Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ имени Н.Э.БАУМАНА
(национальный исследовательский университет)»

Факультет: Информатика и системы управления

Кафедра: Теоретическая информатика и компьютерные технологии

Лабораторная работа № 2

«Решения задачи построения карты местности по геоточкам»

по дисциплине «Моделирование»

Работу выполнил

студент группы ИУ9-82Б

Жук Дмитрий

Москва, 2023

Цель работы

Целью данной работы является построения полигональной карты местности для рендера 3D модели на компьютере по массиву координат геоточек.

Задание

Используя итеративный алгоритм построения триангуляции Делоне, соединить массив точек геоданных в данную триангуляцию и произвести визуализацию конечного результата.

Теория

Одной из важнейших операций, выполняемых при построении триангуляции, является проверка условия Делоне для заданных пар треугольников. На практике обычно используют один из данных способов проверки:

1. Проверка через уравнение описанной окружности.
2. Проверка с заранее вычисленной описанной окружностью.
3. Проверка суммы противолежащих углов.
4. Модифицированная проверка суммы противолежащих углов.

Был выбран способ проверки через уравнение описанной окружности. Уравнение окружности, проходящей через точки $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ можно записать в виде

$$\begin{vmatrix} x^2 + y^2 & x & y & 1 \\ x_1^2 + y_1^2 & x_1 & y_1 & 1 \\ x_2^2 + y_2^2 & x_2 & y_2 & 1 \\ x_3^2 + y_3^2 & x_3 & y_3 & 1 \end{vmatrix} = 0$$

или же как $(x^2 + y^2) \cdot a - x \cdot b + y \cdot c - d = 0$, где

$$a = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}, \qquad b = \begin{vmatrix} x_1^2 + y_1^2 & y_1 & 1 \\ x_2^2 + y_2^2 & y_2 & 1 \\ x_3^2 + y_3^2 & y_3 & 1 \end{vmatrix},$$

$$c = \begin{vmatrix} x_1^2 + y_1^2 & x_1 & 1 \\ x_2^2 + y_2^2 & x_2 & 1 \\ x_3^2 + y_3^2 & x_3 & 1 \end{vmatrix}, \qquad d = \begin{vmatrix} x_1^2 + y_1^2 & x_1 & y_1 \\ x_2^2 + y_2^2 & x_2 & y_2 \\ x_3^2 + y_3^2 & x_3 & y_3 \end{vmatrix}.$$

Тогда условие Делоне для любого заданного треугольника $\Delta\big((x_1, y_1), (x_2, y_2), (x_3, y_3)\big)$ будет выполняться только тогда, когда для любого узла (x_0, y_0) триангуляции будет $(a \cdot (x_0^2 + y_0^2) - b \cdot x_0 + c \cdot y_0 - d) \cdot \operatorname{sign} a \geq 0$, т.е. когда (x_0, y_0) не попадает внутрь окружности, описанной вокруг треугольника $\Delta\big((x_1, y_1), (x_2, y_2), (x_3, y_3)\big)$.

Реализация

```
const det3 = (a: number[][]) => 0
        + a[0][0] * a[1][1] * a[2][2]
        + a[0][1] * a[1][2] * a[2][0]
        + a[0][2] * a[1][0] * a[2][1]
        - a[0][2] * a[1][1] * a[2][0]
        - a[0][1] * a[1][0] * a[2][2]
        - a[0][0] * a[1][2] * a[2][1];

const checkDelaunay = (t: Triangle, p0: Point): boolean => {
        const [p1, p2, p3] = t.points;
        const x0 = p0.x;
        const y0 = p0.y;
        const x1 = p1.x;
        const y1 = p1.y;
        const x2 = p2.x;
        const y2 = p2.y;
        const x3 = p3.x;
        const y3 = p3.y;

        const s1 = x1 ** 2 + y1 ** 2;
        const s2 = x2 ** 2 + y2 ** 2;
        const s3 = x3 ** 2 + y3 ** 2;

        const a = det3([
```

```
                [x1, y1, 1],
                [x2, y2, 1],
                [x3, y3, 1],
        ]);
        const b = det3([
                [s1, y1, 1],
                [s2, y2, 1],
                [s3, y3, 1],
        ]);
        const c = det3([
                [s1, x1, 1],
                [s2, x2, 1],
                [s3, x3, 1],
        ]);
        const d = det3([
                [s1, x1, y1],
                [s2, x2, y2],
                [s3, x3, y3],
        ]);

        return Math.sign(a) * (a * (x0 ** 2 + y0 ** 2) - b * x0 + c * y0 - d)
>= 0;
};
```

Листинг 1 — Алгоритм проверки условия Делоне для заданного треугольника и точки

```
const iterativeDelaunay = async (data: Data, newPoint: Point) => {
        const incorrectTriangles = await data.getIncorrectTriangles(newPoint);
        await Promise.all(incorrectTriangles.map(data.removeTriangle));

        const grad = (p: Point) => Math.atan2(p.y - newPoint.y, p.x -
newPoint.x);

        const lostPoints = [...new Set(incorrectTriangles.flatMap(({ points })
=> points))]
                .sort((a, b) => grad(a) - grad(b));

        await Promise.all(lostPoints.map(
                (p, i) => data.createTriangle(p, lostPoints[(i + 1) %
lostPoints.length], newPoint),
        ));
};
```

Листинг 2 — Алгоритм добавления точки к существующей триангуляции Делоне

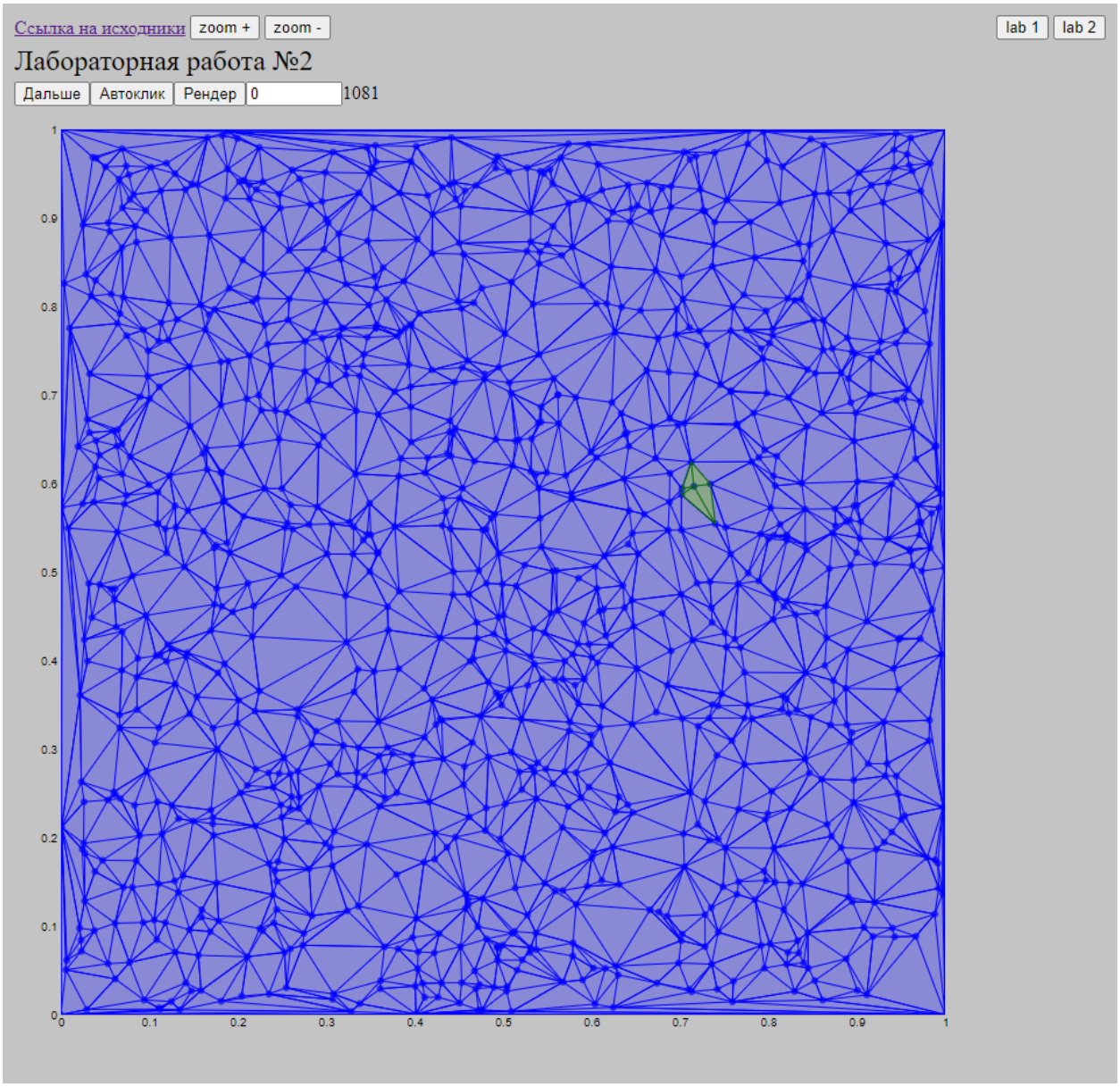

Рисунок 1 – график получившейся триангуляции

Вывод

В ходе лабораторной работы был изучен способ построения триангуляции Делоне, что исходя из её свойств она позволяет избежать очень острых треугольников, которые будут мешать и создавать визуальные артефакты. Также была выявлена неточность в алгоритме проверки суммы

противолежащих углов предложенная во многих источниках – нигде не упоминается важность порядка перебора точек в треугольнике, ведь это влияет на то какие получатся углы. Насчет данной проблемы был обнаружен пост на Хабре: https://habr.com/ru/post/252925/.