ПОСТРОЕНИЕ ПРОСТЕЙШИХ ФРАКТАЛЬНЫХ МНОЖЕСТВ С ПРИМЕРАМИ

А. Р. Баранцев

alexei.barantsev@gmail.com

19 сентября 2024 г.

Системы итерированных функций.

Игра в хаос и треугольник Серпинского. Пусть u_0, u_1, u_2 — три точки на плоскости. Возьмем произвольную точку $v_0 \in \mathbb{R}^2$ и построим последовательность по правилу

$$v_{n+1} := \frac{v_n + u_i}{2}, \qquad n = 0, 1, \ldots, \tag{1}$$

где $i \in 0 : 2$ — случайное число, выбираемое произвольно на каждом шаге. То есть, на каждом шаге случайным образом выбирается одна из трех опорных точек u_0, u_1, u_2 и в качестве следующей точки последовательности выбирается середина отрезка, соединяющего текущую точку v_n и случайно выбранную опорную точку u_i.

Процесс проиллюстрирован с помощью GeoGebra[1] в файле 01-Chaotic-Sequence.ggb.

[1]К тексту прилагается набор ggb-файлов, предназначенных для просмотра в GeoGebra. Файлы ggb можно просмотреть онлайн, не устанавливая приложение GeoGebra нв компьютере. Для этого достаточно зайти на `https://www.geogebra.org/calculator` и загрузить ggb-файл.

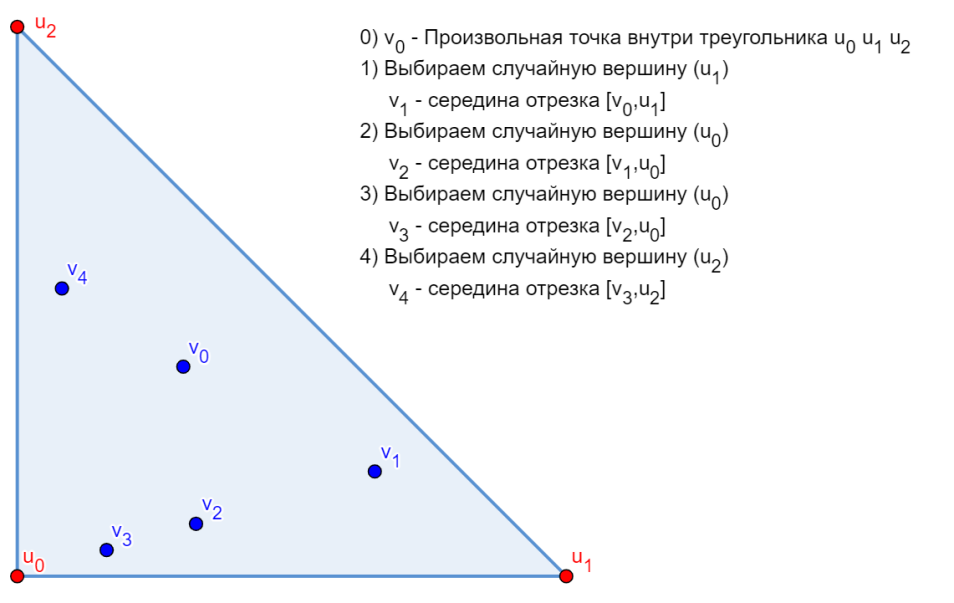

Рис. 1: 01-Chaotic-Sequence.ggb

В GeoGebra можно потаскать мышкой начальную точку v_0 и понаблюдать, как перемещаются другие точки последовательности. Можно также добавить одну или несколько точек в последовательность. Для этого надо выбрать инструмент «Середина», а затем выбрать две точки - последнюю точку последовательности (v_4) и одну из опорных точек u_0, u_1, u_2, а затем переименовать получившуюся точку как v_5.

Программа DEMO01[2] показывает на экране последовательность точек $\{v_n\}$ для начальных данных $u_0 = (0, 0)$, $u_1 = (1, 0)$, $u_2 = (0, 1)$.

[2]К тексту прилагается набор тестовых программ на Turbo Pascal. Для компиляции и запуска среды Turbo Pascal в современном Windows можно установить TPWDB - Turbo Pascal With DOSBox - `https://turbopascal-wdb.sourceforge.io/` Каждый пример записан в отдельном фале: DEMO01.PAS, DEMO02.PAS, и т.д. Для запуска примеров необходимы файлы WORLD.PAS и EGAVGA.BGI, которые должны находиться в той же директории , где и демонстрационные примеры.

Программа DEMO01 выводит на экран точки последовательности, пока не будет нажата клавиша. Для выхода из программы следует нажать клавишу ENTER.

```
PROGRAM Demo01;                                          1
                                                         2
USES Crt,World;                                          3
                                                         4
VAR v: Point;                                            5
CONST N= 2;                                              6
VAR i: 0..N;                                             7
VAR u: ARRAY[0..N] OF Point;                             8
                                                         9
BEGIN                                                    10
  u[0].x:= 0.0; u[0].y:= 0.0;                            11
  u[1].x:= 1.0; u[1].y:= 0.0;                            12
  u[2].x:= 0.0; u[2].y:= 1.0;                            13
  v:= u[0];                                              14
  REPEAT                                                 15
    Dot(v);                                              16
    i:= Random(SUCC(N));                                 17
    v.x:= (v.x+u[i].x)*0.5;                              18
    v.y:= (v.y+u[i].y)*0.5;                              19
  UNTIL KeyPressed;                                      20
                                                         21
  ReadLn;                                                22
  Shutdown;                                              23
END.                                                     24
```

В начале (строки 11-13) инициализируются опорные точки.
Начальная точка последовательности выбирается как u_0 (строка 14).
В цикле REPEAT (строки 15-20):

- Текущая точка последовательности выводится на экран (строка 16)
- Случайным образом выбирается одна из опорных точек (строка 17)
- Вычисляются координаты следующей точки последовательности (строки 18-19)

Процедура Dot выводит точку на экран. Она реализована в модуле World.

Результат работы программы представлен на рис. 2.

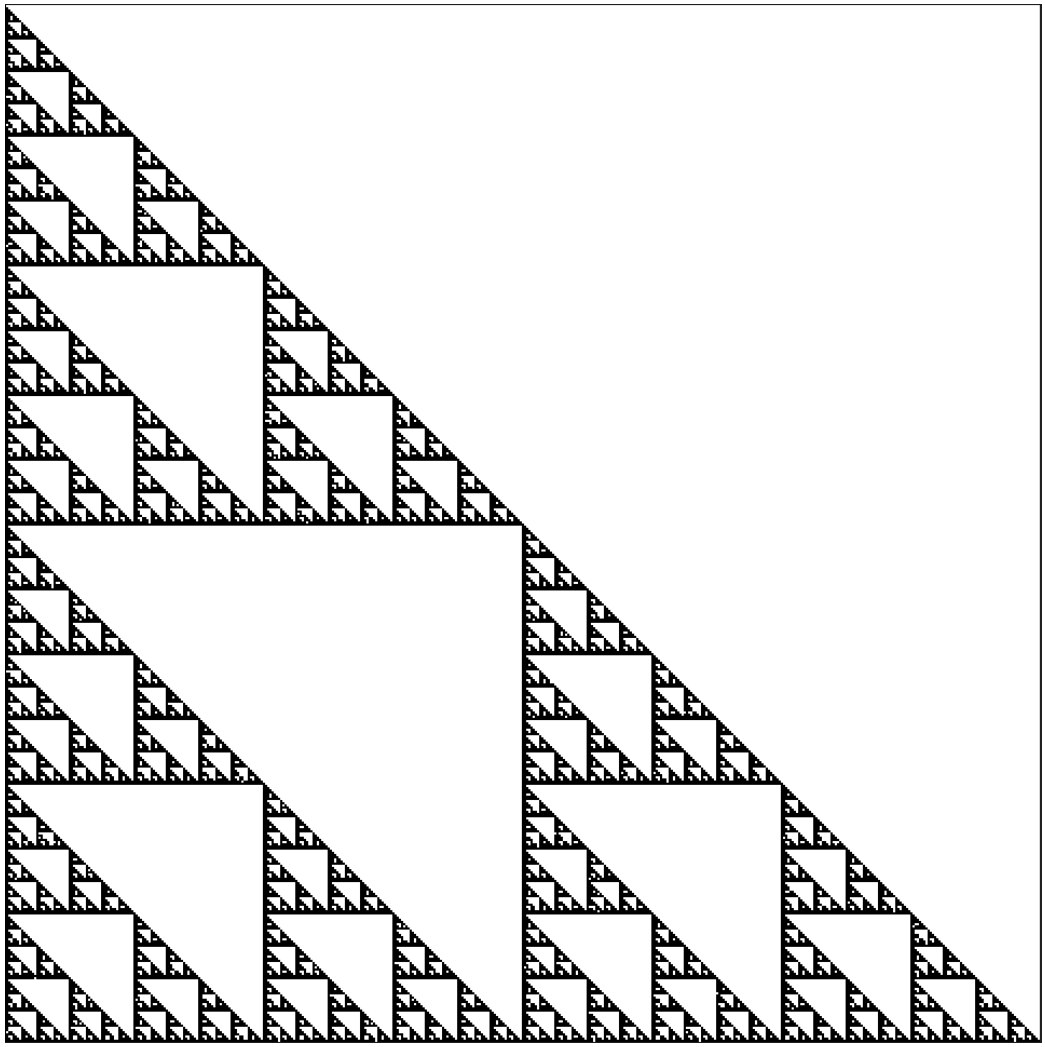

Рис. 2: DEMO01. Треугольник Серпинского. Игра в хаос.

