Лабораторная работа. Рисование.

Рассмотрим, как можно организовать рисование простых фигур и графиков на форме на примере построения самоподобных регулярных кривых. Фрактал — это фигура, состоящая из частей, которые в каком-то смысле подобны целому (текое неформальное определение дано фракталу основоположником фрактальной геометрии Бенуа Мандельбротом). В своей книге «Фрактальная геометрия природы» он приводит множество примеров «самоподобных» объектов в природе — русла рек, береговые линии, ветви деревьев, кровеносная система человека, морские волны... Фрактальные построения актуальны, например, в компьютерной графике для создания реалистичных пейзажей. Для описания фракталов существуют различные методики, например, L-системы (системы Линденмайера), применение систем итерирующих функций (IFS). L-система - это грамматика некоторого языка (достаточно простого), которая описывает инициатор и преобразование, выполняемое над ним. Система итерирующих функций - это совокупность сжимающих аффинных преобразований. Как известно, аффинные преобразования включают в себя масштабирование, поворот и параллельный перенос. Афинное преобразование считается сжимающим, если коэффициент масштабирования меньше единицы.

Ознакомимся с основными приемами рисования в приложении с графическим интерфейсом на примере кривой Коха, салфетки Серпинского и драконовой ломаной (дракон Хартера-Хейтуэя).

Построение триадной кривой Коха в режиме L-системы начинается с отрезка единичной длины (рисунок 1) – это 0-е поколение кривой Коха. Далее каждое звено (в нулевом поколении один отрезок) заменяется на образующий элемент, обозначенный на рисунке 1 через n = 1. В результате такой замены получается следующее поколение кривой Коха. В 1-ом поколении – это кривая из четырех прямолинейных звеньев, каждое длиной по 1/3. Для получения 3-го поколения проделываются те же действия - каждое звено заменяется на уменьшенный образующий элемент. Итак, для получения каждого последующего поколения, все звенья предыдущего поколения необходимо заменить уменьшенным образующим элементом. Кривая n-го поколения при любом конечном n называется предфракталом. На рисунке.1 представлены пять поколений кривой. При n стремящемся к бесконечности кривая Коха становится фрактальным обьектом.

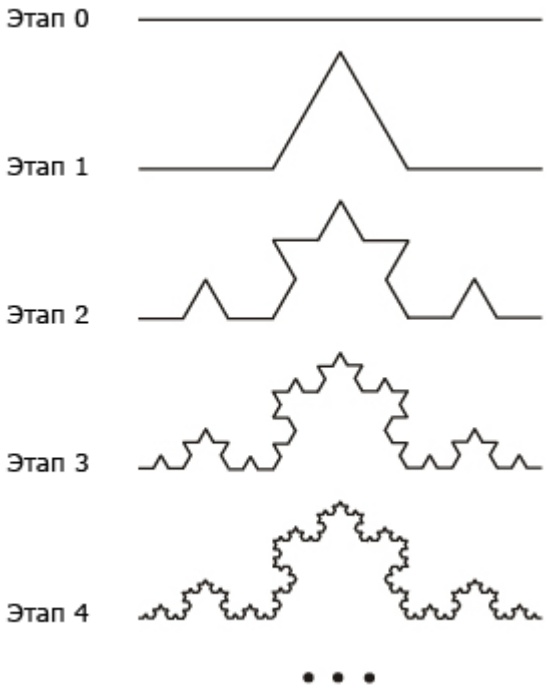


Рис. 1. Кривая Коха

Салфетка Серпинского нулевого порядка состоит из закрашенного равностороннего треугольника (цвет закраски на картинке — черный). Салфетка первого порядка получится, если разделить треугольник на четыре равные части и «извлечь» центральную (покрасить центральную часть белым, снять черную закраску). Если для оставшихся закрашенных частей треугольника повторить процедуру, получится салфетка второго порядка, затем третьего и т.д.

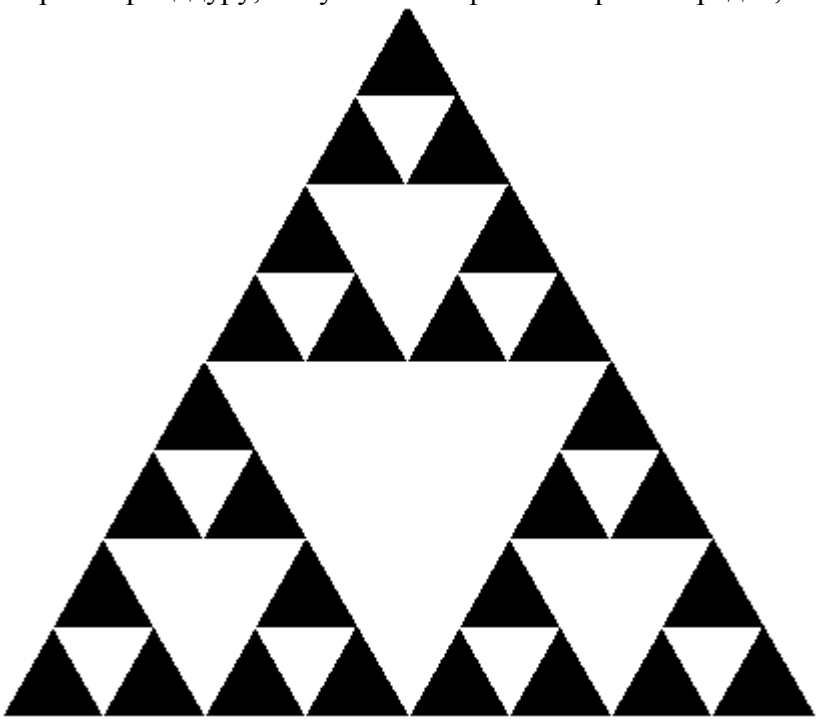


Рис. 2. Салфетка Серпинского 3 порядка

Драконова ломаная получится, если полоску бумаги согнуть пополам K раз (сгиб всегда производится в одну и ту же сторону), а потом разогнуть так, чтобы каждый угол был равен 90 градусам. Предельная фрактальная кривая (при K стремящемся к бесконечности) называется драконом Хартера-Хейтуэя.

