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

Дисциплина Компьютерная графика

Тема Реализация и исследование алгоритма построчного затравочного заполнения сплошных областей

Студент Хетагуров П.К.

Группа ИУ7-45

Оценка (баллы) _____

Преподаватель Куров А. В

Москва.
2020 г.

Цель работы:

Реализация и исследование алгоритма построчного затравочного заполнения.

Задание:

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. Должен быть предусмотрен ввод затравочной точки.
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:

Суть алгоритма:

В алгоритмах затравочного заполнения сплошных областей предполагается, что известна определенная точка (затравка) внутри области и необходимо определить точки, соседние с затравочной и расположенные внутри области.

Данные алгоритмы применимы к гранично-определенным областям, то есть таким, что все пикселы на границе данных областей имеют выделенное значение или цвет , но не один пиксел из внутренней части таких областей не может иметь это выделенное значение.

Преимущества построчного алгоритма заполнения с затравкой:

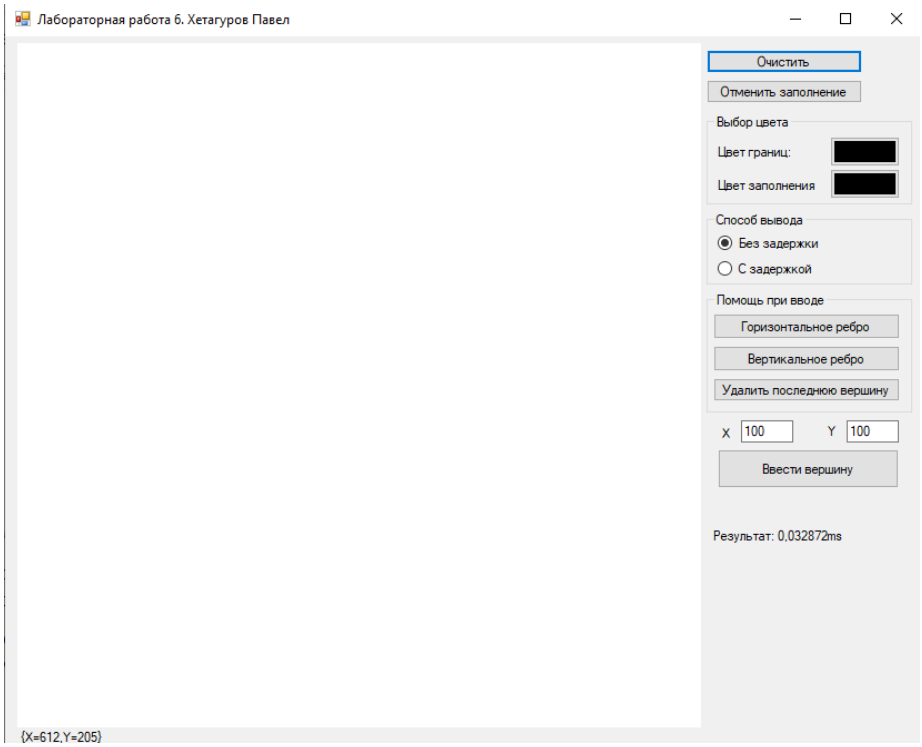
1) Возможность заполнения области, без знания точных её границ (не требует хранения списка ребер)
2) Алгоритм не обрабатывает пикселы вне заполняемой области и в отверстиях
3) В отличии от простого алгоритма с затравкой, меньший требуемый объем памяти

Недостатки:

1) Так как происходят обращения к цвету пикселя (считывание и изменение), скорость алгоритма зависит от скорости считывания и изменения цвета пиксела.
2) Необходимость знания цвета границ области