Видно, что точки не хаотически разбросаны внутри треугольника, а образуют правильную структуру, известную как треугольник Серпинского.

Три следующие файла GeoGebra наглядно объясняют, почему точки распределяются именно таким образом.

В первом файле объяснено, почему центральный треугольник остается пустым.

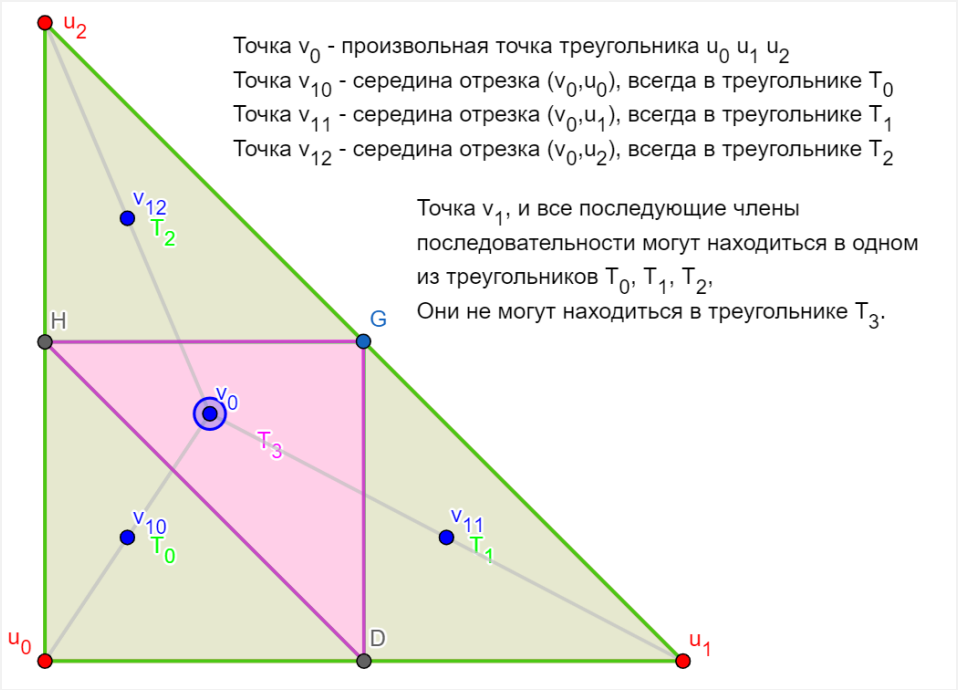

Рис. 3: 02-v0-v1.ggb

Здесь точка v_0 - начальная точка последовательности, а точки v_{10}, v_{11}, v_{12} представляют собой возможных кандидатов на то, чтобы стать следующим членом последовательность v_1 - в зависимости от того, какая из опорных точек u_0, u_1, u_2 будет выбрана случайным образом.

Перемещая точку v_0 внутри опорного треугольника, можно убедиться, что точки v_{10}, v_{11}, v_{12} остаются в пределах треугольников T_0, T_1, T_2.

Видно, что ни при каких обстоятельствах точка v_1, равно как и все последующие члены последовательности, не может оказаться в центральном треугольнике T_3.

В следующем файле прослеживается, куда может попасть точка последоваетльности на второй итерации.

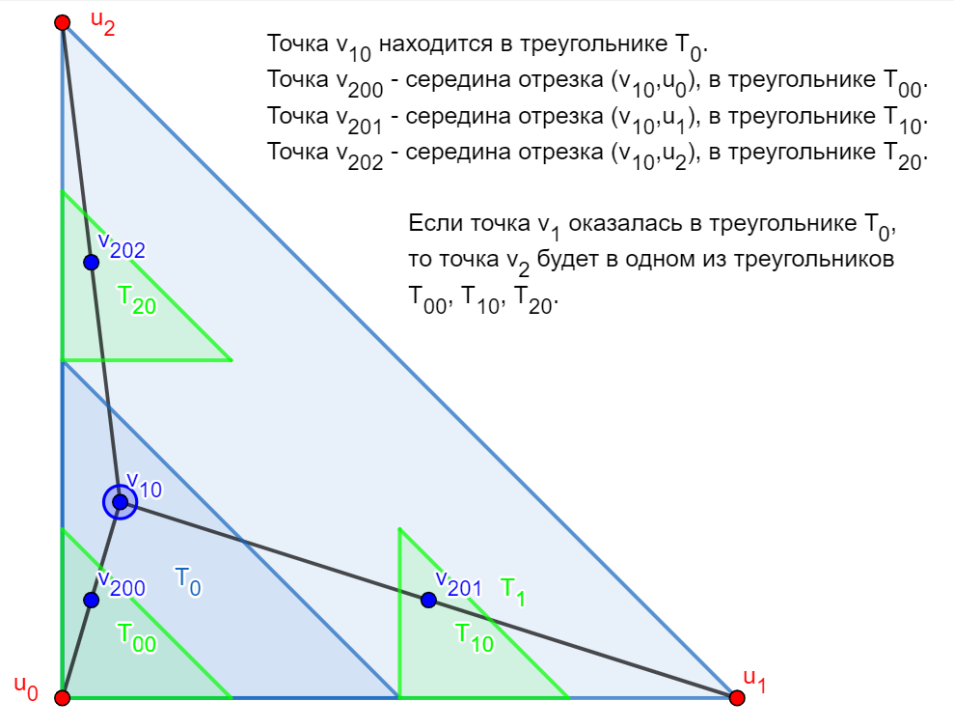

Рис. 4: 03-v10-v20.ggb

Здесь точка v_{10} - возможный кандидат на члена последовательности v_1 - в том случае, если была выбрана опорная точка u_0.
Кандидаты на то, чтобы стать v_2 - это точки v_{200}, v_{201}, v_{202}, опять в зависимости от того, какая из опорных точек выбрана на следующем шаге.
Перемещая точку v_{10} внутри треугольника T_0, можно убедиться, что точки v_{200}, v_{201}, v_{202} остаются в пределах треугольников T_{00}, T_{10}, T_{20}.
Видно, что если на первой итерации была выбрана опорная точка u_0, то на второй итерации точка будет находиться в одном из зеленых треугольников T_{00}, T_{10}, T_{20}.

Наконец, в третьем файле демонстрируется, что все точки последовательности, начиная со второй, будут находиться в одном из зеленых треугольников, и не могут попасть ни в один из красных.

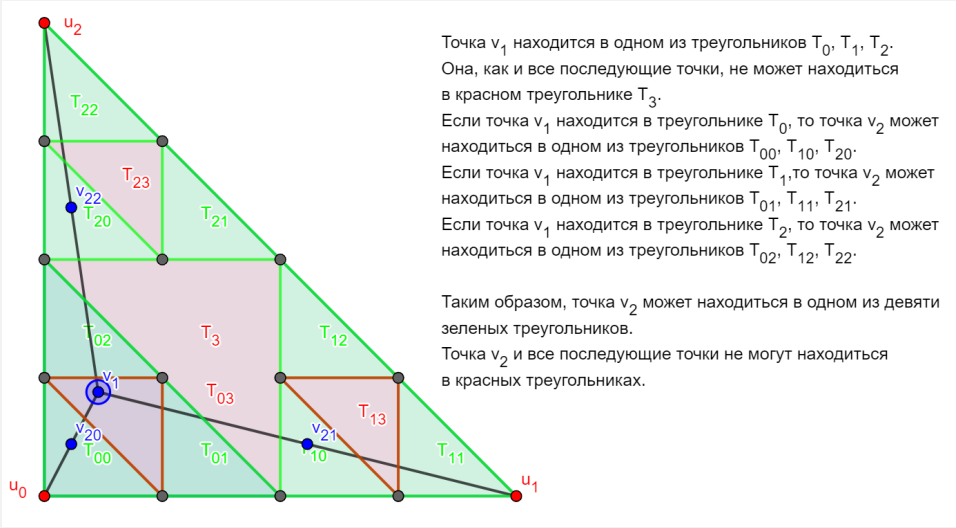

Рис. 5: 04-v1-v2.ggb

Как установлено выше, точка v_1 может находиться в одном из треугольников T_0, T_1, T_2, и не может находиться в центральном треугольнике T_3.
Точки v_{20}, v_{21}, v_{22} - возможные кандидаты на то, чтобы стать v_2, в зависимости от случайного выбора опорной точки на второй итерации.
Если перемещать точку v_1 внутри треугольника T_0, то точки v_{20}, v_{21}, v_{22} перемещаются внутри треугольников T_{00}, T_{10}, T_{20}.
Если перемещать точку v_1 внутри треугольника T_1, то точки v_{20}, v_{21}, v_{22} перемещаются внутри треугольников T_{01}, T_{11}, T_{21}.
Если перемещать точку v_1 внутри треугольника T_2, то точки v_{20}, v_{21}, v_{22} перемещаются внутри треугольников T_{02}, T_{12}, T_{22}.
Таким образом, в каком бы из треугольников T_0, T_1, T_2 ни находилась точка v_1, следующий член последовательности v_2, а также все последующие, попадет в один из зеленых треугольников, и не окажется ни в одном из красных треугольников.

Продолжая рассуждения таким образом, видно, почему точки последовательности образуют треугольник Серпинского.

