|  |  |
| --- | --- |
| 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»**  **Дисциплина Компьютерная графика**  **Студент Кузин Антон**  **Группа ИУ7-42Б**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель Куров А.В.** |  |

Москва.

2020 г.**Цель:**

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

**Техническое задание:**

В рамках данной работы необходимо реализовать Алгоритм заполнения со списком ребер и флагом.

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).

**Теоретические сведения:**

Алгоритм со списком рёбер и флагом является двухшаговым алгоритмом. Первый шаг – обрисовка контура, ограничивающего область, что позволяет определить пары ограничивающих пикселей, вторым шагом активизируются пиксели, расположенные между вычисленными на предыдущем шаге ограничивающими пикселями. При заполнении строк используется переменная флаг, которая меняет своё значение, если рассматриваемый пиксель является граничным, и обозначает, находится ли рассматриваемый пиксель внутри закрашиваемой области, либо вне неё.

Преимуществами данного алгоритма является то, что каждый пиксель обрабатывается только один раз, а также он пригоден для аппаратной реализации.

**Практическая часть:**

void MainWindow::dda(const QLine &line, canvas\_t &canvas, int extr)

{

int dX = line.x2() - line.x1();

int dY = line.y2() - line.y1();

int l = qMax(abs(dX), abs(dY));

double dx = (double) dX / l;

double dy = (double) dY / l;

double xf = line.x1();

double yf = line.y1();

int x, y;

int ly = -1;

if (!extr)

l -= 1;

for (int i = 0; i <= l; i++)

{

x = qRound(xf);

y = qRound(yf);

if (y != ly)

{

ly = y;

if (canvas.image->pixel(x, y) == canvas.boundColor->rgb()) // проверка для острых углов

canvas.image->setPixel(x + 1, y, canvas.boundColor->rgb());

else

canvas.image->setPixel(x, y, canvas.boundColor->rgb());

}

xf += dx;

yf += dy;

}

}

double MainWindow::fillFlag(QVector<QLine> edges, canvas\_t &canvas, float delay)

{

LARGE\_INTEGER t1, t2;

LARGE\_INTEGER frequency;

double elapsedTime = 0;

QueryPerformanceFrequency(&frequency);

int xmax = edges[0].x1(), ymax = edges[0].y1();

int xmin = edges[0].x1(), ymin = edges[0].y1();

for (int i = 1; i < edges.length(); i++)

{

if (edges[i].x1() > xmax)

xmax = edges[i].x1();

if (edges[i].y1() > ymax)

ymax = edges[i].y1();

if (edges[i].x1() < xmin)

xmin = edges[i].x1();

if (edges[i].y1() < ymin)

ymin = edges[i].y1();

}

for (int j = 0; j < ends.length(); j++)

for (int i = starts[j]; i < ends[j]; i++)

dda(edges[i], canvas, extr\_check(edges[i], edges[i + 1]));

bool flag;

for (int y = ymax; y >= ymin; y--)

{

QueryPerformanceCounter(&t1);

flag = false;

for (int x = xmin; x <= xmax; x++)

{

if (canvas.image->pixel(x,y) == canvas.boundColor->rgb())

flag = !flag;

if (flag)

canvas.image->setPixel(x, y, canvas.fillColor->rgb());

else

canvas.image->setPixel(x, y, canvas.bgColor->rgb());

}

QueryPerformanceCounter(&t2);

elapsedTime += (t2.QuadPart - t1.QuadPart) \* 1000. / frequency.QuadPart;

if (ui->delayCheckBox->isChecked())

{

QThread::usleep(delay);

imageView();

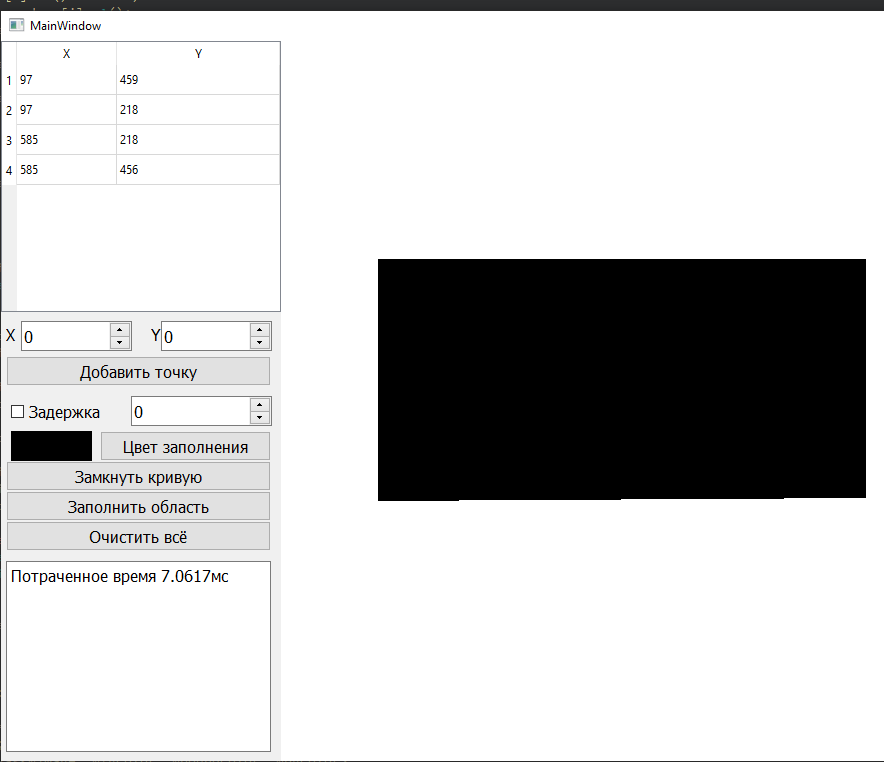
repaint();

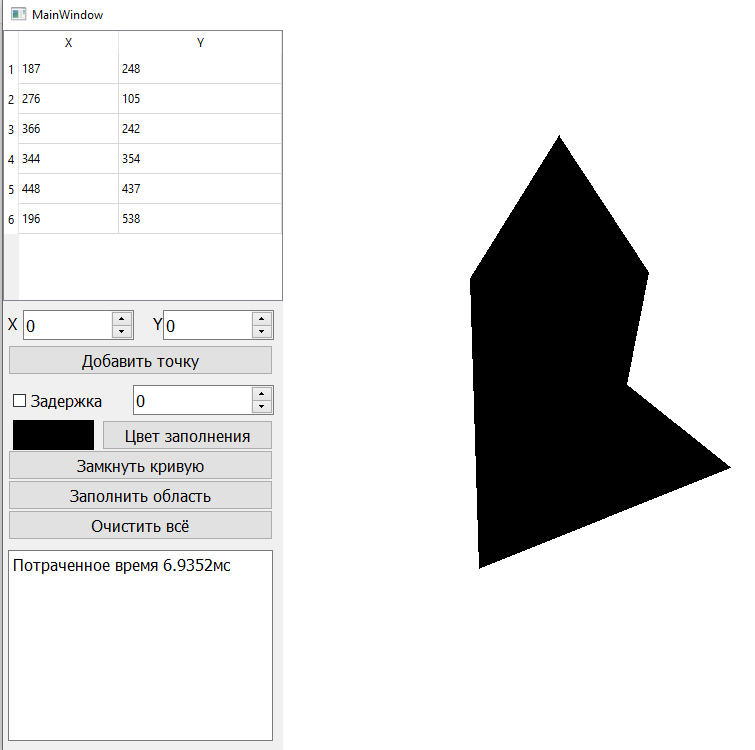
}

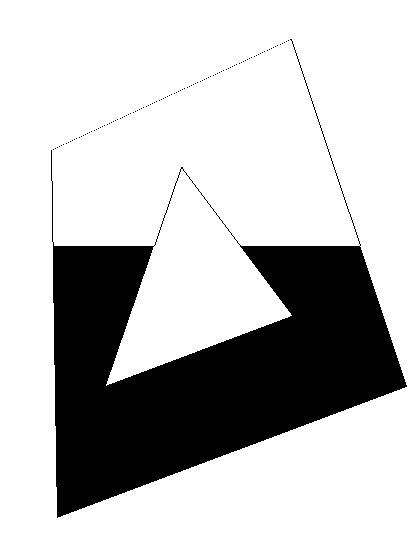
}

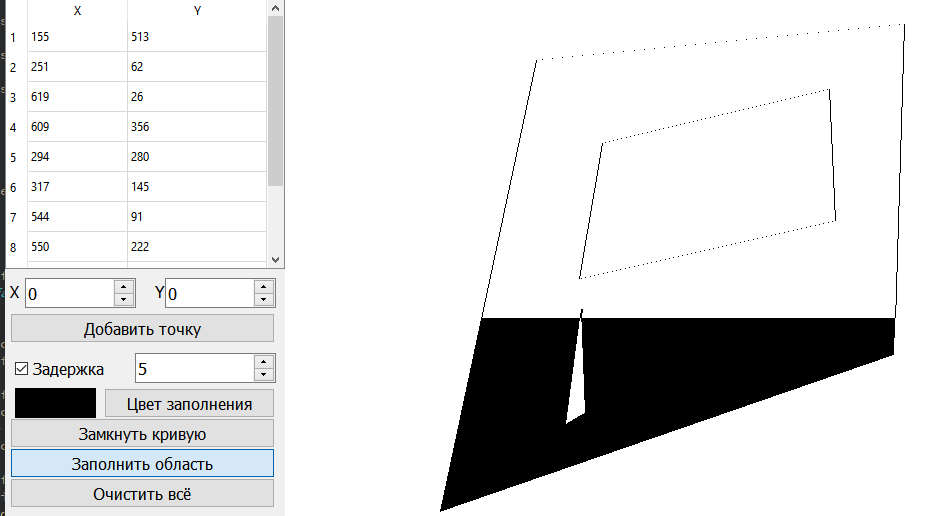
return elapsedTime;

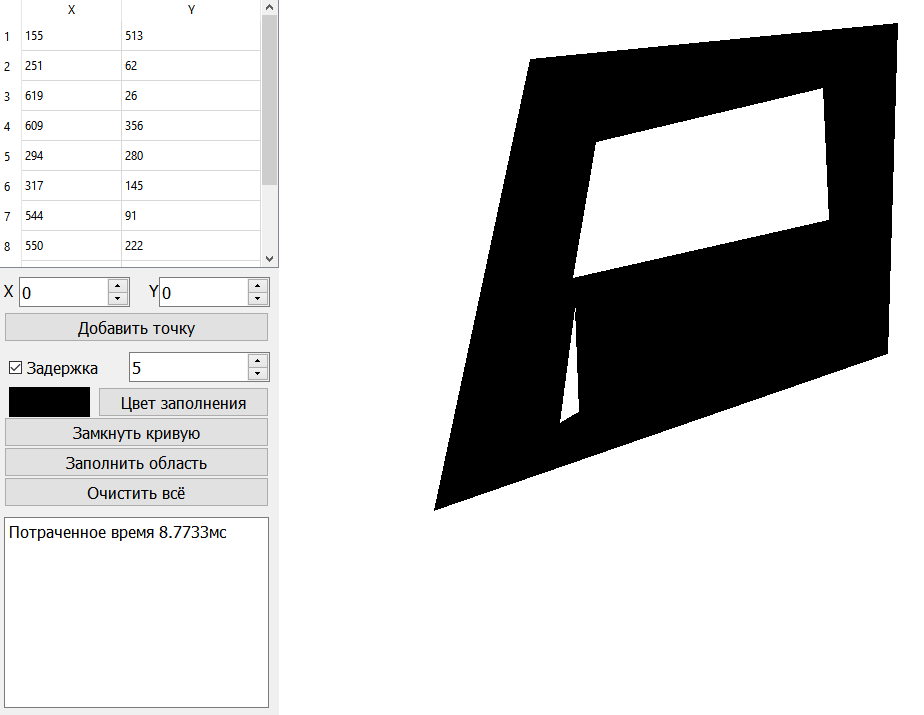
}

**Пример работы:**









**Вывод**

Был реализован алгоритм заполнения со списком рёбер и флагом, проверена его работоспособность на различных примерах.