Фрактальная графика

Понятие фрактала

«***Фрактал***» от латинского *fractus*, что означает *разбитый*, *дробный* (*поделенный на части*).

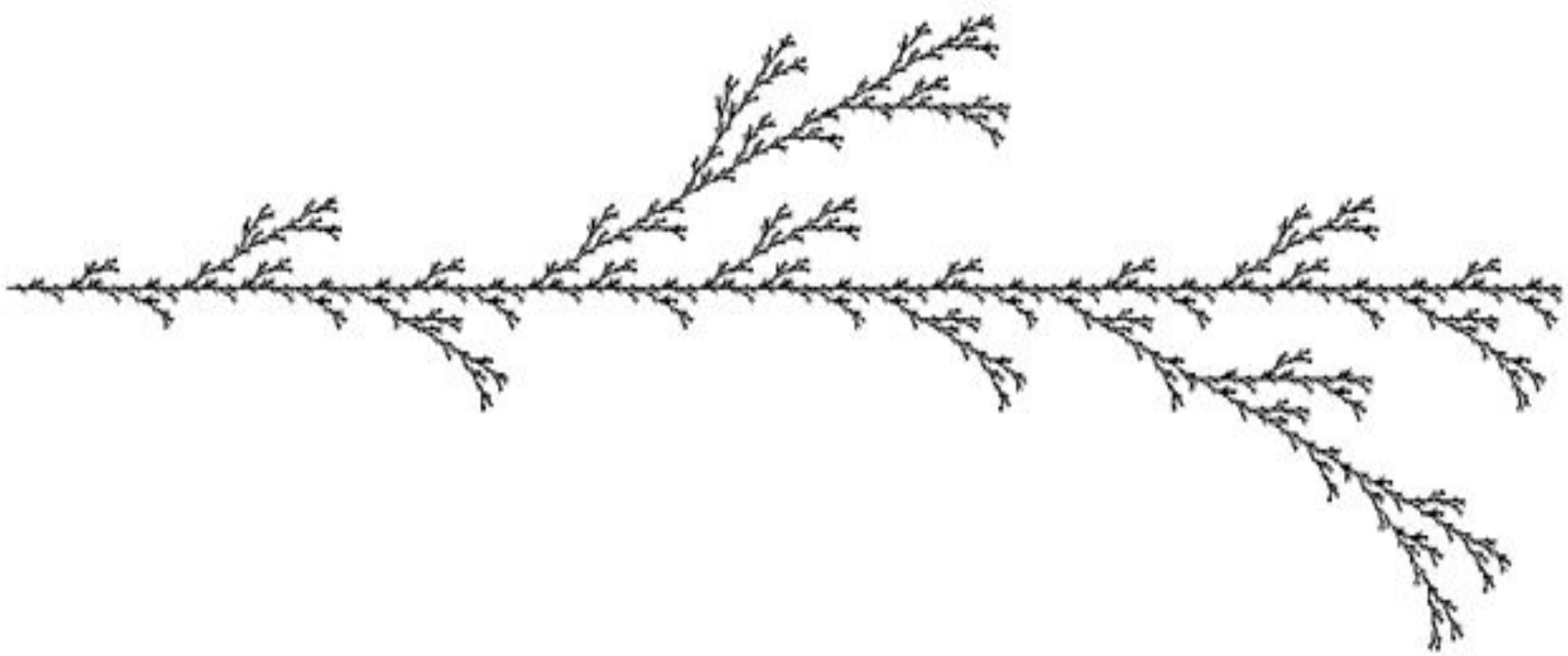
Термин был предложен американским математиком Бенуа МандельбротомТермин был предложен американским математиком Бенуа Мандельбротом в 1975 году.

Одно из определений фрактала:

Фрактал – это геометрическая фигура, состоящая из частей и которая может быть поделена на части, каждая из которых будет представлять уменьшенную копию целого (по крайней мере, приблизительно).

Это определение указывает характерное свойство всех фрактальных объектов – самоподобие.

Пример фрактала



Классификация фракталов

Применение фракталов

- В биологии - используются для моделирования популяций и для описания систем внутренних органов человека и животных (например, системы кровеносных сосудов).

- В физике - фракталы применяют при моделировании нелинейных процессов, таких, как турбулентное течение жидкости, сложные процессы диффузии-адсорбции, и т.п.

- В нефтехимии - используются при моделировании пористых материалов.

Применение фракталов

- В компьютерной графике - для построения изображений природных объектов, таких, как деревья, кусты, облака, горные ландшафты, поверхности морей и т.д.

- Для сжатия графической информации – фрактальное сжатие данных - в основе алгоритма лежит поиск больших фрагментов изображения подобных некоторым маленьким фрагментам. В выходной файл записывается только, какой фрагмент, какому подобен.

- Для анализа временных рядов, например курсов ценных бумаг, валют и цен на различные товары.

1. Геометрические фракталы

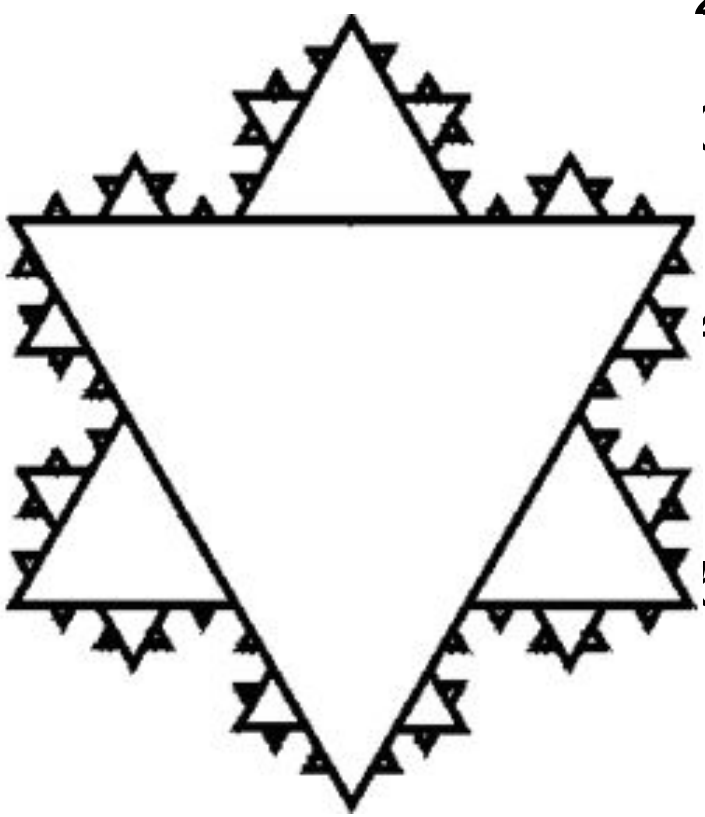
История фракталов началась именно с геометрических фракталов, которые исследовались математиками в XIX веке.

В этих фракталах наиболее очевидно наблюдается ***самоподобие***.

Алгоритмы для их получения наиболее простые и понятные.

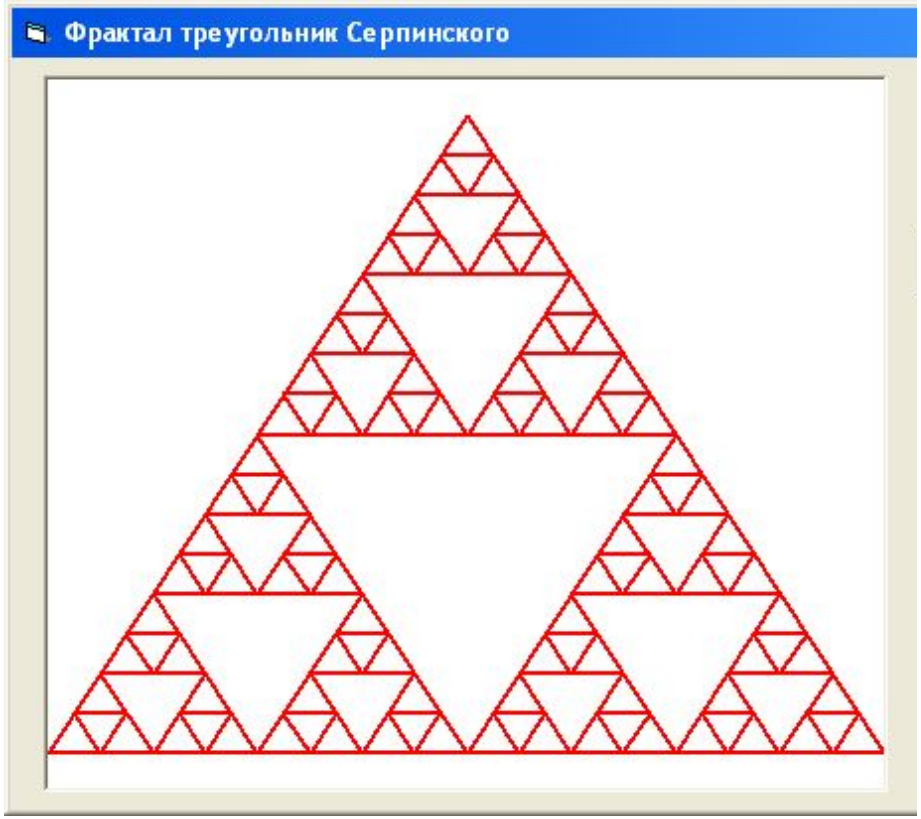
В двумерном случае такие фракталы можно получить, задав некоторую ломаную, называемой ***генератором***. За один шаг алгоритма каждый из отрезков, составляющих ломаную, заменяется на ломаную-генератор в соответствующем масштабе. В результате бесконечного повторения этой процедуры, получается геометрический фрактал.

Фрактальный треугольник



1) Строится равносторонний треугольник со стороной *a*.
2) Разделим каждую из его сторон на 3 отрезка равной длины.
3) На среднем отрезке каждой стороны построим равносторонний треугольник.
4) Со всеми полученными после этого треугольниками повторим те же действия. Процесс можно повторять сколько угодно долго.
5) Получается фрактальная структура, состоящая из треугольников, причем треугольники последующих «поколений» наследуют свойства родительских структур

Фрактал Треугольник Серпинского (салфетка или решето Серпинского)

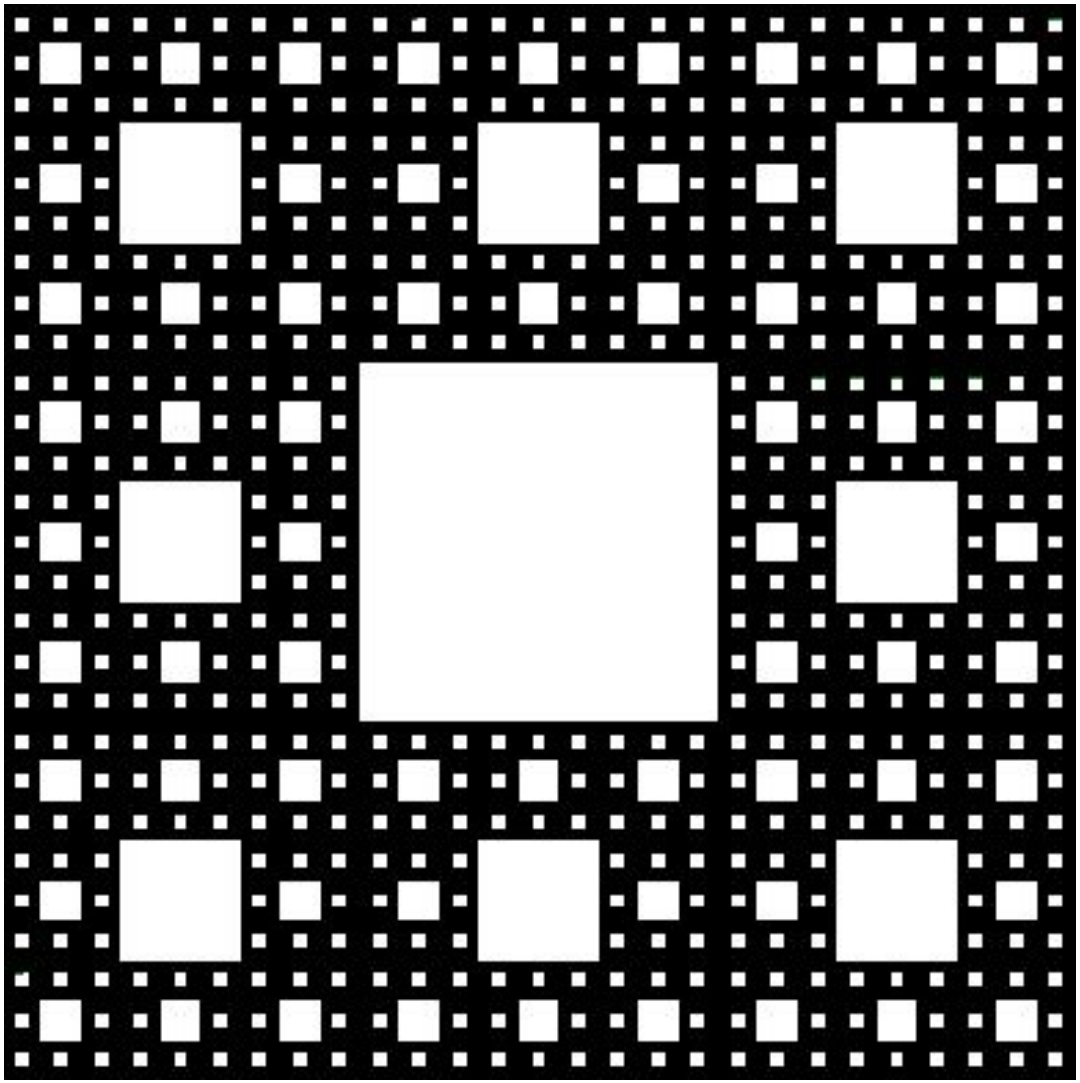


1) Строится большой внешний треугольник;

