

test.js

```
1  window.addEventListener("load", main, false);
2
3  function main() {
4
5      //绘制线段的函数绘制一条从(x1,y1)到(x2,y2)的线段,
6      // cxt和color两个参数意义与绘制点的函数相同,
7      function drawLine(cxt, x1, y1, x2, y2, color) {
8
9          cxt.beginPath();
10         cxt.moveTo(x1, y1);
11         cxt.lineTo(x2, y2);
12         cxt.closePath();
13         cxt.strokeStyle = "rgb(" + color[0] + "," +
14             +color[1] + "," +
15             +color[2] + ")";
16         //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
17         cxt.lineWidth = 2;
18
19         cxt.stroke();
20     }
21
22     var c = document.getElementById("myCanvas");
23     var cxt = c.getContext("2d");
24
25
26     //将canvas坐标整体偏移0.5,
27     // 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
28     cxt.translate(0.5, 0.5);
29
30     document.getElementById("myCanvas").style.position = 'absolute';
31     init();
32
33     var numShapes, shapes;
34     var dragIndex, dragging;
35     var mouseX, mouseY;
36     var dragHoldX, dragHoldY;
37     var pointRad;
38     var colors;
39
40     scan(polygon, colors);
41     drawShapes();
42
43     function init() {
44         numShapes = vertex_pos.length;
45         pointRad = 6;
46         shapes = [];
47         colors = [];
48         for (var i = 0, len = polygon.length; i < len; i++) {
49             colors.push(vertex_color[polygon[i][0]]);
50         }
51         document.getElementById('myCanvas').height = canvasSize.maxY;
52         document.getElementById('myCanvas').width = canvasSize.maxX;
53         makeShapes();
54         cxt.clearRect(0, 0, c.width, c.height);
55         c.addEventListener("mousedown", mouseDownListener, false);
56     }
57
58     function newNETObject(x, dx, ymax) {
59         var ne = {};
60         ne.x = x;
61         ne.dx = dx;
```

```

62     ne.ymax = ymax;
63     return ne;
64 }
65
66 function newAETObject(x, dx, ymax) {
67     var ae = {};
68     ae.x = x;
69     ae.dx = dx;
70     ae.ymax = ymax;
71     return ae;
72 }
73
74 function scanLine(polygon, index, color) {
75     //找点
76     var points = [];
77     var point;
78     let x_;
79     let y_;
80     for (var i = 0, len = polygon[index].length; i < len; i++) {
81         x_ = vertex_pos[polygon[index][i]][0];
82         y_ = vertex_pos[polygon[index][i]][1];
83         point = {x: x_, y: y_};
84         points.push(point);
85     }
86
87     //找边
88     var edges = [];
89     var parallel = [];
90     for (var i = 0, len = points.length; i < len; i++) {
91         if (points[i].y !== points[(i + 1) % len].y) {
92             edges.push({dot: [points[i], points[(i + 1) % len]]});
93         }
94         else {
95             parallel.push({dot: [points[i], points[(i + 1) % len]]});
96         }
97     }
98 }
99
100 //NET
101 //AET
102 var ymax = canvasSize.maxY;
103 var NET = Array(ymax);
104 var AET = Array(ymax);
105 for (var i = 0; i < ymax; i++) {
106     NET[i] = [];
107     AET[i] = [];
108 }
109
110 //构建NET
111
112 for (var y = 0; y < ymax; y++) {
113     for (var i = 0, edges_num = edges.length; i < edges_num; i++) {
114         if (edges[i] !== 0) {
115             for (var j = 0; j < 2; j++) {
116                 if (edges[i].dot[j].y === y) {
117                     var another = 1 - j;
118                     NET[y].push(newNETObject(edges[i].dot[j].x,
119                         (edges[i].dot[j].x - edges[i].dot[another].x)
120                         / (edges[i].dot[j].y - edges[i].dot[another].y),
121                         edges[i].dot[another].y));
122                     edges[i] = 0;
123                     break;
124                 }
125             }
126         }
127     }
128 }

```

```
126         }
127     }
128 }
129
130 // for (var i = 0; i < 768; i++) {
131 //     for(var j = 0, len = NET[i].length; j < len; j++) {
132 //         console.log(NET[i][j].x);
133 //     }
134 // }
135 //构建AET
136 for (var y = 0; y < ymax; y++) {
137     for (var i = 0, len = NET[y].length; i < len; i++) {
138         for (var j = y, max = NET[y][i].ymax; j < max; j++) {
139             AET[j].push(newAETObject(NET[y][i].x + NET[y][i].dx * (j - y),
140                                     NET[y][i].dx, NET[y][i].ymax));
141         }
142     }
143 }
144
145 //排序
146 var tmp;
147 for (var y = 0; y < ymax; y++) {
148     for (var i = 0, len = AET[y].length; i < len; i++) {
149         for (var j = i + 1; j < len; j++) {
150             if (AET[y][i].x > AET[y][j].x) {
151                 tmp = AET[y][i];
152                 AET[y][i] = AET[y][j];
153                 AET[y][j] = tmp;
154             }
155         }
156     }
157 }
158 //画线
159 for (var y = 0; y < ymax; y++) {
160     for (var i = 0, len = AET[y].length; i < len; i += 2) {
161         drawLine(cxt, AET[y][i].x, y, AET[y][i + 1].x, y, color);
162     }
163 }
164
165 //处理平行线
166 for (var i = 0, len = parallel.length; i < len; i++) {
167     drawLine(cxt, parallel[i].dot[0].x, parallel[i].dot[0].y,
168             parallel[i].dot[1].x, parallel[i].dot[1].y, color);
169 }
170 }
171
172 function scan(polygon, colors) {
173     for (var i = 0, len = polygon.length; i < len; i++) {
174         scanLine(polygon, i, colors[i]);
175     }
176 }
177
178
179 function makeShapes() {
180     var Color;
181     var Shape;
182     for (var i = 0; i < numShapes; i++) {
183         Color = "rgb(" + vertex_color[i][0] + "," +
184             vertex_color[i][1] + "," + vertex_color[i][2] + ")";
185         Shape = {
186             x: vertex_pos[i][0], y: vertex_pos[i][1],
187             rad: pointRad, color: Color
188         };
189         shapes.push(Shape);
190     }
191 }
```

```
190     }
191   }
192
193   function mouseDownListener(evt) {
194     var i;
195
196     var highestIndex = -1;
197
198     var bRect = c.getBoundingClientRect();
199     mouseX = (evt.clientX - bRect.left) * (c.width / bRect.width);
200     mouseY = (evt.clientY - bRect.top) * (c.height / bRect.height);
201
202
203     for (i = 0; i < numShapes; i++) {
204       if (hitTest(shapes[i], mouseX, mouseY)) {
205         dragging = true;
206         if (i > highestIndex) {
207           dragHoldX = mouseX - shapes[i].x;
208           dragHoldY = mouseY - shapes[i].y;
209           highestIndex = i;
210           dragIndex = i;
211         }
212       }
213     }
214
215     if (dragging) {
216       window.addEventListener("mousemove", mouseMoveListener, false);
217     }
218     c.removeEventListener("mousedown", mouseDownListener, false);
219     window.addEventListener("mouseup", mouseUpListener, false);
220
221
222     if (evt.preventDefault) {
223       evt.preventDefault();
224     }
225     else if (evt.returnValue) {
226       evt.returnValue = false;
227     }
228     return false;
229   }
230
231   function mouseUpListener() {
232     c.addEventListener("mousedown", mouseDownListener, false);
233     window.removeEventListener("mouseup", mouseUpListener, false);
234     if (dragging) {
235       dragging = false;
236       window.removeEventListener("mousemove", mouseMoveListener, false);
237     }
238   }
239
240   function mouseMoveListener(evt) {
241     var posX;
242     var posY;
243     var shapeRad = 0;
244     var minX = shapeRad;
245     var maxX = c.width - shapeRad;
246     var minY = shapeRad;
247     var maxY = c.height - shapeRad;
248     //获取鼠标位置，进行坐标转换
249     var bRect = c.getBoundingClientRect();
250     mouseX = (evt.clientX - bRect.left) * (c.width / bRect.width);
251     mouseY = (evt.clientY - bRect.top) * (c.height / bRect.height);
252
253     //框定鼠标位置，避免越界
```

```
254     posX = mouseX - dragHoldX;
255     posX = (posX < minX) ? minX : ((posX > maxX) ? maxX : posX);
256     posY = mouseY - dragHoldY;
257     posY = (posY < minY) ? minY : ((posY > maxY) ? maxY : posY);
258
259     shapes[dragIndex].x = posX;
260     shapes[dragIndex].y = posY;
261     vertex_pos[dragIndex][0] = posX;
262     vertex_pos[dragIndex][1] = posY;
263     cxt.clearRect(0, 0, c.width, c.height);
264     scan(polygon, colors);
265     drawShapes();
266 }
267
268 function hitTest(shape, mx, my) {
269
270     var dx;
271     var dy;
272     dx = mx - shape.x;
273     dy = my - shape.y;
274
275     return (dx * dx + dy * dy < shape.rad * shape.rad * 4);
276 }
277
278 function drawShapes() {
279     var i;
280     for (i = 0; i < numShapes; i++) {
281         // cxt.fillStyle = shapes[i].color;
282         //不能用fill, 所以把线宽设定为半径的两倍。。
283         cxt.beginPath();
284         cxt.arc(shapes[i].x, shapes[i].y, shapes[i].rad, 0, 2 * Math.PI, false);
285         cxt.closePath();
286         cxt.lineWidth = shapes[i].rad * 2;
287         cxt.strokeStyle = 'red';
288         cxt.stroke();
289     }
290 }
291 }
292
```