МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

ОТЧЕТ

по лабораторной работе № 4

по дисциплине «Компьютерная графика»

Тема: Исследование алгоритмов отсечения отрезков и многоугольников окнами различного вида

Студентка гр. 1361 _____ Галунина Е.С.

Студентка гр. 1361 _____ Горбунова Д.А.

Студентка гр. 1361 _____ Токарева У.В.

Преподаватель _____ Колев Г.Ю.

Санкт-Петербург

2023

Цель работы.

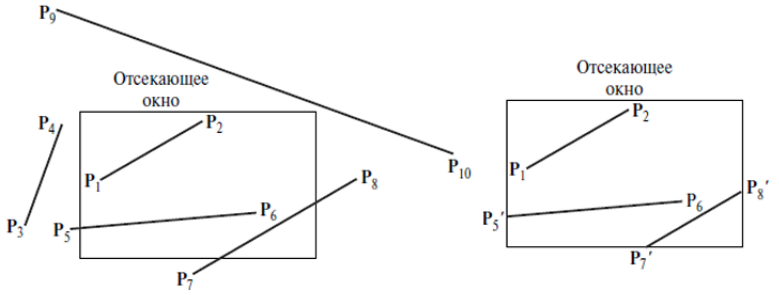
Обеспечить реализацию алгоритма отсечения массива произвольных отрезков заданным прямоугольным окном с использование метода половинного деления. Вначале следует вывести на экран сгенерированные отрезки полностью, а затем другим цветом или яркостью те, которые полностью или частично попадают в область окна.

Основные теоретические положения.

Основные теоретические сведения представлены на рисунках 1 и 2.

ДВУХМЕРНОЕ ОТСЕЧЕНИЕ ЛИНИИ

- На рис. иллюстрируются возможные положения прямых отрезков относительно стандартного отсекающего окна. Алгоритм отсечения линии обрабатывает каждую линию на сцене с помощью последовательности проверок и расчетов точек пересечения, позволяющих определить, нужно ли записывать линию или ее часть.



- Рис. Отсечение прямых отрезков с использованием стандартного прямоугольного отсекающего окна
- а) До отсечения б) После отсечения
- Выполняется расчет точек пересечения линии со сторонами окна.
- **Главная цель** любого алгоритма отсечения линии минимизировать расчет точек пересечения.
- Для этого вначале выполнить проверку - где находится участок линии

1. Полностью внутри или полностью снаружи отсекающего окна.
2. Если линия не относится к полностью внутренним или полностью внешним, нужно вычислить точки пересечения линии с периметром окна.

- Чтобы проверить, находится ли прямой отрезок полностью внутри или полностью снаружи выбранной стороны отсекающего окна, проверяется возможность отсечения точки.

Рисунок 1 – Теоретические сведения

ДВУХМЕРНОЕ ОТСЕЧЕНИЕ ЛИНИИ

1. Если все концы отрезка находятся внутри всех четырех сторон отсекающего прямоугольника, (P_1P_2 на рис.), линия полностью лежит внутри отсекающего окна, и она сохраняется.
2. Если оба конца отрезка находятся вне всех четырех сторон (линия P_3P_4 на рис.), отрезок находится полностью вне окна, и он удаляется из описания сцены.
3. Если ни одна проверка не дала положительного результата, отрезок пересекает по меньшей мере одну сторону отсекающего прямоугольника и может пересекать или не пересекать внутреннюю часть отсекающего окна.

- Параметрическое представление уравнения для прямого отрезка, где точки с координатами (x_0, y_0) и (x_{end}, y_{end}) - концы отрезка.
- $x = x_0 + u(x_{end} - x_0)$;
- $y = y_0 + u(y_{end} - y_0)$; $0 \leq u \leq 1$
- Подставить вместо x или y координаты и найти параметр u, (где отрезок проходит через стороны отсекающего окна).
- Например, левая граница окна - xw_{min}, подставим вместо x, найдем u и вычислим координату y точки пересечения.
- Если $u \notin [0, 1]$, отрезок не пересекает данную границу окна, иначе часть линии находится внутри границ. Затем обработать внутреннюю часть отрезка вместе с другими границами окна, пока не будет удалена вся линия, или пока не будет найден отрезок, полностью лежащий внутри окна.
- Обработка отрезков на сцене с использованием простого подхода отсечения не очень эффективна.
- В настоящее время разработано несколько более быстрых схем отсечения линий.

Рисунок 2 – Теоретические сведения

Формализация.

Работа выполнена на языке программирования Python. Для визуализации данных трехмерной графики используется библиотека matplotlib.

Экспериментальные результаты.

