HZ BOOKS
PEARSON
Addison Wesley
华章程序员书库

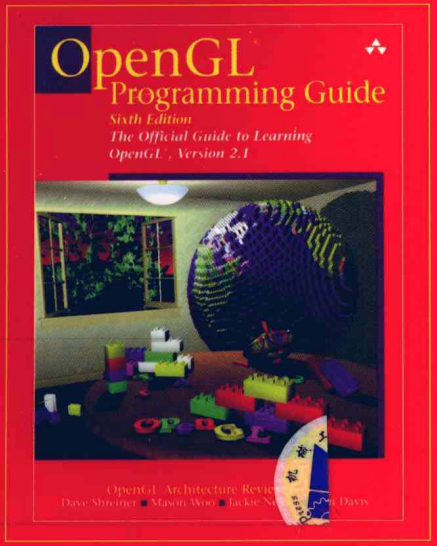
OpenGL 编程指南

（原书第6版）

OpenGL® Programming Guide

The Official Guide to Learning OpenGL, Version 2.1

Sixth Edition



OpenGL Architecture Review Board,
Dave Shreiner, Mason Woo, Jackie Neider, Tom Davis 著
徐波 译

- 详实权威的OpenGL红宝书
- 学习OpenGL的官方指南

机械工业出版社
China Machine Press

OpenGL 编程指南

（原书第6版）

OpenGL® Programming Guide

The Official Guide to Learning OpenGL, Version 2.1

OpenGL Architecture Review Board,
Dave Shreiner, Mason Woo, Jackie Neider, Tom Davis 著

徐波 译

机械工业出版社
China Machine Press

本书以清晰的语言、丰富的实例以及翔实的内容描述了OpenGL的功能以及许多计算机图形技巧。全书共15章，包括OpenGL简介、状态管理和绘制几何物体、视图、颜色、光照、混合、抗锯齿、雾、多边形偏移、显示列表、绘制像素、位图、字体、图像、纹理贴图、帧缓冲区、分格化和二次方程表面、求值器和NURBS、选择和反馈、OpenGL高级技巧以及OpenGL着色语言等内容。本书对OpenGL以及OpenGL实用函数库提供了全面而又权威的介绍，素有“OpenGL红宝书”之美誉。第6版在第5版的基础上进行了全面修订，涵盖了OpenGL2.1版本的最新特性。

本书版权登记号：图字：01-2008-1746

图书在版编目（CIP）数据

OpenGL编程指南（原书第6版）/（美）施润尼（Shreiner, D.）等著；徐波等译. —北京：机械工业出版社，2008.7

（华章程序员书库）

书名原文：OpenGL Programming Guide: The Official Guide to Learning OpenGL,Version 2.1, Sixth Edition

ISBN 978-7-111-24201-7

Ⅰ.O…　Ⅱ.①施…　②徐…　Ⅲ.图形软件，OpenGL－程序设计指南　Ⅳ.TP391.41-62

中国版本图书馆CIP数据核字（2008）第073930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：李南丰

北京瑞德印刷有限公司印刷·新华书店北京发行所发行

2008年8月第1版第1次印刷

186mm×240mm　·　33印张（含彩页1个印张）

标准书号：ISBN 978-7-111-24201-7

定价：75.00元

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广泛流行于游戏、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工业标准。

OpenGL的前身是SGI公司所开发的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的地位。IRIS GL最初是个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，从而使它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，而是更像一个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是个开放的标准，虽然它由SGI首创，但它的标准并不是控制在SGI的手中，而是由OpenGL体系结构审核委员会（ARB）所掌管。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年开一次会，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一直保持与时代的同步。

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小组所编写的官方指南，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6版在第5版的基础上又有改所进。

作为本书的主要翻译者，我在《OpenGL编程指南》第3版问世时（2001年）开始涉猎OpenGL，并对它产生了浓厚的兴趣。虽然由于开发工作的繁忙，在OpenGL上面无法投入太多的精力，但它一直是我感兴趣的一个领域。2006年，我承担了本书第5版的翻译任务，对OpenGL有了更加细致深入的研究。在翻译第6版时，除了OpenGL新特性之外，还对原来的翻译内容进行了仔细的修改和完善，并纠正了第5版所存在的一些小问题。

本书的翻译是个艰苦的过程，其中凝结了许多人的辛勤工作。除了我之外，黄静、李福军、陈永军、毛玉山、朱健儿、韩建正、张汉宏也参加了本书的翻译工作。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！

徐 波

2008年4月

前　言

OpenGL（GL代表图形库，Graphics Library）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所没有的图像。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够在场景中绘制正确着色和光照的三维物体。

