|  |  |
| --- | --- |
| 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** Изучение и программная реализация простого алгоритма отсечения отрезка  **Студент** Якуба Д. В.  **Группа** ИУ7-43  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель** Куров А. В. |  |

Москва

2020 г.

Оглавление

[Цель работы 3](#_Toc40529092)

[Техническое задание 3](#_Toc40529093)

[Теоретическая часть 3](#_Toc40529094)

[Алгоритм построчного затравочного заполнения 3](#_Toc40529095)

[Практическая часть 3](#_Toc40529096)

[Программная реализация алгоритма на ЯП Python 3](#_Toc40529097)

[Пользовательский интерфейс 3](#_Toc40529098)

[Демонстрация работы алгоритма 3](#_Toc40529099)

[Исследование временных характеристик 3](#_Toc40529100)

# Цель работы

Изучение и программная реализация алгоритма отсечения отрезка.

# Техническое задание

Алгоритм отсечения отрезка, согласно варианту 28 – Простой алгоритм отсечения.

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 осуществляется с помощью мыши и нажатия других клавиш.

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

# Теоретическая часть

## Отсечение

Отсечение – операция удаления части изображения, находящейся за пределами выделенной области, называемой отсекателем.

Отсечение используется в алгоритмах удаления невидимых линий поверхностей и в некоторых алгоритмах учёта теней, поэтому от нас не только требуется скопировать или сохранить изображение, расположенное в пределах отсекателя, но и решить геометрическую задачу нахождения геометрических характеристик изображения, чтобы использовать их при решении других задач.

Также существует понятие стирания изображения – это операция удаления части изображения, находящейся в пределах заданной области (то есть отсекателя).

Само по себе отсечение может проводиться в двумерном или трёхмерном пространствах. При этом, трёхмерный случай является обобщением двумерного случая. То есть, умея решать задачу в двумерном пространстве, не составит труда реализовать и трёхмерный случай.

Существует следующая классификация двумерных отсекателей:

Регулярный (стандартный) отсекатель – это отсекатель прямоугольной формы со сторонами, параллельными координатным осям.

Нерегулярный отсекатель – отсекатель формы произвольного выпуклого многоугольника.

Более сложные отсекатели – отсекатели формы произвольного невыпуклого многоугольника.

Также следует определить классификацию трёхмерных отсекателей:

Отсекатели формы прямоугольного параллелепипеда

Отсекатели формы четырёхгранной усечённой пирамиды.

Также следует отметить, что границу отсекателя принято относить к внутренней области отсечения.

## Отсечение отрезков регулярным отсекателем

Перед тем, как обратиться к простому алгоритму отсечения отрезка регулярным отсекателем, нам потребуется решить задачу определения отношения заданного отрезка к области отсечения.

На следующей картинке определены три отрезка:

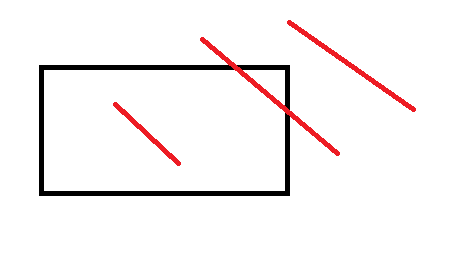


Рисунок , виды отрезков

Каждый из отрезков относительно отсекаемой области будет являться: полностью видимым (крайний левый отрезок), частично видимым (средний отрезок) или полностью невидимым (крайний правый отрезок).

Как видно из рисунка, для того чтобы определить принадлежность каждой из вершин отрезка отсекаемой области, достаточно проверить условие:

Каждый раз при обработке очередного отрезка проверять выполнения данных неравенств не очень удобно, поэтому было предложено формализовать данную процедуру и характеризовать положение точки четырёхразрядным кодом.

Каждый разряд в таком коде определяет положение точки относительно каждой из границ отсекателя. Обозначим такой код буквой .

В таком коде для каждой точки будем иметь:

*Стоит отметить, что в координатах экрана в условиях для знаки строгого неравенства поменяются между собой местами.*

Таким образом, точка будет видима в том случае, если поразрядная сумма её кода будет равна нулю.

Получается, что и отрезок будет полностью видим в том случае, если поразрядные суммы кодов концов этого отрезка равны нулю: .

Если отрезок не является полностью видимым, то он может являться либо частично видимым, либо полностью невидимым.

Возникает необходимость вновь «упростить» задачу и найти способ определить полную невидимость отрезка относительно заданного регулярного отсекателя.

Ниже приведены два вида полностью невидимых отрезков:

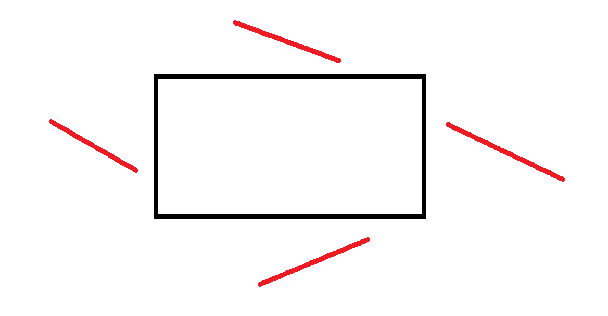


Рисунок , первый вид полностью невидимых отрезков

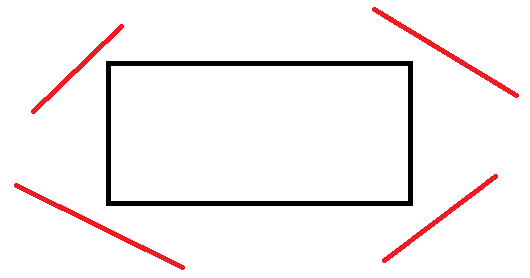


Рисунок , второй вид полностью невидимых отрезков

Сразу стоит обговорить тот факт, что для вида полностью невидимых отрезков, предоставленного на Рисунке 3, с помощью простого анализа кодов концов отрезков определить полную невидимость не получается.

Для определения полной невидимости отрезков, которые располагаются относительно отсекателя вполне определённым способом (Рисунок 2: полностью ниже нижней границы отсекателя, полностью левее левой границы, полностью правее правой границы, полностью выше верхней границы), достаточно провести следующий простой тест:

*, где & - поразрядное логические «и», а - код первого конца отрезка и - код второго конца отрезка.*

При этом мы также имеем возможность определить частичную видимость следующим простым тестом:

*Для таких отрезков от нас потребуется найти вторую вершину видимой части.*

## Простой алгоритм

# Практическая часть

## Программная реализация алгоритма на ЯП Python

## Пользовательский интерфейс

## Демонстрация работы алгоритма

# Исследование временных характеристик