Сжимающие отображения. Пусть $X_0 = \{u_0, u_1, u_2\}$ —множество из трех точек на плоскости. Построим последовательность множеств по правилу

$$X_{n+1} := \frac{X_n + \{u_0\}}{2} \cup \frac{X_n + \{u_1\}}{2} \cup \frac{X_n + \{u_2\}}{2}$$

На каждом шаге строятся три уменьшенных (в два раза) копии множества X_n, состоящие из полусумм точек множества X_n и опорных точек u_i. Объединение этих трех копий дает следующий элемент последовательности X_{n+1}. На следующем шаге получится 9 уменьшенных в четыре раза копий исходного трехточечного множества и т.д.

Программа DEMO02 изображает на экране множество X_8 для тех же начальных данных.

Программа DEMO02 выводит на экран восьмое множество X_8. Для выхода из программы следует нажать клавишу ENTER.

```
PROGRAM Demo02;                                                    1
                                                                   2
USES Crt,World;                                                    3
                                                                   4
CONST N= 2;                                                        5
VAR i: 0..N;                                                       6
TYPE Triangle= ARRAY[0..N] OF Point;                               7
VAR u: Triangle;                                                   8
                                                                   9
VAR d: WORD;                                                       10
PROCEDURE Draw(VAR t: Triangle);                                   11
  VAR v: Triangle;                                                 12
  VAR i,j: 0..N;                                                   13
  PROCEDURE Draw0(VAR t: Triangle);                                14
  VAR i: 0..N;                                                     15
    BEGIN                                                          16
      FOR i:= 0 TO N DO Dot(t[i]);                                 17
    END;                                                           18
  BEGIN                                                            19
    IF d=0 THEN BEGIN Draw0(t); EXIT; END;                         20
    FOR i:= 0 TO N DO                                              21
      BEGIN                                                        22
        FOR j:= 0 TO N DO                                          23
          BEGIN                                                    24
            v[j].x:= (t[j].x+u[i].x)*0.5;                          25
            v[j].y:= (t[j].y+u[i].y)*0.5;                          26
          END;                                                     27
        DEC(d);                                                    28
        Draw(v);                                                   29
        INC(d);                                                    30
      END;                                                         31
  END;                                                             32
                                                                   33
BEGIN                                                              34
  u[0].x:= 0.0; u[0].y:= 0.0;                                      35
  u[1].x:= 1.0; u[1].y:= 0.0;                                      36
  u[2].x:= 0.0; u[2].y:= 1.0;                                      37
  d:= 8;                                                           38
  Draw(u);                                                         39
                                                                   40
  ReadLn;                                                          41
  Shutdown;                                                        42
END.                                                               43
```

Процедура Draw0 (строки 14-18) отображает на экране вершины треугольника.

Процедура Draw либо вызывает процедуру Draw0 (строка 20), либо рекурсивно вызывают саму себя трижды (строка 29), для каждого из подтреугольников.

Результат работы программы представлен на рис. 6.

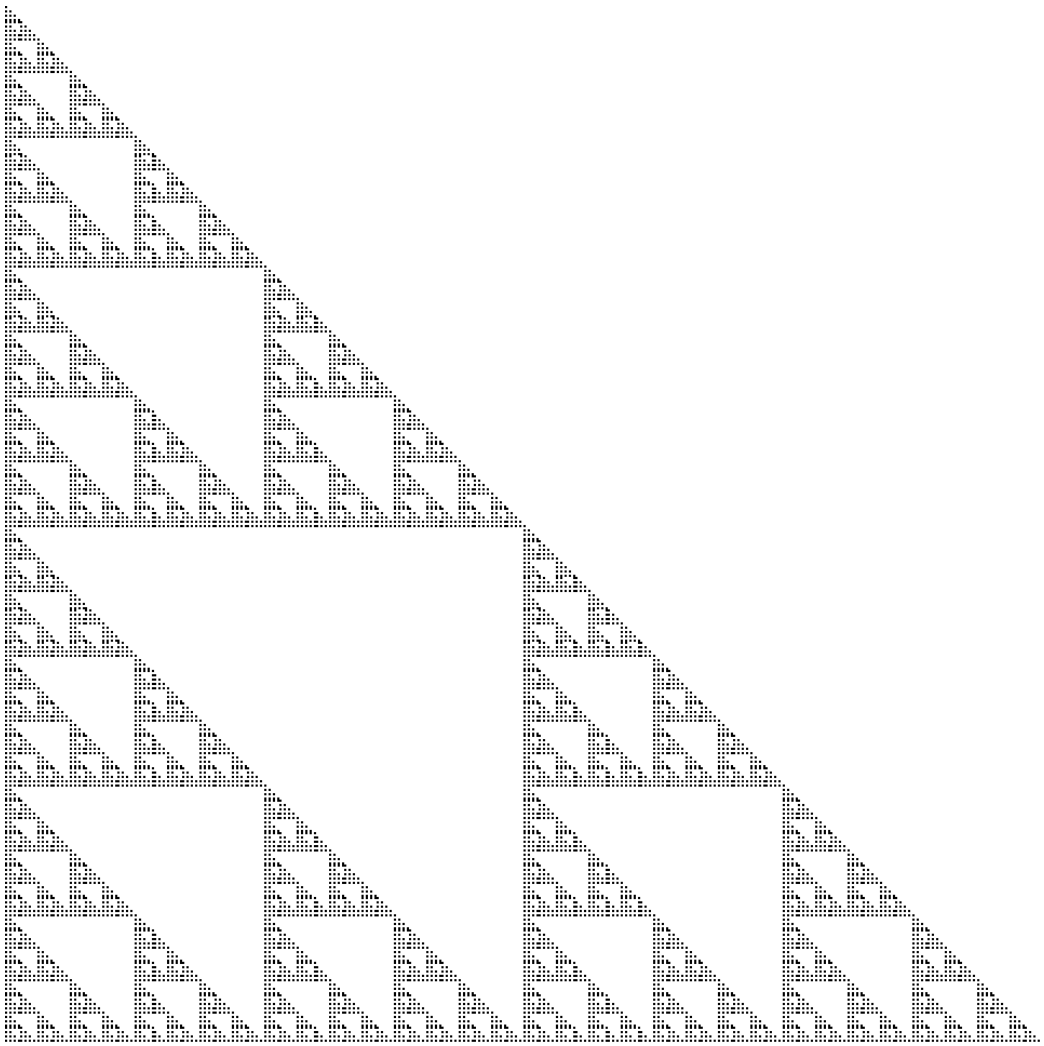

Рис. 6: DEMO02. Треугольник Серпинского. Рекуррентное сжатие трех точек. Восьмая итерация.

В программе DEMO03 в качестве начального множества X_0 взята граница треугольника, натянутого на точки u_0, u_1, u_2. На шаге n множество X_n является объединением 3^n треугольников, каждый из которых получен из исходного треугольника уменьшением в 2^n раз.

Результат X_5 представлен на рис. 7.

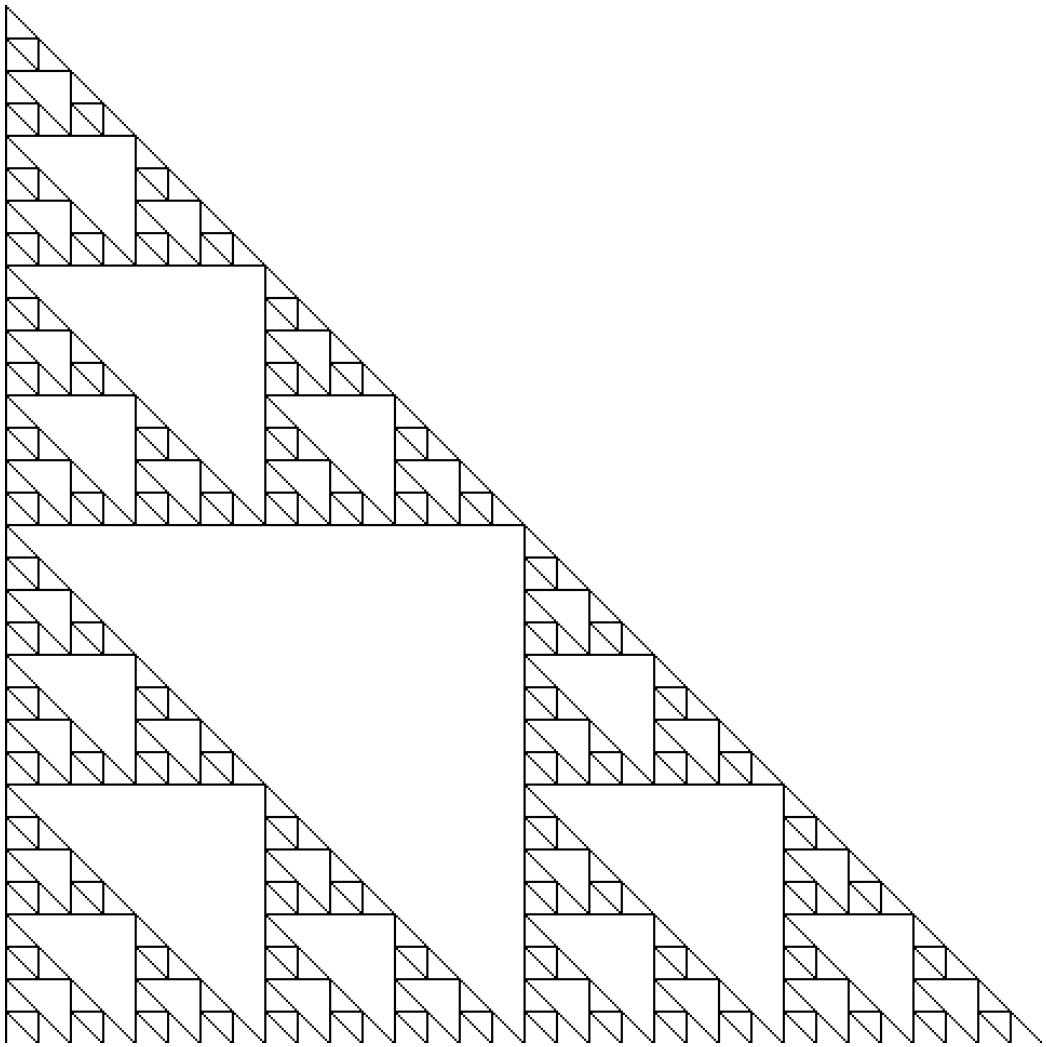

Рис. 7: DEMO03. Треугольник Серпинского. Рекуррентное сжатие треугольника. Пятая итерация.

