# Государственное бюджетное общеобразовательное учреждение "Президентский физико-математический лицей № 239"

# **Нахождение наибольшей площади пересечения множеств широких лучей и треугольников**

Годовой проект по информатике

Работу выполнила Ученица 10-7 класса

Чуприна Анна

Санкт-Петербург

2021

## 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. В качестве ответа: выделить найденные "широкий луч" и треугольник, выделить контур фигуры, которая ограничивает точки внутри найденного «широкого луча» и треугольника, желательно выделить внутреннее пространство фигуры ("залить цветом"). Мне нужно, рассмотрев два множества: множество треугольников и множество широких лучей, найти фигуру, которая находится на пересечении двух множеств и при этом имеет максимальную площадь среди всех подобных себе.

## Уточнение исходных и выходных данных и ограничений на них

### Исходные данные

Входными данными являются множество «широких лучей» и множество треугольников. **"Широкий луч" ("полуполоса").** Бесконечная фигура. "Прямоугольник с одной из сторон, находящейся в бесконечности". Задается двумя точками, являющимися вершинами стороны прямоугольника. Направление "луча" считать "направо на 90 градусов относительно вектора, проведенного из первой точки во вторую". В проекте класс именуется «Ray».

**Треугольник.** Задается тремя точками-вершинами. Точки попарно не совпадают. В проекте класс именуется «Triangle».

- Элементы обоих множеств я задаю с помощью объектов класса «Vector». Класс «Vector» - класс, который задаётся двумя переменными типа Double. Можно сказать, что класс «Vector» задаёт геометрический объект – направленный отрезок. Класс «Vector» я использую в своей работе, т.к. с помощью него легче выполнять задание фигур. Например, не нужно делать сложные расчёты о вычислении положения отложенной точки (её координат), когда можно просто умножить вектор на число и сразу же получить координаты этой точки как координаты нового вектора.

- Координаты вершин фигур ограничены. Если a – искомая фигура (широкий луч/треугольник), то -1<=a.x<=1 и -1<=a.y<=1. Такое ограничения накладываются из-за того, что точки границы рабочего поля программы имеют координаты x и y такие, что (|x|=1 и |y|<=1) или (|y|=1 и |x|<=1).

### Выходные данные

Необходимо вывести на экран фигура, которая получается в пересечении «широкого луча» и треугольника и при этом имеет максимальную площадь. Это многоугольник, являющийся наибольшей точкой пересечения двух множеств. Задаётся динамическим массивом точек, являющихся его вершинами. Точки принадлежат классу «Vector». Массив является динамическим из-за того, что только по ходу выполнения решения становится понятно, какое количество вершин содержит многоугольник с максимальной площадью. В проекте класс именуется «Polygon». Координаты каждой вершины ограничены. Если i – некая вершина многоугольника, то -1<=i.x<=1 и -1<=i.y<=1.

## Математическая модель

### -Класс «Vector»: (в дальнейшем x и y являются координатами полученной фигуры) Имеет поля x, y.

Нахождение координат вектора

Поворот вектора на произвольный угол

Вычисление длины вектора

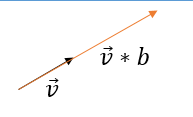
Построение вектора, перпендикулярного данному

90

Сложение данного вектора с вектором b

Разность данного вектора и вектора b

Нормализация вектора - преобразование заданного вектора в вектор в том же направлении, но с единичной длиной.



Умножение вектора на число

Метод, возвращающий выраженный в радианах угол, отсчитываемый против часовой стрелки от положительного направления оси X до точки (x,y). Арктангенс данного угла.

### -Класс «Line». Имеет поля A, B, C – коэффициенты для

задания общего уравнения прямой; точки pointA, pointB, которыми происходит задание прямых.

Проверка прямых на параллельность. В случае параллельности прямых b и c:

Нахождение коэффициентов общего уравнения, заданной по двум точкам

Пересечение данной прямой с прямой t:

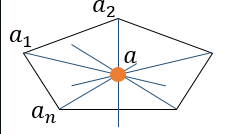
1. Проверка на параллельность (см. метод «isParallel» пред. пункт)
2. Если прямые не параллельны, то можно найти точку пересечения двух прямых:

### -Класс «Triangle»

Нахождение площади S треугольника

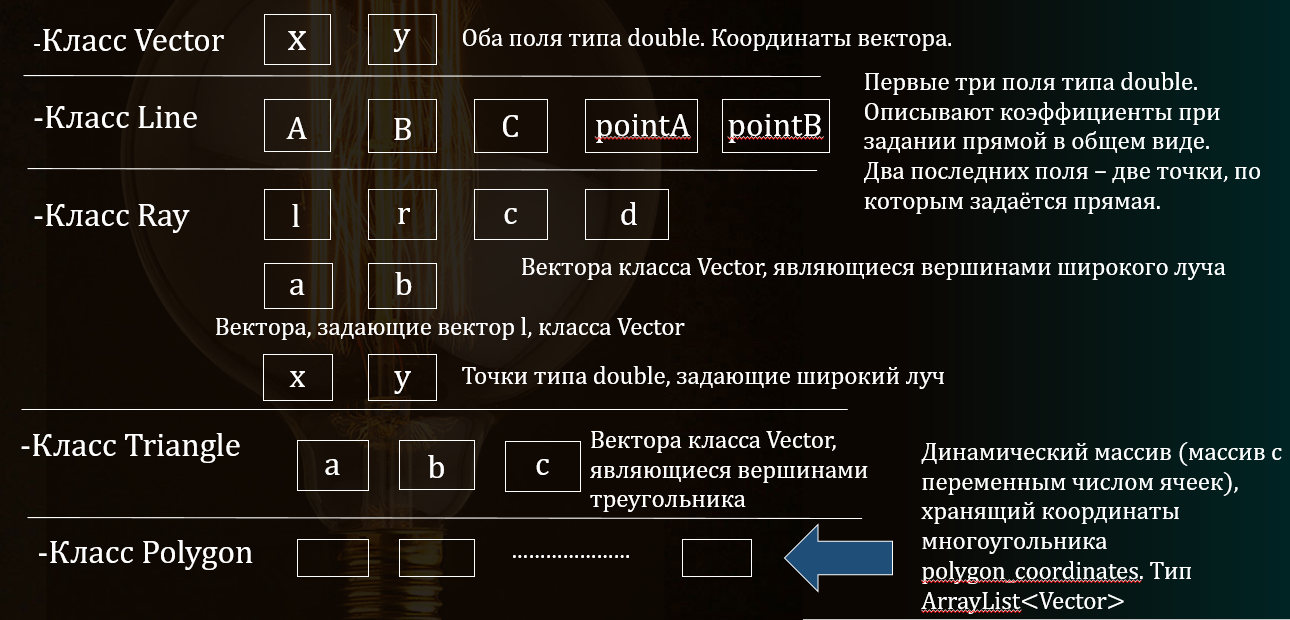
### -Класс «Polygon»

Нахождение геометрического центра многоугольника



## Анализ используемой структуры данных

На картинке представлены поля классов, которые являются частью кода.



Помимо полей в классах, нужно также создать массивы для хранения вершин фигур, являющихся входными данными. Причём, размер массивов, хранящих вершины каждого треугольника и каждого луча строго определены: 3 и 4 соответственно, однако по ходу решения задачи так же придётся хранить точки на координатной плоскости, удовлетворяющие условию конкретного этапа решения. Такие точки я буду заносить в динамические массивы - массивы, размер которых может изменяться во время исполнения программы, а после удалять на каждом этапе неудовлетворяющие условию. На выходе я получу динамический массив точек, являющихся вершинами многоугольника – фигуры пересечения «широкого луча» и треугольника с максимальной площадью. Тип данных всех массивов – переменные класса «Vector»

## Выбор метода решения

1 этап: 1.1. Нахождение прямых, которые задают каждый треугольник и создание массива с ними.

2.2 Нахождение прямых, которые задают каждый широкий луч и создание массива с ними.

2 этап: Пересечение полученных прямых и занесение всех точек пересечения каждого треугольника с каждым «широким лучом» в отдельный массив точек пересечения.

3 этап: Нахождение точек в каждой комбинации пересечений треугольников и лучей, лежащих внутри «широкого луча» и лежащих внутри треугольника, т.к. они точно лежат на границе пересечения или внутри него.

4 этап: После получения только тех точек, которые лежат на границе нашего пересечения или внутри него для каждой комбинации «треугольник – луч», нужно выбрать правильный обход точек для каждого многоугольника. В моей задаче рассматриваются выпуклые многоугольники, а значит, при обходе точек от одной к другой должен получаться выпуклый многоугольник.

