# 用MontageJS创建 3D 应用

创建浏览器3D应用不是轻而易举的。[WebGL](http://www.khronos.org/webgl/)开发了一套针对浏览器3D图形硬件加速的免费插件，它是一套底层图形处理API，对习惯只使用web功能的前端工程师会有比较高的学习门槛。

为了帮助开发者能够以一种简单的方式在浏览器中创建交互式3D应用，MontageJS框架提供一个叫做SceneView的组件。SceneView是一个基于WebGL的3D组件，让开发者可以像操作DOM中的HTML元素一样的简单方式操作3D场景中的元素。

为了让你感受一下这个组件能够做什么（不需要太多的编码）， 在一个支持WebGL的浏览器中打开[Beach Planet 示例](http://montagejs.github.io/beachplanetblog/)。Beach Planet是教程中用来演示3D原理的一个简单解谜游戏。这个游戏的玩法是通过点击不同的位置然后显示隐藏的元素,最后找到MontageJS 元素. 在示例中使用到viewpoints，3D transformations动画，和事件处理。

 ***图 1.*** *找到四个隐藏的元素,真实地体验一下MontageJS的3D组件。*

这个教程讲解MontageJS开发交互式3D应用的基本步骤。大体是这样的：

* 创建一个MontageJS 3D 项目
* 导入一个3D场景到MontageJS项目
* 操作3D场景（使用CSS和绑定）

# 需要的基础知识

要完成这个教程，你需要熟悉MontageJS开发的基础知识。如果你还不熟悉MontageJS框架，你可能需要从[开始使用](http://docs.montagestudio.com/montagejs/montagejs-setup.html)教程开始学习。

同时，这个教程包含详细的示例源码来说明如何使用SceneView组件。在阅读教程的过程中，你可以打开GitHub上面的Beach Planet [源码](https://github.com/montagejs/beachplanetblog)查看；所有的例子都有一个指向源码的链接。当然，你也可以把Beach Planet示例导出然后安装到你本地的机器上，在源码的[readme](https://github.com/montagejs/beachplanetblog)文件中可以查看安装指南。

# SceneView组件介绍

The SceneView component is part of the [mjs-volume](https://github.com/fabrobinet/mjs-volume) module maintained by Fabrice Robinet. The component is designed to help front-end web developers and designers build interactive 3D experiences in the browser using their existing HTML, CSS, and JavaScript skills. Using the component, you can:

SceneView组件是[mjs-volume](https://github.com/fabrobinet/mjs-volume)模块的一部分，这个模块由Fabrice Robinet维护。这个组件让前端web开发者能够用现有的HTML, CSS, 和 JavaScript技术开发交互式3D应用。使用它你可以实现这些：

* 集成3D场景到MontageJS web应用中。
* 用CSS操作3D场景中的元素。
* 用和CSS同样的方式处理3D场景transitions动画。

当然要完成这些，你的3D模型格式必须是SceneView组件能够识别的。

## glTF资源格式介绍

SceneView组件显示的内容是一种JSON运行时资源格式，叫做[OpenGL Transmission Format](https://www.khronos.org/gltf) (glTF)。是由 Khronos Group consortium—the组织根据popular [COLLADA](http://www.khronos.org/collada/)提出的一种开放、标准的Web 3D模型数字资源格式，glTF可以处理网格数据、动画、纹理和阴影。glTF资源通过下面的文件提供对3D场景的描述。

* 一个描述节点层次、素材和摄像机的JSON文件。
* 包含几何图形和动画的二进制文件。
* 纹理的图片（PG、PNG等等）文件
* 用于特定舞台的GLSL源码文本文件。

任何你想在MontageJS 3D应用中使用的3D资源必须转换成glTF格式。

## 转换3D资源为glTF

你可以使用COLLADA组织提供的3D-Asset-to-glTF工具来转换3D资源。

慨括地讲，转换3D资源为glTF格式分成两个步骤：

1. 通过3D设计软件把资源导出为COLLADA文件格式（结果是一个DAE文件）。
2. 使用开源命令行工具[COLLADA-to-glTF](http://www.khronos.org/gltf)把DAE文件转换成glTF模型（结果是一个JSON文件和一系列的二进制文本文件，通过这些文件可以重现3D场景）。

