

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Дисциплина « Компьютерная графика»**

**Лабораторная работа №7**

**по теме:**

**«Реализация алгоритма отсечения отрезка регулярным отсекателем.»**

**Работу выполнил:**

студент группы ИУ7-44Б

Турчанинов А.М.

**Работу проверил:**

Погорелов Д.А.

2020 г.

**Цель работы:**

Изучение и программная реализация алгоритма отсечения отрезка.

**Задание:**

Реализация алгоритма Сазерленда-Коэна.

**Требования:**

1. Необходимо обеспечить ввод регулярного отсекателя - прямоугольника. Высветить его первым цветом.
2. Необходимо обеспечить ввод нескольких (до десяти) различных отрезков (высветить их вторым цветом).
3. Реализовать ввод любых отрезков (горизонтальные, вертикальные, имеющие произвольный наклон).
4. Ввод осуществлять с помощью мыши и нажатия других клавиш.
5. Выполнить отсечение отрезков, показав результат третьим цветом.
6. Исходные отрезки не удалять.

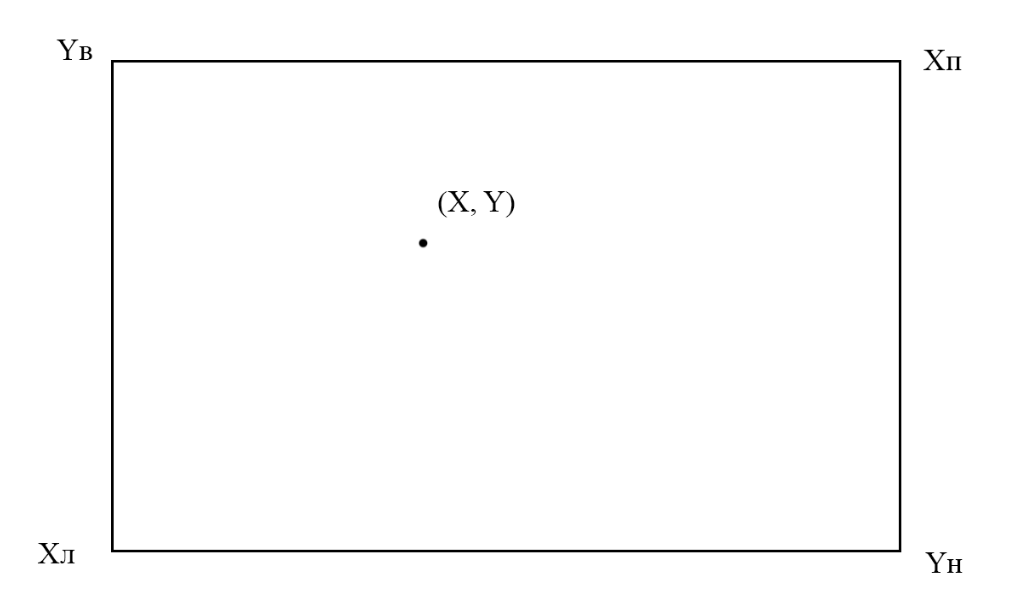
**Теоретический материал:**

Регулярным (стандартным) отсекателем на плоскости является прямоугольник со сторонами, параллельными координатным осям.

Нерегулярным отсекателями является произвольные выпуклые и невыпуклые многоугольники. В данной лабораторной работе мы имеем дело с регулярными отсекателями.

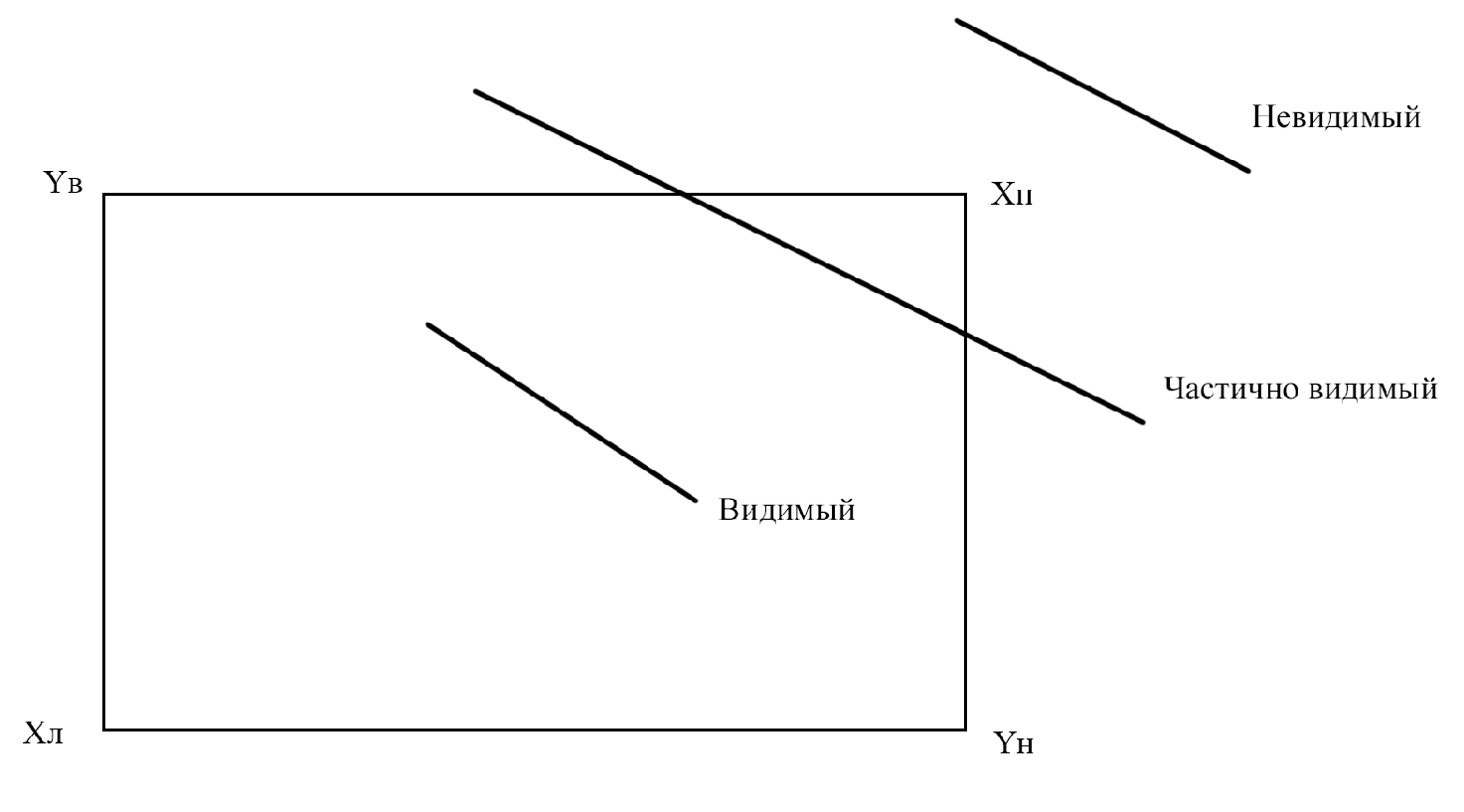
Для выполнения отсечения регулярным отсекателем необходимо задать абсциссы Xл, Xп левого и правого ребер и ординаты Yн,Yв нижнего и верхнего ребер. Цель отсечения будет состоять в определении точек, отрезков или их частей, которые лежат внутри отсекателя.

Рассмотрим отсечение простого объекта, а именно точки:



Для того, чтобы точка была внутри отсекателя (Т.е. видимой) должно выполнятся данной условие: Xл <= X <= Xп и Yн <= Y <= Yв. Мы считаем, что если точка лежит на стороне отсекателя, то она видима.

Введем понятия видимых, невидимых и частично видимых отрезков:



Далее было предложено положение точки характеризовать с помощью четырехразрядного кода (Обозначим как T(1-4)). Он представляет из себя массив из 4-ёх элементов, где каждый элемент содержит булево значение (1 или 0) и задается данными условиями:

Ti - i-ый разряд.

T1 = 1, если X < Xл, иначе 0

T2 = 1, если X > Xп, иначе 0

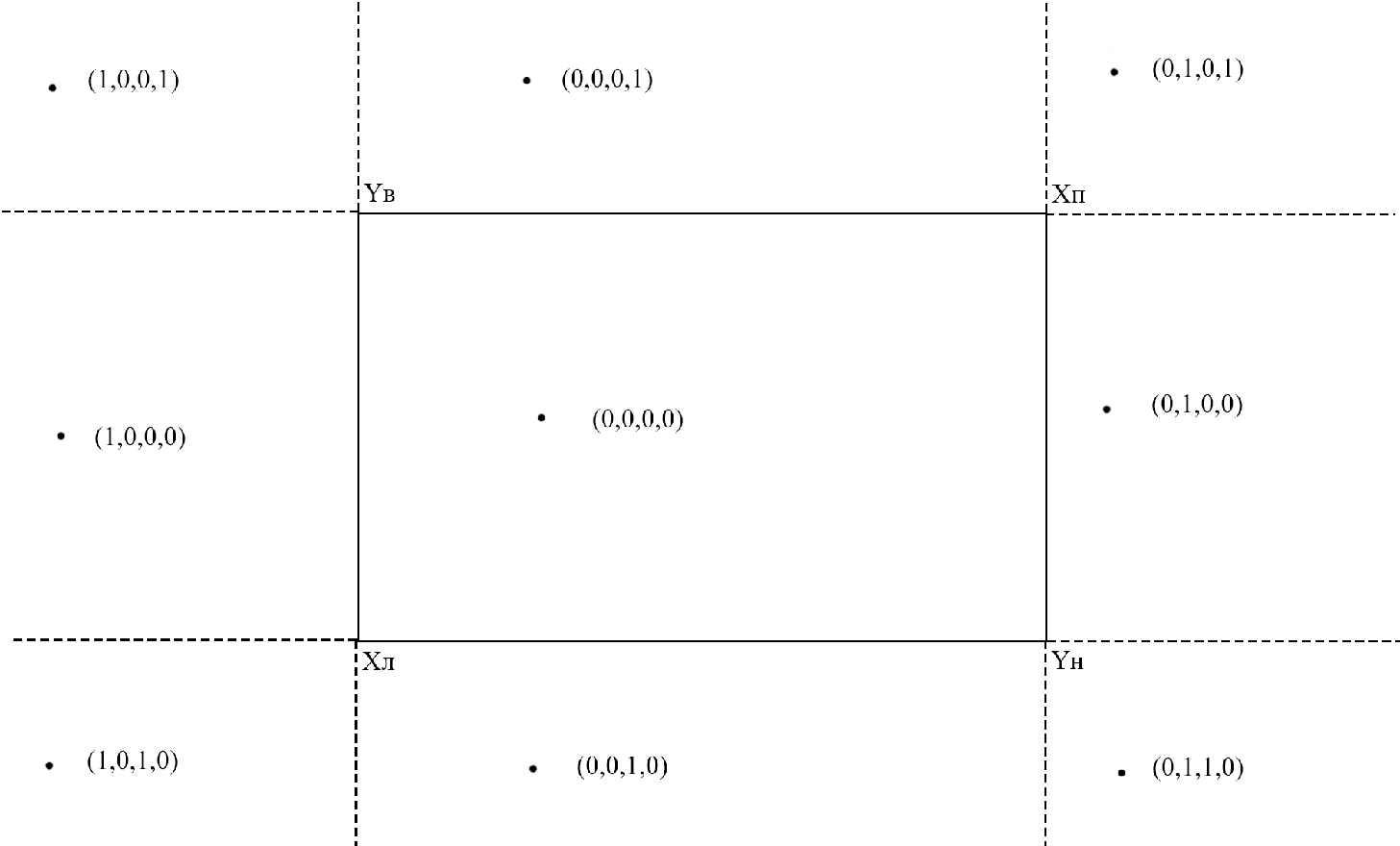
T3 = 1, если Y < Yн, иначе 0

T4 = 1, если Y > Yв, иначе 0

Получается, что отсекатель мы представляем в виде: ( Xл, Xп, Yн,Yв), а четырехразрядной код точки имеет единицу в i-ом разряде, если он расположен по невидимую сторону от данного ребра (Т.е. левее, правее, ниже или выше нашего отсекателя).

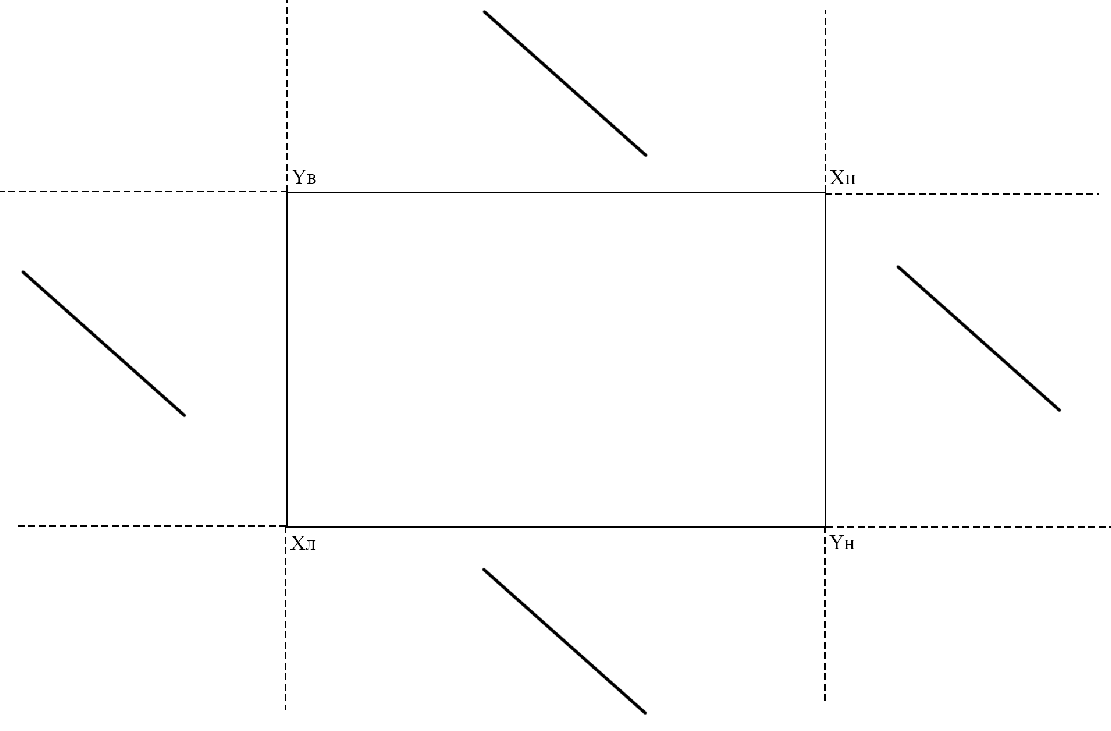
Данный четырехразрядный код позволяет легко определить полностью видимую точку, а именно во всех разрядах должен быть ноль.

Ниже представлены четырехразрядный код для каждой области:

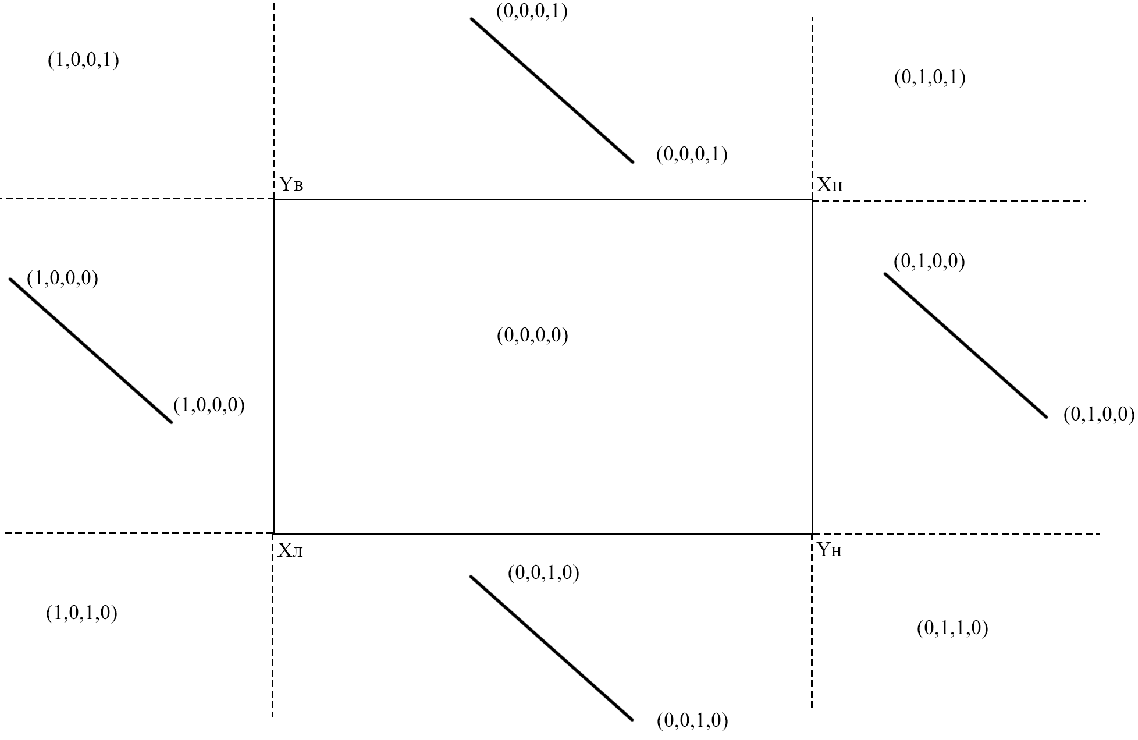


Сразу становится понятно, что для того, чтобы отрезок был полностью видимым нам нужно, чтобы начало и конец были видимы. Т.е. сумма концов его четырехразрядного кода равнялась нулю.

Далее введем понятие тривиально невидимого отрезка:



А теперь посмотрим на их четырехразрядный код:



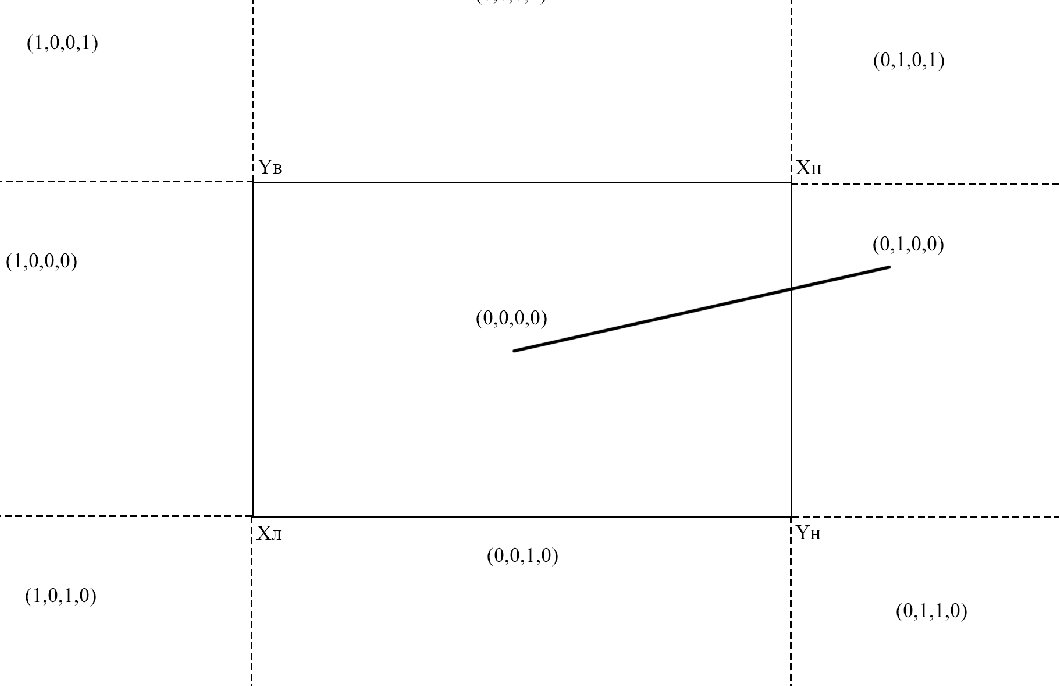
Полную невидимость таких отрезков легко определить, т.к. нам достаточно поразрядно перемножить четырехразрядные коды концов. И мы можем убедиться, что сумма данного перемножения будет отлична от нуля, а это означает, что он тривиально невидимый.

Пример: Возьмем четырехразрядные коды концов отрезка, расположенные левее левой границы отсекателя и перемножим их:

(1,0,0,0) \* (1,0,0,0) = 1 \* 1 + 0 \* 0 + 0 \* 0 + 0 \* 0 = 1

Аналогично для отрезков, расположенных правее правой, ниже нижней и выше верхней границе.

Если одна вершина видима, а другая нет, то отрезок заведомо частично видимый и нам нужно найти вторую вершину видимой части отрезка.



Теперь перейдем непосредственно к нахождению пересечения нашего отрезка и стороной отсекателя.

Мы знаем об отрезке Xначальное, Yначальное (обозначим как (Xн, Yн)) и Xконечное, Yконечное (обозначим как (Xк, Yк)). Мы легко можем найти tg(Обозначим как m). Если tg отличен от ∞ (т.е. отрезок не вертикальный), то рассмотрим нахождение пересечения со сторонами отсекателя:

Пересечение с вертикальной стороной отсекателя:

Нам нужно найти y т.к. x известен (Он совпадает с x-овой координатой стороны отсекателя (x\_отсекателя) ). Также нам известно m. y = mx + b. Отсюда b = y - mx. Подставляем b, m и x\_отсекателя в уравнение прямой:

y = m \* x\_отсекателя + y - m \* x = m\*(x\_отсекателя - x) + y

Под x\_отсекателя понимается Хп или Хл (В зависимости от того, с какой стороной ищем пересечение).

Пересечение с горизонтальной стороной отсекателя:

В данной ситуации нам нужно найти x т.к. y известен (Он совпадает с y-овой координатой стороны отсекателя (y\_отсекателя) ). y = mx + b. Выразим x: x = (y - b) / m = 1/m \* (y - b). Подставим известные нам значения. x = 1/m \* (y\_отсекателя - (y - mx)) = 1/m \* (y\_отсекателя - y + mx) = 1/m \* (y\_отсекателя - y) + x

x = 1/m \* (y\_отсекателя - y) + x. Под y\_отсекателя понимается Yн или Yв (В зависимости от того, с какой стороной ищем пересечение).

**Алгоритм:**

Данный алгоритм ищет пересечение с каждой стороной отсекателя, и, если она есть, то отбрасывает невидимую часть отрезка.

Для начала нам нужно определить положение отрезка и tg (если отрезок не вертикальный). За это будет отвечать flag и m (tg).

Положение отрезка:

flag = 1 - общего положения.

flag = 0 - горизонтальный.

flag = -1 - вертикальный.

Далее начинается сам алгоритм: мы итерируемся 4 раза (т.к. у нас регулярный отсекатель с 4-мя границами). Каждый раз мы определяем видимость отрезка. Видимость:

1 = видимый.

0 = частично видимый.

-1 = невидимый.

Если он полностью видимый, то высвечиваем его. Если невидимый, то завершаем алгоритм. Иначе ищем пересечение (если оно есть). Как именно мы ищем пересечение рассмотрено выше. После чего отбрасываем невидимую часть отрезка.

**Реализация алгоритма:**

**def cohen\_sutherland(line, rectangle):**

**# Возвращает концы видимого отрезка.**

**# Если отрезок невидим, то возвращает INVISIBLE\_LINE**

**# Изначально считаем, что отрезок общего положения.**

**flag, m = NORMAL\_LINE, 1**

**# Проверка на то, что отрезок вертикальный.**

**if line[X1] - line[X2] == 0:**

**flag = VERTICAL\_LINE**

**else:**

**# Если отрезок не вертикальный, то вычисляем тангенс.**

**m = (line[Y2] - line[Y1]) / (line[X2] - line[X1])**

**# Проверка на то, что отрезок горизонтальный.**

**if m == 0:**

**flag = HORIZONTAL\_LINE**

**# Итерируемся по 4 сторонам отсекателя**

**# В порядке: (x\_левое, x\_правое, y\_нижнее, y\_верхнее)**

**for i in range(4):**

**# Формируем четырехразрядный код начала отрезка.**

**code\_1 = create\_code([line[X1], line[Y1]], rectangle)**

**# Формируем четырехразрядный код конца отрезка.**

**code\_2 = create\_code([line[X2], line[Y2]], rectangle)**

**# Определяем видимость отрезка.**

**vis = is\_visible(code\_1, code\_2, rectangle)**

**# Если отрезок видимый, возвращаем его начало и конец.**

**if vis == VISIBLE\_LINE:**

**return line**

**# Если отрезок невидимый возвращаем признак невидимости.**

**elif vis == INVISIBLE\_LINE:**

**return INVISIBLE\_LINE**

**# Проверяем пересечение отрезка и стороны отсекателя.**

**# До этого момента не дойдет отрезок, у которого две**

**# координаты по одну сторону, потому что выше мы это**

**# обработали.**

**# Т.е. в данном случае может могут быть только**

**# Такие значения code\_1[i] и code\_2[i]:0 и 1, 1 и 0, 0 и 0**

**# 1 и 0, 0 и 1 - означает, что данный отрезок пересекает**

**# Данную сторону отсекателя (Т.е. одна его сторона**

**# находится**

**# По невидимую сторону, а вторая по видимую => есть**

**# Пересечение с данной стороной) - это то, что нам нужно.**

**# 0 и 0 - означает обратное => отрезок ее не пересекает.**

**if code\_1[i] == code\_2[i]:**

**continue**

**# Т.к. мы собираемся искать пересечение**

**# Прямой и стороны отсекателя мы должны будем после**

**# Того, как найдем данное пересечение**

**# Присвоить одной вершине отрезка найденное пересечение.**

**# Чтобы корректной вершине присвоить пересечение**

**# Мы должны проверить, что данная вершина находится по**

**# Невидимую сторону стороны отсекателя. Если это не так**

**# То поменять местами. Если данную проверку не произвести**

**# (И обмен),**

**# То получится, что после того, как мы найдем пересечение**

**# Мы можем присвоить вершине, которая и так является**

**# видимой, а та**

**# Вершина, которая находилась вне отсекателя так там и**

**# останется.**

**# Т.е. вместо того, чтобы отбросить невидимую часть**

**# отрезка**

**# Мы можем (Если не произведем данный обмен) отбросить**

**# видимую часть.**

**if not code\_1[i]:**

**line[X1], line[Y1], line[X2], line[Y2] = line[X2], line[Y2], line[X1], line[Y1]**

**# Если не вертикальная линия.**

**if flag != VERTICAL\_LINE:**

**# Т.к. rectangle представлен данном виде:**

**# (x\_левое, x\_правое, y\_нижнее, y\_верхнее)**

**# Это значит, что при i<2 мы ищем пересечение**

**# Либо с x\_левой либо с x\_правой.**

**if i < 2:**

**# Находим пересечение.**

**line[Y1] = m \* (rectangle[i] - line[X1]) + line[Y1]**

**# Мы точно знаем, что раз есть пересечение,**

**# То координата x будет соответствовать x\_левому**

**# или x\_правому**

**# В зависимости от того, с какой стороной**

**# Мы на данный момент работаем.**

**line[X1] = rectangle[i]**

**continue**

**else:**

**line[X1] = (1 / m) \* (rectangle[i] - line[Y1]) + line[X1]**

**line[Y1] = rectangle[i]**

**return line**

**Результат работы:**

Синим цветом показаны отрезки внутри отсекателя.