5 этап: Нахождение площади каждого полученного многоугольника и отбор многоугольника наибольшей площади.

6 этап: Отрисовка полученной фигуры по вершинам.

## Листинг программы

Листинг метода solve.

public void solve() {

double maxSquare = 0; { вспомогательная переменная }

for (Triangle triangle : triangles) { { рассматриваю каждый треугольник из множества треугольников исходных данных }

Line t1 = new Line(triangle.a.x, triangle.a.y, triangle.b.x, triangle.b.y); { прямая между точками 1 и 2 каждого треугольника }

Line t2 = new Line(triangle.b.x, triangle.b.y, triangle.c.x, triangle.c.y); { прямая между точками 2 и 3 каждого треугольника }

Line t3 = new Line(triangle.c.x, triangle.c.y, triangle.a.x, triangle.a.y); { прямая между точками 3 и 1 каждого треугольника }

for (Line triangleLine : new Line[]{t1, t2, t3}) { { рассматриваю массив прямых каждого треугольника }

for (Ray ray : rays) { { рассматриваю и работаю с каждым лучом }

List<Vector> localPoints = new ArrayList<>(); { массив точек пересечения двух множеств }

Line r1 = new Line(ray.a.x, ray.a.y, ray.b.x, ray.b.y); { прямая между точками 1 и 2 каждого «широкого луча» }

Line r2 = new Line(ray.b.x, ray.b.y, ray.c.x, ray.c.y); { прямая между точками 2 и 3 каждого «широкого луча» }

Line r3 = new Line(ray.c.x, ray.c.y, ray.d.x, ray.d.y); { прямая между точками 3 и 4 каждого «широкого луча» }

Line r4 = new Line(ray.d.x, ray.d.y, ray.a.x, ray.a.y); { прямая между точками 1 и 4 каждого «широкого луча»}

for (Line rectLine : new Line[]{r1, r2, r3, r4}) { { Для каждого массива прямых «широкого луча» }

Vector a = triangleLine.intersection(rectLine); { Нахождение точки a, являющейся точкой пересечения треугольника и «широкого луча» }

if (a != null) { если точка а существует }

localPoints.add(a); { добавить точку а в массив точек пересечения }

}

localPoints.addAll(triangle.getTrianglePoints()); { добавить в массив точек пересечения вершины каждого треугольника }

localPoints.addAll(ray.getRaypoints()); { добавить в массив точек пересечения вершины каждого «широкого луча» }

localPoints.removeIf(point -> !new Polygon(triangle.getTrianglePoints()).isInside(point) ||

!new Polygon(ray.getRaypoints()).isInside(point)); { удалить точку из массива точек пересечения, если она не принадлежит точкам внутри треугольника или если она не принадлежит точкам внутри «широкого луча»}

if (localPoints.isEmpty())

continue; { продолжить, если массив точек пустой }

Polygon t = new Polygon(localPoints); { создаём новый многоугольник, верщинами которого являются отобранные точки }

double localSquare = t.getSquare(); { создание переменной, хранящей площадь многоугольника }

if (localSquare > maxSquare) { { проверка условия: если площадь многоугольника больше максимальной площади фигуры, заданной во всём методе }

maxSquare = localSquare; { то максимальная площадь равна площади многоугольника }

resultPolygon = t; { итоговый многоугольник – многоугольник t максимальной площади }

}

points.addAll(localPoints); { добавить в конечный массив точек многоугольника точки данного многоугольника, площадь которого наибольшая }

}

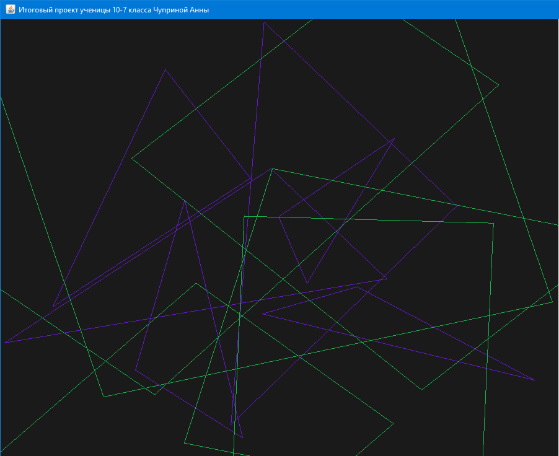
}

}

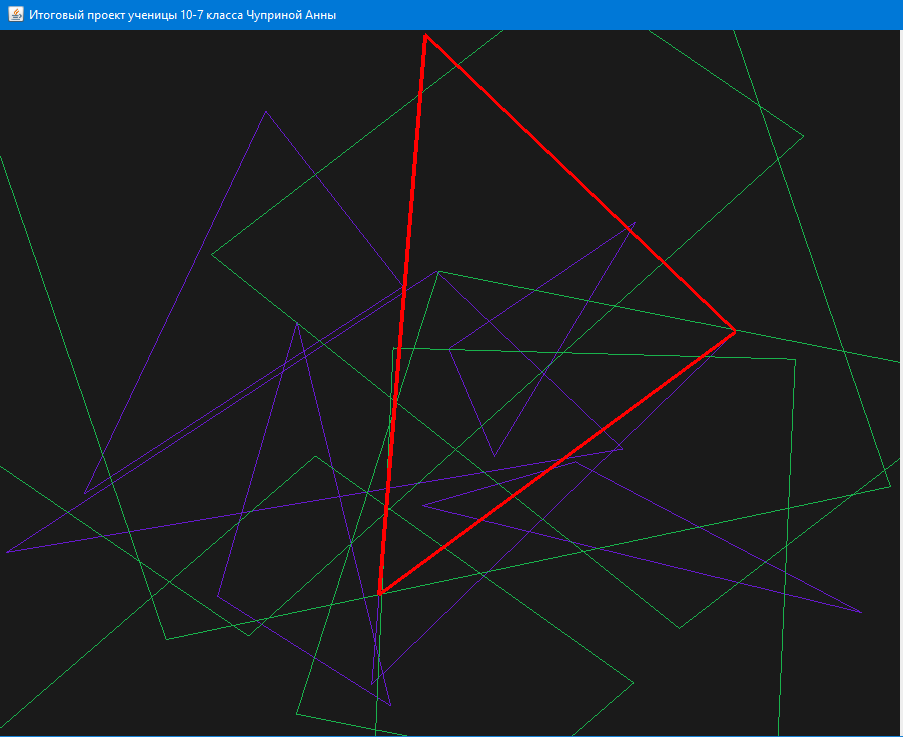
}

## Пример работы программы

