|  |  |
| --- | --- |
| 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 № \_\_6\_\_**

|  |  |
| --- | --- |
| **Тема РЕАЛИЗАЦИЯ И ИССЛЕДОВАНИЕ АЛГОРИТМА ПОСТРОЧНОГО ЗАТРАВОЧНОГО ЗАПОЛНЕНИЯ СПЛОШНЫХ ОБЛАСТЕЙ**  **Студент \_Брянская Е.В.\_\_\_\_\_\_\_\_**  **Группа \_ИУ7-42Б\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_Куров А.В.** |  |

Москва.

2020 г.

**Цель работы:** реализация и исследование алгоритма построчного затравочного заполнения.

Задание:

1. реализовать Алгоритм построчного затравочного заполнения сплошных областей
2. продемонстрировать возможность заполнения с помощью затравочного алгоритма произвольной области, ограниченной замкнутой кривой линией.
3. обеспечить замер времени выполнения алгоритма (без задержки, с выводом на экран только окончательного результата).

Необходимо предусмотреть:

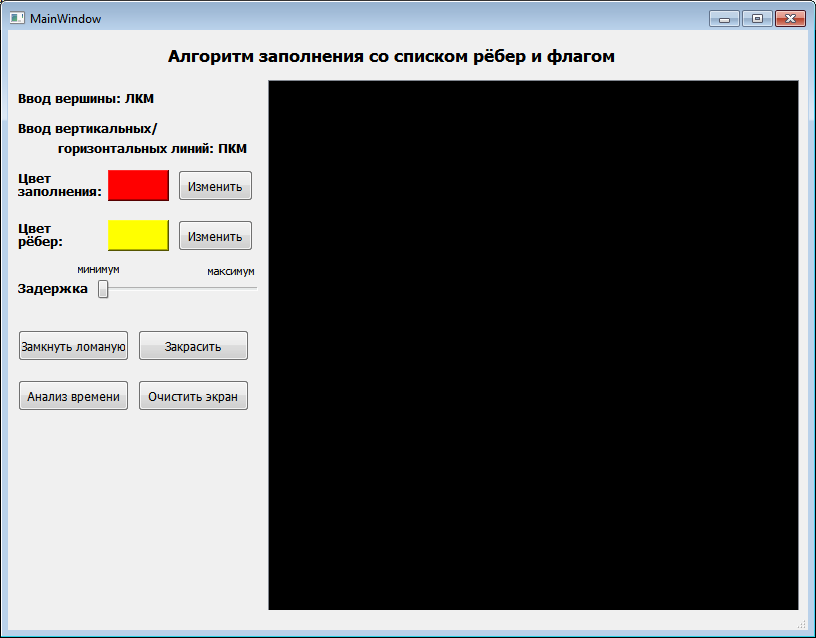
1. Необходимо обеспечить ввод произвольной многоугольной области, содержащей произвольное количество отверстий.

Ввод (вершин многоугольника) производить с помощью мыши, при этом для удобства пользователя должны отображаться ребра, соединяющие вводимые вершины. Предусмотреть ввод горизонтальных и вертикальных ребер.

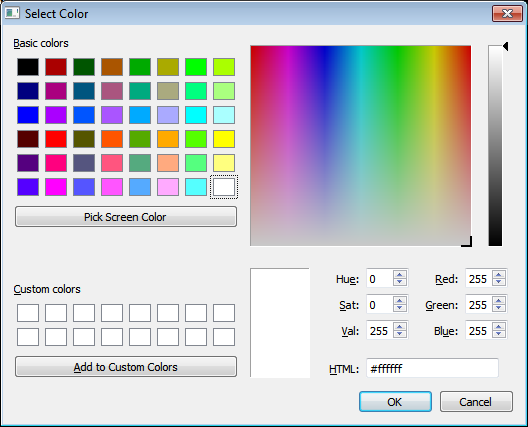
1. Необходимо предусмотреть ввод затравочной точки
2. Пользователь должен иметь возможность задания цвета заполнения.
3. Работа программы должна предусматривать два режима – с задержкой и без задержки.
4. Режим с задержкой должен позволить проследить выполняемую последовательность действий. (Задержку целесообразно выполнять после обработки очередной строки).

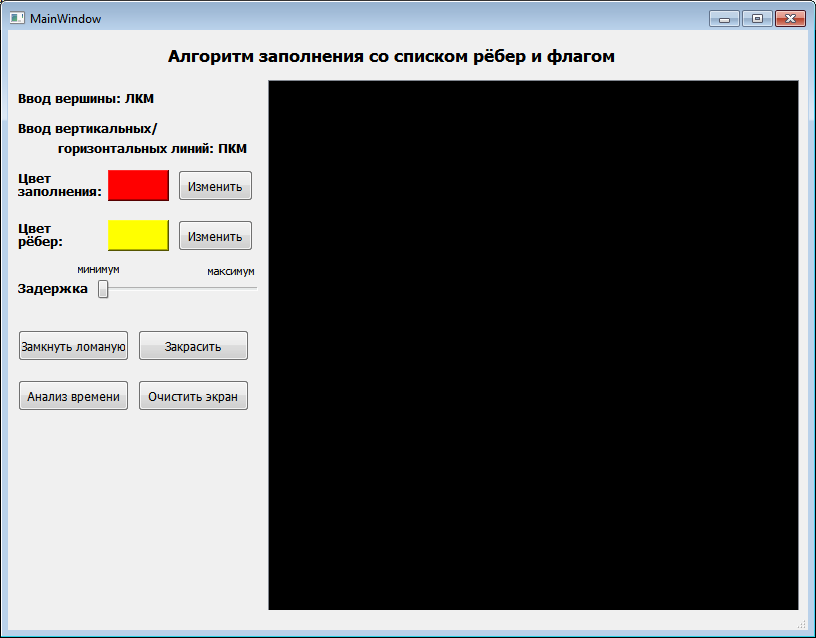
**Интерфейс**

Пользователю предоставляется следующий интерфейс для работы:

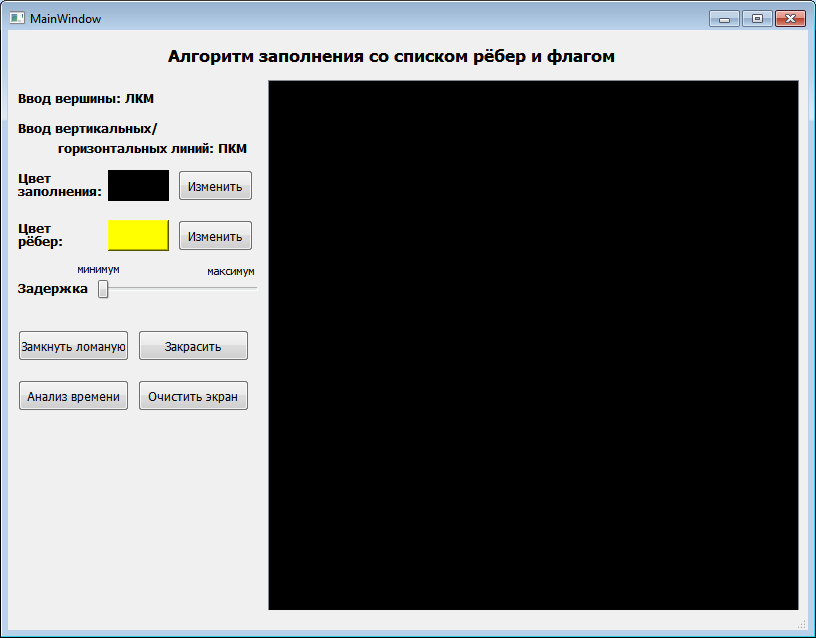


Ввод вершин осуществляется с помощью мыши, путём нажатия на черный холст. Если нажимается левая кнопка мыши, то рисуется соответствующее ребро, если пользователю нужно изобразить вертикальный или горизонтальный отрезок, то он должен сместить курсор на необходимое расстояние и нажать правую кнопку мыши, программа сама скорректирует координаты и выполнит задачу.

