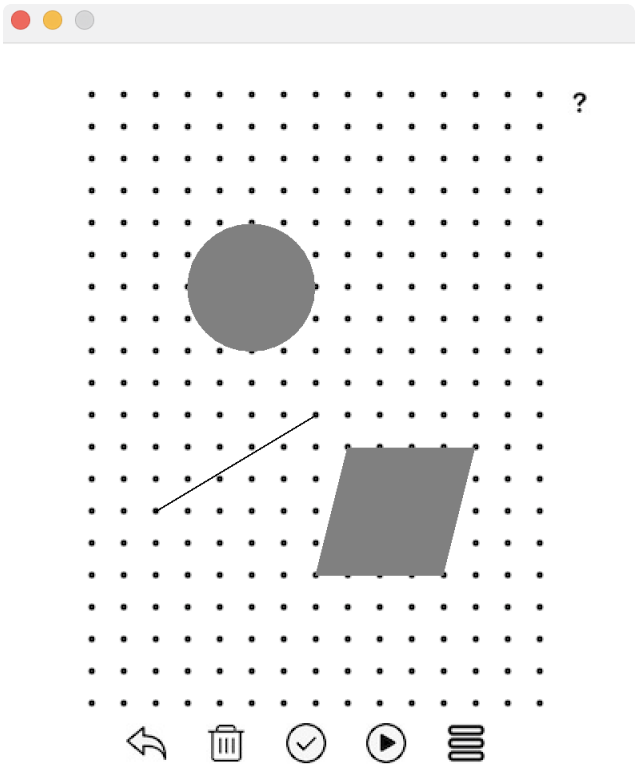
okay?

绘图逻辑

> 画线
> 画圆
> 画多边形

1. 首次点击，将点击的点选中
2. 第二次点击，判断是否与首次点击位置相同
    a. 相同，判断两次点击的时间差
        i. 时间差小于 1s，表示双击，画圆
        ii. 时间差大于 1s，表示取消绘图
    b. 不同，两点连线，开始绘图
3. 第三次点击，判断是否与首次或第二次点击位置相同
    a. 相同，表示画线
    b. 不同，两点连线，继续绘图
4. 第四次点击或以上，判断当前点是否被此次绘图点击过
    a. 是，是否与首次点击相同
        i. 是，两点连线，画多边形完成
        ii. 否，取消绘图
    b. 否，两点连线，继续绘图



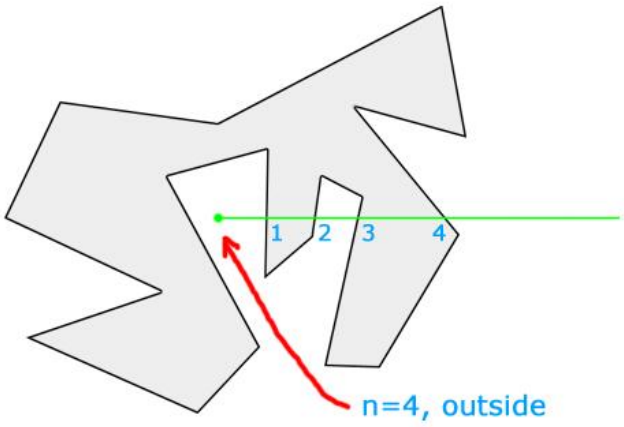
射线法

> 射线法思路，by 米粽粽（原文图片已裂）
> 转载博客，by IndeReChill

逻辑

> 目标：判断某一点是否在图形内

从点做一条射线，计算它跟多边形边界的交点个数，如果交点个数为奇数，那么点在多边形内部，否则点在多边形外

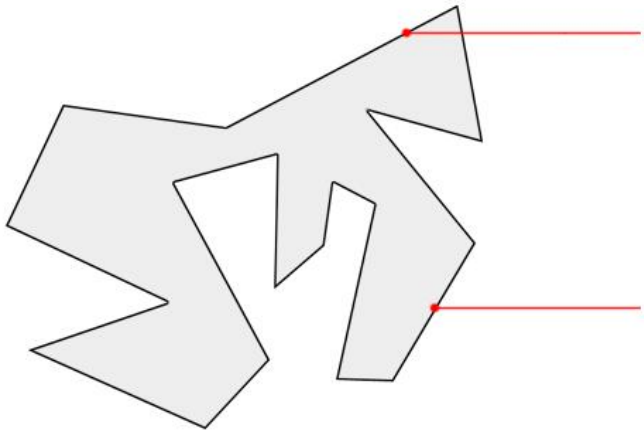


因为该射线不计长度，最终都会穿出多边形。如果点在多边形外部，首次交点会穿入，则穿入和穿出一一对应，与多边形的交点为偶数；如果点在多边形内部，首次交点为穿出，相对于点在内部，少了一次穿入的交点，则与多边形的交点为奇数

其他情况：

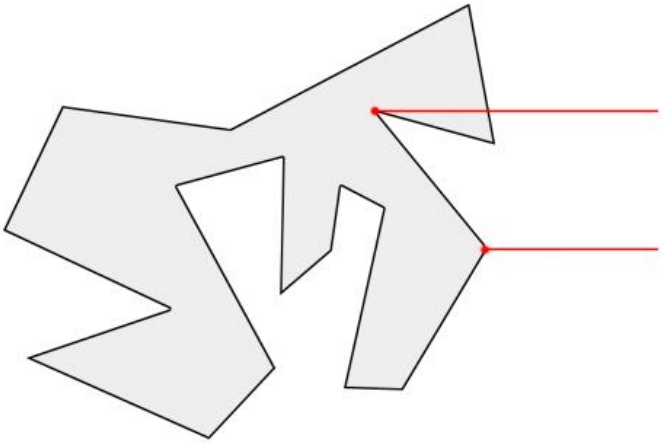
1. 点在多边形边上

解决办法：直接判断交点和多边形边的斜率是否相等，见下面“射线实现逻辑”



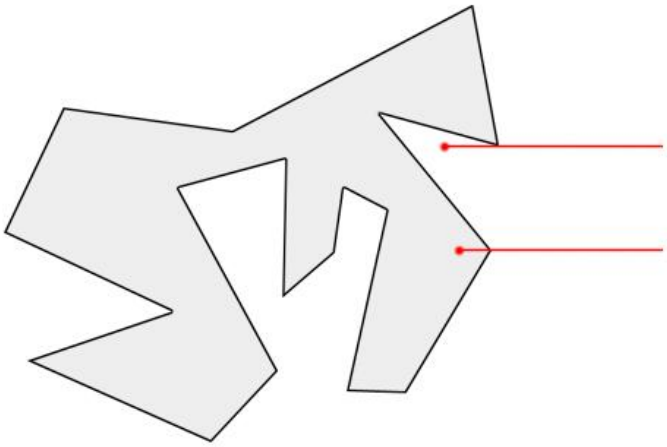
2. 点与多边形顶点重合

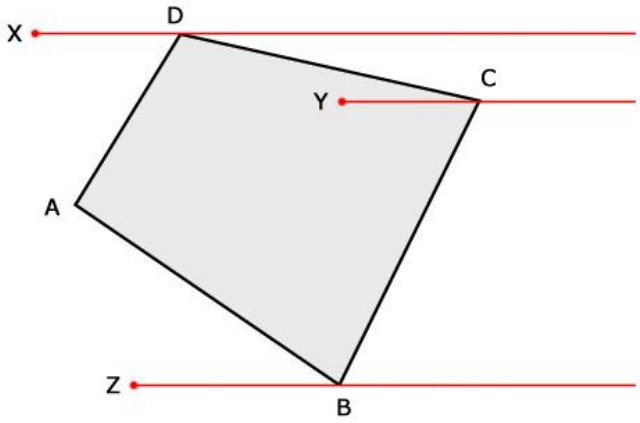
解决办法：直接比较交点和多边形的顶点坐标



3. 射线经过多边形顶点

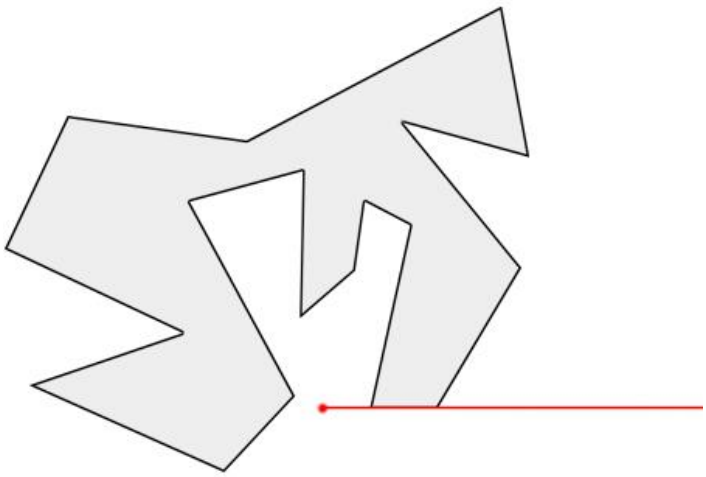
解决办法：假设规定射线经过的点属于射线以上的一侧。图2 中，D和C在射线Y的上方，所以没有与 CD 相交，但与 CB 相交，所以判定Y点在多边形内；射线Z没有与任何一边相交，判定在多边形外；射线X与 AD 和 CD 相交，判定在多边形外





4. 射线经过多边形边（是上一种的特例，经过两个相邻顶点）

解决办法：上述的逻辑已覆盖此情况



实现

传入一个多边形，遍历多边形的每条边

有以下多种情况判断：

1. ‘点’与多边形顶点重合，判定为 true（在内部）
2. ‘点’在多边形的边上，判定为 true
3. 边是平行于x轴的平行线，‘点’的y坐标与之相同，且x坐标在边范围内，则一定在边上，判定为 true
4. 从‘点’开始向右无限延长，计算延长线与边的交叉点
    a. 交叉点在‘点’的左侧，不存在，忽略
    b. 交叉点在‘点’的右侧，表示穿越一次【需要额外判断交叉点与顶点重合情况】

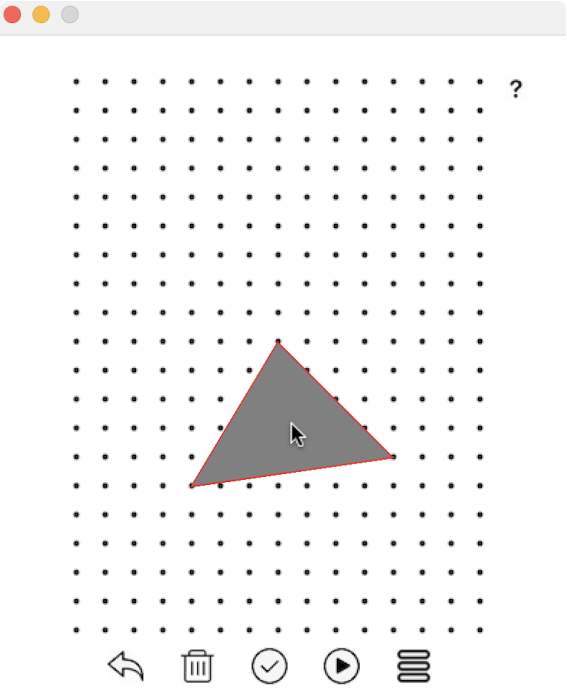
Java | 复制代码

```java
private static boolean isInsidePolygon(Point point, ShapeDTO shapeDTO) {
        List<PointDTO> points = shapeDTO.getPoints();
        int size = points.size();
        int px = point.x;
        int py = point.y;
        boolean flag = false;
        // 遍历图形的每根线
        for (int i = 0; i < size - 1; i++) {
            int x1 = realX(points.get(i).getX());
            int y1 = realY(points.get(i).getY());
            int x2 = realX(points.get(i + 1).getX());
            int y2 = realY(points.get(i + 1).getY());
            // 点与多边形顶点重合
            if ((x1 == px && y1 == py) || (x2 == px && y2 == py)) {
                return true;
            }
            if (y1 != y2) {
                // p点的 y坐标 在线段的 y坐标 之间
                if ((py >= y1 && py <= y2) || (py >= y2 && py <= y1)) {
                    // 计算出线段上 y坐标=pY 点的 x坐标
                    double x = x1 + (double)(py - y1) * (x2 - x1) / (y2 - y1);
                    // 点在多边形的边上
                    if (x == px) {
                        return true;
                    }
                /*
                 思路：以p点开始，做一条平行于x轴，向右无限延长的线
                 x 大于 pX，表示 p点 在线段的左侧，则 (x, pY)点 是 p延长线与线段的交点
                 x 小于 pX，表示 p点 在线段的右侧，此认定为无交点
                 */
                    if (x > px) {
                        // 处理延伸线穿过顶点的情况
                        if (x == x1 || x == x2) {
                            if ((y1 < y2 && py == y1) || (y1 > y2 && py == y2)) {
                                flag = !flag;
                            }
                        } else {
                            flag = !flag;
                        }
                    }
                }
            }
```

