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）：



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：

