HZ BOOKS
PEARSON

开发人员专业技术丛书

经典的OpenGL红宝书，涵盖OpenGL 3.0和3.1的最新特性

（原书第7版）

OpenGL编程指南

OpenGL Programming Guide Seventh Edition

（美）Dave Shreiner The Khronos OpenGL ARB Working Group 著

李军 徐波 等译

机械工业出版社
China Machine Press

PEARSON Education

开发人员专业技术丛书

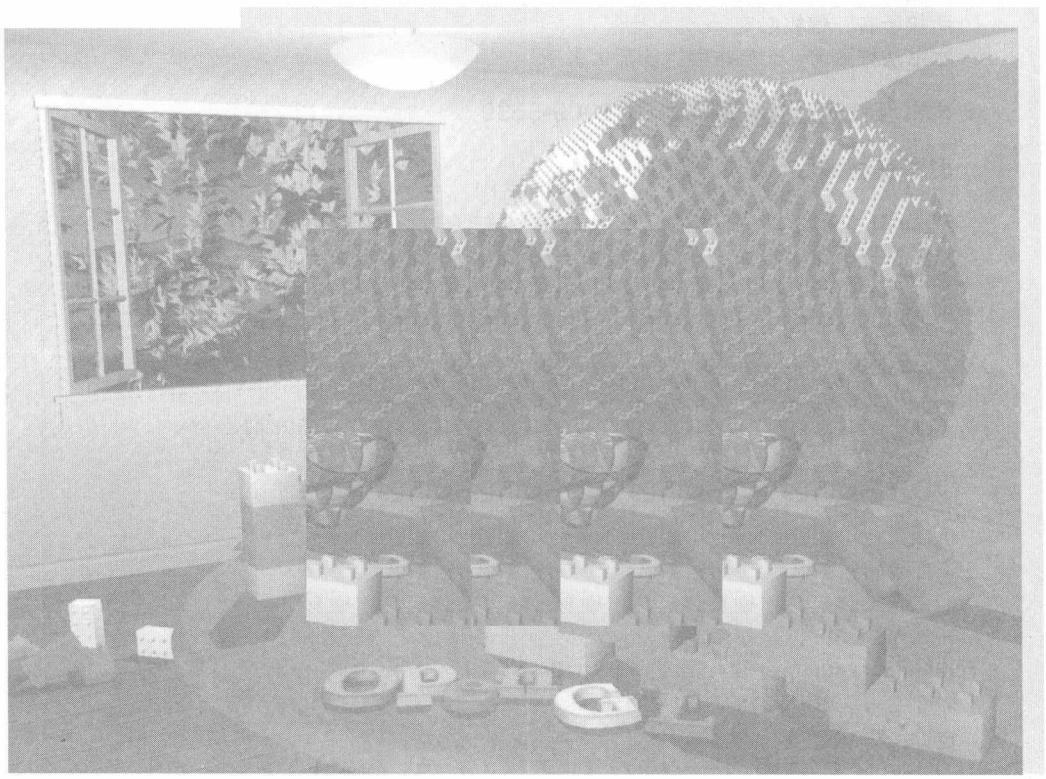
（原书第7版）

OpenGL编程指南

OpenGL Programming Guide *Seventh Edition*

（美）Dave Shreiner　The Khronos OpenGL ARB Working Group 著

李 军　徐 波　等译



机械工业出版社
China Machine Press

本书对OpenGL以及OpenGL实用函数库进行了全面而又权威的介绍，素有“OpenGL红宝书”之誉。本书的上一个版本覆盖了OpenGL 2.1版的所有内容。本版涵盖了OpenGL 3.0和3.1的最新特性。本书以清晰的语言描述了OpenGL的功能以及许多基本的计算机图形技巧，例如，创建和渲染3D模型、从不同的透视角度观察物体、使用着色、光照和纹理贴图使场景更加逼真等。另外，本书还深入探讨了许多高级技巧，包括纹理贴图、抗锯齿、雾和大气效果、NURBS、图像处理等。

