ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА
(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій
(повне найменування факультету)

Кафедра механіки, автоматизації та інформаційних технологій
(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»

«Використання фрактальної графіки»

«Using fractal graphics»

Виконала: здобувачка денної форми навчання
спеціальності 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

Алєксандрова Ольга Леонідівна
(прізвище, ім'я, по-батькові здобувача)

Керівник к.ф.-м.н., доц. Рачинська А.Л.
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент викладач Недєва О.А.
(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
механіки, автоматизації та інформаційних технологій
№ ___ від ____.____. 20___ р.

Завідувачка кафедри
_____ Алла РАЧИНСЬКА
(підпис)

Захищено на засіданні ЕК № _____
протокол № __від ____.____.20___ р.

Оцінка_____/_____/_____
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК
_____ Микола МАЛАКСІАНО
(підпис)

Одеса 2024

АНОТАЦІЯ

Дипломна робота розроблялась на тему «Використання фрактальної графіки».

Метою даної роботи є розробка та реалізація програмного продукту - фрактального редактора з розширеним функціоналом. Основними цілями проекту є створення зручного та потужного інструменту для створення та редагування фрактальних зображень.

Результатом проведеної роботи є завершений програмний продукт — фрактальний редактор, що володіє певним функціоналом:

– відображення трьох основних фракталів з можливістю зміни їхнього положення на екрані шляхом введення координат або за допомогою натискання на відповідну іконку та переміщення за допомогою лівої кнопки миші. Додатково, користувачі можуть змінювати колір і розмір пензля, змінювати розмір фракталів, збільшуючи або зменшуючи їх, а також користуватися двома типами гумок. Редагування робочого полотна передбачає зміну висоти та ширини, розгортання на весь екран, вибір стандартних кольорів для фону або індивідуальне налаштування кольору, а також використання зображення як фону полотна.

– операції з файлами включають імпортування параметрів, відкриття, збереження, збереження з вибором шляху, збереження малюнку і параметрів, а також копіювання фрактала до буферу обміну.

– анімаційні відтворення фракталів з основними діями: запуск, пауза та зупинка. Крім того, для кожного фракталу доступні додаткові функції. Також існує можливість редагування швидкості анімації.

ABSTRACT

The thesis was developed on the topic “Using fractal graphics”.

The purpose of this work is to develop and implement a software product - a fractal editor with advanced functionality. The main goals of the project are to create a convenient and powerful tool for creating and editing fractal images.

The result of the work is a completed software product - a fractal editor with certain functionality:

- displaying three main fractals with the ability to change their position on the screen by entering coordinates or by clicking on the corresponding icon and moving them with the left mouse button. Additionally, users can change the color and size of the brush, resize fractals by enlarging or reducing them, and use two types of erasers. Editing the canvas includes changing the height and width, expanding to full screen, choosing standard colors for the background or customizing the color, and using an image as the canvas background.
- file operations include importing parameters, opening, saving, saving with path selection, saving a drawing and parameters, and copying a fractal to the clipboard.
- animated playback of fractals with basic actions: start, pause, and stop. In addition, additional functions are available for each fractal. You can also edit the animation speed.

ЗМІСТ

ВСТУП

У сучасному світі комп'ютерна графіка відіграє важливу роль у різних галузях науки, техніки та мистецтва. Одним із перспективних напрямків у цій сфері є фрактальна графіка, яка дозволяє створювати складні й красиві зображення за допомогою математичних алгоритмів. Фрактальні структури знаходять застосування у численних областях, таких як комп'ютерна анімація, моделювання природних явищ, криптографія та аналіз даних.

Тема фрактальної графіки, зокрема фрактальних дерев, є надзвичайно актуальною через їхні унікальні властивості та широкий спектр застосувань. Фрактали володіють властивостями самоподібності та нескінченної складності, що дозволяє використовувати їх для моделювання природних об'єктів, таких як гілки дерев, берегові лінії та хмари.

Метою даної дипломної роботи є розробка та реалізація програмного продукту - фрактального редактора з розширеним функціоналом. Основні завдання проекту полягають у створенні інструменту, який дозволяє користувачам легко та зручно створювати та редагувати фрактальні зображення, а також досліджувати їхні властивості.

Актуальність теми обумовлена потребою в потужних інструментах для роботи з фрактальною графікою, які можуть знайти застосування у різних наукових та практичних сферах. Створення фрактального редактора з розширеним функціоналом дозволить значно розширити можливості дослідження та використання фрактальних структур, сприятиме розвитку нових підходів у комп'ютерній графіці та стимулюватиме подальші наукові дослідження у цій галузі.

Ціллю даної дипломної роботи є розробка інтерактивного фрактального редактора, який дозволить користувачам створювати, налаштовувати та візуалізувати різні типи фракталів, а також досліджувати їхні геометричні та математичні властивості.

Щоб досягти поставленої цілі необхідно вирішити наступні задачі:

1. Провести аналіз існуючих методів та алгоритмів побудови фрактальних структур.
2. Обрати платформу та інструментальні засоби для реалізації програмного продукту.
3. Розробити архітектуру фрактального редактора, включаючи модулі для створення, налаштування та візуалізації фракталів.
4. Реалізувати інтерфейс користувача, що забезпечить зручне та інтуїтивне керування всіма функціями редактора.
5. Розробити алгоритми для побудови Піфагорового дерева, кривої Гільберта та змії Госпера.
6. Реалізувати функції збереження та завантаження параметрів фракталів, а також експорту зображень у різних графічних форматах.
7. Провести тестування та діагностику програмного забезпечення для виявлення та виправлення помилок.
8. Проаналізувати отримані результати та оцінити ефективність розробленого редактора для роботи з фрактальною графікою.

Таким чином, виконання цих завдань дозволить створити ефективний інструмент для роботи з фракталами, що може знайти широке застосування у різних галузях науки та техніки.

1 ФРАКТАЛЬНІ СТРУКТУРИ: СИМЕТРИЧНІ БІНАРНІ ДЕРЕВА, ПІФАГОРОВЕ ДЕРЕВО, КРИВА ГІЛЬБЕРТА ТА КРИВА ГОСПЕРА

1.1 Симетричне Бінарне Дерево

Симетричне бінарне дерево - це тип бінарного дерева, в якому кожен вузол має не більше двох дочірніх вузлів, із найлівішим дочірнім вузлом, який має значення менше батьківського вузла, із найправішим дочірнім вузлом, який має значення більше батьківського вузла. У симетричному бінарному дереві, ліве піддерево буде містити всі вузли з меншими значеннями, праве піддерево буде містити всі вузли з більшими значеннями, і кожне піддерево також буде симетричним бінарним деревом. Кожне піддерево також є симетричним бінарним деревом, що робить цю структуру рекурсивною.

Симетричне бінарне дерево є важливим об'єктом в інформатиці та математиці завдяки своїм властивостям і застосуванням. Воно забезпечує ефективний спосіб зберігання та впорядкування даних, дозволяючи швидкий доступ, додавання і видалення елементів.

Побудова Симетричного Бінарного Дерева. Щоб побудувати симетричне бінарне дерево, треба вибрати кут θ при $0° < \theta < 180°$ і коефіцієнт масштабування r при $0 < r < 1$. Почати з вертикального відрізка (стовбура) довжиною 1. У верхній частині стовбур розгалужується на дві гілки, кожна з яких утворює кут θ з лінійним продовженням стовбура, одна ліворуч, а інша праворуч. Кожна гілка має довжину r. Кожна з цих двох гілок утворює стовбур піддерева, яке розгалужується ще на дві гілки за тим же правилом. Кут знову дорівнює θ, а довжина кожної з чотирьох нових гілок дорівнює r^2.

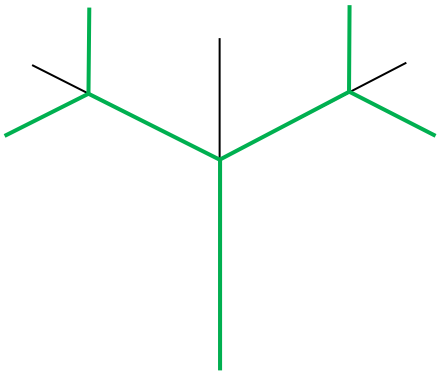


Рисунок 1.1- Стовбур та гілки, де θ = 60° і *r* = 0,65

Симетричне бінарне дерево отримується, продовжуючи додавати нові гілки до нескінченності, використовуючи кут θ і коефіцієнт масштабування *r* для кожного набору нових сегментів гілок. Кінцеві точки гілок називаються вершинами гілок.

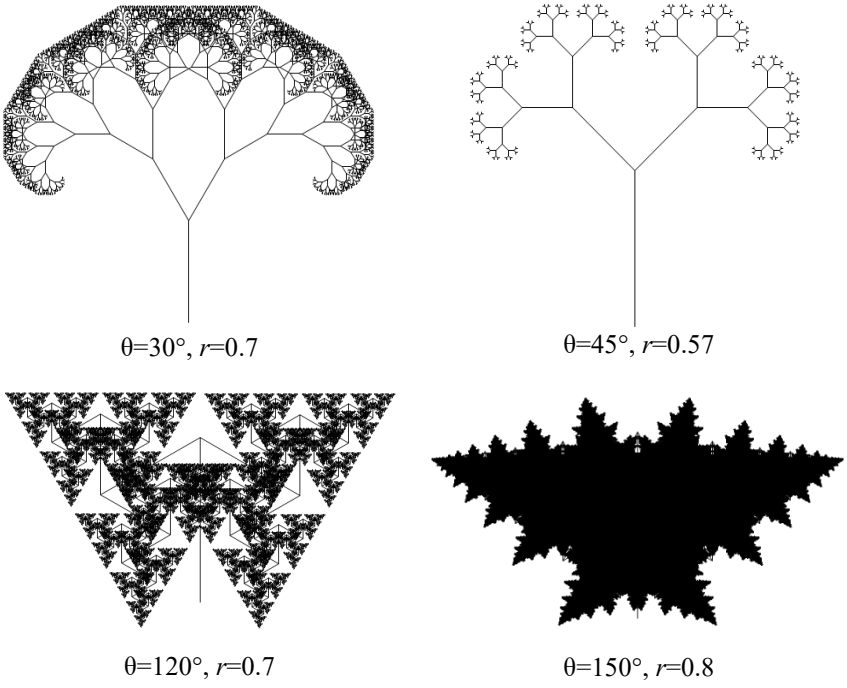


Рисунок 1.2 - Приклад симетричних бінарних дерев

Ітераційні Функції для Побудови Бінарного Дерева базуються на тому, що дві гілки отримані зі стовбура шляхом масштабування в коефіцієнті *r*, повороту проти годинникової стрілки на θ (ліворуч) і -θ (праворуч), а потім трансляції до вершини стовбура. Третя необхідна функція - тотожність. Вона

зберігає вже намальовані гілки в їх поточному розташуванні, в той час як перші дві функції додають нові гілки.

$$f_1(x) = \begin{bmatrix} rcos(\theta) & -\mathrm{rsin}(\theta) \\ \mathrm{rsin}(\theta) & rcos(\theta) \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \text{масштабувати на } r, \text{повертати на } \theta° \tag{1.1}$$

$$f_2(x) = \begin{bmatrix} rcos(\theta) & r\sin(\theta) \\ -r\sin(\theta) & rcos(\theta) \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \text{масштабувати на } r, \text{повертати на } \theta° \tag{1.2}$$

$$f_3(x) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x \tag{1.3}$$

Функція 1.3 у цій ітераційній системі функцій не є контрактивним відображенням. Тому теорія контрактивних ітераційних систем функцій не обов'язково може бути застосована тут. Однак у цьому випадку ітерації початкової множини за цією ітераційною системою функцій збігаються до граничної множини, але межа буде залежати від початкової множини. Цей тип ітераційної системи функцій часто називають ітераційною системою функцій зі згущенням.

Якщо використовувати лише перші дві функції як систему ітераційних функцій, то система буде стиснутою і всі початкові множини зійдуться до вершин гілок.

Шляхи симетричного бінарного дерева можна адресувати скінченними рядками з літер L та R, де L відповідає розгалуженню ліворуч, а R - праворуч, таким чином здійснюючи адресацію та побудову шляхів у симетричному бінарному дереві. Наприклад, на (Рис. 1.3) показано кінці шляхів, що відповідають адресам L, R, LL, LR, RL та RR. Червоні гілки намальовані ліворуч, а сині - праворуч .

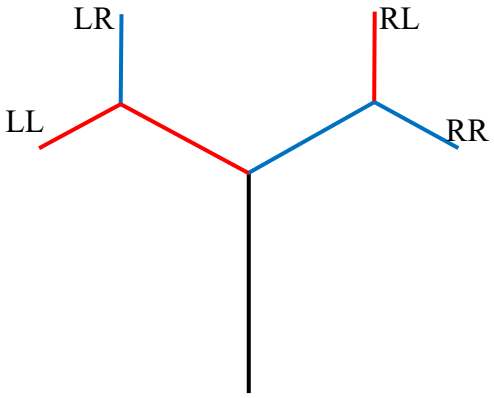
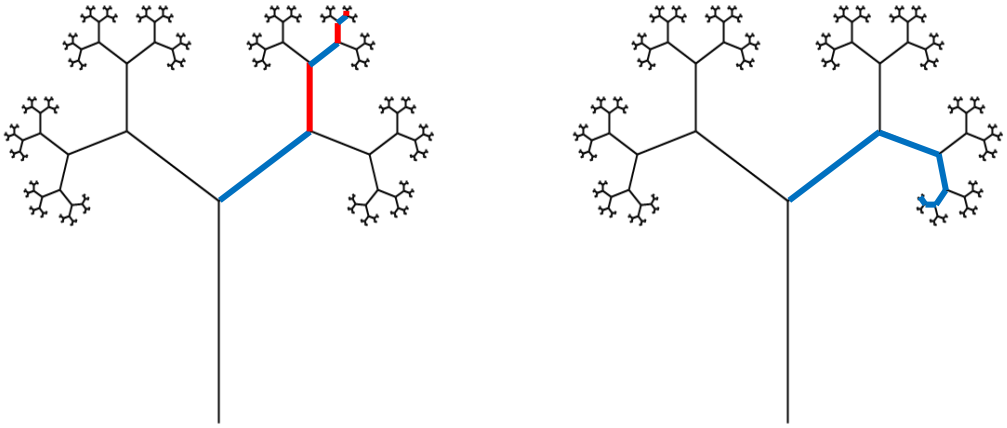


Рисунок 1.3 - Кінці шляхів

На наступному зображенні (Рис. 1.4) показано перші 7 ітерацій побудови симетричного бінарного дерева.



RLRLRLR RRRRRRR

Рисунок 1.4 - Перші 7 ітерацій побудови симетричного бінарного дерева

Кінці гілок отримуються як нескінченні послідовності L і R. Наприклад, послідовність (RL)∞ = RLRLRL... буде шляхом, що чергує праві та ліві гілки. Особлива увага приділяється симетрії з адресою (LR)∞, яка чергує ліві та праві гілки. Адреса R∞ відповідає нескінченній спіралі правих гілок, що утворюють повторювану структуру, і є прикладом самоподібності у фрактальних деревоподібних структурах..

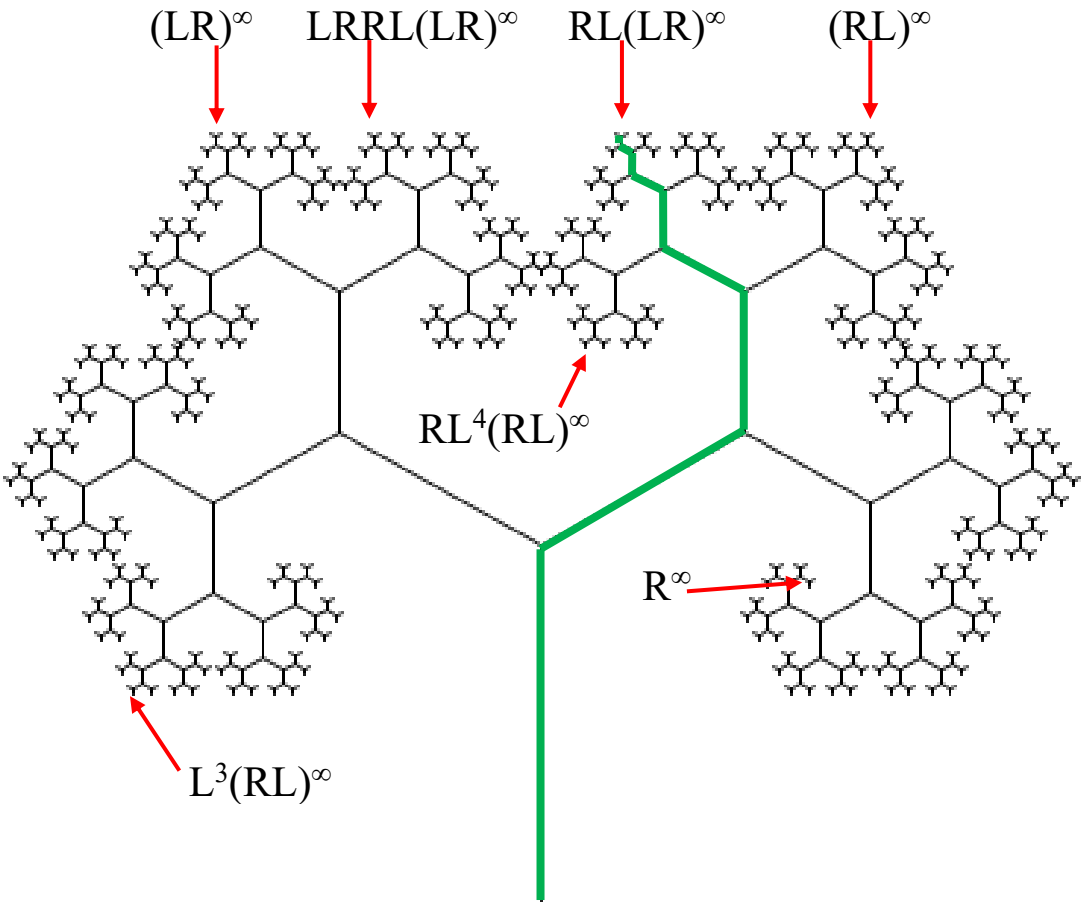
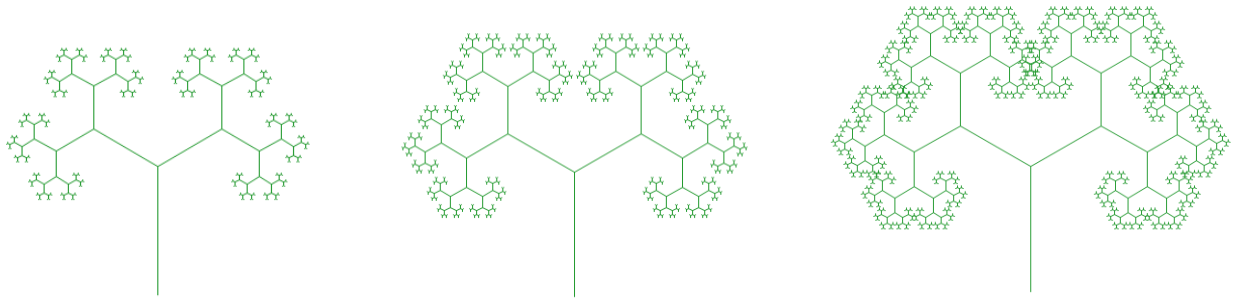


Рисунок 1.5 - Кінці гілок отримуються як нескінченні послідовності L і R

Вибір масштабного коефіцієнта для самоуникаючого симетричного бінарного дерева залежить від величини коефіцієнта *r*: якщо *r* замалий, то гілки дерева будуть самоуникаючими, а якщо завеликий, то гілки перетинатимуться. Мандельброт і Фрейм показали, що для кожного кута θ існує унікальний масштабний коефіцієнт r_{sc}, при якому симетричне бінарне дерево з цим θ і *r* буде самоуникаючим, тобто гілки можуть торкатися в одній точці, але не можуть перетинатися. У цьому випадку верхівки лівих гілок збігатимуться з верхівками деяких правих гілок, але жодна верхівка не збігатиметься з жодною точкою дерева, що не є вершиною. На рисунку (Рис. 1.6) показано три приклади для θ=60°.



Самоуникнення
r = 0.58

Самоконтактний
r = 0.62

Самонакладання
r = 0.65

Рисунок 1.6 - Три дерева для яких θ≈60°

Для $0 < \theta \leq 90°$ нехай N_θ - найменше ціле число, для якого $N_\theta\theta \geq 90°$. На цьому проміжку значення r_{sc} є єдиним розв'язком рівняння:

$$r^2\left(\frac{r^{N_\theta+1}\cos((N_\theta-1)\theta)-r^{N_\theta}\cos(N_\theta\theta)-r\cos(\theta)+1}{r^2+1-2r\cos(\theta)}\right)=\frac{1}{2}. \quad (1.4)$$

Для $90° \leq \theta < 135°$ отримано:

$$r_{sc}=\frac{1}{\sqrt{2-3\cos^2(\theta)}-\cos(\theta)} \quad (1.5)$$

і для $135° \leq \theta < 180°$:

$$r_{sc}=-\frac{1}{2\cos(\theta)} \quad (1.6)$$

Графік r_{sc} як функції θ. Судячи з того, що $\frac{1}{2} < r_{sc} \leq \frac{1}{\sqrt{2}}$, з максимальним значенням при $\theta = 90°$ і $\theta = 135°$. Штрихові лінії з'являються при 135° і $\frac{90°}{N}$ для

N = 1, 2, 3, 4, 5, 6. При $\theta = \frac{90^\circ}{N}$ для N > 6 пунктирних ліній повинно бути набагато більше, але зі збільшенням N вони починають збігатися разом. Ці значення відображають зміну N_θ у рівнянні для r_{sc} для $0^\circ < \theta < 90^\circ$.

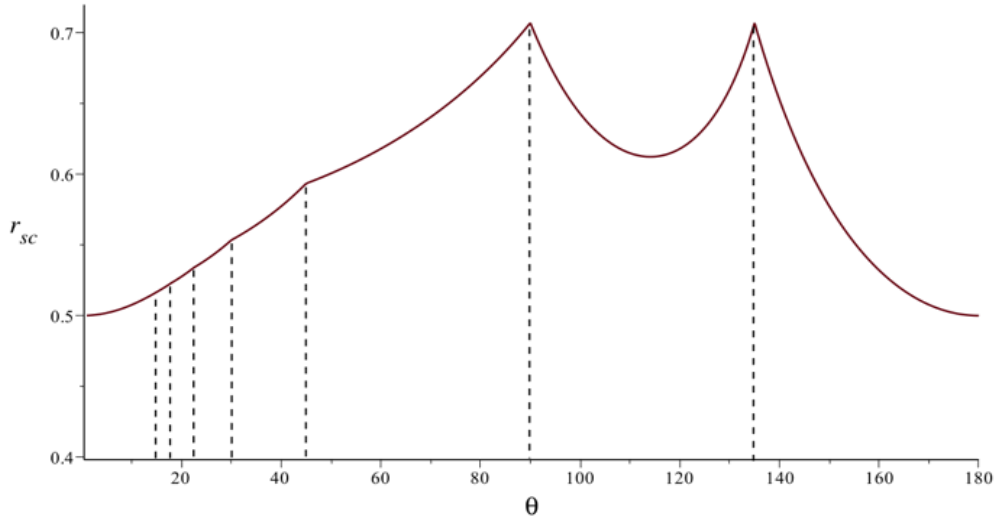
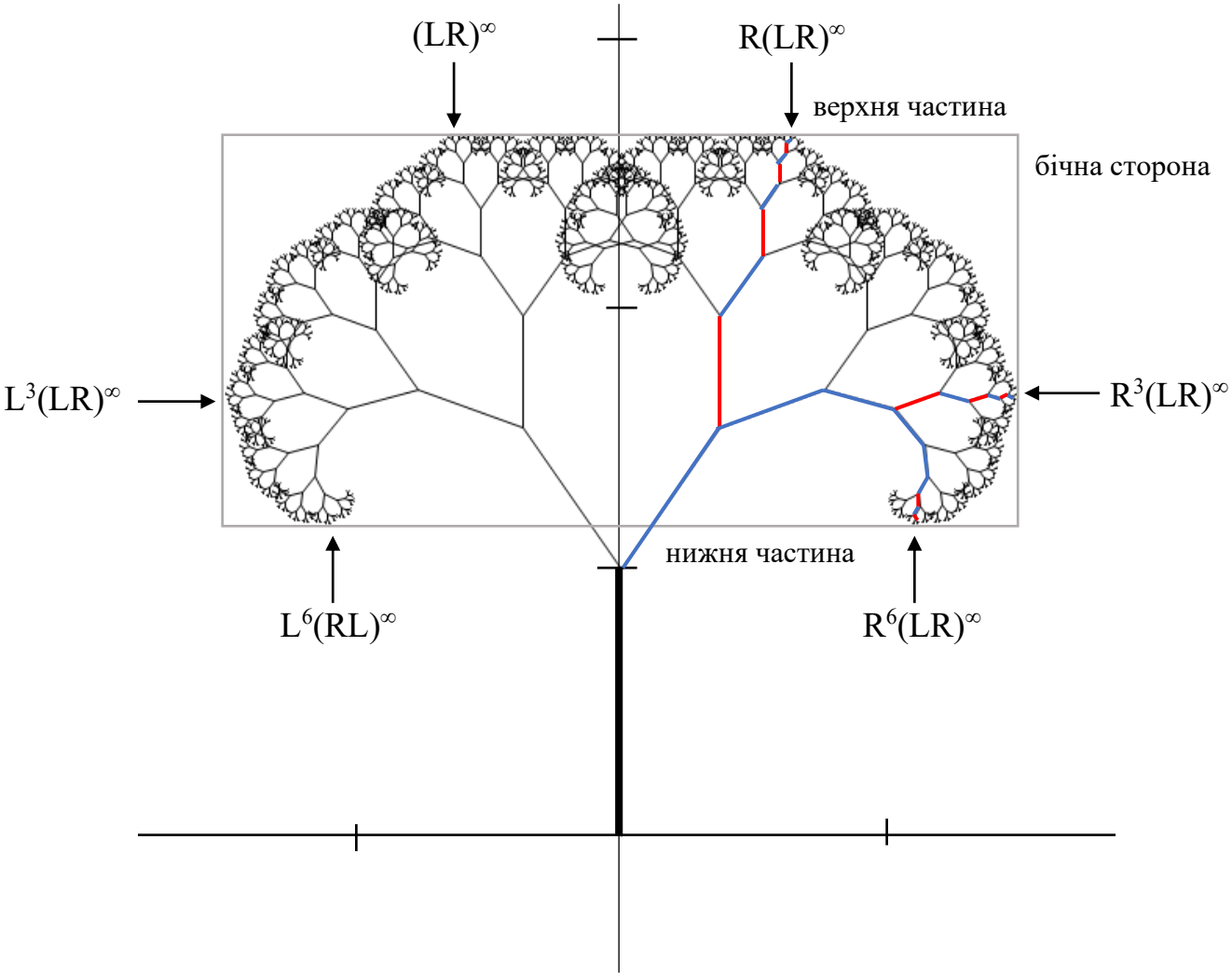


Рисунок 1.7 – Графік з максимальним значенням при θ = 90° і θ = 135°

Геометричні властивості симетричного бінарного дерева визначаються кутом θ і масштабним коефіцієнтом *r*: вважається, що основа дерева знаходиться на початку координат, а початковий вертикальний стовбур має довжину 1. Під "вершиною" дерева розуміється вершина вершин гілок, тобто найбільшу

y-координату вершин гілок. "Низом" дерева названо нижню частину вершин гілок, тобто найменшу *y*-координату вершин гілок. "Правою стороною" дерева буде найбільша *x*-координата вершин гілок (і за симетрією, ліва сторона буде від'ємним значенням цієї *x*-координати). Для більшості симетричних бінарних дерев вершина дерева буде відповідати *y*-координаті вершини гілки шляху $(LR)^\infty$, або еквівалентно $(RL)^\infty = R(LR)^\infty$; права частина буде відповідати *x*-координаті вершини шляху $R^k(LR)^\infty$, де k - найменше ціле число з $k\theta \geq 90^\circ$; а нижня частина буде відповідати *y*-координаті вершини шляху $R^k(LR)^\infty$, де k k - найменше ціле число з $k\theta \geq 180^\circ$.



θ = 35°, r = 0.6

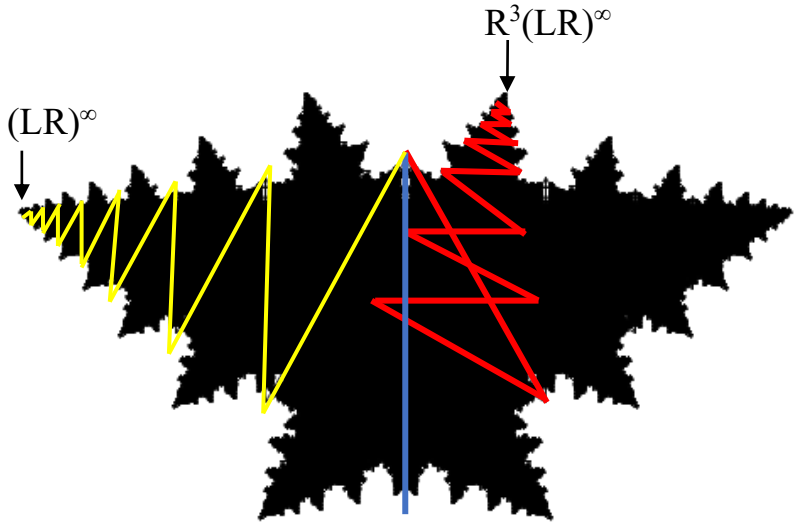
синій = права гілка, червоний = ліва гілка

верхня = 2.6539, бокова = 1.51964, нижня = 1.15588

Рисунок 1.8 - Геометричні Властивості дерева з заданим кутом θ і масштабним коефіцієнтом *r*

Однак, як зауважив Дон Вест для вершини дерева, верхівка гілки для $(LR)^{\infty}$ або $(RL)^{\infty}$ не завжди може мати найбільшу координату *y*. Дійсно, для кожного θ, яке не має вигляду $\frac{360°}{n}$ для деякого цілого n, існує критичне значення r_T, таке, що якщо $r > r_T$, то вершина гілки для шляху $R^k(LR)^{\infty}$, або, еквівалентно, $L^k(RL)^{\infty}$, матиме більшу координату по осі *y*, де k - найменше ціле число, при якому $k\theta \geq 360°$. Ці коефіцієнти масштабування будуть досить

великими, однак, і результуючі бінарні дерева матимуть багато гілок, що перекриваються.

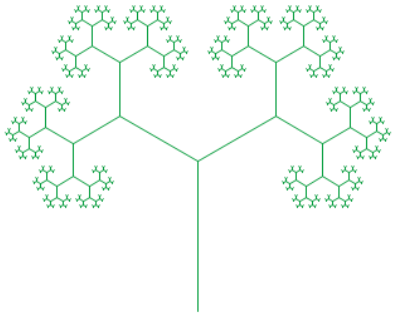


θ = 150°, r = 0.8

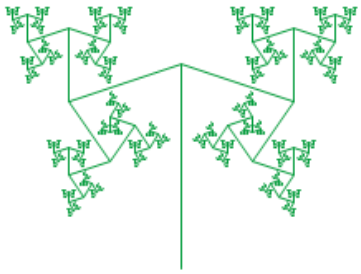
Рисунок 1.9 - Приклад гілок що перетинаються

Аналогічні винятки існують для дна та боків бінарного дерева для певних інтервалів θ. Однак жоден з цих винятків не матиме місця для самоконтактуючих дерев або коли $r < r_{sc}$.

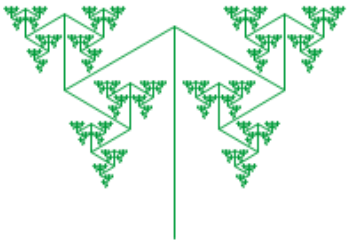
Дослідження самодотичних бінарних дерев із золотим перетином проводила Тара Тейлор, яка дослідила чотири самодотичних симетричних бінарних дерева, для яких коефіцієнт масштабування дорівнює $\frac{1}{\phi}$, де $\phi = \frac{1+\sqrt{5}}{2}$ це золотий перетин. Ці чотири дерева (Рис. 1.10) відповідають кутам 60°, 108°, 120° і 144°.



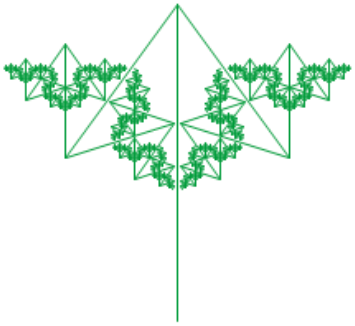
θ = 60°



θ = 108°



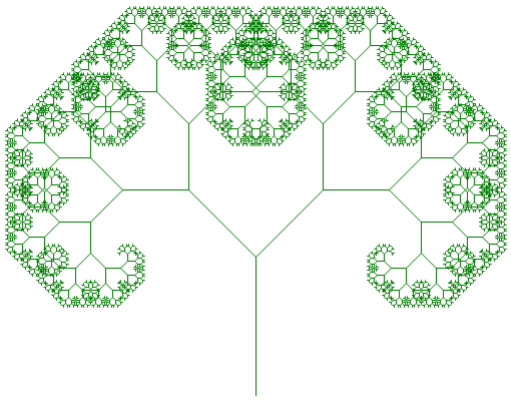
θ = 120°



θ = 144°

Рисунок 1.10 - Золотий перетин в при заданих кутах

Властивості симетричного бінарного дерева Леві включають утворення фрактала, подібного до дракона Леві, коли вершини гілок мають кути з θ = 45° і коефіцієнт масштабування $r = \frac{1}{\sqrt{2}}$.



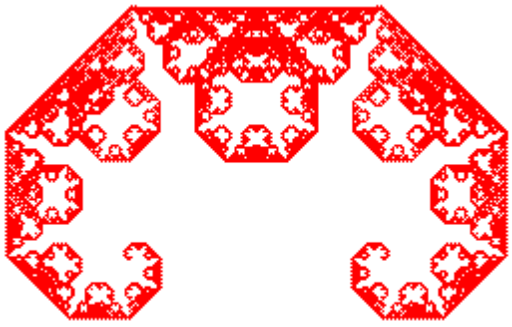


Рисунок 1.11 - Симетричне бінарне дерево та Леві Дракон

Якщо чотири копії симетричного бінарного дерева Леві розмістити під прямим кутом, то кінчики гілок утворять зовнішній гобелен Леві.

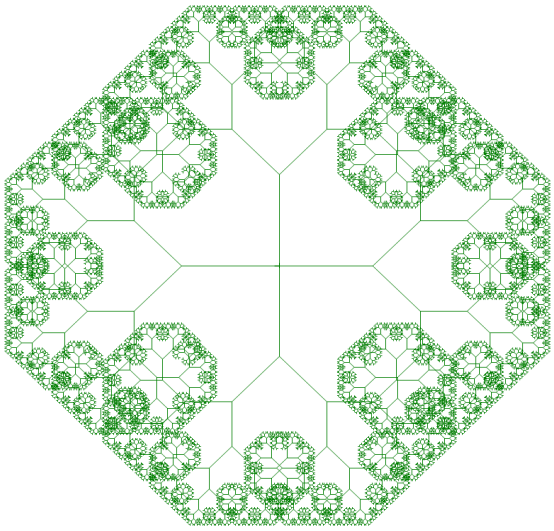


Рисунок 1.12 - Зовнішній гобелен Леві

1.2 Піфагорове Дерево

Історія та геометрія Піфагорового дерева пов'язана з іменем грецького математика Піфагора. Назва дерева походить від Піфагора через його внесок у геометрію та теорему, яка стала відома як теорема Піфагора. Ця конструкція ілюструє геометричне доведення теореми Піфагора, яка стверджує, що сума площ квадратів, побудованих на катетах прямокутного трикутника, дорівнює площі квадрата, побудованого на гіпотенузі. Піфагорове дерево, ймовірно, вперше намалював Альберт Е. Босман (1891-1961) близько 1942 року. Босман був голландським викладачем електротехніки та математики. Копія його першого (намальованого від руки!) ескізу дерева з прямокутними рівнобедреними трикутниками була розміром приблизно 33,5 на 23,5 дюйма. У 1957 році Босман опублікував книгу Het wondere onderzoekingsveld der vlakke meetkunde ("Дивовижне поле досліджень плоскої геометрії"), яка містила опис піфагорового дерева.

Босман зауважує, що кількість квадратів у дереві зростає як 1, 2, 4, 8, 16, 32, 64, 128, 512, 1024, 2048, ... і стверджує, що дивовижна структура піфагорійського дерева має межу з "гострими лініями і тонким мереживом". Він добре усвідомлював самоподібну природу своєї конструкції, оскільки зауважує, що кожен квадрат у дереві "можна розглядати як початковий квадрат, з якого розвивається дерево, так що воно складається з безлічі однорідних, все менших і менших дерев". Босмана також цікавило, "наскільки високим є це дерево, наскільки широкою є його крона вгорі і в середині, де знаходяться центри точок скручування і т.д.". Він використав дві діаграми, щоб допомогти дослідити деякі з цих властивостей.

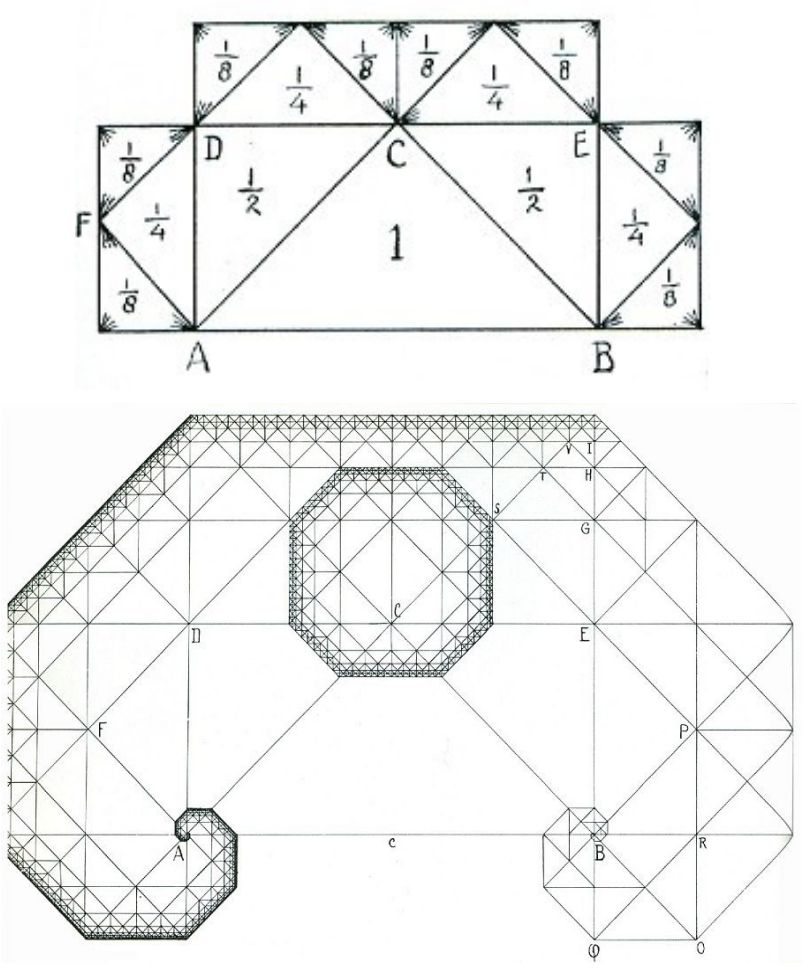


Рисунок 2.1 – Діаграми Босмана, базова конструкція дракона Леві

Босман зауважує, що нижній малюнок повторює верхній "до нескінченно малого". На малюнку (Рис. 2.1) також видно базову конструкцію дракона Леві, що починається з відрізка AB.

У своїй книзі Босман також демонструє розвиток дерева при використанні не рівнобедреного прямокутного трикутника з кутами 60° і 30°, розвиток, який Босман називає "дивовижним".

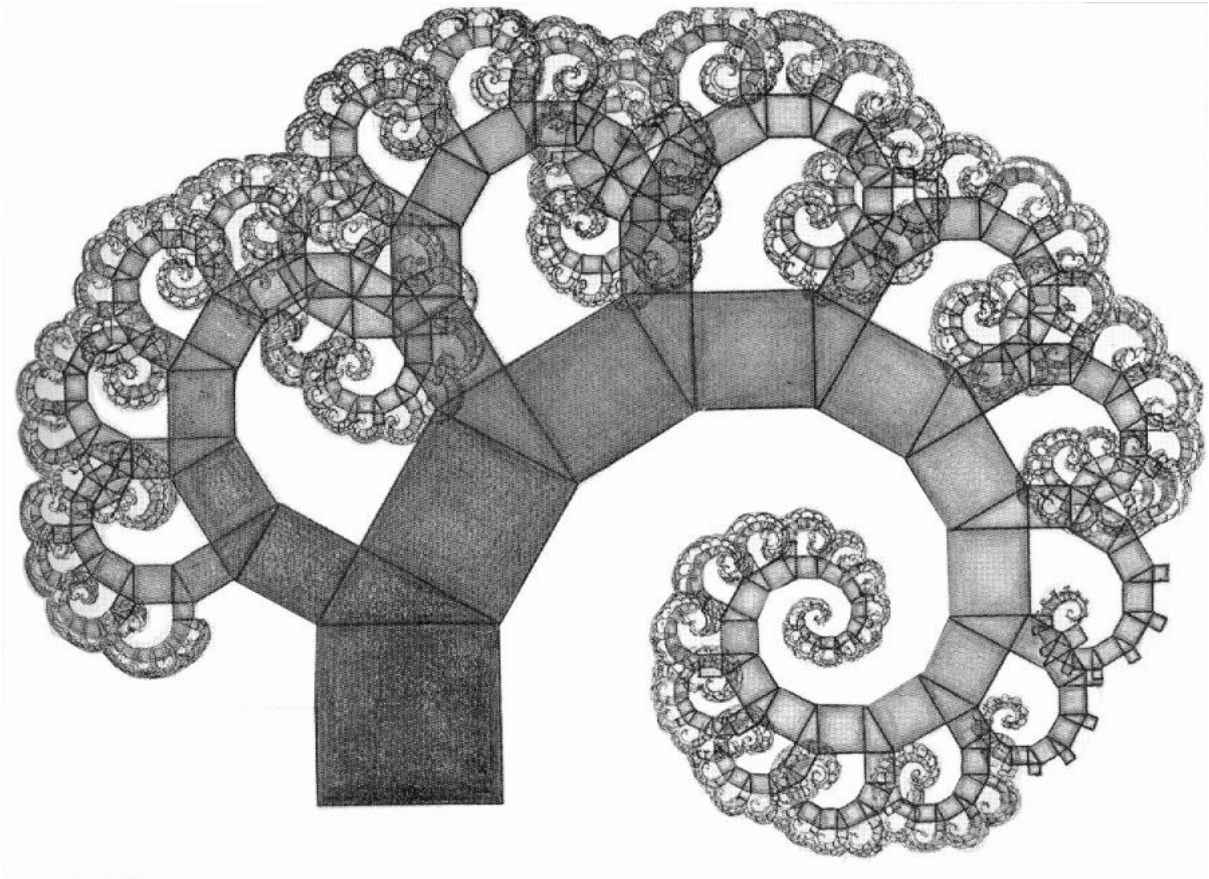


Рисунок 2.2 – Дивовижний розвиток

Він закінчує свій короткий розділ про піфагорове дерево, зазначаючи, що існують всілякі можливості.

Конструкція Піфагорового Дерева розпочинається з квадрата. Будується прямокутний рівнобедрений трикутник, гіпотенуза якого є верхнім краєм квадрата. Далі додаються квадрати вздовж двох інших сторін цього рівнобедреного трикутника.

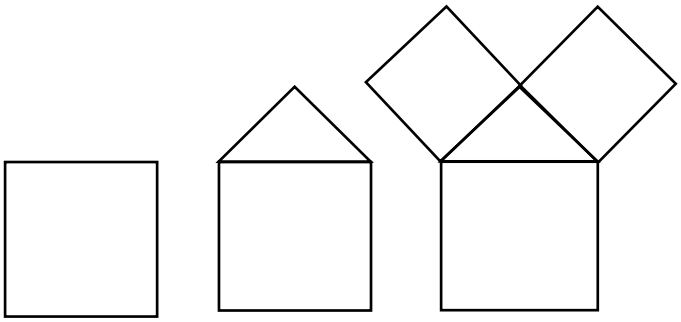


Рисунок 2.3 - Конструкція Піфагорового Дерева

Повторюється ця конструкція рекурсивно для кожного з двох нових квадратів.

Межу конструкції називається піфагорійським деревом (або деревом Піфагора).

Трикутники, які додаються до кожної гіпотенузи, можуть бути будь-якими прямокутними трикутниками з гострими кутами.

Конструкція Піфагорового дерева з використанням ітеративних функцій починається з одиничного квадрата з лівим нижнім кутом в початку координат. Нехай α лівий кут, як показано на (Рис. 2.4).

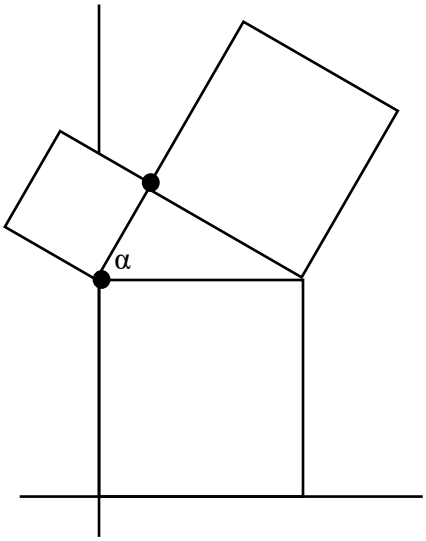


Рисунок 2.4 - Конструкція, коли лівий кут = α

Для першої функції, що відповідає лівому верхньому квадрату, повинно масштабуватися за cos(α) і повернутися проти годинникової стрілки на α, а

потім перевести прямо вгору. Для другої функції, що відповідає правому верхньому квадрату, повинно масштабуватися на sin(α) і повернутися за годинниковою стрілкою на 90°- α, а потім перевести точку на початку координат в точку під прямим кутом в трикутнику. Нарешті, третя функція - це просто тотожність. Вона зберігає вже намальовані квадрати в їх поточному розташуванні, в той час як перші дві функції додають додаткові квадрати.

$$f_1(x) = \begin{bmatrix} cos^2(\alpha) & -\cos(a)\sin(\alpha) \\ \cos(\alpha)\sin(\alpha) & cos^2(\alpha) \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (1.7)$$

$$f_2(x) = \begin{bmatrix} sin^2(\alpha) & \cos(a)\sin(\alpha) \\ -\cos(\alpha)\sin(\alpha) & sin^2(\alpha) \end{bmatrix} x + \begin{bmatrix} cos^2(\alpha) \\ 1 + \cos(\alpha)\sin(\alpha) \end{bmatrix} \quad (1.8)$$

$$f_3(x) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x \quad (1.9)$$

Зауважено, що функція (1.9) у цій ітераційній системі функцій не є контрактивним відображенням. Тому теорія стискання ітераційних систем функцій не обов'язково може бути застосована тут. Однак у цьому випадку ітерації початкового квадрата під цією ітераційною системою функцій сходяться до граничної множини. Насправді, ітерації будуть сходитися до граничної множини, починаючи з будь-якого початкового набору, навіть з неконтрактивною картою ідентичності як частиною IFS. Однак, на відміну від інших, набір обмежень буде залежати від початкового набору, який використовується. На зображенні (Рис. 2.5) ліворуч показано 10 ітерацій, починаючи з квадрата, з α = 45°. На зображенні (Рис. 2.5) праворуч показано 10 ітерацій, починаючи з вертикального відрізка.

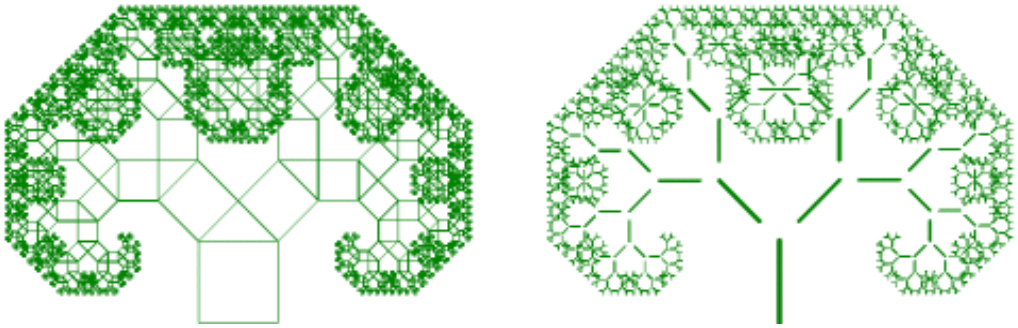


Рисунок 2.5 - 10 ітерацій, починаючи з квадрата, з α = 45°
та з вертикального відрізка

Цікавим є те, що хоча "стовбури" цих двох дерев відрізняються, "зовнішні листки" дерев виглядають дуже схожими. Це пояснюється тим, що якщо видалити третю функцію з ітераційної системи функцій, то дві інші функції, що залишилися, будуть стискатися і, таким чином, матимуть унікальний атрактор, незалежно від того, з якого початкового набору розпочати.

Насправді, функції (2.1) і (2.2) - це, по суті, ті ж самі функції, що і для дракона Леві, тільки з векторами трансляції, зсунутими на додаткову одиницю по вертикалі.

Якщо піфагорове дерево намалювати за допомогою рівнобедрених прямокутних трикутників (α = 45°) та почати з одиничного квадрата як початкової фігури, то дерево точно вписатиметься в прямокутник шириною 6 і висотою 4. Перші чотири ітерації квадрати не перетинатимуться, але після цього вони почнуть перетинатися і розростатися всередину, а також розширюватися назовні, створюючи ефект "листя" по всьому периметру дерева. Однак дерево завжди залишатиметься в межах цього прямокутника.

Припустимо, що пронумеровано квадрати, як показано на зображені (Рис. 2.6). Початковий квадрат позначено індексом 1. Якщо на клітинку з індексом n накласти прямокутний рівнобедрений трикутник, на якому побудовано дві нові клітинки, то нова клітинка зліва матиме індекс 2n, а нова

клітинка справа - індекс 2n+1. Особлива увага, що червоні квадрати мають парні індекси, а сині - непарні. На рисунку (Рис. 2.6) показано лише перші 4 ітерації. Після цього квадрати почнуть перекриватися, але алгоритм індексації все одно буде застосовуватися, оскільки буде виконано більше ітерацій. Після того, як квадрату присвоєно певний індекс, він ніколи не зміниться, поки не того, як квадрату присвоєно певний індекс, він ніколи не зміниться, поки не будуть створені додаткові квадрати.

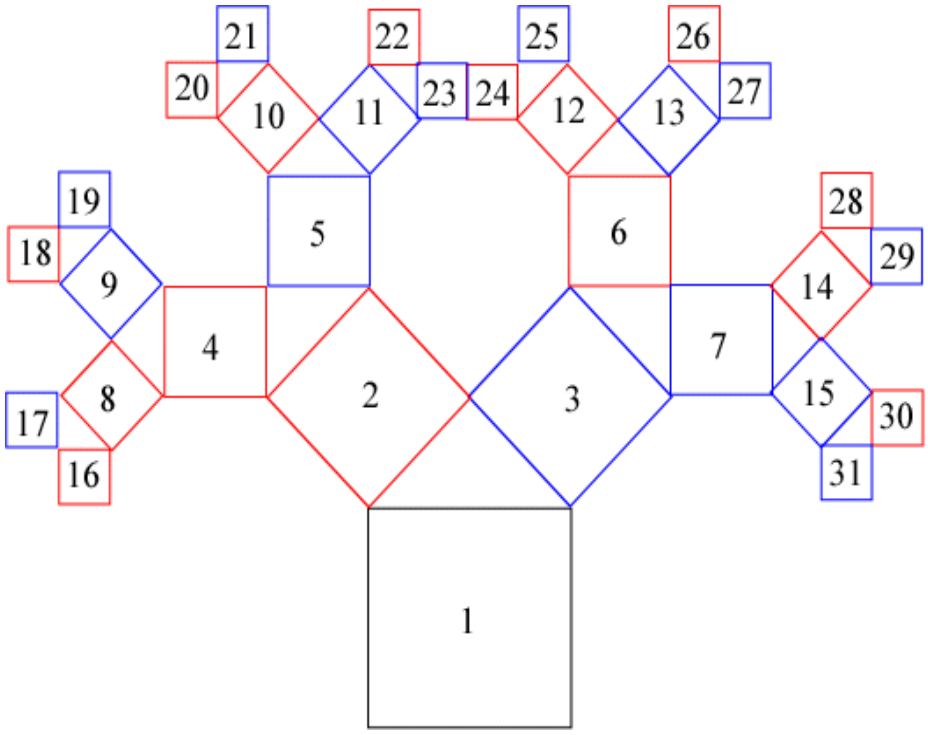


Рисунок 2.6 – Дерево з пронумерованими квадратами

Тепер можливо використовувати двійкову систему числення, щоб знайти будь-яку конкретну клітинку в дереві Піфагора. Наприклад, де знаходиться квадрат 45? Оскільки 45 = 32+8+4+1, то в двійковому записі він буде записаний як 101101. Крайня ліва цифра завжди буде 1 і відповідає базовому (початковому) квадрату. Для решти цифр 0 означає поворот ліворуч, а 1 - поворот праворуч. Таким чином, щоб потрапити до квадрата 45, потрібно повернути ліворуч, праворуч, праворуч, ліворуч, праворуч.

На зображені (Рис. 2.7) показані квадрати з індексами 1, 2, 4, 8, ..., тобто степенями 2. Ці квадрати утворюють логарифмічну спіраль, яка сходиться до

точки (точніше, порівнянні точки на кожному квадраті лежать на кривій, яка є логарифмічною спіраллю; на (Рис. 2.8) праворуч показана крива через кутові точки).

Але насправді немає нічого особливого в тому, щоб починати з початкового квадрата. Якщо починати з будь-якого квадрата в дереві Піфагора і рухатися від нього до квадратів, які завжди йдуть або вліво, або завжди вправо, то ці квадрати також будуть рухатися по логарифмічній спіралі. Таким чином, піфагорове дерево складається з нескінченної кількості спіралей.

1.3 Крива Гільберта

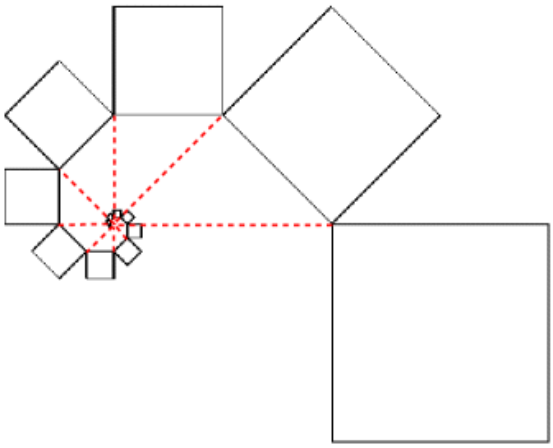


Рисунок 2.7 - Квадрати з індексами 1, 2, 4, 8, ..., тобто степенями 2

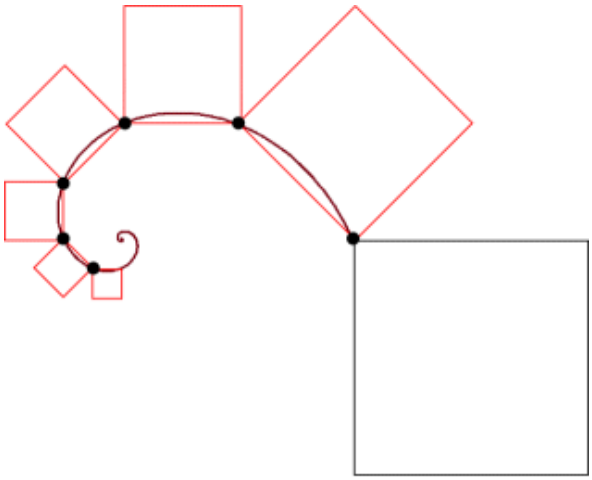


Рисунок 2.8 - Крива через кутові точки

Криві Гільберта вперше були описані у 1891 році німецьким математиком Давидом Гільбертом. Він розробив їх як приклад безперервної кривої, яка заповнює простір. Крива Гільберта є однією з найвідоміших і найчастіше досліджуваних кривих у теорії фракталів.

Крива Гільберта - це безперервна крива, що заповнює простір. Ці криві також є фракталами, вони самоподібні; якщо збільшити масштаб і уважно подивитеся на частину кривої вищого порядку, то можливо побачити, що вона

має такий самий вигляд, як сама крива. Крива Гільберта будується ітераційним методом, при якому кожна ітерація створює все більш детальну версію початкової кривої. Починається з простої лінії, яка при кожній наступній ітерації змінюється відповідно до заданих правил.

Для побудови Кривої Гільберта треба взяти квадрат зі стороною $\frac{1}{2}$, прибрати одну з його сторін і помістити його точно в центр одиничного квадрата.

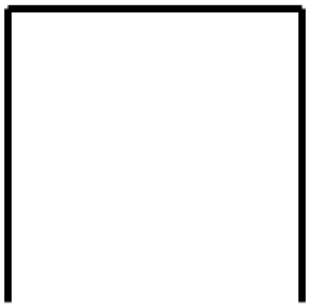


Рисунок 3.1 - Крива Гільберта першого порядку

Отримано криву Гільберта першого порядку. Тепер треба зменшити її рівно вдвічі і зробити з неї чотири копії. Дві треба перемістити, а дві інші також треба перемістити і повернути на чверть оберту в протилежні сторони. Треба з'єднати кінці ліній трьома однаковими відрізками, довжина яких дорівнює стороні нового, зменшеного квадрата

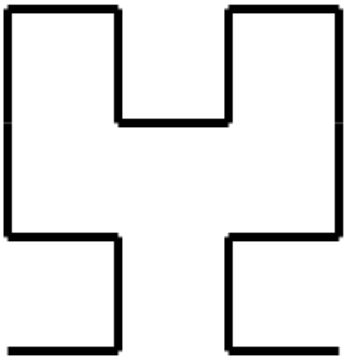


Рисунок 3.2 - Крива другого порядку

Вийшла крива другого порядку. Тепер треба повторити процедуру з отриманою ламаною лінією: треба зменшити вдвічі, зробити чотири копії, дві з яких повернуті, і знову треба з'єднати відрізками, які також треба укоротити вдвічі.

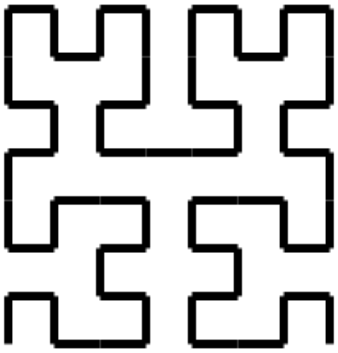


Рисунок 3.3 - Крива третього порядку

Повторювати цей алгоритм можна до нескінченності. Наступні чотири кроки (Рис. 3.4), з кривими порядку від 4 до 5 включно.

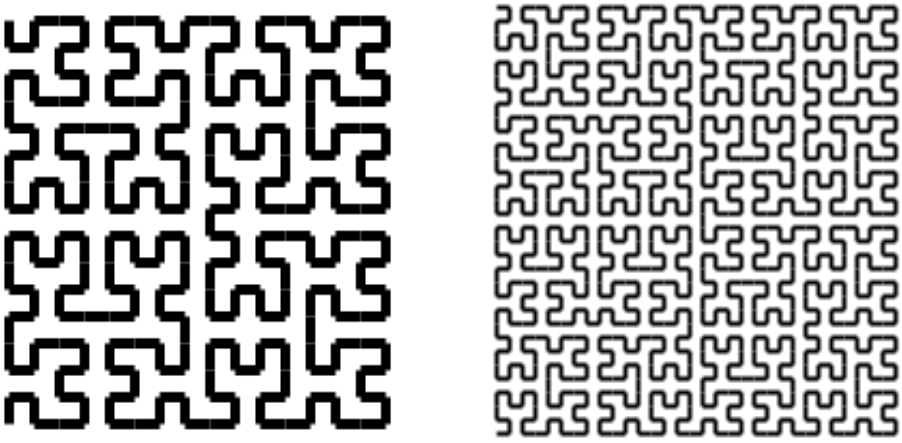


Рисунок 3.4 - Крива четвертого та п’ятого порядку

Крива Гільберта відома тим, що, якщо повторити нашу процедуру нескінченну кількість разів, то вона заповнить собою весь квадрат без проміжків: через будь-яку точку всередині або на межі квадрата вона проходитиме принаймні один раз! До роботи Гільберта математики припускали, що це неможливо.

Звісно, повторити алгоритм нескінченну кількість разів неможливо; говорячи про нескінченності, слід оперувати межами, ε-окрестностями та іншими поняттями математичного аналізу.

Доведення існування граничної кривої базується на міркуванні, що треба розмістити криву другого порядку (червона лінія середньої товщини) поверх кривої першого порядку (синя, найтовстіша лінія). Вочевидь, що перша наче "обвиває" останню. Ще більше це відношення помітно між кривими другого і третього порядку (тонка зелена лінія).

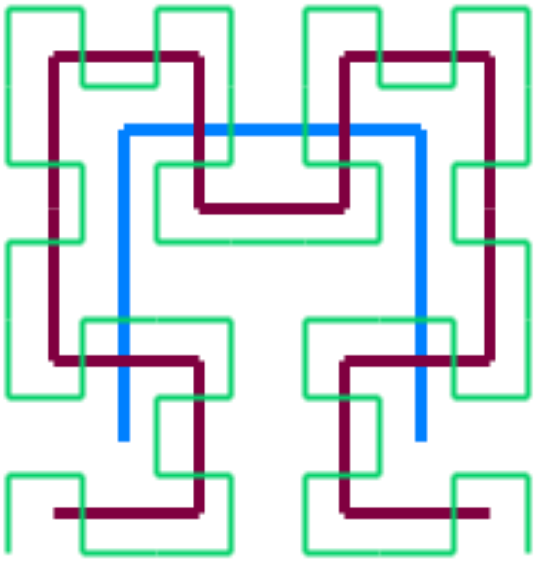


Рисунок 3.5 - Доведення існування граничної кривої

Треба взяти шматок нескінченно розтяжної резинки і нанести на неї шкалу від 0 до 1. Різні ділянки резинки можуть бути розтягнуті по-різному. Треба натягнути її за формою кривої першого порядку так, що кінці відповідатимуть 0 і 1, а середини кожної з трьох сторін — $\frac{1}{4}$, $\frac{2}{4}$ і $\frac{3}{4}$ відповідно. Тепер треба взяти ще одну таку ж резинку і натягнути її вздовж кривої другого порядку. Кінці кривої знову треба прикріпити в точках 0 і 1, а до середини кожного з 15 відрізків, що складають криву, треба прикріпити точки, починаючи з $\frac{1}{16}$ і закінчуючи $\frac{15}{16}$. Легко бачити, що відстань між точками, відповідними одному і тому ж числу на двох резинках, не перевищує діагоналі

квадрата зі стороною, рівною довжині кожного з відрізків кривої другого порядку.

Оскільки з кожною ітерацією сторона буде зменшуватися вдвічі, то встановлене нами "відхилення" однієї резинки від іншої зі збільшенням номера ітерації буде прагнути до нуля разом з довжиною відрізків, з яких складається крива. Важливим є те що, що кожна з резинок — це функція: якщо число на резинці виявилося в якійсь точці квадрата, то функція якраз і відображає це значення на точку в квадраті.

За теоремою Коші, оскільки різниця між будь-якими двома функціями прагне до нуля (транзитивність тут доводиться розкладанням у геометричний ряд), то і гранична функція існує і неперервна. Інтуїтивно це зрозуміло з малюнка: з кожною ітерацією відстань між кривими порядку n+1 і n, першою "обвиваючою" останню, стає все менше і менше.

Цікавим є питання, чи можна говорити про фрактальність цієї кривої. З одного боку, крива, безумовно, самоподібна, за винятком з'єднувальних відрізків, в силу побудови. З іншого боку, під Мандельбротове визначення фрактала вона не підпадає: оскільки крива заповнює весь квадрат, то її Хаусдорфова розмірність збігається з топологічною і дорівнює 2. Отже, відповідь залежить від того, що слід вважати фракталом. У певному сенсі, крива Гільберта знаходиться на межі між фрактальними і нефрактальними об'єктами.

Алгоритмічно криву Гільберта зручно будувати за допомогою переписувальної символьної системи Ліндeнмайера (L-System), кінцевий результат перетворення в якій треба інтерпретувати як "черепашачу" графіку.

.

1.4 Крива Госпера

Крива Госпера, відома також як крива Пеано-Госпера, названа ім'ям Біла Госпера, - це крива, що заповнює простір. Ця крива є фрактальною, подібною до кривих дракона і Гільберта.

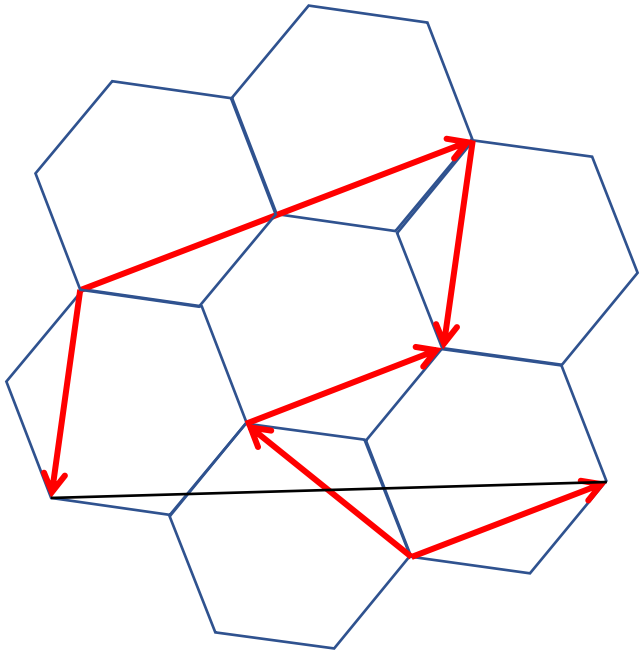


Рисунок 4.1 - Сім правильних шестикутників та основний червоний мотив

Починаючи з візерунка з семи правильних шестикутників. Вісім вершин з'єднайте, як показано на зображені (Рис. 4.1), щоб створити основний червоний мотив. Сегменти червоної лінії зорієнтовані стрілками. Тепер можливо змінити кожен відрізок червоної лінії копією мотиву в масштабі $r = \frac{1}{\sqrt{7}}$ таким чином, щоб копія лежала ліворуч від орієнтованого ребра, яке вона замінює. Важливо, що це помістить копію мотиву всередину шестикутника, що містить це ребро. Цю рекурсивну процедуру продовжуємо і в результаті отримуємо зображення, яке називається «змія Вільяма Госпера».

З шестикутниками і візерунком, розташованими, як показано вище (Рис. 4.1), отриманаступну IFS, генерація Змії Госпера здійснюється за допомогою IFS, де $A = \arcsin\left(\frac{\sqrt{3}}{2\sqrt{7}}\right) \approx 19.1066°$.

$$f_1(x) = \begin{bmatrix} -1/\sqrt{14} & 3/3\sqrt{14} \\ -3/3\sqrt{14} & -1/\sqrt{14} \end{bmatrix} x + \begin{bmatrix} 1/14 \\ 3\sqrt{3}/14 \end{bmatrix}$$, масштабувати на $1/\sqrt{7}$, повернути на A-120° (1.10)

$$f_2(x) = \begin{bmatrix} 5/14 & -\sqrt{3}/14 \\ \sqrt{3}/14 & 5/14 \end{bmatrix} x + \begin{bmatrix} 1/14 \\ 3\sqrt{3}/14 \end{bmatrix}$$, масштабувати на $1/\sqrt{7}$, повернути на A (1.11)

$$f_3(x) = \begin{bmatrix} 5/14 & -\sqrt{3}/14 \\ \sqrt{3}/14 & 5/14 \end{bmatrix} x + \begin{bmatrix} 3/7 \\ 2\sqrt{3}/7 \end{bmatrix}$$, масштабувати на $1/\sqrt{7}$, повернути на A (1.12)

$$f_4(x) = \begin{bmatrix} -1/\sqrt{14} & 3/3\sqrt{14} \\ -3/3\sqrt{14} & -1/\sqrt{14} \end{bmatrix} x + \begin{bmatrix} 11/14 \\ 5\sqrt{3}/14 \end{bmatrix}$$, масштабувати на $1/\sqrt{7}$, повернути на A-120° (1.13)

L-система (Lindenmayer система) — це формальна граматика, яка використовується для моделювання розвитку рослинних структур і генерації фракталів. Вона складається з алфавіту символів, аксіоми (початкового стану) і правил переписування (продукцій).

Компоненти L-системи для кривої Госпера:

a. Алфавіт: F, +, -
   - F: рух вперед на фіксовану відстань, залишаючи слід.
   - +: поворот на кут +60 градусів (за годинниковою стрілкою).
   - -: поворот на кут -60 градусів (проти годинникової стрілки).
b. Аксіома (початковий стан): F
c. Правила переписування.

Графічна інтерпретація. Для візуалізації кривої Госпера необхідно інтерпретувати символи ланцюжка як команди для графічного черепашачого графіка (turtle graphics):

a. F: рух черепашки вперед, малюючи лінію.
b. +: поворот черепашки за годинниковою стрілкою на 60 градусів.
c. -: поворот черепашки проти годинникової стрілки на 60 градусів.

Наступна L-система генерує криву Пеано-Госпера, криву заповнення простору, граничним набором якої є змійка потоку.

Кут 60°

Аксіома F

F→F−G−−G+F++FF+G−

G→+F−GG−−G−F++F+G

Межа може бути отримана за допомогою наступної L-системи:

Кут 60°

Аксіома F+F+F+F+F+F+F+F

F → F-F+F

Початковий напрямок для L-системи для границі ітерації n можна взяти рівним 90°+n * A, де $A = \arcsin\left(\frac{\sqrt{3}}{2\sqrt{7}}\right) \approx 19.1066°$.

Змійка потоку є самоподібною з 7 копіями, що не перетинаються, кожна з яких масштабується з коефіцієнтом $r < 1$. Тому розмірність подібності, d, атрактора IFS є розв'язком рівняння

$$\sum_{k=1}^{7} r^d = 1 \Rightarrow d = \frac{\log(1/7)}{\log(r)} = \frac{\log(1/7)}{\log(1/\sqrt{7})} = 2 \qquad (1.14)$$

Межа змії потоків формується з шести частин, кожна з яких є самоподібною з 3 непересічними копіями самої себе, кожна з яких масштабується з коефіцієнтом $\frac{1}{\sqrt{7}}$. Тому границя має фрактальну розмірність:

$$\frac{\log(1/3)}{\log(1/\sqrt{7})} = \frac{\log(3)}{\log(1/\sqrt{7})} = 1.12915 \qquad (1.15)$$

Змійка також відома як острів Госпера, назва, яку вперше використав Мандельброт. Крива заповнення простору, яка заповнює змійку, відома як крива Пеано-Госпера. Те, що ми описуємо тут, є дзеркальним відображенням опису Мартіна Гарднера в його статті в Scientific American.

Площа змійки дорівнює площі початкового шестикутника. Як показано на рисунку (Рис. 4.3), на першому етапі побудови границі нові відрізки зигзагоподібної лінії вздовж кожного краю початкового шестикутника мають стільки ж площі, скільки додається ззовні шестикутника, скільки забирається зсередини шестикутника. Таким чином, перша ітерація має ту ж саму площу, що і шестикутник. На кожному кроці ітеративної побудови кількість ребр продовжуватиме збільшуватись, але баланс додавання та віднімання площ залишатиметься незмінним, а отже, загальна площа ніколи не зміниться. У кінцевому підсумку площа, обмежена межею, буде точно такою ж, як і площа початкового шестикутника.

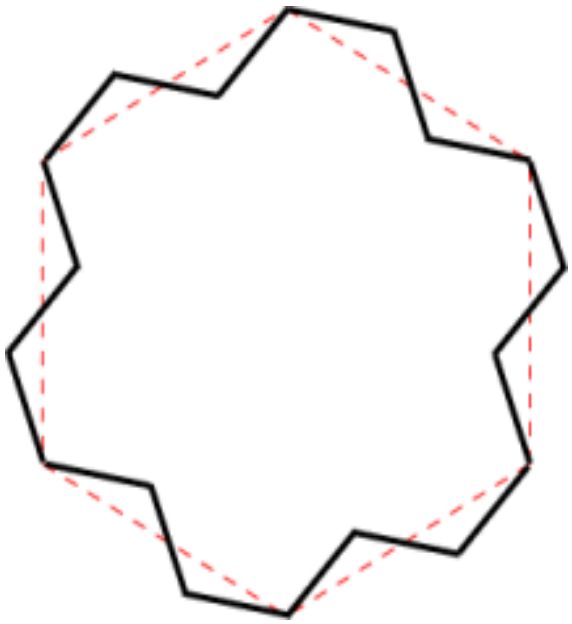


Рисунок 4.3 - Площа змійки дорівнює площі початкового шестикутника

2 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ — ФРАКТАЛЬНОГО РЕДАКТОРА З РОЗШИРЕНИМ ФУНКЦІОНАЛОМ

2.1 Платформа для розробки системи дослідження

У дипломній роботі для створення фрактального редактора з розширеним функціоналом було обрано платформу .NET Framework 4.8. Программа розроблялась у середовищі Visual Studio. Microsoft Visual Studio - це серія продуктів компанії Microsoft, які містять інтегроване середовище розробки (IDE) програмного забезпечення та низку інших інструментальних засобів. Основні можливості включають розробку консольних програм та програм з графічним інтерфейсом (Windows Forms). Система підтримує мови програмування C++, C# та Visual Basic.[1]

Windows Forms (або WinForms) – це бібліотека графічних класів, яка є частиною Microsoft .NET та .NET Framework. Вона забезпечує платформу для створення клієнтських додатків для настільних комп'ютерів, ноутбуків та планшетів на базі Windows. WinForms надає зручні інструменти для розробки інтерфейсу користувача, підтримує роботу з подіями та забезпечує багатий набір вбудованих елементів керування, таких як кнопки, текстові поля, списки та діалогові вікна. Крім того, вона дозволяє розробникам створювати власні елементи керування, легко інтегрується з іншими технологіями Microsoft та підтримує роботу з різними мовами програмування, зокрема C# та Visual Basic.

У дипломній роботі обрано мовою програмування C#. C# - це сучасна мова програмування, розроблена корпорацією Microsoft, що входить до платформи .NET. Вона відрізняється простотою використання, високою продуктивністю та надійністю. C# надає широкі можливості для розробки програмного забезпечення, включаючи об'єктно-орієнтоване програмування, сильну типізацію, збирання сміття, підтримку асинхронного програмування та високу інтеграцію з .NET Framework.[1]

Завдяки використанню цих технологій, у фрактальному редакторі реалізовані зручні та ефективні інструменти для створення, редагування та

візуалізації фракталів. Це забезпечує користувачам простий доступ до складних математичних моделей і надає можливість гнучко налаштовувати параметри для отримання бажаних результатів.

2.2 Декомпозиція системи фрактального редактора

Мета створення Інформаційної системи фрактального редактора полягає в розробці інструменту, спрямованого на створення, налаштування та візуалізацію фрактальних структур.

Ця система пропонує зручні та ефективні інструменти для творчого вияву користувачів, дозволяючи їм експериментувати з різноманітними фрактальними формами. Вона також є інструментом для вивчення основних принципів та властивостей фрактальних структур, сприяючи глибшому розумінню їхньої природи та математичного апарату.

Забезпечуючи можливість проведення досліджень у галузі фрактальної геометрії, система допомагає науковцям та дослідникам в аналізі даних та обміні результатами своїх досліджень.

Крім того, вона може використовуватися в освітніх цілях для викладання курсів з фрактальної геометрії та комп'ютерної графіки, сприяючи активному засвоєнню матеріалу студентами та практичному застосуванню отриманих знань. Таким чином, мета цієї системи полягає в наданні потужного інструменту для творчого вияву, дослідження та освіти в галузі фрактальної геометрії. Декомпозиція системи фрактального редактора передбачає поділ програмного забезпечення на окремі функціональні компоненти, кожен з яких виконує специфічні задачі. Це дозволяє структурувати систему та полегшити її розробку, тестування та підтримку.

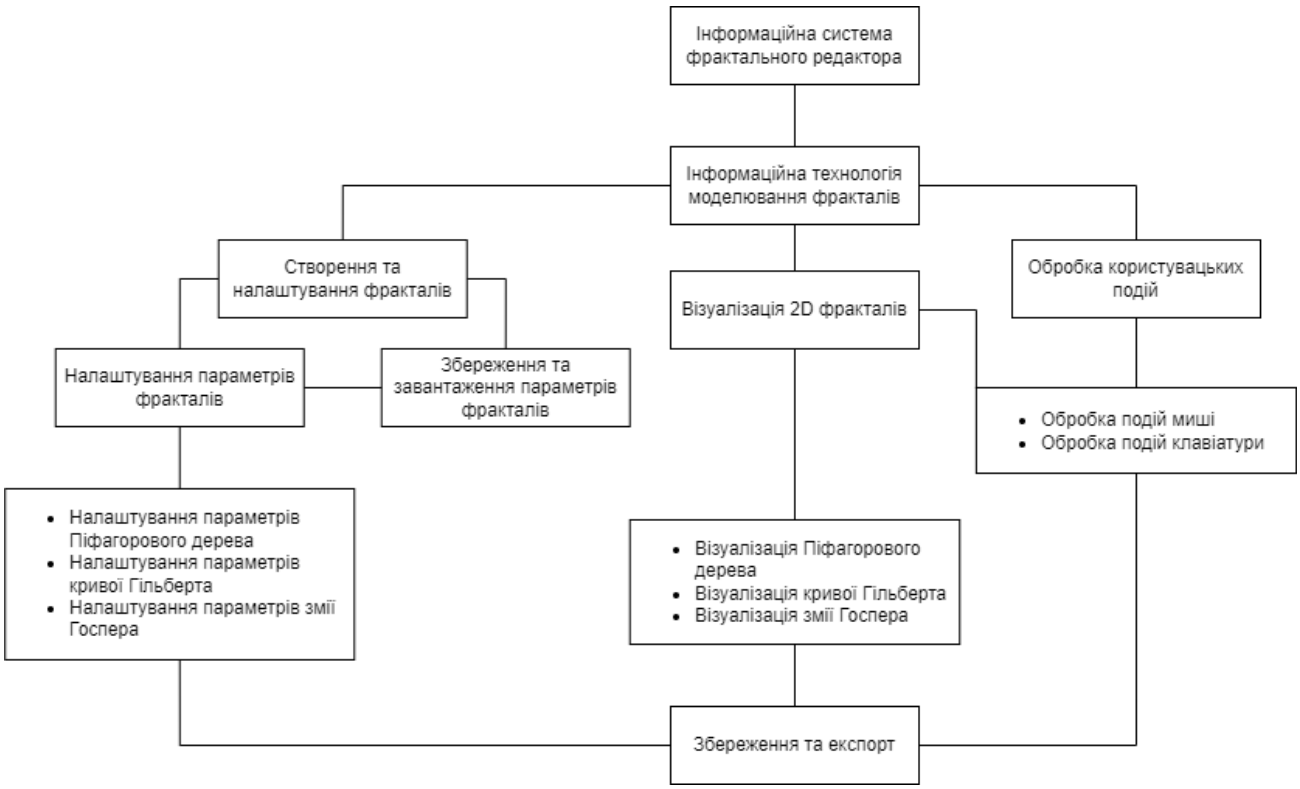


Рисунок 5.1 - Декомпозиція системи фрактального редактора

Інформаційна система фрактального редактора — це комплекс програмних засобів, призначених для створення, моделювання, візуалізації та збереження фрактальних структур. Основна мета цієї системи полягає в наданні користувачам можливості легко і ефективно працювати з різними типами фракталів, налаштовуючи їх параметри та отримуючи графічні зображення для подальшого використання.

Створення та налаштування фракталів у системі фрактального редактора забезпечує користувачам можливість створювати фрактальні структури та налаштовувати їхні параметри. Цей підрозділ складається з декількох основних функцій та підфункцій, що дозволяють точно керувати процесом генерації фракталів. Ось детальніше про кожну з них:

Налаштування параметрів фракталів: ця функція включає налаштування специфічних параметрів для різних типів фракталів. Користувачі можуть змінювати різні параметри для отримання бажаного вигляду та властивостей фрактальних структур.

Наприклад, Налаштування параметрів Піфагорового дерева. Для налаштування Піфагорового дерева користувачі можуть змінювати наступні параметри: Кути нахилу гілок: Встановлюються кути, під якими будуть розташовані квадрати та Масштабний коефіцієнт: Визначає розмір кожного наступного квадрата відносно попереднього.

2.3 Розробка інтерактивного інтерфейсу

У віконних додатках форма є графічним елементом, який відображає вікно програми, де користувач може взаємодіяти з програмою. Форма може мати різні розміри, форми та стилі, і вона є основною частиною інтерфейсу користувача. На формі можуть бути розміщені різні елементи, такі як кнопки, текстові поля, списки, таблиці та інші, які представляють собою елементи управління, призначені для взаємодії з користувачем. Ці елементи дозволяють користувачеві виконувати певні дії або вводити дані в програму, що робить їх важливою частиною будь-якого віконного додатка.[2]

Один із цікавих елементів управління - CollapsibleGroupBox, який комбінує в собі функціональність групи елементів та можливість її згортання або розгортання для зручності користувача. Зазвичай він представляє собою контейнер, який може містити інші елементи управління, такі як кнопки, поля введення тексту, списки тощо.

Головна перевага використання collapsibleGroupBox полягає в тому, що він дозволяє організувати інтерфейс програми таким чином, щоб згруповані елементи були логічно пов'язані між собою та одночасно мали можливість приховання або розкриття за потреби. Це особливо корисно, коли на формі присутні велика кількість елементів управління, і важливо забезпечити їхню структурованість та зручний доступ для користувача.

На головному екрані програмного продукту використовуються різноманітні елементи інтерфейсу для забезпечення зручної та ефективної взаємодії з користувачем. Серед цих елементів можна виділити[2]:

MenuStrip (стрічка меню): розташована зазвичай у верхній частині вікна програми та містить різні команди та опції, які дозволяють користувачеві взаємодіяти з програмою. Наприклад, меню може містити команди для відкриття файлів, збереження даних, налаштувань тощо.

ToolStripMenuItem (пункти меню): представляють собою конкретні команди чи опції у меню. При натисканні на пункт меню виконується певна дія, яка зазвичай відкриває нові вікна або виконує певні операції.

PictureBox (піктограма): використовується для відображення зображень або графіки на головному екрані програми. Цей елемент може відтворювати графічні зображення, такі як фрактали або інші результати роботи програми.

Label (мітка): використовується для відображення текстової інформації на екрані. Мітки зазвичай містять назви елементів або інструкції для користувача.

ComboBox (розкривне меню): дозволяє користувачу вибирати один елемент зі списку доступних варіантів. Використовується для вибору різних параметрів, наприклад, кольору чи розміру.

CheckBox (прапорець): дозволяє користувачу обирати один або декілька варіантів зі списку. Використовується для активації чи деактивації певних опцій або функцій програми.

RadioButton (радіокнопка): дозволяє користувачу обрати один з кількох варіантів. Використовується для вибору режимів роботи чи параметрів програми.

У контексті редагування фракталів ці елементи інтерфейсу використовуються для зміни параметрів фракталів, таких як розмір, колір, тип тощо. Наприклад, ToolStripMenuItem може використовуватися для вибору типу фрактала, а CheckBox – для зміни розміру поля у повноекранний режим.

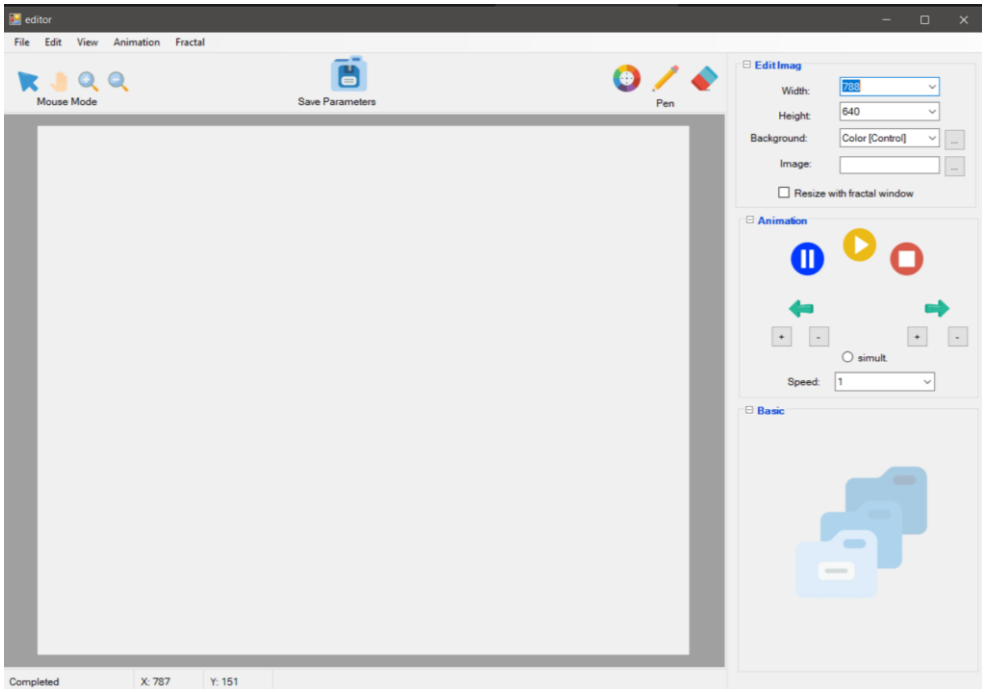


Рисунок 5.2 – Інтерфейс фрактального редактора

Для “collapsibleGroupBoxBasic”, який є частиною інтерфейсу програмного продукту, створено окремий клас “collapsibleGroupBoxBasicPanel”. Цей клас відповідає за динамічне створення різноманітних елементів інтерфейсу. У цьому класі оголошені поля для різних типів елементів, таких як `NumericUpDown`, `ComboBox`, та інші, які використовуються для взаємодії з користувачем.

2.4 Реалізація класів

У продовженні розділу про інтерфейс, елемент `collapsibleGroupBoxBasic` має свій власний клас під назвою `collapsibleGroupBoxBasicPanel`. Цей клас відповідає за динамічне створення елементів інтерфейсу для `collapsibleGroupBoxBasic` . У ньому оголошені публічні статичні поля для різних типів елементів, таких як `NumericUpDown`, `ComboBox` та інші. (див. Додаток А)

Метод `AllNumerikXYatreeLength` створює числові поля для введення координат та довжини сторони дерева. Метод `AllNumerik12` відповідає за створення додаткових числових полів. Метод `NumberAnew` створює нове

числове поле для введення кута. Метод `ComBox` створює комбінований список для вибору значення 0 або 1.

Метод `whatcollapsibleGroupBoxBasicPanel` визначає, які елементи інтерфейсу будуть створені залежно від вибору користувача. Наприклад, якщо вибрано опцію "Додати гілки", будуть відображатися додаткові елементи для налаштування гілок. Якщо вибрано опцію "Крива Госпера", створюються відповідні елементи для налаштування цієї кривої.

У цьому класі використовуються елементи управління, такі як `NumericUpDown`, `ComboBox`, і т. д., для введення різних параметрів, необхідних для редагування фракталів у програмі. Діаграму цього класу представлено на класовій діаграмі (Рис. 5.3).

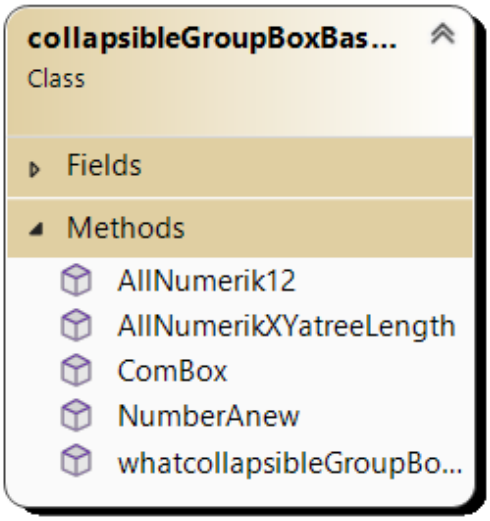


Рисунок 5.3 – Класова діаграма класу collapsibleGroupBoxBasic

Клас Pythagoras є ключовим компонентом для малювання фрактального дерева Піфагора. Він реалізує алгоритми, які дозволяють створювати це дерево за допомогою графічних об'єктів та параметрів, що налаштовуються користувачем. Клас призначений для графічного представлення фрактальних структур за допомогою методів рекурсії. Основною функцією класу є малювання дерева Піфагора на графічному полотні, яке задається користувачем через об'єкт Graphics. Для налаштування зовнішнього вигляду

дерева використовується об'єкт Pen, а параметри анімації та специфічні кути налаштовуються за допомогою числових значень.

Клас має кілька приватних полів, які зберігають важливу інформацію для процесу малювання: graphics — об'єкт для малювання на графічному полотні; pen — об'єкт, що визначає колір та стиль ліній, які будуть намальовані; animationFactor — коефіцієнт, що впливає на зменшення довжини гілок під час рекурсії; newToolStripMenuItemClicked — булеве значення, що визначає стан відповідної кнопки меню і впливає на алгоритм малювання (див. Додаток Б).

Конструктор класу приймає параметри, необхідні для ініціалізації полів. Це включає об'єкти Graphics і Pen, а також числові значення для анімації та стану кнопки меню. Ці параметри дозволяють налаштувати клас відповідно до вимог користувача. Метод DrawTree є головним методом цього класу і реалізує рекурсивний алгоритм малювання дерева. Він приймає координати початкової точки, довжину гілки, кути нахилу та інші параметри, необхідні для визначення форми та структури дерева. Метод перевіряє, чи довжина гілки перевищує певний мінімум, і, якщо так, то малює гілку та викликає себе рекурсивно для малювання дочірніх гілок.

Метод DrawTree використовує кілька параметрів для налаштування вигляду дерева: початкові координати (x і y), довжина гілки (length), кут нахилу гілки (angle), основний кут (baseAngle), кути для лівої та правої дочірніх гілок (leftChildAngle і rightChildAngle), кути для додаткових гілок (offset1Angle і offset2Angle). Метод також враховує стан кнопки меню newToolStripMenuItemClicked для визначення, чи потрібно малювати додаткові гілки з іншими кутами.

Клас Pythagoras забезпечує гнучкий та ефективний спосіб створення фрактальних дерев. Використання рекурсії та можливість налаштування різних параметрів дозволяє створювати складні й естетично привабливі графічні структури. Цей клас є важливим інструментом для дослідження

фрактальної геометрії та створення візуально цікавих зображень у рамках графічних додатків.

Клас Gilbert є спеціалізованим класом для малювання кривої Гільберта. Цей клас використовує рекурсивний алгоритм для створення фрактальної кривої, що заповнює площину, забезпечуючи візуально цікавий та математично важливий графічний об'єкт. Основні поля та методи класу дозволяють ефективно налаштовувати та виконувати процес малювання кривої.

Конструктор класу приймає параметри, необхідні для ініціалізації полів, зокрема посилання на об'єкти PictureBox та Graphics. Це дозволяє класу взаємодіяти з графічним інтерфейсом користувача, забезпечуючи зручний спосіб малювання на формі.

Метод Drawing відповідає за малювання лінії від поточних координат (LastX, LastY) до нових координат, визначених зміщеннями dx і dy. Після малювання метод оновлюється, додаючи зміщення (див. Додаток В).

Метод Hilbert реалізує рекурсивний алгоритм малювання кривої Гільберта. Він приймає параметри dep (глибина рекурсії), dx і dy (зміщення по координатах). В залежності від глибини рекурсії, метод викликає себе з новими параметрами, забезпечуючи побудову складної фрактальної кривої.

Клас Gilbert забезпечує потужний інструмент для створення та відображення кривих Гільберта. Використання рекурсії та гнучких параметрів дозволяє налаштовувати вигляд кривої та адаптувати процес малювання до конкретних вимог. Цей клас є важливим компонентом для дослідження фрактальної геометрії та створення візуально привабливих зображень у графічних додатках.

Клас Gosper призначений для малювання кривої Госпера, яка є ще одним прикладом фрактальної кривої. Цей клас забезпечує необхідні методи для побудови та відображення цієї кривої на графічному інтерфейсі користувача.

Метод Draw виконує основну логіку малювання кривої Госпера. Він приймає початкові координати x та y, довжину лінії l, кут u, глибину рекурсії

t та напрямок q. Якщо глибина рекурсії більше нуля, метод рекурсивно викликає себе з новими параметрами, що відповідають різним сегментам кривої. В іншому випадку, якщо глибина рекурсії дорівнює нулю, метод малює лінію за допомогою об'єкта Graphics.

Метод Paint служить допоміжною функцією для Draw. Він оновлює координати x та y на основі довжини та кута, а також викликає метод Draw для продовження малювання кривої.

Клас Gosper ефективно використовує рекурсивний підхід для створення складних фрактальних кривих (див. Додаток Г). . Використання об'єктів Graphics та Pen дозволяє налаштовувати вигляд кривої та інтегрувати процес малювання в графічний інтерфейс користувача. Це робить клас корисним інструментом для візуалізації та дослідження фрактальних геометричних структур.

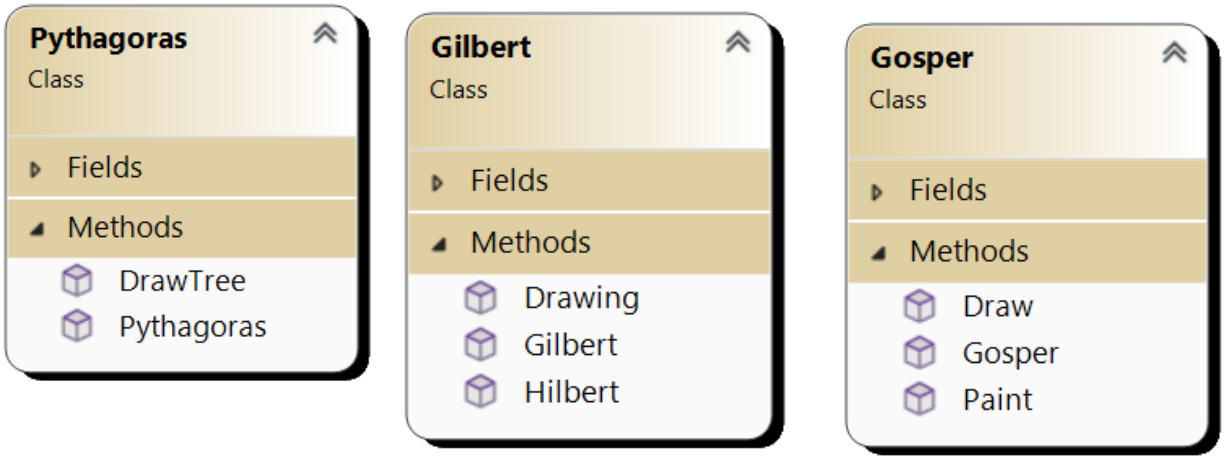


Рисунок 5.4 – Класові діаграми класів Pythagoras, Gilbert, Gosper

3 ДОСЛІДЖЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ ФРАКТАЛЬНОГО РЕДАКТОРА

При запуску програмного продукту відкривається сплеш-скрин, в якому відображені різні фрактальні дерева, створені в редакторі, а також надається коротка інформація про розробника та сам продукт. Після цього автоматично відкривається головне вікно редактора, в якому розгорнуті три секції, що відповідають за редагування полотна для відображення фракталів, анімацію та введення параметрів. Перші дві секції за замовченням заповнені необхідними параметрами, а третя секція залишається порожньою для введення нових даних.

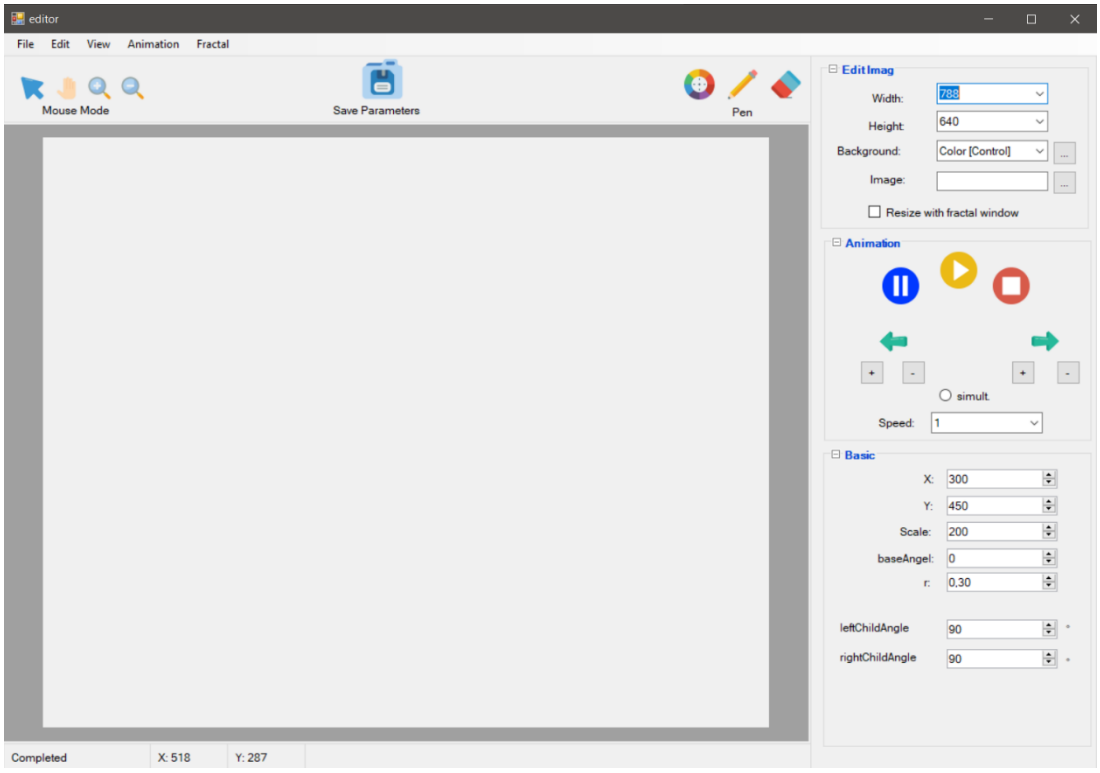


Рисунок 6.1 – Динамічне виведення потрібних параметрів у секції Basic

3.1 Дослідження фракталів дерева Піфагора

Для побудови першого фракталу необхідно вибрати потрібний об'єкт у меню, а саме в розділі Fractal. Наприклад, першим буде дерево Піфагора. Для нього в меню надається можливість вибору кількості гілок, які користувач

бажає відобразити. В якості прикладу обрано стандартне дерево з двома гілками. У секції Basic автоматично з’являються всі необхідні параметри, встановлені за замовченням. Однак, зображення ще не відображається. Тільки після натискання на іконку пензля фрактал відображається згідно з заданими параметрами. Для зручності, перші дві секції згорнуті.

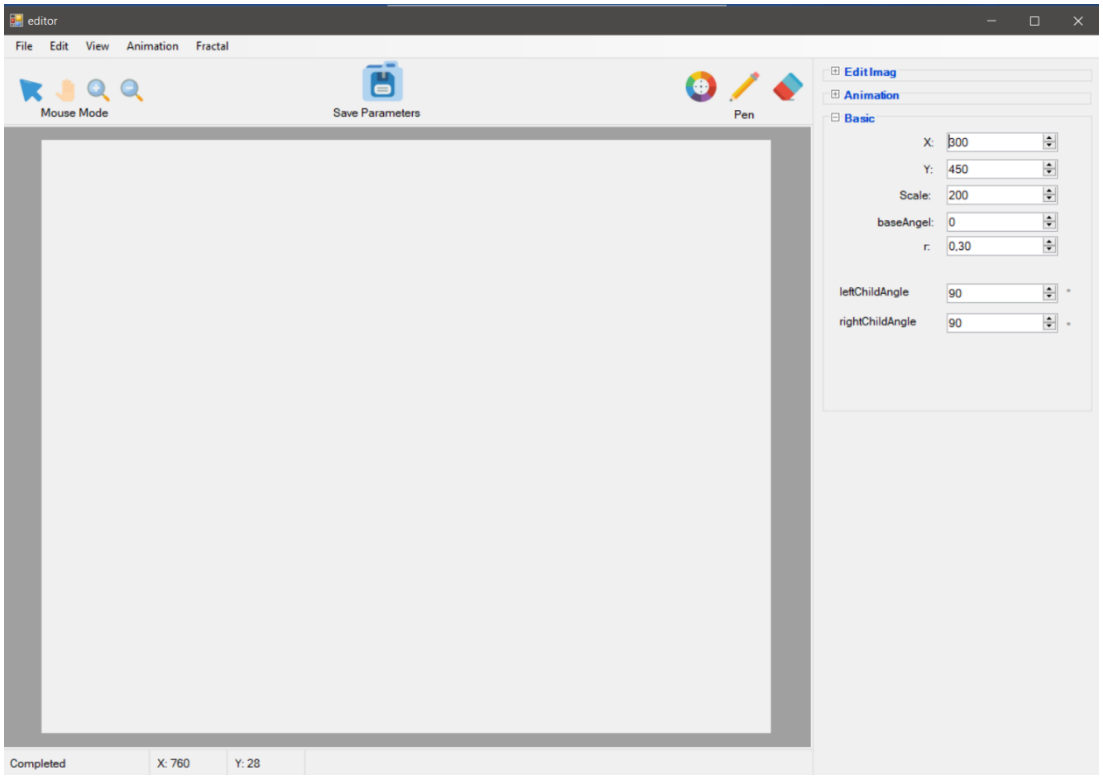


Рисунок 6.2 – Вивід дерева Піфагора за заданими параметрами

Отримане Дерево Піфагора (Рис. 6.2), коли обидві гілки знаходяться під кутом 90 градусів, утворює структуру, де кожна наступна "гілка" зростає перпендикулярно до попередньої. У цьому випадку дерево набуває форми хреста. У кожному вузлі гілки діляться під прямим кутом, створюючи симетричне розгалуження, яке на кожному рівні нагадує форму літер "Т або Н".

Або, якщо не враховувати основу дерева, утворюється Н-фрактал. Це підтверджує, що з Піфагорового дерева можна отримувати інші фрактальні

структури. Для чіткого візуалізування Н-фракталу змінено кількість ітерацій, наприклад, *a* = 0,65[3]. Результат представлено на малюнку (Рис. 6.3).

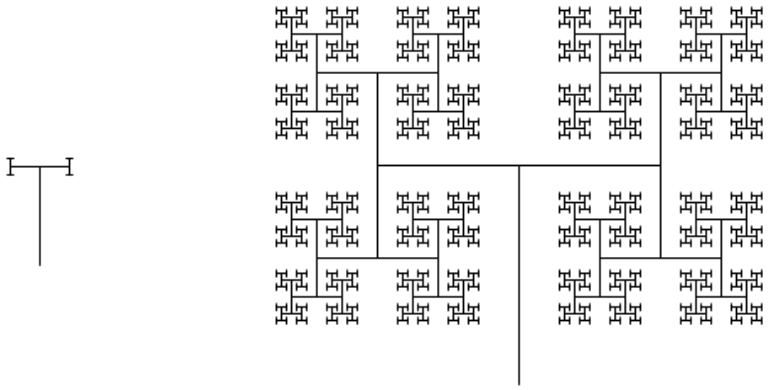


Рисунок 6.3 - Н-фрактал

Однак, при поверненні до стандартної конфігурації дерева Піфагора необхідно змінити кути нахилу гілок з 90° на 45°, залишаючи параметр, що відповідає за кількість ітерацій, незмінним, тобто a = 0,65. Отримане дерево видно на зображені (Рис. 6.4)

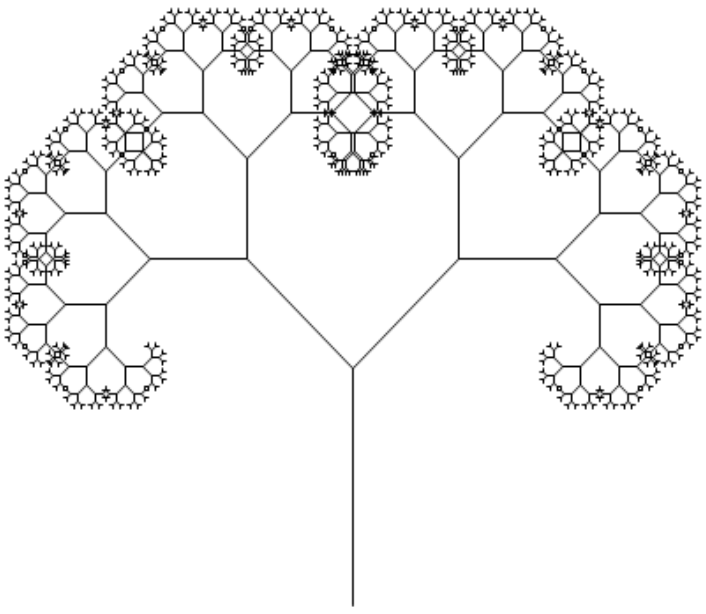


Рисунок 6.4 – Дерево Піфагора, кількість ітерацій 0,65

Після побудови Н-фракталу та стандартного дерева Піфагора можна провести їхню комбінацію, залишивши одну гілку під кутом 90°, а іншу повернувши в положення, коли вона мала кут нахилу 45°. Це призводить до формування спірального дерева Піфагора (Рис. 6.5), у якому під час кожної ітерації гілки збільшуються або зменшуються за розміром, утворюючи спіральну структуру (Рис. 6.6). Кількість ітерацій залишається без змін для збереження властивостей фракталу.

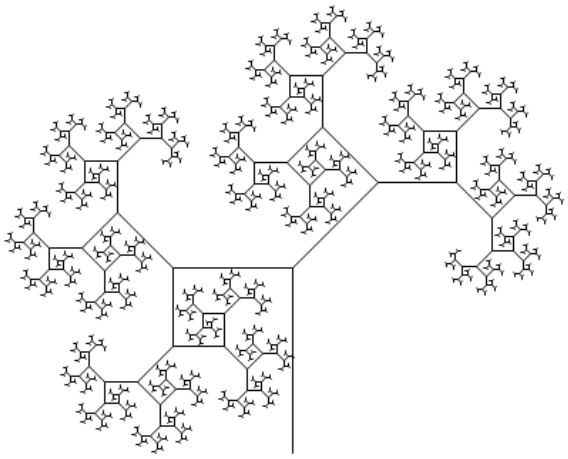


Рисунок 6.5 - Спіральне дерево Піфагора

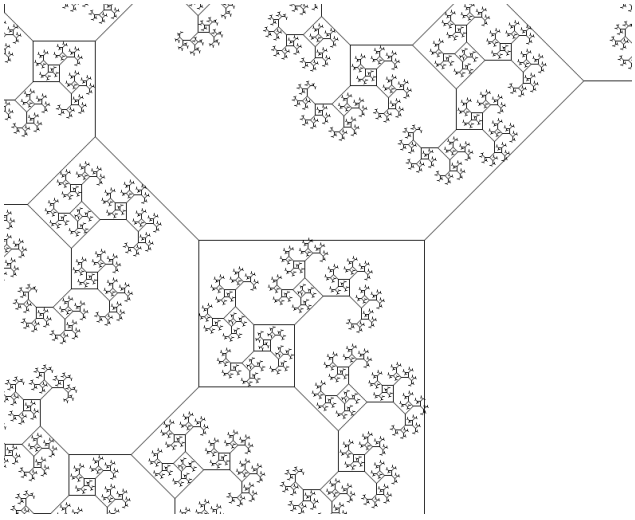


Рисунок 6.6 – Спіральна структура дерева Піфагора

Досліджено різні конфігурації дерева Піфагора, включаючи стандартну, спіральну та Н-фрактальну форми. Проведено детальний аналіз їхніх геометричних та топологічних властивостей. Отримані результати демонструють, що базове дерево Піфагора може бути модифіковане для створення широкого спектра фрактальних структур. Це підтверджує можливість формування складних і цікавих фрактальних конфігурацій з використанням лише основних принципів побудови дерева Піфагора.

3.2 Дослідження кривої Гільберта

Для дослідження кривої Гільберта користувач повинен вибрати відповідний прапорець у меню "Фрактал". Після цього секція "База" буде автоматично оновлена з відображенням лише необхідних параметрів. Після натискання на іконку пензля з'являється крива Гільберта, яка може представляти, наприклад, квадрат зі стороною $\frac{1}{2}$. Це відповідає одній ітерації, як показано на малюнку (Рис. 7.1)[4].

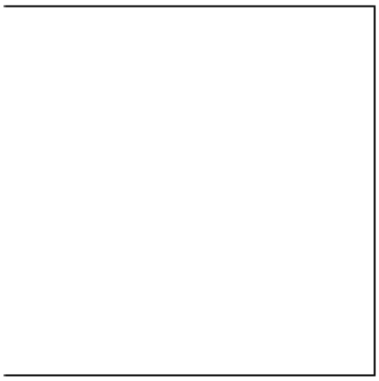


Рисунок 7.1 – Перша, одна ітерація кривої Гільберта

Кількість ітерацій змінено на дві, що відповідало отриманню кривої Гільберта другого порядку. Довжину зменшено удвічі, і створено чотири копії. Дві з них переміщено, а дві – також переміщено і повернено на чверть оберту в протилежних напрямках. Для з’єднання кінців ліній використовувалися три однакових відрізки, довжина яких дорівнювала стороні нового зменшеного квадрата. Застосовано рекурсивну функцію для реалізації процесу створення копій та їх переміщення (Рис. 7.2). Використання рекурсії дозволило досягти високої точності та забезпечити правильне розташування елементів.

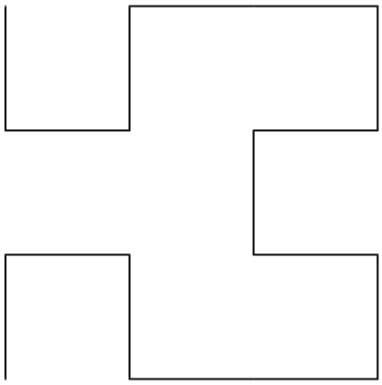


Рисунок 7.2 – Друга ітерація кривої Гільберта

Необхідно провести аналогічну процедуру з отриманою ламаною лінією: зменшити вдвічі, створити чотири копії, повернути дві з них, а потім знову з'єднати відрізками, які також потрібно скоротити вдвічі. Цей алгоритм можна повторювати нескінченно.[4]

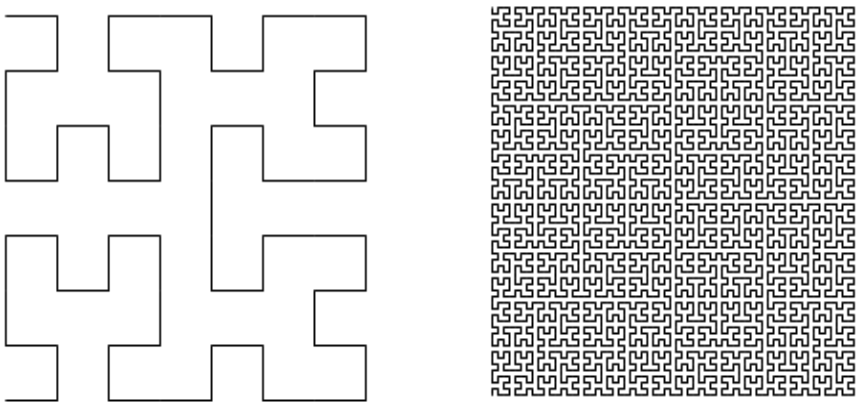


Рисунок 7.3 – Третя та Шоста ітерація кривої Гільберта

3.3 Дослідження кривої Госпера

Для дослідження кривої Госпера треба виконати такі ж самі маніпуляції щоб вивести криву на робоче поле, а саме вибір кривої Госпера у меню фрактального редактора, у секції "Basic" автоматично з’являються необхідні параметри, які можна налаштувати для створення кривої. Основними параметрами є кількість ітерацій та масштабний коефіцієнт. Після

налаштування параметрів користувач натискає кнопку генерації, і редактор будує криву Госпера на полотні[5].

А саме він будує нульову ітерацію, тобто пряму лінію, приклад побудови кривої Госпера за допомогою L-системи:

Нульова ітерація (аксіома): F

Це просто пряма лінія під кутом 60°.

Рисунок 8.1 – Нульова ітерація кривої Госпера

Перша ітерація (застосування правил до аксіоми): F-G--G+F++FF+G-

Тобто, якщо розбирати це правило частинами, то: **F**: Рух вперед і малювання лінії. **-**: Поворот проти годинникової стрілки на 60°. **G**: Рух вперед без малювання лінії. **--**: Поворот проти годинникової стрілки на 120° (два послідовних повороти по 60°). **G**: Рух вперед без малювання лінії. **+**: Поворот за годинниковою стрілкою на 60°. **F**: Рух вперед і малювання лінії. **++**: Поворот за годинниковою стрілкою на 120° (два послідовних повороти по 60°). **FF**: Два послідовних рухи вперед і малювання ліній. **+**: Поворот за годинниковою стрілкою на 60°. **G**: Рух вперед без малювання лінії. **-**: Поворот проти годинникової стрілки на 60°.

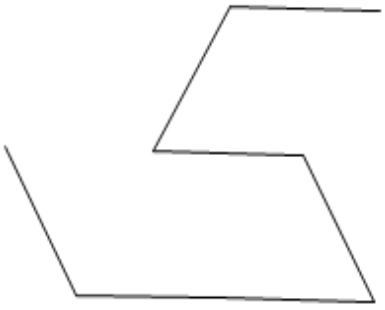


Рисунок 8.2 - Перша ітерація с застосуванням правил до аксіоми

Після застосування цього правила до кожного символу F на кожній ітерації, ми отримуємо складну криву (Рис. 8.2), яка з кожною наступною ітерацією стає все більш деталізованою і набуває характерного фрактального вигляду кривої Госпера.

Друга ітерація (застосування правил до кожного символу F і G з першої ітерації): F-G--G+F++FF+G- - (+F-GG--G-F++F+G) -- (+F-GG--G-F++F+G) + F-G--G+F++FF+G++FF+G-

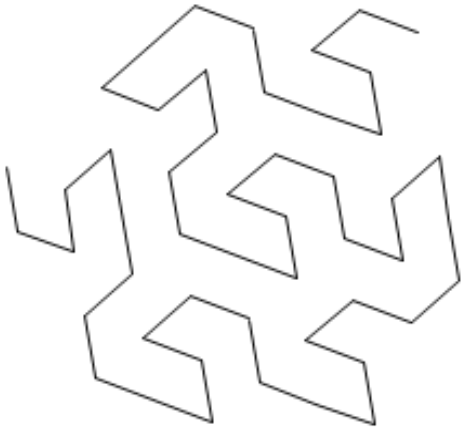


Рисунок 8.3 - Друга ітерація с застосуванням правил до кожного символу F і G з першої ітерації

Третя ітерація (застосування правил до кожного символу F і G з другої ітерації): З кожною ітерацією ланцюжок символів стає все довшим і складнішим, поступово формуючи фрактальну криву.

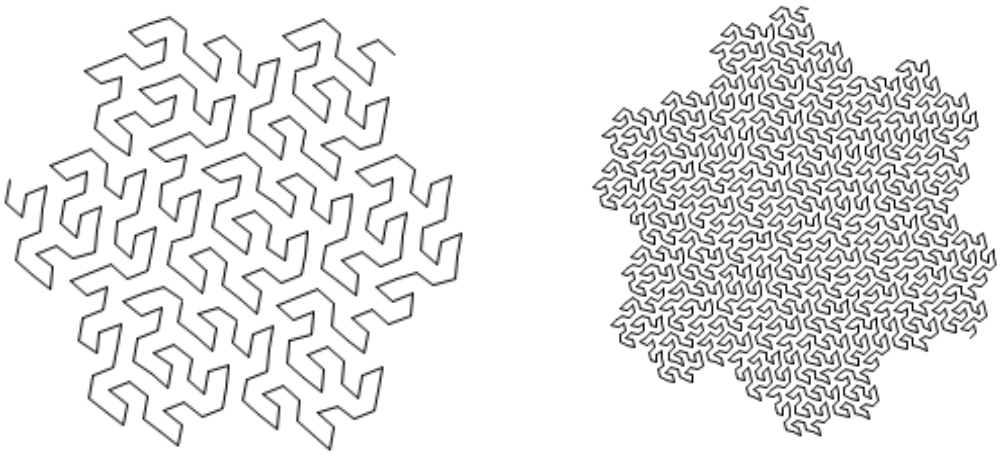


Рисунок 8.4 – Третя та Четверта ітерація кривої Госпера

ВИСНОВОК

В ході роботи було здійснено ґрунтовний аналіз теоретичних аспектів фрактальної геометрії та її застосування у моделюванні. Основну увагу приділено дослідженню і реалізації Піфагорового дерева, кривої Гільберта та змії Госпера. Було розроблено фрактальний редактор з розширеним функціоналом, який дозволяє користувачам створювати, налаштовувати та анімувати різноманітні фрактальні структури.

На етапі розробки програмного забезпечення було створено інтуїтивно зрозумілий інтерфейс користувача, який надає можливості для візуалізації фракталів у двомірному просторі з налаштуванням кольорів, розмірів та інших параметрів. Значна увага приділена забезпеченню зручності користування, що дозволяє навіть недосвідченим користувачам ефективно взаємодіяти з програмою.

Важливою частиною роботи стало впровадження анімаційних можливостей, які дають змогу спостерігати процес побудови фракталів у режимі реального часу, що є корисним як для навчальних цілей, так і для глибшого розуміння поведінки фрактальних структур.

Для забезпечення можливостей збереження та подальшого використання результатів роботи, було реалізовано функції експорту створених фракталів у різних графічних форматах, а також збереження налаштувань у власному форматі редактора.

Таким чином, у процесі виконання дипломної роботи було вирішено всі поставлені задачі, що дало змогу створити інструмент для моделювання та візуалізації фрактальних структур, який має високий потенціал для використання як у навчальних, так і в дослідницьких цілях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Щербаков О.В., Парфьонов Ю.Е., Федорченко В.М., Основи об'єктно-орієнтованого програмування: Навчальний посібник // Харків: ХНЕУ ім. С. Кузнеця, 2019. – 237 с.

2. Скиба О.П., Комп’ютерна графіка: конспект лекцій для студентів усіх форм навчання спеціальностей 122 «Комп’ютерні науки» та 123 «Комп’ютерна інженерія» з курсу «Комп’ютерна графіка» // Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2019. – 88 с.

3. Adventures into the Mathematical Forest of Fractal Trees – Wolfram Blog. Wolfram Blog: News, Views and Insights from Wolfram [Електронний ресурс]. Режим доступу: https://blog.wolfram.com/2014/05/22/adventures-into-the-mathematical-forest-of-fractal-trees/.

4. Taylor, Tara. "Excursions through a Forest of Golden Fractal Trees" // The Beauty of Fractals: 6 Different Views / eds. Denny Gulick and Jon Scott // Mathematical Association of America, 2010.

5. Mendler, Nick and Vincent J. Matsko. "Symmetric Binary Trees with Branching Ratios Larger than 1" // Proceedings of Bridges 2017: Mathematics, Art, Music, Architecture, Education, Culture. 507–510.

6. Gardner M. Penrose Tiles to Trapdoor Ciphers: And the Return of Dr Matrix // Washington, D.C: Mathematical Association of America, 1997. – 319 p.

ДОДАТОК А

Код динамічного модуля редагування фракталів

```
namespace PythagorasTree
{
    public class collapsibleGroupBoxBasicPanel
    {
        public static System.Windows.Forms.NumericUpDown numericUpDowntreeLength;
        public static System.Windows.Forms.NumericUpDown numericUpDownbaseAngle;
        public static System.Windows.Forms.NumericUpDown numericUpDownX;
        public static System.Windows.Forms.NumericUpDown numericUpDownY;
        public static System.Windows.Forms.NumericUpDown numeric_a;
        public static System.Windows.Forms.NumericUpDown numericUpDown1;
        public static System.Windows.Forms.NumericUpDown numericUpDown2;
        public static System.Windows.Forms.NumericUpDown numericUpDown3;
        public static System.Windows.Forms.NumericUpDown numericUpDown4;
        public static System.Windows.Forms.ComboBox comboBoxOneOrZero;

        public static void AllNumerikXYatreeLength(Control groupBox)
        {
            numericUpDownX = new System.Windows.Forms.NumericUpDown();
            numericUpDownX.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            numericUpDownX.Location = new System.Drawing.Point(134, 24);
            numericUpDownX.Maximum = new decimal(new int[] { 2000, 0, 0, 0 });
            numericUpDownX.Minimum = new decimal(new int[] { -2000, 0, 0, -2147483648 });
            numericUpDownX.Name = "numericUpDownX";
            numericUpDownX.Size = new System.Drawing.Size(120, 24);
            numericUpDownX.TabIndex = 3;
            numericUpDownX.Value = new decimal(new int[] { 300, 0, 0, 0 });
            groupBox.Controls.Add(numericUpDownX);

            numericUpDownY = new System.Windows.Forms.NumericUpDown();
            numericUpDownY.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            numericUpDownY.Location = new System.Drawing.Point(134, 53);
            numericUpDownY.Maximum = new decimal(new int[] { 2000, 0, 0, 0 });
            numericUpDownY.Minimum = new decimal(new int[] { -2000, 0, 0, -2147483648 });

            numericUpDownY.Name = "numericUpDownY";
            numericUpDownY.Size = new System.Drawing.Size(120, 24);
            numericUpDownY.TabIndex = 4;
            numericUpDownY.Value = new decimal(new int[] { 450, 0, 0, 0 });
            groupBox.Controls.Add(numericUpDownY);

            numericUpDowntreeLength = new System.Windows.Forms.NumericUpDown();
            numericUpDowntreeLength.Font = new System.Drawing.Font("Microsoft Sans Serif",
9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            numericUpDowntreeLength.Location = new System.Drawing.Point(134, 81);
            numericUpDowntreeLength.Maximum = new decimal(new int[] { 10000, 0, 0, 0 });
            numericUpDowntreeLength.Name = "numericUpDowntreeLength";
```

```
            numericUpDowntreeLength.Size = new System.Drawing.Size(120, 24);
            numericUpDowntreeLength.TabIndex = 14;
            numericUpDowntreeLength.Value = new decimal(new int[] { 200, 0, 0, 0 });
            groupBox.Controls.Add(numericUpDowntreeLength);

            numeric_a = new System.Windows.Forms.NumericUpDown();
            numeric_a.Cursor = System.Windows.Forms.Cursors.Default;
            numeric_a.DecimalPlaces = 2;
            numeric_a.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            numeric_a.Increment = new decimal(new int[] { 1, 0, 0, 131072 });
            numeric_a.Location = new System.Drawing.Point(134, 136);
            numeric_a.Maximum = new decimal(new int[] { 1000, 0, 0, 0 });
            numeric_a.Name = "numeric_a";
            numeric_a.Size = new System.Drawing.Size(120, 24);
            numeric_a.TabIndex = 6;
            numeric_a.ThousandsSeparator = true;
            numeric_a.Value = new decimal(new int[] { 3, 0, 0, 65536 });
            groupBox.Controls.Add(numeric_a);
        }
        public static void AllNumerik12(Control groupBox)
        {
            numericUpDown1 = new System.Windows.Forms.NumericUpDown();
            numericUpDown1.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            numericUpDown1.Location = new System.Drawing.Point(134, 187);
            numericUpDown1.Maximum = new decimal(new int[] { 10000, 0, 0, 0 });
            numericUpDown1.Name = "numericUpDown1";
            numericUpDown1.Size = new System.Drawing.Size(120, 24);
            numericUpDown1.TabIndex = 6;
            numericUpDown1.ThousandsSeparator = true;
            numericUpDown1.Value = new decimal(new int[] { 900, 0, 0, 65536 });
            groupBox.Controls.Add(numericUpDown1);

            numericUpDown2 = new System.Windows.Forms.NumericUpDown();
            numericUpDown2.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            numericUpDown2.Location = new System.Drawing.Point(134, 219);
            numericUpDown2.Maximum = new decimal(new int[] { 10000, 0, 0, 0 });
            numericUpDown2.Name = "numericUpDown2";
            numericUpDown2.Size = new System.Drawing.Size(120, 24);
            numericUpDown2.TabIndex = 18;
            numericUpDown2.ThousandsSeparator = true;
            numericUpDown2.Value = new decimal(new int[] { 900, 0, 0, 65536 });
            groupBox.Controls.Add(numericUpDown2);

            numericUpDown3 = new System.Windows.Forms.NumericUpDown();
            numericUpDown3.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            numericUpDown3.Location = new System.Drawing.Point(134, 249);
            numericUpDown3.Maximum = new decimal(new int[] { 10000, 0, 0, 0 });
            numericUpDown3.Name = "numericUpDown3";
            numericUpDown3.Size = new System.Drawing.Size(120, 24);
```

```
            numericUpDown3.TabIndex = 21;
            numericUpDown3.ThousandsSeparator = true;
            numericUpDown3.Value = new decimal(new int[] { 900, 0, 0, 65536 });
            groupBox.Controls.Add(numericUpDown3);

            numericUpDown4 = new System.Windows.Forms.NumericUpDown();
            numericUpDown4.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            numericUpDown4.Location = new System.Drawing.Point(134, 279);
            numericUpDown4.Maximum = new decimal(new int[] { 10000, 0, 0, 0 });
            numericUpDown4.Name = "numericUpDown4";
            numericUpDown4.Size = new System.Drawing.Size(120, 24);
            numericUpDown4.TabIndex = 22;
            numericUpDown4.ThousandsSeparator = true;
            numericUpDown4.Value = new decimal(new int[] { 900, 0, 0, 65536 });
            groupBox.Controls.Add(numericUpDown4);
        }
        public static void NumberAnew(Control groupBox)
        {
            numericUpDownbaseAngle = new System.Windows.Forms.NumericUpDown();
            numericUpDownbaseAngle.Font = new System.Drawing.Font("Microsoft Sans Serif",
9F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            numericUpDownbaseAngle.Location = new System.Drawing.Point(134, 110);
            numericUpDownbaseAngle.Maximum = new decimal(new int[] { 10000, 0, 0, 0 });
            numericUpDownbaseAngle.Name = "numericUpDownbaseAngle";
            numericUpDownbaseAngle.Size = new System.Drawing.Size(120, 24);
            numericUpDownbaseAngle.TabIndex = 18;
            numericUpDownbaseAngle.ThousandsSeparator = true;
            numericUpDownbaseAngle.Value = new decimal(new int[] { 000, 0, 0, 65536 });
            groupBox.Controls.Add(numericUpDownbaseAngle);
        }
        public static void whatcollapsibleGroupBoxBasicPanel(basicform form, Control
groupBox)
        {
            if (form.treeToolStripMenuItem.Checked && !form.addToolStripMenuItem.Checked)
            {
                var (labelX, labelY, labelScale, labelA, labelRight, labelLeft, label4,
label6, labelLeft2, labelRight2, labelCel1, labelCel2, labelBase) =
LabelManager.AddLabelToGroupBox(groupBox);
                LabelManager.UpdateLabelsTree(labelX, "X:", labelY, "Y:", labelScale,
"Scale:", labelA, "a:", labelRight, "rightChildAngle", labelLeft, "leftChildAngle", label4,
"°C", label6, "°C", labelBase, "baseAngel:");
                AllNumerikXYatreeLength(groupBox);
                AllNumerik12(groupBox);
                numericUpDown3.Visible = false;
                numericUpDown4.Visible = false;
                NumberAnew(groupBox);
            }
            else if (!form.treeToolStripMenuItem.Checked &&
form.addToolStripMenuItem.Checked)
            {
                MessageBox.Show("Ви можете додати ще гілки, тільки для Дерева Піфагора",
"Повідомлення", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

```
            }
            if (form.addToolStripMenuItem.Checked && form.treeToolStripMenuItem.Checked)
            {
                var (labelX, labelY, labelScale, labelA, labelRight, labelLeft, label4,
label6, labelLeft2, labelRight2, labelCel1, labelCel2, labelBase) =
LabelManager.AddLabelToGroupBox(groupBox);
                LabelManager.UpdateLabelsTree(labelX, "X:", labelY, "Y:", labelScale,
"Scale:", labelA, "a:", labelRight, "rightChildAngle", labelLeft, "leftChildAngle", label4,
"°C", label6, "°C", labelBase, "baseAngel:");
                LabelManager.UpdateLabelsTree4(labelLeft2, "leftChildAngle2", labelRight2,
"rightChildAngle2", labelCel1, "°C", labelCel2, "°C");
                AllNumerikXYatreeLength(groupBox);
                AllNumerik12(groupBox);
                NumberAnew(groupBox);

                numericUpDown3.Visible = true;
                numericUpDown4.Visible = true;
            }

            if (form.hilbertCurveToolStripMenuItem.Checked)
            {
                var (labelX, labelY, labelScale, labelA, labelRight, labelLeft, label4,
label6, labelLeft2, labelRight2, labelCel1, labelCel2, labelBase) =
LabelManager.AddLabelToGroupBox(groupBox);
                LabelManager.UpdateLabelsHilbert(labelX, "X:", labelY, "Y:", labelScale,
"Scale:", labelA, "a:");
                AllNumerikXYatreeLength(groupBox);
                numeric_a.Increment = 1.0m;
                numeric_a.Value = 1.0m;
            }
            if (form.gosperCurveToolStripMenuItem.Checked)
            {
                var (labelX, labelY, labelScale, labelA, labelRight, labelLeft, label4,
label6, labelLeft2, labelRight2, labelCel1, labelCel2, labelBase) =
LabelManager.AddLabelToGroupBox(groupBox);
                LabelManager.UpdateLabelsGopster(labelX, "X:", labelY, "Y:", labelScale,
"Scale:", labelA, "a:", labelRight, "Direction:", labelLeft, "Angle:");
                AllNumerikXYatreeLength(groupBox);
                AllNumerik12(groupBox);
                labelLeft.Location = new System.Drawing.Point(15, 187);
                numericUpDown2.Visible = false;
                numericUpDown3.Visible = false;
                numericUpDown4.Visible = false;
                ComBox(groupBox);
            }
        }
        public static void ComBox(Control groupBox)
        {
            comboBoxOneOrZero = new System.Windows.Forms.ComboBox();
            comboBoxOneOrZero.FormattingEnabled = true;
            comboBoxOneOrZero.Location = new System.Drawing.Point(134, 219);
            comboBoxOneOrZero.Name = "comboBoxOneOrZero";
            comboBoxOneOrZero.Size = new System.Drawing.Size(120, 24);
```

```
            comboBoxOneOrZero.TabIndex = 0;
            comboBoxOneOrZero.SelectedItem = "1";
            comboBoxOneOrZero.Items.AddRange(new object[] { "0", "1" });

            groupBox.Controls.Add(comboBoxOneOrZero);
        }
    }
}
```

ДОДАТОК Б

Код для малювання Дерева Піфагора

```
using System;
using System.Drawing;

namespace PythagorasTree
{
    public class Pythagoras
    {
        private Graphics graphics;
        private Pen pen;
        private double animationFactor;
        private bool newToolStripMenuItemClicked;

        public Pythagoras(Graphics graphics, Pen pen, double animationFactor, bool
newToolStripMenuItemClicked)
        {
            this.graphics = graphics;
            this.pen = pen;
            this.animationFactor = animationFactor;
            this.newToolStripMenuItemClicked = newToolStripMenuItemClicked;
        }

        public void DrawTree(double x, double y, double length, double angle, double baseAngle,
double leftChildAngle, double rightChildAngle, double offset1Angle, double offset2Angle, bool
newToolStripMenuItemClicked)
        {
            if (length > 2)
            {
                length *= animationFactor;
                double newX = Math.Round(x + length * Math.Cos(angle));
                double newY = Math.Round(y - length * Math.Sin(angle));

                graphics.DrawLine(pen, (float)x, (float)y, (float)newX, (float)newY);
                x = newX;
                y = newY;

                if (!newToolStripMenuItemClicked)
                {
                    DrawTree(x, y, length, angle + leftChildAngle, baseAngle, leftChildAngle,
rightChildAngle, offset1Angle, offset2Angle, newToolStripMenuItemClicked);
                    DrawTree(x, y, length, angle - rightChildAngle, baseAngle, leftChildAngle,
rightChildAngle, offset1Angle, offset2Angle, newToolStripMenuItemClicked);
                }
                else
                {
                    collapsibleGroupBoxBasicPanel.numeric_a.Maximum = 0.7m;

                    DrawTree(x, y, length, angle + leftChildAngle, baseAngle, leftChildAngle,
rightChildAngle, offset1Angle, offset2Angle, newToolStripMenuItemClicked);
                    DrawTree(x, y, length, angle - rightChildAngle, baseAngle, leftChildAngle,
rightChildAngle, offset1Angle, offset2Angle, newToolStripMenuItemClicked);
                    DrawTree(x, y, length, angle + offset1Angle, baseAngle, leftChildAngle,
rightChildAngle, offset1Angle, offset2Angle, newToolStripMenuItemClicked);
                    DrawTree(x, y, length, angle - offset2Angle, baseAngle, leftChildAngle,
rightChildAngle, offset1Angle, offset2Angle, newToolStripMenuItemClicked);
                }
            }
        }
    }
}
```

ДОДАТОК В

Код для малювання Гільбертової кривої

```
namespace PythagorasTree
{
    public class Gilbert
    {
        public float LastX, LastY;
        public Pen pen;
        public Bitmap HilbertImage;
        public PictureBox pictureBox1;
        public Graphics graphics;

        public Gilbert(PictureBox pictureBox, Graphics graphics)
        {
            pictureBox1 = pictureBox;
            this.graphics = graphics;
        }

        public void Drawing(float dx, float dy)
        {
            graphics.DrawLine(pen, LastX, LastY, LastX + dx, LastY + dy);
            LastX += dx;
            LastY += dy;
        }

        public void Hilbert(int dep, float dx, float dy)
        {
            if (dep > 1) Hilbert(dep - 1, dy, dx);
            Drawing(dx, dy);
            if (dep > 1) Hilbert(dep - 1, dx, dy);
            Drawing(dy, dx);
            if (dep > 1) Hilbert(dep - 1, dx, dy);
            Drawing(-dx, -dy);
            if (dep > 1) Hilbert(dep - 1, -dy, -dx);
        }
    }
}
```

ДОДАТОК Г

Код для малювання Госперової кривої

```
namespace PythagorasTree
{
    public class Gosper
    {
        public PictureBox pictureBox1;
        public Graphics graphics;
        public Pen pen;

        public Gosper(PictureBox pictureBox, Graphics graphics)
        {
            pictureBox1 = pictureBox;
            this.graphics = graphics;
        }

        public void Draw(double x, double y, double l, double u, int t, int q)
        {
            if (t > 0)
            {
                if (q == 1)
                {
                    x += l * Math.Cos(u);
                    y -= l * Math.Sin(u);
                    u += Math.PI;
                }
                u -= 2 * Math.PI / 19;
                l /= Math.Sqrt(7);
                Paint(ref x, ref y, l, u, t - 1, 0);
                Paint(ref x, ref y, l, u + Math.PI / 3, t - 1, 1);
                Paint(ref x, ref y, l, u + Math.PI, t - 1, 1);
                Paint(ref x, ref y, l, u + 2 * Math.PI / 3, t - 1, 0);
                Paint(ref x, ref y, l, u, t - 1, 0);
                Paint(ref x, ref y, l, u, t - 1, 0);
                Paint(ref x, ref y, l, u - Math.PI / 3, t - 1, 1);
            }
            else
            {
                if (graphics != null)
                {
                    graphics.DrawLine(pen, (float)Math.Round(x), (float)Math.Round(y),
(float)Math.Round(x + Math.Cos(u) * l), (float)Math.Round(y - Math.Sin(u) * l));
                }
            }
        }

        public void Paint(ref double x, ref double y, double l, double u, int t, int q)
        {
            Draw(x, y, l, u, t, q);
            x += l * Math.Cos(u);
```

```
            y -= l * Math.Sin(u);
        }
    }
}
```

ДОДАТОК Д

Код таймерів та основної форми

```
namespace PythagorasTree
{
    public partial class basicform : Form
    {
        public Graphics graphics;
        public Bitmap bitmap;
        public static Pen pen;
        public double penwight;
        private Image backgroundImage;

        private double x;
        private double y;
        private double length;
        private double angle;

        public double startX;
        public double startY;

        public double u;
        public double q;


        public Timer timer;
        public Timer windTimerLeft;
        public Timer windTimerLeftPlus;
        public Timer WindTimerLeftPlus { get { return windTimerLeftPlus; } }

        public Timer windTimerLeftMinus;
        public Timer WindTimerLeftMinus { get { return windTimerLeftMinus; } }


        public Timer windTimerRight;
        public Timer windTimerRightPlus;
        public Timer WindTimerRightPlus { get { return windTimerRightPlus; } }

        public Timer windTimerRightMinus;
        public Timer WindTimerRightMinus { get { return windTimerRightMinus; } }


        private bool isPaused = false;
        private double animationFactor;
        public double baseAngle;
        private double leftChildAngle;
        private double rightChildAngle;
        public double treeRootLength;
        public double offset1Angle;
        public double offset2Angle;
```

```
        public double someNewValueX;
        public double someNewValueY;

        public double previousNumericUpDownXValue;
        public double previousNumericUpDownYValue;
        public double previousNumericUpDowntreeLengthValue;
        public double previousNumeric_aValue;
        public double previousNumericUpDown1Value;
        public double previousNumericUpDown2Value;
        private collapsibleGroupBoxBasicPanel panelInstance;
        private System.Windows.Forms.GroupBox groupBox;
        public Gilbert gilbert;
        public Gosper gosper;

        public basicform()
        {
            InitializeComponent();
            InitializeTimer();
            colorDialog1 = new ColorDialog();
            gilbert = new Gilbert(pictureBox1, graphics);
            gosper = new Gosper(pictureBox1, graphics);
        }
        public void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (!MenuPanal.CloseApplicationBool)
            {
                if (e.CloseReason == CloseReason.UserClosing)
                {
                    if (MessageBox.Show("Ви впевнені, що хочете закрити?", "Confirmation",
MessageBoxButtons.YesNo) == DialogResult.No)
                    {
                        e.Cancel = true;
                    }
                    else
                    {
                        windTimerRightPlus.Stop();
                        windTimerRightMinus.Stop();
                        windTimerLeftPlus.Stop();
                        windTimerLeftMinus.Stop();
                        Application.Exit();
                    }
                }
            }
        }

        #region all timers
        private void InitializeTimer()
        {
            timer = new Timer();
            timer.Interval = 1000;
            timer.Tick += timer_a_Tick_1;
```

```
            windTimerLeftPlus = new Timer();
            windTimerLeftPlus.Tick += timer_a_Tick_windLeftPlus;

            windTimerLeftMinus = new Timer();
            windTimerLeftMinus.Tick += timer_a_Tick_windLeftMinus;

            windTimerRightPlus = new Timer();
            windTimerRightPlus.Tick += timer_a_Tick_windRightPlus;

            windTimerRightMinus = new Timer();
            windTimerRightMinus.Tick += timer_a_Tick_windRightMinus;
        }

        public void timer_a_Tick_1(object sender, EventArgs e)
        {
            if (treeToolStripMenuItem.Checked == true)
            {
                double                          currentValue                          =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numeric_a.Value);

                if (currentValue < 0.7)
                {
                    if (currentValue >= 0.50 && newToolStripMenuItemClicked)
                    {
                        timer.Stop();
                        simult.Checked = false;
                        return;
                    }

                    currentValue += 0.01;
                    collapsibleGroupBoxBasicPanel.numeric_a.Value = (decimal)currentValue;
                    UpdateImage();
                }
                else
                {
                    timer.Stop();
                    simult.Checked = false;
                }
            }
            if (hilbertCurveToolStripMenuItem.Checked == true)
            {
                double                          currentValue                          =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numeric_a.Value);

                if (currentValue < 20.0)
                {
                    if (currentValue >= 0.50 && newToolStripMenuItemClicked)
                    {
                        timer.Stop();
                        simult.Checked = false;
                        return;
                    }
```

```
                    currentValue += 0.10;
                    collapsibleGroupBoxBasicPanel.numeric_a.Value = (decimal)currentValue;
                    UpdateImage();
                }
                else
                {
                    timer.Stop();
                    simult.Checked = false;
                }

            }
            if (gosperCurveToolStripMenuItem.Checked == true)
            {
                double                          currentValue                          =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numeric_a.Value);

                if (currentValue < 20.0)
                {
                    if (currentValue >= 0.50 && newToolStripMenuItemClicked)
                    {
                        timer.Stop();
                        simult.Checked = false;
                        return;
                    }

                    currentValue += 1.0;
                    collapsibleGroupBoxBasicPanel.numeric_a.Value = (decimal)currentValue;
                    UpdateImage();
                }
                else
                {
                    timer.Stop();
                    simult.Checked = false;
                }

            }

        }

        public void timer_a_Tick_windLeft(object sender, EventArgs e)
        {
            if (simult.Checked == true)
            {
                if (collapsibleGroupBoxBasicPanel.numericUpDown1.Value >= 11.0m)
                {
                    collapsibleGroupBoxBasicPanel.numericUpDown1.Value -= 1.0m;
                    UpdateImage();
                }
                else if (collapsibleGroupBoxBasicPanel.numericUpDown1.Value <= 1.0m)
                {
                    windTimerLeft.Stop();
                    simult.Checked = false;
                }
```

```
    }
}
public void timer_a_Tick_windLeftPlus(object sender, EventArgs e)
{
    if (collapsibleGroupBoxBasicPanel.numericUpDown1.Value < 180.0m)
    {
        collapsibleGroupBoxBasicPanel.numericUpDown1.Value += 1.0m;
        UpdateImage();
    }
    else if (collapsibleGroupBoxBasicPanel.numericUpDown1.Value == 180.0m)
    {
        windTimerLeftPlus.Stop();
        simult.Checked = false;
        leftplusButtonSelected = false;
    }
}

public void timer_a_Tick_windLeftMinus(object sender, EventArgs e)
{
    if (collapsibleGroupBoxBasicPanel.numericUpDown1.Value > 1.0m)
    {
        collapsibleGroupBoxBasicPanel.numericUpDown1.Value -= 1.0m;
        UpdateImage();
    }
    else if (collapsibleGroupBoxBasicPanel.numericUpDown1.Value == 1.0m)
    {
        windTimerLeftMinus.Stop();
        simult.Checked = false;
        leftminusButtonSelected = false;
    }
}

public void timer_a_Tick_windRightPlus(object sender, EventArgs e)
{
    if (collapsibleGroupBoxBasicPanel.numericUpDown2.Value < 180.0m)
    {
        collapsibleGroupBoxBasicPanel.numericUpDown2.Value += 1.0m;
        UpdateImage();
    }
    else if (collapsibleGroupBoxBasicPanel.numericUpDown2.Value == 180.0m)
    {
        windTimerRightMinus.Stop();
        simult.Checked = false;
        rightplusButtonSelected = false;
    }
}

public void timer_a_Tick_windRightMinus(object sender, EventArgs e)
{
    if (collapsibleGroupBoxBasicPanel.numericUpDown2.Value > 1.0m)
    {
        collapsibleGroupBoxBasicPanel.numericUpDown2.Value -= 1.0m;
```

```
                    UpdateImage();
                }
                else if (collapsibleGroupBoxBasicPanel.numericUpDown2.Value == 1.0m)
                {
                    windTimerRightMinus.Stop();
                    simult.Checked = false;
                    rightminusButtonSelected = false;

                }
            }

            #endregion
            public void UpdateImage()
            {
                if (treeToolStripMenuItem.Checked == true)
                {
                    CreateGraphicsObjects();
                    someNewValueX = startX;
                    someNewValueY = startY;
                    double                          treeRootLength                          =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDowntreeLength.Value);
                    animationFactor                                                         =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numeric_a.Value);
                    DrawTree(someNewValueX, someNewValueY, treeRootLength, baseAngle);
                    pictureBox1.Image = bitmap;

                }
                if (hilbertCurveToolStripMenuItem.Checked == true)
                {
                    CreateGraphicsObjectsGilber();
                    someNewValueX = startX;
                    someNewValueY = startY;
                    double                          treeRootLength                          =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDowntreeLength.Value) + 50;
                    animationFactor                                                         =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numeric_a.Value);
                    DrawGilbert();
                    pictureBox1.Image = bitmap;
                }
                if (gosperCurveToolStripMenuItem.Checked == true)
                {
                    CreateGraphicsObjectsGosper();
                    someNewValueX = startX;
                    someNewValueY = startY;
                    treeRootLength                                                          =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDowntreeLength.Value);
                    animationFactor                                                         =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numeric_a.Value);
                    u = Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDown1.Value);
                    pictureBox1.Image = bitmap;
                    string                              selectedq                           =
collapsibleGroupBoxBasicPanel.comboBoxOneOrZero.SelectedItem?.ToString();
                    if (selectedq == null)
```

```
                        {
                            MessageBox.Show("Будь ласка, виберіть значення ComboBox.");
                        }
                        else
                        {
                            q = Convert.ToDouble(selectedq);
                            gosper.Draw(someNewValueX,    someNewValueY,    treeRootLength,    u,
(int)animationFactor, (int)q);
                        }
                    }
                }
                public void pain_Click(object sender, EventArgs e)
                {
                    if (treeToolStripMenuItem.Checked == true)
                    {
                        CreateGraphicsObjects();
                        startX                                                                  =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDownX.Value);
                        startY                                                                  =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDownY.Value);
                        treeRootLength                                                          =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDowntreeLength.Value);
                        animationFactor                                                         =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numeric_a.Value);
                        pictureBox1.Image = bitmap;
                        DrawTree(startX, startY, treeRootLength, baseAngle);
                        simult.Checked = false;
                    }
                    if (hilbertCurveToolStripMenuItem.Checked == true)
                    {
                        CreateGraphicsObjectsGilber();
                        DrawGilbert();
                    }

                    if (gosperCurveToolStripMenuItem.Checked == true)
                    {
                        CreateGraphicsObjectsGosper();
                        startX = (double)collapsibleGroupBoxBasicPanel.numericUpDownX.Value;
                        startY = (double)collapsibleGroupBoxBasicPanel.numericUpDownY.Value;
                        treeRootLength                                                          =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDowntreeLength.Value);
                        animationFactor = (double)collapsibleGroupBoxBasicPanel.numeric_a.Value;
                        u = Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDown1.Value);
                        pictureBox1.Image = bitmap;
                        string                              selectedq                           =
collapsibleGroupBoxBasicPanel.comboBoxOneOrZero.SelectedItem?.ToString();
                        if (selectedq == null)
                        {
                            MessageBox.Show("Будь ласка, виберіть значення ComboBox.");
                        }
                        else
                        {
                            try
```

```
                        {
                            q = Convert.ToDouble(selectedq);
                            gosper.Draw(startX,        startY,        treeRootLength,        u,
(int)animationFactor, (int)q);
                        }
                        catch (FormatException)
                        {
                            MessageBox.Show("Помилка: Не можна конвертувати вибране значення в
число.");
                        }
                    }
                }
            }
            public void DrawTree(double x, double y, double length, double angle)
            {
                this.x = x;
                this.y = y;
                this.length = length;
                this.angle = angle;
                baseAngle                                                                    =
(Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDownbaseAngle.Value) * Math.PI) / 180.0;
                leftChildAngle                                                               =
(Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDown1.Value) * Math.PI) / 180.0;
                rightChildAngle                                                              =
(Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDown2.Value) * Math.PI) / 180.0;
                offset1Angle                                                                 =
(Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDown3.Value) * Math.PI) / 180.0;
                offset2Angle                                                                 =
(Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDown4.Value) * Math.PI) / 180.0;

                if (length > 2)
                {
                    length *= animationFactor;
                    double newX = Math.Round(x + length * Math.Cos(angle));
                    double newY = Math.Round(y - length * Math.Sin(angle));

                   graphics.DrawLine(pen, (float)x, (float)y, (float)newX, (float)newY);

                    x = newX;
                    y = newY;

                    if (newToolStripMenuItemClicked == false)
                    {
                        DrawTree(x, y, length, angle + leftChildAngle);
                        DrawTree(x, y, length, angle - rightChildAngle);
                    }
                    else if (newToolStripMenuItemClicked == true)
                    {
                        collapsibleGroupBoxBasicPanel.numeric_a.Maximum = 0.7m;
                        DrawTree(x, y, length, angle + leftChildAngle);
                        DrawTree(x, y, length, angle - rightChildAngle);
                        DrawTree(x, y, length, angle + offset1Angle);
                        DrawTree(x, y, length, angle - offset2Angle);
```

```
                }
            }
        }
        public void DrawGilbert()
        {
            CreateGraphicsObjectsGilber();

            animationFactor =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numeric_a.Value);
            pictureBox1.BackColor = colorDialog2.Color;
            pictureBox1.Image = bitmap;
            double start_length, total_length;
            double dep = Double.Parse(animationFactor.ToString());

            if (pictureBox1.Height < pictureBox1.Width)
            {
                total_length =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDowntreeLength.Value);
            }
            else
            {
                total_length = (float)(0.9 * pictureBox1.Width);
            }

            startX = Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDownX.Value);
            startY = Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDownY.Value);

            start_length = (double)(total_length / (Math.Pow(2, dep) - 1));

            gilbert.LastX = (int)startX;
            gilbert.LastY = (int)startY;
            gilbert.Hilbert((int)dep, (int)start_length, 0);
        }
        private void CreateGraphicsObjects()
        {
            bitmap = new Bitmap(pictureBox1.Width, pictureBox1.Height);
            graphics = Graphics.FromImage(bitmap);
            graphics.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
            Color penColor = Color.Black;

            if (pictureBoxColorWasClicked)
            {
                penColor = pen.Color;
            }

            pen = new Pen(penColor, (float)penwight);
        }
        private void CreateGraphicsObjectsGilber()
        {
            bitmap = new Bitmap(pictureBox1.Width, pictureBox1.Height);
            gilbert.graphics = Graphics.FromImage(bitmap);
            gilbert.graphics.SmoothingMode =
System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
```

```
                Color penColor = Color.Black;

                if (pictureBoxColorWasClicked)
                {
                    penColor = pen.Color;
                }

                gilbert.pen = new Pen(penColor, (float)penwight);
            }
            private void CreateGraphicsObjectsGosper()
            {
                bitmap = new Bitmap(pictureBox1.Width, pictureBox1.Height);
                gosper.graphics = Graphics.FromImage(bitmap);
                gosper.graphics.SmoothingMode =
System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
                Color penColor = Color.Black;

                if (pictureBoxColorWasClicked)
                {
                    penColor = pen.Color;
                }

                gosper.pen = new Pen(penColor, (float)penwight);
            }

            #region  animation_Click
            private void animation_Click(object sender, EventArgs e)
            {
                isPaused = false;
                timer.Stop();
                collapsibleGroupBoxBasicPanel.numeric_a.Value = 0.1m;
                pause.Enabled = true;
                pain_Click(this, EventArgs.Empty);
                timer.Start();
            }
            public void DrawAndSaveImage(Control groupBox)
            {
                CreateGraphicsObjects();
                double x =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDownX.Value);
                double y =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDownY.Value);
                treeRootLength =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDowntreeLength.Value);
                baseAngle =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numericUpDownbaseAngle.Value);
                animationFactor = 0.7;
                DrawTree(x, y, treeRootLength, angle);
                SaveImage();
                pictureBox1.BackgroundImage = bitmap;
            }
            private string filePath;
            public void SaveImage()
```

```
{
    string directoryPath = @"C:\Users\alexv\Desktop\diplom\diplom";
    string fileName = $"imageTree_{DateTime.Now:yyyyMMddHHmmss}.png";
    string filePath = Path.Combine(directoryPath, fileName);
    bitmap.Save(filePath, ImageFormat.Png);
}
private void comboBoxSpeed_SelectedIndexChanged(object sender, EventArgs e)
{
    double selectedSpeed = (double)comboBoxSpeed.SelectedItem;
    timer.Interval = (int)(300 / selectedSpeed);
    windTimerLeftPlus.Interval = (int)(300 / selectedSpeed);
    windTimerLeftMinus.Interval = (int)(300 / selectedSpeed);
    windTimerRightPlus.Interval = (int)(300 / selectedSpeed);
    windTimerRightMinus.Interval = (int)(300 / selectedSpeed);
}
#endregion
private void stop_Click(object sender, EventArgs e)
{
    timer.Stop();
    simult.Checked = false;
    windTimerRightPlus.Stop();
    windTimerRightMinus.Stop();
    windTimerLeftPlus.Stop();
    windTimerLeftMinus.Stop();
     leftplusButtonSelected = false;
     leftminusButtonSelected = false;
     rightplusButtonSelected = false;
     rightminusButtonSelected = false;
     pause.Enabled = false;
}
private void pause_Click(object sender, EventArgs e)
{
    if (!isPaused)
    {
        timer.Stop();
        isPaused = true;
    }
    else
    {
        timer.Start();
        isPaused = false;
    }
}
#region menu
public bool ispictureBoxMovementPressed = false;
public bool ispictureBoxSizeReleased = false;
public bool ispictureBoxSizeMinusReleased = false;
public bool ispictureBoxCursorReleased = false;
public bool pictureBoxColorWasClicked = false;
public bool newToolStripMenuItemClicked = false;

public void pictureBoxSize_Click(object sender, EventArgs e)
{
```

```
                ispictureBoxSizeMinusReleased = false;
                ispictureBoxSizeReleased = true;
                ispictureBoxMovementPressed = false;
            }
            public void pictureBoxSizeMinus_Click(object sender, EventArgs e)
            {
                ispictureBoxSizeReleased = false;
                ispictureBoxSizeMinusReleased = true;
                ispictureBoxMovementPressed = false;

            }
            public void pictureBoxMovement_Click(object sender, EventArgs e)
            {
                ispictureBoxMovementPressed = true;
                ispictureBoxSizeReleased = false;
                ispictureBoxSizeMinusReleased = false;
                ispictureBoxCursorReleased = false;
            }
            public void pictureBoxCursor_Click(object sender, EventArgs e)
            {
                ispictureBoxCursorReleased = true;
                ispictureBoxSizeReleased = false;
                ispictureBoxSizeMinusReleased = false;
                ispictureBoxMovementPressed = false;
            }
            private void copyImageToolStripMenuItem_Click(object sender, EventArgs e)
            {
                if (pictureBox1.Image != null)
                {
                    Clipboard.SetImage(pictureBox1.Image);
                }
            }
            private void resetLoToolStripMenuItem_Click(object sender, EventArgs e)
            {
                if (treeToolStripMenuItem.Checked)
                {
                    collapsibleGroupBoxBasicPanel.numericUpDownX.Value = pictureBox1.Width /
2;
                    collapsibleGroupBoxBasicPanel.numericUpDownY.Value =
(decimal)(pictureBox1.Height / 1.3);
                    pain_Click(sender, e);
                }
                if (hilbertCurveToolStripMenuItem.Checked)
                {
                    collapsibleGroupBoxBasicPanel.numericUpDownX.Value = pictureBox1.Width / 2
- 100;
                    collapsibleGroupBoxBasicPanel.numericUpDownY.Value =
(decimal)(pictureBox1.Height / 2 - 100);
                    pain_Click(sender, e);
                }
                if (gosperCurveToolStripMenuItem.Checked)
                {
```

```
                    collapsibleGroupBoxBasicPanel.numericUpDownX.Value = pictureBox1.Width / 2
+ 100;
                    collapsibleGroupBoxBasicPanel.numericUpDownY.Value                    =
(decimal)(pictureBox1.Height / 2 + 50);
                    pain_Click(sender, e);
                }

            }
            private void pictureBoxColor_Click(object sender, EventArgs e)
            {
                timer.Stop();
                pictureBoxColorWasClicked = true;

                if (colorDialog1 != null && colorDialog1.ShowDialog() == DialogResult.OK)
                {
                    Color color = colorDialog1.Color;
                    pen.Color = color;
                    if (bitmap != null)
                    {
                        UpdateImage();
                    }
                }
            }
            public void imagToolStripMenuItem_Click(object sender, EventArgs e)
            {
                collapsibleGroupBoxEditImag.IsCollapsed = false;
                collapsibleGroupBoxEditImag_CollapseBoxClickedEvent(sender);

            }
            private void toolStripComboBoxthickness_Click(object sender, EventArgs e)
            {
                toolStripComboBoxthickness.Items.Clear();
                toolStripComboBoxthickness.Items.AddRange(new object[] { 1.0, 2.0, 5.0, 8.0,
11.0, 14.0, 25.0, 50.0 });
            }
            private void saveAllToolStripMenuItem_Click(object sender, EventArgs e)
            {
                pictureBoxSaveParams_Click(sender, e);
                saveAsToolStripMenuItem_Click(sender, e);
            }
            private   void   toolStripComboBoxthickness_SelectedIndexChanged(object   sender,
EventArgs e)
            {
                double selectedThickness = (double)toolStripComboBoxthickness.SelectedItem;
                penwight = selectedThickness;
                pen = new Pen(pen.Color, (float)selectedThickness);
                UpdateImage();
            }
            public void comboBoxWidth_TextChanged(object sender, EventArgs e)
            {
                if (!string.IsNullOrEmpty(comboBoxWidth.Text))
                {
                    int width;
```

```
            if (int.TryParse(comboBoxWidth.Text, out width))
            {
                pictureBox1.Width = width;
            }
        }
    }
    public void comboBoxHeight_TextChanged(object sender, EventArgs e)
    {
        if (!string.IsNullOrEmpty(comboBoxHeight.Text))
        {
            int height;
            if (int.TryParse(comboBoxHeight.Text, out height))
            {
                pictureBox1.Height = height;
            }
        }
    }
    private void comboBoxColor_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (comboBoxColor.SelectedItem != null && comboBoxColor.SelectedItem is Color)
        {
            Color selectedColor = (Color)comboBoxColor.SelectedItem;
            pictureBox1.BackColor = selectedColor;
        }
    }
    private void colorelse_Click(object sender, EventArgs e)
    {
        if (colorDialog2.ShowDialog() == DialogResult.OK)
        {
            pictureBox1.BackColor = colorDialog2.Color;
        }
    }
    public void checkBoxAll_CheckedChanged(object sender, EventArgs e)
    {
        int x = (panel1.Width / 2) - (788 / 2);
        pictureBox1.Anchor    =    AnchorStyles.Bottom    |    AnchorStyles.Right    |
AnchorStyles.Left | AnchorStyles.Top;
        if (checkBoxAll.Checked)
        {
            pictureBox1.Location = new Point(0, 0);
            pictureBox1.Width = panel1.Width;
            pictureBox1.Height = panel1.Height;
        }
        else if (checkBoxAll.Checked == false)
        {
            pictureBox1.Location = new Point(x, 14);
            comboBoxWidth_TextChanged(null, EventArgs.Empty);
            comboBoxHeight_TextChanged(null, EventArgs.Empty);
        }
    }
    private void deleteToolStripMenuItem_Click(object sender, EventArgs e)
    {
        newToolStripMenuItemClicked = false;
```

```
                deleteToolStripMenuItem.Checked = true;
                addToolStripMenuItem.Checked = false;

                LabelManager.SetLabelRight2Visibility(false);
                LabelManager.SetLabelLeft2Visibility(false);
                LabelManager.SetLabelCel1Visibility(false);
                LabelManager.SetLabelCel2Visibility(false);
                if (treeToolStripMenuItem.Checked)
                {
                    collapsibleGroupBoxBasicPanel.numericUpDown3.Visible = false;
                    collapsibleGroupBoxBasicPanel.numericUpDown4.Visible = false;
                }
            }
            private void addToolStripMenuItem_Click(object sender, EventArgs e)
            {
                newToolStripMenuItemClicked = true;
                addToolStripMenuItem.Checked = true;
                deleteToolStripMenuItem.Checked = false;
                treeToolStripMenuItem_Click(sender, e);
                animationFactor                                                    =
Convert.ToDouble(collapsibleGroupBoxBasicPanel.numeric_a.Value);
                if (treeToolStripMenuItem.Checked)
                {
                    collapsibleGroupBoxBasicPanel.numericUpDown3.Visible = true;
                    collapsibleGroupBoxBasicPanel.numericUpDown4.Visible = true;
                }
            }
            private bool isLeftMouseButtonPressed = false;
            private Point lastMousePosition;
            private void pictureBox1_MouseEnter(object sender, EventArgs e)
            {
                if (ispictureBoxMovementPressed)
                {
                    Cursor = Cursors.Hand;
                }

                if (ispictureBoxSizeReleased)
                {
                    Cursor = Cursors.SizeAll;
                }

                if (ispictureBoxSizeMinusReleased)
                {
                    Cursor = Cursors.SizeAll;
                }

                if (ispictureBoxCursorReleased)
                {
                    Cursor = Cursors.Default;
                }
            }

            private void closeToolStripMenuItem_Click(object sender, EventArgs e)
```

```
    {
        MenuPanal menu = new MenuPanal();
        menu.CloseApplication(this);
    }
    private void saveToolStripMenuItem_Click(object sender, EventArgs e)
    {
        MenuPanal menu = new MenuPanal();
        menu.SaveApplication(this, bitmap, backgroundImage);
    }
    private void saveAsToolStripMenuItem_Click(object sender, EventArgs e)
    {
        MenuPanal menu = new MenuPanal();
        menu.SaveLikeApplication(this, bitmap, BackgroundImage);

    }
#endregion
#region loupe_Click
    private void pictureBox1_MouseLeave(object sender, EventArgs e)
    {
        if (ispictureBoxCursorReleased == true)
        {
            Cursor = Cursors.Default;
        }
    }
    public void pictureBox1_MouseDown(object sender, MouseEventArgs e)
    {
        if (ispictureBoxSizeReleased && e.Button == MouseButtons.Left)
        {
            collapsibleGroupBoxBasicPanel.numericUpDowntreeLength.Value += 10;
            UpdateImage();
        }

        if (ispictureBoxSizeMinusReleased && e.Button == MouseButtons.Left)
        {
            collapsibleGroupBoxBasicPanel.numericUpDowntreeLength.Value -= 10;
            UpdateImage();
        }
        if (e.Button == MouseButtons.Left && ispictureBoxMovementPressed)
        {
            isLeftMouseButtonPressed = true;
            lastMousePosition = e.Location;
        }
    }
    private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
    {
        if (isLeftMouseButtonPressed && ispictureBoxMovementPressed)
        {
            if (e.Button == MouseButtons.Left)
            {
                int dx = e.X - lastMousePosition.X;
                int dy = e.Y - lastMousePosition.Y;
                startX += dx;
                startY += dy;
```

```
                        try
                        {
                            collapsibleGroupBoxBasicPanel.numericUpDownX.Value = (decimal)startX;
                            collapsibleGroupBoxBasicPanel.numericUpDownY.Value = (decimal)startY;
                        }
                        catch (ArgumentOutOfRangeException ex)
                        {
                            MessageBox.Show($"Value of '{ex.ParamName}' is not valid for 'Value'. 'Value' should be between {collapsibleGroupBoxBasicPanel.numericUpDownX.Minimum} and {collapsibleGroupBoxBasicPanel.numericUpDownX.Maximum}.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                            collapsibleGroupBoxBasicPanel.numericUpDownX.Value = (decimal)startX + 100m;
                            collapsibleGroupBoxBasicPanel.numericUpDownY.Value = (decimal)startY + 100m;
                        }
                        UpdateImage();
                        lastMousePosition = e.Location;
                    }
                }
                int mouseX = e.X;
                int mouseY = e.Y;
                labelX.Text = $"X: {mouseX}";
                labelY.Text = $"Y: {mouseY}";
            }
            private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
            {
                    if (e.Button == MouseButtons.Left && ispictureBoxMovementPressed)
                    {
                        isLeftMouseButtonPressed = false;
                    }
            }
            #endregion
            #region mini и radioButton
            public void wind_Click(object sender, EventArgs e)
            {
                collapsibleGroupBoxBasicPanel.numeric_a.Value = 0.5m;
                collapsibleGroupBoxBasicPanel.numericUpDown2.Value = 45.0m;
                collapsibleGroupBoxBasicPanel.numericUpDown1.Value = 45.0m;
            }
            public bool leftplusButtonSelected = false;
            public bool leftminusButtonSelected = false;
            public bool rightplusButtonSelected = false;
            public bool rightminusButtonSelected = false;
            public void leftplus_Click(object sender, EventArgs e)
            {
                leftplusButtonSelected = true;
                if (simult.Checked == false)
                {
                    windTimerLeftPlus.Start();
                    windTimerLeftMinus.Stop();
```

```
    }
    else if (leftplusButtonSelected && rightplusButtonSelected)
    {
        windTimerLeftPlus.Start();
        windTimerRightPlus.Start();
        windTimerRightMinus.Stop();
        windTimerLeftMinus.Stop();
    }
    else if (leftplusButtonSelected && rightminusButtonSelected)
    {
        windTimerLeftPlus.Start();
        windTimerRightMinus.Start();
        windTimerLeftMinus.Stop();
        windTimerRightPlus.Stop();
    }
}
private void leftminus_Click(object sender, EventArgs e)
{
    leftminusButtonSelected = true;

    if (simult.Checked == false)
    {
        windTimerLeftMinus.Start();
        windTimerLeftPlus.Stop();
    }
    else if (leftminusButtonSelected && rightminusButtonSelected)
    {
        windTimerLeftMinus.Start();
        windTimerRightMinus.Start();
        windTimerLeftPlus.Stop();
        windTimerRightPlus.Stop();
    }
    else if (leftminusButtonSelected && rightplusButtonSelected)
    {
        windTimerLeftMinus.Start();
        windTimerRightPlus.Start();
        windTimerRightMinus.Stop();
        windTimerLeftPlus.Stop();
    }
}
private void rightplus_Click(object sender, EventArgs e)
{
    rightplusButtonSelected = true;
    if (simult.Checked == false)
    {
        windTimerRightPlus.Start();
        windTimerRightMinus.Stop();
    }
    else if (leftplusButtonSelected && rightplusButtonSelected)
    {
        windTimerLeftPlus.Start();
        windTimerRightPlus.Start();
        windTimerLeftMinus.Stop();
```

```
            WindTimerRightMinus.Stop();
        }
        else if (rightplusButtonSelected && leftminusButtonSelected)
        {
            windTimerRightPlus.Start();
            windTimerLeftMinus.Start();
            windTimerRightMinus.Stop();
            windTimerLeftPlus.Stop();
        }
    }
    private void rightminus_Click(object sender, EventArgs e)
    {
        rightminusButtonSelected = true;
        if (simult.Checked == false)
        {
            WindTimerRightMinus.Start();
            windTimerRightPlus.Stop();
        }
        else if (leftminusButtonSelected && rightminusButtonSelected)
        {
            windTimerLeftMinus.Start();
            windTimerRightMinus.Start();
            WindTimerLeftPlus.Stop();
            windTimerRightPlus.Stop();
        }
        else if (rightminusButtonSelected && leftplusButtonSelected)
        {
            windTimerRightMinus.Start();
            windTimerLeftPlus.Start();
            windTimerRightPlus.Stop();
            windTimerLeftMinus.Stop();
        }
    }
    private void clean_Click(object sender, EventArgs e)
    {
        timer.Stop();
        bitmap.Dispose();
        bitmap = new Bitmap(pictureBox1.Width, pictureBox1.Height);
        pictureBox1.Image = bitmap;
    }
    private void simult_CheckedChanged(object sender, EventArgs e)
    {
        windTimerLeftMinus.Stop();
        windTimerLeftPlus.Stop();
        windTimerRightPlus.Stop();
        windTimerRightMinus.Stop();
    }
    #endregion

    private void pictureBoxOpen_MouseEnter(object sender, EventArgs e)
    {
        MouseEnterColor();
    }
```

```
        public void MouseEnterColor()
        {
            pictureBoxMovement.BackColor = Color.LightBlue;
            pictureBoxCursor.BackColor = Color.LightBlue;
            pictureBoxSize.BackColor = Color.LightBlue;
            pictureBoxSizeMinus.BackColor = Color.LightBlue;
            pictureBoxColor.BackColor = Color.LightBlue;
            pictureBoxSavePArams.BackColor = Color.LightBlue;
        }
        private void pictureBoxOpen_MouseLeave(object sender, EventArgs e)
        {
            MouseLeaveColor();
        }
        public void MouseLeaveColor()
        {
            pictureBoxMovement.BackColor = SystemColors.Control;
            pictureBoxCursor.BackColor = SystemColors.Control;
            pictureBoxSize.BackColor = SystemColors.Control;
            pictureBoxSizeMinus.BackColor = SystemColors.Control;
            pictureBoxColor.BackColor = SystemColors.Control;
            pictureBoxSavePArams.BackColor = SystemColors.Control;
        }
        private void pictureBoxSaveParams_Click(object sender, EventArgs e)
        {
            string toolChecked = "";
            if (treeToolStripMenuItem.Checked)
            {
                toolChecked = "treeToolStripMenuItem";
            }
            else if (hilbertCurveToolStripMenuItem.Checked)
            {
                toolChecked = "hilbertCurveToolStripMenuItem";
            }
            else if (gosperCurveToolStripMenuItem.Checked)
            {
                toolChecked = "gosperCurveToolStripMenuItem";
            }

            int numericXValue = (int)collapsibleGroupBoxBasicPanel.numericUpDownX.Value;
            int numericYValue = (int)collapsibleGroupBoxBasicPanel.numericUpDownY.Value;
            int                              treeLengthValue                              =
(int)collapsibleGroupBoxBasicPanel.numericUpDowntreeLength.Value;
            double numericAValue = (double)collapsibleGroupBoxBasicPanel.numeric_a.Value;

            if (treeToolStripMenuItem.Checked)
            {
                int                          numericUpDown1Value                          =
(int)collapsibleGroupBoxBasicPanel.numericUpDown1.Value;
                int                          numericUpDown2Value                          =
(int)collapsibleGroupBoxBasicPanel.numericUpDown2.Value;
                string parameters = $"Tool Checked: {toolChecked} \n X: {numericXValue}\n
Y:  {numericYValue}\n  Tree  Length:  {treeLengthValue}\n  A:  {numericAValue}\n  NumericUpDown1:
{numericUpDown1Value}\n NumericUpDown2: {numericUpDown2Value}";
```

```
                    SaveParameters(parameters);
                }
                else if (hilbertCurveToolStripMenuItem.Checked)
                {
                    string parameters = $"Tool Checked: {toolChecked} \n X: {numericXValue}\n
Y: {numericYValue}\n Tree Length: {treeLengthValue}\n A: {numericAValue}\n";
                    SaveParameters(parameters);
                }
                else if (gosperCurveToolStripMenuItem.Checked)
                {
                    int                         numericUpDown1Value                         =
(int)collapsibleGroupBoxBasicPanel.numericUpDown1.Value;
                    string                          selectedValue                           =
collapsibleGroupBoxBasicPanel.comboBoxOneOrZero.SelectedItem?.ToString();
                    int comboxoorzValue;
                    if (int.TryParse(selectedValue, out comboxoorzValue))
                    {
                        string   parameters   =   $"Tool   Checked:   {toolChecked}   \n   X:
{numericXValue}\n  Y:  {numericYValue}\n  Tree  Length:  {treeLengthValue}\n  A:  {numericAValue}\n
NumericUpDown1: {numericUpDown1Value}\n comboxoorzValue: {comboxoorzValue}";
                        SaveParameters(parameters);
                    }
                }
            }
            private void SaveParameters(string parameters)
            {
                SaveFileDialog saveFileDialog1 = new SaveFileDialog();
                saveFileDialog1.Filter = "Текстовые файлы (*.txt)|*.txt|Все файлы (*.*)|*.*";
                saveFileDialog1.Title = " Зберегти параметри як... ";

                if (saveFileDialog1.ShowDialog() == DialogResult.OK)
                {
                    using (StreamWriter writer = new StreamWriter(saveFileDialog1.FileName))
                    {
                        writer.WriteLine(parameters);
                    }
                    MessageBox.Show("Параметри збережені у файлі " + saveFileDialog1.FileName);
                }
            }
            private void importToolStripMenuItem_Click(object sender, EventArgs e)
            {
                OpenFileDialog openFileDialog1 = new OpenFileDialog();
                openFileDialog1.Filter = " Текстові файли (*.txt)|*.txt|Всі файли (*.*)|*.*";
                openFileDialog1.Title = " Виберіть текстовий файл";

                if (openFileDialog1.ShowDialog() == DialogResult.OK)
                {
                    string[] lines = File.ReadAllLines(openFileDialog1.FileName);
                    if (lines.Length >= 1)
                    {
                        string[] firstLineParts = lines[0].Split(':');
                        if   (firstLineParts.Length   ==   2   &&   firstLineParts[1].Trim()   ==
"treeToolStripMenuItem")
```

```
                            {
                                if (lines.Length >= 7)
                                {
                                    treeToolStripMenuItem.Checked = true;
                                    treeToolStripMenuItem_Click(treeToolStripMenuItem,
EventArgs.Empty);
                                    collapsibleGroupBoxBasicPanel.numericUpDownX.Value          =
int.Parse(lines[1].Split(':')[1].Trim());
                                    collapsibleGroupBoxBasicPanel.numericUpDownY.Value          =
int.Parse(lines[2].Split(':')[1].Trim());
                                    collapsibleGroupBoxBasicPanel.numericUpDowntreeLength.Value  =
int.Parse(lines[3].Split(':')[1].Trim());
                                    collapsibleGroupBoxBasicPanel.numeric_a.Value                =
(decimal)double.Parse(lines[4].Split(':')[1].Trim());
                                    collapsibleGroupBoxBasicPanel.numericUpDown1.Value          =
int.Parse(lines[5].Split(':')[1].Trim());
                                    collapsibleGroupBoxBasicPanel.numericUpDown2.Value          =
int.Parse(lines[6].Split(':')[1].Trim());
                                    pain_Click(this, EventArgs.Empty);
                                }
                            }
                            if  (firstLineParts.Length  ==  2  &&  firstLineParts[1].Trim()  ==
"hilbertCurveToolStripMenuItem")
                            {
                                if (lines.Length >= 4)
                                {
                                    hilbertCurveToolStripMenuItem.Checked = true;

hilbertCurveToolStripMenuItem_Click(hilbertCurveToolStripMenuItem, EventArgs.Empty);
                                    collapsibleGroupBoxBasicPanel.numericUpDownX.Value          =
int.Parse(lines[1].Split(':')[1].Trim());
                                    collapsibleGroupBoxBasicPanel.numericUpDownY.Value          =
int.Parse(lines[2].Split(':')[1].Trim());
                                    collapsibleGroupBoxBasicPanel.numericUpDowntreeLength.Value  =
int.Parse(lines[3].Split(':')[1].Trim());
                                    collapsibleGroupBoxBasicPanel.numeric_a.Value                =
(decimal)double.Parse(lines[4].Split(':')[1].Trim());
                                    pain_Click(this, EventArgs.Empty);
                                }

                            }
                            if  (firstLineParts.Length  ==  2  &&  firstLineParts[1].Trim()  ==
"gosperCurveToolStripMenuItem")
                            {
                                if (lines.Length >= 7)
                                {
                                    gosperCurveToolStripMenuItem.Checked = true;

gosperCurveToolStripMenuItem_Click(gosperCurveToolStripMenuItem, EventArgs.Empty);
                                    collapsibleGroupBoxBasicPanel.numericUpDownX.Value          =
int.Parse(lines[1].Split(':')[1].Trim());
                                    collapsibleGroupBoxBasicPanel.numericUpDownY.Value          =
int.Parse(lines[2].Split(':')[1].Trim());
```

```
                                collapsibleGroupBoxBasicPanel.numericUpDowntreeLength.Value  =
int.Parse(lines[3].Split(':')[1].Trim());
                                collapsibleGroupBoxBasicPanel.numeric_a.Value                =
(decimal)double.Parse(lines[4].Split(':')[1].Trim());
                                collapsibleGroupBoxBasicPanel.numericUpDown1.Value           =
int.Parse(lines[5].Split(':')[1].Trim());
                                string selectedValue = lines[6].Split(':')[1].Trim();
                                collapsibleGroupBoxBasicPanel.comboBoxOneOrZero.SelectedItem =
selectedValue;
                                pain_Click(this, EventArgs.Empty);
                            }
                        }

                    }

                }
            }
            private void Form1_Load(object sender, EventArgs e)
            {
                comboBoxSpeed.Items.AddRange(new object[] {0.5, 0.75, 1.0, 1.5, 2.0 });
                comboBoxSpeed.SelectedItem = 1.0;
                collapsibleGroupBoxEditImag.IsCollapsed = false;
                collapsibleGroupBoxanimation.IsCollapsed = false;
                collapsibleGroupBoxBasic.IsCollapsed = false;
                pen = new Pen(Color.Black);
            }
            private void collapsibleGroupBoxEditImag_Enter(object sender, EventArgs e)
            {
                comboBoxWidth.Items.Clear();
                comboBoxHeight.Items.Clear();
                comboBoxColor.Items.Clear();
                comboBoxWidth.Text = pictureBox1.Width.ToString();
                comboBoxWidth.Items.AddRange(new object[] { 72, 576, 1200, 1280, 788  });
                comboBoxHeight.Text = pictureBox1.Height.ToString();
                comboBoxHeight.Items.AddRange(new object[] { 72, 288, 600, 720, 640 });
                comboBoxColor.Text = pictureBox1.BackColor.ToString();
                comboBoxColor.Items.AddRange(new    object[]    {    Color.Pink,    Color.Blue,
Color.Green });
            }
            #region collap
            public bool collapsibleGroupBoxBasic_CollapseBoxEvent;
            private void collapsibleGroupBoxBasic_CollapseBoxClickedEvent(object sender)
            {
                bool collapsibleGroupBoxBasic_CollapseBoxEvent;
                LabelManager.SetLabelLeft2Visibility(false);
                if (collapsibleGroupBoxBasic.IsCollapsed)
                {
                    collapsibleGroupLogic();
                }
                else
                {
                    collapsibleGroupLogic();
                }
```

```
            }
            private void collapsibleGroupBoxanimation_CollapseBoxClickedEvent(object sender)
            {
                if (collapsibleGroupBoxanimation.IsCollapsed == true)
                {
                    collapsibleGroupLogic();
                }
                else
                {
                    collapsibleGroupLogic();
                }

            }
            private void collapsibleGroupBoxEditImag_CollapseBoxClickedEvent(object sender)
            {
                if (collapsibleGroupBoxEditImag.IsCollapsed)
                {
                    collapsibleGroupLogic();
                }
                else
                {
                    collapsibleGroupLogic();
                }
            }
            public void collapsibleGroupLogic()
            {
                collapsibleGroupBoxEditImag.Location = new Point(11, 14);
                collapsibleGroupBoxanimation.Location = new Point(11,
collapsibleGroupBoxEditImag.Location.Y + collapsibleGroupBoxEditImag.Height + 5);
                collapsibleGroupBoxBasic.Location = new Point(11,
collapsibleGroupBoxanimation.Location.Y + collapsibleGroupBoxanimation.Height + 5);
            }
            #endregion collab
            private void treeToolStripMenuItem_Click(object sender, EventArgs e)
            {
                treeToolStripMenuItem.Checked = true;
                gosperCurveToolStripMenuItem.Checked = false;
                hilbertCurveToolStripMenuItem.Checked = false;
                collapsibleGroupBoxBasic.Controls.Clear();
                if (addToolStripMenuItem.Checked)
                {
                    collapsibleGroupBoxBasicPanel.whatcollapsibleGroupBoxBasicPanel(this,
collapsibleGroupBoxBasic);
                    addToolStripMenuItem.Checked = false;
                }
                else if(addToolStripMenuItem.Checked == false)
                {
                    collapsibleGroupBoxBasicPanel.whatcollapsibleGroupBoxBasicPanel(this,
collapsibleGroupBoxBasic);
                    deleteToolStripMenuItem_Click(sender, e);

                }
                buttonsmallchoice();
```

```
        }
        private void hilbertCurveToolStripMenuItem_Click(object sender, EventArgs e)
        {
                collapsibleGroupBoxBasic.Controls.Clear();
                addToolStripMenuItem.Checked = false;
                deleteToolStripMenuItem.Checked =false;
                treeToolStripMenuItem.Checked = false;
                gosperCurveToolStripMenuItem.Checked = false;
                hilbertCurveToolStripMenuItem.Checked = true;
                collapsibleGroupBoxBasicPanel.whatcollapsibleGroupBoxBasicPanel(this,
collapsibleGroupBoxBasic);
                buttonsmallchoice();
        }
        private void gosperCurveToolStripMenuItem_Click(object sender, EventArgs e)
        {
            collapsibleGroupBoxBasic.Controls.Clear();
            addToolStripMenuItem.Checked = false;
            deleteToolStripMenuItem.Checked = false;
            treeToolStripMenuItem.Checked = false;
            hilbertCurveToolStripMenuItem.Checked = false;
            gosperCurveToolStripMenuItem.Checked = true;
            collapsibleGroupBoxBasicPanel.whatcollapsibleGroupBoxBasicPanel(this,
collapsibleGroupBoxBasic);
            buttonsmallchoice();
        }
        public void buttonsmallchoice()
        {
            if (treeToolStripMenuItem.Checked)
            {
                leftplus.Enabled = true; rightplus.Enabled = true;
                leftminus.Enabled = true; rightminus.Enabled = true;
            }
            if (hilbertCurveToolStripMenuItem.Checked)
            {
                leftplus.Enabled = false; rightplus.Enabled = false;
                leftminus.Enabled = false; rightminus.Enabled = false;
            }
            if (gosperCurveToolStripMenuItem.Checked)
            {
                leftplus.Enabled = true; rightplus.Enabled = false;
                leftminus.Enabled = true; rightminus.Enabled = false;
            }
        }
        public void buttonImage_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.InitialDirectory                                          =
@"C:\Users\alexv\Desktop\diplom\PythagorasTree\PythagorasTree\bin\Debug\fon";
            openFileDialog.Filter  =  "Image  Files  (*.jpg,  *.png)|*.jpg;*.png|All  Files
(*.*)|*.*";
            openFileDialog.FilterIndex = 1;
            openFileDialog.RestoreDirectory = true;
            if (openFileDialog.ShowDialog() == DialogResult.OK)
```

```
            {
                try
                {
                    pictureBox1.BackgroundImage = new Bitmap(openFileDialog.FileName);
                    pictureBox1.BackgroundImageLayout = ImageLayout.Center;
                    string imageName = Path.GetFileName(openFileDialog.FileName);
                    textBoxImage.Text = imageName;
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Ошибка при завантаженні зображення: " + ex.Message);
                }
            }
        }
        private void completeCleaningToolStripMenuItem_Click(object sender, EventArgs e)
        {
            clean_Click(sender, e);
            backgroundImage = pictureBox1.BackgroundImage;
            pictureBox1.BackgroundImage = null;
            pictureBox1.BackColor = SystemColors.Control;
            textBoxImage.Text = "";
        }
    }
}
```

ДОДАТОК Е

Код класу панелі меню

```
namespace PythagorasTree
{
    public class MenuPanal
    {
        public static bool CloseApplicationBool = false;
        public void CloseApplication(Form form)
        {
            if (!CloseApplicationBool)
            {
                CloseApplicationBool = true;
                if (MessageBox.Show("Впевнені, що хочете закрити?", "Підтвердження",
MessageBoxButtons.YesNo) == DialogResult.Yes)
                {
                    form.Close();
                }
            }
        }
        public void OpenApplication(Form form)
        {
            string directoryPath =
Path.Combine(Path.GetDirectoryName(Application.ExecutablePath), "fractal");
            if (Directory.Exists(directoryPath))
            {
                Process.Start("explorer.exe", directoryPath);
            }
            else
            {
                MessageBox.Show("Папка fractal не існує.", "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }

        }
        public void SaveApplication(basicform form, Bitmap bitmap, Image backgroundImage)
        {
            string directoryPath = Path.GetDirectoryName(Application.ExecutablePath);
            directoryPath = Path.Combine(directoryPath, "fractal");
            if (!Directory.Exists(directoryPath))
            {
                Directory.CreateDirectory(directoryPath);
            }
            string fileName = $"imageTree_{DateTime.Now:yyyyMMddHHmmss}.png";
            backgroundImage = form.pictureBox1.BackgroundImage;
            string filePath = Path.Combine(directoryPath, fileName);
            Bitmap mergedBitmap = new Bitmap(bitmap.Width, bitmap.Height);
            using (Graphics g = Graphics.FromImage(mergedBitmap))
            {
                if (backgroundImage != null)
                {
```

```
                        g.DrawImage(backgroundImage, Point.Empty);
                    }
                    g.DrawImage(bitmap, Point.Empty);
                }
                mergedBitmap.Save(filePath, ImageFormat.Png);
            }
            public void SaveLikeApplication(basicform form, Bitmap bitmap, Image backgroundImage)
            {
                if (bitmap != null)
                {
                    SaveFileDialog saveFileDialog = new SaveFileDialog();
                    saveFileDialog.Filter = "JPEG Files|*.jpeg|PNG Files|*.png|All Files|*.*";
                    saveFileDialog.Title = "Save Image";
                    if (saveFileDialog.ShowDialog() == DialogResult.OK)
                    {
                        string filePath = saveFileDialog.FileName;
                        backgroundImage = form.pictureBox1.BackgroundImage;
                        Bitmap mergedBitmap = new Bitmap(bitmap.Width, bitmap.Height);
                        using (Graphics g = Graphics.FromImage(mergedBitmap))
                        {
                            if (backgroundImage != null)
                            {
                                g.DrawImage(backgroundImage, Point.Empty);
                            }
                            g.DrawImage(bitmap, Point.Empty);
                        }
                        mergedBitmap.Save(filePath, ImageFormat.Png);
                    }
                }
                else
                {
                    MessageBox.Show("Неможливо зберегти порожнє зображення.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            public void windowToolStripMenuItemMenu(ToolStripMenuItem menu)
            {
                menu.DropDownItems.Clear();
                foreach (Form openedForm in Application.OpenForms)
                {
                    ToolStripMenuItem formMenuItem = new ToolStripMenuItem(openedForm.Text);
                    formMenuItem.Click += (s, ev) => { openedForm.Activate(); };
                    menu.DropDownItems.Add(formMenuItem);
                }     }   }}
```

ДОДАТОК Ж

Код класу менеджера написів

```
namespace PythagorasTree
{
    public static class LabelManager
    {
        private static System.Windows.Forms.Label labelRight2;
        private static System.Windows.Forms.Label labelLeft2;
        private static System.Windows.Forms.Label labelCel1;
        private static System.Windows.Forms.Label labelCel2;
        public static (Label labelX, Label labelY, System.Windows.Forms.Label labelScale,
            System.Windows.Forms.Label labelA, System.Windows.Forms.Label labelRight,
System.Windows.Forms.Label labelLeft,
            System.Windows.Forms.Label label4, System.Windows.Forms.Label label6,
System.Windows.Forms.Label labelLeft2,
            System.Windows.Forms.Label labelRight2, System.Windows.Forms.Label labelCel1,
System.Windows.Forms.Label labelCel2, System.Windows.Forms.Label labelBase)
AddLabelToGroupBox(Control groupBox)
        {
            System.Windows.Forms.Label labelX = new System.Windows.Forms.Label();
            labelX.AutoSize = true;
            labelX.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            labelX.Location = new System.Drawing.Point(106, 26);
            labelX.Size = new System.Drawing.Size(22, 18);
            groupBox.Controls.Add(labelX);

            System.Windows.Forms.Label labelY = new System.Windows.Forms.Label();
            labelY.AutoSize = true;
            labelY.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            labelY.Location = new System.Drawing.Point(106, 55);
            labelY.Size = new System.Drawing.Size(21, 18);
            groupBox.Controls.Add(labelY);

            System.Windows.Forms.Label labelScale = new System.Windows.Forms.Label();
            labelScale.AutoSize = true;
            labelScale.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            labelScale.Location = new System.Drawing.Point(80, 83);
            labelScale.Name = "labelScale";
            labelScale.Size = new System.Drawing.Size(49, 18);
            labelScale.TabIndex = 2;
            groupBox.Controls.Add(labelScale);

            System.Windows.Forms.Label labelA = new System.Windows.Forms.Label();
            labelA.AutoSize = true;
            labelA.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            labelA.Location = new System.Drawing.Point(106, 138);
```

```
                    labelA.Name = "labelA";
                    labelA.Size = new System.Drawing.Size(20, 18);
                    labelA.TabIndex = 4;
                    groupBox.Controls.Add(labelA);

                    System.Windows.Forms.Label labelBase = new System.Windows.Forms.Label();
                    labelBase.AutoSize = true;
                    labelBase.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
                    labelBase.Location = new System.Drawing.Point(56, 112);
                    labelBase.Name = "labelA";
                    labelBase.Size = new System.Drawing.Size(20, 18);
                    labelBase.TabIndex = 4;
                    groupBox.Controls.Add(labelBase);

                    System.Windows.Forms.Label labelRight = new System.Windows.Forms.Label();
                    labelRight.AutoSize = true;
                    labelRight.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
                    labelRight.Location = new System.Drawing.Point(15, 219);
                    labelRight.Name = "right";
                    labelRight.Size = new System.Drawing.Size(105, 18);
                    labelRight.TabIndex = 8;
                    groupBox.Controls.Add(labelRight);

                    System.Windows.Forms.Label labelLeft = new System.Windows.Forms.Label();
                    labelLeft.AutoSize = true;
                    labelLeft.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
                    labelLeft.Location = new System.Drawing.Point(15, 187);
                    labelLeft.Name = "left";
                    labelLeft.Size = new System.Drawing.Size(96, 18);
                    labelLeft.TabIndex = 7;
                    groupBox.Controls.Add(labelLeft);

                    System.Windows.Forms.Label label4 = new System.Windows.Forms.Label();
                    label4.AutoSize = true;
                    label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
                    label4.Location = new System.Drawing.Point(260, 189);
                    label4.Name = "label4";
                    label4.Size = new System.Drawing.Size(25, 18);
                    label4.TabIndex = 25;
                    groupBox.Controls.Add(label4);

                    System.Windows.Forms.Label label6 = new System.Windows.Forms.Label();
                    label6.AutoSize = true;
                    label6.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
                    label6.Location = new System.Drawing.Point(260, 225);
                    label6.Name = "label6";
                    label6.Size = new System.Drawing.Size(25, 18);
                    label6.TabIndex = 26;
```

```
                groupBox.Controls.Add(label6);

                labelLeft2 = new System.Windows.Forms.Label();
                labelLeft2.AutoSize = true;
                labelLeft2.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
                labelLeft2.Location = new System.Drawing.Point(15, 249);
                labelLeft2.Name = "label12";
                labelLeft2.Size = new System.Drawing.Size(104, 18);
                labelLeft2.TabIndex = 24;
                groupBox.Controls.Add(labelLeft2);

                labelRight2 = new System.Windows.Forms.Label();
                labelRight2.AutoSize = true;
                labelRight2.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
                labelRight2.Location = new System.Drawing.Point(15, 279);
                labelRight2.Name = "label11";
                labelRight2.Size = new System.Drawing.Size(113, 18);
                labelRight2.TabIndex = 23;
                groupBox.Controls.Add(labelRight2);

                labelCel1 = new System.Windows.Forms.Label();
                labelCel1.AutoSize = true;
                labelCel1.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
                labelCel1.Location = new System.Drawing.Point(260, 281);
                labelCel1.Name = "label16";
                labelCel1.Size = new System.Drawing.Size(25, 18);
                labelCel1.TabIndex = 28;
                groupBox.Controls.Add(labelCel1);

                labelCel2 = new System.Windows.Forms.Label();
                labelCel2.AutoSize = true;
                labelCel2.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
                labelCel2.Location = new System.Drawing.Point(260, 251);
                labelCel2.Name = "label15";
                labelCel2.Size = new System.Drawing.Size(25, 18);
                labelCel2.TabIndex = 28;
                groupBox.Controls.Add(labelCel2);

                return (labelX, labelY, labelScale, labelA, labelRight, labelLeft, label4,
label6, labelLeft2, labelRight2, labelCel1, labelCel2, labelBase);
            }

            public static void UpdateLabelsTree(
                            System.Windows.Forms.Label labelX, string textX,
                            System.Windows.Forms.Label labelY, string textY,
                            System.Windows.Forms.Label labelScale, string textScale,
                            System.Windows.Forms.Label labelA, string textA,
                            System.Windows.Forms.Label labelRight, string textRight,
                            System.Windows.Forms.Label labelLeft, string textLeft,
```

```
                System.Windows.Forms.Label label4, string text4,
                System.Windows.Forms.Label label6, string text6,
                System.Windows.Forms.Label labelBase, string textBase
               )
{
    labelX.Text = textX;
    labelY.Text = textY;
    labelScale.Text = textScale;
    labelA.Text = textA;
    labelRight.Text = textRight;
    labelLeft.Text = textLeft;
    label4.Text = text4;
    label6.Text = text6;
    labelBase.Text = textBase;
}
public static void UpdateLabelsTree4(

                System.Windows.Forms.Label labelLeft2, string textLeft2,
                System.Windows.Forms.Label labelRight2, string textRight2,
                System.Windows.Forms.Label labelCel1, string textCel1,
                System.Windows.Forms.Label labelCel2, string textCel2)
{

    labelLeft2.Text = textLeft2;
    labelRight2.Text = textRight2;
    labelCel1.Text = textCel1;
    labelCel2.Text = textCel2;
}

public static void UpdateLabelsHilbert(
     System.Windows.Forms.Label labelX, string textX,
     System.Windows.Forms.Label labelY, string textY,
     System.Windows.Forms.Label labelScale, string textScale,
     System.Windows.Forms.Label labelA, string textA)
{
    labelX.Text = textX;
    labelY.Text = textY;
    labelScale.Text = textScale;
    labelA.Text = textA;
}
public static void UpdateLabelsGopster(
               System.Windows.Forms.Label labelX, string textX,
               System.Windows.Forms.Label labelY, string textY,
               System.Windows.Forms.Label labelScale, string textScale,
               System.Windows.Forms.Label labelA, string textA,
               System.Windows.Forms.Label labelRight, string textRight,
               System.Windows.Forms.Label labelLeft, string textLeft
              )
{
    labelX.Text = textX;
    labelY.Text = textY;
    labelScale.Text = textScale;
    labelA.Text = textA;
```

```
            labelRight.Text = textRight;
            labelLeft.Text = textLeft;
        }

        public static void SetLabelRight2Visibility(bool visible)
        {
            if (labelRight2 != null)
            {
                labelRight2.Visible = visible;
            }
        }

        public static void SetLabelLeft2Visibility(bool visible)
        {
            if (labelLeft2 != null)
            {
                labelLeft2.Visible = visible;
            }
        }

        public static void SetLabelCel1Visibility(bool visible)
        {
            if (labelCel1 != null)
            {
                labelCel1.Visible = visible;
            }
        }

        public static void SetLabelCel2Visibility(bool visible)
        {
            if (labelCel2 != null)
            {
                labelCel2.Visible = visible;            }        }    }}
```

ДОДАТОК З

Код сплеш-скрин форми

```
namespace PythagorasTree
{
    public partial class Enter : Form
    {
        private bool isDragging = false;
        private Point lastCursor;
        private Point lastForm;
        private int count = 0;
        private bool form1Opened = false;

        public Enter()
        {
            InitializeComponent();
            this.FormBorderStyle = FormBorderStyle.None;
```

```
    timer1.Interval = 1000;
    timer1.Tick += timer1_Tick;
    timer1.Start();
}

private void timer1_Tick(object sender, EventArgs e)
{
    count++;
    if (count >= 5 && !form1Opened)
    {
        timer1.Stop();
        OpenForm1();
        form1Opened = true;
    }
}

private void OpenForm1()
{
    basicform form1 = new basicform();
    form1.Show();
    this.Hide();
}


private void Enter_Load(object sender, EventArgs e)
{
    int deltaX = Screen.PrimaryScreen.Bounds.Width / 2 - this.Width / 2;
    int deltaY = Screen.PrimaryScreen.Bounds.Height / 2 - this.Height / 2;
    this.Location = new Point(deltaX, deltaY);
}    }}
```