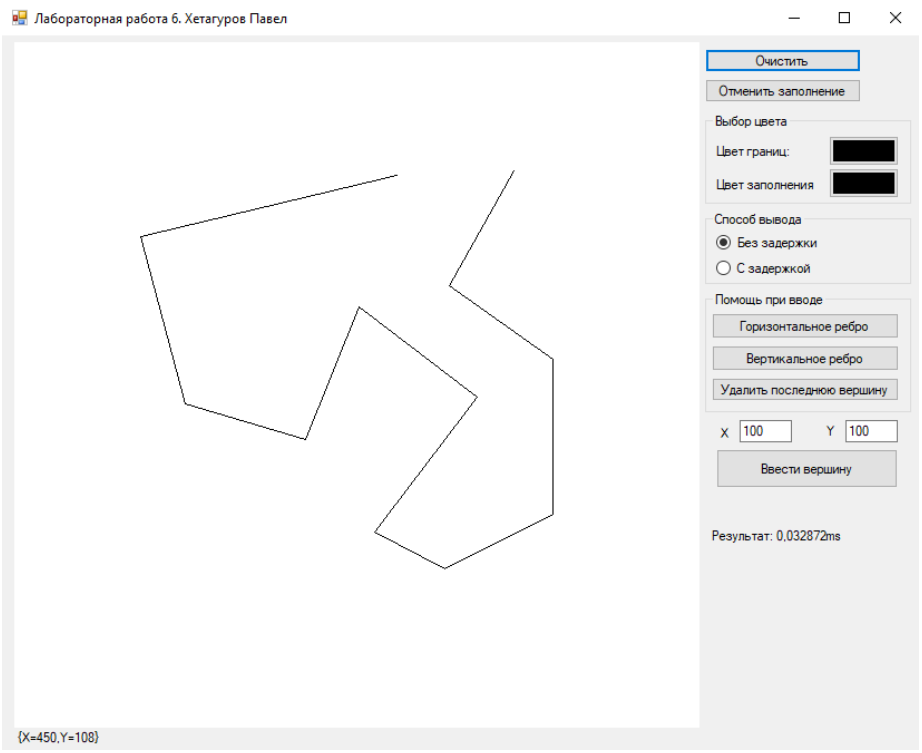
Практическая часть

Интерфейс:



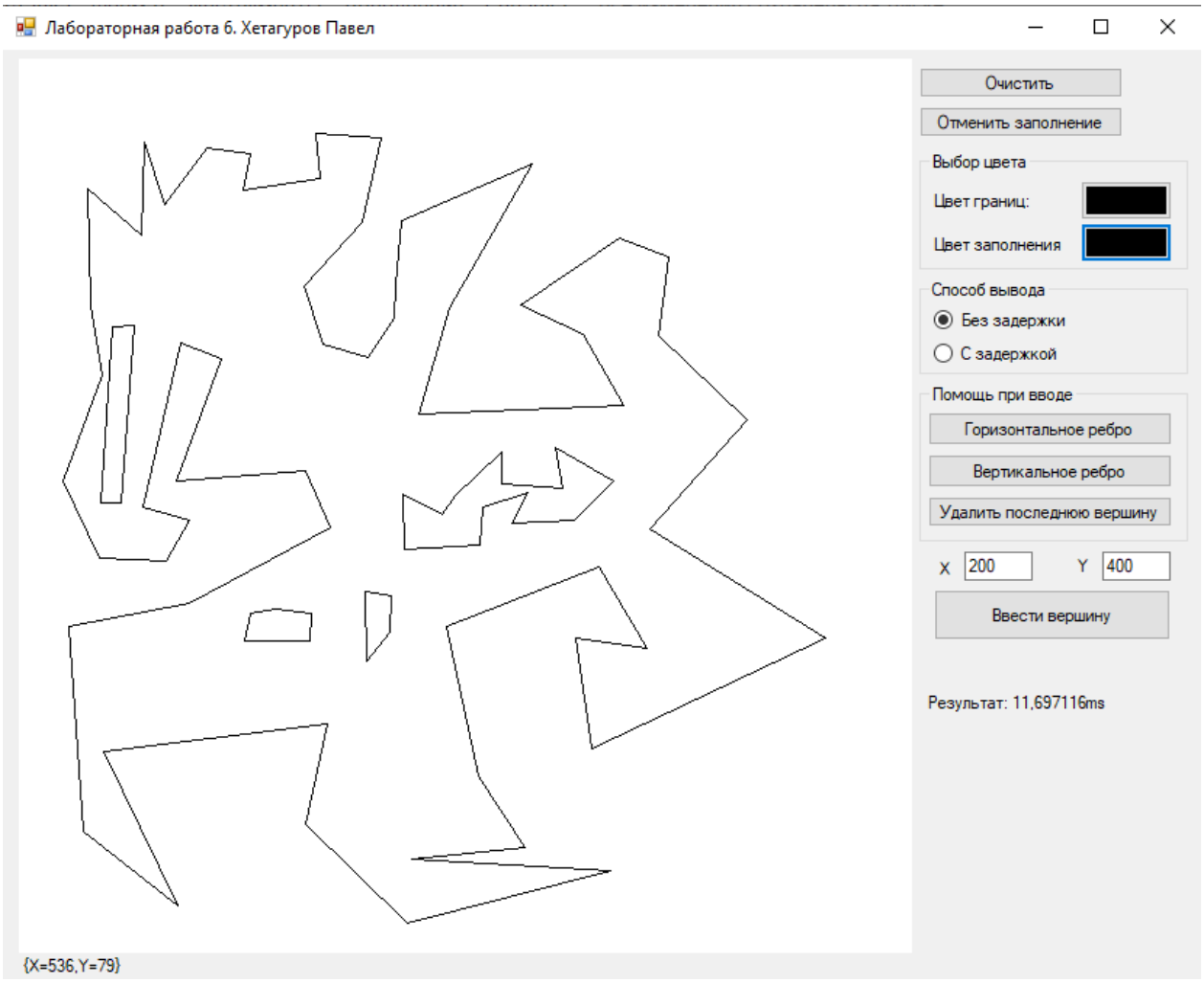
Вводить вершины можно как с помощью мыши (по нажатию на область рисования), так и с помощью текстовых полей (по координатам)