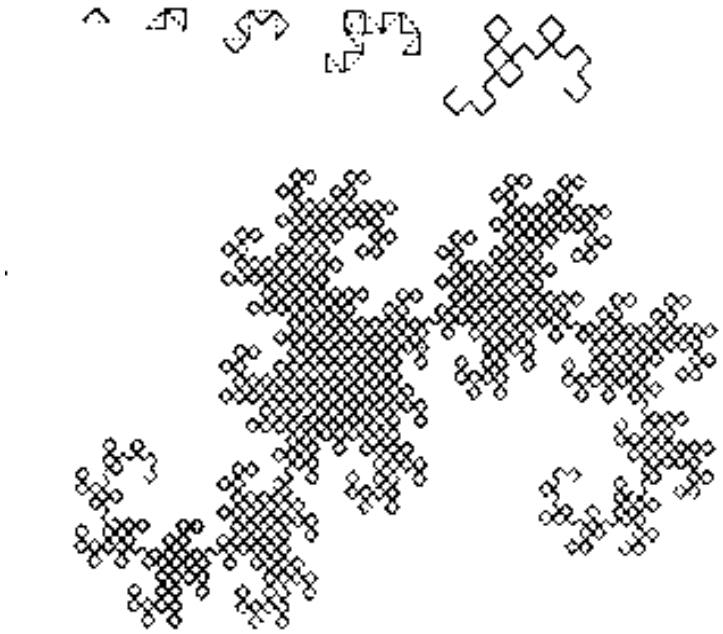


Рис. 3. Драконова ломаная (K=1, 2, 3, 4, 5, 11).

Перейдем теперь непосредственно к разработке приложения для рисования данных фрактальных кривых. Создадим новый python-файл, в котором импортируем tkinter, создадим корневое окно и два фрейма, у которых родительским элементом назначим корневое окно.

```
from tkinter import *
root = Tk() # явно создать корневое окно

pict=Frame(root) # здесь будет канва и рисунок
manage=Frame(root) # здесь будут управляющие элементы

pict.pack(side=LEFT) # разместим фреймы в окне
manage.pack(side=RIGHT)
```

Теперь выделим место под рисунок: организуем канву и зальем ее белым прямоугольником. Канву будем создавать во фрейме pict.

```
canvas=Canvas(pict, width=400, height=400)
canvas.create_rectangle(0,0, 400,
                        400, outline='#fff', fill = '#fff')
canvas.pack(fill=BOTH, expand=1)
```

Добавим элементы управления: радиокнопки для выбора вида фрактала, кнопку вызова диалогового окна для выбора цвета кривой, две шкалы для установки толщины линии и порядка фрактала, а также кнопки для рисования графика и стирания графика. Т.е. в итоге должно получиться что-то похожее:

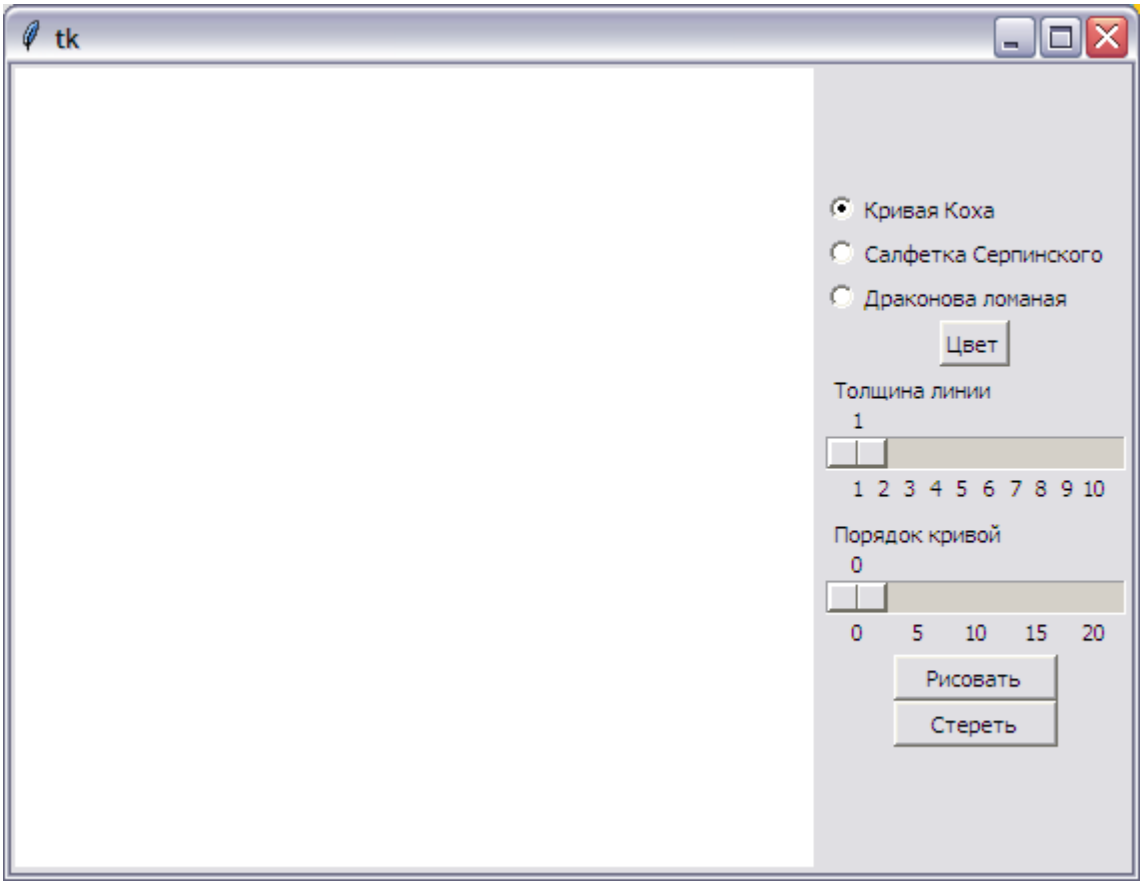


Рис. 4. Внешний вид окна программы

Итак, поместим в код программы следующие элементы:

```
rdVar=IntVar() # ассоциированная переменная для радиокнопок
rdVar.set(0) # по умолчанию выбрана будет первая радиокнопка

# сами радиокнопки, родительский элемент — фрейм!!!
rad0 = Radiobutton(manage,text="Кривая Коха",
          variable=rdVar,value=0)
rad1 = Radiobutton(manage,text="Салфетка Серпинского",
          variable=rdVar,value=1)
rad2 = Radiobutton(manage,text="Драконова ломаная",
          variable=rdVar,value=2)

rad0.pack(side=TOP, anchor=W) # экспериментируем с положением радиобаттонов
rad1.pack(side=TOP, anchor=W)
rad2.pack(side=TOP, anchor=W)

# чтобы можно было управлять цветом кривой, заведем переменную и запишем туда
# сначала черный цвет
myColor="#000"

#теперь, чтобы можно было вызывать диалоговое окно выбора цвета, сделам импорт
from tkinter.colorchooser import askcolor

#и заведем обработчик события, где будет вызываться диалоговое окно для цвета
def setColor(event):
    global myColor #тут уточняем, что работать будем с вышезаведенной переменной
    (RGB, myColor)=askcolor() #запоминаем результат выбора цвета

# а тут заводим кнопку, щелчок по которой вызовет диалог для выбора цвета
# родительский элемент — фрейм
# назначим обработчик и разместим кнопку
butColor=Button(manage, text="Цвет")
butColor.bind('<Button-1>', setColor)
butColor.pack()

#теперь введем 2 шкалы и определим ассоциированные переменные
myPenWidth=IntVar() # для толщины линии фрактала
myPenWidth.set(1)  # установим ее равной 1 по умолчанию
penWidth = Scale(manage, label="Толщина линии", orient=HORIZONTAL, length=150,
             from_=1, to=10, tickinterval=1, resolution=1, variable=myPenWidth)

# и ассоциированная переменная для второй шкалы — порядок фрактала
myCurvePower=IntVar()
myCurvePower.set(0) # по умолчанию это 0
curvePower = Scale(manage, label="Порядок кривой", orient=HORIZONTAL, length=150,
             from_=0, to=20, tickinterval=5, resolution=1, variable=myCurvePower)

# разместим шкалы
penWidth.pack()
curvePower.pack()

#Теперь поставим кнопки для прорисовки фрактала и очищения области рисования
#Введем функцию-обработчик события для рисования кривой
# В зависимости от значения выбранной радиокнопки (а точнее, от значения
# ассоциированной переменной) будем рисовать фрактал
def draw(event):
    if (rdVar.get()==0): # кривая Коха
        # рекурсия для Коха, напишем позже
    elif (rdVar.get()==1):
        #рекурсия для Серпинского, напишем позже
    else:
        #рекурсия для драконовой ломаной, напишем позже
```

```
# Теперь создадим и разместим кнопки для рисования
butDraw=Button(manage, text="Рисовать", width=12)
butDraw.bind("<Button-1>", draw)
butDraw.pack()

#А это примитивный обработчи события «очистить область рисования»
#мы просто закрашиваем все белым прямоугольником. В Python можно и иначе,
#как именно — предлагаю выяснить и реализовать самостоятельно в качестве упражнения
def clear(event):
    canvas.create_rectangle(0, 0, canvas.winfo_width(), canvas.winfo_height(),
                            outline='#fff', fill='#fff')

butClear=Button(manage, text="Стереть", width=12)
butClear.bind("<Button-1>", clear)
butClear.pack()

# не забываем запустить сценарий корневого окна
root.mainloop()
```

Теперь, если запустить проект на выполнение, вы увидите приведенную на рисунке выше форму. Уже работают кнопки вызова диалогового окна выбора цвета, кнопка очистки формы, радиобаттоны и ползунки. Давайте запрограммируем функцию draw, чтобы можно было рисовать фракталы.

Рассмотрим подробнее построение рекурсивных функций для рисования кривой Коха, салфетки Серпинского и драконовой ломаной.
Чтобы построить кривую Коха, надо научиться рисовать ее генератор (см. рис. 5) для любых пар точек (X1, Y1) и (X2, Y2).

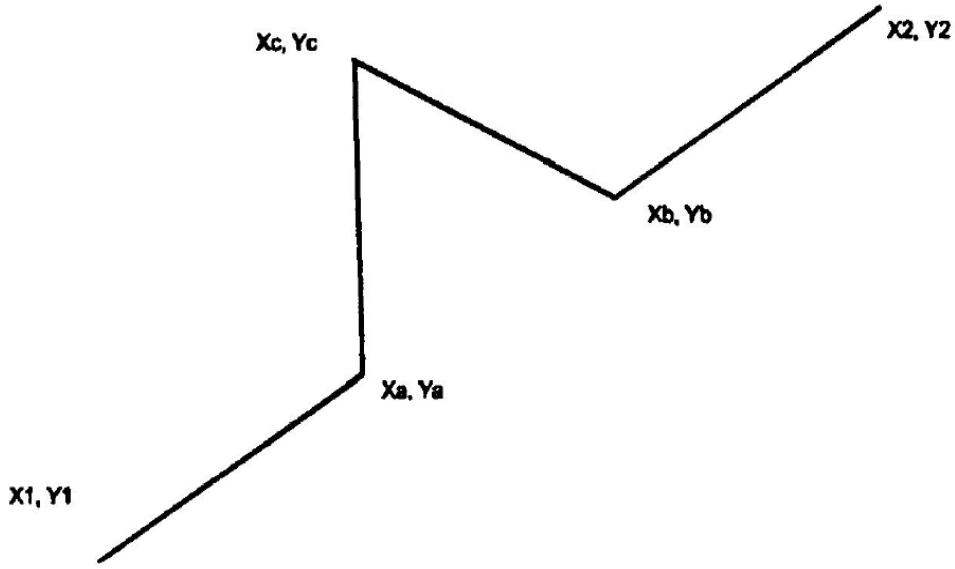


Рис. 5. Генератор кривой Коха

Точки (Xa, Ya), (Xb, Yb) и (Xc, Yc) определяются формулами:

$$\alpha = atan2(Y2-Y1,\ X2-X1)$$

$$R = \sqrt{(X2-X1)^2+(Y2-Y1)^2}$$

$$Xa = X1 + \frac{R}{3}\cos\alpha \quad , \quad Ya = Y1 + \frac{R}{3}\sin\alpha$$

$$Xb = X1 + \frac{2R}{3}\cos\alpha \quad , \quad Yb = Y1 + \frac{2R}{3}\sin\alpha$$

$$Xc = Xa + \frac{R}{3}\cos(\alpha - \frac{\pi}{3}) \quad , \quad Yc = Ya + \frac{R}{3}\sin(\alpha - \frac{\pi}{3})$$

Для справки: функция atan2(y, x) возвращает угол между осью абсцисс и отрезком, соединяющим начало координат с точкой (x, y).

На псведокоде алгоритм построения кривой Коха порядка order на отрезке (X1, Y1, X2, Y2) очень прост:

ЕСЛИ order=0 , построить отрезок (X1, Y1, X2, Y2)
ИНАЧЕ
построить кривую Коха порядка order-1 на отрезке (X1, Y1, Xa, Ya)
построить кривую Коха порядка order-1 на отрезке (Xa, Ya, Xc, Yc)
построить кривую Коха порядка order-1 на отрезке (Xc, Yc, Xb, Yb)
построить кривую Коха порядка order-1 на отрезке (Xb, Yb, X2, Y2)

Реализуем алгоритм в программе. Введем рекурсивную функцию Koch:

```
import math
def Koch(order, x1, y1, x2, y2):
    if (order==0):
        canvas.create_line(x1,y1,x2,y2,fill=myColor, width=myPenWidth.get())
    else:
        alpha=math.atan2(y2-y1, x2-x1)
        R=math.sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))

        # вычислим xA, yA, xB, yB, xC, yC
        xA=x1+(R/3)*math.cos(alpha)
        yA=y1+(R/3)*math.sin(alpha)
        xC=xA+R*math.cos(alpha-math.pi/3)/3
        yC=yA+R*math.sin(alpha-math.pi/3)/3
        xB=x1+2*R*math.cos(alpha)/3
        yB=y1+2*R*math.sin(alpha)/3

        #рекурсивные вызовы
        Koch(order-1, x1, y1, xA, yA)
        Koch(order-1, xA, yA, xC, yC)
        Koch(order-1, xC, yC, xB, yB)
        Koch(order-1, xB, yB, x2, y2)
```

Разумеется, описание функции для рисования кривой Коха надо делать до того, как мы описываем функцию-обработчик draw. Теперь модифицируем функцию draw:

```
def draw(event):
    if (rdVar.get()==0): # кривая Коха
        x1=0
        y1=canvas.winfo_height()
        x2=canvas.winfo_width()
        y2=0
        # рекурсия для Коха
        Koch(myCurvePower.get(),x1, y1, x2, y2)
    elif (rdVar.get()==1):
        #рекурсия для Серпинского, напишем позже
        Sierpinski(myCurvePower.get(),0, canvas.winfo_height()-20,
                                         canvas.winfo_width()-10)
    else:
```

```
#рекурсия для драконовой ломаной
```

Запустите программу и попробуйте нарисовать различными цветами кривые нулевого, первого, второго, третьего, четвертого и пятого порядков.

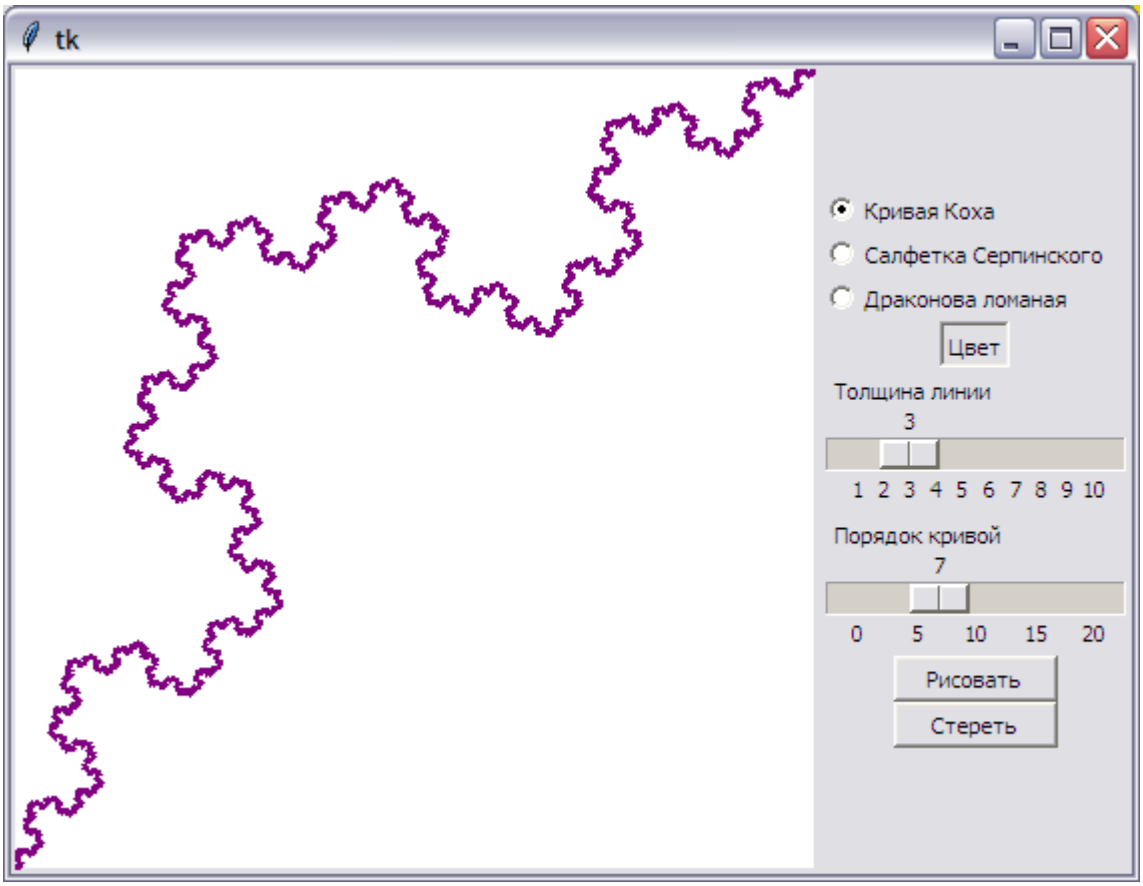


Рис. 6. Кривая Коха 6 порядка

Для прорисовки салфетки Серпинского надо научиться рисовать заполненные цветом треугольники. Фактически, построив начальный цветной треугольник достаточно из него каждый раз «выбрасывать» (зарисовывая цветом фона) центральный треугольник.

Точкой отсчета рисования треугольника будет точка A с координатами (x, y), длина стороны треугольника на рисунке 7 обозначена как len.

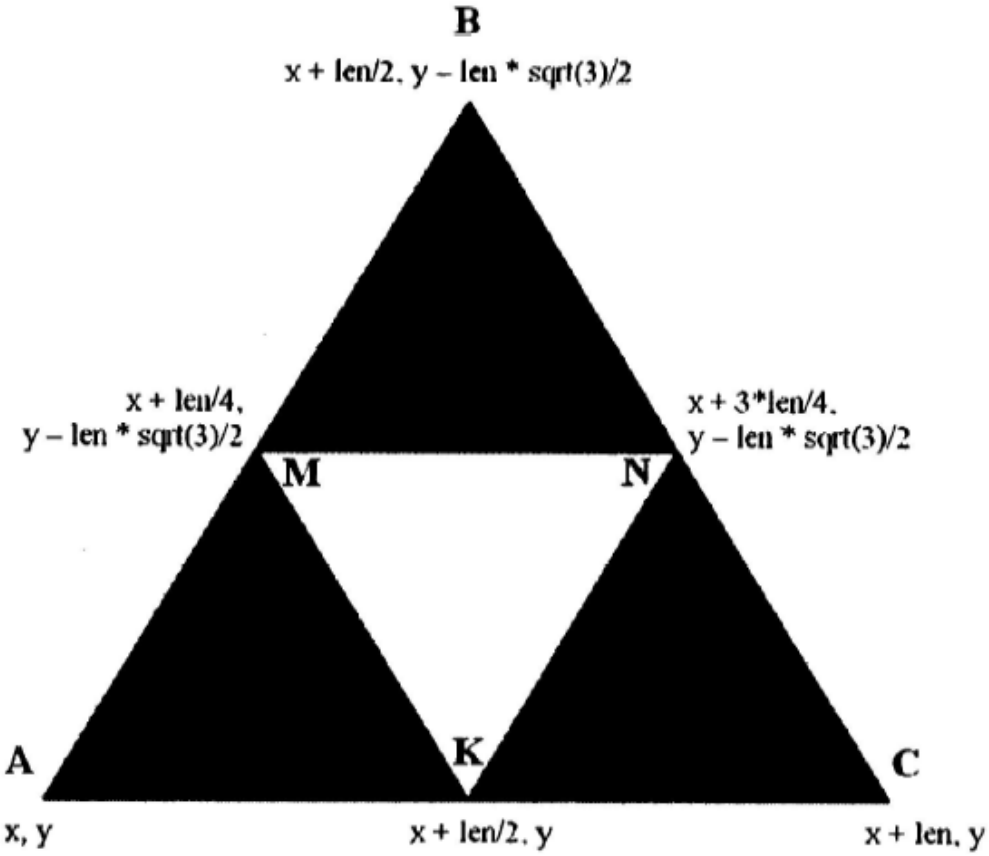


Рис. 7. Базовый элемент салфетки Серпинского

В псевдокоде процедура прорисовки салфетки выглядит просто:

построить «черный» треугольник ABC
ЕСЛИ order больше нуля
построить белый треугольник MNK
построить треугольник Серпинского порядка order-1 со стороной len/2 в тчк A, M, K

Добавим рекурсивную функцию для рисования салфетки Серпинского.

```
def Sierpinski(order, x, y, length):
    s3d2=math.sqrt(3)/2
    #строим цветной треугольник ABC
    #вычисляем координаты вершин треугольника
    points=[x, y, x+length/2, y-length*s3d2, x+length, y]
    canvas.create_polygon(points, outline=myColor, fill=myColor,
                          width=myPenWidth.get())
    #теперь будем «выбрасывать» средние треугольники
    if (order>0):
        #рисуем треугольник MNK цветом фона
        points=[x+length/4, y-length*s3d2/2, x+3*length/4, y-length*s3d2/2,
                x+length/2, y]
        canvas.create_polygon(points, outline='#fff', fill='#fff',
                              width=myPenWidth.get())
        #рекурсивно вызываем фунцкию
        Sierpinski(order-1, x, y, length/2); # в т.A
        Sierpinski(order-1, x+length/2, y, length/2); # в т.K
        Sierpinski(order-1, x+length/4, y-length*s3d2/2, length/2); # в т.M
```

Теперь в обработчике события в функции draw надо вызвать эту рекурсивную функцию, например, так:

```
Sierpinski(myCurvePower.get(),0, canvas.winfo_height()-20, canvas.winfo_width()-10)
```

Попробуйте построить салфетку Серпинского разных порядков.

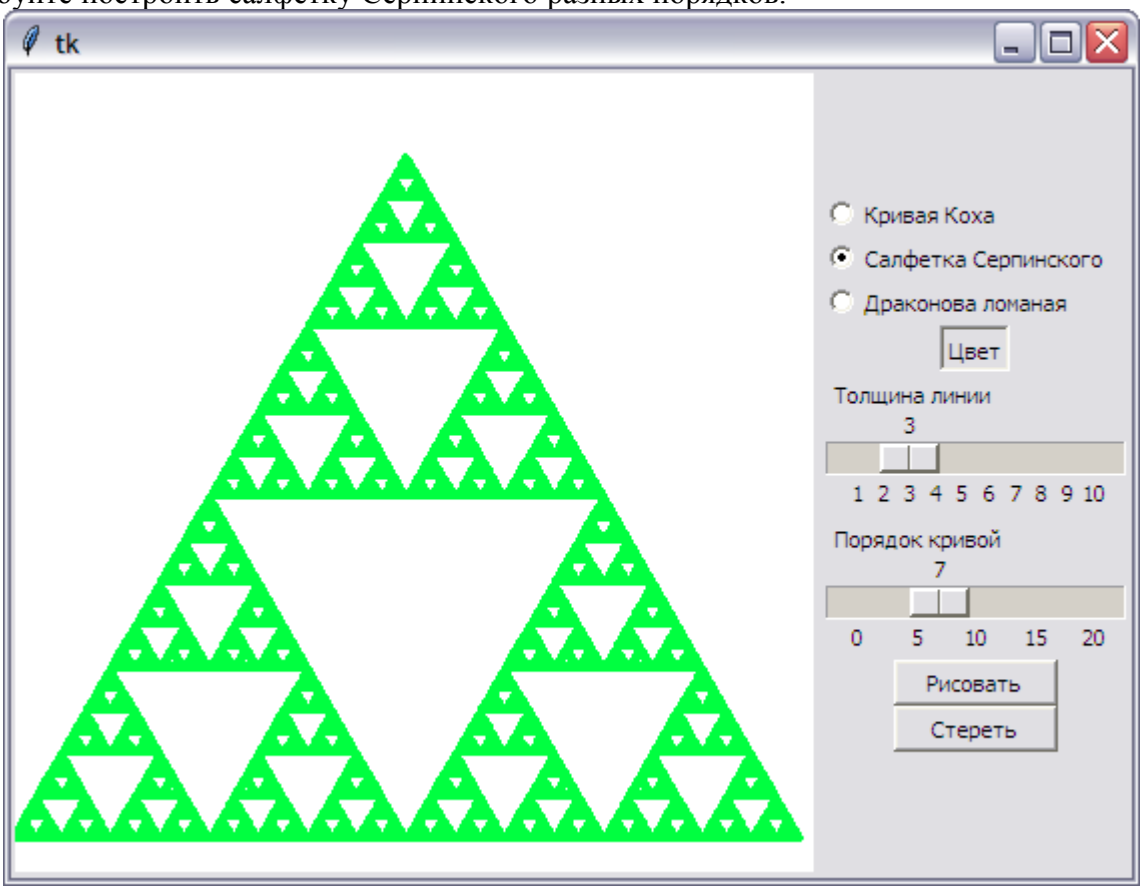


Рис. 8. Салфетка Серпинского 7 порядка.

Черчение драконовой ломаной — задача более сложная. Рассмотрим, как ломаная N-ного порядка строится на основе ломаной порядка N-1.

