|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**Лабораторная работа № 5**

|  |  |
| --- | --- |
| **Тема** Реализация и исследование алгоритмов растрового заполнения сплошных областей  **Студент** Якуба Д. В.  **Группа** ИУ7-43  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель** Куров А. В. |  |

Москва

2020 г.

Оглавление

[Цель работы 3](#_Toc38924886)

[Техническое задание 3](#_Toc38924887)

[Теоретическая часть 3](#_Toc38924888)

[Алгоритм со списком рёбер и флагом 3](#_Toc38924889)

[Обрисовка контура 4](#_Toc38924890)

[Заполнение 4](#_Toc38924891)

[Практическая часть 5](#_Toc38924892)

[Программная реализация алгоритма на ЯП Python 5](#_Toc38924893)

[Пользовательский интерфейс 6](#_Toc38924894)

[Демонстрация работы алгоритма 10](#_Toc38924895)

[Исследование временных характеристик 18](#_Toc38924896)

# Цель работы

Реализация и исследование одного из алгоритмов (по заданию преподавателя) растрового заполнения области.

# Техническое задание

В рамках данной работы необходимо реализовать алгоритм заполнения со списком рёбер и флагом (вариант 28) сплошной области.

Необходимо обеспечить ввод произвольной многоугольной области, содержащей произвольное количество отверстий. Ввод (вершин многоугольника) производить с помощью мыши, при этом для удобства пользователя должны отображаться ребра, соединяющие вводимые вершины. Предусмотреть ввод горизонтальных и вертикальных ребер.

Пользователь должен иметь возможность задания цвета заполнения.

Работа программы должна предусматривать два режима – с задержкой и без задержки.

Режим с задержкой должен позволить проследить выполняемую последовательность действий.

(Задержку целесообразно выполнять после обработки очередной строки).

Обеспечить замер времени выполнения алгоритма (без задержки, с выводом на экран только окончательного результата).

# Теоретическая часть

Растровая развёртка сплошных областей – генерация сплошных областей из описаний рёбер или вершин. В методах растровой развёртки пытаются определить в порядке сканирования строк (сканирующая строка – это строка экрана с определённым текущим, можно сказать, значением координаты Y, то есть совокупность пикселей, имеющих одно определённое значение координаты Y), лежит ли точка внутри многоугольника или контура. Эти алгоритмы обычно идут сверху вниз по заданному многоугольнику (то есть от ymin до ymax).

Итак, обратимся к описанию работы алгоритма.

## Алгоритм со списком рёбер и флагом

Алгоритм, использующий список рёбер и флаг, является двухшаговым. Первый шаг – обрисовка контура, в результате чего на каждой сканирующей строке образуются пары ограничивающих пикселов. Второй шаг состоит в заполнении пикселов, расположенных между ограничивающими.

### Обрисовка контура

Для каждого ребра, пересекающего сканирующую строку, следует отметить ближайший правый от точки пересечения пиксел. Цвет метки не должен совпадать с цветом фона или цветом фигуры, которую мы закрашиваем.

**При использовании этого алгоритма следует учитывать, что ребра многоугольника, образующие очень малый угол, могут сливаться, поэтому при обрисовке контура, начиная со второго ребра, надо проверять, не совпадает ли точка нового ребра с уже высвеченным пикселом. Если совпадает, то следует высветить соседний пиксел.**

### Заполнение

Алгоритм строится на прохождении каждой сканирующей строки с закраской текущего пикселя цветом, который первоначально является цветом фона. При встрече на сканирующей строке метки, цвет меняется на цвет фигуры, если цвет совпадал цветом фона, иначе – на цвет фона, и только после данной проверки, текущий пиксел закрашивается текущим цветом.

При этом стоит отметить, что минимальные и максимальные значения y и x, это максимальные и минимальные значения соответствующих координат вершин заданного многоугольника.

#### Алгоритм, записанный на псевдокоде:

для ***yтек*** от ***ymin***до ***ymax*** делать:

**цвет** = цвет фона

для ***xтек***от ***xmin*** до ***xmax*** делать:

если пиксел [***xтек***; ***yтек***] –метка, то:

если **цвет** = цвет фона, то **цвет** = цвет фигуры, иначе **цвет** = цвет фона

закрасить пиксел [***xтек***; ***yтек***] цветом **цвет**

# Практическая часть

## Программная реализация алгоритма на ЯП Python

def leadRoundEdge(img, edge):

if edge[0][1] == edge[1][1]:

return

if edge[0][1] > edge[1][1]:

edge[1], edge[0] = edge[0], edge[1]

stepX = (edge[1][0] - edge[0][0])/(edge[1][1] - edge[0][1])

curX = edge[0][0]

curY = edge[0][1]

while curY < edge[1][1]:

if img.get(int(curX) + 1, curY) != noteColorCheck:

img.put(noteColor, (int(curX) + 1, curY))

else:

img.put(noteColor, (int(curX), curY))

curX += stepX

curY += 1

def leadRoundFigure(img, edgesArray):

for figure in range(len(edgesArray)):

arrEnd = len(edgesArray[figure]) - 1

for i in range(arrEnd):

leadRoundEdge(img, edgesArray[figure][i])

leadRoundEdge(img, edgesArray[figure][arrEnd])

def rasterScanWithFlag(img, edgesArray, sides):

leadRoundFigure(img, edgesArray)

for curY in range(sides[0], sides[2] + 1):

curColor = curColorBackground

invColor = curColorLines

curPointScanString = sides[3]

for curX in range(sides[3], sides[1] + 3):

if img.get(curX, curY) == noteColorCheck:

img.put(curColor, (curPointScanString, curY, curX, curY + 1))

curColor, invColor = invColor, curColor

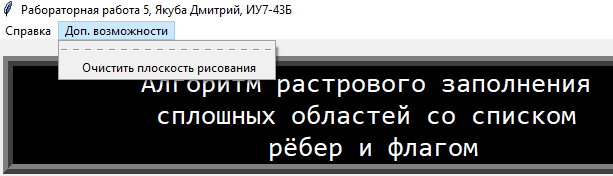
curPointScanString = curX

img.put(curColor, (curPointScanString, curY, curX, curY + 1))

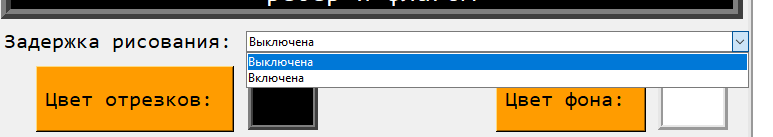
## Пользовательский интерфейс



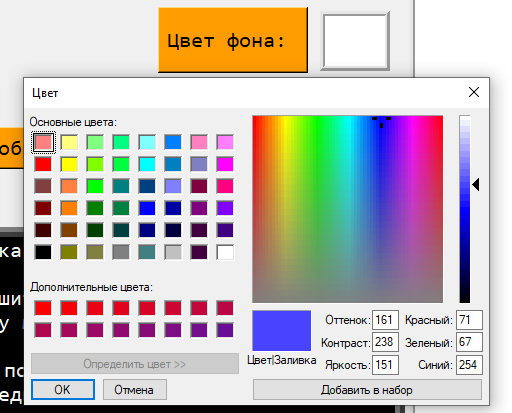
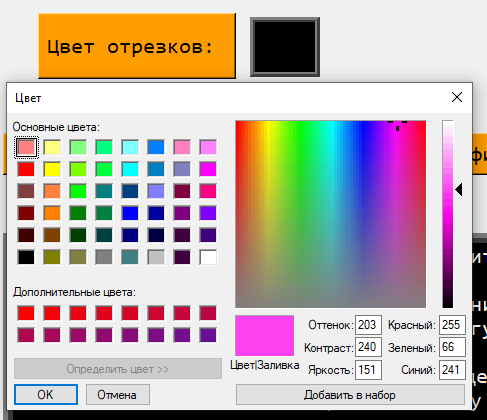
Предусмотрена очистка плоскости рисования

