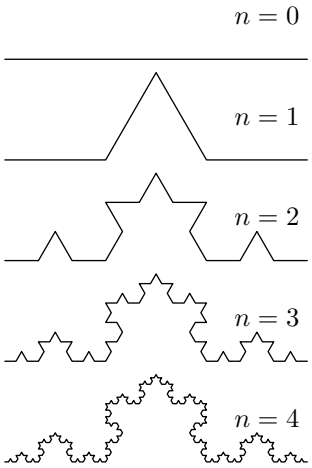
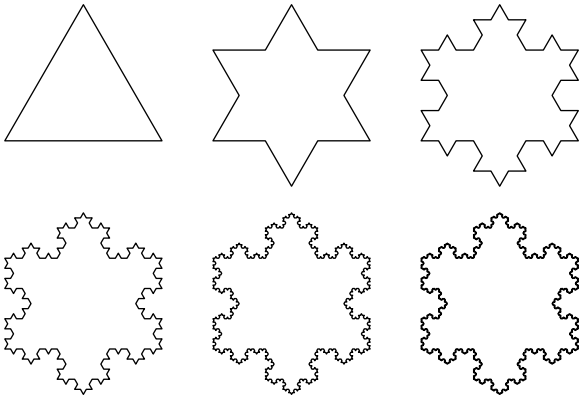
Использование рекурсии для рисования фракталов

При помощи рекурсивных функций можно рисовать более интересные объекты, например фракталы. Начнём с классического фрактала — кривой Коха.

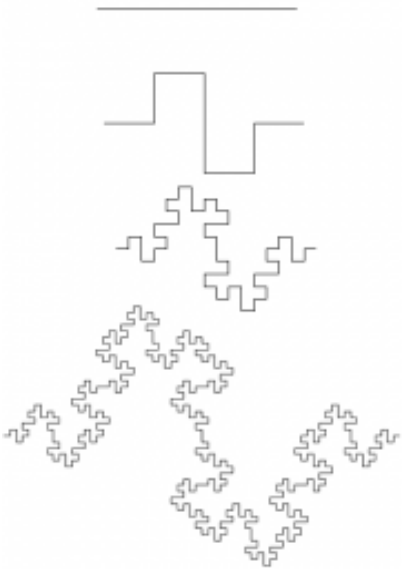
A. Кривая Коха нулевого порядка — это отрезок. Для построения кривой Коха порядка $n+1$ мы разбиваем каждый отрезок кривой Коха порядка n на три равные части и средний сегмент заменяем равносторонним треугольником без этого сегмента.



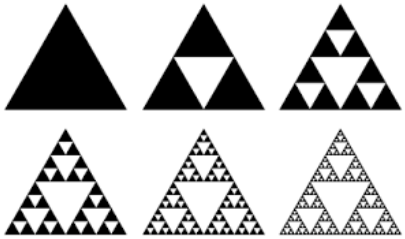
B. Похожим образом определяется т.н. снежинка Коха. Это равносторонний "треугольник", каждая сторона которого представляет собой кривую Коха одного и того же порядка.



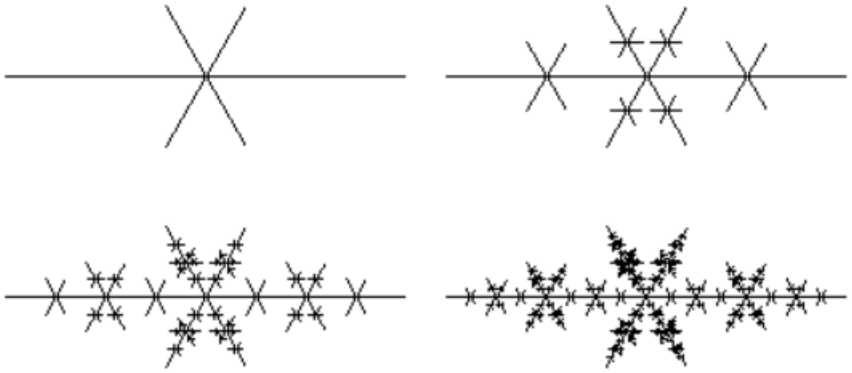
C. Ещё один классический фрактал — кривая Минковского.



D. Треугольник Серпинского — последовательное “выкидывание” центрального треугольника. Двумерный аналог множества Кантора был предложен польским математиком Вацлавом Серпинским в 1915 году.



E. Ледяной фрактал.
Здесь можно поэкспериментировать с размерами “усов”, их формой (углом) и стороной, куда они растут.



Используя функцию, строящую такой отрезок (растущий в одну или обе стороны), напишите фукнцию, строящую правильный N-многоугольник, каждая сторона которого будет представлять собой такой отрезок.

Справочник по модулю `turtle`

Прежде всего, ссылка на стандартную документацию.

Основное

- `T.forward(distance)` — проползти вперёд на distance пикселей
- `T.right(angle)` — повернуть направо (по часовой стрелке) на угол `angle` градусов
- `T.left(angle)` — повернуть налево (против часовой стрелки) на угол `angle` градусов
- `T.pendown()` — опустить перо (начать рисование)
- `T.penup()` — поднять перо (закончить рисование)
- `T.goto(x, y)` — переместить черепашку в точку с координатами (x, y)

Движение и направление

- `T.backward(distance)` — проползти назад на `distance` пикселей
- `T.setx(x)` — установить `x` координату черепашки
- `T.sety(y)` — установить `y` координату черепашки
- `T.setheading(to_angle)` — повернуть черепашку под углом `to_angle` к вертикали (0 — наверх, 90 — направо)
- `T.home()` — вернуть черепашку домой, в точку с координатами $(0, 0)$
- `T.dot(size, color)` — нарисовать точку диаметра `size` цвета `color`. Параметр `color` необязателен
- `T.undo()` — откатить предыдущее действие черепашки

Рисование

- `T.pensize(width)` — установить диаметр пера в `width`
- `T.pencolor(colorstring)` — установить цвет линии, которая рисует черепашка (например, `'brown'` или `'#32c18f'`)
- `T.fillcolor(colorstring)` — установить цвет заполнения
- `T.begin_fill()` — начать следить за черепашкой для заполнения области
- `T.end_fill()` — заполнить цветом `fillcolor` область, пройденную черепашкой начиная с `begin_fill`
- `T.showturtle()` — показать черепашку
- `T.hideturtle()` — спрятать черепашку
- `T.write(text)` — вывести текст `text`

Скорость

- `T.speed(speed)` — установить скорость черепашки. Значение `speed` должно быть от 1 (медленно) до 10 (быстро), или 0 (мгновенно)
- `T.getscreen().tracer(n)` — отрисовывать лишь каждый n-й кадр. Почти в n раз ускоряет рисование.

Информация о черепашке

- `T.position()` — получить текущие координаты черепашки
- `T.towards(x, y)` — получить угол между текущим направление черепашки и прямой от черепашки к точке (x, y)
- `T.xcor()` — получить x координату черепашки
- `T.ycor()` — получить y координату черепашки
- `T.heading()` — получить текущий угол к вертикали
- `T.distance(x, y)` — получить расстояние до точки (x, y)
- `T.isdown()` — узнать, рисует ли сейчас черепашка (опущено ли перо)
- `T.isvisible()` — узнать, видима ли сейчас черепашка

Пример программы

```
from turtle import *    # Подключаем модуль turtle
T = Turtle()            # T - это наша черепашка. Можно создавать много черепашек
T.pendown()             # Опускаем перо (начало рисования)
T.forward(50)           # Проползти 50 пикселей вперёд
T.left(90)              # Поворот влево на 90 градусов
T.forward(50)           # Рисуем вторую сторону квадрата
T.left(90)              # Поворот влево на 90 градусов
T.forward(50)           # Рисуем третью сторону квадрата
T.left(90)              # Поворот влево на 90 градусов
T.forward(50)           # Рисуем четвертую сторону квадрата
T.penup()               # Поднять перо (закончить рисовать)
T.forward(100)          # Отвести черепашку от рисунка в сторону
T.hideturtle()          # Спрятать черепашку
mainloop()              # Задержать окно на экране
```

Или то же самое, но с функцией и без намёка на использование нескольких черепашек

```
from turtle import *

def draw():
    pendown()
    forward(50)
    left(90)
    forward(50)
    left(90)
    forward(50)
    left(90)
    forward(50)
    penup()
    forward(100)
    hideturtle()

draw()
hideturtle()
mainloop()
```