**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

Булова М.И.

Титенков П.В

Минск 2011

Цель:

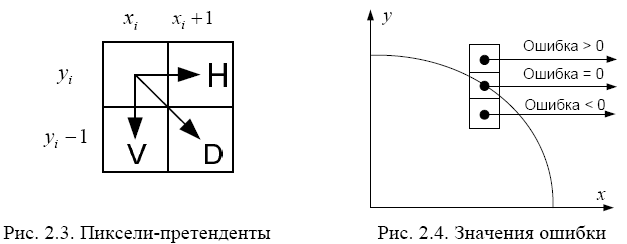
Изучить алгоритмы генерации кривых второго порядка. Разработать алгоритм генерации гиперболы на основе алгоритма Брезенхема. Разработать элементарный графический редактор, реализующий построение линий второго порядка: гипербола, эллипс.

**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,

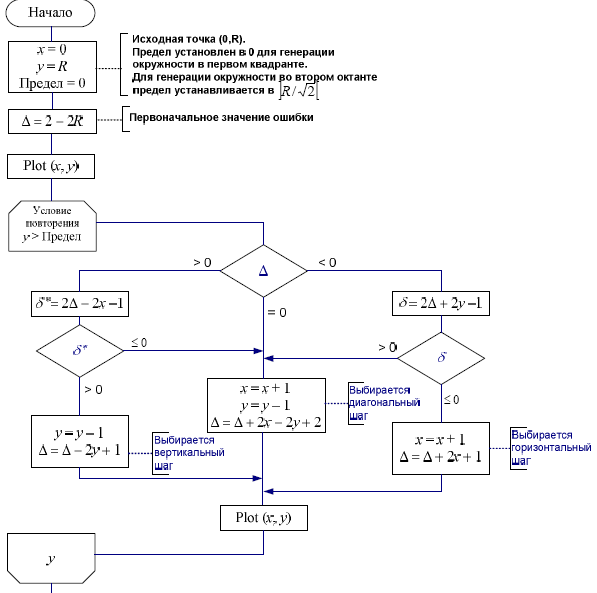
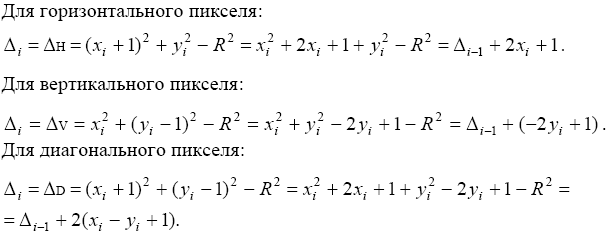
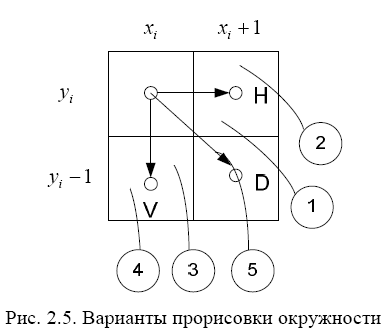
что центр окружности и начальная точка находятся точно в точках растра. Как и при рисовании линии, центральным понятием является ***ошибка*** *– разность между центром пикселя и действительным положением окружности*.



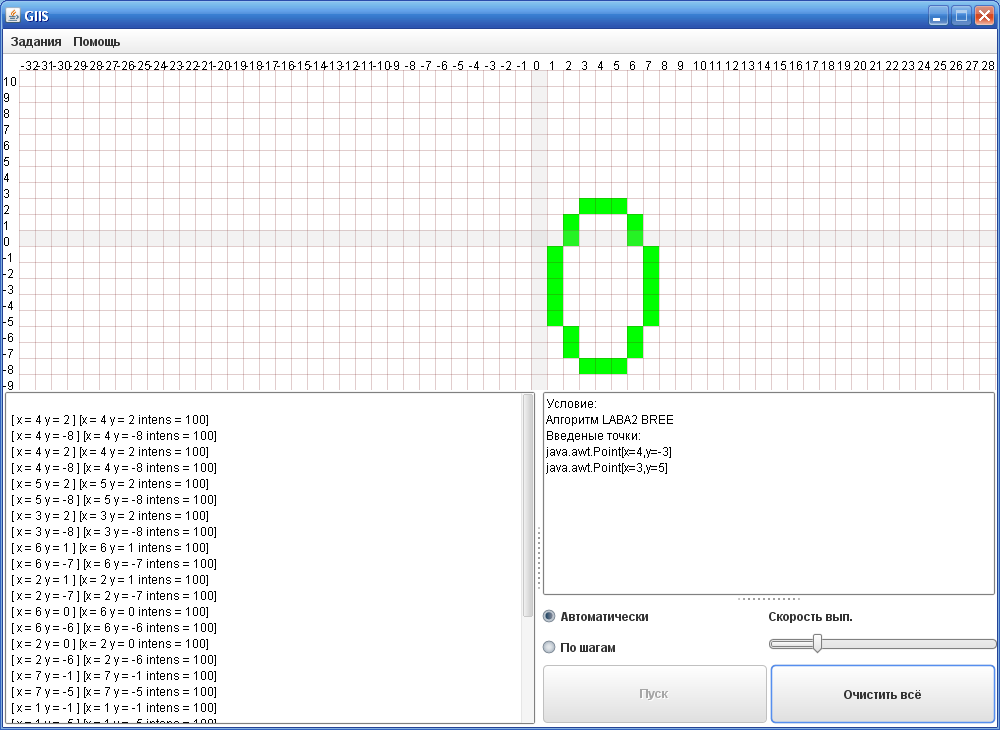
Сущность метода состоит в следующем:

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:**



**Алгоритм построения эллипса:**

public class EllipseBREZ extends Algorithm {

private int a = 0;

private int b = 0;

private int radius = 0;

private int cX = 0;

private int end = 0;

public EllipseBREZ(Point p1, Point p2){

super(p1, p2);

}

@Override

public void prepareAlg() {

MainFrame.setALG(BREZ\_E);

a = p2.x;

b = p2.y;

x = p1.x;

y = p1.y;

cX = 0;

radius = b;

end = 0;

iter = b\*b - 2\*b\*a\*a + a\*a;

}

@Override

public ArrayList<Cell> nextStepAlg() {

ArrayList<Cell> cells = new ArrayList();

String descr = "[x = " + (x+cX) + " y = " + (y+radius) + " intens = 100]";

cells.add(new Cell(round(x+cX), round(y+radius), Color.GREEN, descr));

descr = "[x = " + (x+cX) + " y = " + (y-radius) + " intens = 100]";

cells.add(new Cell(round(x+cX), round(y-radius), Color.GREEN, descr));

descr = "[x = " + (x-cX) + " y = " + (y+radius) + " intens = 100]";

cells.add(new Cell(round(x-cX), round(y+radius), Color.GREEN, descr));

descr = "[x = " + (x-cX) + " y = " + (y-radius) + " intens = 100]";

cells.add(new Cell(round(x-cX), round(y-radius), Color.GREEN, descr));

if(radius < end){

return null;

}

if(iter == 0) {

cX ++;

radius --;

iter += 2\*cX\*b\*b + b\*b -2\*radius\*a\*a + a\*a;

System.out.println("Pixel D");

}

else if(iter > 0) {

if((2\*(iter - b \* b \* cX) - b\*b) <= 0) {

cX ++;

radius --;

iter += 2\*cX\*b\*b + b\*b -2\*radius\*a\*a + a\*a;

System.out.println("Pixel V");

}

else {

radius --;

iter += a\*a -2\*radius\*a\*a;

System.out.println("Pixel D");

}

} else {

if((2\*(iter + a \* a \* radius) - a\*a) <= 0) {

cX ++;

iter += 2\*cX\*b\*b +b\*b;

System.out.println("Pixel D");

}

else {

cX ++;

radius --;

iter += 2\*cX\*b\*b + b\*b -2\*radius\*a\*a + a\*a;

System.out.println("Pixel H");

}

}

return cells;

