WebGL KAIFA YU YINGYONG

WebGL 开发与应用

郑 华 张云佐 著

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

WebGL 开发与应用

郑　华　张云佐　著

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

WebGL 是一项新的 Web 3D 图形标准，也是 HTML5 大家庭中的一员。本书从 WebGL 和 3D 图形学的基础概念讲起，循序渐进，用简单的实例直观地讲解了各个知识点，随后又理论结合实际，介绍了在现实开发环境中需要注意的各种问题，在最后一章，把全书所有讲过的知识综合到一起，讲解了一个完整的基于 WebGL 的 MIS 系统开发过程，让读者能够基本了解 WebGL 这一新技术的开发流程，以便读者可以独立开发自己的 WebGL 应用程序。

本书适合作为高等院校计算机相关专业 WebGL 类课程的教材，也适合 Web 开发人员及对 3D 开发感兴趣的读者参考阅读。

图书在版编目（CIP）数据

WebGL 开发与应用 / 郑华，张云佐著．—北京：中国铁道出版社，2017.12

ISBN 978-7-113-24082-0

Ⅰ. ①W… Ⅱ. ①郑… ②张… Ⅲ. ①网页制作工具－程序设计－高等职业教育－教材 Ⅳ. ①TP392.092.2

中国版本图书馆 CIP 数据核字（2017）第 297810 号

书　　名： WebGL 开发与应用
作　　者： 郑　华　张云佐　著

策　　划： 侯　伟　孙晨光
责任编辑： 秦绪好　徐盼欣
封面设计： 刘　颖
责任校对： 张玉华
责任印制： 郭向伟

读者热线：（010）63550836

出版发行： 中国铁道出版社（100054，北京市西城区右安门西街 8 号）
网　　址： http://www.tdpress.com/51eds/
印　　刷： 虎彩印艺股份有限公司
版　　次： 2017 年 12 月第 1 版　2017 年 12 月第 1 次印刷
开　　本： 787 mm×1 092 mm　1/16　**印张：** 10.75　**字数：** 245 千
书　　号： ISBN 978-7-113-24082-0
定　　价： 28.00 元

PREFACE 前言

2012年底，在一个MIS系统开发过程中，编者碰到了一个如何在Web页面中展示三维模型的问题，在多次的方案讨论过程中，偶然发现了WebGL这个新名词，当时编者并不清楚它究竟能干什么，也不清楚如何使用它，只是知道它是一个面向Web的无插件的3D引擎。之后的两年中，编者不断探索WebGL的开发问题，并逐步接触了Three.js引擎、Sim.js框架等基于WebGL的开发工具。随着微软Internet Explorer 11.0对WebGL的支持，WebGL迎来了新一轮发展的春天，正是在这种形式下，编者萌生了撰写《WebGL开发与应用》一书的想法，因为在国内，这方面的资料非常稀少，几乎没有可供参考的书目。

本书共分6章，分别是：入门篇、基础篇、交互篇、动画篇、应用篇和基于WebGL的MIS系统开发。在由浅入深地对WebGL的各个知识点进行了讲解之后，在最后一章重点讲解了如何将WebGL与传统的软件开发相结合的问题，在传统的Web3D方案中，它们都是依赖插件的，而插件由各公司或厂商主导，WebGL的兴起彻底地解决了这个问题。本书可以在一定程度上推进这一新技术的应用，在虚拟现实、管理信息系统等方面带来全新的用户体验。

WebGL是一个全新的领域，其标准由科纳斯（Khronos）组织开发和维护，该组织致力于软件的开源，基于WebGL的很多二次开发包也都是开源的，这也导致了帮助文档严重不足的现状。在编写本书的过程中，编者不得不通过反复阅读源代码的方式去理解其内部的工作原理，这为本书的编写增加了不少难度。最终成稿时，还参考了很多互联网资料；请教了石家庄铁路职业技术学院的一些数学老师；邀请了一些3D建模方向的学生，做了大量的试验。没有他们的无偿帮助，将难以完成书稿，在此对他们一并表示感谢。

本书适用于对3D感兴趣的Web开发人员，读者最好具备一些计算机图形学、线性代数和JavaScript编程方面的基础知识。

作　者

2017年11月

目录

目 录 CONTENTS

CONTENTS 目 录

目 录 CONTENTS

CONTENTS 目 录

第1章 入门篇

Web 自 20 世纪 90 年代初诞生以来，经过 20 多年的发展，现在已经成为 Internet 上最重要、最普及的应用，从 HTML1.0 发展到 2.0、3.0、4.0、XHTML 以及现在的 HTML5.0。但至今为止，主流的 Web 页面仍然是二维的，随着 3D 技术的日益普及，Web 技术正朝着 3D 方向发展。传统的 3D 页面开发技术主要包括：

1. Flash 3D

Flash 的前身是 Future Wave 公司的 Future Splash，是世界上第一个商用的二维矢量动画软件，用于设计和编辑 Flash 文档。1996 年，美国 Macromedia 公司收购了 Future Wave，并将其改名为 Flash。2005 年，Macromedia 被 Adobe 公司收购。目前，Flash 动画在 Web 上的应用已经非常普及，Internet 上大多数视频都是基于 Flash 播放器的，如优酷、土豆、酷6、搜狐等。正是由于 Flash 在实际中的普遍应用，Flash 3D 引擎也很多，如 Papervision 3D、Alternativa 3D、Away 3D、Sandy 3D、Flash Player 10 等，但它们都不是开源的，而且必须安装 Flash 插件才能运行。

2. Unity 3D

Unity 是由 Unity Technologies 公司开发的一个可以轻松创建三维视频游戏、建筑可视化、实时三维动画等类型的多平台综合型游戏开发工具，是一个全面整合的专业游戏引擎，可发布游戏至 Windows、Mac、Wii、iPhone 和 Android 平台，也可以利用 Unity Web Player 插件发布网页游戏、手机游戏，支持 Mac 和 Windows 的网页浏览。游戏开发是 Unity 3D 的主要应用，熟悉 C# 或 JavaScript 的开发者，可以轻松入门。其典型的产品有 PC 游戏《新仙剑》《将魂三国》《QQ 乐团》和手机游戏《神庙逃亡》《极限摩托车》《公路战士》等。和 Flash 一样，Unity 也不是开源的，基于网页的应用也需要安装插件才能运行，而且 Unity 3D 的针对性很强，即主要针对游戏开发。

3. CopperCube

CopperCube 是由 Ambiera 公司开发的一款用于创建交互式 3D 场景的软件，著名的 3D 引擎 Irrlicht 即是该公司的产品，它可制作从最简单三维全景到复杂的完整三维游戏。Coppercube 可将场景、资源、逻辑直接导出成引擎支持的场景文件，不需编写代码即可生成简单的游戏或应用，并可由 Irrlicht 引擎导出成 Flash-swf、HTML5-Canvas、Android-app 或者独立的 Windows-exe 文件，是一款很有前途的 3D 开发工具，但 CopperCube 是一个商用软件，

并不是一个公共标准，用户需要购买使用。

4．Silverlight

Silverlight 中文名“微软银光”，是由 Microsoft 推出的一种新的 Web 呈现技术，能在各种平台上运行。对于 Web 用户来说，Silverlight 是一个简单的浏览器插件，用户只要安装了这个插件程序，就可以在 Windows 和 Macintosh 上的多种浏览器中运行相应版本的 Silverlight 应用程序；对于开发者来说，它提供了一套开发框架，并通过使用基于向量的图层技术，支持任何尺寸图像的无缝整合，对基于 asp.net、AJAX 在内的 Web 开发环境实现了无缝连接。Silverlight 运行在 .NET 3.0 以上，它不是开源的，对于 Web 来说仍然需要安装插件。

在工业需求的推进下，HTML5 于 2004 年被 Whatwg 提出，2007 年被 W3C 接纳，2008 年 1 月 HTML5 的第一份正式草案公布，2012 年 12 月 HTML5 规范正式定稿，2013 年 5 月，HTML5.1 正式草案公布。目前，支持 HTML5 的浏览器包括 Firefox、IE9 及其更高版本、Chrome（谷歌浏览器）、Safari、Opera、360 浏览器、搜狗浏览器、QQ 浏览器、猎豹浏览器等。

在图形图像方面，HTML5 新增了 Canvas 标记，允许浏览器直接在上面绘制矢量图形，这意味着用户可以脱离各种插件，直接在浏览器中显示图形或动画。目前 HTML5 和 Canvas 2D 规范的制定已经完成，尽管还不能算是 W3C 标准，但是这些规范已经功能完整，它最终代替多媒体框架（如 Flash）也是一个必然趋势。现在一些主流的大公司都已经逐步转向使用 HTML5。

HTML5 只完成了 Canvas 2D 规范，直接利用 Canvas API 只能进行二维平面图形的绘制，如果想制作 3D 场景，用户必须自行开发 3D 引擎，这对于绝大多数的 Web 开发者来说，仍然是一个巨大的挑战。

2011 年 3 月，多媒体技术标准化组织 Khronos 在美国洛杉矶举办的游戏开发大会上发布了 WebGL 标准规范 R 1.0，支持 WebGL 的浏览器不借助任何插件便可提供硬件图形加速，从而提供高质量的 3D 体验。WebGL 标准已经获得了 Apple（Mac OS Safari Nightly Builds）、Google（Chrome 9.0）、Mozilla（Firefox 4.0 Beta）和 Opera（Preview Build）等的支持。

1.1 什么是 WebGL

WebGL 是一种 3D 绘图标准，这种绘图技术标准允许把 JavaScript 和 OpenGL ES（OpenGL for Embedded Systems）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，并能创建复杂的导航和数据可视化。WebGL 技术标准免去了开发网页专用渲染插件的麻烦，可被用于创建具有复杂 3D 结构的网站页面。

WebGL 标准是由科纳斯（Khronos）组织开发和维护的，该组织同样管理着 OpenGL、COLLADA 等标准。该组织在官网上对 WebGL 的描述是这样的：“WebGL 是免费授权的、跨平台的应用程序接口 API，它将 OpenGL ES 2.0 作为在 HTML 网页内的 3D 绘图环境，作为低级别文档对象模型接口开放。它使用 OpenGL 渲染语言 GLSL ES（OpenGL Shading Language for Embedded Systems，编者注），并可被整洁地与其他 3D 内容上层或下层的网页内容捆绑。它使用 JavaScript 编程开发语言，是进行动态 3D 网页应用开发的理想工具，并已被主流互联

网浏览器集成。”

简言之，WebGL 可以看作将 OpenGL ES（OpenGL 嵌入式版本，针对手机、游戏机等设备相对较轻量级的版本）移植到了网页平台，Chrome、Firefox 等浏览器都实现了对 WebGL 标准的支持（IE 从 11.0 以后正式支持 WebGL），使用 JavaScript 就可以进行代码编写。

1.2 WebGL 的运行环境

WebGL 必须运行在特定的浏览器中。表 1–1 和表 1–2 分别列出了 PC 端和移动端支持 WebGL 的浏览器。

表 1–1 PC 端支持 WebGL 的浏览器

Firefox 4 以上	Chrome 9 以上	Safari 5.1 以上	Opera 12 以上	IE 11.0 以上
支持	支持	支持，需要配置	支持，需要配置	支持

表 1–2 移动端支持 WebGL 的浏览器

Android				iOS 8
Firefoxfor Mobile 4 以后	OperaMobile 12 以后	Google Chrome for Android 25 以后	Android 默认浏览器	
支持	支持	支持	支持，需要配置	支持

WebGL 绘图是基于 HTML5 的 Canvas 标记的，由于不是所有的浏览器都支持 WebGL，因此有必要在程序中加入检测机制，这可以保证程序更加得体地选择退出。下面这段代码可用于检测浏览器是否支持 WebGL：

```
function initWebGL(canvas) {
  var gl;
  try {
    gl = canvas.getContext("experimental-webgl")||canvas.getContext("webgl");
  }
  catch(e) {
    alert('您的浏览器不支持 WebGL');
  }
  return gl;
}
```

最后，需注意两点：一是尽管浏览器支持 WebGL，但一些老旧的计算机可能仍然不能运行 WebGL，因为 WebGL 被设计为直接运行在图形显示卡（GPU）上，因此要求较高性能的显卡；二是浏览器对于多媒体文件的支持程度也会影响其表现，比如 Firefox 支持 Ogg 格式的视频，但 IE 不支持；IE 支持 MP4 格式的视频，但 Firefox 不支持；等等。

1.3 Three.js 引擎

WebGL 的原生 API 是相当低级的，对于大部分传统程序员来说遥不可及，只有有经验的 3D 程序员才能驾驭。现在已经有很多人在做这些代码库的集成工作，通过编写用户友好的接口程序，使得 WebGL 编程变得通俗易懂，一些知名的开发框架有 GLGE（http://www.glge.

org)、SceneJS（http://www.scenejs.org)、CubicVR（http://www.cubicvr.org)、Three.js（http://www.threejs.org）等。在众多的引擎中，Three.js（Ricardo Cabello Miguel，西班牙）成为了佼佼者，它以简单、直观的方式封装了 3D 图形编程中的常用对象，使用了很多图形引擎的高级技巧，极大地提高了性能，且是完全免费和开源的。其主要特点包括：

