Three.js设备建模文档

# 引言

## 建模目的

目前建模技术已经比较成熟，许多简化便于使用的建模框架都得到了广泛应用。为了提升用户体验，同时使产品更具竞争力，在UNET网管项目中，我们决定多以图形化的界面向用户展示数据与提供交互的功能。

## 建模工具

作为Javascript编写的WebGL第三方免费类库，three.js提供了非常多的3D显示功能，且相比原生WebGL简单易用许多，故建模框架选定了three.js。

## 功能要求

### 复用性

由于UNET网管项目监控的设备众多，类型型号各异，所以要求建模封装能够组件化，实现组件复用。新增设备时，根据库中已有组件组装设备模型实现建模；若新增设备中含有模型库中没有的组件时，只需向库中新增该组件模型，且该新增模型之后也能复用。

### 可调整性

由于要显示模型的网页区域无法保证一致，故要求建模能够根据调用方传入参数调整模型大小与显示位置。

### 交互性

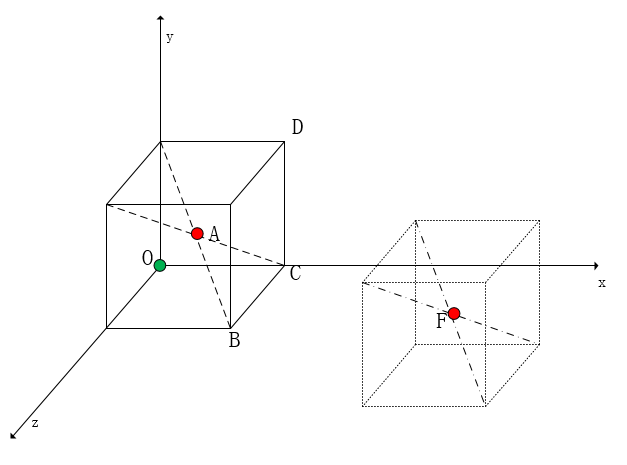
为了提升用户体验，更好地展示数据，要求建模封装提供模型事件监听的接口供调用方使用，调用方可调用监听接口实现自己的事件逻辑。

# 建模详解

## 文件说明

封装文件threeJsCommonUtils.js内封装了three.js各必需组件调用函数，事件监听接口，设备各组件建模函数。需要新增未封装的组件时，只需在该文件中添加该类组件的建模函数即可。使用时引入three.js相关类库文件与该js文件即可。

## 算法说明



如图三维坐标系所示，图中实线立方体为设备外框，同时也是将整个设备包围起来的封闭空间。以该封闭空间左下角O点作为坐标系原点。该设备的中心点A点与封闭空间中心点重合。调用方在建模时，提供参数：

1. 设备的实际宽度realWidth。
2. 设备虚拟宽度virtualWidth，即坐标系中OC线段的长度，实际宽度除以虚拟宽度virtualWidth所得值即为该设备所有组件的统一缩小倍数。
3. 设备虚拟高度virtualHeight，即坐标系中CD线段的长度，virtualHeight乘以设备缩小倍数所得值即为设备实际高度。
4. 设备虚拟深度virtualDepth，即为坐标系中BC线段的长度，virtualDepth乘以设备缩小倍数所得值即为设备实际深度。
5. 设备中心点的X坐标值virtualPositionX，即A点坐标X值。
6. 设备中心点的Y坐标值virtualPositionY，即A点坐标Y值。

若要求模型不绘制在画布正中，调用方可提供设备中心点平移后的坐标值来指定设备在画布中的绘制位置：

1. 设备中心点的Z坐标值virtualPositionZ，即A点坐标Z值。
2. 设备中心平移点的X坐标值centerCoordinateX，即F点坐标X值。
3. 设备中心平移点的Y坐标值centerCoordinateY，即F点坐标Y值。

针对设备上的组件，也需提供组件的外形大小（长、宽、高等），以及组件中心点的坐标值，所有值均是设备缩小后的值，所有组件缩小倍数一致。如此，根据组件本身大小比例、离原点的距离以及缩小倍数，即可还原出真实设备的模型。

