Дополнение к отчёту по лабораторной работе №3

1. Пример вычислений для алгоритма Брезенхема (отрезок)

Рассмотрим отрезок от точки A(2, 1) до точки B(10, 6). Для него `dx = 8`, `dy = 5`, `sign(dx) = 1`, `sign(dy) = 1`, а начальное значение ошибки `d0 = 2·dy − dx = 2`. Ниже приведены промежуточные значения ошибки и принятое решение (смещение вдоль оси `x` или одновременно `x` и `y`).

i	(x_i, y_i)	d_i	Решение	(x_{i+1}, y_{i+1})	d_{i+1}
0	(2, 1)	2	d > 0 ⇒ (x+1, y+1)	(3, 2)	−4
1	(3, 2)	−4	d ≤ 0 ⇒ (x+1, y)	(4, 2)	6
2	(4, 2)	6	d > 0 ⇒ (x+1, y+1)	(5, 3)	0
3	(5, 3)	0	d ≤ 0 ⇒ (x+1, y)	(6, 3)	10
4	(6, 3)	10	d > 0 ⇒ (x+1, y+1)	(7, 4)	4
5	(7, 4)	4	d > 0 ⇒ (x+1, y+1)	(8, 5)	−2
6	(8, 5)	−2	d ≤ 0 ⇒ (x+1, y)	(9, 5)	8
7	(9, 5)	8	d > 0 ⇒ (x+1, y+1)	(10, 6)	2

Итоговый набор целочисленных точек: `(2,1)`, `(3,2)`, `(4,2)`, `(5,3)`, `(6,3)`, `(7,4)`, `(8,5)`, `(9,5)`, `(10,6)` — совпадает с визуализацией в веб-приложении.

2. Временные характеристики реализованных алгоритмов

Измерения выполнялись в Node.js v18.19.1. Каждый алгоритм вызывался 50 000 раз, после чего среднее время вычислялось как `(общее время) / 50000`. Для теста линий использован отрезок `(2, 1) → (120, 70)`, для окружности — центр `(0, 0)` и радиус `60`. Дополнительно оценено количество арифметических операций (сложение, вычитание, умножение, деление, `abs`, `round`, `floor`, `sqrt`, модуль) для одного прохода алгоритма на тех же входных данных.

Алгоритм	Среднее время за вызов, мс	Время работы, операции
Пошаговый (отрезок)	0.018	837
ЦДА (отрезок)	0.018	482
Брезенхем (отрезок)	0.0013	497
Castle–Pitway (отрезок)	0.0069	444
By (отрезок)	0.0347	731
Брезенхем (окружность)	0.0487	1242
By (окружность)	0.115	1727

Целочисленный Брезенхем для отрезка ожидаемо самый быстрый в миллисекундах: он обходится без операций с плавающей точкой и использует минимальный набор вычислений, хотя суммарное число элементарных операций сравнимо с ЦДА. Castle–Pitway «равномерно» распределяет диагональные и горизонтальные шаги, а потому требует меньше арифметики, чем ЦДА, но проигрывает Брезенхему из-за построения последовательностей. Алгоритмы By работают с интенсивностями и плавающей точкой, что даёт заметный рост как по времени, так и по количеству операций. Построение окружности Брезенхема требует больше времени, потому что на каждом шаге генерирует восемь симметричных точек и обновляет три целочисленных переменных; сглаженная окружность By дополнительно тратит ресурсы на вычисление корней и интенсивностей.