|  |
| --- |
| МУНИЦИПАЛЬНОЕ АВТОНОМНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  «ЛИЦЕЙ ИННОВАЦИОННЫХ ТЕХНОЛОГИЙ» |
| **Н.Н. Гончаренко, Е.В. Редько**  **Фракталы на языке Python:**  **учебное пособие** |
| Фракталы: красота математики |
| Хабаровск  2022 |

|  |
| --- |
| **Фракталы на языке Python**: учебное пособие / Н.Н. Гончаренко, Е.В. Редько. – Хабаровск, МАОУ «Лицей инновационных технологий». – 2022. – 40 с. |
| Настоящее пособие может быть использовано педагогами, реализующими дополнительные общеобразовательные программы естественнонаучной направленности, учащимися, учителями информатики, студентами-бакалаврами педагогических направлений подготовки.  Пособие состоит из двух частей. Первая часть содержит теоретический материал по фрактальной графике и базовым принципам создания графических примитивов на языке Python.  Вторая часть представляет собой сборник практических работ для школьников с примерами построения геометрических фракталов и заданиями для самостоятельной работы. Выполнение практических работ предполагает владение базовыми навыками программирования на языке Python. Дополнительные средства рисования (инструменты библиотеки Tkinter, «черепашья» графика) описаны непосредственно в пособии в объеме, достаточном для выполнения заданий по построению фракталов. |
| © МАОУ ЛИТ, 2022 |

# Содержание

[Введение 4](#_Toc100272981)

[1. Основные понятия теории фракталов 5](#_Toc100272982)

[1.1 История формирования понятия «фрактал» 5](#_Toc100272983)

[1.2 Виды фракталов 9](#_Toc100272984)

[1.3 Применение фрактальной геометрии в современном мире 13](#_Toc100272985)

[2. Рисование на языке Python 17](#_Toc100272986)

[2.1 Графическая библиотека Tkinter 17](#_Toc100272987)

[2.2 Исполнитель «Черепашка» 28](#_Toc100272988)

[3. Практические работы 29](#_Toc100272989)

[Практическая работа № 1. Построение фрактала «Снежинка Коха» с использованием графических примитивов Tkinter 29](#_Toc100272990)

[Практическая работа № 2. Построение фрактала «Снежинка Коха» с помощью черепашьей графики 36](#_Toc100272991)

[Практическая работа № 3. Построение фрактала «Треугольник Серпинского» с использованием графических примитивов Tkinter 39](#_Toc100272992)

[Задания для самостоятельной работы 43](#_Toc100272993)

[Ответы к заданиям на построение фракталов 49](#_Toc100272994)

[Список источников 52](#_Toc100272995)

# Введение

Цель обучения информатике в школе – подготовка учеников к работе в информационном обществе. Для этого необходимо хорошо владеть разными типами прикладных программ, уметь самостоятельно осваивать программные продукты, выбирать адекватный инструмент под задачу, развивать системный подход, алгоритмическое мышление. Интерпретируемый язык программирования Python идеально подходит для этих целей.

Язык Python – язык высокого уровня общего назначения, допускающий возможность создания настоящих, графически оформленных, работоспособных программ. Изучение Python развивает математическую интуицию и геометрические представления, формирует алгоритмический, структурный, логический и комбинаторный типы мышления, повышает творческую активность и самостоятельность школьников.

Фрактальная графика – быстро развивающийся и перспективный вид компьютерной графики. Понятия фрактал, фрактальная графика, фрактальная геометрия появились в конце 70-х годов XX века. Фрактал – структура, состоящая из самоподобных частей. Фракталы используются для оформления полиграфической продукции, при создании игр.

Обучение школьников приемам построения компьютерной графики имеет не только прикладное значение, так как способствует развитию алгоритмического мышления. При этом построение фракталов затрагивает такую тему, как рекурсия, и требует применения специальных математических или даже геометрических знаний.

В пособии рассмотрены принципы построения фракталов с использованием графических примитивов языка Python, а также исполнителя Turtle, разработаны задания для проверки усвоения материала.

Учебное пособие «Фракталы на языке Python» актуально для использования учителями информатики в рамках проведения уроков по программам углубленного изучения информатики, а также при организации занятий внеурочной деятельности.

# 1. Основные понятия теории фракталов

## 1.1 История формирования понятия «фрактал»

Математики пренебрегли вызовом и предпочли бежать от природы путём изобретения всевозможных теорий, которые никак не объясняют того, что мы видим или ощущаем.

**Бенуа Мандельброт**

Язык науки стремительно меняется в современном мире. Уже изучено огромное количество разнообразных явлений природы, открыты фундаментальные законы физики, объясняющие различные экспериментальные факты. Каждый раз, сталкиваясь с новыми природными объектами, ученые вводят в язык науки новые термины и понятия. Математики при описании всего богатства и удивительной красоты окружающего нас мира подчас мыслят образами, используя эстетические категории.

До недавнего времени геометрические модели различных природных конструкций традиционно строились на основе сравнительно простых фигур: прямых, многоугольников, окружностей, многогранников, сфер. Однако, очевидно, что этот классический набор становится плохо применимым для характеристики многих сложных объектов (рис. 1):

* очертание береговых линий материков,
* поле скоростей в турбулентном потоке жидкости,
* разряд молнии в воздухе,
* пористые материалы,
* форма облаков, снежинки,
* пламя костра,
* контуры дерева,
* кровеносно-сосудистая система человека,
* поверхность клеточной мембраны и др.



Рисунок . Фрактальная геометрия в природе (река с притоками)

В последние 15-20 лет для описания перечисленных и подобных им образований ученые всё чаще используют новые геометрические понятия.

Одним из таких понятий, изменившим многие традиционные представления о геометрии, явилось понятие фрактала.

Изучение фракталов на рубеже XIX и XX веков носило скорее эпизодический, нежели систематический характер, потому что раньше математики в основном изучали «хорошие» объекты, которые поддавались исследованию при помощи общих методов и теорий.

Первые идеи фрактальной геометрии возникли в 19 веке. Кантор с помощью простой рекурсивной (повторяющейся) процедуры превратил линию в набор несвязанных точек (так называемая Пыль Кантора). Он брал линию и удалял центральную треть и после этого повторял то же самое с оставшимися отрезками (рис. 2).

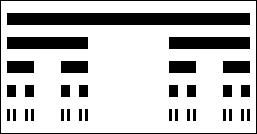


Рисунок . Канторово множество

В 1872 году немецкий математик Карл Вейерштрасс построил пример непрерывной функции, которая нигде не дифференцируема, то есть не имеет касательной ни в одной своей точке. Однако его построение было целиком абстрактно и трудно для восприятия. Поэтому в 1904 году швед Хельге фон Кох придумал такую непрерывную кривую, которая нигде не имеет касательной, причем ее довольно просто нарисовать. Оказалось, что она обладает свойствами фрактала. Один из вариантов этой кривой носит название «снежинка Коха».

Идеи самоподобия фигур подхватил француз Поль Пьер Леви, будущий наставник Бенуа Мандельброта. В 1938 году вышла его статья «Плоские и пространственные кривые и поверхности, состоящие из частей, подобных целому», в которой описан еще один фрактал – С-кривая Леви.

В 1918 году была опубликована работа Жулиа, посвященная итерациям комплексных рациональных функций, в которой описаны множества Жулиа – целое семейство фракталов, близко связанных с множеством Мандельброта. Этот труд был удостоен приза Французской академии, однако в нем не содержалось ни одной иллюстрации, так что оценить красоту открытых объектов было невозможно

Термин «фрактал» был введен в обращение французским математиком польского происхождения Бенуа Мандельбротом в 1975 году. Важную роль в широком распространении идей фрактальной геометрии сыграла замечательная книга Б. Мандельброта «Фрактальная геометрия природы» (рис. 3, 4).

|  |  |
| --- | --- |
| Мандельброт, Бенуа - Wikiwand  Рисунок . Бенуа Мандельброт | Фрактальная геометрия природы - Мандельброт Бенуа  Рисунок . Книга Бенуа Мандельброта «Фрактальная геометрия природы» |

Фрактальные объекты, согласно своему определению, обладают размерностью, строго превышающей топологическую размерность элементов, из которых они построены.

Основой новой геометрии является идея самоподобия. Она выражает собой тот факт, что иерархический принцип организации фрактальных структур не претерпевает значительных изменений при рассмотрении их через микроскоп с различным увеличением. В результате эти структуры на малых масштабах выглядят в среднем так же, как и на больших.

Таким образом, слово фрактал произошло от латинского *fractus* и переводится как дробный, ломаный. Фракталами называют геометрические объекты, которые удовлетворяют одному или нескольким из следующих свойств:

* обладает сложной структурой при любом увеличении;
* является самоподобной;
* обладает дробной фрактальной размерностью, которая больше топологической;
* может быть построена рекурсивными процедурами.

## 1.2 Виды фракталов

Фракталы можно разделить на несколько видов:

* конструктивные (геометрические),
* стохастические,
* динамические (алгебраические).

***Геометрические*** фракталы строятся поэтапно. Сначала изображается основа. Затем некоторые части основы заменяются фрагментом. На каждом следующем этапе части уже построенной фигуры вновь заменяются фрагментом, взятым в подходящем масштабе.

Всякий раз масштаб уменьшается. Когда изменения становятся визуально незаметными, считают, что построенная фигура хорошо приближает фрактал и дает представление о его форме. Однако на самом деле для получения фрактала нужно бесконечное число этапов. Меняя основу и фрагмент, можно получить много разных геометрических фракталов.

Примером геометрического фрактала является снежинка Коха (рис. 5).

Но кривая Коха, как бы она ни была похожа на границу берега, не может выступать в качестве ее модели из-за того, что она слишком «правильна». Все природные объекты создаются по капризу природы, в этом процессе всегда есть случайность (рис. 6).

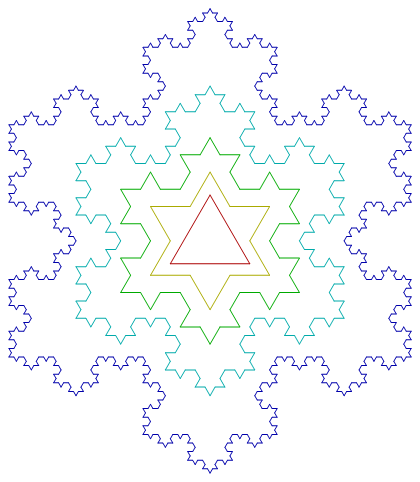


Рисунок . Снежинка Коха – первые пять этапов формирования (автор fregimus/ livejournal.com)

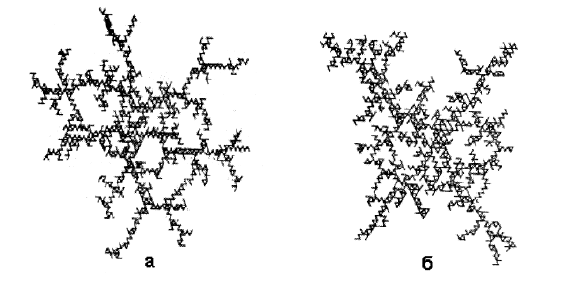


Рисунок . Фрактальность нефтегазовых пластов

Фракталы, при построении которых в итеративной системе случайным образом изменяются какие-либо параметры, называются ***стохастическими*** (рис. 7). К этому классу фракталов относится и фрактальная монотипия, или стохатипия. Термин «стохастичность» происходит от греческого слова, обозначающего «предположение».

Стохатипия (фрактальная монотипия) Л. Лившиц «Зимний пейзаж» интересна тем, что кисть художника ее не касалась (рис. 8). Она является чистым (природным) продуктом «самоорганизации» на молекулярном уровне. В отличие от нанофракталов, которые являются черно-белыми, стохатипии – цветные. Чтобы получить нанорисунок, нанохудожнику приходится наноструктуры окрашивать с помощью программ. Стохатипии относятся к фрактальному искусству (*fractal art*), а также к научному искусству.

|  |  |
| --- | --- |
| Рисунок . Пример стохастического фрактала | Рисунок . Леа-Тути Лившиц. Зимний пейзаж. Стохатипия. 1983. 39 х 60. |

Другие примеры стохастических фракталов:

* траектория броуновского движения на плоскости и в пространстве;
* граница траектории броуновского движения на плоскости;
* эволюции Шрамма-Лёвнера – случайная кривая y, которая задается уравнением Лёвнера;
* различные виды рандомизированных фракталов, то есть фракталов, полученных с помощью рекурсивной процедуры, в которую на каждом шаге введён случайный параметр.

***Динамические*** фракталы строятся по однозначному правилу , которое переводит каждую точку плоскости ровно в одну точку этой же плоскости. Начав с точки , можно построить последовательность точек:

; ; ; и так далее.

В зависимости от начальной точки получаются последовательности, которые могут вести себя по-разному (говорят, что у них разная динамика):

* сходиться к какому-либо пределу;
* зацикливаться;
* расходиться к бесконечности (точки неограниченно удаляются от начала).

Можно считать, что правило делит плоскость на несколько областей, в каждой из которых точки ведут себя одинаково – например, сходятся к одному из возможных пределов. Оказывается, что во многих случаях границы таких областей устроены очень сложно и являются фракталами.

Примером динамического фрактала является множество Мандельброта (рис. 9).

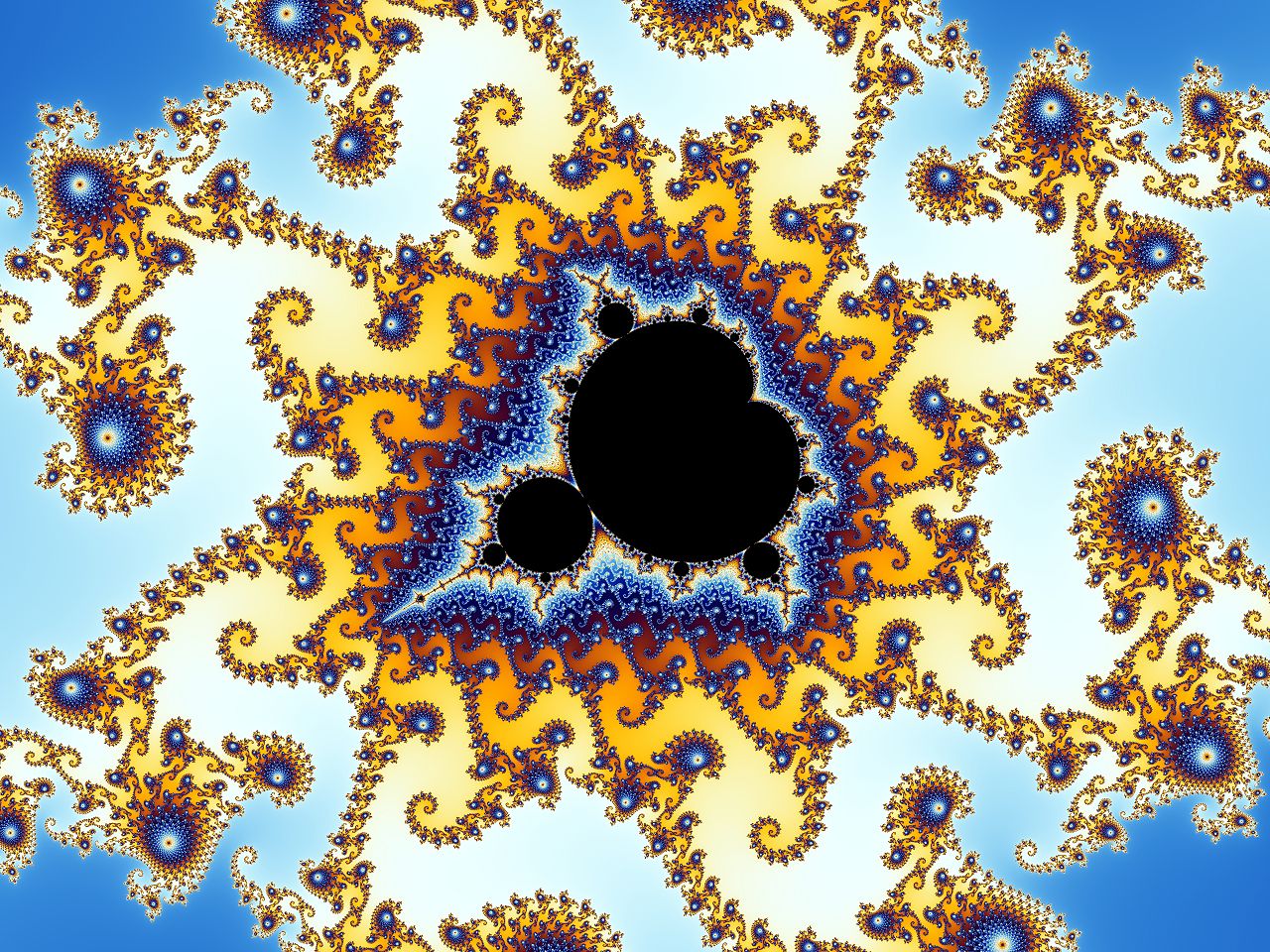


Рисунок . Множество Мандельброта

## 1.3 Применение фрактальной геометрии в современном мире

Фракталы нашли широкое применение в различных областях науки и техники. Основная причина этого заключается в том, что они описывают реальный мир иногда даже лучше, чем традиционная физика или математика. Многие объекты в природе, например, дерево, молния, береговая линия, коралл (рис. 11), горный рельеф, обычная мятая бумага, – имеют фрактальные свойства. Это используют при их компьютерном моделировании для достижения большей реалистичности. Поэтому применять фрактальные изображения можно в самых разных сферах, начиная от создания обычных текстур и фоновых изображений и заканчивая фантастическими ландшафтами для компьютерных игр или книжных иллюстраций.

Применение фрактальных правил построения широко распространено и в архитектуре (рис. 10).

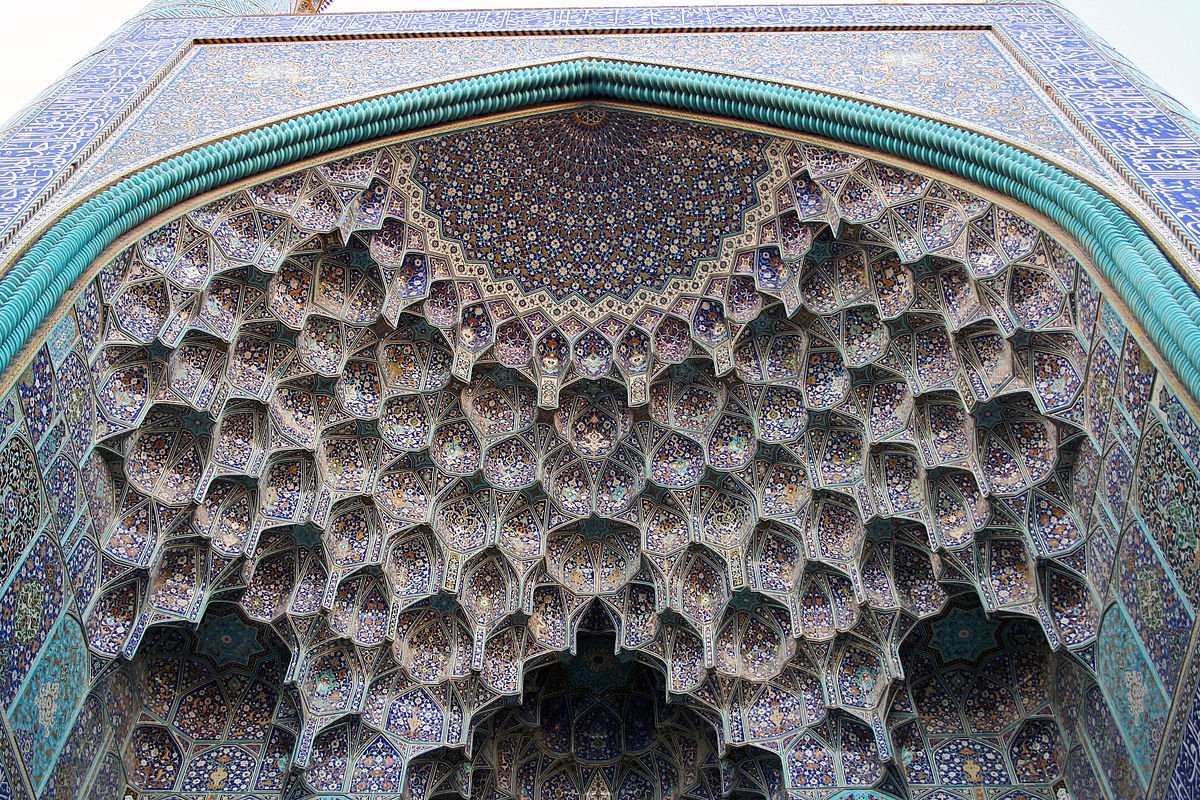


Рисунок . Самоподобные элементы в архитектуре

В физике фракталы естественным образом возникают при моделировании нелинейных процессов, таких, как турбулентное течение жидкости, сложные процессы диффузии-адсорбции, пламя, облака и т. п. Фракталы используются при моделировании пористых материалов, например, в нефтехимии. В биологии они применяются для моделирования популяций и для описания систем внутренних органов (система кровеносных сосудов).

