangi and J. S. Sivia, "A compact hybrid fractal antenna using Koch and Minkowski curves," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2018.
55. A. S. Brar and J. S. Sivia, "Modified Minkowski fractal antenna for wireless applications," *J. Inst. Eng. (India) Ser. B*, vol. 99, no. 4, pp. 391–396, 2018.
56. E. C. Lee *et al.*, "Design and fabrication of a flexible Minkowski fractal antenna for VHF applications," *European Conference on Antennas and Propagation*, 2011.
57. S. Saleekaw, C. Mahatthanajatuphat, P. Akkaraekthalin, and M. Krairiksh, "Monopole antennas with modified Minkowski fractal geometry," in *2008 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, 2008.
58. Journal of Computer Science IJCSIS, G. Bharti, and J. Singh, "A compact hybrid Minkowski fractal antenna for C and X – band applications," 2017.
59. V. Radonić, K. Palmer, G. Stojanović, and V. Crnojević-Bengin, "Flexible Sierpinski carpet fractal antenna on a Hilbert slot patterned ground," *Int. J. Antennas Propag.*, vol. 2012, pp. 1–7, 2012.
60. M. Jena, G. P. Mishra, A. B. Sahoo, and B. B. Mangaraj, "An improved Sierpinski fractal MIMO array antenna with enhanced isolation for next generation wireless applications," 2020.

61. R. Mohanamurali and T. Shanmuganatham, "Sierpinski carpet fractal antenna for multiband applications," *Int. J. Comput. Appl.*, vol. 39, no. 14, pp. 19–23, 2012.
62. N. Lu and X. Xu, "Sierpinski carpet fractal antenna in third iteration," in *2013 IEEE INTERNATIONAL CONFERENCE ON MICROWAVE TECHNOLOGY & COMPUTATIONAL ELECTROMAGNETICS*, 2013.
63. P. Iyampalam and I. Ganesan, "Complementary Sierpinski Knopp fractal antenna for emergency management system," *Int. J. Microw. Wirel. Technol.*, vol. 12, no. 10, pp. 1029–1038, 2020.
64. H. Jiang, W. Li, and Z. Xue, "Modified microstrip aperture coupled patch antenna with Sierpinski fractal geometry," *Int. J. Antennas Propag.*, vol. 2014, pp. 1–8, 2014.
65. M. Ranjan Jena, B. B. Mangaraj, and R. Pathak, "Design of a novel Sierpinski fractal antenna arrays based on circular shapes with low side lobes for 3G applications," *Am. J. Electr. Electron. Eng.*, vol. 2, no. 4, pp. 137–140, 2014.
66. R. Gurjar, D. K. Upadhyay, B. K. Kanaujia, and A. Kumar, "A compact modified sierpinski carpet fractal UWB MIMO antenna with square-shaped funnel-like ground stub," *Int. J. Electron. Commun.*, vol. 117, no. 153126, p. 153126, 2020.
67. F. De Nicola *et al.*, "Multiband plasmonic Sierpinski carpet fractal antennas," *ACS Photonics*, vol. 5, no. 6, pp. 2418–2425, 2018.
68. A. Karimbu Vallappil, B. A. Khawaja, I. Khan, and M. Mustaqim, "Dual-band Minkowski–Sierpinski fractal antenna for next generation satellite communications and wireless body area networks," *Microw. Opt. Technol. Lett.*, vol. 60, no. 1, pp. 171–178, 2018.
69. A. Karmakar, "Fractal antennas and arrays: a review and recent developments," *Int. J. Microw. Wirel. Technol.*, vol. 13, no. 2, pp. 173–197, 2021.
70. V. A. S. Ponnappalli and P. V. Y. Jayasree, "Fractal array antennas and applications," in *Emerging Microwave Technologies in Industrial, Agricultural, Medical and Food Processing*, K. Y. You, Ed. London, England: InTech, 2018.

