test.js

```
window.addEventListener("load", main, false);

function main() {

    //绘制线段的函数绘制一条从(x1,y1)到(x2,y2)的线段，
    // cxt和color两个参数意义与绘制点的函数相同，
    function drawLine(cxt, x1, y1, x2, y2, color) {

        cxt.beginPath();
        cxt.moveTo(x1, y1);
        cxt.lineTo(x2, y2);
        cxt.closePath();
        cxt.strokeStyle = "rgb(" + color[0] + "," +
            +color[1] + "," +
            +color[2] + ")";
        //这里线宽取1会有色差，但是类似半透明的效果有利于debug，取2效果较好
        cxt.lineWidth = 2;

        cxt.stroke();
    }

    var c = document.getElementById("myCanvas");
    var cxt = c.getContext("2d");


    //将canvas坐标整体偏移0.5，
    // 用于解决宽度为1个像素的线段的绘制问题，具体原理详见project文档
    cxt.translate(0.5, 0.5);

    document.getElementById("myCanvas").style.position = 'absolute';
    init();

    var numShapes, shapes;
    var dragIndex, dragging;
    var mouseX, mouseY;
    var dragHoldX, dragHoldY;
    var pointRad;
    var colors;

    scan(polygon, colors);
    drawShapes();

    function init() {
        numShapes = vertex_pos.length;
        pointRad = 6;
        shapes = [];
        colors = [];
        for (var i = 0, len = polygon.length; i < len; i++) {
            colors.push(vertex_color[polygon[i][0]]);
        }
        document.getElementById('myCanvas').height = canvasSize.maxY;
        document.getElementById('myCanvas').width = canvasSize.maxX;
        makeShapes();
        cxt.clearRect(0, 0, c.width, c.height);
        c.addEventListener("mousedown", mouseDownListener, false);
    }

    function newNETObject(x, dx, ymax) {
        var ne = {};
        ne.x = x;
        ne.dx = dx;
        ne.ymax = ymax;
        return ne;
    }

    function newAETObject(x, dx, ymax) {
        var ae = {};
        ae.x = x;
        ae.dx = dx;
        ae.ymax = ymax;
        return ae;
    }

    function scanLine(polygon, index, color) {
        //找点
        var points = [];
        var point;
        let x_;
        let y_;
        for (var i = 0, len = polygon[index].length; i < len; i++) {
            x_ = vertex_pos[polygon[index][i]][0];
            y_ = vertex_pos[polygon[index][i]][1];
            point = {x: x_, y: y_};
            points.push(point);
        }


        //找边
        var edges = [];
        var parallel = [];
        for (var i = 0, len = points.length; i < len; i++) {
            if (points[i].y !== points[(i + 1) % len].y) {
                edges.push({dot: [points[i], points[(i + 1) % len]]});
            }
            else {
                parallel.push({dot: [points[i], points[(i + 1) % len]]});
            }
        }

        //NET
        //AET
        var ymax = canvasSize.maxY;
        var NET = Array(ymax);
        var AET = Array(ymax);
        for (var i = 0; i < ymax; i++) {
            NET[i] = [];
            AET[i] = [];
        }

        //构建NET

        for (var y = 0; y < ymax; y++) {
            for (var i = 0, edges_num = edges.length; i < edges_num; i++) {
                if (edges[i] !== 0) {
                    for (var j = 0; j < 2; j++) {
                        if (edges[i].dot[j].y === y) {
                            var another = 1 - j;
                            NET[y].push(newNETObject(edges[i].dot[j].x,
                                (edges[i].dot[j].x - edges[i].dot[another].x)
                                / (edges[i].dot[j].y - edges[i].dot[another].y),
                                edges[i].dot[another].y));
                            edges[i] = 0;
                            break;
                        }
                    }
                }
            }
        }

        // for (var i = 0; i < 768; i++) {
        //  for(var j = 0, len = NET[i].length; j < len; j++) {
        //      console.log(NET[i][j].x);
        //  }
        // }
        //构建AET
        for (var y = 0; y < ymax; y++) {
            for (var i = 0, len = NET[y].length; i < len; i++) {
                for (var j = y, max = NET[y][i].ymax; j < max; j++) {
                    AET[j].push(newAETObject(NET[y][i].x + NET[y][i].dx * (j - y),
                        NET[y][i].dx, NET[y][i].ymax));
                }
            }
        }

        //排序
        var tmp;
        for (var y = 0; y < ymax; y++) {
            for (var i = 0, len = AET[y].length; i < len; i++) {
                for (var j = i + 1; j < len; j++) {