Затравочный пиксель вводится нажатием правой кнопки мыши и сразу запускает алгоритм заполнения

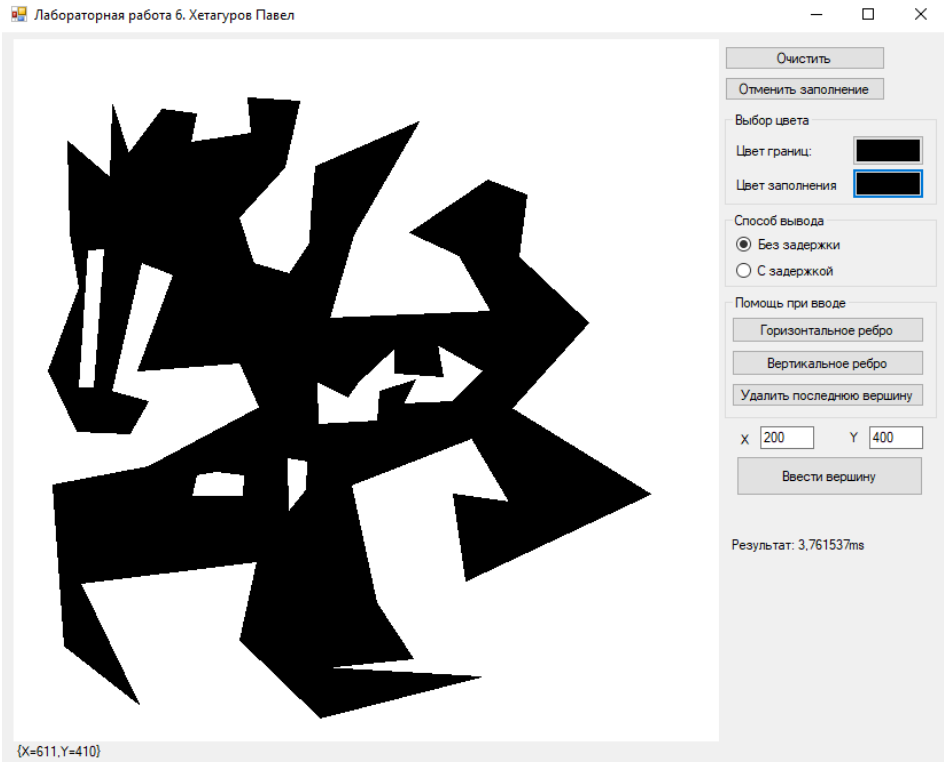


При вводе вершин они соединяются с помощью ребер.

Заполнение произвольной сложной фигуры:

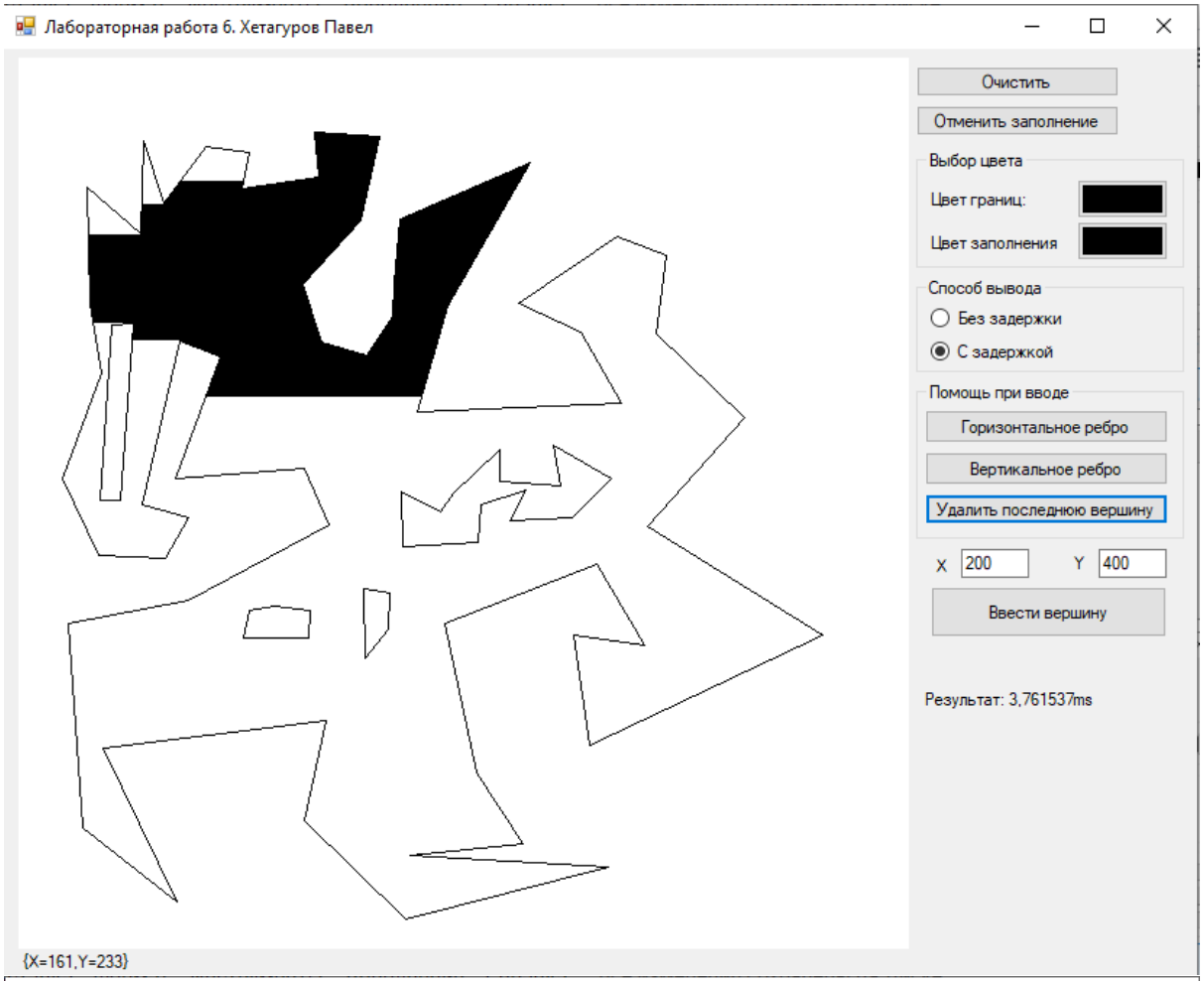


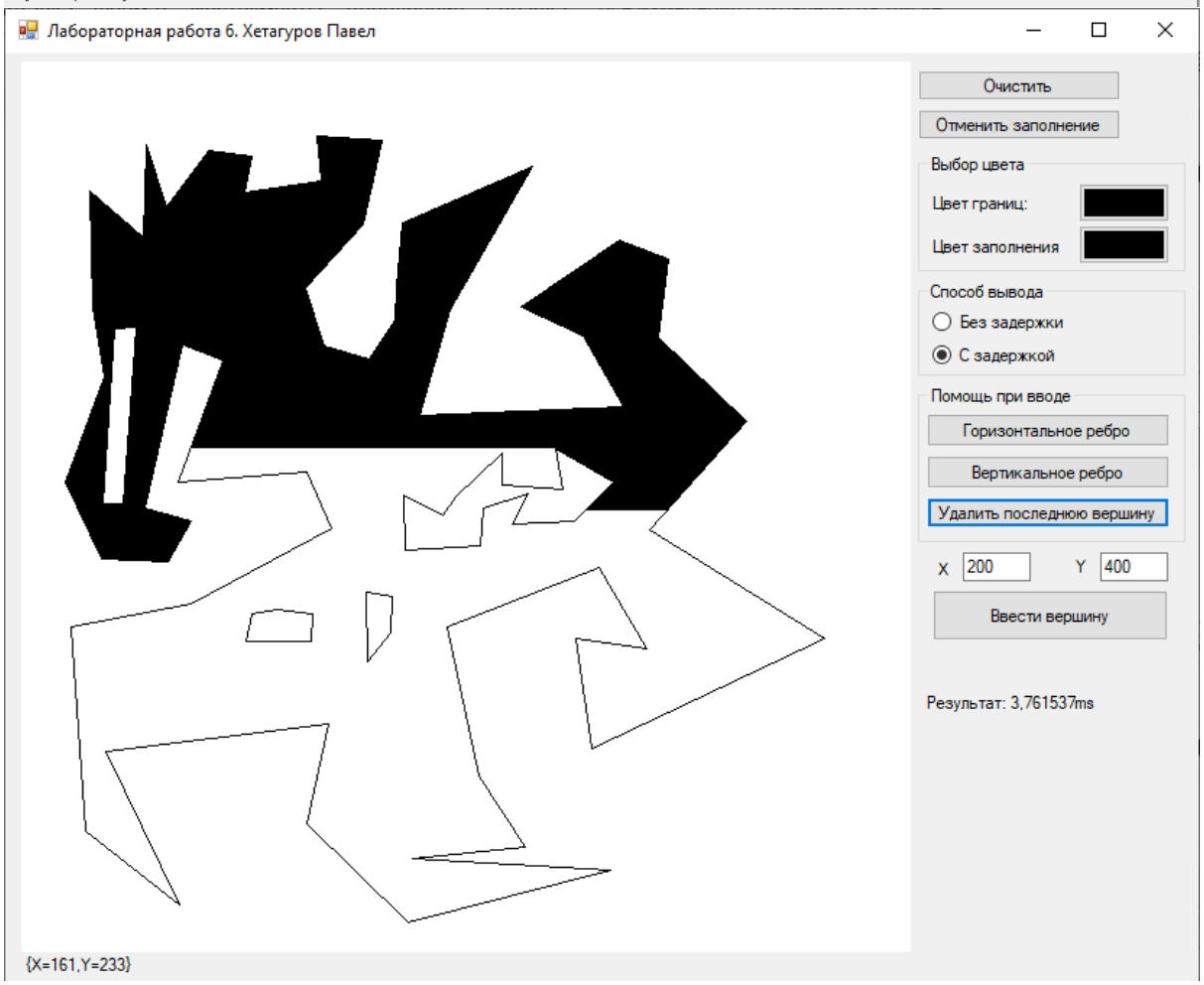
(На скриншоте появляются артефакты на границах фигуры, на экране их нет. Скорее всего из-за сжатия)

