**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

отчет

**по лабораторной работе №4**

**по дисциплине «Компьютерная графика»**

Тема: **Исследование алгоритмов отсечения отрезков и многоугольников окнами различного вида**

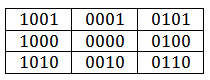
|  |  |  |
| --- | --- | --- |
| Студент гр. 7307 |  | Торопов В.А. |
| Преподаватель |  | Матвеева И.В. |

**Цель работы**

Обеспечить реализацию алгоритма отсечения массива произвольных отрезков заданным прямоугольным окном с использованием четырех битового кода для концов каждого анализируемого отрезка. Вначале следует вывести на экран сгенерированные отрезки полностью, а затем другим цветом или яркостью те, которые полностью или частично попадают в область окна.

**Основные теоретические положения**

Алгоритм Сазерленда-Коэна заключается в том, что концам отрезка присваивается четырёхбитный код b0, b1, b2, b3. Этот четырёхбитный код содержит информацию о положении точки относительно области вывода. На практике возможны 9 комбинаций:



Коды составляются в соответствии со следующими правилами:

* b0=0, если x≥xmin;
* b0=1, если x<xmin;
* b1=0, если x≤xmax;
* b1=1, если x>xmax;
* b2=0, если y≥ymin;
* b2=1, если y<ymin;
* b3=0, если y≤ymax;
* b3=1, если y>ymax;

После того, как коды получены, возможны следующие варианты:

1. Коды содержат только 0, а значит отрезок целиком лежит внутри окна и должен быть отрисован целиком;
2. Коды содержат единичный бит в одной и той же позиции, а значит, отрезок лежит за пределами окна и не будет отрисован;
3. Во всех остальных случаях в окне лежит только часть отрезка, и это значит, что есть необходимость в отсечении.

В случае с 3-м случаем идёт перемещение концов отрезков по отрезку, составляются их коды и после этого вновь сравниваются. Операция происходит до того момента пока не выполнится условие 1 или 2.

**Математическое обоснование (Python)**

Получение 4-х разрядного битового кода для вершины отрезка (В качестве битового кода возвращается целое число, чтобы упростить операции проверки):

def get\_bytes(lines, i, j):  
 *"""  
 Getting array of bytes as an integer (for ease of operation) according to the rule* ***:param*** *lines: generated segments* ***:param*** *i: number of segment* ***:param*** *j: number of point of segment* ***:return****: array of bytes as an integer  
 """* if (lines[i][j][0] >= min(window[0][0], window[1][0])) and (lines[i][j][0] <= max(window[1][0], window[0][0])):  
 if lines[i][j][1] >= min(window[0][1], window[1][1]):  
 if lines[i][j][1] <= max(window[0][1], window[1][1]):  
 return 0  
 else:  
 return 1  
 else:  
 return 2  
 elif (lines[i][j][0] >= min(window[0][0], window[1][0])) and (lines[i][j][0] > max(window[1][0], window[0][0])):  
 if lines[i][j][1] >= min(window[0][1], window[1][1]):  
 if lines[i][j][1] <= max(window[0][1], window[1][1]):  
 return 4  
 else:  
 return 5  
 else:  
 return 6  
 else:  
 if lines[i][j][1] >= min(window[0][1], window[1][1]):  
 if lines[i][j][1] <= max(window[0][1], window[1][1]):  
 return 8  
 else:  
 return 9  
 else:  
 return 10

Функция проверки вхождения путем сравнения битов:

def proof(i):  
 *"""  
 function for checking the inclusion of segment* ***:param*** *i: index of segment* ***:return****: 0 if not included and 1 if included  
 """* line = lines[i].copy()  
 bytes = [0, 0]  
  
 # finding minimum and maximum x of points of segment  
 min\_p = line[0].copy()  
 max\_p = line[1].copy()  
 if min(line[0][0], line[1][0]) == line[1][0]:  
 min\_p, max\_p = max\_p.copy(), min\_p.copy()  
  
 points = [[min\_p.copy(), max\_p.copy()]]  
  
 # moving along the segment and checking the entry conditions  
 for i in range(min\_p[0], max\_p[0], 1):  
 points[0][0][0] = i  
 points[0][0][1] = get\_y(i, line)  
 bytes[0] = get\_bytes(points, 0, 0)  
 bytes[1] = get\_bytes(points, 0, 1)  
 if bytes[0] == 0 or bytes[1] == 0:  
 return 1  
 elif bytes[0] == bytes[1]:  
 return 0  
 points[0][1][0] = max\_p[0] - (i - min\_p[0] + 1)  
 points[0][1][1] = get\_y(points[0][1][0], line)  
 bytes[1] = get\_bytes(points, 0, 1)  
 if bytes[0] == 0 or bytes[1] == 0:  
 return 1  
 elif bytes[0] == bytes[1]:  
 return 0  
 return 2

**Экспериментальные результаты.**

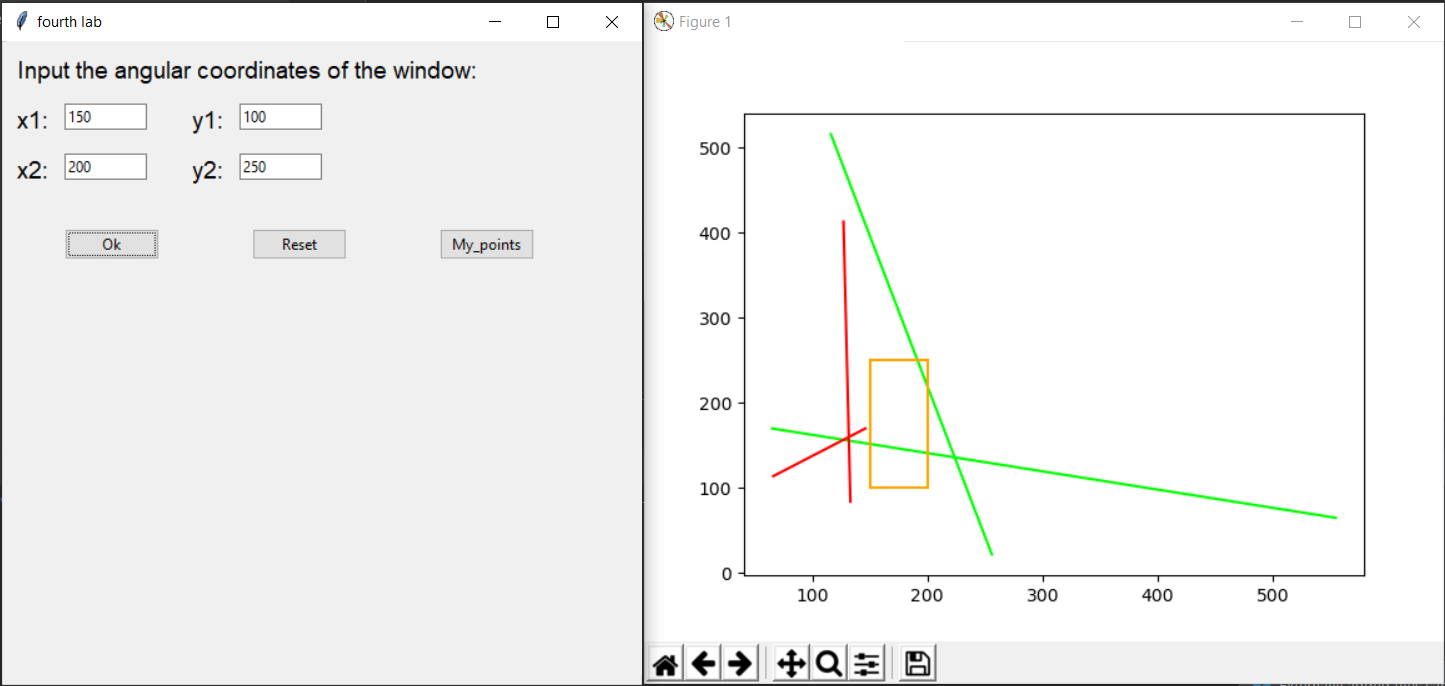
На рисунке 1 представлено отсечение сгенерированных отрезков прямоугольным окном, заданным координатами в форме.

Рисунок 1 – Отсечение отрезков

**Вывод**

В данной лабораторной работе был реализован алгоритм отсечения сгенерированных отрезков заданным прямоугольным окном с использованием четырех битового кода для концов каждого анализируемого отрезка.