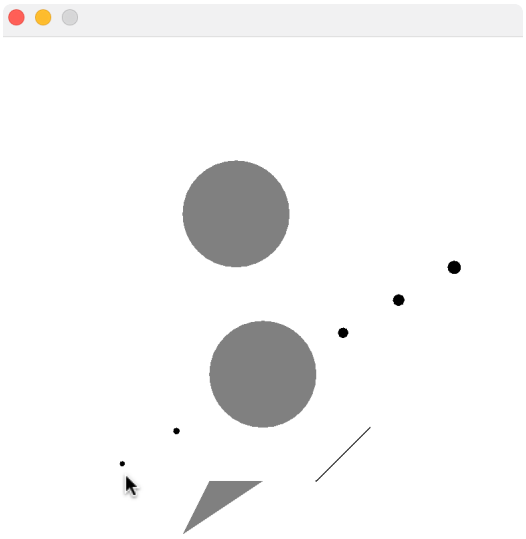
```
43                  } else if (py == y1 && ((px >= x1 && px <= x2) || (px >= x2 &&
    px <= x1))) {
44                      // 处理p点在平行于x轴的线段上
45                      return true;
46                  }
47              }
48              return flag;
        }
```

应用

- 绘图板中鼠标可选中图形
- 被图形覆盖的`点`，不触发点击



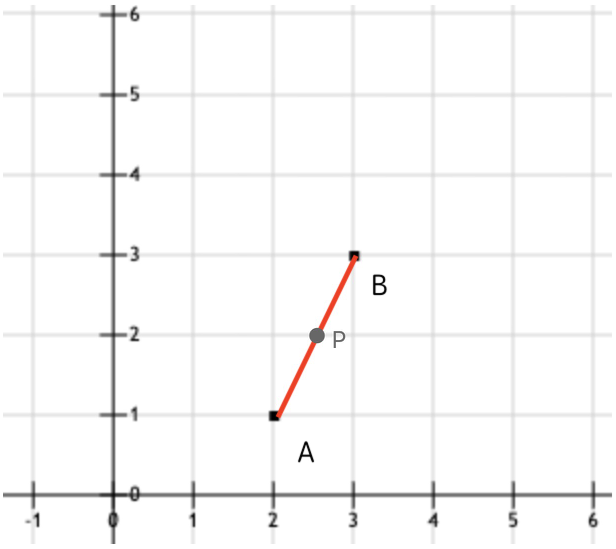
- 拉伸线的小球不覆盖图形



斜率的应用

判断点是否在线上

在上文射线法的实现中，需判断点是否在线上



假设点 `a(2, 1)`， `b(3, 3)`， `p(2.5, 2)`
目标：根据 py，求出若在 AB 线上则对应的 x坐标
公示：ax + (py – ay) * (bx – ax) / (by – bx)
= 2 + (2 – 1) * (3 – 2) / (3 – 1)
= 2.5
因为 `px = 2.5`，等于计算结果，所以 p点在线上

计算小球坐标增量

在 JPanel 上显示的坐标类型为 int，为实现小球**平滑移动**，则将坐标增量变量（即moveX 和 moveY）设置为 double 类型。具体代码如下：

Java 复制代码

```java
public void calculationIncr(int startX, int startY, int endX, int endY) {
    double dStartX = startX;
    double dStartY = startY;
    double dEndX = endX;
    double dEndY = endY;
    if (startX != endX && startY != endY) {
        moveX = (dEndX - dStartX) / Math.abs(dEndY - dStartY);
        if (Math.abs(moveX) > 1) {
            moveX = moveX > 1 ? 1.0 : -1.0;
        }
        moveY = (dEndY - dStartY) / Math.abs(dEndX - dStartX);
        if (Math.abs(moveY) > 1) {
            moveY = moveY > 1 ? 1.0 : -1.0;
        }
    } else {
        if (startX == endX && startY != endY) {
            moveX = 1.0;
            moveY = 0.0;
        } else if (startX != endX) {
            moveX = 0.0;
            moveY = 1.0;
        }
    }
}
```

小球的反弹

未完成...