# Lab1 说明文档

## 一、功能实现

1. 多边形渲染的实现：

实现了 render(poly, color)方法，这个方法传入多边形（这里的多边形是指多边形的顶点集，以一个三角形为例 eg. [[0,0], [50,0], [25,25]]）和颜色作为参数。然后以该多边形的 Y 值范围作为要扫描的 Y 值范围，以与 X 轴平行的线与多边形进行相交，得到交点，然后将交点按照 X 值进行排序，再这些交点两两配对，配对的交点用提供的 drawLine()方法进行画线。当完成了整个 Y 值范围的画线，再将多边形的边画好，整个多边形也就实现了渲染。

在这次的 lab 中，因为是要实现四个四边形的组合图形。所以又实现了 renderAll(polygon)方法，传入的参数 polygon 是 config.js 中提供的，在这个方法中将 polygon 的每一个四边形提取出来，单独进行渲染，当四个四边形渲染完成，也就完成了整体多边形的渲染。

最后使用 canvas 的画圆方法，实现了 drawRedPoint(cxt, x, y)方法和 drawNinePoint(cxt, points)方法，完成了对 9 个红色节点的渲染啊。

2. 可拖动变形的多边形实现：

因为红色节点并不是以一个 DOM 树节点的形式创建，所以无法对其直接添加事件，于是采用在 canvas 上接收鼠标事件来替代。首先监听了 onmousedown 事件，获得鼠标点下时的 x 和 y 值，然后以该 x 和 y 作为参数，调用我实现的 drag()方法。

drag()方法先是调用 findPoint 方法，findPoint()方法的作用是根据参数 x 和 y，判断这个 x 和 y 是否在某个红色节点圆上，若在，返回对应点的下标，若不在，返回-1。回到 drag()方法中，当其接收到>=0 的返回值后，监听 onmousemove 事件，并获得鼠标移动后的 x2 和 y2 值，然后将 x2 和 y2 设为对应点的新值，最后重新渲染，这个时候渲染出来的就是要实现拖动后的图形。

## 二、遇到的问题和解决的办法

1. X 轴平行线与多边形交点的交点问题：

这一个问题上课着重讲过，但是，理论上都能理解在什么时候取两点，什么时候取一点。但是，在算法实现上，却遇到了很多问题。最后的做法是：每与一条边获得一个交点，调用 contains(nodes, point)方法，判断其是否已经存在于交点组中。若不在，返回-1 若在，再判断这个点是在交点组的开头还是末尾。若是开头，返回 1，末尾返回 2。（判断开头还是末尾非常重要，因为这个关系到后面用什么方式来判断取点数）。

当 contains()方法返回正值以后，说明这个点是多边形两条边的交点，然后判断这两条边的另外两个点在 y 值上是否在交点的同一侧，若在，记为两点，若不在，记为一点。如下所示（x3 和 y3 是另一条边的另一个端点）：

```
//从上到下进行扫描
for (var y = minY;y <= maxY;y++) {
    //将多边形的每一条边与横向的射线相交，得到交点，并按照x值排序
    for (var i = 0;i < lengthOfpoly;i++) {
        //首先得到两个端点的坐标，
        y1 = poly[i][1];
        x1 = poly[i][0];
        y2 = poly[(i+1)%lengthOfpoly][1];
        x2 = poly[(i+1)%lengthOfpoly][0];
        if (y === y1 && y2 === y) { //当三个y值相同时
            p = [poly[i][0]+1,y];
            nodes.push(p);
        }
        else if (((y >= y1 && y <= y2) || (y <= y1 && y >= y2))) { //当y值在y1和y2的范围内时
            var x = parseInt(((y - y1) * (x2 - x1) / (y2 - y1)) + x1);

            p = [x,y];
            var flag = contains(nodes,p);
            var y3;
            if ( flag === 0)
                nodes.push(p);
            else if (flag === 1){
                // 同一点出现两次，说明这个点是图形两边的交点，然后判断这两边的另外两个端点在y值上是否是同一侧
                y3 = poly[(i+2)%lengthOfpoly][1];
                if ((y3-y2) * (y1 - y2) > 0)
                    nodes.push(p);
            }
            else {
                y3 = poly[(i+lengthOfpoly - 1)%lengthOfpoly][1];
                if ((y3-y1) * (y2 - y1) > 0)
                    nodes.push(p);
            }
```

2. 多边形出现于 X 轴平行的边的取点问题

这种边的取点不管是取哪个端点，都会影响到第一个问题的解决。因为这种边会在渲染边的时候得到渲染，所以，这个点只是取一下过渡的作用，所以，最后的解决办法是取了其中一个端点(x+1)的 x 值，以及对应的 y 值。这样既不会影响第一个问题的解决，又不会影响整体的渲染效果。

3. 滑动条拖动以后，鼠标取点 x,y 偏移的问题

这个问题本质上是鼠标取点的坐标系与 canvas 的坐标系不统一的问题。鼠标取点是屏幕的坐标系，而 canvas 是本身的坐标系。比如说有一个点在屏幕下方，y 值是 1000，当滑动条拉下来，让它处于中间时，鼠标去点它，得到的 y 坐标却是 500。这样一来，在发生拖动事件时，这个点就没有被捕获到。解决办法是鼠标取点后，给它加上 document.body 和 document.documentElement 的偏移量。如下所示：

```
var e = ev||event;
var x = e.clientX + document.body.scrollLeft + document.documentElement.scrollLeft;
var y = e.clientY + document.body.scrollTop + document.documentElement.scrollTop;
```