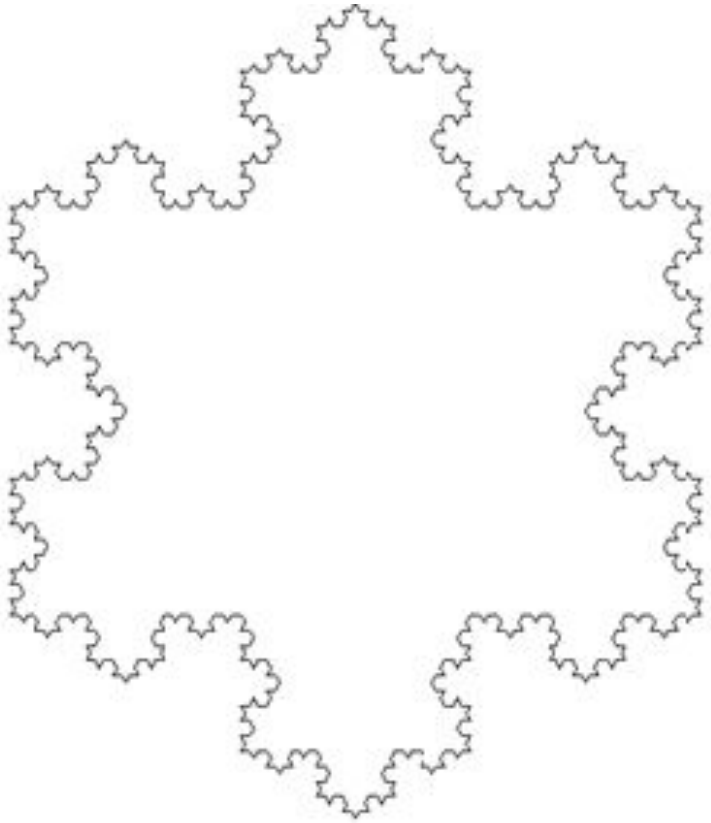
2) Строится треугольник, получающийся при соединении середин сторон большого треугольника;

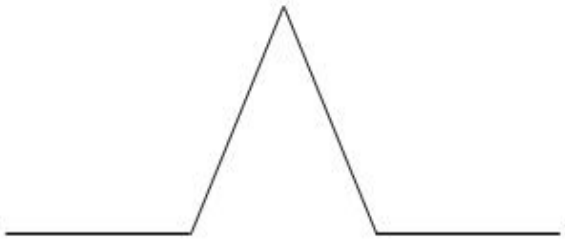
3) аналогично строятся остальные треугольники,.

Ковер Серпинского



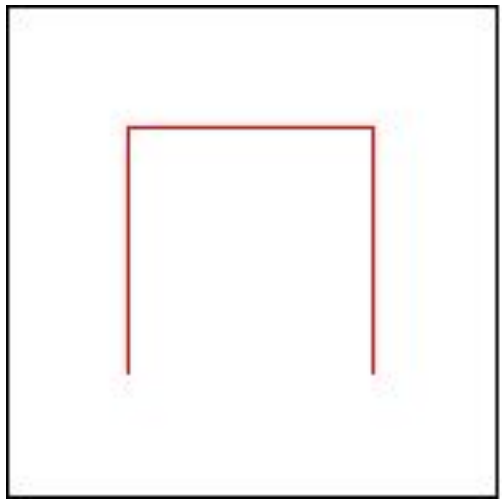
Звезда Коха



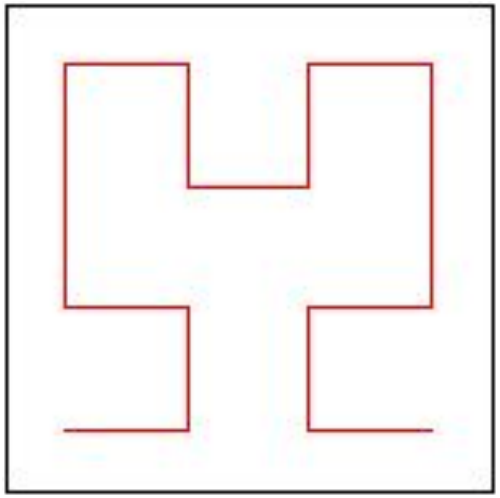


- Если каждый раз из фрактального треугольника удалять выделяемые средние отрезки, то получится многоугольник, называемый звездой (или снежинкой) Коха.
- При построении звезды Коха на каждом шаге один отрезок прямой заменяется четырьмя, длиной исходного отрезка, связанными между собой.
- Нижний рисунок демонстрирует результат одного шага алгоритма для построения звезды Коха

Фрактальная кривая Гильберта

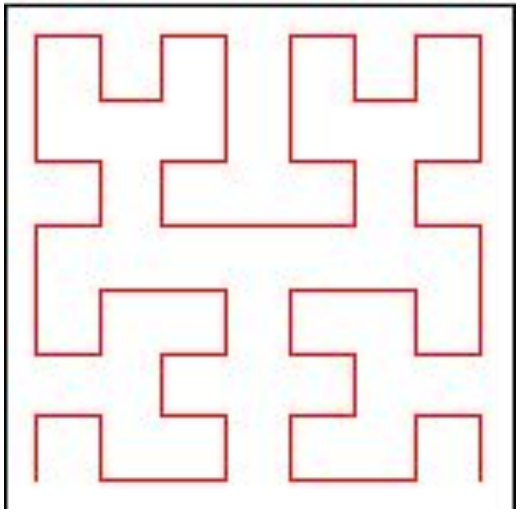


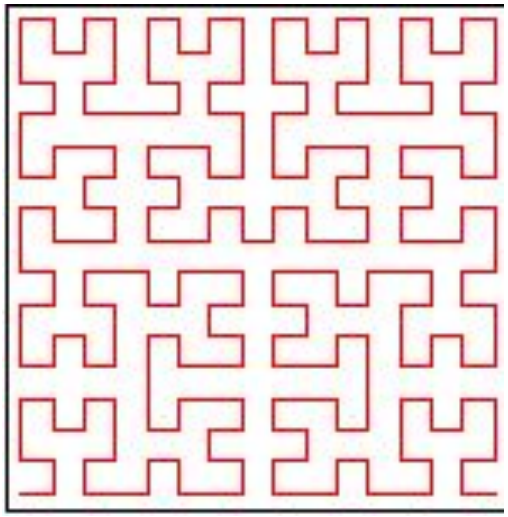
Кривая Гильберта первого порядка, обозначаемая H1, похожа на букву «П», в виде трех сторон квадрата.



Кривая Гильберта второго порядка H2, состоит из четырех кривых H1, ориентированных в разные стороны и соединенных тремя дополнительными отрезками прямых – связками

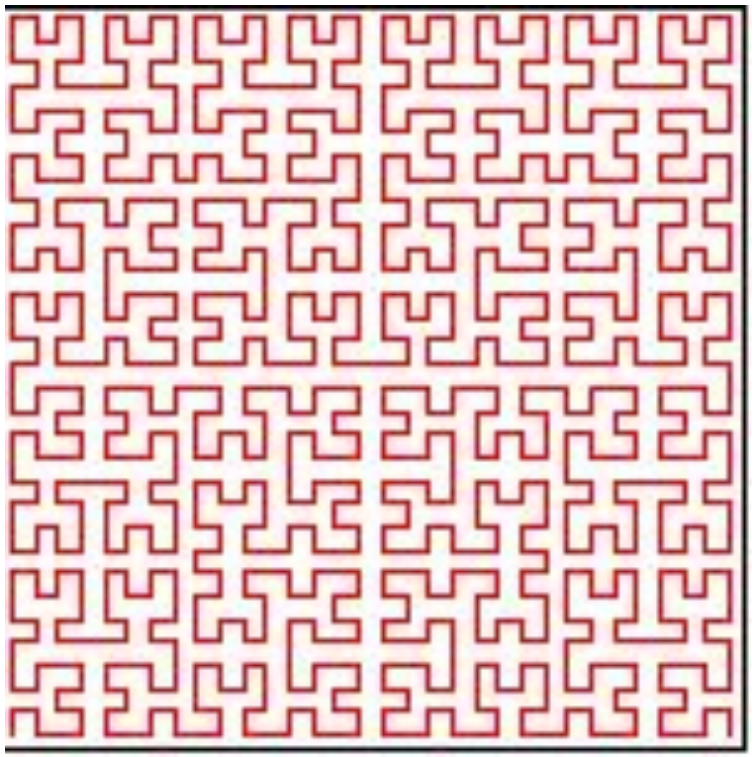
Фрактальная кривая Гильберта 3-го и 4-го порядка

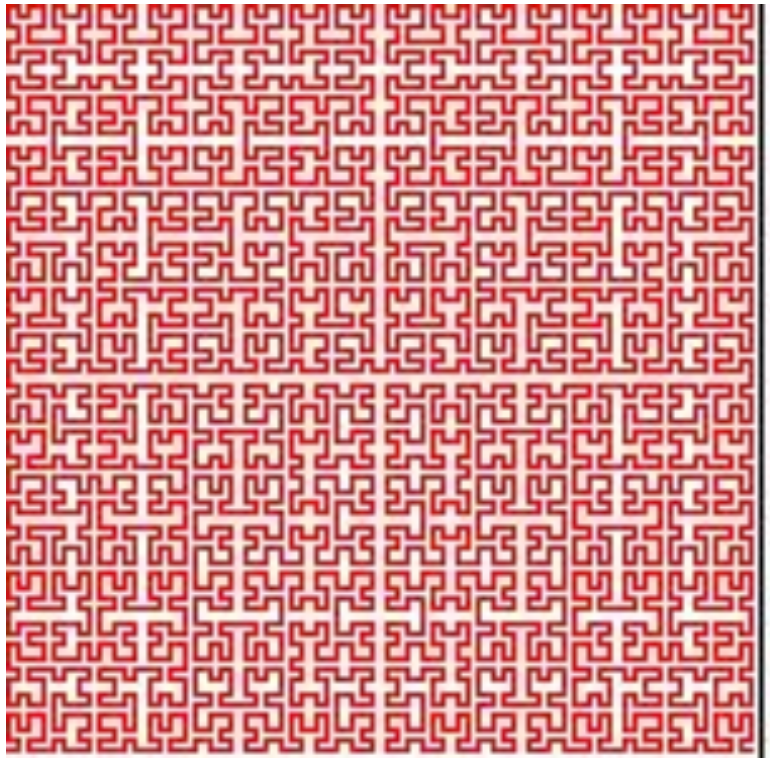




- Кривую H3 можно рассматривать как состоящую из четырех кривых H2, ориентированных в разные стороны и трех связок.
- Таким образом, кривую Гильберта *i*-го порядка можно получить из четырех кривых H *i* -1-го порядка, ориентированных в разные стороны и трех связок.
- Отрезки, образующие линию , можно рассматривать как связки, соединяющие четыре точки – кривые Гильберта нулевого порядка.

Фрактальная кривая Гильберта 5-го и 6-го порядка





Рекурсивный алгоритм для получения кривой Гильберта

Если процедуры рисования кривых , ориентированных вверх, вниз, влево и вправо, обозначить соответственно $GU(i)$, $GD(i)$, $GL(i)$ и $GR(i)$, то можно составить следующие рекурсивные схемы построения этих кривых:

$GU(i)$: $GR(i\text{-}1) \uparrow GU(i-1) \rightarrow GU(i-1) \downarrow GL(i-1)$
$GR(i)$: $GU(i\text{-}1) \rightarrow GR(i-1) \uparrow GR(i-1) \leftarrow GD(i-1)$
$GD(i)$: $GL(i\text{-}1) \downarrow GD(i-1) \leftarrow GD(i-1) \uparrow GR(i-1)$
$GL(i)$: $GD(i\text{-}1) \leftarrow GL(i-1) \downarrow GL(i-1) \rightarrow GU(i-1)$

Здесь символы «↑», «↓», «→», «←» обозначают связки, направленные вверх, вниз, вправо и влево, соответственно.

Процедура построения кривой Гильберта на псевдокоде

```
алг GU (цел i)
нач
если i > 0 то
    GR(i-1)
    LineUp
    GU(i-1)
    LineRight
    GU(i-1)
    LineDown
    GL(i-1)
все
кон
```

Аналогично можно оформить процедуры *GD*, *GR* и *GL* рисования кривых Гильберта, ориентированных вниз, вправо и влево.

