|  |  |
| --- | --- |
| 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 № 7**

|  |  |
| --- | --- |
| **Тема**  Реализация алгоритма отсечения отрезка регулярным отсекателем.  **Студент** Козлов М. А.  **Группа** ИУ7-45Б  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_**Куров А. В.**\_\_\_\_\_** |  |

Москва.

2020 г.

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

**Техническое задание.**

Необходимо обеспечить ввод регулярного отсекателя - прямоугольника. Высветить его первым цветом. Также необходимо обеспечить ввод нескольких (до десяти) различных отрезков (высветить их вторым цветом). Отрезки могут иметь произвольное расположение: горизонтальные, вертикальные, имеющие произвольный наклон.

Ввод осуществлять с помощью мыши и нажатия других клавиш.

Выполнить отсечение отрезков, показав результат третьим цветом. Исходные отрезки не удалять.

Реализовать алгоритм Сазерленда-Коэна (вариант 11).

**Теоретический материал.**

Алгоритмы отсечения бывают дву- или трёх мерными и применяются к регулярным или нерегулярными областями и объёмами. Рассмотрим двумерное отсечение регулярным отсекателем.

Определение видимости точки. Введём четырёхбитный код T.

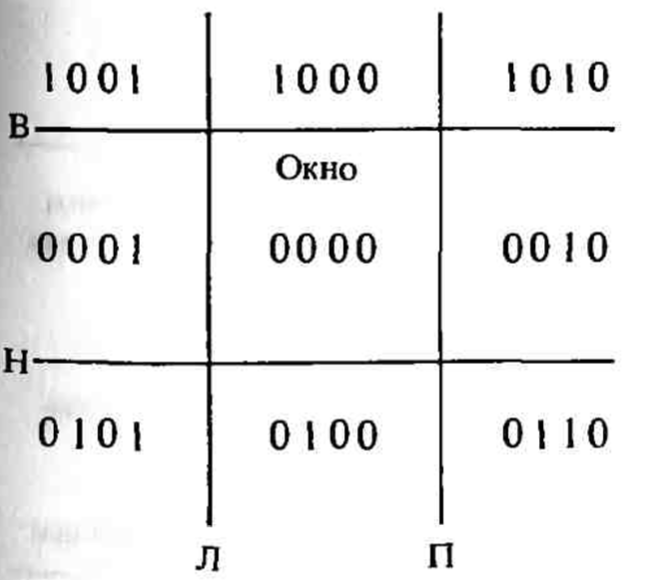
Первый бит – точка левее окна.

Второй бит – точка правее окна.

Третий бит - точка ниже окна.

Четвёртый бит – точка выше окна.

Получаем 9 возможных комбинаций кода точки.



Отсюда следует, что если сумма кодов концов отрезка равна нулю, то отрезок полностью видим. Если побитовое И кодов концов отрезка не равно нулю, то отрезок тривиально невидим.

В алгоритме Сазерленда-Коэна отрезок разбивается сторонами окна.

Для каждого отрезка P1P2 определяется его полная видимость или тривиальная невидимость.

Если P1 внутри окна, поменять местами P1 и P2.

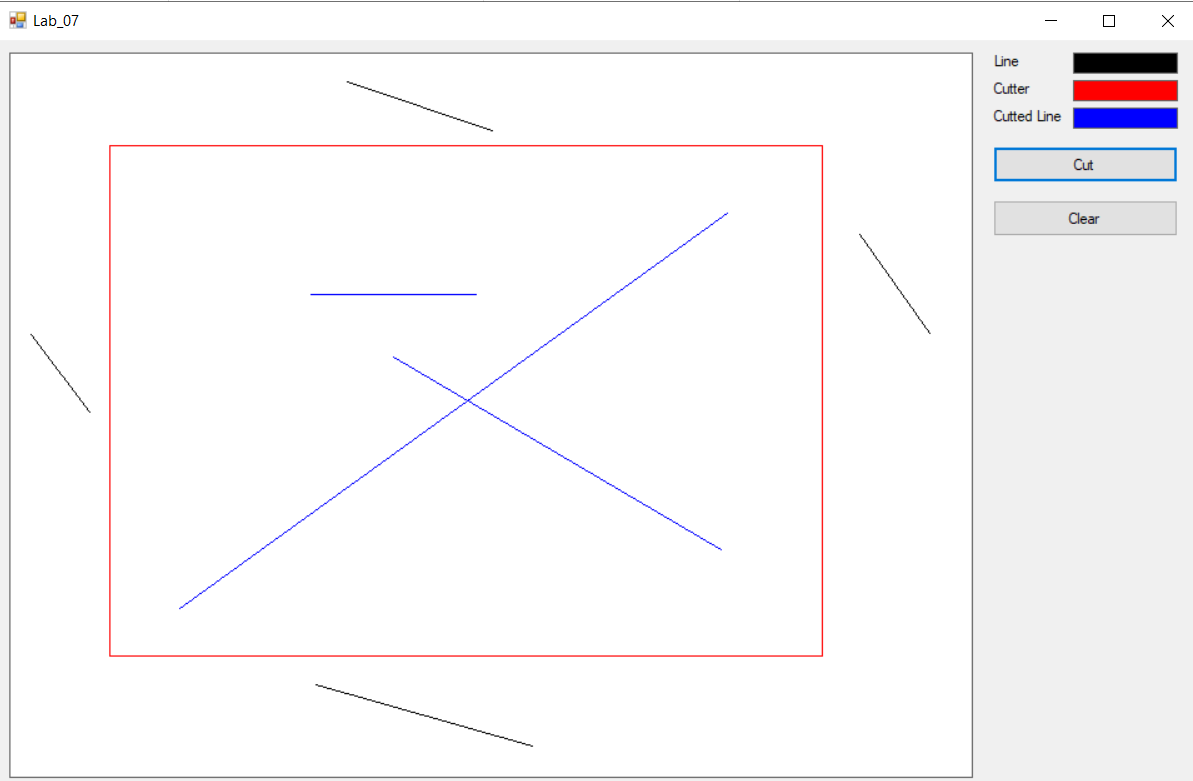
Заменить P1 на точку пересечения P1P2 со стороной окна.

**Примеры работы программы.**

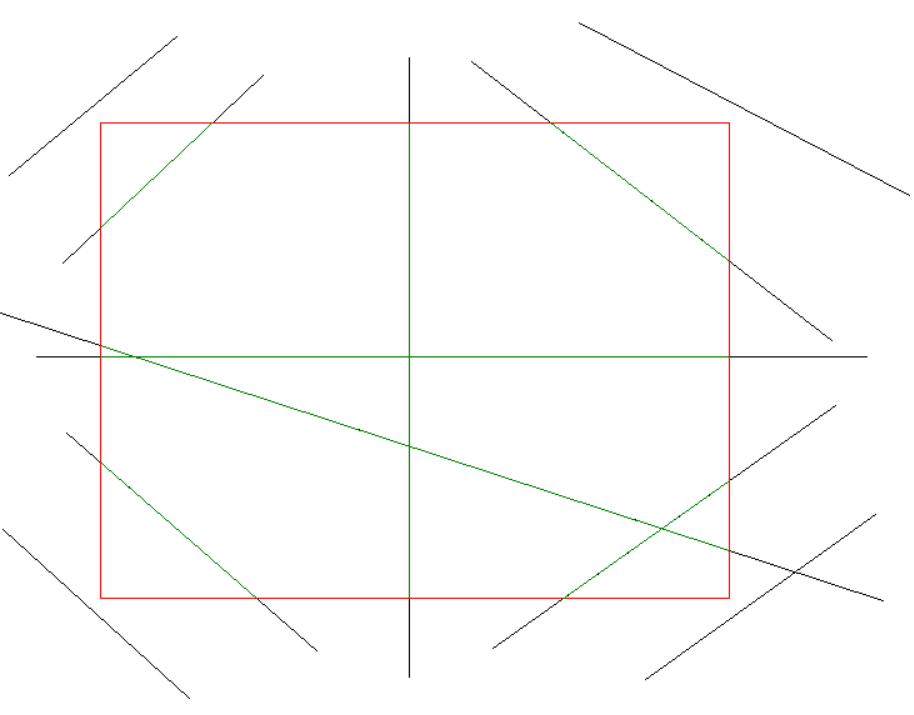
Ввод горизонтальных и вертикальных рёбер осуществляется при зажатии клавиши SHIFT.

Ввод координат отсекателя осуществляется кликами правой кнопкой мыши.

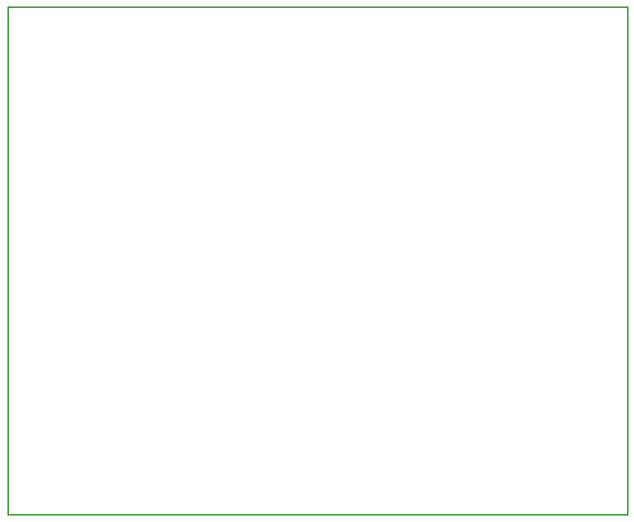
Все отрезки или полностью видимые, или тривиально невидимые.



Частично видимые отрезки и нетривиально невидимые.



Отрезки лежат на границе отсекателя.



**Исходный код.**

// Получение кода точки

protected int GetCode(PointF p)

{

int sum = 0b0000;

if (p.X < A.X) // left

sum |= 0b1000;

if (p.X > B.X) // right

sum |= 0b0100;

if (p.Y < A.Y) // down

sum |= 0b0010;

if (p.Y > B.Y) // up

sum |= 0b0001;

return sum;

}

// отрезок полносью видимый

protected bool IsLineVisible(int t0, int t1)

{

return (t0 | t1) == 0;

}

// отрезок тривиально невидим

protected bool IsTrivialInvisible(int t0, int t1)

{

return (t0 & t1) != 0;

}

// массив c координатами границ отсекателя

protected int this[int index]

{

get

{

switch(index)

{

case 0:

return A.X; // left

case 1:

return B.X; // right

case 2:

return A.Y; // down

case 3:

return B.Y; // up

default:

throw new IndexOutOfRangeException();

}

}

}

public PointF[] Cut(PointF P1, PointF P2)

{

float m = 0;

float Im = 0;

float dx = P2.X - P1.X;

float dy = P2.Y - P1.Y;

bool isVert = dx == 0;

bool isHor = dy == 0;

if (!isVert)

m = dy / dx;

if (!isHor)

Im = dx / dy;

int mask = 0b1000; // для обращения к i-ому разряду кода вершины

for (int i = 0; i < 4; i++)

{

int t0 = GetCode(P1);

int t1 = GetCode(P2);

if (IsLineVisible(t0, t1))

return new PointF[] { P1, P2};

if (IsTrivialInvisible(t0, t1))

return null;

bool t0i = (t0 & mask) != 0;

bool t1i = (t1 & mask) != 0;

// если t0i или t1i равен единице.

if (t0i != t1i)

{

if (!t0i) // Если точка P1 по видимую сторону, поменять местами вершины

Swap(ref P1, ref P2);

if (!isVert && i < 2)

{

P1.Y = m \* (this[i] - P1.X) + P1.Y;

P1.X = this[i];

}

else

{

if (!isHor)

{

P1.X = Im \* (this[i] - P1.Y) + P1.X;

P1.Y = this[i];

}

}

}

mask /= 2;

}

return new PointF[] { P1, P2};

}