- 第1章“OpenGL简介”：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，并解释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“状态管理和绘制几何物体”：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“视图”：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“颜色”：描述如何指定颜色以及用于绘制物体的着色方法。
- 第5章“光照”：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。在还没有精通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“混合、抗锯齿、雾和多边形偏移”：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“显示列表”：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“绘制像素、位图、字体和图像”：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“纹理贴图”：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“帧缓冲区”：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“分格化和二次方程表面”：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“求值器和NURBS”：介绍生成曲线和表面的高级技巧。
- 第13章“选择和反馈”：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“OpenGL高级技巧”：描述如何用一些巧妙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而总结出来的。
- 第15章“OpenGL着色语言”：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“操作顺序”：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B“状态变量”：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“OpenGL和窗口系统”：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。本章讨论了X窗口系统、Apple Macintosh、IBM OS/2和Microsoft Windows的窗口系统接口。
- 附录D“GLUT（OpenGL实用工具库）基础知识”：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E“计算法线向量”：描述了如何计算不同类型的几何物体的法线向量。
- 附录F“齐次坐标和变换矩阵”：介绍了隐藏在矩阵转换后面的一些数学知识。
- 附录G“编程提示”：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H“OpenGL的不变性规则”：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I“OpenGL着色语言内置的变量和函数”：列出了OpenGL着色语言所提供的所有内置的变量和函数。

最后，本书还提供了一个词汇表，对本书所使用的一些关键术语进行了定义。

第6版的新增内容

本书涵盖OpenGL 2.1版本，增加了一些新内容，并对一些原有的内容进行了更新。

- 增加了下面这些核心功能：
- 在缓冲区对象中存储像素矩形。

- 支持sRGB格式化纹理。sRGB颜色空间大致对应于gamma校正的RGB空间（使用2.2的gamma值）。
- 在OpenGL着色语言的着色器程序中指定非正方形矩阵为uniform变量。
- 对OpenGL着色语言进行了扩展讨论，包括新增的对OpenGL 2.1版本的功能支持。
- 修正了一些bug，并做了其他一些澄清。

阅读本书所需要的预备知识

本书假定读者知道如何用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多数内容。当然，计算机图形学是个巨大的主题，因此读者最好能够扩展自己在这个领域的知识。下面是我们所推荐的两本参考书：

- 《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对计算机图形学已经有了一定程度的了解。
- 《3D Computer Graphics》：作者Andrew S.Glassner（The Lyons Press，1994）：该书对计算机图形学进行了一些非技术性的、通俗易懂的介绍。这本书的重点是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 2.1版的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所出版的《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，说明了各种OpenGL编程技巧的具体用法。这些程序使用了Mark Kilgard所编写的OpenGL实用工具库（GLUT）。读者可以在Mark Kilgard所著的

该书已由机械工业出版社引进出版，书号为7-111-13026-X，中译名为《计算机图形学原理及实践》。——编辑注

《OpenGL Programming for the X Windows System》（Addison Wesley，1996）中了解GLUT的详细信息。本书第1章中的“OpenGL相关函数库”以及附录D提供了使用GLUT的更多信息。如果读者可以访问Internet，可以通过匿名FTP（文件传输协议）免费获取本书的示例程序以及GLUT的源代码。

为了获取本书示例程序的源代码，可以访问：

http://www.opengl-redbook.com/code/

为了获取Mark Kilgard的GLUT的源代码（用于Microsoft Windows或X窗口系统），可以访问下面这个网站，了解当前可用的GLUT版本，并下载它的源代码：

http://www.opengl.org/resources/libraries/glut/

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所使用的系统相关的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

本书无疑也会存在一些错误。下面这个网站维护了本书的错误列表：

http://www.opengl-redbook.com/errata/

如果读者发现了书中的任何bug，可以在这个网站中报告它们。

格式约定

本书的正文使用宋体（中文）和Times New Rome（英文），字号为五号。本书的源代码使用Courier New字体。函数的说明具有灰色背景。

在函数说明中，函数的名称和矩阵用粗体表示，参数的名称用斜体表示。花括号用于标识可选数据类型选项。在下面这个例子中，**glCommand**具有4个可能的后缀：s、i、f和d，它们分别表示数据类型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*);

目 录

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，其内容主要分布于下面几节中。

- “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，还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”：介绍了一些与实用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。

1.1 什么是OpenGL

OpenGL是图形硬件的一种软件接口。这个接口所包含的函数超过700个（其中核心OpenGL大约包括650个函数，另外50个左右的函数位于OpenGL工具库），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，可以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，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，现在让我们来讨论它可以做什么。读者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，这个模型的大部分将被房间的墙壁遮挡。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，就是通过组装低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显得更暗淡一些，显然这更接近于现实。它提供了一种称为深度的视觉提示。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所产生的。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，它们看上去显得更为逼真，而且更具立体效果，它们的表面就像被磨圆了一样。
- 彩图6在前一个版本的场景的基础之上添加了阴影（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之上。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显得比较模糊。