                    if (AET[y][i].x > AET[y][j].x) {
                        tmp = AET[y][i];
                        AET[y][i] = AET[y][j];
                        AET[y][j] = tmp;
                    }
                }
            }
        }
        //画线
        for (var y = 0; y < ymax; y++) {
            for (var i = 0, len = AET[y].length; i < len; i += 2) {
                drawLine(cxt, AET[y][i].x, y, AET[y][i + 1].x, y, color);
            }
        }

        //处理平行线
        for (var i = 0, len = parallel.length; i < len; i++) {
            drawLine(cxt, parallel[i].dot[0].x, parallel[i].dot[0].y,
                parallel[i].dot[1].x, parallel[i].dot[1].y, color);
        }
    }

    function scan(polygon, colors) {
        for (var i = 0, len = polygon.length; i < len; i++) {
            scanLine(polygon, i, colors[i]);
        }
    }


    function makeShapes() {
        var Color;
        var Shape;
        for (var i = 0; i < numShapes; i++) {
            Color = "rgb(" + vertex_color[i][0] + "," +
                vertex_color[i][1] + "," + vertex_color[i][2] + ")";
            Shape = {
                x: vertex_pos[i][0], y: vertex_pos[i][1],
                rad: pointRad, color: Color
            };
            shapes.push(Shape);
        }
    }

    function mouseDownListener(evt) {
        var i;

        var highestIndex = -1;

        var bRect = c.getBoundingClientRect();
        mouseX = (evt.clientX - bRect.left) * (c.width / bRect.width);
        mouseY = (evt.clientY - bRect.top) * (c.height / bRect.height);


        for (i = 0; i < numShapes; i++) {
            if (hitTest(shapes[i], mouseX, mouseY)) {
                dragging = true;
                if (i > highestIndex) {
                    dragHoldX = mouseX - shapes[i].x;
                    dragHoldY = mouseY - shapes[i].y;
                    highestIndex = i;
                    dragIndex = i;
                }
            }
        }

        if (dragging) {
            window.addEventListener("mousemove", mouseMoveListener, false);
        }
        c.removeEventListener("mousedown", mouseDownListener, false);
        window.addEventListener("mouseup", mouseUpListener, false);


        if (evt.preventDefault) {
            evt.preventDefault();
        }
        else if (evt.returnValue) {
            evt.returnValue = false;
        }
        return false;
    }

    function mouseUpListener() {
        c.addEventListener("mousedown", mouseDownListener, false);
        window.removeEventListener("mouseup", mouseUpListener, false);
        if (dragging) {
            dragging = false;
            window.removeEventListener("mousemove", mouseMoveListener, false);
        }
    }

    function mouseMoveListener(evt) {
        var posX;
        var posY;
        var shapeRad = 0;
        var minX = shapeRad;
        var maxX = c.width - shapeRad;
        var minY = shapeRad;
        var maxY = c.height - shapeRad;
        //获取鼠标位置，进行坐标转换
        var bRect = c.getBoundingClientRect();
        mouseX = (evt.clientX - bRect.left) * (c.width / bRect.width);
        mouseY = (evt.clientY - bRect.top) * (c.height / bRect.height);

        //框定鼠标位置，避免越界
        posX = mouseX - dragHoldX;
        posX = (posX < minX) ? minX : ((posX > maxX) ? maxX : posX);
        posY = mouseY - dragHoldY;
        posY = (posY < minY) ? minY : ((posY > maxY) ? maxY : posY);

        shapes[dragIndex].x = posX;
        shapes[dragIndex].y = posY;
        vertex_pos[dragIndex][0] = posX;
        vertex_pos[dragIndex][1] = posY;
        cxt.clearRect(0, 0, c.width, c.height);
        scan(polygon, colors);
        drawShapes();
    }

    function hitTest(shape, mx, my) {

        var dx;
        var dy;
        dx = mx - shape.x;
        dy = my - shape.y;

        return (dx * dx + dy * dy < shape.rad * shape.rad * 4);
    }

    function drawShapes() {
        var i;
        for (i = 0; i < numShapes; i++) {
            // cxt.fillStyle = shapes[i].color;
            //不能用fill，所以把线宽设定为半径的两倍。。。
            cxt.beginPath();
            cxt.arc(shapes[i].x, shapes[i].y, shapes[i].rad, 0, 2 * Math.PI, false);
            cxt.closePath();
            cxt.lineWidth = shapes[i].rad * 2;
            cxt.strokeStyle = 'red';
            cxt.stroke();
        }
    }
}
```