基本动作使用（0.1）

var mapoper = {

zoomIn: function(){

map.zoomIn();

},

zoomOut: function(){

map.zoomOut();

},

panLeft: function(){

map.panBy([256,0]);

},

panRight: function(){

map.panBy([-256,0]);

},

panTop: function(){

map.panBy([0,256]);

},

panBottom: function(){

map.panBy([0,-256]);

},

restore: function(){

map.setZoom(10);

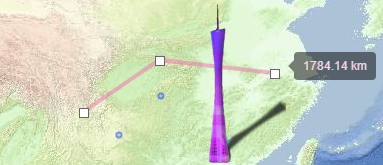
map.panTo([21.999, 113.660]);

}

}  
map是实例化的地图，panBy实现上下左右的像素移动，这个包含了地图缩放工具栏所需要的功能点

# 测量层的使用（0.7）

var measure = new L.Polyline.Measure(map);  
measure.enable();

  
measure.disable();

# 旋转标记的使用（0.7）

//现在旋转标记的旋转点在图片中心

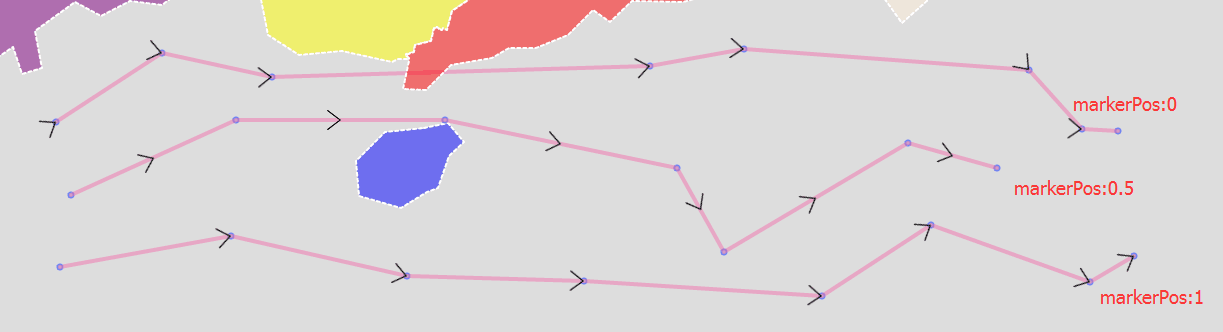
var icon = {iconUrl:'/js/tool/zmap/images/marker-arrow.png', iconSize:[20,28], iconAnchor:[10,14]};

var angleMarker = L.angleMarker(latlng, {icon: new L.Icon(icon), iconAngle: angle});

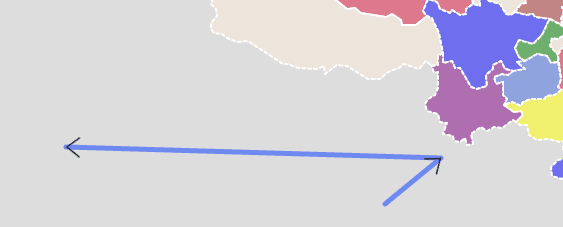
angleMarker.addTo(map);

map.removeLayer(angleMarker);

# 手绘标记多段线的使用

逃生路线与之前不同的在于它是线与标记的组合，我起了一个新的type为markerline，  
这个可以做飞机航线、轮船航线、步行路线等不同图标的方向指示多段线  
var escapeline = new L.Draw.Markerline(map, {markerIcon: L.icon({iconUrl:'/js/tool/zmap/images/marker-arrow.png', iconSize:[20,28], iconAnchor:[10,14]}), markerPos: 1})  
escapeline.enable();  
ask:  
第一条的角度好像有问题  
answer:

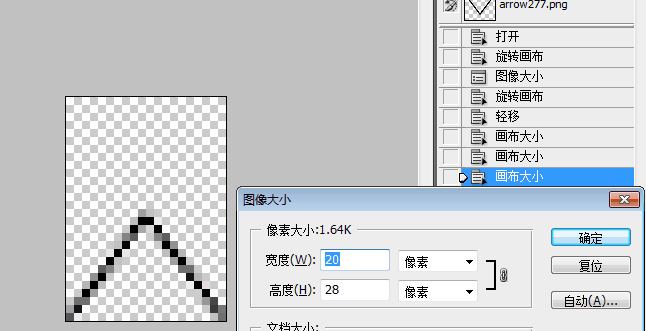
没有问题,它是以线段的0%处开始画,每一个箭头与视觉上的下一条线段匹配

标记多段线的使用  
var escapleline = L.markerline([[16,94],[20,99],[21,66]], {markerIcon:L.icon({iconUrl:'/js/tool/zmap/images/marker-arrow.png', iconSize:[20,28], iconAnchor:[10,14]}), markerPos:1});  
escapleline.addTo(map);  
map.removeLayer(escapleline);  
  