71. V. P. Joseph, "Multiband fractal patch antenna: Modification and miniaturization of Sierpinski gasket," *Int. J. Sci. Res. (Raipur)*, vol. 4, no. 12, pp. 825–829, 2015.
72. K. Ismail and S. H. Ishak, "Sierpinski gasket fractal antenna with defected ground structure (DGS)," in *2012 International Conference on ICT Convergence (ICTC)*, 2012.
73. "A fun triangular fractal antenna with laboratory test results," *Antenna Test Lab Company*, 16-Mar-2021. [Online]. Available: <https://antennatestlab.com/biz-cards/pcb0046-fractal-antenna>. [Accessed: 12-Oct-2023].
74. M. Waqas, Z. Ahmed, and M. B. Ihsan, "Multiband Sierpinski fractal antenna," in *2009 IEEE 13th International Multitopic Conference*, 2009.
75. A. Singh and S. Singh, "Design and optimization of a modified Sierpinski fractal antenna for broadband applications," *Appl. Soft Comput.*, vol. 38, pp. 843–850, 2016.
76. T. Benyetho *et al.*, "Fractal multiband planar antenna for wireless power transmission," *2014 International Renewable and Sustainable Energy Conference (IRSEC)*, Ouarzazate, Morocco, 2014, pp. 407-410, doi: 10.1109/IRSEC.2014.7059746.
77. O. A. Safia and M. Nedil, "Ultra-broadband V-band fractal T-square antenna," *2017 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*, San Diego, CA, USA, 2017, pp. 2611-2612, doi: 10.1109/APUSNCURSINRSM.2017.8073348.
78. M. Kaur and A. P. Deepinder Singh, "Design of penta band T-shaped fractal patch antenna for wireless applications," *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, 2016, pp. 675-679.
79. R. Kadwane and J. Singh, "Compact printed multiband fractal antenna for C, X and ku band applications: Design and analysis," in *Proceedings of Trends in*

Electronics and Health Informatics, Singapore: Springer Nature Singapore, 2022, pp. 501–512.

80. A. Husain, M. N. Nanda, M. S. Chowdary, and M. Sajid, “Fractals: An eclectic survey, part II,” *Fractal Fract.*, vol. 6, no. 7, p. 379, 2022.

81. T. N. Cao, T. Q. Hoa Nguyen, T. N. Nguyen, N. H. Nguyen, and D. T. Le, “Improved performance for antenna based on a combination of fractal geometry with CSRR,” *J. Inf. Telecommun.*, vol. 7, no. 2, pp. 144–154, 2023.

82. Mircea and R. Baic, “Fractal Antenna Applications,” in *Microwave and Millimeter Wave Technologies from Photonic Bandgap Devices to Antenna and Applications*, I. Minin, Ed. London, England: InTech, 2010.

83. Malvika and R. Yadav, “Design and Analysis of Fractal Antennas for Wideband, 3 – 8 GHz Frequency Range,” 2017.

84. A. Aggarwal and M. V. Kartikeyan, “Pythagoras tree: A fractal patch antenna for multi-frequency and ultra-wide bandwidth operations,” *Prog. Electromagn. Res. C Pier C.*, vol. 16, pp. 25–35, 2010.

85. S. Sahni, V. Gupta, and S. Negi, “Design and Simulation of Pythagorean Tree Monopole Fractal Antenna,” *Ijser.org*. [Online]. Available: <https://www.ijser.org/paper/Design-and-Simulation-of-Pythagorean-Tree-Monopole-Fractal-Antenna.html>. [Accessed: 12-Oct-2022].

86. S. Khobragade, S. Nalbalwar, and A. Nandgaonkar, “Study of fractal tree antenna for multiband applications,” in *Proceedings of the International Conference on Communication and Signal Processing 2016 (ICCASP 2016)*, 2017.

87. M. Miftahul Amri and G. Ramadhan Faqih, “Recent trends in fractal antenna research for in-body communications,” *Bincang Sains dan Teknologi*, vol. 1, no. 02, pp. 63–75, 2022.