В программе DEMO04 в качестве начального множества X_0 взята граница единичного квадрата. Результат X_6 представлен на рис. 8.

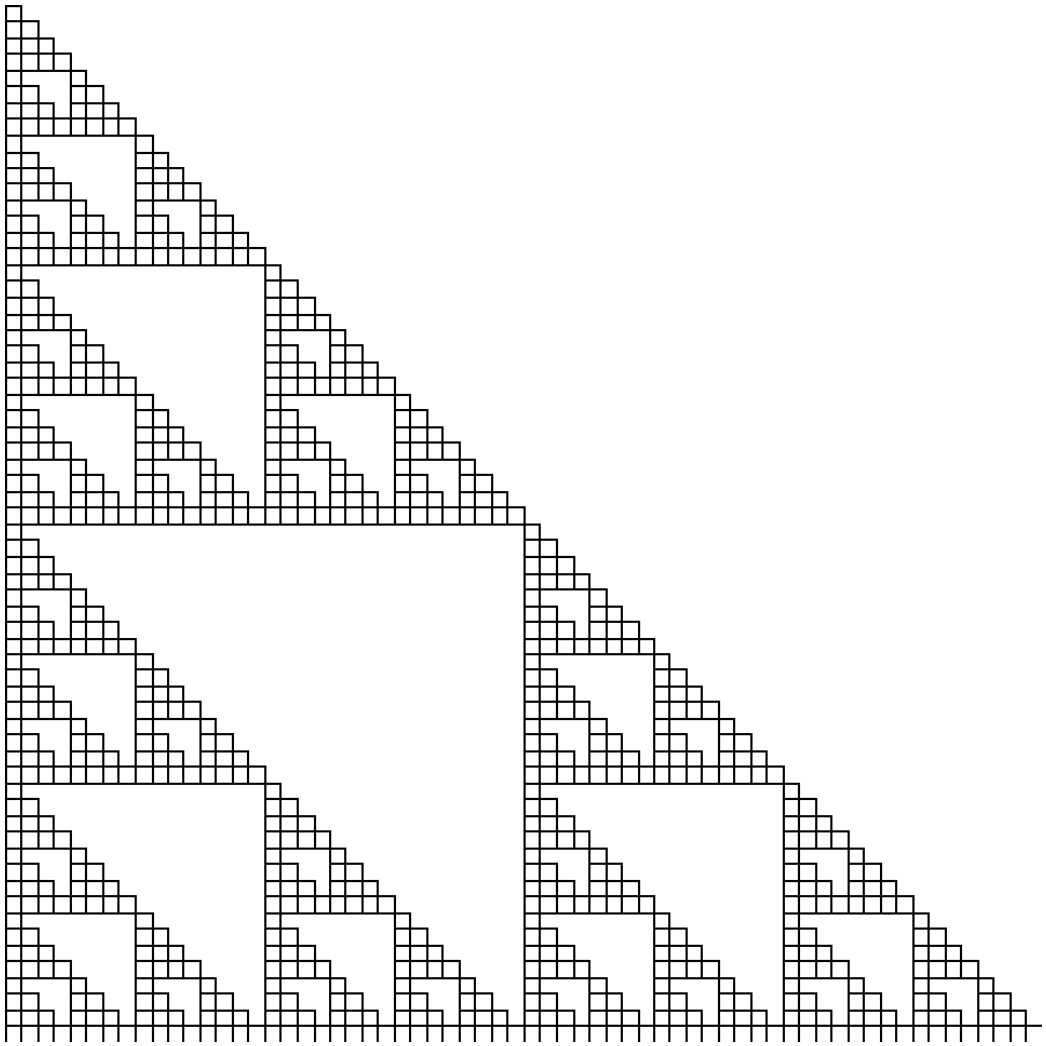

Рис. 8: DEMO04. Треугольник Серпинского. Рекуррентное сжатие пустого квадрата. Шестая итерация.

В программе DEMO05 в качестве начального множества X_0 взят единичный квадрат. Результат X_6 представлен на рис. 9.

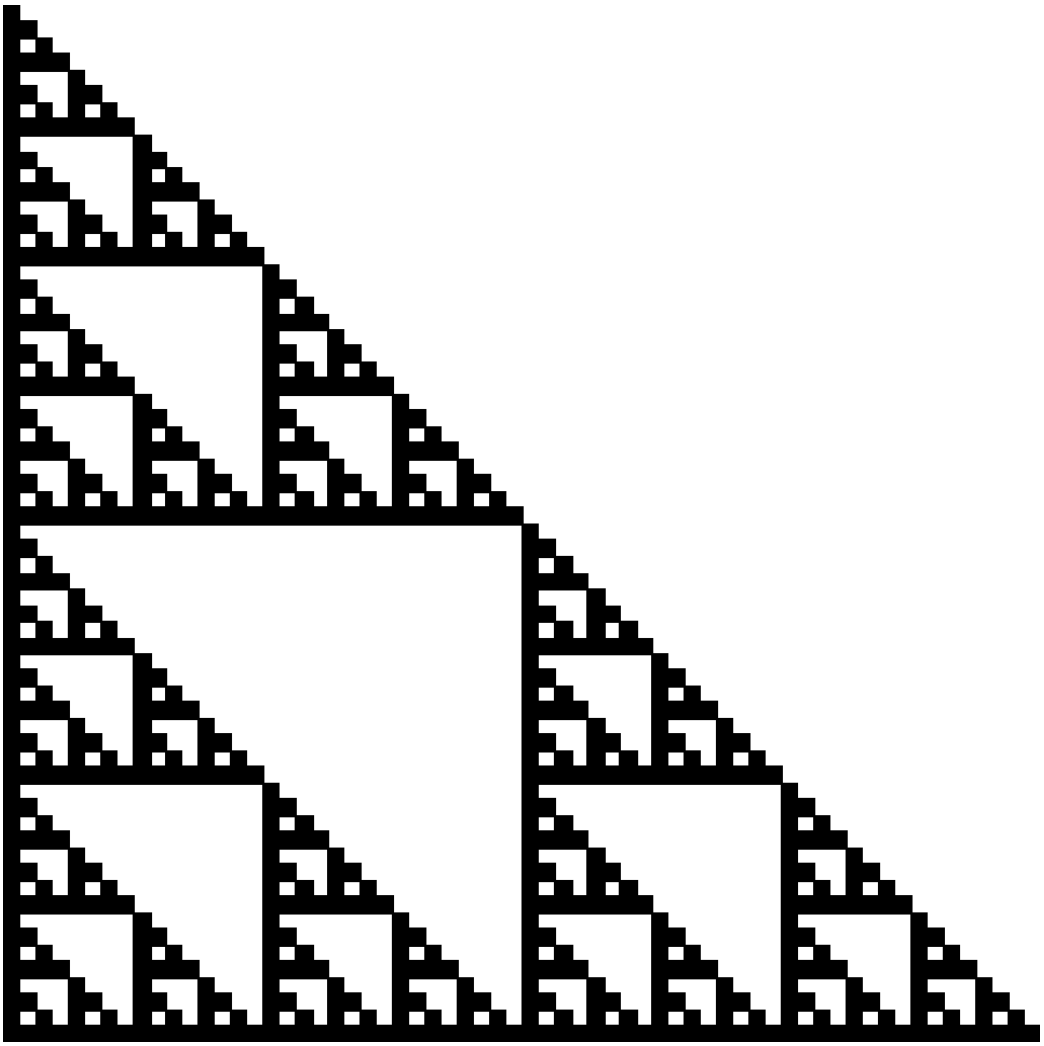

Рис. 9: DEMO05. Треугольник Серпинского. Рекуррентное сжатие заполненного квадрата. Шестая итерация.

Во всех представленных вариантах DEMO02–DEMO05 результат при $n \to \infty$ оказывается одинаковым, вне зависимости от начального множества X_0.

Сжимающие аффинные преобразования. Процесс, задаваемый (1) можно описать в терминах *аффинных преобразований* на плоскости.

Варьируя коэффициенты и число преобразований, можно получать самые экзотические изображения.

Аффинные преобразования на плоскости имеют вид

$$w(v) = Av + B = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot v + \begin{pmatrix} B_1 \\ B_2 \end{pmatrix}$$

Для процесса (1), аффинные преобразования w_0, w_1, w_2 задаются диагональными матрицами A, равными

$$A_0 = A_1 = A_2 = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}$$

и векторами $B_i = u_i/2$.

В этом случае

$$v_{n+1} = w_i(v_n), \qquad n = 0, 1, \ldots, \tag{2}$$

где по-прежнему $i \in 0 : 2$ — случайное число, выбираемое произвольно на каждом шаге.

Программа DEMO06 показывает на экране последовательность точек $\{v_n\}$ для начальных данных $u_0 = (0, 0)$, $u_1 = (1, 0)$, $u_2 = (0, 1)$.

Программа DEMO06 делает то же самое, что и программа DEMO01, но на этот раз с использованием сжимающих аффинных преобразований.

```
PROGRAM Demo06;                                                    1
USES Crt,World;                                                    2
CONST N= 2;                                                        3
VAR i: 0..N;                                                       4
TYPE Map= RECORD a11,a12,a21,a22,b1,b2: Scalar; END;               5
VAR w: ARRAY[0..N] OF Map;                                         6
VAR v: Point;                                                      7
                                                                   8
PROCEDURE Transform(VAR m: Map; VAR p: Point);                     9
  VAR x,y: Scalar;                                                 10
  BEGIN                                                            11
    x:= m.a11*p.x + m.a12*p.y + m.b1;                              12
    y:= m.a21*p.x + m.a22*p.y + m.b2;                              13
    p.x:= x; p.y:= y;                                              14
  END;                                                             15
                                                                   16
BEGIN                                                              17
  WITH w[0] DO BEGIN                                               18
    a11:= 0.5; a12:= 0.0; a21:= 0.0; a22:= 0.5;                    19
    b1:= 0.0; b2:= 0.0;                                            20
    END;                                                           21
  WITH w[1] DO BEGIN                                               22
    a11:= 0.5; a12:= 0.0; a21:= 0.0; a22:= 0.5;                    23
    b1:= 0.5; b2:= 0.0;                                            24
    END;                                                           25
  WITH w[2] DO BEGIN                                               26
    a11:= 0.5; a12:= 0.0; a21:= 0.0; a22:= 0.5;                    27
    b1:= 0.0; b2:= 0.5;                                            28
    END;                                                           29
  v.x:= 0.0; v.y:= 0.0;                                            30
  REPEAT                                                           31
    Dot(v);                                                        32
    i:= Random(SUCC(N));                                           33
    Transform(w[i],v);                                             34
  UNTIL KeyPressed;                                                35
  ReadLn;                                                          36
  Shutdown;                                                        37
END.                                                               38
```

Процедура Transform (строки 9-15) выполняет аффинное преобразование.
Матрицы A и столбцы B трех аффинных преобразований инициализированы в строках 18-29.
В цикле REPEAT (строки 31-35) текущая точка последовательности отображается на экране (строка 32), затем случайным образом выбирается одно из преобразований (строка 33), и вызывается процедура Transform (строка 34), которая применяет это преобразование к текущей точке последовательностию

Результат работы программы представлен на рис. 10. Как и следовало ожидать, он совпадает с рисунком 2.

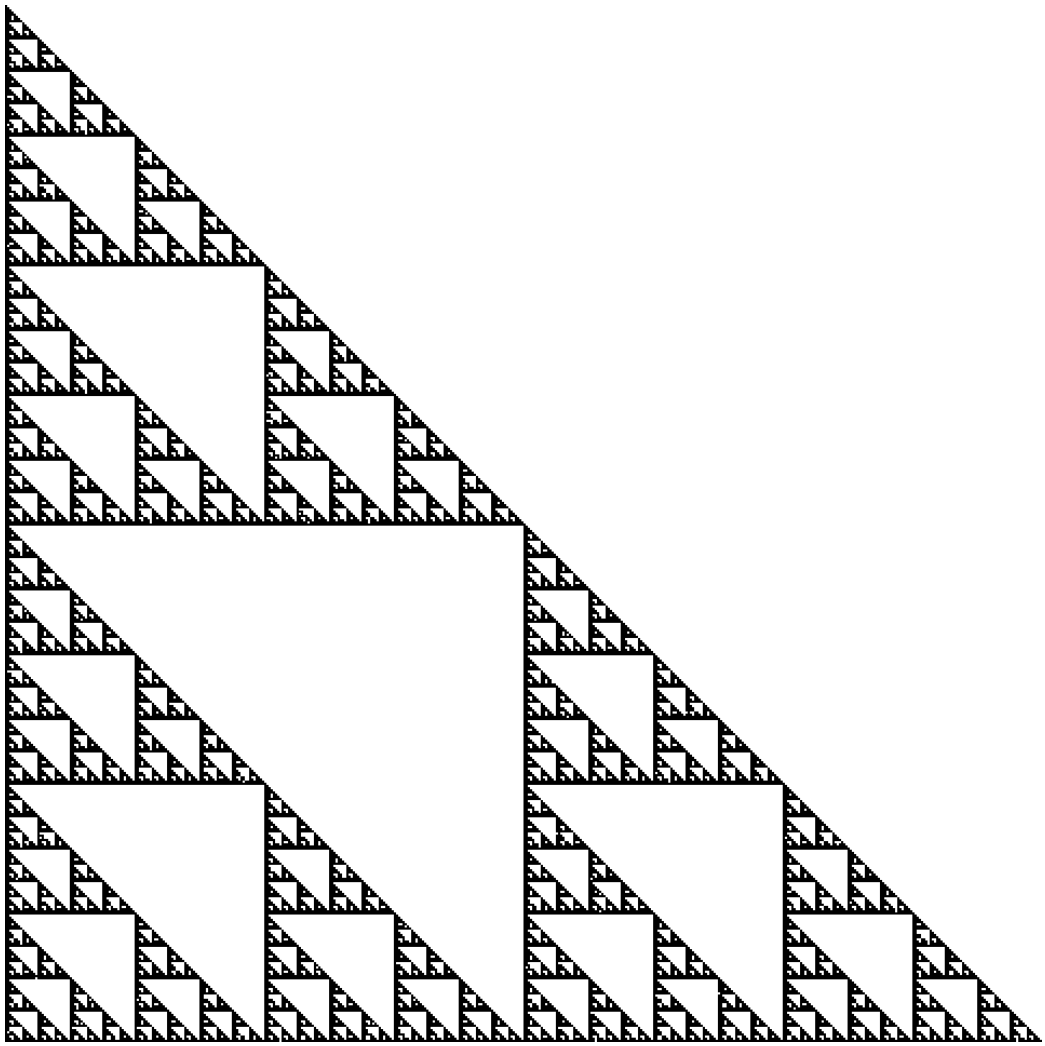

Рис. 10: DEMO06. Треугольник Серпинского. Сжимающие аффинные отображения.

В программе DEMO07 строится последовательность точек для четырех аффинных преобразований

$$A_0 = \begin{pmatrix} 0.849 & 0.037 \\ -0.037 & 0.849 \end{pmatrix} \qquad B_0 = \begin{pmatrix} 0.075 \\ 0.1830 \end{pmatrix}$$

$$A_1 = \begin{pmatrix} 0.197 & -0.226 \\ -0.226 & 0.197 \end{pmatrix} \qquad B_1 = \begin{pmatrix} 0.400 \\ 0.0490 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} -0.150 & 0.283 \\ 0.283 & 0.237 \end{pmatrix} \qquad B_2 = \begin{pmatrix} 0.575 \\ -0.0840 \end{pmatrix}$$

$$A_3 = \begin{pmatrix} 0.0 & 0.0 \\ 0.0 & 0.160 \end{pmatrix} \qquad B_3 = \begin{pmatrix} 0.5 \\ 0.0 \end{pmatrix}$$

Результат представлен на рис. 11.

Рис. 11: DEMO07. Папоротник Барнсли. Равновероятный выбор.

В программе DEMO08 аффинные преобразования те же, но случайное число i принимает значения $0, 1, 2, 3$ с вероятностями $0.73, 0.13, 0.11, 0.03$ соответственно.

Программа DEMO08 применяет аффинные преобразования с разными вероятностями.

```
PROGRAM Demo08;                                                    1
USES Crt,World;                                                    2
CONST N= 3;                                                        3
VAR i: 0..N;                                                       4
TYPE Map= RECORD                                                   5
  a11,a12,a21,a22,b1,b2: Scalar; p: 0..100;                        6
  END;                                                             7
VAR w: ARRAY[0..N] OF Map;                                         8
VAR u,v: Point;                                                    9
                                                                   10
PROCEDURE Transform(VAR m: Map; VAR p: Point);                     11
  VAR x,y: Scalar;                                                 12
  BEGIN                                                            13
    x:= m.a11*p.x + m.a12*p.y + m.b1;                              14
    y:= m.a21*p.x + m.a22*p.y + m.b2;                              15
    p.x:= x; p.y:= y;                                              16
  END;                                                             17
                                                                   18
BEGIN                                                              19
  WITH w[0] DO BEGIN                                               20
    a11:= 0.849; a12:= 0.037; a21:= -0.037; a22:= 0.849;           21
    b1:= 0.075; b2:= 0.1830; p:= 73; END;                          22
  WITH w[1] DO BEGIN                                               23
    a11:= 0.197; a12:= -0.226; a21:= 0.226; a22:= 0.197;           24
    b1:= 0.400; b2:= 0.0490; p:= 13; END;                          25
  WITH w[2] DO BEGIN                                               26
    a11:= -0.150; a12:= 0.283; a21:= 0.260; a22:= 0.237;           27
    b1:= 0.575; b2:= -0.0840; p:= 11; END;                         28
  WITH w[3] DO BEGIN                                               29
    a11:= 0.0; a12:= 0.0; a21:= 0.0; a22:= 0.160;                  30
    b1:= 0.5; b2:= 0.0; p:= 3; END;                                31
                                                                   32
  v.x:= 0.0; v.y:= 0.0;                                            33
  REPEAT                                                           34
    u.x:= 1.2*v.y; u.y:= 1.4*v.x-0.3;                              35
    Dot(u);                                                        36
    i:= SUCC(Random(100));                                         37
    IF i<w[0].p THEN i:=0                                          38
    ELSE IF i<w[0].p+w[1].p THEN i:=1                              39
    ELSE IF i<w[0].p+w[1].p+w[2].p THEN i:=2                       40
    ELSE i:= 3;                                                    41
    Transform(w[i],v);                                             42
  UNTIL KeyPressed;                                                43
                                                                   44
  ReadLn;                                                          45
  Shutdown;                                                        46
END.                                                               47
```


В строке 32 точка отражается относительно диагонали (x меняется на y), масштабируется и сдвигается из эстетических соображений. Неравновероятный случайный выбор преобразования осуществляется в строках 35-38.