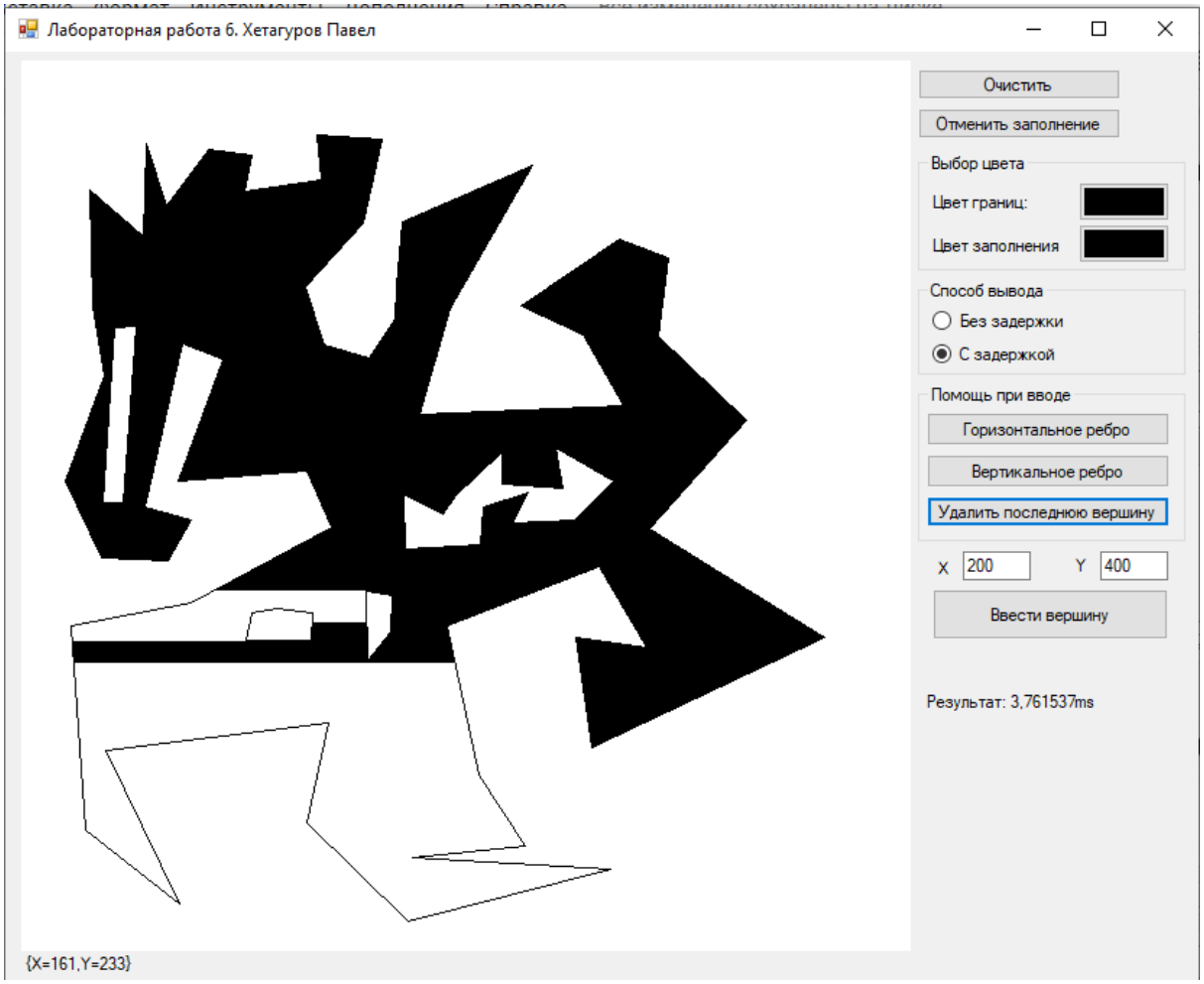


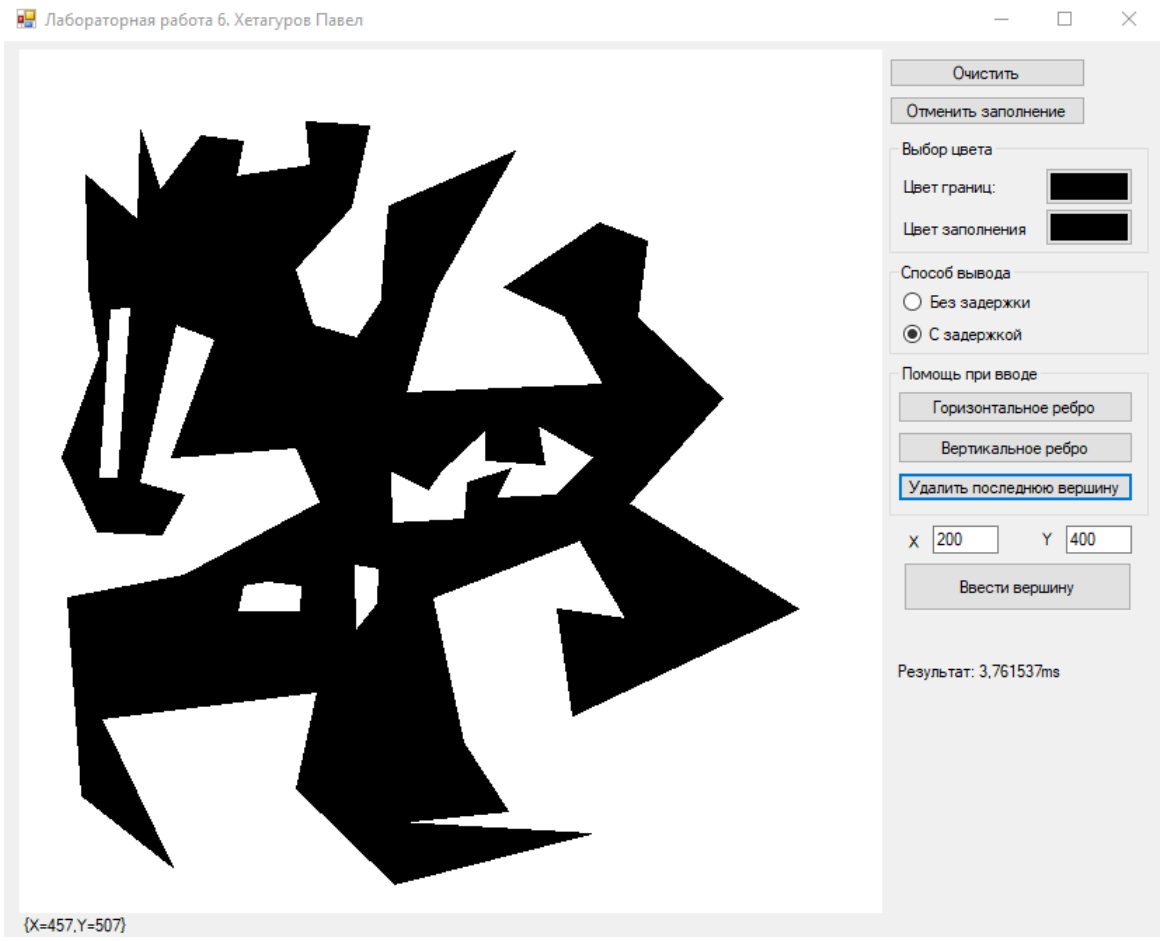
Время полного заполнения: 3.76 миллисекунд (быстрее, чем заполнение аналогичной фигуры при использовании растрового заполнения)

