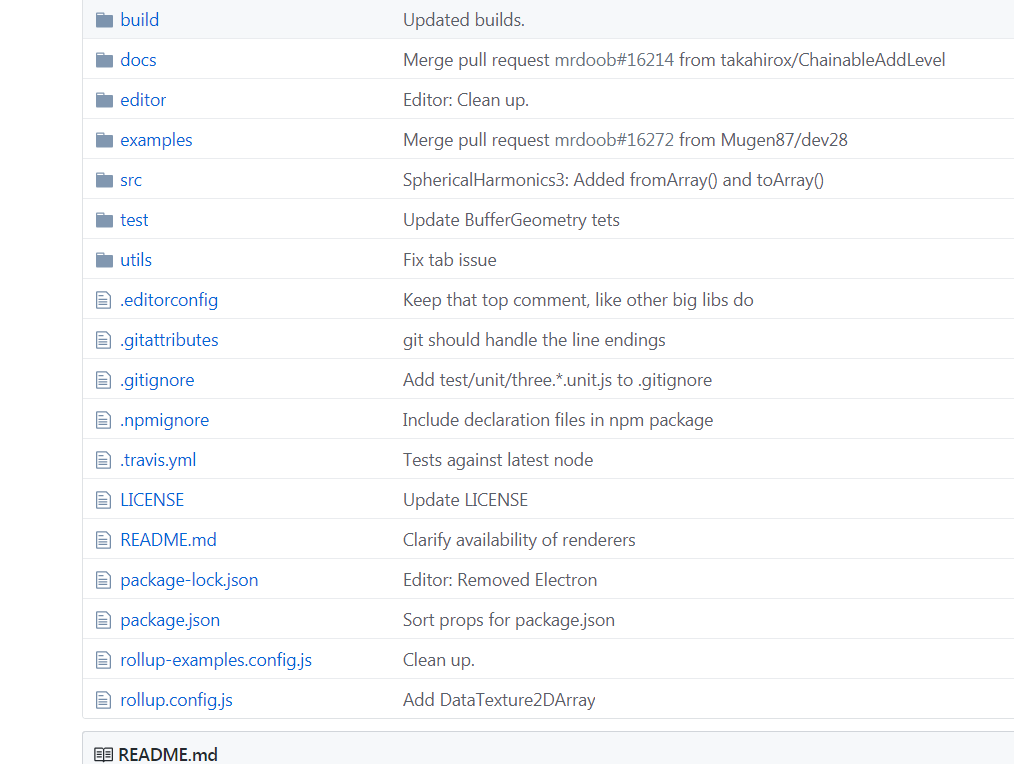
# webGL介绍

WebGL是一种3D绘图标准，这种绘图技术标准允许把JavaScript和OpenGL ES 2.0结合在一起，通过增加OpenGL ES 2.0的一个JavaScript绑定，WebGL可以为HTML5 Canvas提供硬件3D加速渲染，这样Web开发人员就可以借助系统显卡来在浏览器里更流畅地展示3D场景和模型了，还能创建复杂的导航和数据视觉化。

# three.js框架

<https://github.com/mrdoob/three.js>

目录结构



Build目录：包含两个文件，three.js 和three.min.js 。

Docs目录：这里是three.js的帮助文档，里面是各个函数的api，可惜并没有详细的解释。

Editor目录：一个类似3D-max的简单编辑程序，它能创建一些三维物体。

Examples目录：一些很有趣的例子demo，可惜没有文档介绍。

Src目录：源代码目录，里面是所有源代码。

Utils目录：存放一些脚本，python文件的工具目录。例如将3D-Max格式的模型转换为three.js特有的json模型。

.gitignore文件：git工具的过滤规则文件，没有用。

CONTRIBUTING.md文件：一个怎么报bug，怎么获得帮助的说明文档。

LICENSE文件：版权信息。

README.md文件：介绍three.js的一个文件，里面还包含了各个版本的更新内容列表。

# three.js引入

<script src="https://raw.github.com/mrdoob/three.js/master/build/three.js"></script>

验证启动成功 开发者调试工具控制台输入THREE.VERSION查看是否显示版本信息

# three.js基础

var scene = new THREE.Scene(); // 场景

var camera = new THREE.PerspectiveCamera(75, window.innerWidth/window.innerHeight, 0.1, 1000);// 透视相机

var renderer = new THREE.WebGLRenderer(); // 渲染器

renderer.setSize(window.innerWidth, window.innerHeight); // 设置渲染器的大小为窗口的内宽度，也就是内容区的宽度

document.body.appendChild(renderer.domElement); //将渲染器对象加入dom中

render( scene, camera, renderTarget, forceClear )

scene：前面定义的场景

camera：前面定义的相机

renderTarget：渲染的目标，默认是渲染到前面定义的render变量中

forceClear：每次绘制之前都将画布的内容给清除，即使自动清除标志autoClear为false，也会清除。

场景中生成物体，渲染器渲染相机视角下的画面载入页面中

工程下的编码一遍要对这几组对象分类进行

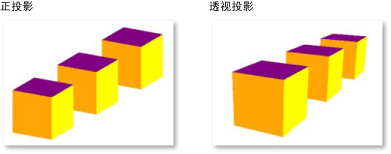
Threejs使用的是右手坐标系，这源于opengl默认情况下，也是右手坐标系。

# 监视器

资源监视器 <https://github.com/mrdoob/stats.js>

在 animation函数中调用stats.update()可进行统计

# 相机对象



正投影相机THREE.OrthographicCamera( left, right, top, bottom, near, far )

