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 Блохин Д.М.

Группа ИУ7-43Б

Оценка (баллы) _____

Преподаватель Куров А.В.

Москва.
2020 г.

Тема: Реализация алгоритма отсечения отрезка регулярным отсекателем.

Цель работы: Изучение и программная реализация алгоритма отсечения отрезка. 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.в

Ввод осуществлять с помощью мыши и нажатия других клавиш.

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

Мой вариант 3: Алгоритм разбиения отрезка средней точкой

Теоретическое описание алгоритма

1. Ввод координат отсекателя $X_л$, $X_п$, $Y_н$, $Y_в$.
2. Ввод координат концов отрезка $P_1(X_1,Y_1)$, $P_2(X_2,Y_2)$.
1. Ввод точности ε вычисления точки пересечения отрезка с границей отсекателя.
2. Установка номера шага отсечения i=1.
5. Вычисление кодов концевых точек и запись их в соответствующие массивы T1 и T2 размерностью 1x 4, вычисление сумм кодов концов $S_1=$, $S_2=$ =
6. Проверка полной видимости отрезка. Если коды обоих концов отрезка равны нулю (полная видимость отрезка), то переход к п. 9.

7. Проверка полной невидимости отрезка. Вычисление побитного логического произведения кодов концевых точек отрезка. Если произведение отлично от нуля (отрезок невидим), то переход к п. 10.
8. Анализ частично видимого отрезка в том случае, если побитовое логическое произведение кодов его концов равно нулю:
8.1. Поиск наиболее удаленной от P_1 видимой точки S исследуемого отрезка. Запоминание исходной точки P_1 в промежуточной переменной R.
8.2. Проверка на окончание процесса решения: если i>2, то определение логического произведения pr кодов концов отрезка. Если pr≠0, то переход к п.10, иначе переход к п.9.

8.3. Проверка точки P_2 на наиболее удаленную от P_1 видимую точку отрезка. Если сумма всех элементов массива T2 равна нулю (S_2), то переход к пункту 8.12.
8.4. Проверка нахождения точки пересечения отрезка с границами отсекателя. Если $|P_1 - P_2| \le \varepsilon$ (расстояние между концевыми точками исследуемого отрезка меньше допустимой погрешности), то переход к пункту 8.12.
8.5. Вычисление средней точки $P_{ср.}$ отрезка: $P_{ср.} = (P_1 + P_2)/2$ ($P_{ср.x} = (P_{1.x} + P_{2.x})/2$; $P_{ср.y} = (P_{1.y} + P_{2.y})/2$).
8.6. Запоминание текущей точки P_1: Pm=P1.
8.7. Замена точки P_1 на среднюю точку: $P_1 = P_{ср}$.
8.8. Вычисление нового кода T1 точки P_1.
8.9. Вычисление произведения pr кодов концов нового отрезка P_1P_2.
8.10.Проверка полной невидимости отрезка P_1P_2. Если побитовое логическое произведение pr кодов концевых точек равно нулю, то переход к пункту 8.4. В противном случае отрезок P_1P_2 невидим.
8.11. Возврат к предыдущему отрезку P_1P_2 : $P_1 = Pm$, $= P_{ср}$, переход к пункту 8.4. (Вычислена наиболее удаленная от точки P_1 видимая точка отрезка).
8.12..Поиск наиболее удаленной от P_2 видимой точки отрезка. Замена мест точек P_1 и P_2 : $P_1 = P_2$, P_2=R. Увеличение шага выполнения отсечения i=i+1. Переход к п.5.
9. Визуализация отрезка.
10.Конец.

