**Министерство образования Республики Беларусь**

**Учреждение образования**

**«Белорусский государственный университет информатики   
и радиоэлектроники»**

Кафедра интеллектуальных

информационных технологий

**Отчёт**

по лабораторной работе №6

по курсу «Графический интерфейс интеллектуальных систем»

Выполнили:

студенты группы 821701

Бозбей М.С.

Михневич Е.Д.

Минск 2011

**Цель:**

Изучить алгоритмы отсечения в двумерном и трехмерном пространстве, реализовать алгоритм Коэна-Сазерленда. Разобрать графический редактор с возможностью отсечения невидимых частей линий пересекающих некоторую заданную фигуру.

**Алгоритм Коэна-Сазерленда:**

Целью алгоритмаявляется определение тех точек, отрезков или их частей, которые лежат внутри отсекаемого окна. Эти точки, отрезки или их части остаются для визуализации, а все остальное отбрасывается.

В реальных сценах приходится реально отсекать большое количество отрезков и точек, поэтому основное требование к алгоритмам отсечения – *эффективность.* Во многих случаях большое число точек или отрезков лежит целиком внутри или вне отсекаемого окна. Поэтому важно уметь быстро отбирать отрезки, подобные *ab* или точки, подобные *p,* и отбрасывать отрезки, подобные *ij* или точки, подобные *q.*Точки, лежащие внутри отсекаемого окна, удовлетворяют условию:

*xл* ≤ *x* ≤ *xп;*

*yн* ≤ *y* ≤ *yв.*

Знак равенства здесь показывает, что точки, лежащие на границе окна, считаются находящимися внутри него.

Отрезок лежит внутри окна, и, следовательно, является видимым, если обе его концевые точки лежат внутри окна, например отрезок *ab.* Однако, если оба конца отрезка лежат вне окна, то этот отрезок не обязательно лежит целиком вне окна, например отрезок *gh.* Если же оба конца отрезка лежат справа, слева, выше или ниже окна, то этот отрезок целиком лежит вне окна, а значит, невидим. Проверка последнего условия устраняет все отрезки, помеченные как *ij,* но не устраняет ни отрезка *gh,* видимого частично, ни отрезка *kl,* которые целиком невидим.

**Реализация:**

**Пример работы программы:**

var CohenSutherland = {

INSIDE : 0,

LEFT : 1,

RIGHT : 2,

BOTTOM : 4,

TOP : 8,

xmin : 0,

xmax : 0,

ymin : 0,

ymax : 0,

initClipRect : function(xmin, xmax, ymin, ymax) {

this.xmin = xmin < xmax ? xmin : xmax;

this.xmax = xmax > xmin ? xmax : xmin;

this.ymin = ymin < ymax ? ymin : ymax;

this.ymax = ymax > ymin ? ymax : ymin;

},

ComputeOutCode : function(x, y) {

var code = this.INSIDE;

if (x < this.xmin)

code |= this.LEFT;

else if (x > this.xmax)

code |= this.RIGHT;

if (y < this.ymin)

code |= this.BOTTOM;

else if (y > this.ymax)

code |= this.TOP;

return code;

},

CohenSutherlandLineClipAndDraw : function(x0, y0, x1, y1) {

var line = new BresenhamLine({

pointStart: new Point({x: x0, y: y0}),

pointEnd: new Point({x: x1, y: y1}),

color: 'red'

});

canvas.draw(line);

var outcode0 = this.ComputeOutCode(x0, y0);

var outcode1 = this.ComputeOutCode(x1, y1);

var accept = false;

while (true) {

if (!(outcode0 | outcode1)) {

accept = true;

break;

} else if (outcode0 & outcode1) {

break;

} else {

var x, y;

var outcodeOut = outcode0? outcode0 : outcode1;

if (outcodeOut & this.TOP) {

x = (x0 + (x1 - x0) \* (this.ymax - y0) / (y1 - y0));

y = this.ymax;

} else if (outcodeOut & this.BOTTOM) {

x = (x0 + (x1 - x0) \* (this.ymin - y0) / (y1 - y0));

y = this.ymin;

} else if (outcodeOut & this.RIGHT) {

y = (y0 + (y1 - y0) \* (this.xmax - x0) / (x1 - x0));

x = this.xmax;

} else if (outcodeOut & this.LEFT) {

y = (y0 + (y1 - y0) \* (this.xmin - x0) / (x1 - x0));

x = this.xmin;

}

if (outcodeOut == outcode0) {

x0 = x;

y0 = y;

outcode0 = this.ComputeOutCode(x0, y0);

} else {

x1 = x;

y1 = y;

outcode1 = this.ComputeOutCode(x1, y1);

}

}

}

if (accept) {

var line = new BresenhamLine({

pointStart: new Point({x: Math.round(x0), y: Math.round(y0)}),

pointEnd: new Point({x: Math.round(x1), y: Math.round(y1)}),

color: 'blue'

});

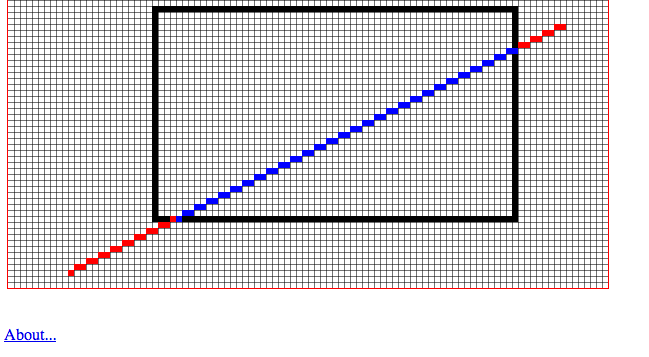
canvas.draw(line);

}

},

};

**Пример работы программы:**



**Выводы:**

В ходе лабораторной работы был изучен и реализован алгоритм Коэна-Сазерленда.