## 代码说明

### 设备类

model3D即为封装后的设备模型类，代表建模的整体设备，也即是设备外框。调用者在创建模型时，首先创建model3D对象，并传入建模所需的必需参数，此时所传入的参数与设备外框相关，封装数据等也是针对整个设备的，不包含组成整体设备的其他组件参数与数据。

#### 属性

##### 代码详解

封装了创建设备外框所需的参数，分为必需与非必需参数。必需参数是调用方在建模时必须传入的参数，非必需参数若不传入，则使用默认值。

必需参数解释：

1. canvasId：调用方指定用作绘制画布的html元素id。
2. id：该设备模型的id，由调用方指定后绘制模型，之后可根据该id获取绘制后的模型，为其增添事件监听函数或修改某些业务数据等。
3. type：当前设备的类型，其可选值封装在组件类型库中，详情参考2.2.3。
4. realWidth：调用方指定设备模型的真实宽度。
5. virtualWidth：调用方指定设备模型的虚拟宽度，realWidth除以virtualWidth则为所有模型组件的统一缩小倍数。
6. virtualHeight：调用方指定设备模型的虚拟高度，根据比例关系，该值乘以缩小倍数获得的值即为设备的实际高度。
7. virtualDepth：调用方指定设备模型的虚拟深度，根据比例关系，该值乘以缩小倍数获得的值即为设备的实际深度。

其余非必需参数解释详见threeJsCommonUtils.js文件中注释。

##### 代码展示



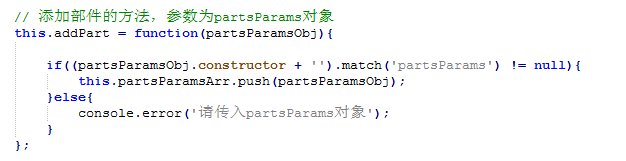
#### 函数

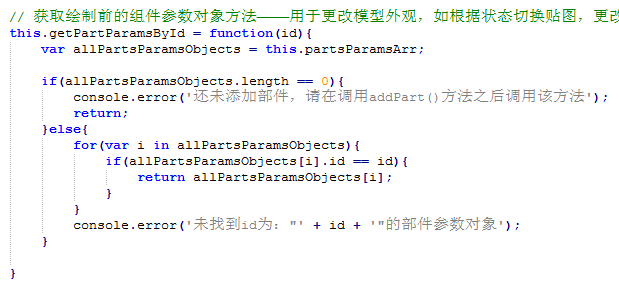
##### 代码详解

对于整体设备而言，提供以下函数：

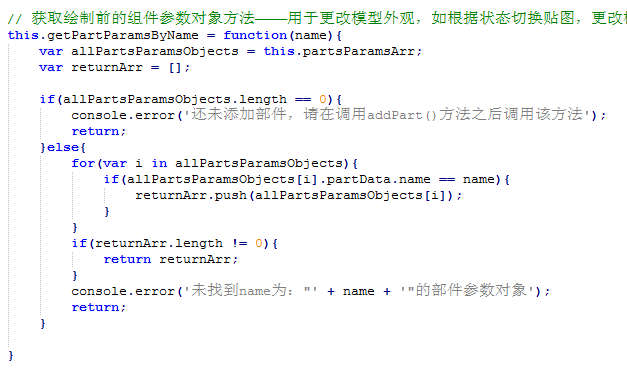
1. addPart(partsParamsObj)：添加部件的函数。参数为partsParams对象。paramsParams类详见2.3.2。
2. getPartParamsById(id)：根据id获取指定部件对象，返回partsParams对象。参数为调用方创建部件模型时传入的id值。
3. getPartParamsByType(type)：根据type获取指定部件对象，返回数组。参数为调用方创建部件模型时传入的type值。
4. getPartParamsByName(name)：根据name获取指定部件对象，返回数组。参数为调用方创建部件模型时封装进业务数据busData中的name值。
5. addEventById(id,eventType,func)：给指定id的部件对象注册监听事件。参数id为部件id，eventType为事件类型，数据类型为字符串，func为执行的事件函数。
6. addEventByType(type,eventType,func)：给指定type的部件对象注册监听事件。参数type为部件type，eventType为事件类型，数据类型为字符串，func为执行的事件函数。

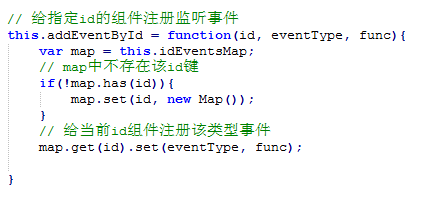
##### 代码展示

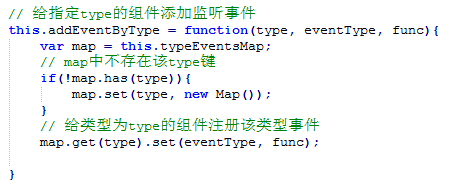












### 组件类

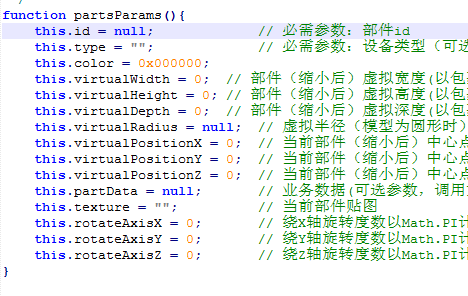
封装后的组件类为partsParams，该类对象代表除设备外框以外的所有组件模型参数对象。例如：交换机的板卡、端口、电源开关、指示灯等部件。调用者在创建部件模型时，需创建partsParams对象，并传入相关参数。

#### 代码详解

组件类只有初始化组件模型需提供的各项属性参数，没有函数。其属性介绍如下：

1. id：部件的id。
2. type：部件类型。
3. color：部件颜色，为十六进制颜色值。
4. virtualWidth：部件虚拟宽度，即缩小指定倍数后的宽度。
5. virtualHeight：部件虚拟高度，即缩小指定倍数后的高度。
6. virtualDepth：部件虚拟深度，即缩小指定倍数后的深度。
7. virtualRadius：部件虚拟半径。（针对圆形部件）
8. virtualPositionX：部件中心点坐标X值。
9. virtualPositionY：部件中心点坐标Y值。
10. virtualPositionZ：部件中心点坐标Z值。
11. partData：当前部件需要绑定的业务数据。
12. texture：当前部件贴图。
13. rotateAxisX：部件绕X轴旋转度数。（如旋转90度，为Math.PI/2）
14. rotateAxisY：部件绕Y轴旋转度数。
15. rotateAxisZ：部件绕Z轴旋转度数。

#### 代码展示

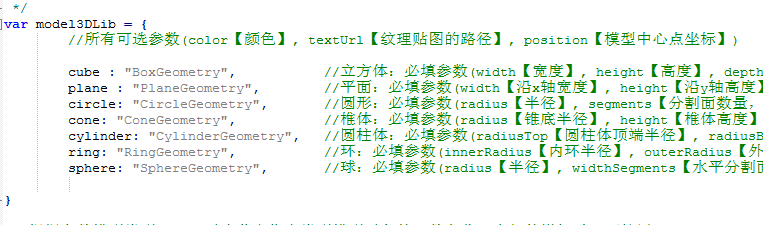


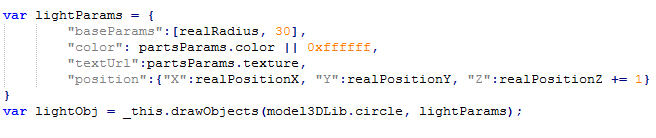
### 模型库

#### 代码详解

封装常用模型形状库，在新增组件建模函数时，调用封装好的模型绘制函数，传入所需模型的形状与需要的参数即可。新增部件模型创建函数时使用，调用者无需关心。

#### 代码展示



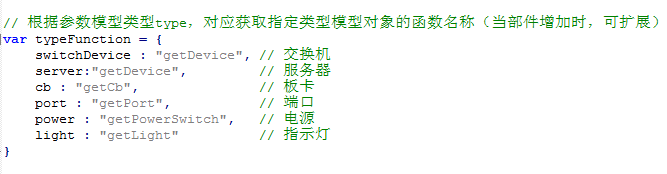


### 组件函数库

#### 代码详解

调用方创建模型时，指定设备type与组件type，绘制模型时执行type对应的函数，即可绘制该类型模型。只在新增库中没有的部件模型时使用，调用方无需关心。

#### 代码展示





### 监听事件

#### 代码详解

给鼠标拾取到的模型部件注册监听事件。形参eventType为监听事件类型，数据类型为字符串，可选值为’click’、’contextmenu’，’dblclick’、’mouseup’、’mouseover’。无需使用方调用。

#### 代码展示



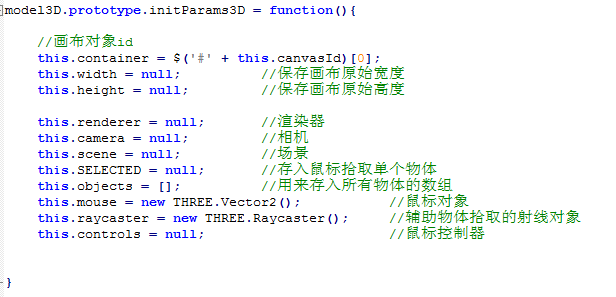
### three.js各组件封装

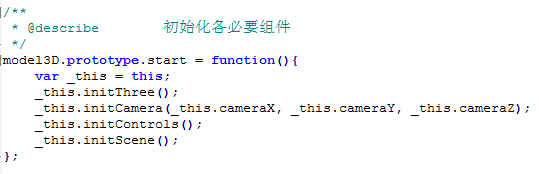
#### 代码详解

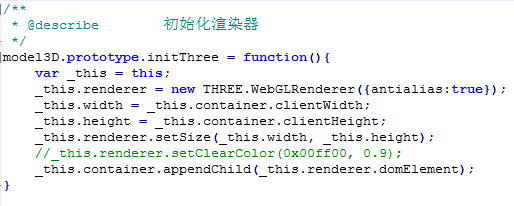
three.js原类库组件相关函数封装如下：

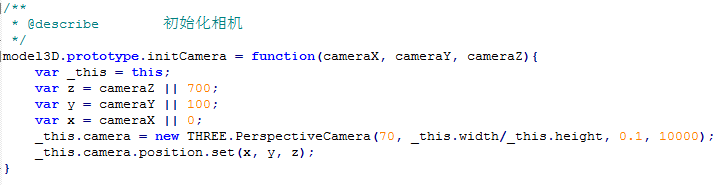
1. initParams3D：声明需要初始化的参数。
2. start：初始化渲染器、相机、鼠标控制器、场景等函数。
3. initThree：初始化渲染器。
4. initCamera(camera,cameraY,cameraZ)：初始化相机。参数分别为相机位置在坐标系中的X、Y、Z坐标值。（可通过传入参数更改相机位置，从而达到从视觉上更改画布中模型大小的目的）
5. initControls：初始化鼠标控制器。如此方可拖动相机实现全方位查看设备模型的功能。
6. initScene：初始化场景。
7. drawObjects(\_objNameStr, \_objParams)：绘制部件模型。\_objNameStr为模型库中的形状对应的three.js模型类，\_objParams为绘制模型时需要传入的json参数。包括baseParams：外形长宽高等；color：部件颜色；textUrl：贴图路径；position：中心点坐标值。
8. bindData(\_objModel, \_data)：给指定部件绑定业务数据。\_objModel为绑定的模型对象，\_data为业务数据。
9. setPosition(\_objMesh, \_objX, \_objY, \_objZ)：设置模型对象的中心点坐标。\_objMesh为指定模型对象，\_objX为X坐标值，\_objY为Y坐标值，\_objZ为Z坐标值。
10. drawBorder(\_objMesh, \_borderColor)：绘制边框。\_objMesh为指定模型对象，\_borderColor为边框颜色。
11. addObjToScene(\_objMesh)：将模型对象添加进场景中。\_objMesh为指定的模型对象。
12. render：渲染循环函数。

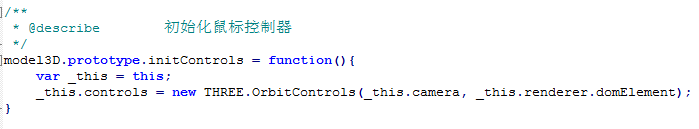
#### 代码展示

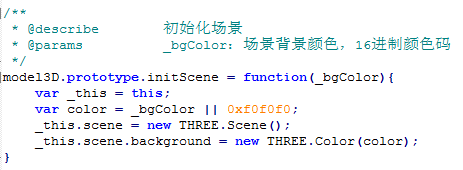




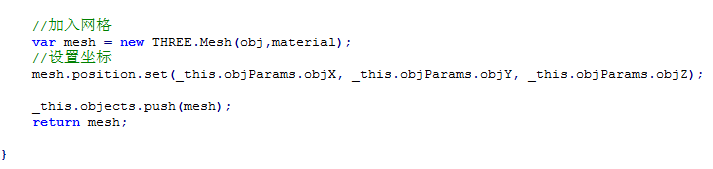


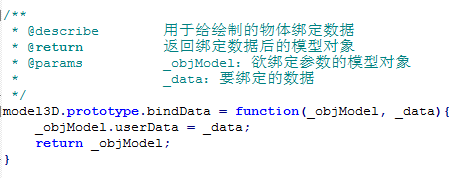


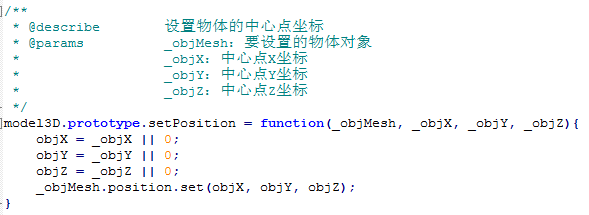


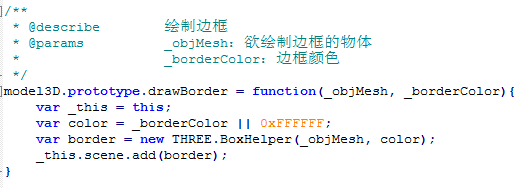


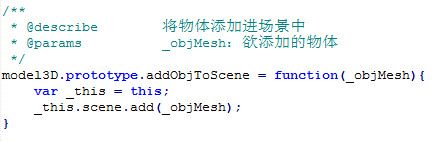


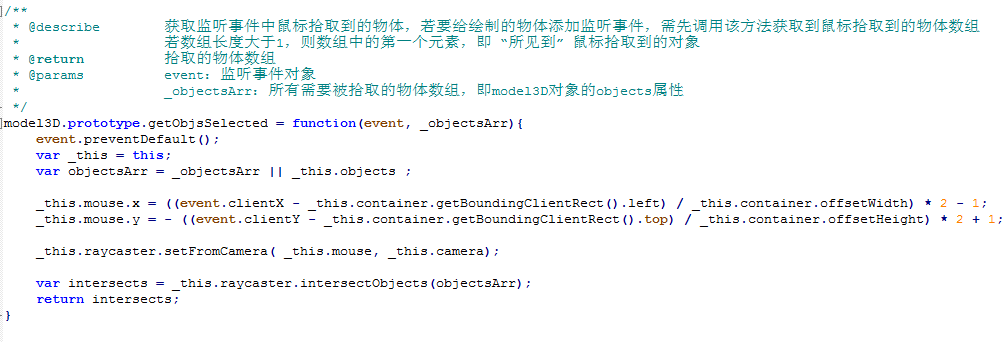


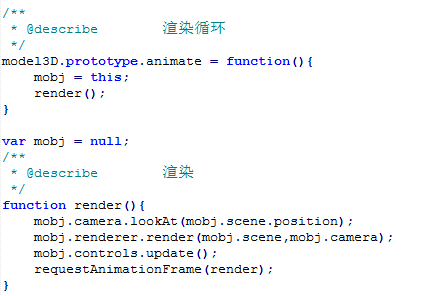












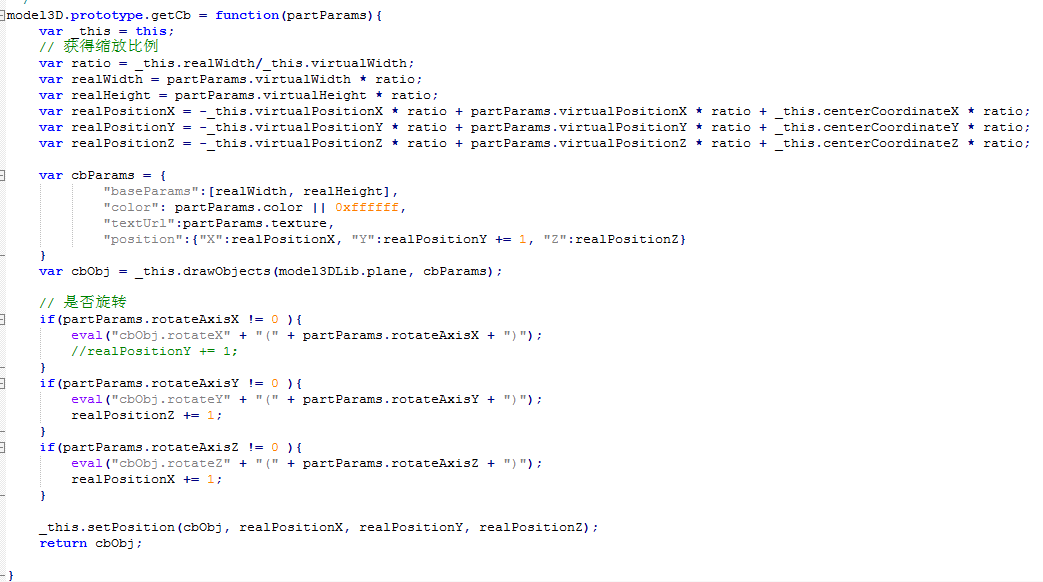
### 组件绘制函数

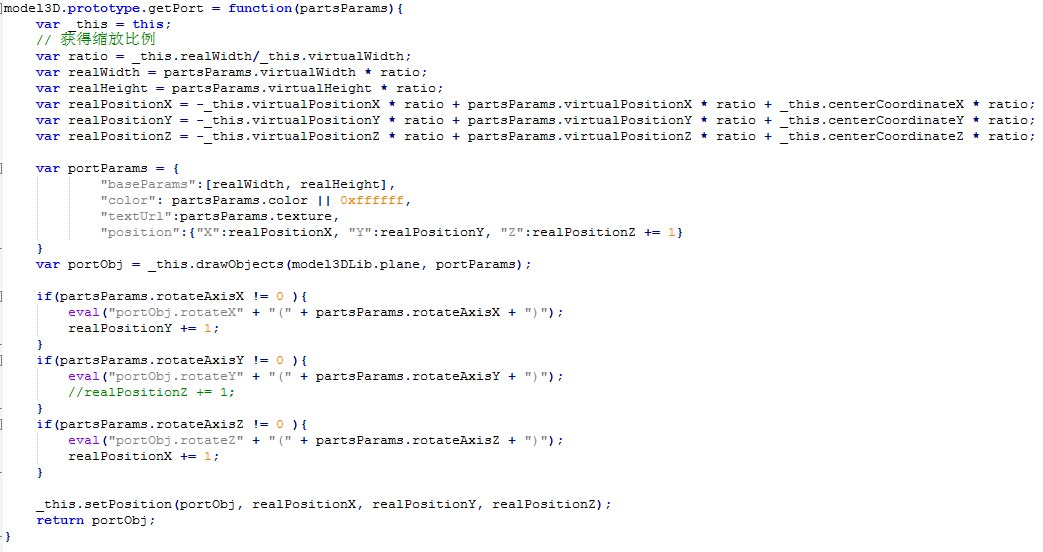
#### 代码详解

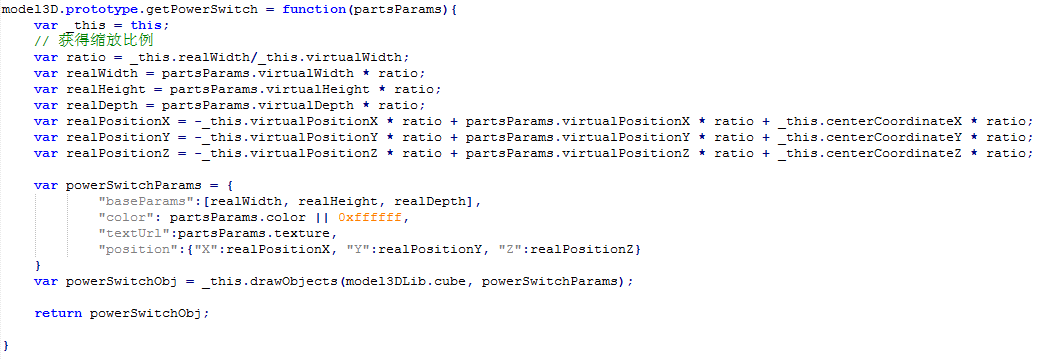
组件绘制函数以组件为单位，返回一类组件模型对象。新增未封装的组件时，可新增该类组件函数。目前封装的组件函数如下：

1. getCb(partParams)：获取交换机的板卡部件。partParams为创建板卡模型需要的json参数。包括：virtualWidth虚拟宽度、virtualHeight虚拟高度、virtualPositionX中心点X坐标值、virtualPositionY中心点Y坐标值、virtualPositionZ中心点Z坐标值、color颜色、texture贴图路径、rotateAxisX绕X轴旋转度数、rotateAxisY绕Y轴旋转度数、rotateAxisZ绕Z轴旋转度数。
2. getPort(partParams)：获取交换机的端口部件。partParams为创建端口模型需要的json参数。
3. getPowerSwitch(partsParams)：获取交换机的电源开关部件。partsParams为创建电源开关模型所需要的json参数。
4. getLight(partsParams)：获取交换机的指示灯部件。partsParams为创建指示灯模型所需要的参数。
5. getDevice()：获取设备外框。该函数创建立方体模型。

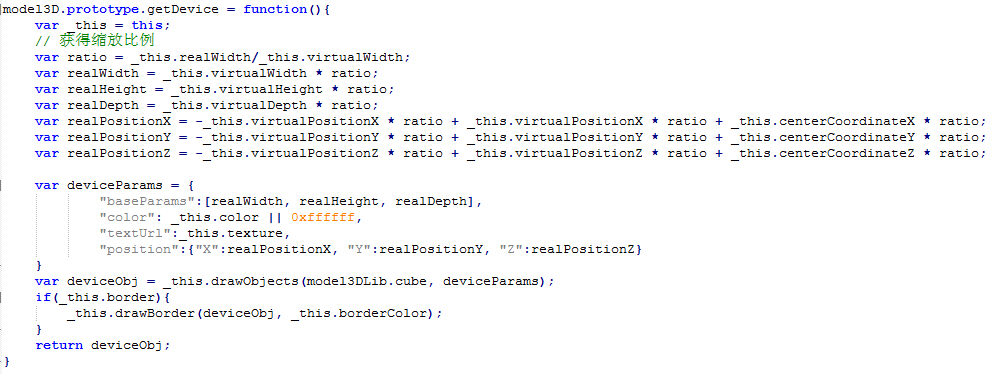
#### 代码展示









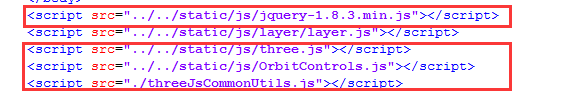


# 使用方式

使用方式可参见threeDemo.html文件中的交换机demo。

## 引入文件

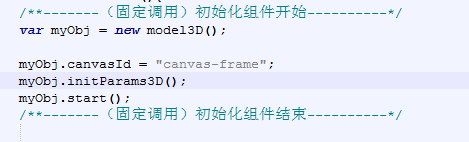
在欲建模的html文档中引入jquery.js、three.js、OrbitControls.js以及threeJsCommonUtils.js文件。



## 初始化组件

给出画布元素，创建设备对象，并指定画布元素id，调用初始化组件的函数。

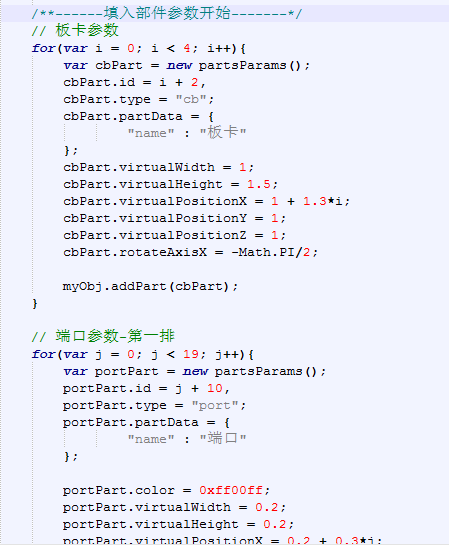


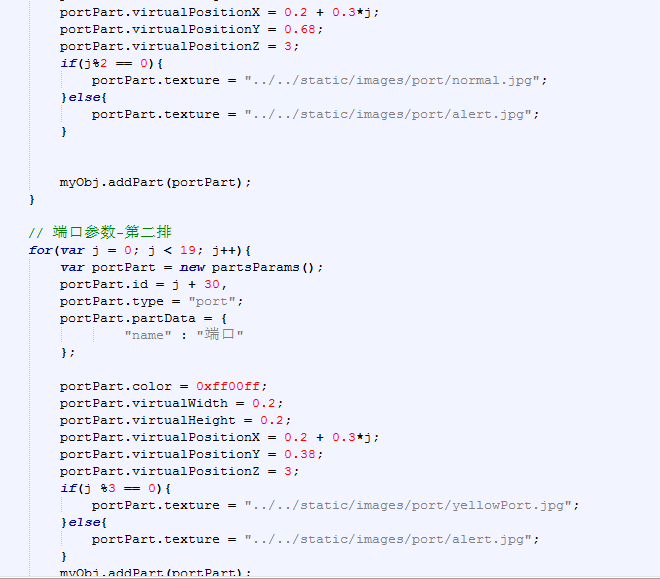


## 填入参数

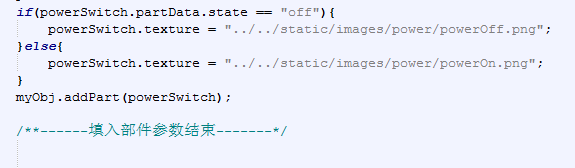
填入参数分为两部分。一部分是设备参数，一部分是组件部分。设备属性参数为model3D对象属性，组件属性参数为partsParams对象属性。





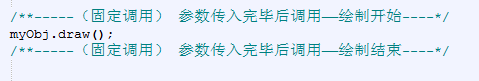






## 调用模型绘制函数

在传入参数完毕之后，即可调用绘制函数绘制模型。



## 添加监听事件

添加事件--事件可在model3D对象创建后的任意位置添加，但对模型外观有影响的事件，应在事件内部处理完逻辑之后，调用redraw()函数，并传入更改了外观的对象，或对象数组，进行重绘。



## 渲染

最后调用渲染函数对组件与模型进行渲染。

