POINT IN POLYGON ALGORITHMUS

PointInPolygon

```
public boolean PointInPolygon(int px, int py) {
  // Keine oder eine Ecke → return false
  // Zähle Kanten mit echtem Schnitt
  // mit horizontaler Geraden durch px, py
  // links von px
  // Wenn Anzahl ungerade, return true
  // Wenn Anzahl gerade, return false
}
```

PointInPolygon

```
public boolean PointInPolygon(int px, int py) {
  if (first == last) // keine oder eine Ecke
    return false;
  // Zähle Kanten mit echtem Schnitt
  // mit horizontaler Geraden durch px, py
  // links von px
  // Wenn Anzahl ungerade, return true
  // Wenn Anzahl gerade, return false
}
```

PointInPolygon

```
public boolean PointInPolygon(int px, int py) {
   if (first == last) // keine oder eine Ecke
      return false;
   Vertex v = first; // Invariante: first.next != first
   int count = 0;
   do {
      // wenn Schnitt von v – v.next
      // mit Horizontaler durch px, py links von px
      // dann inkrementiere count
      v = v.next;
   } while (v != first);
   return (count % 2 == 1);
}
```

PointInPolygon

```
public boolean PointInPolygon(int px, int py) {
   if (first == last) // keine oder eine Ecke
      return false;
   Vertex v = first;  // Invariante: first.next != first
   int count = 0;
   do {
      if (IntersectHorizontalLeft(px, py, v, v.next))
         ++count;
      v = v.next;
   } while (v != first);
   return (count % 2 == 1);
}
```

PointInPolygon

```
public boolean PointInPolygon(int px, int py) {
   if (first == last) // keine oder eine Ecke
      return false;
   Vertex v = first;  // Invariante: first.next != first
   boolean b = false; // "Zähl"variable (false -> true -> false ... )
   do {
      if (IntersectHorizontalLeft(px, py, v, v.next))
         b = !b;
      v= v.next;
   } while (v != first);
   return b;
}
```

Schnitt?

```
boolean IntersectHorizontalLeft(int x, int y, Vertex a, Vertex b) {
   // y-Wert zwischen p.y und q.y, nicht jedoch unten
   if (y <= p.y && y > q.y || y > p.y && y <= q.y) {
      // Schnittpunktberechnung
      double sx = p.x + (q.x - p.x) * (double)(y-p.y) / (q.y-p.y);
      // Schnittpunkt links
      return (sx <= x); // Schnittpunkt links
   }
   return false; // Kein Schnittpunkt
}
```

Konversion und Fliesskommarechnung nötig? → Umformen

DRAW LINE

PointInPolygon

```
public static void Line(ImageViewer v, int px, int py, int qx, int qy)
{
// bestimme dx, dy
// wenn abs(dx) > (dy) flache Linie entlang x
// sonst steile Linie entlang y
}
```

PointInPolygon

```
public static void Line(ImageViewer v, int px, int py, int qx, int qy){
        int dx = qx-px;
        int dy = qy-py;
        if (abs(dx)>abs(dy)) { // flache Linie entlang x
            for (int x = px; x != qx; x++) // immer richtig? → nein.
            {}
        }
        else { //  steile Linie entlang y
        }
}
```

PointInPolygon

```
public static void Line(ImageViewer v, int px, int py, int qx, int qy){
        int dx = qx-px;
        int dy = qy-py;
        int sx = 1;       if (dx < 0) sx = -1;
        int sy = 1;       if (dy < 0) sy = -1;
        if (dx*sx > dy*sy) // da haben wir nun sogar etwas gespart
        { // flache Linie entlang x
            for (int x = px; x != qx+sx; x+=sx){ // traversiert alle Pixel
                // berechne y zu x und
                // zeichne Punkt
            }
        }
        else // steile Linie entlang y
        { }
}
```

PointInPolygon

```
public static void Line(ImageViewer v, int px, int py, int qx, int qy){
        int dx = qx-px;
        int dy = qy-py;
        int sx = 1;       if (dx < 0) sx = -1;
        int sy = 1;       if (dy < 0) sy = -1;
        if (dx*sx > dy*sy) { // flache Linie entlang x
            for (int x = px; x != qx+sx; x+=sx){
                int y = py + (int)((x-px) * (double)dy / dx+0.5);
                v.dot(x, y);
            }
        }
        else // steile Linie entlang y
        {
             // nun analog wie oben
        }
}
```

PointInPolygon

```
public static void Line(ImageViewer v, int px, int py, int qx, int qy){
        int dx = qx-px;
        int dy = qy-py;
        int sx = 1;         if (dx < 0) sx = -1;
        int sy = 1;         if (dy < 0) sy = -1;
        if (dx*sx > dy*sy) { // flache Linie entlang x
            for (int x = px; x != qx+sx; x+=sx){
                int y = py + (int)((x-px) * (double)dy / dx+0.5);
                v.dot(x, y);
            }
        }
        else {// steile Linie entlang y
            for (int y = py; y != qy+sy; y+=sy){
                int x = px + (int)((y-py) * (double)dx / dy+0.5);
                v.dot(x, y);
            }
        }
}
```