与L.Draw.markerline一样，拥有markerIcon与markerPos两个属性需要设定，线条样式可以按自己需要设定  
L.Draw.Markerline会直接以L.Markerline的方式传出created事件，可以不做任何处理就自动加上路线与图标  
map.on('draw:created', function (e) {  
var type = e.layerType,  
layer = e.layer;  
if (type === 'marker') {  
// Do marker specific actions  
}else if (type === 'markerline'){  
// Do marker line actions  
}  
// Do whatever else you need to. (save to db, add to map etc)  
drawnItems.addLayer(layer);  
});  
  
  
L.Draw.Markerline与L.Markerline这两个类的markerPos不传的时候，默然0.5，markerIcon不传的时候，默然是那个黑色箭头

对于自定义的markerIcon，实际上是实例化了L.AngleMarker，有以下要求：

1、图片的旋转点在中心位置，可能图片需要做处理将定位点移到中心位置

2、图片的0度为朝上，所有图片需要以朝上的姿态作为初始状态



# 风场层的使用（0.7）

## 格点风场添加

var us = [[10,15,-5],[6,-10,7],[-3,7,8]]

var vs = [[6,5,3],[1,-7,9],[-6,-10,8]]

var gridInfo = {"top":40,"bottom":20,"left":110,"right":130,xgap:10, ygap:10, xdim:3, ydim:3}

var type = 'line'; //circle,arrow,wave

var windlayer=L.windLayer(us, vs, gridInfo, 'circle');

windlayer.addTo(map);

## 格点风场插值

风场插值任意经纬的值，如果落在模式范围外返回为空

var pointWind = windlayer.interp(120, 30);

# 离散点风场（1.2）

## 离散点风场添加

var gps = [{x:113.103749,y:22.860310,dir:25,vel:8.8},{x:113.247590,y:22.958820,dir:354,vel:9.4},{x:113.244698,y:22.848431,dir:44,vel:7.5},{x:113.337178,y:22.805262,dir:231,vel:7.0},{x:113.286409,y:22.883030,dir:194,vel:9.0},{x:113.156120,y:22.976888,dir:236,vel:7.0},{x:113.091316,y:22.977097,dir:183,vel:7.5},{x:113.116605,y:22.783421,dir:354,vel:8.1},{x:113.245382,y:22.751382,dir:90,vel:7.4},{x:113.146950,y:22.728616,dir:326,vel:9.9},{x:113.159897,y:22.926778,dir:138,vel:8.9},{x:113.162115,y:22.837693,dir:340,vel:9.3},{x:113.070345,y:22.815547,dir:169,vel:3.8},{x:113.292719,y:22.809430,dir:346,vel:7.1},{x:113.335294,y:22.760696,dir:118,vel:6.0},{x:113.191313,y:22.890526,dir:0,vel:8.5},{x:113.248962,y:22.918163,dir:208,vel:7.5},{x:113.098611,y:22.974722,dir:352,vel:10.1},{x:113.131667,y:22.959444,dir:51,vel:4.7},{x:113.265108,y:22.828927,dir:0,vel:8.7},{x:113.256569,y:22.790836,dir:6,vel:8.2},{x:113.200328,y:22.867369,dir:293,vel:8.6},{x:113.228497,y:22.872448,dir:34,vel:6.5},{x:113.211681,y:22.805555,dir:354,vel:8.4},{x:113.138614,y:22.865188,dir:17,vel:8.7},{x:113.069556,y:22.859829,dir:3,vel:7.9},{x:113.034821,y:22.850600,dir:326,vel:8.8},{x:113.105108,y:22.901234,dir:6,vel:5.5},{x:113.112888,y:22.956696,dir:17,vel:7.2},{x:113.109000,y:22.719500,dir:39,vel:6.1},{x:113.207174,y:22.723134,dir:349,vel:4.9},{x:113.364344,y:22.812909,dir:25,vel:7.8},{x:113.293196,y:22.782129,dir:357,vel:8.0},{x:113.205421,y:22.930132,dir:357,vel:6.8},{x:113.256198,y:22.946616,dir:31,vel:8.2},{x:113.188737,y:22.966917,dir:42,vel:0.0},{x:113.198690,y:22.986962,dir:312,vel:8.6},{x:113.133128,y:22.816656,dir:340,vel:6.0},{x:113.163333,y:22.791667,dir:3,vel:7.5},{x:113.070345,y:22.815547,dir:0,vel:0.0},{x:113.194722,y:22.965833,dir:236,vel:9.3},{x:113.293611,y:22.850506,dir:236,vel:7.5},{x:113.111944,y:22.915556,dir:87,vel:9.1},{x:113.280024,y:22.824304,dir:338,vel:7.7}];

var layerWindDis = L.windLayer(gps, 'line');

layerWindDis.addTo(map);

## 离散点风场插值

风场插值任意经纬的值，如果落在模式范围外返回为空

var pointWind = windlayer.interp(120, 30);

# 业务层的使用（0.7）

## 业务动画添加

var urls = [

'http://10.148.10.80:8097/images/home/img1.jpg',

'http://10.148.10.80:8097/images/home/img2.jpg',

'http://10.148.10.80:8097/images/home/img3.jpg',

'http://10.148.10.80:8097/images/home/img4.jpg'

];

//loop:true为死循环动画,loop:false只动画一次

var busilayer = L.businessOverlay(urls, { loop:true,time:500, left:110,right:115,top:23,bottom:22});

//调用之后，自动产生动画

busilayer.addTo(map, function(index, url){

console.info("picture["+index+"] is ok, url is:"+url);

});

//动画控制

busilayer.startAnimate();

busilayer.stopAnimate();

## 业务区域添加

var busilayer = L.businessOverlay('https://ss0.bdstatic.com/5aV1bjqh\_Q23odCf/static/superman/img/logo/bd\_logo1\_31bdc765.png', {left:110,right:115,top:23,bottom:22});

busilayer.addTo(map);

busilayer.setUrl('http://10.148.10.80:8097/images/home/img3.jpg')

## 业务全屏添加

var busilayer = L.businessOverlay('https://ss0.bdstatic.com/5aV1bjqh\_Q23odCf/static/superman/img/logo/bd\_logo1\_31bdc765.png');

busilayer.addTo(map);

busilayer.setUrlAutoFit('http://10.148.10.80:8097/images/home/img3.jpg')

## 图片请求完毕回调

初始添加的回调  
busilayer.addTo(map, function(url){

console.info("load ok:"+url)

});

设置新图片的回调

busilayer.setUrl('http://10.148.10.80:8097/images/home/img3.jpg', function(url){

console.info("load ok:"+url)

} )

busilayer.setUrlAutoFit('http://10.148.10.80:8097/images/home/img3.jpg', function(url){

console.info("load ok:"+url)

})

# 填色层的使用（1.2）

## 格点填色层添加

var grid = [[10,15,-5],[6,-10,7],[-3,7,8]]

var gridInfo = {"top":40,"bottom":20,"left":110,"right":130,xgap:10, ygap:10, xdim:3, ydim:3}

//填色支持色表方式1

var colortable = {colors:['rgba(100,100,0,0.7)', 'rgba(120,80,0,0.7)', 'rgba(140,60,0,0.7)', 'rgba(160,60,0,0.7)', 'rgba(180,40,0,0.7)', 'rgba(200,20,0,0.7)', 'rgba(240,0,0,0.7)'], startVal:-10, stepVal:3};

//填色支持色表方式2

var colortable = [{color:'rgba(0,100,0,0.7)',value:-8}, {color:'rgba(20,80,0,0.7)',value:-2}, {color:'rgba(140,60,0,0.7)',value:5}, {color:'rgba(160,60,0,0.7)',value:10}];

var shadeLayer = L.shadeLayer(grid, gridInfo, colortable);

shadeLayer.addTo(map) ;

## 格点填色层插值

格点填色层插值任意经纬的值，如果落在模式范围外返回为空

var pointVal = shadeLayer.interp(120, 30);

## 格点填色层裁剪

var grid = [[10,15,-5],[6,-10,7],[-3,7,8]]

var gridInfo = {"top":26,"bottom":20,"left":108,"right":120,xgap:6, ygap:3, xdim:3, ydim:3}

var colortable = [{color:'rgba(0,100,0,0.7)',value:-8}, {color:'rgba(20,80,0,0.7)',value:-2}, {color:'rgba(140,60,0,0.7)',value:5}, {color:'rgba(160,60,0,0.7)',value:10}];

var shadeLayer = L.shadeLayer(grid, gridInfo, colortable, L.Util.Bounds.SimpleGuangDong, {debug:false});

//或者

var shadeLayer = L.shadeLayer(grid, gridInfo, colortable, L.Util.Bounds.SimpleGuangDong)

//或者

var shadeLayer = L.shadeLayer(grid, gridInfo, colortable)

shadeLayer.addTo(map)

## 离散点填色层添加

var gps = [{x:113.103749,y:22.860310,val:27.8}, {x:113.247590,y:22.958820,val:28.9}, {x:113.244698,y:22.848431,val:29.4},{x:113.337178,y:22.805262,val:32.4},{x:113.286409,y:22.883030,val:28.7},{x:113.156120,y:22.976888,val:28.9},{x:113.116605,y:22.783421,val:28.7},{x:113.245382,y:22.751382,val:30.3},{x:113.146950,y:22.728616,val:27.5},{x:113.159897,y:22.926778,val:29.6},{x:113.162115,y:22.837693,val:28.9},{x:113.070345,y:22.815547,val:28.9},{x:113.292719,y:22.809430,val:30.5},{x:113.335294,y:22.760696,val:29.8},{x:113.191313,y:22.890526,val:29.3},{x:113.248962,y:22.918163,val:29.7},{x:113.098611,y:22.974722,val:27.4},{x:113.131667,y:22.959444,val:28.7},{x:113.256569,y:22.790836,val:28.9},{x:113.200328,y:22.867369,val:29.3},{x:113.228497,y:22.872448,val:29.6},{x:113.211681,y:22.805555,val:29.0},{x:113.138614,y:22.865188,val:28.3},{x:113.069556,y:22.859829,val:27.9},{x:113.034821,y:22.850600,val:29.0},{x:113.105108,y:22.901234,val:29.3},{x:113.112888,y:22.956696,val:28.1},{x:113.109000,y:22.719500,val:28.1},{x:113.207174,y:22.723134,val:28.9},{x:113.293196,y:22.782129,val:30.8},{x:113.205421,y:22.930132,val:29.1},{x:113.256198,y:22.946616,val:29.8},{x:113.188737,y:22.966917,val:28.8},{x:113.198690,y:22.986962,val:28.3},{x:113.133128,y:22.816656,val:29.1},{x:113.163333,y:22.791667,val:28.4},{x:113.070345,y:22.815547,val:32.8},{x:113.194722,y:22.965833,val:29.8},{x:113.293611,y:22.850506,val:28.6},{x:113.111944,y:22.915556,val:28.7},{x:113.280024,y:22.824304,val:29.5}];

//填色支持色表方式1

var colortable = {colors:['rgba(100,100,0,0.7)', 'rgba(120,80,0,0.7)', 'rgba(140,60,0,0.7)', 'rgba(160,60,0,0.7)', 'rgba(180,40,0,0.7)', 'rgba(200,20,0,0.7)', 'rgba(240,0,0,0.7)'], startVal:26, stepVal:0.5};

//填色支持色表方式2

var colortable = [{color:'rgba(0,100,0,0.7)',value:26}, {color:'rgba(20,80,0,0.7)',value:27}, {color:'rgba(140,60,0,0.7)',value:28}, {color:'rgba(160,60,0,0.7)',value:29}];

var shadeLayer = L.shadeLayer(gps, colortable);

shadeLayer.addTo(map) ;

## 离散点填色层插值

离散点填色层插值任意经纬的值，如果落在模式范围外返回为空

var pointVal = shadeLayer.interp(120, 30);

## 离散填色层裁剪

var shadeLayer = L.shadeLayer(gps, colortable, L.Util.Bounds.SimpleGuangDong, {debug:false});

//或者

var shadeLayer = L.shadeLayer(grid, colortable, L.Util.Bounds.SimpleGuangDong)

//或者

var shadeLayer = L.shadeLayer(gps, colortable)

shadeLayer.addTo(map)

# 固定层的使用(0.7)

## 固定层添加

var fixedlayer = L.fixedLayer('http://10.148.83.228/thunderdecisionindex/static/imgs/no\_thunder\_chapter.png', {size: '250px 100px', position:'50% 50%'});

fixedlayer.addTo(map);

## 固定层调整

fixedlayer.setPosition('100px 100px')

fixedlayer.setPosition('50% 100px')

fixedlayer.setSize('400px 70px')

fixedlayer.setSize('100% 30%')