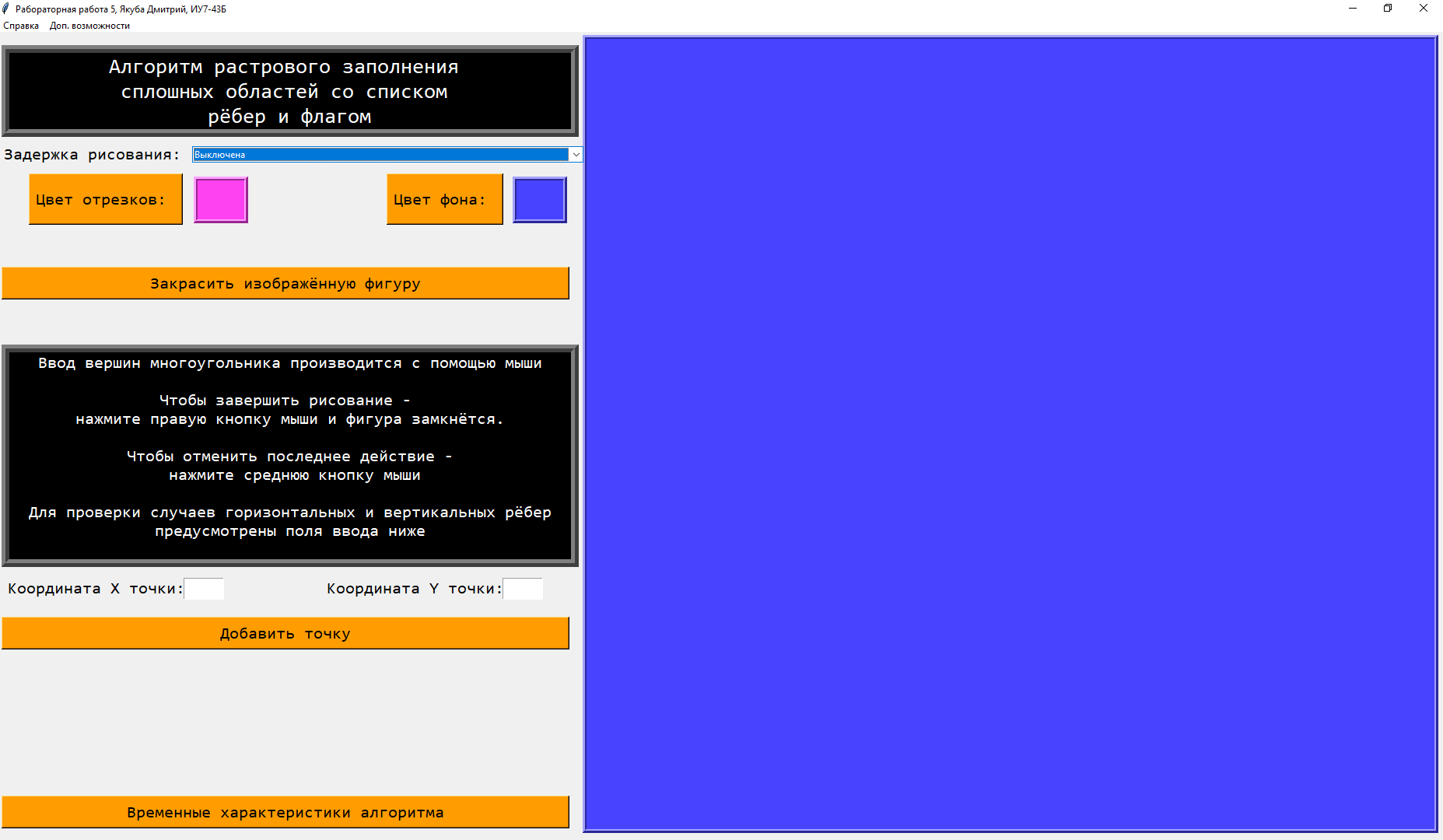


Предусмотрен выбор, каким образом будет отрисовываться закраска фигуры

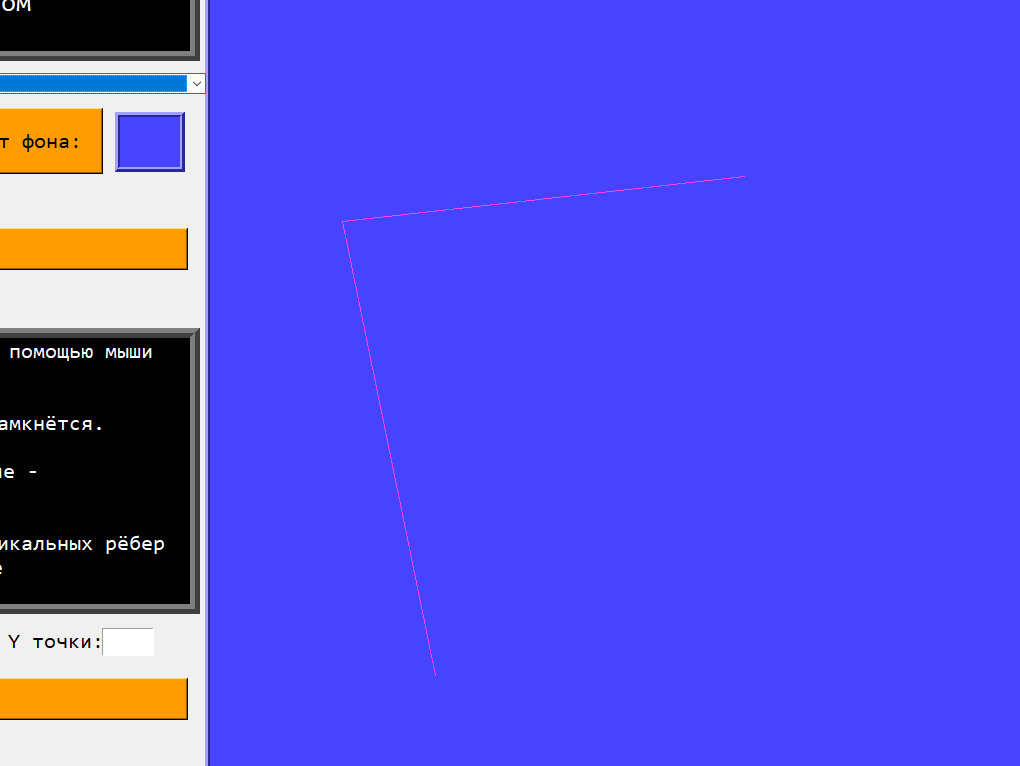


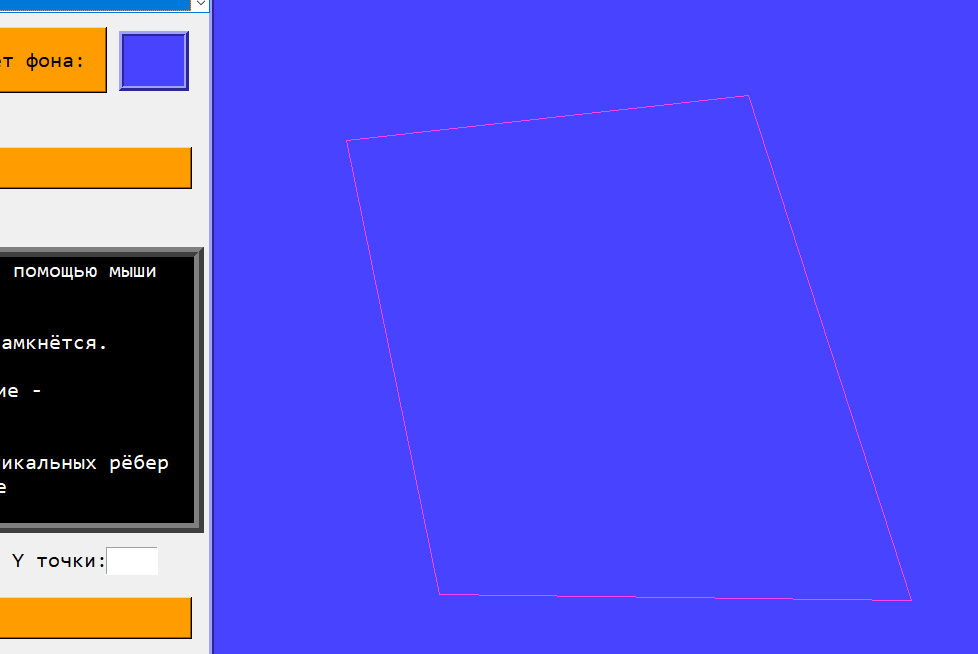
Предусмотрен выбор цвета отрезков и фона плоскости рисования