本书内容详实，讲解生动，图文并茂，是OpenGL程序员的绝佳编程指南。

本书版权登记号：图字：01-2009-5639

图书在版编目（CIP）数据

OpenGL编程指南（原书第7版）/（美）施瑞奈尔（Shreiner, D.）等著；李军等译. —北京：机械工业出版社，2010.1

（开发人员专业技术丛书）

书名原文：OpenGL Programming Guide，Seventh Edition

ISBN 978-7-111-29450-4

Ⅰ.O… Ⅱ.① 施… ② 李… Ⅲ.图形软件，OpenGL Ⅳ.TP391.41

中国版本图书馆CIP数据核字（2010）第000465号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：陈佳媛

北京市荣盛彩色印刷有限公司印刷

2010年3月第1版第1次印刷

186mm×240mm · 34.5印张 （含1印张彩插）

标准书号：ISBN 978-7-111-29450-4

定价：89.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991；88361066

购书热线：（010）68326294；88379649；68995259

投稿热线：（010）88379604

读者信箱：hzjsj@hzbook.com

译 者 序

OpenGL是图形硬件的一种软件接口。从本质上说，它是一个3D图形和模型库，具有高度的可移植性，并且具有非常快的渲染速度。如今，OpenGL广泛应用于游戏、医学影像、地理信息、气象模拟等领域，是高性能图形和交互性场景处理的行业标准。

OpenGL的前身是SGI公司开发的IRIS GL图形函数库。SGI是一家久负盛名的公司，在计算机图形和动画领域处于业界领先地位。IRIS GL最初是一个2D图形函数库，后来逐渐演化为SGI的高端IRIS图形工作站所使用的3D编程API。后来，由于图形技术的发展，SGI对IRIS GL的移植性进行了改进和提高，使它逐步发展成如今的OpenGL。在此期间，OpenGL得到了各大厂商的支持，从而成为一种广泛流行的三维图形标准。

OpenGL并不是一种编程语言，而更像是一个C运行时函数库。它提供了一些预包装的功能，帮助开发人员编写功能强大的三维图形应用程序。OpenGL可以在多种操作系统平台上运行，例如各种版本的Windows、UNIX/Linux、Mac OS和OS/2等。

OpenGL是一个开放的标准，虽然它由SGI首创，但是它的标准并不控制在SGI的手中，而是由OpenGL体系结构审核委员会（ARB）掌管。ARB由SGC、DEC、IBM、Intel和Microsoft等著名公司于1992年创立，后来又陆续添加了nVidia、ATI等图形芯片领域的巨擘。ARB每隔4年举行一次会议，对OpenGL规范进行维护和改善，并出台计划对OpenGL标准进行升级，使OpenGL一直保持与时代同步。

2006年，SGI公司把OpenGL标准的控制从ARB移交给一个新的工作组——Khronos小组（www.khronos.org）。Khronos是一个由成员提供资金的行业协会，专注于开放媒体标准的创建和维护。目前，Khronos负责OpenGL的发展和升级。

《OpenGL编程指南》就是由Khronos小组编写的官方指南，是OpenGL领域的权威著作，有“OpenGL红宝书”之称，曾经帮助许多程序员走上了OpenGL专家之路。第7版在第6版的基础上又有所改进，介绍了OpenGL 3.0和OpenGL 3.1的新的和更新的内容。

本书历经多次版本升级，其中文版的翻译也是一项延续性的工作，凝结了许多人的辛勤工作。徐波等曾承担《OpenGL编程指南》第5版和第6版的主要翻译工作。李军在第6版的中文版的基础上，负责了第7版新增内容的翻译和更新工作。参与第7版翻译工作的还有刘金华、刘伟超、罗庚臣、刘二然、郑芳菲、庄逸川、王世高、郭莹、陈垚、邓勇、何进伟、贾晓斌、汪蔚和齐国涛。机械工业出版社华章分社的编辑为本书的出版付出了辛勤劳动，感谢他们！