1、 left参数

left：左平面距离相机中心点的垂直距离。从图中可以看出，左平面是屏幕里面的那个平面。

2、 right参数

right：右平面距离相机中心点的垂直距离。从图中可以看出，右平面是屏幕稍微外面一点的那个平面。

3、 top参数

top：顶平面距离相机中心点的垂直距离。上图中的顶平面，是长方体头朝天的平面。

4、 bottom参数

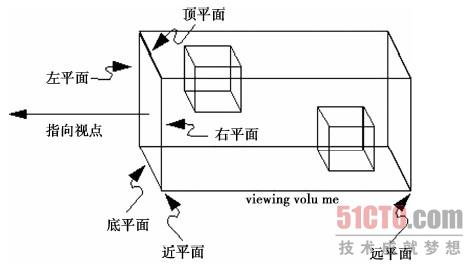
bottom：底平面距离相机中心点的垂直距离。底平面是头朝地的平面。

5、near参数

near：近平面距离相机中心点的垂直距离。近平面是左边竖着的那个平面。

6、far参数

far：远平面距离相机中心点的垂直距离。远平面是右边竖着的那个平面。



透视投影相机THREE.PerspectiveCamera( fov, aspect, near, far )

1、视角fov：这个最难理解,我的理解是,眼睛睁开的角度,即,视角的大小,如果设置为0,相当你闭上眼睛了,所以什么也看不到,如果为180,那么可以认为你的视界很广阔,但是在180度的时候，往往物体很小，因为他在你的整个可视区域中的比例变小了。

2、近平面near：这个呢，表示你近处的裁面的距离。补充一下，也可以认为是眼睛距离近处的距离，假设为10米远，请不要设置为负值，Three.js就傻了,不知道怎么算了,

3、远平面far：这个呢，表示你远处的裁面,

4、纵横比aspect：实际窗口的纵横比，即宽度除以高度。这个值越大，说明你宽度越大，那么你可能看的是宽银幕电影了，如果这个值小于1，那么可能你看到的是如下的图中的LED屏幕了。

# 光源

光源基类THREE.Light ( hex )

hex，接受一个16进制的颜色值

环境光

环境光是经过多次反射而来的光称为环境光，无法确定其最初的方向。环境光是一种无处不在的光。环境光源放出的光线被认为来自任何方向。因此，当你仅为场景指定环境光时，所有的物体无论法向量如何，都将表现为同样的明暗程度。 （这是因为，反射光可以从各个方向进入您的眼睛）

环境光

THREE.AmbientLight( hex )

环境光将照射场景中的所有物体，让物体显示出某种颜色

4 点光源

点光源：由这种光源放出的光线来自同一点，且方向辐射自四面八方。

PointLight( color, intensity, distance )

Color：光的颜色

Intensity：光的强度，默认是1.0,就是说是100%强度的灯光，

distance：光的距离，从光源所在的位置，经过distance这段距离之后，光的强度将从Intensity衰减为0。 默认情况下，这个值为0.0，表示光源强度不衰减。

5 聚光灯

聚光灯：这种光源的光线从一个锥体中射出，在被照射的物体上产生聚光的效果。使用这种光源需要指定光的射出方向以及锥体的顶角α

THREE.SpotLight( hex, intensity, distance, angle, exponent )

Hex：聚光灯发出的颜色，如0xFFFFFF

Intensity：光源的强度，默认是1.0，如果为0.5，则强度是一半，意思是颜色会淡一些。和上面点光源一样。

Distance：光线的强度，从最大值衰减到0，需要的距离。 默认为0，表示光不衰减，如果非0，则表示从光源的位置到Distance的距离，光都在线性衰减。到离光源距离Distance时，光源强度为0.

Angle：聚光灯着色的角度，用弧度作为单位，这个角度是和光源的方向形成的角度。

exponent：光源模型中，衰减的一个参数，越大衰减约快。

平行光又称为方向光（Directional Light），是一组没有衰减的平行的光线，类似太阳光的效果。

THREE.DirectionalLight = function ( hex, intensity )

Hex：关系的颜色，用16进制表示

Intensity：光线的强度，默认为1。因为RGB的三个值均在0~255之间，不能反映出光照的强度变化，光照越强，物体表面就更明亮。它的取值范围是0到1。

# 纹理

纹理类

THREE.Texture( image, mapping, wrapS, wrapT, magFilter, minFilter, format, type, anisotropy )

THREE.MeshBasicMaterial({map:texture})

Image：这是一个图片类型，基本上它有ImageUtils来加载

// url 是一个http://xxxx/aaa.jpg 的类似地址，javascript没有从本地加载数据的能力，

// 所以没有办法从您电脑的C盘加载数据。

var image = THREE.ImageUtils.loadTexture(url);

Mapping：是一个THREE.UVMapping()类型，它表示的是纹理坐标。下一节，我们将说说纹理坐标。

wrapS：表示x轴的纹理的回环方式，就是当纹理的宽度小于需要贴图的平面的宽度的时候，平面剩下的部分应该p以何种方式贴图的问题。

wrapT：表示y轴的纹理回环方式。 magFilter和minFilter表示过滤的方式，这是OpenGL的基本概念，我将在下面讲一下，目前你不用担心它的使用。当您不设置的时候，它会取默认值，所以，我们这里暂时不理睬他。

format：表示加载的图片的格式，这个参数可以取值THREE.RGBAFormat，RGBFormat等。THREE.RGBAFormat表示每个像素点要使用四个分量表示，分别是红、绿、蓝、透明来表示。RGBFormat则不使用透明，也就是说纹理不会有透明的效果。

type：表示存储纹理的内存的每一个字节的格式，是有符号，还是没有符号，是整形，还是浮点型。不过这里默认是无符号型（THREE.UnsignedByteType）。

anisotropy：各向异性过滤。使用各向异性过滤能够使纹理的效果更好，但是会消耗更多的内存、CPU、GPU时间

纹理可以通过canves添加，通过