УДК 004.92

Рисование правильного многоугольника и фрактала «треугольник Серпинского» на языке Python

Черкашин Александр Михайлович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Целью исследования является написание скрипта по рисованию правильного многоугольника и фрактала «треугольник Серпинского». В работе приведен скрипт рисования фрактала из треугольников Серпинского. В результате нарисован фрактал, представляющий собой многоугольник, состоящий из треугольников Серпинского.
Ключевые слова: треугольник Серпинского, фрактал, многоугольник, python.

Drawing a regular polygon and a fractal "Sierpinski triangle" in Python

Cherkashin Alexander Mihailovich
Sholom-Aleichem Priamursky State University
Student

Abstract
The purpose of the study is to write a script for drawing a regular polygon and a fractal Sierpinski triangle. The paper describes a script for drawing a fractal from Sierpinski triangles. As a result of the work, a fractal is drawn, which is a polygon consisting of Sierpinski triangles.
Keywords: Sierpinski triangle, fractal, polygon, python.

1 Введение

1.1 Актуальность исследования

Данная статья описывает возможность написание скрипта для рисования правильного многоугольника и фрактала.

1.2 Цель исследования

Целью исследования является написание скрипта на языке Python для рисования правильного многоугольника и фрактала.

1.3 Обзор исследований

Т. Хованова и др. описывают треугольник Серпинского на некоторых клеточных автоматах, например, правило 90, в том числе, которых относятся к игре в жизни [1]. Е. Конверсано исследует дизайн и рисунки треугольник Серпинского на церквях в Риме 11 веке [2]. П. Брунори, П. Магрон, Л. Т. Лолли исследуют рисунки треугольника Серпинского в средневековых церквях [3]. Ж. И. Латропа, Ж. Х. Луц, С. М. Саммерс рассматривают

проблему строгой самосборки дискретных треугольников Серпинского [4]. Ж. Тайнзер показывает большое разнообразие решений для генерации треугольники Серпинского [5]. К. Гдавиес представил использование комбинированный метод нахождения корня в генерации фрактальных узоров [6].

2. Результаты и обсуждение

2.1 Многоугольник

Скрипт написан на языке Python и использует библиотеки Numpy, Pygame. Скрипт использует несколько алгоритмов для рисования многоугольника и фракталов.

Рисования многоугольник используется формула (рис 1) [7].

$$x_i = x_C + R\cos\left(\phi_0 + \frac{2\pi i}{n}\right)$$
$$y_i = y_C + R\sin\left(\phi_0 + \frac{2\pi i}{n}\right)$$

Рисунок 1. Формула декартовы координаты вершин правильного n-угольника, где: x_i, y_i — координаты вершины n-угольника, x_c, y_c — центральная точка n-угольника, R — радиус n-угольника, φ_0 — угол, i — инкремент, n - n-угольник

Для рисования n-угольника используется цикл (рис 1) для рисования вершины xi, yi.

Для рисования (рис 3) применяется цикл для создание уменьшенного копия n-угольника, используется формула (рис 2). n-угольник создается копия само себя уменьшенным, радиус, полученный из предыдущий n-угольника умножается из формула (рис 2).

$$r = R\cos\frac{\pi}{n},$$

Рисунок 2. Радиус вписанной окружности в n-угольник

Для примера взято радиус 400 (таблица 1) используется формула (рис 2).

Таблица 1. Вычисление радиус уменьшенного копия n-угольник

n-угольник	j (кол-во копия n-угольника)	Радиус R
	0	400
3	1	200
	2	100
	3	50

4	1	282.842712474619
	2	200
	3	141.42135623730954
5	1	323.60679774997897
	2	261.8033988749895
	3	211.80339887498948