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。（请参见稍后的1.5节“OpenGL渲染管线”，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。颜色可以是由应用程序明确指定的，也可以是根据特定的光照条件所确定的，或者是通过把纹理贴到物体的表面而获得的，或者是上述这三种操作的混合产物。

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。这个过程称为*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必须要实现这样一个目标：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并不相同。由许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。

在观察具体的OpenGL代码之前，我们首先介绍几个术语。*渲染*（rendering）就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，也可称为*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。

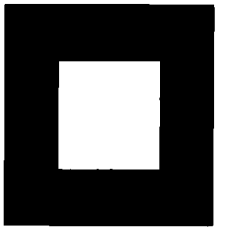


图1-1 黑色背景中的白色矩形

最终完成了渲染的图像是由在屏幕上所绘制的像素组成的。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）被组织成*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。

现在让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。

示例程序1-1 一段OpenGL代码

```
#include <whateverYouNeed.h>

main() {
   InitializeAWindowPlease();

   glClearColor(0.0, 0.0, 0.0, 0.0);
```

```
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
    glBegin(GL_POLYGON);
       glVertex3f(0.25, 0.25, 0.0);
       glVertex3f(0.75, 0.25, 0.0);
       glVertex3f(0.75, 0.75, 0.0);
       glVertex3f(0.25, 0.75, 0.0);
    glEnd();
    glFlush();
    UpdateTheWindowAndCheckForEvents();
}
```

main()函数的第一行代码在屏幕上初始化一个窗口：**InitializeAWindowPlease()**函数是个占位符，是个特定于窗口系统的函数，它通常并不是OpenGL函数。接下来的两个函数都是OpenGL函数，它们把窗口颜色清除为黑色：**glClearColor()**函数确定了窗口将被清除成什么颜色，而**glClear()**函数实际完成清除窗口的任务。在设置了清除颜色之后，以后每次调用**glClear()**时，窗口就会被清除为这种颜色。当然，可以再次调用**glClearColor()**函数，更改当前的清除颜色。类似，**glColor3f()**函数确定了绘制物体时所使用的颜色（在此例中为白色）。在此之后，所有被绘制的物体都将使用这种颜色，除非再次调用这个函数更改绘图颜色。

这个程序所使用的下一个OpenGL函数是**glOrtho()**，它指定了OpenGL在绘制最终图像时所使用的*坐标系统*（cooridinate system），决定了图像将如何映射到屏幕上。接下来的几个函数位于一对**glBegin()**和**glEnd()**调用之间，它们定义了将要被绘制的物体，在本例中为一个具有4个顶点的多边形。这个多边形的“角”是由**glVertex3f()**函数定义的。该函数所使用的参数表示（x, y, z）坐标。根据这些值，可以猜出这个多边形是个位于$z = 0$平面的矩形。

最后，**glFlush()**函数保证了绘图命令将被实际执行，而不是存储在*缓冲区*中等待其他的OpenGL命令。**UpdateTheWindowAndCheckForEvents()**也是个占位符函数，它管理窗口的内容，并开始进行事件处理。

实际上，这段OpenGL代码的结构并不是很好。读者可能会问：“如果我试图移动或改变窗口的大小，它会变成怎么样子？”或者“每次当我绘制这个矩形时，是不是需要重置坐标系统？”在本章的后面，我们将用实际执行任务的代码来替换**InitializeAWindowPlease()**和**UpdateTheWindowAndCheckForEvents()**占位符函数。但是，这就要求我们对这段代码的结构进行修改，使它更有效率。

1.3 OpenGL函数的语法

在前一节的那个简单程序中，读者可能已经发现OpenGL使用了前缀“**gl**”，并把组成函数的每个单词的首字母用大写形式表示（例如，**glClearColor()**）。类似，OpenGL还定义了一些以前缀GL_开头的常量，所有的单词都使用大写形式，并以下划线分隔（例如GL_COLOR_BUFFER_BIT）。

除此之外，读者可能还注意到有些OpenGL函数中有些似乎不相关的字母（例如**glColor3f()**和**glVertex3f()**中的“**3f**”）。确实，**glColor3f()**函数名中的“**Color**”部分就足以定义这个用于设置当前绘图颜色的函数。但是，OpenGL定义了这个函数的多个不同版本，以便使用不同类型的参数。具体地说，这个后缀中的“**3**”表示这个函数接受3个参数。**Color**函数还存在接受4个参数的版本。这个后缀中的“**f**”表示这些参数都是浮点数。OpenGL之所以为同一个函数定义了不同参数类型的版本，是为了允许用户根据他们自己的数据格式向OpenGL传递参数。

有些OpenGL函数可以在它们的参数中接受多达8种的不同数据类型。表1-1列出了一些后缀字母，