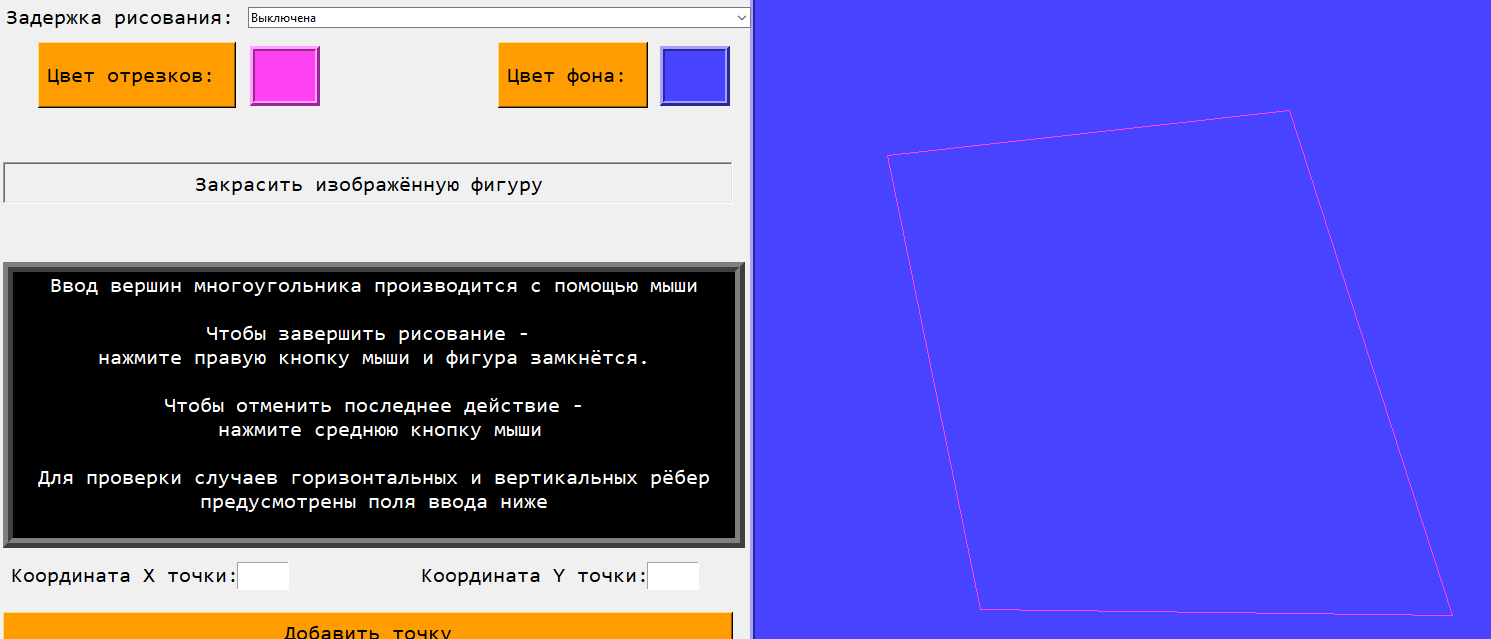




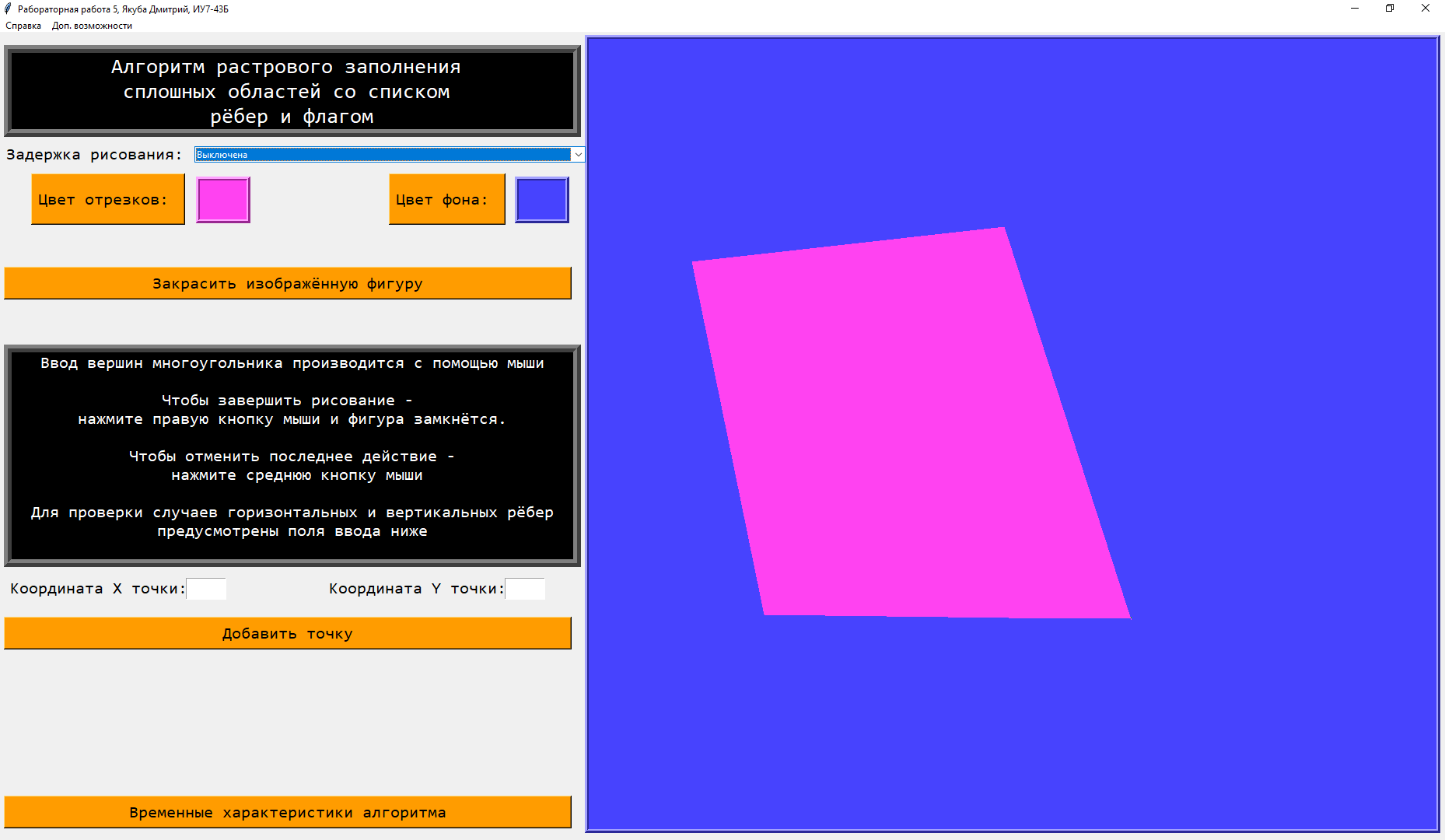
Ввод вершин производится посредством клика правой кнопкой мыши в точке, соответствующей вершине многоугольника. Отрисовка происходит по нажатию кнопки мыши. Отмена последнего действия – средняя кнопка мыши. Замыкание фигуры – правая кнопка мыши.

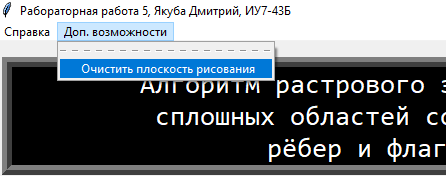


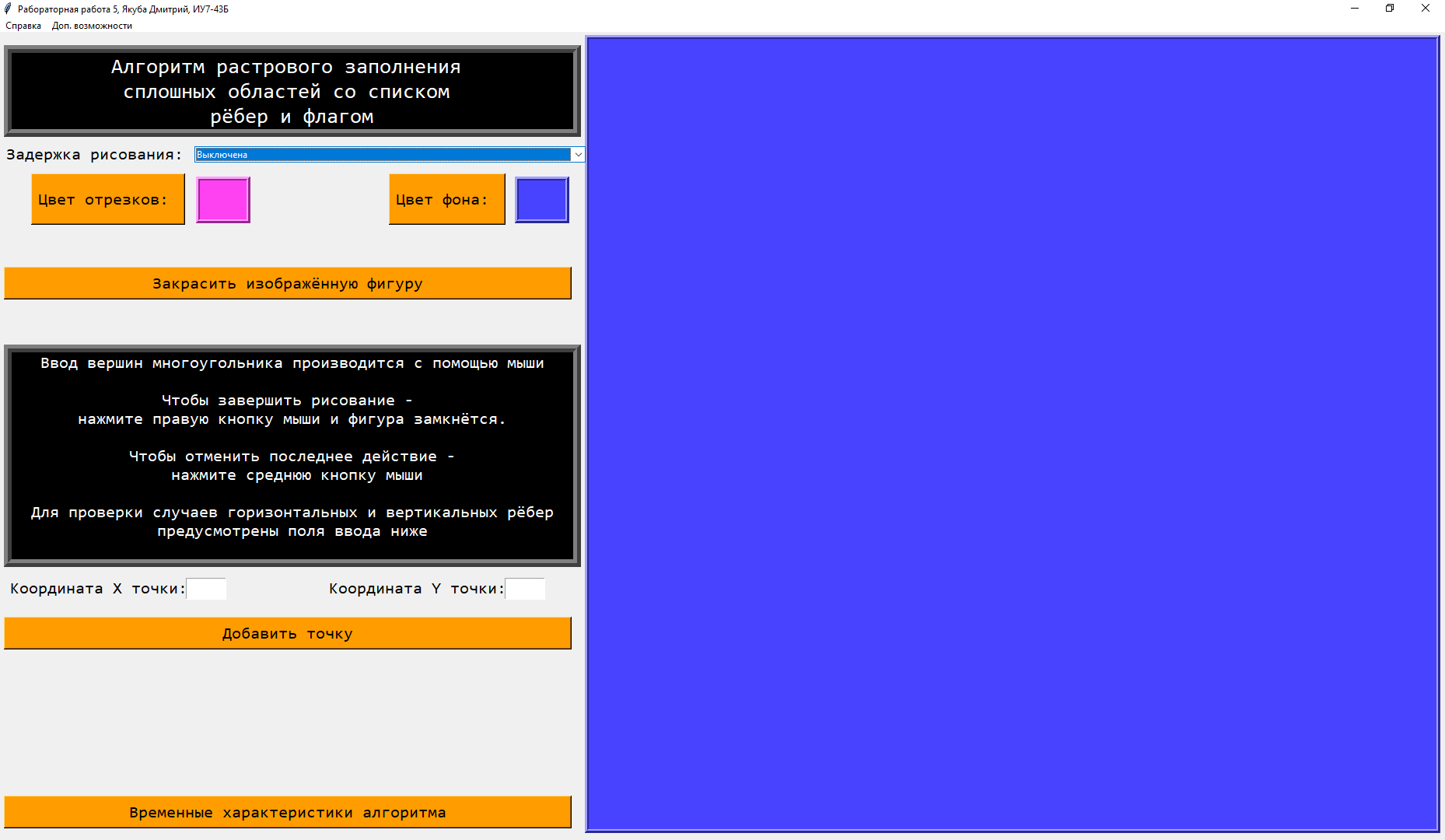




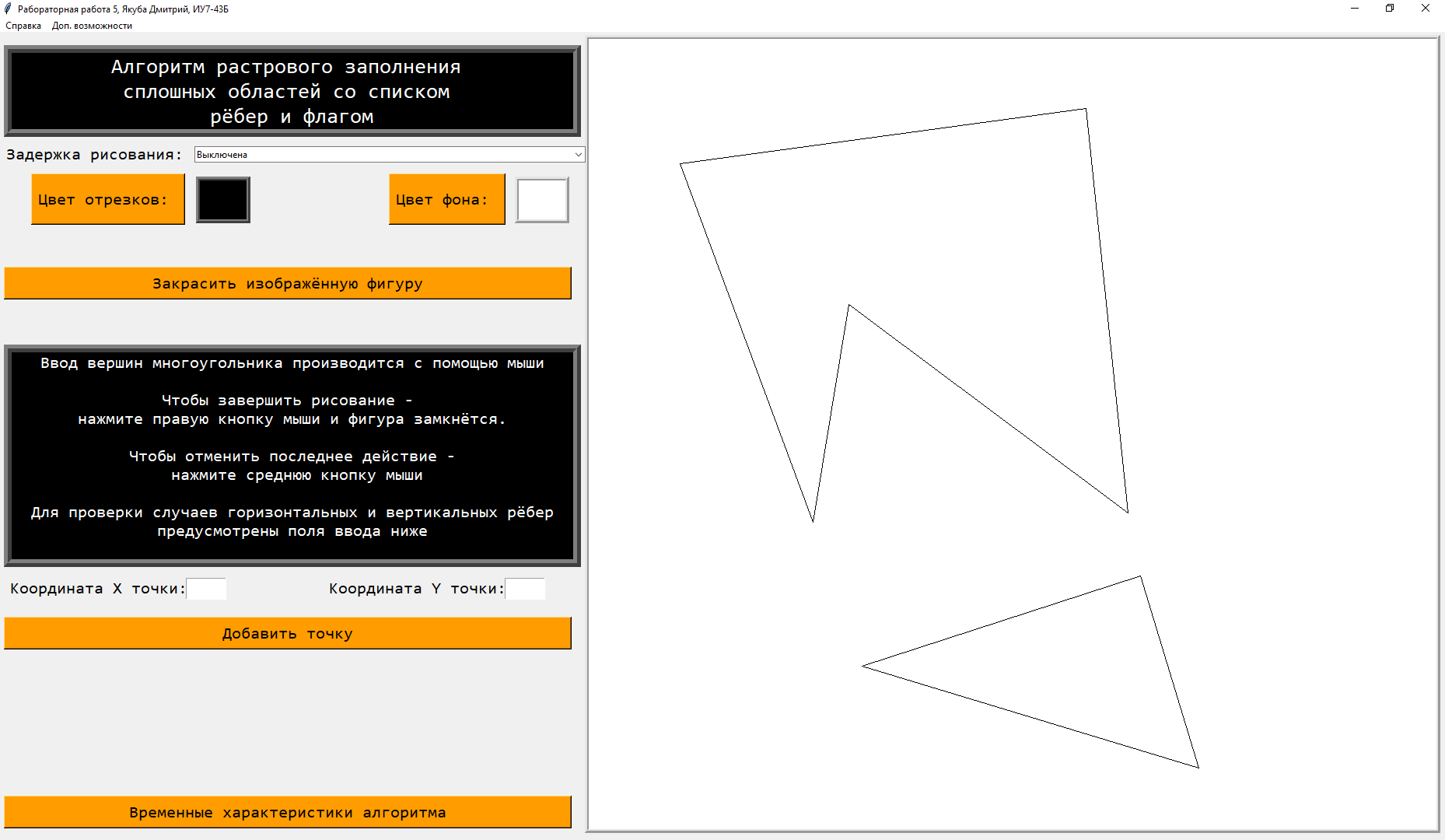
Закраска заданной фигуры происходит по нажатию кнопки «закрасить изображённую фигуру»