Результат, отраженный относительно главной диагонали, представлен на рис. 12.

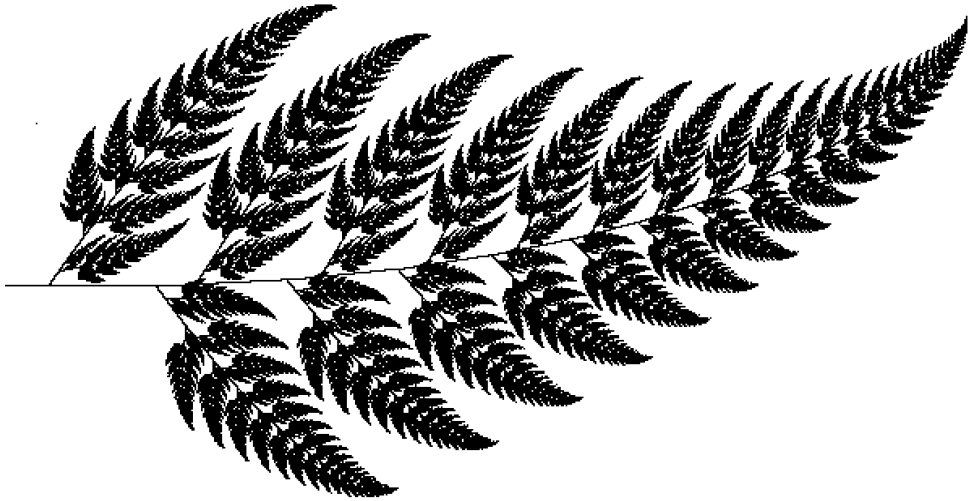

Рис. 12: DEMO08. Папоротник Барнсли со специально подобранными вероятностями.

Хаусдорфова метрика. Пусть (X, d) — метрическое пространство. Будем обозначать через $\mathcal{H}(X)$ множество всех компактных подмножеств X. *Расстоянием между точкой* $x \in X$ *и множеством* $B \in \mathcal{H}(X)$ назовем

$$d(x, B) := \min_{y \in B} d(x, y).$$

Расстоянием от множества $A \in \mathcal{H}(X)$ *до множества* $B \in \mathcal{H}(X)$ называется

$$d(A, B) := \max_{x \in A} d(x, B).$$

В общем случае, $d(A, B) \neq d(B, A)$.

Хаусдорфово расстояние между $A \in \mathcal{H}(X)$ и $B \in \mathcal{H}(X)$ есть

$$h(A, B) := \max\{d(A, B), d(B, A)\}.$$

Здесь всегда $h(A, B) = h(B, A)$ и $(\mathcal{H}(X), h)$ — метрическое пространство.

Если метрическое пространство (X, d) полно, то метрическое пространство $(\mathcal{H}(X), h)$ также полно.

Понятие Хаусдорфова расстояния наглядно проиллюстрировано в презентации GeoGebra [3].

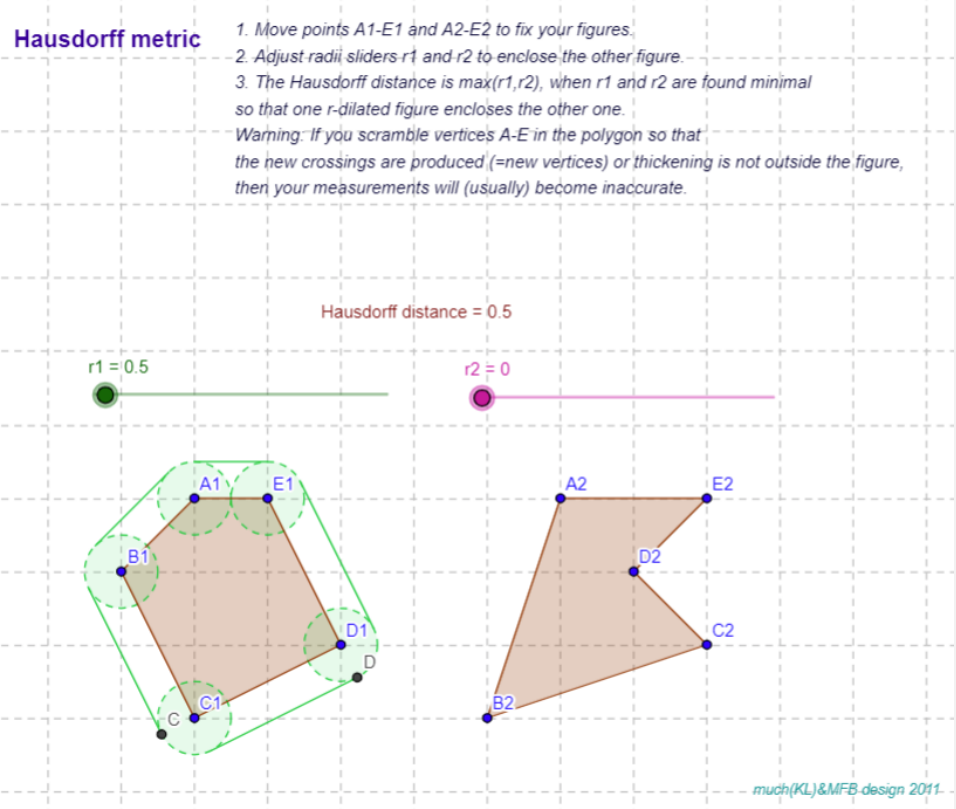

Рис. 13: Hausdorff_distance.ggb

Здесь вычисляется расстояние между двумя пятиугольниками $A1$-$E1$ и $A2$-$E2$. Двигая левый бегунок $r1$ можно определить расстояние от первого множества до второго.
Двигая правый бегунок $r2$ можно определить расстояние от второго множества до первого.
Хаусдорфово расстояние - максимальное из $r1$ и $r2$.
Перемещая вершины пятиугольников, можно почувствовать как при этом меняется хаусдорфово расстояние.
В том числе, можно определить расстояние между пересекающимися множемтвами.

Преобразование $f : X \to X$ называется *сжимающим*, если существует константа $s \in [0, 1)$ такая, что

$$d(f(x), f(y)) \leq s \cdot d(x, y) \quad \forall x, y \in X.$$

Число s называется *фактором сжатия* f.

Если $f(x) = x$, то точка x — *фиксированная точка отображения* f.

Теорема о сжимающем отображении. Пусть $f : X \to X$ — сжимающее отображение на метрическом пространстве (X, d). Тогда существует единственная фиксированная точка x_f для отображения f. Более того, для всякой точки $x \in X$ последовательность $\{f^{\circ n}(x)\}$, $n = 0, 1, 2, \ldots$ сходится к x_f.

Система итерированных функций состоит из полного метрического пространства (X, d) и конечного множества сжимающих отображений $w_n : X \to X$ с соответствующими факторами сжатия $s_n, n = 0, 1, 2, \ldots, N$. Ее фактор сжатия есть $s := \max_{n \in 0:N} s_n$.

Теорема об аттракторе. Пусть $\{X; w_n, n \in 0 : N\}$ — система итерированных функций с фактором сжатия s. Тогда преобразование $W : \mathcal{H}(X) \to \mathcal{H}(X)$

$$W(B) := \bigcup_{n=0}^{N} w_n(B) \qquad \forall B \in \mathcal{H}(X)$$

есть сжимающее отображение в полном метрическом пространстве $\mathcal{H}(X), h(d))$ с фактором сжатия s. Именно,

$$h(W(B), W(C)) \leq s \cdot h(B, C)$$

для всех $B.C \in \mathcal{H}X)$. Его единственная фиксированная точка, $A \in \mathcal{H}(X)$ обладает свойством

$$A = W(A) = \bigcup_{n=0}^{N} w_n(A), \tag{3}$$

и для всякого $B \in \mathcal{H}(X)$

$$A = \lim_{n \to \infty} W^{\circ n}(B)$$

Фиксированная точка $A \in \mathcal{H}(X)$ называется *аттрактором* системы итерированных функций.

В (3) множество A представлено в виде объединения своих уменьшенных копий. Такое представление называют *коллажем* аттрактора A.

Теорема о коллаже (Барнсли 1985). Пусть (X, d) — полное метрическое пространство. Пусть $L \in \mathcal{H}(X)$ — некоторое множество, а число ϵ неотрицательно. Пусть удалось подобрать такую систему итерированных функций $\{X; w_0, w_1, \ldots, w_N\}$, с факторм сжатия $s \in [0, 1)$, что

$$h\left(L, \bigcup_{n=0}^{N} w_n(L)\right) \leq \epsilon,$$

где $h(d)$ — Хаусдорфова метрика. Тогда

$$h(L, A) \leq \epsilon/(1-s),$$

где A есть аттрактор упомянутой системы итерированных функций. Формально,

$$h(L, A) \leq (1-s)^{-1} h\left(L, \bigcup_{n=0}^{N} w_n(L)\right) \qquad \forall L \in \mathcal{H}(X).$$

С помощью теоремы о коллаже можно строить систему итерированных функций, аттрактор которой близок к заданному множеству в смысле Хаусдорфовой метрики.

