РЕФЕРАТ

Пояснювальна записка: ___ с., ___ рис., ___ табл., __ дод., ___ джерел.

Об'єкт розробки: структури багатовимірних даних, отримані за допомогою фрактальних алгоритмів.

Мета кваліфікаційної роботи: дослідження способів моделювання фракталів та розроблення програмного засобу для побудови фрактальних зображень.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні додатка, за допомогою якого у зручній формі можна ознайомитися з моделями фракталів, а також дослідити закономірності в побудові фрактальних зображень при зміні параметрів фрактальної моделі.

Актуальність розробленої інформаційної системи визначається тим, що на даний час теорія фракталів знаходить широке застосування в різних областях практичної діяльності. Використання фрактальних моделей дозволяє знаходити нові підходи у вирішенні завдань обробки цифрової інформації, вивченні питань турбулентного руху рідин, радіолокації, дослідження фінансових ринків, отримання нових наноматеріалів з заданими властивостями тощо.

Список ключових слів: ФРАКТАЛ, РОЗРОБКА, МАТЕМАТИЧНА МОДЕЛЬ, САЙТ, СТОРІНКА, АЛГОРИТМ, ПРОЕКТУВАННЯ, МЕНЮ, ЗОБРАЖЕННЯ, ПАРАМЕТРИ.

ABSTRACT

Explanatory note: ___ pp., ___ fig., ___ table, __ appendix, ___ sources.

Object of development: structures of multidimensional data obtained using fractal algorithms.

The purpose of the qualification work: research of methods of modeling fractals and development of software for construction of fractal images.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the program, determines the input and output data, provides characteristics of the parameters of hardware, describes the call and download program .

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance lies in the creation of an application with which you can easily get acquainted with the models of fractals, as well as to explore the patterns in the construction of fractal images when changing the parameters of the fractal model.

The relevance of the developed information system is determined by the fact that currently the theory of fractals is widely used in various fields of practice. The use of fractal models allows to find new approaches in solving problems of digital information processing, studying the turbulent motion of liquids, radar, research of financial markets, obtaining new nanomaterials with given properties, etc.

List of keywords: FRACTAL, DEVELOPMENT, MATHEMATICAL MODEL, SITE, PAGE, ALGORITHM, DESIGN, MENU, IMAGE, PARAMETERS.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ОС - операційна система;

ПЗ - програмне забезпечення;

ПК - персональний комп'ютер;

CSS - Cascading Style Sheets, каскадні таблиці стилів;

HTML - Hyper Text Markup Language, мова розмітки гіпертекстових документів;

IFS - Iterated Functions Systems, системи ітеративних функцій.

ЗМІСТ

ВСТУП

Фрактал є однією з багатьох складових частин певної субстанції, тому зникнення однієї з таких складових призводить до втрати візуальної гармонії, що людське око розпізнає одразу. Присутність фракталу з першого погляду можна і не помітити, якщо не заглиблюватись у досконале вивчення математики. Проте ця наука, дійсно, не має меж і постійно спонукає до різноманітних досліджень.

В даний час теорія фракталів знаходить широке застосування в різних областях практичної діяльності. Використання фрактальних моделей дозволяє знаходити нові підходи у вирішенні завдань обробки цифрової інформації, вивченні питань турбулентного руху рідин, радіолокації, дослідження фінансових ринків, отримання нових наноматеріалів з заданими властивостями тощо.

Застосування фракталів у комп'ютерних технологіях є досить новим напрямком. Їх використання дозволяє вирішувати багато актуальних на сьогодні задач з більшою ефективністю, ніж існуючі алгоритми. Одне з головних застосувань фракталів – це фрактальна графіка. За допомогою якої можна створити (описати) поверхні дуже складної форми, а змінюючи всього декілька коефіцієнтів в рівнянні домогтися практично нескінченних варіантів початкового зображення.

Метою кваліфікаційної роботи є дослідження способів моделювання фракталів та розроблення програмного засобу для побудови фрактальних зображень.

В наш час область застосування фракталів досить велика – від дослідження утворення грузлих пальців у пористих середовищах до побудови фрактальних пейзажів, які мало в чому поступаються роботам кращих художників.

Основні задачі, що вирішуються під час виконання даної роботи, це:

1. Аналіз математичних моделей фрактальних множин.

2. Розробка алгоритмів моделювання фрактальних структур, оцінка характеристик алгоритмів, а також властивостей множин, породжуваних запропонованими алгоритмами.

3. Розробка програмного комплексу для проведення обчислювального експерименту і дослідження структур даних на основі фрактальних моделей, отриманих з використанням запропонованих алгоритмів.

4. Тестування розробленого програмного забезпечення при різних значеннях параметрів моделей.

За допомогою розробленого в роботі веб-ресурсу можна в зручній формі ознайомитися з моделями фракталів, дослідити закономірності у побудові фрактальних зображень при зміні параметрів фрактальної моделі.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

1.1.1. Основні поняття фракталів

Перші ідеї, що стосуються аналізу фрактальних структур, виникли в XIX столітті. Кантор за допомогою простої рекурсивної процедури перетворив лінію в набір незв'язаних точок (так званий Пил Кантора). Він брав лінію і вилучав центральну третину і після цього повторював ту ж саму процедуру з відрізками, що залишилися. Пеано намалював особливий вид лінії. Для її створення він використовував наступний алгоритм. На першому кроці він брав пряму лінію і замінював її на дев’ять відрізків довжиною в три рази меншою, ніж довжина початкової лінії. Далі він робив те ж саме з кожним відрізком отриманої лінії. І так до нескінченності. Унікальність отриманої лінії в тому, що вона заповнює всю площину.

Доведено, що для кожної точки на площині можна знайти точку, що належить лінії Пеано. Крива Пеано і пил Кантора виходили за рамки звичайних геометричних об'єктів. Вони не мали чіткої розмірності. Пил Кантора будувався начебто на підставі одновимірної прямої, але складався з точок (розмірність нуль). Крива Пеано будувалася на основі одновимірної лінії, а в результаті виходила площина. Аналогічно цьому у багатьох інших областях науки з'являлися задачі, розв’язання яких приводило до дивних результатів. Аж до XX століття йшло нагромадження даних про такі дивні об'єкти, без будь-якої спроби їх систематизувати. Істотне зрушення відбулося наприкінці 70-х – середини 80-х років, коли вийшла в світ книга Бенуа Мандельброта «Фрактальна геометрія природи». Ця яскрава робота викликала загальний інтерес до фрактальної геометрії - поняттю, уведеному самим Мандельбротом.

Щоб уявити собі, що таке фрактал, розглянемо приклад, що став класичним - "Яка довжина південного узбережжя Криму?". Відповідь на це

питання не така проста, як здається на перший погляд. Усе залежить від довжини інструмента, яким ми будемо користуватися. Виміривши берег за допомогою кілометрової лінійки, ми одержимо якусь довжину. Однак ми пропустимо багато невеликих заток та напівостровів, чий розмір набагато менший за нашу лінійку. Зменшивши розмір лінійки до, скажімо, одного метра, ми врахуємо ці деталі ландшафту, і, відповідно, довжина берега стане більшою. Підемо далі і будемо вимірювати довжину берега за допомогою міліметрової лінійки. Зараз ми враховуємо деталі, що більше міліметра, тобто довжина буде ще більшою. В підсумку, відповідь на таке, здавалося б, просте питання може здивувати кого завгодно - довжина берега Кримського півострова нескінченна.

1.1.2. Основні види побудов фрактальних зображень

Незважаючи на те, що більшість людей асоціює поняття фракталу та фрактальної графіки з чимось хаотичним, з тим, що не можна структурувати, фрактали можна поділити на три основні групи (відповідно до того, що лежить в їх основі, будь-то геометричні операції, формули чи певні дії з кольором): геометричні, алгебраїчні та стохастичні.

1. Геометричні фрактали. Фрактали цього класу найбільш наочні. Саме з них починалася історія фракталів. Зображення цього типу отримують за допомогою геометричних маніпуляцій над початковою фігурою, яку називають ініціюючим елементом, наприклад, у двовимірному просторі такою фігурою є деяка ламана, у тривимірному ж – площина. Шляхом застосування деякого набору правил до початкового елементу, він перетворюється на більш складну геометричну фігуру, фрактал з якої можна отримати, якщо провести нескінченну кількість ітерацій над кожною частиною новоутвореної фігури.

Фрактали цього типу будуються поетапно. Спочатку зображується основа, потім деякі частини основи заміняються на фрагмент. На кожному наступному етапі частини вже побудованої фігури, аналогічні замінених частинам основи, знову заміняються на фрагмент, взятий у потрібному масштабі. Кожного разу

масштаб зменшується. Коли зміни стають візуально непомітними, вважається, що побудована фігура наближена до фракталу і дає уявлення про його форму. Проте для отримання самого фракталу, потрібна нескінченна кількість етапів. Змінюючи основу, можна отримати багато різних геометричних фракталів.

Геометричні фрактали з одного боку являються об’єктом серйозних наукових досліджень, а з іншого – їх можна легко розпізнати і "побачити", навіть не будучи спеціалістом у даній галузі. Таке поєднання рідко зустрічається в сучасній математиці, де всі об’єкти задаються за допомогою слів і символів, зрозумілих лише людям із відповідною освітою.

Виявляється, що більшість геометричних фракталів можна намалювати на листочку у клітинку. Слід звісно ж пам’ятати, що отримані таким шляхом зображення будуть лише кінцевим наближенням нескінченних по своїй суті фракталів. Наприклад, маючи достатньо великий аркуш паперу, олівець та запас вільного часу можна намалювати такий трикутник Серпінського, що з відстані в кілька метрів неозброєне око буде сприймати це зображення як справжній фрактал. Очевидно, що комп’ютер виконає це набагато краще, збільшивши точність намальованого.

Прикладами геометричних фракталів слугують: сніжинка Коха, лист, трикутник Серпінського, криві Гільберта, криві Серпінського, килим Серпінського тощо.

Одним із перших досліджень вчених у галузі фрактальної графіки була сніжинка Коха. Її отримують із трьох копій кривої Коха, інформація про яку вперше з’явилась у статті шведського математика Хельге фон Коха в 1904 р. Ця крива була придумана як приклад неперервної лінії, до якої неможливо провести дотичну в жодній точці. Лінії з такою властивістю були відомі й раніше (наприклад, Карл Вейерштрас побудував таку лінію ще у 1872 р.), але особливістю кривої Коха була простота її конструкції.

Процес її побудови виглядає наступним чином: береться одиничний відрізок, ділиться на три рівні частини і замінюється середнім інтервалом рівностороннього трикутника без цього сегмента (див. мал. 1.6). У результаті

утворюється ламана, що складається з чотирьох ланок довжиною 1/3. На наступному кроці потрібно повторити операцію для кожної з чотирьох одержаних частин. Таким чином, для отримання кожного з наступних поколінь, всі ланки попереднього покоління необхідно замінити за даним правилом. Крива n-го покоління при будь-якому кінцевому n називається предфракталом. На рис. 1.1. представлені 7 поколінь кривої. Коли n буде прямувати до нескінченності, крива Коха стає фрактальним зображенням.

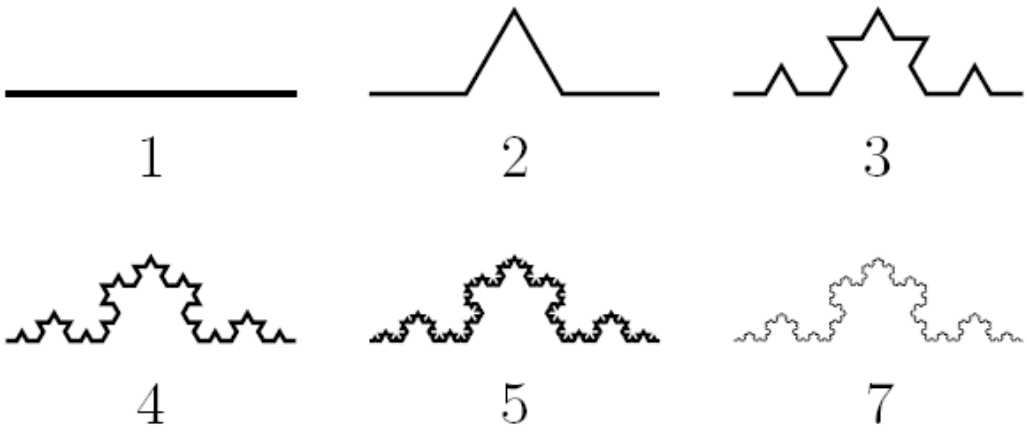

Рис. 1.1. Етапи побудови кривої Коха

Для того, щоб отримати сніжинку Коха, за початковий елемент, замість одиничного відрізка, береться рівносторонній трикутник, кожна сторона якого і дорівнюватиме даному відрізку. Провівши відповідні перетворення з кожною стороною, отримаємо видозмінену замкнуту фігуру, яку прийнято називати сніжинкою Коха.

Для одержання іншого фрактального об'єкта потрібно змінити правила побудови. Для цього потрібно взяти за нульове покоління фракталу трикутник, знайти середину кожної сторони цього трикутника та з'єднати їх. Ця дія повторюється для кожного отриманого трикутника безліч кількість разів.

Отриманий таким чином фрактал був запропонований ще в 1915 р. польським математиком і названий трикутник Серпінського. Проте фрактал цього типу можна отримати ще й іншими способами (рис. 1.2).

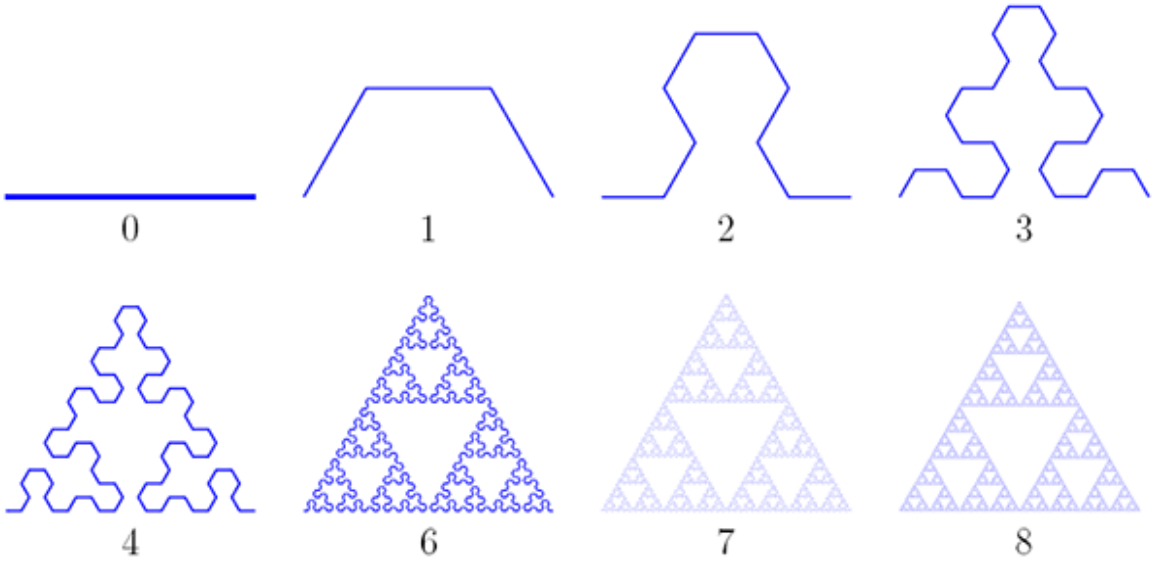

Рис. 1.2. Один з способів побудови трикутника Серпінського

Як і інші фрактали, він має свої властивості:

– трикутник Серпінського має нульову площу. Після кожної ітерації площа того, що залишилось множиться на ¾, тобто стає все меншою і поступово наближається до нуля;

– цікавим є несподіваний зв'язок фракталу з комбінаторикою. Якщо в трикутнику Паскаля, всі парні числа виділити білим, а непарні – чорним, то видимі числа в деякому наближені сформують трикутник Серпінського.

На сьогодні існує безліч різновидів даного фракталу: квадрат Серпінського, піраміда Серпінського, Губка Менгера, що різняться зовнішньо, але будуються за одним і тим же правилом, змінюється лише початковий елемент або розмірність простору, як наприклад в піраміді Серпінського.

У машинній графіці застосування геометричних фракталів необхідне під час отримання зображень дерев, кущів, берегової лінії. Двовимірні геометричні фрактали застосовуються для створення об'ємних текстур.

2. Алгебраїчні фрактали. Алгебраїчні фрактали – це найбільша група фракталів, яка отримала назву через те, що будують їх на основі алгебраїчних формул, іноді досить простих. Вони виникають при дослідженні нелінійних

динамічних систем (звідки й інша назва – динамічні фрактали). Поведінку такої системи можна описати комплексною нелінійною функцією (многочленом)

Як приклад візьмемо будь-яку початкову точку z_0 на комплексній площині. Розглянемо нескінчену послідовність чисел на комплексній площині, кожне наступне з яких отримується з $f(z)$. попереднього

$$z_0, z_1 = f(z_0), \quad z_2 = f(z_1), \dots z_{n+1} = f(z_n).$$

В залежності від початкової точки z_0, така послідовність може вести себе по різному: прямувати до нескінченності при $n \to \infty$;; прямувати до 0; приймати декілька фіксованих значень і не виходити за їх межі; можливі і більш складні варіанти.

Щоб проілюструвати алгебраїчні фрактали звернемося до множини Manlelbrota (рис. 1.3).

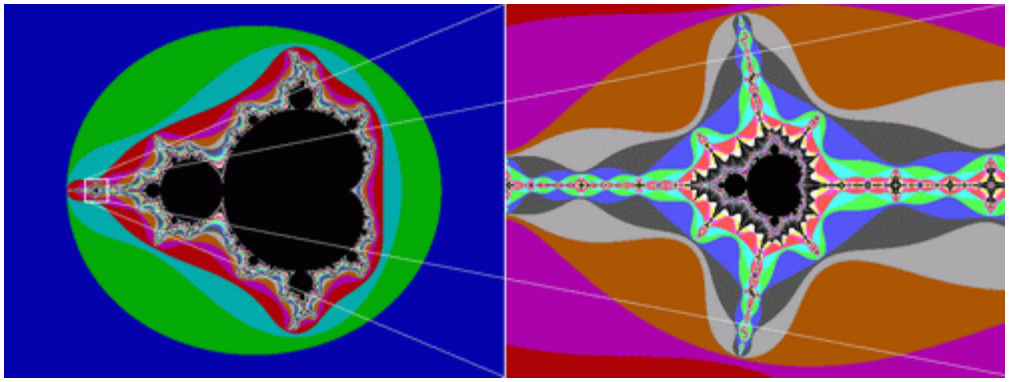

Рис. 1.3. Множина Мандельброта

Для побудови цього фракталу необхідні комплексні числа. Комплексне число – це число, що складається з двох частин: дійсної та уявної, і позначається воно $a+b_i$ Дійсна частина a– це звичайне число в нашому уявленні, а b_i – уявна частина (i – називають уявною одиницею, і прийнято вважати, що при піднесенні до квадрату вона дає -1) /

Комплексне число можна зобразити як точку на площині, у якої координата X це дійсна частина a, а Y це коефіцієнт при уявній частині b. Функціональна множина Мандельброта визначається за правилом:

$$z_{i+1} = z_i * z_i + C, \quad (1.1)$$

де i – комплексні змінні.

В залежності від параметра C функція f(z) може прямувати до нескінченності або залишатись обмеженою. При цьому всі значення C при яких послідовність обмежена і утворюють таким чином множину Мандельброта. Дана множина була детально вивчена самим Мандельбротом та іншими математиками, які відкрили немало цікавих властивостей цієї множини.

Ще один тип фракталів, що включає в себе клас динамічних фракталів – так звані басейни. Одним із яскравих прикладів є басейни Ньютона. Формула для їх побудови засновані на методі розв'язання нелінійних рівнянь, який був відкритий великим математиком ще в XVII столітті. Застосовуючи загальну формулу метода Ньютона отримаємо послідовність $z_{n+1} = ((k-1)z_n^k - a)/kz_n^{k-1}$.

Якщо в останній формулі підставити k =2, а в якості початкового наближення взяти z_0=a, то отриманий фрактал будуватиметься за формулою $f(z)=z^3-1$ (рис. 1.4).

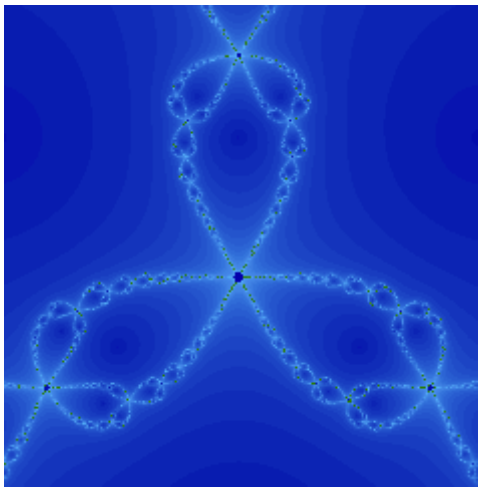

Рис. 1.4. Басейн Ньютона 3 порядку

Ще одним із різновидів алгебраїчних фракталів є пузирі Галлея. (рис. 1.5) Такі фрактали виходять, якщо в якості правила для побудови використовувати

формулу Галлея для пошуку наближених значень коренів функцій. На відміну від метода Ньютона, даний спосіб ефективніший: послідовність сходиться до нуля функції швидше.

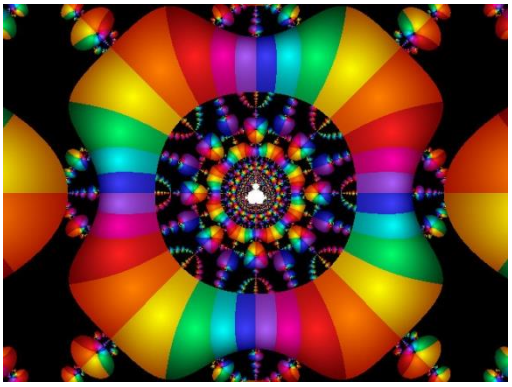

Рис. 1.5. Метод Галлея

Крім вищезгаданих до динамічних фракталів також належить множина Жюліа, яку можна зобразити як множину точок, що мають конкретний тип поведінки. Фрактальні зображення даного типу часто використовують в моделюванні, кінематографі та для створення різного роду спецефектів.

3. Стохастичні фрактали. Фрактали, що отримуються у випадку, якщо в ітераційному процесі випадковим чином замінити якісь з параметрів називаються стохастичними. Термін "стохастичність" походить від грецького і означає "припущення".

Стохастичним природним процесом є броунівський рух. За допомогою комп'ютера такі процеси задаються досить просто: потрібно задати послідовність випадкових чисел і налаштувати відповідний алгоритм.

При цьому отримуються об'єкти досить схожі на природні – несиметричні дерева, порізані берегові лінії тощо. Двохвимірні стохастичні фрактали використовуються для моделювання рельєфу, ландшафтів, поверхні морів.

Спроможність фрактальної графіки моделювання образів живої природи обчислювальним шляхом часто застосовують для автоматичної генерації незвичайних ілюстрацій.

Типовим представником даного класу фракталів є "Плазма" (рис. 1.6).

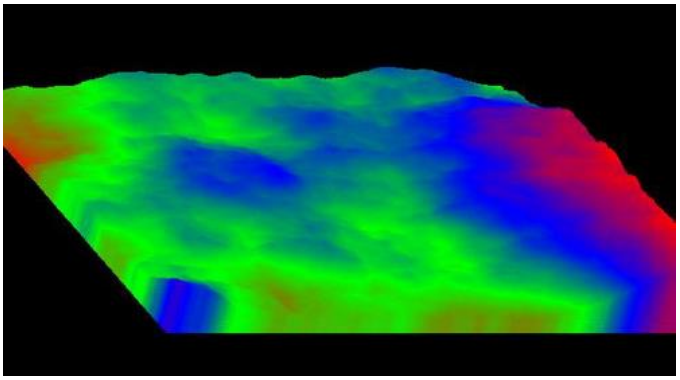

Рис 1.6. Плазма

Для її побудови візьмемо прямокутник і для кожного його кута визначимо колір. Далі знаходимо центральну точку прямокутника і розфарбовуємо її у колір рівний середньому арифметичному кольорів по кутах прямокутника плюс деяке випадкове число. Чим більше випадкове число – тим більш "рваним" буде малюнок. Якщо, наприклад, взятий колір крапки за висоту над рівнем моря, то отримаємо замість плазми – гірський масив.

Саме на цьому принципі моделюються гори в більшості програмах. За допомогою алгоритму, схожого на плазму будується карта висот, до неї застосовуються різні фільтри, накладаються текстури.

4. Системи ітерованих функцій (IFS). Група фракталів стала відомою завдяки роботам Майкла Барнслі. Він намагався кодувати зображення за допомогою фракталів. Запатентувавши ідеї з кодування за допомогою фракталів, він створив фірму «Iterated Systems». І через деякий час вона випустила програмний продукт «Images Incorporated», в якому зображення переводлись з растрової форми у фрактальну. Це дозволило домогтися високих ступенів стиснення. При низьких ступенях стиснення якість малюнків поступалося якістю формату JPEG, але при високих картинки виходили більш якісними (рис. 1.7).

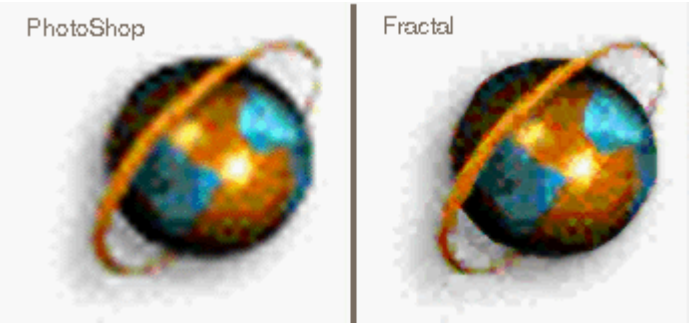

Рис. 1.7. Приклад зображення виконаного в PhotoShop та фрактального

Якщо в L-systems (алгебраїчних фракталах) йшлося про заміну прямої лінії якимсь полігоном, то в IFS ми в ході кожної ітерації замінюємо якийсь полігон (квадрат, трикутник, коло) на набір полігонів, кожен їх яких підданий афінних перетворень. При афінних перетвореннях вихідне зображення змінює масштаб, паралельно переноситься уздовж кожної з осей і обертається на деякий кут.

1.1.3. Практичне застосування фракталів

Сьогодні фрактальна геометрія пронизує багато наукових галузей, таких як: астрофізика, біологія, криптографія, навіть музика та література. Розглянемо деякі з прикладів використання фракталів та фрактальних зображень в науці.

1. Економіка. Фрактальний аналіз ринків – новий напрямок аналізу валютного та фондового ринку. Родоначальником фрактального аналізу ринків є Бенуа Мандельброт, який окреслив теорію у своїй книзі у співавторстві з Річардом Л. Хадсоном "(Не)слухняні ринки: фрактальна революція у фінансах." Фрактальний аналіз ринків, на відміну від теорії ефективних ринків, постулює залежність майбутніх цін від їх минулих змін. Таким чином, процес ціноутворення на ринках глобально детермінований, залежний від "початкових умов ", тобто минулих значень. Локально ж процес ціноутворення випадковий, тобто в кожному конкретному випадку ціна має два варіанти розвитку. Фрактальний аналіз ринків безпосередньо виходить з фрактальної теорії і запозичує властивості фракталів для отримання прогнозів.

2. Природничі науки. У фізиці фрактали природним чином виникають при моделюванні нелінійних процесів, таких як: турбулентний перебіг рідини, складні процеси дифузії-адсорбції, полум'я, хмари тощо. Фрактали використовуються при моделюванні пористих матеріалів, наприклад, в нафтохімії. У біології вони застосовуються для моделювання популяцій і для опису систем внутрішніх органів (система кровоносних судин).

3. Фрактали в астрофізиці. Ніхто не знає скільки точно зірок на небі, але, дивлячись на них, ви коли-небудь задумувались над тим як вони формувались? Астрофізики вважають, що ключем рішення цієї проблеми є фрактальна природа міжзоряного газу. Фрактальні розподіли є ієрархічними так як сліди диму, чи наприклад купчасті хмари в небі. Турбулентність формує як хмари в небі, так і своєрідні хмари у космосі, надаючи їм нелінійного, але повторюваного патерну, який неможливо було б описати без допомоги фрактальної геометрії та людини, що на належному рівні змогла б пояснити особливості цих перетворень і спрогнозувати можливі подальші перспективи, використовуючи засоби обробки фрактальних зображень [1].

4. Фрактали в біології. Біологи традиційно моделюють природу та природні процеси з використанням евклідової геометрії. Вони представляють серцебиття, як синусоїду, хвойні дерева, як конуси, клітинні мембрани у вигляді графіків або простої поверхні. Тим не менш, вчені прийшли до висновку, що багато природних об'єктів краще характеризується з використанням фрактальної геометрії. Біологічні системи і процеси, як правило, характеризуються багаторівневими підструктурами і водночас з повторювальною загальною картиною. Вчені виявили, що базова структура хромосома є деревоподібною, кожна хромосома складається з безлічі "міні-хромосом", і тому може розглядатися як фрактал. Самоподібність була виявлена також в наборі ДНК. На думку деяких учених-біологів фрактальні властивості ДНК можуть бути використані для вирішення еволюційних відносин серед тварин. Можливо, у майбутньому біологи, співпрацюючи з кваліфікованими спеціалістами з комп'ютерних наук, використають фрактальну

геометрію для створення всеосяжної моделі структур і процесів, що спостерігаються в природі.

5. Радіотехніка. Використання фрактальної геометрії при проектуванні антенних пристроїв було вперше застосоване американським інженером Натаном Коеном, який жив в центрі Бостона, де була заборонена установка зовнішніх антен на будівлях. Натан вирізав із алюмінієвої фольги фігуру у формі кривої Коха і наклеїв її на лист паперу, потім приєднав до приймача. Коен заснував власну компанію і налагодив серійний випуск такого типу радіоантен.

6 .Інформатика. Стиснення зображень. Фрактальне стиснення зображень – це алгоритм стиснення зображень із втратами, заснований на застосуванні систем ітеруючих функцій (IFS, як правило є афінними перетвореннями) до зображень. Даний алгоритм відомий тим, що в деяких випадках дозволяє отримати дуже високі коефіцієнти стиснення для реальних фотографій природних об'єктів, що недоступно для інших алгоритмів стиснення зображень у принципі. Ці алгоритми були засновані на ідеї про те, що замість самого зображення можна зберігати стискаюче відображення, для якого це зображення (або деяке близьке до нього) є нерухомою крапкою. Один з варіантів даного алгоритму був використаний фірмою Microsoft при виданні своєї енциклопедії.

Через складну ситуацію з патентуванням широкого розповсюдження алгоритм не отримав. Майклом Барнслі та іншими було отримано кілька патентів на фрактальний стиск у США. Ці патенти покривають широкий спектр можливих змін фрактального стиснення і серйозно стримують його розвиток. Дані патенти не обмежують досліджень у цій області, тобто можна придумувати свої алгоритми на основі запатентованих і публікувати їх. Також можна продавати алгоритми в країни, на які не поширюються отримані патенти. Крім того, термін дії більшості патентів – 17 років з моменту прийняття і він закінчується для більшості патентів найближчим часом, відповідно використання методів, що покривалися цими патентами, стане гарантовано вільним.

7. Децентралізовані мережі. Система призначення IP-адрес у мережі Netsukuku використовує принцип фрактального стиснення інформації для компактного збереження інформації про вузли мережі. Кожен вузол мережі Netsukuku зберігає всього 4 Кб інформації про стан сусідніх вузлів, при цьому будь-який новий вузол підключається до загальної мережі без необхідності в центральному регулюванні роздачі IP-адрес, що, наприклад, характерний для мережі Інтернет.

8. Криптографія. У криптографії використовують фрактальні хаотичні системи, які мають таку властивість: їх поведінка суттєво залежить від заданих початкових умов, при цьому стан системи в певний момент часу неможливо виразити простою формулою. Хаотична поведінка фракталів використовується в криптографії для шифрування даних і генерації випадкових послідовностей чисел. У фрактальній криптографії немає циклів, замість них використовуються ітерації. Так, для обчислення n-ої ітерації функції, необхідно спочатку обчислити (n-1)-шу ітерацію і т. д. до найпершої. Як і в звичайній криптографії, чим більше виконано ітерацій – тим краще, тому що проітерована багато разів точка фрактала далеко віддаляється від початку і неможливо обчислити траєкторію точки не виконавши кожен крок рекурсії. Отже, якість шифрування виходить значно вищою в порівняні зі звичайними методами. Ще одна з характеристик фракталів, що використовується в криптографії – це чутливість до початкових умов, тобто два близьких початкових значення будуть розходитись по мірі роботи системи. Таку поведінку важко передбачити аналітичними методами без знання секретного ключа. Це усуває один тип атак, а саме атаки перебору ключів, які перебирають усі можливі ключі для розшифровки даних. Атаки є успішними дуже рідко, оскільки будь-яке намагання зламати ключ залежить від довжини ключа [2].

9. Фрактальна графіка. Фрактали широко застосовуються в комп'ютерній графіці для побудови зображень природних об'єктів, таких як: дерева, кущі, гірські ландшафти, поверхні морів тощо. Фрактали широко використовують і в кінематографі, де, наприклад, режисерська задумка вимагає реалістичного

пейзажу неіснуючої планети. Так, приміром, в "Поверненні Джедая" фрактали були використані для створення географії Місяця. З використанням фракталів можуть будуватися не тільки ірреальні зображення, але і цілком реалістичні (наприклад, фрактали нерідко використовуються при створенні хмар, снігу, берегових ліній, дерев і кущів тощо). Застосовувати фрактальні зображення можна в різних сферах, починаючи від створення звичайних текстур і фонових зображень, і закінчуючи фантастичними ландшафтами для комп'ютерних ігор або книжкових ілюстрацій. А створюються подібні фрактальні шедеври (так само як і векторні) шляхом математичних розрахунків. Будь-хто може створити своє власне оригінальне зображення. Так як це зробила італійка Сільвія Кордедда (рис. 1.8) Створивши серію фрактальних квітів, що тепер зачаровують не лише людей з інформаційною освітою, а й художників та мистецтвознавців.

Рис. 1.8. Фрактальнне зображення Сільвії Кордедди

Народження фрактальної геометрії прийнято пов'язувати з виходом у 1977 р. книги "Фрактальна геометрія природи" Бенуа Мандельбротта. Його і вважають батьком сучасної фрактальної геометрії і слова фрактал. Визначення фрактала, дане Мандельбротом, звучить так: "фракталом називається

структура, що складається з частин, які в якомусь сенсі подібні цілому". Мандельброт вивів слово fractal від латинського слова fractus, що означає розбитий (поділений на частини). І одне з визначень фрактала – це геометрична фігура, що складається з частин і яка може бути поділена на частини, кожна з яких представлятиме зменшену копію цілого.

Фрактальна графіка, як і векторна, заснована на математичних обчисленнях. Однак, базовим елементом є математична формула, ніяких об'єктів у пам'яті комп'ютера не зберігається і зображення будується виключно по рівняннях. Фрактальна графіка міститься у пакетах для наукової візуалізації для побудови, як найпростіших структур так і складних ілюстрацій, що імітують природні процеси та тривимірні об'єкти.

Фрактали можна поділити на три основні групи (відповідно до того, що лежить в їх основі, будь-то геометричні операції, формули чи певні дії з кольором): геометричні, алгебраїчні, стохастичні та системи ітерованих функцій IFS.

Основними галузями застосування є: економіка (фрактальний аналіз ринків), фізика (моделювання нелінійних процесів), біологія (моделювання природних процесів), радіотехніка (використання фрактальної геометрії при проектуванні антенних пристроїв), інформатика (фрактальне стиснення зображень при застосуванні систем ітеруючих функцій IFS), децентралізовані мережі (призначення IP-адрес у мережі Netsukuku) [17].

1.1.4. Програми фрактальної графіки

1.1.4.1. Art Dabbler

Знайомство з основами фрактальної графіки краще всього почати з пакету Art Dabbler. Цей редактор (створений фірмою Fractal Design, а зараз належить Corel) фактично є частковим варіантом програми Painter. Це відмінна програма для навчання не тільки комп'ютерній графіці, але перш за все азам малювання. Малий обсяг необхідної пам'яті (для його установки необхідно всього 10

Мбайт), а також простий інтерфейс, доступний навіть дитині, дозволяють використовувати його в шкільній програмі. Як і растровий редактор MS Paint, фрактальний редактор Art Dabbler особливо ефективний на початковому етапі освоєння комп'ютерної графіки.

Головна увага розробниками пакету Art Dabbler була приділена двом чинникам:

– створенню спрощеного інтерфейсу, основним елементом якого є коробки інструментальних наборів (званих тут висувними ящиками);

– можливості використання пакету як повчальна програма. Для реалізації цієї мети в комплект постачання пакету разом з самою програмою включений самовчитель "Вчися малювати" і повчальний фільм на компакт-диску. Пропоновані в них уроки малювання дозволяють крок за кроком спостерігати за процесом створення досвідченими художниками кольорових зображень засобами пакету Art Dabbler.

Рядок меню включає шість пунктів: стандартні для більшості програм – File, Edit і Help, а також Effects, Options і Tutors, які присутні в більшості графічних програм і не потребують додаткових коментарів.

Art Dabbler надає комплект ефектів (меню Effects), які можуть бути використані для зміни або спотворення зображень. Наприклад, ефект Texturize створює текстури паперу, полотна і тому подібне, розширюючи творчі можливості художника.

Слід зазначити, що в Art Dabbler висувними ящиками називаються всі інструментальні засоби точно так, як і, наприклад, в Photoshop аналогічні засоби називаються палітрами, а в CORELDRAW – докерами. У них зберігаються пензлі, олівці, гумка і інші інструменти, для активізації яких досить натиснути відповідну їм піктограму. На передніх стінках ящиків відображається невелика кількість кнопок і ручка, натиснувши яку користувач дістає доступ до всього набору здійснюваних через нього операцій завдяки додатковим кнопкам, що відкриваються (рис. 1.9).

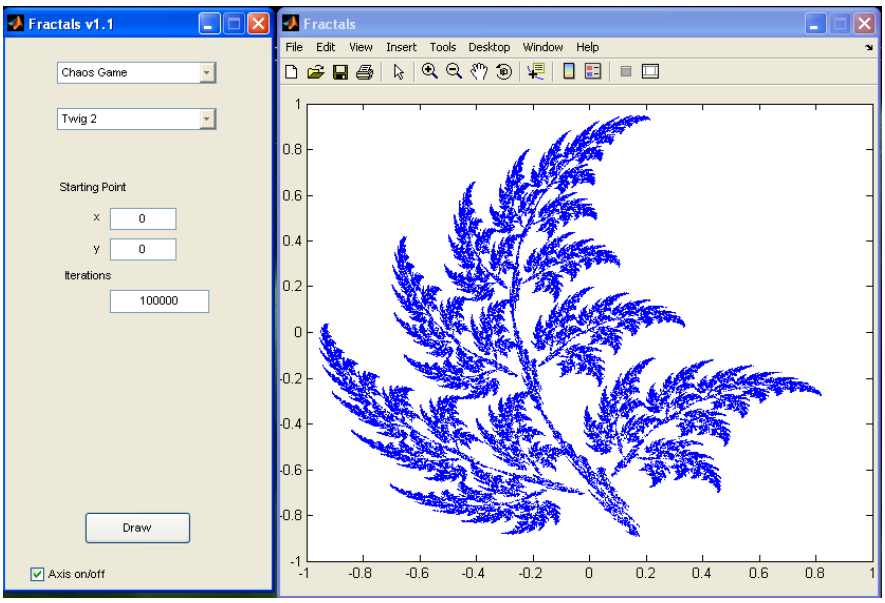

Рис. 1.9. Вигляд програми Art Dabbler

1.1.4.2. Ultra Fractal

Ultra Fractal – створення унікальних фрактальних зображень високої якості. Пакет відрізняється інтерфейсом, багато елементів якого нагадують інтерфейс Photoshop і супроводжується детальною і ілюстрованою документацією з серією туториалов, в яких поетапно розглядаються всі аспекти роботи з програмою. Ultra Fractal представлений двома редакціями: Standard Edition і розширеною Animation Edition, можливості якої дозволяють не лише генерувати фрактальні зображення, але і створювати анімацію на їх основі. Створені зображення можна візуалізувати у високій роздільній здатності, придатні для поліграфії, і зберегти у власному форматі програми або в одному з популярних фрактальних форматів. Візуалізовані зображення також можуть бути експортовані в один з растрових графічних форматів (jpg, bmp, png і psd), а готові фрактальні анімації – в AVI-формат.

Створення фрактальних зображень традиційне – скористатися формулою, що додається в постачанні, а потім редагувати параметри формули. Коли експеримент виявився невдалим, то останні дії легко скасувати. Готових фрактальніх формул дуже багато.

Однак не слід вважати, що фрактальне зображення криється лише у вдалій формулі, важливі й інші аспекти. Колірне налаштування, що задає вибір варіанту кольору і точне налаштування його параметрів. Налаштування кольору реалізоване на рівні солідних графічних пакетів, наприклад градієнти можна створювати і настроювати самостійно, коректуючи безліч параметрів, включаючи напівпрозору, і зберігати їх в бібліотеці для подальшого використання. Застосування шарів із можливістю зміни режимів їх змішування і коректуванням напівпрозорості дозволяє генерувати багатошарові фрактали і за рахунок накладення фрактальних зображень один на одного добиватися унікальних ефектів. Використання масок непрозорості забезпечує маскування певних областей зображення. Фільтри трансформації дозволяють виконувати відносно виділених фрагментів зображення різноманітні перетворення: масштабувати, дзеркально відображати, обрізати за шаблоном, спотворювати за допомогою завихорення або мигтінь, розмножувати за принципом калейдоскопа і так далі (рис. 1.10).

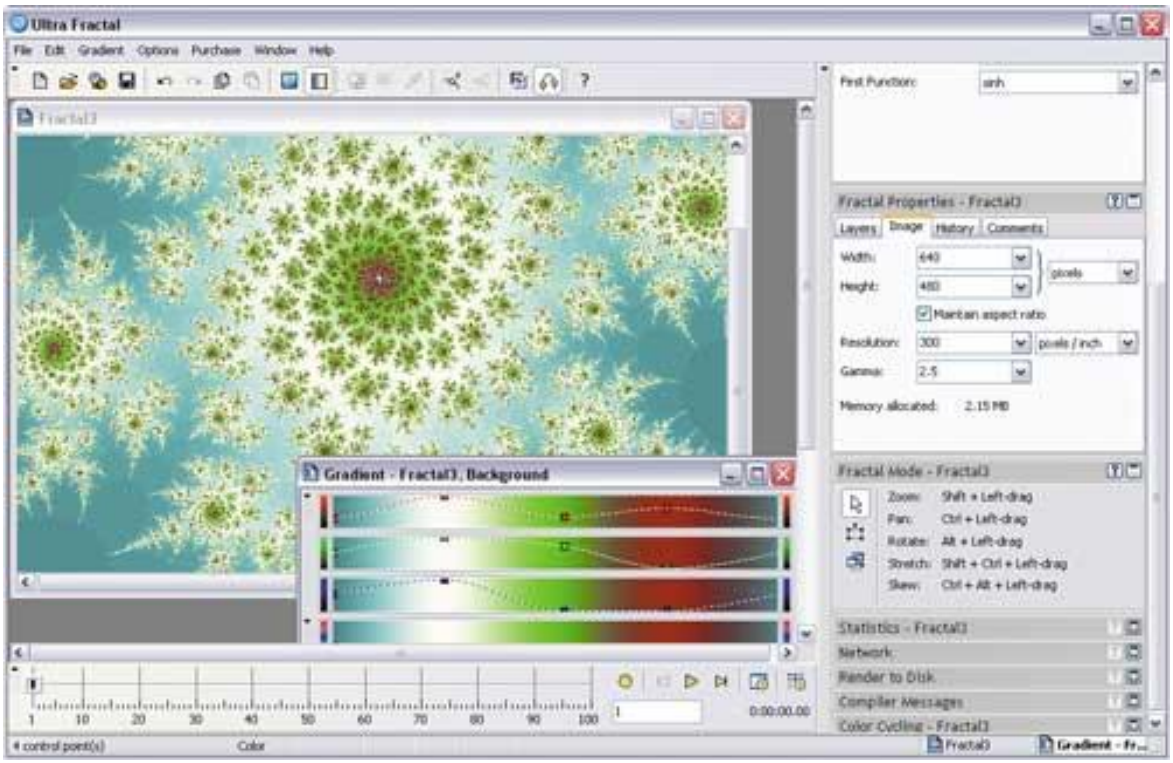

Рис. 1.10. Вигляд програми Ultra Fractal

1.1.4.3. Fractal Explorer

Fractal Explorer – створення зображень фракталів і трьохмірних аттракторов. Має зрозумілий класичний інтерфейс, який може бути настроєний для користувача. Підтримує стандартні формати фрактальних зображень (*.frp; *.frs; *.fri; *.fro; *.fr3, *.fr4 та ін.). Готові фрактальні зображення зберігаються у форматі *.frs і можуть бути експортовані в один з растрових графічних форматів (jpg, bmp, png і gif), а фрактальні анімації зберігаються як AVI-файли.

Генерація фракталів можлива двома способами – на основі базових фрактальних зображень, побудованих по вхідних в постачання формулах, або з нуля. Перший варіант дозволяє отримати цікаві результати просто, адже вибрати формулу не важко. Біля отриманого таким шляхом фрактального зображення можна змінити колірну палітру, добавити до нього фонове зображення. Можна фрактальне зображення трансформатувати, при необхідності масштабувати, змінити розміри зображення.

Створення зображення з нуля набагато складніше і передбачає вибір одного з двох способів. Можна вибрати тип фрактала майже з 150 варіантів. А потім вже перейти до зміни різноманітнихпараметрів: налаштуванню палітри, фону і ін. А можна спробувати створити свою призначену для користувача формулу, скориставшись вбудованим компілятором. Перед рендерингом готового зображення може потрібно проведення автоматичної корекції колірного балансу корекції, яскравості і контрастності (рис. 1.11).

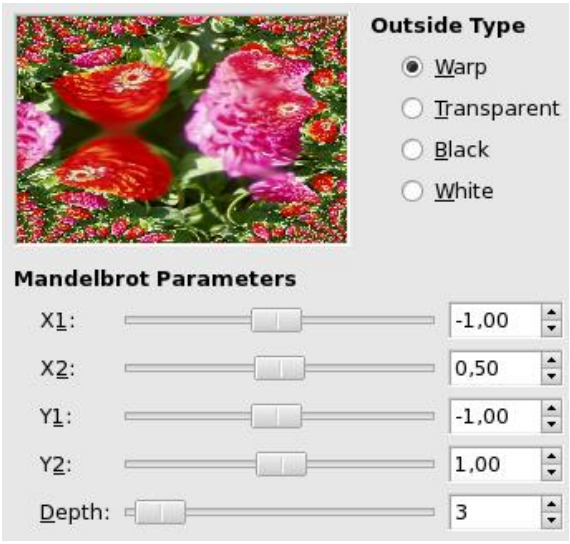

Рис. 1.11. Діалогове вікно програми Fractal Explorer

1.1.4.4. Fractracer

Fractracer – створення тривимірних зображень на базі фрактальної геометрії, що представляє собою щось середнє між генератором фракталів і 3D-редактором. Отримані в цій програмі форми зберігаються у вигляді зображень або перетворюються в тривимірні mesh-об'єкти, які в подальшому можуть бути використані в популярних пакетах 3D-графіки.