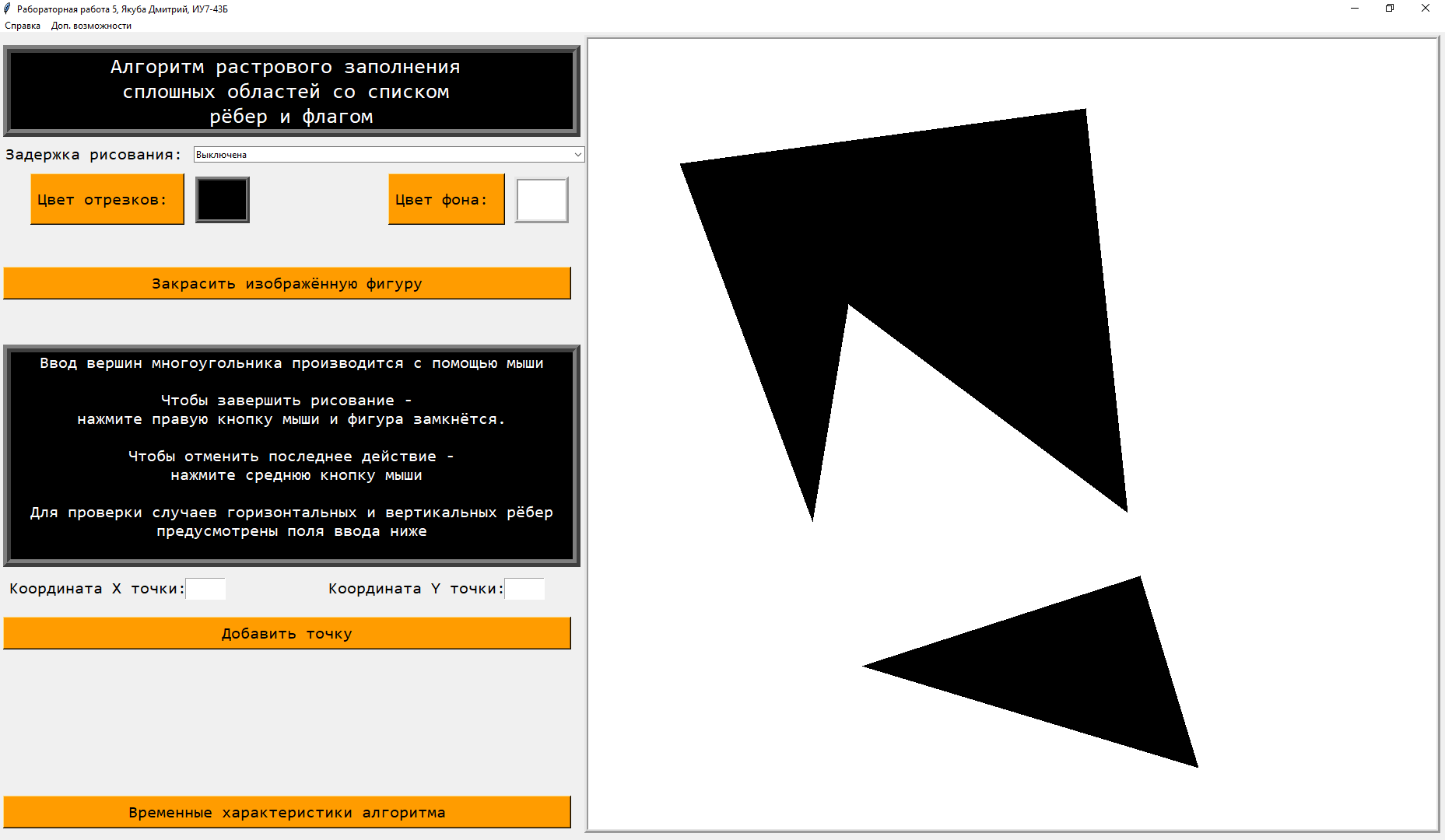


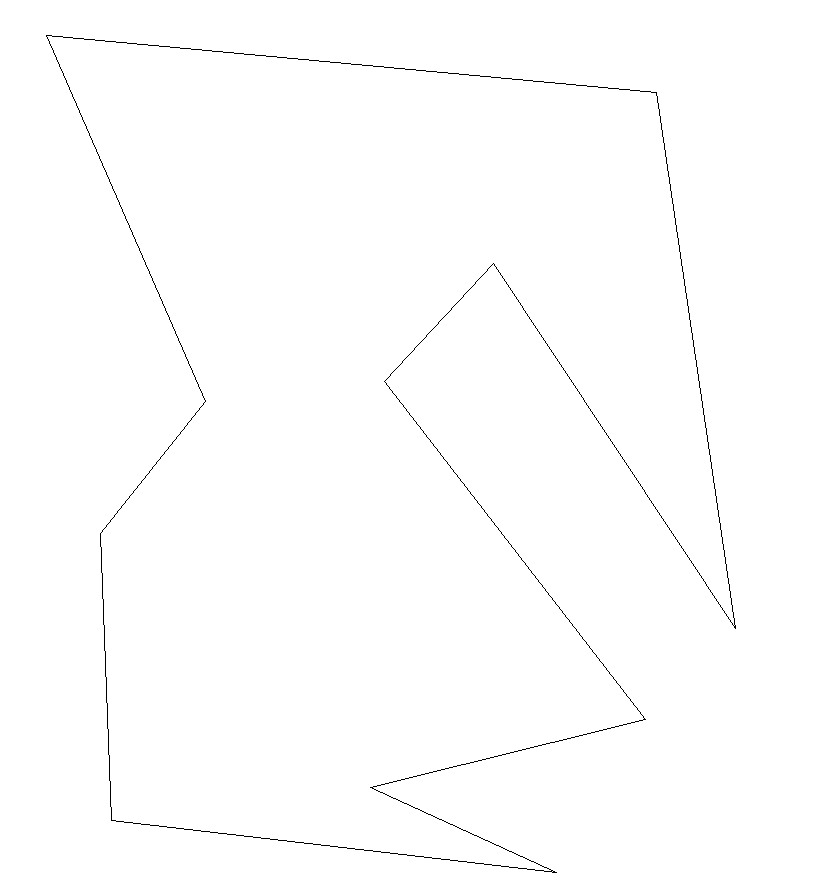
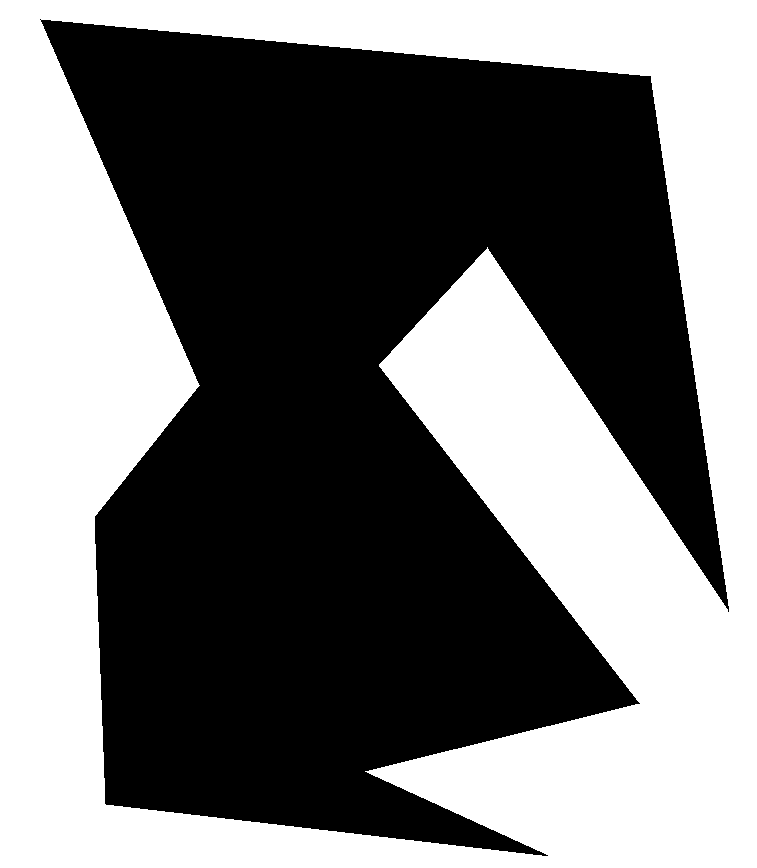


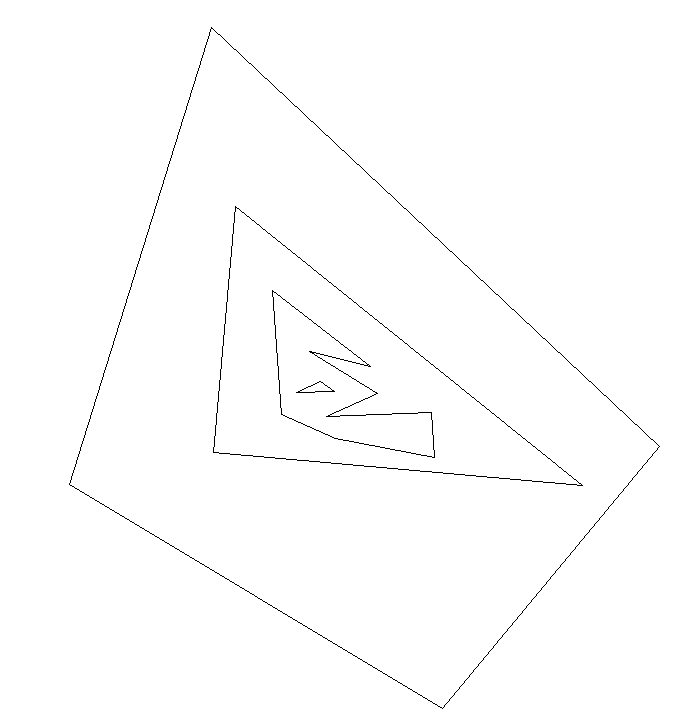
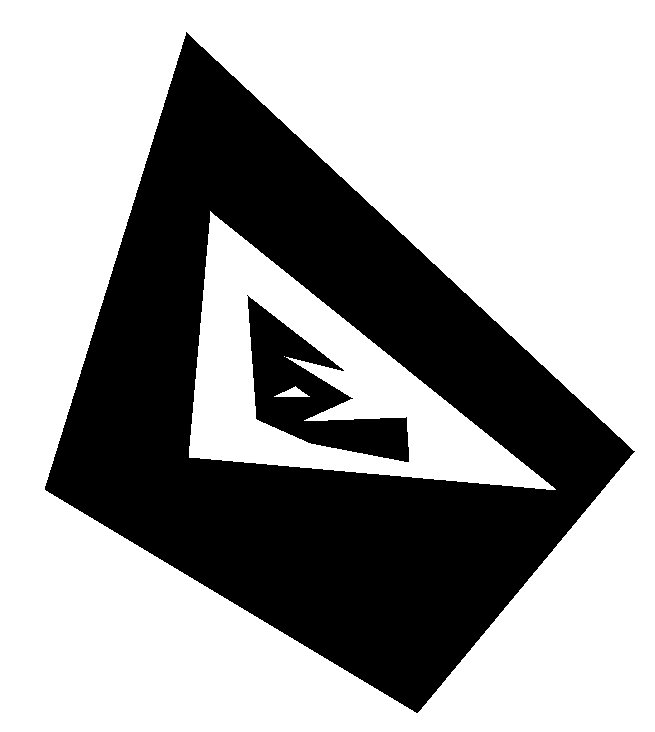