### Исходные данные



### Выходные данные

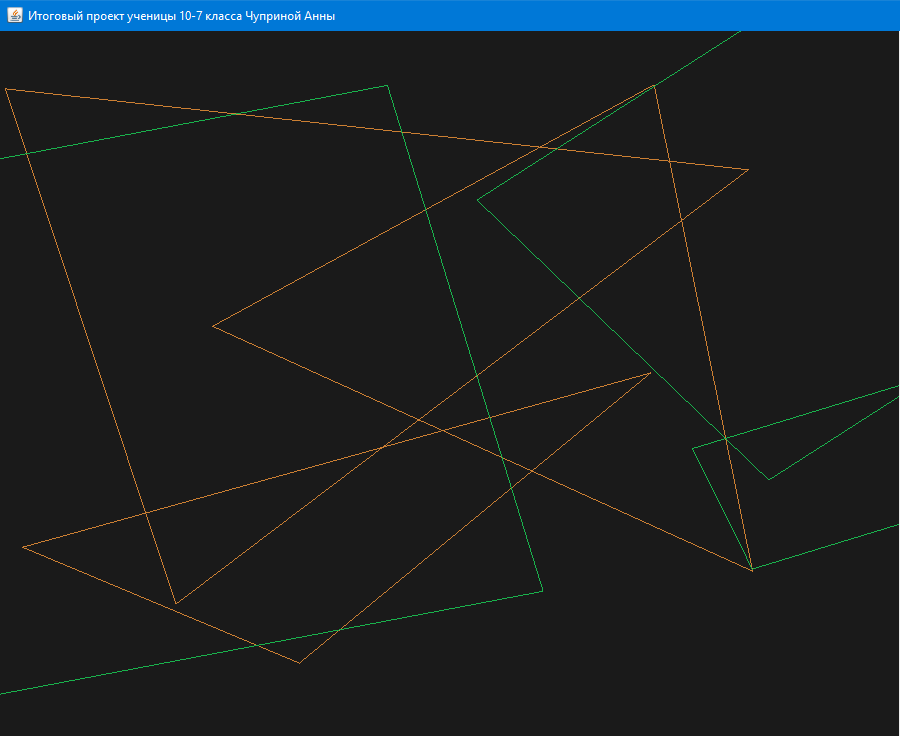
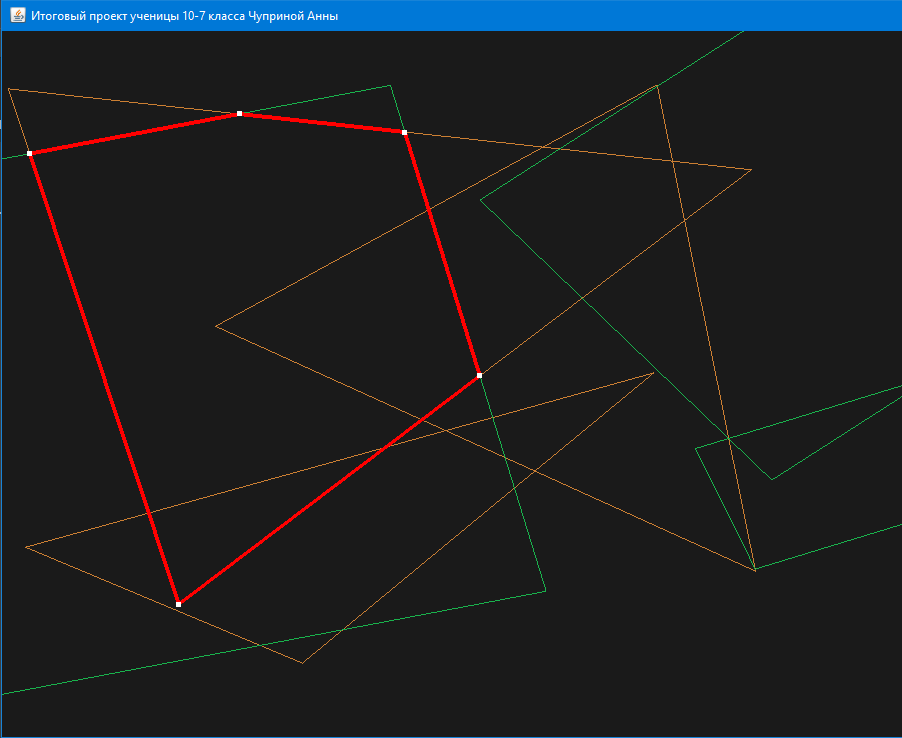