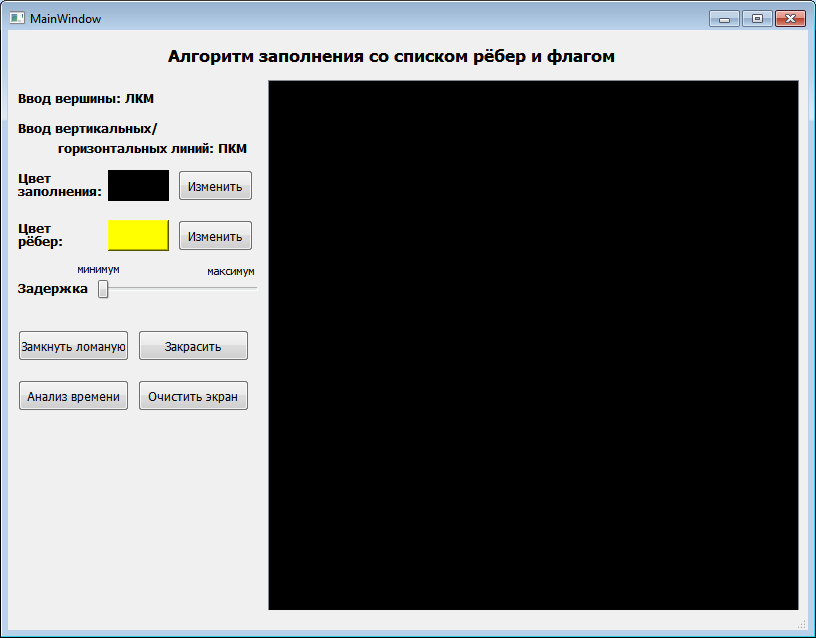
Пользователь может выбрать цвет заполнения и цвет вводимых рёбер:



Программа предоставляет возможность работы алгоритма с задержкой, которую можно регулировать с помощью соответствующего горизонтального ползунка. Если пользователю необходимо использовать режим без задержки нужно выставить значение ползунка на «минимум» (кроме того, изначально при запуске выбран этот параметр).



Пользователь может задавать программе следующие команды:



Команда «Замкнуть ломаную» автоматически соединяет начальную точку с конечной. Пользователь может сделать это и самостоятельно, достаточно относительно близко к начальной вершине нажать мышкой (предусмотрена погрешность, попав в которую координаты программно корректируются) и рисуется соответствующее ребро.

Команда «Закрасить» запускает процесс заполнения области. После нажатия на эту кнопку пользователю предлагается ввести затравочный пиксель.

Ещё предусмотрены такие команды, как «Анализ времени» и «Очистить экран».

**Алгоритм построчного затравочного заполнения сплошных областей**

*Об алгоритме*

В этом алгоритме используется такая структура данных, как стек. Стек используется для хранения затравочных пикселей. Сам алгоритм основан на построчном заполнении области, и что важно, в стеке хранится только один затравочный пиксель из очередного непрерывного интервала пикселей.

Алгоритм предусматривает следующие этапы:

1. Извлечь очередной затравочный пиксель из стека
2. Заполнить интервал, которому принадлежит этот затравочный пиксель влево и вправо, пока не найдутся соответствующие границы (запомнить их: )
3. Просмотреть соседние строки (сверху и снизу) в граничном интервале (от до ). В случае, когда найдены пиксели, которые не являются ни граничными, ни заполненными, нужно добавить в стек крайний правый пиксель (очередной затравочный пиксель)

Дальше все действия повторяются, но уже для следующего элемента из стека. Маркером конца работы алгоритма является пустой стек.

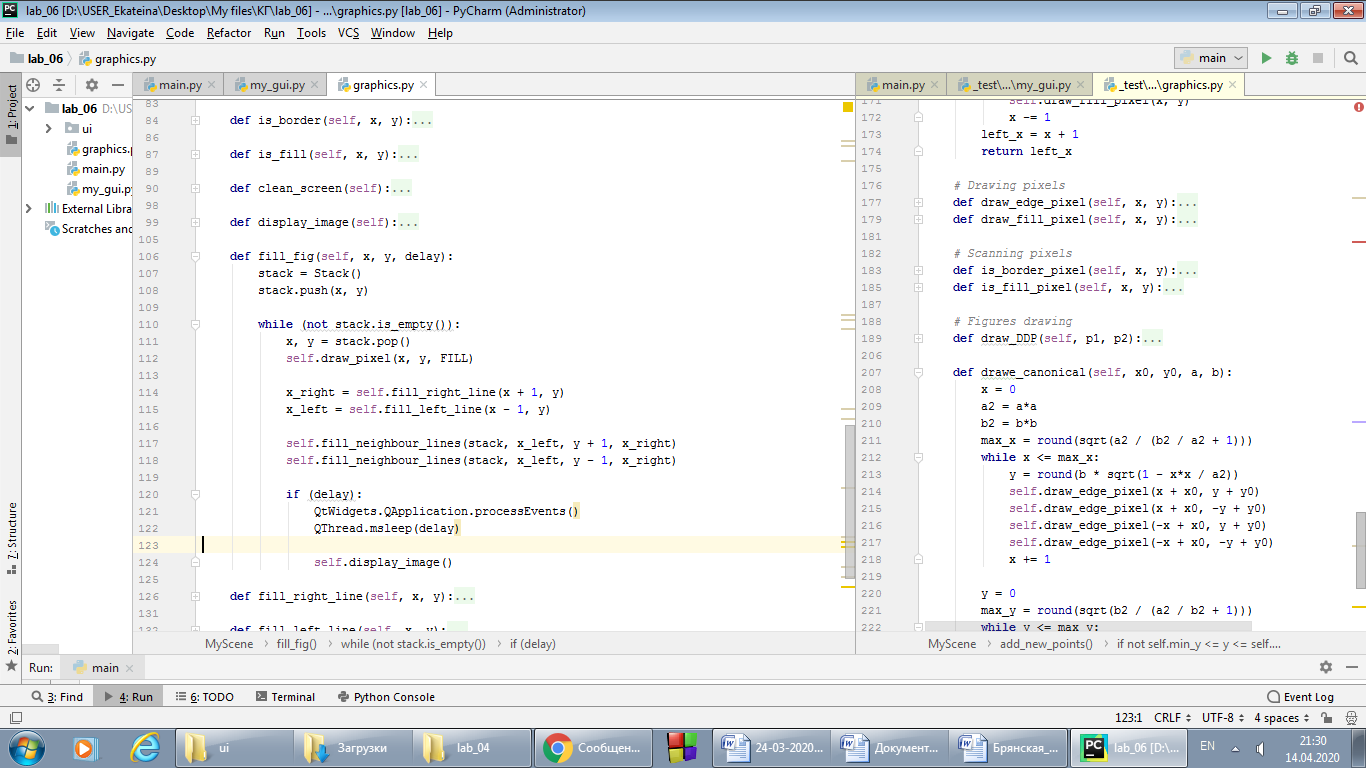
Изначально в стек помещается затравочная точка, которую введёт пользователь.

*Преимущества алгоритма*

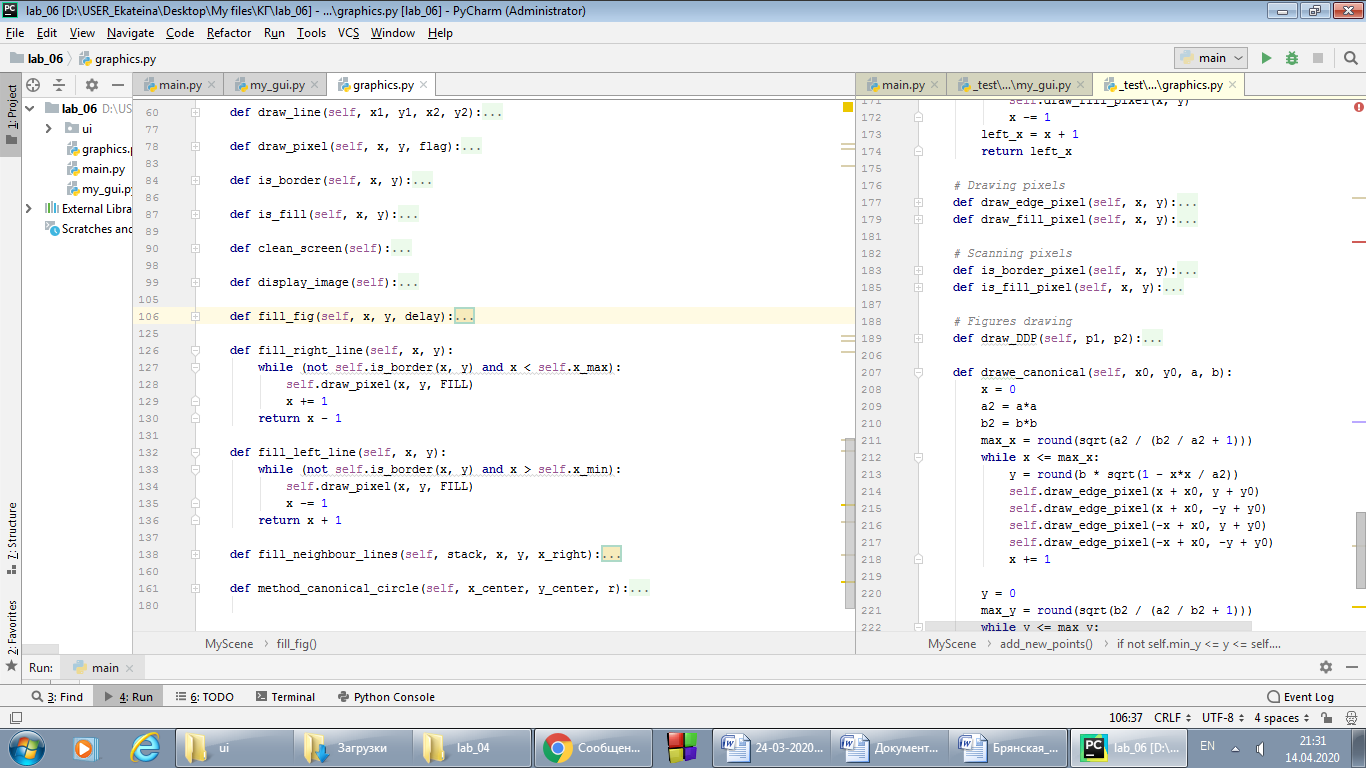
Для данного алгоритма не требуется стек большого размера (так как хранится только один пиксель на каждый непрерывный интервал), что влияет и на время работы, и на объём требуемой памяти.

