**Государственное бюджетное общеобразовательное учреждение "Президентский физико-математической лицей № 239"**

**НАХОЖДЕНИЕ ТАКОЙ ПАРЫ "ШИРОКИЙ ЛУЧ"-ТРЕУГОЛЬНИК, В КОТОРОЙ ФИГУРА, НАХОДЯЩИЙСЯ ВНУТРИ "ШИРОКОГО ЛУЧА" И ТРЕУГОЛЬНИКА ИМЕЕТ МАКСИМАЛЬНУЮ ПЛОЩАДЬ.**

Годовой проект по информатике

Работу выполнил

ученик 10–2 класса

Малков Максим

Санкт-Петербург

2020

1. **Постановка задачи**

На плоскости задано множество "широких лучей" и множество треугольников. Найти такую пару "широкий луч"-треугольник, что фигура, находящаяся внутри "широкого луча" и треугольника, имеет максимальную площадь

1. **Уточнение исходных и выходных данных и ограничений на них**

**2.1. Исходные данные**

Во входном файле [triangles.txt](https://github.com/Haha-bip/JavaGl239Project-master/blob/master/triangles.txt) содержатся строки с информацией о каждом заданном треугольнике в формате: x1 y1 x2 y2 x3 y3, где x1 и y1 — координаты вершины, x2 , y2, x3, y3 – координаты второй и третьей вершин треугольника соответственно. Например, 0,4273 -0,5748 0,2809 -0,2952 0,9426 -0,1814. Количество строк заранее неизвестно. Каждая из заданных координат представляет из себя вещественное число, в котором дробная часть (если она есть) отделена запятой от целой части. Дробная часть может отсутствовать. Число может быть отрицательным (знак минус перед числом). Ограничения на координаты заданы, это вещественные числа в области [(-1,-1), (1,1)].