Например, в случае треугольника Серпинского коллаж собирается из трех уменьшенных копий треугольника, как показано на рис. 14.

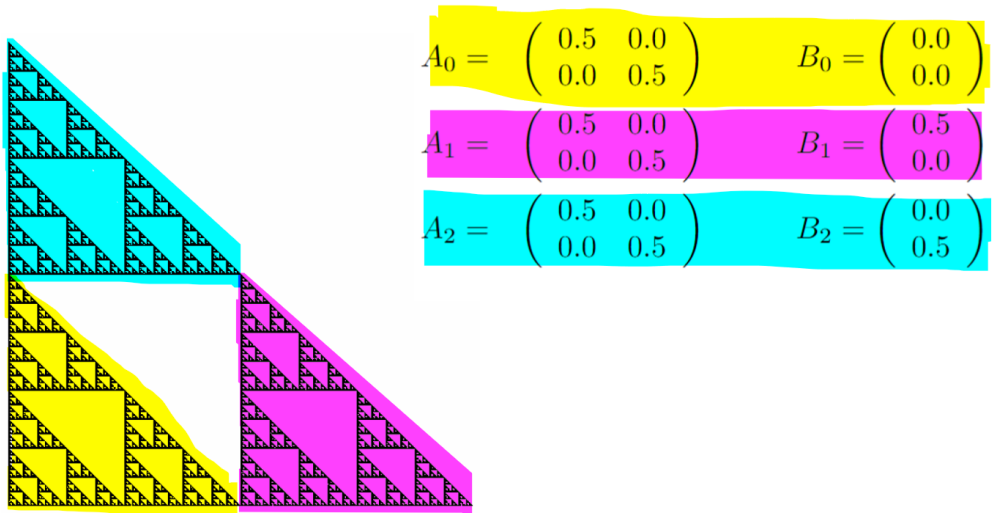

Рис. 14: Коллаж треугольника Серпинского.

Коллаж листа папоротника Барнсли собирается из четырех уменьшенных копий.

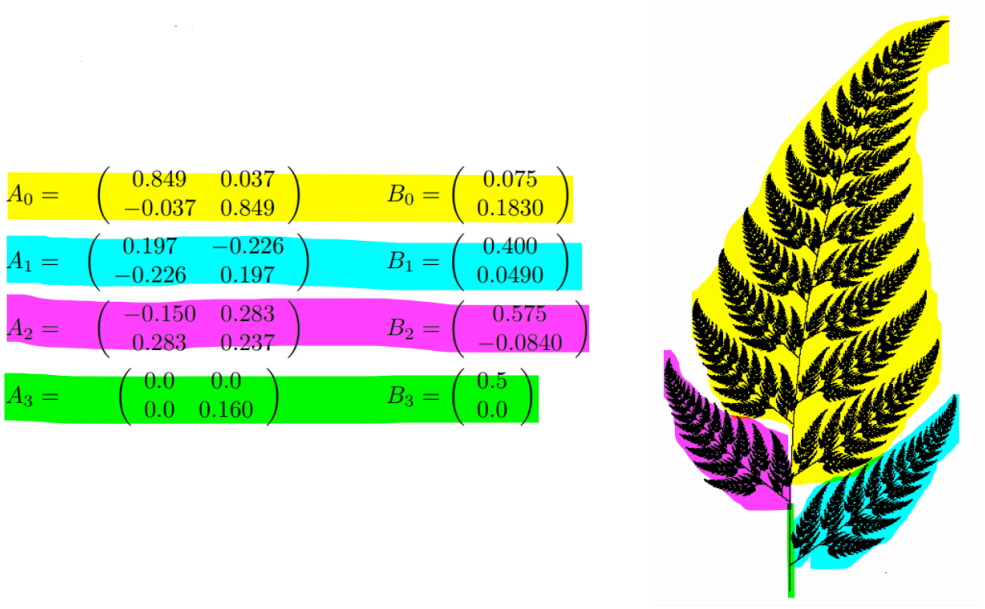

Рис. 15: Коллаж треугольника Серпинского.

При этом, последнее отображение (зеленый цвет) сжимает лист папоротника в кусочек черенка.

Адресные схемы

Пусть S — треугольник Серпинского. Тогда нижний левый под-треугольник равен $w_0(S)$, два других равны $w_1(S)$ и $w_2(S)$. Их объединение дает S:

$$S = w_0(S) \cup w_1(S) \cup w_2(S).$$

Присвоим под-треугольникам номера $0, 1, 2$ соответственно. В под-треугольнике с номером 0 можно выделить три под-под-треугольника, которым присвоим адреса $00, 01, 02$. Продолжая, получим адресную схему рис. 16. Например, треугольник с адресом 021 равен $w_0(w_2(w_1(S)))$.

Каждая точка z треугольника Серпинского является пересечением бесконечного числа вложенных под-треугольников, ее содержащих. Таким образом можно идентифицировать точки треугольника по их адресам:

$$\text{адрес}(z) = s_1 s_2 s_3 \dots$$

Если читать адрес слева направо, то он интерпретируется как последовательность вложенных треугольников, содержащих точку z, все более точ-

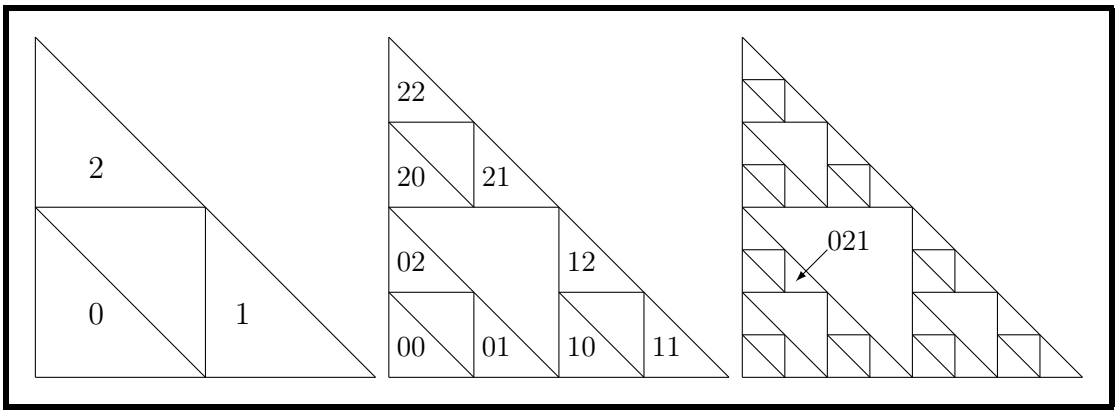

Рис. 16: Треугольник Серпинского. Адресная схема.

но локализующих ее местоположение. Не все точки обладают уникальными адресами. Например, точка с адресом $0222\ldots$ совпадает с точкой с адресом $2000\ldots$. Эта ситуация аналогична с десятичным представлением вещественных чисел.

Всякому аттрактору системы итерированных функций может быть сопоставлено соответствующее пространство адресов. Именно, если система итерированных функций задается сжимающими отображениями $w_0, w_1, \ldots, w_N$, то любая точка из аттрактора A_∞ имеет адрес в пространстве Σ_N всех бесконечных последовательностей $s_1 s_2 s_3 \ldots$, где каждое число s_i принадлежит множеству $0 : N$.

Возьмем некоторую последовательность k чисел $s_1 s_2 \ldots s_k$, $s_i \in 0 : 2$. Пусть последние три числа суть $1, 2, 0$. Построим последовательность точек по правилу (2), выбирая номера сжимающих отображений из нашей последовательности. Тогда последняя точка имеет вид $w_0(w_2(w_1(...)))$. Адрес треугольника, в который она попадает, равен 021..., то есть, определяется выбранной последовательностью, если читать ее справа налево.

Таким образом, для каждого треугольника произвольного уровня, можно утверждать, что точка попадет в него, если среди сгенерированной последовательности случайных чисел найдется подстрока, совпадающая с адресом этого треугольника, прочитанным в обратном порядке. Это соображение объясняет поведение процесса (2).

Три следующие демонстрационных файла GeoGebra наглядно демонстрируют сказанное на примере адреса 021.

В первом файле изображены вложенные треугольники, соответствующие адресу 021.