COLLADA-to-glTF转换工具可以成功转换由[SketchUp](http://www.sketchup.com/)和[其它3D设计软件](http://collada.org/mediawiki/index.php/Portal:Products_directory)这些3D模式设计软件导出的COLLADA文件。

可以打开mjs-volume项目的readme文件中[Converting 3D Assets to glTF](https://github.com/fabrobinet/mjs-volume/blob/master/README.md#converting-3d-assets-to-gltf)部分查看如何使用COLLADA-to-glTF转换工具。

想知道更多关于SceneView组件的细节，比如API文档，请访问GitHub上的[mjs-volume](https://github.com/fabrobinet/mjs-volume)项目。

# 配置MontageJS 3D项目

和使用其它技术开发3D应用一样，使用MontageJS也需要一些准备工作：首先你需要把原始的3D资源转换成SceneView组件能够识别的格式，然后在项目中添加3D资源和mjs-volume包。（SceneView组件在安装MontageJS时候没有安装，需要单独安装）

**备注:** 在学习教程过程中你不需要从头创建一个项目。教程中每个例子结束后都有一个指向源码的链接。

和创建其它MontageJS项目一样，你可以使用minit命令行工具创建3D项目（在[开始使用](http://montagejs.org/docs/montagejs-setup.html)查看具体步骤）；例如：

minit create:app -n beachplanet

然后把mjs-volume包和转换好的3D资源加入到项目中。

## 添加SceneView组件

用minit命令行创建一个项目后是不包含SceneView组件。你需要手动添加到你的MontageJS项目中：

1. 打开项目根目录里面的package.json文件。
2. 在dependencies节点中添加mjs-volume包：

* ...  
  "dependencies": {  
   ...,  
   "mjs-volume" : "git://github.com/fabrobinet/mjs-volume.git"  
  },  
  ...

1. 在命令行窗口切换到项目当前目录，运行以下命令：

* npm install

1. 按回车键执行命令后，mjs-volume包就成功添加到项目。

## 添加3D资源

把包含JSON、二进制和GLSL文件的3D资源文件夹复制到项目的assets文件夹。（例如，如果要在你的项目中使用Beach Planet示例的资源，首先从GitHub[下载](https://github.com/montagejs/beachplanetblog/archive/master.zip)，解压Beach Planet源码，然后把assets文件夹里面的3d文件夹复制到你项目的assets文件夹中。）

## 创建3D场景组件

和其它项目一样，首先在项目的ui目录中创建组件，然后在项目的Main组件模板中声明引用它.

**备注:** Main是MontageJS应用的主用户界面组件。和一个网站的首页或者单页面应用的入口一样：它可以包含任意数量的子组件来绘制页面和实现应用的功能。

现在为止你已经成功创建MontageJS 3D项目，可以开始编写应用代码。

# 导入3D场景

一个3D场景由节点层次构成并包括需要绘制的网格，几何图形，光线，阴影等等。在构建MontageJS 3D 应用时，你需要用到两个组件：

* Scene组件，通过这个组件来加载JSON-based glTF资源。
* SceneView界面组件，通过这个组件把场景内容显示在浏览器上。

在SceneView组件中引用一个Scene组件，就可以显示3D场景。

...  
"scene": {  
 "prototype": "mjs-volume/runtime/scene",  
 "properties": {  
 "path": "models/beachplanet/beachplanet.json"  
 }  
},  
"sceneView": {  
 "prototype": "mjs-volume/ui/scene-view.reel",  
 "properties": {  
 "element": { "#": "sceneView" },  
 "scene": { "@": "scene" }  
 }  
},  
...

<div data-montage-id="sceneView"></div>

这个示例中：

* scene声明一个scene组件对象，组件的定义是在mjs-volume/runtime/scene.js文件中。 对象的path属性设置为glTF资源的路径（beachplanet.json）。
* sceneView声明一个SceneView界面组件对象，组件的定义是在mjs-volume模块的scene-view.reel文件夹中。对象的scene属性设置为scene对象（用来下载场景数据）。element属性通过data-montage-id关联到HTML元素sceneView（浏览器会把场景内容显示在这个"容器"里面）。

3D场景已经能够加载并且显示在浏览器中。包括海洋、沙滩、树木、小黄鸭和一个房子。你可以用鼠标或者手势来旋转，放大，缩小场景。

 ***图 2.*** *导入一个简单3D场景*

**备注:** SceneView组件默认是没有尺寸设置，我们需要在组件的JS文件中处理。SceneView组件会自动设置THML元素的CSS属性和尺寸信息。我们可以在JS文件中改变SceneView的大小或者背景颜色。 在GitHub查看[完整源码](https://github.com/montagejs/beachplanetblog/tree/master/ui/planet.reel) 。

# 操作3D场景

在项目中引入并且显示3D场景后，你可以有几种方式操作场景中的元素。例如让一些元素动画响应用户动作，让用户可以切换视角等。通过在glTF资源和mjs-volume组件上应用CSS规则和MontageJS绑定模块，你可以很容易实现以上功能。实现的第一个步骤是在应用中引用3D场景中元素。

## 引用3D元素节点

在beachplanet.json文件中，每一个3D场景元素对象称作为一个节点。在一个场景中每个节点有一个唯一标识符ID。把glTF文件中需要操作的节点引入到应用。

例如，要操作Beach Planet示例中的小黄鸭元素(buoy)，你需要定义一个新对象然后设置它的id属性为glTF对应节点的id（这个例子中是：node\_31）,对象的类型是mjs-volume模块中node。

"scene": {  
 "prototype": "mjs-volume/runtime/scene",  
 "properties": {  
 "path": "models/beachplanet/beachplanet.json"  
 }  
},  
  
"buoy": {  
 "prototype": "mjs-volume/runtime/node",  
 "properties": {  
 "id": "node\_31",  
 "scene": { "@": "scene" }  
 }  
}

如上面例子所示，我们需要知道glTF中节点的ID值，所以在设计3D模型的时候就需要为节点设置一个容易记住的名字。在设计工具中为节点定义的名字会在转换成COLLADA(DAE)文件然后转换为glTF格式后保留。如果要知道一个节点的ID值，打开glTF JSON文本文件，然后搜索节点名就可以找到（这个例子中是beachplanet/beachplanet.json）。

Note that you can expose any individual material's properties in a 3D scene to MontageJS in much the same way that you expose a node, using the material.js runtime component. 你也可以在应用中引入3D场景中的其它元素，比如材质。和引入节点一样，只是使用 material.js组件。

## 用CSS操作3D节点

在组件的模板中引入节点后，你可以通过对它应用CSS规则来操作它。

node组件支持visibility属性和3D transforms。material组件支持opacity属性。

**备注:** 其它的功能，比如改变节点的纹理和透明度，会在以后的版本支持（你可以在[mjs-volume](https://github.com/fabrobinet/mjs-volume)查看最新功能）。 上面的两种组件同时也支持使用CSS动画。还有active和hover选择器，所以组件可以响应点击和滑过事件。可以像其它组件一样为节点设置一个CSS类名，但是只能够使用MontageJS绑定方式。

 ***图 2.*** *小黄鸭获取焦点后放大。*

在Buoy组件（buoy.reel）中设置以下的CSS规则后，当鼠标滑过小黄鸭（buoy）时就会放大：

* 在组件的CSS文件中定义animateCSS类，当鼠标滑过的时执行scale3d变化。
* .animate:hover {  
   transform: scale3d(3, 3, 3);  
   cursor: pointer;  
  }  
  .animate {  
   transition-property: transform;  
   transition-duration: 5s;  
   -montage-transform-z-origin: 0%;  
  }
* scale3d变化会增加元素的大小。transition属性控制动画的效果和时长；在这个例子中5秒内小黄鸭的大小变为全尺寸。当鼠标滑出之后，小黄鸭的尺寸变回原来大小。
* 在组件模板中，节点（buoy）通过classList.has绑定CSS类。
* "scene": {  
   "prototype": "mjs-volume/runtime/scene",  
   "properties": {  
   "path": "models/beachplanet/beachplanet.json"  
   }  
  },  
  "sceneView": {  
   "prototype": "mjs-volume/ui/scene-view.reel",  
   "properties": {  
   "element": { "#": "sceneView" },  
   "scene": { "@": "scene" }  
   }  
  },  
  "buoy": {  
   "prototype": "mjs-volume/runtime/node",  
   "properties": {  
   "id": "node\_31",  
   "scene": { "@": "scene" }  
   },  
   "bindings": {  
   "classList.has('animate')": { "<-": "true" }  
   }  
  }

在GitHub查看完整 [源码](https://github.com/montagejs/beachplanetblog/tree/master/ui/buoy.reel)

## 添加事件监听器

在真实的交互式3D应用中，需要响应用户动作事件。响应事件的方法是在节点对象上添加事件监听器，然后实现事件处理方法。

在一个3D节点上添加事件处理和在一个MontageJS按钮组件上添加事件处理的方式是一样。下面的代码实现当点击小屋节点之后弹出一个对话框。

...  
"door": {  
 "prototype": "mjs-volume/runtime/node",  
 "properties": {  
 "id": "node\_6",  
 "scene": { "@": "scene" }  
 },  
 "listeners": [{  
 "type": "action",  
 "listener": { "@": "owner" }  
 }]  
},  
...

var Component = require("montage/ui/component").Component;  
  
exports.Door = Component.specialize({  
 handleDoorAction: {  
 value: function(event) {  
 alert("The user clicked the door!");  
 }  
 }  
});

这里我们使用action监听点击事件，你也可以替换为监听滑动事件hover。

## 使用绑定操作3D节点

完成上面的例子后，你还可以让场景响应点击（触屏）事件。在小黄鸭放大例子中，绑定3D节点动画CSS类的值是固定值true。你可以用MontageJS绑定模块动态绑定，这样就可以实现是否需要应用动画CSS类。

下面的代码展示如何实现当小房子门被点击之后打开或者关闭。

"door": {  
 "prototype": "mjs-volume/runtime/node",  
 "properties": {  
 "id": "node\_6",  
 "scene": { "@": "scene" }  
 },  
 "bindings": {  
 "classList.has('animate')": { "<-": "true" },  
 "classList.has('open')": { "<-": "@owner.doorOpen" }  
 },  
 "listeners": [{  
 "type": "action",  
 "listener": { "@": "owner" }  
 }]  
}

在组件的JS文件事件处理函数中每次对doorOpen属性进行取反操作，这样的效果是保证门被点击之后doorOpen属性值在true和false之间来回切换。

Component = require("montage/ui/component").Component;  
  
exports.Door = Component.specialize({  
  
 doorOpen: { value: false },  
  
 handleDoorAction: {  
 value: function(event) {  
 this.doorOpen = ~this.doorOpen;  
 }  
 },  
 . . .  
});

在组件的CSS文件中，门打开的CSS类使用rotateZ属性控制门的角度，实现打开状态。 animateCSS类实现一动画的方式打开，关闭。

.open {  
 transform: rotateZ(-130deg);  
}  
.animate {  
 transition-property: transform;  
 transition-duration: 5s;  
 transform-origin: 0% 0%;  
}

注意动画相关CSS类是单独定义的，这样在门打开或者关闭的过程中都会被应用。

 ***图 4.*** *用绑定方式控制门节点的CSS类。*

在这个例子中还涉及到transform-origin属性，这个属性控制旋转过程中门的左边缘保持不动。如果没有设置它,门就会围绕中心旋转而不是门栓。在使用transformsCSS时，一般都需要按照需求设置相应的transform-origin属性。

在GitHub查看完整[源码](https://github.com/montagejs/beachplanetblog/tree/master/ui/door.reel)

## 改变视角

在一个复杂3D场景中，需要能够控制用户视角。SceneView组件可以很容易地做到在一个3D场景中在不同视角之间切换。一个视角就是一个摄像头节点。你可以像绑定其它场景元素一样的方式绑定视角。

在下面的例子中，sceneView对象有一个viewPoint属性，属性的值绑定到planetVP节点：

"sceneView": {  
 "prototype": "mjs-volume/ui/scene-view.reel",  
 "properties": {  
 "allowsViewPointControl" : false,  
 "element": { "#": "sceneView" },  
 "scene": { "@": "scene" },  
 "viewPoint": { "@" : "planetVP" }  
 }  
},  
"planetVP": {  
 "prototype": "mjs-volume/runtime/node",  
 "properties": {  
 "id": "node-Camera\_cabin",  
 "scene": { "@": "scene" }  
 }  
}

allowsViewPointControl属性控制是否允许用户通过拖拽改变视角。这个功能在应用需要在一个场景中通过拖拽显示更多内容时非常有用。你也可以用绑定的方式设置viewPoint属性，这样就可以通过编程改变视角。

Beach Planet示例也提供一个菜单来帮助用户快速选择视角：行星，海鸥，小黄鸭，小屋，和海豚。下面的示例代码展示如何在Beach Planet场景中切换两个不同的视角。viewPoint属性绑定摄像头到当前选中的菜单选项。

"planetVP": {  
 "prototype": "mjs-volume/runtime/node",  
 "properties": {  
 "id": "node-Camera\_cabin",  
 "scene": { "@": "scene" }  
 }  
},  
"cabinVP": {  
 "prototype": "mjs-volume/runtime/node",  
 "properties": {  
 "id": "node-Camera\_cabin",  
 "scene": { "@": "scene" }  
 }  
},  
"seaGullVP": {  
 "prototype": "mjs-volume/runtime/node",  
 "properties": {  
 "id": "node-Camera\_SeaGull",  
 "scene": { "@": "scene" }  
 }  
},  
"sceneView": {  
 "prototype": "mjs-volume/ui/scene-view.reel",  
 "properties": {  
 "allowsViewPointControl" : false,  
 "element": { "#": "sceneView" },  
 "scene": { "@": "scene" },  
 "viewPoint": { "@" : "planetVP" }  
 }  
},  
"nav": {  
 "prototype": "montage/ui/repetition.reel",  
 "properties": {  
 "element": { "#": "nav" },  
 "content": [  
 { "label": "Planet", "value": { "@": "planetVP" } },  
 { "label": "Cabin", "value": { "@": "cabinVP" } },  
 { "label": "Seagull", "value": { "@": "seaGullVP" } }  
 ]  
 }  
}

在GitHub查看示例的完整 [源码](https://github.com/montagejs/beachplanetblog/tree/master/ui/menu.reel)

 ***图 5.*** *通过viewPoint属性控制用户视角。*

现在你已经知道SceneView组件的基本使用方法，试试在自己的项目里实现可交互的3D功能吧。从从[3D Warehouse](http://sketchup.google.com/3dwarehouse/)可以下载到glTF格式的3D资源。SceneView组件虽然已经可以帮助你实现类似于Beach Planet效果项目。但我们并不满足，更多的功能正在开发中。

# Next Steps

# 下一步

继续研究Beach Planet示例 [源码](https://github.com/montagejs/beachplanetblog/tree/master/ui/beachplanet.reel)。

获取MontageJS 3D 组件最新开发版本，在GitHub上关注或者加星[mjs-volume](https://github.com/fabrobinet/mjs-volume) 项目。

我们非常欢迎你有意见（或者代码）帮助优化组件。你可以加入我们的[MontageJS邮件列表](https://groups.google.com/forum/?fromgroups#!forum/montagejs)或者从Twitter[@MontageJS](https://twitter.com/montagejs)上联系我们。

下面的资源帮助你了解更多关于MontageJS应用开发的知识：

* [MontageJS 文档](http://montagejs.org/docs/)
* [MontageJS 手册](http://seg.phault.net/montage/cookbook/)
* [开始使用 MontageJS](http://montagejs.org/docs/montagejs-setup.html)，一步一步教你配置MontageJS开发环境。