**2.2. Выходные данные**

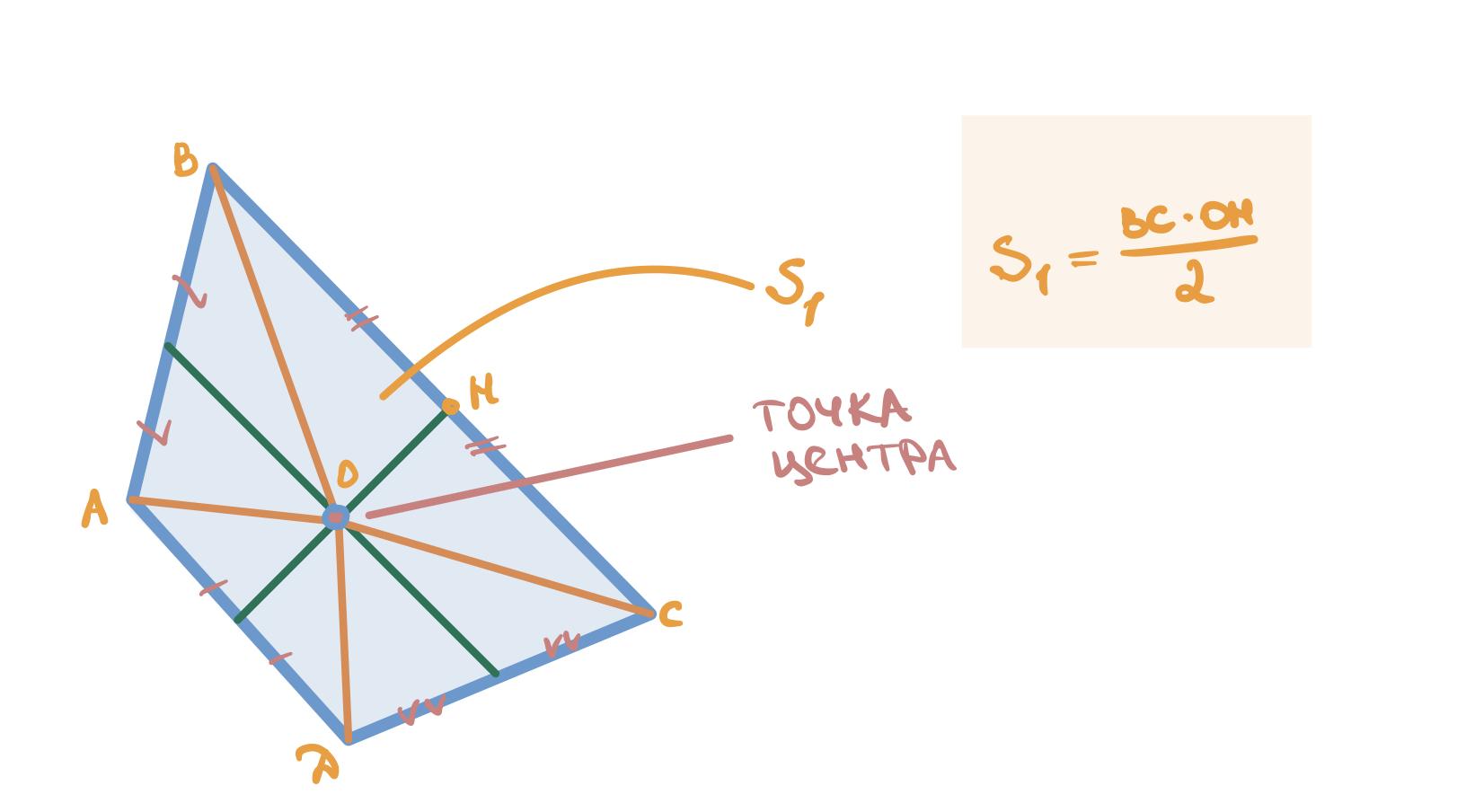
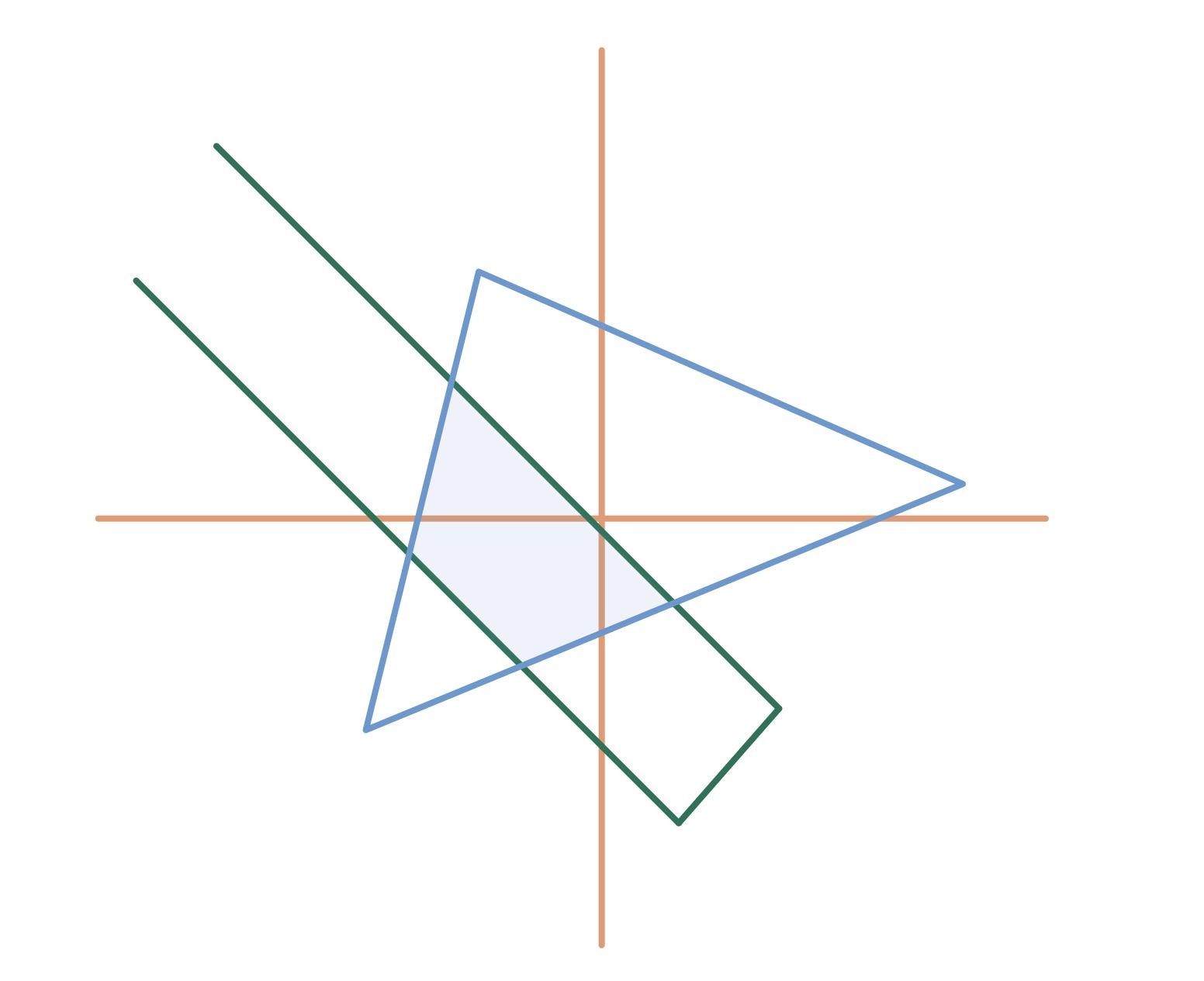
Необходимо найти и выделить такую пару "широкий луч"-треугольник, что фигура, находящаяся внутри "широкого луча" и треугольника, имеет максимальную площадь.

