WebGL 开发手册

目录

一、 简单的示例

1.1 引用 js 库

在<head>标签中添加引用 js 文件的代码，详细代码如下：

```
<script type="text/javascript" src="script/stats.min.js"></script>
<script type="text/javascript" src="script/lsjworker/LSJPWMHelper.min.js"></script>
<script type="text/javascript" src="script/lsjfc.min.js"></script>
<script type="text/javascript" src="script/lsjcode.min.js"></script>
<script type="text/javascript" src="script/TransformControls.js"></script>
<script type="text/javascript" src="script/Projector.js"></script>
<script type="text/javascript" src="script/DragControls.js"></script>
```

1.2 初始化场景并加载数据

在页面的加载完成事件或者<body>标签中添加初始化场景的函数，详细代码如下：

```
<!--初始化场景-->
var container = document.createElement('div');
```

```
document.body.appendChild(container);
initSceneControl(container);
<!--加载数据-->
openLFP("http://192.168.1.101/data / 倾 斜 摄影.json",true);
<!--激活场景-->
animateSceneControl();
```

二、 具体接口

2.1 函数

2.1.1 加载数据

加载单个 **lob** 文件

```
var pageLodNode = new LSJPageLODNode();
pageLodNode.strDataPath ="http://192.168.1.101/data/ /Model/Model.lob";
addFeaturePageLODNode(pageLodNode);
```

加载 **lfp** 文件

```
openLFP("http://192.168.1.101/data /倾斜摄影.lfp");
```

加载 **json** 文件

```
openLFP("http://192.168.1.101/data /倾斜摄影.json",true);
```

2.1.2 场景设置

设置场景的最大、最小倾斜角

```
setControlMaxTilt(60);     //单位：度
setControlMinTilt(0.001);     //单位：度
```

设置双击飞行的速度

```
setControlDoubleClickZoomRatio(0.8);
```

设置缩放时距离模型的最小距离

```
controlMinZoomDist = 20;    //单位：米
```

场景惯性

在<head>标签中添加两个函数即可，详细代码如下：

```
function beginZoom(){
var mouseX = window.document.documentElement.clientWidth/2;
var mouseY = window.document.documentElement.clientHeight/2;
beginMomentumZoomScreen(mouseX,mouseY,0.01);
}
```

```
function endZoom(){
    stopSceneMomentum();
}
```

2.1.3 场景常用操作

场景倾斜

```
var mouseX = window.document.documentElement.clientWidth/2;
var mouseY = window.document.documentElement.clientHeight/2;
inertiaPitchScreen(mouseX,mouseY,22.5);
```

说明：

mouseX、mouseY 代表的是场景倾斜时的中心点的 x、y 坐标；

22.5 代表的是以当前倾斜角为基础向下倾斜的角度，单位：度，正数代表向下倾斜，负数代表向上倾斜；

场景旋转

```
var mouseX = window.document.documentElement.clientWidth/2;
var mouseY = window.document.documentElement.clientHeight/2;
inertiaRollScreen (mouseX,mouseY,-0.5);
```

说明：

mouseX、mouseY 代表的是场景旋转时的中心点的 x、y 坐标；

-0.5 代表的是以当前位置为基础顺时针旋转的角度，单位：度，正数代表逆时针旋转，负数代表顺时针旋转；

场景缩放

```
var mouseX = window.document.documentElement.clientWidth/2;
var mouseY = window.document.documentElement.clientHeight/2;
inertiaZoomScreen (mouseX,mouseY,0.5);
```

说明：

mouseX、mouseY 代表的是场景缩放时的中心点的 x、y 坐标；

0.5 代表的是以当前位置为基础放大的弧度，单位：弧度，正数代表放大，负数代表缩小；

屏幕坐标转场景坐标

```
var mouseX = window.document.documentElement.clientWidth/2;
var mouseY = window.document.documentElement.clientHeight/2;
var center = screenToScene(mouseX,mouseY);
```

2.1.4 飞行

定位

```
var initPosition=newTHREE.Vector3(-3629.74587406,-978.784402266,3010.651689);
var initQuaternion=newTHREE.Quaternion(0.3963012,-0.272445,-0.4966985,0.722502279);
var euler = new THREE.Euler();
```

```
euler.setFromQuaternion(initQuaternion,"XYZ",true);
getFlyToCameraControls().flyTo(initPosition,euler.x,euler.y,euler.z,10*1000);
```

说明：

-3629.74587406,-978.784402266,3010.651689:场景坐标 x、y、z,相当于经纬度高程；

0.3963012,-0.272445,-0.4966985,0.722502279：场景 x、y、z、w 方向的倾斜度数；

停止定位飞行

```
getFlyToCameraControls().stop();
```

绕点旋转

```
var mouseX = window.document.documentElement.clientWidth/2;
var mouseY = window.document.documentElement.clientHeight/2;
var center = screenToScene(mouseX,mouseY);
flyAroundPosition(center,-0.5);
```

说明：

center：场景旋转中心点

-0.5：代表顺时针一直旋转，0.5 代表速度，正数代表逆时针旋转，负数代表顺时针旋转；

停止旋转飞行

```
getFlyAroundCenterControls().stop();
```

沿线飞行

```
getFlyWithLineControls().attach(curve);
```

说明：

curve 是线对象，代表飞行的路线；

停止沿线飞行

```
getFlyWithLineControls().detach();
```

2.1.5 添加对象

添加标注对象

```
var layers =    getLayers();                    //图层集合
    var layer = new LSJLayer();                 //图层
    var geometry = new LSJGeoMarker();    //标注对象
    var style = new LSJMarkerStyle();         //标注风格
    style.iconPath = "resource/image/marker.png";      //标注图标的路径
    style.iconSize = 48;                              //标注图标大小
  var textStyle = style.getTextStyle();     //文字风格
    textStyle.setFontName('STCaiyun');     //文字类型
    textStyle.getFillColor().setRGB(1,0,0);    //文字颜色
```

```
    geometry.setStyle(style);
  geometry.setName('我的标注');       //标注内容
    geometry.setPosition(-52.047264099121094,29.460622787475586,2
65.41644287109375);
    //标注的位置，分别是经度、纬度、高程
    layer.addGeometry(geometry);       //添加标注对象到图层
    layers.addLayer(layer);                  //添加图层对象到图层集合
```

添加线对象

```
   var geometry = new THREE.Geometry();
    for ( var i = 0; i < 200; i ++ ) {
        geometry.vertices.push( new THREE.Vector3() );
    }
    var curve = new THREE.CatmullRomCurve3();      //线对象
    curve.type = 'catmullrom';
    curve.mesh = new THREE.Line( geometry.clone(), new
THREE.LineBasicMaterial( {
        color: 0xff0000,                          //线颜色
        opacity: 0.35,                            //线透明度
        linewidth: 2                              //线宽度
    } ) );
    curve.mesh.castShadow = true;
    curve.mesh.visible = false;                   //线是否显示
    getScene().add( curve.mesh );
```

```
var editorLineControls = getEditorLineControls();
editorLineControls.attach(curve);
editorLineControls.addPoint(new THREE.Vector3(-100.047264099121094,10.460622787475586,300.41644287109375));
editorLineControls.addPoint(new THREE.Vector3(-52.047264099121094+30,40.460622787475586,285.41644287109375));
editorLineControls.addPoint(new THREE.Vector3(-52.047264099121094+30,40.460622787475586+20,285.41644287109375));
editorLineControls.addPoint(new THREE.Vector3(-52.047264099121094,50.460622787475586,285.41644287109375));
editorLineControls.setMode("none");
```

2.2 事件

事件名称不能更改，可以更改事件的实现

2.2.1 对象单击事件

```
function onControlFeatureClick(mouse,intersect,object){

}
```

2.2.2 场景单击事件

PC 端

```
function onControlPageLODClick(mouse,intersect){

}
```

移动端

```
function onCustomTouchEnd(event){

}
```