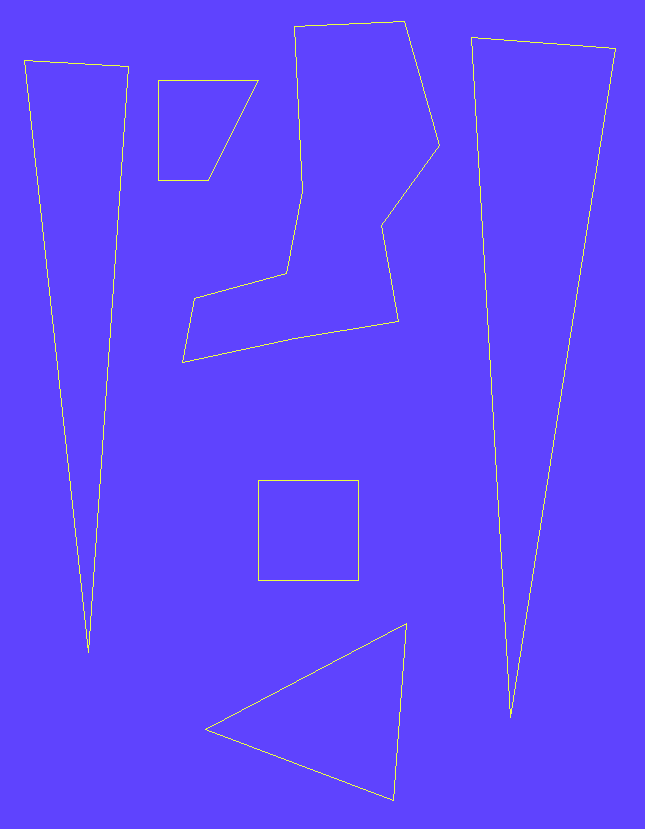
## Демонстрация работы алгоритма



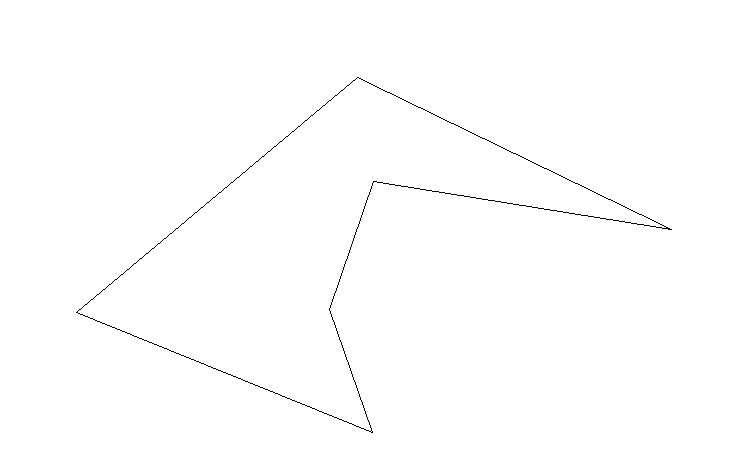


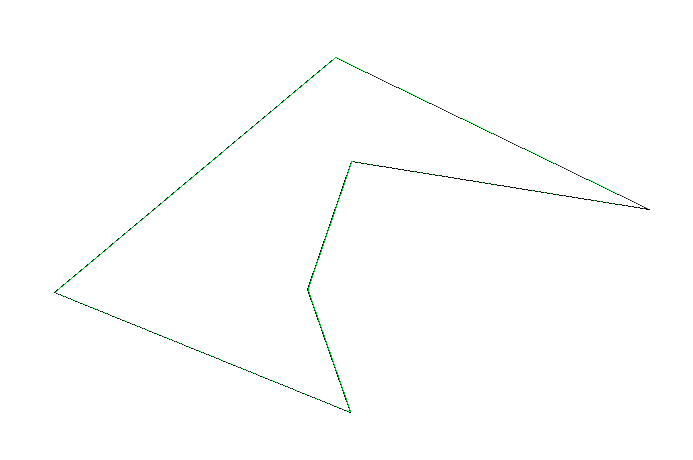
 

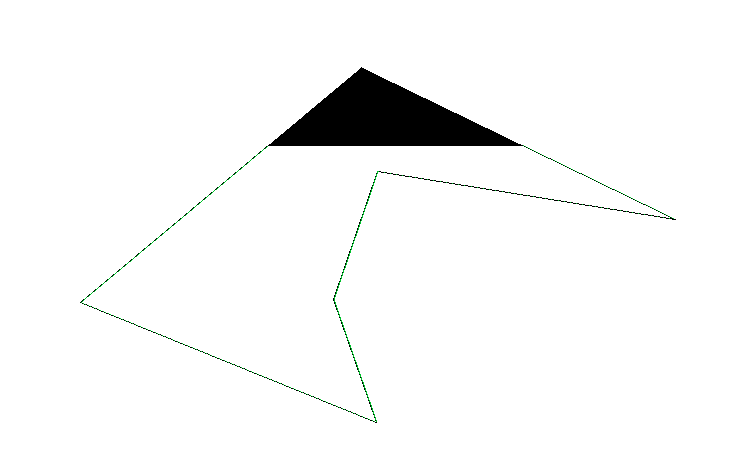
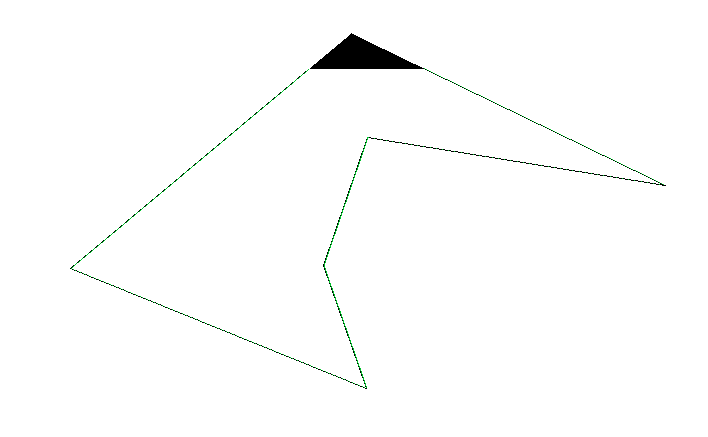
 

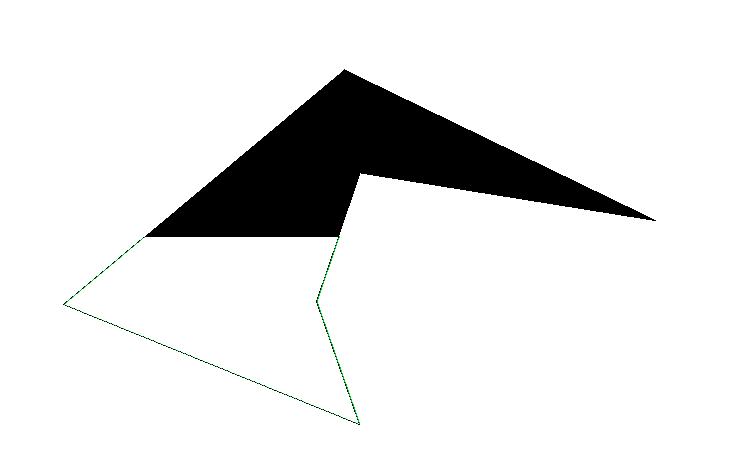
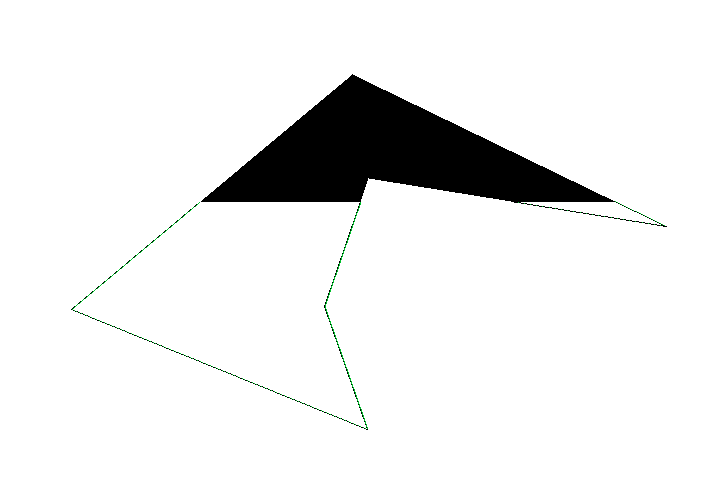