В них *LineUp*, *LineDown*, *LineRight* и *LineLeft* – процедуры рисования связок, направленных соответственно вверх, вниз, право и влево.

Кривая Гильберта является частным случаем кривой Пеано. Всякая кривая Пеано вписана в квадрат. Чем выше порядок кривой, тем более плотно кривая заполняет квадрат.

L-системы

***L*-системы** названы в честь своего создателя биолога Аристида Линдермауера.

***L-системы* -** универсальные алгоритмы, которые в зависимости от заданных значений своих параметров получают различные геометрические фракталы.

С помощью *L*-систем можно создавать и «чисто математические» объекты, подобные кривой Гильберта, и изображения, очень похожие на природные: деревья, кусты, травинки. Причем имеется возможность получать несимметричные растения, например, как бы изогнувшиеся на ветру.

L-системы

- ***L-системой*** называют набор, состоящий из ***алфавита***, ***аксиомы***, и ***множества правил***.

- ***Алфавитом*** называется конечное множество, а его элементы — *символами*.

Природа символов не важна, их единственная функция — отличаться друг от друга.

- ***Строкой*** *над алфавитом* называют конечную последовательность символов алфавита..
- ***Аксиома*** — это некоторая строка над алфавитом.

L-системы

Алгоритмы *L*-систем для рисования фракталов основаны на «черепашьей» графике.

Представим себе исполнителя-черепашку, который умеет ползать по плоскости и выполнять всего несколько простых команд.

Черепашка имеет память, организованную в виде стека.

L-системы

Текущее состояние черепашки описывается тремя параметрами:

- **x,y** – текущие координаты черепашки;
- **α** – угол, определяющий направление, в котором черепашка ползет по команде «вперед».

Кроме того, на начальном шаге алгоритма задаются еще 2 параметра:

- **Δd** – величина шага, который делает черепашка по команде «вперед»;
- **Δα** – угол поворота, показывает насколько меняется угол при выполнении команд «налево» и «направо».

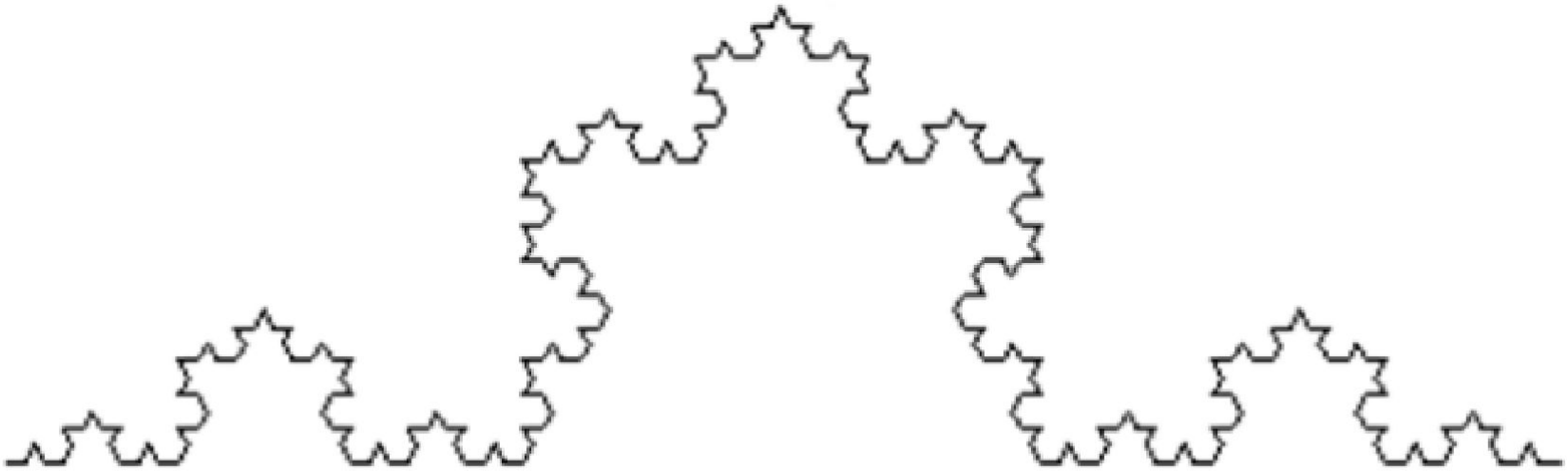
Команды, которые умеет выполнять черепашка, и их символьные обозначения приведены в таблице.

Символ, обозначающий команду	Описание команды
F	Переместиться вперед на Δd, в направлении α, оставив след.
+	Повернуться направо (по часовой стрелке) на угол $\Delta\alpha$ (при исполнении этой команды просто меняется угол α).
-	Повернуться налево (против часовой стрелки) на угол $\Delta\alpha$ (при исполнении этой команды просто меняется угол α).
[	Запомнить в стеке текущее состояние (x, y, α), т.е. текущие координаты и угол α.
]	Извлечь из стека последнее запомненное состояние.

Программой для черепашки является строка, т.е. последовательность символов, в которой кроме команд, приведенных в таблице, могут встречаться и любые другие символы.

Черепашка просматривает строку-программу слева направо символ за символом. Команды она выполняет, а все другие символы пропускает.

Пример выполнения программы черепашкой



Кривая Коха

Аксиома: F

Правило: $F \rightarrow F\text{-}F{+}{+}F\text{-}F$

Угол: $\frac{\pi}{3}$

Строки-программы для черепашки получаются не вручную. Для этого в *L*-системах предусмотрен специальный алгоритм.

Пусть имеется начальная строка, называемая ***аксиомой***, и набор строк, называемых ***правилами***. Аксиома может быть любой, а правила должные иметь вид «символ → строка».

Например:

Аксиома: F++F++F

Правило: F → F-F++F-F.

Сначала создаваемая строка-программа принимается равной аксиоме.

Она просматривается символ за символом слева направо. Если встречается символ, стоящий в левой части одного из правил (правил может быть несколько), то он заменяется правой частью этого правила.

В нашем примере после первого шага получается следующая строка-программа:

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

Далее строка-программа опять просматривается слева направо и опять каждый символ сравнивается с левыми частями правил.

В случае обнаружения совпадения, символ заменяется правой частью соответствующего правила.

Теоретически этот процесс можно продолжать до бесконечности.

На практике обычно задают **глубину** – количество шагов обработки строки-программы.

Системы итерируемых функций (IFS)

Идея метода заключается в представлении изображения несколькими простыми преобразованиями точек.

Обычно используются простые аффинные преобразования на плоскости.

Аффинные преобразования включают в себя масштабирование, поворот и параллельный перенос.

Метод систем итерируемых функций можно описать следующим образом:

Изображение кодируется несколькими простыми преобразованиями.

Таким образом, достаточно хранить только коэффициенты этих преобразований (в случае аффинных преобразований A, B, C, D, E, F).

Например, закодировав некоторое изображение двумя аффинными преобразованиями, мы определяем его с помощью двенадцати коэффициентов.

Если теперь задаться какой-либо начальной точкой, и запустить итерационный процесс, то после первой итерации получим две точки, после второй – четыре, после третьей – восемь и т.д.

Через несколько десятков итераций совокупность полученных точек будет описывать закодированное изображение. Но проблема в том, что трудно найти коэффициенты преобразований, которые кодировали бы произвольное изображение.

Для построения IFS применяют кроме аффинных и другие классы простых геометрических преобразований, которые задаются небольшим числом параметров. Например, проективные:

$$X' = (A_1 \cdot X + B_1 \cdot Y + C_1)/(D_1 \cdot X + E_1 \cdot Y + F_1),$$
$$Y' = (A_2 \cdot X + B_2 \cdot Y + C_2)/(D_2 \cdot X + E_2 \cdot Y + F_2)$$

или квадратичные:

$$X' = A_1 \cdot X^2 + B_1 \cdot X \cdot Y + C_1 \cdot Y^2 + D_1 \cdot X + E_1 \cdot Y + F_1,$$
$$Y' = A_2 \cdot X^2 + B_2 \cdot X \cdot Y + C_2 \cdot Y^2 + D_2 \cdot X + E_2 \cdot Y + F_2$$

преобразования на плоскости.

Детерминированный метод построения фракталов на основе IFS

Существуют два подхода к реализации IFS:

- детерминированный
- рандомизированный.

В общем виде **детерминированный алгоритм построения фрактала с помощью IFS** можно описать следующим образом:

Пусть для некоторого фрактала требуются *N* аффинных преобразований.

На начальном шаге детерминированного алгоритма создается исходное множество точек .

Переменной *j* присваивается значение 0.

Общий шаг алгоритма:

- $j = j + 1$
- К каждой точке множества применить каждое из N аффинных преобразований. В результате получим новое множество точек .

Общий шаг можно повторять заданное число раз либо до тех пор, пока детали изображения не станут мельче заданной величины.

При реализации следует учесть, что в результате преобразования некоторые точки могут выйти за границы области, в которую осуществляется вывод фрактала. Такие точки следует исключить из дальнейшей работы.

Рандомизированный метод построения фракталов на основе IFS

Задается вероятность применения того или иного аффинного преобразования.

Пример. **Фрактальная решетка**

В общей виде аффинное преобразование на плоскости задается следующей системой уравнений:

$X' = A * X + B * Y + E$

$Y' = C * X + D * Y + F.$

Рандомизированный метод IFS построения фрактальной решетки

Для получения фрактальной решетки берутся следующие четыре системы:

(1, вероятность выбора 0.25)
X' = 0.3 * X - 0.3 * Y + 1
Y' = 0.3 * X + 0.3 * Y + 1

(2, вероятность выбора 0.25)
X' = 0.3 * X - 0.3 * Y + 1
Y' = 0.3 * X + 0.3 * Y - 1

(3, вероятность выбора 0.25)
X' = 0.3 * X - 0.3 * Y - 1
Y' = 0.3 * X + 0.3 * Y + 1

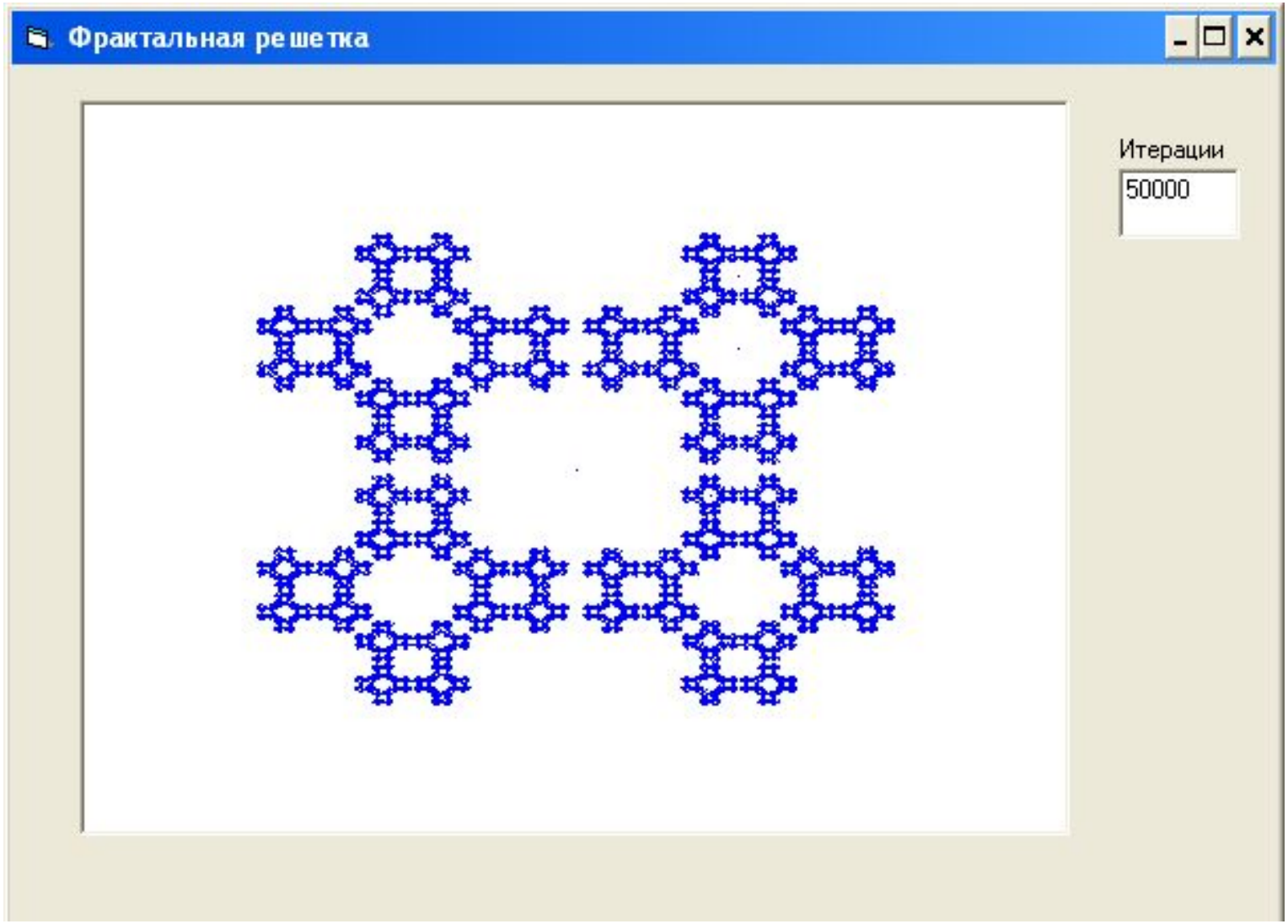
(4, вероятность выбора 0.25)
X' = 0.3 * X - 0.3 * Y - 1
Y' = 0.3 * X + 0.3 * Y - 1

Рандомизированный метод IFS построения фрактальной решетки

Алгоритм построения фрактала:

- Взять произвольную точку (X, Y) на плоскости.
- Выбрать, используя известные вероятности выбора, одно из аффинных преобразований,
- Вычислить координаты (X', Y') следующей точки.
- Принять X = X' и Y = Y'.
- Повторить п.п. 2 и 3 алгоритма заданное число раз.

