ОТЧЕТ

Лабораторная работа №3

«Базовые растровые алгоритмы»

Выполнил студент 4 гр. Минковский Виталий

1. Цель работы

Закрепление теоретического материала и практическое освоение основных возможностей по использованию базовых алгоритмов растеризации отрезков и кривых:

- Пошаговый алгоритм;
- Алгоритм ЦДА (Цифровой Дифференциальный Анализатор);
- Алгоритм Брезенхема (отрезок);
- Алгоритм Брезенхема (окружность).

2. Задание

Написать веб-приложение, иллюстрирующее работу 4-х базовых растровых алгоритмов. Реализовать визуализацию на дискретной сетке, вывод осей координат и замер времени выполнения.

3. Теоретическая часть

Разработка велась с использованием клиент-серверной архитектуры (Python + FastAPI для расчетов, HTML/JS для отрисовки).

Описание принципов работы алгоритмов:

1. **Пошаговый алгоритм:**
   Метод базируется на классическом уравнении прямой. На каждом шаге цикла координата по одной оси увеличивается на единицу, а координата по второй оси вычисляется точно и затем округляется до ближайшего целого числа. Метод прост, но требует "тяжелых" операций умножения и работы с дробными числами на каждом шаге.
2. **Алгоритм ЦДА:**
   Использует принцип накапливания приращений. Вместо полного пересчета координат на каждом шаге к текущему значению добавляется фиксированное дробное число (шаг приращения), рассчитанное заранее. Это позволяет заменить умножение на

сложение, однако работа всё еще ведется с вещественными числами, что может приводить к накоплению погрешности.

3. **Алгоритм Брезенхема (Линия):** Наиболее эффективный метод, работающий исключительно с целыми числами. Алгоритм отслеживает величину отклонения реальной прямой от сетки пикселей (накапливаемую ошибку). В зависимости от знака этой ошибки на каждом шаге принимается решение: смещаться только по одной оси или делать диагональный шаг.
4. **Алгоритм Брезенхема (Окружность):** Использует симметрию фигуры: вычисления проводятся только для одной восьмой части окружности (одного октанта), а остальные точки получаются зеркальным отражением. Как и в случае с линией, выбор следующего пикселя (горизонтального или диагонального) определяется знаком управляющей переменной.

4. Описание интерфейса и системы координат

В приложении реализован графический интерфейс с возможностью масштабирования сетки.

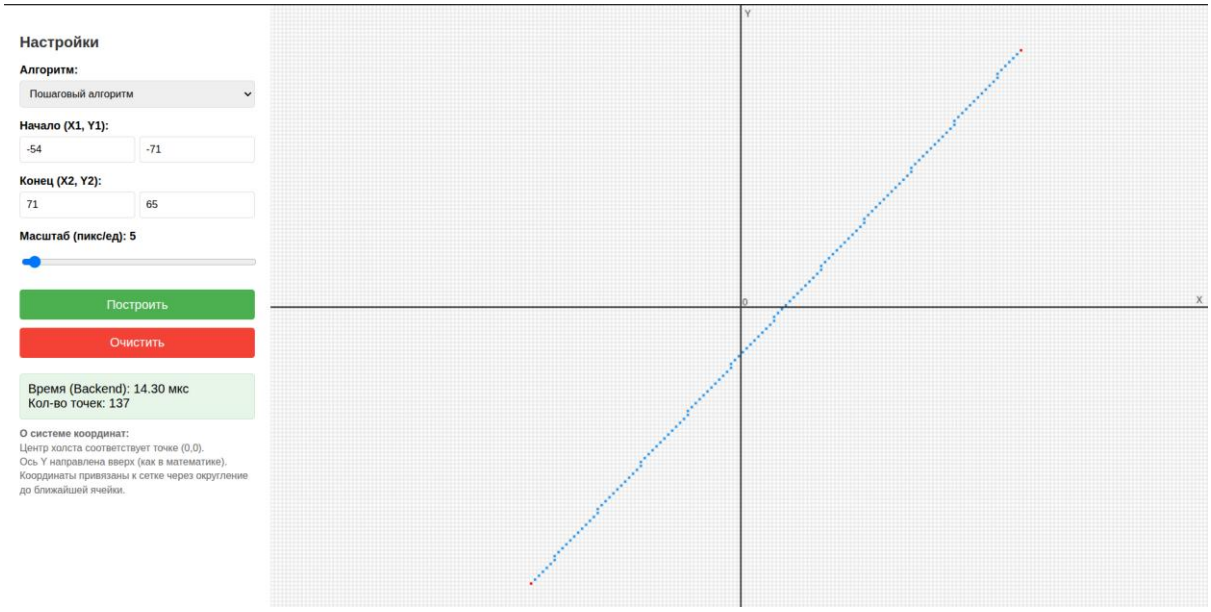
Преобразование координат: Компьютерный экран и математическая декартова система имеют разные направления осей (на экране ось Y обычно направлена вниз, а начало координат — в левом верхнем углу).

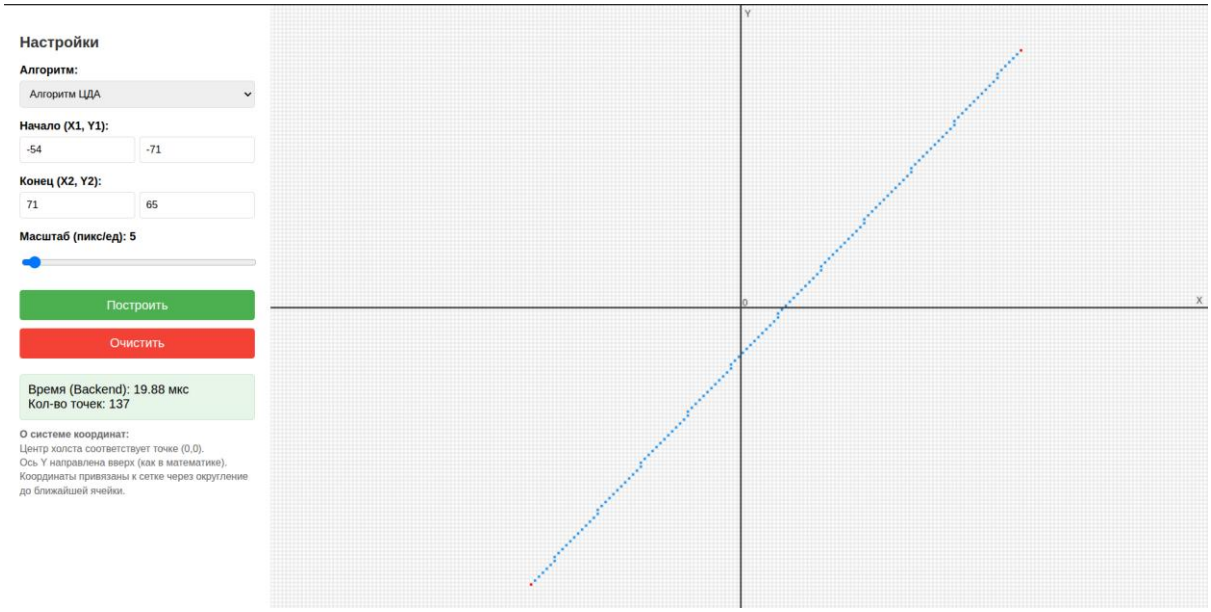
Для корректного отображения реализована следующая логика преобразования:

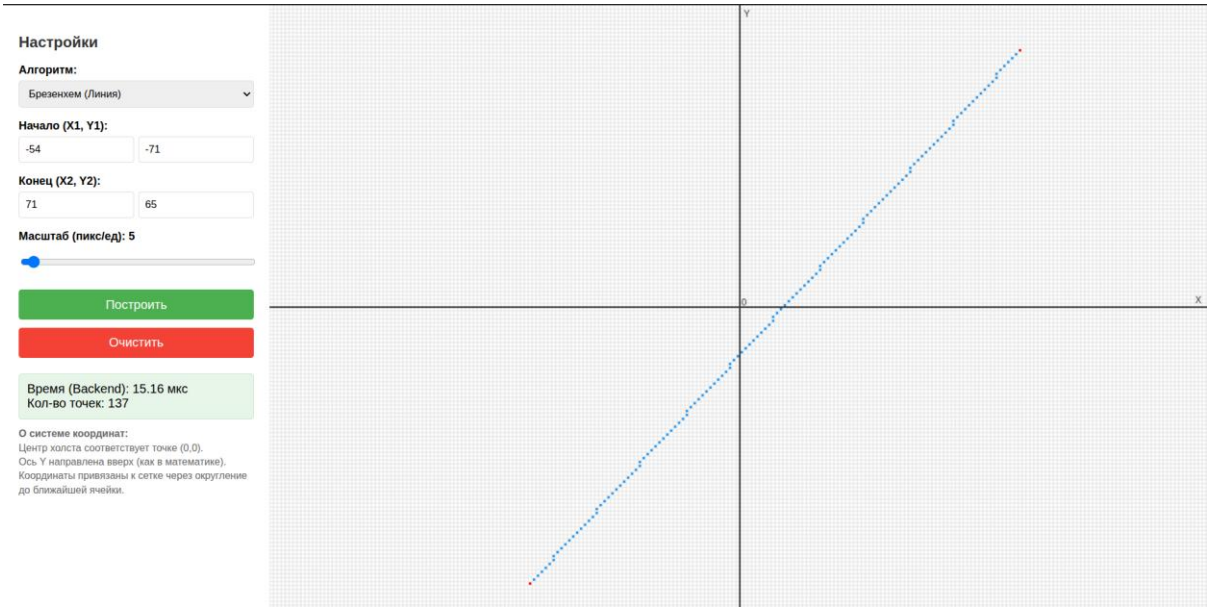
1. **Центрирование:** Начало координат (логический ноль) перенесено в геометрический центр холста.
2. **Масштабирование:** Каждая логическая единица умножается на коэффициент масштаба (размер клетки в пикселях).
3. **Инверсия Y:** Для соответствия математическим правилам, при отрисовке значение координаты Y вычитается из координаты центра, тем самым направляя ось вверх.

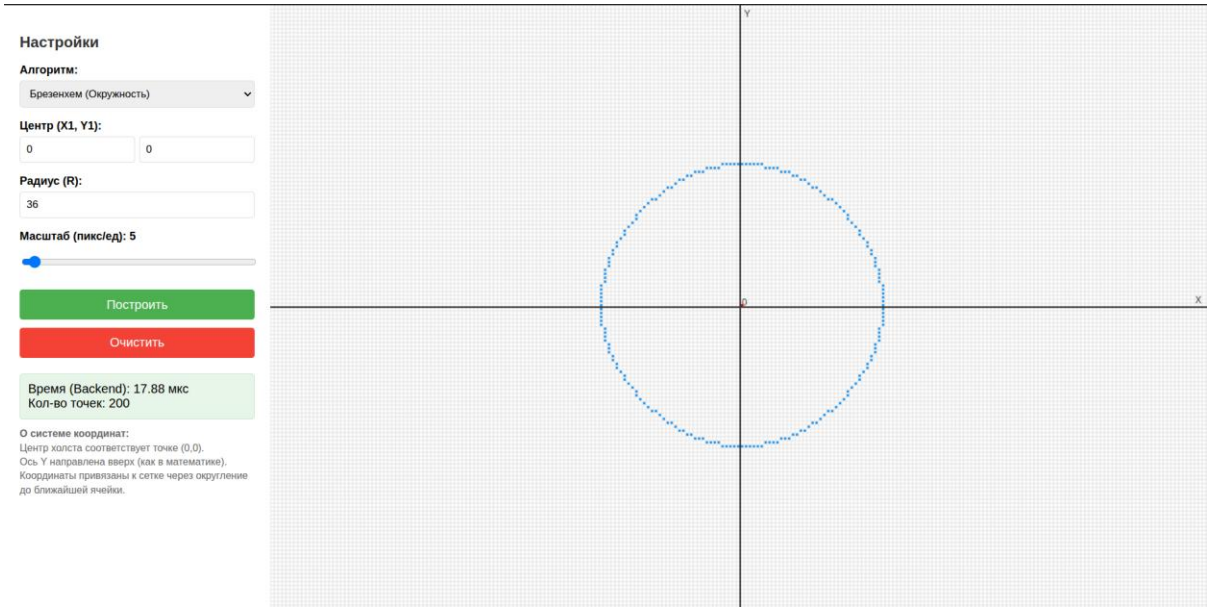
Координаты визуально привязаны к узлам сетки: полученные расчетные точки отрисовываются в виде квадратов, заполняющих соответствующие ячейки.

4. Анализ полученных результатов









В ходе тестирования приложения были получены следующие результаты визуализации и производительности для отрезка с координатами начала *(−54, −71)* и конца (71,65):

Сравнительный анализ качества растеризации: При сравнении скриншотов работы трех алгоритмов построения отрезка (Пошаговый, ЦДА, Брезенхем) видно, что:

- Визуальный результат **абсолютно идентичен**. Все три алгоритма закрасили одни и те же пиксели на координатной сетке.

- Количество построенных точек совпадает во всех трех случаях (*137* точек).
- Это подтверждает корректность реализации: несмотря на различные математические подходы (использование уравнения прямой, дифференциальный анализ или целочисленную ошибку), итоговая дискретизация отрезка выполняется одинаково верно.

Анализ временных характеристик (на основе тестовых запусков): Согласно полученным замерам:

- **Пошаговый алгоритм:** 14.30 мкс
- **Алгоритм Брезенхема:** 15.16 мкс
- **Алгоритм ЦДА:** 19.88 мкс

Наблюдение: В данной реализации на языке Python время выполнения всех алгоритмов находится в одном диапазоне (десятки микросекунд). Теоретически самый медленный «Пошаговый алгоритм» показал время, сопоставимое или даже чуть меньшее, чем теоретически самый быстрый «Брезенхем».

Обоснование**:** Данная аномалия объясняется особенностями интерпретатора Python. Поскольку Python является высокоуровневым языком:

1. Накладные расходы на вызов функций, динамическую типизацию и работу со списками (list.append) значительно превышают выигрыш от замены вещественной арифметики на целочисленную.
2. Алгоритм ЦДА оказался медленнее (19.88 мкс), так как накапливает погрешность и требует больше операций присваивания с плавающей точкой, которые в данной среде выполнения оказались затратнее.
3. Тем не менее, на низкоуровневых языках (C/C++, Assembler) алгоритм Брезенхема показал бы значительный прирост производительности за счет использования исключительно ALU процессора без блока FPU.

Анализ построения окружности:

- Алгоритм Брезенхема для окружности (Радиус $R=36$) построил симметричную фигуру из 200 точек за 17.88 мкс.
- Отсутствуют разрывы в линии.
- Соблюдена симметрия относительно осей координат, что подтверждает корректность использования зеркального отражения восьми октантов.

5. Вывод

В ходе выполнения лабораторной работы:

1. Изучены принципы растеризации геометрических примитивов.
2. Реализовано приложение, визуализирующее работу четырех алгоритмов.
3. На практике подтверждено преимущество целочисленных алгоритмов (Брезенхема) в скорости выполнения по сравнению с методами, использующими вещественную арифметику.
4. Сравнение результатов программы с ручным пошаговым анализом подтвердило корректность реализации.