译者

2009年10月

前　言

OpenGL（Graphics Library，GL图形库）图形系统是图形硬件的一个软件接口，它允许我们创建交互性的程序，产生移动三维物体的彩色图像。使用OpenGL，我们可以对计算机图形技术进行控制，产生逼真的图像或者虚构出现实世界没有的图像。本书解释了如何使用OpenGL图形系统进行编程，实现所需要的视觉效果。

本书内容

本书共分15章。前5章描述了一些基本信息，读者需要理解这些内容，才能在场景中绘制正确着色和光照的三维物体。

- 第1章对OpenGL可以实现的功能进行简要的介绍。该章还提供了一个简单的OpenGL程序，并介绍需要了解的一些基本编程细节，有助于学习后续章节的内容。
- 第2章解释如何创建一个物体的三维几何图形描述，并最终把它绘制到屏幕上。
- 第3章描述三维模型在绘制到二维屏幕之前如何进行变换。我们可以控制这些变换，显示模型的特定视图。
- 第4章描述如何指定颜色以及用于绘制物体的着色方法。
- 第5章解释如何控制围绕一个物体的光照条件，以及这个物体如何对光照作出反应（也就是说，它是如何反射或吸收光线的）。光照是一个重要的主题，因为物体在没有光照的情况下看上去往往没有立体感。

接下来的几章说明如何对三维场景进行优化以及如何添加一些高级特性。在没有精通OpenGL之前，读者可以选择不使用这些高级特性。有些特别高级的主题在出现时会有特殊的标记。

- 第6章描述创建逼真场景所需要的一些基本技巧：alpha混合（创建透明物体）、抗锯齿（消除锯齿状边缘）、大气效果（模拟雾和烟雾）以及多边形偏移（在着重显示填充多边形的边框时消除不良视觉效果）。
- 第7章讨论如何存储一系列的OpenGL命令，用于在以后执行。我们可以使用这个特性来提高OpenGL程序的性能。
- 第8章讨论如何操作表示位图或图像的二维数据。位图的一种常见用途就是描述字体中的字符。
- 第9章解释如何把称为纹理的一维、二维和三维图像映射到三维物体表面。纹理贴图可以实现许多非常精彩的效果。
- 第10章描述OpenGL实现中可能存在的所有缓冲区，并解释如何对它们进行控制。我们可以使用这些缓冲区实现诸如隐藏表面消除、模板、屏蔽、运动模糊和景深聚焦等效果。
- 第11章显示了如何使用GLU（OpenGL Utility Library，OpenGL工具函数库）中的分格化和二次方程函数。
- 第12章介绍生成曲线和表面的高级技巧。
- 第13章说明如何使用OpenGL的选择机制来选择屏幕上的一个物体。此外，该章还解释了反馈

机制，它允许我们收集OpenGL所产生的绘图信息，而不是在屏幕上绘制物体。

- 第14章描述如何用巧妙的或意想不到的方法来使用OpenGL，产生一些有趣的结果。这些技巧是通过对OpenGL及其技术前驱Silicon Graphics IRIS图形函数库的多年应用和实践总结出来的。
- 第15章讨论OpenGL 2.0所引入的变化，包括对OpenGL着色语言的介绍。OpenGL着色语言通常又称为GLSL，它允许对OpenGL的顶点和片断处理阶段进行控制。这个特性可以极大地提高图像的质量，充分体现OpenGL的计算威力。

