Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования Московский государственный технический университет имени Н.Э. Баумана

Лабораторная работа №2
«Триангуляция Делоне»
по курсу
«Моделирование»

Студент группы ИУ9-82 Белогуров А.А.

Преподаватель Домрачева А.Б.

Москва, 2018

Содержание

1 Постановка задачи

Изучить различные алгоритмы триангуляции Делоне и реализовать итеративный алгоритм с динамическим кэшированием поиска на произвольном наборе точек.

2 Теоретические сведения

2.1 Основные определения

Триангуляцией называется планарный граф, все внутренние области которого являются треугольниками. [1]

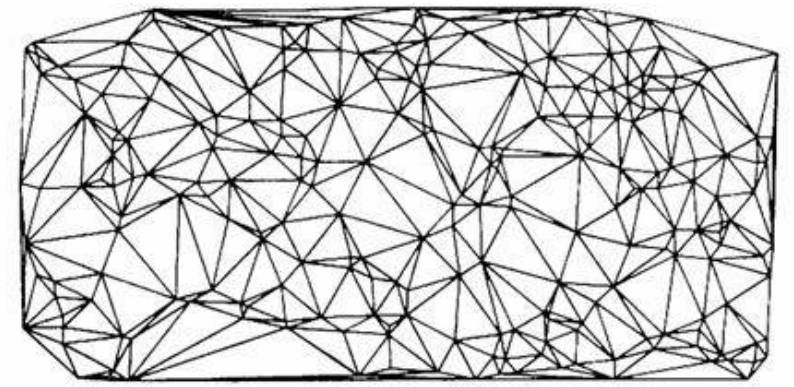

Рис. 1: Пример триангуляции

Выпуклой триангуляцией называется такая триангуляция, для которой минимальный многоугольник, охватывающий все треугольники, будет выпуклым. Триангуляция, не являющаяся выпуклой, называется **невыпуклой**. [1]

Триангуляция удовлетворяет **условию Делоне**, если внутрь окружности, описанной вокруг любого построенного треугольника, не попадает ни одна из заданных точек триангуляции. [1]

Триангуляция называется **триангуляцией Делоне**, если она является выпуклой и удовлетворяет условию Делоне (Рис. 2)

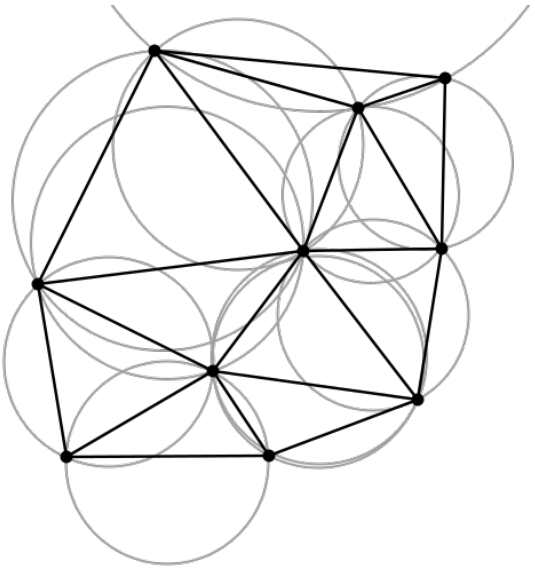

Рис. 2: Триангуляция Делоне

2.2 Итеративный алгортим с динамическим кэшированием поиска

1. Создание суперструктуры, которая будет покрывать все точки. *(В данном случае создается прямоугольник, который разбивается диагональю на два треугольника. Следовательно, получаем на этом шаге триангуляцию, которая состоит их двух одинаковых треугольников)*

2. Выполняем цикл по n точкам для пунктов 3-5.

3. Очередная n-ая точка добавляется в триангуляцию следующим образом. Вначале производится локализация точки с помощью динамического кэша.

   3.1. Если точка попадает в эпсилон-окрестность любой другой вершины триангуляции, то она отбрасывается.

   3.2. Если точка попала на некоторое ребро, то оно разбивается на два новых, а оба смежных с ребром треугольника также делятся на два меньших.

3.3. Если точка попала строго внутрь какого-нибудь треугольника, он разбивается на три новых.

3.4. Если точка попала вне триангуляции, то строится один или более треугольников. *(Так как на первом шаге была построена суперструктура, то все точки заведомо будут лежать внутри нее. Поэтому выполнение этого пункта невозможно)*

4. Проводятся локальные проверки вновь полученных треугольников на соответствие условию Делоне и выполняются необходимые перестроения.

3 Практическая реализация

Далее приведена реализация некоторых методов программы на языке JavaScript, визуализация выполненяется с помощью библиотеки THREE.JS.

Листинг 1 Создание суперструктуры.

```
/**
 * Создание суперструктуры, внутри которой будет происходить
 ↪  триангуляция
 * @param {Point} topLeft
 * @param {Point} bottomRight
 */
Triangulation.prototype.createSuperstructure = function(topLeft,
 ↪  bottomRight) {
    // Инициализация двух треугольников
    var left = new Triangle();
    var right = new Triangle();

    var bottomLeft = new Point(topLeft.X, bottomRight.Y);
    var topRight = new Point(bottomRight.X, topLeft.Y);

    // Инициализация ребер суперструктуры
    var diagonal = new Rib(topLeft, bottomRight, left, right);
    var leftRib = new Rib(topLeft, bottomLeft, left, null);
    var rightRib = new Rib(topRight, bottomRight, right, null);
    var topRib = new Rib(topLeft, topRight, right, null);
    var bottomRib = new Rib(bottomLeft, bottomRight, left, null);

    // Определение ребер для треугольников
    left.setRibs(leftRib, bottomRib, diagonal);
    right.setRibs(rightRib, topRib, diagonal);

    // Обновлние струтуры треугольников
    left.update();
    right.update();

    // Добавление треугольников в кэш
    this.cache.initialize(left, right, left, right);
```

```
31
32      return [left, right];
33  };
```

Листинг 2 Вставка новой точки в триангуляцию

```
1   /**
2    * Вставка новой точки в триангулцию
3    * 1) Нахождение треугольника иди ребра, куда попала новая точка
4    * 2) Если точка попадает в эпсилон-окрестность любой другой
     ↪  вершины триангуляции - игнорировать ее
5    * 3) Если точка попадает на ребро, то треугольники с этим ребром
     ↪  разбиваются на два новых
6    * 4) Если точка попадает внутрь треугольника - треугольник
     ↪  разбивается на три новых
7    * @param {Point} node Новая точка
8    * @returns {array} Массив новых\измененных треугольников
9    */
10  Triangulation.prototype.calcTriangulation = function (node) {
11      // Шаг 1)
12      var initTriangle = this.cache.get(node);
13      var targetTriangle = this.findTriangleBySeparatingRibs(node,
         ↪  initTriangle);
14
15      // Шаг 2)
16      for (i = 0; i < targetTriangle.verticies.length; i++) {
17          var vert = targetTriangle.verticies[i];
18          if (vert.isInEpsilonAreaPoint(node)) {
19              return null;
20          }
21      }
22
23      // Массив новых и измененных треугольников
24      var newAndModifiedTriangles = null;
25      var newTriangles = null;
26
27      // Шаг 3)
28      var targetTriangleRibs = targetTriangle.ribs;
29      for (var i = 0; i < targetTriangleRibs.length; i++) {
```

```
30          if
             ↪  (isInEpsilonArea(distanceToLine(targetTriangleRibs[i].A,
             ↪  targetTriangleRibs[i].B, node), 0)) {
31              newAndModifiedTriangles =
                 ↪  this.putPointOnRib(targetTriangleRibs[i], node,
                 ↪  newTriangles);
32              break;
33          }
34      }
35
36      // Шаг 4)
37      if (newAndModifiedTriangles == null) {
38          var triangles = this.putPointInTriangle(targetTriangle,
             ↪  node, newTriangles);
39          newAndModifiedTriangles = triangles.mdTr;
40          newTriangles = triangles.newTr;
41
42      }
43      // Увеличение кол-ва вершин в кэше
44      this.cache.incrementNodeCount(newTriangles.length);
45
46      // Добавление новых треугольников в кэш
47      for (i = 0; i < newTriangles.length; i++) {
48          this.cache.update(newTriangles[i]);
49      }
50
51      return newAndModifiedTriangles;
52  };
```

Листинг 3 Попадание точки внутрь треугольника

```
1   /**
2    * Попадание узла внутрь треугольника
3    * @param {Triangle} T
4    * @param {Point} node
5    * @param {Array<Triangle>} newTriangles
6    * @return {Array<Triangle>} Измененные треугольники
7    */
8   Triangulation.prototype.putPointInTriangle = function(T, node,
     ↪  newTriangles){
```

```
9       // Вершины
10      // node == 0
11      var A = T.verticies[0];
12      var B = T.verticies[1];
13      var C = T.verticies[2];
14
15      // Треугольники
16      var LT = new Triangle();
17      var RT = new Triangle();
18
19      newTriangles = [LT, RT];
20
21      // Ребра
22      var AB = T.getRib(A, B);
23      var BC = T.getRib(B, C);
24      var AC = T.getRib(A, C);
25
26      // Новые ребра.
27      var OA = new Rib(node, A, LT, T);
28      var OB = new Rib(node, B, LT, RT);
29      var OC = new Rib(node, C, RT, T);
30
31      // Обновление ссылок на смежные треугольники
32      AB.update(T, LT);
33      BC.update(T, RT);
34
35      // Обновление старых ребер на новые
36      T.updateRib(AB, OA);
37      T.updateRib(BC, OC);
38
39      // Определение новых ребер треугольника
40      LT.setRibs(AB, OB, OA);
41      RT.setRibs(BC, OB, OC);
42
43      // Добавление новых треугольников в триангуляцию
44      this.triangles.push(LT);
45      this.triangles.push(RT);
46
47      // Обновление структур треугольников
48      T.update();
49      LT.update();
50      RT.update();
```

```
51
52      var modifiedTriangles = [T, LT, RT];
53      // Return new and modified triangles.
54      return {
55          newTr: newTriangles,
56          mdTr: modifiedTriangles
57      }
58  };
```

Листинг 4 Локализация точки

```
1   /**
2    * Найти треугольник в кэше по точке
3    * @param {Point} node Точка
4    * @return {Triangle} Треугольник из кэша
5    */
6   DynamicCache.prototype.get = function(node)
7   {
8       var row = this.getRow(node.Y);
9       var col = this.getCol(node.X);
10      return this.cache[row][col];
11  };
```

4 Результаты

Далее будут приведены результаты программы на разном количестве данных.

С отображением суперструктуры:

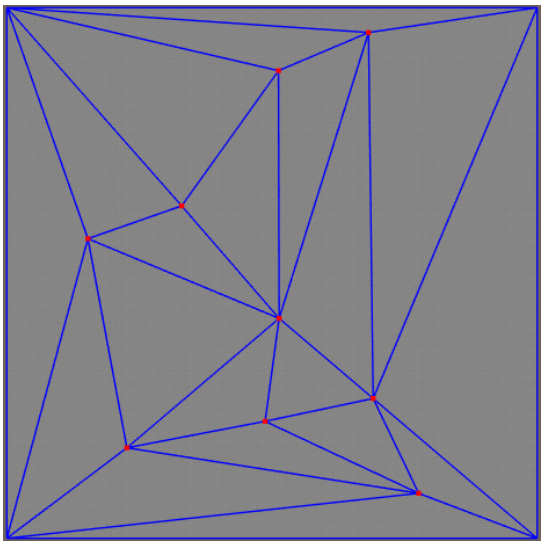

Рис. 3: 9 точек

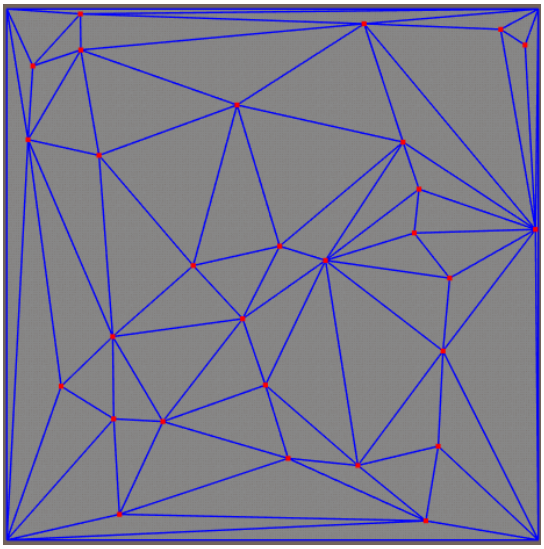

Рис. 4: 29 точек

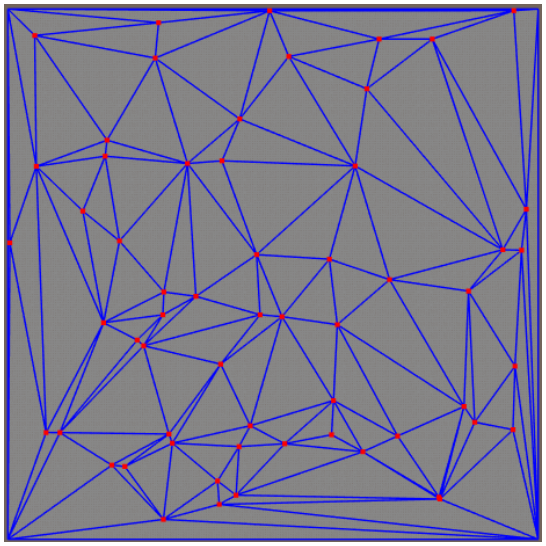

Рис. 5: 59 точек

Без отображениея суперструктуры:

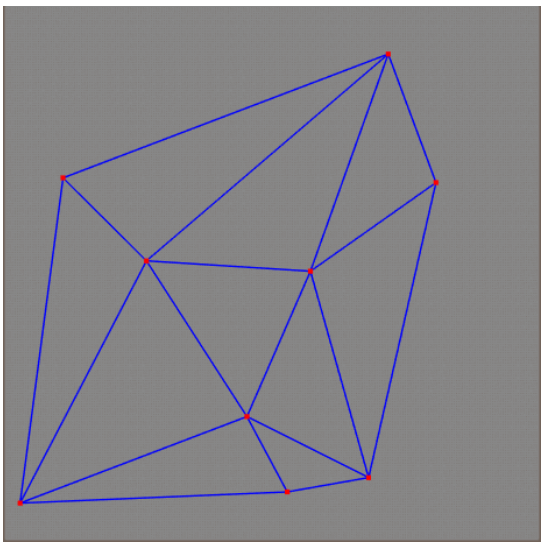

Рис. 6: 9 точек

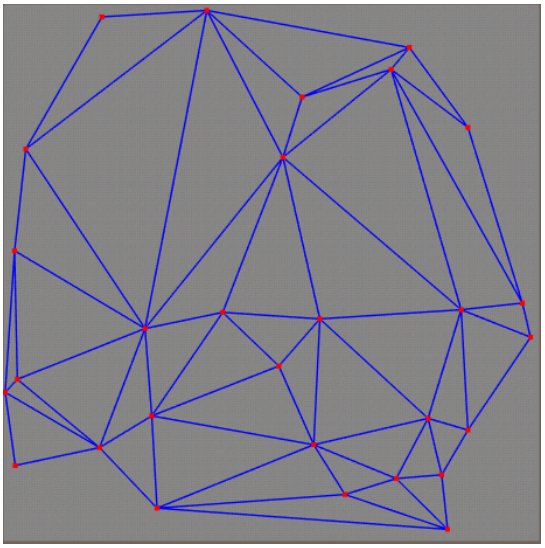

Рис. 7: 29 точек

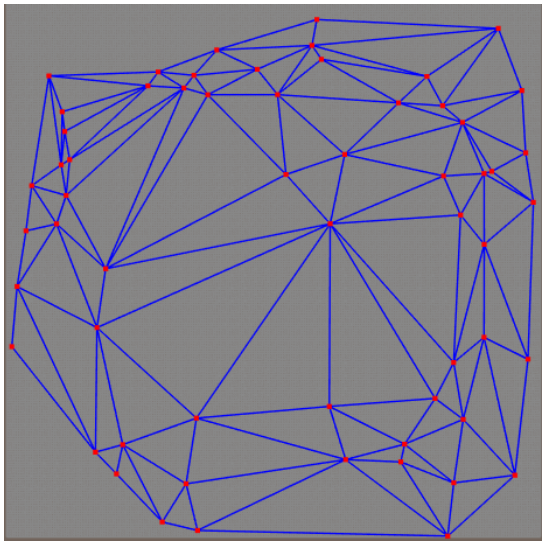

Рис. 8: 59 точек

5 Вывод

В ходе выполнения лабораторной работы было изучены алгоритмы построения триангуляции Делоне и реализован итеративный алгоритм с динамическим кэшированием поиска. Он выигрывает по быстродействию у всех существующих, так как имеют динамическую структуру - кэш, который служит для быстрой локализации точки. Данный факт был подтвержден для большого количества данных.

Список литературы

[1] Скворцов А. В. Триангуляция Делоне и её применение. 2002. 128 с.