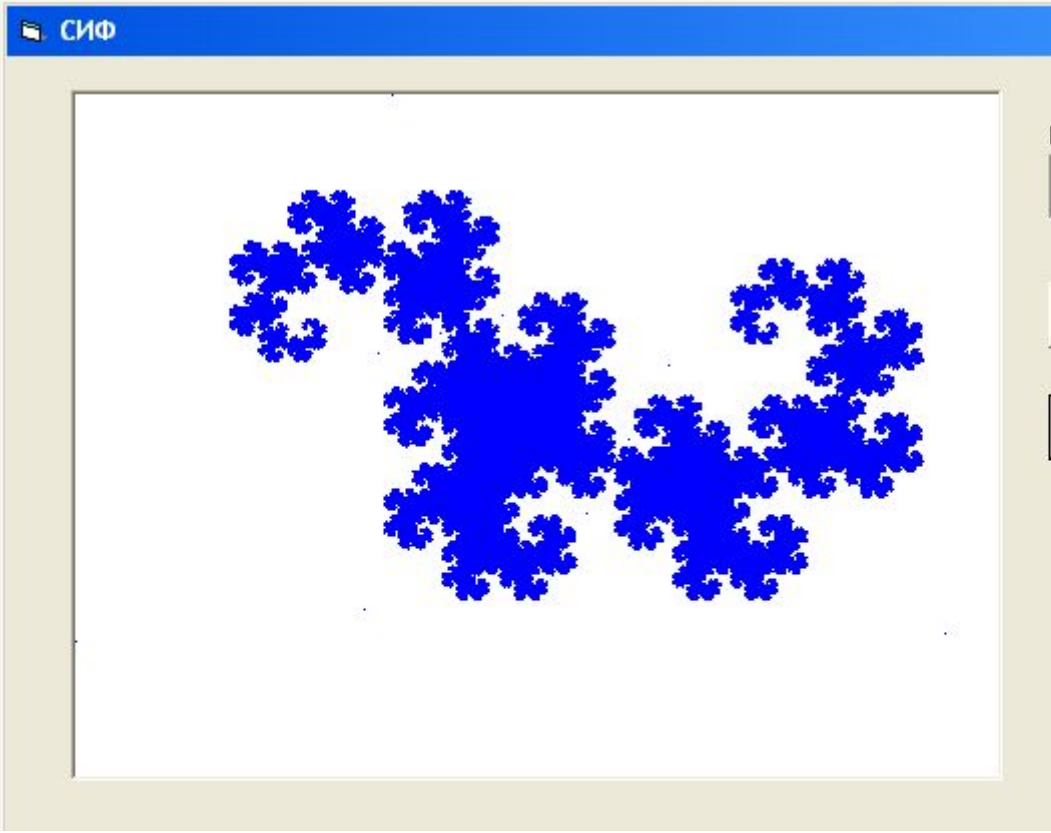
Рандомизированный метод IFS построения фрактальной решетки



Фрактал дракона Хартера-Хейтуэя на основе IFS

На основе двух аффинных преобразований:

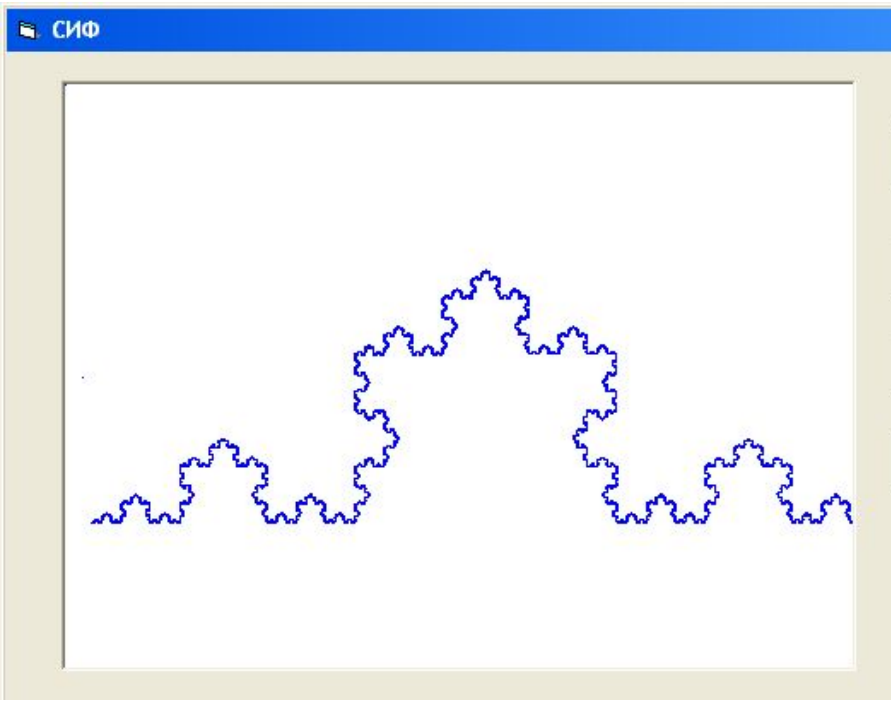
1) X' = -0.5*X -0.5*Y + 490
   Y' = 0.5*X -0.5*Y + 120
2) X' = 0.5*X -0.5*Y + 340
   Y' = 0.5*X +0.5*Y - 110



Фрактал кривая Коха на основе IFS

Требуется набор аффинных преобразований, состоящий из четырех преобразований:

1) X' = 0.333*X + 13.333
   Y' = 0.333*Y + 200
2) X' = 0.333*X + 413.333
   Y' = 0.333*Y + 200
3) X' = 0.167*X + 0.289*Y + 130
   Y' = -0.289*X + 0.167*Y + 256
4) X' = 0.167*X - 0.289*Y + 403
   Y' = 0.289*X + 0.167*Y + 71



2. АЛГЕБРАИЧЕСКИЕ ФРАКТАЛЫ

- Алгебраические фракталы строят, используя простые алгебраические формулы.
- Пример - множество Мандельброта – один из самых известных фрактальных объектов. Это множество было построено Бенуа Мандельбротом в 1980 году.

Алгоритм его построения достаточно прост и основан на простом итеративном выражении:

$$Z[k+1] = Z[k] * Z[k] + C, \qquad (1)$$

где $Z[k]$ и C – комплексные переменные, k – номер итерации, $k = 1, 2, \ldots.;\ Z[0] = 0.$

Любое комплексное число Z можно представить в виде:

$$Z = x + y * i,$$

где x и y это действительные числа,

i – мнимая единица.

Мнимая единица – такой математический объект, который имеет следующее свойство: $i^2 = -1.$

Если интерпретировать числа x и y как декартовы координаты, то получим соответствие комплексных чисел и точек на плоскости (рис. 1).

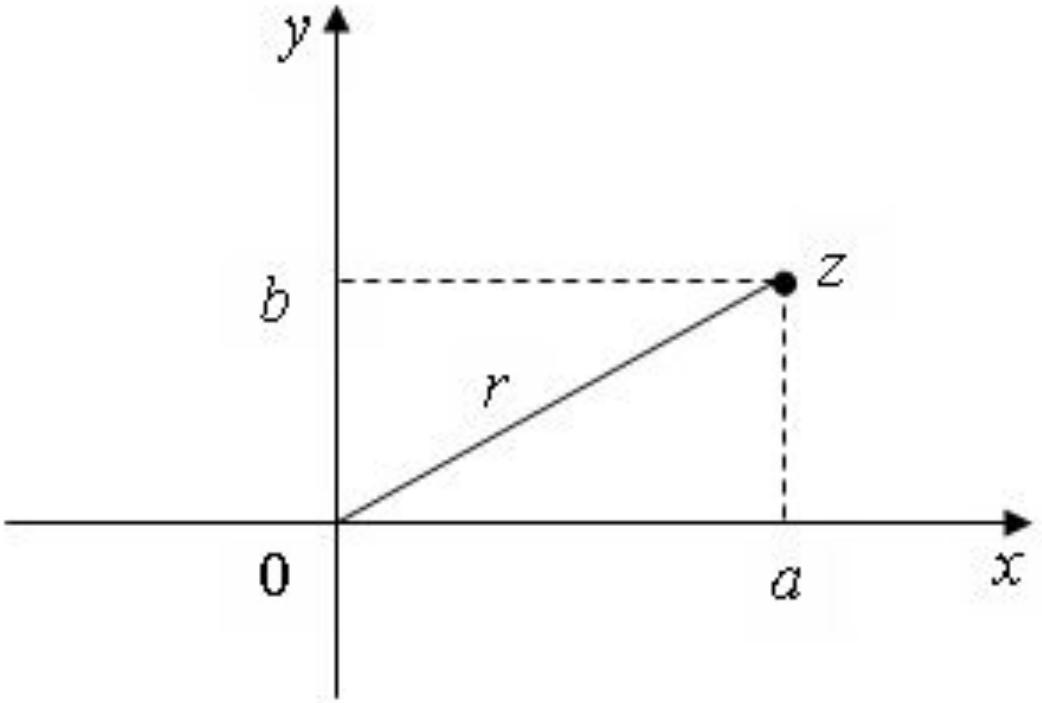


Рис. 1. Соответствие комплексных чисел на плоскости

Число x называется абсциссой, а y – ординатой комплексного числа $x+y^*i$. Эта система координат называется *комплексной плоскостью*. Расстояние r от начала координат до точки z комплексной плоскости с координатами (x, y) определяется выражением:

$$r = |x + y^* i| = \sqrt{x^2 + y^2}, \qquad (2).$$

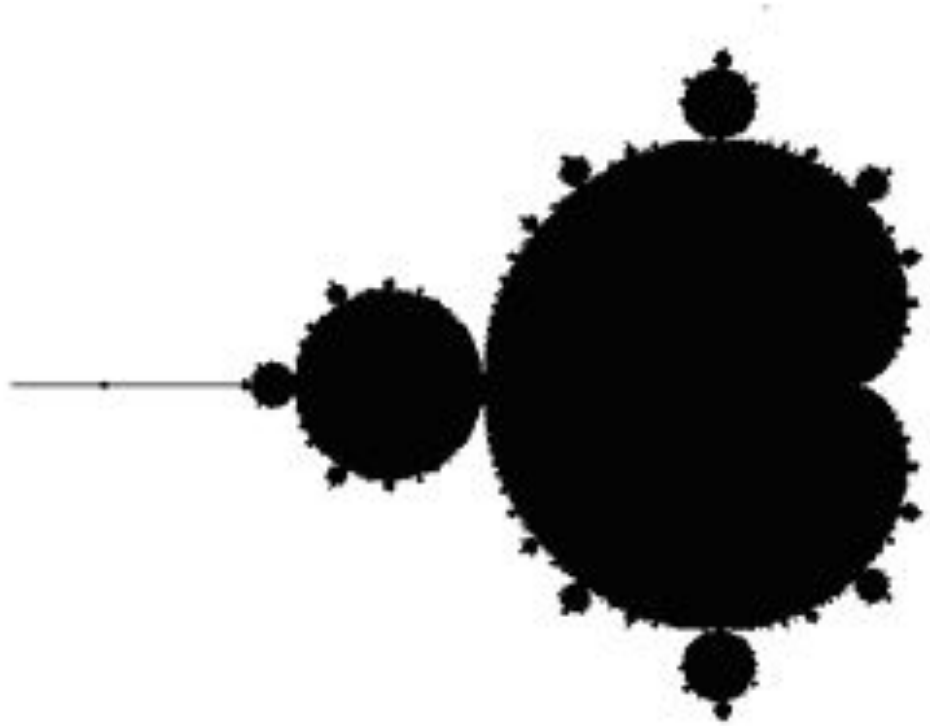
Величина r называется *модулем комплексного числа*.