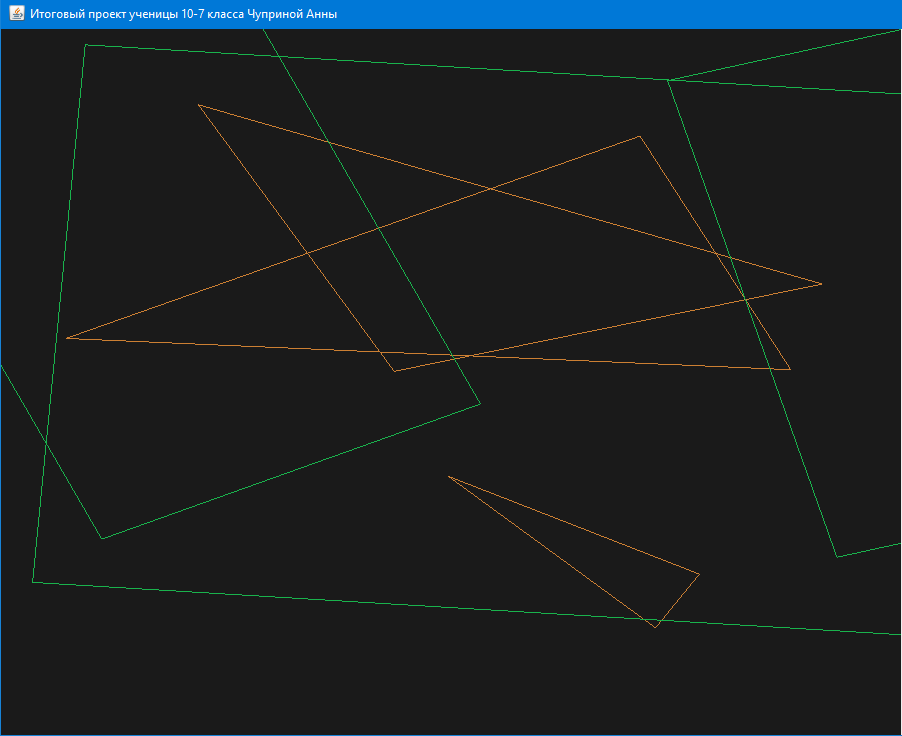
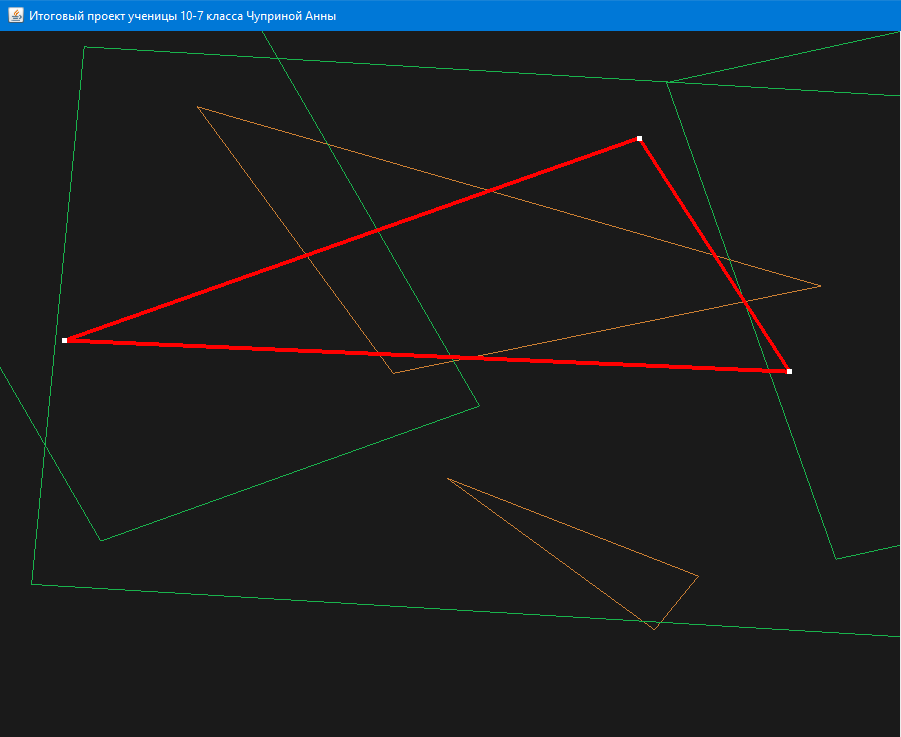
## Анализ правильности решения

