**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»**

**Вариант 1**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 9301 |  | Примакова Е.Е. |
|  |  | Русанова К.В. |
| Преподаватель |  | Матвеева И.В. |

Санкт-Петербург

2022

**Цель работы**

Обеспечить реализацию алгоритма отсечения массива произвольных отрезков заданным прямоугольным окном с использование определенного алгоритма.

**Задание**

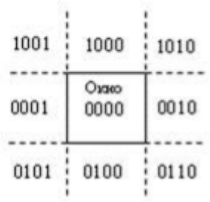
Обеспечить реализацию простого алгоритма отсечения массива произвольных отрезков следует формировать генератором случайных чисел. Вначале следует вывести на экран сгенерированные отрезки полностью, а затем другим цветом или яркостью те, которые полностью или частично попадают в область окна.

**Математическая модель**

Суть алгоритма 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:

|  |  |  |
| --- | --- | --- |
| 1001 | 0001 | 0101 |
| 1000 | 0000 | 0100 |
| 1010 | 0010 | 0110 |

Схема расположения кодов



Поясним эти коды:

b0 =0, если

b0 =1, если

b1 =0, если

b1 =1, если

b2 =0, если

b2 =1, если

b3 =0, если

b3 =1, если

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.

**Контрольный пример**

Контрольные примеры представлены на рисунках 1, 2 и 3.

По нажатию на кнопку «Generate lines» программа генерирует случайным образом линии черного цвета, потом пользователь выделяет прямоугольную область на экране и все линии, попавшие полностью или частично в эту область, становятся красными.

