|  |  |
| --- | --- |
| **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 \_ Реализация алгоритма отсечения отрезка регулярным отсекателем. \_**  **Студент \_Чаушев Александър Красимиров\_\_\_\_\_\_\_\_**  **Группа \_ИУ7-46Б\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_Куров А. В.\_\_\_\_** |  |

Москва.

2020 г.

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

Алгоритм разбиения отрезка средней точкой (Вариант 12)

**Входные данные:** Координаты вершин многоугольника, цвет .

**Выходные данные:** Пользовательское меню, содержащее поля ввода и конечннoe изображение. Вывод замера времени выполнения алгоритма( без задержки ).

**Ошибочные ситуации:** Программа прекращается, если хотя бы один из входных данных не корректен.

**Теоретическая часть:**

Если отрезок частично видимый, разбиваем его средней точкой на два отрезка. Если средняя точка вне окна, то одна из половин невидима, отбрасываем ее, а алгоритм деления применяем к другой половине. Если одна из половин видна – вывод. Если же каждая из половин является частично видимой, то алгоритм применяем к каждой из половин. Деление прекращается, когда длина отрезка равна одному пикселю. Получим либо две таких точки на сторонах окна, которые соединяем, либо одну точку вне окна.

Алгоритм:

1.Ввод координат отсекателя Xл, Xп, Yн, Yв.

2.Ввод координат концов отрезка P1(X1,Y1), P2(X 2,Y 2).

3.Ввод точности ε вычисления точки пересечения отрезка с границей отсекателя.

4.Установка номера шага отсечения i=1.

5.Вычисление кодов концевых точек и запись их в соответствующие массивы T1 и T2 размерностью 1х 4, вычисление сумм кодов концов S1 S2

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 P1 видимой точки S исследуемого отрезка. Запоминание исходной точки P1 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 P2 на наиболее удаленную от P1 видимую точку отрезка. Если сумма всех элементов массива T2 равна нулю (S2), то переход к пункту 8.12.

8.4. Проверка нахождения точки пересечения отрезка с границами отсекателя. Если  P1- P2≤ε (расстояние между концевыми точками исследуемого отрезка меньше допустимой погрешности), то переход к пункту 8.12.

8.5. Вычисление средней точки Pср. отрезка: Pср. = (P1 + P2 )/2 (Pср.x = (P1.x + P2..x )/2 ; Pср.y = (P1.y + P2.y )/2).

8.6. Запоминание текущей точки P1: Pm=P1.

8.7. Замена точки P1 на среднюю точку: P1= Pср .

8.8. Вычисление нового кода T1 точки P1.

8.9. Вычисление произведения pr кодов концов нового отрезка P1P2.

8.10.Проверка полной невидимости отрезка P1P2. Если побитовое логическое произведение pr кодов концевых точек равно нулю, то переход к пункту 8.4. В противном случае отрезок P1P2 невидим.

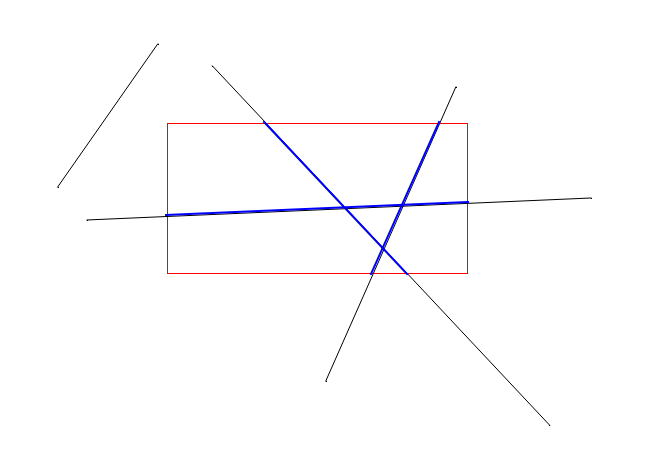
8.11. Возврат к предыдущему отрезку P1P2 : P1 = Pm, = Pср , переход к пункту 8.4. ( Вычислена наиболее удаленная от точки P1 видимая точка отрезка).