另外，本书还包括几个非常实用的附录：

- 附录A讨论了用于处理窗口系统操作的函数库。GLUT（OpenGL实用工具库）具有可移植性，它可以使代码更短、更紧凑。
- 附录B列出了OpenGL所维护的状态变量，并描述了如何获取它们的值。
- 附录C介绍了隐藏在矩阵转换后面的一些数学知识。
- 附录D简单描述了窗口系统特定的函数库所提供的函数，它们进行了扩展，以支持OpenGL渲染。本附录讨论了X窗口系统、Apple的Mac OS和Microsoft Windows的窗口系统接口。
- 附录E对OpenGL所执行的操作提供了一个技术性的浏览，简要描述了当应用程序执行时这些操作的出现顺序。
- 附录F列出了一些基于OpenGL设计者思路的编程提示，这些可能对读者有用。
- 附录G描述了OpenGL实现在什么时候以及什么地方必须生成OpenGL规范所描述的精确像素。
- 附录H描述了如何计算不同类型的几何物体的法线向量。
- 附录I列出了OpenGL着色语言所提供的所有内置的变量和函数。
- 附录J介绍了各种浮点数、共享指数像素和纹理单元格式。
- 附录K介绍了存储单成分和双成分压缩纹理的纹理格式。
- 附录L介绍了GLSL 1.40的uniform变量缓存区的标准内存布局。

最后，本书还提供了一个术语表，对本书所使用的一些关键术语进行了定义。

第7版的新增内容

本书包含了OpenGL 3.0和OpenGL 3.1的新的和更新的内容。通过这些版本（这也是本书值得庆祝的18岁生日），OpenGL经历了与其之前的版本最显著的改变。3.0版添加了很多新的功能，并且添加了废弃模型，它建立了一种方法把陈旧的功能从库中删除。注意，只有新功能添加到了3.0版中，才会使其在源代码和二进制文件上都和之前的版本向后兼容。然而，很多功能标记为废弃的，表示可能在API未来的版本中删除。

本书介绍的和OpenGL 3.0相关的更新内容包括：

OpenGL中的新功能：

- OpenGL着色语言更新，创建了GLSL 1.30版。
- 条件渲染。
- 对映射缓冲区对象的内存的细粒度访问以用于更新和读取。
- 除了纹理图像格式（在OpenGL 2.1中加入），还有用于帧缓冲区的浮点数像素格式。
- 帧缓冲区和渲染缓冲区对象。
- 为小的动态范围数据采用紧凑的浮点表示，以减少内存存储占用。

- 改进了对复制数据时的多采样缓冲区交互的支持。
- 纹理图像和渲染缓冲区中的非规范化的整数值保留它们最初的表示，相对于OpenGL将这些值映射到范围[0,1]的常规操作。
- 支持一维纹理数组和二维纹理数组。
- 附加的包装像素格式支持访问新的渲染缓冲区。
- 针对多渲染目标，分开混合和写屏蔽控制。
- 纹理压缩格式。
- 纹理的单成分和双成分的内部格式。
- 转换反馈。
- 顶点数组对象。
- sRGB帧缓冲区格式。
- 废弃模式的深入讨论。
- 修复错误并更新标记名。

对于OpenGL 3.1：

- 标识出OpenGL 3.0中因废弃而要删除的功能。

新的功能：

- OpenGL着色语言更新，创建了GLSL 1.40版。
- 实例化渲染。
- 缓冲区之间高效的服务器端数据复制。
- 在单个调用作用渲染多个类似的图元，用一个特殊标记（由用户指定）来表示何时重新启动一个图元。
- 纹理缓冲区对象。
- 纹理矩形。
- uniform缓冲区对象。
- 带符号的规范化纹理单元格式。

阅读本书所需要的预备知识

本书假定读者知道如何使用C语言编写程序，并且了解一些数学背景知识（几何、三角、线性代数、积分和微分几何）。即使读者对计算机图形技术领域了解不多，仍然能够理解本书所讨论的绝大部分内容。当然，计算机图形学是一个巨大的主题，因此读者最好能够扩展自己在这个领域的知识。下面是我们推荐的两本参考书：

- 《Computer Graphics: Principles and Practice》：作者James D.Foley、Andries van Dam、Steven K.Feiner和John F.Hughes（Addison-Wesley，1990）。该书是计算机图形学领域的百科全书，它包括了丰富的信息。但是，读者在阅读这本书之前最好已经对计算机图形学有一定程度的了解。
- 《3D Computer Graphics》：作者Andrew S. Glassner（The Lyons Press，1994）。该书对计算机图形学进行了非技术性的、通俗易懂的介绍。这本书重点介绍计算机图形学可以实现的视觉效果，而不是实现这些效果所需要的技术。

