**利用HTML5构筑物理模拟环境～ WebGL库Three.js入门（１／３）**

发布时间：2012-05-09 作者：小想 来源：未知 [我要评论](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#postform)

这是目前质量较好、讲解比较清楚的一篇 Three.js 的入门教程，原作者是日本人遠藤理平，由 HiWebGL 技术讨论群群友瀡风翻译，HiWebGL 授权转载。感谢原作者的创作和瀡风辛苦翻译！…

作者：遠藤 理平  
翻译：瀡风(2012年5月6日)  
如需转载本中文译文，请联系翻译者瀡风！  
中文翻译转自：<http://www.hiwebgl.com/?p=1058>  
点击这里查看：[原文](http://www.natural-science.or.jp/article/20120220155529.php)

目录

* [1 HTML5和WebGL](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#HTML5WebGL)
  + [1.1 HTML5的新成员「canvas元素」](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#HTML5canvas)
  + [1.2 OpenGL 的浏览器版 WebGL 登场](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#OpenGL_WebGL)
  + [1.3 WebGL例](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#WebGL)
  + [1.4 WebGL的利用方法](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#WebGL-2)
* [2 三维物理模拟：2重单摆模拟器的作成](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#2)
  + [2.1 环境的准备](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i)
  + [2.2 Three.js下载](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#Threejs)
* [3 Three.js的动作确认](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#Threejs-2)
  + [3.1 例１：tutorial1.html（设置视点和光源、绘制立方体）](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#tutorial1)
  + [3.2 例１的解说](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-2)
  + [3.3 ０．画布(Canvas)的准备](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#Canvas)
    - [3.3.1 body 元素中追加 div 标签](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#body_div)
    - [3.3.2 style 元素中追加 CSS样式](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#style_CSS)
  + [3.4 １．设置渲染器](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-3)
  + [3.5 ２．设置相机](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-4)
    - [3.5.1 透视投影模式下相机的设置](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-5)
    - [3.5.2 设置相机的位置坐标](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-6)
    - [3.5.3 设置相机的上方向](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-7)
    - [3.5.4 设置相机的视野中心](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-8)
    - [3.5.5 动作确认](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-9)
    - [3.5.6 其他投影方式](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-10)
  + [3.6 ３．声明场景](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-11)
  + [3.7 ４．设置光源并追加到场景](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-12)
    - [3.7.1 动作确认](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-13)
    - [3.7.2 其他光源](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-14)
  + [3.8 ５．声明立方体并追加到场景](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-15)
    - [3.8.1 动作确认](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-16)
    - [3.8.2 其他形状类](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-17)
  + [3.9 ６．执行渲染（threeStart() ）](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#threeStart)
  + [3.10 稍微再试一下（追加平面和球体）](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-18)
    - [3.10.1 执行结果](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-19)
* [4 Three.js 实现动画](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#Threejs-3)
  + [4.1 使用「window.requestAnimationFrame」函数实现动画](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#windowrequestAnimationFrame)
  + [4.2 例２：tutorial2.html（实现动画）](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#tutorial2html)
  + [4.3 例２的解说](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-20)
    - [4.3.1 创建多个对象](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-21)
    - [4.3.2 「loop()」函数的定义和执行](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#loop)
    - [4.3.3 指定物体的回转角度的方法](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-22)
  + [4.4 例２－２：tutorial2\_2.html（相机的移动）](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#tutorial2_2)
  + [4.5 例２－３：tutorial2\_3.html（设置环境光）](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#tutorial2_3)
  + [4.6 例２－４：tutorial2\_4.html（影子的设置）](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#tutorial2_4)
  + [4.7 例２－５：tutorial2\_5.html（纹理）](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#tutorial2_5)
* [5 鼠标事件](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-23)
  + [5.1 鼠标事件例](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#i-24)
  + [5.2 例３：tutorial3.html（鼠标事件）](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html#tutorial3)

**HTML5和WEBGL**

简单认识一下HTML5吧。HTML5是目前构成WEB页的主流的HTML4和XHTML的后续语言、 2008年制定了草案、约定各大浏览器提供商一起力争在2014年前形成正式的版本。 HTML5在2012年1月还处于「草案」阶段、虽然还处于规范在变动的准备阶段、但是无论是开发者还是用户都非常的关注。 一个主要的理由就是、随着iPhone和Android等智能手机的崛起，人们期待HTML5能够对各种各样的WEB内容跨平台化做 出重要的贡献。 在智能手机登上舞台之前、Adobe公司的 FLASH 在动画和音频等多媒体内容领域已成为了事实上的标准、通过导入FLASH插件在各种操作系统或者浏览器中都能够支持FLASH格式、事实上通过FLASH 应该是可以构建一个跨平台的执行环境、但是因为Apple公司的iPhone（和iPad）上拒绝支持FLASH使得这一构想成为了泡影。 于是、一个崭新的跨平台执行环境闪耀登场、这就是WEB标准化团体的作为下一代WEB页开发语言的HTML5。 HTML从此由单纯的支持静态文本内容的语言开始， 进化成支持动画和音频，而且还可以直接操作画像数据的语言。 并且、因为不再需要导入插件了、当然可以实现包括智能手机在内的大多数环境中的跨平台。出于对WEB内容跨平台化迫切的需求、虽然还只是在【草案】阶段的 HTML5已经被各浏览器所支持。2012年2月为止、HTML5智能手机中的浏览器基本上100%支持HTML5了（电脑平台的各浏览器也基本都支 持）。 另外，HTML5不是一个单体，而是需要和控制视觉体裁的CSS3及浏览器支持的Javascript语言联合起来才能发挥出真实的价值。

**HTML5的新成员「CANVAS元素」**

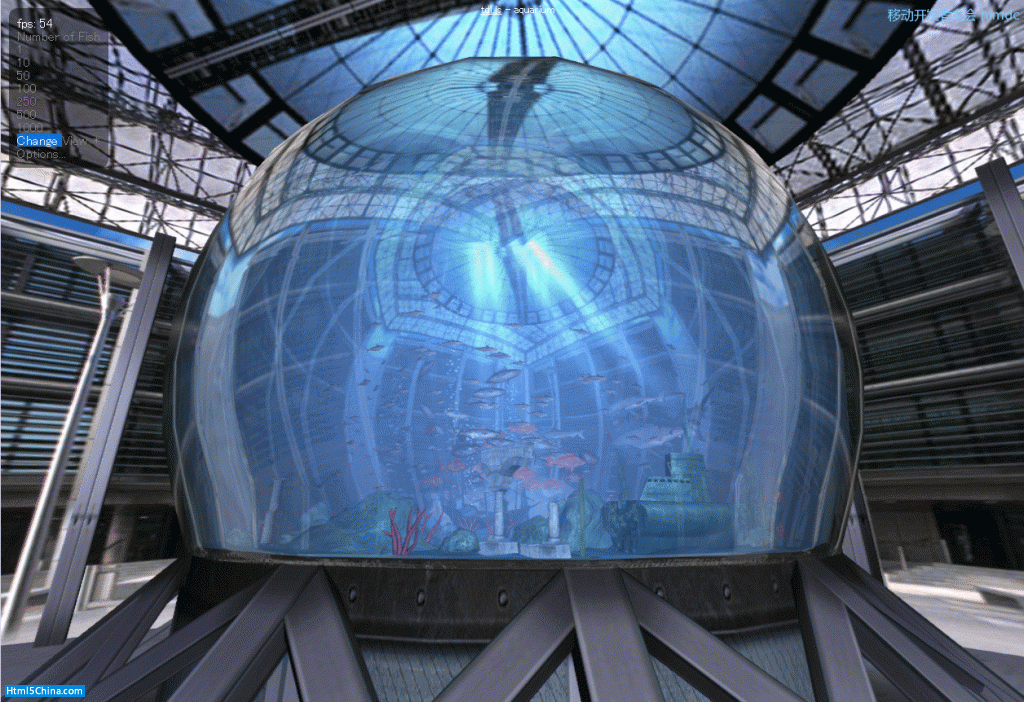
在HTML5中、新登场的 canvas 元素所表现的领域里、可以以像素为单位指定颜色。 这意味着不依靠插件也能在HTML中直接绘制二维和三维图像。

**OPENGL 的浏览器版 WEBGL 登场**

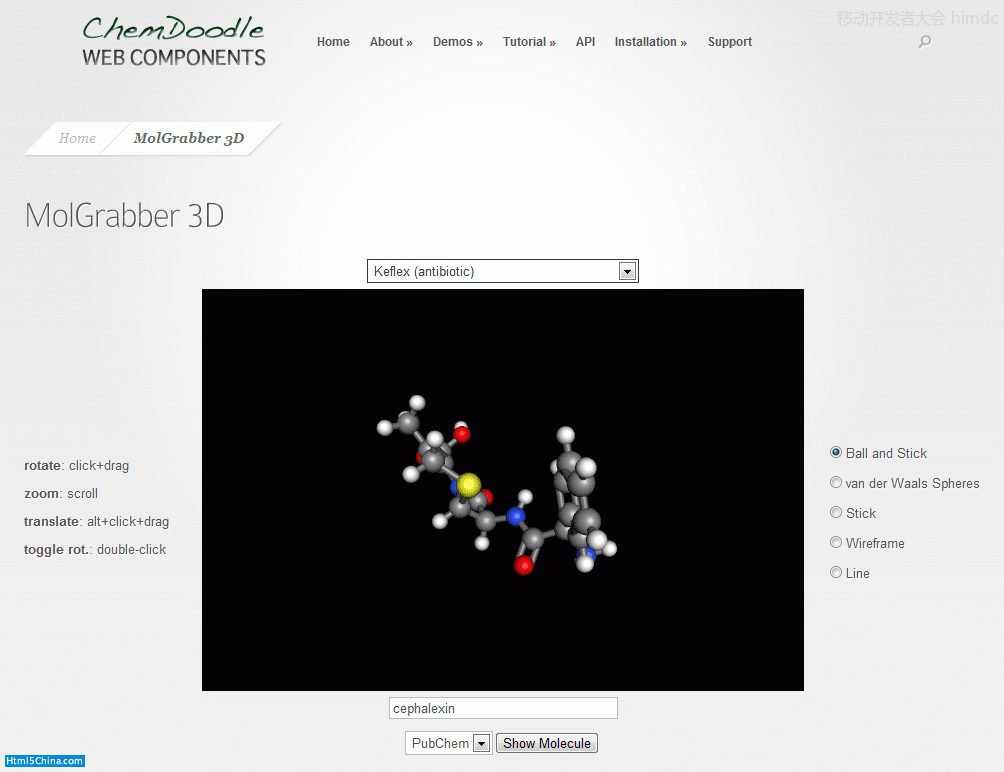
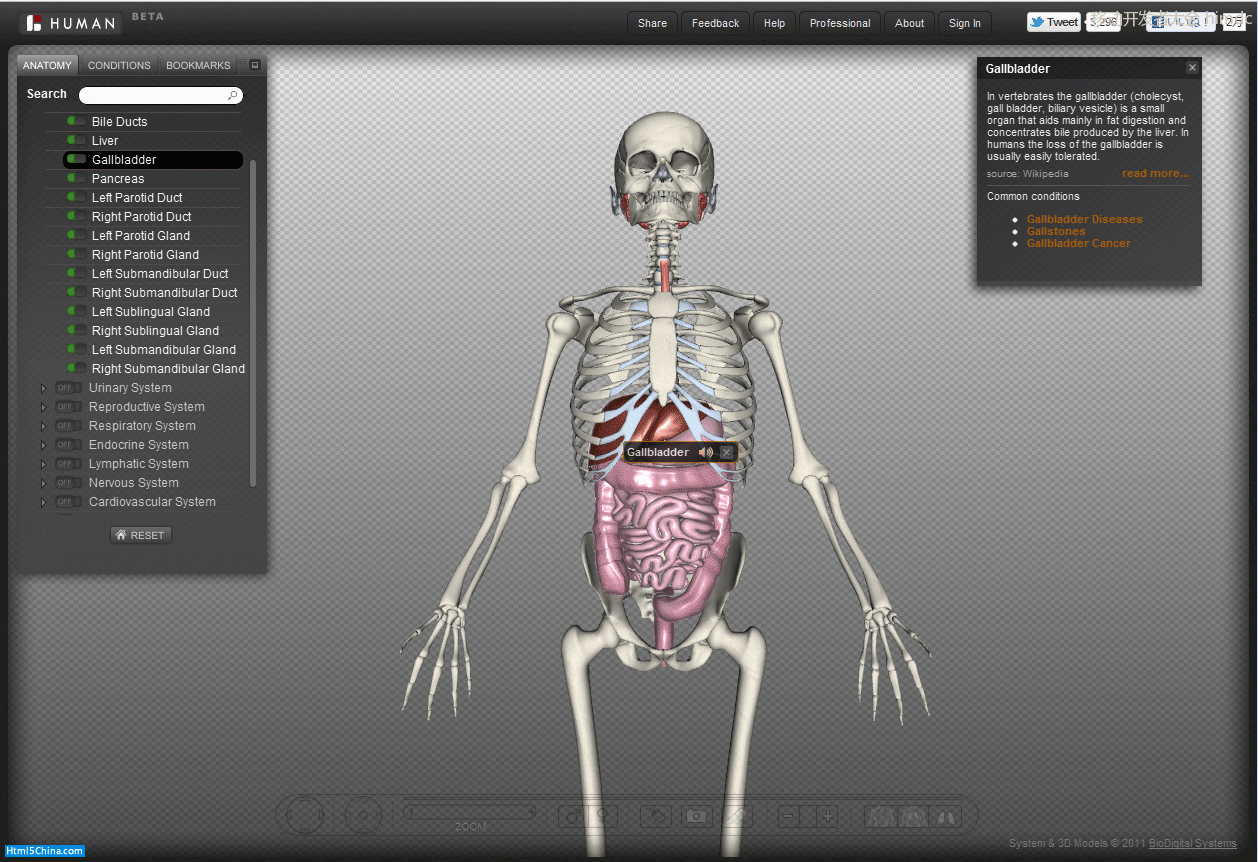
关于2维3维计算机图形技术、Khronos Group对各种技术进行了标准化并制定了[OpenGL]、作为Windows、Mac、Linux上的一种跨平台技术并一直活跃着。 OpenGL 能够从任意视点出发，对三维空间中的物体进行二维投影的自动计算。 而且因为可以直接操作计算机的图形卡、OpenGL能够非常高速和高精度地描绘三维图像。2009年，Khronos Group把这样的OpenGL技术运用到WEB浏览器并制定了WebGL。WebGL 是 OpenGL 和 JavaScript 之间的结晶、HTML5 的 canvas 元素里、利用和OpenGL同样的API、可以绘制高精度的三维图像。 但问题也出来了，占有最大浏览器份额的InternetExplore反对采用WebGL。 因为一直以来 OpenGL 和 开发InternetExplore 的 Microsoft公司独自开发的三维图形API「DirectX」处于竞争者关系。 可因为 DirectX 只是 Microsoft 公司的产品、不能成为跨平台的选择、 而且在浏览器中描绘性能也没有超过WebGL的迹象、所以胜败已经是很明显的事情了。 关于WebGL的中需要注意的是、浏览器要支持WebGL、电脑中安装的图形显示卡中的OpenGL的版本必须要是2.0以上。

**WEBGL例**

关于WebGL非常有名的例子是、Human Engines和Gregg Tavares 2010年末开发的「[Aquarium](http://webglsamples.googlecode.com/hg/aquarium/aquarium.html)」。下图是画面截图、请使用支持WebGL的浏览器进行浏览。

[](http://www.himdc.com/wp-content/uploads/2012/05/052920pCS.gif)

在浏览器中能够实现如此精致的三维图形，而且还有动画效果真是让人吃惊。 最近、侧重学习的实用性、[化学构造的三维绘制库「ChemDoodle」（下图左）](http://web.chemdoodle.com/) 和 [三维人体解剖图「BIODIGITAL HUMAN」（下图右）](http://www.biodigitalhuman.com/)等已经陆续出现了。近期教育界正在议论的教材数字化，也在考虑是不是可以直接推进到类似这样的层次。

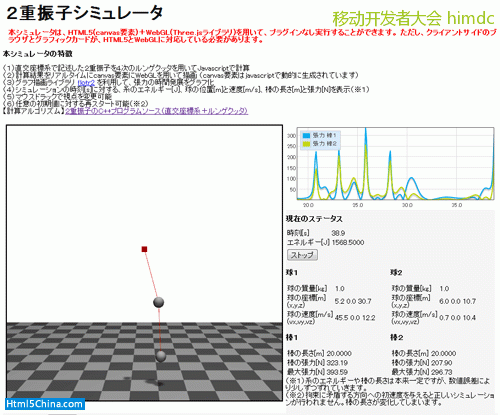
[](http://web.chemdoodle.com/)    [](http://www.biodigitalhuman.com/)

**WEBGL的利用方法**

在做WebGL开发的时候，可以利用「glMatrix.js」库矩阵运算的功能。 2009年末、Giles发表了使用WebGL的入门向导文章「[Learning WebGL](http://learningwebgl.com/blog/?page_id=1217)」、WebGL的使用方法变得更加容易理解了。虽说通过WebGL的绘制效果甚至可能达到本地的高性能游戏专用机的绘制效果，但如果没有非常精通OpenGL的人在还是不太现实的。 因此、2010年末为了能够更加简单的利用WebGL，库「[Three.js（Mr.doob）](http://mrdoob.github.com/three.js/)」 被发表了。因为 Three.js 非常容易使用，所以很快变得非常有人气。 发表的初期每周一回的频率、到2012年2月还保持在1.5个月１回的频率进行版本更新、期待它在未来有更远的发展。 本文的三维物理模拟的绘制也使用了Three.js。 WebGL的javascript框架除了Three.js之外，还有「[J3D](http://creativejs.com/)」和 「[SceneJS](http://scenejs.com/)」和、日本开发的「[gl.enchant.js β版（株式会社ユビキタスエンターテインメント）](http://9leap.net/games/1109)」等。

**三维物理模拟：2重单摆模拟器的作成**

本文中、利用 WebGL 的 Three.js 库作成了三维物理模拟实例「[２重振り子シミュレータ](http://www.natural-science.or.jp/WebGL/DoublePendulum_ver1.0.html)」。 支持WebGL的浏览的话[查看这边](http://www.natural-science.or.jp/WebGL/DoublePendulum_ver1.0.html)。下图是「[２重振り子シミュレータ](http://www.natural-science.or.jp/WebGL/DoublePendulum_ver1.0.html)」的截图。

[](http://www.natural-science.or.jp/WebGL/DoublePendulum_ver1.0.html)

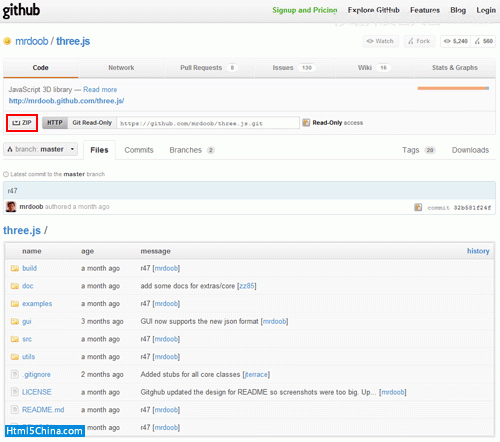
**环境的准备**

１．文本编辑器（什么都没有也没关系）  
２．运行WebGL的浏览器（推荐：Google Chrome ver.17）  
３．调试器（推荐：Google Chrome）  
４．WebGL 库 Three.js

本文的内容按照下面的环境进行过测试。  
OS：Windows7 Professional  
浏览器：Google Chrome（ver.17）  
GPU：NVIDIA 550i（OpenGL 4.2） Three.js 的版本：r47

**THREE.JS下载**

Three.js 的下载、先打开「[mrdoob/three.js – GitHub](https://github.com/mrdoob/three.js/)」、点击下图的 「ZIP」就可以下载所有的文件。使用 Three.js 的时候只要有「build」目录下的「Three.js」就足够了。



拷贝「build」目录下的「Three.js」到工作目录下。这样准备工作就做好了。

**THREE.JS的动作确认**

绘制一个长方体，顺便确认下 Three.js 的运行。 启动文本编辑器、把下面的源代码拷贝&粘贴、并保存到工作目录下。

**例１：TUTORIAL1.HTML（设置视点和光源、绘制立方体）**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Three.js チュートリアル１</title>

<script src="Three.js"></script>

<style type="text/css">

div#canvas-frame{

border: none;

cursor: pointer;

width: 600px;

height: 600px;

background-color: #EEEEEE;

}

</style>

<script>

var renderer;

function initThree() {

width = document.getElementById('canvas-frame').clientWidth;

height = document.getElementById('canvas-frame').clientHeight;

renderer = new THREE.WebGLRenderer({antialias: true});

renderer.setSize(width, height );

document.getElementById('canvas-frame').appendChild(renderer.domElement);

renderer.setClearColorHex(0xFFFFFF, 1.0);

}

var camera;

function initCamera() {

camera = new THREE.PerspectiveCamera( 45 , width / height , 1 , 10000 );

camera.position.x = 100;

camera.position.y = 20;

camera.position.z = 50;

camera.up.x = 0;

camera.up.y = 0;

camera.up.z = 1;

camera.lookAt( {x:0, y:0, z:0 } );

}

var scene;

function initScene() {

scene = new THREE.Scene();

}

var light;

function initLight() {

light = new THREE.DirectionalLight(0xFF0000, 1.0, 0);

light.position.set( 100, 100, 200 );

scene.add(light);

}

var cube;

function initObject(){

cube = new THREE.Mesh(

new THREE.CubeGeometry(50,50,50), //形状の設定

new THREE.MeshLambertMaterial({color: 0xff0000}) //材質の設定

);

scene.add(cube);

cube.position.set(0,0,0);

}

function threeStart() {

initThree();

initCamera();

initScene();

initLight();

initObject();

renderer.clear();

renderer.render(scene, camera);

}

</script>

</head>

<body onload="threeStart();">

<div id="canvas-frame"></div>

</body>

</html>

执行的结果就象下面这样。 使用支持WebGL的浏览器就可以看到一个红色的长方体被绘制出来了。

**例１的解说**

使用Three.js 绘制的步骤参照如下

* ０．画布的准备
* １．Three.js 的初期化（initThree()）
* ２．相机的准备（initCamera()）
* ３．场景的准备（initScene()）
* ４．光源的准备（initLight()）
* ５．物体的装备（initObject()）
* ６．执行绘制（threeStart()）

**０．画布(CANVAS)的准备**

准备和画布框大小一致的领域用于WebGL绘制。 具体来说、  
(1) body 标签中添加 id 为「canvas-frame」的 div 元素。  
(2) style 标签中指定 「canvas-frame」的CSS样式。

**BODY 元素中追加 DIV 标签**

<[div](http://december.com/html/4/element/div.html) id="canvas-frame"></[div](http://december.com/html/4/element/div.html)>

**STYLE 元素中追加 CSS样式**

通过【元素#ID名】的CSS[选择器](http://www.html5china.com/HTML5features/selector/)方式，为【canvas-frame】元素指定如下属性。  
(1) 隐藏框线  
(2) 鼠标光标设置为【pointer】  
(3) 宽度设置为600px  
(4) 高度设置为600px  
(5) 背景设置为「#EEEEEE」

<[style](http://december.com/html/4/element/style.html) type="text/css">

div#canvas-frame{

border: none; //(1)

cursor: pointer; //(2)

width: 600px; //(3)

height: 600px; //(4)

background-color: #EEEEEE; //(5)

}

</[style](http://december.com/html/4/element/style.html)>

**１．设置渲染器**

三维空间里的物体映射到二维平面的过程被称为三维渲染。 一般来说我们都把进行渲染操作的软件叫做渲染器。 具体来说要进行下面这些处理。

(0) 声明全局变量（对象）  
(1) 获取画布「canvas-frame」的高宽  
(2) 生成渲染器对象（属性：抗锯齿效果为设置有效）  
(3) 指定渲染器的高宽（和画布框大小一致）  
(4) 追加 【canvas】 元素到 【canvas-frame】 元素中。  
(5) 设置渲染器的清除色(clearColor)

var width, height; //(0)

var renderer; //(0)

function initThree() {

width = document.getElementById('canvas-frame').clientWidth; //(1)

height = document.getElementById('canvas-frame').clientHeight; //(1)

renderer = new THREE.WebGLRenderer({antialias: true}); //(2)

renderer.setSize(width, height); //(3)

document.getElementById('canvas-frame').appendChild(renderer.domElement); //(4)

renderer.setClearColorHex(0xFFFFFF, 1.0); //(5)

}

**２．设置相机**

OpenGL（WebGL）中、三维空间中的物体投影到二维空间的方式中，存在透视投影和正投影两种相机。 透视投影就是、从视点开始越近的物体越大、远处的物体绘制的较小的一种方式、和日常生活中我们看物体的方式是一致的。 正投影就是不管物体和视点距离，都按照统一的大小进行绘制、在建筑和设计等领域需要从各个角度来绘制物体，因此这种投影被广泛应用。在 Three.js 也能够指定透视投影和正投影两种方式的相机。 本文按照以下的步骤设置透视投影方式。

(0) 声明全局的变量（对象）  
(1) 设置透视投影的相机  
(2) 设置相机的位置坐标  
(3) 设置相机的上为「z」轴方向  
(4) 设置视野的中心坐标

var camera; //(0)

function initCamera() {

var camera = new THREE.PerspectiveCamera( 45 , width / height , 1 , 10000 ); //(1)

camera.position.x = 100; //(2)

camera.position.y = 20; //(2)

camera.position.z = 50; //(2)

camera.up.x = 0; //(3)

camera.up.y = 0; //(3)

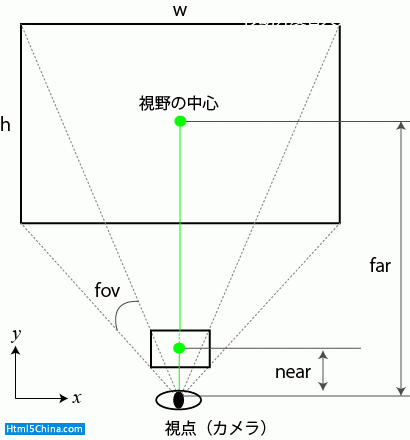
camera.up.z = 1; //(3)

camera.lookAt( {x:0, y:0, z:0 } ); //(4)

}

**透视投影模式下相机的设置**

透视投影中，会把称为视体积领域中的物体作成投影图。 视体积是通过以下4个参数来指定。  
视野角：fov  
纵横比：aspect  
相机离视体积最近的距离：near  
相机离视体积最远的距离：far



通过上面4个参数确定的透视投影设置相机。 默认情况下相机的上方向为Y轴，右方向为X轴，沿着Z轴朝里。

var camera = new THREE.PerspectiveCamera( fov , aspect , near , far );

通过 Three.js 的 「PerspectiveCamera」类声明对象[camera]，同时我们可以修改对象[camera]的各种属性值来设置相机的详细属性。

**设置相机的位置坐标**

相机的位置坐标和视野的中心坐标，按照(2)那样的方式进行设置。 和上面(2)的方式一样，下面这样的方法也可以。

camera.position.set(50,50,100); //(2)

**设置相机的上方向**

相机的上方向按照(3)这样的方式设置对象的属性。

**设置相机的视野中心**

利用[lookAt]方法来设置相机的视野中心。 「lookAt()」的参数是一个属性包含中心坐标「x」「y」「z」的对象。 「lookAt()」方法不仅是用来设置视点的中心坐标、 在此之前设置的相机属性要发生实际作用，也需要调用 [lookAt] 方法。

**动作确认**

请尝试修改(1),(2),(3),(4),(5)、来确认相机各个设定的实际作用。

**其他投影方式**

在 Three.js 中、有各种各样的类，用来来实现透视投影、正投影或者复合投影（透视投影和正投影）这样的相机。

var camera = THREE.OrthographicCamera = function ( left, right, top, bottom, near, far ) //正投影

var camera = THREE.CombinedCamera = function ( width, height, fov, near, far, orthonear, orthofar ) //複合投影

**３．声明场景**

场景就是一个三维空间。 用 [Scene] 类声明一个叫 [scene] 的对象。

var scene;

function initScene() {

scene = new THREE.Scene();

}

**４．设置光源并追加到场景**

OpenGL（WebGL）的三维空间中，存在点光源和聚光灯两种类型。 而且，作为点光源的一种特例还存在平行光源(无线远光源)。另外，作为光源的参数还可以进行 [环境光] 等设置。 作为对应， Three.js中可以设置 [点光源(Point Light)] [聚光灯(Spot Light)] [平行光源(Direction Light)]，和 [环境光(Ambient Light)]。 和OpenGL一样、在一个场景中可以设置多个光源。 基本上，都是环境光和其他几种光源进行组合。 如果不设置环境光，那么光线照射不到的面会变得过于黑暗。 本文中首先按照下面的步骤设置平行光源，在这之后还会追加环境光。  
(0) 声明全局变量(对象)  
(1) 设置平行光源  
(2) 设置光源向量  
(3) 追加光源到场景

var light; //(0)

function initLight() {

light = new THREE.DirectionalLight(0xFF0000, 1.0, 0); //(1)

light.position.set( 100, 100, 200 ); //(2)

scene.add(light); //(3)

}

用「DirectionalLight」类声明一个叫 [light] 的对象来代表平行光源，之后在构造函数中进行属性设置

var light = new THREE.DirectionalLight( hex, intensity)

「hex」是用来以16进制指定光源的颜色。默认是白色「0xFFFFFF」。 「intensity」是光源的强度。默认为1。 另外，还需要通过对象的属性来设置平行光源的方向。

light.position.set( x, y, z );

光源方向向量是从通过[set()]方法指定的点(x, y, z)开始到 点(0, 0, 0) 的方向，和向量的大小没有关系。 和相机的位置坐标设置一样，直接修改「position.x, position.y, position.z」属性也可以。 最后通过 [add] 方法追加光源到场景中去。

**动作确认**

请试着修改(1),(2)来明确平行光源的方向以及光线的颜色设置原理。

※注意：三维世界中物体的颜色不只是由光源的颜色来决定，而且还和物体本身的颜色有关。 之后会再次说明， 因为这个立方体的颜色为红色(0xFF0000),光源中即使存在绿色或者蓝色，立方体看起来的颜色也不会改变。也就是说， 就这个立方体来说，光源的颜色不论是「0xFFFF00」还是「0xFF0000」看起来颜色都没有变化。

**其他光源**

var light = new THREE.AmbientLight( hex ); //(1)環境光源

var light = new THREE.PointLight( hex, intensity, distance ); //(2)点光源

var light = new THREE.SpotLight( hex, intensity, distance, castShadow ); //(3)スポットライト光源

**５．声明立方体并追加到场景**

终于要向三维空间中追加物体了。

var cube; //(0)

function initObject(){

cube = new THREE.Mesh( //(1)

new THREE.CubeGeometry(50,50,50), //(2)

new THREE.MeshLambertMaterial({color: 0xff0000}) //(3)

);

scene.add(cube); //(4)

cube.position.set(0,0,0); //(5)

}

(1) 声明 [Mesh] 类的对象 [cube]。 设置构造函数的第一个参数「形状对象（geometory）」、第二个参数[材质对象（material）]。 第一参数中设置的代表立方体的形状对象，通过「CubeGeometry」类可以创建。

var geometry = THREE.CubeGeometry ( width, height, depth, segmentsWidth, segmentsHeight, segmentsDepth, materials, sides );

构造函数中的「width」「height」「depth」代表立方体的宽高及深度，其他的参数可以省略。 第二个参数中代入的材质对象， 本文中使用的是反射来自光源的光线的材质 「MeshLambertMaterial」类创建的对象。

var geometry = THREE.MeshLambertMaterial( parameters );

构造函数中的参数是联想队列。 本例中、物体的颜色指定为「color: 0xff0000」。另外，通过联想队列参数还可以指定环境光的颜色。关于这些之后我们再介绍。 通过步骤(4),(5)追加立方体到场景，并指定位置坐标。

**动作确认**

请试着修改(2),(3),(5)来明确怎么设置立方体的大小， 颜色和位置。

**其他形状类**

THREE.CubeGeometry ( width, height, depth, segmentsWidth, segmentsHeight, segmentsDepth, materials, sides );//立方体

THREE.CylinderGeometry ( radiusTop, radiusBottom, height, segmentsRadius, segmentsHeight, openEnded );//円錐型

THREE.OctahedronGeometry ( radius, detail ) //八面体

THREE.PlaneGeometry ( width, height, segmentsWidth, segmentsHeight ); //平面型

THREE.SphereGeometry ( radius, segmentsWidth, segmentsHeight, phiStart, phiLength, thetaStart, thetaLength );//球型

THREE.TorusGeometry ( radius, tube, segmentsR, segmentsT, arc )//トーラス型

**６．执行渲染（THREESTART() ）**

最后，按照已设定的相机来观察场景中的物体的方式来进行实际绘制。

function threeStart() {

initThree(); //(0)

initCamera(); //(0)

initScene(); //(0)

initLight(); //(0)

initObject(); //(0)

renderer.clear(); //(1)

renderer.render(scene, camera); //(2)

}

**稍微再试一下（追加平面和球体）**

在上面的例子中再追加平面和球体。

var cube, sphere, plane;

function initObject(){

cube = new THREE.Mesh(

new THREE.CubeGeometry(50,50,50),

new THREE.MeshLambertMaterial({color: 0xff0000})

);

scene.add(cube);

cube.position.set(0,-50,0);

sphere = new THREE.Mesh(

new THREE.SphereGeometry(20,20,20),

new THREE.MeshLambertMaterial({color: 0x00ff00})

);

scene.add(sphere);

sphere.position.set(0,0,0);

plane = new THREE.Mesh(

new THREE.PlaneGeometry(50, 50),

new THREE.MeshLambertMaterial({color: 0x0000ff})

);

scene.add(plane);

plane.position.set(0,50,0);

}

**执行结果**

**Three.js 实现动画**

通过 WebGL，也能够像电视机一样连续绘制静态画像(frame)来实现动画效果。 WEB浏览器中能够按照一定时间间隔调用Javascript函数。最大可以支持60fps(fps就是一秒中的帧数）（译者注：这里是特定指用 window.requestAnimationFrame这样的方式）。 当然根据所调用函数的执行时间，帧数可能会下降。接下来说一下具体如何实现动画效果。

**使用「WINDOW.REQUESTANIMATIONFRAME」函数实现动画**

「window.requestAnimationFrame」函数能够在一定时间间隔后调用指定函数。这个函数只有在所在浏览器页正在被浏览的时候才会执行，因此更加节省资源。

function loop() {

（処理）

window.requestAnimationFrame(loop);

}

看下上面的例子就能明白，「window.requestAnimationFrame」和 loop 函数配合形成一个无限循环的调用。

function threeStart() {

（処理）

loop();

}

按照这样连续绘制就实现了动画效果。

**例２：TUTORIAL2.HTML（实现动画）**

拷贝&粘贴下面这段javascript代码，看一看效果吧。

var width, height;

var renderer;

function initThree() {

width = document.getElementById('canvas-frame').clientWidth;

height = document.getElementById('canvas-frame').clientHeight;

renderer = new THREE.WebGLRenderer({antialias: true});

renderer.setSize(width, height );

document.getElementById('canvas-frame').appendChild(renderer.domElement);

renderer.setClearColorHex(0xFFFFFF, 1.0);

}

var camera;

function initCamera() {

camera = new THREE.PerspectiveCamera( 45 , width / height , 1 , 10000 );

camera.position.x = 400;

camera.position.y = 20;

camera.position.z = 50;

camera.up.x = 0;

camera.up.y = 0;

camera.up.z = 1;

camera.lookAt( {x:0, y:0, z:0 } );

}

var scene;

function initScene() {

scene = new THREE.Scene();

}

var light;

function initLight() {

light = new THREE.DirectionalLight(0xFF0000, 1.0, 0);

light.position.set( 100, 100, 200 );

scene.add(light);

}

var cube = Array();

function initObject(){

for(var i=0; i<3; i++){

cube[i] = new THREE.Mesh(

new THREE.CubeGeometry(50,50,50), //形状の設定

new THREE.MeshLambertMaterial({color: 0xff0000}) //材質の設定

);

scene.add(cube[i]);

cube[i].position.set(0,-100+100\*i,0);

}

}

var t=0;

function loop() {

t++;

cube[0].rotation.set( t/100, 0, 0 );

cube[1].rotation.set( 0, t/100, 0 );

cube[2].rotation.set( 0, 0, t/100 );

renderer.clear();

renderer.render(scene, camera);

window.requestAnimationFrame(loop);

}

function threeStart() {

initThree();

initCamera();

initScene();

initLight();

initObject();

loop();

}

例2的执行结果： 三个立方体分别按照「x」「y」「z」轴进行回转。

**例２的解说**

和例1相比有三处新追加的地方

(1) 创建多个对象  
(2) 「loop()」函数的定义和执行  
(3) 追加指定物体回转角度的方法

**创建多个对象**

例２中通过数组来保存立方体。 Javascript 中数组可以按照

var cube = Array();

这样的方式进行声明。 和C语言不一样，Javascript中定义数组对象不用指定长度。

配列名[識別子].position.set(x, y, z);

**「LOOP()」函数的定义和执行**

「loop()」函数中通过「window.requestAnimationFrame(loop);」实现无限循环调用。 在本节开始的地方已接触过这种方式。 如果不让物体动起来，即使不断重复绘制，必然还是同样的静态的画面。 让例2中的物体回转起来。

**指定物体的回转角度的方法**

一般来说通过指定物体的位置和回转角度，三维空间中的物体就能展示任意的姿势。 例１中、接触过指定物体位置坐标的方法「position.set(x,y,z)」。 例２中、会同时指定物体的位置坐标和回转角度。 利用 [Mesh]类的对象的方法，让物体回转起来。

オブジェクト名.rotation.set( theta\_x, theta\_y, theta\_z );

而且、通过「theta\_x」「theta\_y」「theta\_z」可以实现「x軸」「y軸」「z軸」各种角度的回转。 单位是弧度。

オブジェクト名.rotation.x = theta\_x;

オブジェクト名.rotation.y = theta\_y;

オブジェクト名.rotation.z = theta\_z;

实现动画的时候，值的变更一定要在渲染之前进行。 另外，渲染之前不要忘记了 「renderer.clear();」。如果不执行「renderer.clear();」 ， 上一帧绘制的内容就会遗留下来。

**例２－２：TUTORIAL2\_2.HTML（相机的移动）**

在上一节、实现了物体的移动动画效果、按照同样的步骤也可以来移动相机（视点）。 在例2的基础上，追加了相机(视点）的移动。

var t=0;

function loop() {

t++;

renderer.clear();

cube[0].rotation.set( t/100, 0, 0 );

cube[1].rotation.set( 0, t/100, 0 );

cube[2].rotation.set( 0, 0, t/100 );

camera.position.set( 400\*Math.cos(t/100), 400\*Math.sin(t/200), 50\*Math.cos(t/50));

camera.lookAt( {x:0, y:0, z:0 } );

renderer.render(scene, camera);

window.requestAnimationFrame(loop);

}

可以看到相机（视点）发生了移动。 关于代码中，调用了Javascript中[Math]类的一些方法， 这些方法应该曾经在学校数学中学习过。（一览[点击这里](http://www.scollabo.com/banban/java/ref_19.html)） 比如

Math.PI //\pi

Math.cos(theta) //\cos(\theta)

Math.asin(x) //\arcsin(x)

Math.pow(x,2) //x^2

Math.sqrt(x) //\sqrt{x}

Math.exp(x) //\exp{x}

>等。 更改相机参数的时候、一定要在「camera.lookAt」函数之前进行。 因为「camera.lookAt」函数、不仅仅是设置视点的中心、还能够让设置的相机参数发生实际的效果。

**例２－３：TUTORIAL2\_3.HTML（设置环境光）**

在Three.js里，光源有直接照射到物体的光源（平行光源，聚灯光源，点光源）和间接照射到物体的环境光源。 在此之前设置了直接光源， 本节开始设置环境光。 首先，确认一下加上环境光后物体的照射例子。

和例２－２相比、可以看出直接光照射不到的地方变得明亮一些了。 现实中，关于环境光不是光源直接照射形成的，而是因为各种物体漫反射形成的。 在计算机图形处理世界中，这样间接照射在物体上的光线被称为环境光。 在OpenGL中，环境光是通过直接光源的参数添加进去的，而Three.js中把环境光定义为光源的一种。 具体来说像下面这样向平行光源添加环境光。

var light, light2;

function initLight() {

light = new THREE.DirectionalLight(0xFF0000, 1.0, 0);

light.position.set( 50, 50, 200 );

scene.add(light);

light2 = new THREE.AmbientLight(0x555555);

scene.add(light2);

}

环境光和直接光都一样，利用相关构造函数声明对象然后添加到场景。 另外，环境光的强度可以通过构造函数的参数来指定（上面的例子中：0×555555）。 同时要让环境光照射到物体，只是设置光源是不够的，还需要向物体的材质中设置 [ambient] 属性。

var cube = Array();

function initObject(){

for(var i=0; i<3; i++){

cube[i] = new THREE.Mesh(

new THREE.CubeGeometry(50,50,50),

new THREE.MeshLambertMaterial({color: 0xff0000, ambient:0xFF0000})

);

scene.add(cube[i]);

cube[i].position.set(0,-100+100\*i,0);

}

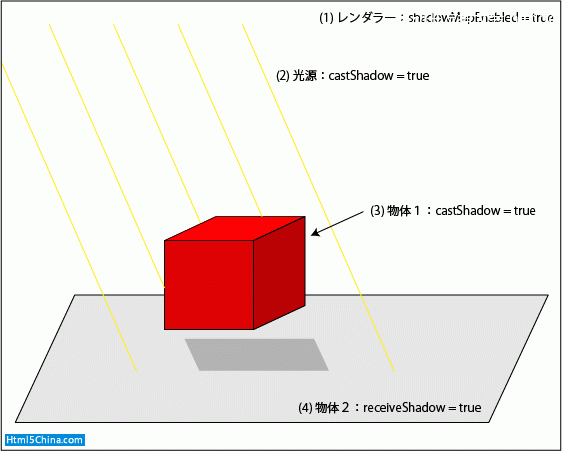
}

声明物体材质的构造函数中的参数里，设置 [ambient] 属性。 这个属性，通过16进制指定了对于环境光反射的强度。简单说，实际物体绘制的颜色，是根据环境光 和这个属性的乘积决定的， 如果环境光和ambient属性都是「0xFFFFFF」的话，那么物体将把环境光都反射出去而看起来为白色。

**例２－４：TUTORIAL2\_4.HTML（影子的设置）**

WebGL（OpenGL）中，没有绘制相对于某个光源的物体阴影的API, 需要使用其它途径来计算。 利用Three.js, 只要简单设置一下就可以实现。

设置物体的阴影， 需要设置下面这4个对象属性。  
(1) 渲染器  
(2) 光源  
(3) 生成阴影的原物体  
(4) 绘制阴影的物体  
模式图就象下面这样。



具体就是在各函数中追加一些对象的属性。

function initThree() {

（省略）

renderer.shadowMapEnabled = true;//影をつける(1)

}

function initLight() {

（省略）

light.castShadow = true;//影をつける（2）

（省略）

}

function initObject(){

（省略）

cube[i].castShadow = true;//影をつける（3）

（省略）

plane.receiveShadow = true; //影をつける（4）

（省略）

}

使用阴影的时候需要注意的是，阴影的计算成本非常的高， 过于使用阴影功能会影响处理速度。

**例２－５：TUTORIAL2\_5.HTML（纹理）**

利用Three.js 能够非常简单就实现纹理的粘贴。

到这里指定颜色物体的材质中，按照选择的纹理就能够自动进行纹理粘贴。 具体的代码如下。

var sphere, sphere2 ;

function initObject(){

var texture1 = new THREE.ImageUtils.loadTexture('earthmap1k.jpg');

sphere1 = new THREE.Mesh(

new THREE.SphereGeometry(50,50,50),

new THREE.MeshLambertMaterial({map: texture1})

);

scene.add(sphere1);

sphere1.position.set(0,0,0);

var texture2 = new THREE.ImageUtils.loadTexture('moonmap1k.jpg');

sphere2 = new THREE.Mesh(

new THREE.SphereGeometry(5,20,20),

new THREE.MeshLambertMaterial({map: texture2})

);

scene.add(sphere2);

}

但是，因为WebGL禁止使用跨域纹理资源，所以也不能读取本地的纹理资源。

**鼠标事件**

**鼠标事件例**

|  |  |
| --- | --- |
| 事件名 | 意思 |
| onmousedown | 鼠标被按下的时候 |
| onmouseup | 鼠标弹上的时候 |
| onmousemove | 鼠标移动的时候 |
| onmouseover | 鼠标移动到物体的上面的时候 |
| onmouseout | 鼠标从物体上面移出物体范围的时候 |

利用鼠标事件，在浏览器中进行特定的动作时， 就能够调用Javascript功能。 而且，事件发生时鼠标的位置也能够获取到。 关于鼠标事件的处理，具体参照下面。

window.onmousedown = function (ev){

var x1 = ev.clientX; //マウスのx座標（ブラウザ上の座標）の取得

var y1 = ev.clientY; //マウスのy座標（ブラウザ上の座標）の取得

var x2 = ev.screenX; //マウスのx座標（ディスプレイ上の座標）の取得

var y2 = ev.screenY; //マウスのy座標（ディスプレイ上の座標）の取得

}

**例３：TUTORIAL3.HTML（鼠标事件）**

var down = false;

var sy = 0, sz = 0;

window.onmousedown = function (ev){ //マウスダウン

if (ev.target == renderer.domElement) {

down = true;

sy = ev.clientX; sz = ev.clientY;

}

};

window.onmouseup = function(){ //マウスアップ

down = false;

};

window.onmousemove = function(ev) { //マウスムーブ

var speed = 2;

if (down) {

if (ev.target == renderer.domElement) {

var dy = -(ev.clientX - sy);

var dz = -(ev.clientY - sz);

camera.position.y += dy\*speed;

camera.position.z -= dz\*speed;

sy -= dy;

sz -= dz;

}

}

}

window.onmousewheel = function(ev){ //マウスホイール

var speed = 0.2;

camera.position.x += ev.wheelDelta \* speed ;

}

到这里为止，对如何使用 Three.js 进行了简单概要性的介绍。 下一节就进入如何实现物理模拟的环节了。

**你可能喜欢阅读以下文章**

* [THREE.js 学习笔记（一）](http://www.html5china.com/HTML5features/WebGL/20111129_2989.html)
* [3D格式，难得的文章](http://www.html5china.com/HTML5features/WebGL/20111129_2988.html)
* [如何学习Three.JS 3D引擎](http://www.html5china.com/HTML5features/WebGL/20111129_2987.html)
* [WebGL All in One 全傻瓜简介](http://www.html5china.com/HTML5features/WebGL/20111129_2985.html)

**【责任编辑：小想】**

>

数据统计中！！

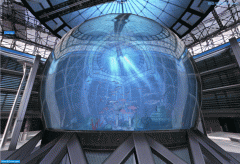
上一篇：没有了   
下一篇：没有了   
[返回WebGL](http://www.html5china.com/HTML5features/WebGL/)

分享到： QQ空间 腾讯微博 新浪微博 开心网 豆瓣 打印 收藏夹 [更多](http://www.jiathis.com/share) 5

**编辑推荐文章**

* [利用HTML5构筑物理模拟环境～ WebGL库Three.js入门](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html)
* [使用粒子系统来表现音乐旋律](http://www.html5china.com/HTML5features/WebGL/20111129_2990.html)
* [THREE.js 学习笔记（一）](http://www.html5china.com/HTML5features/WebGL/20111129_2989.html)
* [3D格式，难得的文章](http://www.html5china.com/HTML5features/WebGL/20111129_2988.html)
* [如何学习Three.JS 3D引擎](http://www.html5china.com/HTML5features/WebGL/20111129_2987.html)
* [WebGL All in One 全傻瓜简介](http://www.html5china.com/HTML5features/WebGL/20111129_2985.html)

[**利用HTML5构筑物理模拟环境～ WebGL库Three.js入门**](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html)

[](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html)

这是目前质量较好、讲解比较清楚的一篇 Three.js 的入门教程，原作者是日本人遠藤理平，由 HiWe...

**网友评论TOP10**

[查看所有评论](http://www.html5china.com/plus/feedback.php?aid=3611)

窗体顶端

**我要评论请自觉遵守互联网相关的政策法规，严禁发布色情、暴力、反动的言论。**



验证码: 

窗体底端

窗体顶端

窗体底端

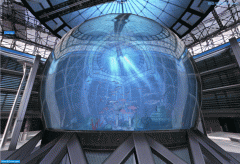
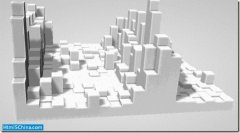
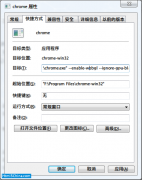
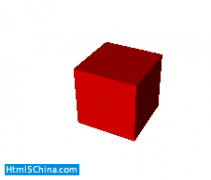
**更多相关文章**

* [THREE.js 学习笔记（一）](http://www.html5china.com/HTML5features/WebGL/20111129_2989.html)
* [3D格式，难得的文章](http://www.html5china.com/HTML5features/WebGL/20111129_2988.html)
* [如何学习Three.JS 3D引擎](http://www.html5china.com/HTML5features/WebGL/20111129_2987.html)
* [WebGL All in One 全傻瓜简介](http://www.html5china.com/HTML5features/WebGL/20111129_2985.html)
* [使用 Three.js 绘制三维树模型](http://www.html5china.com/HTML5features/WebGL/20111101_2312.html)

**更多推荐文章**

* [HTML5 2D游戏性能测试分析](http://www.html5china.com/HTML5features/WebGL/20111129_2984.html)
* [使用 Three.js 绘制三维树模型](http://www.html5china.com/HTML5features/WebGL/20111101_2312.html)
* [chrome 13， FireFox 5 浏览器不支持WEBGL设置](http://www.html5china.com/HTML5features/WebGL/20111005_2170.html)
* [chrome、firefox如何关闭WebGL禁用webGL](http://www.html5china.com/HTML5features/WebGL/20111005_2169.html)
* [GLGE系列教程–第12节：设置六面贴图的场景](http://www.html5china.com/HTML5features/WebGL/20111005_2168.html)

**栏目热门**

* [](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html)  
  [利用HTML5构筑物理模拟环境～ We](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html)
* [](http://www.html5china.com/HTML5features/WebGL/20111129_2990.html)  
  [使用粒子系统来表现音乐旋律](http://www.html5china.com/HTML5features/WebGL/20111129_2990.html)
* [](http://www.html5china.com/HTML5features/WebGL/20111129_2989.html)  
  [THREE.js 学习笔记（一）](http://www.html5china.com/HTML5features/WebGL/20111129_2989.html)
* [](http://www.html5china.com/HTML5features/WebGL/20111129_2985.html)  
  [WebGL All in One 全傻瓜简介](http://www.html5china.com/HTML5features/WebGL/20111129_2985.html)
* [](http://www.html5china.com/HTML5features/WebGL/20111101_2312.html)  
  [使用 Three.js 绘制三维树模型](http://www.html5china.com/HTML5features/WebGL/20111101_2312.html)

热门标签：　[Html5下载](http://www.html5china.com/plus/search.php?keyword=Html5%E4%B8%8B%E8%BD%BD) 　[开发工具](http://www.html5china.com/plus/search.php?keyword=%E5%BC%80%E5%8F%91%E5%B7%A5%E5%85%B7) 　[HTML5工具](http://www.html5china.com/plus/search.php?keyword=HTML5%E5%B7%A5%E5%85%B7) 　[canvas](http://www.html5china.com/plus/search.php?keyword=canvas) 　[html5](http://www.html5china.com/plus/search.php?keyword=html5) 　[html5视频](http://www.html5china.com/plus/search.php?keyword=html5%E8%A7%86%E9%A2%91) 　[css3](http://www.html5china.com/plus/search.php?keyword=css3) 　[HTML5 下载](http://www.html5china.com/plus/search.php?keyword=HTML5+%E4%B8%8B%E8%BD%BD) 　[视频](http://www.html5china.com/plus/search.php?keyword=%E8%A7%86%E9%A2%91) 　[webgl](http://www.html5china.com/plus/search.php?keyword=webgl)

**头条**[**THREE.js 学习笔记（一）**](http://www.html5china.com/HTML5features/WebGL/20111129_2989.html)

[](http://www.html5china.com/HTML5features/WebGL/20111129_2989.html)一. 获取THREE.js three.js的代码托管在github上面， https://github.com/mrdoob/three.js/ 我们可以用...

* [THREE.js 学习笔记（一）](http://www.html5china.com/HTML5features/WebGL/20111129_2989.html)
* [3D格式，难得的文章](http://www.html5china.com/HTML5features/WebGL/20111129_2988.html)

**栏目热点**

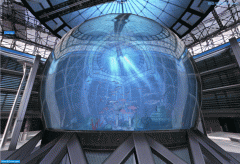
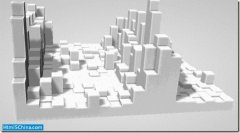
* [WebGL教程：第2课，添加颜色](http://www.html5china.com/HTML5features/WebGL/20110921_1882.html)
* [WebGL教程：第8课，深度缓存、透明度和混合](http://www.html5china.com/HTML5features/WebGL/20110921_1888.html)
* [WebGL系列中文教程！（共16课）](http://www.html5china.com/HTML5features/WebGL/20110921_1879.html)
* [WebGL教程：第1课，三角与方块的故事](http://www.html5china.com/HTML5features/WebGL/20110921_1881.html)
* [WebGL教程：第9课，优化代码结构实现多物体](http://www.html5china.com/HTML5features/WebGL/20110921_1889.html)
* [WebGL教程：第0课，从零开始](http://www.html5china.com/HTML5features/WebGL/20110921_1880.html)
* [WebGL教程：第5课，引入纹理贴图](http://www.html5china.com/HTML5features/WebGL/20110921_1885.html)
* [WebGL教程：第3课，动起来！](http://www.html5china.com/HTML5features/WebGL/20110921_1883.html)

**热点视频教程**

[](http://www.html5china.com/video/basic/20110924_1958.html)[HTML5入门视频教程：第一章，概论](http://www.html5china.com/video/basic/20110924_1958.html)

[](http://www.html5china.com/video/basic/20110925_1963.html)[HTML5入门视频教程：第六章，实例讲解](http://www.html5china.com/video/basic/20110925_1963.html)

**最新文章**

* [](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html)  
  [利用HTML5构筑物理模拟环境～ WebGL库Three.js入门](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html)
* [](http://www.html5china.com/HTML5features/WebGL/20111129_2990.html)  
  [使用粒子系统来表现音乐旋律](http://www.html5china.com/HTML5features/WebGL/20111129_2990.html)
* [利用HTML5构筑物理模拟环境～ WebGL库Three.js入门](http://www.html5china.com/HTML5features/WebGL/20120509_3611.html)
* [使用粒子系统来表现音乐旋律](http://www.html5china.com/HTML5features/WebGL/20111129_2990.html)
* [THREE.js 学习笔记（一）](http://www.html5china.com/HTML5features/WebGL/20111129_2989.html)
* [3D格式，难得的文章](http://www.html5china.com/HTML5features/WebGL/20111129_2988.html)
* [如何学习Three.JS 3D引擎](http://www.html5china.com/HTML5features/WebGL/20111129_2987.html)

**推荐图书**

[](http://www.html5china.com/book/basic/20120115_3348.html)

书名：[HTML5 Canvas基础教程](http://www.html5china.com/book/basic/20120115_3348.html)  
出版社：人民邮电出版社  
作者：盛海艳  
页数：

* [HTML5开发精要与实例详解](http://www.html5china.com/book/basic/20120114_3347.html)
* [HTML5和CSS3实例教程](http://www.html5china.com/book/basic/20120114_3346.html)
* [HTML5与CSS3实战指南](http://www.html5china.com/book/basic/20120114_3345.html)
* [jQuery开发技术详解(含CD光盘1张)](http://www.html5china.com/book/basic/20111110_2368.html)

**最热跟帖**

* [国人写的强大的html5植物大战僵尸（源码）](http://bbs.html5china.com/thread-5-1-1.html)
* [老外写的3D旋转的不可思议的俄罗斯方块(源码)](http://bbs.html5china.com/thread-8-1-1.html)
* [HTML5开发的太阳系演示！源码下载！](http://bbs.html5china.com/thread-417-1-1.html)
* [html5从入门到精通](http://bbs.html5china.com/thread-433-1-1.html)
* [一步一步学做HTML5游戏教程讨论](http://bbs.html5china.com/thread-2-1-1.html)
* [德州21点扑克牌（源码，适合用来学习）](http://bbs.html5china.com/thread-11-1-1.html)
* [撞撞球（源码）](http://bbs.html5china.com/thread-9-1-1.html)
* [曲线谜团(经典游戏，经典源码)](http://bbs.html5china.com/thread-15-1-1.html)
* [关于我们](http://www.html5china.com/help/about.html)|
* [广告服务](http://www.html5china.com/help/adservice.html)|
* [友情链接](http://www.html5china.com/plus/flink.php)|
* [网站地图](http://www.html5china.com/sitemap.html)|
* [HTML5中文论坛](http://bbs.html5china.com)

Copyright © 2010-2011 [HTML5中文网](http://www.html5china.com) 版权所有 粤ICP备10202793号-1   
Powered by [html5China Team](http://www.html5china.com/help/about.html) 合作联系QQ 1294899260 [站长统计](http://www.cnzz.com/stat/website.php?web_id=2297124)