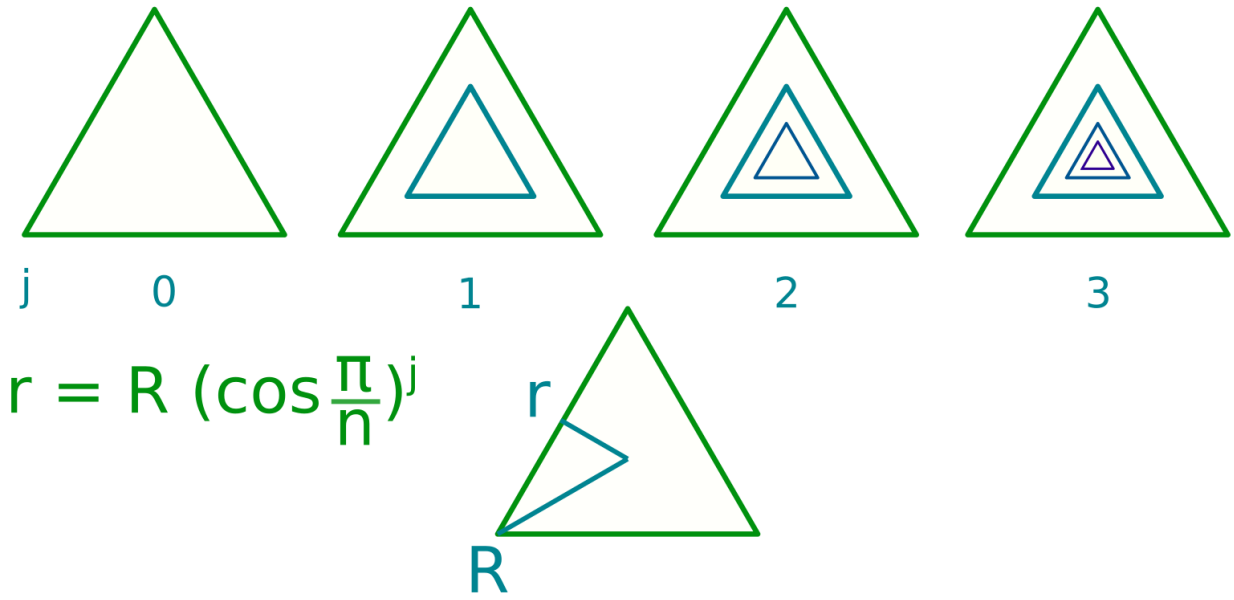

Рисунок 3. Схема уменьшения копии многоугольника, для примера используется треугольник, j — глубина копия многоугольника, m — максимальная глубина копия многоугольника

Для поворота многоугольника используется формула (рис 4) [8]. Однако формула считает сумма все углов многоугольника, чтобы найти один угол разделить на n (рис 5).

$$180^\circ \cdot (n-2)$$

Рисунок 4. Теорема о сумме углов многоугольника [8]

$$\alpha = \frac{(n-2)}{n} \cdot 18$$

Рисунок 5. Каждый угол вершины правильный многоугольника [9]

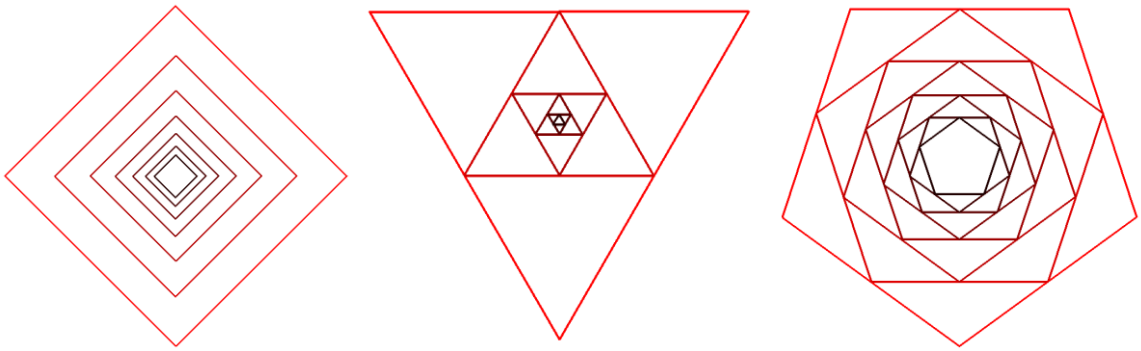

Рисунок 6. Нечетные n-угольники повернуты пересекая вершины на ребро, четные n-угольники не пересекаются