另外，读者还可以访问OpenGL的官方网站。这个网站包含了各种类型的通用信息，包括软件、

示例程序、文档、FAQ、讨论版和新闻等。如果读者遇到任何OpenGL问题，这是首先应该想到的去处：http://www.opengl.org/。

另外，OpenGL官方网站记录了组成OpenGL 3.0版和OpenGL 3.1版所有过程的完整文档。这些文档代替了OpenGL体系结构审核委员会和Addison Wesley出版的《OpenGL Reference Manual》(OpenGL参考手册)。

OpenGL实际上是一种独立于硬件的程序接口规范。在一种特定类型的硬件上，所使用的是它的一种特定实现。本书解释了如何使用所有的OpenGL实现进行编程。但是，由于各种OpenGL实现可能存在微小的差别（例如，在渲染性能以及提供额外的、可选的特性方面），因此读者可能需要寻找与自己所使用的特定OpenGL实现相关的补充文档。另外，读者所使用的系统还可能提供了与OpenGL相关的实用函数库、工具箱、编程和调试支持工具、各种窗口部件、示例程序和演示程序等。

如何获取本书示例程序的源代码

本书包含了许多示例程序，以说明各种OpenGL编程技巧的具体用法。由于本书的读者背景各不相同，从新手到老手，既有计算机图形学的知识又有OpenGL的知识，本书中的示例通常都展示了一个特定渲染条件下的最简单的方法，并且使用OpenGL 3.0接口来说明。这么做的目的主要是为了让那些刚开始接触OpenGL的读者能够更直接和更容易地接受本书所介绍的内容。对于那些有着广泛经验、想要寻找使用最新的API功能实现的读者，我们首先感谢你耐心地阅读那些早已掌握的内容，并且建议访问我们的Web站点：http://www.opengl-redbook.com/。

在那里，你将会找到本书中所有示例的源代码，使用最新功能的实现，以及介绍从一个版本的OpenGL迁移到另一个版本所需的修改的额外讨论。

本书中包含的所有程序都使用了OpenGL Utility Toolkit (GLUT)，其最初的作者是Mark Kilgard。对于这个版本，我们使用了开发freeglut项目的人们所编写的GLUT接口的开源版本。他们扩展了Mark最初的工作（那些工作在Mark Kilgard的《OpenGL Programming for the X Window System》Addison-Wesley，1996，一书中有详细的介绍）。可以从如下地址找到他们的开源项目页面：http://freeglut.sourceforge.net/。

可以从这个站点获取它们的实现的代码和二进制文件。

本书1.6节和附录A给出了关于使用GLUT的更多信息。其他有助于更快地学习和使用OpenGL和GLUT编程的资源，可以在OpenGL Web的资源页面找到：http://www.opengl.org/resources/。

许多OpenGL实现还可能包含一些示例代码，作为系统的一部分。这些源代码可能是这种OpenGL实现应该使用的最佳代码，因为它可能针对当前的系统进行了优化。读者可以参阅与自己使用的系统相关的OpenGL文档，了解从哪里下载这些示例程序。

Nate Robin的OpenGL教程

Nate Robin编写了一套教学程序，用于演示OpenGL编程的基本概念。它允许用户修改函数的参数，以交互的方式观察它们的效果。这套教程所涵盖的主题包括变换、光照、雾和纹理。我们极力推荐使用这些教程。它们具有可移植性，并且使用了前面所提到的GLUT。要获取这些教程的源代码，可以访问下面这个网站：

http://www.xmission.com/~nate/tutors.html

勘误表

本书也会存在一些错误。此外，在本书出版过程中，OpenGL也在更新：随着规范和新规范的发布，错误也会更正并加以澄清。我们在Web站点http://www.opengl-redbook.com/维护了一个错误和更新的列表，那里还提供了工具来报告你可能会发现的任何新的错误。如果你发现一个错误，请接受我们的道歉，并且提前感谢你的报告，我们将尽快地更正。