*Реализация:*

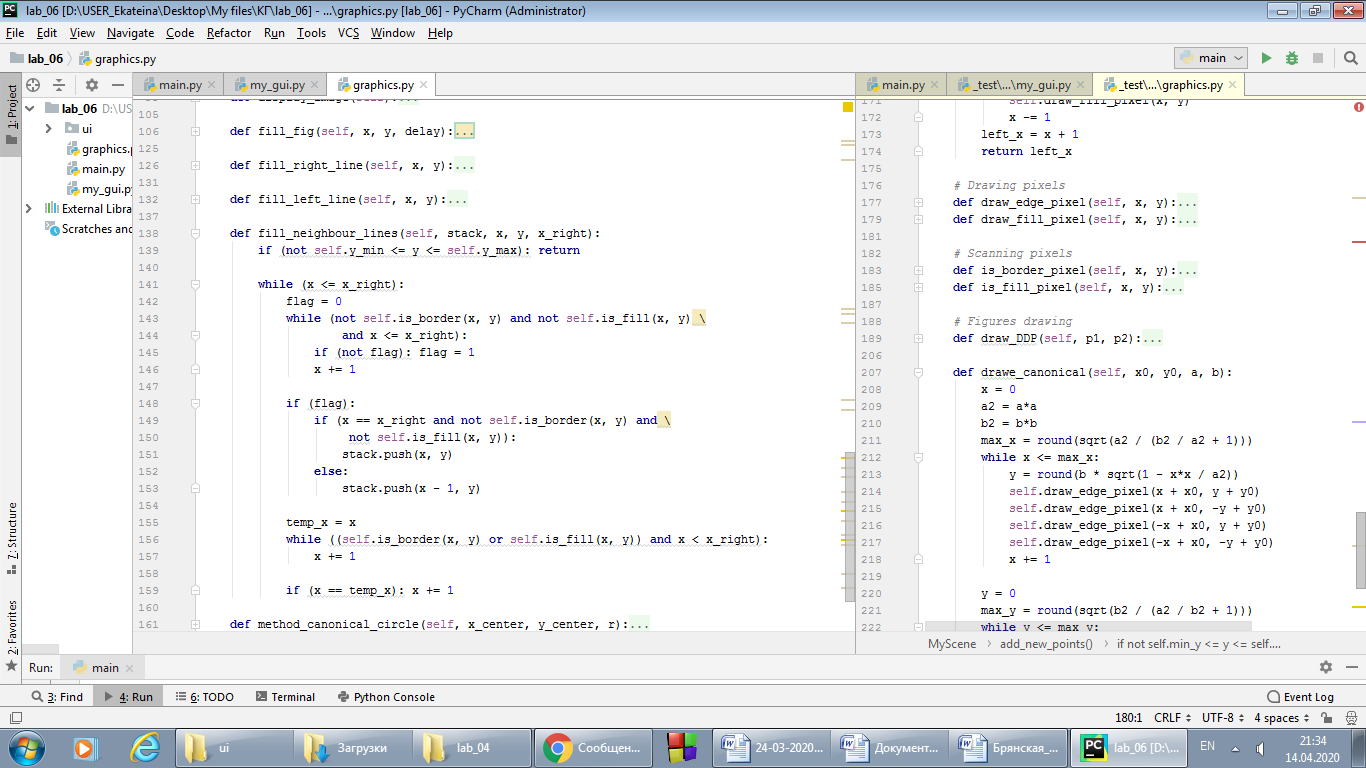
Основной цикл:



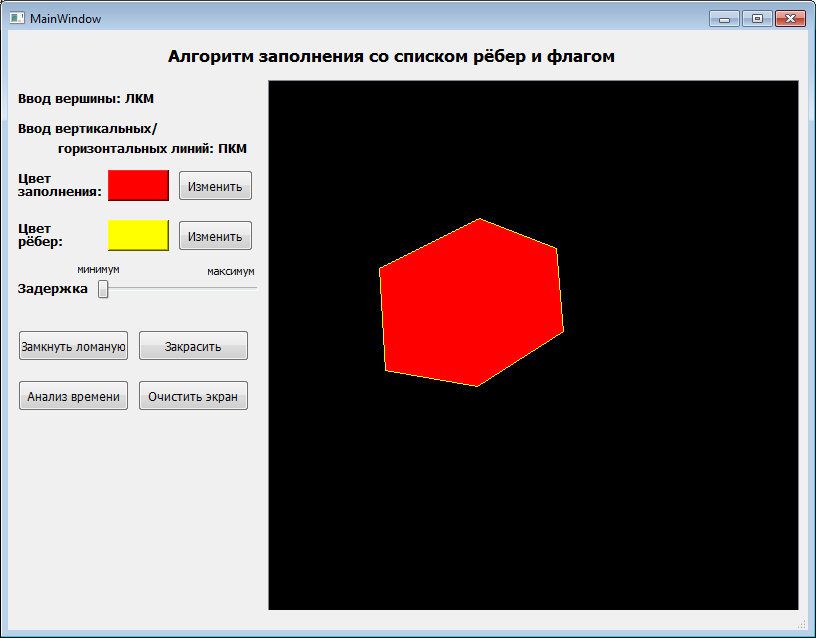
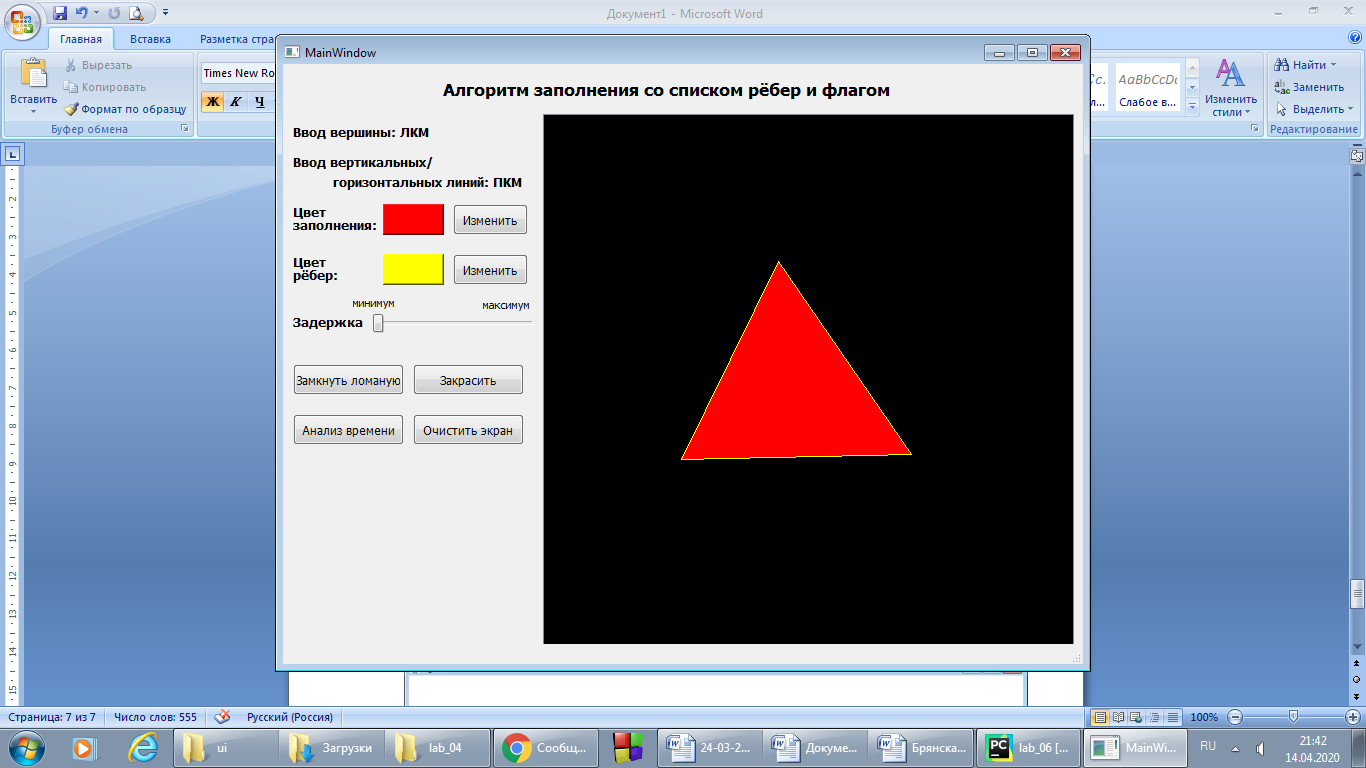
Заполнение строк справа и слева от затравочного пикселя:



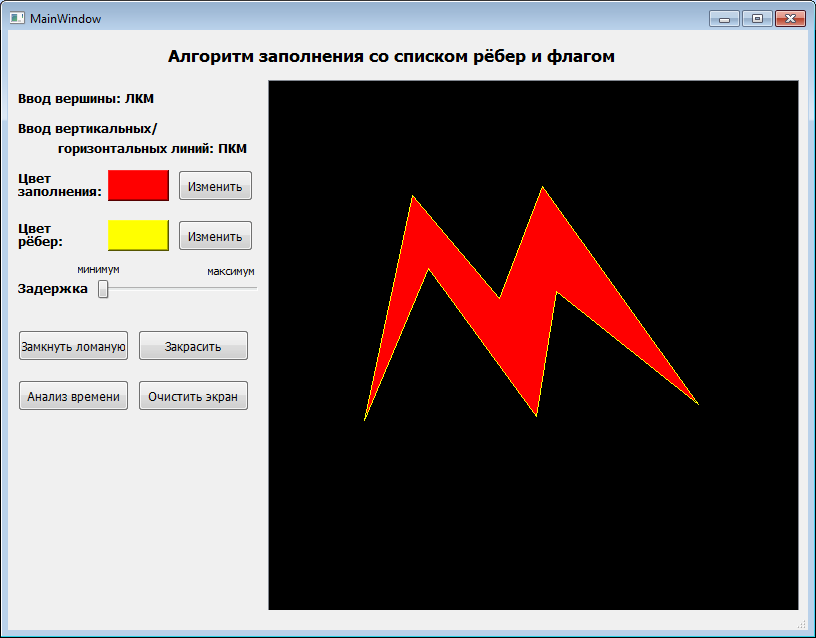
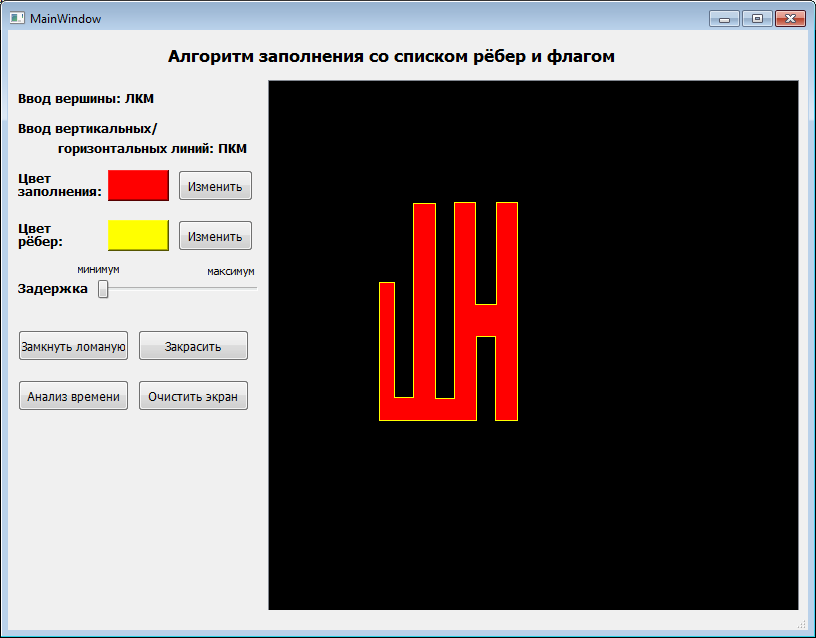
Просмотр соседних строк:

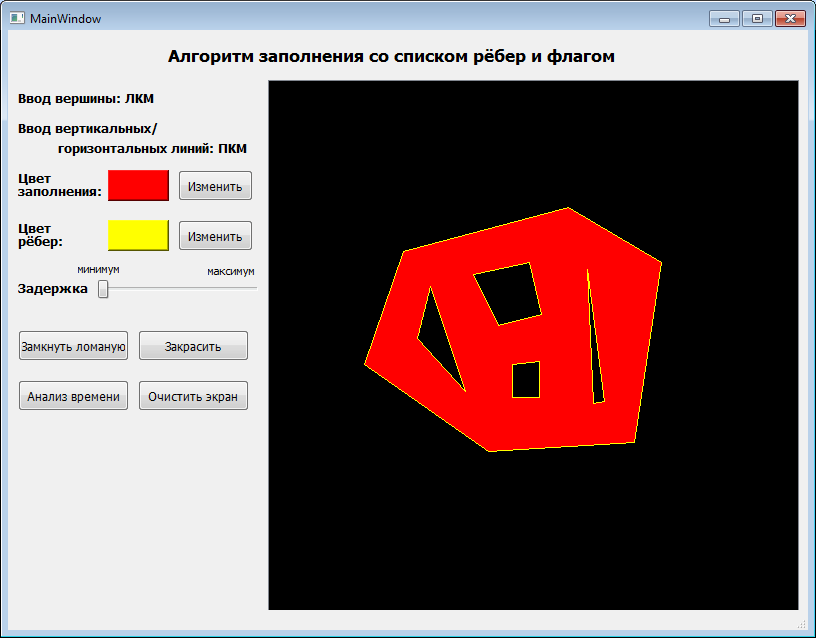
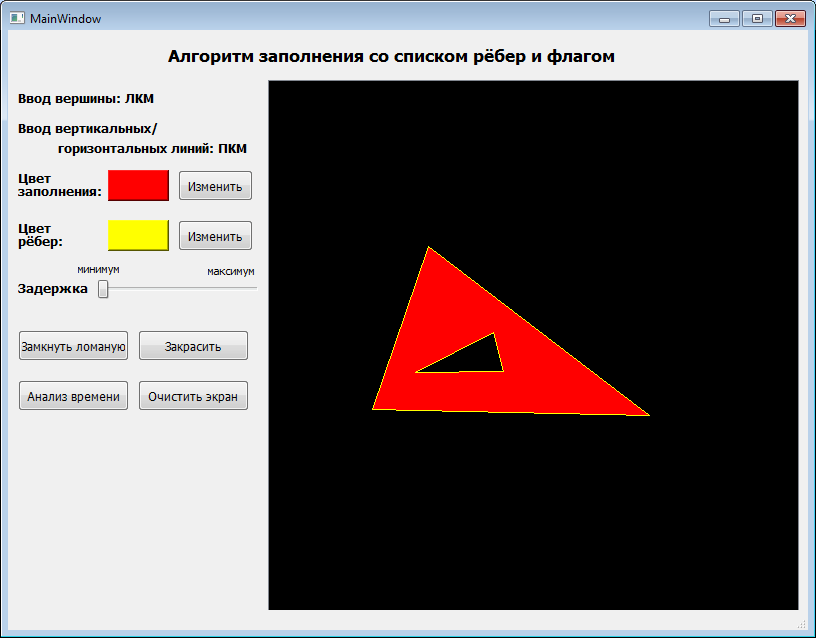