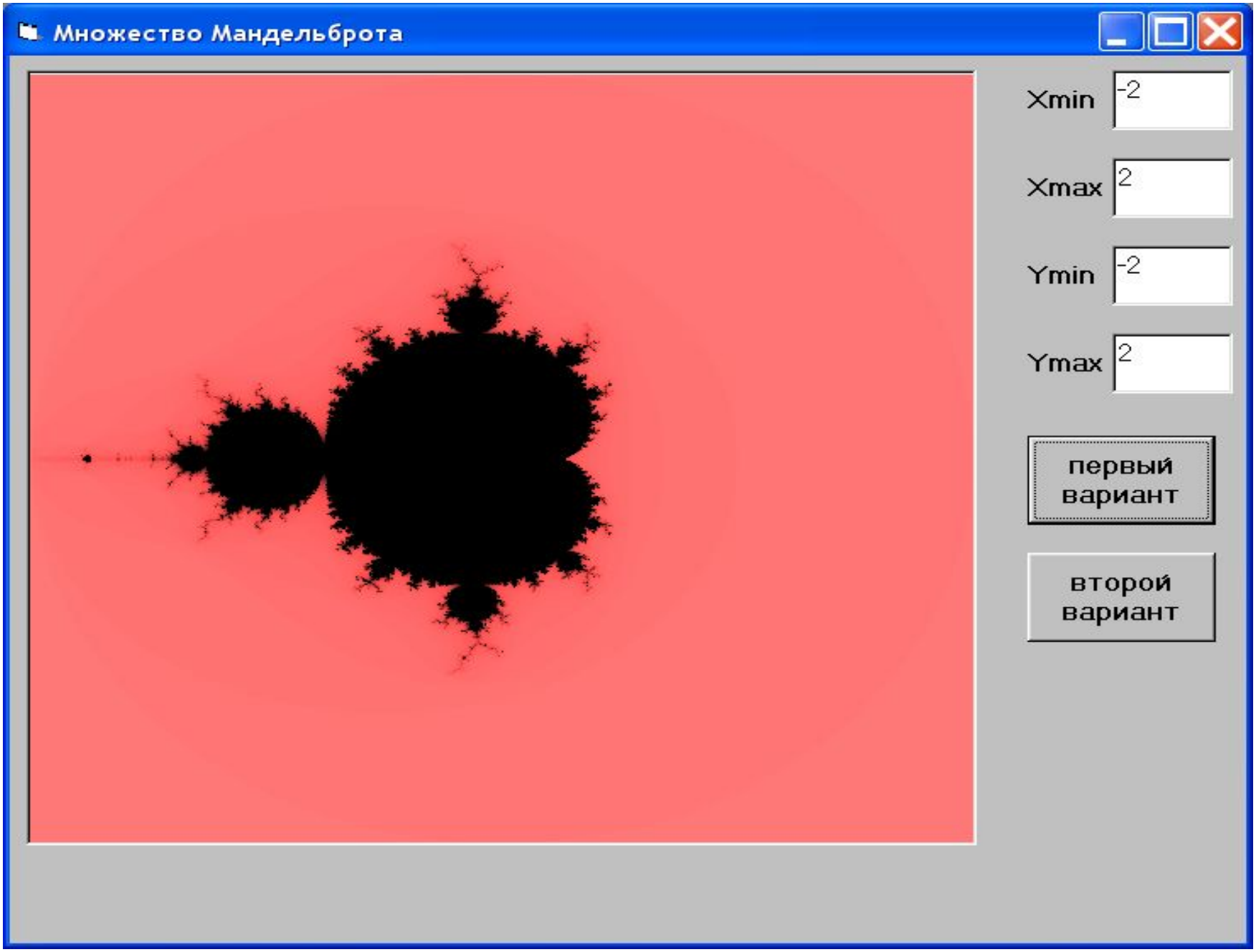
Множество Мандельброта определяется как множество комплексных чисел C, для которых последовательность чисел $\{z_k\}$, построенная по формуле (1), является ограниченной, т.е. существует такое число R, что для любого k выполняется условие $|z_k| < R$, где $|z_k|$ – модуль комплексного числа, вычисляемый по формуле (2).

Для построения фрактала рассматривается некоторая прямоугольная область комплексной плоскости. Для каждой точки C этой области выполняется N итераций, т.е. находят N членов последовательности $\{z_k\}$ по формуле (1), при этом $k = 1, 2, \ldots, N$; $z_0 = 0$; число N – задается пользователем. Доказано, что множество Мандельброта размещается в круге радиуса 2 с центром в начале координат. Поэтому если для какой-либо точки C на некоторой итерации k получается комплексное число z_k, модуль которого r превышает число 2, то можно сделать вывод, что точка C не принадлежит множеству Мандельброта. В этом случае итерационный процесс можно завершить досрочно, не достигая заданного числа N. Если будут проделаны все N итераций, и при этом окажется, что все получаемые числа z_k принадлежат кругу радиуса 2, то это не гарантирует, что точка C принадлежит множеству Мандельброта.

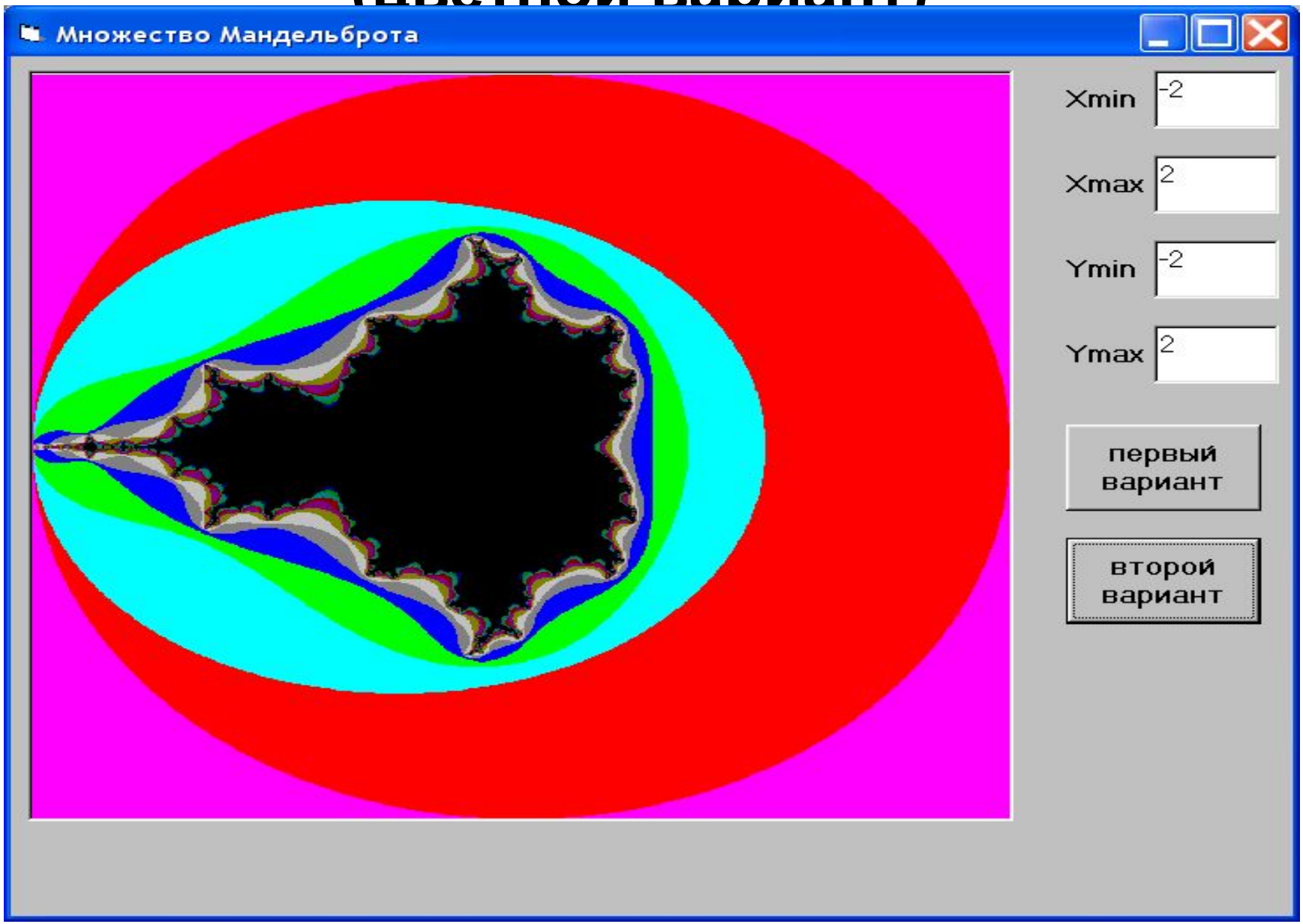
Пусть точки *C* комплексной плоскости, не принадлежащие множеству Мандельброта, окрашиваются в белый цвет, а точки C, которые в данном приближении считаются принадлежащими множеству – в черный.

Тогда будет получено черно-белое изображение фрактала

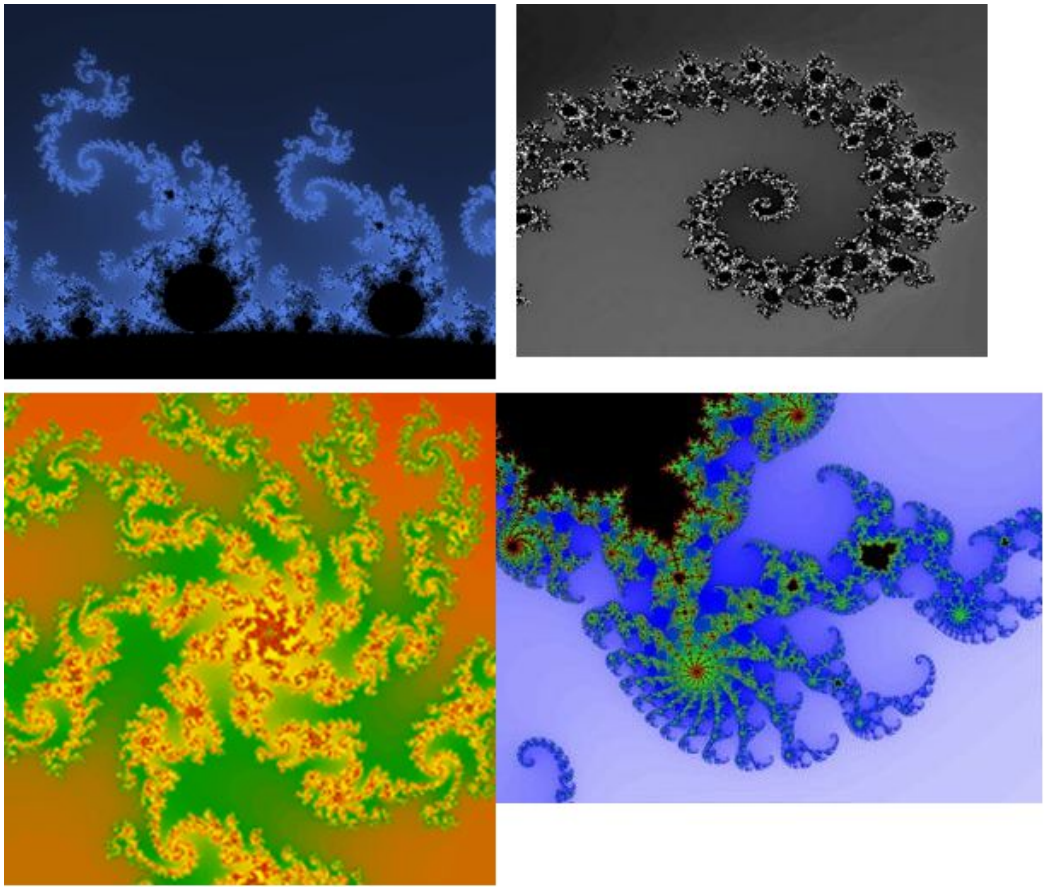



Множество Мандельброта
Xmin
-2
Xmax
2
Ymin
-2
Ymax
2
первый вариант
второй вариант

Множество Мандельброта (цветной вариант)



- Наиболее интересные сложные структуры возникают на границах множества.
- Можно получать и многоцветные изображения множества Мандельброта или изображения, содержащие различные оттенки серого.



Для этого точки плоскости, не принадлежащие множеству, окрашиваются не всегда в белый цвет, как говорилось выше. Таким точкам присваивается цвет в зависимости от номера итерации, на которой произошел выход за пределы круга радиуса 2. Например, можно принять правило, что чем больше итераций последовательность $\{z_k\}$ остается в круге, тем цвет точки должен быть ближе к цвету, выбранному для множества Мандельброта. В случае, когда для изображения фрактала используются только оттенки серого, чем больше итераций последовательность $\{z_k\}$ остается в круге, тем более темный оттенок серого присваивается точке.

Первый вариант алгоритм получения изображения множества Мандельброта

алг Мандельброт1 (**цел** N)
нач
вещ r
комплекс z, C
цел k
нц для всех точек C выделенной области комплексной плоскости
$z = 0$
$k = N$
нц пока $r <= 2$ **и** $k > 0$
$z = z^2 + C$
$r = \sqrt{(z.x)^2 + (z.y)^2}$
$k = k - 1$
кц
окрасить точку C в цвет k
кц
кон

Работа с цветом может быть реализована по-разному. В данном случае считаем, что в нашем распоряжении имеется N цветов, т.е. число цветов равно числу итераций (если нужно использовать меньшее количество цветов, то нетрудно выполнить масштабирование палитры, т.е. присвоить каждый цвет некоторому интервалу номеров итераций). Цвета занумерованы от 0 до $N-1$. Причем черный цвет имеет номер 0.

Полученный алгоритм следует доработать с целью повышения скорости построения изображения. Для ускорения вычислений заменим операции возведения в квадрат умножением. Кроме того, избавимся от извлечения квадратного корня, а величину $(z.x)^2+(z.y)^2$ будем сравнивать с числом 4.

Для того чтобы сделать алгоритм более удобным для компьютерной реализации избавимся от операций с мнимой единицей. Для этого будем отдельно вычислять абсциссу и ординату точки комплексной плоскости. Вспомним правила выполнения умножения и сложения комплексных чисел:

Произведением двух комплексных чисел $z_1 = x_1 + y_1 \cdot i$ и $z_2 = x_2 + y_2 \cdot i$ называется комплексное число

$$(x_1 \cdot x_2 - y_1 \cdot y_2) + (x_1 \cdot y_2 + y_1 \cdot x_2) \cdot i.$$

Суммой двух комплексных чисел $z_1 = x_1 + y_1 \cdot i$ и $z_2 = x_2 + y_2 \cdot i$ называется комплексное число

$$(x_1 + x_2) + (y_1 + y_2) \cdot i, \qquad (3)$$

Пользуясь этими правилами, выполним преобразование формулы (1). Пусть $Z[k] = x_k + y_k \cdot i$. Тогда

$$Z[k+1] = Z[k] * Z[k] + C = (x_k + y_k \cdot i) \cdot (x_k + y_k \cdot i) + C =$$

$$= x_k \cdot x_k - y_k \cdot y_k + (x_k \cdot y_k + y_k \cdot x_k) \cdot i + C =$$

$$= \underbrace{\underbrace{(x_k \cdot x_k - y_k \cdot y_k)}_{\text{абсцисса}} + \underbrace{2 \cdot x_k \cdot y_k}_{\text{ордината}} \cdot i + C}_{\text{комплексное число}}.$$

Пусть $C = x_C + y_C \cdot i$. Тогда требуется выполнить сложение двух комплексных чисел $(x_k \cdot x_k - y_k \cdot y_k) + 2 \cdot x_k \cdot y_k \cdot i$ и $x_C + y_C \cdot i$. Как видно из формулы (3) при сложении комплексных чисел отдельно складываются их абсциссы и ординаты. Поэтому формулу (1) можно записать в координатной форме:

$$x_{k+1} = x_k \cdot x_k - y_k \cdot y_k + x_C;$$

$$y_{k+1} = 2 \cdot x_k \cdot y_k + y_C.$$

Для окончательной формулировки алгоритма осталось обсудить вопрос, как организовать перебор точек выделенной области комплексной плоскости. Пусть для построения фрактала выделена прямоугольная область, координаты верхнего левого угла которой $(x_{\min}, y_{\min})$, а нижнего правого – $(x_{\max}, y_{\max})$. Можно задать количество интервалов разбиения d, и, используя ее, рассчитать шаг по каждой из координатных осей: $hx = \frac{x_{\max} - x_{\min}}{d}$;

$hy = \frac{y_{\max} - y_{\min}}{d}$.

В алгоритме организовываем перебор абсцисс точек от $x_{\min}$ до $x_{\max}$ с шагом hx и ординат от $y_{\min}$ до $y_{\max}$ с шагом hy.

алг Мандельброт2 (**цел** $N, d, x_{\min}, y_{\min}, x_{\max}, y_{\max}$)
нач
вещ $r, x, y, x2, y2, Cx, Cy$
цел k
$hx = (x_{\max} - x_{\min})/d$
$hx = (y_{\max} - y_{\min})/d$
нц для Cx **от** $x_{\min}$ **до** $x_{\max}$ **шаг** hx
нц для Cy **от** $y_{\min}$ **до** $y_{\max}$ **шаг** hy
$x = 0$
$y = 0$
$k = N$

нц пока $r <= 4$ **и** $k > 0$

$x2 = x * x$

$y2 = y * y$

$x = x2 + y2 + Cx$

$y = 2 * x * y + Cy$

$r = x2 + y2$

$k = k - 1$

кц

окрасить точку C в цвет k

кц

кц

кон

3. СТОХАСТИЧЕСКИЕ ФРАКТАЛЫ

Фракталы, при построении которых случайным образом изменяются какие-либо параметры, называются **стохастическими**.

Такие фракталы лучше подходят для описания природных объектов и процессов, поскольку в природе нет абсолютно правильных линий и строгой симметрии.

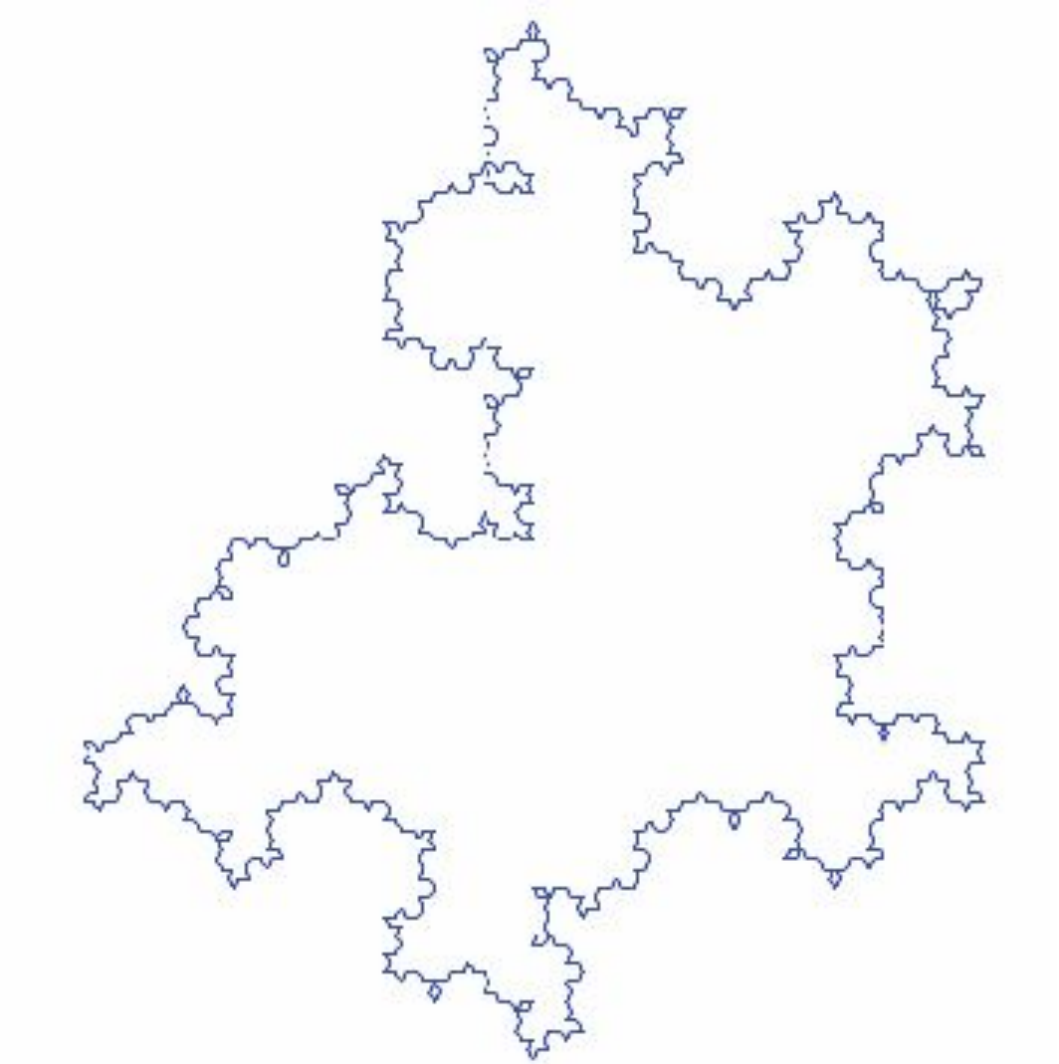
На формирование природных объектов оказывают влияние множество случайных факторов.

Нельзя найти, например, две совершенно одинаковых снежинки или два одинаковых листочка на одном и том же дереве. Внесение случайности в построение фракталов позволяет получать искусственные объекты, удивительно схожие с природными.

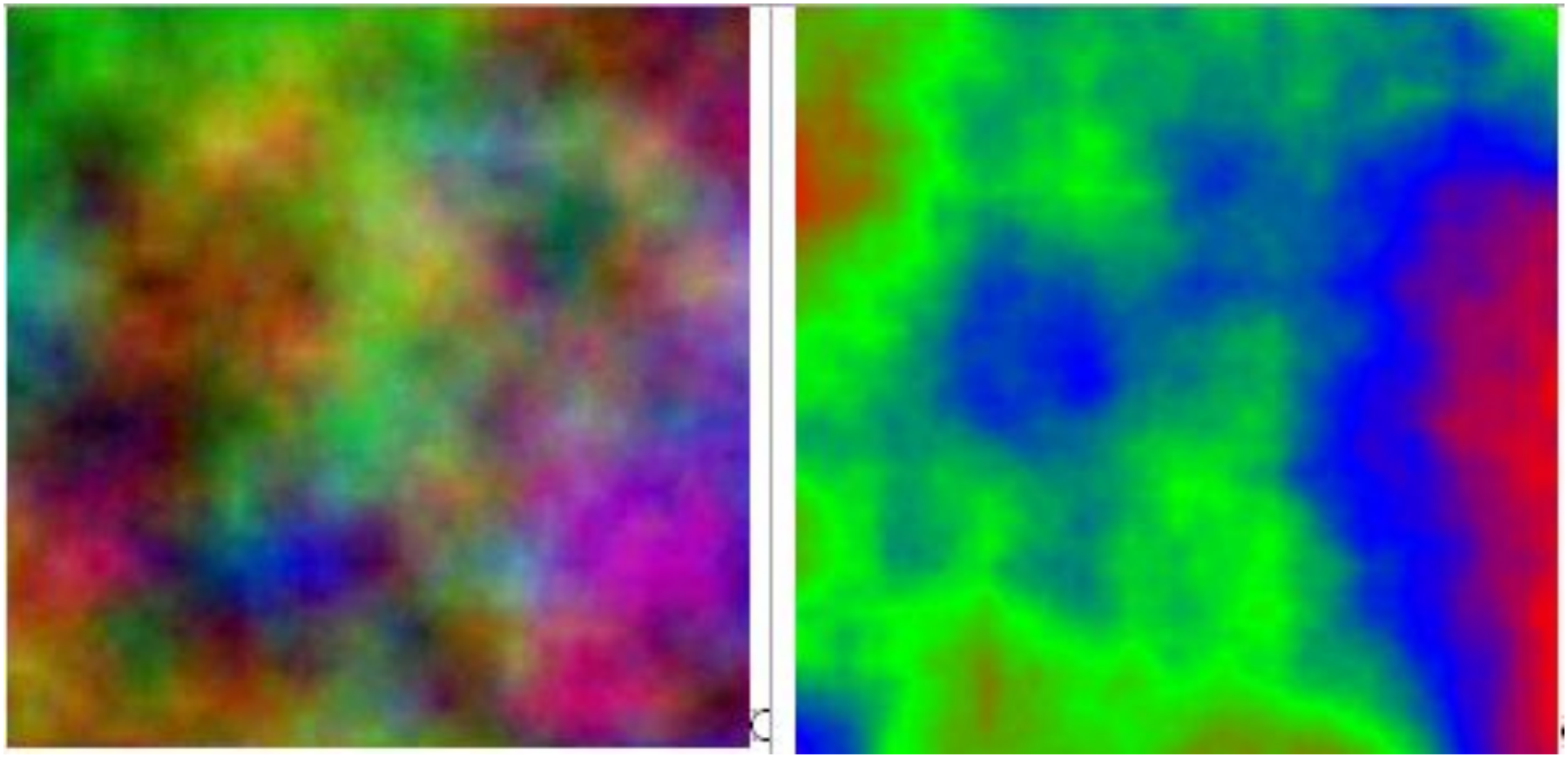
Рандомизированная звезда Коха

- Для превращения этого фрактала в стохастический немного изменим алгоритм его построения.
- Вообще, при построении звезды Коха, на каждом шаге алгоритма один отрезок прямой заменяется четырьмя.
- Теперь же с помощью датчика случайных чисел будем каждый раз выбирать направление нового участка внутрь или наружу, т.е. будет ли получаемая ломаная выпуклой или вогнутой. Построенный в результате стохастический фрактал уже не будет симметричным как обычная звезда Коха.

Рандомизированная звезда Коха 5-го порядка



Фрактал «плазма»



Для построения фрактала задается прямоугольная область. Пусть (x_1, y_1) – координаты верхнего левого угла, а (x_2, y_2) – координаты нижнего правого угла прямоугольника.

Для четырех точек растра, соответствующих четырем углам прямоугольника выбираются случайным образом цвета (при компьютерной реализации используется датчик случайных чисел).

Затем вычисляются координаты середины каждой из четырех сторон, и каждой из этих новых точек присваивается цвет, получаемый как среднее арифметическое цветов вершин соответствующей стороны плюс некоторое случайное число.

Чем больше случайное число, тем более «рваным» будет рисунок. Далее определяются координаты центра прямоугольника, и этой точке присваивается цвет, равный среднему арифметическому цветов по углам прямоугольника.

После этого исходный прямоугольник делится на четыре части линиями, параллельными сторонам прямоугольника и проходящими через его центр (рис. 6), и процедура назначения цветов серединам сторон и центру прямоугольника повторяется для каждого нового прямоугольника.

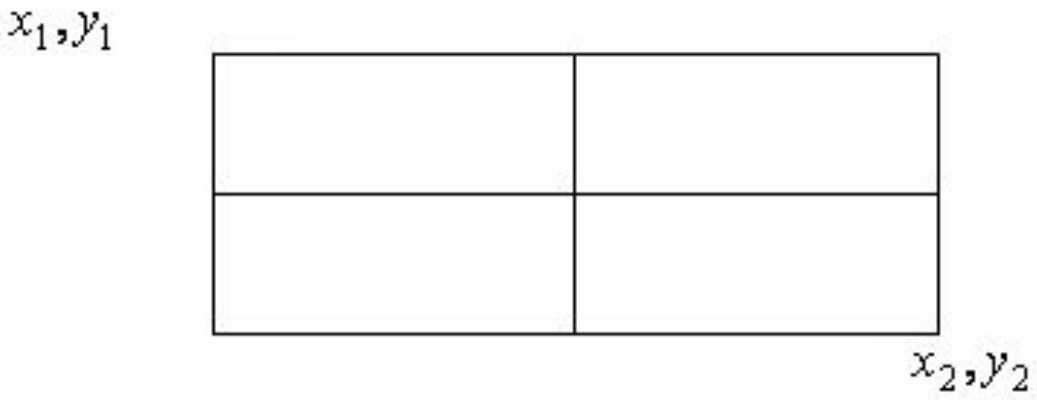


Рис. 6. Область построения фрактала

При этом получается, что длина и ширина каждого нового прямоугольника в 2 раза меньше по сравнению с длиной и шириной исходного прямоугольника. Процесс останавливается, когда размеры очередных четырех выделяемых прямоугольников становятся меньше двух.

Главный алгоритм Создать_плазму(**цел** $x1$, $y1$, $x2$, $y2$, ЧислоЦветов) принимает в качестве аргументов координаты левого верхнего ($x1$, $y1$) и нижнего правого углов ($x2$, $y2$) прямоугольной области, а также количество цветов в используемой палитре (ЧислоЦветов). Этот алгоритм назначает случайно выбранные цвета вершинам области и вызывает рекурсивный алгоритм построения «плазмы» (Рек_Плазма):

алг Создать_плазму(**цел** $x1, y1, x2, y2$, ЧислоЦветов)
нач
цел таб $P[1 : x2\text{-}x1+1, 1 : y2\text{-}y1+1]$ | массив цветов точек растра
$P[x1, y1)]$:= СлучЧисло(ЧислоЦветов)
$P[x2, y1)]$:= СлучЧисло(ЧислоЦветов)
$P[x1, y2)]$:= СлучЧисло(ЧислоЦветов)
$P[x2, y2)]$:= СлучЧисло(ЧислоЦветов)
Рек_Плазма ($x1, y1, x2, y2$, ЧислоЦветов, P)
кон

Пояснение: СлучЧисло(x) – функция, которая возвращает случайное целое число из интервала [0, x-1].

Следующий алгоритм Рек_Плазма(**цел** $x1$, $y1$, $x2$, $y2$, ЧислоЦветов, **цел таб** P[1 : $x2$-$x1$+1, 1 : $y2$-$y1$+1]) принимает те же аргументы, что и алгоритм Создать_плазму плюс массив цветов точек растра P. Алгоритм Рек_Плазма вызывает алгоритм Цвет_точки для того чтобы назначить цвет середине каждой стороны прямоугольника ($x1$, $y1$) – ($x2$, $y2$). Затем алгоритм Рек_Плазма вычисляет цвет центра этого прямоугольника. И, наконец, вызывает сам себя для каждой из четырех прямоугольных частей исходного прямоугольника ($x1$, $y1$) - ($x2$, $y2$), полученных как описывалось выше.

```
алг Рек_Плазма (цел x1, y1, x2, y2, ЧислоЦветов,
                цел таб P[1 : x2-x1+1, 1 : y2-y1+1])
нач
цел x, y
если НЕ(x2 – x1 < 2 и y2 – y1 < 2) то
   x := div(x1 + x2, 2)
   y := div(y1 + y2, 2)
   Цвет_точки(x, y1, x1, y1, x2, y1, ЧислоЦветов, P)
   Цвет_точки(x2, y, x2, y1, x2, y2, ЧислоЦветов, P)
   Цвет_точки(x, y2, x1, y2, x2, y2, ЧислоЦветов, P)
   Цвет_точки(x1, y, x1, y1, x1, y2, ЧислоЦветов, P)
    если P[x, y] = 0 то
       P[x, y] := div(P[x1, y1] + P[x2, y1] + P[x2, y2] + P[x1, y2], 4)
    все
    Рек_Плазма(x1, y1, x, y, ЧислоЦветов, P)
    Рек_Плазма(x, y1, x2, y, ЧислоЦветов, P)
    Рек_Плазма(x, y, x2, y2, ЧислоЦветов, P)
    Рек_Плазма(x1, y, x, y2, ЧислоЦветов, P)
все
кон
```

- Пояснение: div(*x*, *y*) – функция, которая возвращает частное от целочисленного деления *x* на *y*.
- Алгоритм Цвет_точки (**цел** *xc*, *yc*, *xa*, *ya*, *xb*, *yb*) принимает в качестве аргументов координаты точки *xc*, *yc*, координаты концов отрезка, серединой которого эта точка является – *xa*, *ya*, *xb*, *yb*, а также *ЧислоЦветов* и массив *P*. Середине отрезка присваивается цвет, получаемый как среднее арифметическое цветов концов отрезка плюс случайное число, зависящее от длины отрезка.

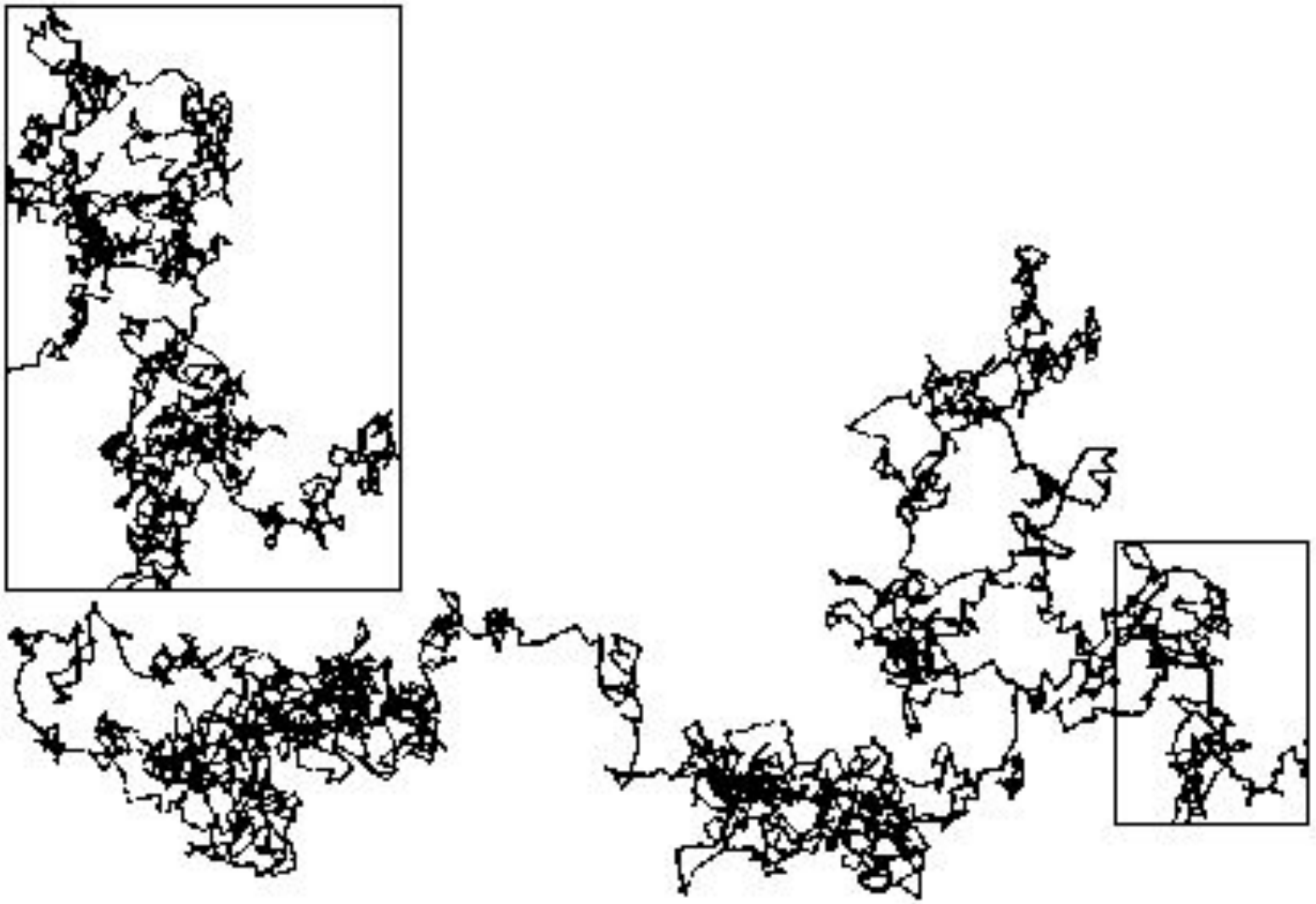
```
алг Цвет_точки (цел xc, yc, xa, ya, xb, yb, ЧислоЦветов,
                цел таб P[1 : x2-x1+1, 1 : y2-y1+1])
нач
цел d
вещ v
    если P[xc, yc] = 0 то
        d := Abs(xa - xb) + Abs(ya - yb)
        v := (P[xa, ya] + P[xb, yb]) / 2 + (СлучЧисло – 0.5) * d * 2
        если v < 0 то
            v := 0
        иначе
            если v > ЧислоЦветов то
                v := ЧислоЦветов
            все
        все
        P[xc, yc] = ОкруглВниз(v)
    все
```

- Пояснение: СлучЧисло – функция, которая случайное вещественное число из интервала [0, 1]; ОкруглВниз(*v*) – функция, которая округляет вещественное число *v* путем отбрасывания дробной части.
- Если представить, что цвет точки «плазмы» это высота над уровнем моря, то фрактал можно считать изображением горного массива. Именно на этом принципе моделируются горы в большинстве графических программ. С помощью похожего алгоритма строится карта высот, к ней применяются различные фильтры, накладывается текстура и получаются фотореалистичные горы.

Броуновское движение

- В 1827 году шотландский ботаник Роберт Броун открыл необычное явление. Он обнаружил, что маленькие частицы цветочной пыльцы, взвешенные в жидкости, непрерывно двигались, описывая причудливые траектории. Это беспорядочное движение взвешенных частиц в жидкости стали называть *броуновским движением*.
- В 1905году Альберт Эйнштейн объяснил это движение хаотическими столкновениями с молекулами жидкости. Молекулы жидкости совершают тепловое нерегулярное движение, которое становится более сильным при возрастании температуры. Молекулы соударяются с более крупными частицами, заставляя менять направление движения.
- Траекторию броуновского движения частицы в жидкости можно рассматривать как фрактальную кривую

Траектория броуновского движения частицы



- Зададим блуждание частицы по плоскости по правилу: если случайное число из интервала [0, 1] меньше 0.5, то частица делает шаг вверх, в противном случае – вниз.
- Аналогично, выбираем, сделает ли она шаг вправо или влево.
- Величину шага также будем выбирать случайным образом из интервала [0, *h_max*], где *h_max* – максимально возможная длина шага. Будем моделировать движение точки в замкнутой прямоугольной области.
- Считаем, что при столкновении частицы с границами области происходит абсолютно упругое (зеркальное) отражение от границ области.

Алгоритм получения траектории броуновского движения

```
алг Броун(цел x1, y1, x2, y2, h_max)
нач
цел x, y, hx, hy
вещ z1, z2
x := div (x1 + x2, 2)
y := div (y1 + y2, 2)
Точка(x, y)
нц пока НЕ НажатаКлавиша
   z1 := СлучЧисло
   z2 := СлучЧисло
   hx := СлучЧисло(h_max)
   hy := СлучЧисло(h_max)
   если z1 < 0.5 то
      x := x + hx
   иначе
      x := x - hx
   все
   если z2 < 0.5 то
      y := y + hy
   иначе
      y := y - hy
   все
   если x <= x1 то
      x := x + 2 * (x1 - x)
   все
   если x >= x1 то
      x := x - 2 * (x - x2)
   все
   если y <= y1 то
      y := y + 2 * (y1 - y)
   все
   если y >= y1 то
      y := y - 2 * (y - y2)
   все
   ЛинияДо(x, y)
кц
кон
```

Рандомизированный метод построения фракталов на основе IFS

Пусть для некоторого фрактала требуются *N* аффинных преобразований.

На начальном шаге рандомизированного алгоритма выбирается одна точка.

Общий шаг алгоритма:

- Случайным образом выбирается одно из *N* аффинных преобразований.
- Образуется новая точка путем применения к предыдущей («старой») выбранного аффинного преобразования.
- Новая точка изображается.
- Выполняется присваивание: новая точка будет рассматриваться в качестве «старой» для следующего шага алгоритма.

Поясним, как выполняется случайный выбор преобразования. Пусть каждому из N аффинных преобразований приписана некоторая вероятность p_i (число от 0 до 1) , $i = 1, \ldots, N$, причём сумма вероятностей всех преобразований равна единице: $\sum_{i=1}^{N} p_i = 1$.

Разобьем интервал $[0;1)$ на числовой оси точками с координатами p_1, $p_1 + p_2$, $p_1 + p_2 + p_3$, …, $p_1 + \ldots + p_{N-1}$ на N частичных интервалов Δ_1, Δ_2, …, Δ_N: Δ_1 – $[0; p_1)$, Δ_2 – $(p_1; p_1 + p_2)$, …, Δ_N - $(p_1 + \ldots + p_{N-1}; 1)$.

Выберем с помощью датчика случайных чисел R – случайное вещественное число от 0 до единицы.

Если R попало в частичный интервал Δ_i, то выберем i-ое преобразование.

Для того чтобы в процессе случайного выбора преобразования с малым детерминантом не появлялись слишком часто, имеет смысл производить выбор с вероятностями, пропорциональными детерминантам. Для этого определим вероятности следующим образом:

$$p_i = \det(T_i) \Big/ \sum_{i=1}^{N} T_i \,,\ i = 1, \ldots, N.$$

Очевидно, что условие $\sum_{i=1}^{N} p_i = 1$ выполняется.

Алгоритм построения фрактала «Папоротник» на основе IFS

Одним из самых известных фракталов, для построения которых эффективно применяются системы итерируемых функций, является лист папоротника. Для его построения применяются четыре аффинных преобразования со следующими коэффициентами и соответствующими вероятностями:

№	Вероятность	Коэффициенты
1	0.01	$A=0, B=0, C=0, D=0.16, E=0, F=0$
2	0.85	$A=0.85, B=0.04, C=-0.04, D=0.85, E=0, F=1.6$
3	0.07	$A=0.2, B=-0.26, C=0.23, D=0.22, E=0, F=1.6$
4	0.07	$A=-0.15, B=0.28, C=0.26, D=0.24, E=0, F=0.44$

Лист папоротника

```
алг Папоротник (цел MaxIt)
| MaxIt – максимальное число итераций
нач
вещ x, y, x1, y1, A, B, C, D, E, F, r
цел i
ИнитСлуч
x := 0
y := 0
нц для i от 1 до MaxIt
    r := СлучЧисло
    если r <= 0.01 то
        A = 0; B = 0; C = 0; D = 0.16; E = 0; F = 0
    иначе
        если r <= 0.86 то
            A = 0.85; B = 0.04; C = -0.04; D = 0.16; E = 0.85; F = 1.6
        иначе
```

```
        иначе
            если r <= 0.93 то
                A = 0.2; B = -0.26; C = 0.23; D = 0.22; E = 0; F = 1.6
            иначе
                A = -0.15; B = 0.28; C = 0.26; D = 0.24; E = 0; F = 0.44
            все
        все
    все
    x1 := A * x + B * y + E
    y1 := C * x + D * y + F
    x := x1
    y := y1
    Изобразить точку (x, y) зелёным цветом
кц
кон
```

Фрактальные деревья

Рассмотрим процесс построения фрактального дерева.

Сначала строится ствол дерева случайной длины, от него проводятся несколько ветвей тоже случайной длины, при этом толщина уменьшается, далее от каждой ветки строится еще несколько веток (от некоторых ничего не строится), и т.д.

При этом на каждом шаге проверяется длина ветки, если она меньше некоторой заранее определенной величины, то вместо ветки рисуется лист, и для этой ветки процесс прекращается.

Опишем рекурсивный алгоритм построения фрактального дерева на псевдокоде.

Будем считать, что объявлены глобальные величины:

- ***Ветвь*** – параметр, влияющий на «ветвистость» дерева, чем он меньше, тем более пышным получается дерево;
- ***l_min*** – минимальная длина ветки,
- ***l_max*** – максимальная длина ветки.

Алгоритм принимает в качестве аргументов координаты; угол наклона ветки (***a***), который вначале представляет собой угол наклона ствола дерева; длину ветки (***l***), которая в начале представляет собой длину ствола.

```
алг Дерево (цел x, y, вещ a, цел l)
нач
цел x1, y1, p
если l >= l_min то
   x1 := Округл(x + l*cos(a));
   y1 := Округл(y + l*sin(a));
   если l > l_max то
      p := l_max
   иначе
      p := l
   все
   если p < 40 то
   | Генерация листьев
       если СлучЧисло > 0.5 то
          УстановитьЦвет(1)
```

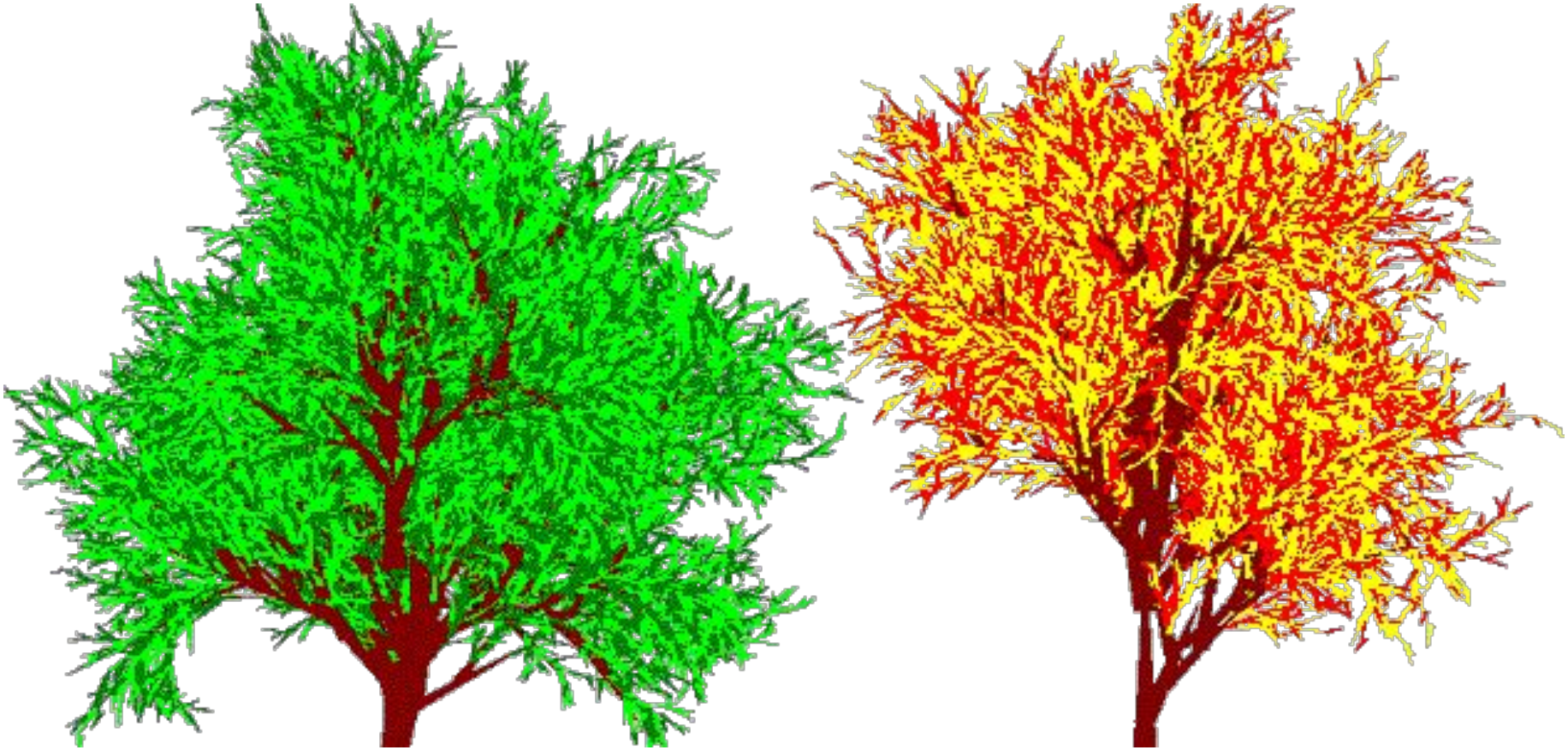
```
        иначе
            УстановитьЦвет(2)
        все
        нц для i от 0 до 3
            Линия(x + i, y, x1, y1)
        кц
    иначе
      | Генерация веток
        УстановитьЦвет(3);
        нц для i от 0 до div(p, 6)
            Линия(x + i - div(p, 12), y, x1, y1)
        кц
    все
    | Следующие ветки
    нц для i от 0 до 9 - СлучЧисло(9)
        s := СлучЧисло (l - div(1, 6)) + div(l, 6);
        x1:= Округл(x + s * cos(a));
        y1:= Округл(y + s * sin(a));
        Дерево(x1, y1, a + 1.6 * (0.5 - СлучЧисло),
                    p - 5 - СлучЧисло(Ветвь))
     кц
    все
кон
```

Пояснение:

- Округл(*v*) – функция, которая округляет вещественное число до ближайшего целого;
- УстановитьЦвет(*k*) – процедура, которая устанавливает цвет №*k* из палитры в качестве текущего цвета рисования линий;
- Линия(*xa*, *ya*, *xb*, *yb*) – процедура, которая рисует линию от точки (*xa*, *ya*) до точки (*xb*, *yb*).

В зависимости от выбранной палитры получается летнее или осеннее дерево

Варианты фрактальных деревьев



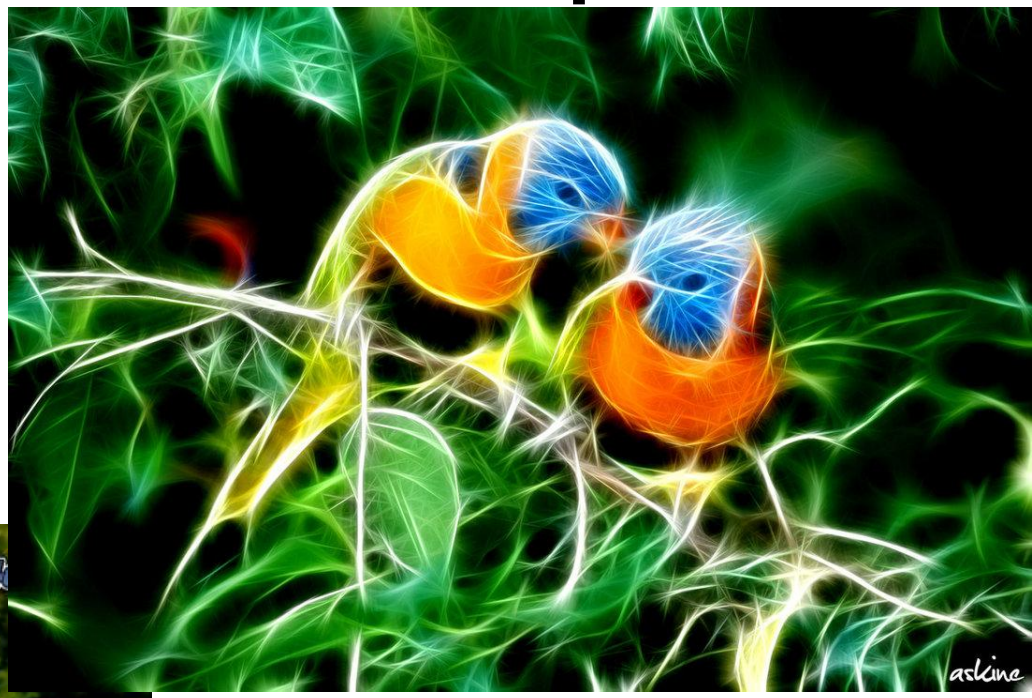
Трехмерное фрактальное дерево

- И еще раз о применении фракталов…

Фракталы о природе


http://svezduh.ru/

Фракталы о животном мире



Фракталы в архитектуре

В музыке

Музыкант Джонатан Колтон на основе фрактальных алгоритмов пишет музыку.

По его утверждениям фрактальные мелодии наиболее полно соответствуют природной гармонии. Все свои произведения Колтон публикует под лицензией, предусматривающей их свободное распространение, копирование и передачу.

И даже в дизайне мебели….

Японский дизайнер Такеши Миякава использовал принцип фрактальности при создании мебели, а именно одной из моделей тумбочек.

Она состоит из 23 ящиков, причем ящики расположены так, что практически полностью используют все выделенное под тумбочку пространство в форме куба.

Полезные ресурсы

- http://www.fractalworld.xaoc.ru/
- http://fraktalsworld.blogspot.ru/