88. M. Kaur and J. S. Sivia, “Tree-shaped hybrid fractal antenna for biomedical applications using ANN and PSO,” *Research Square*, 2021.

89. A. Reha and A. O. Said, “Tri-Band Fractal Antennas for RFID Applications,” *Wirel. Eng. Technol.*, vol. 04, no. 04, pp. 171–176, 2013.

90. К. О. Струк, А. В. Кавецька, Н. В. Сачанюк-Кавецька, “Варіант класифікації фракталів”, *Матеріали Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)»*, 2023.

91. V. P. Silva Neto and A. G. D’Assunção, “Iterative full-wave analysis of Mandelbrot-inspired fractal patch antenna on textile substrate for UWB applications,” *Int. J. Antennas Propag.*, vol. 2017, pp. 1–6, 2017.

92. M. J. Duarte, V. P. da Silva Neto, and A. G. D’Assunção, “Synthesis and mechanical reconfiguration of ground plane tilted microstrip antennas based on tetra-circle fractals,” *J. Microw. Optoelectron. Electromagn. Appl.*, vol. 19, no. 2, pp. 228–241, 2020.

93. D. R. Minervino, A. G. D’Assunção, and C. Peixeiro, “Mandelbrot fractal microstrip antennas,” *Microw. Opt. Technol. Lett.*, vol. 58, no. 1, pp. 83–86, 2016.

94. В.В. Демченко, Є.В. Бородавка, “Геометричне моделювання і комп’ютерна графіка”, *К.: КНУБА*, pp. 288 с, 2010, ISBN 966-627-061-7

95. R. Azaro, G. Boato, M. Donelli, G. Franceschini, A. Martini, and A. Massa, “Design of a miniaturized ism-band fractal antenna,” 2005.

96. Cohen, Nathan (Summer 1995), “Fractal antennas Part 1”, *Communications Quarterly* 5: 7–22. ISSN 1053-9433, 1995.

97. N. Cohen. “U.S. Patent № 6140975A. H01Q 1/48”, *Fractal. Antenna Ground Counterpoise, Ground Planes and Loading. Elements*, 2000.

98. Cohen; Nathan (Belmont, MA), Earle; Daniel (Waltham, MA), Mitchell; Justin (Waltham, MA) “U.S. patent number 10,594,038 [Application Number 15/528,397] was granted by the patent office on 2020-03-17 for fractal metamaterial cage antennas”, *This patent grant is currently assigned to Fractal Antenna Systems, Inc.. The grantee listed for this patent is Fractal Antenna Systems, Inc.. Invention is credited to Nathan Cohen, Daniel Earle, Justin Mitchell*, 2020

99. S. Hassan Ghadeer *et al.*, “An innovative fractal monopole MIMO antenna for modern 5G applications,” *Int. J. Electron. Commun.*, vol. 159, no. 154480, p. 154480, 2023.

100. T. Janani, Shennes Mathew, S. Deepa. Survey of fractal image compression techniques. *International Journal of Pure and Applied Mathematics. Volume 118 No. 20* 2018, 637-643.

101. О. Залевська, П. Яблонський, Ю. Сидоренко, О. Фіногенов, А. Ситник, “Реалізація алгоритму фрактального стиснення графічного зображення”, *Збірник наукових праць. Сучасні проблеми моделювання*, no. 22, 2021

102. Y. Zheng, X. Li, and M. Sarem, “Fast fractal image compression algorithm using specific update search,” *IET Image Process.*, vol. 14, no. 9, pp. 1733–1739, 2020.

103. O. Svychnuk, O. Barabash, J. Nikodem, R. Kochan, and O. Laptiev, “Image compression using fractal functions,” *Fractal Fract.*, vol. 5, no. 2, p. 31, 2021.

104. J. He, H.-X. Gong, and H.-Y. Lu, “Design of fractal image coding compression and transmission model based on wavelet transform,” in *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Cham: Springer International Publishing, 2022, pp. 15–25.

105. Іяс Каялі Ріад Вафаа, “Методи маршрутизації самоподібних інформаційних потоків у мультисервісних комп'ютерних мережах”, *Дисертація*, Харків, 2014.

106. В. Т. Грінченко, О. Б. Курилко, В. Т. Маципура, “Фрактальні властивості складних сигналів і випадкові хвилеві поля”, *Навчальний посібник*, 2022

107. V. V. Kashin *et al.*, “Information technologies based on noise-like signals: II. Statistical and fractal properties of chaotic algorithms,” *Radioelectron. Nanosyst. Inf. Technol.*, vol. 14, no. 2, pp. 151–164, 2022.

108. P. Herman, S. A. Wahab, A. Eke, and F. Hyder, “Fractal properties of neurophysiologic signals in rat somatosensory cortex,” *J. Cereb. Blood Flow Metab.*, vol. 25, no. 1_suppl, pp. S186–S186, 2005.

109. R. Ursulean, A. M. Lazar, and M. Istrate, “The sensitivity of the fractal spectrum: A means of controlling the statistical properties of chaotic signals,” *Elektron. Ir Elektrotech.*, vol. 114, no. 8, 2011.

110. О. М. Юнак, Б. М. Пелешак, Н. Л. Охремчук, Я. Р. “Метлевич Перетворення зображення фрактальної структури типу «Фрактальна пиль» (Множина кантора) в рандомізовану систему ітераційних функцій”, *XII Міжнародна науково практична конференція «Последние достижения на Европейската наука - 2016»*, Том 13, София «Бял ГРАД-БГ» ООД, 90с, 2016.

111. О. Yunak, О. Shpur, В. Strykhaliuk, М. Klymash, “Algorithm forming randomized system of iterative functions by based cantor structure”, *Infocommunication Technologies and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія*, vol. 1, no. 2, pp. 71–80, 2021.

112. О. М. Юнак, М. М. Климаш, О. М. Шпур, В. Б. Мрак, “Математична модель розпізнавання фрактальних структур з використанням технології нейронних мереж”, *Infocommunication Technologies and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія*, vol. 3, no. 1, pp. 1-9, 2023.

113. О. М. Юнак, Б. М. Стрихалюк, М. М. Климаш, “Ефективність рандомізованою системою ітераційних функцій над детермінованою системою ітераційних функцій при побудові фрактальних зображень з обмеженою роздільною здатністю”, *Вимірювальна та обчислювальна техніка в технологічних процесах*, no. 1, pp. 5–12, 2023.

114. О. М. Юнак, Б. М. Стрихалюк, М. М. Климаш, О. М. Шпур, “Побудова фрактального зображення типу “канторів пил”, з використанням рандомізованої системи ітераційних функцій”, *Infocommunication Technologies*

and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія, vol. 2, no. 1, pp. 19–25, 2022.

115. O. Yunak , B. Strykhaliuk , M. Klymash, Y. Pyrih, O. Shpur A “Neuron Network Learning Algorithm for the Recognition of Fractal Image and the Restoration of Their Quality”, *Lecture Notes in Electrical Engineering*, vol. 965 LNEE, pp. 574–584, (SciVerse SCOPUS), 2023.

116. O. Yunak, “Protection of documents with the help of fractal images formed by a randomized system of iterating functions”, *Infocommunication Technologies and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія*, vol. 2, no. 2, pp. 50–57, 2022.

117. O. M. Юнак, Б. М. Стрихалюк, О. П. Юнак, “Шифрування графічної інформації за допомогою матриць перетворень для захисту від дешифрування нейронними алгоритмами”, *Штучний інтелект*, no 2 (88), pp. 15–20, 2020.

118. J. Duckett, *HTML and CSS: Design and Build Websites*, 1st ed. Nashville, TN: John Wiley & Sons, 2014.

119. D. Woods, *HTML5 and CSS: Complete, International Edition*, 7th ed. Florence, AL: South-Western College Publishing, 2012.

120. R. Tomar and S. Dangi, *JavaScript: Syntax and practices*. Philadelphia, PA: Chapman & Hall/CRC, 2021.

121. B. Craig, *JavaScript: JavaScript programming made easy for beginners & intermediates (step by step with hands on projects)*. USA: Fanton, 2019.

122. S. B. Uzayr, Ed., *Mastering jQuery: A beginner’s guide*. London, England: CRC Press, 2023.

123. A. Kumar and M. Chaudhary, *Practical jQuery*, 1st ed. New York, NY: APress, 2015.

124. P. Matsinopoulos, *Practical bootstrap: Learn to develop responsively with one of the most popular CSS frameworks*, 1st ed. Berlin, Germany: APress, 2020.

125. N. Siru, *Bootstrap - A front-end framework that helps you build mobile responsive websites more quickly and easily*. Independently Published, 2021.

126. “Npm: Brain.Js,” *npm*. [Online]. Available: <https://www.npmjs.com/package/brain.js?activeTab=readme>. [Accessed: 10-Oct-2022]
127. “Brain.js. Побудова нейронної мережі. Уроки для початківців. W3Schools українською,” *Github.io*. [Online]. Available: https://w3schoolsua.github.io/ai/ai_brainjs.html. [Accessed: 11-Oct-2022].
128. B. Öggl and M. Kofler, *Git: Project management for developers and DevOps teams*. Quincy, MA: Rheinwerk Publishing, 2022.
129. J. Dahmer, *Github and git: A beginner's guide to git and github*. Independently Published, 2023.
130. Atlassian, “Bitbucket,” *Bitbucket*. [Online]. Available: <https://bitbucket.org/product/>. [Accessed: 11-Oct-2022].

Додаток А. Акти впровадження

Акціонерне товариство «Укртелеком»
Львівська філія
вул. Дорошенка, 43
м. Львів, 79007, Україна
Тел.: (032)261-21-21
Факс: (032)297-02-17



№ 46 G 500-9
від «12» липня 2023 р.
на № _____

Г

- 1

АКТ

про використання результатів дисертаційної роботи
Юнака Остапа Мироновича
«Методи побудови та обробки фрактальних зображень з використанням
рандомізованої системи ітераційних функцій»

Даний акт складений про те, що у Львівська філія ПАТ «Укртелеком» для підвищення ефективності засобів технічного захисту інформації корпоративної інформаційно-комунікаційної мережі використані результати дисертаційної роботи Юнака Остапа Мироновича «Методи побудови та обробки фрактальних зображень з використанням рандомізованої системи ітераційних функцій», представленої на здобуття наукового ступеня доктора філософії.

Зокрема, представниками компанії Львівська філія ПАТ «Укртелеком» за згодою автора Юнака О.М. використано метод шифрування фрактальних зображень, який використовує надлишковість та рандомізовані матриці, що дало змогу забезпечити прозорість передачі даних, зберігаючи при цьому конфіденційність і цілісність інформації.


Алгоритм виявився ефективним не тільки для фрактальних зображень, а й для будь-яких існуючих зображень, і дає змогу шифрувати в реальному часі з малими обчислювальними потужностями.

Директор



Т.В. Андрухів

«ЗАТВЕРДЖУЮ»
Директор
ТЗОВ «Астра-Львів»
Юрій ЖОВНІР
«13» _____ 2023 р.



АКТ

про використання результатів дисертаційної роботи
Юнака Остапа Мироновича
«Методи побудови та обробки фрактальних зображень з використанням
рандомізованої системи ітераційних функцій»

Даний акт складений про те, що у ТЗВО «Астра-Львів» для підвищення ефективності засобів технічного захисту інформації корпоративної інформаційно-комунікаційної мережі використані результати дисертаційної роботи Юнака Остапа Мироновича «Методи побудови та обробки фрактальних зображень з використанням рандомізованої системи ітераційних функцій», представленої на здобуття наукового ступеня доктора філософії.

Зокрема, представниками компанії ТЗВО «Астра-Львів» за згодою автора Юнака О.М. використано метод шифрування фрактальних зображень, який має просту реалізацію і дозволяє створювати шифрувальні ключі, що є необхідними для розшифрування зображення. Алгоритм маскує розмір зображення, що забезпечує додатковий рівень захисту. Використання надлишкових елементів перешкоджає нейронним мережам у співставленні пікселів. Зміна кольорів, яскравості та інверсія з використанням випадкових процесів ускладнюють завдання по підбору функції розшифрування. Враховуючи те, що матриці формуються випадковим чином, нейронні мережі не можуть встановити шифрувальну функцію. Для розшифрування зображення з 1000 пікселів потрібно перебрати надзвичайно велику кількість комбінацій (більше $5.1 \cdot 10^{2576}$), що на сьогоднішній день не є реалістичним технічним рішенням. Великою перевагою даного алгоритму є можливість передавати зашифровані зображення по відкритим каналам

Цей алгоритм проявив свою ефективність не лише у випадку фрактальних зображень, але й для будь-яких наявних зображень, дозволяючи шифрувати їх в реальному часі з використанням обмежених обчислювальних ресурсів.

Директор



Юрій ЖОВНІР

«ЗАТВЕРДЖУЮ»
Директор
ТзОВ «АПДЕЙТ
ТЕХНОЛОДЖІС СИСТЕМЗ»

Василь ВІТІВ
« 15 » травня 2023 р.

АКТ

про використання результатів дисертаційної роботи
Юнака Остапа Мироновича
«Методи побудови та обробки фрактальних зображень з використанням
рандомізованої системи ітераційних функцій»

Даний акт складений про те, що у ТзОВ «АПДЕЙТ ТЕХНОЛОДЖІС СИСТЕМЗ» для підвищення засобів технічного захисту інформації корпоративної мережі використані результати дисертаційної роботи Юнака Остапа Мироновича «Методи побудови та обробки фрактальних зображень з використанням рандомізованої системи ітераційних функцій», представленої на здобуття наукового ступеня доктора філософії.

Зокрема, представниками ТзОВ «АПДЕЙТ ТЕХНОЛОДЖІС СИСТЕМЗ» за згодою автора Юнака О.М. використано метод шифрування фрактальних зображень. Метод шифрування фрактальних зображень, використовуючи матриці перетворень, що запобігають дешифруванню з використанням нейронних алгоритмів.

Ефективність цього методу полягає у використанні рандомізованого алгоритму побудови матриць перетворень, що гарантує неможливість розшифрування інформації. Цей метод є ефективним для будь-яких існуючих зображень і дозволяє проводити шифрування в реальному часі з використанням обчислювальних ресурсів низької потужності.

Директор



Василь ВІТІВ

«Затверджую»

Проректор з наукової роботи
Національного університету
«Львівська політехніка»
д.т.н., проф. Івану ДЕМИДОВУ
«_____» _____ 2023 р.



АКТ

про використання результатів дисертаційної роботи Юнака Остапа Мироновича на тему «Методи побудови та обробки фрактальних зображень з використанням рандомізованої системи ітераційних функцій».

Комісія у складі начальника науково-дослідної частини, д.т.н., Небесного Р.В., В.О. заступника начальника планово-фінансового відділу Фаст І.І., завідувача кафедри телекомунікацій, д.т.н., проф. Климаша М.М., склала цей акт у тому, що у держбюджетній науково-дослідній роботі: «Стратегічні напрямки, методи і засоби цифровізації та інтелектуалізації енергетичних систем з використанням сучасних інформаційно-комунікаційних технологій» (ДБ/Smart Grid), (№ держреєстрації 0123U101692, (2023-2025 рр.) – учасник) використані наступні результати дисертаційної роботи Юнака Остапа Мироновича на тему «Методи побудови та обробки фрактальних зображень з використанням рандомізованої системи ітераційних функцій»:

- програмний модуль навчання нейронної мережі для автоматичного визначення параметрів рандомізованої системи ітераційних функцій, що забезпечує швидку і ефективну обробку даних для визначення оптимальних параметрів фрактальних структур;
- спосіб захисту документів з використанням побудови фрактальних структур за допомогою рандомізованої системи ітераційних функцій;

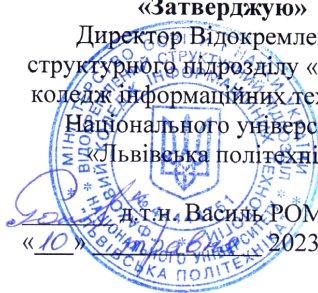
Загалом, використання фрактальних зображень та рандомізованих систем ітераційних функцій може покращити ефективність та інтелектуалізацію Smart Grid, допомагаючи забезпечити стабільну та ефективну роботу електромережі та знижуючи витрати на її управління та обслуговування.

Члени комісії:

Three handwritten signatures in blue ink are shown. The first signature is the most prominent and appears to be 'R. Nebesnyy'. The other two are less distinct but appear to be 'I. Fast' and 'M. Klimash'.

Роман НЕБЕСНИЙ
Ірина ФАСТ
Михайло КЛИМАШ

«Затверджую»
Директор Відокремленого
структурного підрозділу «Фаховий
коледж інформаційних технологій
Національного університету
Львівська політехніка»
д.т.н. Василь РОМАНЧУК
«10» травня 2023 р.



АКТ

про використання результатів дисертаційної роботи Юнака Остапа Мироновича на тему «Методи побудови та обробки фрактальних зображень з використанням рандомізованої системи ітераційних функцій», у навчальному процесі відділення «Апаратних засобів інформатизації».

Даний акт складений комісією у складі:

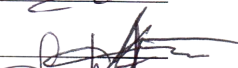
- Володимир СТАХІВ, завідувач відділення «Комп'ютерних систем та мереж», в.о. завідувача відділення «Апаратних засобів інформатизації»;
- Богдан ПЕЛЕЩАК, голова циклової комісії «Апаратних засобів інформатизації»;
- Марія-Вікторія ПОЛЕЦЬ, завідувача лабораторії «Апаратних засобів інформатизації».

про те, що в навчальному процесі відділення «Апаратних засобів інформатизації» використано результати дисертаційної роботи Юнака Остапа Мироновича «Методи побудови та обробки фрактальних зображень з використанням рандомізованої системи ітераційних функцій», метод побудови фрактальних структур, що дає змогу без значних обчислювальних ресурсів в реальному часі представляти будь-яке зображення у вигляді фрактальних структур. Зокрема, результати використані для модернізації курсу лекцій (Лекція 7 «Поняття про комп'ютерну графіку, основні галузі застосувань») з дисципліни «Інженерна та комп'ютерна графіка» спеціальності 172 «Телекомунікації та радіотехніка». А також використано метод шифрування фрактальних зображень, який використовує надлишковість та рандомізовані матриці, що дало змогу забезпечити прозорість передачі даних, зберігаючи при цьому конфіденційність і цілісність інформації для (Лекція 4 «Забезпечення надійності і підвищення якості програм» з дисципліни «Надійність, діагностика та експлуатація комп'ютерних систем та мереж» спеціальності 123 «Комп'ютерна інженерія».

Члени комісії:



Володимир СТАХІВ



Богдан ПЕЛЕЩАК



Марія-Вікторія ПОЛЕЦЬ

**Додаток Б. Список публікацій здобувача за темою дисертації та відомості
про апробацію результатів дисертації**

Наукові праці, у яких опубліковані основні результати дисертації

1. Yunak O., Strykhaliuk B., Klymash M., Pyrih Y., Shpur O. A Neuron Network Learning Algorithm for the Recognition of Fractal Image and the Restoration of Their Quality // Lecture Notes in Electrical Engineering. – 2023. – Vol. 965 LNEE. – P. 574–584. (SciVerse SCOPUS).

2. Юнак О. М., Климаш М. М., Шпур О. М., Мрак В. Б. Математична модель розпізнавання фрактальних структур з використанням технології нейронних мереж // Infocommunication Technologies and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія. – 2023. – Vol. 3, № 1. – P. 1–9.

3. Юнак О. М., Стрихалюк Б. М., Климаш М. М. Ефективність рандомізованою системою ітераційних функцій над детермінованою системою ітераційних функцій при побудові фрактальних зображень з обмеженою роздільною здатністю // Вимірювальна та обчислювальна техніка в технологічних процесах. – 2023. – № 1. – С. 5–12.

4. Yunak O. Protection of documents with the help of fractal images formed by a randomized system of iterating functions // Infocommunication Technologies and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія. – 2022. – Vol. 2, № 2. – P. 50–57.

5. Юнак О. М., Стрихалюк Б. М., Климаш М. М., Шпур О. М. Побудова фрактального зображення типу “канторів пил”, з використанням рандомізованої системи ітераційних функцій // Infocommunication Technologies and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія. – 2022. – Vol. 2, № 1. – P. 19–25.

6. Юнак О. М., Стрихалюк Б. М., Юнак О. П. Шифрування графічної інформації за допомогою матриць перетворень для захисту від дешифрування нейронними алгоритмами // Штучний інтелект. – 2020. – № 2 (88). – С. 15–20.

7. Yunak O., Shpur O., Strykhaliuk B., Klymash M. Algorithm forming randomized system of iterative functions by based cantor structure // Infocommunication Technologies and Electronic Engineering = Інфокомунікаційні технології та електронна інженерія. – 2021. – Vol. 1, № 2. – P. 71–80.

Наукові праці, які засвідчують апробацію матеріалів дисертації

8. Юнак О., Пелешак Б., Охремчук О., Метлевич Я. Перетворення зображення франктальної структури типу «Фрактальний пил» (Множина Кантора) в рандомізовану систему ітераційних функцій // Последните постижения на европейската наука: за XII международна научна практична конференция, 7-25 юни 2016. София «БялГРАД-БГ», 2016. Том 10. P. 27-29.

9. Щерба В., Юнак О., Юнак О. Віртуальний паспорт для входу в нову мережу TELEGRAM OPEN NETWORK // Вітчизняна наука на зламі епох: проблеми та перспективи розвитку: матеріали Всеукраїнської наук.-практ. інтернет- конф., 18 березня 2020р.: зб. наук. праць. Переяслав, 2020. Вип. 59. С. 100-102.

10. Юнак О., Цебенко О., Юнак О. TON BLOCKCHAIN – компонент мережі TELEGRAM OPEN NETWORK // Вітчизняна наука на зламі епох: проблеми та перспективи розвитку: матеріали Всеукраїнської наук.-практ. інтернет- конф., 18 березня 2020р.: зб. наук. праць. Переяслав, 2020. Вип. 59. С. 102-103.

11. Пелешак Б., Охремчук Н., Юнак О., Кащук В. Використання веб-сервісів для хостингу проектів (систем контролю версій) та їх спільної розробки як засобів інтерактивного навчання методом проектів // матеріали XXV Всеукраїнської наук.- практ. інтернет-конф., 16-17 верес. 2016 р.:зб. наук. праць. Переяслав-Хмельницький, 2016.Вип. 25. С. 147-150

12. Пелешак Б., Юнак О., Охремчук О., Метлевич Я. Використання систем моделювання мереж зв'язку // Вітчизняна наука на зламі епох: проблеми та перспективи розвитку: матеріали XXIV Всеукраїнської наук.- практ. інтернет-конф., 25-26 червня 2016р.: зб.наук. праць. Переяслав-Хмельницький, 2016. Вип. 24. С.87-90.