# **Документация для программиста к приложению растеризации линий**

## **Введение**

Данное приложение реализует визуализацию растеризации линий с использованием различных алгоритмов. Написано на C++ с использованием библиотеки Qt. Основные функции приложения включают создание графического интерфейса, обработку пользовательских вводов и выполнение алгоритмов растеризации.

## **Архитектура приложения**

Приложение состоит из класса Rasterisation, который наследуется от QMainWindow. Основные компоненты включают:

* **UI**: Интерфейс пользователя, созданный с помощью Qt Designer.
* **Графика**: Использует QPainter для рисования линий и координатной сетки.
* **Алгоритмы растеризации**: Реализованы алгоритмы Брезенхема, ЦДА и других.

## **Основные классы и методы**

### **Класс Rasterisation**

* **Конструктор и деструктор**
  + Rasterisation(QWidget \*parent): Конструктор, инициализирует интерфейс и начальные параметры.
  + ~Rasterisation(): Деструктор, освобождает память.
* **Методы**
  + void MakeOsi(): Создает координатную ось и инициирует отрисовку линий.
  + void UpdateLabel(): Обновляет отображение графики на метке.
  + void SetRange(): Устанавливает диапазоны для полей ввода координат.
  + void DrawLine(): Отрисовывает линию на основе выбранных алгоритмов.
  + void DrawWuLine(): Отрисовывает линию с использованием алгоритма Ву.
  + void ChangeStart(), void ChangeCdStart(): Преобразуют координаты для корректной работы алгоритмов.

### **Алгоритмы растеризации**

* **Пошаговый алгоритм**
  + void on\_StepAlgorithm\_clicked(): Реализует пошаговое растеризирование.
* **Алгоритм Брезенхема**
  + void on\_Bresenhem\_clicked(): Реализует алгоритм Брезенхема для растеризации линий.
* **Цифровой дифференциальный алгоритм**
  + void on\_CDA\_clicked(): Реализует ЦДА алгоритм для растеризации линий.
* **Алгоритм Ву**
  + void on\_AlWu\_clicked(): Реализует алгоритм Ву с поддержкой сглаживания.

### **Обработка событий**

* **События изменения значений**
  + void on\_spinBox\_valueChanged(int arg1): Обрабатывает изменения начальной точки X.
  + void on\_spinBox\_2\_valueChanged(int arg1): Обрабатывает изменения начальной точки Y.
  + void on\_spinBox\_3\_valueChanged(int arg1): Обрабатывает изменения конечной точки X.
  + void on\_spinBox\_4\_valueChanged(int arg1): Обрабатывает изменения конечной точки Y.
* **Управление слайдером**
  + void on\_horizontalSlider\_valueChanged(int value): Устанавливает значение масштаба для отрисовки.

### **Вспомогательные методы**

* void InitPoints(): Инициализирует точки для растеризации.
* QString to\_number(const int & n) const: Преобразует целое число в строку.
* QString to\_float(const float & f) const: Преобразует число с плавающей точкой в строку.

## **Использование**

1. Запустите приложение.
2. Укажите начальные и конечные точки с помощью полей ввода или слайдера.
3. Выберите алгоритм растеризации.
4. Нажмите кнопку для выполнения алгоритма и отображения результата.

## **Заключение**

Данное приложение предоставляет мощные инструменты для визуализации растеризации линий и может быть дополнено новыми алгоритмами или функциями. Код написан с учетом читаемости и возможности расширения, что делает его удобным для дальнейшей разработки.