**Министерство образования Республики Беларусь**

**Учреждение образования**

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

Кафедра интеллектуальных

информационных технологий

**Отчёт**

по лабораторной работе № 2

по курсу «Графический интерфейс интеллектуальных систем»

Выполнили:

студенты группы 821701

Бозбей М.С.

Михневич Е.Д.

Минск 2011

Цель:

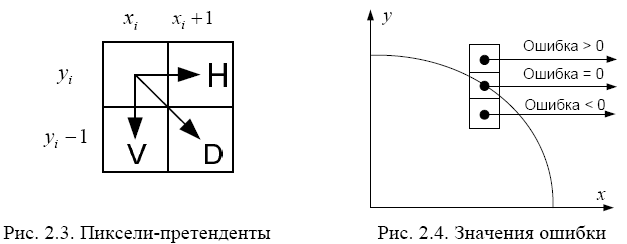
Изучить алгоритмы генерации кривых второго порядка. Разработать алгоритм генерации парабола на основе алгоритма Брезенхема. Разработать элементарный графический редактор, реализующий построение линий второго порядка: параболы, эллипс.

**Описание алгоритма построения окружности:**

Алгоритм Брезенхема начинает работать с точки (0, *R*) и рисует часть ок-

ружности по часовой стрелке, лежащую в первом квадранте. Предполагается,

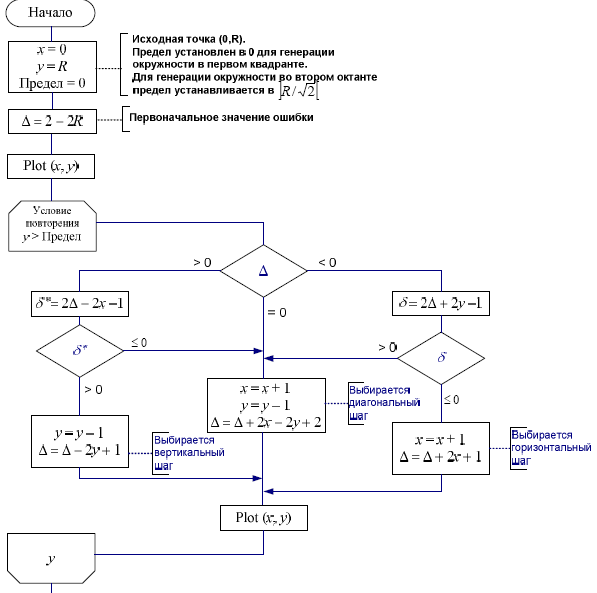
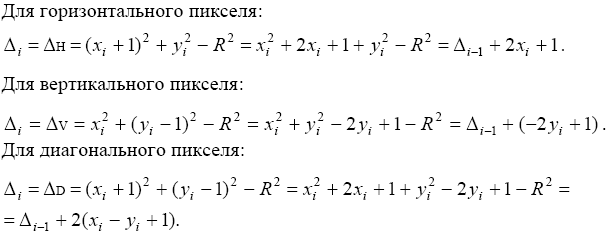
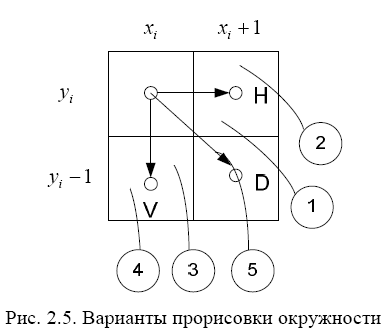
что центр окружности и начальная точка находятся точно в точках растра. Как и при рисовании линии, центральным понятием является ***ошибка*** *– разность между центром пикселя и действительным положением окружности*.



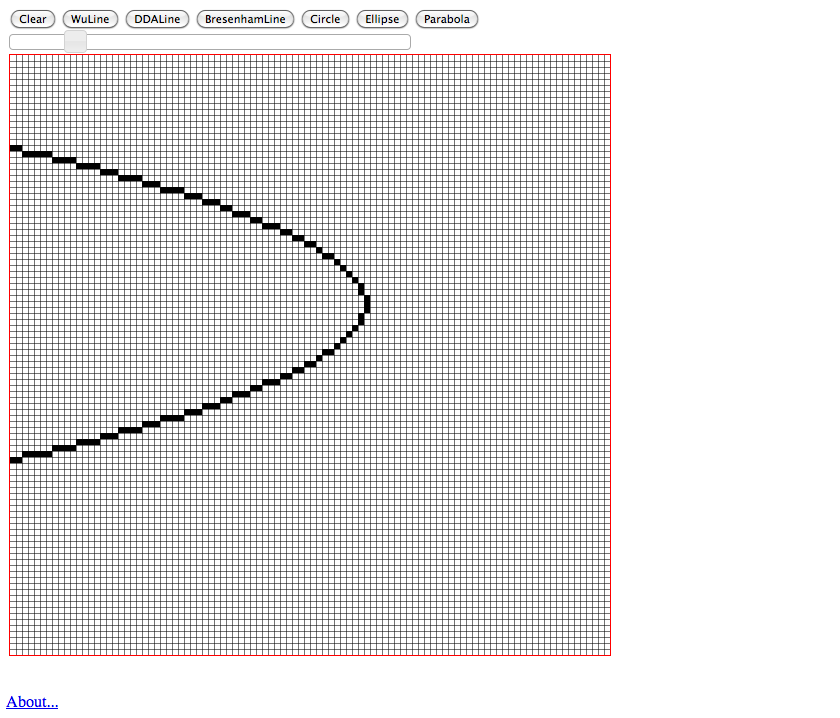
Сущность метода состоит в следующем:

1) вычислить ошибки для пикселей претендентов;

2) выбрать среди них пиксель с минимальным абсолютным значением разности квадратов расстояний от центра окружности до пикселя-претендента и до окружности.