На рисунках 3, 4, 5 представлены результаты эксперимента, где прямые, входящие полностью или частично в необходимое окно автоматически окрашиваются в зеленый цвет.

```
Выберите один из следующих вариантов: 1 -- ввести координаты окна в ручную;     2 --
оставить координаты окна по умолчанию:  1
Введите x-координату верхнего левого угла окна:  0.6
Введите y-координату верхнего левого угла окна:  0.3
Введите x-координату нижнего правого угла окна:  0.5
Введите y-координату нижнего правого угла окна:  0.2
```

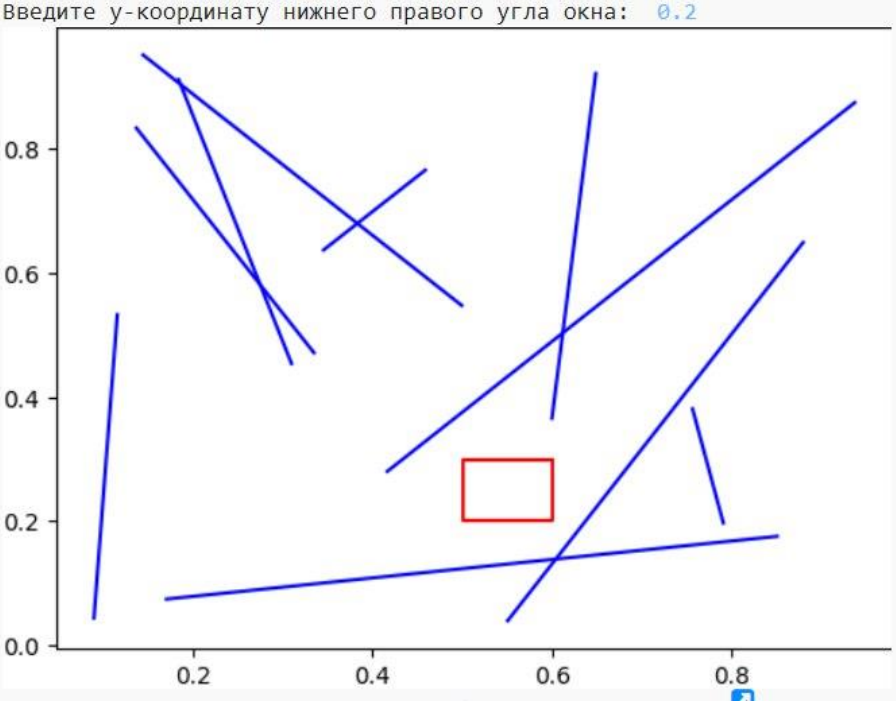


Рисунок 3 – Результат эксперимента

```
Выберите один из следующих вариантов: 1 -- ввести координаты окна в ручную;     2 --
оставить координаты окна по умолчанию:  1
Введите x-координату верхнего левого угла окна:  0.3
Введите y-координату верхнего левого угла окна:  0.6
Введите x-координату нижнего правого угла окна:  0.4
Введите y-координату нижнего правого угла окна:  0.9
```

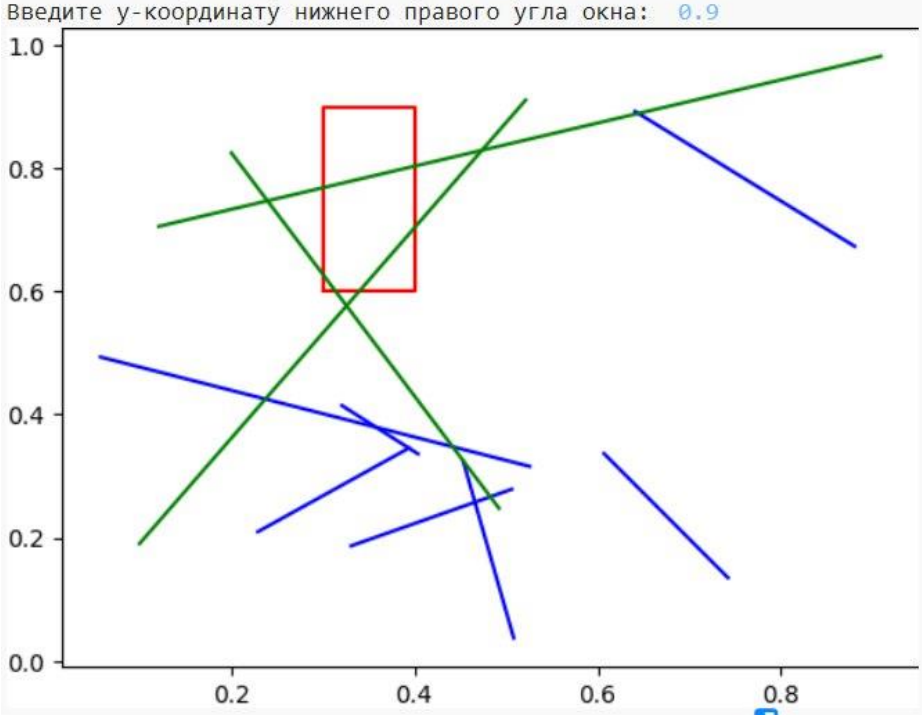


Рисунок 4 – Результат эксперимента

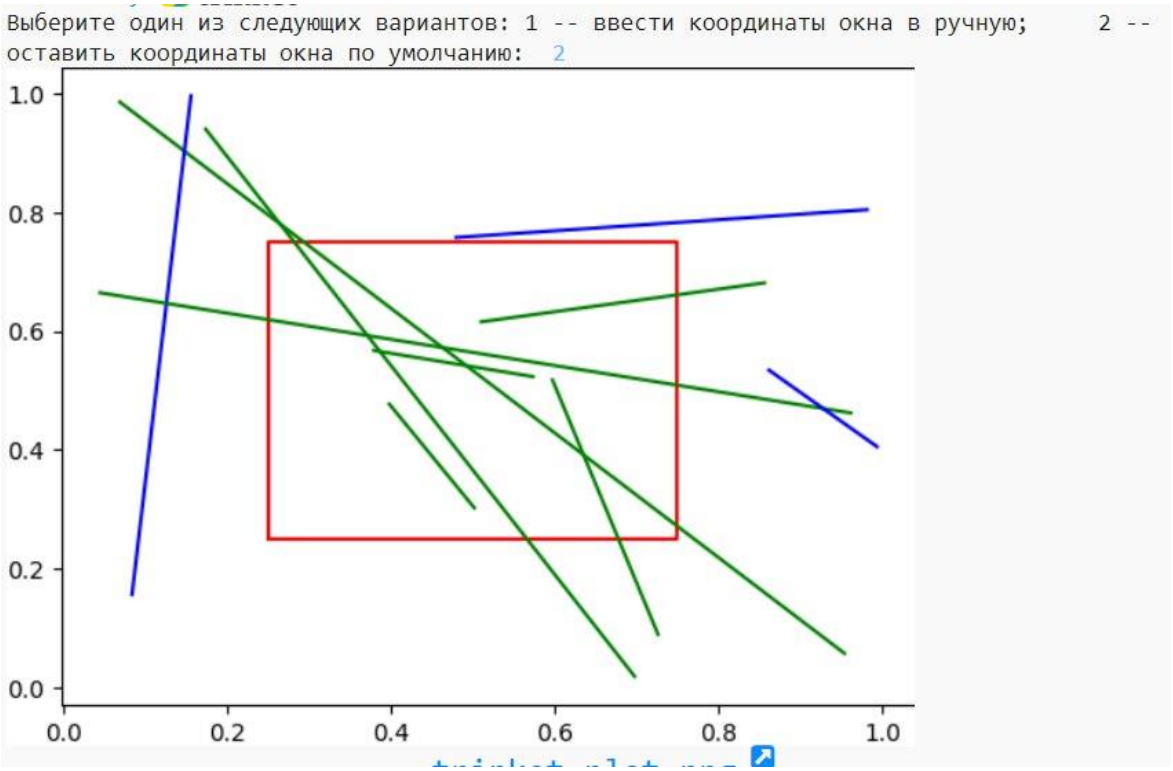


Рисунок 5 – Результат эксперимента

Исходный код программы

```
import matplotlib.pyplot as plt
import numpy as np
import time
#генерация отрезков
np.random.seed(int(time.time()))
num_segments = 10
segments = np.random.rand (num_segments, 2, 2)
# Определение пространственных точек
select = int (input ("Выберите один из следующих вариантов: 1 -- ввести координаты окна в ручную; 2 -- оставить координаты окна по умолчанию: "))
if (select == 1):
 # Задаем прямоугольное окно
  x1 = float(input("Введите x-координату верхнего левого угла окна: "))
```

```
      y1 = float(input("Введите y-координату верхнего
левого угла окна: "))
      x2 = float(input("Введите x-координату нижнего
правого угла окна: "))
      y2 = float(input("Введите y-координату нижнего
правого угла окна: "))
      window = np.array([[x1, y1], [x2, y2]])
    else :
      window = np.array([[0.25, 0.25],[0.75, 0.75]])

    #бинарный поиск
    intersections = []
    for segment in segments:
        if ((segment[:, 0] >= window[0, 0]).any() and
(segment[:, 0] <= window[1, 0]).any() and
            (segment[:, 1] >= window[0, 1]).any() and
(segment[:, 1] <= window[1, 1]).any()):
          intersections.append(True)
        else:
          intersections.append(False)


    #визулизация
    plt.figure()
    plt.plot([window[0,0],window[1,0],window[1,0],windo
w[0,0],window[0,0]],

[window[0,1],window[0,1],window[1,1],window[1,1],window
[0,1]], 'r-')
    for i, segment in enumerate (segments):
```

```
        if intersections[i]:
          plt.plot(segment[:,0], segment[:,1], 'g-')
        else :
          plt.plot(segment[:,0], segment[:,1], 'b-')
    plt.show()
```

Выводы.

В результате выполнения работы нами был реализован код алгоритма отсечения массива произвольных отрезков заданным прямоугольным окном на языке Python.