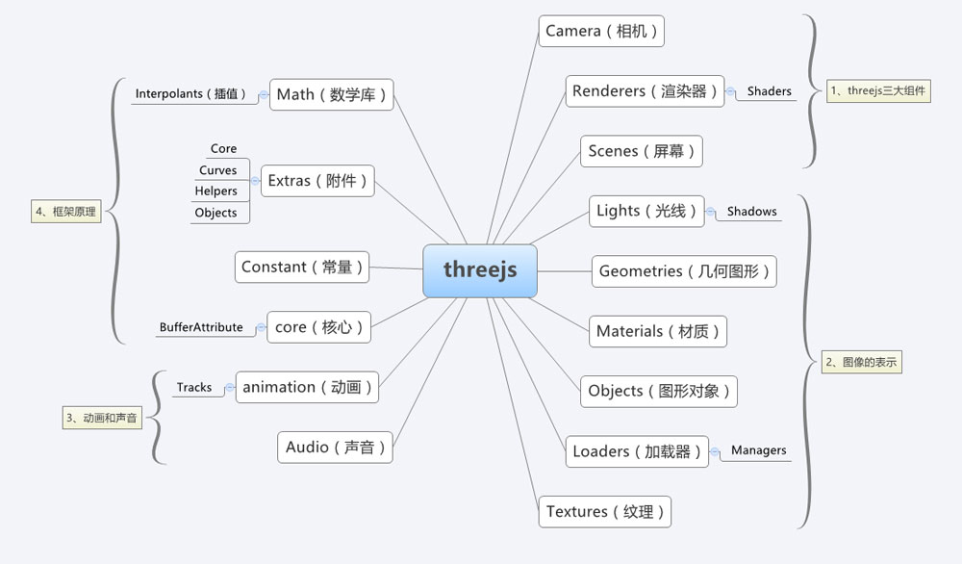
**Three.js入门**

# Three.js 简介



[**Three.js**](https://link.jianshu.com?t=https%3A%2F%2Fthreejs.org%2F)**源自github的一个开源项目**

**开源地址：**

[https://github.com/mrdoob/three.js](https://link.jianshu.com?t=https%3A%2F%2Fgithub.com%2Fmrdoob%2Fthree.js)

**应用场景及发展：**

应用场景有全景图，3D网页游戏、3D模型、VR等等。

[Three.js](https://link.jianshu.com?t=https%3A%2F%2Fthreejs.org%2F)已经更新迭代到了[r118](https://github.com/mrdoob/three.js/commit/440dd963240698ab6c99fb91af121a76d6effc89)版本，即更新100多个版本了。

**目的：**

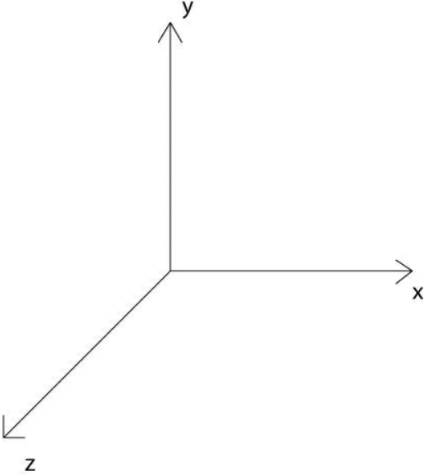
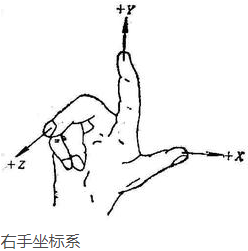
Three.js目的在于简化WebGL开发复杂度和降低入门难度，是在WebGL标准基础上封装了一个轻量级的JS 3D库。

**渲染：**

Three.js 是基于canvas渲染的，所创建出来的东西最终会被渲染到canvas上面。

**坐标系：**

Three.js 使用的坐标系是右手坐标系

坐标系的原点在画布中心（canvas.width / 2, canvas.height / 2）。我们可以通过 Three.js 提供的 THREE.AxisHelper() 辅助工具方法将坐标系可视化。

ps: 在 Three 中是没有「长度单位」这个概念的，它的数值都是根据比例计算得出，因此这里提到的 0.1 或 1000 都没有具体的含义，而是一种相对长度。

# Threejs的常用对象

渲染器（Renderer）、场景（Scene）、相机（Camera）、光源（light）

#### 渲染器（Renderer）

**渲染器则负责用如何渲染出图像，**初始化渲染器Renderer的两种方法：

（1）利用Dom中Canvas元素与渲染器进行绑定，即之前在 HTML中手动定义了 id 为 myCanvas 的 Canvas 元素，那么可以这样初始化Renderer渲染器：

var renderer = new THREE.WebGLRenderer({

  canvas: document.getElementById('myCanvas');

});

（2）利用Threejs生成Canvas元素，在HTML中就不需要定义 Canvas 元素，通过Threejs设置了一个宽高为600的Canvas，并添加到了body中

var renderer = new THREE.WebGLRenderer();

renderer.setSize(600, 600); // canvas 的大小

document.body.appendChild(renderer.domElement);

#### 场景（Scene）

场景用来容纳图形元素，场景相当于宇宙，而图形元素就是星星，**图形元素只有添加到场景中，其坐标、大小等才有意义。**

var scene = new THREE.Scene(); // 实例化一个场景Scene

var something;

scene.add(something); // 添加something到scene中

#### 照相机（Camera）

**相机的作用是定义可视域，即确定哪些图形元素是可见的**

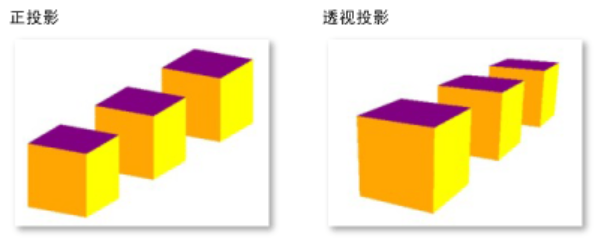
相机有自己的位置（position）、朝向（lookAt）和 可视域

我们创建的图形一定要放在相机的可视域中才能看得见。可视域与相机的类型、相机的位置和朝向都有关。有了照相机才能把看的的东西渲染在canvas画布中。

**相机的类型**

正投影相机（OrthographicCamera）

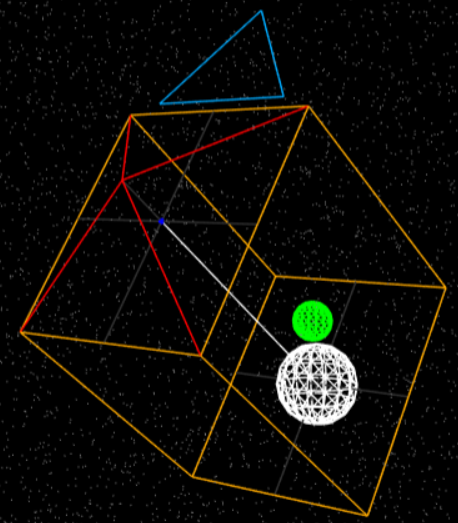
透视投影相机（PerspectiveCamera）

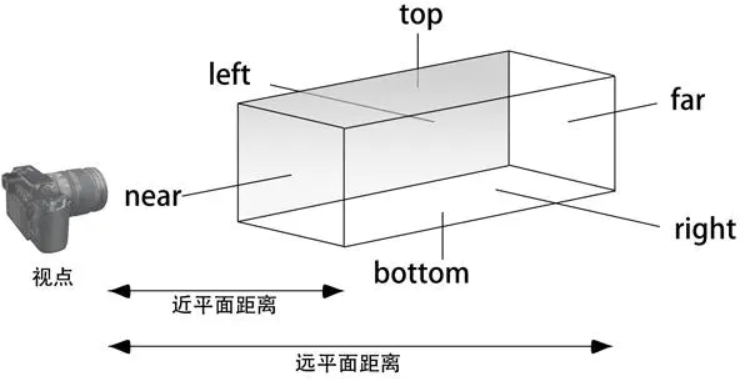


**正投影相机**

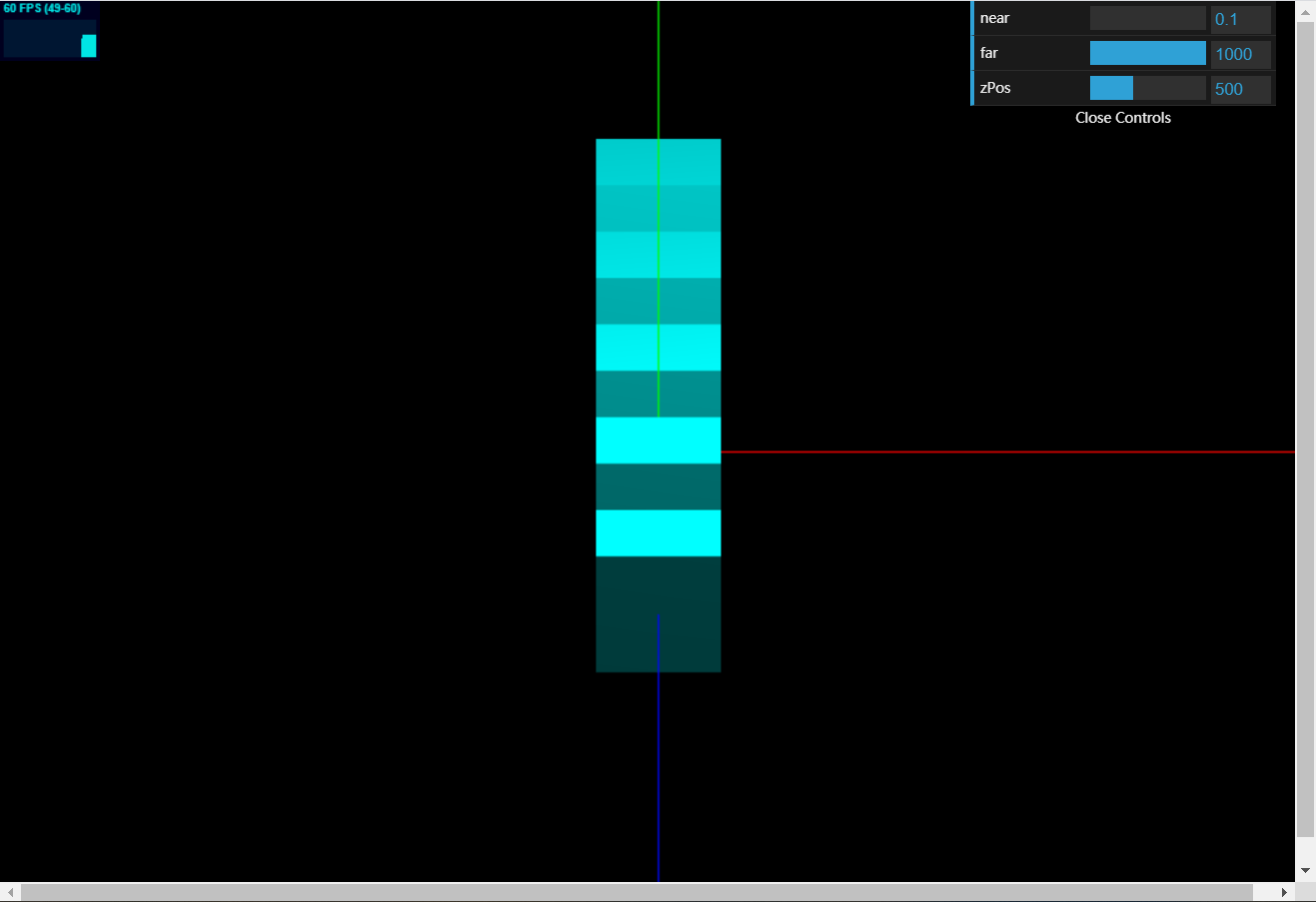
**THREE.OrthographicCamera(**左偏移, 右偏移，上偏移，下偏移，近平面距离、远平面距离**)，**偏移是相对于相机中心位置，不是真实坐标

**一般来说正投影相机（right-left）/ (top-bottom)的比例和画布的比例是一致的，才不会出现物体变形。**



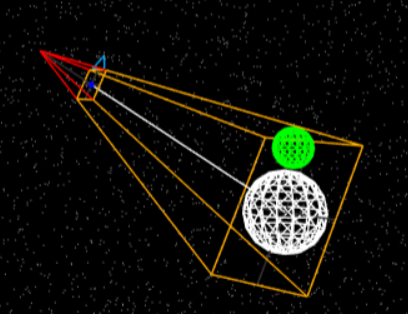


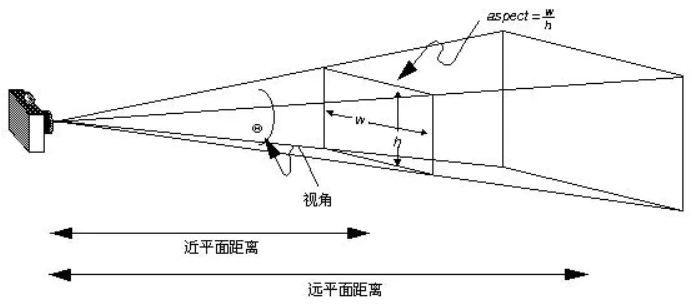
正交相机拍摄的效果类似设计图，重在表现物体的实际尺寸，没有近大远小的效果



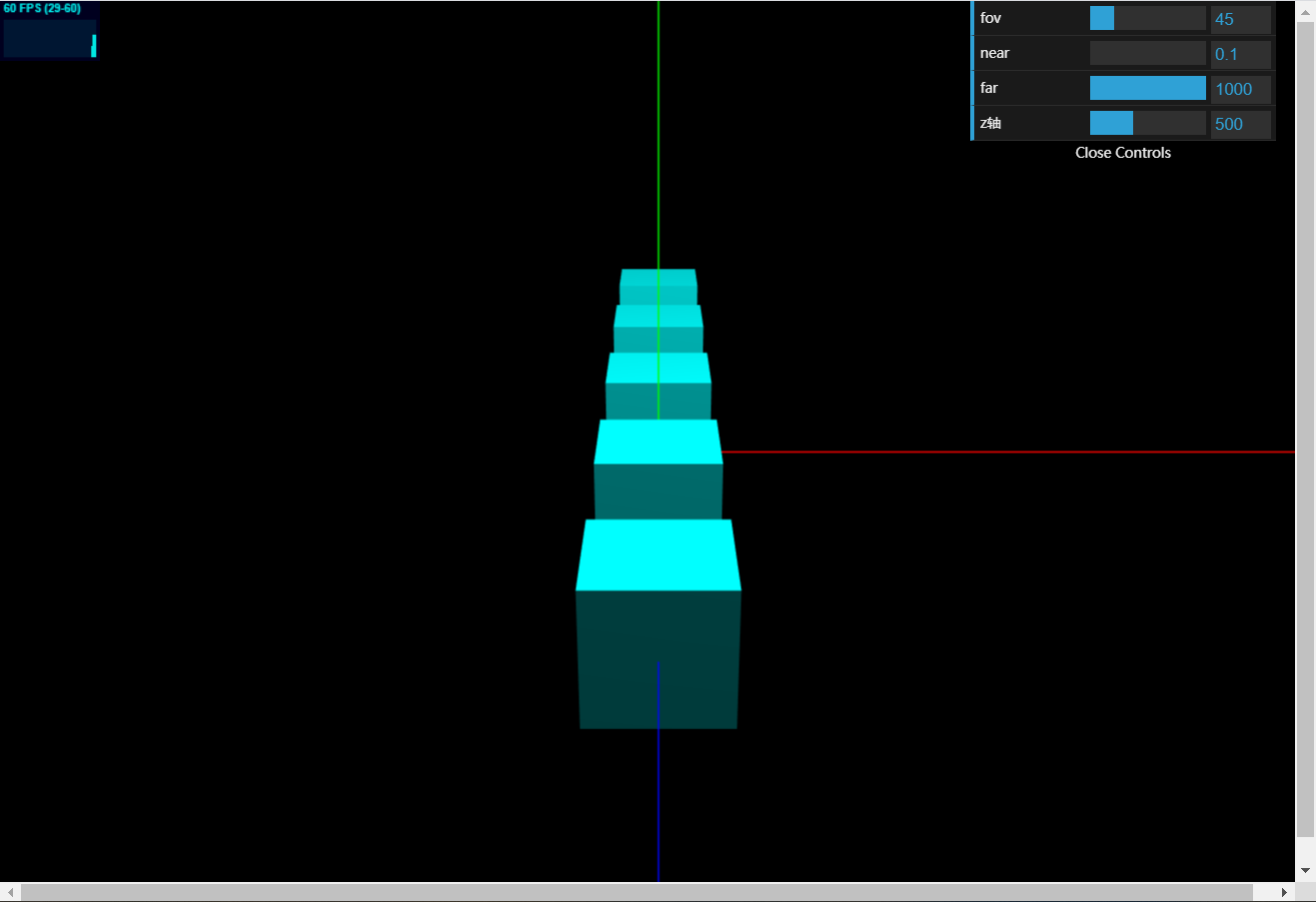
**透视投影相机**

**THREE.PerspectiveCamera(**视角, 宽高比, 近平面距离, 远平面距离**)**





透视投影相机拍摄的效果和人眼看的效果是一致的，有近大远小的效果



#### 光源（Light）

**光源能照亮在环境中的物体，常用的光源有：**

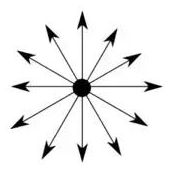
点光源 **PointLight**

聚光灯光源 **SpotLight**

平行光源 **DirectionalLight**

环境光源 **AmbientLight**

**点光源**



**点光源从一个点向各个方向发射，实际情况中可以理解为灯泡发出的光，可以将物体的阴影投射出。**

 new THREE.PointLight() 用于创建点光源，该方法中有四个变量，依次是 color ， intensity ， distance ， decay 。

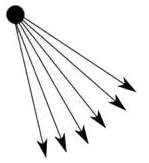
color ：(可选参数)) 十六进制光照颜色。 缺省值 0xffffff (白色)，不支持 rgba 格式颜色。

intensity ：(可选参数) 光照强度。 缺省值 1，值越大光照越强，实测最小值为0，最大值不限。

distance ：这个距离表示从光源到光照强度为0的位置。 当设置为0时，光永远不会消失(距离无穷大)。缺省值 0，可以设置任意非负数。

decay ：沿着光照距离的衰退量。缺省值 1。 模拟现实世界光衰减只需要设置 decay 值为2便可

**聚光灯光源**



**聚光灯是从一个方向上的一个点发出，沿着一个圆锥体，它离光越远，它的尺寸就越大。**

new THREE.SpotLight() 用于创建点光源，该方法中有四个变量，依次是 color ， intensity ， distance ， angle ， penumbra ， decay。

color ：(可选参数)) 十六进制光照颜色。 缺省值 0xffffff (白色)，不支持 rgba 格式颜色。

intensity ：(可选参数) 光照强度。 缺省值 1，值越大光照越强，实测最小值为0，最大值不限。

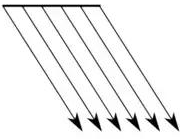
distance ：这个距离表示从光源到光照强度为0的位置。 当设置为0时，光永远不会消失(距离无穷大)。缺省值 0，可以设置任意非负数。

angle：光线散射角度，最大为Math.PI/2。

penumbra：聚光锥的半影衰减百分比。在0和1之间的值。默认为0。

decay ：沿着光照距离的衰退量。缺省值 1。 模拟现实世界光衰减只需要设置 decay 值为2便可。

**平行光源**



**平行光是沿着特定方向发射的光，这种光的表现像是无限远,从它发出的光线都是平行的，常常用平行光来模拟太阳光的效果，太阳足够远，因此我们可以认为太阳的位置是无限远，所以我们认为从太阳发出的光线也都是平行的。**

new THREE.DirectionalLight() 用于创建平行光源，创建方法中只需要两个变量，依次是 color ， intensity 。

color ：(可选参数)) 十六进制光照颜色。 缺省值 0xffffff (白色)，不支持 rgba 格式颜色。

intensity ：(可选参数) 光照强度。 缺省值 1，值越大光照越强，实测最小值为0，最大值不限。

**环境光源**

**环境光会均匀的照亮场景中的所有物体，该光源无法投影物体的阴影，因为环境光源是没有方向的**

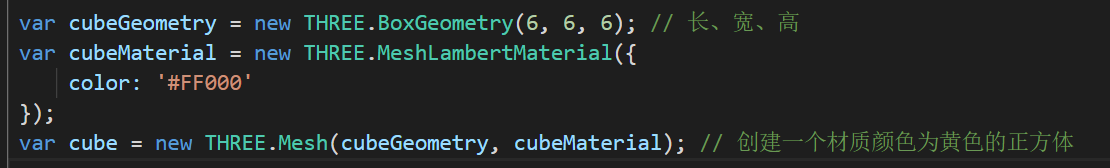
new THREE.AmbientLight() 用于创建环境光源，环境光源比较简单，创建方法中只需要两个变量，依次是 color ， intensity 。

color ：(可选参数)) 十六进制光照颜色。 缺省值 0xffffff (白色)，不支持 rgba 格式颜色。

intensity ：(可选参数) 光照强度。 缺省值 1，值越大光照越强，实测最小值为0，最大值不限。

#### 网格（Mesh）

**创建一个几何体是通过Mesh来创建的，Mesh需要物体外壳和物体(形状)的材质(颜色等)**通常创建一个Mesh网格对象如下：



拓展：

Threejs网站：https://threejs.org/

纹理、精灵、骨骼、动画和声音