**Пример работы программы:**



var x0 = this.center.get('x');

var y0 = this.center.get('y');

if(this.p < 0) {

var limit = this.canvas.width;

} else {

var limit = 0;

}

var p = Math.abs(this.p);

var x = 0;

var y = 0;

if(p != 0) {

var lim = - Math.abs(limit - x0);

} else {

var lim = 0;

}

if(p == 1) {

var delta = 0;

} else {

var delta = -1;

}

var point = new Point({

x: x,

y: y

});

while(x >= lim) {

point.set({x: x0 + Math.sign(this.p) \* x, y: y0 + y});

this.canvas.drawPoint(point, this.color);

point.set({x: x0 + Math.sign(this.p) \* x, y: y0 - y});

this.canvas.drawPoint(point, this.color);

var error = 2 \* (delta - p + 2 \* y + 1);

if(delta < 0 && error <= 0) {

y++;

delta += 2 \* y + 1;

continue;

}

var error = 2 \* (delta - 2 \* p + y) + 1;

if(delta > 0 && error > 0) {

x--;

delta += - 2 \* p;

continue;

}

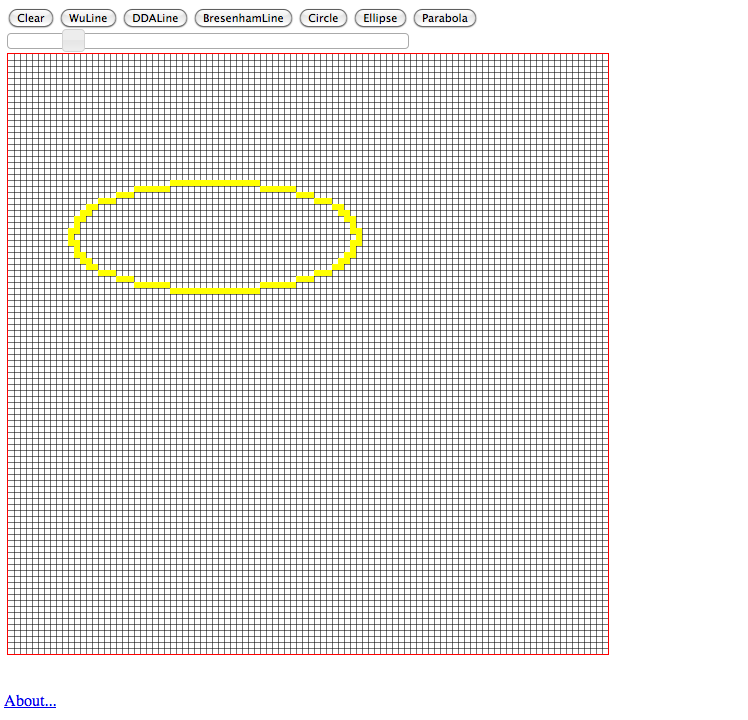
x--;

y++;

delta += 2 \* y - 2 \* p + 1;

}

}



var x0 = this.center.get('x');

var y0 = this.center.get('y');

var x = 0;

var y = this.b;

var p = new Point({

x: x,

y: y

});

var delta = Math.sqr(this.a) + Math.sqr(this.b) - 2 \* Math.sqr(this.a) \* this.b;

var error = 0;

while(y >= 0) {

p.set({x: x0 + x, y: y0 + y});

this.canvas.drawPoint(p, this.color);

p.set({x: x0 + x, y: y0 - y});

this.canvas.drawPoint(p, this.color);

p.set({x: x0 - x, y: y0 + y});

this.canvas.drawPoint(p, this.color);

p.set({x: x0 - x, y: y0 - y});

this.canvas.drawPoint(p, this.color);

var error = 2 \* (delta + Math.sqr(this.a) \* y) - Math.sqr(this.a);

if(delta < 0 && error <= 0) {

x++;

delta += Math.sqr(this.b) \* (2 \* x + 1);

continue;

}

error = 2 \* (delta + Math.sqr(this.b) \* x) - Math.sqr(this.b);

if(delta > 0 && error > 0) {

y--;

delta += Math.sqr(this.a) \* (1 - 2 \* y);

continue;

}

++x;

delta += Math.sqr(this.b) \* (2 \* x + 1) + Math.sqr(this.a) \* (1 - 2 \* y);

--y;

}

}

**Выводы:**

В ходе лабораторной работы был изучен и реализован алгоритм Брезенхема для генерации кривых второго порядка, в частности алгоритм для генерации эллипса. Был разработан и реализован алгоритм для генерации параболы на основе алгоритма Брезенхема. Были изучены принципы создания алгоритмов для генерации любых кривых второго порядка, которые основываются на вычисление ошибок для претендентов и выборе между H, V и D пикселями.