Реализация алгоритма в коде

```
void MainWindow::on_cut_button_clicked()
{
    painter->setRenderHint(QPainter::Antialiasing, true);

    // если на холсте присутствуют и отсекатель и отрезки
    if (flag_line_exist && flag_rect_set)
    {
        for (int j = 0; j < lines.size(); j += 2)
        {
            int i = 1; // шаг отсечения

            QPoint P1 = lines[j];
            QPoint P2 = lines[j + 1];

            qDebug() << P1 << P2;

            int T1[NUM_BITS];
            int T2[NUM_BITS];

            int S1, S2;
            while (1)
            {
                set_bits(rect, P1, T1);
                set_bits(rect, P2, T2);

                S1 = get_sum(T1, NUM_BITS);
                S2 = get_sum(T2, NUM_BITS);

                if (S1 == 0 && S2 == 0)
                {
                    qDebug() << "Отрезок тривиально видим.";
                    painter->setPen(QPen(outline_color, 2)); // цвет
```

```
    painter->drawLine(P1.x(), P1.y(), P2.x(), P2.y());
    ui->draw_label->setPixmap(*scene);
    break;
}

int P = logic_mult(T1, T2, NUM_BITS); //

QPoint R;
QPoint tmp;

if (P == 0)
{
    if (i > 2)
    {
        qDebug() << P1 << P2;

        painter->setPen(QPen(outline_color, 2));

        painter->drawLine(P1.x(), P1.y(), P2.x(), P2.y());
        ui->draw_label->setPixmap(*scene);

        break;
    }
    R = P1; // запоминаем первую концевую точку
    if (S2 == 0)
    {
        P1 = P2;
        P2 = R;
        i++;
    }
    else
    {

        while (abs(P1.x() - P2.x()) > ACCURACY || abs(P1.y() - P2.y()) > ACCURACY)
        {
            QPoint Pm;

            Pm.setX((P1.x() + P2.x()) >> 1);
            Pm.setY((P1.y() + P2.y()) >> 1);

            tmp = P1;
            P1 = Pm;

            set_bits(rect, P1, T1);

            int pr = logic_mult(T1, T2, NUM_BITS);

            if (pr != 0)
            {
                P1 = tmp;
                P2 = Pm;
            }
        }
        P1 = P2;
        P2 = R;
        i++;
    }
}

else
{
    qDebug() << "Отрезок тривиально невидим.";
```

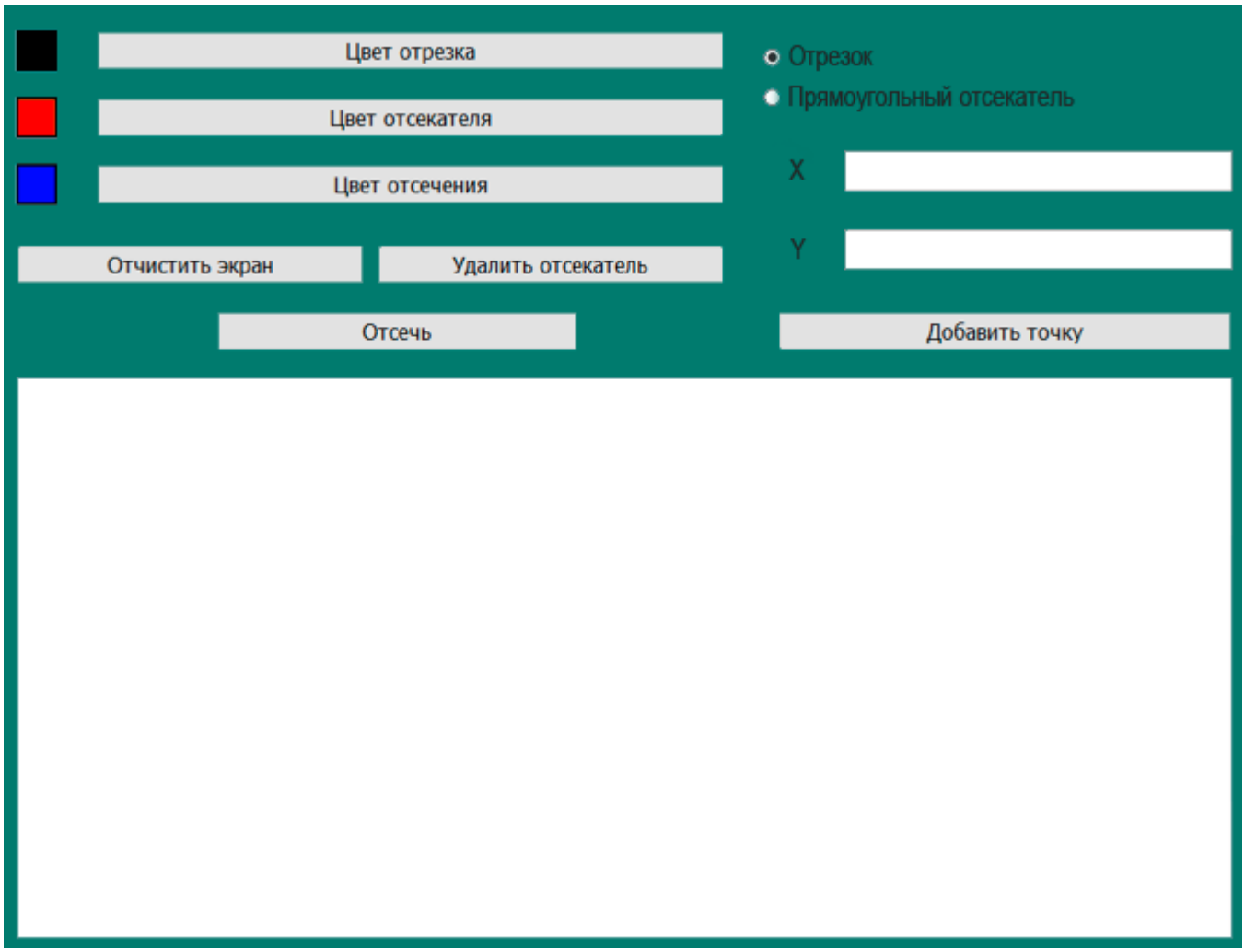
```
                break;
            }
        }
    }
}
else
{
    if (!flag_rect_set && flag_line_exist)
    {
        message_box(QString("Вы не ввели отсекатель!"));
    }
    else if (!flag_line_exist && flag_rect_set)
    {
        message_box(QString("Вы не нарисовали линии!"));
    }
    else
    {
        message_box(QString("Вы не нарисовали линии и не ввели отсекатель!"));
    }

}
}
```

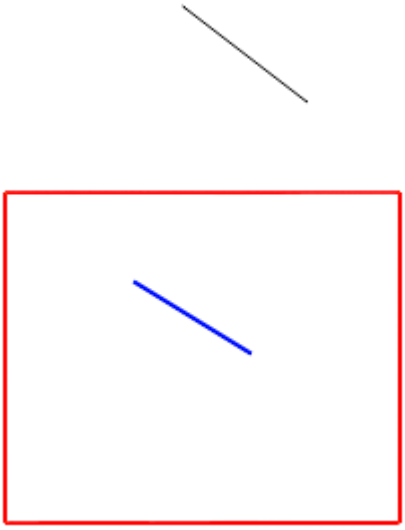
Интерфейс и демонстрация работы

Предоставлен выбор цвета отрезка, отсекателя, отсечения. Также доступен ввод вершин посредством координат или мышки. Для упрощенного построения горизонтальных и вертикальных линии используется горячая клавиша ctrl. Присутствует возможность очистки экрана, удаление отсекателя.

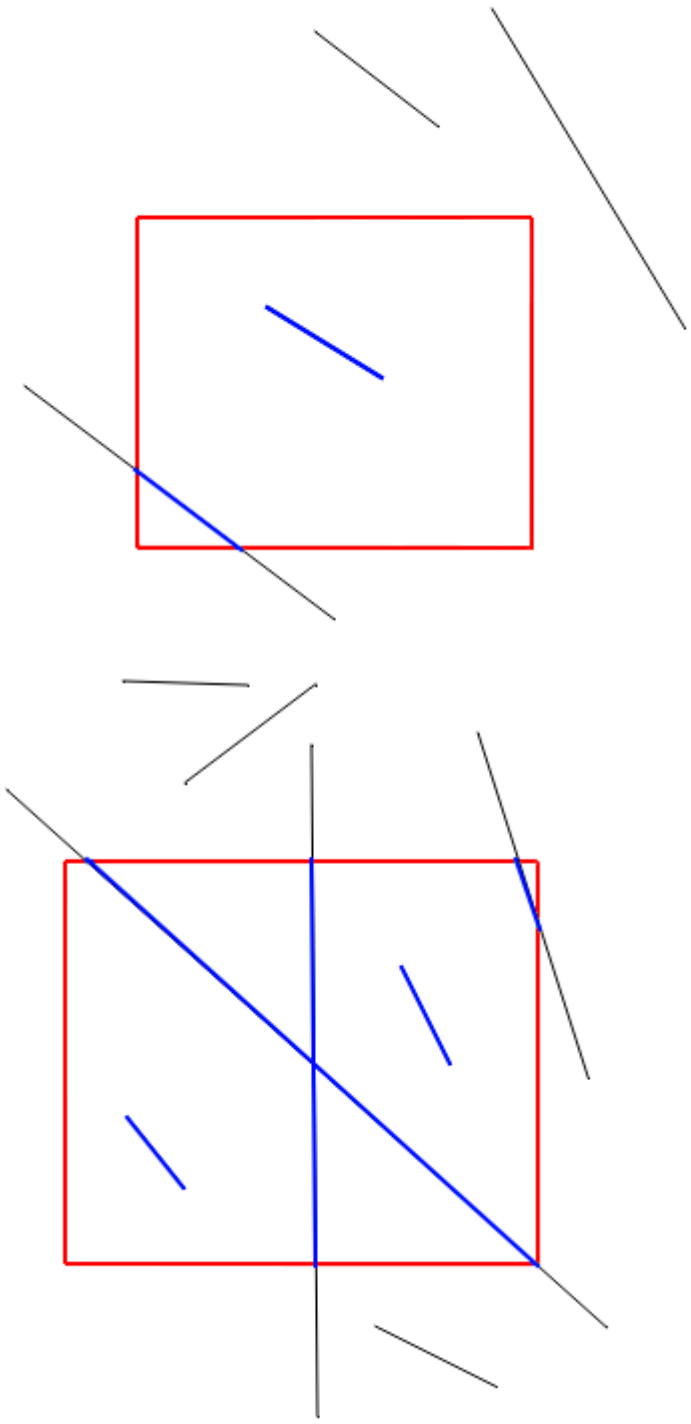
При вводе с помощью мышки отсекатель задается 2 углами: левый нижний (для компьютера, для нас – верхний) и правый верхний (для нас – нижний).



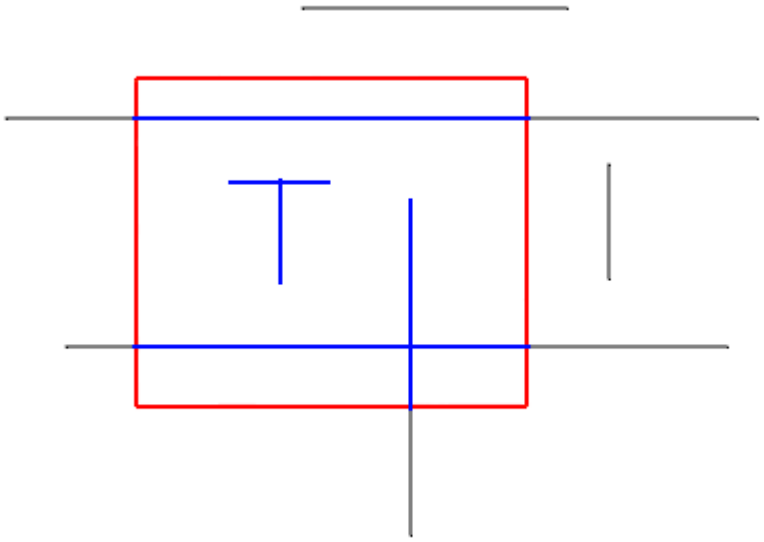
Тривиальные случаи



Нетривиальные случаи



Случай с горизонтальными и вертикальными линиями



Случаи с прохождением вдоль границы отсекателя