Фракталы не только окружают нас, они отражают принципы строения различных систем организма человека, животного, растений. Бронхи и бронхиолы легкого образуют «дерево» с многочисленными разветвлениями. Количественный анализ ветвления дыхательных путей показал, что оно имеет фрактальную геометрию (рис. 12).

|  |  |
| --- | --- |
| фракталы  Рисунок . Фрактал в структуре коралла | Индивидуальный итоговый проект по теме: «Фракталы в искусстве» - PDF  Скачать Бесплатно  Рисунок . Фрактал в структуре бронхов человека |

Также, принципы фрактальной геометрии были использованы Натаном Коэном при проектировании антенных устройств. Преимуществом фрактальных антенн является их многодиапазонность и широкополосность при сравнительно меньших размерах (рис. 13).

В последнее время фракталы стали популярным инструментом у трейдеров для анализа состояния биржевых рынков. Фракталы рынка являются одним из индикаторов в торговой системе Била Вильямса. Считается, что он же впервые и ввел это название в трейдинг.

Более того, на основе фрактальных объектов программы моделируют сочинение музыки. Одну из наиболее известных подобных программ – MusiNum – разработал Ларс Киндерман. Модули программы позволяют делать выбор голосов, устанавливать темп композиции, задавать сценарий, который позволяет изменять параметры синтеза музыки в процессе исполнения композиции. Можно выбрать характер звучания, панораму и громкость голоса.

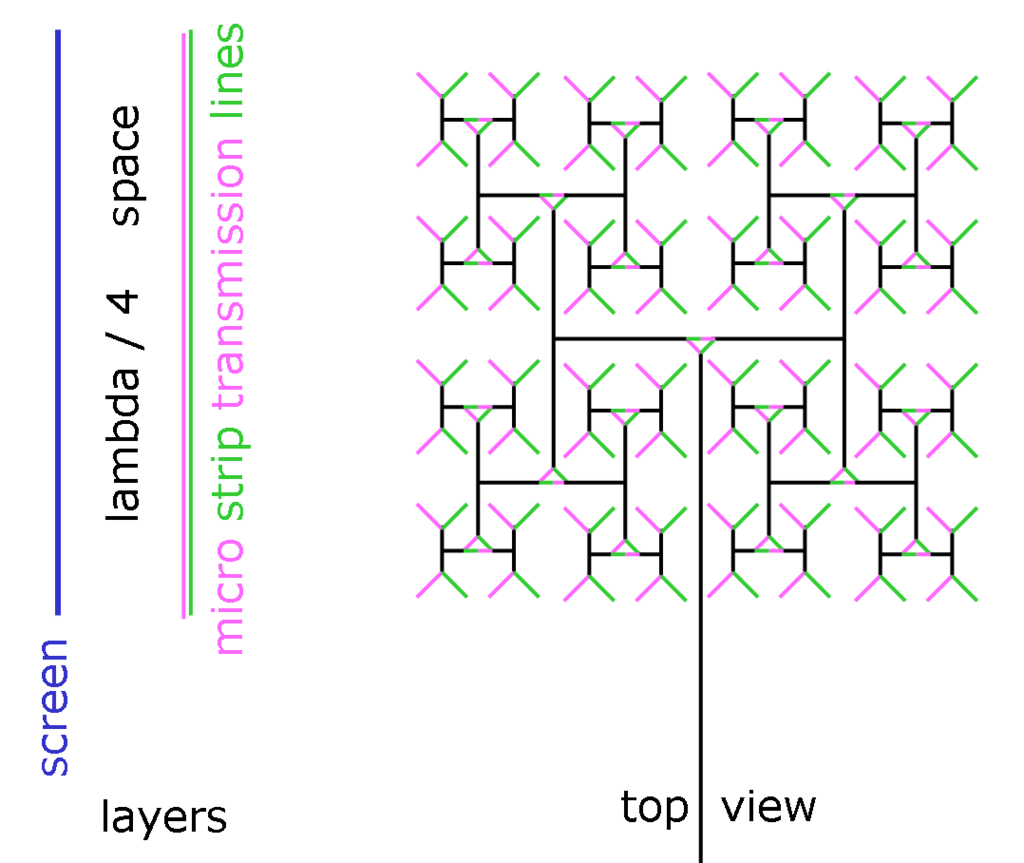


Рисунок . Фрактальная антенна

# 2. Рисование на языке Python

## 2.1 Графическая библиотека Tkinter

**Tkinter (Tk)** – графическая библиотека Python, которая предназначена для создания программ с оконным интерфейсом. Она кроссплатформенная, то есть с ее помощью можно писать приложения для Windows, Linux, macOS.

Кроссплатформенная графическая библиотека **Tk** содержит компоненты графического интерфейса пользователя (*graphical user interface* – GUI). Эта библиотека написана на языке программирования **Tcl** (**Tcl** – это скриптовый язык). Область применения **Tcl/Tk** – быстрое прототипирование. С его помощью создают графические интерфейсы, встраивают новые сценарии в программы, тестируют.

Чтобы работать с **Tcl/Tk** на Python, используется библиотека **Tkinter**. Таким образом, **Tkinter**– это интерфейс **Tcl/Tk**. На английском его обычно называют «тикль-ток», на русском – «так-тикль».

Удобство **Tkinter** в том, что она входит в стандартный набор Python. Если в системе установлен Python, то **Tkinter** тоже работает.

### 2.1.1 Общая схема создания окна

Листинг 1:

From tkinter import \* # Импортируем библиотеку

window = Tk() # Создаем новое окно

window.mainloop() # Запускаем бесконечный цикл окна

**Tk** является базовым классом любого **Tkinter** приложения. При создании объекта этого класса запускается интерпретатор **tcl/tk** и создаётся базовое окно приложения.

**Tkinter** является событийно-ориентированной библиотекой. В приложениях такого типа имеется главный цикл обработки событий. В **Tkinter** такой цикл запускается методом **mainloop()**. Окно будет ждать любого действия от пользователя до тех пор, пока пользователь его не закроет (рис. 14). Без **mainloop()** окно не отобразится.

Для явного выхода из интерпретатора и завершения цикла обработки событий используется метод **quit()**.

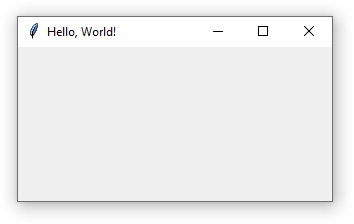


Рисунок 14 – Пустое окно (результат работы программы листинга 1)

Особенность **Tkinter** в том, что библиотека автоматически подстраивает внешний вид окна под стиль операционной системы. На Windows, Linux и macOS оно будет различным, соответствующим стилю системы (рис. 15).



Рисунок . Различие стилей базового окна, объекта класса Tk, в зависимости от операционной системы

### 2.1.2 Базовые виджеты

Виджеты – это основа библиотеки **Tkinter**. Через них пользователи взаимодействуют с программой.

Каждый виджет определен классом (см. таблицу 1).

Таблица 1 – Классы виджетов

|  |  |  |
| --- | --- | --- |
| **Класс виджета** | **Что делает** | **Пример использования** |
| **Frame** | Помогает организовать пользовательский интерфейс как визуально, так и на уровне кода. Отображается как простой прямоугольник. | Разделение интерфейса на блоки. |
| **Label** | Виджет для отображения текста или изображения. Пользователи могут его просматривать, но не могут взаимодействовать с ним. | Заголовки, подписи, иконки в интерфейсе. |
| **Button** | Это элемент интерфейса, с которым пользователи могут взаимодействовать. На кнопку нажимают, чтобы выполнить действие. **Button** может отображать текст и изображение, как и **Label**, но также может принимать дополнительные параметры для изменения поведения. | Кнопки в интерфейсе: для перехода, сохранения, выхода из программы. |
| **Checkbutton** | Разновидность кнопки, которая содержит двоичное значение. При нажатии переключатель переворачивается, затем происходит обратный вызов. | Включение и выключение опций с помощью галочек. |
| **Radiobutton** | Кнопка, которая позволяет выбрать один из нескольких взаимоисключающих вариантов. | Список с опциями, например, для выбора языка интерфейса. |
| **Entry** | Виджет для ввода одной строки текста. | Указание имени, пароля, города и других данных пользователей. |
| **Combobox** | Объединяет **Entry** со списком опций. Пользователи могут выбирать из предложенных вариантов или указывать свои. | Выпадающий список. |

Это лишь базовые виджеты. Посмотреть их подробное описание можно в официальной документации **Tk**.

### 2.1.3 Общие для всех виджетов свойства

Все виджеты в **Tkinter** обладают некоторыми общими свойствами. Опишем их, перед тем как перейти к рассмотрению конкретных виджетов. Виджеты создаются вызовом конструктора соответствующего класса.

Первый аргумент (как правило неименованный, но можно использовать имя **master**) это родительский виджет, в который будет упакован (помещен) создаваемый виджет. Родительский виджет можно не указывать, в таком случае будет использовано главное окно приложения.

Далее следуют именованные аргументы, конфигурирующие виджет. Это может быть используемый шрифт (**font**=...), цвет виджета (**bg**=...), команда, выполняющаяся при активации виджета (**command**=...) и так далее. Полный список всех аргументов можно посмотреть в **manoptions** и **man**-странице соответствующего виджета (например **manbutton**, см. разделы «STANDARD OPTIONS» и «WIDGET-SPECIFIC OPTIONS»).

Приведем пример кода, создающего окно с двумя кнопками: первая кнопка будет закрывать окно (завершать работу программы), вторая – печатать заданное слова в консоль Python (рис.16).

Листинг 2:

from tkinter import \*

def button\_clicked():

print ("Клик!")

root=Tk()

button1 = Button(root,bg="grey",

width=10,

font='Arial 14',

text="Close",

command=root.destroy)

button1.pack()

button2 = Button(

root,

bg="red",

text="Кликни меня!",

command=button\_clicked

)

button2.pack()

root.mainloop()

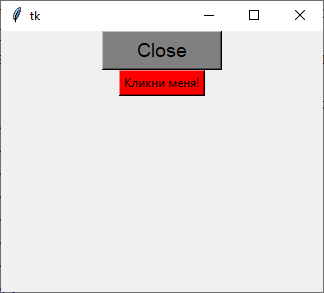


Рисунок 16 – Окно с двумя кнопками, созданными в листинге 2

В любом приложении виджеты не разбросаны по окну как попало, а организованы, интерфейс продуман до мелочей и обычно подчинен определенным стандартам. В примере листинга 2 элементы расположены друг под другом с помощью наиболее простого менеджера геометрии – **pack()**.

### 2.1.4 Виджет Canvas

Рисование в **Tkinter** реализовано при помощи виджета **Canvas**. Виджет **Canvas** имеет функционал высокого уровня, который позволяет создавать графику в **Tkinter**. Рисование можно использовать для создания графиков статистики, самодельных пользовательских виджетов и даже небольших игр.

Общая схема программы, использующей виджет Canvas (рис. 17), представлена в листинге 3.

Листинг 3:

from tkinter import \*

root=Tk()

canvas=Canvas(root, width=400, height=200)

canvas.create\_line(

60, 90, 160, 90, 110, 185, 60, 90,

fill="red",

width=4

)

canvas.pack()

button = Button(

root,

bg="grey",

text="Close",

command=root.destroy

)

button.pack()

root.mainloop()

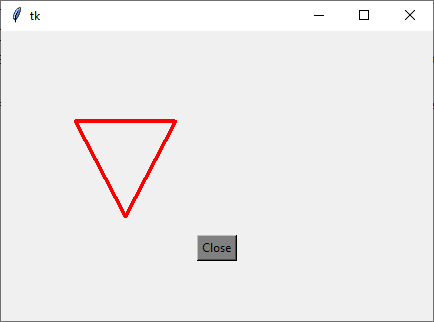


Рисунок . Пример окна, на котором размещен графический виджет Canvas

Рассмотрим методы класса **Canvas** для построения графических объектов (см. таблицу 2).

Таблица 2 – Методы класс Canvas

|  |  |  |
| --- | --- | --- |
| **Графический объект** | **Параметры** | **Примеры** |
| Линия (отрезок прямой)  **create\_line()** | **x1, y1, x2, y2** – координаты начальной и конечной точек;  **dash(a, b)** – пунктирная линия, где **a** – длина пунктирного тире в пикселях, **b** – пустой промежуток между тире;  **fill**– **цвет линии;**  **width** – толщина линии | **canvas.create\_line(20, 30, 220, 30)**  #горизонтальная сплошная линия  **canvas.create\_line(300, 35, 300, 150, dash=(3, 2))**  #вертикальная пунктирная линия |
| Ломаная линия  **create\_line()** | *n*-пар координат **x, y** точек, которые будут соединяться линией  **fill**– **цвет линии**  **width** – толщина линии | **canvas.create\_line(**  **60, 90, 160, 90,**  **110, 185, 60, 90,**  **fill="red",**  **width=4**  **)**  #замкнутая ломаная красного цвета (треугольник) с координатами вершин (60, 90), (160, 90), (110, 185) и толщиной линии 4  **canvas.create\_line(**  **250, 20, 380, 90,**  **300, 130, 160, 10,**  **fill="black",**  **width=3**  **)**  # ломаная черного цвета с четырьмя вершинами |
| Прямоугольник  **create\_rectangle()** | **x1, y1, x2, y2** – координаты левой верхней и правой нижней вершин;  **outline** – цвет контурной линии;  **fill** – закраска внутренней области прямоугольника;  **width** – толщина контура. | **canvas.create\_rectangle(**  **30, 10, 320, 180,**  **outline="#fb0", fill="#fb0"**  **)** |
| Многоугольник  **create\_polygon()** | *n*-пар координат **x, y** вершин многоугольника;  **outline** – цвет контурной линии;  **fill** – закраска внутренней области многоугольника;  **width** – толщина контура. | **points = [**  **150, 100, 200, 120,**  **240, 180, 210, 200,**  **150, 150, 100, 200**  **]**  **canvas.create\_polygon(**  **points, outline='maroon',**  **fill='yellow', width=4**  **)** |
| Круг (овал)  **create\_oval()** | Первые четыре параметра определяют ограничивающие координаты фигуры. Иными словами, это **x** и **y** координаты верхней левой и правой нижней точек квадрата (прямоугольника), в который помещен круг (овал);  **outline** – цвет контурной линии;  **fill** – закраска внутренней области фигуры;  **width** – толщина контура. | **canvas.create\_oval(**  **50, 50, 280, 180,**  **outline="green",**  **fill="yellow",**  **width=3**  **)** |
| Сектор  **create\_arc()** | Первые четыре параметра определяют ограничивающие координаты фигуры. Иными словами, это **x** и **y** координаты верхней левой и правой нижней точек квадрата (прямоугольника), в который помещен сектор;  **start** – начальный угол сектора;  **extent** – конечный угол сектора;  **outline** – цвет контурной линии;  **fill** – закраска внутренней области фигуры;  **width** – толщина контура. | **canvas.create\_arc(**  **100, 30, 290, 130,**  **start=0,**  **extent=270,**  **outline="#dbd",**  **fill="#0bb",**  **width=4**  **)** |
| Текст  **canvas.create\_text()** | Первые два параметра – это **x** и **y** координаты центральной точки текста;  Если мы закрепим текстовый объект по направлению запада **anchor=W**, текст будет начинаться в этой части окна;  **font** – шрифт текста;  **text** – строка, выводимая на область рисования Canvas. | **canvas.create\_text(**  **100, 100,**  **anchor=W,**  **font=("Courier",20,"bold"),**  **text="На пылающий город**  **падает тень",**  **fill="red",**  **width=150**  **)** |

## 2.2 Исполнитель «Черепашка»

Библиотека **turtle** – это расширение языка Python, позволяющее рисовать на экране несложные рисунки. Представьте себе, что по экрану компьютера ползает маленькая черепашка (**turtle**). Вы можете управлять движением черепашки, отдавая ей различные команды вида «Проползти вперед на 10 пикселей», «Повернуть направо», «Повернуть налево». После того, как вы отдадите ей команду «Начать рисовать», черепашка будет оставлять за собой след, пока не получит команду «Закончить рисовать». Управлять черепашкой можно при помощи инструкций языка Python.

Чтобы использовать исполнителя «черепашку», достаточно сделать импорт библиотеки **turtle**:

import turtle

Далее в программе необходимо создать экземпляр класса turtle:

t = turtle.Turtle()

Если сейчас запустить программу, то появится графическое окно и тут же пропадет. Чтобы этого не происходило, в конце добавим команду:

turtle.done()

Далее «черепашка» может ползти вперед (то есть, куда глаза глядят) с помощью команды **forward()**, поворачиваться влево, вправо с помощью команд **left()** и **right()**.

Начало координат в окне для графики модуля **turtle** находится в центре окна. Положительное направление оси X определяется слева направо, положительное направление оси Y определяется снизу вверх, чем больше X, тем правее черепашка, чем больше Y, тем выше черепашка.

Работу с исполнителем «черепашка» продемонстрируем непосредственно при построении фрактала в практической работе № 2.

# 3. Практические работы

## Практическая работа № 1. Построение фрактала «Снежинка Коха» с использованием графических примитивов Tkinter

Снежинка Коха является фрактальной фигурой, начальной формой которой служит треугольник, а на каждой следующей итерации отдельный линейный сегмент заменяется ломаной. Получить снежинку Коха можно выполнив построение трех кривых Коха одинаковой длины, но под разными углами и с различными стартовыми точками (рис. 18).

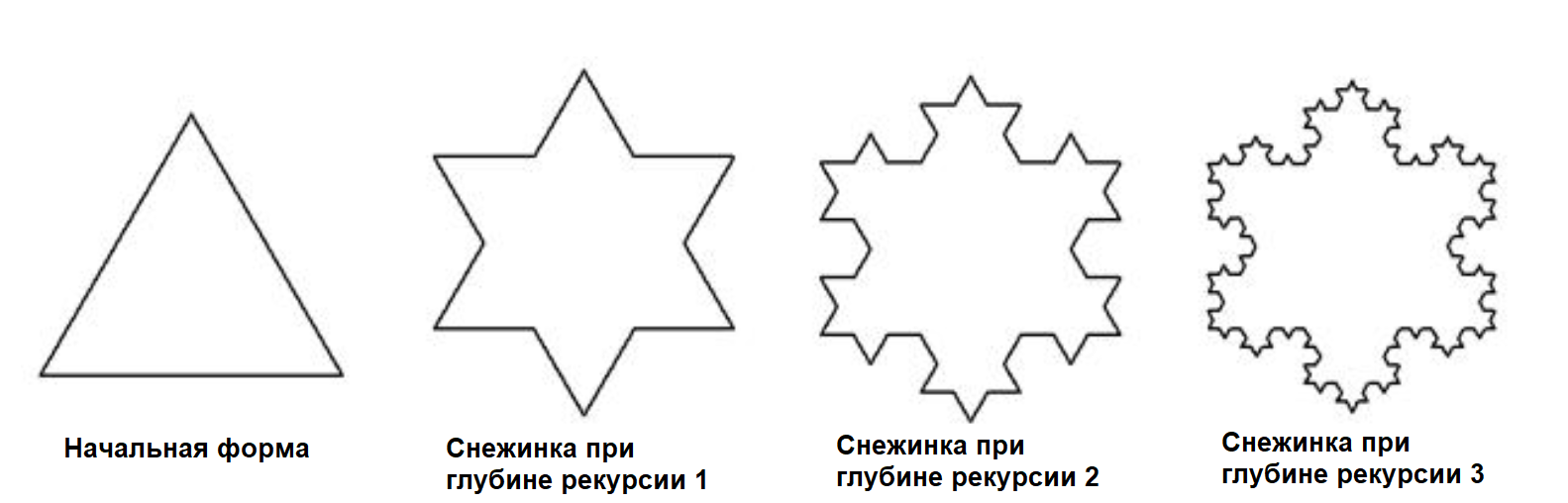


Рисунок 18. Глубина рекурсии и вид снежинки Коха

Для построения начальной формы договоримся, что одна из сторон треугольника (обозначим ее AC) параллельна оси Ох (рисунок 19).

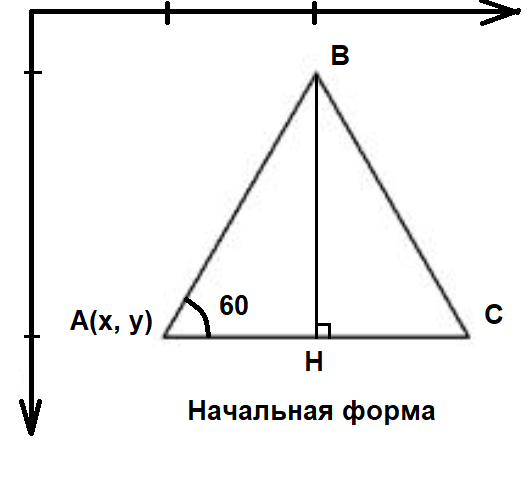


Рисунок 19. Определение координат точки B при известных координатах точки А

Для первого линейного сегмента начальной формы AB определим параметры:

**x, y** – координаты стартовой точки A;

**lnght** – длина линейного сегмента;

**ang** – угол наклона AB к оси Ох.

Тогда для линейных сегментов BC и CA координаты стартовых точек можно рассчитать последовательно следующим образом:

**x+=lngth\*cos(ang\*pi/180)**

**#**изменяется на длину катета AH

**y-=lngth\*sin(ang\*pi/180)**

**#**изменяется на длину катета BH

А угол наклона каждого следующего линейного сегмента изменяется на 120 градусов по часовой стрелке:

ang-=120

Определив параметры для всех трех кривых Коха, перейдем к построению самой кривой.

Кривая Коха строится по простому правилу: каждый ее линейный сегмент на последующей итерации заменяется ломаной (рис. 20). Или, также можно сказать, что центральная часть отрезка вырезается, а на его месте надстраивается «фиорд» из двух отрезков, образующих с вырезанной частью равносторонний треугольник.

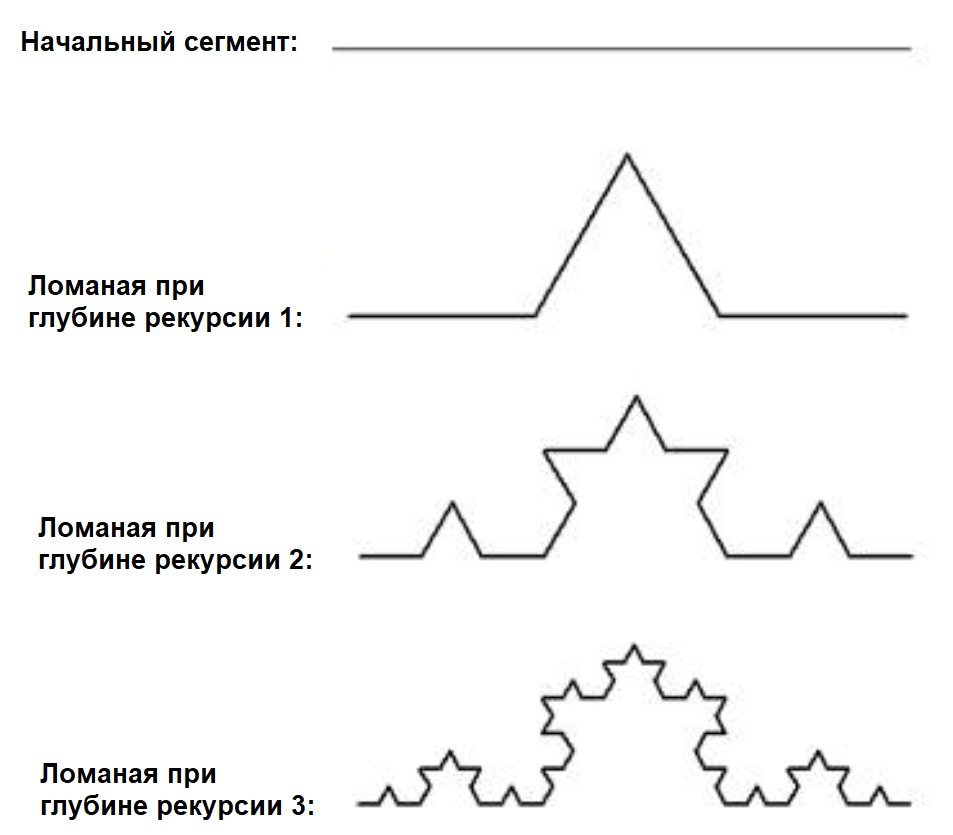


Рисунок 20. Глубина рекурсии и вид ломаной Коха

Начинаем с построения прямой линии при глубине рекурсии достигшей нулевого значения. Так как для каждого линейного фрагмента известны координаты только стартовой точки, то рассчитаем координаты второй точки через длину линейного сегмента и угол его наклона к оси Ох аналогично тому, как были рассчитаны координаты вершин треугольника при построении основы снежинки Коха.

dx=3\*len\*cos(ang\*pi/180)

dy=-3\*len\*sin(ang\*pi/180)

canvas.create\_line(x,y,x+dx,y+dy, fill='black', width=1)

Если же уровень рекурсии еще не нулевой, то заменяем прямую линию ломаной, состоящей из 4-х линейных сегментов равной длины:

for i in range(4):

Koch\_curve(canvas,x,y,len/3,ang,n-1)

#рекурсивный вызов для прорисовки сегмента ломаной

dx=len\*cos(ang\*pi/180)

dy=-len\*sin(ang\*pi/180)

x=x+dx

y=y+dy

if i%2==0:

ang+=60

else:

ang-=120

Длина одного отрезка этой ломаной равна длине линейного сегмента, уменьшенного в три раза, углы между прямыми изменяются на 60 градусов против часовой стрелки или на 120 градусов по часовой стрелке. В результате рекурсивных спусков, получается самоподобная кривая.

Приведем полный листинг программы.

Листинг 4:

from tkinter import \*

from math import \*

def Koch\_curve(canvas,x,y,len,ang,n):

if n==0:

dx=3\*len\*cos(ang\*pi/180)

dy=-3\*len\*sin(ang\*pi/180)

canvas.create\_line(

x,y,x+dx,y+dy,

fill='black', width=1

)

else:

for i in range(4):

Koch\_curve(canvas,x,y,len/3,ang,n-1)

dx=len\*cos(ang\*pi/180)

dy=-len\*sin(ang\*pi/180)

x=x+dx

y=y+dy

if i%2==0:

ang+=60

else:

ang-=120

def start\_draw():

c.delete('all')

level=int(entry\_level.get())

if level>7:

c.create\_text(

100, 100, anchor=W,

font=("Courier",20,"bold"),

text="Недопустимое значение,

введите число <= 7",

fill="red", width=450

)

else:

lngth=660

ang=60

x=80

y=h/3\*2

for \_ in range(3):

Koch\_curve(c,x,y,lngth/3,ang,level)

x+=lngth\*cos(ang\*pi/180)

y-=lngth\*sin(ang\*pi/180)

ang-=120

root=Tk()

root.title("Фрактал - снежинкаКоха")

w=900

h=900

c = Canvas(root, width=w, height=h, bg='white')

c.grid(rowspan=w//10, column=0)

Label(root, width=10,

font='Arial 14',

text="Глубина:"

).grid(row=7, column=1,sticky=S)

entry\_level=Entry(root, width=10, font='Arial 16')

entry\_level.grid(row=8, column=1)

button1 = Button(root,bg="grey",

width=10,

font='Arial 14',

text="Построить",

command=start\_draw)

button1.grid(row=9, column=1, sticky=N)

button2 = Button(root,bg="grey",

width=10,

font='Arial 14',

text="Close",

command=root.destroy)

button2.grid(row=11, column=1, sticky=N)

root.mainloop()

Результат работы программы приведен на рисунке 21.

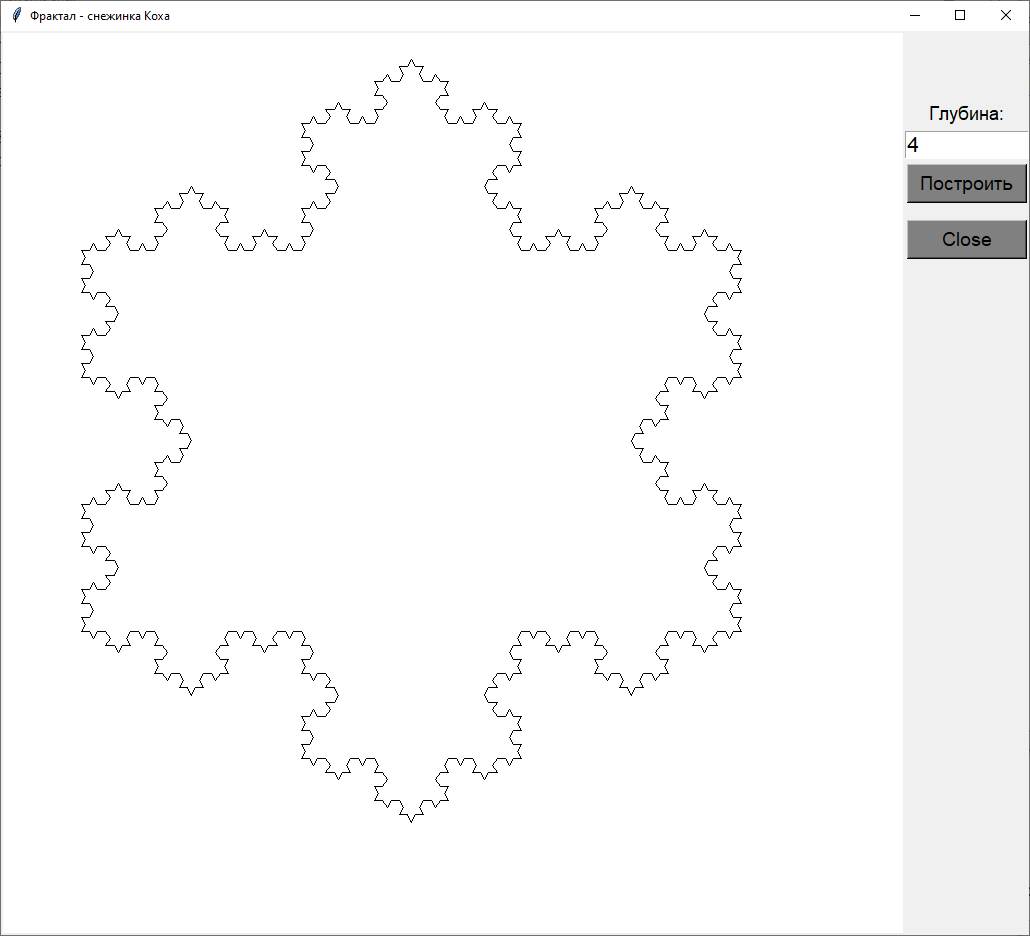


Рисунок 21. Демонстрация работы программы построения снежинки Коха

**Примечание**: на каждом шаге рекурсии длина образующейся ломаной увеличивается в сравнении с предыдущей в раза и составляет , где – уровень рекурсии. Тогда периметр «острова» Коха на -ом уровне рекурсии составляет , а значит с увеличением , растет неограниченно – до бесконечности. При этом, площадь «острова» Коха конечна и равна . *Интересный факт – конечная площадь фигуры ограничена периметром бесконечной длины!*

## Практическая работа № 2. Построение фрактала «Снежинка Коха» с помощью черепашьей графики

Черепаха находится на координатной плоскости и может передвигаться по ней только вперёд, но может также поворачиваться на месте. При движении черепаха способна чертить линию карандашом, или двигаться, не оставляя за собой след. Есть возможность заменять карандаши, и тем самым управлять толщиной линии и её цветом.

Вот перечень команд черепашьего языка (он не является исчерпывающим):

* forward(x) #сдвинуться вперёд на расстояние x, оставляя за собой линию, если карандаш опущен;
* left(angl) #повернуться влево на заданный угол относительно текущего курса черепахи;
* right(angl) #повернуться вправо на заданный угол относительно текущего курса черепахи.

Идея построения

Нарисовать один сегмент кривой Коха можно следующим образом:

**for i in range(4):**

**t.fd(ln)**

**if i<3:**

**if i%2==0:**

**t.left(60)**

**else:**

**t.right(120)**

Чтобы рисование происходило быстрее, мы ускорим черепашку командой:

**t.speed(100)**

Итак, одна итерация у нас есть. Теперь, каждый линейный фрагмент также нужно прорисовать такой же ломаной, но уменьшенной в 3 раза. Проще всего это сделать через рекурсию:

Мы заменяем линейные сегменты, пока длина сегмента ln больше 6 пикселей. Соответственно, при выполнении этого условия, уходим вглубь рекурсии и черепашка готова прорисовывать более мелкие детали. Когда условие не выполняется, то начинается рисование, причем, только самых мелких деталей. Фактически, рекурсией мы здесь сформировали множество однотипных команд для прорисовки кривой Коха.

Листинг 5:

import turtle

**def draw\_koch\_segment(t, ln):**

**if ln > 6:**

**for i in range(4):**

**draw\_koch\_segment(t, ln//3)**

**if i<3:**

**if i%2==0: t.left(60)**

**else: t.right(120)**

**else:**

**for i in range(4):**

**t.fd(ln)**

**if i<3:**

**if i%2==0: t.left(60)**

**else: t.right(120)**

#основная программа

t = turtle.Turtle()

ln = 150

t.penup()

t.backward(200)

t.pendown()

t.speed(100)

t.ht()

for \_ in range(3):

draw\_koch\_segment(t, ln)

t.right(120)

turtle.done()

Результат работы программы листинга 5 представлен на рисунке 22.

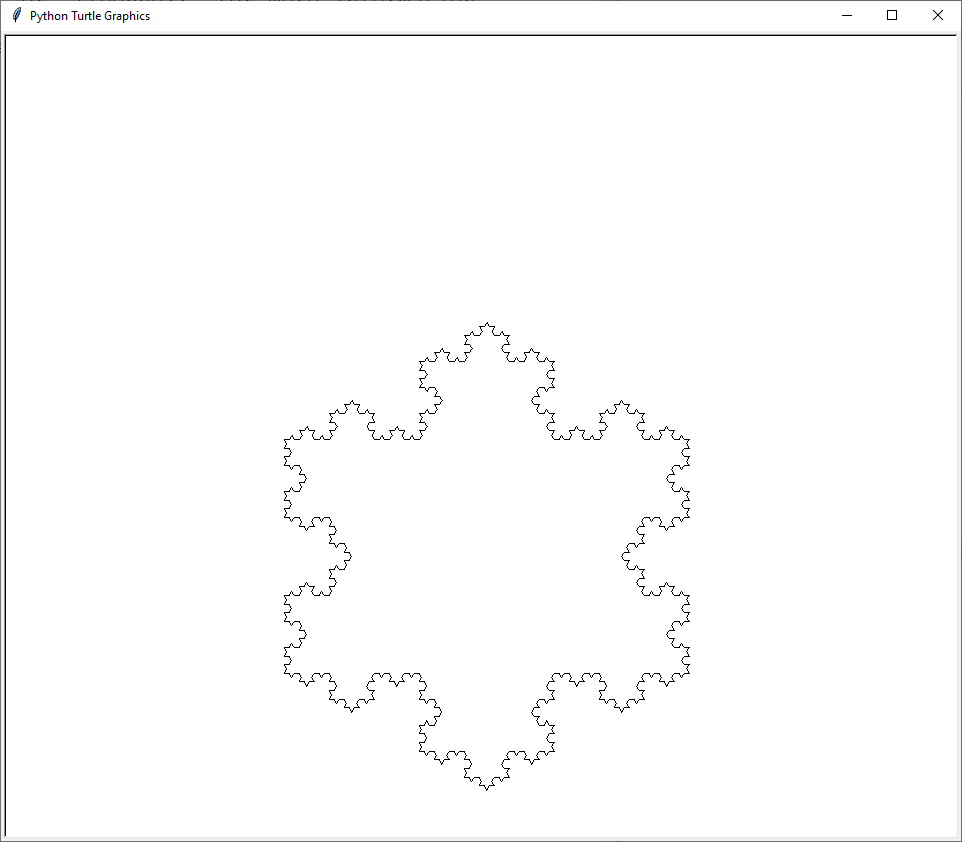


Рисунок . Снежинка Коха, построенная «черепашьей» графикой

## Практическая работа № 3. Построение фрактала «Треугольник Серпинского» с использованием графических примитивов Tkinter

Еще один классический пример геометрического фрактала – треугольник Серпинского (также называется салфеткой Серпинского, рис. 23).

Его построение начинается с закрашенного равностороннего треугольника. Из первоначального треугольника «вырезается» перевернутый центральный равносторонний треугольник, со стороной, равной половине длины стороны исходного треугольника.

Остаются три равносторонних треугольника со сторонами, вдвое меньше стороны исходного треугольника. К ним также применяется операция удаления центрального треугольника, после чего образуется девять треугольников, и так до бесконечности.

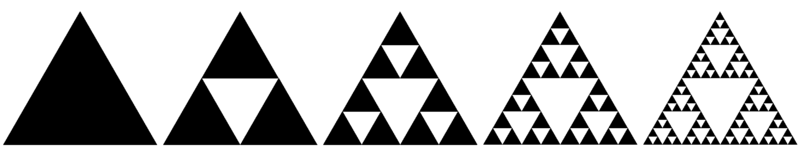


Рисунок . Треугольник Серпинского на 1-5 уровнях рекурсии

Для начала необходимо построить черный треугольник, который будет основой нашего фрактала. Для этого определим параметры:

x, y – координаты стартовой точки A (рис. 24);

lngth – длина стороны треугольника.

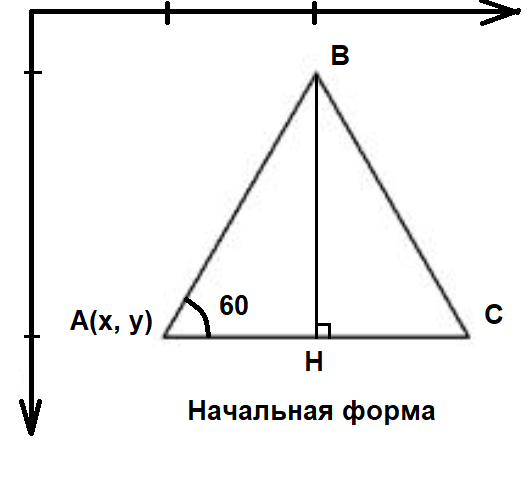


Рисунок . Основа треугольника Серпинского

Для расчета координат точек B и C понадобятся следующие величины:

dx=lngth/2 #половина длины катета AC

dy=sqrt(lngth\*\*2-(lngth/2)\*\*2) #длина высоты AH

Тогда для построения треугольника воспользуемся функцией canvas.create\_polygon() и определим координаты четырех точек следующим образом:

canvas.create\_polygon(

[x,y,x+dx,y+dy,xdx,y+dy,x,y],

outline='black',

fill='black',

width=1

)

Далее нам необходимо «вырезать» из каждого черного треугольника его серединный треугольник, для этого будем строить внутри треугольника белый, вершины которого совпадают с серединами сторон черного. Именно это действие и будет составлять рекурсивный переход, так как «вырезав» из черного треугольника белый, сразу получаем три черных треугольника меньшего размера, для которых процедура должна повториться.

Листинг 6:

from tkinter import \*

from math import \*

def treugol(canvas,x,y,lngth):

dy=sqrt(lngth\*\*2-( lngth/2)\*\*2)

dx=lngth/2

canvas.create\_polygon(

[x,y,x+dx,y+dy,x-dx,y+dy,x,y],

outline='black',

fill='black',

width=1

)

def salfetka(canvas,x,y,lngth,n):

if n>0:

dy=sqrt(lngth\*\*2-(lngth/2)\*\*2)

dx=lngth/2

canvas.create\_polygon(

[x+dx,y+dy,x,y+2\*dy,x-dx,y+dy,x+dx,y+dy],

outline='white',

fill='white',

width=1

)

salfetka(canvas,x,y,lngth/2,n-1)

salfetka(canvas,x-dx,y+dy,lngth/2,n-1)

salfetka(canvas,x+dx,y+dy,lngth/2,n-1)

root=Tk()

root.title("Фрактал – треугольник Серпинского")

w=900

h=900

c = Canvas(root, width=w, height=h, bg='white')

c.pack()

lngth=800

treugol(c,w/2,0,lngth)

salfetka(c,w/2,0,lngth/2,6)

root.mainloop()

Результат работы программы представлен на рисунке 25.

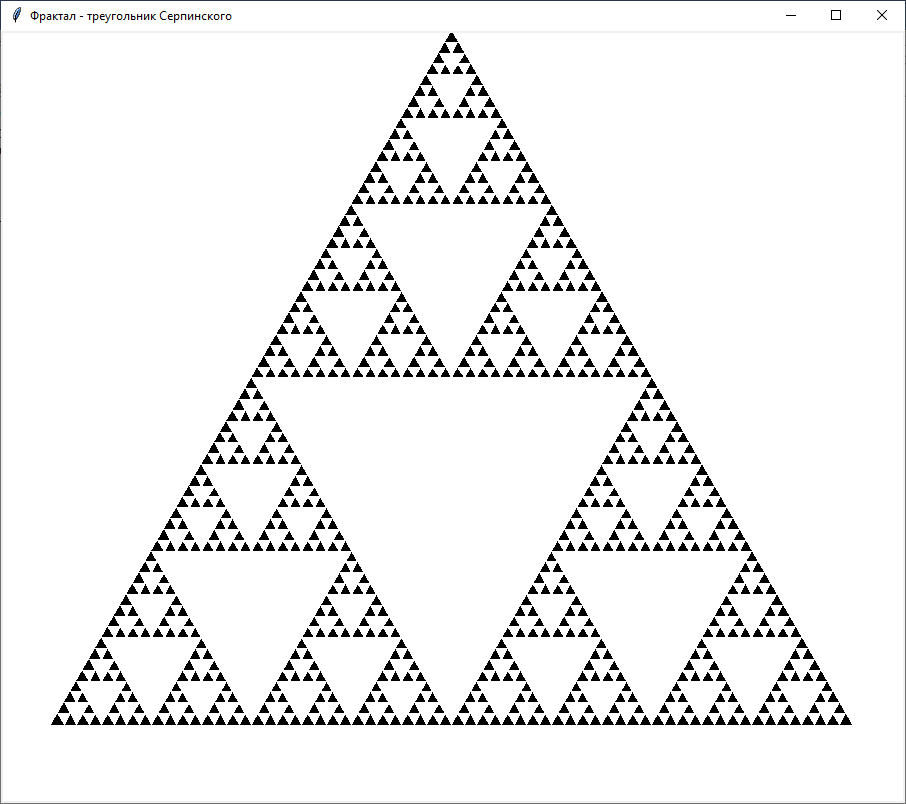
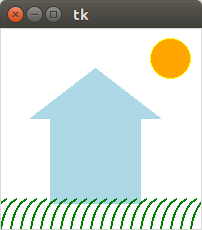


Рисунок . Треугольник Серпинского с глубиной рекурсии, равной 6

## Задания для самостоятельной работы

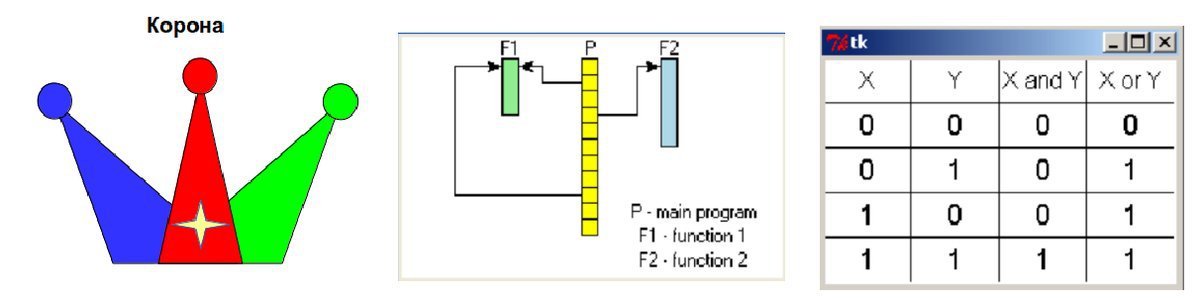
### Создание изображения на холсте в Tkinter

1. Нарисовать окружность с центром в точке (300, 200), и радиусом 50.
2. Нарисовать закрашенную окружность в точке (300, 200), и радиусом 50.
3. Нарисовать линию – отрезок, соединяющий точки (350, 30) и (120, 190).
4. Нарисовать красную линию между точками (20, 300) и (420, 90).
5. Нарисовать красную линию толщиной 3 из точки (10,300) в точку (300,300).
6. Создать на холсте подобное изображение:
   1. «Домик»

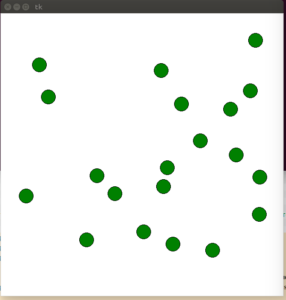


*Примечание: для создания травы используется цикл.*

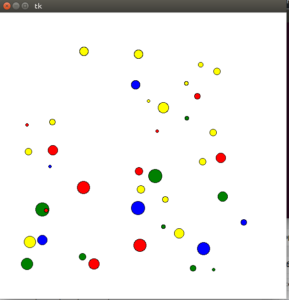
* 1. «Корона»



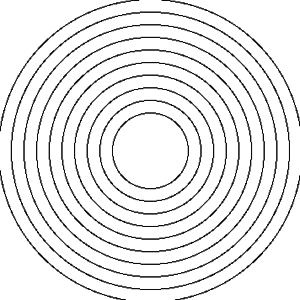
* 1. «Сосуд с молекулами газа» – 20 кругов в случайных местах

[](http://progras.ru/wp-content/uploads/2016/07/%D0%A1%D0%BD%D0%B8%D0%BC%D0%BE%D0%BA-%D1%8D%D0%BA%D1%80%D0%B0%D0%BD%D0%B0-%D0%BE%D1%82-2016-10-09-22-18-50.png)

* 1. «Конфетти» – 40 кругов в случайных местах случайного размера, случайного цвета

[](http://progras.ru/wp-content/uploads/2016/07/%D0%A1%D0%BD%D0%B8%D0%BC%D0%BE%D0%BA-%D1%8D%D0%BA%D1%80%D0%B0%D0%BD%D0%B0-%D0%BE%D1%82-2016-10-09-22-21-20.png)

* 1. «Жесткий диск» – 10 концентрических окружностей

[](http://progras.ru/wp-content/uploads/2016/07/%D0%A1%D0%BD%D0%B8%D0%BC%D0%BE%D0%BA-%D1%8D%D0%BA%D1%80%D0%B0%D0%BD%D0%B0-%D0%BE%D1%82-2016-10-09-22-31-41.png)

* 1. «Улица» – несколько домов в ряд

[](http://progras.ru/wp-content/uploads/2016/07/%D0%A1%D0%BD%D0%B8%D0%BC%D0%BE%D0%BA-%D1%8D%D0%BA%D1%80%D0%B0%D0%BD%D0%B0-%D0%BE%D1%82-2016-10-25-14-56-26.png)

* 1. Нарисовать таблицу со случайно выбранными числами из последовательности: 2, 4, 8, 16, 32, 64, 128, 256, 512. Каждому числу должен соответствовать свой цвет.

[](http://progras.ru/wp-content/uploads/2016/07/%D0%A1%D0%BD%D0%B8%D0%BC%D0%BE%D0%BA-%D1%8D%D0%BA%D1%80%D0%B0%D0%BD%D0%B0-%D0%BE%D1%82-2016-10-17-02-05-51.png)

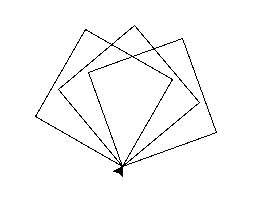
### Создание изображения графикой Turtle

1. Нарисовать квадрат с длиной стороны 150:



Угол квадрата – прямой, то есть равен 90 градусов.

1. Нарисовать несколько квадратов, с поворотом относительно друг друга:

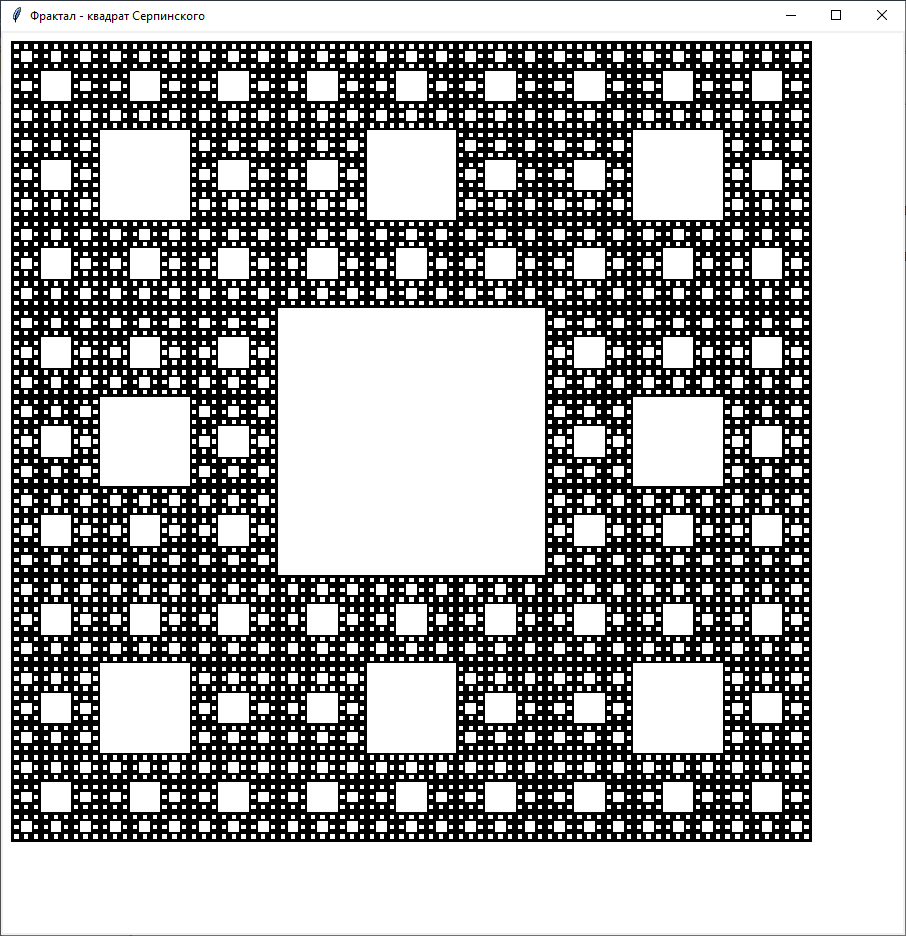


На изображении показан поворот черепашки на 20 градусов при переходе к следующему квадрату. Попробуйте задавать разные углы, например, 30 или 40.

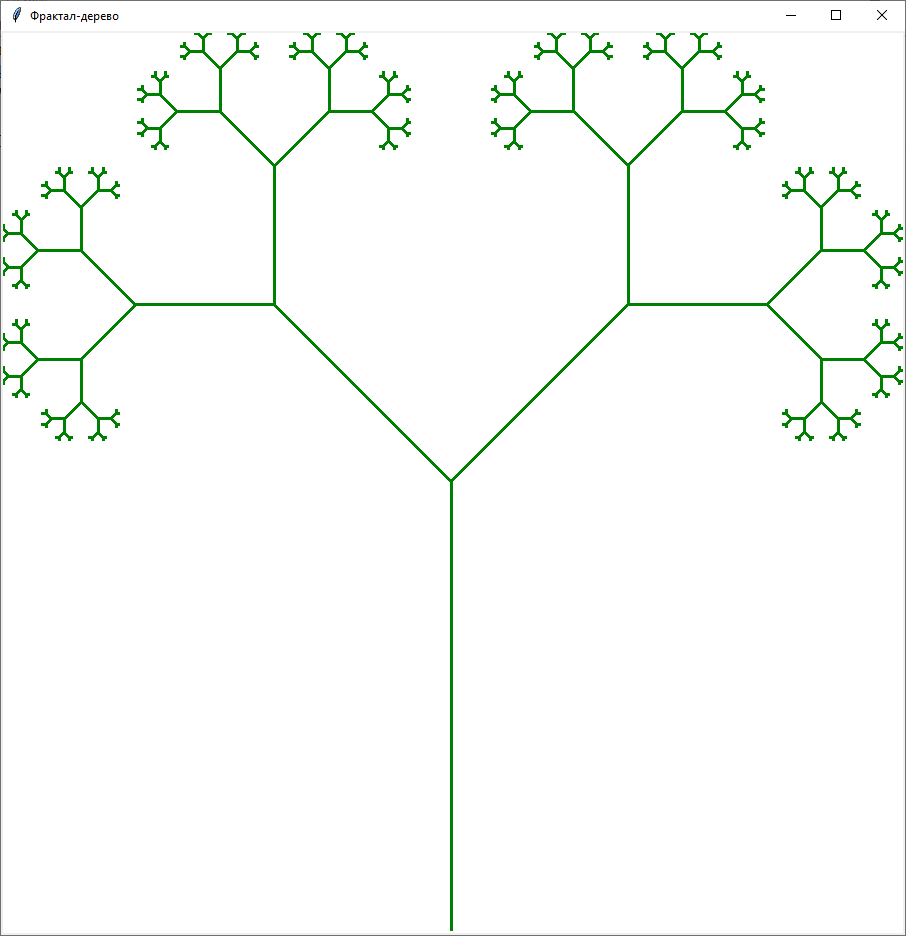
1. Построить изображение «домика» (квадрат под треугольником) *без подъёма пера* при условии однократного перемещения по каждой линии.

### Построение фракталов

1. Постройте квадратный ковер Серпинского



1. Постройте фрактал-дерево



## Ответы к заданиям на построение фракталов

### Ковер Серпинского

Листинг 7

from tkinter import \*

from math import \*

#функция отрисовки квадрата с координатами левого

#верхнего угла (x, y) и длиной стороны len

def kvadrat(canvas,x,y,len):

canvas.create\_rectangle(

[x,y,x+len,y+len,],

outline='black',

fill='black',

width=1

)

#Рекурсивная функция, которая вызывает себя 9 раз

#через вложенные циклы с тремя итерациями.

#Предварительно в центре черного исходного квадрата

#прорисовывается белый квадрат с длиной стороны len/3.

def kover(canvas,x,y,len,n):

if n>0:

d=len/3

canvas.create\_rectangle(

[x+d,y+d,x+d\*2,y+d\*2],

outline='white',

fill='white',

width=1

)

for i in range(3):

for j in range(3):

if i!=1 or j!=1:

kover(canvas,x+d\*j,y+d\*i,len/3,n-1)

#основная программа

root=Tk()

root.title("Фрактал - квадрат Серпинского")

w=900

h=900

c = Canvas(root, width=w, height=h, bg='white')

c.pack()

lngth=800

kvadrat(c,10,10,lngth)

kover(c,10,10,lngth,5)

root.mainloop()

### Фрактал-дерево

Листинг 8

from tkinter import \*

from math import \*

#рекурсивная функция

def fractal(canvas,x,y,len,ang,n):

if n > 0:

#по известным координатам начальной точки текущего

#отрезка **x,y** и углу наклона отрезка **ang** определяем

#смещение по X и Y, которые позволяют получить

#конечную точку отрезка и построить такой отрезок

dy = len\*(-sin(ang\*pi/180))

dx = len\*(-cos(ang\*pi/180))

canvas.create\_line(

x,y,x+dx,y+dy,

fill='green',

width=3

)

#далее дважды вызываем рекурсивную функцию с новыми

#значениями параметров: начало каждого из двух следующих

#отрезков – это конец текущего отрезка, а угол

#относительно текущего изменяется на 45 градусов

#вправо и влево

fractal(canvas,x+dx,y+dy,len/1.8,ang+45,n-1)

fractal(canvas,x+dx,y+dy,len/1.8,ang-45,n-1)

#основная программа

root=Tk()

root.title("Фрактал-дерево")

w = 900

h = 900

c = Canvas(root, width=w, height=h, bg='white')

c.pack()

fractal(c, w//2, h, h//2, 90, 9)

root.mainloop()

# Список источников

1. Божокин, С. В. Фракталы и мультифракталы / С. В. Божокин, Д. А. Паршин. – Ижевск : Регулярная и хаотическая динамика, Институт компьютерных исследований, 2019. – 128 c. – ISBN 978-5-4344-0780-9. – Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/92075.html (дата обращения: 12.03.2022). – Режим доступа: для авторизир. Пользователей
2. Иудин Д.И. Фракталы: от простого к сложному / Д.И. Иудин, Е.В. Копосов; Нижегор. гос. архитектур.-строит. ун-т – Н. Новгород: ННГАСУ. – 2012. – 200 с.
3. Махоркин, А. В. Математика фракталов : учебное пособие / А. В. Махоркин, В. В. Махоркин. – Калининград : Балтийский федеральный университет им. Иммануила Канта, 2011. – 156 c. – ISBN 978-5-9971-0163-3. – Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/23794.html (дата обращения: 12.02.2022). – Режим доступа: для авторизир. пользователей
4. Поляков К. Ю. Программирование. Python. C++. Часть 2: учебное пособие / К. Ю. Поляков. – М. : БИНОМ. Лаборатория знаний, 2019. – 176 с. : ил. ISBN 978-5-9963-4135-1
5. Тренькин, А. А. Введение в теорию фракталов. Математические аспекты и некоторые физические приложения : учебное издание / А. А. Тренькин. – Саров : Российский федеральный ядерный центр – ВНИИЭФ, 2007. – 40 c. – ISBN 978-5-9515-0088-5. – Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/60841.html (дата обращения: 27.02.2022). – Режим доступа: для авторизир. пользователей
6. Шеремет, Г. Г. Геометрические преобразования и фрактальная геометрия : учебник / Г. Г. Шеремет. – Пермь : Пермский государственный гуманитарно-педагогический университет, 2013. – 188 c. – Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. – URL: https://www.iprbookshop.ru/32031.html (дата обращения: 04.03.2022). – Режим доступа: для авторизир. пользователей
7. PROPROPROGS / Кривая Коха и снежинка Коха. – URL: https://proproprogs.ru/fractals/krivaya-koha-i-snezhinka-koha

*Учебное пособие*

***ФРАКТАЛЫ НА ЯЗЫКЕ PYTHON***

*Гончаренко Наталья Николаевна*

*Редько Евгения Владимировна*

*Отпечатано с авторского оригинал-макета*

МУНИЦИПАЛЬНОЕ АВТОНОМНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

«ЛИЦЕЙ ИННОВАЦИОННЫХ ТЕХНОЛОГИЙ»

680000, г. Хабаровск, ул. Гоголя, 24

тел. (4212) 32-47-36

http://lit.khv.ru