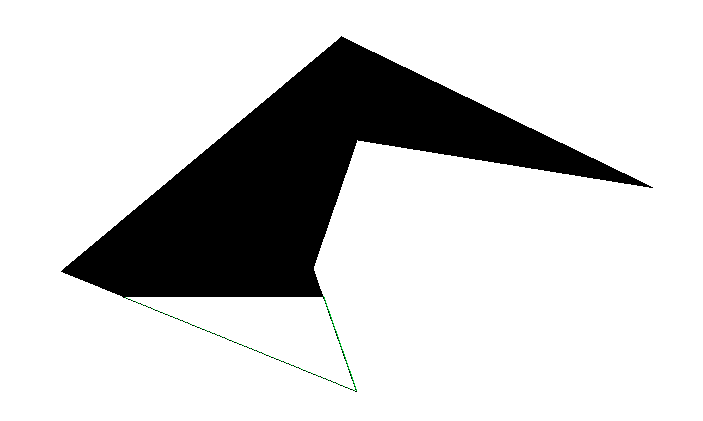
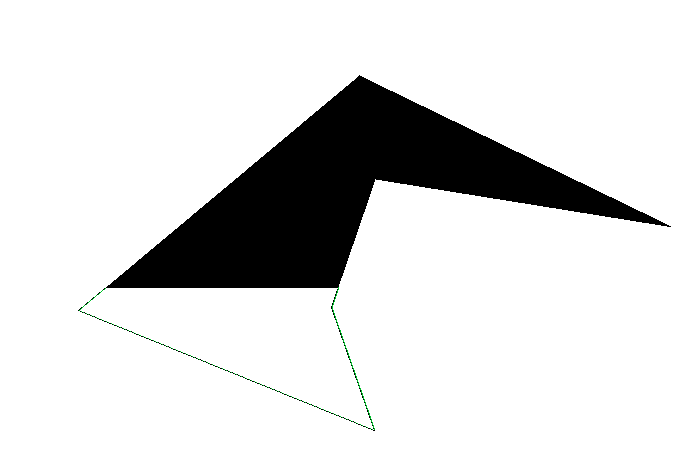
Ниже предоставлены снимки выполнения с задержкой

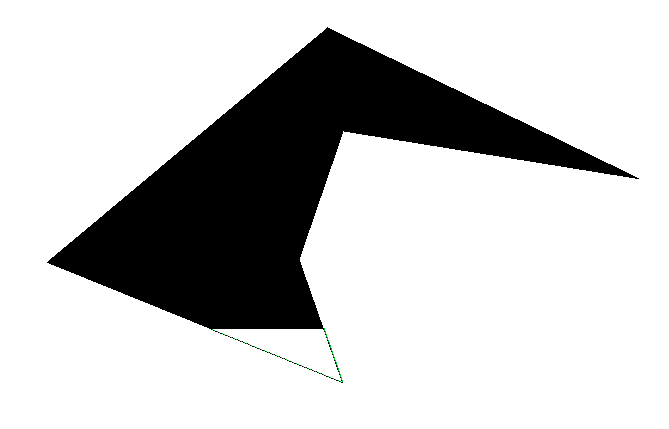


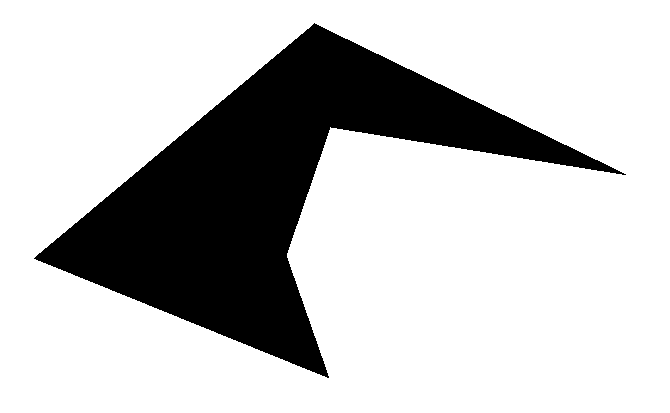




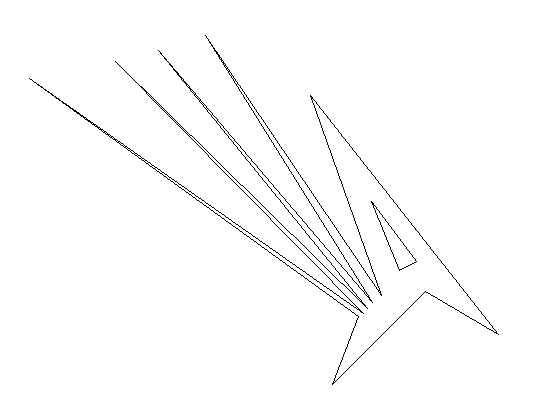


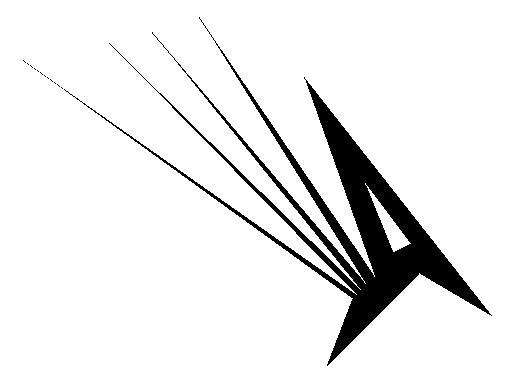
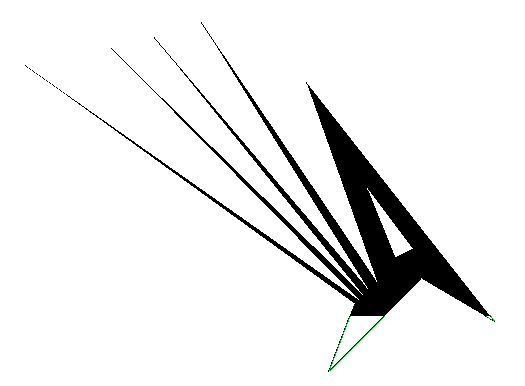
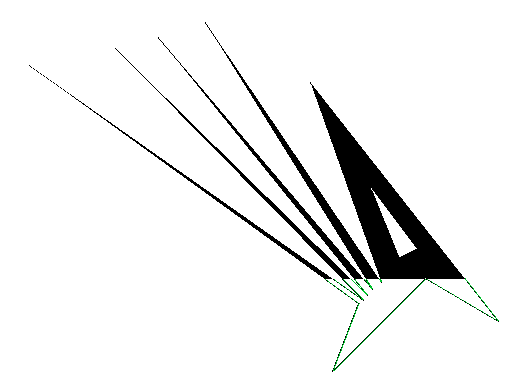
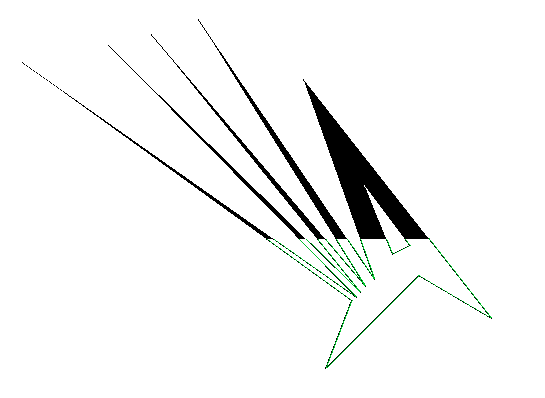
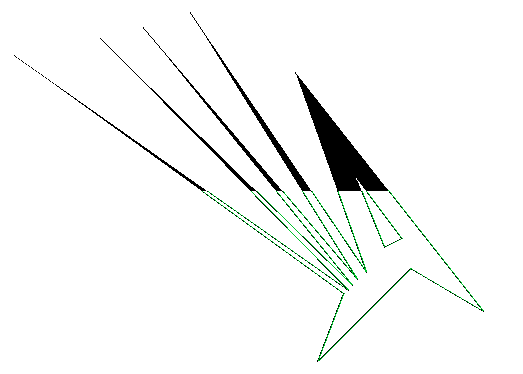
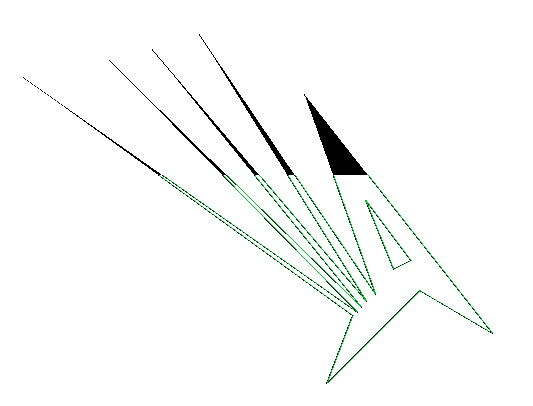
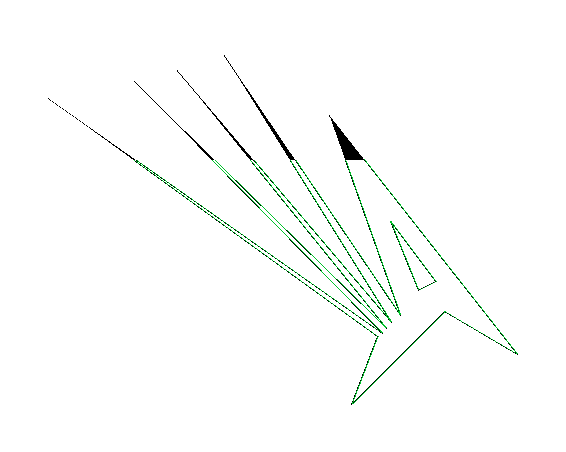
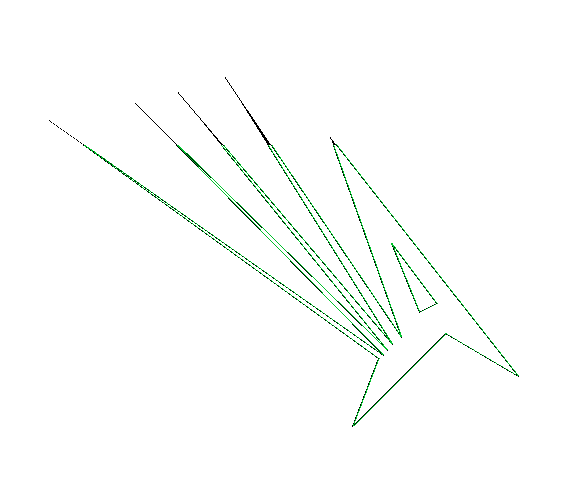
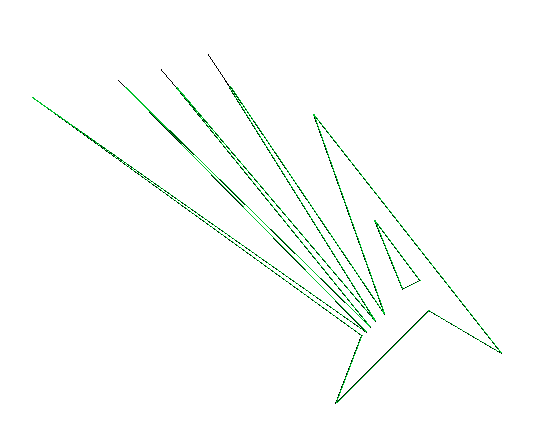
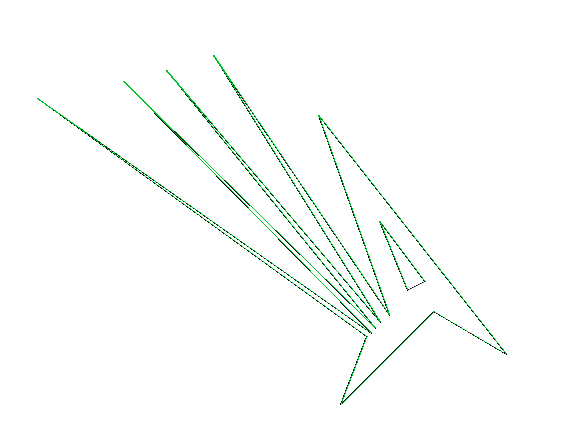