Для створення зображень передбачено два варіанти: використання встановлених прикладів фрактальних об'єктів, які потім нескладно видозмінити бажаним чином, і створення проектів з нуля – на базі програмного коду. Масштабування, обертання і переміщення отриманого фрактального зображення виробляються мишею, а всі інші настройки - через численні панелі. Найважливішими панелями є панель об'єктів (Objects) і панель параметрів (Parameters). Усе в нові на зображення елементи – це окремі незалежні об'єкти, а саме зображення являє собою наповнену ними сцену. Тому будь-який з доданих об'єктів, незалежно від того, на якому етапі він був впроваджений у сцену, в подальшому може бути без проблем видалений. Об'єкти розташовані в ієрархічній структурі, і від їх положення по відношенню один до одного залежить вид зображення. Набір об'єктів регулюється на панелі Objects

(додаткові об'єкти вставляються з вбудованої бібліотеки), а властивості об'єктів – на панелі Parameters. Тут, крім зміни параметрів фрактальних формул і колірних налаштувань, можливе корегування положення камери, джерел світла, трансформацій і багато чого іншого.

Рендеринг у програмі також нагадує візуалізацію в 3D-рішеннях, оскільки при цьому враховуються встановлені параметри освітлення, наявність / відсутність тіней і необхідність розмивання країв. Можливе створення анімації на основі фрактальних зображень з урахуванням визначення ключових кадрів і бажаної тривалості (рис. 1.12).

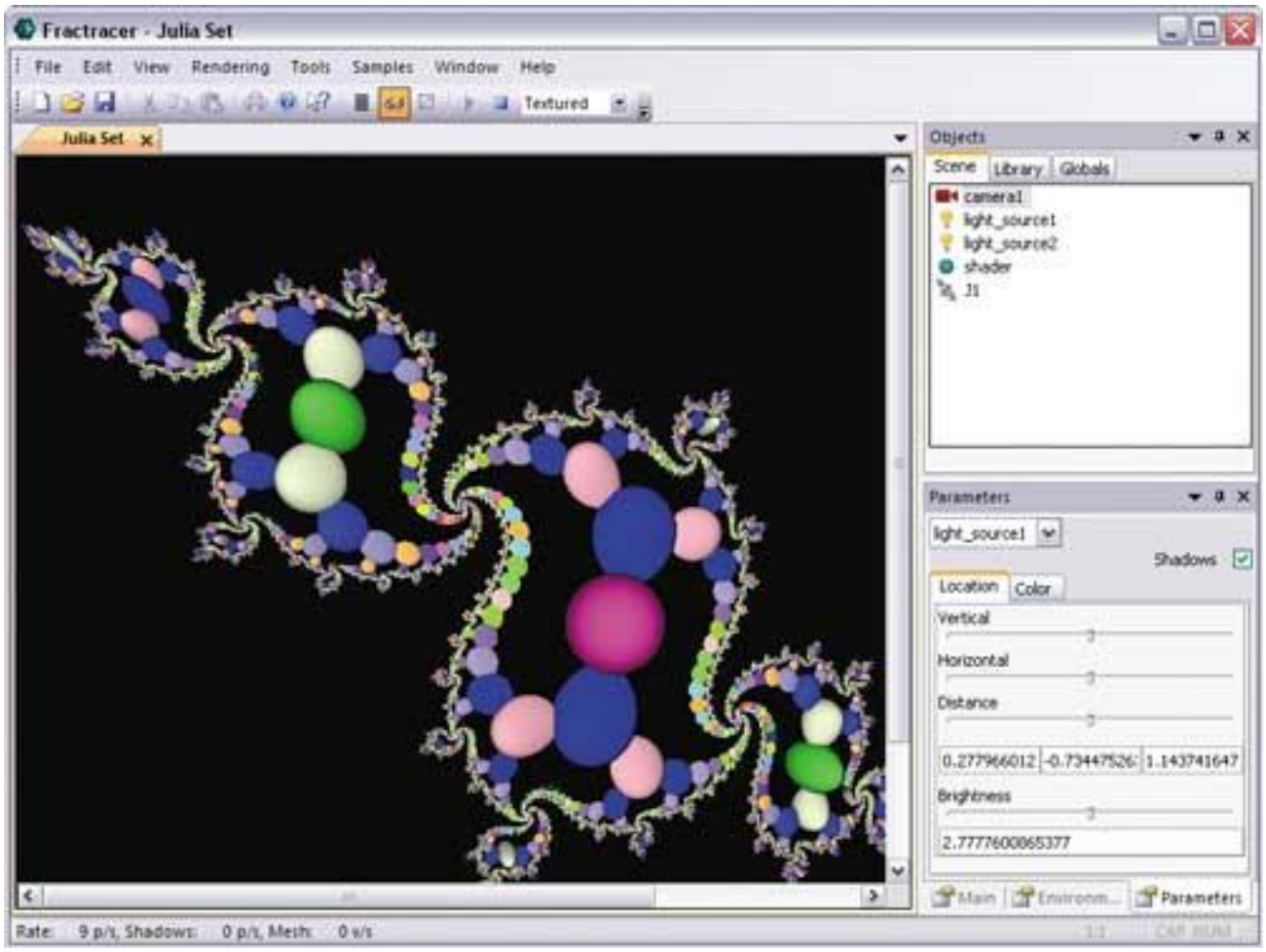

Рис. 1.12. Вигляд програми Fractracer

1.2. Призначення розробки та галузь застосування

Мета кваліфікаційної роботи - це дослідження способів моделювання фракталів та розроблення програмного засобу для побудови фрактальних зображень.

Розроблений програмний додаток призначений для вирішення наступних задач:

1. Аналіз математичних моделей фрактальних множин.

2. Розробка алгоритмів моделювання фрактальних структур, оцінка характеристик алгоритмів, а також властивостей множин, породжуваних запропонованими алгоритмами.

3. Розробка програмного комплексу для проведення обчислювального експерименту та дослідження структур даних на основі фрактальних моделей, отриманих з використанням запропонованих алгоритмів.

4. Тестування розробленого програмного забезпечення при різних значеннях параметрів моделей.

Область застосування фракталів надзвичайно велика. Вони використовуються в комп’ютерній графіці, медицині, природничих науках, телекомунікаціях тощо.

Фрактали знаходять все більше і більше застосування в науці. Основна причина цього полягає в тому, що вони описують реальний світ іноді навіть краще, ніж традиційна фізика або математика. Ось кілька прикладів:

1. Комп’ютерні системи. Найбільш корисним використанням фракталів в комп'ютерній науці є фрактальне стиснення даних. В основі цього виду стиснення лежить той факт, що реальний світ добре описується фрактальною геометрією. При цьому, картинки стискаються набагато краще, ніж це робиться звичайними методами (такими як jpeg або gif). Інша перевага фрактального стиснення в тому, що при збільшенні картинки, не спостерігається ефекту пікселізації (збільшення розмірів точок до розмірів, що спотворюють

зображення). При фрактальному ж стисненні, після збільшення, картинка часто виглядає навіть краще, ніж до нього.

2. Фізика рідин і вогню:

— вивчення турбулентності в потоках дуже добре підлаштовується під фрактали. Турбулентні потоки хаотичні і тому їх складно точно змоделювати. І тут допомагає перехід до фракталів, що сильно полегшує роботу інженерам і фізикам, дозволяючи їм краще зрозуміти динаміку складних потоків;

— за допомогою фракталів також можна змоделювати язики полум'я.

3. Телекомунікації. Для передачі даних на відстані використовуються антени, що мають фрактальні форми, що сильно зменшує їх розміри і вагу.

4. Фізика поверхонь. Фрактали використовуються для опису кривизни поверхонь. Нерівна поверхня характеризується комбінацією з двох різних фракталів.

5. Біологія. Моделювання хаотичних процесів, зокрема при описі моделей популяцій.

Їх допомога необхідна, наприклад, коли потрібно задати лінії та поверхні дуже складної форми за допомогою декількох коефіцієнтів. Фактично можна сказати, що знайдений спосіб легкої та зручної уяви складних неевклідових об’єктів, образи яких схожі на природні.

Результатом виконання даної роботи є створення додатка, за допомогою якого у зручній формі можна ознайомитися з моделями фракталів, а також дослідити закономірності в побудові фрактальних зображень при зміні параметрів фрактальної моделі.

1.3. Підстава для розробки

Підставою для розробки кваліфікаційної роботи на тему «Розробка інформаційної системи моделювання фрактальних зображень

інструментальними засобами» є наказ по Національному технічному університету «Дніпровська політехніка» від __.__. 2021р. № ____-л.

1.4. Постановка завдання

Мета кваліфікаційної роботи - дослідження способів моделювання фракталів та розроблення програмного засобу для побудови фрактальних зображень.

Основною задачею для побудови фрактальних зображень є знаходження координат точок початку та кінця ланок предфракталу в двовимірному просторі (x, y).

Для спрощення задачі доцільно використовувати полярну систему координат – двовимірну систему координат, в якій кожна точка на площині визначається двома числами – кутом та відстанню.

Будь-яку точку в полярній системі координат можна представити двома полярними координатами, що зазвичай мають назву r (радіальна координата) та φ (кутова координата, полярний кут, азимут, інколи пишуть θ або t). Координата r відповідає відстані до полюса, а координата φ дорівнює куту в протигодинниковому напрямі від променя через 0° (інколи називається полярною віссю) [17].

Пару полярних координат r та φ можна перевести в Декартові координати x та y шляхом застосування тригонометричних функцій синуса та косинуса:

$$\begin{aligned} x &= r\cos\varphi \\ y &= r\sin\varphi \end{aligned} \qquad (1.2)$$

де r = 1, як довжина ланки предфракталу, φ – кут повороту.

Для визначення кутової координати φ, слід взяти до уваги два такі міркування:

– для r = 0, φ може бути довільним дійсним числом;

– для r≠ 0, аби отримати унікальне значення φ, слід обмежитись інтервалом в 2π. Зазвичай, обирають інтервал [0, 2π) або (−π, π].

Для знаходження координат кожної наступної з ланок предфракталу використаємо формули:

$$x = \sum_{n=0}^{k} Cos(2 \times \pi \times \varphi \times n)$$
$$y = \sum_{n=0}^{k} Sin(2 \times \pi \times \varphi \times n) \quad (1.3)$$

де k – кількість ітерацій.

В результаті k повторень отримується масив координат ланок фракталів, що з’єднані між собою і утворюють малюнок

Розроблений програмний додаток призначений для вирішення наступних задач:

1. Аналіз математичних моделей фрактальних множин.

2. Розробка алгоритмів моделювання фрактальних структур, оцінка характеристик алгоритмів, а також властивостей множин, породжуваних запропонованими алгоритмами.

3. Розробка програмного комплексу для проведення обчислювального експерименту та дослідження структур даних на основі фрактальних моделей, отриманих з використанням запропонованих алгоритмів.

4. Тестування розробленого програмного забезпечення при різних значеннях параметрів моделей.

З урахуванням проведеного аналізу фрактальних моделей під час виконання кваліфікаційної роботи необхідно:

1. Розробити веб-сервіс для побудови фрактальних структур на основі:
   - ітераційного алгоритму;
   - рекурсивного алгоритму;
   - алгоритму генерації згідно метода IFS.

2. Провести обчислювальний експеримент і дослідження структур даних на основі фрактальних моделей, отриманих з використанням запропонованих алгоритмів.

3. За допомогою представленого в роботі сайту забезпечити можливість ознайомитися з моделями фракталів, дослідити закономірності в побудові фрактальних зображень при зміні параметрів фрактальної моделі.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Структура розробленого веб-додатку повинна складатися з наступних складових:

1. Стартова сторінка, яка містить в собі інформацію про ресурс та переходи до вирішення завдань.

2. Сторінка побудови кривої Коха.

3. Сторінка побудови папороті Барнслі.

4. Сторінка побудови дерева Піфагора.

5. Сторінка побудови множини Мандельброта.

Кожна із сторінок з вирішенням задач повинна містити короткі теоретичні відомості про застосований метод та саму реалізацію метода за введеними користувачем даними та виводити результат в графічному виді.

1.5.2. Вимоги до інформаційної безпеки

До розробленого додатку сформовані наступні вимоги забезпечення безпеки і цілісності даних:

- при відмові системи не повинні бути пошкоджені;
- як система, так і оброблені дані, повинні легко переноситися;
- час відновлення після збою 5 хвилин.

Для надійної роботи системи необхідно:

1. Використовувати ліцензійне програмне забезпечення на сервері.
2. Здійснювати захист від вірусів на сервері.
3. Здійснювати захист від несанкціонованого доступу.
4. Застосовувати на сервері джерело безперебійного живлення для захисту від перепадів напруги або збоїв у живленні.
5. Здійснювати контроль даних, що вводяться користувачами.
6. Автоматично завершувати сеанс роботи з користувачем у разі тривалої перерви його активності.

1.5.3. Вимоги до складу та параметрів технічних засобів

Даний додаток має використовуватися на ПК з наступними характеристиками:

- процесор: 1 ГГц і швидше з підтримкою PAE, NX і SSE2;
- RAM: 1 Гбайт (32 біт) або 2 Гбайт (64 біт);
- HDD: 16 Гбайт (32 біт) або 20 Гбайт (64 біт);
- відеокарта: підтримка Microsoft DirectX 9 з драйвером WDDM.

1.5.4. Вимоги до інформаційної та програмної сумісності

Сервіс повинний бути створений за допомогою HTML5/CSS3 та мови програмування JavaScript та сумісний з операційними системами Microsoft Windows XP SP3 та її пізнішими версіями.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Кваліфікаційна робота на тему «Розробка інформаційної системи моделювання фрактальних зображень інструментальними засобами» призначена для створення веб-сервісу, який забезпечує побудову фрактальних структур та проведення обчислювального експерименту і дослідження змін у структурах, що будуються при різних значеннях параметрів фрактальних моделей.

Веб-сервіс дозволяє побудувати такі фрактали: крива Коха, дерево Піфагора, папороть Барнслі та множина Мандельброта.

2.2. Опис застосованих математичних методів

2.2.1. Тріадна крива Коха

Крива Коха – фрактальна крива, описана в 1904 році шведським математиком Хельге фон Кохом. Крива Коха цікава тим, що ніде не має дотичних, тобто ніде недиференційована, хоча всюди неперервна.

Тріадна крива Коха – один зі стандартних прикладів, що наводяться для підтвердження того, що крива може мати фрактальну розмірність D>1.

Побудова кривої Коха починається з прямолінійного відрізка одиничної довжини $L(1)=1$. Цей початковий відрізок називається затравкою і може бути замінений яким-небудь багатокутником, наприклад рівностороннім трикутником, квадратом. Затравка – це 0-е покоління кривої Коха.

Побудова кривої Коха продовжується в такий спосіб: кожну ланку затравки ми заміняємо утворюючим елементом. У результаті такої заміни ми одержуємо перше покоління – криву з чотирьох прямолінійних ланок, кожна

довжиною по 1/3. Довжина всієї кривої першого покоління складає величину L(1/3)=4/3. Наступне покоління одержується при заміні кожної прямолінійної ланки зменшеним утворюючим елементом. У результаті ми одержуємо криву другого покоління, що складена з $N=4^2=16$ ланок, кожна довжиною $\sigma=3^{-2}=9$. Довжина кривої другого покоління дорівнює $L(1/9)=(4/3)^2=16/9$. Замінюючи всі ланки попереднього покоління кривої зменшеним утворюючим елементом, одержуємо нове покоління кривої. Крива n-го покоління при будь-якому кінцевому n називається передфракталом.

Простежимо в подробицях за тим, як одержується вираз для D. Довжина передфрактала n-го покоління визначається формулою $L(\sigma)=(4/3)^n$. Довжина кожної ланки складає $\sigma=3^{-n}$. Приймаючи до уваги, що число поколінь n можна переписати у вигляді $n=-\ln\sigma/\ln 3$, запишемо довжину передфрактала таким чином:

$$L(\delta) = (4/3)^n = \exp(-\frac{\ln \delta(\ln 4 - \ln 3)}{\ln 3}) = \delta^{i-D} \qquad (2.1)$$

.

У наближеному вигляді маємо: $L(\sigma)=a*\sigma^{1-D}$, в якій $D=\ln 4/\ln 3\approx 1{,}2628$. Число сегментів дорівнює $N(\sigma)=4^n=4^{-\ln\sigma/\ln 3}$ і може бути записане у виді $N(\sigma)=\sigma^{-D}$.

Як буде показано далі, D - фрактальна розмірність тріадної кривої Коха. Насамперед зауважимо, що дана побудова дозволяє в будь-якому поколінні одержувати нормальну криву кінцевої довжини. Але при збільшенні числа поколінь величина σ прагне до нуля і довжина кривої розходиться. Ясно, що множина точок, яка одержана як границя нескінченно великої кількості ітерацій процедури Коха, не є кривою, для якої довжина є зручною мірою. Але якщо ми виберемо пробну функцію $h(\sigma)=\sigma^d$, то одержимо d-міру:

$$M_d=\Sigma h(\sigma)=N(\sigma)h(\sigma)=\sigma^{-D}\sigma^d \cdot \qquad (2.2)$$

Ми бачимо, що міра M_d залишається скінченою і дорівнює одиниці тільки в тому випадку, якщо розмірність d, що входить у пробну функцію h(σ), дорівнює D. Виходячи з цього, можна зробити висновок, що критична розмірність і, отже, розмірність Хаусдорфа – Безиковича для тріадної кривої Коха дорівнює D=ln4/ln3. На кожній стадії побудови передфрактали Коха можуть бути розтягнуті в пряму лінію, тому топологічна розмірність тріадної кривої Коха дорівнює D_T =1. З того, що розмірність Хаусдорфа – Безиковича D для кривої Коха більше її топологічної розмірності D_T, ми робимо висновок, що крива Коха є фрактальною множиною з фрактальною розмірністю D=ln4/ln3.

2.2.2. Дракон Хартера-Хейтуея

Крива дракона вперше була описана в популярній літературі в журналі “Scientific American” у 1967 році. Замітка про неї з’явилась в колонці “Математичні ігри”, яку редагував Мартин Гарднер. Спочатку використовувалась повна назва кривої – “дракон Хартера-Хейтуея”, яку їй дав засновник комп’ютерної фрактальної геометрії Бенуа Мандельброт.

Опишемо алгоритм побудови даної кривої. Нехай утворюючим елементом будуть два рівних відрізка, з’єднаних під прямим кутом. Розглянемо горизонтальний відрізок як криву дракона нульового порядку. Розділимо відрізок навпіл та побудуємо на ньому прямий кут, як показано на рис. 2.1. Отримаємо криву дракона першого порядку. На сторонах прямого кута знову побудуємо прямі кути. При цьому вершина першого кута завжди знаходиться справа, якщо дивитися з точки A (початку кривої) уздовж першого відрізка кривої, а напрямки, в яких будуються вершини інших кутів, чергуються.

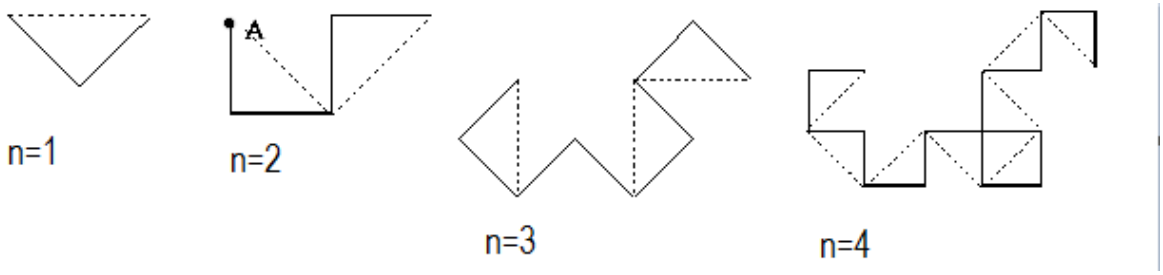

Рис. 2.1. Побудова кривої дракона 4-го порядку

Крива дракона – один з найпопулярніших об’єктів, який зустрічається майже в усіх напрямках фрактальної геометрії.

2.2.3. Дерево Піфагора

Дерево Піфагора - плоский фрактал, заснований на фігурі, що відома як «Піфагорові штани».

Піфагор під час доведення теореми побудував фігуру, де на сторонах прямокутного трикутника розташовані квадрати. В наш час ця фігура виросла в ціле дерево. Вперше дерево Піфагора побудував Босман (1891—1961) під час другої світової війни, з використанням звичайної креслярської лінійки.

Однією з властивостей дерева Піфагора є те, що, якщо площа першого квадрата дорівнює одиниці, тоді на кожному рівні сума площ квадратів також буде дорівнювати одиниці (рис. 2.2).

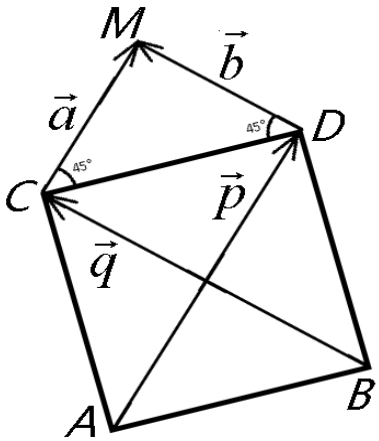

Рис. 2.2. Класичне дерево Піфагора

Якщо в класичному дереві Піфагора кут дорівнює 45 градусам, то також можна побудувати і узагальнене дерево при використанні інших кутів. Таке дерево називають "обдуваним". Якщо зображати тільки відрізки, що поєднують яким-небудь чином обрані "центри" трикутників, тоді отримаємо "оголене дерево Піфагора".

Опишемо рекурсивну функцію, яка буде малювати дерево Піфагора. Функція отримує наступні параметри:

- кількість ітерацій, яку потрібно виконати;
- координати точок вершин квадрата;
- координати вектора $\vec{v} = \overrightarrow{AC} = \overrightarrow{BD}$.

Спочатку обчислимо координати точки M. Для цього обчислимо вектори $\vec{p} = \overrightarrow{AD}, \vec{q} = \overrightarrow{BC}$. Зауважимо, що $\vec{a} = \frac{\vec{p}}{2}, \vec{b} = \frac{\vec{q}}{2}$.

Тепер нескладно обчислити координати точки M, додавши до точки C вектор $\vec{a}$. Далі ми можемо без труднощів обчислити координати вершин квадратів, використовуючи рівність відповідних векторів.

2.2.4. Фрактали на основі метода IFS

Цю групу містять фрактали, які генеруються згідно метода «систем ітеративних функцій» – IFS (Iterated Functions Systems). Даний метод може бути описаний як послідовний ітеративний розрахунок координат нових точок у просторі:

$$\begin{cases} x_{k+1} = F_x(x_k, y_k), \\ y_{k+1} = F_y(x_k, y_k), \end{cases} \tag{2.3}$$

де F_x та F_y – функції перетворення координат, наприклад афінного перетворення.

Ці функції визначають форму фрактала. У випадку афінного перетворення необхідно знайти відповідні числові значення коефіцієнтів.

Прикладом такого фрактала являється лист папороті (рис 2.3).

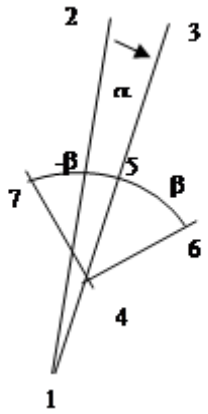

Рис. 2.3. Побудова листа папороті на основі метода IFS

Для початку ітерацій необхідно задати початкові координати лінії – точки 1 і 2. На кожному кроці ітерації розраховуються координати інших точок.

Точка 3 виходить поворотом точки 2 на кут **α** відносно центра в точці 1:

$$x3=(x2-x1)\cos a-(y2-y1)\sin a+x1;\quad y3=(x2-x1)\sin a+(y2-y1)\cos a+y1. \qquad (2.4)$$

Якщо **α** = 0, то ствол та всі гілки прямі.

Знаходимо точку 4. Від неї будуть поширюватися гілки. Нехай відношення довжин відрізків 1-4 та 1-3 дорівнює k ($0<k<1$). Тоді точка 4 має наступні координати:

$$x4=x1(1-k)+kx3;\ y4=y1(1-k)+ky3. \qquad (2.5)$$

Задаємо довжину та кут нахилу гілок, які виходять з точки 4. Знайдемо координати точки 5: параметр k1 – відношення довжин відрізків 4-5 та 4-3 ($0<k1<1$):

$$x5=x4(1-k1)+k1x3;\ \ y5=y4(1-k1)+k1y3. \quad (2.6)$$

Точка 6 виходить поворотом точки 5 відносно точки 4 на кут β, а точка 7 – на кут (-β):

$$x6=(x5-x4)\cos b-(y5-y4)\sin b+x4;\ y6=(x5-x4)\sin b+(y5-y4)\cos b+y4, \quad (2.7)$$

$$x7=(x5-x4)\cos b+(y5-y4)\sin b+x4;\ y7=-(x5-x4)\sin b+(y5-y4)\cos b+y4. \quad (2.8)$$

Після обчислення опорних точок на кожному кроці малюється відрізок 1-4. В залежності від номера ітерації колір відрізка можна змінювати.

Таким чином, фрактал задається як послідовність афінних перетворень координат точок. Величини a, b, k, k1 – це параметри, які описують вид фрактала в цілому. Вони являють собою константи на протязі всього ітеративного процесу. Це дає нам можливість використовувати тільки операції додавання, віднімання та множення в ітераціях, якщо ми обчислимо заздалегідь наступні значення:

$$A=\cos\alpha;\ B=\sin\alpha;\ C=(1-k);\ D=k;E=1-k1;\ F=k1;\ G=\cos\beta;\ H=\sin\beta. \quad (2.9)$$

2.3. Опис використаних технологій та мов програмування

2.3.1. Опис типу веб-сайту

Сайти за своїм призначенням, за своєю цільовою аудиторією бувають різні. В залежності від цього розрізняють декілька типів веб-сайтів в Інтернет:

- сайт-візитка,
- промо-сайт,
- електронний магазин,
- інформаційний сайт,
- корпоративне представництво,
- портал,

– система управління підприємством.

Беручи до уваги цілі і завдання, які перед нами поставлені, наш веб-сайт відноситься до інформаційних та буде містити вичерпну інформацію з певної предметної області. Сайти цього типу, як правило, містять статті, а також такі сервіси як: опитування, голосування, розсилки, некомерційного типу, а також реалізують де-які розрахункові дії.

Мета такого веб-сайту – представити певну інформацію в Інтернет, надання інформаційних послуг.

Властивості:

– загальне призначення сайту – надання докладної, вичерпної інформації користувачеві;

– характеристика і основні елементи – побудови фрактальних структур та проведення обчислювального експерименту і дослідження змін у структурах, що будуються при різних значеннях параметрів фрактальних моделей;

– кількість сторінок – варіюється залежно від наповнення, в нашому випадку 5;

– тип і характеристика дизайну – в залежності від специфіки інформаційного наповнення, може бути як креативним, так і суворо діловим;

– система навігації – проста;

– частота і необхідність поновлення – базова;

– ким є відвідувачі – цільова аудиторія, випадкові відвідувачі;

Завдання:

– дати користувачеві максимальну та вичерпну інформацію;

– легкість за контентом, щоб у відвідувача не було проблем з завантаженням.

2.3.2. Опис технологій і мов програмування

Сервіс створений за допомогою HTML5/CSS3 та мови програмування JavaScript та сумісний з операційними системами Microsoft Windows XP SP3 та її пізнішими версіями.

HTML (англ. Hyper Text Markup Language - мова розмітки гіпертекстових документів) - стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML. Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді.

HTML разом із каскадними таблицями стилів та вбудованими скриптами - це три основні технології побудови веб-сторінок.

HTML впроваджує засоби для:

– створення структурованого документу шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;

– отримання інформації із Всесвітньої мережі через гіперпосилання;

– створення інтерактивних форм;

– включення зображень, звуку, відео, та інших об'єктів до тексту.

Каскадні таблиці стилів (англ. Cascading Style Sheets або скорочено CSS) – спеціальна мова, що використовується для опису сторінок, написаних мовами розмітки даних.

CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг – можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS).

Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш структурованим та простим, прибрати повтори тощо. CSS також дозволяє адаптувати контент до різних умов відображення (на

екрані монітора, мобільного пристрою, у роздрукованому вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових браузерах та ін.)

JavaScript – динамічна, об'єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. Мова JavaScript також використовується для програмування на стороні серверу (подібно до таких мов програмування, як Java і C#), розробки ігор, стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite), всередині PDF-документів тощо.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтова¬ної), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частковофункціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

jQuery – популярна JavaScript - бібліотека з відкритим сирцевим кодом. Вона була представлена у січні 2006 року у BarCamp NYC Джоном Ресіґом (John Resig). Згідно з дослідженнями організації W3Techs, JQuery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

jQuery є вільним програмним забезпеченням під ліцензією MIT (до вересня 2012 було подвійне ліцензування під MIT та GNU General Public License другої версії).

Синтаксис jQuery розроблений, щоб зробити орієнтування у навігації

зручнішим завдяки вибору елементів DOM, створенню анімації, обробки подій, і розробки AJAX-застосунків. jQuery також надає можливості для розробників, для створення плагінів у верхній частині бібліотеки JavaScript. Використовуючи ці об'єкти, розробники можуть створювати абстракції для низькорівневої взаємодії та створювати анімацію для ефектів високого рівня. Це сприяє створенню потужних і динамічних веб-сторінок.

AJAX (Asynchronous JavaScript And XML) – підхід до побудови користувацьких інтерфейсів веб-застосунків, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані. AJAX – один з компонентів концепції DHTML.

AJAX — це не самостійна технологія, а швидше концепція використання декількох суміжних технологій. AJAX підхід до розробки, який призначених для користувача інтерфейсів, комбінує кілька основних методів і прийомів:

– використання DHTML для динамічної зміни змісту сторінки;

– використання XMLHttpRequest для звернення до сервера «на льоту», не перезавантажуючи всю сторінку повністю;

– альтернативний метод — динамічне підвантаження коду JavaScript в тег <SCRIPT> з використанням DOM, що здійснюється із використанням формату JSON);

– динамічне створення дочірніх фреймів.

Денвер (від скор. Д.н.w.р або ДНВР – джентльменський набір Web-розробника) – набір дистрибутивів і програмна оболонка, призначені для створення та налагодження сайтів (веб-додатків, іншого динамічного вмісту інтернет-сторінок) на локальному ПК (без необхідності підключення до мережі Інтернет) під управлінням ОС Windows.

2.4. Опис структури програми та алгоритмів її функціонування

2.4.1. Алгоритм побудови кривої Коха

Алгоритм побудови кривої Коха включає наступні кроки:

1. беремо одиничний відрізок;
2. поділяємо цей відрізок на три рівні частини та замінюємо середній інтервал рівностороннім трикутником без цього сегмента. В результаті утворюється ламана, що складається з чотирьох ланок довжини 1/3.
3. повторюємо операцію для кожної з чотирьох отриманих ланок.

Кількість повторних виконань алгоритму залежить від кількості введених користувачем ітерацій.

Програмна реалізація побудови кривої Коха: алгоритм реалізовано мовою JavaScript. В основі програми лежить рекурсивна функція, яка рахує параметри побудови для кожної прямої, яка входить до кривої Коха (рис. 2.5).

```
var drawKochCurve = function(x1, y1, x2, y2, angle, k) {
    if( ((x2 - x1)*(x2 - x1) + (y2 - y1)*(y2 - y1)) < k) {
        drawLine(x1,y1,x2,y2);
   return;
}

var xa = x1 + 1/3 * (x2 - x1),
ya = y1 + 1/3 * (y2 - y1),

xb = x1 + 2/3 * (x2 - x1),
yb = y1 + 2/3 * (y2 - y1),

xc = xa + (xb - xa) * Math.cos(-angle) - (yb - ya) * Math.sin(-angle),
yc = ya + (yb - ya) * Math.cos(-angle) + (xb - xa) * Math.sin(-angle);

drawKochCurve(x1,y1, xa,ya, angle, k);
drawKochCurve(xa,ya, xc,yc, angle, k);
drawKochCurve(xc,yc, xb,yb, angle, k);
drawKochCurve(xb,yb, x2,y2, angle, k);

}
```

Рис. 2.5. Програмна реалізація алгоритму побудови кривої Коха

Ця функція використовує сторонню функцію drawLine, яка будує пряму за заданими параметрами.

2.4.2. Побудова папороті Барнслі

Папороть Барнслі будується за допомогою 4-ох афінних перетворень наступного виду:

$$f(x,y) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}. \tag{2.10}$$

Для побудови використана матрицю значень, створену М. Барнслі (рис. 2.6):

w	a	b	c	d	e	f	p
$f1$	0	0	0	0.16	0	0	0.01
$f2$	0.85	0.04	-0.04	0.85	0	1.6	0.85
$f3$	0.2	-0.26	0.23	0.22	0	1.6	0.07
$f4$	-0.15	0.28	0.26	0.24	0	0.44	0.07

Рис. 2.6. Матриця значень М. Барнслі

Стовпці a - f – коефіцієнти рівняння, а p – коефіцієнт вірогідності, x та y – координати. Дана таблиця відповідає наступним перетворенням:

$$f(x,y) = \begin{bmatrix} 0.00 & 0.00 \\ 0.00 & 0.16 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{2.11}$$

$$f(x,y) = \begin{bmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.00 \\ 1.60 \end{bmatrix}, \tag{2.12}$$

$$f(x,y) = \begin{bmatrix} 0.20 & -0.26 \\ 0.23 & 0.22 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.00 \\ 1.60 \end{bmatrix}, \tag{2.13}$$

$$f(x,y) = \begin{bmatrix} -0.15 & 0.28 \\ 0.26 & 0.24 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.00 \\ 0.44 \end{bmatrix}. \tag{2.14}$$

Зображення побудови папороті Барнслі наведено на рис. 2.7.

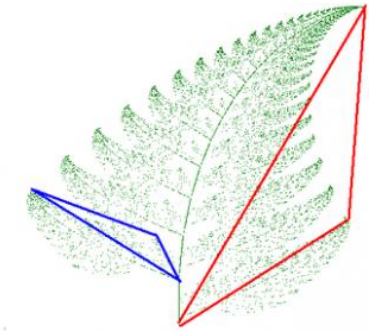

Рис. 2.7. Побудова папороті

Алгоритм побудови папороті Барнслі має наступний вигляд:

1. Зафіксувати першу точку на початку координат (x0 = 0, y0 = 0).
2. Обчислити коефіцієнт ймовірності p.
3. На наступній ітерації будується точка A1 (a * x0 + b * y0 + c, e * x0 + f * y0 + q), де коефіцієнти обираються з урахуванням “попадання” p у відповідний діапазон значень:

— якщо p = 0.01, то xn + 1 = 0; yn + 1 = 0.16 yn (побудова точки в основі стовбура);

— якщо p потрапляє в проміжок 0..0.85, то xn + 1 = 0.85 xn + 0.04 yn; yn + 1 = −0.04 xn + 0.85 yn + 1.6. Це точки, що потрапили в червоний трикутник (рис.2.7), і з цього перетворення виникає множина точок, загальна гущина і висота листя папороті;

— якщо p потрапляє в проміжок 0.85..0.93, то xn + 1 = 0.2 xn − 0.26 yn; yn + 1 = 0.23 xn + 0.22 yn + 1.6 (випадки потрапляння точки в синій трикутник);

— якщо p потрапляє в проміжок 0.93..0.99, то xn + 1 = −0.15 xn + 0.28 yn; yn + 1 = 0.26 xn + 0.24 yn + 0.44 (випадки потрапляння точки в трикутник, що симетричний синьому).

Програмна реалізація побудови папороті Барнслі виконана за допомогою мови програмування JavaScript. В основі програми лежить рекурсивна функція, яка працює за допомогою метода IFS (рис. 2.8).

```
function pBarnsleyFern(lim, ferncolor) {
    initVariables();
    // Главный цикл
    for(var i = 0; i < lim; i++) {
        r = randgp(100);

        if (r <= 1) {
          xw = 0;
          yw = 0.16 * y;
        }

        else if (r <= 8) {
          xw = 0.2 * x - 0.26 * y;
          yw = 0.23 * x + 0.22 * y + 1.6;
        }

        else if (r <= 15) {
          xw = -0.15 * x + 0.28 * y;
          yw = 0.26 * x + 0.24 * y + 0.44;
        }

        else {
          xw = 0.85 * x + 0.04 * y;
          yw = -0.04 * x + 0.85 * y + 1.6;
        }

        x = xw;
        y = yw;
        ctx.fillStyle=ferncolor;
        ctx.fillRect(x * 90 + 480, -y * 90 + 1000, 1, 1);
    }
}
```

Рис. 2.8. Програмна реалізація алгоритму побудови папороті Барнслі

Ця функція використовує сторонні функції initVariables та randgp. Функція initVariables ініціалізує потрібні змінні, а функція randgp повертає випадкове число в діапазоні від 0 до заданого аргументу.

2.4.3. Побудова дерева Піфагора

Алгоритм побудови дерева Піфагора має наступний вигляд:

1. Будуємо вертикальну пряму.
2. З верхнього кінця цієї прямої будуємо ще дві прямі під кутами, заданими користувачем.
3. Викликаємо функцію побудови двох наступних відрізків для кожного з побудованих відрізків стільки разів, скільки ітерацій ввів користувач.

Програмна реалізація побудови дерева Піфагора виконана за допомогою мови програмування JavaScript. В основі програми лежить рекурсивна функція, яка будує пряму з певною довжиною та під певним кутом до попередньої прямої (рис. 2.9).

```
function drawTreeWithTwoI(x, y, length, angle) {

            if (length >= minlength)
            {
                drawLine(x, y, length, angle);
                x += length * Math.cos(angle); y -= length * Math.sin(angle);
                angle+=dAngle;
                if (length > 200)
                length *= 0.31;
                else length *= 0.72;
                i++;

            if (i < iterations) {
                drawTreeWithTwoI(x, y, length, angle);
                drawTreeWithTwoI(x, y, length, angle-Math.PI/2);
            }
```

Рис.2.9. Програмна реалізація алгоритму побудови дерева Піфагора

Ця функція використовує сторонні функції drawline та angle. Функція drawline призначена для проведення прямої, а функція angle призначена для визначення кута між попередньою прямою (для можливості користувача змінювати цей кут).

Щоб мати можливість змінювати кількість розгалужень, створено додатково ще дві функції для побудови дерева Піфагора відповідно з трьома та чотирма розгалуженнями.

2.4.4. Побудова множини Мандельброта

Алгоритм побудови множини Мандельброта має наступний опис: розглянемо послідовність чисел, які отримаємо за допомогою наступного рівняння:

$$C_{n+1} = C_n^2 + c_0. \qquad (2.15)$$

Якщо така послідовність не виходить за межі двох комплексних чисел $C_1(1, 1i)$ і $C_2(-1, -1i)$, то дане комплексне число c_0 належить множині Мандельброта. Це значить, що множина Мандельброта – це множина всіх таких чисел c_0, при якій розглядана вище послідовність залишається в межах C_1 и C_2.

Програмна реалізація побудови множини Мандельброта виконана за допомогою мови програмування JavaScript. Множина Мандельброта існує між $C_1(1, 1i)$ і $C_2(-1, -1i)$, тому потрібна нам система координат буде мати центр в точці (0; 0), вісі абсцис та ординат будуть обмежені значеннями від -1.0 до 1.0 (рис. 2.10).

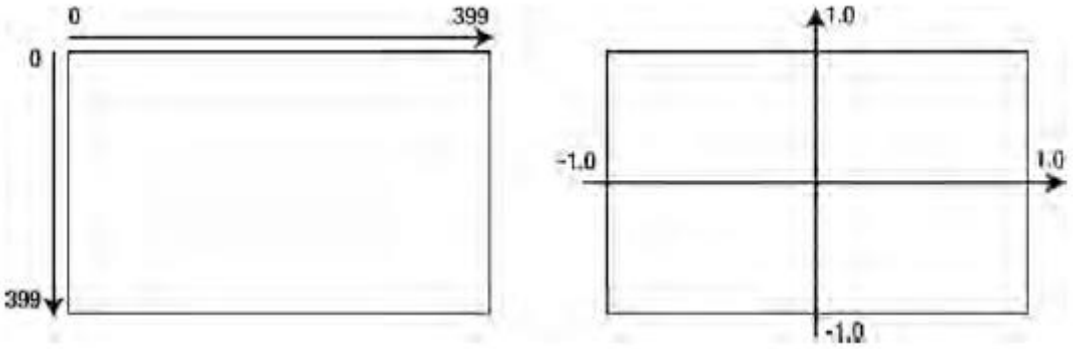

Рис. 2.10. Зображення системи множини Мандельброта

В програмі створено функцію, яка перевіряє належність точки комплексної площини до множини. В функції виконується цикл, збільшивши кількість ітерацій якого ми збільшимо точність результату (тож чим більша кількість ітерацій, тим менша кількість точок належить до множини). Якщо точка належить до множини – функція поверне true, інакше – false (рис. 2.11).

```
var realComponentOfResult = x;
var imaginaryComponentOfResult = y;

for(var i = 0; i < 10; i++) {
     var tempRealComponent = realComponentOfResult * realComponentOfResult
                           - imaginaryComponentOfResult * imaginaryComponentOfResult
                           + x;

     var tempImaginaryComponent = 2 * realComponentOfResult * imaginaryComponentOfResult
                           + y;

     realComponentOfResult = tempRealComponent;
     imaginaryComponentOfResult = tempImaginaryComponent;
}

if (realComponentOfResult * imaginaryComponentOfResult < 5)
    return true; // Належить Множині Мандельброта

return false; // Не належить Множині Мандельброта
```

Рис. 2.11. Програмна реалізація алгоритму побудови множини Мандельброта

Після виконання програми, яка використовує цю функцію, отримаємо наступне зображення (рис. 2.12).

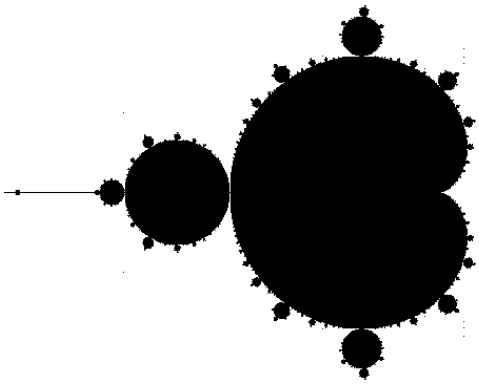

Рис. 2.12. Зображення побудованої множини Мандельброта

Модифікуємо цю множину додаванням кольору. Легше за все це зробити за допомогою схеми кольорів HSL. Щоб це реалізувати, змінимо значення, які повертає функція: замість булевого значення вона повинна повертати числове, яке буде залежати від кількості ітерацій, яка була потрібна для ідентифікації цієї точки. Тепер, якщо точка належить до множини, ми будемо повертати i/maxIterations, де i – кількість виконаних ітерацій, maxIterations – максимальна

кількість ітерацій, задана користувачем. Напишемо функцію, яка буде малювати точку з відповідним кольором. $Z_1 = Z^2 + C$ (рис. 2.13):

```
if(belongsToSet == 0) {
    ctx.fillStyle = '#000';
    ctx.fillRect(x,y, 1,1); // Малюємо чорну точку
} else {
    ctx.fillStyle = 'hsl(0, 100%, ' + belongsToSet + '%)';
    ctx.fillRect(x,y, 1,1); // Малюємо кольорову точку
}
```

Рис. 2.13. Фрагмент коду функції

При зміні процентів в схемі кольорів HSL буде змінюватись колір. В моїй програмі реалізовано чотири кольори побудови. Отримаємо наступне зображення (рис.2.14).

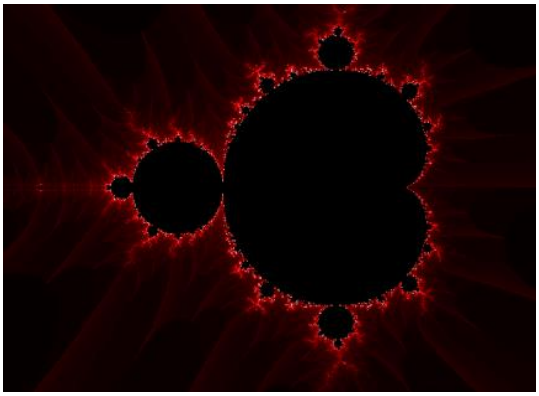

Рис. 2.14. Зміна кольору

2.4.5. Опис структури програми

Структура розробленого веб-додатку складається з наступних складових:

1. Стартова сторінка, яка містить в собі інформацію про ресурс та переходи до вирішення завдань.

2. Сторінка побудови кривої Коха.

3. Сторінка побудови папороті Барнслі.

4. Сторінка побудови дерева Піфагора.

5. Сторінка побудови множини Мандельброта.

Кожна із сторінок з вирішенням задач містить короткі теоретичні відомості про застосований метод та саму реалізацію метода за введеними користувачем даними.

Схема роботи програми зображено на рис. 2.15.

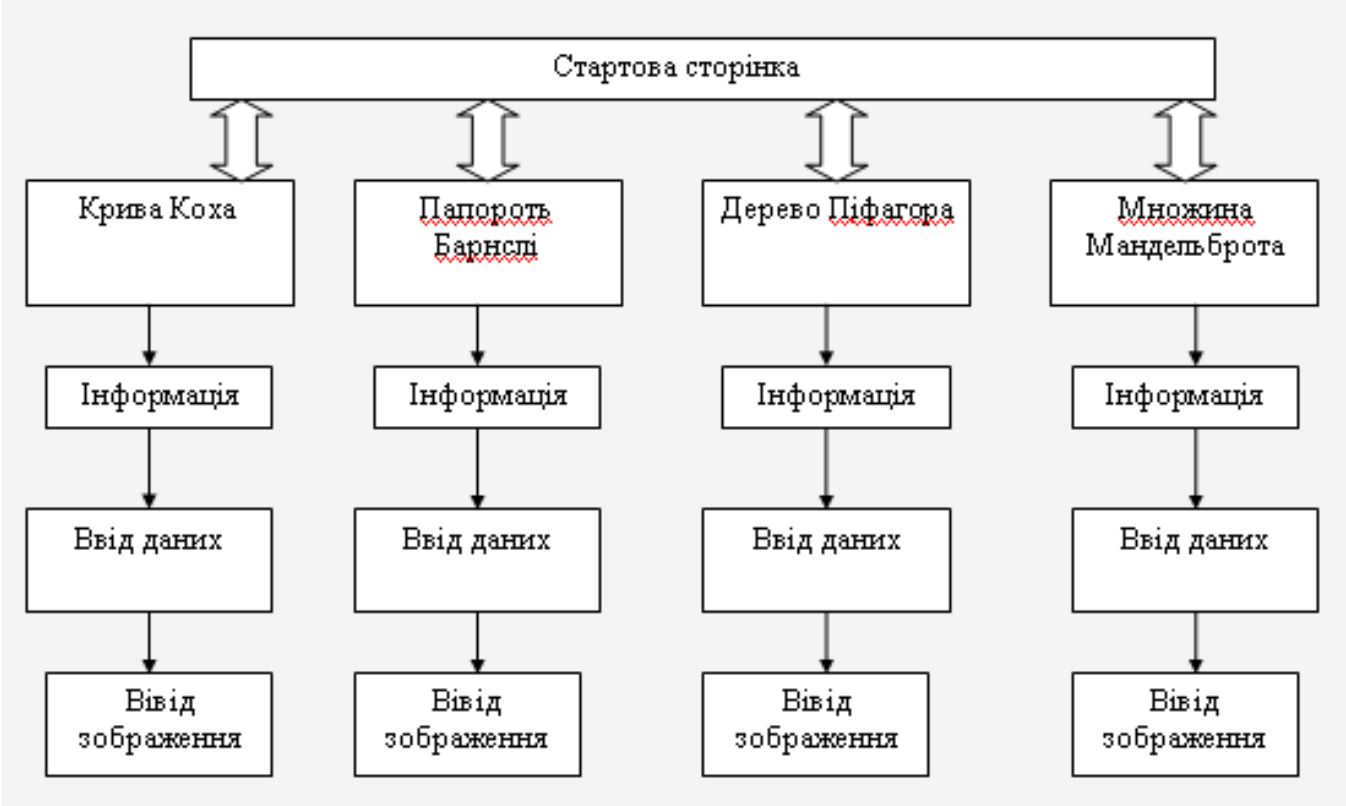

Рис. 2.15. Схема роботи сайту

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Вхідні дані в програмі вносяться користувачем вручну до відповідних полів, або використовуються дані, що встановлені розробником за замовчуванням. До вхідних даних відносять:

1. Для побудови кривої Коха: кількість ітерацій, колір побудови та тла.
2. Для побудови папороті Барнслі: кількість ітерацій, кількість та згорнутість листя, колір побудови та тла.

3. Для побудови дерева Піфагора: кількість ітерацій, кут нахилу, кількість розгалужень, колір побудови та тла.

4. Для побудови множини Мандельброта: кількість ітерацій, змінити масштаб, обрати колір побудови та тла.

Вихідними даними для всіх типів задач є зображення побудованого фракталу, що виводиться на поле.

2.6. Опис роботи розробленої системи

2.6.1. Використані технічні засоби

Для користування версією для Windows необхідно мати комп'ютер чи ноутбук з встановленою операційною системою Windows 7 або Windows 10, та наступні характеристики ПК:

- процесор: 1 ГГц і швидше з підтримкою PAE, NX і SSE2;
- RAM: 1 Гбайт (32 біт) або 2 Гбайт (64 біт);
- HDD: 16 Гбайт (32 біт) або 20 Гбайт (64 біт);
- відеокарта: підтримка Microsoft DirectX 9 з драйвером WDDM.

2.6.2. Використані програмні засоби

Додаток створений за допомогою HTML5/CSS3 та мови програмування JavaScript.

2.6.3. Виклик та завантаження програми

Для завантаження розробленої програми необхідно перейти на сторінку, розміщену в мережі Інтернет (http://fractal.pp.ua/site1/main_page/ index.php), після чого завантажиться стартова сторінка створеного web-додатку.

2.6.4. Опис інтерфейсу користувача

Перше, що користувач бачить при запуску сайту – головну сторінку. З неї за допомогою відповідного надпису можна перейти до відповідної частини сайту (рис. 2.16).

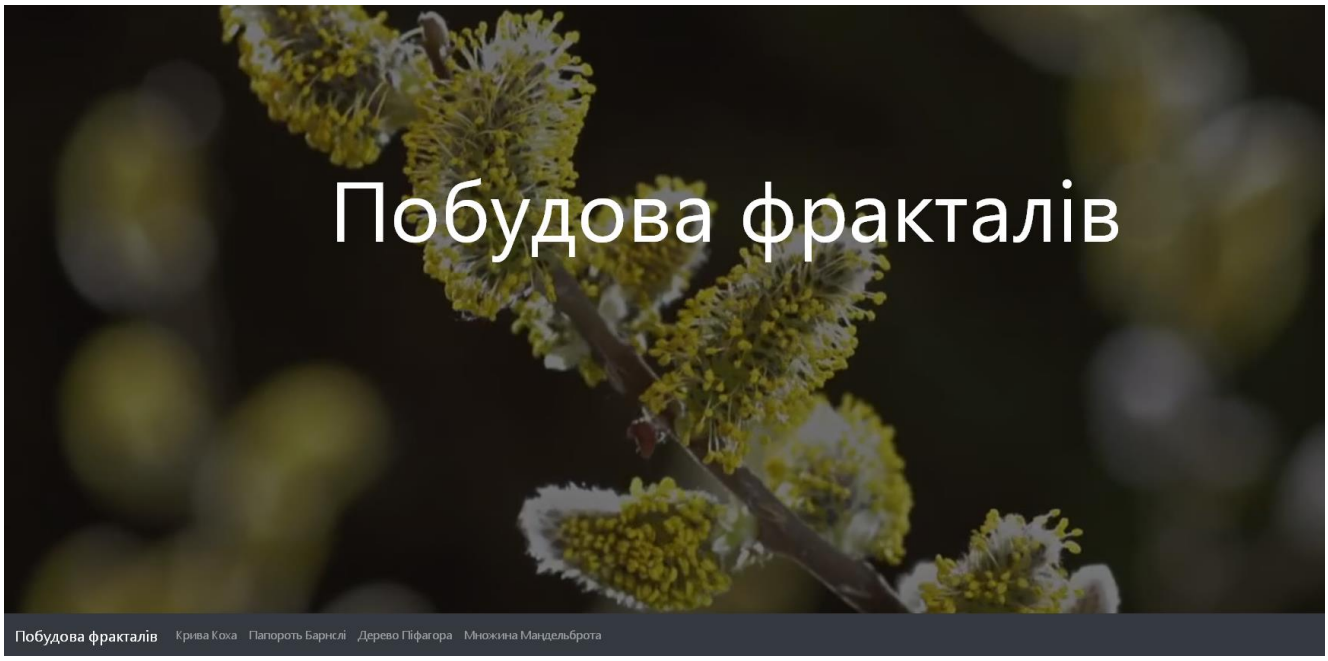

Рис. 2.16. Стартова сторінка додатку

Стартова сторінка інформує користувача про суть додатку, внизу сторінки розташовані блоки з інформацією щодо певних фракталів та посиланнями на сторінки з їх побудовою: кривої Коха, папороті Барнслі, дерева Піфагора та множини Мандельброта (рис. 2.17-2.20).

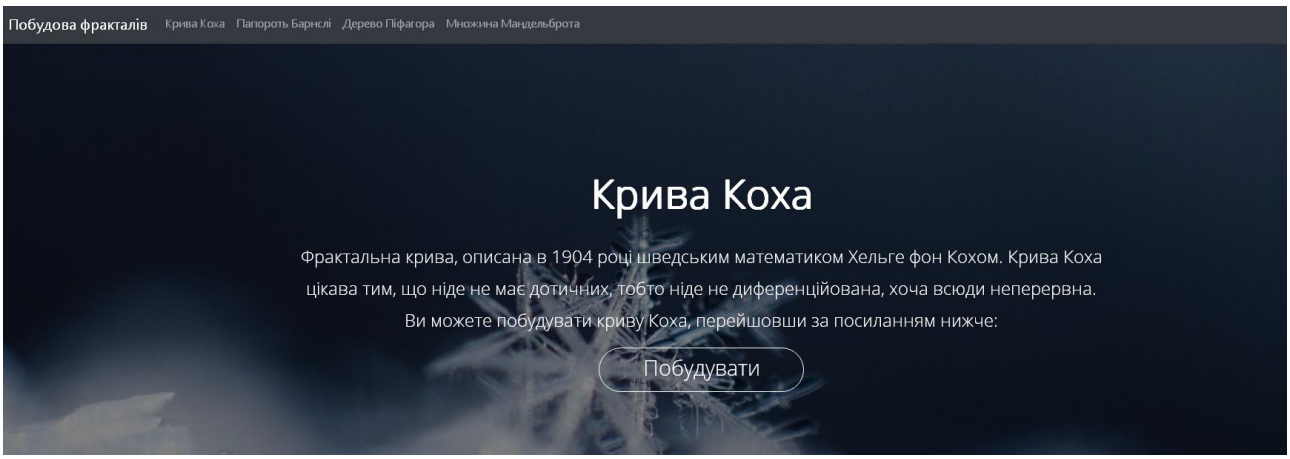

Рис. 2.17. Інформація про сторінку «Крива Коха»

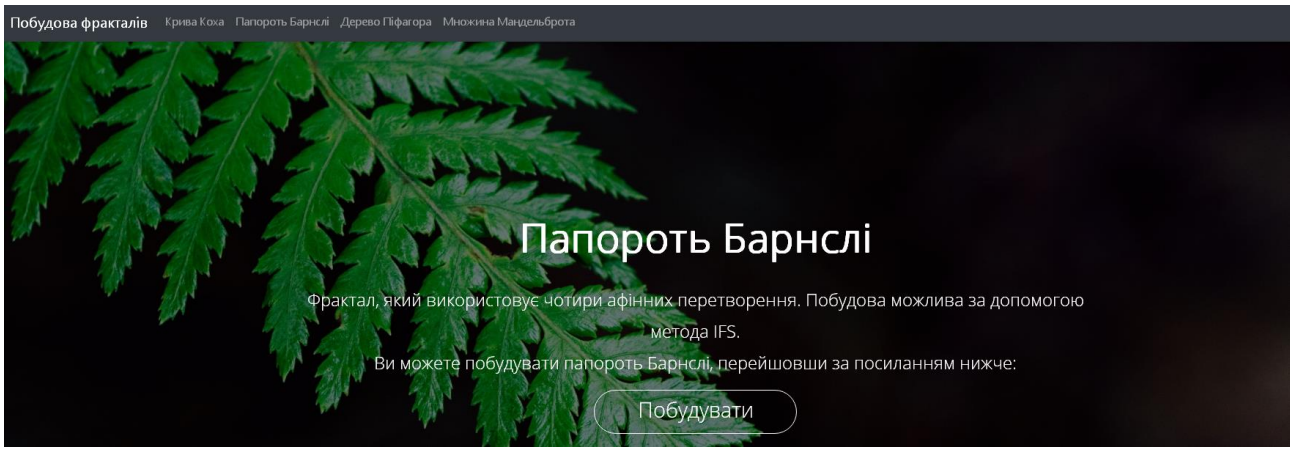

Рис. 2.18. Інформація про сторінку «Папороть Барнслі»

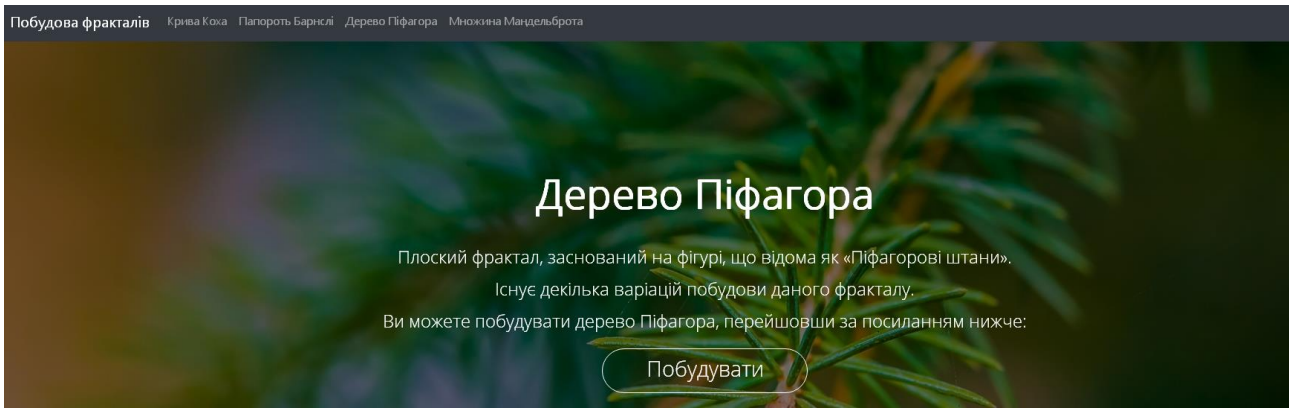

Рис. 2.19. Інформація про сторінку «Дерево Піфагора»

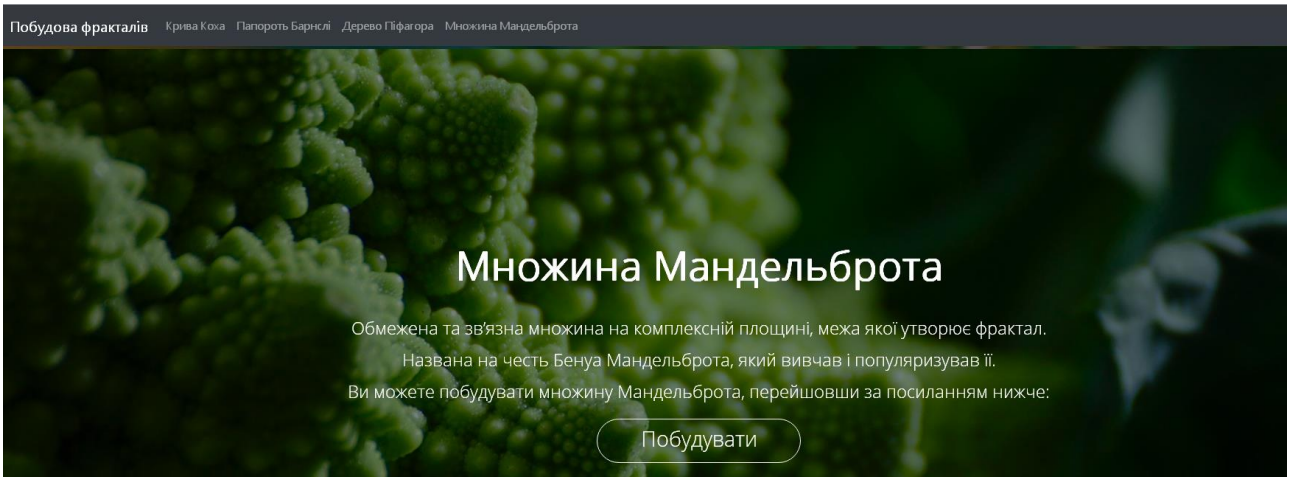

Рис. 2.20. Інформація про сторінку «Множина Мандельброта»

Сторінка “Побудова кривої Коха” (рис. 2.21).

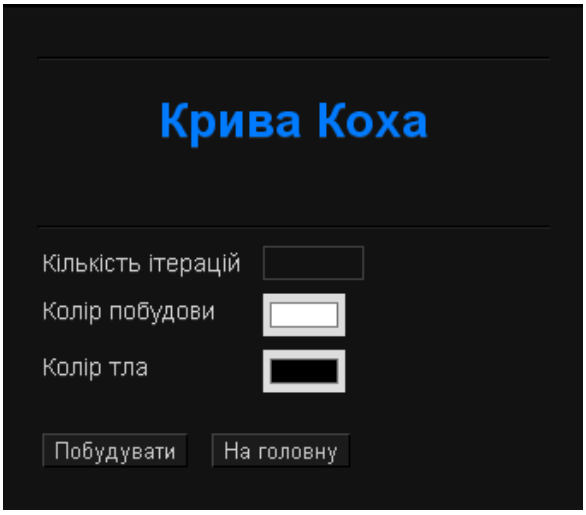

Рис. 2.21. Сторінка «Крива Коха»

На цій сторінці потрібно задати параметри побудови за допомогою форми введення параметрів. Користувач може ввести кількість ітерацій, обрати колір побудови та тла. Після натискання кнопки «Побудувати» побудується фрактал з даними параметрами.

На рис. 2.22. побудовано фрактал при кількості ітерацій 7 (значенням за замовчуванням), на рис. 2.23 кількість ітерацій дорівнює 4 та змінено колір побудови.

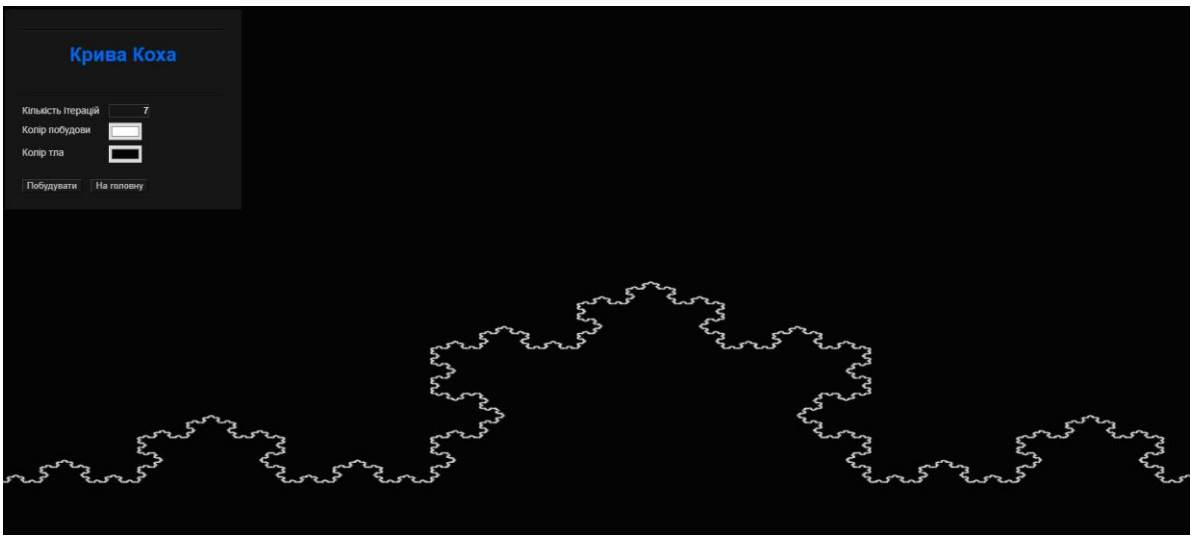

Рис. 2.22. Результат побудови кривої Коха з кількістю ітерацій 7

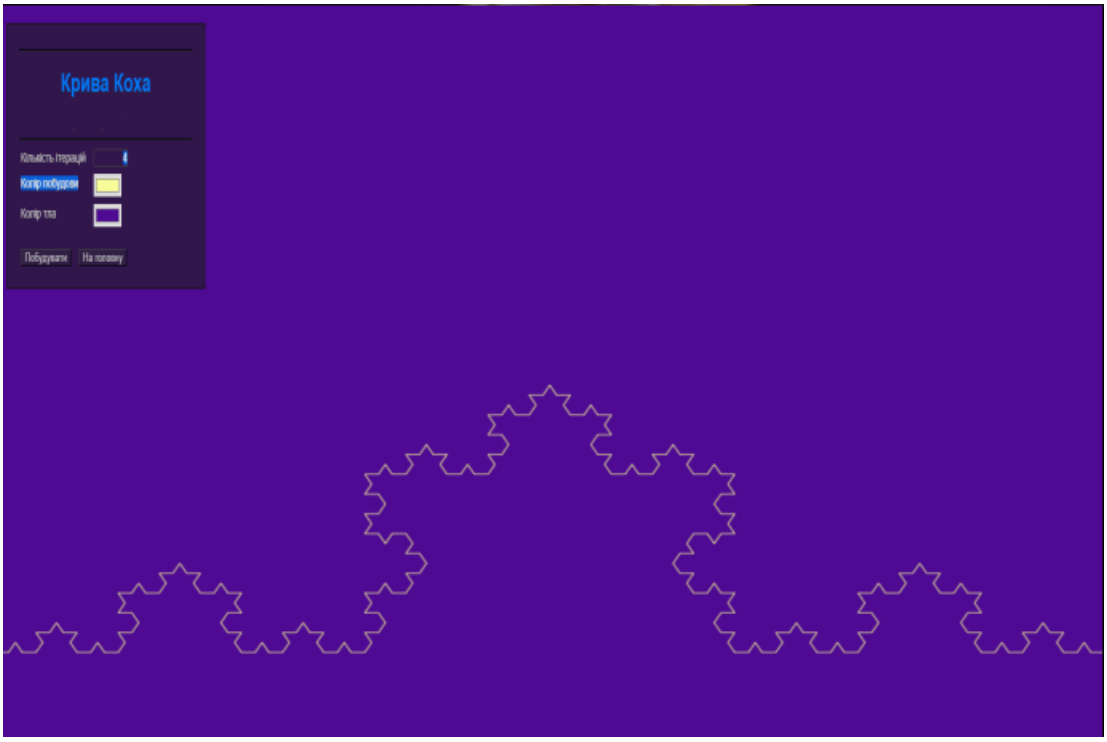

Рис. 2.23. Результат побудови кривої Коха з кількістю ітерацій 4 та обраними кольорами

Сторінка «Папороть Барнслі» (рис. 2.24) має аналогічну з попереднім фракталом форму введення параметрів побудови.

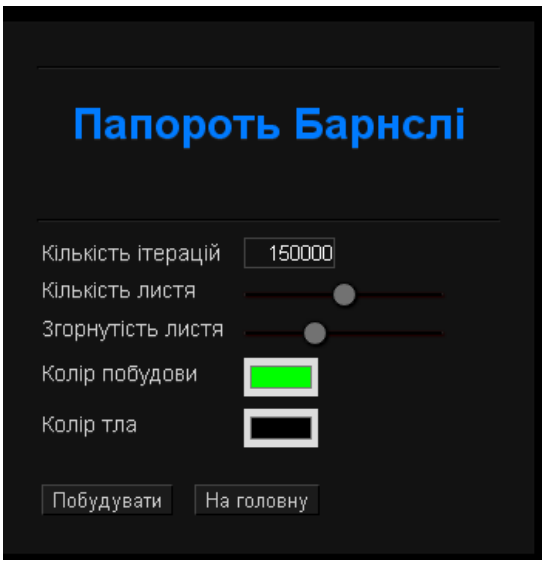

Рис. 2.24. Сторінка «Папороть Барнслі»

Користувач може ввести у відповідні поля кількість ітерацій, змінити кількість та згорнутість листя, обрати колір побудови та тла. На рис. 2.25 побудовано даний фрактал при кількості ітерацій 150000 (значення за замовчуванням), кількість та згорнутість листя за замовчуванням.

Рис. 2.25. Результат побудови папороті Барнслі при кількості ітерацій 150000

На рис. 2.26 побудований фрактал, кількість ітерацій якого зменшена до 5000. Ми можемо спостерігати, що якість зображення залежить від кількості побудованих точок.

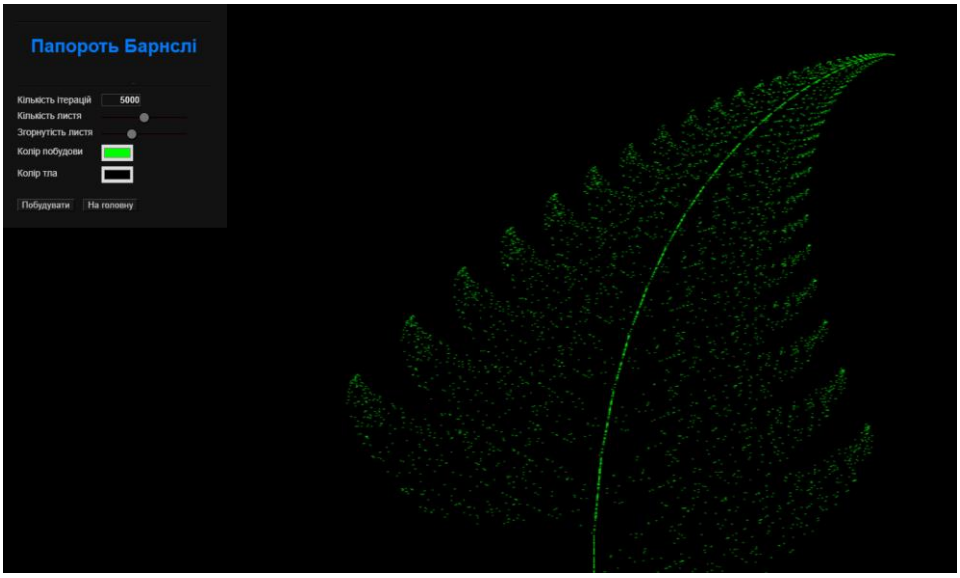

Рис. 2.26. Результат побудови папороті Барнслі при кількості ітерацій 5000

На рис. 2.27 побудований фрактал, кількість ітерацій якого дорівнює 150000 (значення за замовчуванням), зменшена кількість листя.

Рис. 2.27. Результат побудови папороті Барнслі при кількості ітерацій 150000 та зменшеної кількості листя

На рис. 2.28 залишимо кількість ітерацій та повернемо кількість листя до значення за замовчуванням, збільшимо згорнутість листя.

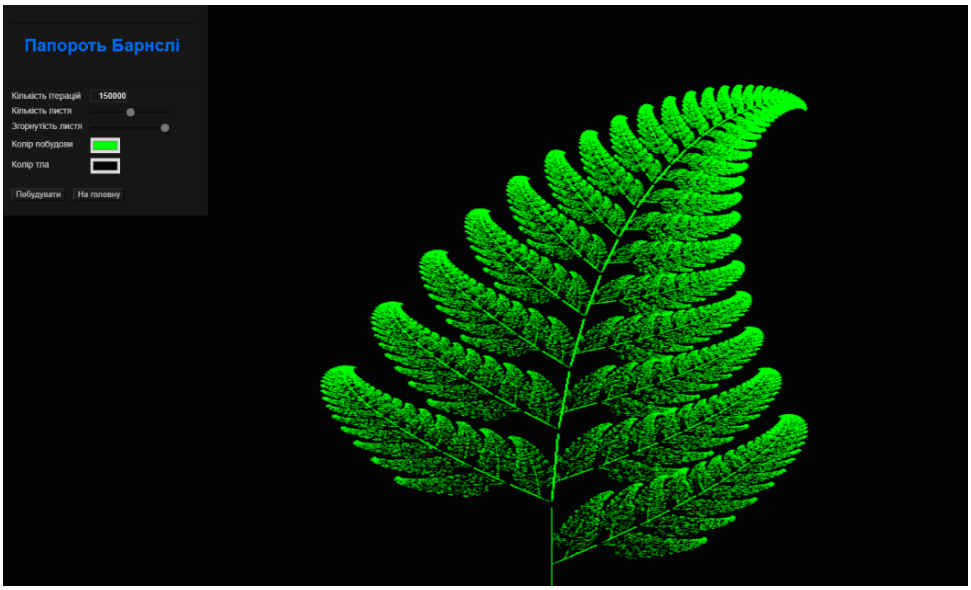

Рис. 2.28. Результат побудови папороті Барнслі при кількості ітерацій 150000, кількість листя за замовчуванням а згорнутість збільшено

Таким чином, змінюючи значення констант в таблиці, можна отримувати безліч різноманітних моделей, які відрізняються від стандартної папороті Барнслі.

Сторінка “Дерево Піфагора” (рис. 2.29).

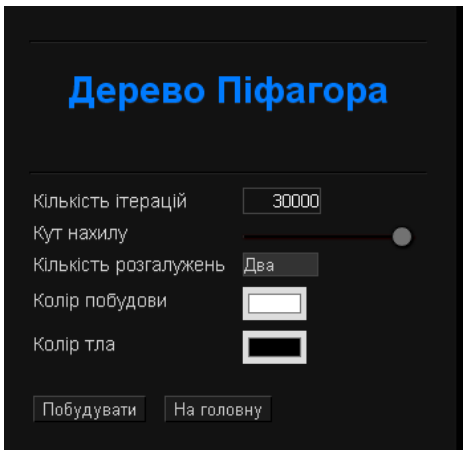

Рис. 2.29. Сторінка «Дерево Піфагора»

Після переходу на сторінку побудови дерева Піфагора ми бачимо

аналогічну з попереднім фракталом форму введення параметрів побудови. Користувач може ввести кількість ітерацій, обрати кут нахилу, кількість розгалужень, колір побудови та тла. На рис. 2.30. побудований фрактал кількість ітерацій якого дорівнює 30000 (значення за замовчуванням).

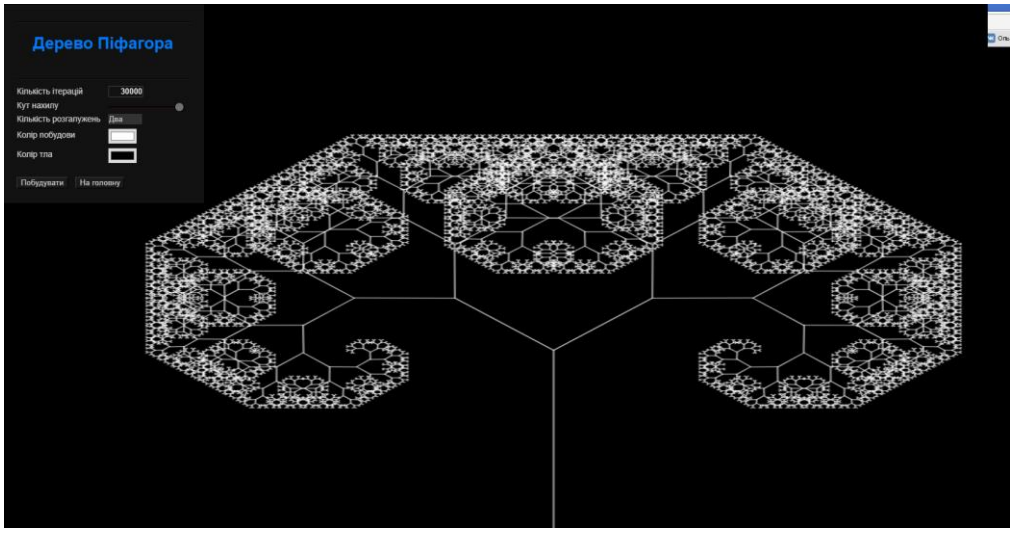

Рис. 2.30. Результат побудови дерева Піфагора при кількості ітерацій 30000

На рис. 2.31 побудований фрактал при 17000 ітераціях, кількість розгалужень збільшена до 3-ох, змінений кут нахилу.

Рис. 2.31. Результат побудови дерева Піфагора при кількості ітерацій 17000, кількість розгалужень 3 та зміненому куті нахилу

Ми можемо спостерігати наслідки обраних параметрів: кількість розгалужень збільшилась з двох до трьох, змінився кут нахилу гілок. Недостатня кількість ітерацій привела до неповної побудови дерева. Таким чином, за допомогою моєї програми ми можемо досліджувати модель реального дерева.

Сторінка «Множина Мандельброта» (рис. 2.32).

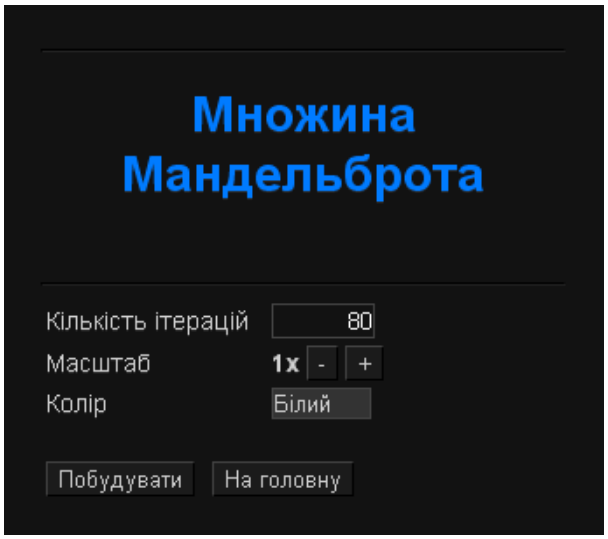

Рис. 2.32. Сторінка «Множина Мандельброта»

Після переходу на сторінку побудови множини Мандельброта ми бачимо аналогічну з попереднім фракталом форму введення параметрів побудови. Користувач може ввести кількість ітерацій, змінити масштаб, обрати колір побудови. На рис. 2.33 побудований даний фрактал з кількістю ітерацій 80 (значення за замовчуванням).

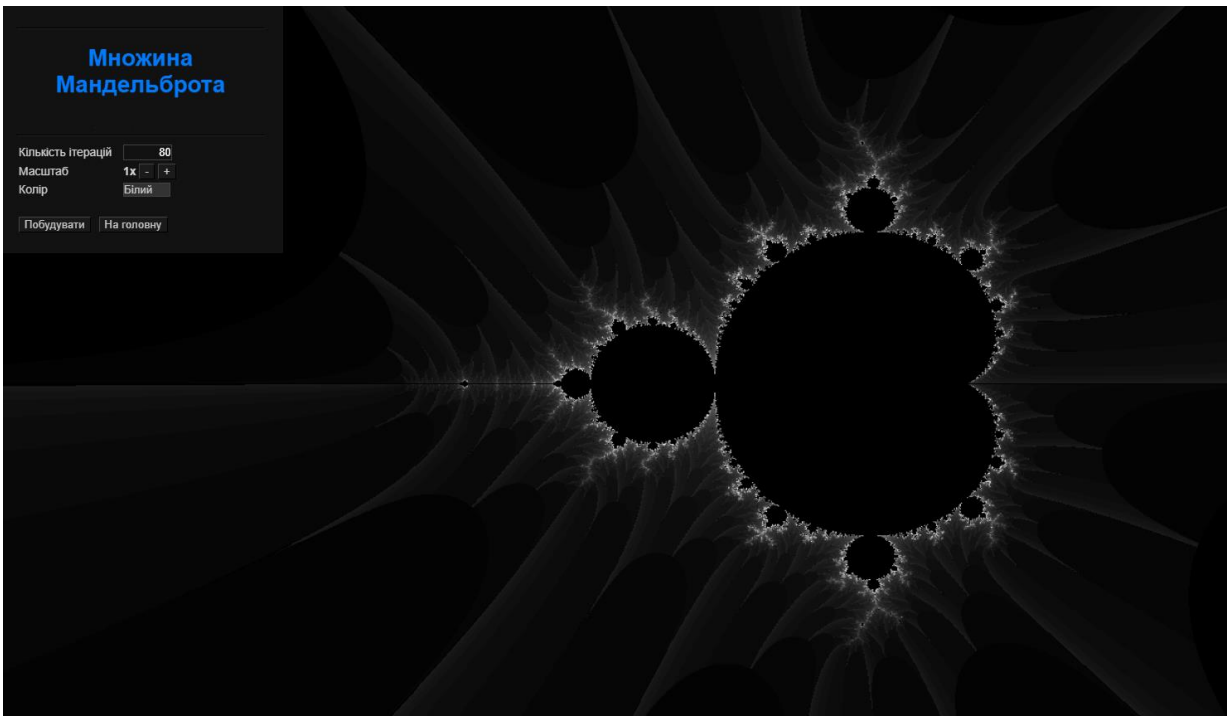

Рис. 2.33. Результат побудови фракталу множини Мандельброта з кількістю ітерацій 80

На рис. 2.34 побудований фрактал з кількістю ітерацій 650, збільшеним масштабом до 7x та зміненим кольором побудови.

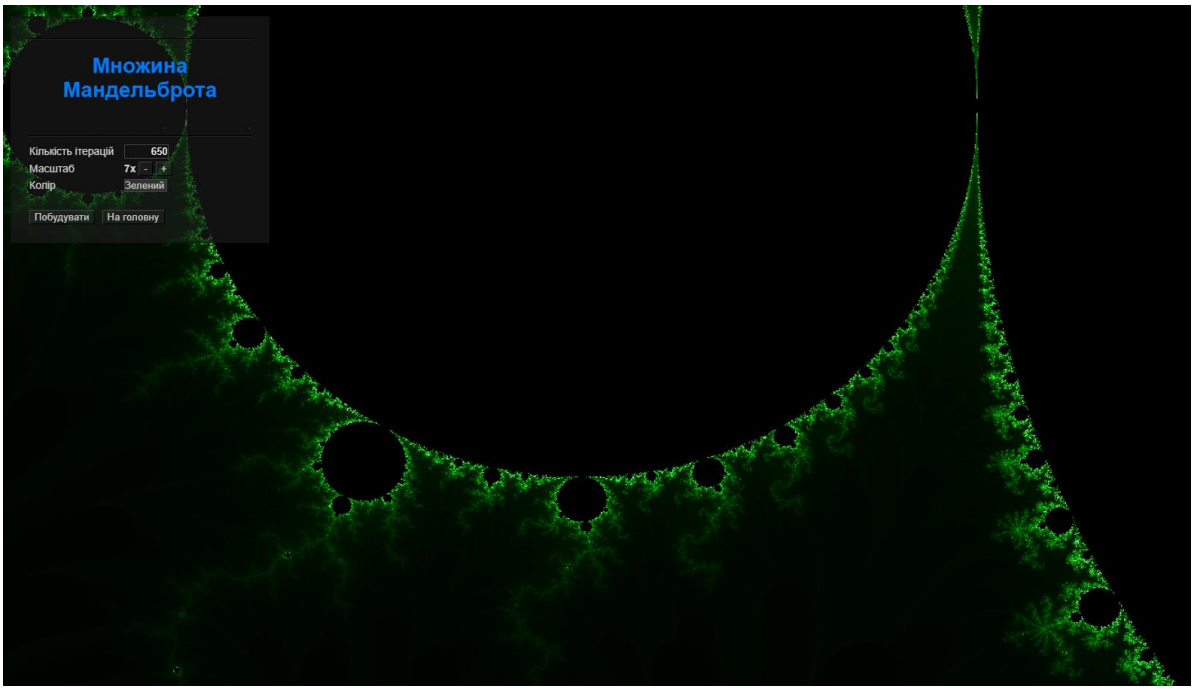

Рис. 2.34. Результат побудови фракталу множини Мандельброта з кількістю ітерацій 650, масштаб 7 та зміненим кольором побудови

На рис. 2.35 кількість ітерацій даного фракталу зменшена до 10, масштаб дорівнює 4х.

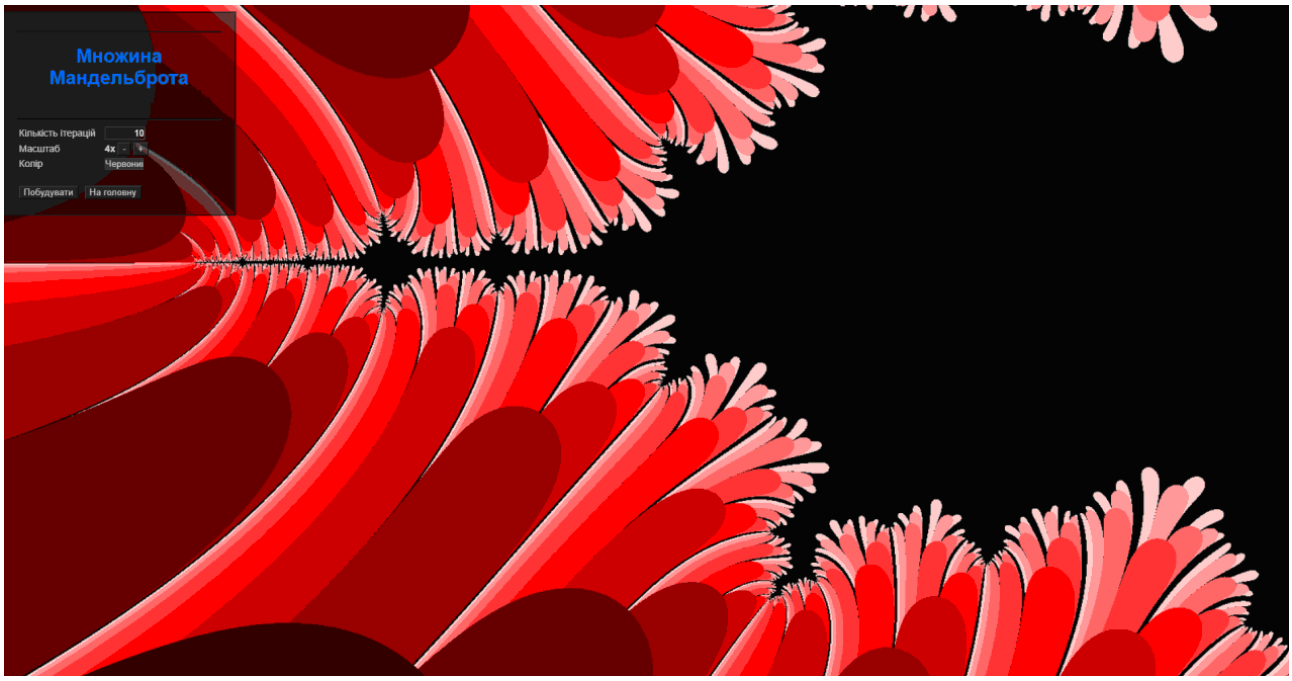

Рис. 2.35. Результат побудови фракталу множини Мандельброта з кількістю ітерацій 10 та масштабом 4

РОЗДІЛ 3
ЕКОНОМІКА

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Вхідні дані:

- передбачуване число операторів - 1200;
- коефіцієнт складності програми - 1,7;
- коефіцієнт корекції програми в ході її розробки - 0,1;
- годинна заробітна плата програміста, грн / год - 100.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де to - витрати праці на підготовку й опис поставленої задачі (приймається 50);

ти - витрати праці на дослідження алгоритму рішення задачі;

та - витрати праці на розробку блок-схеми алгоритму;

тп - витрати праці на програмування по готовій блок-схемі;

тотл - витрати праці на налагодження програми на ЕОМ;

тд - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \times C \times (1+ p) \text{ , людино-годин,} \qquad (3.2)$$

де q - передбачуване число операторів;

С - коефіцієнт складності програми;

р - коефіцієнт кореляції програми в ході її розробки.

$$Q = 1200 \cdot 1{,}7 \cdot (1 + 0{,}1) = 2244 \text{ людино-годин.}$$

Витрати праці на вивчення опису задачі tи визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{QB}{(75...85)K} \text{ , людино-годин,} \qquad (3.3)$$

де В - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі; В=1.2 ... 1.5;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Витрати праці на розробку алгоритму рішення задачі:

$$t_u = \frac{2244 \cdot 1{,}2}{80 \cdot 0{,}8} = 42 \text{ , людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t\alpha = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \qquad (3.4)$$

$$t_a = \frac{2244}{20 \cdot 0{,}8} = 140 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \quad \text{людино-годин.} \tag{3.5}$$

$$t_n = \frac{2244}{25 \cdot 0{,}8} = 112 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4...5)K} \quad \text{людино-годин.} \tag{3.6}$$

$$t_{отл} = \frac{2244}{4 \cdot 0{,}8} = 701 \text{ людино-годин.}$$

За умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1{,}2 \cdot t_{отл}; \tag{3.7}$$

$$t_{отл} = 701 * 1{,}2 = 841{,}5$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{ людино-годин,} \tag{3.8}$$

де tдр - трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)K}, \text{ людино-годин.} \tag{3.9}$$

$$t_{др} = \frac{2244}{15 \cdot 0{,}8} = 187 \text{ людино-годин,}$$

tдо - трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75 \cdot t_{др}, \text{ людино-годин.} \quad (3.10)$$

$$tdo = 0{,}75 \cdot 187 = 140$$

$$t_{до} = 140 + 187 = 327 \text{ людино-годин.}$$

Отримуємо трудомісткість розробки програмного забезпечення:

$$t = 50 + 42 + 140 + 112 + 841 + 327 = 1513 \text{ людино-годин.}$$

3.2. Витрати на створення програмного забезпечення

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,} \quad (3.12)$$

де t - загальна трудомісткість, людино-годин;

Спр - середня годинна заробітна плата програміста, грн/година.

$$Ç_{Çï} = 1513 \cdot 100 = 151300 \text{ ãðí.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \times С_{М}, \text{ грн,} \quad (3.13)$$

де tотл - трудомісткість налагодження програми на ЕОМ, год.

Смч - вартість машино-години ЕОМ, 5 грн/год.

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$З_{МВ} = 841 \times 5 = 4207 \text{ грн.}$$

$$Êïï = 4207 + 151300 = 195507 \text{ ãðí.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p} \text{ міс.} \qquad (3.14)$$

де Bk - число виконавців;

Fp - місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = \frac{1531}{1 \cdot 176} = 8{,}6 \text{ міс.}$$

Визначено трудомісткість розробленої інформаційної системи (1513 люд-год), проведений підрахунок вартості роботи по створенню програми (195507 грн.) та розраховано час на його створення (8,6 міс).

ВИСНОВКИ

На даний час теорія фракталів знаходить широке застосування в різних областях практичної діяльності.

Використання фрактальних моделей дозволяє знаходити нові підходи у вирішенні завдань обробки цифрової інформації, вивченні питань турбулентного руху рідин, радіолокації, дослідження фінансових ринків, отримання нових наноматеріалів з заданими властивостями тощо.

Метою кваліфікаційної роботи є дослідження способів моделювання фракталів та розроблення програмного засобу для побудови фрактальних зображень.

Розроблений програмний додаток призначений для вирішення наступних задач:

1. Аналіз математичних моделей фрактальних множин.
2. Розробка алгоритмів моделювання фрактальних структур, оцінка характеристик алгоритмів, а також властивостей множин, породжуваних запропонованими алгоритмами.
3. Розробка програмного комплексу для проведення обчислювального експерименту та дослідження структур даних на основі фрактальних моделей, отриманих з використанням запропонованих алгоритмів.
4. Тестування розробленого програмного забезпечення при різних значеннях параметрів моделей.

Під час виконання даної роботи були досліджені закономірності у побудові фрактальних зображень при зміні параметрів фрактальної моделі та проаналізовані наступні алгоритми побудови фракталів:

1. Ітеративний.
2. Рекурсивний.
3. За допомогою метода IFS.

Додаток створений за допомогою HTML5/CSS3 та мови програмування JavaScript.

В результаті виконання роботи, створено веб-додаток, за допомогою якого можна не тільки побудувати деякі фрактали, але й дослідити їх побудову за допомогою зміни параметрів побудови. Таким чином, змінюючи параметри побудови можна отримувати безліч фрактальних зображень.

Веб-сервіс призначений для побудови фрактальних структур та проведення обчислювального експерименту та дослідження змін у структурах, що будуються при різних значеннях параметрів фрактальних моделей. WEB-сервіс дозволяє побудувати такі фрактали: крива Коха, дерево Піфагора, папороть Барнслі та множина Мандельброта.

Працездатність даного програмного продукту підтверджується вдалими експлуатаційними випробуваннями.

В економічному розділі визначено трудомісткість розробленої інформаційної системи (1513 люд-год), проведений підрахунок вартості роботи по створенню програми (195507 грн.) та розраховано час на його створення (8,6 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Алгоритм побудови множини Мандельброта. Algolist.ru/ URL. – Режим доступу: http://algolist.manual.ru/graphics/mandelbrot.php/. дата звернення: 21.05.2021.
2. Божокин С.В., Паршин Д.А. Фракталы и мультифракталы. – Ижевск: НИЦ «Регулярная и хаотическая динамика», 2001 г., – 128 с.
3. Гринченко В. Т., Мацыпура В. Т., Снарский А. А. Введение в нелинейную динамику: Хаос и фракталы. - М.: URSS, 2010. - 280 с.
4. Гринченко В. Т., Мацыпура В. -., Снарский А. А. Фракталы: от удивления к рабочему инструменту. - К.: Наукова думка, 2013. - 270 с.
5. Детальніше про Криву Коха. Elementy.ru – URL: https://elementy.ru/posters/ fractals/Koch/ . дата звернення: 21.05.2021.
6. Дерево Піфагора. Wikipedia – URL: https://uk.wikipedia.org/wiki/ Дерево_Піфагора. дата звернення: 21.05.2021.
7. Комп’ютерна графіка. Векторна та фрактальна графіка URL: http://www.victoria.lviv.ua/html/oit/html/lesson12_II.htm дата звернення: 21.05.2021.
8. Комплексні числа. Wikipedia – URL: https://uk.wikipedia.org/wiki/ Комплексне_число/. дата звернення: 21.05.2021.
9. Кроновер Р. М. Фракталы и хаос в динамических системах. - М.: Техносфера, 2006. - 488 с.
10. Крива Коха. Wikipedia – URL: https://uk.wikipedia.org/wiki/ Крива_Коха. дата звернення: 21.05.2021.
11. Ландэ Д. В. Фракталы и кластеры в информационном пространстве // Корпоративные системы (2005) (6) С. 35-39.
12. Мандельброт Б. Фрактальная геометрия природы. - Ижевск: ИКИ, 2010. - 656 с.
13. Мандельброт Б. Фракталы и хаос. - Ижевск: РХД, 2009. - 400 с.

14. Мандельброт Б. Фракталы, случай и финансы. - Ижевск: РХД, 2004. - 256 с.

15. Методи перетворення графічних зображень URL: http://inter.ptngu.com/головна/оксм/компютерна-графіка/методи-перетворення-графічних-зобр дата звернення: 21.05.2021.

16. Множина Мандельброта. Wikipedia – URL: https://uk.wikipedia.org/ wiki/ Множина_Мандельброта/. дата звернення: 21.05.2021.

17. Морозов А.Д. Введение в теорию фракталов //2- е изд., 2002 г.– 163стр.

18. Пайтген Х.-О., Рихтер П. Х. Красота фракталов. - М.: Мир, 1993. - 176 с.

19. Площина комплексних чисел. Wikipedia – URL: https://uk.wikipedia.org/wiki/ -_площина. дата звернення: 21.05.2021.

20. Поняття фрактала та історія появи фрактальної графіки //Физика, математика ТОЕ. Лекции, задачи, учебники // URL:http:// Программы для генерации фракталов. URL: http://m-rush.ru/praktik /programmy.html дата звернення: 21.05.2021.

21. Ричард М. Кроновер «Фракталы и хаос в динамических системах». – М., Постмаркет, 2000 г., – 352 с.

22. Федер Е. Фракталы. - М.: Мир, 1991. - 254 с.

23. Фрактальна графіка URL:http://lib.lntu.info/book/knit/auvp/2012/12-14/page15.html дата звернення: 21.05.2021.

24. Шевельова М. К., Карплюк С. О., Вакалюк Т. А. Інструментальні засоби створення фрактальних зображень // Науковий пошук молодих дослідників. – Вип. 6. – Житомир: ЖДУ, 2013. – С. 35-38.

25. Шредер М. Фракталы, хаос, степенные законы. - Ижевск: РХД, 2005. - 528 с.

26. Edyta Patrzalek, Fractals: Useful Beauty (General Introduction to Fractal Geometry) // Stan Ackermans Institute, IPO, Centre for User-System Interaction, Eindhoven University of Technology, 2000. – 6 p.

27. Kumar S., Public key cryptography system using Mandelbrot sets // Military Communications Conference, 2006.

28. Mandelbrot B.B. The Fractal Geometry Of Nature // San Francisco 1982. – 656 p.

29. Windows Dev Center Home: «Обзор интегрированной среды разработки» URL: http://msdn.microsoft.com/ruru/library/windows/apps/xaml/ms165088(v=vs.90).aspx дата звернення: 21.05.2021.

ДОДАТОК А

КОД ПРОГРАМИ

```
/*!
 * Bootstrap v4.0.0-beta (https://getbootstrap.com)
 * Copyright 2011-2021 The Bootstrap Authors
(https://github.com/twbs/bootstrap/graphs/contributors)
 * Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)
 */
If ("undefined"==typeof jQuery) throw new Error
("Bootstrap's JavaScript requires jQuery. jQuery must be included before Bootstrap's JavaScript.");
!function(t)
{var e=jQuery.fn.jquery.split(" ")[0].split(".");
if(e[0]<2&&e[1]<9||1==e[0]&&9==e[1]&&e[2]<1||e[0]>=4)
throw new Error("Bootstrap's JavaScript requires at least jQuery v1.9.1 but less than v4.0.0")}(),
function()
{function t(t,e)
{if(!t)throw new ReferenceError("this hasn't been initialised - super() hasn't been called");
return!e||
"object"!=typeof e&&"function"!=typeof e?t:e}
function e(t,e)
{if("function"!=typeof e&&null!==e)throw new TypeError("Super expression must either be null or
a function, not "+typeof e);
t.prototype=Object.create(e&&e.prototype,
{constructor:
{value:t,enumerable:!1,writable:!0,configurable:!0}}),
e&&(Object.setPrototypeOf?Object.setPrototypeOf(t,e):t.__proto__=e)}
function n(t,e)
{if(!(t instanceof e))throw new TypeError("Cannot call a class as a function")}
var i="function"==typeof Symbol&&"symbol"==typeof Symbol.iterator?function(t){return typeof
t}:
function(t){return t&&"function"==typeof
Symbol&&t.constructor===Symbol&&t!==Symbol.prototype?"symbol":typeof t},
o=function()
```

```
{function t(t,e)
{for(var n=0;n<e.length;n++)
{var i=e[n];
i.enumerable=i.enumerable||!1,
i.configurable=!0,"value"in i&&(i.writable=!0),
Object.defineProperty(t,i.key,i)}}
return
function(e,n,i)
{return n&&t(e.prototype,n),i&&t(e,i),e}}
(),
r=function(t)
{function e(t){return{}.toString.call(t).match(/\s([a-zA-Z]+)/)[1].toLowerCase()}
function n(t){return(t[0]||t).nodeType
}function i()
{return{bindType:s.end,delegateType:s.end,handle:function(e)
{if(t(e.target).is(this))return e.handleObj.handler.apply(this,arguments)}}}
function o()
{if(window.QUnit)return!1;
var t=document.createElement("bootstrap");
for(var e in a)
if(void 0!==t.style[e])
return{end:a[e]};
return!1}
function r(e)
{var n=this,i=!1;
return t(this).one(l.TRANSITION_END,
function()
{i=!0}),
setTimeout(function()
{i||l.triggerTransitionEnd(n)},e),this}
var s=!1,
a={WebkitTransition:"webkitTransitionEnd",
MozTransition:"transitionend",
OTransition:"oTransitionEnd otransitionend",
transition:"transitionend"},
```

```
l={TRANSITION_END:"bsTransitionEnd",
getUID:function(t)
{do{t+=~~(1e6*Math.random())}
while(document.getElementById(t));
return t},
getSelectorFromElement
:function(e)
{var n=e.getAttribute("data-target");
n&&"#"!==n||(n=e.getAttribute("href")||"");
try
{return t(n).length>0?n:null}
catch(t)
{return null}},reflow:function(t)
{return t.offsetHeight},triggerTransitionEnd:function(e)
{t(e).trigger(s.end)},
supportsTransitionEnd:function()
{return Boolean(s)},
typeCheckConfig:function(t,i,o)
{for(var r in o)if(o.hasOwnProperty(r))
{var s=o[r],a=i[r],l=a&&n(a)?"element":e(a);
if(!new RegExp(s).test(l))throw new Error(t.toUpperCase()+': Option "'+r+'" provided type "'+l+'"
but expected type "'+s+'".')}}};
return s=o(),t.fn.emulateTransitionEnd=r,
l.supportsTransitionEnd()&&(t.event.special[l.TRANSITION_END]=i()),l}(jQuery),s=(function(t)
{var e="alert",
i=t.fn[e],
s={DISMISS:'[data-dismiss="alert"]'},
a={CLOSE:"close.bs.alert",CLOSED:"closed.bs.alert",CLICK_DATA_API:"click.bs.alert.data-
api"},
l={ALERT:"alert",FADE:"fade",SHOW:"show"},h=function()
{function e(t){n(this,e),this._element=t}
return e.prototype.close=function(t){t=t||this._element;
var e=this._getRootElement(t);
this._triggerCloseEvent(e).isDefaultPrevented()||
this._removeElement(e)},e.prototype.dispose=function()
```

```
{t.removeData(this._element,"bs.alert"),this._element=null},
e.prototype._getRootElement=function(e)
{var n=r.getSelectorFromElement(e),i=!1;
return n&&(i=t(n)[0]),i||(i=t(e).closest("."+l.ALERT)[0]),i},
e.prototype._triggerCloseEvent=function(e){var n=t.Event(a.CLOSE);
return t(e).trigger(n),n},
e.prototype._removeElement=function(e)
{var n=this;t(e).removeClass(l.SHOW),
r.supportsTransitionEnd()&&t(e).hasClass(l.FADE)?
t(e).one(r.TRANSITION_END,function(t){return
n._destroyElement(e,t)}).emulateTransitionEnd(150):this._destroyElement(e)}
,e.prototype._destroyElement=function(e){t(e).detach().trigger(a.CLOSED).remove()},
e._jQueryInterface=function(n){return this.each(function(){var i=t(this),o=i.data("bs.alert");
o||(o=new e(this),
i.data("bs.alert",o)),"close"===n&&o[n](this)})},
e._handleDismiss=function(t)
{
return function(e){e&&e.preventDefault(),t.close(this)}},
o(e,null,[{key:"VERSION",get:function(){return"4.0.0-beta"}}]),
e}();
t(document).on(a.CLICK_DATA_API,s.DISMISS,
h._handleDismiss(new h)),
t.fn[e]=h._jQueryInterface,t.fn[e].Constructor=h,t.fn[e].noConflict=function()
{return t.fn[e]=i,h._jQueryInterface}}
(jQuery),function(t)
{var e="button",
i=t.fn[e],
r={ACTIVE:"active",BUTTON:"btn",FOCUS:"focus"},
s={DATA_TOGGLE_CARROT:'[data-toggle^="button"]',
DATA_TOGGLE:'[data-toggle="buttons"]',
INPUT:"input",ACTIVE:".active",BUTTON:".btn"},
a={CLICK_DATA_API:"click.bs.button.data-api",
FOCUS_BLUR_DATA_API:"focus.bs.button.data-api blur.bs.button.data-api"},
l=function(){function e(t){n(this,e),this._element=t}
return e.prototype.toggle=function()
```

```
{var e=!0,n=!0,i=t(this._element).closest(s.DATA_TOGGLE)[0];
if(i)
{var o=t(this._element).find(s.INPUT)[0];
if(o)
{if("radio"===o.type)
if(o.checked&&t(this._element).hasClass(r.ACTIVE))e=!1;
else
{var a=t(i).find(s.ACTIVE)[0];
a&&t(a).removeClass(r.ACTIVE)}if(e)
{if(o.hasAttribute("disabled")||i.hasAttribute("disabled")
||o.classList.contains("disabled")||i.classList.contains("disabled"))
return;
o.checked=!t(this._element).hasClass(r.ACTIVE),
t(o).trigger("change")}o.focus(),n=!1}}
n&&this._element.setAttribute("aria-pressed",
!t(this._element).hasClass(r.ACTIVE)),
e&&t(this._element).toggleClass(r.ACTIVE)},
e.prototype.dispose=function()
{t.removeData(this._element,"bs.button"),this._element=null},
e._jQueryInterface=function(n)
{return this.each(function()
{var i=t(this).data("bs.button");
i||(i=new e(this),t(this).data("bs.button",i)),
"toggle"===n&&i[n]()})},
o(e,null,[{key:"VERSION",
get:function(){return"4.0.0-beta"}}]),
e}();
t(document).on(a.CLICK_DATA_API,s.DATA_TOGGLE_CARROT,
function(e)
{e.preventDefault();
var n=e.target;
t(n).hasClass(r.BUTTON)||(n=t(n).closest(s.BUTTON)),
l._jQueryInterface.call(t(n),"toggle")}).on(a.FOCUS_BLUR_DATA_API,s.DATA_TOGGLE_CA
RROT,function(e)
{var n=t(e.target).closest(s.BUTTON)[0];
```

```
t(n).toggleClass(r.FOCUS,/^focus(in)?$/.test(e.type))}),
t.fn[e]=l._jQueryInterface,t.fn[e].Constructor=l,t.fn[e].noConflict=function()
{return t.fn[e]=i,l._jQueryInterface}
}(jQuery),function(t)
{var e="carousel",s="bs.carousel"
,a="."+s,
l=t.fn[e],
h={interval:5e3,keyboard:!0,slide:!1,
pause:"hover",wrap:!0},
c={interval:"(number|boolean)",
keyboard:"boolean",slide:"(boolean|string)",
pause:"(string|boolean)",wrap:"boolean"},
u={NEXT:"next",PREV:"prev",LEFT:"left",RIGHT:"right"},
d={SLIDE:"slide"+a,SLID:"slid"+a,KEYDOWN:"keydown"+a,MOUSEENTER:"mouseenter"+a,
MOUSELEAVE:"mouseleave"+a,
TOUCHEND:"touchend"+a,LOAD_DATA_API:"load.bs.carousel.data-api",
CLICK_DATA_API:"click.bs.carousel.data-api"},
f={CAROUSEL:"carousel",ACTIVE:"active",SLIDE:"slide",RIGHT:"carousel-item-
right",LEFT:"carousel-item-left",
NEXT:"carousel-item-next",PREV:"carousel-item-prev",ITEM:"carousel-
item"},p={ACTIVE:".active",
ACTIVE_ITEM:".active.carousel-item",
ITEM:".carousel-item",NEXT_PREV:"
.carousel-item-next,
.carousel-item-prev",
INDICATORS:".carousel-indicators",
DATA_SLIDE:"[data-slide], [data-slide-to]",
DATA_RIDE:'[data-ride="carousel"]'},_=function()
{
function l(e,i)
{n(this,l),
this._items=null,this._interval=null,this._activeElement=null,this._isPaused=!1,this._isSliding=!1,th
is.touchTimeout=null,this._config=this._getConfig(i),this._element=t(e)[0],
this._indicatorsElement=t(this._element).find(p.INDICATORS)[0],this._addEventListeners()}
return l.prototype.next=function()
```

{this._isSliding||this._slide(u.NEXT)},l.prototype.nextWhenVisible=function()
{document.hidden||this.next()},l.prototype.prev=function(){this._isSliding||this._slide(u.PREV)},
l.prototype.pause=function(e){e||(this._isPaused=!0),
t(this._element).find(p.NEXT_PREV)[0]&&r.supportsTransitionEnd()&&(r.triggerTransitionEnd(t
his._element),this.cycle(!0)),clearInterval(this._interval),this._interval=null},
l.prototype.cycle=function(t){t||(this._isPaused=!1),
this._interval&&(clearInterval(this._interval),
this._interval=null),this._config.interval&&!this._isPaused&&(this._interval=setInterval((document
.visibilityState?this.nextWhenVisible:this.next).bind(this),this._config.interval))},
l.prototype.to=function(e)
{var n=this;this._activeElement=t(this._element).find(p.ACTIVE_ITEM)[0];
var i=this._getItemIndex(this._activeElement);if(!(e>this._items.length-1||e<0))
if(this._isSliding)t(this._element).one(d.SLID,function()
{return n.to(e)});
else{if(i===e)
return this.pause(),
void this.cycle();
var o=e>i?u.NEXT:u.PREV;
this._slide(o,this._items[e])}},
l.prototype.dispose=function(){t(this._element).off(a),
t.removeData(this._element,s),
this._items=null,this._config=null,
this._element=null,this._interval=null,this._isPaused=null,this._isSliding=null,
this._activeElement=null,this._indicatorsElement=null},
l.prototype._getConfig=function(n){return
n=t.extend({},h,n),r.typeCheckConfig(e,n,c),n},l.prototype._addEventListeners=function()
{var e=this;this._config.keyboard&&t(this._element).on(d.KEYDOWN,function(t)
{return e._keydown(t)}),"hover"===this._config.pause&&(t(this._element).on(d.MOUSEENTER,
function(t){return e.pause(t)}).on(d.MOUSELEAVE,function(t){return
e.cycle(t)}),"ontouchstart"in
document.documentElement&&t(this._element).on(d.TOUCHEND,function()
{e.pause(),e.touchTimeout&&clearTimeout(e.touchTimeout),e.touchTimeout=setTimeout(function(
t)
{return e.cycle(t)},500+e._config.interval)}))},
l.prototype._keydown=function(t){if(!/input|textarea/i.test(t.target.tagName))switch(t.which){case

```
37:t.preventDefault(),this.prev();break;case 39:t.preventDefault(),this.next();break;default:return}},
l.prototype._getItemIndex=function(e){return
this._items=t.makeArray(t(e).parent().find(p.ITEM)),this._items.indexOf(e)},
l.prototype._getItemByDirection=function(t,e)
{var n=t===u.NEXT,i=t===u.PREV,
o=this._getItemIndex(e),
r=this._items.length-1;if((i&&0===o||n&&o===r)&&!this._config.wrap)
return e;
var s=(o+(t===u.PREV?-1:1))%this._items.length;
return-1===s?this._items[this._items.length-1]:this._items[s]},
l.prototype._triggerSlideEvent=function(e,n)
{var i=this._getItemIndex(e),
o=this._getItemIndex(t(this._element).find(p.ACTIVE_ITEM)[0]),
r=t.Event(d.SLIDE,{relatedTarget:e,direction:n,from:o,to:i});
return t(this._element).trigger(r),r},
l.prototype._setActiveIndicatorElement=function(e)
{if(this._indicatorsElement){t(this._indicatorsElement).find(p.ACTIVE).removeClass(f.ACTIVE);
var n=this._indicatorsElement.children[this._getItemIndex(e)];n&&t(n).addClass(f.ACTIVE)}},
l.prototype._slide=function(e,n)
{var i=this,o=t(this._element).find(p.ACTIVE_ITEM)[0],
s=this._getItemIndex(o),a=n||o&&this._getItemByDirection(e,o),l=this._getItemIndex(a),
h=Boolean(this._interval),c=void 0,_=void 0,
g=void 0;
if(e===u.NEXT?(c=f.LEFT,_=f.NEXT,g=u.LEFT):(c=f.RIGHT,_=f.PREV,g=u.RIGHT),
a&&t(a).hasClass(f.ACTIVE))this._isSliding=!1;
else if(!this._triggerSlideEvent(a,g).isDefaultPrevented()&&o&&a)
{this._isSliding=!0,h&&this.pause(),this._setActiveIndicatorElement(a);
var m=t.Event(d.SLID,{relatedTarget:a,direction:g,from:s,to:l});
.supportsTransitionEnd()&&t(this._element).hasClass(f.SLIDE)?(t(a).addClass(_),r.reflow(a),
t(o).addClass(c),t(a).addClass(c),
t(o).one(r.TRANSITION_END,function()
{t(a).removeClass(c+" "+_).addClass(f.ACTIVE
),t(o).removeClass(f.ACTIVE+" "+_+" "+c),
i._isSliding=!1,setTimeout(function(){return
t(i._element).trigger(m)},0)}).emulateTransitionEnd(600)):(t(o).removeClass(f.ACTIVE),
```

t(a).addClass(f.ACTIVE),
this._isSliding=!1,t(this._element).trigger(m)),h&&this.cycle()}},
l._jQueryInterface=function(e){return this.each(function()
{var n=t(this).data(s),o=t.extend({},h,t(this).data());
"object"===(void 0===e?"undefined":i(e))&&t.extend(o,e);var r="string"==typeof e?e:o.slide;
if(n||(n=new l(this,o),t(this).data(s,n)),"number"==typeof e)n.to(e);
else if("string"==typeof r){if(void 0===n[r])throw new Error('No method named "'+r+'"');
n[r]()}
else o.interval&&(n.pause(),n.cycle())})}
,l._dataApiClickHandler=function(e)
{var n=r.getSelectorFromElement(this);
if(n){var i=t(n)[0];if(i&&t(i).hasClass(f.CAROUSEL))
{var o=t.extend({},t(i).data(),t(this).data()),
a=this.getAttribute("data-slide-to");a&&(o.interval=!1),
l._jQueryInterface.call(t(i),o),a&&t(i).data(s).to(a),e.preventDefault()}}},
o(l,null,[{key:"VERSION",get:function(){return"4.0.0-beta"}},
{key:"Default",get:function()
{return h}}]),l}();
t(document).on(d.CLICK_DATA_API,p.DATA_SLIDE,_._dataApiClickHandler),
t(window).on(d.LOAD_DATA_API,function()
{t(p.DATA_RIDE).each(function()
{var e=t(this);_._jQueryInterface.call(e,e.data())})}),
t.fn[e]=_._jQueryInterface,t.fn[e].Constructor=_,
t.fn[e].noConflict=function(){return t.fn[e]=l,_._jQueryInterface}}(jQuery),function(t)
{var e="collapse",s="bs.collapse",a=t.fn[e],
l={toggle:!0,parent:""},h={toggle:"boolean",parent:"string"},
c={SHOW:"show.bs.collapse",SHOWN:"shown.bs.collapse",HIDE:"hide.bs.collapse",HIDDEN:"h
idden.bs.collapse",
CLICK_DATA_API:"click.bs.collapse.data-api"},
u={SHOW:"show",COLLAPSE:"collapse",
COLLAPSING:"collapsing",
COLLAPSED:"collapsed"},
d={WIDTH:"width",HEIGHT:"height"},
f={ACTIVES:".show, .collapsing",DATA_TOGGLE:'[data-toggle="collapse"]'},
p=function()

{function a(e,i){n(this,a),this._isTransitioning=!1,
this._element=e,
this._config=this._getConfig(i),this._triggerArray=t.makeArray(t('[data-
toggle="collapse"][href="#'+e.id+'"],[data-toggle="collapse"][data-target="#'+e.id+'"]'));for(var
o=t(f.DATA_TOGGLE),s=0;s<o.length;s++)
{var l=o[s],h=r.getSelectorFromElement(l);
null!==h&&t(h).filter(e).length>0&&this._triggerArray.push(l)}
this._parent=this._config.parent?this._getParent():null,this._config.parent||this._addAriaAndCollaps
edClass(this._element,this._triggerArray),this._config.toggle&&this.toggle()}
return a.prototype.toggle=function()
{t(this._element).hasClass(u.SHOW)?this.hide():this.show()},
a.prototype.show=function()
{var e=this;if(!this._isTransitioning&&!t(this._element).hasClass(u.SHOW))
{var n=void 0,i=void
0;if(this._parent&&((n=t.makeArray(t(this._parent).children().children(f.ACTIVES))).length||(n=nu
ll)),!(n&&(i=t(n).data(s))&&i._isTransitioning))
{var o=t.Event(c.SHOW);if(t(this._element).trigger(o),!o.isDefaultPrevented())
{n&&(a._jQueryInterface.call(t(n),"hide"),i||t(n).data(s,null));
var l=this._getDimension();
t(this._element).removeClass(u.COLLAPSE).addClass(u.COLLAPSING),
this._element.style[l]=0,
this._triggerArray.length&&t(this._triggerArray).removeClass(u.COLLAPSED).attr("aria-
expanded",!0),
this.setTransitioning(!0);
var h=function()
{t(e._element).removeClass(u.COLLAPSING).addClass(u.COLLAPSE).addClass(u.SHOW),
e._element.style[l]="",e.setTransitioning(!1),
t(e._element).trigger(c.SHOWN)};
if(r.supportsTransitionEnd())
{var d="scroll"+(l[0].toUpperCase()+l.slice(1));
t(this._element).one(r.TRANSITION_END,h).emulateTransitionEnd(600),
this._element.style[l]=this._element[d]+"px"}
else h()}}}},a.prototype.hide=function()
{var e=this;if(!this._isTransitioning&&t(this._element).hasClass(u.SHOW))
{var n=t.Event(c.HIDE);

```
if(t(this._element).trigger(n),!n.isDefaultPrevented())
{var i=this._getDimension();
if(this._element.style[i]=this._element.getBoundingClientRect()[i]+"px",r.reflow(this._element),
t(this._element).addClass(u.COLLAPSING).removeClass(u.COLLAPSE).removeClass(u.SHOW),
this._triggerArray.length)
for(var o=0;o<this._triggerArray.length;o++)
{var s=this._triggerArray[o],
a=r.getSelectorFromElement(s);
null!==a&&(t(a).hasClass(u.SHOW)||t(s).addClass(u.COLLAPSED).attr("aria-expanded",!1))}
this.setTransitioning(!0);
var l=function(){e.setTransitioning(!1),
t(e._element).removeClass(u.COLLAPSING).addClass(u.COLLAPSE).trigger(c.HIDDEN)};
this._element.style[i]="",
r.supportsTransitionEnd()?t(this._element).one(r.TRANSITION_END,l).emulateTransitionEnd(600
):l()}},
a.prototype.setTransitioning=function(t){this._isTransitioning=t},
a.prototype.dispose=function(){t.removeData(this._element,s),
this._config=null,this._parent=null,this._element=null,
this._triggerArray=null,this._isTransitioning=null},
a.prototype._getConfig=function(n){return n=t.extend({},l,n),n.toggle=Boolean(n.toggle),
r.typeCheckConfig(e,n,h),n},
a.prototype._getDimension=function()
{return t(this._element).hasClass(d.WIDTH)?d.WIDTH:d.HEIGHT},
a.prototype._getParent=function(){var e=this,n=t(this._config.parent)[0],
i='[data-toggle="collapse"][data-parent="'+this._config.parent+'"]';return
t(n).find(i).each(function(t,n)
{e._addAriaAndCollapsedClass(a._getTargetFromElement(n),[n])}),n},
a.prototype._addAriaAndCollapsedClass=function(e,n){if(e)
{var i=t(e).hasClass(u.SHOW);
n.length&&t(n).toggleClass(u.COLLAPSED,!i).attr("aria-expanded",i)}},
a._getTargetFromElement=function(e){var n=r.getSelectorFromElement(e);
return n?t(n)[0]:null},
a._jQueryInterface=function(e){return this.each(function()
{var n=t(this),
o=n.data(s),
```

r=t.extend({},l,n.data(),
"object"===(void 0===e?"undefined":i(e))&&e);
if(!o&&r.toggle&&/show|hide/.test(e)&&(r.toggle=!1),o||(o=new a(this,r),
n.data(s,o)),"string"==typeof e)
{if(void 0===o[e])throw new Error('No method named "'+e+'");o[e]()})},
o(a,null,[{key:"VERSION",
get:function()
{return"4.0.0-beta"}},
{key:"Default",get:function()
{return l}}]),a}();t(document).on(c.CLICK_DATA_API,f.DATA_TOGGLE,function(e)
{/input|textarea/i.test(e.target.tagName)||e.preventDefault();
var n=t(this),i=r.getSelectorFromElement(this);t(i).each(function()
{var e=t(this),i=e.data(s)?"toggle":n.data();p._jQueryInterface.call(e,i)})}),
t.fn[e]=p._jQueryInterface,t.fn[e].Constructor=p,t.fn[e].noConflict=function()
{return t.fn[e]=a,p._jQueryInterface}}
(jQuery),function(t){if("undefined"==typeof Popper)t
hrow new Error("Bootstrap dropdown require Popper.js (https://popper.js.org)");
var e="dropdown",s="bs.dropdown",a="."+s,l=t.fn[e],
h=new RegExp("38|40|27"),
c={HIDE:"hide"+a,HIDDEN:"hidden"+a,SHOW:"show"+a,
SHOWN:"shown"+a,CLICK:"click"+a,CLICK_DATA_API:"click.bs.dropdown.data-api",
KEYDOWN_DATA_API:"keydown.bs.dropdown.data-api",
KEYUP_DATA_API:"keyup.bs.dropdown.data-api"},
u={DISABLED:"disabled",SHOW:"show",
DROPUP:"dropup",MENURIGHT:"dropdown-menu-right",
MENULEFT:"dropdown-menu-left"},
d={DATA_TOGGLE:'[data-toggle="dropdown"]',FORM_CHILD:".dropdown form",
MENU:".dropdown-menu",
NAVBAR_NAV:".navbar-nav",
VISIBLE_ITEMS:".dropdown-menu .dropdown-item:not(.disabled)"},f={TOP:"top-start",
TOPEND:"top-end",
BOTTOM:"bottom-start",
BOTTOMEND:"bottom-end"},p={placement:f.BOTTOM,offset:0,flip:!0},
_={placement:"string",offset:"(number|string)",flip:"boolean"},
g=function()

{function l(t,e){n(this,l),
this._element=t,
this._popper=null,
this._config=this._getConfig(e),
this._menu=this._getMenuElement(),
this._inNavbar=this._detectNavbar(),
this._addEventListeners()}return l.prototype.toggle=function()
{if(!this._element.disabled&&!t(this._element).hasClass(u.DISABLED))
{var e=l._getParentFromElement(this._element),
n=t(this._menu).hasClass(u.SHOW);
if(l._clearMenus(),!n)
{var i={relatedTarget:this._element},
o=t.Event(c.SHOW,i);if(t(e).trigger(o),
!o.isDefaultPrevented())
{var
r=this._element;t(e).hasClass(u.DROPUP)&&(t(this._menu).hasClass(u.MENULEFT)||t(this._men
u).hasClass(u.MENURIGHT))&&(r=e),
this._popper=new Popper(r,this._menu,
this._getPopperConfig()),"ontouchstart"in
document.documentElement&&!t(e).closest(d.NAVBAR_NAV).length&&t("body").children().on(
"mouseover",null,t.noop),
this._element.focus(),
this._element.setAttribute("aria-expanded",!0),
t(this._menu).toggleClass(u.SHOW),
t(e).toggleClass(u.SHOW).trigger(t.Event(c.SHOWN,i))}}},
l.prototype.dispose=function()
{t.removeData(this._element,s),
t(this._element).off(a),
this._element=null,
this._menu=null,null!==this._popper&&this._popper.destroy(),this._popper=null
},l.prototype.update=function()
{this._inNavbar=this._detectNavbar(),
null!==this._popper&&this._popper.scheduleUpdate()},
l.prototype._addEventListeners=function()
{var e=this;

```
t(this._element).on(c.CLICK,function(t)
{t.preventDefault(),t.stopPropagation(),e.toggle()})},
l.prototype._getConfig=function(n)
{var i=t(this._element).data();
return void 0!==i.placement&&(i.placement=f[i.placement.toUpperCase()]),
n=t.extend({},
this.constructor.Default,t(this._element).data(),n),r.typeCheckConfig(e,n,this.constructor.DefaultTy
pe),n},
l.prototype._getMenuElement=function()
{if(!this._menu)
{var e=l._getParentFromElement(this._element
);this._menu=t(e).find(d.MENU)[0]}
return this._menu},
l.prototype._getPlacement=function()
{var e=t(this._element).parent(),
n=this._config.placement;
return
e.hasClass(u.DROPUP)||this._config.placement===f.TOP?(n=f.TOP,t(this._menu).hasClass(u.ME
NURIGHT)&&(n=f.TOPEND)):t(this._menu).hasClass(u.MENURIGHT)&&(n=f.BOTTOMEND)
,n},
l.prototype._detectNavbar=function()
{return t(this._element).closest(".navbar").length>0},
l.prototype._getPopperConfig=function()
{var t={placement:this._getPlacement(),modifiers:{offset:{offset:this._config.offset},
flip:{enabled:this._config.flip}}};
return this._inNavbar&&(t.modifiers.applyStyle={enabled:!this._inNavbar}),t},
l._jQueryInterface=function(e)
{return this.each(function()
{var n=t(this).data(s),o="object"===(void 0===e?"undefined":i(e))?e:null;
if(n||(n=new l(this,o),t(this).data(s,n)),"string"==typeof e)
{if(void 0===n[e])throw new Error('No method named "'+e+'"');n[e]()}})},
l._clearMenus=function(e){if(!e||3!==e.which&&("keyup"!==e.type||9===e.which))
for(var n=t.makeArray(t(d.DATA_TOGGLE)),
i=0;i<n.length;i++)
{var o=l._getParentFromElement(n[i]),r=t(n[i]).data(s),
```

a={relatedTarget:n[i]};
if(r)
{var
h=r._menu;if(t(o).hasClass(u.SHOW)&&!(e&&("click"===e.type&&/input|textarea/i.test(e.target.t
agName)||"keyup"===e.type&&9===e.which)&&t.contains(o,e.target)))
{var f=t.Event(c.HIDE,a);t(o).trigger(f),f.isDefaultPrevented()||("ontouchstart"in
document.documentElement&&t("body").children().off("mouseover",null,t.noop),
n[i].setAttribute("aria-expanded","false"),
t(h).removeClass(u.SHOW),t(o).removeClass(u.SHOW).trigger(t.Event(c.HIDDEN,a)))}}},
l._getParentFromElement=function(e)
{var n=void 0,i=r.getSelectorFromElement(e);
return i&&(n=t(i)[0]),n||e.parentNode},
l._dataApiKeydownHandler=function(e)
{if(!(!h.test(e.which)||/button/i.test(e.target.tagName)&&32===e.which||/input|textarea/i.test(e.targe
t.tagName)||(e.preventDefault(),e.stopPropagation(),
this.disabled||t(this).hasClass(u.DISABLED))))
{var n=l._getParentFromElement(this),
i=t(n).hasClass(u.SHOW);
if((i||27===e.which&&32===e.which)&&(!i||27!==e.which&&32!==e.which))
{var o=t(n).find(d.VISIBLE_ITEMS).get();if(o.length)
{var r=o.indexOf(e.target);
38===e.which&&r>0&&r--,
40===e.which&&r<o.length-1&&r++,r<0&&(r=0),
o[r].focus()}}else{if(27===e.which)
{var s=t(n).find(d.DATA_TOGGLE)[0];
t(s).trigger("focus")}
t(this).trigger("click")}}},
o(l,null,[{key:"VERSION",get:function()
{return"4.0.0-beta"}},{key:"Default",get:function()
{return p}},
{key:"DefaultType",
get:function()
{return _}}]),l}();
t(document).on(c.KEYDOWN_DATA_API,d.DATA_TOGGLE,g._dataApiKeydownHandler).on(c
.KEYDOWN_DATA_API,d.MENU,g._dataApiKeydownHandler).on(c.CLICK_DATA_API+"

"+c.KEYUP_DATA_API,g._clearMenus).on(c.CLICK_DATA_API,d.DATA_TOGGLE,
function(e)
{e.preventDefault(),e.stopPropagation(),
g._jQueryInterface.call(t(this),"toggle")}).on(c.CLICK_DATA_API,d.FORM_CHILD,
function(t)
{t.stopPropagation()}),
t.fn[e]=g._jQueryInterface,
t.fn[e].Constructor=g,
t.fn[e].noConflict=function()
{return t.fn[e]=l,g._jQueryInterface}}
(jQuery),
function(t)
{var e="modal",
s=".bs.modal",
a=t.fn[e],
l={backdrop:!0,keyboard:!0,focus:!0,show:!0},
h={backdrop:"(boolean|string)",
keyboard:"boolean",focus:"boolean",show:"boolean"},
c={HIDE:"hide.bs.modal",
HIDDEN:"hidden.bs.modal",
SHOW:"show.bs.modal",
SHOWN:"shown.bs.modal",
FOCUSIN:"focusin.bs.modal",
RESIZE:"resize.bs.modal",
CLICK_DISMISS:"click.dismiss.bs.modal",
KEYDOWN_DISMISS:"keydown.dismiss.bs.modal",
MOUSEUP_DISMISS:"mouseup.dismiss.bs.modal",
MOUSEDOWN_DISMISS:"mousedown.dismiss.bs.modal",
CLICK_DATA_API:"click.bs.modal.data-api"},
u={SCROLLBAR_MEASURER:"modal-scrollbar-measure",
BACKDROP:"modal-backdrop",OPEN:"modal-open",FADE:"fade",SHOW:"show"},
d={DIALOG:".modal-dialog",DATA_TOGGLE:'[data-toggle="modal"]',
DATA_DISMISS:'[data-dismiss="modal"]',
FIXED_CONTENT:".fixed-top, .fixed-bottom, .is-fixed,
.sticky-top",NAVBAR_TOGGLER:".navbar-toggler"},

```
f=function()
{function a(e,i){n(this,a),this._config=this._getConfig(i),
this._element=e,
this._dialog=t(e).find(d.DIALOG)[0],
this._backdrop=null,
this._isShown=!1,
this._isBodyOverflowing=!1,
this._ignoreBackdropClick=!1,
this._originalBodyPadding=0,
this._scrollbarWidth=0}
return a.prototype.toggle=function(t)
{return this._isShown?this.hide():this.show(t)},
a.prototype.show=function(e)
{var n=this;
if(!this._isTransitioning){r.supportsTransitionEnd()&&t(this._element).hasClass(u.FADE)&&(this.
_isTransitioning=!0);
var i=t.Event(c.SHOW,
{relatedTarget:e});
t(this._element).trigger(i),
this._isShown||i.isDefaultPrevented()||(this._isShown=!0,
this._checkScrollbar(),this._setScrollbar(),
t(document.body).addClass(u.OPEN),
this._setEscapeEvent(),this._setResizeEvent(),
t(this._element).on(c.CLICK_DISMISS,d.DATA_DISMISS,function(t)
{return n.hide(t)}),
t(this._dialog).on(c.MOUSEDOWN_DISMISS,function()
{t(n._element).one(c.MOUSEUP_DISMISS,function(e)
{t(e.target).is(n._element)&&(n._ignoreBackdropClick=!0)})}),this._showBackdrop(function(){ret
urn n._showElement(e)}))}},a.prototype.hide=function(e)
{var n=this;if(e&&e.preventDefault(),!this._isTransitioning&&this._isShown)
{var
i=r.supportsTransitionEnd()&&t(this._element).hasClass(u.FADE);i&&(this._isTransitioning=!0);v
ar o=t.Event(c.HIDE);
t(this._element).trigger(o),
this._isShown&&!o.isDefaultPrevented()&&(this._isShown=!1,
```

```
this._setEscapeEvent(),
this._setResizeEvent(),t(document).off(c.FOCUSIN),
t(this._element).removeClass(u.SHOW),
t(this._element).off(c.CLICK_DISMISS),
t(this._dialog).off(c.MOUSEDOWN_DISMISS),
i?t(this._element).one(r.TRANSITION_END,function(t)
{return n._hideModal(t)}).emulateTransitionEnd(300):this._hideModal())}},
a.prototype.dispose=function()
{t.removeData(this._element,"bs.modal"),
t(window,document,this._element,this._backdrop).off(s),this._config=null,
this._element=null,
this._dialog=null,
this._backdrop=null,
this._isShown=null,
this._isBodyOverflowing=null,
this._ignoreBackdropClick=null,
this._scrollbarWidth=null},
a.prototype.handleUpdate=function()
{this._adjustDialog()},
a.prototype._getConfig=function(n)
{return n=t.extend({},l,n),
r.typeCheckConfig(e,n,h),n},
a.prototype._showElement=function(e)
{var n=this,
i=r.supportsTransitionEnd()&&t(this._element).hasClass(u.FADE);
this._element.parentNode&&this._element.parentNode.nodeType===Node.ELEMENT_NODE||do
cument.body.appendChild(this._element),
this._element.style.display="block",this._element.removeAttribute("aria-hidden"),
this._element.scrollTop=0,
i&&r.reflow(this._element),
t(this._element).addClass(u.SHOW),
this._config.focus&&this._enforceFocus();
var o=t.Event(c.SHOWN,
{relatedTarget:e}),s=function()
{n._config.focus&&n._element.focus(),n._isTransitioning=!1,
```

t(n._element).trigger(o)};
i?t(this._dialog).one(r.TRANSITION_END,
s).emulateTransitionEnd(300):s()},a
.prototype._enforceFocus=function()
{var e=this;
t(document).off(c.FOCUSIN).on(c.FOCUSIN,function(n){document===n.target||e._element===n.t
arget||t(e._element).has(n.target).length||e._element.focus()})},a.prototype._setEscapeEvent=functio
n()
{var
e=this;this._isShown&&this._config.keyboard?t(this._element).on(c.KEYDOWN_DISMISS,functi
on(t){27===t.which&&(t.preventDefault(),e.hide())}):this._isShown||t(this._element).off(c.KEYD
OWN_DISMISS)},a.prototype._setResizeEvent=function()
{var e=this;this._isShown?t(window).on(c.RESIZE,function(t)
{return e.handleUpdate(t)}):t(window).off(c.RESIZE)},
a.prototype._hideModal=function()
{var e=this;this._element.style.display="none",this._element.setAttribute("aria-
hidden",!0),this._isTransitioning=!1,
this._showBackdrop(function()
{t(document.body).removeClass(u.OPEN),
e._resetAdjustments(),e._resetScrollbar(),
t(e._element).trigger(c.HIDDEN)})},
a.prototype._removeBackdrop=function()
{this._backdrop&&(t(this._backdrop).remove(),this._backdrop=null)},
a.prototype._showBackdrop=function(e)
{var n=this,i=t(this._element).hasClass(u.FADE)?u.FADE:"";
if(this._isShown&&this._config.backdrop)
{var o=r.supportsTransitionEnd()&&i;
if(this._backdrop=document.createElement("div"),
this._backdrop.className=u.BACKDROP,i&&t(this._backdrop).addClass(i),
t(this._backdrop).appendTo(document.body),
t(this._element).on(c.CLICK_DISMISS,
function(t)
{n._ignoreBackdropClick?n._ignoreBackdropClick=!1:t.target===t.currentTarget&&("static"===n.
_config.backdrop?n._element.focus():n.hide())}),
o&&r.reflow(this._backdrop),

```
t(this._backdrop).addClass(u.SHOW),!e)
return;
if(!o)return void e();
t(this._backdrop).one(r.TRANSITION_END,e).emulateTransitionEnd(150)}
else
 if(!this._isShown&&this._backdrop)
{t(this._backdrop).removeClass(u.SHOW);
var s=function(){n._removeBackdrop(),e&&e()};
r.supportsTransitionEnd()&&t(this._element).hasClass(u.FADE)?t(this._backdrop).one(r.TRANSIT
ION_END,s).emulateTransitionEnd(150):s()}
else e&&e()},
a.prototype._adjustDialog=function()
{var t=this._element.scrollHeight>document.documentElement.clientHeight;
!this._isBodyOverflowing&&t&&(this._element.style.paddingLeft=this._scrollbarWidth+"px"),
this._isBodyOverflowing&&!t&&(this._element.style.paddingRight=this._scrollbarWidth+"px")},
a.prototype._resetAdjustments=function()
{this._element.style.paddingLeft="",
this._element.style.paddingRight=""},
a.prototype._checkScrollbar=function()
{this._isBodyOverflowing=document.body.clientWidth<window.innerWidth,
this._scrollbarWidth=this._getScrollbarWidth()},
a.prototype._setScrollbar=function()
{var e=this;if(this._isBodyOverflowing)
{t(d.FIXED_CONTENT).each(function(n,i)
{var o=t(i)[0].style.paddingRight,r=t(i).css("padding-right");
t(i).data("padding-right",o).css("padding-right",parseFloat(r)+e._scrollbarWidth+"px")}),
t(d.NAVBAR_TOGGLER).each(function(n,i)
{var o=t(i)[0].style.marginRight,r=t(i).css("margin-right");
t(i).data("margin-right",o).css("margin-right",parseFloat(r)+e._scrollbarWidth+"px")});var
n=document.body.style.paddingRight,i=t("body").css("padding-right");
t("body").data("padding-right",n).css("padding-right",
parseFloat(i)+this._scrollbarWidth+"px")}},
a.prototype._resetScrollbar=function()
{t(d.FIXED_CONTENT).each(function(e,n)
{var i=t(n).data("padding-right");void 0!==i&&t(n).css("padding-right",i).removeData("padding-
```

```
right")}),t(d.NAVBAR_TOGGLER).each(function(e,n){var i=t(n).data("margin-right");
void 0!==i&&t(n).css("margin-right",i).removeData("margin-right")});
var e=t("body").data("padding-right");
void 0!==e&&t("body").css("padding-right",e).removeData("padding-right")},
a.prototype._getScrollbarWidth=function()
{var t=document.createElement("div");
t.className=u.SCROLLBAR_MEASURER,document.body.appendChild(t);
var e=t.getBoundingClientRect().width-t.clientWidth;
return document.body.removeChild(t),e},a._jQueryInterface=function(e,n){return
this.each(function()
{var o=t(this).data("bs.modal"),
r=t.extend({},a.Default,t(this).data(),
"object"===(void 0===e?"undefined":i(e))&&e);
if(o||(o=new a(this,r),t(this).data("bs.modal",o)),
"string"==typeof e)
{if(void 0===o[e])throw new Error('No method named "'+e+'"');o[e](n)}
else r.show&&o.show(n)})},
o(a,null,[{key:"VERSION",get:function()
{return"4.0.0-beta"}},{key:"Default",get:function()
{return l}}]),a}();
t(document).on(c.CLICK_DATA_API,d.DATA_TOGGLE,
function(e){var n=this,i=void 0,o=r.getSelectorFromElement(this);o&&(i=t(o)[0]);
var s=t(i).data("bs.modal")?"toggle":t.extend({},
t(i).data(),t(this).data());
"A"!==this.tagName&&"AREA"!==this.tagName||e.preventDefault();
var a=t(i).one(c.SHOW,function(e)
{e.isDefaultPrevented()||a.one(c.HIDDEN,function()
{t(n).is(":visible")&&n.focus()})});f._jQueryInterface.call(t(i),s,this)}),t
.fn[e]=f._jQueryInterface,t.fn[e].Constructor=f,t.fn[e].noConflict=function()
{return t.fn[e]=a,f._jQueryInterface}}(jQuery),
function(t)
{var "function"==typeof
this.config.content?this.config.content.call(this.element):this.config.content)},
h.prototype._cleanTipClass=function()
{var t=r(this.getTipElement()),
```

```
e=t.attr("class").match(c);
null!==e&&e.length>0&&t.removeClass(e.join(""))},
h._jQueryInterface=function(t){return this.each(function()
{var e=r(this).data("bs.popover"),
n="object"===(void 0===t?"undefined":i(t))?t:null;
if((e||!/destroy|hide/.test(t))&&(e||(e=new h(this,n),r(this).data("bs.popover",e)),
"string"==typeof t))
{if(void 0===e[t])throw new Error('No method named "'+t+'");e[t]()})},
o(h,null,[{key:"VERSION",
get:function(){return"4.0.0-beta"}},
{key:"Default",get:function(){return u}},
{key:"NAME",get:function(){return a}},
{key:"DATA_KEY",get:function(){return"bs.popover"}},
{key:"Event",get:function(){return _}},
{key:"EVENT_KEY",get:function(){return l}},
{key:"DefaultType",get:function(){return d}}]),h}
(s);r.fn[a]=g._jQueryInterface,
r.fn[a].Constructor=g,r.fn[a].noConflict=function()
{return r.fn[a]=h,g._jQueryInterface}}(jQuery)}().
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_ doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Диплом_.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_.ppt	Презентація кваліфікаційної роботи