При повороте многоугольника по формуле (рис 5)[9], в результате получается что четные n-угольников вершины не пересекаются в соседний копия многоугольника (рис 6). При использовании формулы (рис 7) нечетные и четные n-угольники вершины пересекаются в соседний n-угольника (рис 8).

$$\alpha = \frac{(n-1)}{n}1\dot{8}0 - \frac{(n-2)}{n}1\dot{8}0 \Rightarrow \frac{(n-1)-(n-2)}{n}1\dot{8}0 \Rightarrow \frac{180}{n}$$

Рисунок 7. Формула углов для пересечения n-угольника копией n-угольника

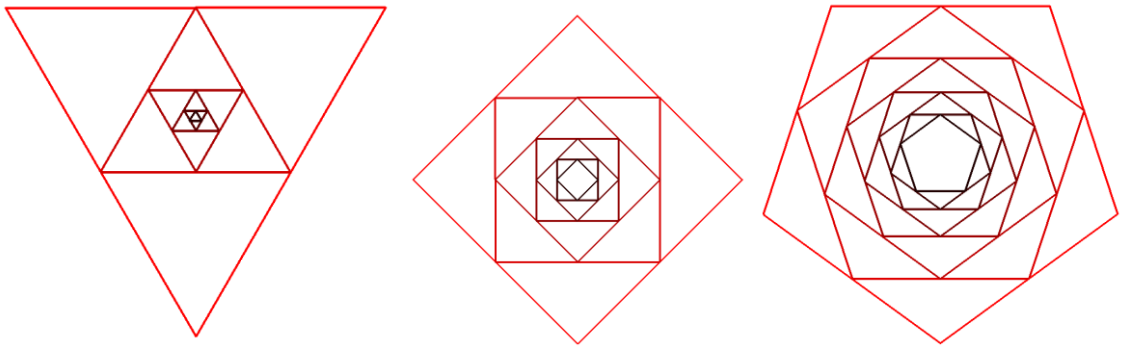

Рисунок 8. Нечетные и четные n-угольники повернуты пересекая вершины на ребро

Таблица 2. Расчеты углов для поворота n-угольника

n	Угол (рис 5) n - 2	Угол (рис 5) n - 1	Угол (рис 7)
3	60	120	60
4	90	135	45
5	108	144	36
6	120	150	30
7	128.57142857142858	154.28571428571428	25.71428571428571
8	135	157.5	22.5

9	140	160	20
10	144	162	18
11	147.27	163.64	16.36
12	150	165	15

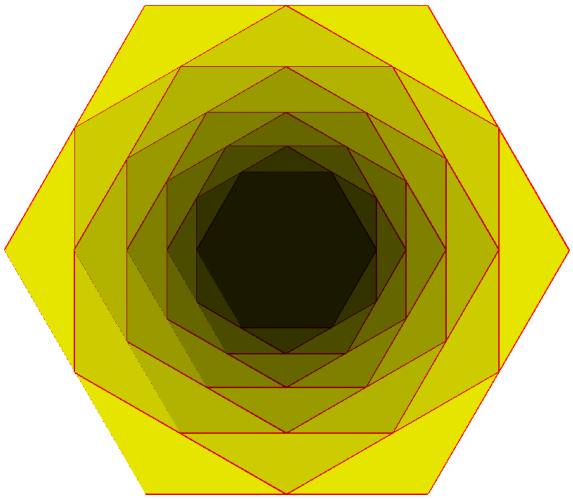

Рисунок 9. Множественный шестиугольник

Для поворачивания многоугольника по формуле (рис 7) умножается на j.

2.2 Треугольник Серпинского

В данной статье для построение фракталов используется итеративный метод, в скрипте определено функция draw_fract(p, dep = 3), где p — 3 вершины треугольника, dep — глубина рекурсия.

Алгоритм:

1. Если глубина меньше 0 то возвращаем функции
2. Итерация от 0 до количество p
3. $d_i = (p_{i-1} + p_i) / 2$
4. Конец итерация
5. draw_fract(p0, d0, d1, dep - 1)
6. draw_fract(p1, d1, d2, dep - 1)
7. draw_fract(p2, d0, d2, dep - 1)
8. Рисуем по линии на d

Для примера glob_dep равен 6.

Листинг 1. Рисования треугольник Серпинского.

```
1  def draw_fract(p, dep = 6):
2          if dep < 0:
3                  return
4          d = []
5          for i in range(0, len(p)):
6                  d += [(p[i-1] + p[i]) / 2]
```

```
7           draw_fract([p[0], d[0], d[1]], dep-1)
8           draw_fract([p[1], d[1], d[2]], dep-1)
9           draw_fract([p[2], d[0], d[2]], dep-1)
10          gdw.lines(screen, conv_hsv_rgb(0, 1.0, 1.0) * (glob_dep-dep) / glob_dep) * 255,
True, d, 2)
```

В результате получаем треугольник Серпинского (рис 10).

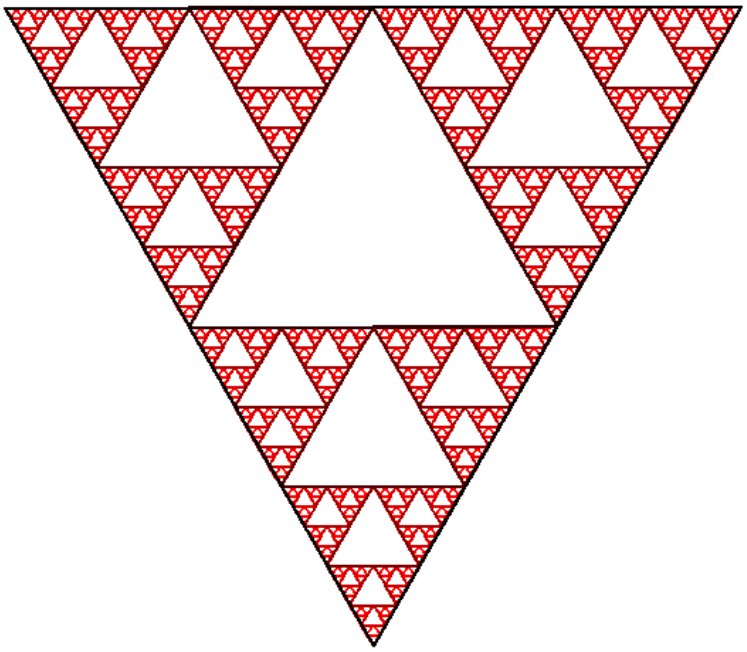

Рисунок 10. Треугольник Серпинского

Покажем получение рис 11. В многоугольник и уменьшенный копия его многоугольников видны по краям треугольников (рис. 9), каждый сторону рисуется треугольники Серпинского (Листинг 2).

Листинг 2. Многоугольник из треугольник Серпинского.

```
1   uv = screen_size / 2
2   c = np.array([0, 0])
3   pd = []
4   d = []
5   dd = []
6   s = 500
7   n = int(max(3, 16 * mv[1]))
8   m = 8
9   angle = 0
10  for j in range(0, m):
11          o = np.array([0, 0])
12          pd = d
13          d = []
14          if j > 0:
15                  s *= np.cos(np.pi / n)
16          angle += 180 / n
17          for i in range(0, n):
```

```
                o = uv + draw_pol(s, angle * np.pi / 180 + 90 * np.pi / 180, n, i)
                d += [o]
        if j > 0:
                for i in range(0, n):
                        dd = [pd[i], d[i], d[i-1]]
                        draw_fract(dd, 3)
        gdw.lines(screen, conv_hsv_rgb(60, 1.0, 1.0) * 255, True, d, 2)
```

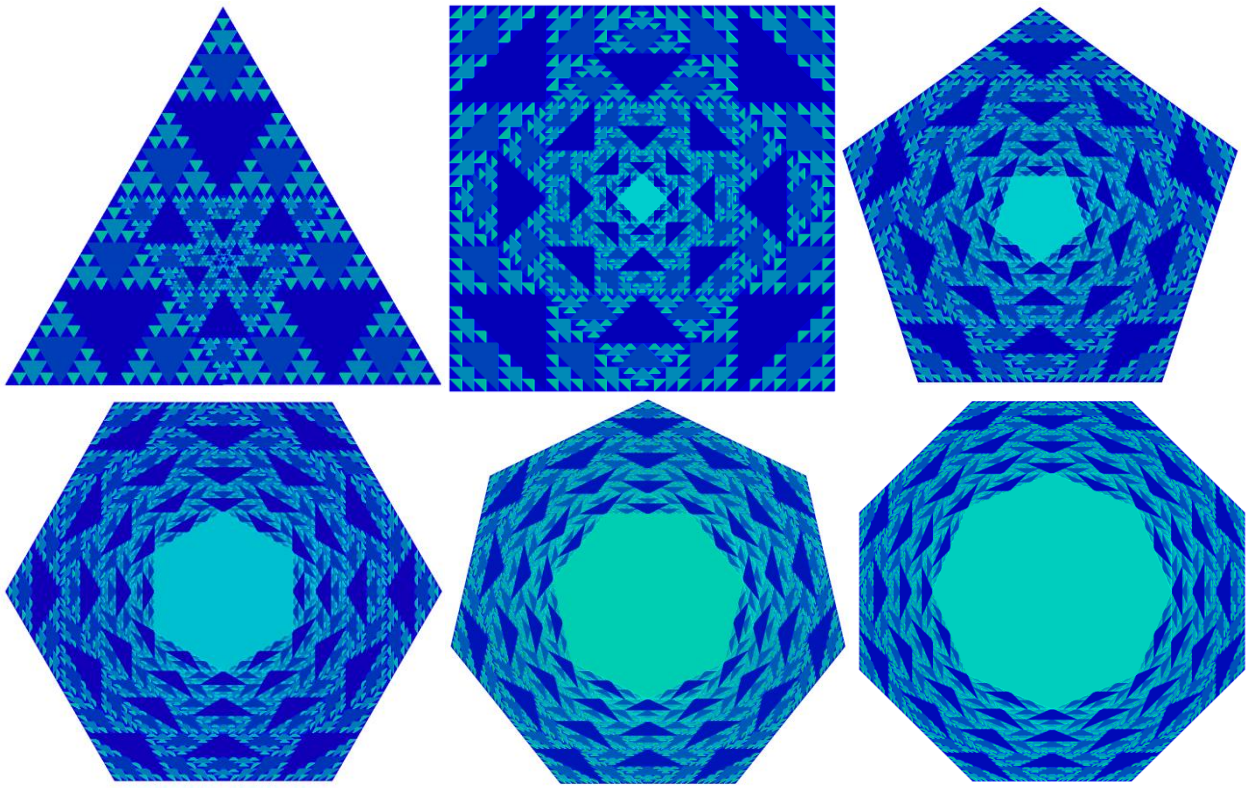

Рисунок 11. Правильный многоугольник на треугольник Серпинского

Листинг 3. Исходный код

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
import pygame
import pygame.locals as lgm
import numpy as np
screen_size = (np.array([1600, 1200]) / 1.0).astype(np.int)
mv = [0, 0]
glob_dep = 3
def conv_hsv_rgb(h, s, v):
        c = v * s
        if h < 0:
                return np.array([0, 0, 0])
        h = h / 60
        x = c * (1 - np.abs(h % 2 - 1))
        if 0 <= h <= 1:
                return np.array([c, x, 0])
        elif 1 < h <= 2:
                return np.array([x, c, 0])
        elif 2 < h <= 3:
                return np.array([0, c, x])
        elif 3 < h <= 4:
                return np.array([0, x, c])
        elif 4 < h <= 5:
                return np.array([x, 0, c])
        elif 5 < h <= 6:
                return np.array([c, 0, x])
        return np.array([0, 0, 0])

