|  |  |
| --- | --- |
| 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 № 9**

|  |  |
| --- | --- |
| **Тема «Отсечение произвольного многоугольника выпуклым отсекателем (алгоритм Сазерленда-Ходжмена)»**  **Дисциплина Компьютерная графика**  **Студент Кузин Антон**  **Группа ИУ7-42Б**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель Куров А.В.** |  |

Москва.

2020 г.**Цель:**

Изучение и программная реализация алгоритма Сазерленда-Ходжмена отсечения многоугольников.

**Техническое задание:**

должна быть разработана программа, позволяющая осуществлять ввод отсекателя, отсекаемого многоугольника и выполнять отсечение многоугольника по границам отсекателя.

Необходимо обеспечить ввод отсекателя – произвольного многоугольника. Высветить его первым цветом. Также необходимо обеспечить ввод отсекаемого многоугольника (высветить вторым цветом). Должна присутствовать проверка отсекателя на выпуклость. Должен быть предусмотрен ввод вершин многоугольника в произвольных точках ребер отсекателя (включая его вершины).

Ввод осуществлять с помощью мыши и нажатия других клавиш.

Выполнить отсечение многоугольника, показав результат третьим цветом. Исходный многоугольник не удалять.

**Теоретические сведения:**

Алгоритм Сазерленда-Ходжмена позволяет провести отсечение произвольного (выпуклого или невыпуклого) многоугольника по границам выпуклого отсекателя. Идея алгоритма заключается в том, что на каждом шаге отсечения исходный и промежуточные многоугольники отсекаются последовательно очередной границей отсекателя. Отсечение многоугольника относительно одной прямой не представляет больших затруднений.

На каждом шаге отсечения алгоритм работает со списком вершин многоугольника и в результате также получается список вершин нового многоугольника. Причем все вершины нового многоугольника лежат по видимую сторону очередной границы отсекателя. Каждое ребро многоугольника отсекается независимо от других, поэтому для подробного рассмотрения сути алгорита достаточно рассмотреть все возможные комбинации взаимного расположения ребра отсекаемого многоугольника и ребра отсекателя.

* Ребро целиком лежит по видимую сторону от ребра отсекателя – в результирующий список заносится одна вершина(конечная).
* Ребро целиком лежит по невидимую сторону от ребра отсекателя – в результирующий список ничего не заносится.
* Ребро видно частично, и оно входит в область видимости – в результирующий список заносится точка пересечения с отсекателя и конечная точка.
* Ребро видно частично, и оно выходит из области видимости – в результирующий список заносится точка пересечения с отсекателя.

Специально рассматривается первая точка. Если точка видима, то она заносится в результирующий список и становится начальной точкой первого ребра. Если же она невидима, то она просто становится начальной точкой ребра и в результирующий список не заносится.

Для определения видимости точки вычисляется векторное произведение векторов A1A3\* A1A2, где A1, A2 – лежат на ребре отсекателя, а A3 – пробная точка. Если оно равно 0, то точка лежит на ребре, в противном случае знак полученного произведения сравнивается со знаком векторного произведения, полученного при проверке отсекателя на выпуклость – если знаки противоположны, то точка видима, иначе невидима.

Для определения точки пересечения, как и в алгоритме Кируса-Бека, используется параметрическое задание отрезков.

При реализации алгоритма Сазерленда - Ходжмена одна из возникающих проблем - рисование ложных ребер. Ложные ребра - это ребра, которые являются дополнительными, они не являются границами результирующих многоугольников. Ложные ребра появляются в том случае, когда в результате отсечения получается несколько не связанных друг с другом многоугольников. Ложные ребра (они совпадают с ребрами отсекателя или их частями) как раз и соединяют между собой получаемые многоугольники

**Практическая часть:**

float crossProduct(const QPointF &v1, const QPointF &v2)

{

return v1.x() \* v2.y() - v2.x() \* v1.y();

}

float dotProduct(const QPointF &v1, const QPointF &v2)

{

return v1.x() \* v2.x() + v1.y() \* v2.y();

}

int on\_side(const QPoint &check, const QPoint &a1, const QPoint &a2)

{

return sign(crossProduct(a1 - check, a1 - a2));

}

bool checkIntersection(const QPoint &p1, const QPoint &p2, const QPoint &a1, const QPoint &a2)

{

return (on\_side(p1, a1, a2) \* on\_side(p2, a1, a2)) <= 0;

}

QPoint intersection(const QPoint &p1, const QPoint &p2, const QPoint &a1, const QPoint &a2)

{

float det = crossProduct(p2 - p1, a1 - a2);

double t = crossProduct(a1 - p1, a1 - a2) / det;

return p1 + (p2 - p1) \* t;

}

QVector<QPoint> clip(const QVector<QPoint> &polygon\_t, const QVector<QPoint> &clipper\_t, int direction)

{

QVector<QPoint> result;

QVector<QPoint> polygon = polygon\_t;

QVector<QPoint> clipper = clipper\_t;

QPoint first, start;

for (int i = 0; i < clipper.length() - 1; i++){

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

if (!j)

first = polygon[j];

else if (checkIntersection(start, polygon[j], clipper[i], clipper[i + 1])){

result.append(intersection(start, polygon[j], clipper[i], clipper[i + 1]));

}

start = polygon[j];

if (on\_side(start, clipper[i], clipper[i+1]) \* direction <= 0)

result.append(start);

}

if (!result.isEmpty()){

if (checkIntersection(start, first, clipper[i], clipper[i + 1]))

result.append(intersection(start, first, clipper[i], clipper[i + 1]));

}

polygon = result;

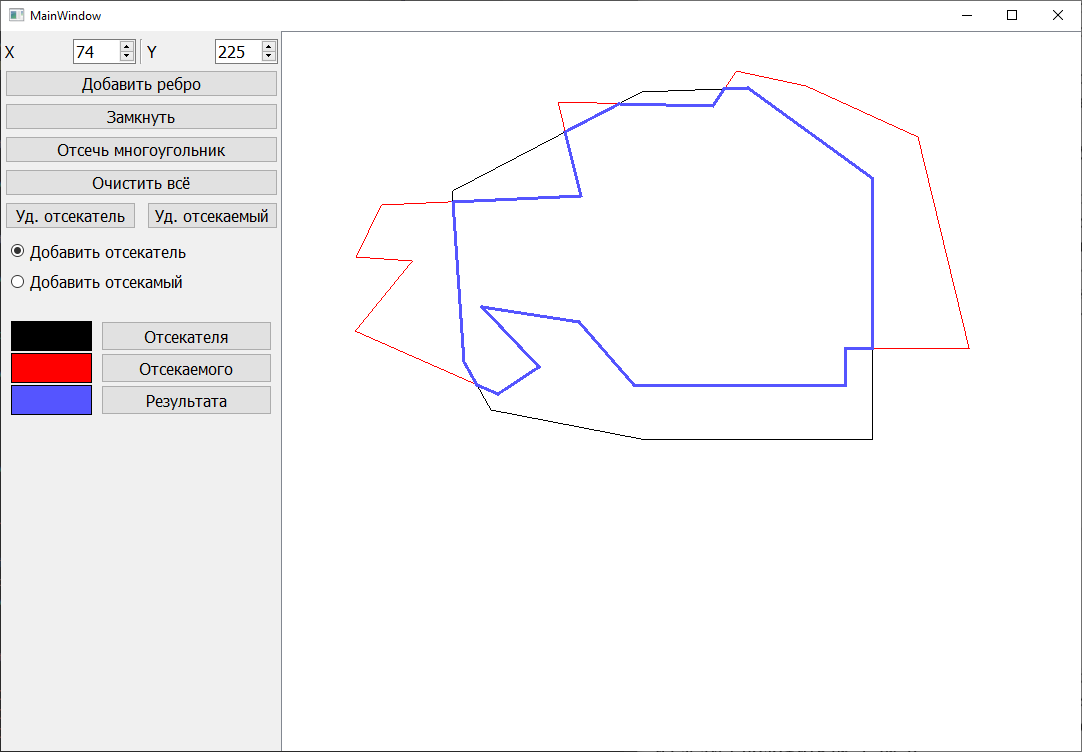
result.clear();

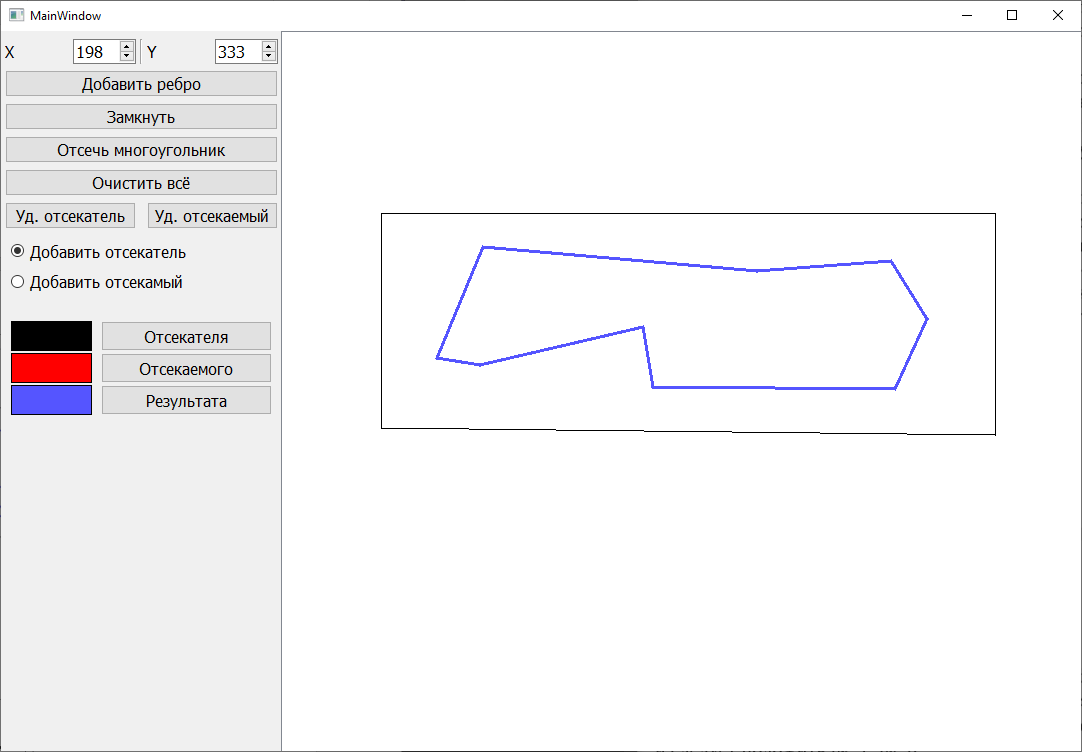
}

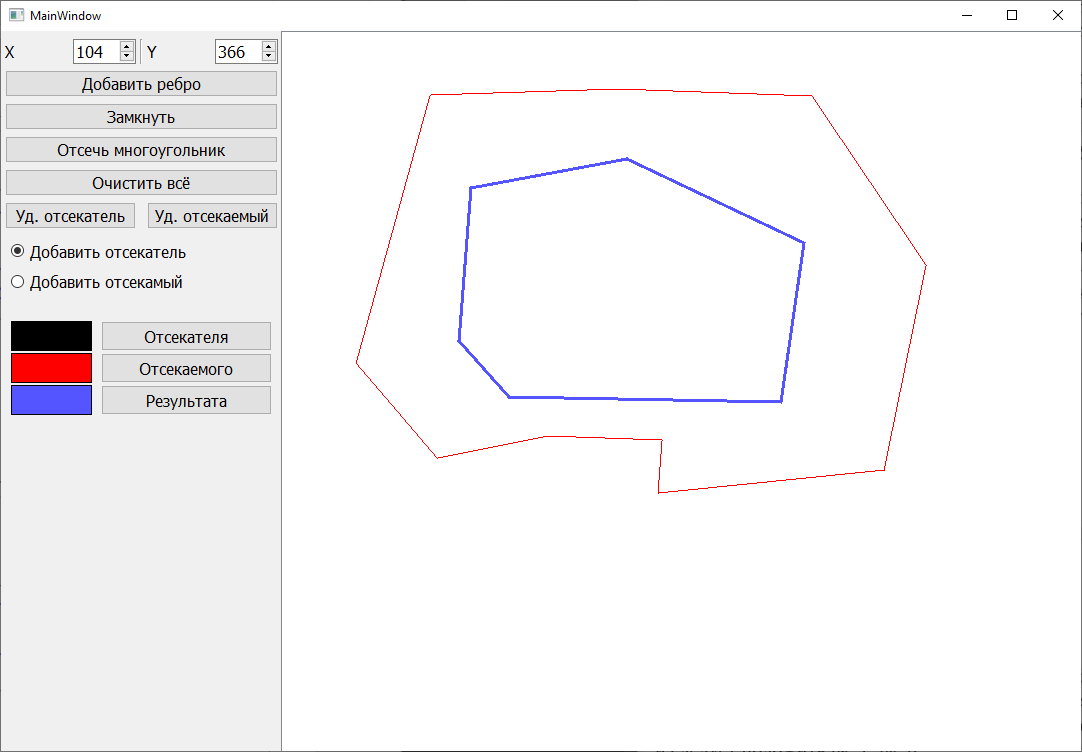
return polygon;

}

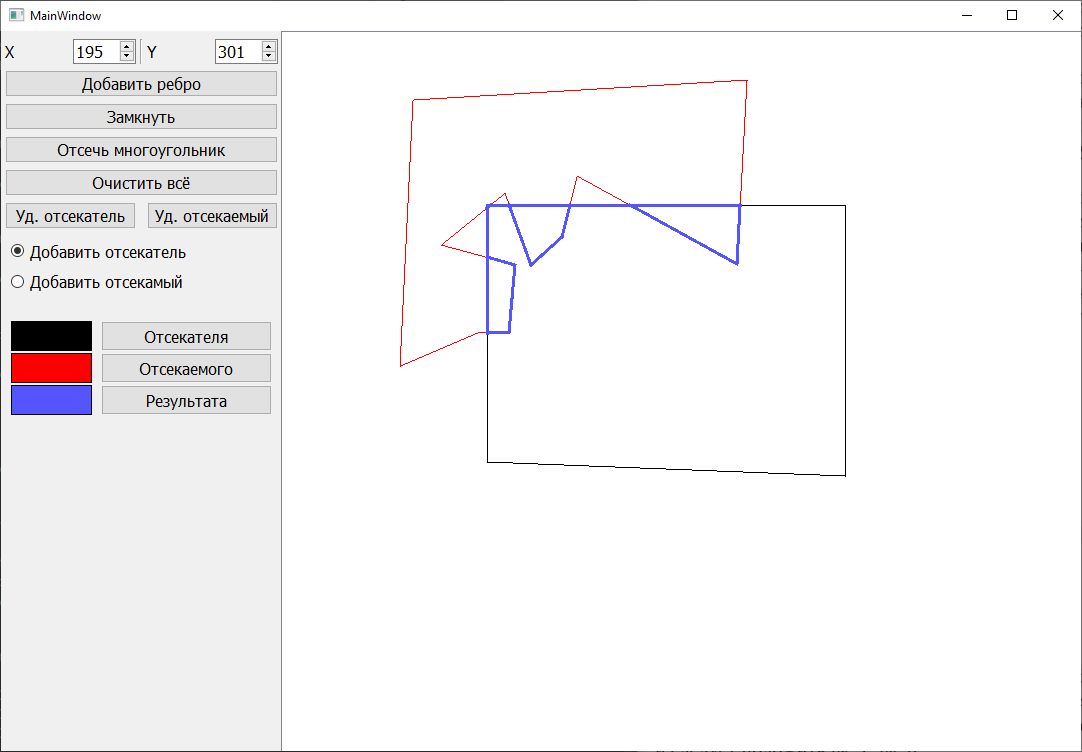
**Пример работы:**







Пример с ложными ребрами:



**Вывод**

В ходе выполнения лабораторной работы был реализован и изучен алгоритм Сазерленда-Ходжмена и проверена его работа.