*Заявка*

*на участие в научно-практической конференции «Старт в инновации»*

*Фамилия, Имя, Отчество Капаклы Вадим Сергеевич*

*Число, месяц, год рождения 09.11.2004*

*Город, место учебы, класс г. Апрелевка, Апрелевская средняя общеобразовательная школа №3 с углубленным изучением отдельных предметов, 10 «Б» класс*

*Секция Числа и данные*

*Название доклада Фракталы*

*Предметные олимпиады Математика, Информатика*

*Контактный телефон, e-mail 89268850633, kapakly-vadim@mail.ru*

Капаклы Вадим, 10 класс, МАОУ Апрелевская СОШ № 3, г. Апрелевка

# Фракталы

Руководитель: Гузева Елена Геннадьевна

**Оглавление**

## Введение

**Актуальность работы.** Окружающий мир нельзя описать только с помощью правильных геометрических фигур и тел. Природные объекты в большинстве своем имеют фрактальную структуру. Фракталы позволяют построить математическую модель, максимально приближенную к реальному миру с его ветвистостью, шероховатостью и изъеденностью, одним словом, смоделировать тот неидеальный мир, который вокруг нас. Знания о фракталах необходимы и математикам, и физикам, и художникам, и программистам, и специалистам других направлений.

**Обоснование выбора.** Слегка ознакомившись с данной темой, возник интерес изучать и развивать ее дальше, не останавливаться на уже полученных знаниях.

**Цель:** создать и проследовать свой фрактал.

**Задачи:**

- изучить теоретический материал;

- познакомиться с наиболее известными фракталами;

- познакомиться с методами построения фракталов;

- на основе изученного сформулировать правило для построения нового фрактала;

- вывести основные математические закономерности для нового фрактала;

- выбрать наиболее подходящие алгоритмы для построения фракталов на персональном компьютере.

## 1. Основная часть

### 1. 1. Теоретическая часть. Что же такое фрактал?

Разделим отрезок прямой на $N$ равных частей. Тогда каждую часть можно считать копией всего отрезка уменьшенной в $1/r$ раз. Очевидно, $N$ и $r$ связаны соотношением $Nr = 1$. Если квадрат разбить на $N$ равных квадратов (с площадью, в $1/r^2$ раз меньше исходного), то соотношение запишется как $Nr^2 = 1$. Если куб разделить на $N$ равных кубов (с объёмом, в $1/r^3$ раз меньше исходного), то соотношение примет следующий вид: $Nr^3 = 1$. Заметим, что размерность d объекта, будь то одномерный отрезок, двумерный квадрат или трехмерный куб, появляется как степень $r$ в соотношении между $N$, числом равных подобъектов, и коэффициентом подобия $r$. А именно:

$$Nr^d = 1.$$

Прологарифмируем обе части уравнения:

$$\ln(Nr^d) = \ln 1$$

$$\ln N \ln r^d = 0$$

$$\ln N + d\ln r = 0$$

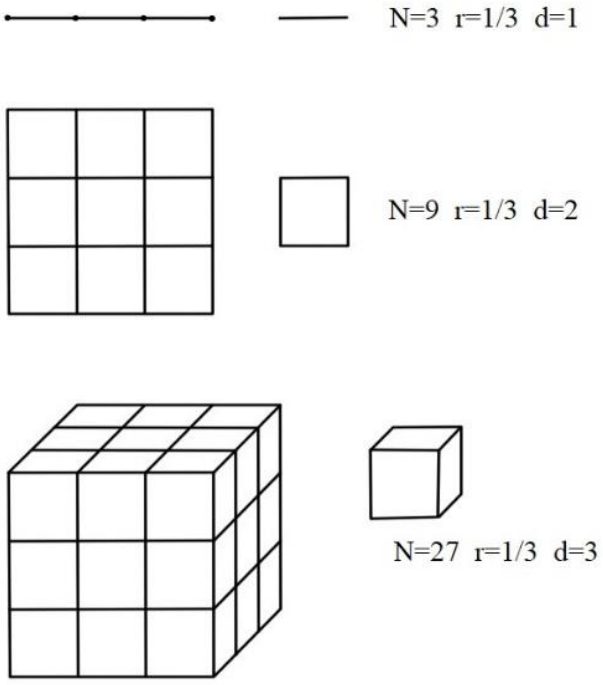
$$d \ln r = -\ln N$$

$$d = -\frac{\ln N}{\ln r}$$

$$d = \frac{\ln N}{(-1)\ln r}$$

$$d = \frac{\ln N}{\ln r^{-1}}$$

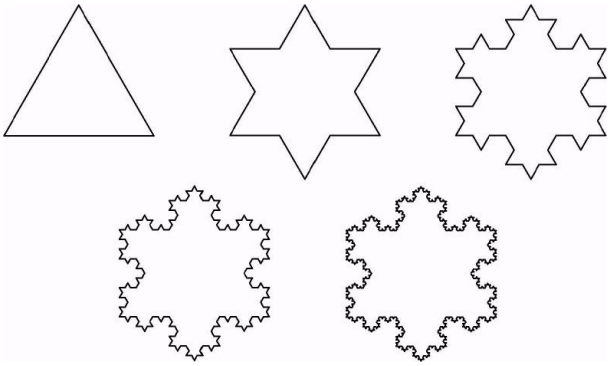
$$d = \frac{\ln N}{\ln \frac{1}{r}}$$




Величина $d$ может не являться целым числом, тогда такое множество называют самоподобным фракталом, $d$ – фрактальная (дробная) размерность.

### 1.2. Снежинка Коха.

Эта фигура — один из первых, исследованных учеными фракталов. Граница *снежинки*, придуманной Гельгом фон Кохом в 1904 году, составлена из трех одинаковых непрерывных линий, к которым нельзя провести касательную ни в одной точке, называются эти линии *кривыми Коха*, и являются фракталами с размерностью $d \approx 1{,}2618$. Каждая треть снежинки строится итеративно, начиная с одной из сторон равностороннего треугольника. Пусть $K_0$ – начальный отрезок. Уберем среднюю треть и добавим два новых отрезка такой же длины. Назовем полученное множество $K_1$. Повторим данную процедуру многократно, на каждом шаге заменяя среднюю треть двумя новыми отрезками. Обозначим через $K_n$ фигуру, получившуюся после $n$-го шага.



Интуитивно ясно, что последовательность кривых $\{K_n\}_{n=1}^{\infty}$ сходится к некоторой предельной кривой *K*.

Если взять копию *K*, уменьшенную в три раза (*r* = 1/3), то все множество *K* можно составить из $N = 4$ таких копий. Следовательно, отношение самоподобия выполняется при указанных N и r, а размерность фрактала будет:

$$d = \ln 4 / \ln 3 \approx 1{,}2618$$

*Граница снежинки Коха имеет бесконечную длину*

Достаточно показать, что каждый из трех идентичных фракталов *K*, полученных итерациями, имеет бесконечную длину. Пусть исходный отрезок $K_0$ имеет единичную длину. Тогда длина кривой $K_1$ равна 4/3. Длина кривой $K_2$ равна $4^n/3^n$. Продолжая таким образом имеем, что кривая К, после *n*-го шага имеет длину $4^n/3^n$. Следовательно, длина предельной кривой *K* равна бесконечности:

$$\lim_{n\to\infty} 4^n/3^n = \infty$$

*Снежинка Коха имеет площадь*

Площадь снежинки Коха равна сумме площадей всех треугольников, из которых она состоит. Пусть сторона исходного правильного треугольника равна $a$, а площадь этого треугольника равна $S_0$, тогда площадь снежинки Коха после перовой итерации равна $S_1$, после второй $S_2$, после *n*-ой $S_n$. Зная логику построения Снежинки Коха, мы с уверенностью можем сказать, что при каждой итерации образуются треугольники, подобные исходному, имеющие коэффициент подобия $k = 1/3$. Причем с каждой итерацией их количество увеличивается с геометрической прогрессией, знаменатель которой $q = 4$.

$$S_0 = \frac{\sqrt{3}}{4} a^2$$

$$S_1 = S_0 + S_0 \cdot \frac{1}{9} \cdot 3 = S_0 \left(1 + \frac{4}{9} \cdot \frac{3}{4}\right)$$

$$S_2 = S_0 + S_0 \cdot \frac{1}{9} \cdot 3 + S_0 \cdot \frac{1}{9^2} \cdot 3 \cdot 4 = S_0 \left(1 + \frac{4}{9} \cdot \frac{3}{4} + \left(\frac{4}{9}\right)^2 \cdot \frac{3}{4}\right)$$

$$S_3 = S_0 + S_0 \cdot \frac{1}{9} \cdot 3 + S_0 \cdot \frac{1}{9^2} \cdot 3 \cdot 4 + S_0 \cdot \frac{1}{9^3} \cdot 4^2 \cdot 3 = S_0 \left(1 + \frac{4}{9} \cdot \frac{3}{4} + \left(\frac{4}{9}\right)^2 \cdot \frac{3}{4} + \left(\frac{4}{9}\right)^3 \cdot \frac{3}{4}\right)$$

$$S_n = S_0\left(1 + \frac{4}{9}\cdot\frac{3}{4} + \left(\frac{4}{9}\right)^2\cdot\frac{3}{4} + \left(\frac{4}{9}\right)^3\cdot\frac{3}{4} + \cdots + \left(\frac{4}{9}\right)^n\cdot\frac{3}{4} + \cdots\right) = S_0\left(1 + \frac{3}{4}\sum_{n=1}^{\infty}\left(\frac{4}{9}\right)^n\right)$$

$\left\{\left(\frac{4}{9}\right)^n\right\}$ – бесконечно убывающая геометрическая прогрессия.

Распишем последовательность $\frac{3}{4}\sum_{n=1}^{\infty}\left(\frac{4}{9}\right)^n$:

$$S = \frac{b_1}{1-q}, \qquad b_1 = \frac{4}{9} \quad q = \frac{4}{9}$$

$$S = \frac{\frac{4}{9}}{1-\frac{4}{9}} = \frac{4}{5}$$

Подставим значение:

$$S = S_0\left(1 + \frac{4}{5}\cdot\frac{3}{4}\right) = \frac{8}{5}S_0 = \frac{8}{5}\cdot\frac{\sqrt{3}}{4}a^2$$

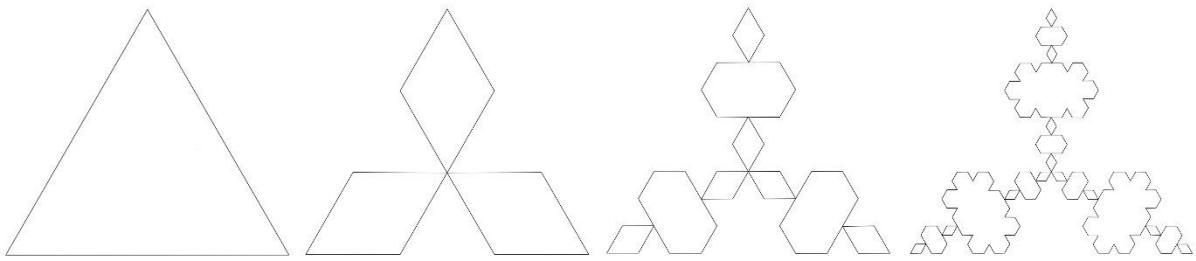
Таким образом, мы можем сделать вывод, что площадь можно вычислить по формуле:

$$S = \frac{2\sqrt{3}}{5}a^2$$

Если $a$ = 1, то $S = \frac{2\sqrt{3}}{5}$.

### 1.2. Обратная снежинка Коха.

У снежинки Коха есть множество вариаций, одной из которых является фрактал с названием *обратная снежинка Коха*. Этот фрактал очень многим схож со снежинкой Коха, начиная от свойств, заканчивая построением. Он также имеет бесконечную длину, как и снежинка Коха, и собственную площадь, которую мы можем вычислить. Общая идея построения обратной снежинки Коха точно такая же, как и у снежинки Коха. Алгоритмы их построения отличаются лишь тем, что для построения одной снежинки треугольники достраиваются к исходному треугольнику с каждой итерацией, а для построения второй, наоборот – вырезаются.



Поэтому для вычисления площади обратной снежинки Коха мы будем вычитать из площади исходного треугольника площади треугольников, образованных итерациями. Тогда формула для вычисления площади обратной снежинки будет выглядеть так:

$$S = S_0\left(1 - \frac{4}{5}\cdot\frac{3}{4}\right) = \frac{2}{5}S_0 = \frac{2}{5}\cdot\frac{\sqrt{3}}{4}a^2 = \frac{\sqrt{3}}{10}a^2$$

Если $a$ = 1, то $S = \frac{\sqrt{3}}{10}$.

Зная формулы для вычисления площади снежинки Коза и обратной снежинки Коха, мы можем узнать отношение площадей двух снежинок для одного и того же исходного треугольника:

$$\frac{S_{\text{снеж.Коха}}}{S_{\text{обр.снеж.Коха}}} = \frac{2\sqrt{3}}{5} \div \frac{\sqrt{3}}{10} = 4$$

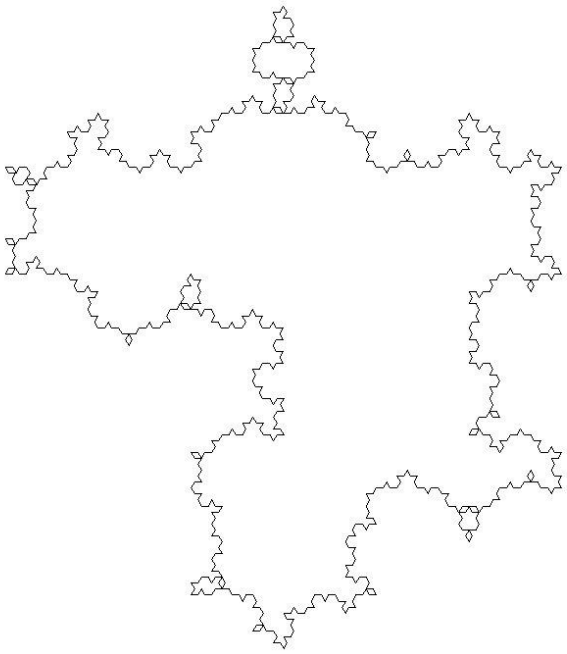
Таким образом, можем сделать вывод, что площадь снежинки Коха, основанной на одном исходном треугольнике, в 4 раза больше площади обратной снежинки Коха, основанной на том же треугольнике.

### 1.3. Случайные возмущения.

Однако рассмотренные нами фракталы обладают одним явным недостатком, ограничивающим их применение для моделирования естественных объектов. Они детерминированы. Хотя каждый может распознать кленовый лист, фактически, никакие два листа не будут в точности подобны друг другу. Одной из причин такого положения вещей является то, что случайность есть неотъемлемое свойство реального мира. Конечно, с большой степенью достоверности можно предполагать, что существует детерминированный генетический код для кленового листа. Однако реальный рост листа в значительной степени зависит от таких факторов, как наличие воды, солнечного света, питательных веществ, а также болезней и множества других подобных воздействий. Все эти факторы окружающей среды приводят к возмущениям в процессах роста и, следовательно, определяют появление случайных различий в конфигурациях листа.

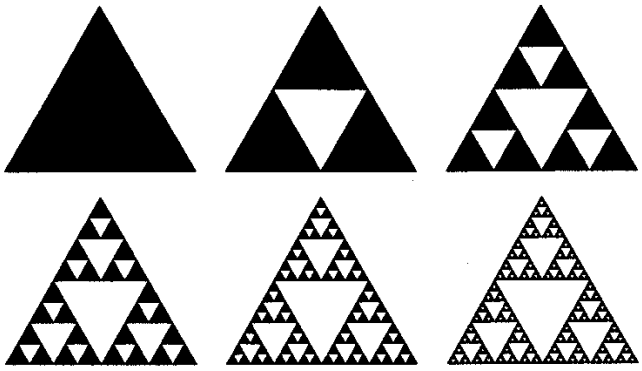
Можно построить рандомизированную снежинку Коха, добавляя на каждом шаге равносторонние треугольники, обращенные как внутрь, так и наружу.

Если на каждом шаге мы сделаем выбор направления внутрь или наружу случайным, то в итоге получим фрактальную кривую, которая по-прежнему имеет размерность $d = \ln 4/\ln 3 \approx 1{,}2618$ (при условии, что перекрытий не слишком много). Кривые, полученные таким образом, могут использоваться для моделирования береговых линий островов.



**1.4. Ковер Серпинского.**

Еще один пример простого самоподобного фрактала - ковер Серпинского, придуманный польским математиком Вацлавом Серпинским в 1915 году. Сам термин *ковер* (gasket) принадлежит Мандельброту. В способе построения, следующем ниже, мы начинаем с некоторой области и последовательно выбрасываем внутренние подобласти.



Пусть, начальное множество $S_0$ – равносторонний треугольник вместе с областью, которую он замыкает. Разобьем $S_0$ на четыре меньшие треугольные области, соединив отрезками середины сторон исходного треугольника. Удалим внутренность маленькой центральной треугольной области. Назовем оставшееся множество $S_1$.Затем повторим процесс для каждого из трех оставшихся маленьких треугольников и получим следующее приближение $S_2$. Продолжая таким образом, получим последовательность вложенных множеств $S_n$, чье пересечение и образует ковер S.

Из построения видно, что весь ковер представляет собой объединение $N = 3$ существенно непересекающихся уменьшенных в два раза копий; коэффициент подобия $r = 1/2$ (как по горизонтали, так и по вертикали). Следовательно, $S$ самоподобный фрактал с размерностью:

$$d = \ln 3/\ln 2 \approx 1{,}5850$$

Очевидно, что суммарная площадь частей, выкинутых при построении, в точности равна площади исходного треугольника. На первом шаге мы выбросили $1/4$ часть площади. На следующем шаге мы выбросили три треугольника, причем площадь каждого равна $1/4^2$ площади исходного. Рассуждая таким образом, мы убеждаемся, что полная доля выкинутой площади составила:

$$1/4 + 3(1/4^2) + 3^2\,(1/4^3) + \cdots + 3^{n-1}\,(1/4^n) + \cdots =$$

$$= \frac{1}{3}\left(\left(\frac{3}{4}\right)^1 + \left(\frac{3}{4}\right)^2 + \left(\frac{3}{4}\right)^3 + \cdots + \left(\frac{3}{4}\right)^n + \cdots\right) =$$

$$= \frac{1}{3}\left(-1 + 1 + \left(\frac{3}{4}\right)^1 + \left(\frac{3}{4}\right)^2 + \left(\frac{3}{4}\right)^3 + \cdots + \left(\frac{3}{4}\right)^n + \cdots\right) =$$

$$= \frac{1}{3}\left(-1 + \sum_{n=1}^{\infty}\left(\frac{3}{4}\right)^n\right) =$$

Распишем отдельно последовательность $\sum_{n=1}^{\infty}\left(\frac{3}{4}\right)^n$ :

$$S = \frac{b_1}{1-q}, \qquad b_1 = 1 \quad q = \frac{3}{4}$$

Подставим значение:

$$= \frac{1}{3}\left(-1 + \frac{1}{1-\frac{3}{4}}\right) = \frac{1}{3}(-1+4) = 1$$

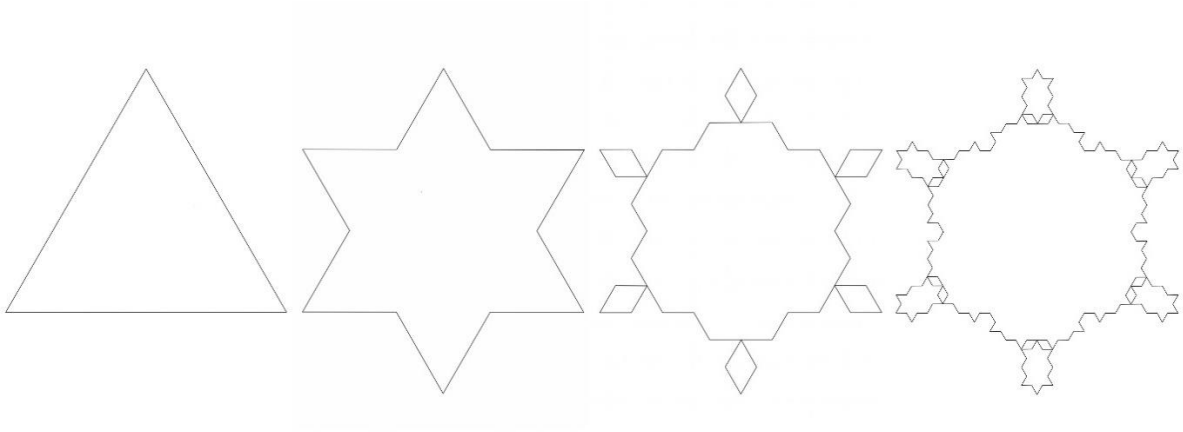
Следовательно, мы можем утверждать, что оставшееся множество *S*, то есть ковер, имеет площадь меры нуль. Это выделяет множество S в разряд «совершенного», в том числе, что оно разбивает свое дополнение на бесконечное число треугольных областей, обладая при этом нулевой толщиной.

## 2. Практическая часть.

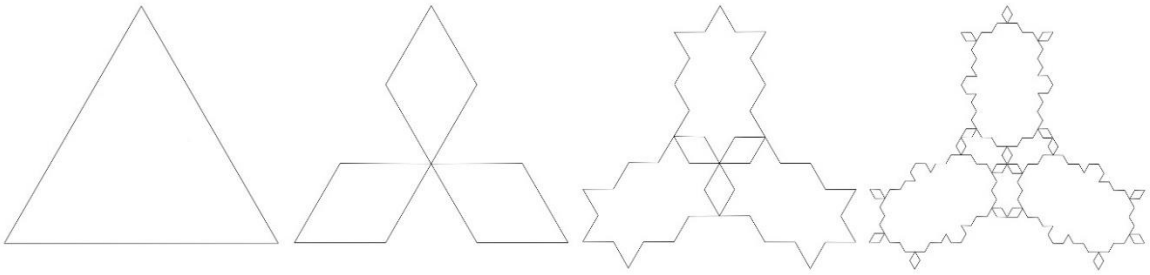
### 2.1. Новые фракталы Тригон и Гексагон на основе снежинки Коха и обратной снежинки Коха.

Изучив снежинку Коха и обратную снежинку Коха, мне в голову пришла идея о двух новых фракталах, включающих в себя элементы этих двух снежинок Коха. Я подумал, а что если объединить идеи двух этих фракталов в одну? А именно, создать фрактал, в котором при каждой итерации будет чередоваться идея построение снежинок, упомянутых ранее. Так как построение двух этих фракталов основывается на равностороннем треугольнике, то и мои фракталы тоже будут основываться на правильном треугольнике. А далее, если при первой итерации мы использовали алгоритм построения снежинки Коха, то есть наращивали на исходный треугольник треугольники поменьше, то при второй итерации мы будем использовать алгоритм построения обратной снежинки Коха, и будем вырезать треугольники, а при третьей итерации снова наращивать. Так мы можем создать два фрактала: первый – *Гексагон*, у которого при нечетных итерациях будет использоваться алгоритм построения снежинки Коха, а при четных - алгоритм обратной снежинки Коха, и второй – *Тригон*, у которого при нечетных итерациях будет использоваться алгоритм построения обратной снежинки Коха, а при четных – алгоритм базовой снежинки Коха.

Этапы построения Гексагона:



Этапы построения Тригона:



Свойства этих фракталов так же будут схожи со свойствами обеих снежинок Коха. Они также будут обладать бесконечной длиной и также будут иметь площадь. Попробуем вывести формулы для нахождения площадей двух этих фракталов, таким же способом, как и до этого.

Сначала выведем формулу площади фрактала, первая итерация которого приходит по алгоритму обратной снежинки Коха, обозначим ее как $S_{\text{Тригона}}$:

$$S_0 = \frac{\sqrt{3}}{4}a^2$$

$$S_1 = S_0 - S_0 \cdot \frac{1}{9} \cdot 3 = S_0\left(1 - \frac{4}{9} \cdot \frac{3}{4}\right)$$

$$S_2 = S_0 - S_0 \cdot \frac{1}{9} \cdot 3 + S_0 \cdot \frac{1}{9^2} \cdot 3 \cdot 4 = S_0\left(1 - \frac{4}{9} \cdot \frac{3}{4} + \left(\frac{4}{9}\right)^2 \cdot \frac{3}{4}\right)$$

$$S_3 = S_0 - S_0 \cdot \frac{1}{9} \cdot 3 + S_0 \cdot \frac{1}{9^2} \cdot 3 \cdot 4 - S_0 \cdot \frac{1}{9^3} \cdot 4^2 \cdot 3 = S_0\left(1 - \frac{4}{9} \cdot \frac{3}{4} + \left(\frac{4}{9}\right)^2 \cdot \frac{3}{4} - \left(\frac{4}{9}\right)^3 \cdot \frac{3}{4}\right)$$

$$S_{\text{Тригона}} = S_0\left(1 - \frac{3}{4}\sum_{n=0}^{\infty}\left(\frac{4}{9}\right)^{2n+1} + \frac{3}{4}\sum_{k=1}^{\infty}\left(\frac{4}{9}\right)^{2n}\right) = S_0\left(1 - \frac{3}{4}\left(\sum_{n=0}^{\infty}\left(\frac{4}{9}\right)^{2n+1} - \sum_{k=1}^{\infty}\left(\frac{4}{9}\right)^{2n}\right)\right)$$

Распишем отдельно последовательность $\sum_{n=0}^{\infty}\left(\frac{4}{9}\right)^{2n+1}$:

$$\frac{4}{9};\left(\frac{4}{9}\right)^{3};\left(\frac{4}{9}\right)^{5};\left(\frac{4}{9}\right)^{7};\cdots$$

$$S=\frac{b_1}{1-q},\qquad b_1=\frac{4}{9}\quad q=\left(\frac{4}{9}\right)^{2}$$

$$S=\frac{\frac{4}{9}}{1-\left(\frac{4}{9}\right)^{2}}=\frac{4}{9}\div\left(1-\frac{16}{81}\right)=\frac{36}{65}$$

Распишем отдельно последовательность $\sum_{k=1}^{\infty}\left(\frac{4}{9}\right)^{2n}$:

$$\left(\frac{4}{9}\right)^{2};\left(\frac{4}{9}\right)^{4};\left(\frac{4}{9}\right)^{6};\left(\frac{4}{9}\right)^{8};\cdots$$

$$S=\frac{b_1}{1-q},\qquad b_1=\left(\frac{4}{9}\right)^{2}\quad q=\left(\frac{4}{9}\right)^{2}$$

$$S=\frac{\left(\frac{4}{9}\right)^{2}}{1-\left(\frac{4}{9}\right)^{2}}=\frac{16}{81}\div\left(1-\frac{16}{81}\right)=\frac{16}{65}$$

Вернемся к нашей формуле и подставим значения последовательностей:

$$S_{\text{Тригона}}=S_0\left(1-\frac{3}{4}\left(\frac{36}{65}-\frac{16}{65}\right)\right)=S_0\cdot\frac{10}{13}$$

Делаем вывод, что площадь этого фрактала находится по формуле:

$$S_{\text{Тригона}}=\frac{10}{13}\cdot\frac{\sqrt{3}}{4}a^2=\frac{5\sqrt{3}}{26}a^2$$

Теперь выведем формулу площади фрактала, первая итерация которого проходит по алгоритму снежинки Коха, обозначим ее как $S_{\text{Гексагона}}$. Она выводится аналогично, поэтому пропустим некоторые этапы:

$$S_{\text{Гексагона}}=S_0\left(1+\frac{3}{4}\sum_{n=0}^{\infty}\left(\frac{4}{9}\right)^{2n+1}-\frac{3}{4}\sum_{k=1}^{\infty}\left(\frac{4}{9}\right)^{2n}\right)=$$

$$=S_0\left(1+\frac{3}{4}\left(\sum_{n=0}^{\infty}\left(\frac{4}{9}\right)^{2n+1}-\sum_{k=1}^{\infty}\left(\frac{4}{9}\right)^{2n}\right)\right)$$

Подставим значения последовательностей:

$$S_{\text{Гексагона}} = S_0\left(1 + \frac{3}{4}\left(\frac{36}{65} - \frac{16}{65}\right)\right) = S_0 \cdot \frac{16}{13}$$

Площадь этого фрактала находится по формуле:

$$S_{\text{Гексагона}} = \frac{16}{13} \cdot \frac{\sqrt{3}}{4} a^2 = \frac{4\sqrt{3}}{13} a^2$$

Теперь мы можем узнать отношение площадей этих двух фракталов для одного и того же исходного треугольника:

$$\frac{S_{\text{Гексагона}}}{S_{\text{Тригона}}} = \frac{4\sqrt{3}}{13} \div \frac{5\sqrt{3}}{26} = 1{,}6$$

Таким образом, можем сделать вывод, что площадь Гексагона, основанного на одном исходном треугольнике в 1,6 раза больше площади Тригона, основанного на том же треугольнике.

А так же узнаем отношения площадей $\frac{S_{\text{снеж.Коха}}}{S_{\text{Тригона}}}$ ; $\frac{S_{\text{снеж.Коха}}}{S_{\text{Гексагона}}}$ :

$$\frac{S_{\text{снеж.Коха}}}{S_{\text{Тригона}}} = \frac{2\sqrt{3}}{5} \div \frac{5\sqrt{3}}{26} = 2{,}08$$

$$\frac{S_{\text{снеж.Коха}}}{S_{\text{Гексагона}}} = \frac{2\sqrt{3}}{5} \div \frac{4\sqrt{3}}{13} = 1{,}3$$

### 2.2. Методы построения фракталов. L-системы

Понятие *L-систем*, тесно связанное с самоподобными фракталами, появилось только в 1968 году благодаря Аристриду Линденмаеру. Изначально L-системы были введены при изучении формальных языков, а так же использовались в биологоческих моделях селекции. С их помощью можно строить многие известные самоподобные фракталы, включая снежинку Коха и ковер Серпинского.

L-системы открывают путь к бесконечному разнообразию новых фракталов, что и послужило причиной их широкого применения в компьютерной графике для построения фракталов для построения фрактальных деревьев и растений.

Для графической реализации L-систем в качестве подсистемы вывода используется так называемая *тертл-графика* (turtle – черепаха). При этом точка (черепашка) движется по экрану дискретными шагами, как правило, прочерчивая свой след, но при необходимости может перемещаться без рисования. В нашем распоряжении имеются три параметра (*x, y, a*), где (*x, y*) – координаты черепахи, *a* – направление, в котором она смотрит. Черепашка обучена интерпретировать и выполнять последовательность команд, задаваемых кодовым словом, буквы которого читаются слева направо. Кодовое слово представляет собой результат работы L-системы и может включать следующие буквы:

F переместиться вперед на один шаг, прорисовывая след

b переместиться вперед на один шаг, *не* прорисовывая след

[ открыть ветвь

] закрыть ветвь

\+ увеличить угол альфа на величину $\theta$ (поворот направо на величину $\theta$)

\- уменьшить угол альфа на величину $\theta$(поворот налево на величину $\theta$)

Размер шага и величина приращения по углу $\theta$ задаются заранее и остаются неизменными для всех перемещений черепашки. Если начальное направление движения *a* (угол, отсчитываемый от положительного направления оси X), то полагаем *a* равным нулю.

L-система состоит из *алфавита*, слова инициализации, называемого *аксиомой* или *инициатором*, и набором *порождающих правил*, указывающих, как следует преобразовывать слово при переходе от уровня к уровню (от итерации к итерации). К примеру, можно заменить букву F с помощью порождающего правила *rule* = F- F++F-F, что соответствует L-системе для снежинки Коха. Символы +, -, ], [ не обновляются, а просто остаются на тех местах, где они встретились. Обновление букв в данном слове предполагается одновременным, то есть все буквы слова одного уровня обновляются раньше любой буквы следующего уровня.

### 2.3. L-система для снежинки Коха

L-система, соответствующая снежинке Коха, задается следующим образом:

$\theta = \pi/3$

Аксиома: F++F++F

Порождающее правило: *rule* = F-F++F-F

Графическое представление аксиомы F++F++F – равносторонний треугольник. Черепашка делает один шаг вперед, затем угол *a* увеличивается на $2\pi/3$ и черепашка делает еще один шаг вперед, угол *a* снова увеличивается на $2\pi/3$ и черепашка делает еще шаг.

На первом шаге каждая буква F слове-инициаторе F++F++F заменяется на F-F++F-F:

(F-F++F-F)++(F-F++F-F)++(F-F++F-F).

Убирая скобки, получаем:

F-F++F-F++ F-F++F-F++ F-F++F-F.

Повторяя этот процесс, на втором шаге получим:

F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++ F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F

и т.д.

Программа на языке Python приведена в *Приложении 1*.

### 2.4. L-система для обратной снежинки Коха, фракталов Тригон и Гексагон и случайного возмущения снежинки Коха

L-система, соответствующая обратной снежинке Коха, задается следующим образом:

$\theta = \pi/3$

Аксиома: F++F++F

Порождающее правило: *rule* = F+F--F+F

Программа на языке Python приведена в *Приложении 2*.

L-система, соответствующая фракталам Тригон, Гексагон и случайному возмущению снежинки Коха, задается следующим образом:

$\theta = \pi/3$

Аксиома: F++F++F

Порождающее правило: *rule1* = F-F++F-F

*rule2* = F+F--F+F

В случае фрактала Тригон для нечетных итераций выбирается rule2, а для четных – rule1.

Программа на языке Python приведена в *Приложении 3*.

В случае фрактала Гексагон для нечетных итераций выбирается rule1, а для четных – rule2.

Программа на языке Python приведена в *Приложении 4*.

А в случае случайного возмущения снежинки Коха эти правила будут выбираться случайным образом с вероятностью 1/2.

Программа на языке Python приведена в *Приложении 5*.

### 2.5. L-система для ковра Серпинского

L-система, соответствующая ковру Серпинского, задается следующим образом:

$\theta = 2\pi/3$

Аксиома: F

Порождающее правило: *rule1* = F-G+F+G-F

*rule2* = GG

В случае символа F применяется *rule1*, а в случае символа G применяет *rule2*.

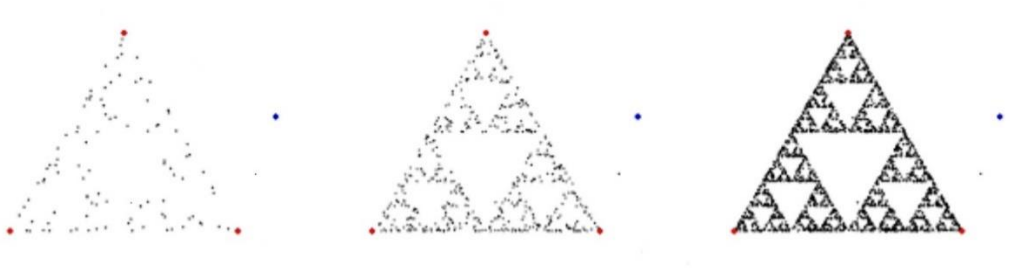
Программа на языке Python приведена в *Приложении 6*.

### 2.6. Построение ковра Серпинского с помощью игры «Хаос»

Когда мы говорим о методах построения фракталов нельзя не упомянуть о таком методе как *игра «Хаос»*, или, как его по-другому называют, *Метод случайных итераций*.

Для примера описания работы данного метода, рассмотрим его на основе треугольника. Суть «игры» такова. На плоскости зафиксирован правильный треугольник, а также в совершенно любом месте отмечена точка, от которой мы в дальнейшем будем отталкиваться. Затем случайным образом выбирают одну из трех вершин треугольника и ставят новую точку, которая будет делить пополам расстояние

от первоначальной точки до этой вершины. Тоже самое повторяют с новообразовавшейся точкой, опять случайным образом выбирают вершину, и повторяют те же действия. Повторяем эти действия с новыми точками. После *n*-ого количества повторений мы можем заметить, что из точек прорисовывается уже знакомая нам фигура, а именно *треугольник Серпинского*. Руководствуясь той же логикой, мы можем сыграть в эту игру, взяв за основу любую фигуру.



Программа на языке Python приведена в *Приложении 7*.

### 2.7. Применение L-систем для других известных фракталов: дракона Хартера-Хайтвея, дерева и куста

При построении фракталов с использованием только одного порождающего правила возникает следующая трудность. Мы не можем изменить направление чтения правила на некоторых шагах, то есть читать его не слева направо, а справа налево.

Для того, чтобы построить фрактал под названием «дракон Хартера-Хайтвея» необходимо иметь возможность менять направление чтения порождающего правила. В качестве инициатора, или аксиомы, используется кривая слева. Порождающее правило в данном случае заключается в том, чтобы нарисовать инициатор сначала в прямом, а затем в обратном направлении. Подобная схема не вписывается в рамки L-систем, использующих только одно порождающее правило. Эту проблему можно решить, введя две различных команды для передвижения вперед, например X и Y. Будем считать, что черепашка интерпретирует X и Y одинаково, то есть как один шаг вперед.

С помощью этих двух букв порождающее правило для дракона можно записать следующим образом:

$\theta = \pi/2$

Аксиома: XY

Порождающее правило: *rule1* = X+YF+

*rule2* = -FX-Y

В случае символа X применяется *rule1*, а в случае символа Y применяет *rule2*.

Программа на языке Python приведена в *Приложении 8*.

Когда мы встречаем символ [ (открыть ветвь), мы запоминаем положение и направление черепашки, то есть переменные (*x, y, a*), и возвращаемся к этим установкам позднее. Для хранения триплетов (*x, y, a*) используется стек $\begin{bmatrix} x_1 & y_1 & a_1 \\ x_2 & y_2 & a_2 \\ & \dots & \\ x_n & y_n & a_n \end{bmatrix}$, причем новые данные записываются в конец стека. Когда ветвь закрывается, переменным (*x, y, a*)

присваиваются значения, считанные из конца стека. Затем эти значения из стека удаляются.

Таким образом, ветвление задается двумя символами:

[ Открыть ветвь. Сохранить ($x$, $y$, $a$) в конце стека.

] Закрыть ветвь. Присвоить переменным ($x$, $y$, $a$) значения, считанные из конца стека, после чего удалить их из стека.

L-система, соответствующая дереву, задается следующим образом:

$0 \leq \theta \leq \pi/4$

Аксиома: XY

Порождающее правило: *rule* = F[@[-X]+X]

В случае символа X применяется *rule*, а символ @ позволяет изменять толщину ветви и ее цвет.

Программа на языке Python приведена в *Приложении 9*.

L-система, соответствующая кусту, задается следующим образом:

$\theta = \pi/8$

Аксиома: F

Порождающее правило: *rule* = -F+F+[+F-F-]-[-F+F+F]

В случае символа F применяется *rule*.

Программа на языке Python приведена в *Приложении 10*.

### 2.8. Универсальность фракталов. Фракталы в литературе

Фрактальные структуры настолько интересны и многогранны, что они нашли свое применение в том числе и в литературе.

При поиске фракталов в литературе, являющейся одним из видов искусств, будем рассматривать фрактал именно как произведение искусства, причем характеризующееся двумя основными характеристиками: 1) часть его неким образом подобна целому (в идеале, эта последовательность подобий распространяется на бесконечность); 2) его восприятие происходит по последовательности вложенных уровней. Заметим, что очарование фрактала как раз и возникает на пути следования по этой завораживающей и головокружительной системе уровней, возвращение с которой не гарантировано.

Самыми простым бесконечным текстом будет текст из бесконечного количества дублирующихся элементов, или куплетов, повторяющейся частью которого является его «хвост» – тот же текст с любым количеством отброшенных начальных куплетов. Схематически такой текст можно изобразить в виде не разветвляющегося дерева или периодической последовательности повторяющихся куплетов. Единица текста – фраза, строфа или рассказ, начинается, развивается и заканчивается, возвращаясь в исходную точку, точку перехода к следующей единице текста, повторяющей исходную.

Среди таких бесконечных произведений – стихи для детей или народные песенки, как, например, стишок о попе и его собаке из русской народной поэзии: «У попа был

двор, на дворе был кол, на колу мочало – не начать ли сказочку сначала?.. У попа был двор...»

Схематично эти тексты могут быть записаны как $«A \to A \to A \to A \to \ldots»$.

Другие стихотворения этого класса обладают схемой типа $«A \to B \to A \to B \ \ldots»$, например:

Еду я и вижу мост, под мостом ворона мокнет,
Взял ворону я за хвост, положил ее на мост, пусть ворона сохнет.
Еду я и вижу мост, на мосту ворона сохнет,
Взял ворону я за хвост, положил ее под мост, пусть ворона мокнет…

К последним текстам можно отнести знаменитую притчу о бабочке Чжуан Цзы, если перефразировать ее следующим образом: «Притча о философе, которому снится, что он бабочка, которой снится, что она философ, которому снится, что он бабочка, которой снится, что она философ, которому снится…».

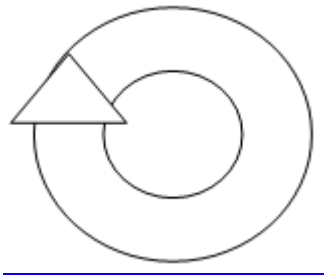
Заметим, что тема сна и сновидений еще встретится нам в других произведениях фрактальной литературы.

Подвариант схемы – $«A \to B \Rightarrow A \to B \Rightarrow \ldots»$: «Игры, в которые играют люди, которые играют в игры, в которые играют люди, которые…».

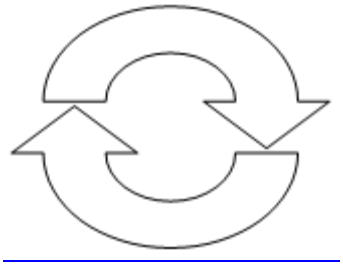
В качестве суждения В может выступать и суждение, обратное к $A: \neg A$. Тогда схему можно записать как: $«A \to \neg A \to A \to \neg A \to A \to \neg A \to A \to ...»$. К подобным текстам относится знаменитый парадокс «Лжеца», модифицированный в суждения, записанные на двух сторонах одной карточки. Первое из суждений гласит: «Утверждение на обратной стороне карточки ложно», а другое: «Утверждение на обратной стороне карточки истинно». Перефразируя их в единый бесконечный лексически фрактальный текст, получаем: «ложным является утверждение, что истинным является утверждение, что ложным является утверждение, что истинным…».

В работах А. Зенкина было показано, что «"истинной" формой "Лжеца" (и ему подобных парадоксов) является именно *бесконечное* "рассуждение"» приведенного выше вида. Именно такая запись позволяет наконец разрешить старинный парадокс, выводя читателя из «состояния интеллектуально-психологического шока», вызываемое традиционной конечной записью.

Фрактальный рисунок для такого рода стихотворений возникает из самой записи текста. Структуру их можно изобразить или в виде замкнутой кривой (первый случай) или двух замыкающих друг друга кривых (второй случай).



Змея, кусающаяся себя за хвост.



Две змеи, пожирающие друг друга.

В кольце может быть и большее число звеньев, как, например, в случае замкнутой логической схемы из трех элементов, описывающей строение стихотворения для детей Р. Сефа «Бесконечные стихи»:

Кто вечно хнычет и скучает –
Тот ничего не замечает,
Кто ничего не замечает –
Тот ничего не изучает,
Кто ничего не изучает,
Тот вечно хнычет и скучает.
Если скучно стало – почитай сначала.

Заметим, что последнее стихотворение, в отличие от большинства бесконечных песенок, предоставляет читателю выбор – кружиться и дальше на бесконечной карусели или сойти с нее: «Если скучно стало – почитай сначала».

Простота этих стишков не должна служить основанием для отказа им во фрактальной сущности, вспомним, что и фракталы Мандельброта, и кривые с дробной размерностью задаются достаточно простыми алгебраическими формулами или алгоритмами, и это также одна из особенностей фракталов – простая задача имеет необычайно сложные решения.

А первое основное свойство фракталов, свойство гомоморфности частного целому, нашим детским стихотворениям присуще.

В результате возникают в точности повторяющиеся «куплеты», которые производят бесконечную последовательность повторений, никогда и ничем не исчерпывающуюся. Эти повторы точны и математически строги.

## Заключение

В современном мире виртуальное пространство занимает порой не менее важное место в жизни людей, чем реальный мир. Фракталы позволяют воссоздать с помощью компьютерных вычислений реальный мир во всей его многогранности, неповторимости и красоте, создавая настоящее разнообразие похожих, но при этом таких разных форм. Без фракталов сейчас невозможно представить компьютерную графику, а также многие другие научные отрасти – например, теорию телетрафика и системы массового обслуживания, где изменения нагрузки начинают иметь самоподобный фрактальный характер. Инженеры активно внедряют фрактальные структуры в микросхемы, а биологи моделируют работу организма человека и его устройство на клеточном уровне с помощью фракталов.

Развитие фрактальных технологий на сегодняшний день – одна из прогрессирующих областей науки. Возможно, если ученым удастся докопаться до сути фракталов, человек начнет намного лучше понимать этот мир.

Мною были исследованы следующие фракталы: Снежинка Коха и ее разновидности, ковер Серпинского, фрактальные растения, дракон Хартера-Хайтвея. Были изучены методы построения фракталов. Также мною на основе снежинок Коха были созданы новые фракталы Тригон и Гексагон. Выведены формулы площадей для них и разновидностей снежинок Коха. Все фракталы были запрограммированны мною на языке программирования Питон для визуализации красоты и простоты работы L-систем. Также мною была создана модель пирамиды Серпинского.

Задачи выполнены, в ходе чего достигнута и цель.

«В течение многих лет я слышал высказывания, что фракталы создают только красивые картинки, но совершенно бесполезны. Меня огорчало такое непонимание сути вещей, потому что для осознания самого важного и сокрытого внутри всегда требуется время. В отношении фракталов нам не пришлось долго ждать…»

Бенуа Мандельброт

## Библиография

1. Введение в современную теорию динамических систем. Каток А. Б., Хасселблад Б., – М.: Факториал, 1999.

2. Введение в теорию фракталов. Морозов А.Д., 2002.

3. Детерминированный хаос. Шустер Г., – М.: Мир, 1988

4. Изучаем Python. Марк Лутц.

5. Изучаем программирование на Python. Пол Бэрри.

6. Красота фракталов. Х.-О. Пайтген, П. Х. Ритхер, - М. Мир, 1993.

7. Основы математического анализа. Рудин У., – М.: Мир, 1966.

8. Познание сложного. Николис Г., Пригожин И., – М.: Мир, 1990.

9. Порядок из хаоса: новый диалог человека с природой. Пригожин И., Стенгерс И. – М.: Мир, 1986.

10. Фракталы и хаос в динамических системах. Ричард М. Кронвер,

11. Фракталы и хаос. Множество Мандельброта и другие чудеса.; Мандельброт Б. Б.

12. Фракталы, хаос, степенные законы. Шредер М., – Ижевск: Изд. Дом «Удмуртский университет», 2000.

## Приложения

**Приложение 1. Программа на языке Python для построения *Снежинки Коха* с использованием L-систем.**

```
import turtle


#Screen settings
WIDTH, HEIGHT = 1400, 900
screen = turtle.Screen()
screen.setup(WIDTH, HEIGHT)
screen.screensize(3 * WIDTH, 3 * HEIGHT)
screen.bgcolor('black')
screen.delay(0)


#turtle settings
leo = turtle.Turtle()
leo.pensize(4)
leo.speed(1)
leo.setpos(-WIDTH // 6, HEIGHT // 6)
leo.color('gold')

#l-system settings
gens = 5
axiom = 'F++F++F'
chr_1, rule_1 = 'F', 'F-F++F-F'
step = 600
angle = 60

def apply_rules(axiom):
    return ''.join([rule_1 if chr == chr_1 else chr for chr in axiom])

def get_result(gens, axiom):
    for gen in range(gens):
        axiom = apply_rules(axiom)
    return axiom

for gen in range(gens):
    turtle.pencolor('white')
    turtle.goto(-WIDTH // 2 + 60,-HEIGHT // 2 - 100)
    turtle.clear()
    turtle.write(f'generation: {gens}', font=("Arial", 60, "normal"))

    leo.setheading(0)
    leo.goto(-WIDTH // 6, HEIGHT // 6)
```

```
    leo.clear()

    length = step // pow(3, gen)
    for chr in axiom:
        if chr == chr_1:
            leo.forward(length)
        elif chr == '+':
            leo.right(angle)
        elif chr == '-':
            leo.left(angle)
    axiom =apply_rules(axiom)

screen.exitonclick()
```

**Приложение 2. Программа на языке Python для построения *обратной Снежинки Коха* с использованием L-систем.**

```
        import turtle

#Screen settings
WIDTH, HEIGHT = 1400, 900
screen = turtle.Screen()
screen.setup(WIDTH, HEIGHT)
screen.screensize(3 * WIDTH, 3 * HEIGHT)
screen.bgcolor('black')
screen.delay(0)

#turtle settings
leo = turtle.Turtle()
leo.pensize(4)
leo.speed(1)
leo.setpos(-WIDTH // 6, HEIGHT // 6)
leo.color('gold')

#l-system settings
gens = 5
axiom = 'F++F++F'
chr_1, rule_1 = 'F', 'F+F--F+F'
step = 600
angle = 60

def apply_rules(axiom):
    return ''.join([rule_1 if chr == chr_1 else chr for chr in axiom])

def get_result(gens, axiom):
    for gen in range(gens):
        axiom = apply_rules(axiom)
    return axiom

for gen in range(gens):
    turtle.pencolor('white')
    turtle.goto(-WIDTH // 2 + 60,-HEIGHT // 2 - 100)
```

```
    turtle.clear()
    turtle.write(f'generation: {gens}', font=("Arial", 60, "normal"))

    leo.setheading(0)
    leo.goto(-WIDTH // 6, HEIGHT // 6)
    leo.clear()

    length = step // pow(3, gen)
    for chr in axiom:
        if chr == chr_1:
            leo.forward(length)
        elif chr == '+':
            leo.right(angle)
        elif chr == '-':
            leo.left(angle)
    axiom =apply_rules(axiom)

screen.exitonclick()
```

**Приложение 3. Программа на языке Python для построения *Тригона* с использованием L-систем.**

```
        import turtle

#Screen settings
WIDTH, HEIGHT = 1400, 900
screen = turtle.Screen()
screen.setup(WIDTH, HEIGHT)
screen.screensize(3 * WIDTH, 3 * HEIGHT)
screen.bgcolor('black')
screen.delay(0)

#turtle settings
leo = turtle.Turtle()
leo.pensize(1)
leo.speed(0)
leo.setpos(-WIDTH // 6, HEIGHT // 6)
leo.color('gold')

#l-system settings
gens = 6
axiom = 'F++F++F'
chr_1, rule_1 = 'F', 'F-F++F-F'
rule_2 = 'F+F--F+F'
step = 600
angle = 60

def apply_rules(axiom):
    if gen % 2 == 1:
        return ''.join([rule_1 if chr == chr_1 else chr for chr in axiom])
    else:
        return ''.join([rule_2 if chr == chr_1 else chr for chr in axiom])
```

```
def get_result(gens, axiom):
    for gen in range(gens):
        axiom = apply_rules(axiom)
    return axiom

for gen in range(gens):
    turtle.pencolor('white')
    turtle.goto(-WIDTH // 2 + 60,-HEIGHT // 2 - 100)
    turtle.clear()
    turtle.write(f'generation: {gens}', font=("Arial", 60, "normal"))

    leo.setheading(0)
    leo.goto(-WIDTH // 6, HEIGHT // 6)
    leo.clear()

    length = step // pow(3, gen)
    for chr in axiom:
        if chr == chr_1:
            leo.forward(length)
        elif chr == '+':
            leo.right(angle)
        elif chr == '-':
            leo.left(angle)
    axiom =apply_rules(axiom)

screen.exitonclick()
```

**Приложение 4. Программа на языке Python для построения *Гексагона* с использованием L-систем.**

```
import turtle

#Screen settings
WIDTH, HEIGHT = 1400, 900
screen = turtle.Screen()
screen.setup(WIDTH, HEIGHT)
screen.screensize(3 * WIDTH, 3 * HEIGHT)
screen.bgcolor('black')
screen.delay(0)

#turtle settings
leo = turtle.Turtle()
leo.pensize(1)
leo.speed(0)
leo.setpos(-WIDTH // 6, HEIGHT // 6)
leo.color('gold')

#l-system settings
gens = 6
axiom = 'F++F++F'
chr_1, rule_1 = 'F', 'F-F++F-F'
rule_2 = 'F+F--F+F'
```

```
step = 600
angle = 60

def apply_rules(axiom):
    if gen % 2 == 1:
        return ''.join([rule_2 if chr == chr_1 else chr for chr in axiom])
    else:
        return ''.join([rule_1 if chr == chr_1 else chr for chr in axiom])
def get_result(gens, axiom):
    for gen in range(gens):
        axiom = apply_rules(axiom)
    return axiom

for gen in range(gens):
    turtle.pencolor('white')
    turtle.goto(-WIDTH // 2 + 60,-HEIGHT // 2 - 100)
    turtle.clear()
    turtle.write(f'generation: {gens}', font=("Arial", 60, "normal"))

    leo.setheading(0)
    leo.goto(-WIDTH // 6, HEIGHT // 6)
    leo.clear()

    length = step // pow(3, gen)
    for chr in axiom:
        if chr == chr_1:
            leo.forward(length)
        elif chr == '+':
            leo.right(angle)
        elif chr == '-':
            leo.left(angle)
    axiom =apply_rules(axiom)

screen.exitonclick()
```

**Приложение 5. Программа на языке Python для построения фрактала снежинки со случайными возмущениями с использованием L-систем.**

```
        import turtle
import random

#Screen settings
WIDTH, HEIGHT = 1400, 900
screen = turtle.Screen()
screen.setup(WIDTH, HEIGHT)
screen.screensize(3 * WIDTH, 3 * HEIGHT)
screen.bgcolor('white')
screen.delay(0)

#turtle settings
leo = turtle.Turtle()
leo.pensize(1)
```

```
leo.speed(1)
leo.penup()
leo.hideturtle()
leo.setpos(-WIDTH // 6, HEIGHT // 6)
leo.pendown()
leo.color('black')

#l-system settings
gens = 5
axiom = 'F++F++F'
chr_1, rule_1 = 'F', 'F-F++F-F'
rule_2 = 'F+F--F+F'
step = 600
angle = 60

def apply_rules(axiom):
    res = ''
    for chr in axiom:
        t = random.randint(1,2)
        if chr == chr_1 and t % 2 == 0:
            res += rule_2
        if chr == chr_1 and t % 2 == 1:
            res += rule_1
        if chr == '+' or chr == '-':
            res += chr
    return res

def get_result(gens, axiom):
    for gen in range(gens):
        axiom = apply_rules(axiom)
    return axiom

for gen in range(gens):
    turtle.pencolor('white')
    turtle.goto(-WIDTH // 2 + 60,-HEIGHT // 2 - 100)
    turtle.clear()
    turtle.write(f'generation: {gens}', font=("Arial", 60, "normal"))

    leo.setheading(0)
    leo.goto(-WIDTH // 6, HEIGHT // 6)
    leo.clear()

    length = step // pow(3, gen)
    for chr in axiom:
        if chr == chr_1:
            leo.forward(length)
        elif chr == '+':
            leo.right(angle)
        elif chr == '-':
            leo.left(angle)
    axiom =apply_rules(axiom)
```

```
screen.exitonclick()
```

**Приложение 6. Программа на языке Python для построения *ковра Серпинского* с использованием L-систем.**

```
        import turtle

#Screen settings
WIDTH, HEIGHT = 900, 600
screen = turtle.Screen()
screen.setup(WIDTH, HEIGHT)
screen.screensize(3 * WIDTH, 3 * HEIGHT)
screen.bgcolor('black')
screen.delay(0)

#turtle settings
leo = turtle.Turtle()
leo.pensize(3)
leo.speed(0)
leo.setpos(-WIDTH // 3, -HEIGHT // 2)
leo.color('green')

#l-system settings
gens = 7
axiom = 'F'
chr_1, rule_1 = 'F', 'F-G+F+G-F'
chr_2, rule_2 = 'G', 'GG'
step = 8
angle = 120

def apply_rules(axiom):
    return ''.join([rule_1 if chr == chr_1 else
                rule_2 if chr == chr_2 else chr for chr in axiom])

def get_result(gens, axiom):
    for gen in range(gens):
        axiom = apply_rules(axiom)
    return axiom


turtle.pencolor('white')
turtle.goto(-WIDTH // 2 + 60,-HEIGHT // 2 + 60)
turtle.clear()
turtle.write(f'generation: {gens}', font=("Arial", 60, "normal"))

axiom = get_result(gens, axiom)
for chr in axiom:
    if chr == chr_1 or chr == chr_2:
        leo.forward(step)
    elif chr == '+':
        leo.right(angle)
```

```
    elif chr == '-':
        leo.left(angle)

screen.exitonclick()
```

**Приложение 7. Программа на языке Python для построения *ковра Серпинского* с использованием игры «Хаос».**

```
        import pygame
import random

pygame.init()

win = pygame.display.set_mode((600,600))

pygame.display.set_caption("Serpinsky triangle")

Triangle = [[400,50], [100,550], [500,550]]

x, y = random.randint(0,600), random.randint(0,600)

RUN = True
while RUN:
   for event in pygame.event.get():
      if event.type == pygame.QUIT:
         RUN = False
   A = random.randint(0,2)
   x, y = .5 * (x + Triangle[A][0]), .5 * (y + Triangle[A][1])

   pygame.draw.line(win, (255, 255, 255), (x, y), (x, y), 1)
   pygame.display.update()
pygame.quit()
```

**Приложение 8. Программа на языке Python для построения *дракона Хартера-Хайтвея* с использованием L-систем.**

```
        import turtle

#Screen settings
WIDTH, HEIGHT = 1400, 900
screen = turtle.Screen()
screen.setup(WIDTH, HEIGHT)
screen.screensize(3 * WIDTH, 3 * HEIGHT)
screen.bgcolor('black')
screen.delay(0)

#turtle settings
leo = turtle.Turtle()
leo.pensize(2)
leo.speed(0)
leo.setpos(WIDTH // 4, -HEIGHT // 4 - 25)
leo.color('magenta')
```

```
#l-system settings
gens = 13
axiom = 'XY'
chr_1, rule_1 = 'X', 'X+YF+'
chr_2, rule_2 = 'Y', '-FX-Y'
step = 4
angle = 90

def apply_rules(axiom):
    return ''.join([rule_1 if chr == chr_1 else
                    rule_2 if chr == chr_2 else chr for chr in axiom])

def get_result(gens, axiom):
    for gen in range(gens):
        axiom = apply_rules(axiom)
    return axiom


turtle.pencolor('white')
turtle.goto(-WIDTH // 2 + 60,-HEIGHT // 2 + 60)
turtle.clear()
turtle.write(f'generation: {gens}', font=("Arial", 60, "normal"))

axiom = get_result(gens, axiom)
for chr in axiom:
    if chr == chr_1 or chr == chr_2:
        leo.forward(step)
    elif chr == '+':
        leo.right(angle)
    elif chr == '-':
        leo.left(angle)

screen.exitonclick()
```

**Приложение 9. Программа на языке Python для построения дерева с использованием L-систем.**

```
        import turtle
from random import randint

#screen settings
WIDTH, HEIGH = 1600, 900
screen = turtle.Screen()
screen.setup(WIDTH, HEIGH)
screen.screensize(3 * WIDTH, 3 * HEIGH)
screen.bgcolor('black')
screen.delay(0)
#turtle settings
leo = turtle.Turtle()
leo.pensize(3)
leo.speed(0)
leo.setpos(0, -HEIGH // 2)
```

```
leo.color('green')
# l-system settings
gens = 9
axiom = 'XY'
chr_1, rule_1 = 'X', 'F[@[-X]+X]'
step = 80
angle =lambda: randint(0, 45)
stack = []
color = [0.35, 0.2, 0.0]
thickness = 20

def apply_rules(axiom):
    return ''.join([rule_1 if chr == chr_1 else chr for chr in axiom])

def get_result(gens, axiom):
    for gen in range(gens):
        axiom = apply_rules(axiom)
    return axiom

turtle.pencolor('white')
turtle.goto(-WIDTH // 2 + 60, -HEIGH // 2 - 100)
turtle.clear()
turtle.write(f'generation: {gens}', font=("Arial", 60, "normal"))

axiom = get_result(gens, axiom)
leo.left(90)
leo.pensize(thickness)
for chr in axiom:
    leo.color(color)
    if chr == 'F' or chr == 'X':
        leo.forward(step)
    elif chr == '@':
        step -=6
        color[1] +=0.04
        thickness -= 2
        thickness = max(1,thickness)
        leo.pensize(thickness)
    elif chr == '+':
        leo.right(angle())
    elif chr == '-':
        leo.left(angle())
    elif chr == '[':
        angle_, pos_ = leo.heading(), leo.pos()
        stack.append((angle_, pos_, thickness, step, color[1]))
    elif chr == ']':
        angle_, pos_, thickness, step, color[1] = stack.pop()
        leo.pensize(thickness)
        leo.setheading(angle_)
        leo.penup()
        leo.goto(pos_)
        leo.pendown()
```

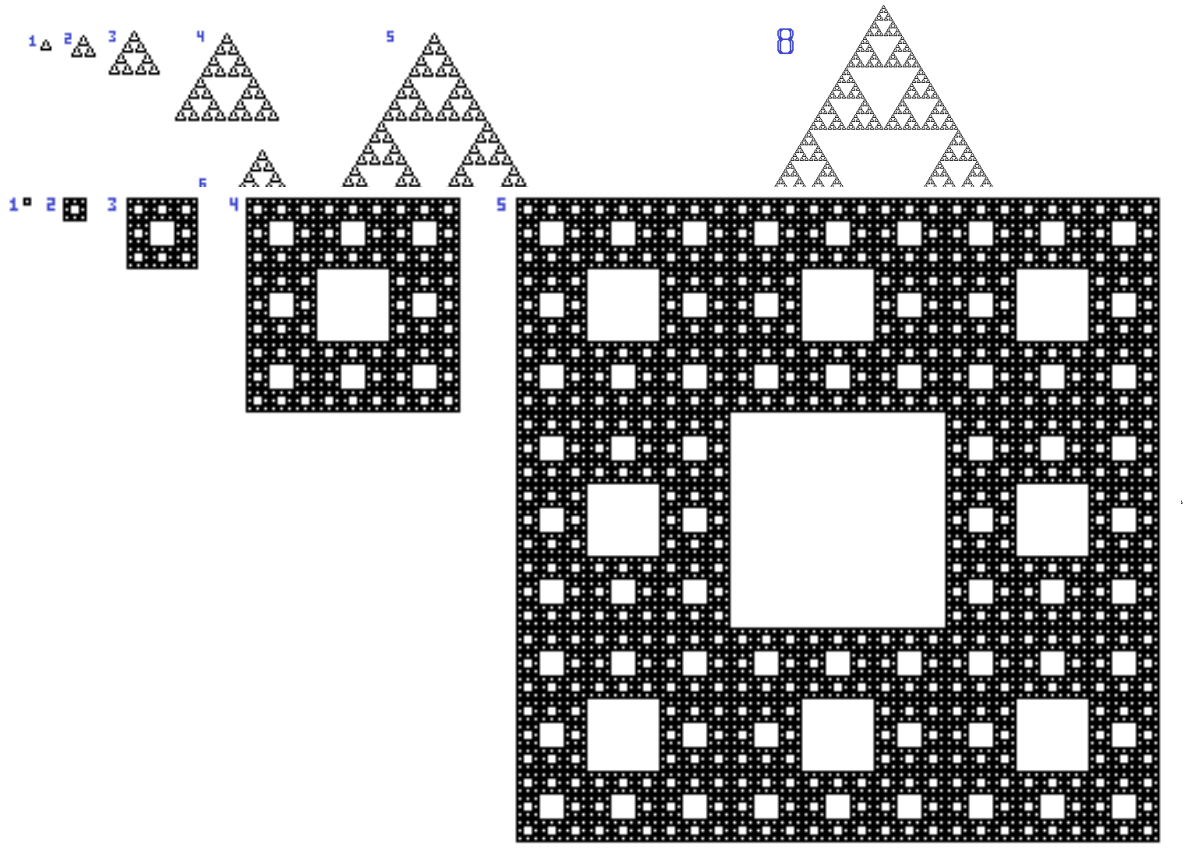
```
screen.exitonclick()
```

**Приложение 10. Программа на языке Python для построения куста с использованием L-систем.**

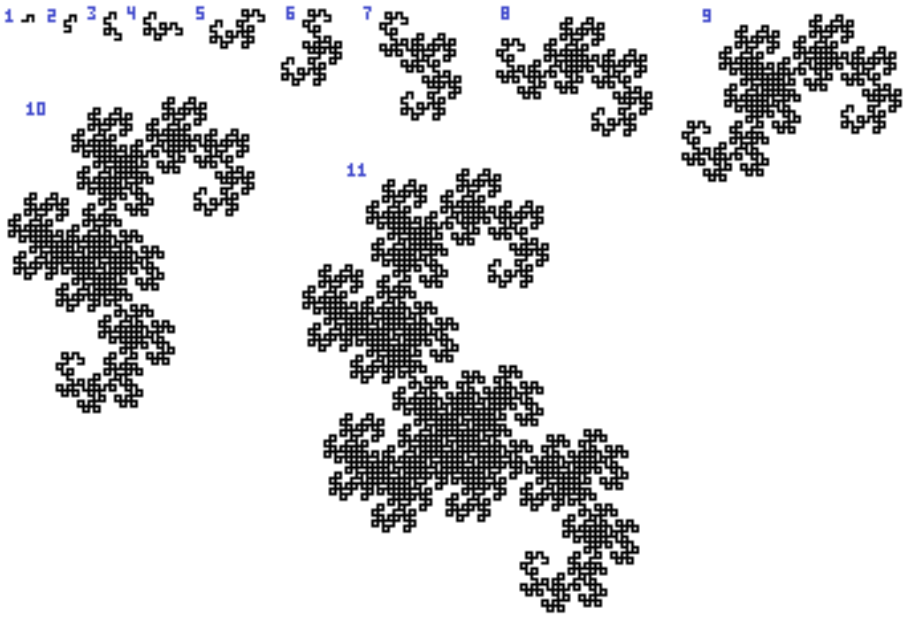
```
        import turtle
from random import randint

#screen settings
WIDTH, HEIGH = 900, 600
screen = turtle.Screen()
screen.setup(WIDTH, HEIGH)
screen.screensize(3 * WIDTH, 3 * HEIGH)
screen.bgpic('bottom.gif')
screen.bgcolor('black')
screen.delay(1)
#turtle settings
leo = turtle.Turtle()
leo.pensize(3)
leo.speed(0)
leo.penup()
leo.setpos(0, -HEIGH // 2)
leo.pendown()
leo.color('green')
# l-system settings
gens = 4
axiom = 'F'
chr_1, rule_1 = 'F', '-F+F+[+F-F-]-[-F+F+F]'
step = 13
angle =22.5
stack = []
color = [0.35, 0.2, 0.0]
thickness = 2

def apply_rules(axiom):
    return ''.join([rule_1 if chr == chr_1 else chr for chr in axiom])

def get_result(gens, axiom):
    for gen in range(gens):
        axiom = apply_rules(axiom)
    return axiom

turtle.pencolor('white')
#turtle.penup()
turtle.goto(-WIDTH // 2 + 60, -HEIGH // 2 - 100)
#turtle.pendown()
turtle.clear()
turtle.write(f'generation: {gens}', font=("Arial", 60, "normal"))

axiom = get_result(gens, axiom)
leo.left(60)
```

