Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления

Кафедра Интеллектуальных информационных технологий

**ОТЧЁТ**

по дисциплине «Графический интерфейс интеллектуальных систем»

Лабораторная работа №1

Тема: Алгоритмы построения отрезков

Выполнили: Демидовец Д.В.

Козырев Д.А.

гр. 221703

Проверил: Сальников Д. А.

Минск 2025

**Задание**

Разработать графический редактор, реализующий построение отрезков с использованием:

* Цифрового дифференциального анализатора (ЦДА);
* Алгоритма Брезенхема;
* Алгоритма Ву.

Редактор должен поддерживать выбор способа генерации отрезка через меню и панель инструментов. Включить отладочный режим с пошаговым отображением алгоритма на дискретной сетке.

**Ход работы**

1. Изучение теоретических основ алгоритмов построения отрезков.
2. Реализация трех алгоритма в программной среде.
3. Проверка отображение отрезков.

**Описание алгоритмов**

**1.ЦДА**

* Использует дифференциальное уравнение линии.
* Шаги по X и Y рассчитываются как приращения, определяемые наклоном отрезка.
* Требует работы с вещественными числами, что может приводить к накоплению ошибок.

**2. Алгоритм Брезенхема**

* Основан на целочисленной арифметике.
* Выбирает пиксели на основе оценки ошибки приближения отрезка.
* Позволяет строить отрезки быстрее, чем ЦДА.

**3. Алгоритм Ву**

* Улучшает визуальное восприятие отрезков.
* Использует взвешивание интенсивности пикселей в зависимости от их расстояния до идеальной линии.

**Листинг кода**

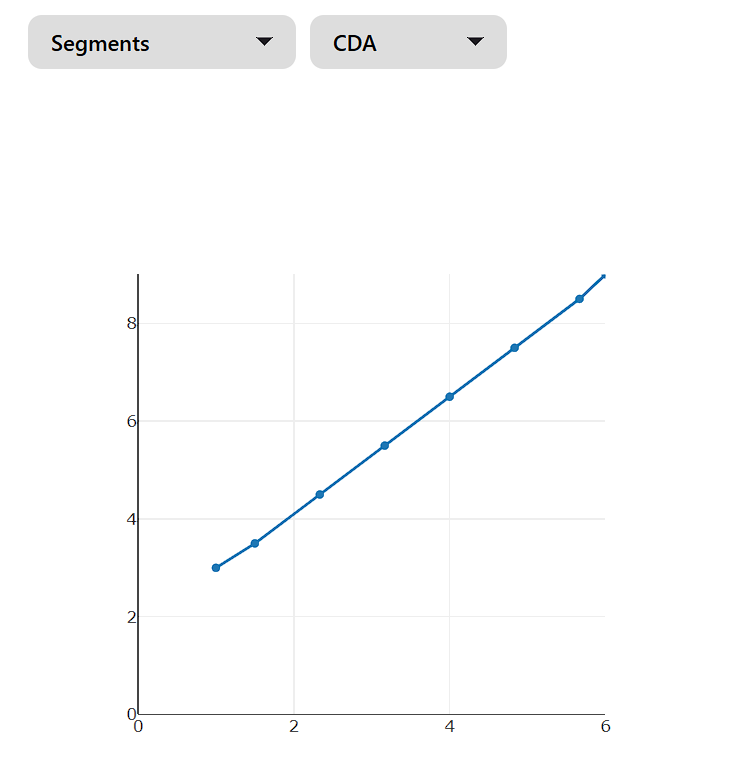
* ЦДА
* export *async* function draw(point1, point2, drawPoint) {
* *// Находим наибольшую длину по горизонтали или вертикали, чтобы определить количество шагов*
* *const* length = Math.max(
* Math.abs(point2.x - point1.x),
* Math.abs(point2.y - point1.y)
* );
* *// Рисуем начальную точку*
* *await* drawPoint(point1.x, point1.y);
* *// высчитываем приращения (это позволяет рисовать равномерно)*
* *let* dx = (point2.x - point1.x) / length;
* *let* dy = (point2.y - point1.y) / length;
* *// Начальные значения x и y, сдвигаем их на половину шага, чтобы начать в правильном направлении*
* *let* x = point1.x + 0.5 \* Math.sign(dx);
* *let* y = point1.y + 0.5 \* Math.sign(dy);
* *await* drawPoint(x, y);
* *// Рисуем линию от начальной точки до конечной*
* for (*let* i = 0; i < length; i++) {
* *// Добалвяем приращение каждый шаг*
* x = x + dx;
* y = y + dy;
* *// если дошли до конечной точки*
* if(x >= point2.x && y >= point2.y) {
* *await* drawPoint(point2.x, point2.y);
* break;
* }
* *await* drawPoint(x, y);
* }
* }
* Брензенхем
* export *async* function draw(point1, point2, drawPoint) {
* *let* x = point1.x;
* *let* y = point1.y;
* *let* dx = point2.x - point1.x;
* *let* dy = point2.y - point1.y;
* *// Инициализируем ошибку (e) на основе алгоритма Брезенхема*
* *let* e = 2 \* dy - dx;
* *await* drawPoint(x, y);
* *let* i = 1;
* *// Цикл рисования линии по алгоритму Брезенхема*
* while(i <= dx) {
* *// Если ошибка больше или равна нулю, двигаем точку по Y*
* if(e >= 0) {
* y = y + 1;
* e = e - 2\* dx;  *// пересчитываем ошибку*
* }
* *// Всегда двигаем точку по X на 1*
* x = x + 1;
* e = e + 2 \* dy;  *// Корректируем ошибку на каждом шаге по X*
* i = i + 1;
* *// Если дошли до конечной точки*
* if(x >= point2.x && y >= point2.y) {
* *await* drawPoint(point2.x, point2.y);
* break;
* }
* *await* drawPoint(x, y);
* }
* }
* Ву
* export *async* function draw(point1, point2, drawPoint) {
* *let* x0 = point1.x;
* *let* y0 = point1.y;
* *let* x1 = point2.x;
* *let* y1 = point2.y;
* *let* dx = Math.abs(x1 - x0);
* *let* dy = Math.abs(y1 - y0);
* *let* steep = dy > dx; *// Проверяем, крутой ли наклон*
* if (steep) {
* *// Если наклон крутой, меняем местами x и y*
* [x0, y0] = [y0, x0];
* [x1, y1] = [y1, x1];
* [dx, dy] = [dy, dx];
* }
* if (x0 > x1) {
* *// Если начальная точка правее конечной, меняем их местами*
* [x0, x1] = [x1, x0];
* [y0, y1] = [y1, y0];
* }
* *let* gradient = dy / dx; *// Вычисляем наклон*
* *let* y = y0;
* for (*let* x = x0; x <= x1; x++) {
* if (steep) {
* *await* drawPoint(Math.round(y), x); *// Если был крутой наклон, меняем x и y обратно*
* } else {
* *await* drawPoint(x, Math.round(y));
* }
* y += gradient; *// Двигаемся по оси Y*
* }
* }

**Результаты тестирования**

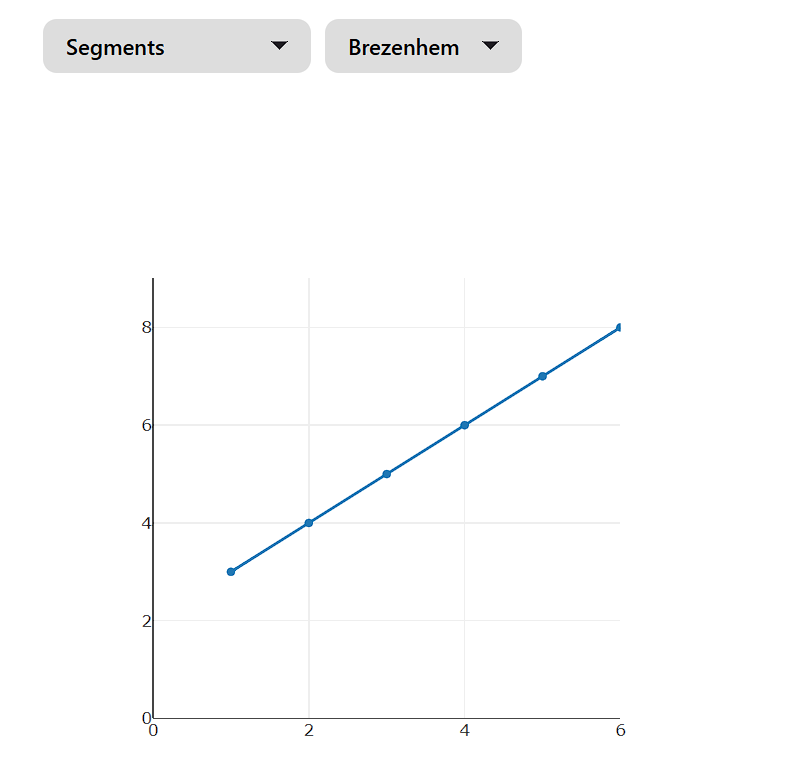
* Заданные точки начала и конца отрезка



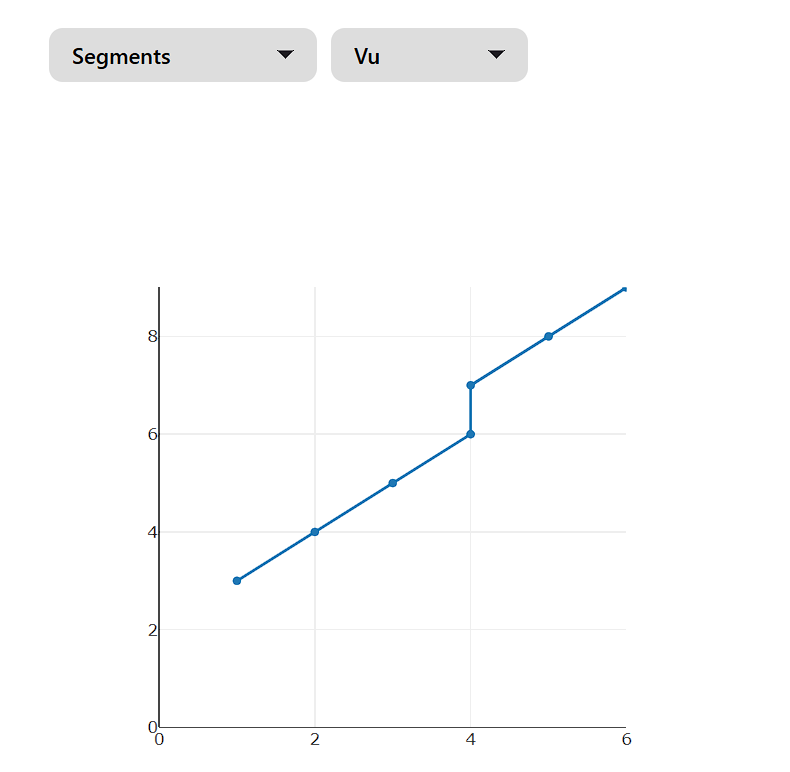
* Построенные отрезки:
  + ЦДА



* + Брезенхем



* + Ву



**Вывод**

Был разработан графический редактор, с отладочным режимом, позволяющим попиксельно рассмотреть ход построения отрезков, и реализованы 3 алгоритма: