**三维文档**

# 三维主要文件目录结构详解

## BIM3D

项目主文件

## Bim3Djs

三维项目外部引入的各种依赖文件例如：three.js、JQuery.mini.js、viewer3D.js等

## Extension

Extension是存放所需要添加的控件文件

## JS

### Controls

文件夹内存放three.js的两个扩展包：OrbitControls.js、TrackballControls.js

1. OrbitControls.js

three.js利用OrbitControls插件（轨道控制）控制模型交互动作

1. TrackballControls.js

Three.js利用TrackballControls插件实现对场景的操控

### Event

文件夹内存放了一些扩展API

1. AddIcon.js

在初始化的时候调用addIcon()加载模型标记

1. ClickAddSensor.js

点击添加传感器的扩展API

1. Svg.js

设置模型标记的API

### Init

1. InitSensor.js

初始化传感器，并且根据照相机跟传感器的距离实时计算大小的API

### Introduce

项目依赖包

# 三维主要功能及实现流程

## 本地模型加载

本地模型加载的功能在入口文件index.html内写入

1. 定义viewer基本配置：

config = {

extensions:[] //自定义的扩展API声明的地方

disabledExtensions:{}

}

1. 声明viewer的作用的DOM对象

var element = document.getElementbyID();

1. 实例化viewer对象

Var viewer = new Autodesk.Viewing.Private.GuiViewer3D()

传入参数：element、config

1. 配置viewer的初始化的选项options
2. 调用Initializer() viewer的初始化方法

传入参数：options、callback

1. 在callback函数内

Viewer.start（）;

调用viewer.start（）

传入参数：url、sharePropertyDbPath、succerrcallback、errorcallback

## 用户自定义切换模型

用户自定义切换模型的功能在入口文件index.html内写入

1. 通过Post请求，获取所有的模型数据，并且循环出模型列表
2. 在每个列表的上面绑定click事件，触发loadModel函数
3. 构造loadModel()函数
4. 设置该模型的Path
5. 调用viewer.impl.unloadCurrentModel()
6. 重新调用viewer.load()加载模型
7. 把当前模型额相关信息转存到新建的对象中

## 三维坐标到二维坐标转换

调用viewer.wordToClient()API，实现三维坐标到二维坐标的转换，此功能使用在Svg.js中，即在调用initSvg方法的时候，把传入的三维坐标转换成屏幕坐标后，就可以调用此屏幕坐标创建svg和label的正确位置

## 导入模型标记

设置模型标记的功能在入口文件index.html内写入

1. 把svgContent节点通过viewer.container.appendChild()的方法插入viewer.container中
2. 获取要标记的模型三维坐标
3. 从数据中读取出模型的msg、color、svg的序列号、infowindow等信息
4. 实例化之前创建好的svg对象
5. 调用initSvg的方法，传入三维坐标、viewer对象、infowindow参
6. 数，来创建模型标记
7. 把创建好的所有的svg都push到svgSensorList数组中，用来存储

## 清空模型标记

清空模型标记的功能在入口文件index.html内写入

1. 获取svgContent节点的所有子节点
2. 用for循环，循环svgContent的所有子节点，让每个子节点都添加上display属性
3. 判断如果其display的值为none则替换成block，反之如此

## 标记数据存储

标记数据存储的功能在入口文件index.html内写入

1. 创建setPosition()的构造函数，传入标记所在模型的ID和模型的pos值（存储三维坐标的对象）
2. 在函数内统一JSON对象的格式
3. 调用父窗体的setPositio()方法

## 添加模型标记

添加模型标记的功能在AddIcon.js

1. 给viewer添加API扩展
2. 通过hitTest方法获取用户点击的屏幕坐标，并且进行存储
3. 用THREE.TextureLoader().load()方法在此坐标处调用three的方法，创建小几何体然后给几何体贴上指定的图标
4. 并且在THREE.TextureLoader().load()方法内调用THREE的lookAT()方法，使镜头看向目前的添加图标的地方，即完成了镜头的移动
5. 再在ADDIcon的原型链上添加load方法，即给viewer.container绑定一个单机事件去执行hitTest这个方法

## 移动控件位置

移动控件位置的功能在文件svg.js内写入

1. 创建divDrag()的构造函数，传入id和id2
2. 给id2节点元素添加一个点击（松开后触发）事件
3. 通过计算分别获取控件离左上角额位置\_x和\_y
4. 给id的节点元素添加fadeTo方法使其点击后开始拖动并且透明显示
5. 监听鼠标移动事件，计算出控件距离左上角的绝对位置x,y
6. 给id的节点元素通过设置其left:x;top:y来设置控件的新位置
7. 添加鼠标抬起事件，即id节点元素停止移动且恢复成不透明

## 点击选择构件

1. 给viewer.container对象绑定单机事件
2. 通过viewer.getSelection()方法获取所点击构件的dbid，返回的是一个数组
3. viewer.getSelection()[0]即使所选构件的唯一识别码

## 获取构件详细信息

1. 通过getSelection()获取到构件的dbid
2. viewer.getProperties(dbid,function(res){console.log(res.properties)},function(err,msg){console.error(err,msg)})的方法就可以得到当前构件的详细信息

## 完全隐藏构件

1. 通过getSelection()获取到构件的dbid
2. 调用viewer.hide(dbid)方法
3. 调用viewer.impl.visibilityManager.setNodeOff(dbid,true)方法

## 显示隐藏构件

1. 通过getSelection()获取到构件的dbid
2. 调用viewer.show(dbid)方法
3. 调用viewer.impl.visibilityManager.setNodeOff(dbid,false)方法

## 鼠标移动到模型构件上不高亮当前构件

1. NOP\_VIEWER.impl.disableRollover(false)

## 鼠标点击改变模型构件颜色

1. 给viewer绑定单模型下鼠标单击事件
2. 调用THREE.Vector4（）设置构件颜色
3. 调用viewer.setThemingColor(dbId,color)就可以改变选中构件的颜色

添加按钮及改变颜色

https://forge.autodesk.com/blog/happy-easter-setthemingcolor-model-material

## 清除模型构件颜色

1. 调用viewer.clearThemingColors()

## 镜头移动

镜头移动的功能在入口文件index.html内写入

1. 创建selectModel()的构造函数，传入标记所在模型的ID
2. 调用viewer.select()方法通过模型ID选择模型
3. 调用viewer.fitToView()方法通过模型ID获取当前控件的最佳视觉

## 撰写viewer扩展方法

继承Autodesk.Viewing.Extension编写自己的扩展代码，再透过viewer3D.函数名Extension这个函数就可以了，卸载也要调用这个函数

例：

Class ChangeColor extends Autodesk.Viewing.Extension{

constructor(viewer,options){

super(viewer,options);

}

load(){

this.viewer.setBackgroundColor(255,0,0,255,255,255);

return true;

}

unload(){

//将颜色改回来viewer3D自带

this.viwer.setBackgroundColor(160,176,184,190,207,216);

return true;

}

}

Autodesk.Viewing.theExtensionManager.registerExtension(‘Demo’,ChangeColor )

//调用

viewer.loadExtension(‘Demo’,null);

//卸载

viewer.unloadExtension(‘Demo’,null)

## 添加Button控件

调用createButton()方法，直接 var btn = createButton()其中传入的参数：

1. ID：Button的ID
2. Url：Button的Icon的地址
3. String：当鼠标移到Button上面时显示的提示文字
4. Function：鼠标点击Button后所触发的函数

## 添加单选控件组

1. 撰写viewer扩展格式
2. 调用getToolbar（）创建一个外部盒子
3. 在外部盒子里面调用getControl()方法，返回当前所在控制组（盒子）的信息
4. 实例化一个Autodesk.Viewing.UI.RadioButtonGroup()新对象
5. 给该对象调用addclass()添加一个类名
6. 创建若干Button
7. 调用addControl()将Button添加到该对象中
8. 将该对象通过addControl()添加到第三步的返回值中

## 添加自定义上拉小菜单

1. 撰写viewer扩展格式
2. 调用getToolbar（）创建一个外部盒子
3. 在外部盒子里面调用getControl()方法，返回当前所在控制组（盒子）的信息
4. 实例化一个Autodesk.Viewing.UI.ComboButton新对象
5. 在第四部的对象上调用setToolTip()即当鼠标移到盒子上时，所显示的提示信息
6. 创建若干个Button用addControl()方法插入到第四步的对象中
7. 把第四步的对象通过addControl()方法插入到第三步所返回的值中

## 删除自定义控件

1. 调用removeControl()方法删除各个节点传入的参数为节点ID

例：



## 添加默认的工具列跟ViewCube

调用Autodesk.Viewing.Private.GuiViewer3D

## 添加传感器

添加传感器的功能在ClickAddSensor.js

1. 给viewer添加API扩展
2. 在ClickAddSensor对象里面写入load()方法
3. 在load()方法里面给viewer.container节点绑定点击事件
4. 当点击viewer.container时先调用RemoveModelMarker()函数，在创建下一个标记的时候清除上一个标记点
5. 调用viewer.getProperties()获取dbid的属性。一旦属性返回，该方法就会触发函数去获取当前标记的坐标，并且调用svg函数去添加图标，再调用createSensor()在场景中添加传感器。
6. 调用setSensor()设置传感器的位置

## 提交保存传感器内容

提交保存传感器的内容在ClickAddSensor.js

1. 给submit节点绑定点击事件去执行以下函数
2. 判断tOrf的状态，放置重复提交保存传感器信息
3. 调用setSensorContent()方法
4. setSensorContent()方法中获取sesorId、sensorType、sensorName的节点，判断它们的value值不为空的话，就把它们的value值分别存入json和markerData数组中
5. 把json数据push到jsonArr数组中
6. 把markerData数组push到markerDataArr数组中
7. 调用setPosition()方法把markerData传入该方法
8. 清空sesorId、sensorType、sensorName的value值

# Viewer3D API解析（部分）

## getSelection()

获取点击后的构件dbid，返回的是一个数组

## getSelectionCount()

获取选中构件的数量，返回的是number

## getSelectionVisibility()

返回关于当前选择的可见性的信息

## getSelectionVisibility()

返回关于当前选择的可见性的信息

## search(text,onSuccessCallback,onErrorCallback)

通过构件名称(text)可以检索出对应的dbid

例：

Viewer.search(‘主梁’,function(dbId){

Console.log(dbId)

},function(err,msg){

Console.error(err,msg)

})

## setThemingColor(dbid,color)

改变指定构件的颜色

## setBackgroundColor(color)

改变viewer颜色

## setFocalLength(mm)

设置当前相机的焦距

## clearThemingColors()

清除构件颜色

## resize()

调整viewer的大小

## fitToview(dbId)

摄像头聚焦API

## Autodesk.Viewing.CAMERA\_TRANSITION\_COMPLETED

适用范围：

1. 回到默认视角（HOME）相关的变动
2. 是摄像头聚焦在构件上 viewer.fitToView(dbId)
3. 布满监视整个模型 viewer.fitToViewer()
4. 透过viewer.restoreState()回复摄像头状态（viewport）

例：

viewer.addEventListener(

Autodesk.Viewing.CAMERA\_TRANSITION\_COMPLETED,function(){

console.log(‘摄像头不再移动’)

}

)

Viewer.fitToView(dbId)

当摄像头的Transition完成后就会触发该事件，会让摄像头聚焦到指定的构件上面

## getProperties(dbid,onSuccessCallback,onErrorCallback)

获取一个ID的属性。一旦属性返回，该方法就会引发一个onproperties准备事件。即可以通过此方法获取到该dbid控件的所有详细属性值

例：

viewer.getProperties(dbid,function(res){

console.log(res.properties)

},function(err,msg){

console.error(err,msg)

})

## Autodesk.Viewing.SELECTION\_CHANGED\_EVENT

单模型下鼠标点击与viewer交互行为

例：

viewer.addEventListener(

Autodesk.Viewing.SELECTION\_CHANGED\_EVENT,

function(event){

console.log(event)

}

)

注：没有特别备注例子的，其使用方法都是viewer.xx()