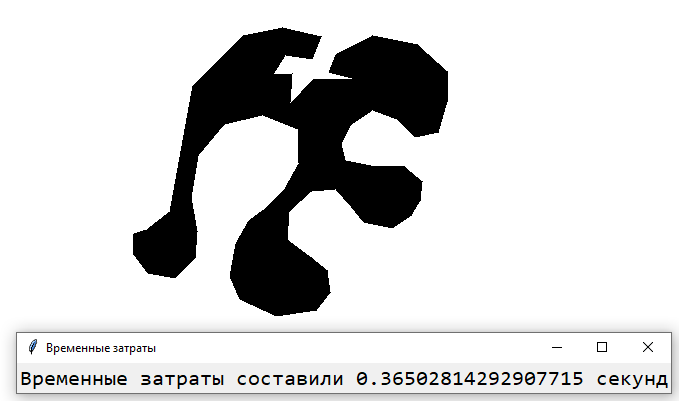


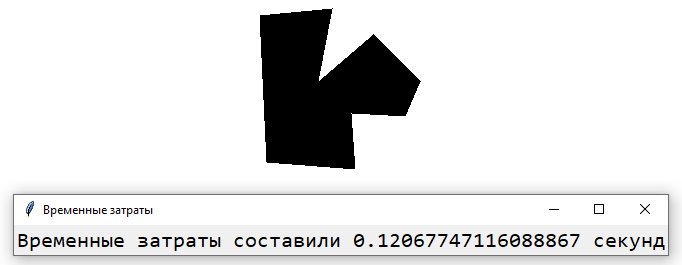
И на более сложный вариант:

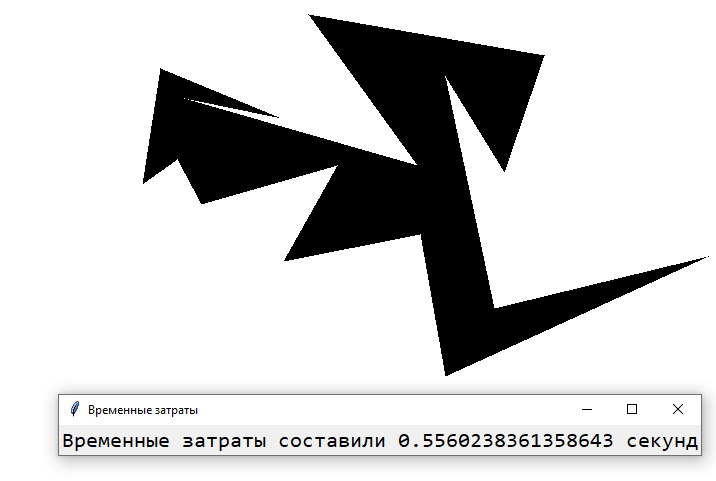


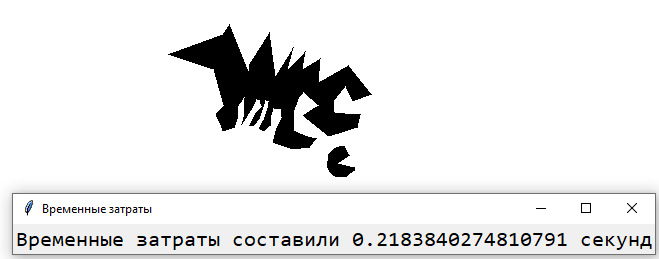


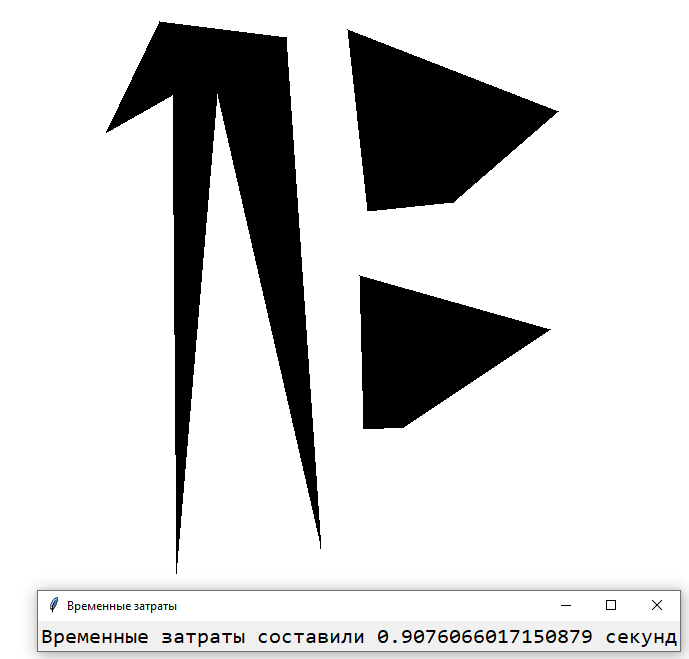
# Исследование временных характеристик

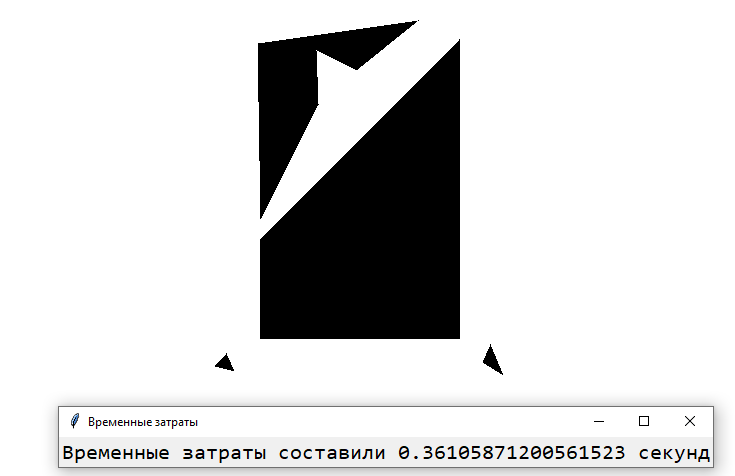












Как видно из предоставленных выше снимков, чем больше размер обрабатываемой плоскости, тем дольше длится работа алгоритма. При этом стоит отметить, что при повышении количества рёбер, разница не так заметна, чем при увеличении размера обрабатываемой плоскости.