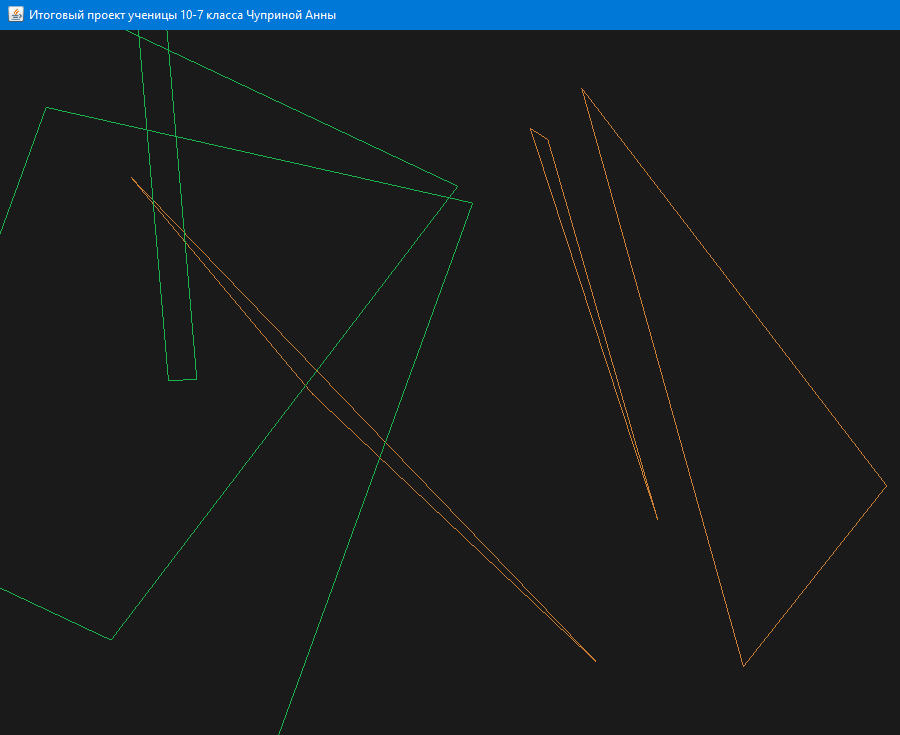
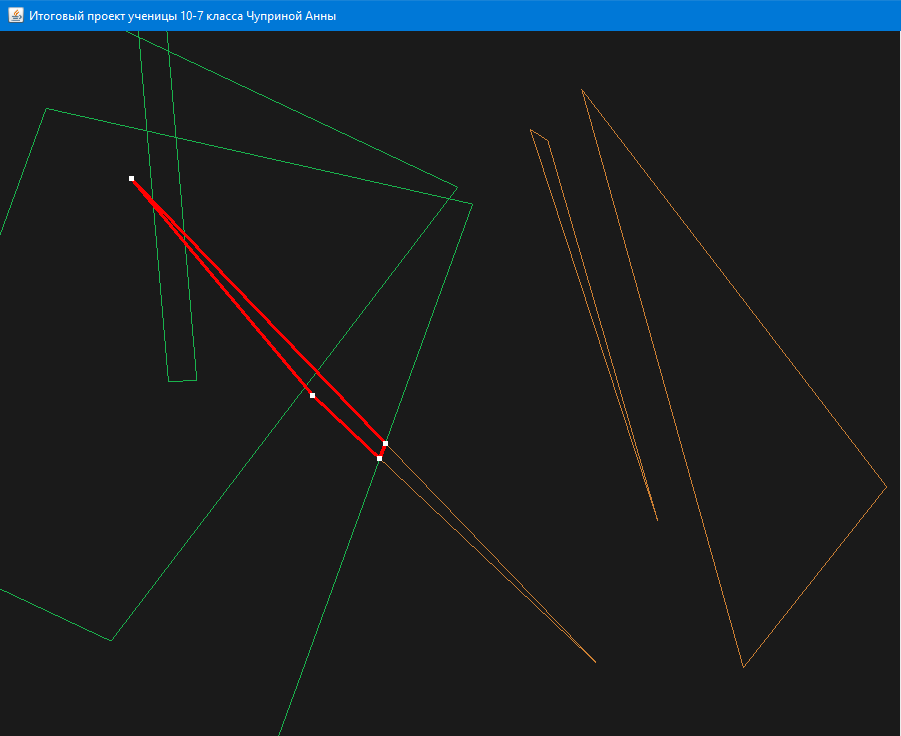
Я считаю, что моя программа работает верно. Для того, чтобы в этом убедиться, рассмотрим несколько наглядных примеров пересечения множеств «широких лучей» и треугольников. Предлагаю рассмотреть множества, состоящие из 3х «широких лучей» и 3х треугольников. Понятно, что решением данной задачи будет являться пятиугольник, образованный самым большим треугольником и «широким лучом», уходящим влево. Нажав кнопку «Решить», можно убедиться в правильности решения, тем самым убедившись в трёх пунктах:

1) программа правильно выбирает обход для полученного многоугольника;

2) программа правильно отбирает точки для выходного многоугольника;

3) программа умеет решать задачи, в которых пересечение имеет более 3х вершин (3 вершины – самый простой случай, т.к. там не нужно выбирать направление обхода). Надо сказать, что точность решения 100%, т.к. программа математически считает площадь треугольника, получая точные значения, в отличие от методов решения, использованных моими одноклассниками и основанных на подсчёте количества точек сетки.

Можно убедиться, что программа отлично справляется с вырожденными случаями, когда, например, весь треугольник лежит в «широком луче». Пример ниже:

Также можно увидеть, что программа рисует и распознаёт другие виды многоугольников, площадь которых является наибольшей, отличные от треугольника и пятиугольника. Например, четырёхугольник: