МИНОБРНАУКИ РОССИИ
ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАННЫХ
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ОТЧЁТ
ПО УЧЕБНОЙ ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКЕ

Студент
гр. БПИ-22-2 _____ Д.Д. Козлов

Руководитель
старший преподаватель каф. ИТС _____ Е.Г. Лаврушина

Владивосток 2024

ЗАДАНИЕ

на учебную ознакомительную практику

Студенту гр. БПИ-22-2 Козлову Данилу Дмитриевичу

1 Тема работы: Разработка программы

2 Срок сдачи работы: 13 июля 2024 г.

3 Содержание задания

3.1 Решение практических задач

Задача 1.

Используя произвольные язык программирования и среду разработки создайте программу, которая отображает на экране монитора график кривой или поверхности (в соответствии с вариантом задания № 54 , Кривая Пеано) в декартовой и полярной системах координат с центром в центре экрана монитора (окна или иной прямоугольной области экрана). При изменении размеров окна, график и все его атрибуты (координатная сетка, метки на шкале, подписи и т.д.) должны автоматически масштабироваться.

Задача 2.

Используя результаты предыдущего задания (для вариантов с 42 по 60) подготовить реферат на тему «История создания и практическое применение фрактала кривая Пеано»

3.2 Оформление и защита отчета

Отчет по учебной исследовательской практике должен содержать: титульный лист; индивидуальное задание; содержание; цель и задачи; описание выполненных заданий (в соответствии с методическим руководством и требованиями преподавателя); выводы и предложения; список использованных источников; графический материал (схемы, графики, технологические карты).

Календарный план-график работ прилагается к отчету отдельным листом.

Требования к оформлению отчетов.

Отчёт предоставляется в печатном виде с выполнением требований нормоконтроля и состоит из следующих разделов:

Введение. Во введении обосновывается цель и задачи прохождения практики.

В разделе 1 выполняется краткий обзор литературы по теме индивидуального задания, подбирается необходимый математический инструментарий (аппарат), производится его анализ и разрабатывается алгоритм решения задачи.

В разделе 2 осуществляется обоснование выбора среды реализации разработанного алгоритма.

В разделе 3 описывается процесс кодирования в выбранной среде и языке программирования и основные элементы управления создаваемым приложением.

В разделе 4 описывается процесс тестированию и отладки программного кода.

Раздел 5 (рекомендуемый) должен содержать краткое иллюстрированное руководство пользователя.

Заключение. В заключении обобщается изложенный в отчёте материал, делаются выводы.

Объем отчёта составляет 20-25 страниц.

Отчёт по практике оформляется в соответствии с «Требованиями к оформлению текстовой части выпускных квалификационных работ, курсовых работ (проектов), рефератов, контрольных работ, отчётов по практикам, лабораторным работам (СК-СТО-ТР04-1.005-2020)». Для оформления графического материала блок-схем, алгоритмов использовать ГОСТ 19.701-90 который распространяется на условные обозначения (символы) в схемах алгоритмов, программ, данных и систем и устанавливает правила выполнения схем, применяемых для отображения различных видов задач обработки данных и средств их решения.

Руководитель
старший преподаватель каф. ИТС _____Е.Г. Лаврушина

Задание получил: _____Д.Д. Козлов

Содержание

Введение

Параметрические кривые в математике представляют собой кривые, которые определяются с помощью параметров. В отличие от явных функций, где зависимость между переменными выражена напрямую, параметрические кривые задаются с помощью одной или нескольких параметрических функций.

В данном отчете будет рассмотрена история открытия параметрической кривой Пеано, её математические свойства, а также приложения в различных областях науки и техники. Особое внимание будет уделено современным исследованиям и разработкам, связанным с кривой Пеано, а также её роли в развитии теории фракталов и топологии [1].

Кривая Пеано, названная в честь итальянского математика Джузеппе Пеано, представляет собой один из первых примеров непрерывной кривой, которая заполняет пространство. Впервые представленная в 1890 году, эта кривая стала важным объектом изучения в теории фракталов и топологии. Кривая Пеано демонстрирует, что возможно построить непрерывную функцию, которая отображает единичный отрезок на единичный квадрат, таким образом, что каждая точка квадрата будет достигнута.

Основная идея кривой Пеано заключается в том, что она заполняет двумерное пространство, несмотря на то, что её параметризация осуществляется через одномерный интервал. Это свойство делает её важным объектом для изучения в различных областях математики и её приложений, включая компьютерную графику, обработку изображений, моделирование и анализ данных.

Кривая Пеано является примером фрактала, что означает, что она обладает самоподобием на различных масштабах. Это свойство позволяет использовать её для моделирования природных объектов и явлений, которые также демонстрируют фрактальные свойства. Например, кривая Пеано может быть использована для моделирования структуры рек, линий берегов, облаков и других природных форм. Если рассматривать эту кривую, как доказательство того, что линию можно представить в виде плоскости, то, несомненно, это будет отличным открытием со стороны самого Пеано.

Кроме того, кривая Пеано имеет важное значение в теории измерений и размерности. Она показывает, что одномерное множество может иметь двумерную меру, что противоречит интуитивному представлению о размерности. Это открытие привело к развитию новых методов и подходов в математике, включая теорию хаоса и динамические системы.

Цель данного отчета заключается в том, чтобы предоставить подробный и всесторонний анализ кривой Пеано, её математических свойств и приложений. Отчет предназначен для студентов, исследователей и специалистов в области математики,

компьютерной графики, обработки изображений и других смежных областей, которые заинтересованы в изучении и применении фрактальных структур и топологических концепций.

1 Анализ технического задания

1.1 Кривая дракона: основные сведения и область применения

Кривая Пеано (рисунок 1) — общее название для параметрических кривых, образ которых содержит квадрат (или, в более общем смысле, открытые области пространства). Другое название — заполняющая пространство кривая.

Названа в честь Джузеппе Пеано (1858—1932), первооткрывателя такого рода кривых, в частном смысле кривой Пеано называется конкретная кривая, которую нашёл Пеано.

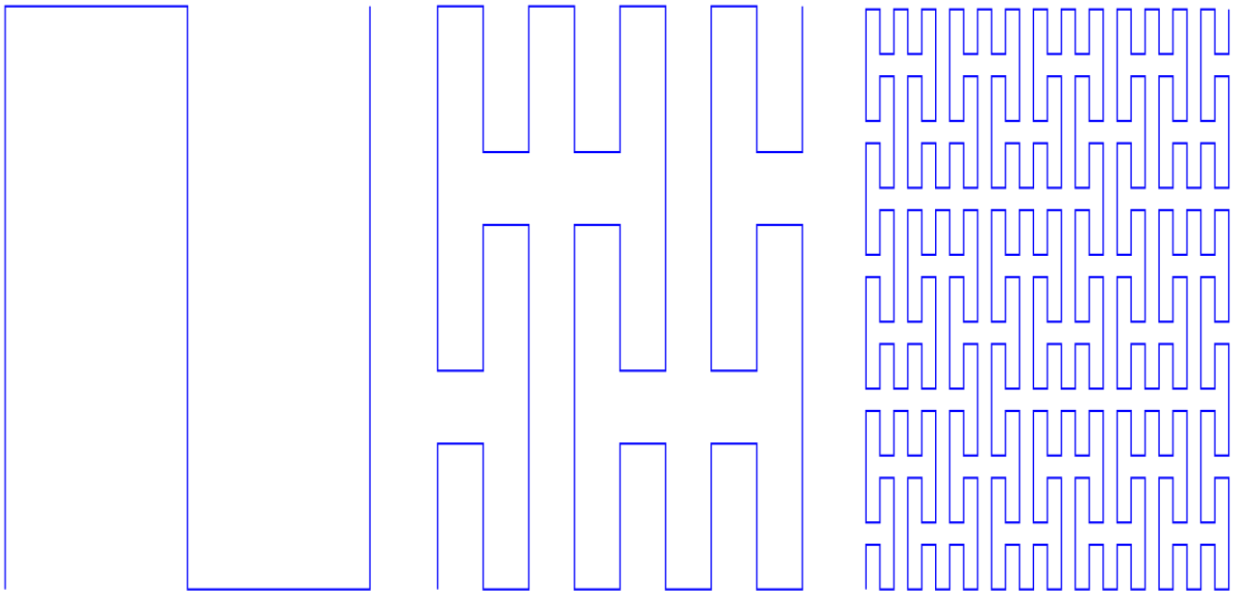


Рисунок 1 – Один из видов прямой Пеано

Джузеппе Пеано (1858-1932) был выдающимся итальянским математиком, известным своими работами в области математической логики, теории множеств и анализа. Он внёс значительный вклад в развитие математической нотации и формализацию математических понятий. Одним из его наиболее известных достижений является создание аксиом Пеано, которые формализуют понятие натуральных чисел [2].

В конце XIX века математики активно исследовали свойства непрерывных функций и их поведение на различных множествах. Одной из ключевых проблем того времени было понимание, как непрерывные функции могут отображать одномерные множества на многомерные. В 1878 году немецкий математик Георг Кантор [3] доказал, что существует биективное соответствие между точками отрезка и точками квадрата, но его доказательство не включало построение непрерывной функции.

В 1890 Пеано открыл непрерывную кривую, ныне называемую кривой Пеано [4], которая проходит через любую точку единичного квадрата. Его целью было построение непрерывного отображения из единичного отрезка в единичный квадрат. Заняться проблемой

Пеано побудил более ранний неожиданный результат Георга Кантора о том, что множество точек единичного интервала имеет ту же мощность, что и множество точек любого конечномерного многообразия, в частности, единичного квадрата. Задача, которую решал Пеано, заключалась в вопросе — может ли быть такое отображение непрерывным, то есть может ли кривая заполнить пространство. Решение Пеано не устанавливает непрерывное взаимно однозначное отображение между единичным интервалом и единичным квадратом, и более того, такого отображения не существует.

Годом позже Давид Гильберт опубликовал в том же журнале другой вариант построения Пеано (рисунок 2) Статья Гильберта была первой статьёй, в которой был помещен рисунок, помогающий представить технику построения. По существу, это был тот же рисунок, что и приведённый здесь. Аналитическая форма кривой Гильберта, однако, существенно сложнее, чем у Пеано.

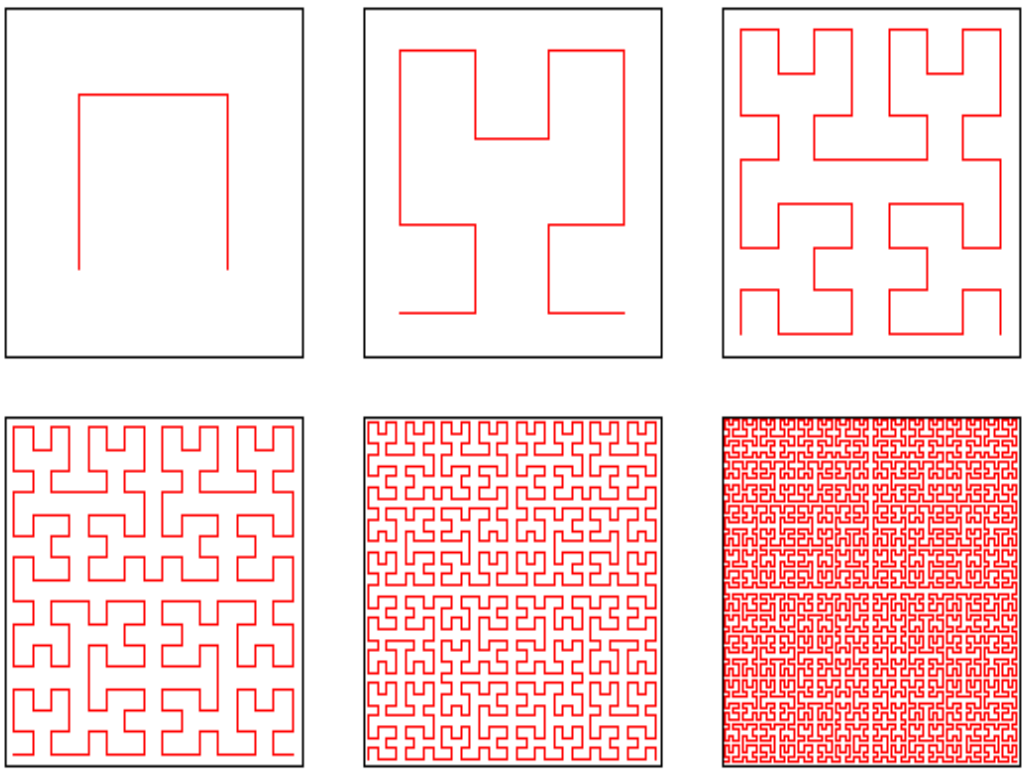


Рисунок 2 – Кривая Пеано построенная Давидом Гильбертом

Как видим, при построении кривой Гилберта, учитывается порядок, но плоскость разбивается на 4, а не на 9 частей.

1.2 Математический аппарат для построения графика

Аналитическое построение кривой Пеано основывается на рекурсивной функции, которая отображает одномерный отрезок на двумерный квадрат. Для этого используется параметрическая функция [5], которая задает координаты точек кривой в зависимости от параметра t на интервале (0,1).

Функция Пеано $P(t)$ может быть определена следующим образом:

– Разделим интервал (0,1) на три равные части: (0,1/3), (1/3,2/3), (2/3,1).

– Для каждой части определим координаты точек кривой в зависимости от t.

Пусть $t \in (0,1)$. Тогда координаты (x, y) точки кривой Пеано могут быть заданы таким образом, как показано на рисунке 3:

$$P(t) = \begin{cases} (3t, 0), & \text{если } t \in [0, \frac{1}{3}] \\ (1, 3t-1), & \text{если } t \in [\frac{1}{3}, \frac{2}{3}] \\ (3-3t, 1), & \text{если } t \in [\frac{2}{3}, 1] \end{cases}$$

Рисунок 3 – Система определения координат точек кривой Пеано

Этот процесс можно продолжить рекурсивно, разделяя каждый интервал на три части и определяя координаты точек для каждой части.

Геометрическое построение кривой Пеано можно описать следующим образом:

Начальный шаг: Начнем с единичного отрезка, который будет разделен на три равные части.

Первый шаг: Разделим отрезок на три равные части и соединим их так, чтобы получился путь, проходящий через все три части.

Рекурсивный шаг: Для каждой из трех частей повторим процесс, разделяя каждую часть на три равные части и соединяя их так, чтобы получился путь, проходящий через все девять частей.

На каждом шаге кривая становится все более детализированной и заполняет все больше пространства. В результате получается кривая, которая заполняет весь квадрат. Кривая Пеано является примером фрактала, который обладает самоподобием на различных масштабах. Аналитическое и геометрическое построение кривой Пеано демонстрирует, как одномерный отрезок может быть отображен на двумерное пространство с помощью непрерывной функции. Этот процесс имеет важное значение в теории фракталов и топологии, а также находит применение в различных областях науки и техники.

1.3 Блок-схема работы программы

Для реализации программы, отрисовывающей кривую Пеано, можно использовать следующую блок-схему (Приложение А).:

1. Инициализация программы. Подготовка необходимых библиотек и функций для дальнейшей работы, инициализация начальных параметров и выбор системы кооринат пользователем, а так же проверка их верности.

2. Проверка порядка и Создание пустого списка для хранения координат точек кривой. Если порядок равен 0, функция возвращает начальную точку (x, y). Использовать аксиому и правила для генерации строки, представляющей последовательность отрезков кривой Пеано.

3. Вызов функции для отрисовки кривой. Функция для построения кривой вызывается с заданным порядком, и возвращается список точек. Извлечение координат x и y: Координаты x и y извлекаются из списка точек. Функция вызывается рекурсивно для меньшего порядка с новыми координатами и уменьшенным размером.

4. Создание фигуры и осей для графика. Создается фигура и оси для отображения графика с помощью функции. Настраиваются параметры фигуры, такие как размер и отступы.

5. Построение начального графика. График строится для начального порядка кривой (например, 3) с помощью функции для отрисовки кривой.

6. Обновление графика при изменении значения слайдера. Устанавливается обработчик событий для слайдера, который вызывает функцию для отрисовки кривой при изменении значения слайдера.

7. Отображение графика. График отображается на экране с помощью функции show библиотеки matplotlib.

8. Завершение. Программа завершает свою работу при нажатии пользователем кнопки завершения работы.

2 Выбор языка программирования и среды разработки

2.1 Языки программирования

Для решения задачи требуется язык программирования Python.

Выбор языка программирования Python для реализации кривой дракона обусловлен несколькими ключевыми преимуществами [6]:

– Богатая Библиотека: Python имеет обширный набор библиотек для математических вычислений и визуализации, таких как Math [7] и Matplotlib [8]. Эти инструменты упрощают создание и отображение сложных геометрических фигур, таких как кривая Пеано.

– Интерактивность: Использование интерактивных сред, таких как Jupyter Notebook, позволяет разработчикам пошагово создавать и визуализировать фракталы, что способствует лучшему пониманию процесса их генерации.

– Простота и Читаемость: Python известен своим чистым и легко читаемым синтаксисом, который делает код доступным даже для начинающих программистов. Это особенно важно при работе с фракталами, так как алгоритмы могут быстро стать сложными. Читаемость кода упрощает отладку и позволяет другим разработчикам легче понять и модифицировать код.

– Мощные Инструменты Визуализации: Python предоставляет мощные инструменты для визуализации, такие как Matplotlib, которые могут быть использованы для создания детализированных и красочных изображений фракталов. Это делает Python идеальным выбором для демонстрации математических концепций через визуальные средства.

– Поддержка Сообщества: Python имеет одно из самых больших и активных сообществ разработчиков. Это означает, что существует множество ресурсов для обучения и поддержки, а также готовые решения и библиотеки, которые можно использовать для ускорения разработки.

– Переносимость и Масштабируемость: Python является кроссплатформенным языком, что позволяет запускать написанные на нем программы на различных операционных системах без изменений в коде. Это делает его идеальным для разработки программ, которые могут быть распространены среди широкой аудитории.

– Производительность: Хотя Python не самый быстрый язык в плане времени выполнения по сравнению с компилируемыми языками, такими как C или C++, он предлагает достаточную производительность для большинства задач визуализации и может быть оптимизирован с помощью специализированных библиотек, таких как Cython.

– Образовательный Потенциал: Python часто используется в образовательных целях для преподавания программирования, математики и информатики. Реализация кривой пеано на

Python может служить отличным учебным примером, демонстрирующим как программирование, так и математические концепты.

В совокупности эти факторы делают Python отличным выбором для реализации кривой дракона и других фрактальных структур, предоставляя удобные инструменты для разработки, тестирования и демонстрации сложных математических фигур и алгоритмов, например, кривой Пеано.

2.2 Среды разработки

Для разработки приложения рассматривались такие интегрированные среды, как:

1. Thonny. Простая и пользовательский IDE, особенно подходящая для начинающих. Thonny предлагает простой интерфейс и базовые функции отладки, что делает её хорошим выбором для тех, кто только начинает изучать программирование [9].

2. «Jupyter Notebook». Это интерактивная среда, которая позволяет выполнять код по частям, что идеально подходит для экспериментов с алгоритмами и визуализацией данных. Jupyter поддерживает визуализацию в реальном времени, что делает его отличным выбором для создания и демонстрации фракталов, таких как кривая Пеано [10].

3. Sublime Text или Atom. Эти текстовые редакторы могут быть настроены как полноценные IDE с помощью плагинов и расширений. Они подходят для тех, кто предпочитает легковесные и быстрые инструменты [11].

4. «Microsoft Visual Studio 2022». Это легковесная, но мощная среда разработки, которая поддерживает Python через расширения. VS Code предлагает широкий спектр функций, включая отладку, управление Git и интеграцию с Docker. Его можно настроить для различных нужд разработки, и он подходит для работы с фрактальной графикой [12].

5. Spyder. Эта среда разработки включена в дистрибутив Anaconda и ориентирована на научные вычисления и анализ данных. Spyder предлагает интегрированный доступ к документации и графический интерфейс для отладки, что может быть полезно при работе с математическими алгоритмами [13].

6. «PyCharm» [14], так как, это одна из самых популярных интегрированных сред разработки (IDE) для Python, предлагающая мощные инструменты для разработки, такие как интеллектуальное завершение кода, поддержка веб-разработки и инструменты для профилирования кода. PyCharm подходит как для начинающих, так и для опытных разработчиков, предоставляя глубокую интеграцию с различными библиотеками Python. К тому же, мой личный опыт пользования этим программным обеспечением, демонстрирует, что именно в PyCharm проще всего писать и тестировать код. В итоге именно эта среда и была выбрана.

3 Разработка программы

3.1 Реализация графического интерфейса

Следующим этапом был реализован графический интерфейс приложения с помощью библиотеки matplotlib и модуля math. Приложение состоит из экрана отображающего декартову систему координат и сам фрактал. Так же есть кнопки управления перемещением по полотну отрисовку и координатам, кнопка уменьшения и увеличения масштаба изображения, а также кнопка для выхода из программы. Ещё присутствует элемент, позволяющий менять количество итераций фигуры для наглядности (рисунок 4).

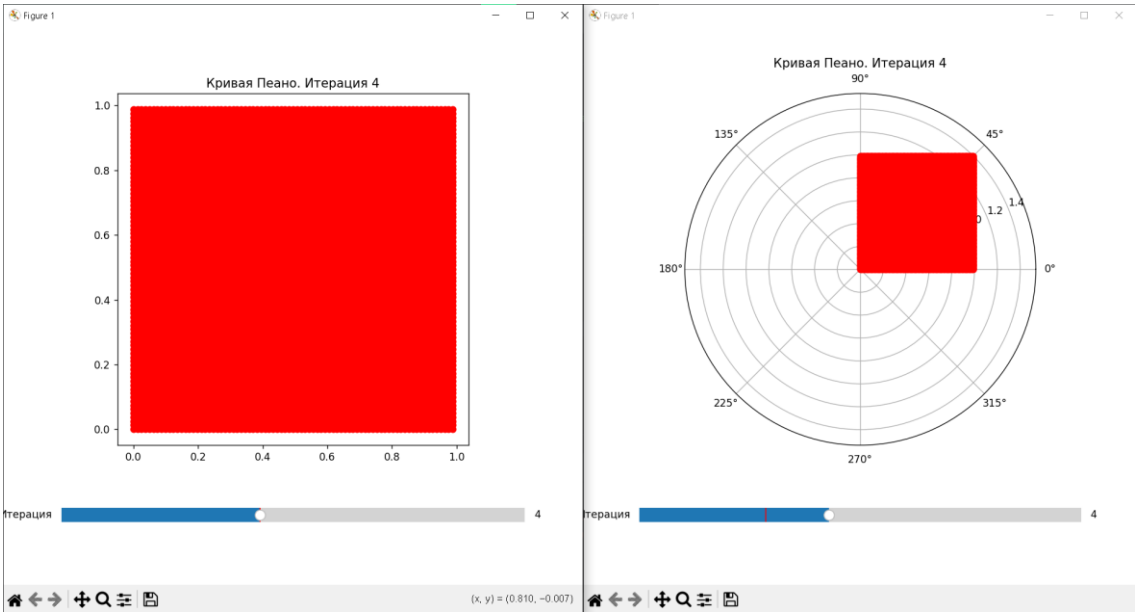


Рисунок 4 – Результат работы программы в двух системах координат

3.2 Реализация программного кода

Определившись с языком программирования и средой разработки, ориентируясь на блок-схему программы по реализации алгоритма, рисующего кривую дракона, напишем код с комментариями [16]. Код каждого из модулей представлен в приложениях Б и В. Полный код представлен в приложении Г.

В декартовой или в полярной, как демонстрируется на рисунке 5.

```
ask = int(input('В какой системе координат вы хотели бы увидеть кривую? 1 - в Декартовой, 2 - в полярной: '))
#Условный оператор для выяснения системы
if ask == 1:
    car_peano()
elif ask == 2:
    polar_peano()
else:
    print('Вы ошиблись, такой системы нет')
```

Рисунок 5 – Оператор проверяющий ввод пользователя

Значит, запуская программу даётся выбор в какой системе выводить график.

При выборе Декартовой системы (то есть, при введении цифры 1), будет реализован код из приложения А.

Разберём подробнее код для реализации кривой в декартовой системе:

Изначально импортируем необходимые библиотеки необходимые для реализации (рисунок 6).

```
import matplotlib.pyplot as plt  # Импортируем библиотеку для построения графиков
from matplotlib.widgets import Slider  # Импортируем слайдер из библиотеки matplotlib
import math  # Импортируем модуль math для математических операций
```

Рисунок 6 – Подключение библиотек math и matplotlib

Прописывается функция, которая инициализирует список точек, тем самым запуская рекурсивный алгоритм. Сама функция возвращает список точек. Эта рекурсивная функция строит кривую Пеано заданного порядка. Если порядок равен 0, функция возвращает начальную точку. В противном случае, размер текущего квадрата делится на 3, и функция рекурсивно вызывает саму себя для каждого из 9 меньших квадратов, добавляя полученные точки в список (рисунок 7).

```
def peano_curve(order, x=0, y=0, size=1):
    # Если порядок равен 0, вернуть начальную точку
    if order == 0:
        return [(x, y)]
    points = []  # Инициализация списка точек
    size /= 3  # Деление размера на 3
    # Цикл по смещениям (dx, dy)
    for dx, dy in [(0, 0), (0, 1), (0, 2), (1, 2), (1, 1), (1, 0), (2, 0), (2, 1), (2, 2)]:
        # Рекурсивный вызов функции для меньшего порядка
        sub_points = peano_curve(order - 1, x + dx * size, y + dy * size, size)
        points.extend(sub_points)  # Добавление точек в список
    return points  # Возвращение списка точек
```

Рисунок 7 – Функция для рекурсии кривой

Она строит новую кривую и обновляет заголовок графика, как показано на рисунке 8.

```
def plot_peano_curve(ax, order):
    points = peano_curve(order)  # Вызов функции построения кривой
    x, y = zip(*points)  # Извлечение координат x и y
    ax.clear()  # Очистка графика
    ax.plot(x, y, marker='o', color='red')  # Построение графика, задаём цвет
    ax.set_title(f'Кривая Пеано. Итерация {order}')  # Настройка заголовка графика
    ax.set_aspect('equal')
    plt.draw()  # Обновление графика
```

Рисунок 8 – Функция обновления кривой

Эта функция строит кривую Пеано заданного порядка на переданных осях.

Далее, здесь создается фигура и оси для графика. Размер фигуры устанавливается на 8x8 дюймов, а параметры subplots_adjust позволяют оставить место для слайдера внизу графика. Так же инициируется график сривой с начальной итерацией 4, как показано на рисунке 9.

```
# Создание фигуры и осей
fig, ax = plt.subplots(figsize=(8, 8))
plt.subplots_adjust(left=0.1, bottom=0.25)
# Начальный график
plot_peano_curve(ax, order: 4)
ax_slider = plt.axes( arg: [0.1, 0.1, 0.8, 0.05], facecolor='lightgoldenrodyellow')
slider = Slider(ax_slider, label: 'Итерация', valmin: 1, valmax: 8, valinit=4, valstep=1)
# Обновление графика при изменении значения слайдера
slider.on_changed(lambda val: plot_peano_curve(ax, int(val)))
```

Рисунок 9 – Отображение слайдера и самй программы

Здесь создается слайдер для изменения порядка кривой Пеано. Параметры axes задают положение и размер слайдера, а Slider создает сам слайдер с диапазоном значений от 1 до 8, начальным значением 4 и шагом 1.

Теперь рассмотрим работу модуля из приложения В, то есть, если пользователь выбрал полярную систему координат. При введении числа «2», первым делом импортируются необходимые библиотеки, такие как math и matplotlib (рисунок 10):

```
import matplotlib.pyplot as plt  # Импортируем библиотеку для построения графиков
from matplotlib.widgets import Slider  # Импортируем слайдер из библиотеки matplotlib
import math  # Импортируем модуль math для математических операций
```

Рисунок 10

Далее, по подобию, декартовой системы, вставляется функция, которая инициализирует список точек, тем самым запуская рекурсивный алгоритм (рисунок 11).

```
def peano_curve(order, x=0, y=0, size=1):
    if order == 0:
        return [(x, y)]  # Возвращаем начальную точку, если порядок равен 0
    points = []  # Список для хранения точек кривой
    size /= 3  # Уменьшаем размер на 3 для следующего уровня рекурсии
    for dx, dy in [(0, 0), (0, 1), (0, 2), (1, 2), (1, 1), (1, 0), (2, 0), (2, 1), (2, 2)]:
        sub_points = peano_curve(order - 1, x + dx * size, y + dy * size, size)  # Рекурсивно строим подмножества точек
        points.extend(sub_points)  # Добавляем подмножества точек в общий список
    return points  # Возвращаем список точек
```

Рисунок 11 – Функция peano_curve()

Вышеприведенный код позволяет быстро реализовать рекурсивный алгоритм обработки координат начальной точки.

В следующей функции описано преобразование координат в полярную систему

```
def plot_peano_curve_polar(ax, order):
    points = peano_curve(order)  # Получаем точки кривой Пеано заданного порядка
    x, y = zip(*points)  # Разделяем координаты x и y
    # Преобразование координат в полярную систему
    r = [math.sqrt(xi ** 2 + yi ** 2) for xi, yi in zip(x, y)]  # Вычисляем радиус для каждой точки
    theta = [math.atan2(yi, xi) for xi, yi in zip(x, y)]  # Вычисляем угол для каждой точки
    ax.clear()  # Очищаем оси
    ax.plot(theta, r, marker='o', color='red')  # Строим график в полярной системе координат
    ax.set_title(f'Кривая Пеано. Итерация {order}')  # Устанавливаем заголовок графика
    ax.set_ylim(0, max(r) * 1.1)  # Центрируем кривую
    ax.set_aspect('equal')
    plt.draw()  # Обновляем график
```

Рисунок 12 – Функция отрисовки кривой в полярных координатах

По аналогии с декартовой системой в которой создаётся сама фигура вместе с осями координат, в полярной происходит ровно то же самое с поправкой на расположение слайдера и самой кривой, это отображено на рисунке 13.

```
fig, ax = plt.subplots(figsize=(8, 8), subplot_kw={'projection': 'polar'})
plt.subplots_adjust(left=0.1, bottom=0.25)  # Настраиваем отступы
# Начальный график
plot_peano_curve_polar(ax, order: 3)  # Строим кривую Пеано порядка 3
ax_slider = plt.axes( arg: [0.1, 0.1, 0.8, 0.05], facecolor='lightgoldenrodyellow')
slider = Slider(ax_slider, label: 'Итерация', valmin: 1, valmax: 8, valinit=3, valstep=1)
# Обновление графика при изменении значения слайдера
slider.on_changed(lambda val: plot_peano_curve_polar(ax, int(val)))  # Добавляем обработчик изменения значения слайдера
plt.show()
```

Рисунок 13 – Функция для отрисовки кривой в полярной системе координат

На рисунке 13 показан финальный этап работы программы, а именно – изменение кривой при изменении слайдера и отображение графика.

В программе по отрисовке дракона использовался условный оператор, который считывал ответ пользователя на вопрос в какой системе он хочет видеть график (рисунок 14).

```
ask = int(input('В какой системе координат вы хотели бы увидеть кривую? 1 - в Декартовой, 2 - в полярной: '))
#Условный оператор для выяснения системы
if ask == 1:
    car_peano()
elif ask == 2:
    polar_peano()
else:
    print('Вы ошиблись, такой системы нет')
```

Рисунок 14 – Запрос пользователю

Таким образом, при помощи интегрированной среды разработки, языка программирования python и дополнительных библиотек, была реализована программа по отображению кривой Пеано.

4 Тестирование программы

4.1 Проверка работы программы при вводе корректных данных

При вводе верных значений системы координат, получаем правильную работу программы (рисунок 17).

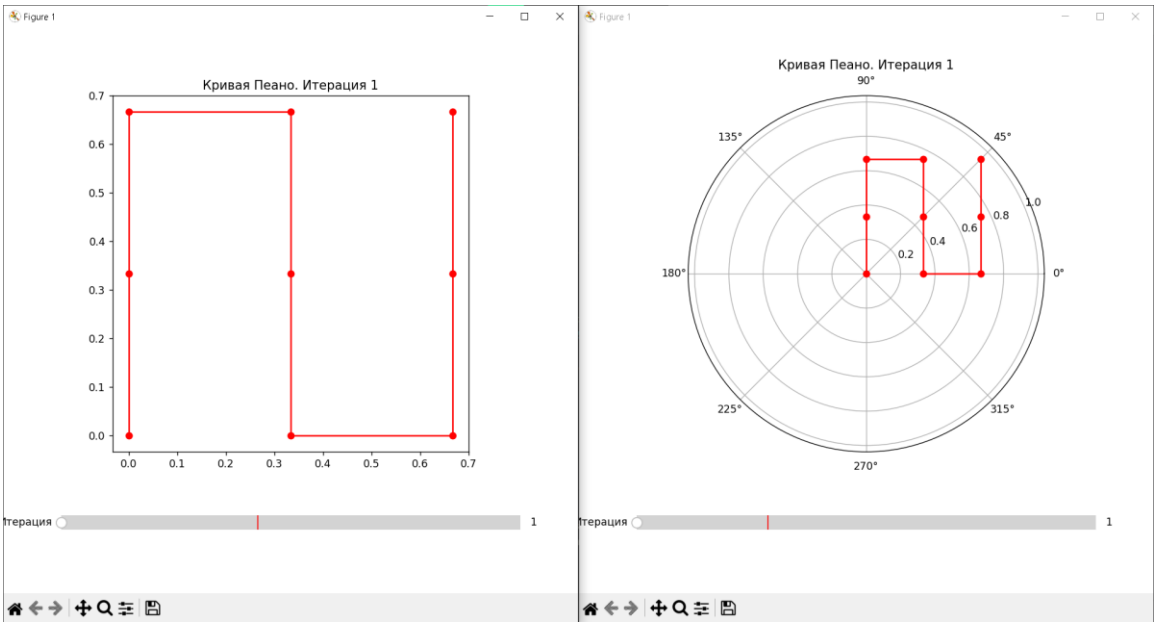


Рисунок 15 – Первая итерация кривой Пеано

4.2 Проверка работы программы при вводе корректных данных

При введении некорректных данных, программа говорит, что такой системы нет и в помине, на рисунке 16 это наглядно показано.

```
В какой системе координат вы хотели бы увидеть кривую? 1 - в Декартовой, 2 - в полярной: 3
Вы ошиблись, такой системы нет
```

Рисунок 16 – Некорректная работа приложения

Таким образом, если пользователь ошибётся с выбором системы, ему выдастся ошибка и предупреждение о том, что нет такой системы.

Заключение

В ходе прохождения учебно-ознакомительной практики, я получил глубокое понимание этой уникальной математической конструкции. Кривая Пеано, впервые описанная Джузеппе Пеано в 1890 году, является одним из первых примеров фракталов и представляет собой непрерывную кривую, заполняющую пространство. Она демонстрирует, как можно пройти через каждую точку двумерного пространства, не пересекаясь и не оставляя пробелов.

Я узнал о значении кривой Пеано в истории математики и её роли в развитии теории фракталов и топологии. Кривая Пеано является важным примером, иллюстрирующим концепцию заполняющих пространство кривых. Я изучил различные методы построения кривой Пеано, включая итеративные и рекурсивные подходы. Это позволило мне понять, как небольшие изменения на каждом шаге могут привести к созданию сложных и красивых структур. В процессе написания кода для построения кривой Пеано я улучшил свои навыки программирования на Python, особенно в использовании библиотек matplotlib и math для визуализации и интерактивного управления графиками.

Кривая Пеано и другие фракталы широко используются в компьютерной графике для создания сложных и реалистичных изображений, таких как текстуры и ландшафты. Методы, основанные на кривой Пеано, применяются в алгоритмах сжатия и обработки изображений, что позволяет эффективно кодировать и передавать визуальную информацию. Кривая Пеано используется в моделировании физических процессов и систем, где требуется равномерное покрытие пространства, например, в моделировании потоков жидкости или распределения тепла.

Изучение кривой Пеано позволило мне глубже понять природу фракталов и их свойства. Я осознал, как простые правила могут приводить к созданию сложных и самоподобных структур. Написание кода для построения кривой Пеано помогло мне улучшить навыки программирования, особенно в области рекурсивных алгоритмов и визуализации данных. Использование интерактивных инструментов, таких как слайдеры, позволило мне создать более наглядные и удобные для пользователя приложения, что является важным аспектом в разработке программного обеспечения.

В заключение, исследование кривой Пеано и её практического применения показало, что эта математическая конструкция имеет широкое применение в различных областях науки и техники. Она не только является интересным объектом для изучения, но и имеет важное практическое значение. Этот опыт обогатил мои знания и навыки, а также вдохновил на дальнейшие исследования в области фракталов и их применения.

Список использованных источников

1. Что такое кривая Пеано // Открытый интернет-блог «Obuchonok» [сайт] – URL: https://obuchonok.ru/node/1670 (дата обращения: 17.06.2024)

2. Что такое фракталы и как они устроены // Информационный портал «Techinsider» [сайт] – URL: https://www.techinsider.ru/science/8906-krasota-povtora-fraktaly/ (дата обращения: 17.06.2024)

3. Идеи Кантора // Информационный портал «Habr» [сайт] – URL: https://habr.com/ru/companies/itglobalcom/articles/751676/ (дата обращения: 17.06.2024)

4. Что такое линия. Кривая Пеано // Информационный портал «Statum» [сайт] – URL: https://stratum.ac.ru/education/textbooks/kgrafic/additional/addit32.html (дата обращения: 17.06.2024)

5. Как задаётся параметрическая функция //Всероссийский математический портал «WebMath.ru» [сайт] – URL: https://www.webmath.ru/poleznoe/formules_8_15.php (дата обращения: 17.06.2024)

6. Почему Python // Информационный портал «Tproger» [сайт] – URL: https://tproger.ru/articles/pochemu-python-takoj-populyarnyj-napisano-chelovekom (дата обращения: 23.06.2024)

7. Официальная документация модуля math// Официальный сайт «Python» [сайт] – URL: https://docs.python.org/3/library/math.html (дата обращения: 23.06.2024)

8. Официальная документация и примеры использования библиотеки Matplotlib // Официальный сайт «Matplotlib» [сайт] – URL: https://matplotlib.org/stable/gallery/index.html (дата обращения: 23.06.2024)

9. Официальный репозиторий и документация Thonny // Информационный ресурс «Github» [сайт] – URL: https://github.com/thonny/thonny (дата обращения: 30.06.2024)

10. Преимущества Jupyter Notebook // Информационный портал «Яндекс Практикум» [сайт] – URL: https://practicum.yandex.ru/blog/chto-takoe-jupyter-notebook (дата обращения: 30.06.2024)

11. Нестандартные решения Sublime Text и Atom // Информационный блог «Skillfactory» [сайт] – URL: https://blog.skillfactory.ru/glossary/sublime-text/ (дата обращения: 30.06.2024)

12. Для чего стоит использовать Visual Studio // Официальный сайт Microsoft Visual Studio [сайт] – URL: https://www.microsoft.com/ru-ru/microsoft-365/business/copilot-for-microsoft-365 (дата обращения: 30.06.2024)

13. Почему Spyder лучший // Информационный блог «Континент свободы» [сайт] – URL: https://xn--90abhbolvbbfgb9aje4m.xn--p1ai/spyder-ide-sreda-programmirovaniya-python/ (дата обращения: 30.06.2024)

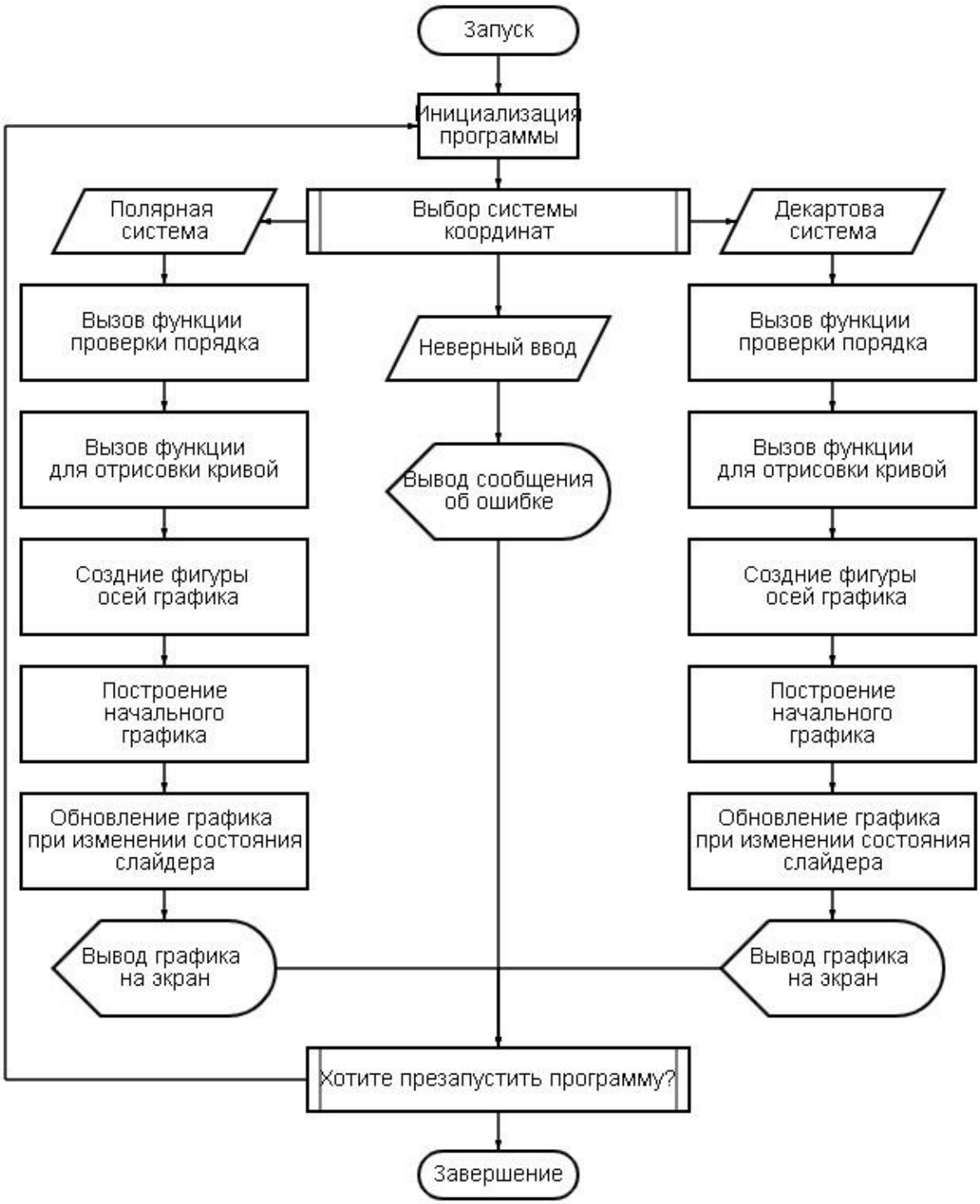
14. Почему Pycharm популярен // Информационный ресурс «Skillbox» – URL: https://skillbox.ru/media/code/pycharm-kak-eye-ustanovit-i-ispolzovat/ (дата обращения: 11.07.2024)

15. Как IDE помогает в управлении проектами // Информационный портал «Atlassian» [сайт] – URL: https://www.atlassian.com/ru/work-management/project-management (дата обращения: 11.07.2024)

16. Алгоритмы построения фракталов // Информационный портал – URL: «hmath» https://hmath.spbstu.ru/userfiles/files/documents/conference/2018/11 (дата обращения: 11.07.2024)

Приложение А

Блок-схема работы программы



Приложение Б

Исходный код программы (отрисовка в декартовой системе координат)

```
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider
import math

def peano_curve(order, x=0, y=0, size=1):
    # Если порядок равен 0, вернуть начальную точку
    if order == 0:
        return [(x, y)]

    points = []  # Инициализация списка точек
    size /= 3  # Деление размера на 3
    # Цикл по смещениям (dx, dy)
    for dx, dy in [(0, 0), (0, 1), (0, 2), (1, 2), (1, 1), (1, 0), (2, 0), (2, 1), (2, 2)]:
        # Рекурсивный вызов функции для меньшего порядка
        sub_points = peano_curve(order - 1, x + dx * size, y + dy * size, size)
        points.extend(sub_points)  # Добавление точек в список

    return points  # Возвращение списка точек


def plot_peano_curve(ax, order):
    points = peano_curve(order)  # Вызов функции построения кривой
    x, y = zip(*points)  # Извлечение координат x и y
    ax.clear()  # Очистка графика
    ax.plot(x, y, marker='o', color='red')  # Построение графика, задаём цвет
    ax.set_title(f'Peano Curve of Order {order}')  # Настройка заголовка графика
    plt.draw()  # Обновление графика


# Создание фигуры и осей
fig, ax = plt.subplots(figsize=(8, 8))
plt.subplots_adjust(left=0.1, bottom=0.25)

# Начальный график
plot_peano_curve(ax, 3)

# Создание слайдера
ax_slider = plt.axes([0.1, 0.1, 0.8, 0.05], facecolor='lightgoldenrodyellow')
slider = Slider(ax_slider, 'Order', 1, 8, valinit=3, valstep=1)

# Обновление графика при изменении значения слайдера
slider.on_changed(lambda val: plot_peano_curve(ax, int(val)))

plt.show()  # Отображение графика
```

Приложение В

Исходный код программы (отрисовка в полярной системе координат)

```
import matplotlib.pyplot as plt  # Импортируем библиотеку для построения графиков
from matplotlib.widgets import Slider  # Импортируем слайдер из библиотеки matplotlib
import math  # Импортируем модуль math для математических операций

def peano_curve(order, x=0, y=0, size=1):
    if order == 0:
        return [(x, y)]  # Возвращаем начальную точку, если порядок равен 0
    points = []  # Список для хранения точек кривой
    size /= 3  # Уменьшаем размер на 3 для следующего уровня рекурсии
    for dx, dy in [(0, 0), (0, 1), (0, 2), (1, 2), (1, 1), (1, 0), (2, 0), (2, 1), (2, 2)]:
        sub_points = peano_curve(order - 1, x + dx * size, y + dy * size, size)  # Рекурсивно строим
подмножества точек
        points.extend(sub_points)  # Добавляем подмножества точек в общий список
    return points  # Возвращаем список точек

def plot_peano_curve_polar(ax, order):
    points = peano_curve(order)  # Получаем точки кривой Пеано заданного порядка
    x, y = zip(*points)  # Разделяем координаты x и y

    # Преобразование координат в полярную систему
    r = [math.sqrt(xi ** 2 + yi ** 2) for xi, yi in zip(x, y)]  # Вычисляем радиус для каждой точки
    theta = [math.atan2(yi, xi) for xi, yi in zip(x, y)]  # Вычисляем угол для каждой точки
    ax.clear()  # Очищаем оси
    ax.plot(theta, r, marker='o', color='red')  # Строим график в полярной системе координат
    ax.set_title(f'Кривая Пеано. Итерация {order}')  # Устанавливаем заголовок графика
    ax.set_ylim(0, max(r) * 1.1)  # Центрируем кривую
    ax.set_aspect('equal')
    plt.draw()  # Обновляем график

# Создание фигуры и осей
fig, ax = plt.subplots(figsize=(8, 8), subplot_kw={'projection': 'polar'})
plt.subplots_adjust(left=0.1, bottom=0.25)  # Настраиваем отступы
# Начальный график
plot_peano_curve_polar(ax, 3)  # Строим кривую Пеано порядка 3
ax_slider = plt.axes([0.1, 0.1, 0.8, 0.05], facecolor='lightgoldenrodyellow')
slider = Slider(ax_slider, 'Итерация', 1, 8, valinit=3, valstep=1)
# Обновление графика при изменении значения слайдера
slider.on_changed(lambda val: plot_peano_curve_polar(ax, int(val)))  # Добавляем обработчик
изменения значения слайдера

plt.show()
```

Приложение Г

```
def polar_peano():
    import matplotlib.pyplot as plt  # Импортируем библиотеку для построения графиков
    from matplotlib.widgets import Slider  # Импортируем слайдер из библиотеки matplotlib
    import math  # Импортируем модуль math для математических операций

    def peano_curve(order, x=0, y=0, size=1):
        if order == 0:
            return [(x, y)]  # Возвращаем начальную точку, если порядок равен 0
        points = []  # Список для хранения точек кривой
        size /= 3  # Уменьшаем размер на 3 для следующего уровня рекурсии
        for dx, dy in [(0, 0), (0, 1), (0, 2), (1, 2), (1, 1), (1, 0), (2, 0), (2, 1), (2, 2)]:
            sub_points = peano_curve(order - 1, x + dx * size, y + dy * size, size)  # Рекурсивно
строим подмножества точек
            points.extend(sub_points)  # Добавляем подмножества точек в общий список
        return points  # Возвращаем список точек

    def plot_peano_curve_polar(ax, order):
        points = peano_curve(order)  # Получаем точки кривой Пеано заданного порядка
        x, y = zip(*points)  # Разделяем координаты x и y
        # Преобразование координат в полярную систему
        r = [math.sqrt(xi ** 2 + yi ** 2) for xi, yi in zip(x, y)]  # Вычисляем радиус для каждой
точки
        theta = [math.atan2(yi, xi) for xi, yi in zip(x, y)]  # Вычисляем угол для каждой точки
        ax.clear()  # Очищаем оси
        ax.plot(theta, r, marker='o', color='red')  # Строим график в полярной системе координат
        ax.set_title(f'Кривая Пеано. Итерация {order}')  # Устанавливаем заголовок графика
        ax.set_ylim(0, max(r) * 1.1)  # Центрируем кривую
        ax.set_aspect('equal')
        plt.draw()  # Обновляем график

    # Создание фигуры и осей
    fig, ax = plt.subplots(figsize=(8, 8), subplot_kw={'projection': 'polar'})
    plt.subplots_adjust(left=0.1, bottom=0.25)  # Настраиваем отступы
    # Начальный график
    plot_peano_curve_polar(ax, 3)  # Строим кривую Пеано порядка 3
    ax_slider = plt.axes([0.1, 0.1, 0.8, 0.05], facecolor='lightgoldenrodyellow')
    slider = Slider(ax_slider, 'Итерация', 1, 8, valinit=3, valstep=1)
    # Обновление графика при изменении значения слайдера
    slider.on_changed(lambda val: plot_peano_curve_polar(ax, int(val)))  # Добавляем обработчик
изменения значения слайдера

    plt.show()

def car_peano():
    import matplotlib.pyplot as plt
    from matplotlib.widgets import Slider
    import math

    def peano_curve(order, x=0, y=0, size=1):
        # Если порядок равен 0, вернуть начальную точку
```

```
        if order == 0:
            return [(x, y)]
        points = []  # Инициализация списка точек
        size /= 3  # Деление размера на 3
        # Цикл по смещениям (dx, dy)
        for dx, dy in [(0, 0), (0, 1), (0, 2), (1, 2), (1, 1), (1, 0), (2, 0), (2, 1), (2, 2)]:
            # Рекурсивный вызов функции для меньшего порядка
            sub_points = peano_curve(order - 1, x + dx * size, y + dy * size, size)
            points.extend(sub_points)  # Добавление точек в список
        return points  # Возвращение списка точек

    def plot_peano_curve(ax, order):
        points = peano_curve(order)  # Вызов функции построения кривой
        x, y = zip(*points)  # Извлечение координат x и y
        ax.clear()  # Очистка графика
        ax.plot(x, y, marker='o', color='red')  # Построение графика, задаём цвет
        ax.set_title(f'Кривая Пеано. Итерация {order}')  # Настройка заголовка графика
        ax.set_aspect('equal')
        plt.draw()  # Обновление графика

    # Создание фигуры и осей
    fig, ax = plt.subplots(figsize=(8, 8))
    plt.subplots_adjust(left=0.1, bottom=0.25)
    # Начальный график
    plot_peano_curve(ax, 4)
    ax_slider = plt.axes([0.1, 0.1, 0.8, 0.05], facecolor='lightgoldenrodyellow')
    slider = Slider(ax_slider, 'Итерация', 1, 8, valinit=4, valstep=1)
    # Обновление графика при изменении значения слайдера
    slider.on_changed(lambda val: plot_peano_curve(ax, int(val)))

    plt.show()

ask = int(input('В какой системе координат вы хотели бы увидеть кривую? 1 - в Декартовой, 2
- в полярной: '))
#Условный оператор для выяснения системы
if ask == 1:
    car_peano()
elif ask == 2:
    polar_peano()
else:
    print('Вы ошиблись, такой системы нет')
```

КАЛЕНДАРНЫЙ ПЛАН-ГРАФИК (ДНЕВНИК)

прохождения учебной ознакомительной практики

студента ВВГУ

Студент Козлов Данил Дмитриевич направляется для прохождения **учебной практики по получению навыков исследовательской работы**.

С 10 июня 2024 г. по 13 июля 2024 г.

№ п/п	Содержание выполняемых работ по программе	Сроки выполнения		Заключение и оценка руководителя или консультанта	Подпись руководителя или консультанта
		Начало	Окончание		
1	Анализ поставленной задачи	10.06.2024	20.06.2024	отл.	
2	Сбор и анализ информации	21.06.2024	23.06.2024	отл	
3	Разработка решения поставленных задач	24.06.2024	01.07.2024	хор.	
4	Оформление отчета по практике	02.07.2024	13.07.2024	хор.	

Согласовано:

Студент-практикант_____ Д.Д. Козлов

подпись

дата

Руководитель от кафедры _____ Е.Г. Лаврушина

подпись

дата