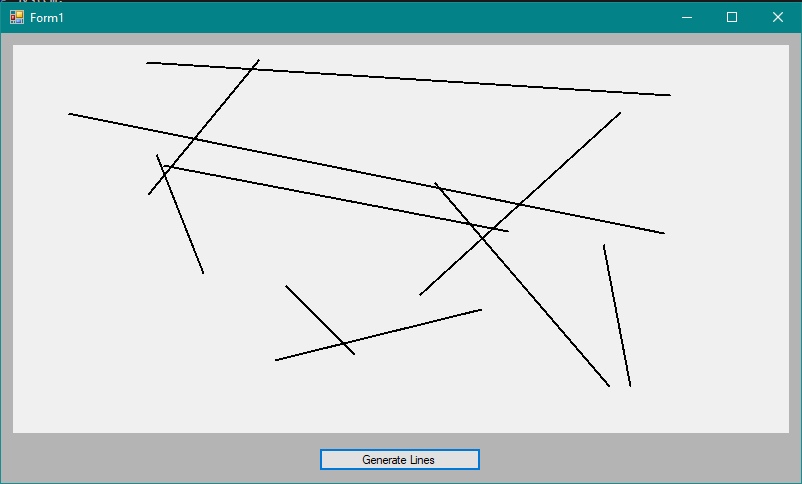


Рис. 1. Генерация линий

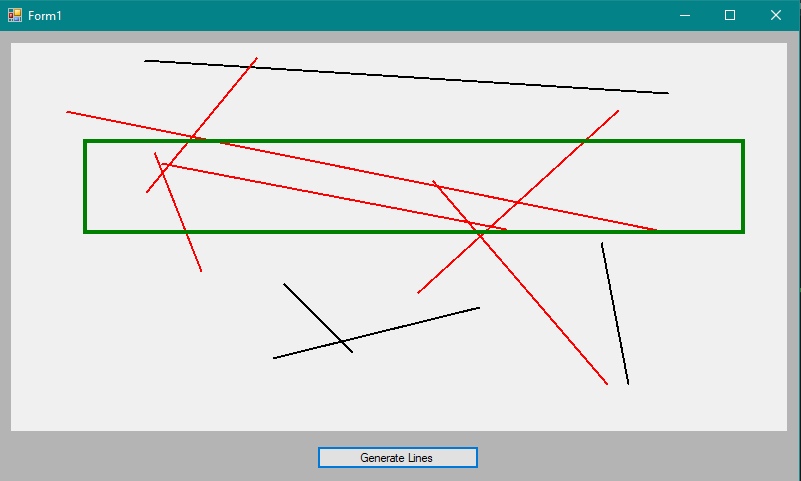


Рис. 2. Выделение области и закраска линий

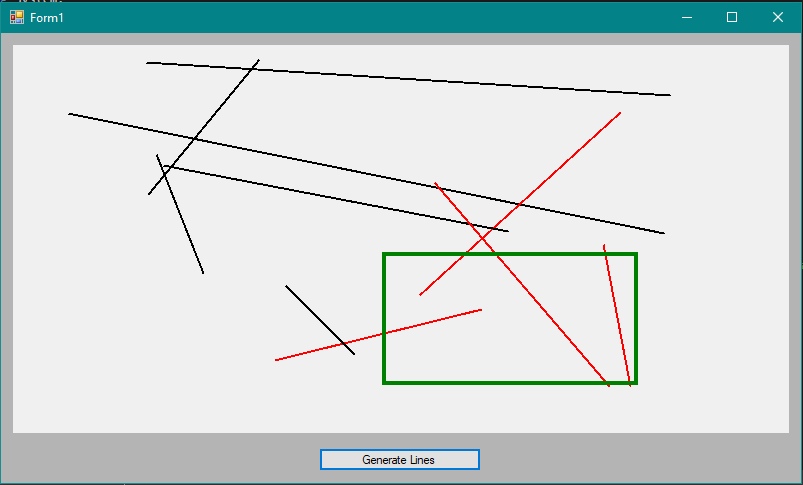


Рис. 3. Выделение области и закраска линий

На рис. 1 показана генерация линий, на рис. 2 и 3 два варианта выделения прямоугольной области и линий, попавших в нее.

**Код программы**

**Файл Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace lab4

{

public partial class Form1 : Form

{

bool isClicked = false;

Line rectangleLine = new Line(new Point(0, 0), new Point(0,0));

List<Line> lines = new List<Line>();

MyRectangle rectangle;

public Form1()

{

InitializeComponent();

}

private void genLinesButton\_Click(object sender, EventArgs e)

{

lines.Clear();

pictureBox.Image = new Bitmap(pictureBox.Width, pictureBox.Height);

Graphics g = Graphics.FromImage(pictureBox.Image);

Random rnd = new Random();

for(int i = 0; i < 10; i++)

{

int x1 = rnd.Next(0, pictureBox.Width);

int y1 = rnd.Next(0, pictureBox.Height);

int x2 = rnd.Next(0, pictureBox.Width);

int y2 = rnd.Next(0,pictureBox.Height);

lines.Add(new Line(new Point(x1, y1), new Point(x2, y2)));

}

for(int i = 0; i < lines.Count; i++)

{

g.DrawLine(new Pen(Brushes.Black, 2), lines.ElementAt(i).P1.X,

lines.ElementAt(i).P1.Y, lines.ElementAt(i).P2.X,

lines.ElementAt(i).P2.Y);

}

}

private void pictureBox\_MouseDown(object sender, MouseEventArgs e)

{

isClicked = true;

rectangleLine.P1.X = e.X;

rectangleLine.P1.Y = e.Y;

}

private void pictureBox\_MouseUp(object sender, MouseEventArgs e)

{

isClicked = false;

pictureBox.Image = new Bitmap(pictureBox.Width, pictureBox.Height);

Graphics g = Graphics.FromImage(pictureBox.Image);

CohenSutherland cohenSutherland = new CohenSutherland();

for (int i = 0; i < lines.Count; i++)

{

int check = cohenSutherland.Cohen\_Sutherland(rectangle, lines.ElementAt(i).P1,

lines.ElementAt(i).P2);

if (check == 0)

{

g.DrawLine(new Pen(Brushes.Red, 2), lines.ElementAt(i).P1.X,

lines.ElementAt(i).P1.Y, lines.ElementAt(i).P2.X, lines.ElementAt(i).P2.Y);

}

else

{

g.DrawLine(new Pen(Brushes.Black, 2), lines.ElementAt(i).P1.X,

lines.ElementAt(i).P1.Y, lines.ElementAt(i).P2.X, lines.ElementAt(i).P2.Y);

}

}

}

private void pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

if (isClicked)

{

rectangleLine.P2.X = e.X;

rectangleLine.P2.Y = e.Y;

pictureBox.Invalidate();

}

}

private void pictureBox\_Paint(object sender, PaintEventArgs e)

{

rectangle = new MyRectangle();

Pen pen = new Pen(Color.Black);

if(rectangleLine.P1.X < rectangleLine.P2.X && rectangleLine.P1.Y > rectangleLine.P2.Y)

{

e.Graphics.DrawRectangle(new Pen(Brushes.Green, 4), rectangleLine.P1.X,

rectangleLine.P2.Y, Math.Abs(rectangleLine.P2.X - rectangleLine.P1.X),

Math.Abs(rectangleLine.P2.Y - rectangleLine.P1.Y));

Initialize\_Rectangle(rectangleLine.P1.X, rectangleLine.P2.X,

rectangleLine.P2.Y, rectangleLine.P1.Y, rectangle);

}

if(rectangleLine.P1.Y < rectangleLine.P2.Y && rectangleLine.P1.X < rectangleLine.P2.X)

{

e.Graphics.DrawRectangle(new Pen(Brushes.Green, 4), rectangleLine.P1.X,

rectangleLine.P1.Y, Math.Abs(rectangleLine.P2.X - rectangleLine.P1.X),

Math.Abs(rectangleLine.P2.Y - rectangleLine.P1.Y));

Initialize\_Rectangle(rectangleLine.P1.X, rectangleLine.P2.X,

rectangleLine.P1.Y, rectangleLine.P2.Y, rectangle);

}

if(rectangleLine.P1.Y < rectangleLine.P2.Y && rectangleLine.P1.X > rectangleLine.P2.X)

{

e.Graphics.DrawRectangle(new Pen(Brushes.Green, 4), rectangleLine.P2.X,

rectangleLine.P1.Y, Math.Abs(rectangleLine.P2.X - rectangleLine.P1.X),

Math.Abs(rectangleLine.P2.Y - rectangleLine.P1.Y));

Initialize\_Rectangle(rectangleLine.P2.X, rectangleLine.P1.X,

rectangleLine.P1.Y, rectangleLine.P2.Y, rectangle);

}

if (rectangleLine.P1.Y > rectangleLine.P2.Y && rectangleLine.P1.X > rectangleLine.P2.X)

{

e.Graphics.DrawRectangle(new Pen(Brushes.Green, 4), rectangleLine.P2.X,

rectangleLine.P2.Y, Math.Abs(rectangleLine.P2.X - rectangleLine.P1.X),

Math.Abs(rectangleLine.P2.Y - rectangleLine.P1.Y));

Initialize\_Rectangle(rectangleLine.P2.X, rectangleLine.P1.X,

rectangleLine.P2.Y, rectangleLine.P1.Y, rectangle);

}

}

private void Initialize\_Rectangle(int x\_min, int x\_max, int y\_min, int y\_max, MyRectangle rectangle)

{

rectangle.X\_min = x\_min; rectangle.X\_max = x\_max;

rectangle.Y\_min = y\_min; rectangle.Y\_max = y\_max;

}

}

}

**Файл CohenSutherland.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CompGrapf\_lab4

{

class CohenSutherland

{

private const byte INSIDE = 0; // 0000

private const byte LEFT = 1; // 0001

private const byte RIGHT = 2; // 0010

private const byte BOTTOM = 4; // 0100

private const byte TOP = 8; // 1000

// возвращает бит-код точки

public byte Vcode(MyRectangle rect, Point point)

{

return (byte)(((point.X < rect.X\_min) ? LEFT : 0) + // +1 если точка левее прямоугольника

((point.X > rect.X\_max) ? RIGHT : 0) + // +2 если точка правее прямоугольника

((point.Y < rect.Y\_min) ? BOTTOM : 0) + // +4 если точка ниже прямоугольника

((point.Y > rect.Y\_max) ? TOP : 0)); // +8 если точка выше прямоугольника

}

// если отрезок ab не пересекает прямоугольник, функция возвращает -1;

// иначе возвращает 0;

public int Cohen\_Sutherland(MyRectangle rect, Point a, Point b)

{

int code\_a, code\_b, code; // код концов отрезка

Point c;

Point aa = new Point(a.X, a.Y); // альтернативные точки, чтобы не менять координаты настоящих точек

Point bb = new Point(b.X, b.Y);

code\_a = Vcode(rect, aa);

code\_b = Vcode(rect, bb);

//пока одна из точек отрезка вне прямоугольника

while ((code\_a | code\_b) != 0)

{

//если обе точки с одной стороны прямоугольника, то отрезок не пересекает прямоугольник

if ((code\_a & code\_b) != 0)

return -1;

// выбираем точку c с ненулевым кодом

if (code\_a != 0)

{

code = code\_a;

c = aa;

}

else

{

code = code\_b;

c = bb;

}

if ((code & LEFT) != 0)

{

c.Y += (aa.Y - bb.Y) \* (rect.X\_min - c.X) / (aa.X - bb.X);

c.X = rect.X\_min;

}

else if ((code & RIGHT) != 0)

{

c.Y += (aa.Y - bb.Y) \* (rect.X\_max - c.X) / (aa.X - bb.X);

c.X = rect.X\_max;

}

else if ((code & BOTTOM) != 0)

{

c.X += (aa.X - bb.X) \* (rect.Y\_min - c.Y) / (aa.Y - bb.Y);

c.Y = rect.Y\_min;

}

else if ((code & TOP) != 0)

{

c.X += (aa.X - b.X) \* (rect.Y\_max - c.Y) / (aa.Y - bb.Y);

c.Y = rect.Y\_max;

}

// обновляем код точки

if (code == code\_a)

{

aa = c;

code\_a = Vcode(rect, aa);

}

else

{

bb = c;

code\_b = Vcode(rect, bb);

}

}

// оба кода равны 0, следовательно обе точки в прямоугольнике

return 0;

}

}

}

**Вывод**

В процессе выполнения данной лабораторной работы был получен практический опыт реализации алгоритма отсечения массива произвольных отрезков заданным прямоугольным окном с использование алгоритма Коэна-Сазерленда.