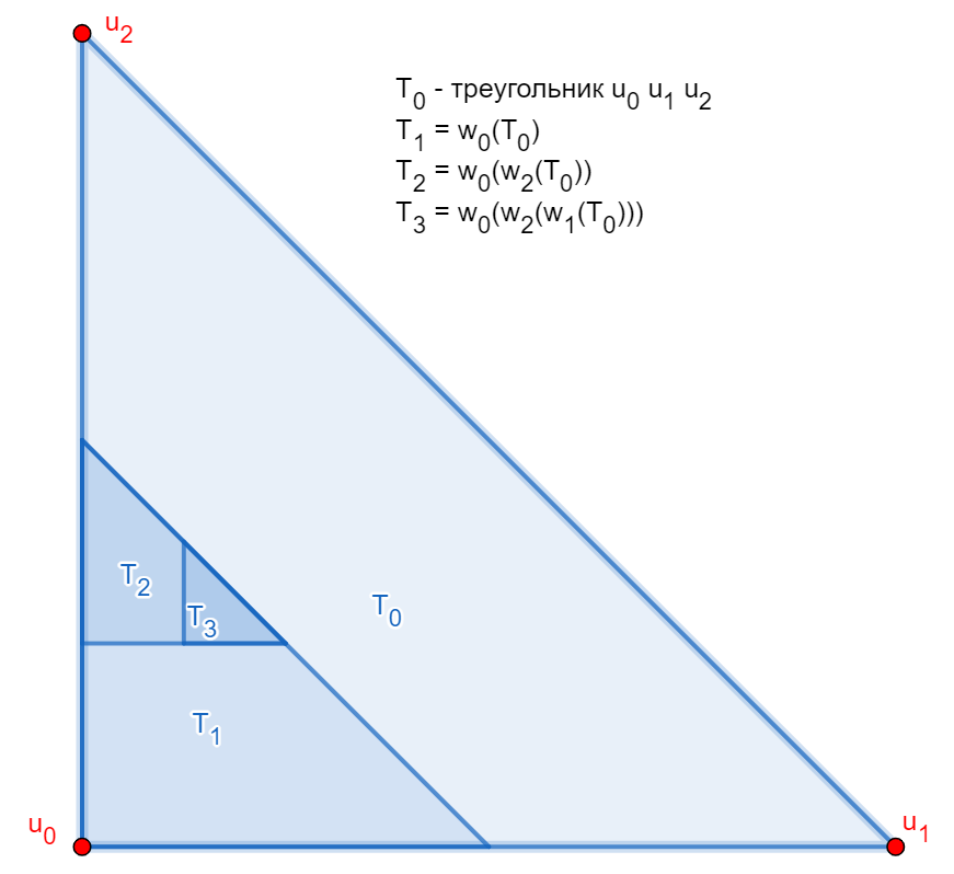

Рис. 17: 05-address-T.ggb

T_0 - треугольник, натянутый на опорные точки u_0, u_1, u_2. Треугольники T_1, T_2, T_3 строятся путем применения преобразований w_i в соответствии с адресом 021, если читать его слева направо. Получается последовательность вложенных треугольников.

Во втором файле отображения w_i применяются справа налево, образуя хаотическую последовательность.

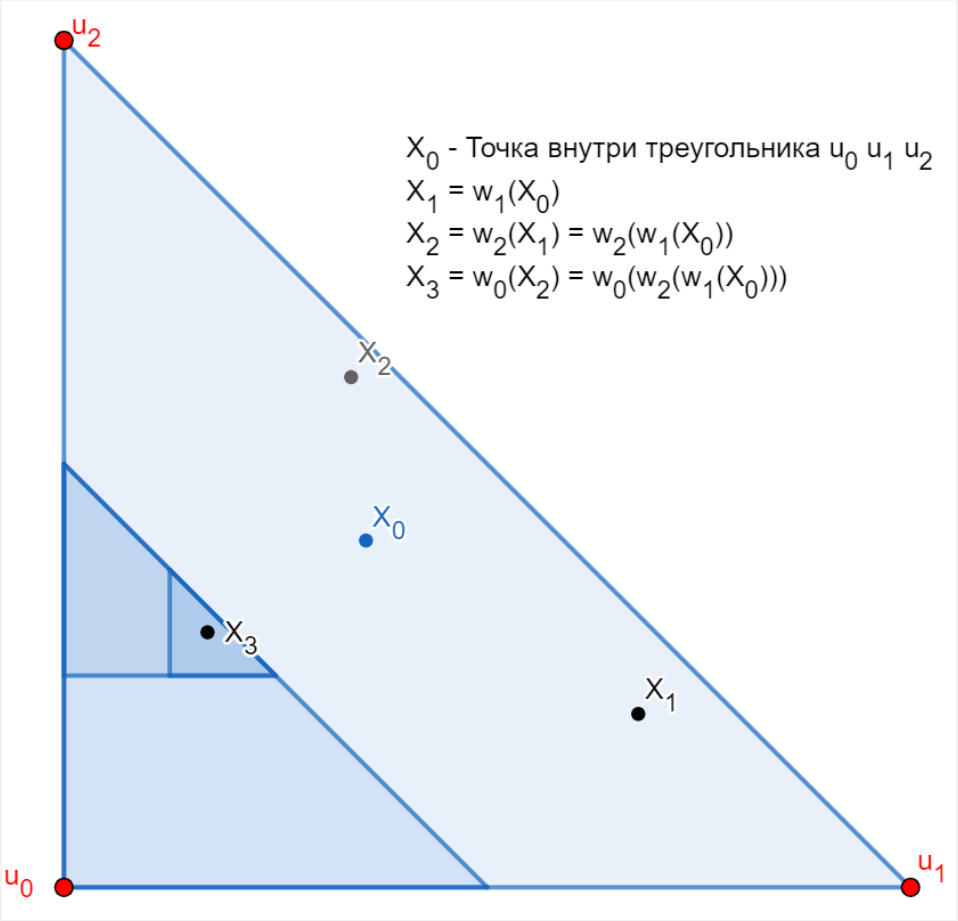

Рис. 18: 06-address-X.ggb

X_0 - начальная точка последовательности, которую можно перемещать мышкой внутри большого треугольника.
Точки X_1, X_2, X_3 - результат последовательного применения отображений w_1, w_2, w_0.
В данном случае адрес 021 читается справа налево.
Точки хаотично разбросаны по треугольнику, но последняя точка находится внутри треугольника T_3 рис. 17.

Наконец, во третьем файле отображения w_i применяются слева направо, образуя сходящуюся последовательность.

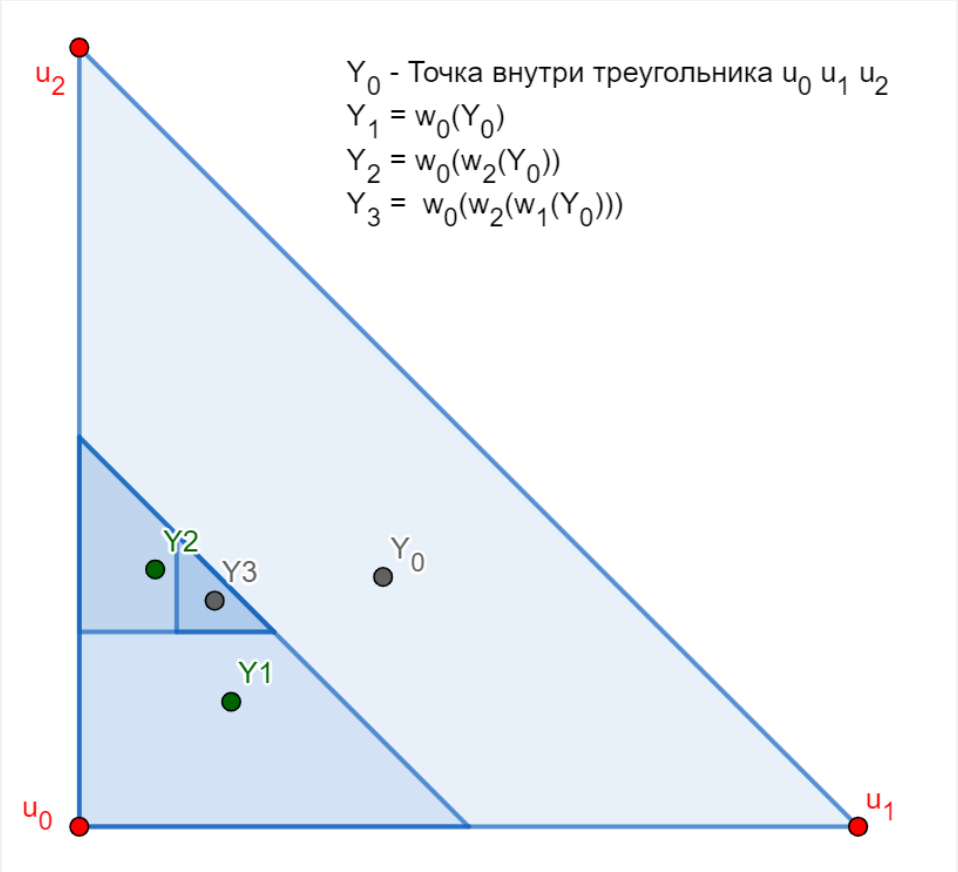

Рис. 19: 07-address-Y.ggb

Y_0 - начальная точка последовательности, которую можно перемещать мышкой внутри большого треугольника.
Точки Y_1, Y_2, Y_3 - результат последовательного применения отображений w_0, w_2, w_1.
В данном случае адрес 021 читается слева направо.
Видно, что точки образуют сходящуюся последовательность, а последняя точка находится внутри треугольника T_3 рис. 17.

Таким образом, последовательность X_n на рисунке 18 и последова-

тельность Y_n на рисунке 19 дают один и тот же результат, но в первом случае адрес считывается справа налево, а во втором - слева направо.

В первом случае получается хаотически разбросанная последовательность, которая формирует аттрактор системы итерированных функций.

В последнем случае последовательность сходится к некой точке треугольника Серпинского, но не покрывает его весь.

Список литературы

[1] M. F. Barnsley. *Fractals everywhere.* ACADEMIC PRESS, 1988. 399 p.

[2] H. -O. Peitgen, H. Jürgens, D. Saupe *Chaos and Fractals. New Frontiers of Science.* SPINGER VERLAG, 1992. 984 p.

[3] megaloxantha *Hausdorff distance between two sets.* `https://www.geogebra.org/u/megaloxantha`

[4] slik *Chaos Game.* `https://www.geogebra.org/m/GfSknh5u`

[5] rob *Fractal Fern.* `https://www.geogebra.org/m/UqC7FRZP`