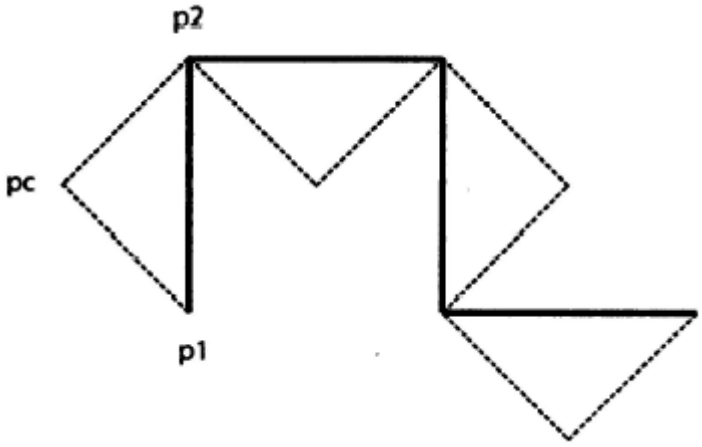


Рис. 9. Построение драконовой ломаной.

При увеличении порядка ломаной каждый ее отрезок заменяется двумя отрезками, расположенными под прямым углом. Первая пара отрезков находится слева от сегмента

ломаной предыдущего порядка, вторая пара — справа, третья — снова слева, четвертая справа и т.д. Построение ломаной можно свести к получению массива точек ломаной N-ного порядка из массива точек ломаной N-1 порядка. Т.е., как показано на рис. 9, зная пару точек (p1, p2) можно вычислить координаты точки pc для следующего порядка, не забывая о чередовании положении точки pc слева-справа от отрезка ломаной предыдущего порядка. Полученный в итоге массив точек можно просто соединить линиями, чтобы получился дракон.

Опишем рекурсивную функцию, которая формирует список точек нашего дракона.

```
def getDragonPoints(order):
    x=canvas.winfo_width()/5
    y=canvas.winfo_height()/2
    if (order==0):
        # ломаная нулевого порядка состоит из одного сегмента
        res = []
        res.append(x)
        res.append(y+x/2)
        res.append(canvas.winfo_width()-x)
        res.append(y+x/2)
        return res
    prevRes=getDragonPoints(order-1)
    res=[]
    # направление: 1 - влево, -1 - вправо
    DirSign=1
    # начальная точка ломаной не изменяется
    res.append(prevRes[0])
    res.append(prevRes[1])
    for  i in range(0,len(prevRes)-3,2):
        # считаем очередной сегмент ломаной
        p1x=prevRes[i]
        p1y=prevRes[i+1]
        p2x=prevRes[i+2]
        p2y=prevRes[i+3]
        alpha = math.atan2(p2y - p1y, p2x - p1x)-DirSign*math.pi/4
        R = math.sqrt(((p1x - p2x) * (p1x - p2x) +
                       (p1y - p2y) * (p1y - p2y))/2)
        # найдем новую точку ломаной
        pcx=p1x+R*math.cos(alpha)
        pcy=p1y+R*math.sin(alpha)
        # добавляем ее и конечную точку в список точек ломаной
        res.append(pcx)
        res.append(pcy)
        res.append(p2x)
        res.append(p2y)
        # меняем направление
        DirSign *=-1
    return res
```

Т.к. в списке храним поочередно координаты x и y для каждой точки, обратите внимание на обход списка точек в цикле for.

Теперь в обработчкие события щелчок по кнопке «Рисовать» во фрагменте кода, ответственном за рисование дракона, надо получить вектор точек драконовой ломаной и потом между каждой парой соседних точек нарисовать линии. Например, так:

```
points=getDragonPoints(myCurvePower.get())
canvas.create_line(points, fill=myColor, width=myPenWidth.get())
```

Запустите программу на выполнение и постройте драконов разного порядка. Драконы высоких порядков (12<=order) выглядят уже очень красиво:

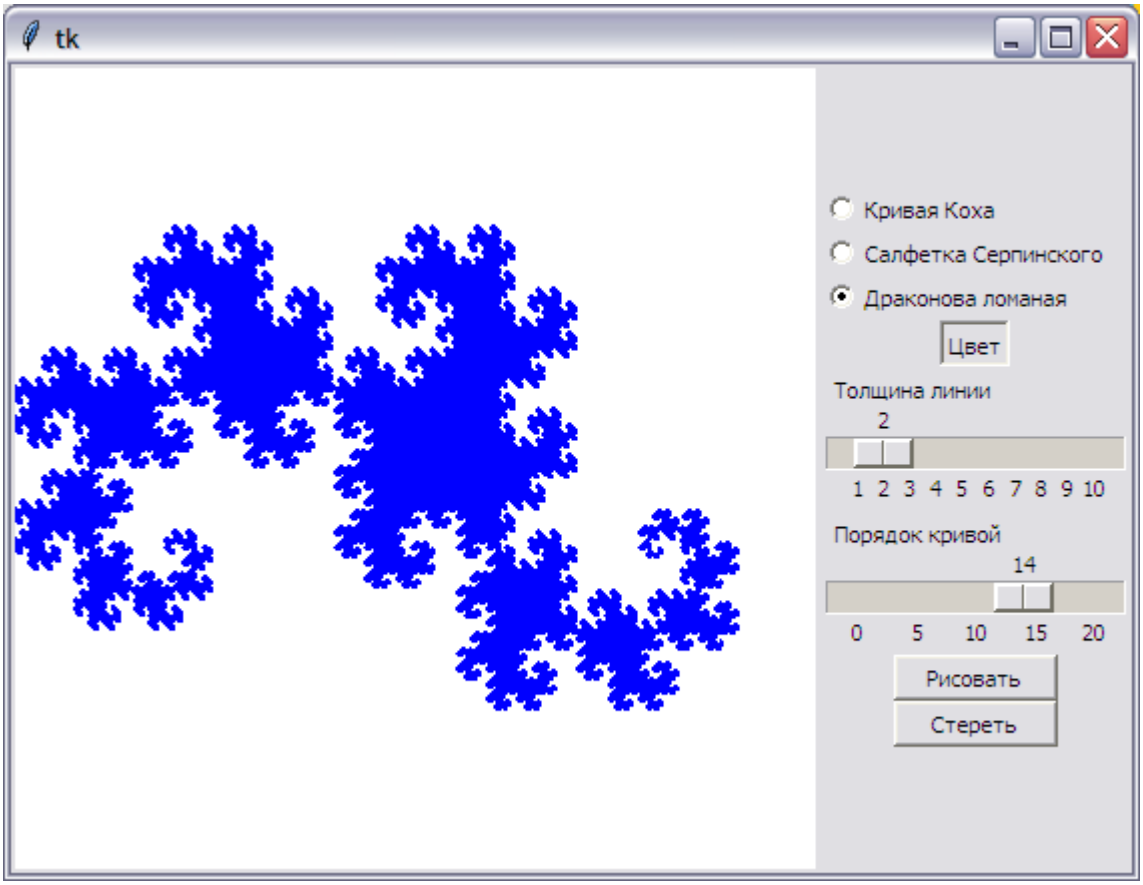


Рис. 10. Драконова ломаная 14 порядка

Задания.

Часть 1. Работа с заготовкой проекта. (до 5 баллов)

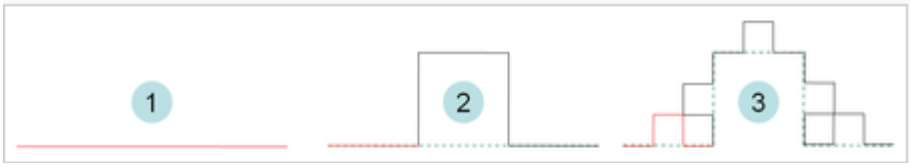
1. Доделайте программу, чтобы в ней можно было строить не только кривую Коха, но и салфетку Серпинского и драконову ломаную. (1 балл)
2. Измените проект так, чтобы вместо кривой Коха строилась снежинка Коха. (2 балла)
3. Модифицируйте программу так, чтобы порядок кривой вводился не при помощи шкалы, а, например, из текстового поля. Дайте пользователю возможность задавать начальные и конечные координаты кривых (либо, координату точки и длину стороны для салфетки Серпинского). (2 балла)

Часть 2.1 Индивидуальное задание. (до 5 баллов)

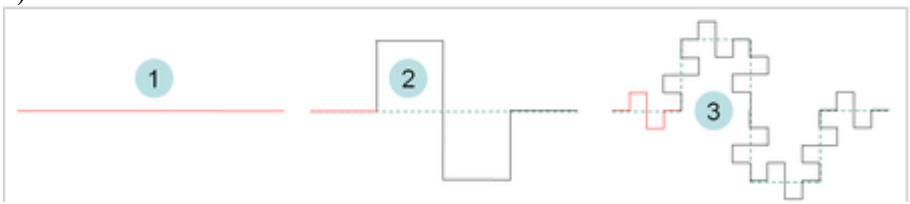
Дополните имеющийся проект (например, добавьте еще одну радиокнопку) рисованием самоподобной кривой в соответствии с вариантом:

1. Канторово множество. Для канторова множества порядка K надо зарисовать все предыдущие K отрезков, чтобы наглядно проиллюстрировать процесс построение множества.
2. Фрактал Чезаро (разновидность кривой Коха, но угол между отрезком (Xa, Ya, Xc, Yc) и основанием не 60 градусов, а больше (но не превышает 90)).

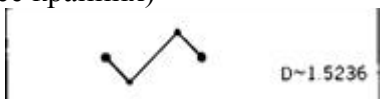
3. Квадратичная кривая Коха (вид 1 — на рисунке показаны основа и генератор, первые 2 итерации)



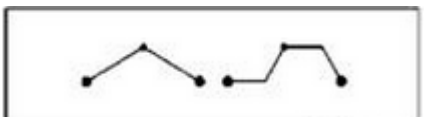
4. Квадратичная кривая Коха (вид 2 — на рисунке показаны основа и генератор, первые 2 итерации)



5. Ковер Серпинского.
6. Кривая Леви.
7. Кривая Гильберта.
8. Кривая Мура (Moore curve).
9. Кривая Госпера.
10. H-дерево.
11. Пифагорово дерево.
12. Фрактал Вишека (Vicsek fractal)
13. Т-квадрат
14. Шкура двойного дракона (вариант кривой Коха с генератором, показанном на рисунке, средняя линяя вдвое длиннее крайних)



15. «Перекошенная снежинка» - вариант драконовой ломаной, но угол между отрезками не 90 градусов, а 120.



Вопросы.

1. Что такое фрактал? Что такое самоподобная кривая?
2. Что такое генератор самоподобной кривой? Поясните на примере построения любой самоподобной кривой.
3. Какие способы компоновки виджетов вы знаете? Как компоновать компоненты при помощи фреймов? Что такое позиционирование виджетов при помощи якоря?
4. Какие виджеты и методы виджетов используются для рисования?
5. Как вызвать диалоговое окно выбора цвета? Как задать пользовательский цвет?
6. Поясните принципы построение рекурсивных функций для прорисовки самоподобных кривых (Коха, салфетки Серпинского, драконовой ломаной, кривой по своему варианту).
7. Для чего используюся ассоциированные переменные? Предложите вариант использования ассоциированной переменной для выбора цвета.