|  |  |
| --- | --- |
| **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 № \_8\_**

|  |  |
| --- | --- |
| **Тема \_ Реализация алгоритма отсечения отрезка произвольным выпуклым отсекателем.** (Алгоритм Кируса-Бека)**. \_**  **Студент \_Чаушев Александър Красимиров\_\_\_\_\_\_\_\_**  **Группа \_ИУ7-46Б\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_Куров А. В.\_\_\_\_** |  |

Москва.

2020 г.

**Цель работы:** Изучение и программная реализация алгоритма отсечения отрезка.

Необходимо обеспечить ввод отсекателя – произвольного многоугольника.

**Входные данные:** Координаты вершин многоугольника, цвет .

**Выходные данные:** Пользовательское меню, содержащее поля ввода и конечннoe изображение. Вывод замера времени выполнения алгоритма( без задержки ).

**Ошибочные ситуации:** Программа прекращается, если хотя бы один из входных данных не корректен.

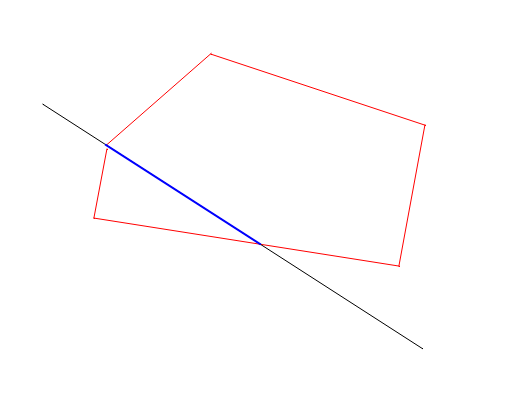
**Теоретическая часть:**

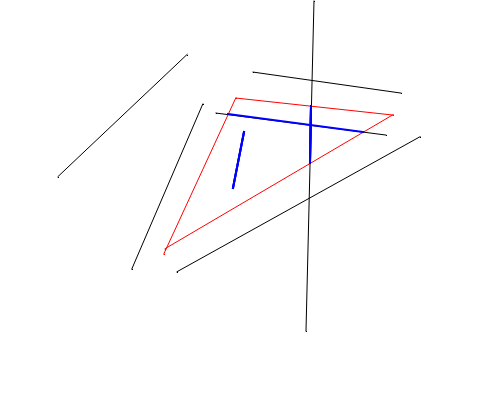
В работе алгоритма Кируса-Бека используется параметрическое представление отрезка: Ps (t)=P1+(P2-P1)\*t;t∈[0;1]. Данный алгоритм позволяет выполнять отсечение не только прямоугольным окном, но и любым выпуклым многоугольником.

Алгоритм

1. D=P2-P1
2. tн=0, tв=1
3. Цикл по всем граням i=1:N
   1. Вычисление nвнi
   2. Di=D\*nвнi
   3. Wi=(P1-fi)\*nвнi
   4. Di=0?
      1. да. Wi≥0? нет - Конец. да - продолжаем
      2. нет. t=-Wi/Di; Di>0?
         1. да. t>1?
            1. нет. tн=max(tн, t)
            2. да. Конец
         2. нет. t<0?
            1. нет. tв=min(tв, t)
            2. да. Конец
4. tн≤tв? да – нарисовать(P(tн), P(tв))
5. Конец

**Пример работы:**





void CyrusBekaAlgorithm(QPoint P1, QPoint P2, int obhod, QVector<edge\_t> rect, QPainter &painter)  
{  
 double tb = 0;  
 double te = 1;  
 double t\_curr = 0;  
 QPoint D;  
 D.setX(P2.x() - P1.x());  
 D.setY(P2.y() - P1.y());  
 for (int i = 0; i < rect.size(); i++)  
 {  
 QPoint W;  
 W.setX(P1.x() - rect[i].x1);  
 W.setY(P1.y() - rect[i].y1);  
  
 QPoint nVector;  
 if (i == rect.size() - 1)  
 {  
 if (obhod == 1)  
 {  
 nVector.setX(-(rect[0].y1 - rect[i].y1));  
 nVector.setY((rect[0].x1 - rect[i].x1));  
 }  
 else  
 {  
 nVector.setX((rect[0].y1 - rect[i].y1));  
 nVector.setY(-(rect[0].x1 - rect[i].x1));  
 }  
 }  
 else  
 {  
 if (obhod == 1)  
 {  
 nVector.setX(-(rect[i + 1].y1 - rect[i].y1));  
 nVector.setY((rect[i + 1].x1 - rect[i].x1));  
 }  
 else  
 {  
 nVector.setX((rect[i + 1].y1 - rect[i].y1));  
 nVector.setY(-(rect[i + 1].x1 - rect[i].x1));  
 }  
 }  
 int Dscalar = scalar(D, nVector);  
 int Wscalar = scalar(W, nVector);  
 // проверка на вырожденость отрезка в точку  
 if (Dscalar == 0)  
 {  
 //qDebug() << "Dscalar == 0";  
 // отрезок выродился в точку  
 if (Wscalar < 0)  
 {  
 //qDebug() << "Wscalar == 0";  
 // Точка видима относительно текущей границы  
 return;  
 }  
 }  
 else  
 {  
 t\_curr = -(double)Wscalar / Dscalar;  
 if (Dscalar > 0)  
 {  
 if (t\_curr > 1)  
 {  
 return;  
 }  
 else  
 {  
 tb = qMax(t\_curr, tb);  
 }  
 }  
 else if (Dscalar < 0)  
 {  
 if (t\_curr < 0)  
 {  
 return;  
 }  
 else  
 {  
 te = qMin(t\_curr, te);  
 }  
 }  
 }  
 }  
 if (tb <= te)  
 {  
 painter.drawLine(P1.x() + (P2.x() - P1.x()) \* te,  
 P1.y() + (P2.y() - P1.y()) \* te,  
 P1.x() + (P2.x() - P1.x()) \* tb,  
 P1.y() + (P2.y() - P1.y()) \* tb);  
 }  
 else  
 {  
 qDebug() << "else";  
 }  
 return;  
}