（1）掩盖了 3D 渲染的细节。Three.js 将原生 API 的细节进行了抽象，将 3D 场景抽象为网格（mesh)、材质（material)、光源（light)、摄像机（camera）等 Object3D 对象。

（2）内置了很多常用 3D 几何对象。例如，球体（SphereGeometry)、立方体（CubeGeometry)、圆柱（CylinderGeometry)、3D 文字（TextGeometry）等，并提供了相关 API，允许用户自由组合这些 3D 对象。

（3）支持交互。Three.js 提供了在 3D 场景中的拾取（pick)、拖动（drag）等操作，使得用户可以轻松创建支持交互的应用。

（4）内置了数学库。Three.js 内置了 3D 图形学中的常用矩阵、投影和矢量运算的数学库，使得没有图形学基础的程序员也可轻松进行 3D 编程。

难能可贵的是，通常情况下更高的封装程度往往意味着灵活性的牺牲，但是 Three.js 在这方面做得很好，几乎不会有 WebGL 支持而 Three.js 实现不了的情况，因此本书主要针对 Three.js 进行说明。

Mr.doob（https://github.com/mrdoob）是 Three.js 项目发起人和主要贡献者之一，但由于 Three.js 是 Github（http://github.com/mrdoob/three.js/）上的一个开源项目，因此有非常多的贡献者。

Three.js 并没有非常明确的设计目标，目前大量的爱好者正在不断修正不良的设计，添加新的功能，最新的版本（大约每两个月左右更新一次）和所有的旧版本都可以在 http://threejs.org/ 上下载，它是完全免费的。

本书的所有示例是基于 Three.js R62 版本的，Three.js 各版本之间并不能完全兼容，在阅读这些示例时请注意这点。

1.4 Sim.js 框架

Three.js 提供了一个中间层来掩盖 WebGL 原生 API 的底层细节，降低了 WebGL 编程的门槛。很多代码都是可以重用的，如创建网格、设置纹理、添加子类、添加鼠标事件等，但在实际应用中，程序员不得不去做大量重复的工作，因此，在大型的程序中，很容易出现逻辑混乱、条理不清、可读性差等问题。

Sim.js（[美]Tony Parisi，https://github.com/tparisi/sim.js）把这些工作抽象成了一个更高等级的可重用对象，它采用面向对象的方式封装了 Three.js 中的常用对象，简化了 Three.js 中许多重复的任务，比如设置渲染器、循环重绘、处理 DOM 时间等。Sim.js 是一个轻量级的开发框架，它是完全开源和免费的。

Sim.js 中包含三个核心的类：

（1）Publisher 类（Sim.Publisher）：封装了所有可触发的事件，处理事件回调，这个类也是后面两个类的基类。

（2）Application 类（Sim.App）：封装了所有建立或删除操作的代码，管理应用中所有的对象列表。

（3）Object 类（Sim.Object）：封装了用户自定义的 3D 对象，例如在场景中增加或移除物体、添加子类、位置变化等。通常情况下，这些类会在 App 类中被实例化，生成三维场景。

大多数情况下，程序通过实例化一个 Application 类和多个 Object 类来构建一个完整的三维场景。利用 Sim.js 框架，程序员可以按照面向对象的方式进行 WebGL 应用的开发。

要注意的是，由于 Three.js 不同版本之间存在细微差别，因此在使用 Sim.js 框架时，要根据版本的不同进行相应的修正。

1.5 关于 Canvas 标记

WebGL 是基于 HTML5 的 Canvas 标记的，尽管 Canvas 2D 规范与本书的主要内容关系不是很大，但了解一些基本的编程接口仍然是很必要的。我们经常会将 Canvas 2D 绘图用于 3D 对象的纹理，这是很常用的一种编程技巧。