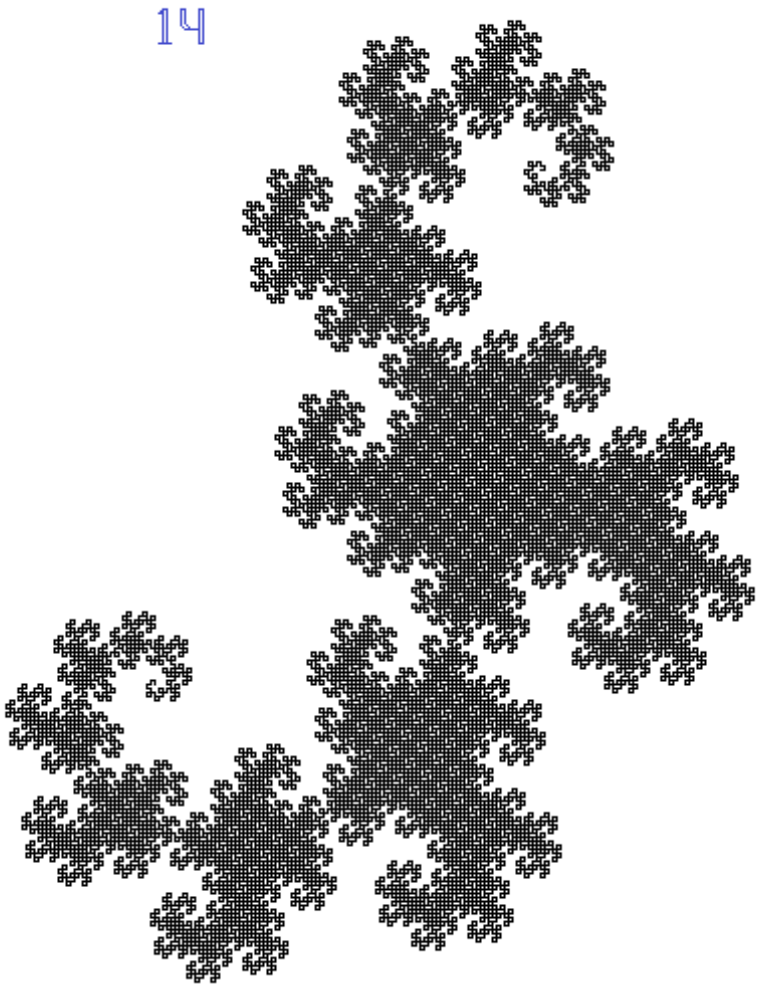
```
leo.pensize(thickness)
for chr in axiom:
    leo.color(color)
    if chr == 'F':
        leo.forward(step)
    elif chr == '+':
        leo.right(angle)
    elif chr == '-':
        leo.left(angle)
    elif chr == '[':
        angle_, pos_ = leo.heading(), leo.pos()
        stack.append((angle_, pos_, thickness, step, color[1]))
    elif chr == ']':
        angle_, pos_, thickness, step, color[1] = stack.pop()
        leo.pensize(thickness)
        leo.setheading(angle_)
        leo.penup()
        leo.goto(pos_)
        leo.pendown()

screen.exitonclick()
```

**Приложение 11. Фрактал *Ковер Серпинского* (треугольник и квадрат), созданный в графическом редакторе Paint:**

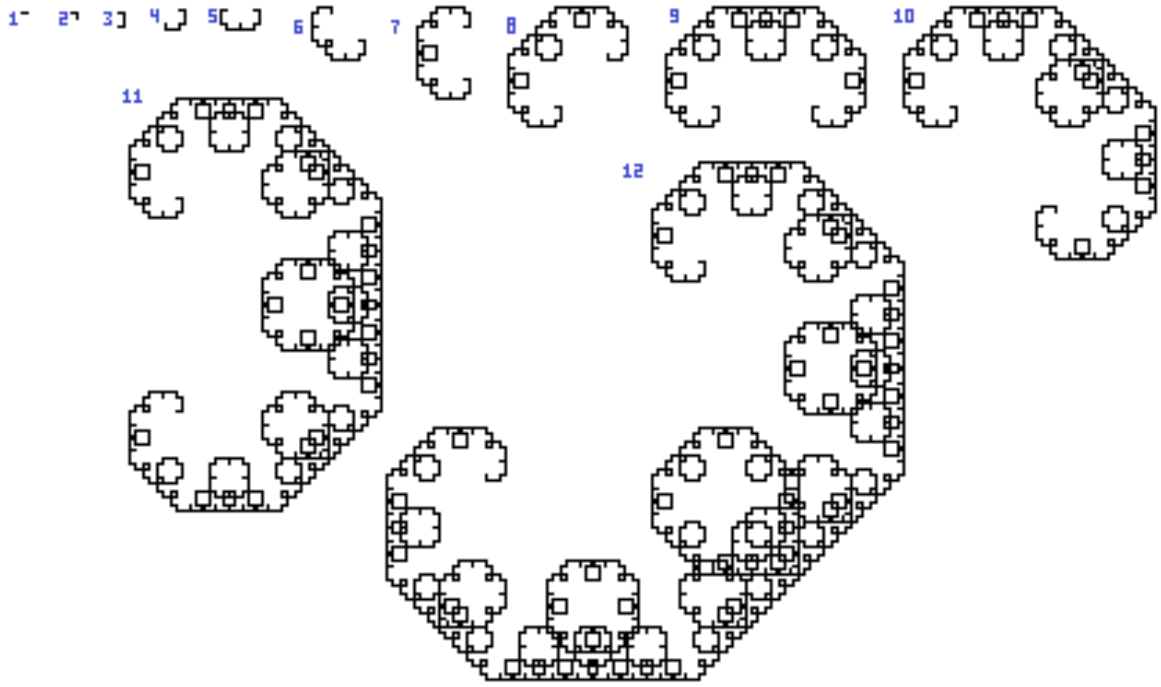


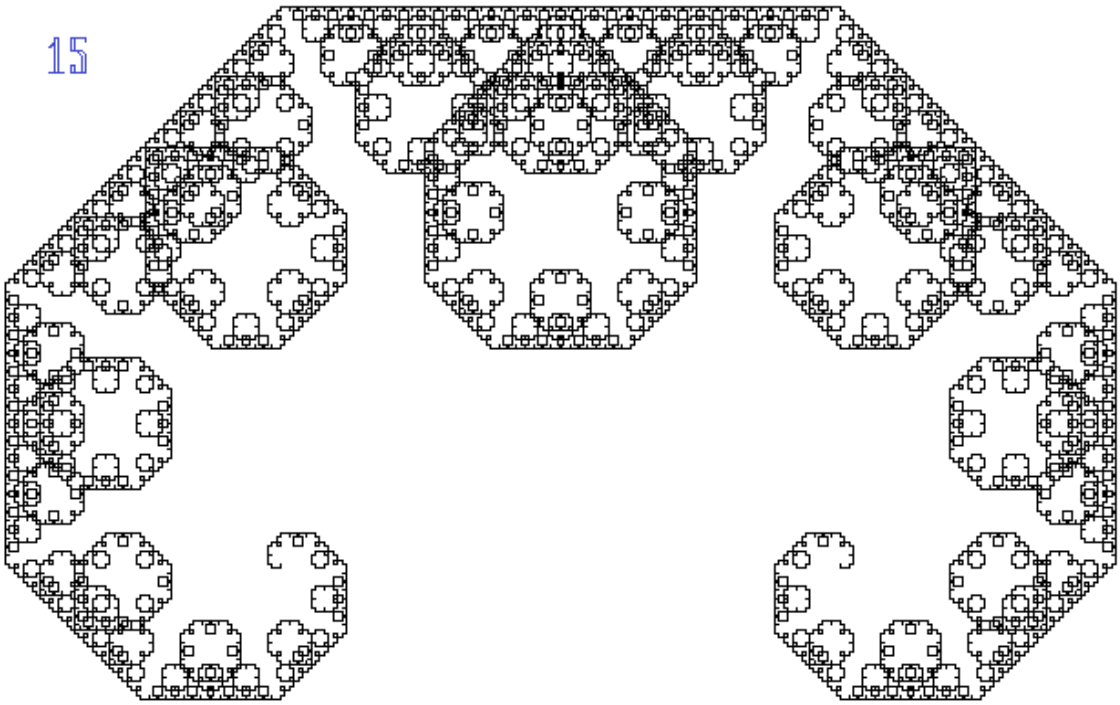
**Приложение 12. Фрактал *Дракон Хартера-Хайтвея,* созданный в графическом редакторе Paint:**



**Приложение 13. Фрактал *Кривая Леви,* созданный в графическом редакторе Paint:**



**Приложение 14. Этапы создания трехмерной пирамиды Серпинского:**