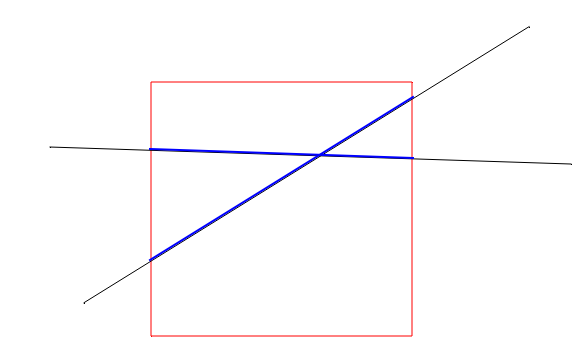
8.12..Поиск наиболее удаленной от P2 видимой точки отрезка. Замена мест точек P1 и P2 : P1 = P2 , P2=R. Увеличение шага выполнения отсечения i=i+1. Переход к п.5.

9.Визуализация отрезка.

10.Конец. При программной реализации алгоритма целесообразно для вычисления кодов концевых точек отрезков и их логических произведений использовать отдельные функции.

**Пример работы:**





**Код программы**

void MainWindow::on\_cut\_button\_clicked()  
{  
 painter->setRenderHint(QPainter::Antialiasing, true);  
 if (flag\_line\_exist && flag\_rect\_set)  
 {  
 for (int j = 0; j < lines.size(); j += 2)  
 {  
 // lines[i] = P1 and lines[i + 1] = P2 // P1, P2 - points  
 int i = 1; // шаг отсечения  
 QPoint P1 = lines[j];  
 QPoint P2 = lines[j + 1];  
 qDebug() << P1 << P2;  
 int T1[BITS\_COUNT];  
 int T2[BITS\_COUNT];  
 int S1, S2; // Summ of BITS T1 and BITS T2  
 while (1)  
 {  
 set\_bits(rect, P1, T1);  
 set\_bits(rect, P2, T2);  
  
 S1 = get\_sum(T1, BITS\_COUNT);  
 S2 = get\_sum(T2, BITS\_COUNT);  
 QPoint R;  
 QPoint temp\_memory;  
 if (S1 == 0 && S2 == 0)  
 {  
 qDebug() << "Отрезок полностью видим";  
 painter->setPen(QPen(outline\_color, 2));  
 painter->drawLine(P1.x(), P1.y(), P2.x(), P2.y());  
 //painter->drawLine(round(P1.x()), round(P1.y()), round(P2.x()), round(P2.y()));  
 ui->draw\_label->setPixmap(\*scene);  
 break;  
 }  
 int P = get\_p(T1, T2, BITS\_COUNT);  
 if (P != 0)  
 {  
 qDebug() << "Отрезок тривиально невидим";  
 break;  
 }  
 R = P1;  
  
 if (i > 2)  
 {  
 int Pr = get\_p(T1, T2, BITS\_COUNT);  
 if (Pr == 0)  
 {  
 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;  
 }  
  
 if (S2 == 0)  
 {  
 P1 = P2;  
 P2 = R;  
 i++;  
 }  
 else  
 {  
 while (abs(P1.x() - P2.x()) > EPS || abs(P1.y() - P2.y()) > EPS)  
 {  
  
 QPoint Pm;  
 Pm.setX((P1.x() + P2.x()) >> 1);  
 Pm.setY((P1.y() + P2.y()) >> 1);  
 temp\_memory = P1;  
 P1 = Pm;  
 set\_bits(rect, P1, T1);  
 int pr = get\_p(T1, T2, BITS\_COUNT);  
 if (pr != 0)  
 {  
 P1 = temp\_memory;  
 P2 = Pm;  
 }  
 }  
 P1 = P2;  
 P2 = R;  
 i++;  
 }  
 }  
 }  
 }  
 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!"));  
 }  
  
 }  
}