约定字体

本书使用如下的约定字体：

- 粗体——表示命令和函数的名称和矩阵。
- 斜体——变量、参数、参数名、空间维度、矩阵成员和第一次出现的关键术语。
- 常规字体——每句类型和定义的常量。

代码示例以等宽字体显示，命令概览放在专门的框线中。

在命令概览部分，花括号表示可选的数据类型。在下面的例子中，glCommand具有4个可能的后缀：s、i、f和d，分别表示数据类型GLshort、GLint、GLfloat和GLdouble。在glCommand函数的原型中，TYPE表示这些后缀所提示的数据类型。

void **glCommand**{sifd}(TYPE x1, TYPE y1, TYPE x2, TYPE y2);

区分废弃的功能

兼容性扩展
glBegin
GL_POLYGON

正如前面所提到的，本书的这一版本主要针对OpenGL 3.0和OpenGL 3.1。OpenGL 3.0对于目前可用的任何版本完全向后兼容。然而，OpenGL 3.1采用了废弃模式，删除了很多与现代图形系统不太兼容的旧功能。尽管很多功能从“核心的”OpenGL中删除了，为了消除版本之间的过渡，OpenGL ARB发布了GL_ARB_compatibility扩展。如果你的实现支持这个扩展，它将能够使用所有删除的功能。为了便于识别那些从OpenGL 3.1中删除了的、仍然得到兼容扩展支持的功能，在本书中，介绍命令或函数的边框的旁边会给出一个信息表格，其中列出受到影响的函数或符号。

尽管有些功能从OpenGL中废弃并删除了，但一些这样的功能影响着库，例如OpenGL Utility Library，通常称之为GLU。这些在OpenGL 3.1的变化中受到影响的函数，也会在旁边的表格中列出。

目　录

⊖ 附录E～附录L请读者查阅华章网站（www.hzbook.com/）。

第 1 章　OpenGL简介

本章目标

- 大致了解OpenGL的功能。
- 了解不同程度的渲染复杂性。
- 理解OpenGL程序的基本结构。
- 了解OpenGL函数的语法。
- 了解OpenGL渲染管线的操作序列。
- 大致了解如何在OpenGL程序中实现动画。

本章对OpenGL进行了简单的介绍，主要包含下面几节：

- 什么是OpenGL：介绍OpenGL是什么，它能够做什么，不能够做什么，以及它的工作原理。
- 一段简单的OpenGL代码：展示一个小型的OpenGL程序，并对它进行了简单的讨论，并定义了一些基本的计算机图形术语。
- OpenGL函数的语法：解释OpenGL函数所使用的一些约定和记法。
- OpenGL是一个状态机：描述OpenGL状态变量的用法，并介绍一些查询、启用和禁用OpenGL状态的函数。
- OpenGL渲染管线：展示一个用于处理几何和图像数据的典型操作序列。
- 与OpenGL相关的函数库：介绍一些与实用OpenGL相关的函数，包括对GLUT（Graphics Library Utility Toolkit，一种可移植的工具库）的详细介绍。
- 动画：简单介绍如何创建能够在屏幕上移动的图片。

OpenGL及其废弃机制：介绍在OpenGL的最新版本中有哪些废弃修改，这些修改如何影响到应用程序，以及根据这些修改，OpenGL未来会如何发展。

1.1　什么是OpenGL

OpenGL是图形硬件的一种软件接口。这个接口包含的函数超过700个（纳入OpenGL 3.0的函数大约有670个，另外50个函数位于OpenGL工具库中），这些函数可以用于指定物体和操作，创建交互式的三维应用程序。