При заполнении с задержкой можно проследить промежуточные результаты:


Лабораторная работа 6. Хетагуров Павел
Очистить
Отменить заполнение
Выбор цвета
Цвет границ:
Цвет заполнения
Способ вывода
Без задержки
С задержкой
Помощь при вводе
Горизонтальное ребро
Вертикальное ребро
Удалить последнюю вершину
X 200
Y 400
Ввести вершину
Результат: 3,761537ms
{X=161,Y=233}


Лабораторная работа 6. Хетагуров Павел
Очистить
Отменить заполнение
Выбор цвета
Цвет границ:
Цвет заполнения
Способ вывода
Без задержки
С задержкой
Помощь при вводе
Горизонтальное ребро
Вертикальное ребро
Удалить последнюю вершину
X 200
Y 400
Ввести вершину
Результат: 3,761537ms
{X=161,Y=233}


Лабораторная работа 6. Хетагуров Павел
Очистить
Отменить заполнение
Выбор цвета
Цвет границ:
Цвет заполнения
Способ вывода
Без задержки
С задержкой
Помощь при вводе
Горизонтальное ребро
Вертикальное ребро
Удалить последнюю вершину
X 200
Y 400
Ввести вершину
Результат: 3,761537ms
{X=161,Y=233}


Лабораторная работа 6. Хетагуров Павел
Очистить
Отменить заполнение
Выбор цвета
Цвет границ:
Цвет заполнения
Способ вывода
Без задержки
С задержкой
Помощь при вводе
Горизонтальное ребро
Вертикальное ребро
Удалить последнюю вершину
X 200
Y 400
Ввести вершину
Результат: 3,761537ms
{X=457,Y=507}

Код:

```
    internal static void FillPolygon(Point seed, Color workColor, Color
borderColor, Bitmap workBitmap)
     {
        Stack<Point> pointStack = new Stack<Point>();
        pointStack.Push(seed);
        Point currentPoint;
        Color curColor;
        int X, Y;
        int xRight, xLeft;
        while (!pointStack.IsEmpty)
        {
           currentPoint = pointStack.Pop();
           X = currentPoint.X;
           Y = currentPoint.Y;

           curColor = workBitmap.GetPixel(X, Y);
           while (!IsColorEquals(curColor, borderColor) && X + 1 <
workBitmap.Width)
           {
              workBitmap.SetPixel(X, Y, workColor);
              X++;
              curColor = workBitmap.GetPixel(X, Y);
           }

           int xRight = X - 1;
           X = currentPoint.X;

           X--;
           curColor = workBitmap.GetPixel(X, Y);
           while (!IsColorEquals(curColor, borderColor) && X > 0)
           {
              workBitmap.SetPixel(X, Y, workColor);
              X--;
              curColor = workBitmap.GetPixel(X, Y);
           }
           X++;
           int xLeft = X;

           if (Y + 1 < workBitmap.Height)
           {
              FindSeed(pointStack, xLeft, xRight, Y + 1, workColor, borderColor,
workBitmap);
```

```
            }
            if (Y > 0)
            {
                FindSeed(pointStack, xLeft, xRight, Y - 1, workColor, borderColor,
workBitmap);
            }
        }
    }

    private static void FindSeed(Stack<Point> pointStack, int X, int xRight, int
Y, Color workColor, Color borderColor, Bitmap workBitmap)
    {
        bool finded = false;
        Color curColor = workBitmap.GetPixel(X, Y);

        while (X <= xRight)
        {
            while (!IsColorEquals(curColor, borderColor) && !
IsColorEquals(curColor, workColor) && X <= xRight)
            {
                finded = true;
                X++;
                curColor = workBitmap.GetPixel(X, Y);
            }

            if (finded)
            {
                pointStack.Push(new Point(X - 1, Y));
            }

            while ((IsColorEquals(curColor, borderColor) ||
IsColorEquals(curColor, workColor)) && X <= xRight)
            {
                X++;
                curColor = workBitmap.GetPixel(X, Y);
            }
        }
    }
```