下面的例子很随意地在画布上绘制了一些图形元素，代码都很简单，希望通过该例子向读者展示基本的 Canvas 2D 编程方法，并对后期的编程有所帮助。

```
<html>
<head>
<title>Canvas 标记基础 </title>
</head>
<body>
<div style="position:absolute;left:50%;top:50%">
<canvas id="myCanvas" width="800" height="600"
    style="border:1px solid #c3c3c3; position:absolute;
    left:-400px;top:-300px;background-color:#efefef;">
    Your browser does not support the canvas element.
</canvas>
<script type="text/javascript">
    //绘制矩形
    var c=document.getElementById("myCanvas");
    var cxt=c.getContext("2d");
    cxt.fillStyle="#FF0000";
    cxt.fillRect(0,0,200,200);          //左上角坐标、宽度、高度
    //绘制渐变矩形
    var grd=cxt.createLinearGradient(300, 0, 500, 0);//左上角、右下角坐标
    grd.addColorStop(0, "#FF0000"); //开始颜色
    grd.addColorStop(1, "#00FF00"); //结束颜色
    cxt.fillStyle=grd;
    cxt.fillRect(300,0,200,200);
    //绘制圆饼
    cxt.fillStyle="#00ff00";
    cxt.beginPath();
    cxt.arc(100, 300, 40, 0, Math.PI * 2, true);
    cxt.closePath();
    cxt.fill();
```

```
        //加载文字
        cxt.font="30px 隶书";
        cxt.fillStyle="#336699";
        cxt.fillText("Hello", c.width/2 , c.height/2);
        //加载图像
        var img=new Image()
        img.src="welcome.png"
        cxt.drawImage(img, 0, 0);
    </script>
    </div>
    </body>
    </html>
```

1.6 本书的配套资源

一些形象的动画过程无法通过书面材料来传达，因此，编者在编写本书的同时开发了WebGL 中文学习网（http://222.30.167.210），网站的主要内容包括：

- 重要知识点的视频讲解；
- 逐步进阶的网页实例及其源代码；
- 一些综合运用的案例（如赛车游戏）；
- 各种工具下载（如 Three.js 开发包）；
- 学习指导、随堂作业与其他。

尤为重要的是，本书不可能附上每一个网页的所有源代码，只能是对一些核心的代码进行解释和说明，完整的代码可以从网站上免费获得。

最后，由于 IE 11.0 以后才支持 WebGL，因此建议读者使用 Firefox 浏览器来浏览本网站，还需要做一些配置工作以便在 Firefox 中启用 WebGL，具体配置过程请参考 http://222.30.167.210/download/Firefox 开启 Webgl.txt。

1. 参考本章 1.5 节的例子，结合 DIV + CSS 的网页开发布局，制作一个围棋棋盘。

2. 结合本教材配套网站中所介绍的跑车动画（http://222.30.167.210/103.html），尝试开发一个简单的 Canvas 2D 赛车游戏。

第2章 基础篇

WebGL 的原生 API 是相当低级的，程序员必须很清楚 OpenGL 引擎的渲染机制并熟悉 3D 图形学才能开始编程。与 Three.js 相比，实现同样功能的网页，使用原生 WebGL 接口需要 5 倍以上的代码量，而且大多数代码对于初学者来说生涩难懂，这并不是本书的写作目的。

Three.js 则要轻量很多，而且 WebGL 支持的功能，Three.js 几乎都能实现，因此本书主要介绍基于 Three.js 引擎的 Web3D 页面开发技术。本章先介绍 Three.js 中的一些基本概念。

2.1 Three.js 引擎中的基本概念

2.1.1 三维坐标系

三维坐标系是进行 Web3D 开发的基础（图 2-1 中，红、绿、蓝三条射线分别代表了 x、y、z 轴），在 Three.js 中，三维坐标系统如下（右手系）：

x 轴：水平向右；

y 轴：垂直向上；

z 轴：垂直与屏幕向外；

原点：画布中心，即坐标（0，0，0）。

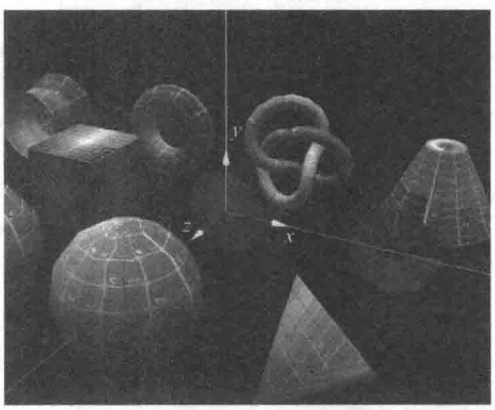


图 2-1 三维坐标系

举例来说，图 2-1 中空心圆饼的坐标是(10,0,0)，空心圆环的坐标是（-5，0，-5），此处的数值单位本身意义并不大，它只是表示了一个相对位置，但通常用米做单位可以达到最好的效果，尤其是在从建模工具导出模型到 Three.js 时。

需要注意的是，这样的坐标系统与大多数的三维建模软件（如 3Dmax、Revit）都是不同的，它们一般在水平面上使用 x 轴和 y 轴，在垂直方向上使用 z 轴，因此，从这些建模软件中导出模型到 Three.js 中时，要进行 y 轴和 z 轴的变换。

2.1.2 透视摄像机

摄像机定义了三维空间到二维屏幕的投影方式，用“摄像机”这样一个类比，可以使我们直观地理解这一投影方式，它反映了在一个 3D 场景中哪部分内容可以显示在用户画布上。

一个典型的透视摄像机（Perspective Camera）包含 4 个参数，比如：

```
var camera = new THREE.PerspectiveCamera( fov , w/h , near , far );
```

其中，fov 代表摄像机的视野广度；w/h 代表摄像机的宽高比，实践中一般设置为画布本身的宽高比；near 代表近视点，小于这个距离的对象不能显示；far 代表远视点，大于这个距离的对象不能显示。通过改变摄像机的位置、朝向和角度，即可改变画布中的内容，比如：

```
camera.position.set(0, 0, 5);
camera.lookAt( {x:0, y:0, z:0 } );
```

这两行代码的意思是将摄像机置于画布正外侧 5 m 的位置上，朝向原点。

2.1.3 正交投影摄像机

使用透视摄像机获得的结果类似于人眼在真实世界中所看到的效果（近大远小），如图 2-2 所示；而使用正交投影摄像机获得的结果则像在几何课上所画的效果，对于在三维空间内平行的线，投影到二维空间中也一定是平行的，如图 2-3 所示。一般说来，对于制图、建模软件通常使用正交投影，这样不会因为投影而改变物体比例；而对于其他大多数应用通常使用透视投影，因为这更接近人眼的观察效果。当然，摄像机的选择并没有对错之分，可以根据应用的特性选择一个效果更佳的摄像机。

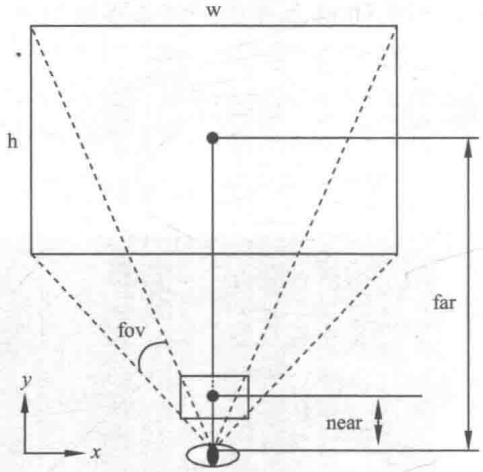
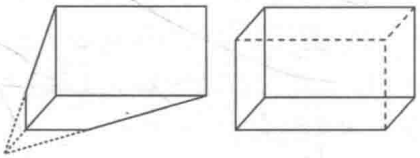


图 2-2 透视摄像机



（a）透视摄像机　（b）正交投影摄像机

图 2-3 三维空间内平行的线

正交投影摄像机（Orthographic Camera）设置起来较为直观，它的构造函数是：

```
THREE.OrthographicCamera(left,right,
top, bottom, near, far) ·
```

这 6 个参数分别代表正交投影摄像机拍摄到的空间的 6 个面的位置，这 6 个面围成一个长方体，我们称其为视景体（Frustum），如图 2-4 所示。只有在视景体内部的物体才可能显示在屏幕上，而视景体外的物体会在显示之前被裁剪掉。

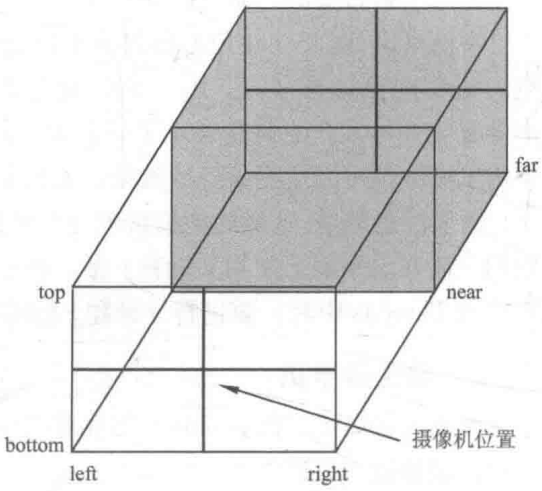


图 2-4 正交投影摄像机视景体