OpenGL的设计目标就是作为一种流线型的、独立于硬件的接口，在许多不同的硬件平台上实现。为了实现这个目标，OpenGL并未包含用于执行窗口任务或者获取用户输入之类的函数。反之，必须通过具体的窗口系统来控制OpenGL应用程序所使用的特定硬件。类似地，OpenGL并没有提供用于描述三维物体模型的高级函数。这类函数可能允许指定相对较为复杂的形状，例如汽车、身体的某个部位、飞机或分子等。在OpenGL中，程序员必须根据一些为数不多的基本几何图元（如点、直线和多边形）来创建所需要的模型。

当然，程序员可以在OpenGL的基础之上，创建提供这些特性的高级函数库。OpenGL工具库（GLU）提供了许多建模功能，例如二次曲面以及NURBS曲线和表面。GLU是所有OpenGL实现的一

个标准组成部分。

既然读者已经知道了OpenGL不能够做什么，现在我们来讨论它可以做什么。可以看一下本书所附的彩图，它们展示了OpenGL的典型用法。本书封面上的场景就是在一台计算机上使用OpenGL通过一系列复杂的方法渲染（即绘制）产生的。下面，我们简单地说明这些图像是如何产生的。

- 彩图1使用线框模型（wireframe model）显示整个场景。也就是说，场景中的所有物体都是用线型构成的。每条线对应于图元（一般为多边形）的一条边。例如，桌子的表面就是由许多三角形构成的，这些三角形就像切开的蛋糕一样排列在一起。

注意，如果物体是实心的而不是线框的，可能只能看到物体的一部分。例如，在这张彩图中，可以看到窗外小山坡的整个模型。但是，在正常情况下，这个模型的大部分会被房间的墙壁遮挡。图中的地球仪看上去几乎是实心的，因为它是由几百个彩色块组成的。读者可以看到所有彩色块的所有边的线框，甚至可以看到构成地球仪背面的那些彩色块。这个地球仪的构建方式提供了一种思路，就是通过组装底层的简单物体来创建复杂的物体。

- 彩图2显示了同一个线框场景的深度提示（depth-cued）版本。注意，距离眼睛较远的线型看上去更暗淡一些，显然这更接近于现实。它提供了一种叫做深度的视觉提示。OpenGL使用各种大气效果（合称为雾）来实现深度提示。
- 彩图3显示了这个线框场景的抗锯齿（antialiased）版本。抗锯齿是一种用于消除锯齿状边缘的技术。锯齿状边缘是在使用像素（pixel，它是图片元素的缩写，指一个矩形网格大小）近似地模拟平滑边缘时产生的。当直线接近水平或垂直时，这种锯齿现象通常最为明显。
- 彩图4显示了这个场景进行单调着色（flat-shading）后的不带光照的版本。现在，场景中的物体以实心的形式显示。它们在场景中看上去像是平面的，这是因为每个多边形只用一种颜色进行渲染。因此，它们之间的过渡显得不够平滑。此外，这个场景也没有应用任何光照效果。
- 彩图5显示了这个场景使用平滑着色（smooth-shading）并且带光照的版本。注意，当物体根据房间内的光源进行着色时，它们看上去更为逼真，而且更具立体效果，它们的表面就像被磨圆了一样。
- 彩图6在前一个版本场景的基础上添加了阴影（shadow）和纹理（texture）效果。阴影并不是OpenGL特性（并不存在“阴影函数”），但是可以使用第9章和第14章所描述的技巧自行创建这种效果。纹理贴图（texture mapping）允许把一幅二维图像应用到一个三维物体上。在这个场景中，桌面就是一个典型的纹理贴图例子。地板和桌面上的木纹都是使用纹理贴图的结果，墙面和桌上玩具的表面也是如此。
- 彩图7显示了这个场景中一个运动模糊（motion-blurred）的物体。图中的积木好像是在向前运动，它们的背后留下了一道模糊的运动轨迹。
- 彩图8从另一个角度显示了本书封面的那个场景。这张图说明了图像实际上只不过是三维物体模型的一张快照而已。
- 彩图9再次使用了雾，用它来模拟空气中的烟尘。彩图2已经显示了雾的效果，但与之相比，彩图9的雾效果更为明显。
- 彩图10显示了景深效果（depth-of-field effect），它模拟了照相机无法对场景中的所有物体进行聚焦的现象。当照相机聚焦于场景中一个特定的点时，远离这个点的物体看起来就会比较模糊。

看了这些彩图之后，读者应该对OpenGL图形系统的功能有了一个大致的了解。下面，我们简单地描述当OpenGL对场景中的图像进行渲染时所执行的主要图形操作。（请参见1.5节，了解和这些操

作顺序有关的详细信息。）

1）根据几何图元创建形状，从而建立物体的数学描述。（OpenGL把点、直线、多边形和位图作为基本的图元。）

2）在三维空间中排列物体，并选择观察复合场景的有利视角。

3）计算所有物体的颜色。颜色可以由应用程序明确指定，可以根据特定的光照条件确定，也可以通过把纹理贴到物体的表面而获得，或者是上述三种操作的混合产物。这些操作可能使用着色器来执行，这样可以显式地控制所有的颜色计算，或者可能使用OpenGL的预编程算法在其内部执行（我们常用术语固定功能的管线来表示后者）。

4）把物体的数学描述以及与物体相关的颜色信息转换为屏幕上的像素。这个过程叫做光栅化（rasterization）。

在这些阶段期间，OpenGL可能还会执行其他操作，例如消除被其他物体所遮挡的物体（或该物体的一部分）。此外，在场景被光栅化之后但在绘制到屏幕之前，仍然可以根据需要对像素数据执行一些操作。

在有些OpenGL实现（例如X窗口系统的OpenGL实现）中，OpenGL必须实现这样一个目标：显示程序员所创建的图形的计算机和运行图形程序的计算机可以不相同。由多台计算机通过一个数字网络彼此连接在一起所组成的网络计算机环境就属于这种情况。在这种情况下，运行图形程序并发出绘图命令的计算机称为客户机，接收这些命令并执行绘图任务的计算机称为服务器。客户机发送给服务器的命令的传输格式（称为协议）总是相同的，因此OpenGL程序可以通过网络运行，即使客户机和服务器并不是同种类型的计算机。如果OpenGL程序并不是通过网络运行的，那就只涉及一台计算机，它既是客户机也是服务器。

1.2 一段简单的OpenGL代码

由于OpenGL图形系统的功能非常强大，因此OpenGL程序可能相当复杂。但是，许多实用的OpenGL程序的基本结构可能非常简单，它的任务就是初始化一些状态（这些状态用于控制OpenGL的渲染方式），并指定需要进行渲染的物体。

在察看具体的OpenGL代码之前，首先介绍几个术语。渲染（rendering）是计算机根据模型创建图像的过程，我们已经在前面看到过这个术语。模型（model）是根据几何图元创建的，也称为物体（object）。几何图元包括点、直线和多边形等，它们是通过顶点（vertex）指定的。

最终完成了渲染的图像是由在屏幕上绘制的像素组成的。像素是显示硬件可以在屏幕上显示的最小可视元素。在内存中，和像素有关的信息（例如像素的颜色）组织成位平面（bitplane）的形式。

位平面是一块内存区域，保存了屏幕上每个像素的1个位的信息。例如，它指定了一个特定像素的颜色中红色成分的强度。位平面又可以组织成帧缓冲区（framebuffer）的形式，后者保存了图形硬件为了控制屏幕上所有像素的颜色和强度所需要的全部信息。

现在我们观察一个简单的OpenGL程序。示例程序1-1在黑色背景中渲染了一个白色的矩形，如图1-1所示。

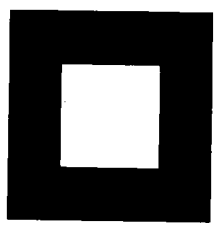


图1-1 黑色背景中的白色矩形