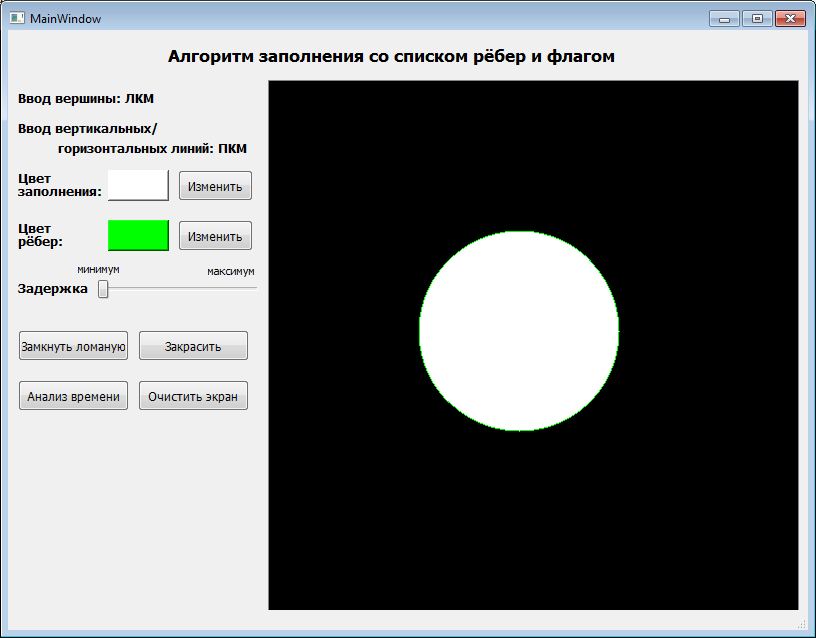
*Примеры работы:*

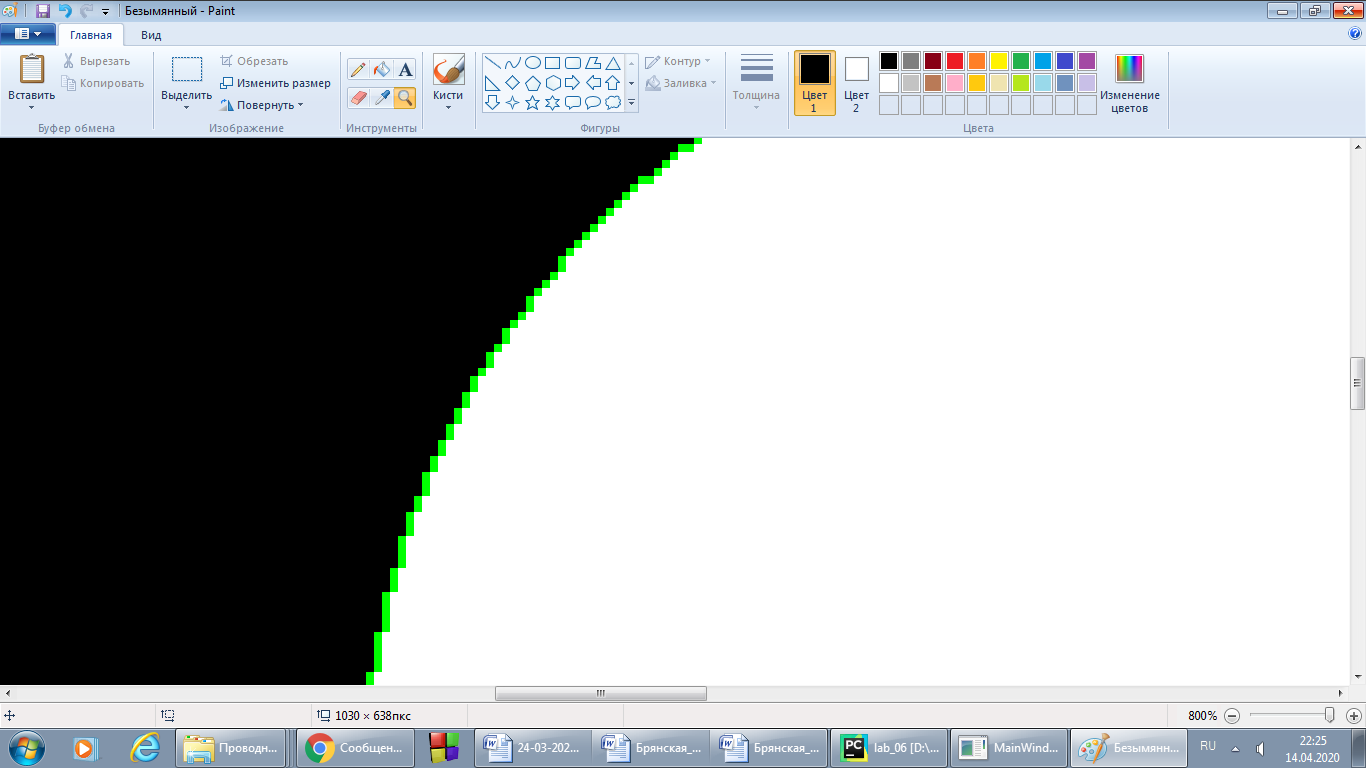
1. Выпуклые фигуры
2. Невыпуклые фигуры 3) Содержит вертикальные/

горизонтальные ребра

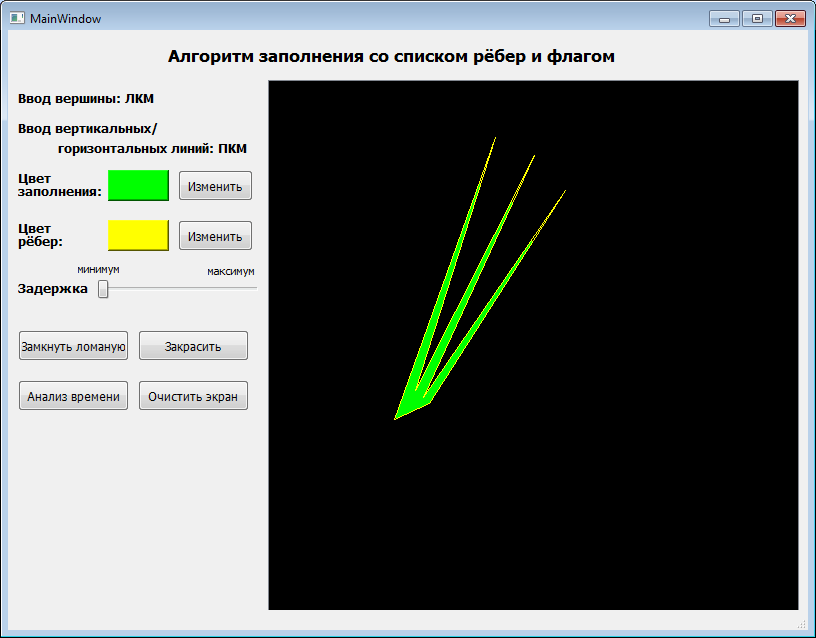
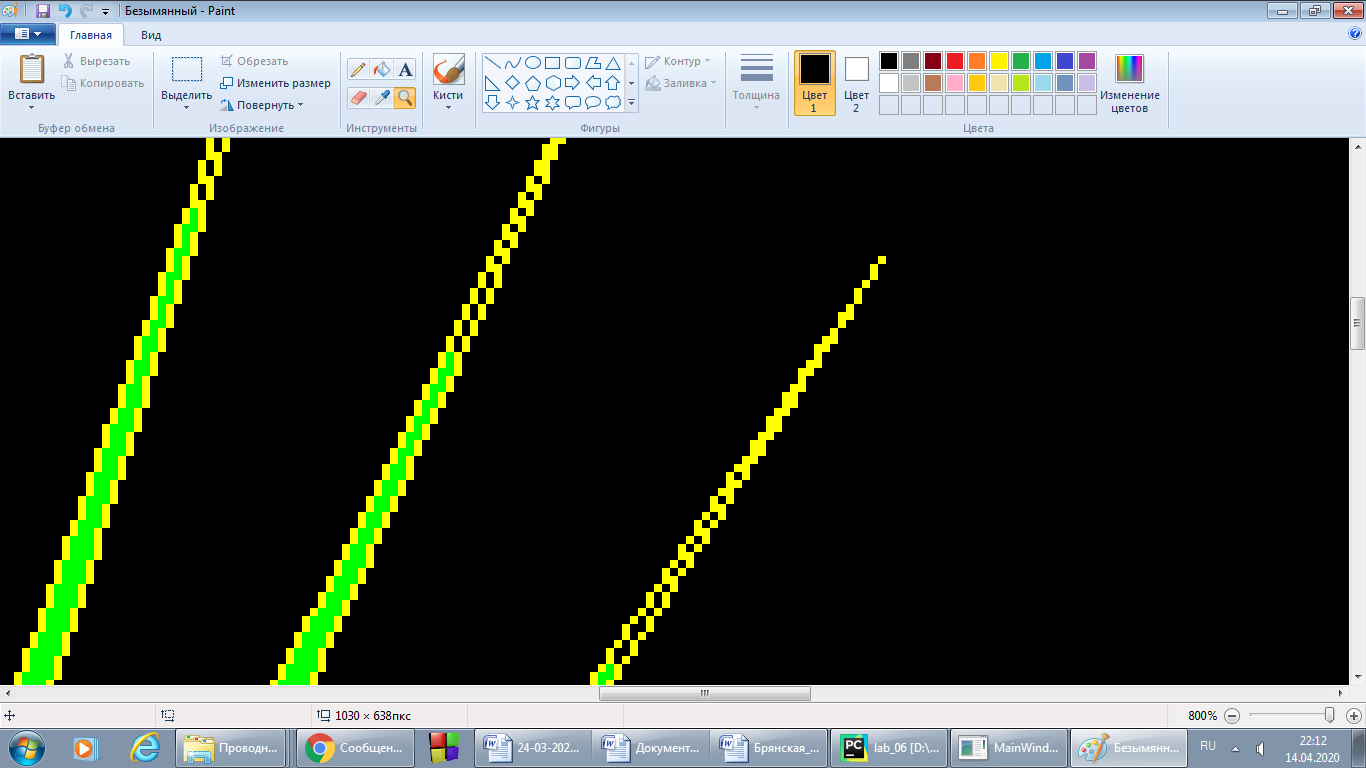


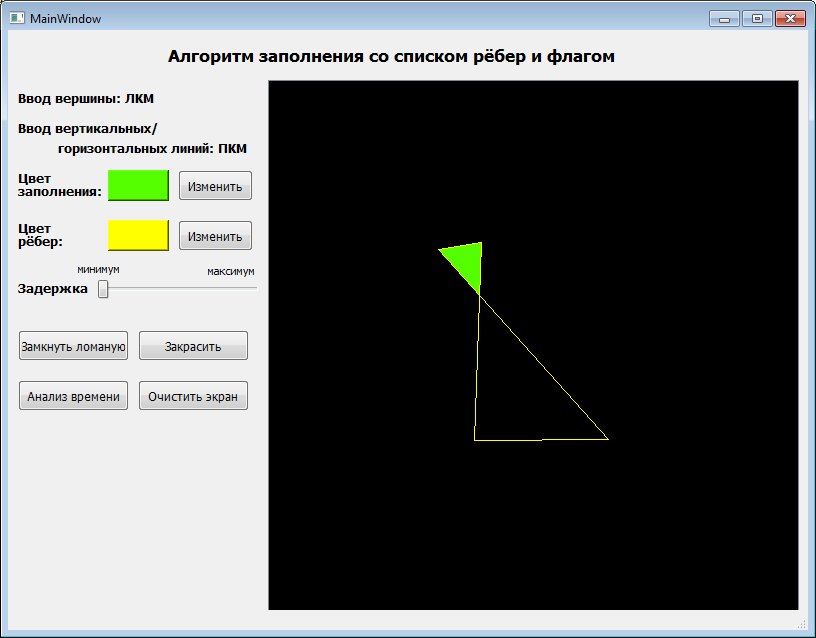
1. С отверстиями
2. Области, ограниченной замкнутой кривой линией

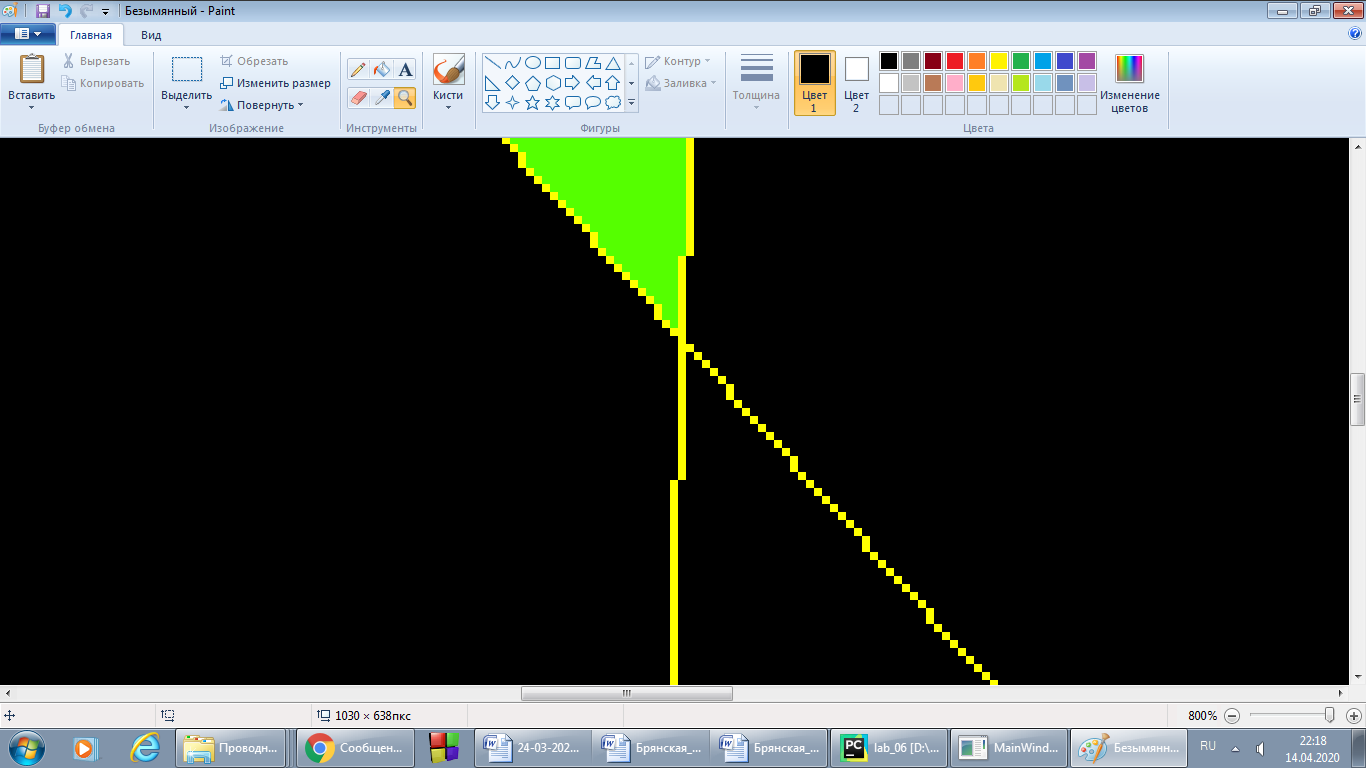




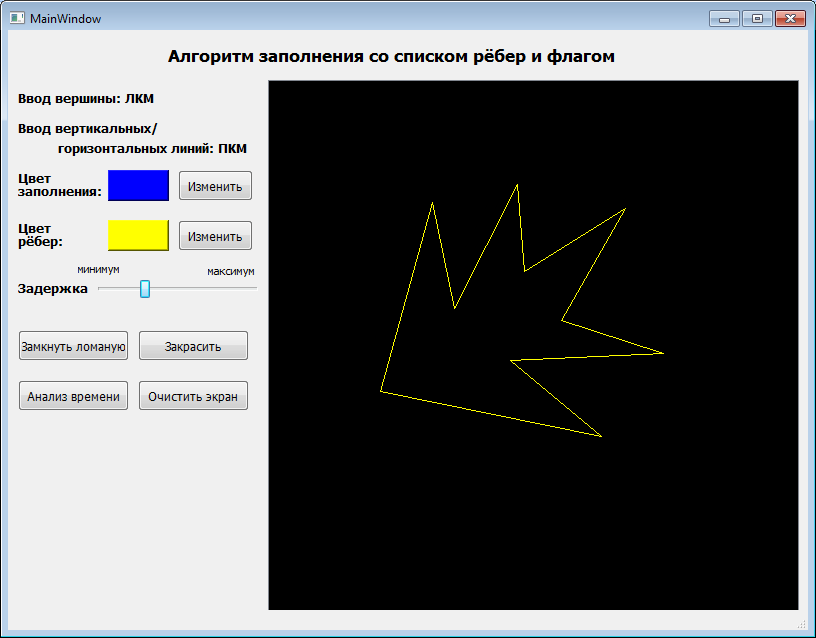
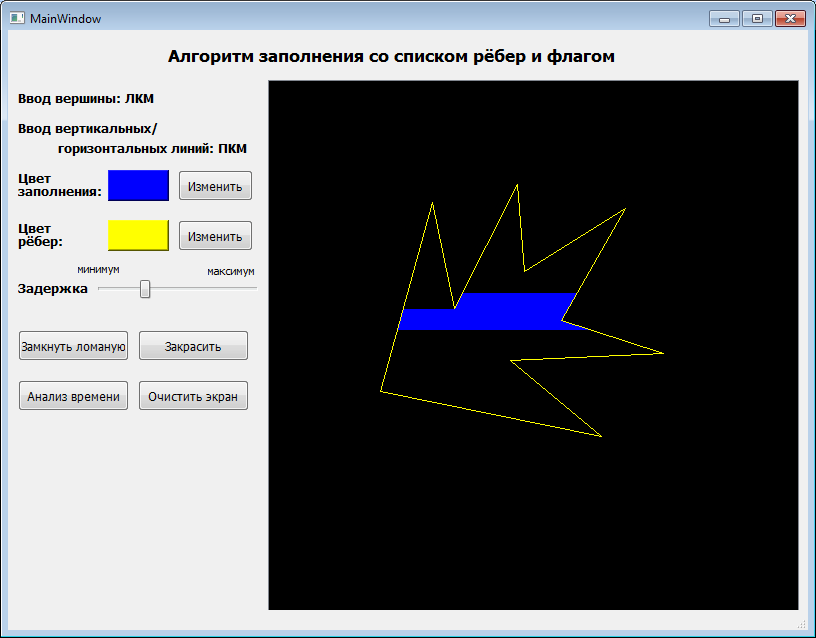
*Важный момент*

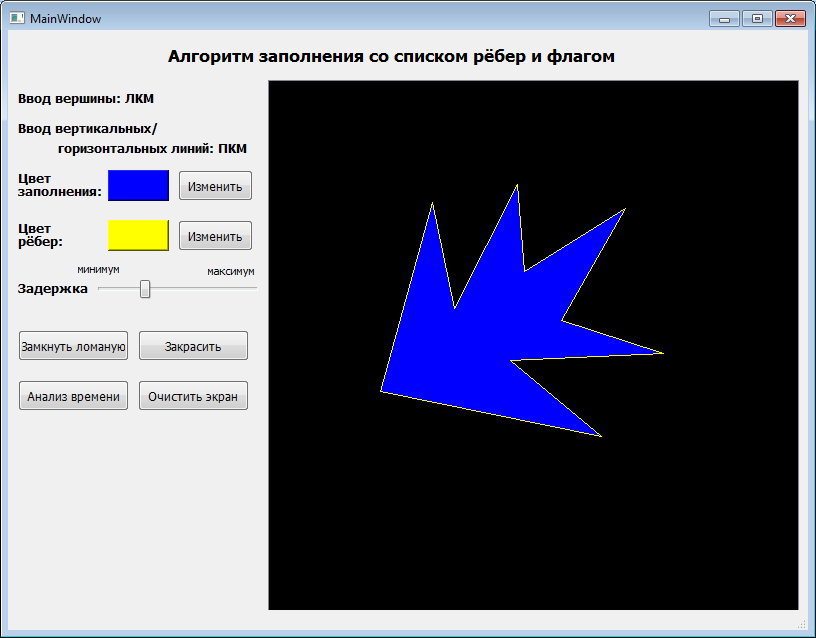
Важно помнить, что данный алгоритм обрабатывает 4х связные области, и поэтому обработать подобную ситуацию, он не может, поэтому остаются необработанные пиксели:

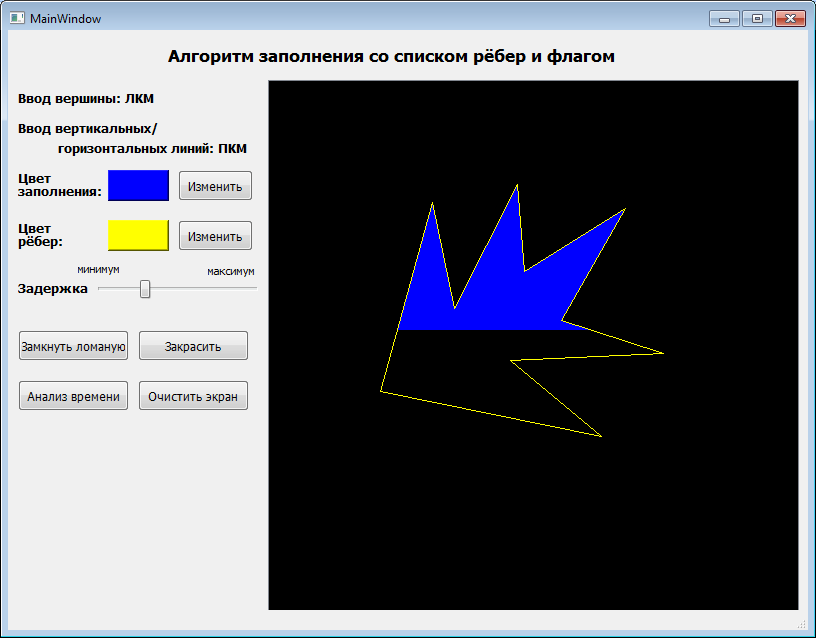
Похожая ситуация возникает, когда фигура содержит пересекающиеся ребра:



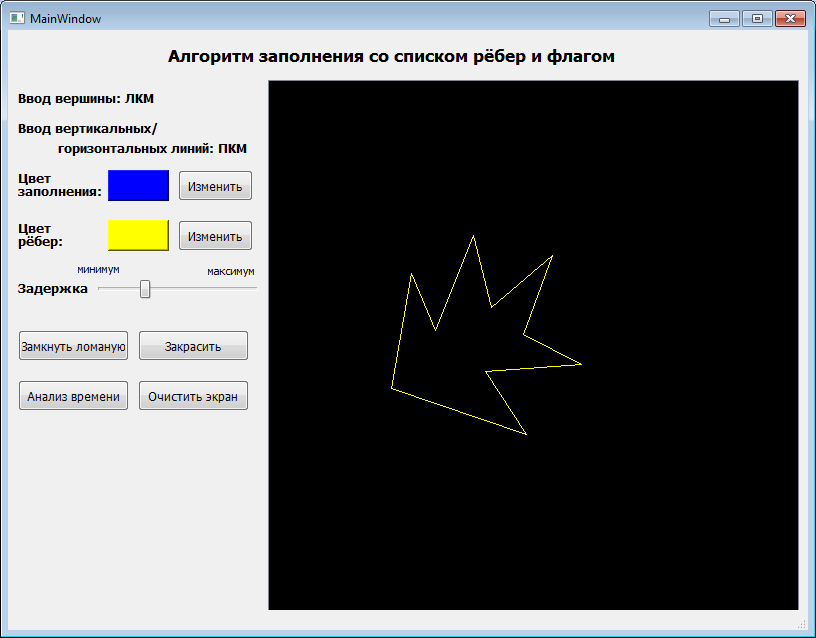
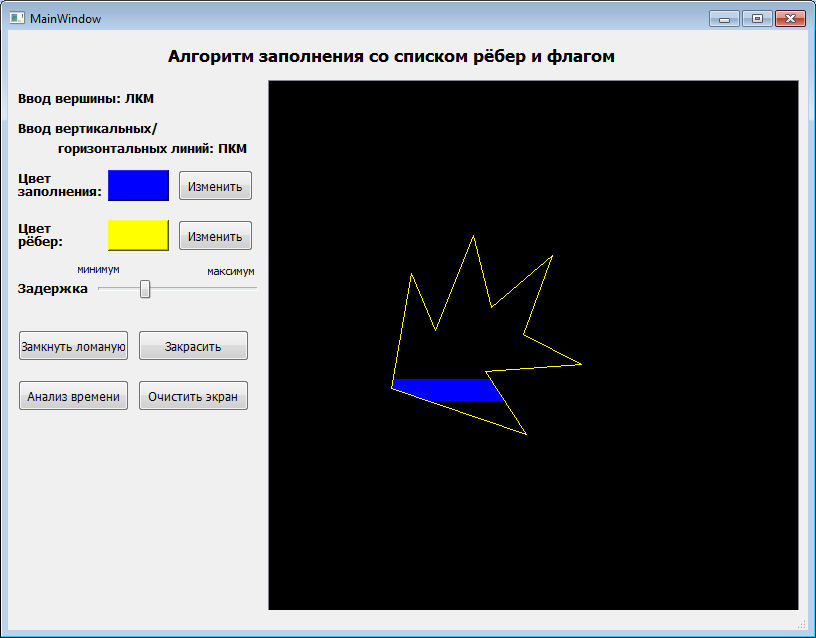
*Работа с задержкой:*

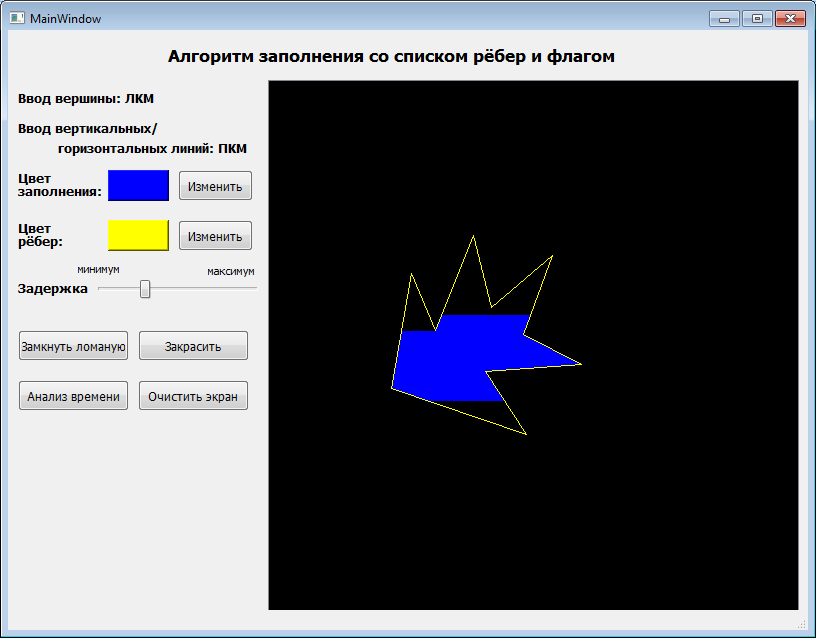
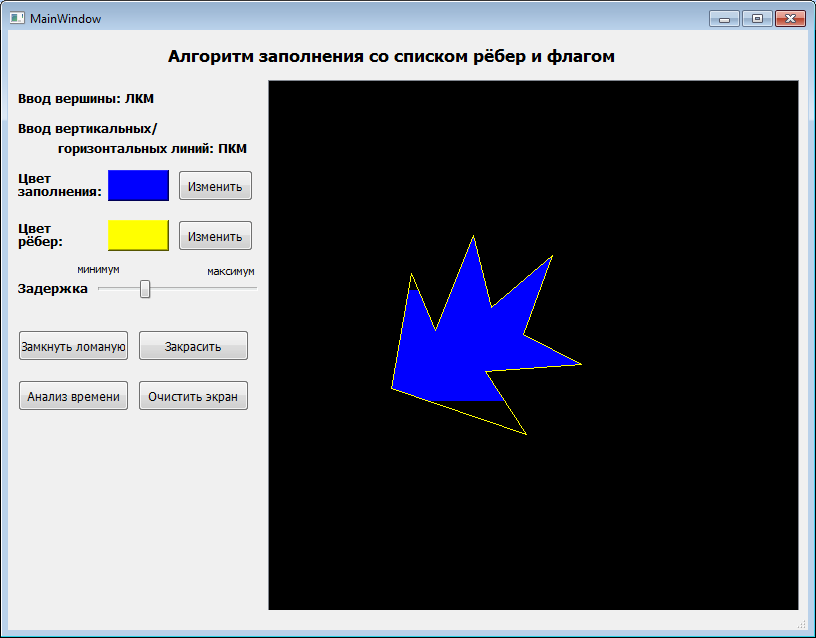
Затравочный пиксель был выбран в середине фигуры:

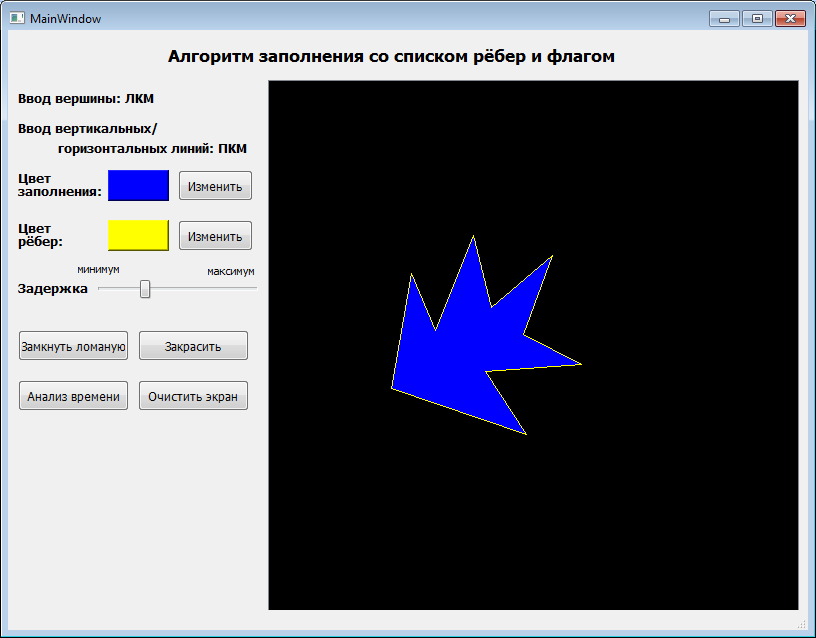




Затравочный пиксель был выбран в нижнем лепестке:

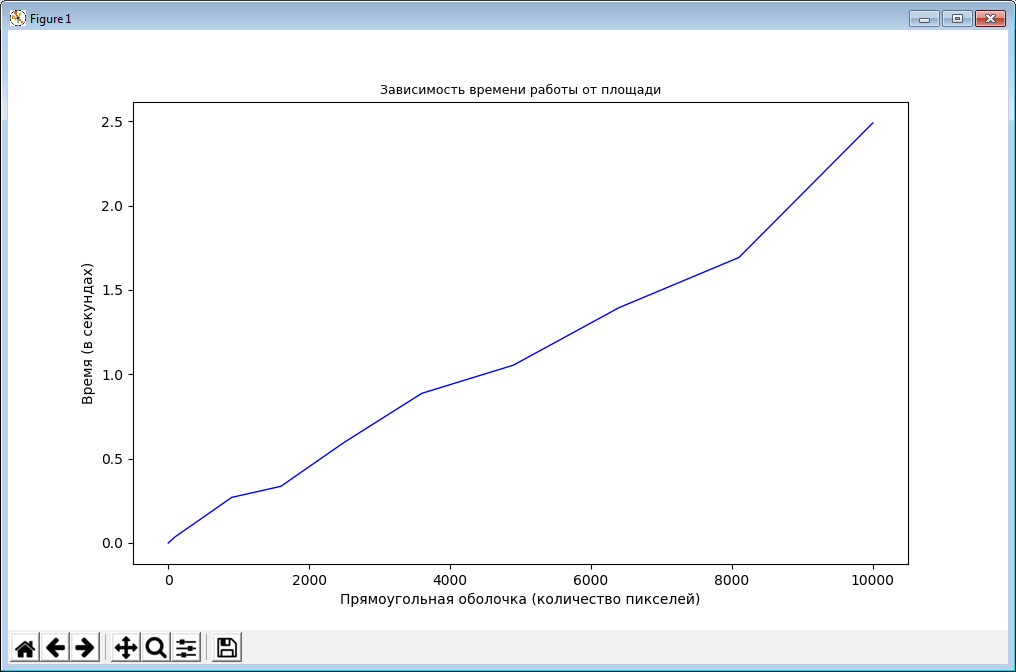






*Анализ времени:*

Производится анализ зависимости времени работы алгоритма от площади заданной фигуры. В качестве примера был взят квадрат, затравочный пиксель находится в середине. По оси ОХ указано значение площади.



На основе этого графика можно сделать вывод о том, что время работы линейно зависит от площади фигуры.

К основным факторам, влияющим на время работы алгоритма, относится количество обрабатываемых пикселей, количество раз считывания, обработки и изменения одного и того же пикселя.

Также этот алгоритм выигрывает по времени, по сравнению с простым алгоритмом, благодаря небольшому по размеру стеку затравочных пикселей.

**Вывод**

В ходе лабораторной работы был изучен и реализован алгоритм построчного затравочного заполнения сплошных областей, а также проведен его анализ. Рассмотрены различные случаи многоугольников для наблюдения за работой алгоритма на различных по форме областях и выявлен недостаток заполнения (при малых углах между ребрами многоугольника, при пересечении рёбер).