1. **Математическая модель**

Как найти площадь выделенной фигуры?

В связи с произвольностью формы, буду делать так:

считаю сумму площадей треугольника из каждой стороны и центра.

Полная площадь фигуры - сумма всех таких треугольников.Новую фигуру сравниваю с предыдущей, и, при большей площади, запоминаю новое число. 

Теперь приступим к решению самой задачи. Точка О является центром фигуры.

Введём обозначения:

* h - высота BOC
* a – BC (сторона, к которой проведена высота)
* s = (a\*b)/2; - площадь
* так же считается для других треугольников, составленных вершиной и каждой стороной фигуры
* сумма этих площадей S

лизирую другие такие многоугольники и, если площадь новой фигуры больше предыдущей, запоминаю новое значение.

В конце выявляется фигура максимальной площади, это и есть ответ. Пара - широкий луч и треугольник – выделяется цветом, ответ закрашивается.

1. **Анализ используемой структуры данных**

* задаче требуется использовать информацию о конкретных треугольниках. Отметим, что данные могут изменяться во время работы программы, у пользователя есть возможность как добавить новые треугольники, так и удалит все предыдущие. В связи с этим рационально использование динамического списка, как способа хранения данных. Все входящие данные являются вещественными числами, поэтому имеет смысл использовать в решении тип данных double.

1. **Выбор метода решения**

Считываем построчно данные из файла по каждому треугольнику, извлекаем из них все необходимые координаты. Сохраняем их в динамический список как объекты класса треугольник.

После чтения исходных данных и сохранения их в массиве перебираем все пары широкий луч-треугольник, чтобы найти искомую. Каждый раз сравниваем площадь, являющуюся пересечением широкого луча и треугольника с максимальной площадью (изначально задаём её как 0). Если она больше предыдущей максимальной, то мы запоминаем эту новую пару.

Перебрав все пары, выводим ответ. Т. е. меняем цвет искомых широкого луча и треугольника и закрашиваем полученную путем пересечения фигуру.

1. **Листинг программы**

public void addTriangle(double x1, double y1, double x2, double y2, double x3, double y3) {

triangles.add(new Triangle(new Vec2(x1, y1), new Vec2(x2, y2), new Vec2(x3, y3)));

}

//создание треугольника

public void addBeam(double x1, double y1, double x2, double y2)

{

beams.add(new Beam(new Vec2(x1, y1), new Vec2(x2, y2)));

}

//создание широкого луча

public void solve() {

Triangle tri = new Triangle();

Beam bea = new Beam();

for (Triangle first : triangles) {

for (Beam second : beams)

{

Polygon p = first.intersect(second);

if (p.area > polygon.area) {

polygon = p;

tri = first;

bea = second;

}

}

}

// Сравниваем площади пересечения и выделяем новые фигуры, если площадь оказывается больше

for (Triangle triangle : triangles) {

if (tri == triangle)

triangle.setType(CHOOSEN);

else

triangle.setType(UNUSED);

}

for (Beam beam : beams)

{

if (beam == bea)

beam.type = CHOOSEN;

else

beam.type = UNUSED;

}

}

// если мы уже выбрали нужные фигуры, дальше мы их не используем

try {

File file = new File(TRIANGLES\_FILE\_NAME);

Scanner sc = new Scanner(file);

while (sc.hasNextLine()) {

Triangle tri = new Triangle(

new Vec2(sc.nextDouble(), sc.nextDouble()),

new Vec2(sc.nextDouble(), sc.nextDouble()),

new Vec2(sc.nextDouble(), sc.nextDouble())

);

sc.nextLine();

tri.setType(TriangleType.valueOf(sc.nextLine()));

triangles.add(tri);

}

} catch (Exception ex) {

System.out.println("Ошибка чтения из файла: " + ex);

}

// Загрузка треугольников

try {

File file = new File(POLYGON\_FILE\_NAME);

Scanner sc = new Scanner(file);

while (sc.hasNextLine()) {

polygon.points.add(new Vec2(sc.nextDouble(), sc.nextDouble()));

sc.nextLine();

}

} catch (Exception ex) {

System.out.println("Ошибка чтения из файла: " + ex);

}

// Загрузка полигона

try {

File file = new File(BEAMS\_FILE\_NAME);

Scanner sc = new Scanner(file);

while (sc.hasNextLine()) {

Beam beam = new Beam(

new Vec2(sc.nextDouble(), sc.nextDouble()),

new Vec2(sc.nextDouble(), sc.nextDouble()),

new Vec2(sc.nextDouble(), sc.nextDouble()),

new Vec2(sc.nextDouble(), sc.nextDouble())

);

sc.nextLine();

beam.type = TriangleType.valueOf(sc.nextLine());

beams.add(beam);

}

// Загрузка широких лучей

public void render(GL2 gl) {

for (Triangle tri : triangles)

if (tri.type != CHOOSEN)

tri.render(gl);

for (Beam beam : beams)

if (beam.type != CHOOSEN)

beam.render(gl);

for (Triangle tri : triangles)

if (tri.type == CHOOSEN)

tri.render(gl);

for (Beam beam : beams)

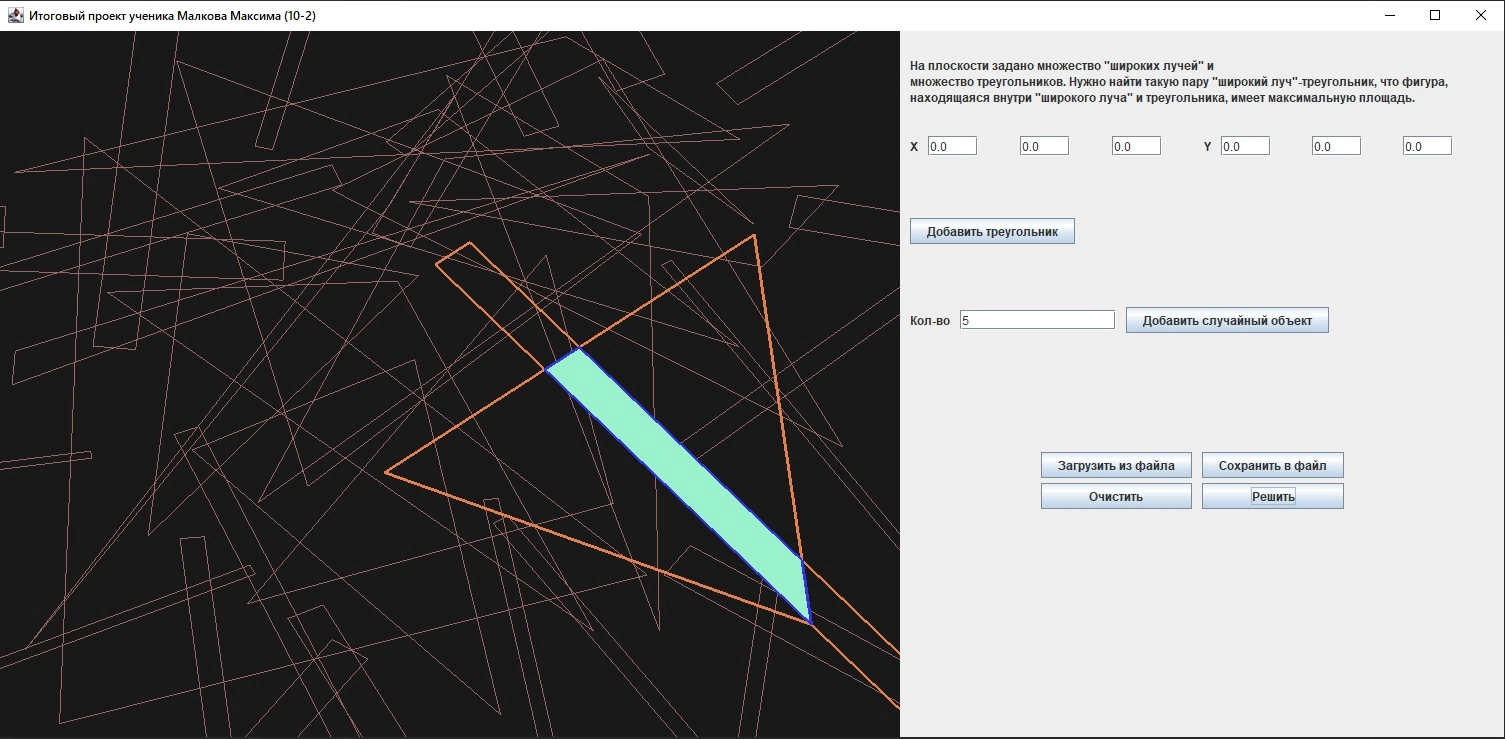
if (beam.type == CHOOSEN)

beam.render(gl);

polygon.render(gl);

}

//Закрашиваем треугольники и широкие лучи в соответствии с их типом : "CHOOSEN" красим цветом, то есть получаем ответ

**7.Пример работы программы**

**8.Анализ правильности решения**

В связи с тем, что задача геометрическая, исходные данные и ответ имеют графическое представление, т. е. для доказательства правильности решения достаточно несколько раз получить верный чертёж.

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |