Государственное бюджетное общеобразовательное учреждение города Москвы “Школа №67”

“Приложение - графический калькулятор на основе Python”

Выполнил: Васильев Тимофей Константинович
Принял: Поликарпов Александр Дмитриевич

1. Введение

1.1 Вступление

В современном мире разработка программного обеспечения становится все более востребованной и динамичной областью. Проект "Графический калькулятор" представляет собой разработку интерактивного приложения, которое позволяет пользователю визуализировать математические функции и выполнять расчеты с использованием графического интерфейса. Данный калькулятор предоставляет пользователям возможность вводить математические выражения, строить графики функций и исследовать их поведение в реальном времени. На сегодняшний день среди как учеников школ, так и преподавателей становится все более востребованной потребность в простых, удобных и эффективных вычислительных программах. Одним из видов таких программ как раз и являются графические калькуляторы - визуализаторы математических и геометрических данных. В данном проекте мы научимся работать с новыми библиотеками в языке программирования Python, которые позволят визуализировать такие данные, а также помогут в разработке интерфейса.

1.2 Цель проекта

Основная цель проекта — создать инструмент, который упростит процесс изучения математических функций и их графического представления. В нем должны присутствовать параметры, которые помогут упростить в образовательном процессе и смогут отличить наш собственный продукт от конкурентов. Используя PySide6 для создания рабочей среды и Manim для обработки данных, пользователи смогут облегчить себе работу с графическими и математическими данными.

1.3 Задачи проекта

- **Создание пользовательского интерфейса для ввода математических выражений**
  Разработать интуитивно понятный интерфейс, где пользователи смогут вводить математические выражения в удобной форме.

Интерфейс должен поддерживать базовые и сложные операции, такие как экспоненты, прямые, параболы/гиперболы и т. д.

- **Реализация механизма для построения графиков функций**
  Используя библиотеки Python PySide6 и Manim, реализовать возможность визуализации графиков по введенным функциям.
- **Поддержка динамического обновления графиков**
  Обеспечить возможность изменения параметров функции и моментального обновления графика без необходимости перезагрузки программы. Это позволит пользователю исследовать поведение функции при изменении переменных.
- **Тестирование и оптимизация производительности**
  Провести тестирование программы для выявления и устранения неисправностей. Оптимизировать производительность при работе с большими объемами данных или сложными функциями, чтобы обеспечить плавность и быстроту работы приложения.

1.4 Актуальность проекта

Можно с уверенностью сказать, что проект на сегодняшний день сможет пользоваться спросом среди как учеников, так и преподавателей, так как и тем и другим необходим удобный и простой инструмент для решения тех или иных задач, в основном связанных с алгеброй, матанализом, геометрией и т.д.

1.5 Практическая применимость

Если кому то из учеников или преподавателей будет необходим графический калькулятор, работающий с анимациями и не требующий подключения к интернету, то данная программа должна быть одним из самых наилучших решений. Она позволит централизовать образовательный процесс и решение математических задач.

1.6 Новизна проекта

- **Интерактивность и динамичность**
  В отличие от большинства традиционных калькуляторов, которые предлагают статические расчеты, проект предлагает динамическое обновление графиков в реальном времени. Пользователь может изменять параметры функций и моментально видеть, как это влияет на график. Это делает изучение математики и визуализацию данных

более интерактивными и понятными.

- **Совмещение вычислительных возможностей Python и графической визуализации**
  Проект использует возможности Python, одного из наиболее популярных языков программирования, что делает его мощным и гибким инструментом. За счет применения библиотеки Manim, калькулятор обеспечивает качественную графическую визуализацию, доступную на уровне профессиональных инструментов.
- **Простота и доступность**
  Графический калькулятор будет разработан с акцентом на простоту использования, что делает его доступным для пользователей с любым уровнем подготовки. Это особенно важно для студентов, преподавателей и исследователей, которым требуются мощные, но простые в использовании инструменты.
- **Открытость и расширяемость**
  Благодаря использованию Python, проект легко расширяем. Пользователи смогут добавлять собственные функции, расширять возможности визуализации и даже интегрировать калькулятор в свои исследовательские проекты или учебные курсы.

2. Анализ предметной области

2.1 Введение

Как было сказано выше, наш графический калькулятор представляет из себя вычислительный инструмент на основе Python и его библиотек - PySide6 для интерфейса и Manim для построения графиков. Он рассчитан на простоту использования, а целевой аудиторией будут являться ученики и преподаватели средних и старших школ.

2.2 Основные характеристики

Графический калькулятор напрямую связан с математическими расчетами и графическим представлением различных типов функций. Основные аспекты математики, которые будут задействованы:

- **Арифметические операции**: выполнение базовых операций (сложение, вычитание, умножение, деление), возведение в степень, вычисление корней.
- **Алгебраические выражения**: поддержка более сложных выражений, содержащих многочлены, дробные выражения.
- **Экспоненциальные функции**: для построения и вычисления различных экспоненциальных зависимостей.

2.3 Основные преимущества

Одной из ключевых особенностей проекта является способность строить графики функций на основе введенных пользователем выражений. Графическая визуализация играет важную роль в понимании поведения математических функций. Основными преимуществами проекта в данной сфере являются:

- **Manim для построения графиков**: одна из самых удобных библиотек для построения математических моделей, которая в данном случае будет использоваться для создания графических представлений данных.
- **Интерактивность**: пользователи должны иметь возможность изменять параметры графиков и исследовать поведение функций в реальном времени.
- **Многозадачность:** пользователям необходима расширенная палитра инструментов для работы с математическими данными, что сможет предоставить данный проект

2.4 Требования к интерфейсу и разработке

- интуитивно понятный интерфейс
- максимальное сокращение кода в не нужных областях для оптимизации работы приложения
- использование исключительно самых известных и самых задокументированных библиотек для разработки, так как это может сильно облегчить задачу
- выявить и устранить ошибки, связанные как с разработанным интерфейсом, так и с построением графиков функций

3. Техническое задание

3.1 Общие сведения

В техническом задании мы подробно опишем процесс разработки данного приложения, выявим требования которые необходимы для оптимизации процесса и проведем тестирование на выявление ошибок.

3.2 Структура разработки

1. Планирование:
    - определение целей и требований проекта
    - анализ предметной области и конкурентов
2. Проектирование:
    - разработка архитектуры приложения
    - создание пользовательского интерфейса
    - выбор технологий и инструментов разработки
    - написание технического задания
3. Разработка:
    - создание основных модулей приложения
    - реализация функционала построения графиков
    - реализация связи между интерфейсом и функционалом
4. Тестирование:
    - планирование и проведение функционального тестирования
    - проверка производительности и стабильности приложения
    - отладка и исправление найденных неисправностей
5. Внедрение и доработка функций:
    - сбор обратной связи пользователей
    - доработка и внедрение новых функций на ее основе
6. Сопровождение и дальнейшая поддержка программы:
    - обновление и поддержка приложения после выпуска
    - мониторинг производительности, обратной связи и популярности среди клиентов
    - решение возникающих проблем и добавление нового функционала

3.3 Функциональные требования

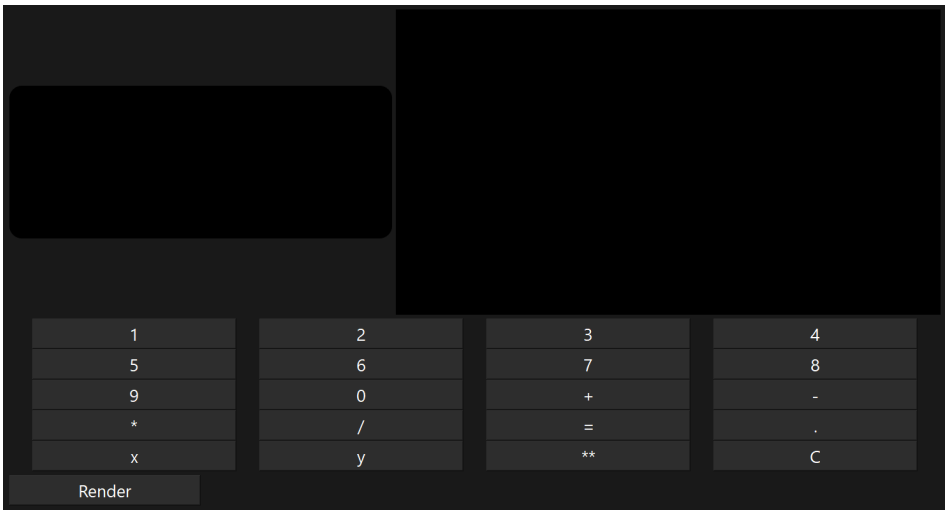
- распознавать введенные символы с виртуальной клавиатуры
- распознавать введенные математические формулы
- заносить в отдельный текстовой файл введенные формулы для дальнейшей их обработки
- обрабатывать формулы для создания анимаций и вывода их на второй экран

3.4 Нефункциональные требования и требования к интерфейсу

- интуитивно понятный интерфейс (цвет фона - темно серый, цвета окон ввода и вывода - черные)
- не слишком громоздкий шрифт и цвет текста (был выбран Segoe UI с размером в 20 пунктов, цвет текста - белый)
- окно ввода и окно вывода - ввод формулы и вывод анимации графика соответственно (цвет фона обоих окон - черный)
- при нажатии кнопки “Render” на окно вывода в кратчайшее время должна выводится анимация графика

3.5 Отрисовка макетов

Реализуем макет интерфейса по выявленным требованиям. Для этого будет использована библиотека PySide6, в которой вместо PyQt6 есть больше инструментов для управления разработкой интерфейса. Каждый аспект функционала будет прописан собственноручно, так как это упростит задачу и лишит нас проблем с параметрами CSS и другими вещами. На представленном изображении готовый вид интерфейса:



4. Состав пакета

4.1 Frontend

Frontend отвечает за визуальную составляющую проекта и взаимодействие с пользователем. В этом проекте PySide6 будет использоваться для создания графического интерфейса. Основные элементы frontend:

1. **Главное окно и структура приложения**
   PySide6 будет обеспечивать создание главного окна калькулятора, включая следующие основные компоненты:

   1.1 Текстовое поле для ввода математических выражений — область, где пользователь будет вводить функции и выражения.

   1.2 Кнопки для математических операций — базовые арифметические операции (сложение, вычитание, умножение и деление)

   1.3 Область для построения графиков — здесь будут отображаться визуализированные графики с помощью Manim.

   1.4 Кнопка для построения графика — пользователь вводит выражение и нажимает кнопку для отображения графика.

2. **Интерактивные элементы**

   2.1 Поле ввода и обработка ошибок: должно быть реализовано удобное текстовое поле с функциями валидации, где при вводе некорректных выражений будут показываться ошибки.

   2.2 Интерактивность с графиками: возможность взаимодействовать с графиком, изменение его состава в реальном времени

3. **Визуализация с использованием Manim**

   3.1 Manim будет интегрирован в интерфейс с использованием PySide6, чтобы динамически показывать анимации графиков, изменяющихся в реальном времени.

   3.2 PySide6 будет управлять вызовами функций Manim для

отрисовки графиков и обновления визуализаций по запросу пользователя.

4. **Навигация и настройка графиков**
   Реализация интерактивных панелей для задания параметров построения графических данных

4.2 Backend

Backend отвечает за математические вычисления, построение графиков и взаимодействие между интерфейсом и системой визуализации Manim. В этом проекте backend будет реализован с использованием Manim и других математических библиотек Python.

1. Модуль вычислений:

- Реализация логики для парсинга математических выражений, введенных пользователем. В этом можно использовать функцию eval для преобразования введенных функций в lambda вид.
- Поддержка различных функций: алгебраических, экспоненциальных
- Валидация введенных данных (например, проверка деления на ноль или использования некорректных символов).

2. Интеграция с Manim:

- Backend должен формировать функции, которые будут передаваться в Manim для их визуализации.
- Manim будет отвечать за создание анимаций построения графиков и обновления визуальных данных в реальном времени.
- Backend должен динамически генерировать код для Manim на основе пользовательских данных, чтобы можно было отображать графики любых сложностей.

3. Обработка запросов и обновлений

- Backend будет получать от frontend команды для построения графиков, обработки ошибок ввода и других взаимодействий пользователя.
- При изменении параметров backend должен пересчитывать значения

и передавать их Manim для обновления анимации.

4.3 Базы данных

Так как на данный момент структура проекта максимально проста, особого смысла в введении систем со сложной структурой баз данных, таких как сохранение пользователей, нет. В основном коде фигурирует лишь сохранение введенных данных в отдельный файл для дальнейше обработки, после чего эти данные стираются и перезаписываются с каждым новым использованием программы.

4.4 Ядро

1. **Модуль вычислений**
   Модуль вычислений является ключевым компонентом, отвечающим за парсинг и интерпретацию математических выражений, вводимых пользователем. Он выполняет следующие задачи:
   - Парсинг математических выражений: ядро должно уметь распознавать математические выражения
   - Валидация ввода: система проверяет введенные выражения на корректность (например, синтаксические ошибки, деление на ноль, неподдерживаемые символы и т.д.).
   - Численные расчеты: после парсинга выражения ядро выполняет необходимые вычисления для получения значений функции в заданных диапазонах.
2. **Модуль визуализации с использованием Manim**
   Ядро также включает в себя интеграцию с библиотекой Manim, которая отвечает за построение и анимацию графиков. Этот модуль управляет следующими процессами:
   - Подготовка данных для Manim: после вычисления функции, ядро передает данные в Manim, где строится визуализация графиков. Manim отвечает за создание анимаций и рендеринг графиков в интерактивной форме.
   - Обновление визуализации: при изменении параметров, ядро вызывает перерисовку графиков с помощью Manim, обеспечивая динамическое обновление графиков.
3. **Модуль взаимодействия с интерфейсом PySide6**
   Этот модуль отвечает за интеграцию с фронтендом и управление пользовательским интерфейсом:

- Передача данных в интерфейс: после обработки выражений и построения графиков, ядро передает результат обратно в интерфейс PySide6, чтобы пользователи могли видеть визуализацию.
- Обработка взаимодействия пользователя: если пользователь меняет какие-либо параметры через интерфейс, этот модуль обрабатывает запрос и передает его на дальнейшую обработку в вычислительный модуль и модуль визуализации.
- Синхронизация с интерфейсом: ядро должно поддерживать постоянную связь между расчетами, визуализацией и пользовательским интерфейсом для обеспечения быстрого отклика и интерактивности.

Логика взаимодействия компонентов:

1. **Пользовательский ввод**
   Пользователь вводит математическое выражение в текстовое поле интерфейса (PySide6). Это выражение передается в ядро для обработки.
2. **Парсинг и валидация**
   Модуль вычислений ядра проверяет введенные данные на корректность и выполняет парсинг выражения для дальнейших расчетов.
3. **Численные вычисления**
   После парсинга выражения ядро вычисляет значения функции на основе введенных параметров.
4. **Визуализация графика**
   После получения данных модуль визуализации передает их в Manim для создания графиков. Manim анимирует графики и рендерит их в заданной области интерфейса.
5. **Интерактивное взаимодействие**
   Если пользователь меняет параметры графиков, ядро обновляет вычисления и вызывает перерисовку графика с помощью Manim.

Дополнительные возможности ядра:

Расширяемость
Структура ядра должна быть спроектирована так, чтобы легко добавлять новые математические функции или типы графиков

4.5 CRM

CRM (Customer Relationship Management) - это система управления взаимодействием с пользователями, направленная на поддержание эффективного взаимодействия, анализа потребностей, улучшения сервиса и поддержки пользователей. Хотя CRM обычно ассоциируется с бизнес-приложениями, она может быть адаптирована для управления пользователями и поддержания связи с ними в рамках проектов с разработкой программного обеспечения.

Цели CRM для данного проекта:

1. Сбор отзывов и рекомендаций для улучшения функциональности и интерфейса.
2. Техническая поддержка пользователей.
3. Управление обновлениями и рассылками об обновлениях программы.

4.6 Техническая документация

Данный проект будет находиться в полном свободном доступе на самом известном ресурсе среди разработчиков ПО GitHub. Благодаря встроенным функциям GitHub, создание технической документации становится намного легче и удобнее. Вся основная документация будет лежать в текстовом файле README.md, классическом файле для предоставления данных о программе на этом ресурсе.

5. Описание логико - математического аппарата

5.1 Формулы и правила подсчета величин. Обработка функций

Так как целью является разработка графического калькулятора, очевидно, что будет большая опора именно на точные вычисления и математические формулы. Как уже было описано, вместо обычной клавиатуры в приложение встроена виртуальная. Сделано это для того, чтобы пользователь не запутался в символах, необходимых для введения. Так, среди всех цифр от 0 до 9 вместе с +, -, /, * и = были добавлены кнопки возведения в степень (“**”) и ввода переменных X и Y.

Стоит также уточнить, что все введенные данные функции будут

обрабатываться модулем Manim, само же преобразование происходит с помощью встроенной функции eval, после чего превращает их в lambda-функцию. Далее функция в преобразованном виде подается на вход функции FunctionGraph() модуля Manim, откуда и идет построение графика.

Основной модуль обработки формул, Manim, использует концепцию сцен для создания анимаций. По своей сути, это класс, в котором с помощью методов описывается расположение, анимация и передвижение тех или иных объектов. Как и было указано раньше, программа берет введенные данные из отдельного текстового файла и импортирует их в сцену, после чего обрабатывает ее и загружает в окно вывода данных. Сама анимация представляет собой файл формата .gif или .mp4, сгенерированного заданными в коде параметрами.

5.2 Набор алгоритмов на формальном языке

Лексический анализ

- **Алгоритм регулярных выражений**: Определяет токены в математическом выражении (числа, операторы, скобки и функции).

Алгоритм упрощения

Применяется для упрощения выражений, таких как факторизация, объединение подобных членов и сокращение дробей.

Алгоритм построения 2D графиков

Отображает функцию f(x) на координатной плоскости.

- Формула: Вычисление значений функции в дискретных точках и соединение их линиями.

Алгоритм анимации графиков

Используется для отображения динамических изменений графиков, например, изменения параметров функции.

6. Ход разработки

6.1 Написание кода

Весь код по итогу состоит из трех файлов - *main.py* (интерфейс, обработка текста), *anim_render.py* (обработка функций и визуализация графиков) и *rendering_queue.txt* (для хранения введенной информации). Разберем каждый из них.

- main.py - в начале создается интерфейс, после чего вводятся методы add_text, cancel_text и render_text. Названия говорят сами за себя. После введения формулы данные переносятся в rendering_queue.txt.
- rendering_queue.txt - сюда копируется мат. формула вида y=f(x), которая принимается на вход в файле anim_render.py.
- anim_render.py - данные из rendering_queue.txt преобразуются в lambda функцию, после чего отправляются в сцену для визуализации графика. После окончания рендеринга файл main.py инициализирует медиаплеер в основном окне.

6.2 Разработка интерфейса

Разработка интерфейса упиралась в создание максимально простой и интуитивно понятной рабочей среды. Для этого было решено уподобить интерфейс виду стандартного калькулятора, но с добавлением окна вывода. Основным элементом цветовой палитры являются темные и серые цвета, которые не сильно мешают глазам и помогают сильнее выделяться белому шрифту текста. Как было сказано ранее, вся разработка интерфейса упиралась в библиотеку PySide6. Лишь малая часть (визуализация графиков) была использована из библиотеки Manim.

6.3 Тестирование и отладка

В ходе первичного тестирования были выявлены следующие проблемы:

- разрешение обработанного видео не соответствовало окну приложения
- проигрыватель .gif изображений не работал
- видео rendering.mp4 не удавалось сохраняться в нужную директорию

- из-за ошибок создания сетки текст был слишком маленьким

Соответственные решения:

- разрешение окна вывода было изменено на 854x480, чтобы соответствовать стандартам рендеринга модуля Manim
- проигрыватель изменен на зацикленный медиаплеер видео .mp4
- была изменена структура обработки видео, чтобы подстроиться под путь сохранения видео
- размер шрифта текста был изменен собственноручно на каждом QLabel и кнопке

7. Развитие и продвижение проекта

На данный момент проект находится в виде, который имеет практическую применимость только для студентов и учеников, так как функционал еще не расширен до конца. Планируется добавление обработки фигур, статистических данных, графиков НЕ функций, визуализация интегральных и дифференциальных уравнений, а также спектр инструментов для создателей контента и преподавателей. Также возможна и реклама на различных сайтах для учеников и учителей, демонстрация продукта на различных образовательных платформах и соцсетях.

8. Заключение

В заключение можно сказать, что все поставленные цели и задачи были выполнены. Я научился работать с библиотеками разработки интерфейса и визуализации математических данных. Данный проект может приоткрыть мне в будущем возможности разработки программного обеспечения для образовательных целей.