def draw_pol(r, deg, n, i):
```

```
30          return r * np.array([np.cos(deg + 2 * np.pi * i / n), np.sin(deg + 2 * np.pi * i / n)])
31
32  def draw_fract(p, dep = 3):
33          if dep < 0:
34                  return
35          d = []
36          for i in range(0, len(p)):
37                  d += [(p[i-1] + p[i]) / 2]
38          draw_fract([p[0], d[0], d[1]], dep-1)
39          draw_fract([p[1], d[1], d[2]], dep-1)
40          draw_fract([p[2], d[0], d[2]], dep-1)
41          col = -mv[0] * 30 + 180 + 70 * dep / glob_dep
42          gdw.lines(screen, conv_hsv_rgb(col, 1.0, 1.0) * 255, True, d, 2)
43          gdw.polygon(screen, conv_hsv_rgb(col, 0.9, 0.8) * 255, d)
44
45  if __name__ == '__main__':
46          pygame.init()
47          screen = pygame.display.set_mode(screen_size)
48          clock = pygame.time.Clock()
49          gdw = pygame.draw
50          pygame.display.set_caption("Draw")
51          dcycles = False
52          screen.fill(np.array([0.9, 0.9, 0.9, 1.0]) * 255)
53          uv = screen_size / 2
54          c = np.array([0, 0])
55          pd = []
56          d = []
57          dd = []
58          while dcycles==False:
59                  mouse = pygame.mouse.get_pos()
60                  mv[0] = mouse[0] / screen_size[0]
61                  mv[1] = mouse[1] / screen_size[1]
62                  for event in pygame.event.get():
63                          pygame.display.update()
64                          if event.type == lgm.QUIT or (event.type == lgm.KEYDOWN and event.key == lgm.K_ESCAPE):
65                                  dcycles = True
66                  pygame.display.flip()
67                  screen.fill(np.array([1.0, 1.0, 1.0, 0.5]) * 255)
68                  s = 550
69                  n = int(max(3, 16 * mv[1]))
70                  m = 8
71                  angle = 0
72                  for j in range(0, m):
73                          o = np.array([0, 0])
74                          pd = d
75                          d = []
76                          if j > 0:
77                                  s *= np.cos(np.pi / n)
78                          angle += 180 / n
79                          for i in range(0, n):
80                                  o = uv + draw_pol(s, angle * np.pi / 180 + 90 * np.pi / 180, n, i)
81                                  d += [o]
82                          if j > 0:
83                                  for i in range(0, n):
84                                          dd = [pd[i], d[i], d[i-1]]
85                                          draw_fract(dd, glob_dep)
86                          col = -mv[0] * 30 + 180 + 70 * (m-j) / m
87                          gdw.lines(screen, conv_hsv_rgb(col, 1.0, 1.0) * 255, True, d, 2)
88                          gdw.polygon(screen, conv_hsv_rgb(col, 0.9, 0.9) * 255, d)
89                  clock.tick(60)
90          pygame.quit()
```

3 Выводы

В данной статье были рассмотрены правильный многоугольник и фрактал «треугольник Серпинского», и разработан скрипт, который их рисует.

Библиографический список

1. Khovanova T., Nie E., Puranik A. The Pioneering Role of the Sierpinski Gasket

//The Best Writing on Mathematics 2016. Princeton University Press, 2017. С. 140-148.
2. Conversano E., Lalli L. T. Sierpinski triangles in stone on medieval floors in Rome //J. Appl. Math. 2011. Т. 4. С. 114-122.
3. Brunori P., Magrone P., Lalli L. T. Imperial Porphiry and Golden Leaf: Sierpinski Triangle in a Medieval Roman Cloister //International Conference on Geometry and Graphics. Springer, Cham, 2018. С. 595-609.
4. Lathrop J. I., Lutz J. H., Summers S. M. Strict self-assembly of discrete Sierpinski triangles //Theoretical Computer Science. 2009. Т. 410. №. 4-5. С. 384-405.
5. Taentzer G. et al. Generation of Sierpinski triangles: A case study for graph transformation tools //International Symposium on Applications of Graph Transformations with Industrial Relevance. Springer, Berlin, Heidelberg, 2007. С. 514-539.
6. Gdawiec K. Fractal patterns from the dynamics of combined polynomial root finding methods //Nonlinear Dynamics. 2017. Т. 90. №. 4. С. 2457-2479.
7. Правильный многоугольник. // Википедия URL: https://ru.wikipedia.org/wiki/Правильный_многоугольник (дата обращения: 2021-08-25).
8. Теорема о сумме углов многоугольника. // Википедия URL: https://ru.wikipedia.org/wiki/Теорема_о_сумме_углов_многоугольника (дата обращения: 2021-08-25).
9. Правильный пятиугольник. // Википедия URL: https://ru.wikipedia.org/wiki/Правильный_пятиугольник (дата обращения: 2021-08-25).