}

@Override

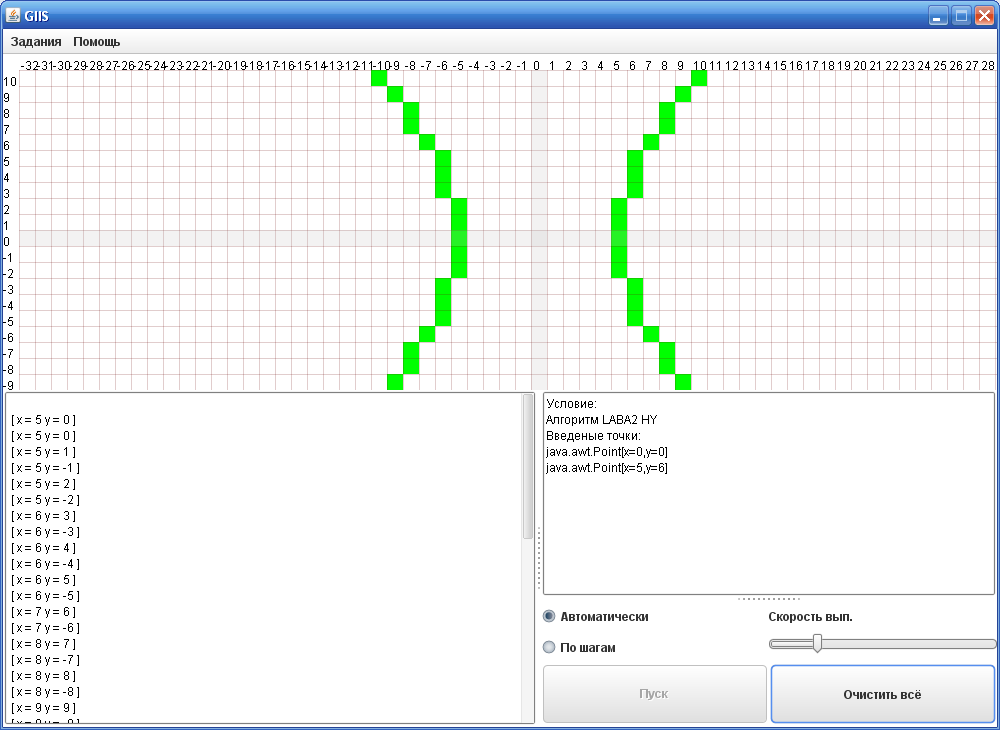
public Class<?> getAlgClass() {

return EllipseBREZ.class;

}

}

**Пример построения гиперболы:**



**Алгоритм построения гиперболы:**

public class Hyberbola extends Algorithm {

private int a = 0;

private int b = 0;

private int radius = 0;

private int cX = 0;

private int end = 0;

public Hyberbola(Point p1, Point p2) {

super(p1,p2);

}

@Override

public void prepareAlg() {

MainFrame.setALG(Hy);

a = p2.x;

b = p2.y;

x = p1.x;

y = p1.y;

cX = 0;

radius = b;

end = 0;

iter = b\*b - 2\*b\*a\*a + a\*a;

}

@Override

public ArrayList<Cell> nextStepAlg() {

if (step >= 2)

return null;

ArrayList<Cell> cells = new ArrayList();

String descr = "";

int limY = 10;

int limX = 32;

a = p2.x;

b = p2.y;

int a2 = a \* a;

int b2 = b \* b;

int y = 0;

int x = a;

int d1 = 2 \* a \* b2 + b2 - a2;

int d2 = 0;

int \_goto = 1;

do {

switch (\_goto) {

case 1:

if (step == 0) {

cells.add(new Cell(round(0 + x), round(0 + y), Color.GREEN, descr));

cells.add(new Cell(round(0 + x), round(0 - y), Color.GREEN, descr));

}

if (step == 1) {

cells.add(new Cell(round(0 - x), round(0 + y), Color.GREEN, descr));

cells.add(new Cell(round(0 - x), round(0 - y), Color.GREEN, descr));

}

if (y >= limY || x >= limX) {

\_goto = 0;

continue;

}

if (d1 < 0) {

\_goto = 2;

continue;

}

if (d1 > 0) {

\_goto = 3;

continue;

}

if (d1 == 0) {

\_goto = 20;

continue;

}

case 2:

d2 = 2 \* d1 + 2 \* y \* a2 + a2;

if (d2 <= 0) {

\_goto = 30;

continue;

}

if (d2 > 0) {

\_goto = 20;

continue;

}

case 3:

d2 = 2 \* d1 - 2 \* x \* b2 - b2;

if (d2 <= 0) {

\_goto = 20;

continue;

}

if (d2 > 0) {

\_goto = 10;

continue;

}

case 10:

y = y + 1;

d1 = d1 - 2 \* y \* a2 - a2;

\_goto = 1;

continue;

case 20:

x = x + 1;

y = y + 1;

d1 = d1 - 2 \* y \* a2 - a2 + 2 \* x \* b2 + b2;

\_goto = 1;

continue;

case 30:

x = x + 1;

d1 = d1 + 2 \* x \* b2 + b2;

\_goto = 1;

continue;

}

} while (\_goto != 0);

++step;

return cells;

}

private int step = 0;

@Override

public Class<?> getAlgClass() {

return Hyberbola.class;

}

}

**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 гиперболы 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.