国外经典教材·计算机科学与技术

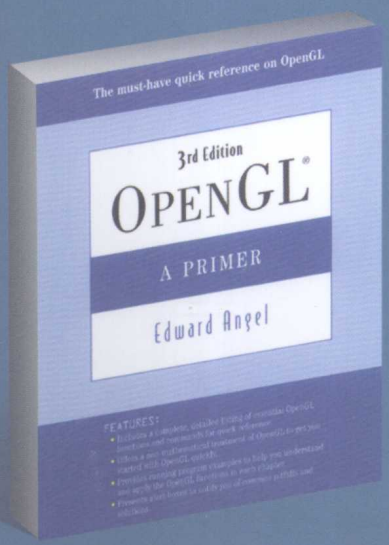
PEARSON Addison Wesley

OpenGL编程基础

（第3版）

OpenGL: A Primer

3rd Edition



- 完整、详细介绍基本的OpenGL函数和命令，便于读者快速查找
- 从非数学角度介绍如何进行OpenGL程序设计，有利于读者迅速入门
- 各章提供可运行的范例，有助于读者理解和应用OpenGL函数
- 提供“提示框”，提醒读者注意常见的陷阱和解决方案

(美) Edward Angel 著

段 菲 译

PEARSON Education

清华大学出版社

国外经典教材·计算机科学与技术

OpenGL 编程基础

(第 3 版)

(美) Edward Angel 著

段 菲 译

清华大学出版社

北 京

内 容 简 介

本书简明扼要地介绍了基本的 OpenGL 命令。它既可当作计算机图形学教材的配套教参，供计算机专业学生使用，也可以单独作为 OpenGL 程序设计指南，供有一定计算机图形背景的程序员参考。

图书在版编目(CIP)数据
OpenGL 编程基础(第 3 版)/(美)安吉尔(Angle E.)著；段菲译.—北京：清华大学出版社，2008.3
(国外经典教材・计算机科学与技术)
书名原文：OpenGL: A Primer, 3rd Edition
ISBN 978-7-302-17102-7

Ⅰ. O… Ⅱ. ①安…②段… Ⅲ. 图形软件－OpenGL－程序设计－教材 Ⅳ. TP391. 41

中国版本图书馆 CIP 数据核字(2008)第 022668 号

责任编辑：文开琪
责任校对：李凤茹
责任印制：王秀菊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
c－service@tup.tsinghua.edu.cn
社 总 机：010-62770175　　**邮购热线**：010-62786544
投稿咨询：010-62772015　　**客户服务**：010-62776969
印 刷 者：北京密云胶印厂
装 订 者：三河市李旗庄少明装订厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：17.25　**字　数**：335 千字
版　　次：2008 年 3 月第 1 版　　**印　次**：2008 年 3 月第1 次印刷
印　　数：1～4500
定　　价：35.00 元

出 版 说 明

近年来，我国的高等教育特别是计算机学科教育，进行了一系列大的调整和改革，急需一批门类齐全、具有国际先进水平的计算机经典教材，以适应当前我国计算机科学的教学需要。通过使用国外先进的经典教材，可以了解并吸收国际先进的教学思想和教学方法，使我国的计算机科学教育能够跟上国际计算机教育发展的步伐，从而培育出更多具有国际水准的计算机专业人才，增强我国计算机产业的核心竞争力。为此，我们从国外知名的出版集团 Pearson 引进这套“国外经典教材·计算机科学与技术”教材。

作为全球最大的图书出版机构，Pearson 在高等教育领域有着不凡的表现，其下属的 Prentice Hall 和 Addison Wesley 出版社是全球计算机高等教育的龙头出版机构。清华大学出版社与 Pearson 出版集团长期保持着紧密友好的合作关系，这次引进的“国外经典教材·计算机科学与技术”教材大部分出自 Prentice Hall 和 Addison Wesley 两家出版社。为了组织该套教材的出版，我们在国内聘请了一批知名的专家和教授，成立了一个专门的教材编审委员会。

教材编审委员会的运作从教材的选题阶段即开始启动，各位委员根据国内外高等院校计算机科学及相关专业的现有课程体系，并结合各个专业的培养方向，从 Pearson 出版的计算机系列教材中精心挑选针对性强的题材，以保证该套教材的优秀性和领先性，避免出现“低质重复引进”或“高质消化不良”的现象。

为了保证出版质量，我们为该套教材配备了一批经验丰富的编辑、排版、校对人员，制定了更加严格的出版流程。本套教材的译者，全部来自于对应专业的高校教师或拥有相关经验的 IT 专家。每本教材的责编在翻译伊始，就定期不间断地与该书的译者进行交流与反馈。为了尽可能地保留与发扬教材原著的精华，在经过翻译、排版和传统的三审三校之后，我们还请编审委员或相关的专家教授对文稿进行审读，以最大程度地弥补和修正在前面一系列加工过程中对教材造成的误差和瑕疵。

由于时间紧迫和受全体制作人员自身能力所限，该套教材在出版过程中很可能还存在一些遗憾，欢迎广大师生来电来信批评指正。同时，也欢迎读者朋友积极向我们推荐各类优秀的国外计算机教材，共同为我国高等院校计算机教育事业贡献力量。

清华大学出版社

国外经典教材·计算机科学与技术

编 审 委 员 会

译 者 序

我们常常会惊叹于中震撼人心的电影场景和无比逼真的游戏画面，在享受这些前所未有的视觉冲击的同时，难免会让人联想到到底是什么力量在幕后提供支撑。毋庸置疑，这一切都应归功于计算机三维图形学领域所取得的最新成果。

目前，以科学计算可视化、计算机动画和虚拟现实技术为主题的三维图形技术已经在军事、航空、航天、医学、地质勘探、文化娱乐以及艺术造型等领域得到了广泛的应用。每种快速发展的技术一定拥有强大工具的支持，正所谓“工欲善其事，必先利其器”。计算机图形学当然也不例外。OpenGL 是一个优秀的三维图形硬件的软件接口，同时也是一个跨平台的、开放性的三维图形和模型库。

自 1992 年 OpenGL 正式成为适于各种计算机环境下的三维应用程序接口以来，它便得到计算机行业巨头的极大关注和支持，先后涌现了大批支持 OpenGL 的图形卡，为计算机硬件的迅速更新起到了推波助澜的作用。它的出现，极大地增强了图形工作者的生产力，使他们从以往充满艰辛的工作中解脱出来，站在更高的抽象层次来思考和解决问题。计算机图形学的发展推动了 OpenGL 的持续发展和不断更新，可以说 OpenGL API 的每个新版本都体现了当时图形学发展的最高水平。目前，OpenGL 在专业绘图和科研领域仍然是占据统治地位的 API，在游戏开发领域也具有极其重要的位置。

本书的作者 Edward Angel 是一位有着多年丰富图形学教学经验的教授，除了在大学授课之外，他还曾经给来自世界各地的数以千计的科学家、工程师和程序员讲授计算机图形学方面的课程，并多次在计算机图形学领域的最高级别会议 SIGGRAPH 上讲授 OpenGL。正是这种授课对象的极端复杂性促使作者对传统的授课理念进行深入的反思和创新，因而大胆地在课堂中采用自顶向下、面向编程的方法来讲授计算机图形学。这种方式使得该门课程一开始便充满了趣味性，充分调动了学生的学习兴趣，使他们一开始就可以动手编写出具有一定难度的三维程序。如此一来，很容易使他们产生成就感并激发进一步深入学习的热情。事实证明，这种方式收到了非常良好的效果。令我们感到高兴的是，作者把他的这种教学理念也完全体现在本书的内容编排之中。相信读者朋友在阅读本书时也会从这种理念中获得愉悦的学习体验。

阅读本书，并不需要太多的先修知识，您只需具有一定的图形学基础，并且熟悉 C 语言的基本语法以及具备基本的线性代数基础(如向量、矩阵的基本运算)即可。本书的突出特点是实践性较强、自成体系，是迅速了解 OpenGL 编程的捷径之一。但须知，只有在实践中不断总结、不断加强巩固和加深对相关的理论的理解，才能在图形学之路上渐行渐远，

修炼成真正的高手，从而真正做到以不变应万变。

在译者翻译本书的过程中，OpenGL 正在酝酿着新一轮的变革，OpenGL 3.0 标准即将出台，新版本按照面向对象的思想将 API 完全重新组织。这无疑是一件好事，但其间必然会经历一番“阵痛”，因此正在学习“旧版本”OpenGL 的您大可不必因此而惊慌失措，因为以往的 API 虽然是面向过程的，但基本思想不会变，以往的知识积累仍然大有用武之地。

作为译者，本人在翻译过程中力求忠实于原作。书中所涉及的所有专业术语，都查阅了大量国内外相关文献，并请教了许多专家学者，力求找到最准确、最完美的译法，并在可能引起歧义或冲突的地方做了适当调整。同时，在专业术语第一次出现或在容易混淆的环境中出现时，都注上了该术语的英文原文，以便读者进行对比理解。尽管如此，由于计算机图形学发展迅速，每年都有大量新名词涌现，许多都尚未规范化，如果读者朋友对哪些术语的译法有不同的见解，非常乐意与我一同探讨。

特别感谢清华大学出版社第三事业部。在与他们合作的过程中，始终非常愉快。特别要感谢文开琪编辑，是她给了我许多非常有价值的建议，让我受益颇多。本书翻译工作的顺利完成，离不开她的帮助和指点。

翻译过程中，好友陈正华、段恺、郭山、李登、李硕、廖超、刘永、王汉奎、严严、朱云峰、李德武、姜媛媛、李黎明、夏琳等均参与其中，为我提供相当多的帮助和建议。限于译者水平，书中错误和不妥在所难免，恳请读者朋友们批评指正，欢迎发送邮件到duanfeis@gmail.com。

关于译著，本书特别在书后保留原书主题索引和函数索引，并在译著的相应地方用粗体方括号标注其原书页码，以期进一步帮助读者查阅和掌握相关主题。

最后祝愿大家在精彩无限的图形编程之旅中，一帆风顺，日益精进！

段　菲

2008 年 3 月于北京清华园

献给 Rose Mary

前　言

在2000年当我编写本书的第1版时，我希望它既可作为我的图形学教科书 *Interactive Computer Graphics: A Top-Down Approach Using OpenGL*(第4版，Addison-Wesley于2005年出版)的配套教材，也希望它能够自成体系，为已经掌握了一些计算机图形学知识，而且希望了解OpenGL的开发人员提供一本OpenGL入门指南。对于第二种读者群，我并不是想要取代OpenGL的标准参考书："红宝书"(*OpenGL Programming Guide*, 第5版，Addison-Wesley出版)和"蓝宝书"(*OpenGL 1.4 Reference Manual*, 第4版，Addison-Wesley出版)。相反，我希望奉献一本能够使程序员无需参考这些价值不菲的参考书便能快速入门的书。

我认为本书的第1版达到了我所期望的目标。很多情况下，我惊奇地发现，第二类读者群(至少以我所接收到的反馈来看)的数量超过了我的预期。我还惊奇地发现，本书已经作为一本独立的教科书为各种层次的图形学课所采用。

当我开始考虑编写本书的第2版时，我不得不做出一些艰难的决定，因为我必须对一本我认为已经非常成功的书籍进行修订。我希望这本书仍然保持简捷的风格，以将价格维持在一个较低的层次上，但我也希望对一些用户的反馈做出反应，同时希望体现图形学和OpenGL API的最新进展。为了实现这些看上去自相矛盾的愿望，本书的第2版与第1版相比，篇幅只增加了20%。在第2版中，我添加了一些新例子。另外我们专门增加了一章用于介绍可编程图形流水线，该工具代表了计算机图形学的一个主要进展。

可编程流水线的进一步发展促成了OpenGL 2.0，该版本中包含OpenGL着色语言(OpenGL Shading Language, GLSL)，它是促使我编写本书第3版的主要动机。我所面临的挑战是在短短一章中通过一些简单而生动的示例介绍GLSL的一些核心内容。

本书遵循了自顶向下的教学理念，这一理念是我在 *Interactive Computer Graphics* 一书中引入的。这种方法所基于的思想是如果学生能够尽快地开始编写一些有意义的应用程序，则他们在学习现代计算机图形学时就能达到最好的效果。OpenGL API非常有助于这一方法的实现。全世界的许多大中专院校都采用这种方法，充分表明这种教学理念的成功。

使用过 *Interactive Computer Graphics* 一书的学生都会有这样的认识：虽然这本书大量使用了OpenGL，但它绝非一本OpenGL编程指南，更不是一本用户手册。所以，在这本教材中，关于OpenGL的信息实际上是不完整的。并非所有的OpenGL函数都被涵盖，书中也没有函数及其参数的详细清单。对学生来说，前面的形式问题不大，但后面的形式确实会带来一些不便。于是，本书的诞生就填补了这个缺憾，这样学生们就不必去购买非常有参考价值但却价格不菲的"红宝书"(*OpenGL Programming Guide*)和"蓝宝书"(*OpenGL 1.4 Reference Manual*)。

大学里选修图形学课程的学生只是图形学社区中规模很小的一个群体，如果放在整个编程社区，则比例更小。许多人至少都对计算机图形学产生过一时的兴趣，而且也编写过

图形程序。对他们来说，使用 OpenGL 是涉足图形学领域的一种非常诱人的方式。本书编写的第二个动机是为读者学习 OpenGL 提供一条捷径。在我看来，即使对那些最终将在 Windows 平台上使用 DirectX 进行开发的人，从研究和使用 OpenGL 开始，并借助 OpenGL 来掌握一些图形学概念，这种学习方式更容易，效果也更好。对于那些像我们一样长期从事科学应用，而且需要在跨平台环境中工作的读者，OpenGL 是一个绝佳的选择。

本书自始至终几乎都没有介绍我那本图形学教材中所用到的数学原理。所以在很多章节中，例如第 9 章，我们更关注编程实现的细节，而没有做任何的数学推导。第 5 章介绍了如何使用旋转、平移及比例变换，但是并未对其中涉及的矩阵做任何推导。本书的章节编排顺序大致与我那本图形学教材一致，但同时也遵循了 OpenGL 的循序渐进的学习规律。我们首先从研究二维问题入手，这是第 2 章的内容，紧接着在第 3 章中我们转而研究程序的交互性，在第 4 章和第 5 章中，我们对三维程序进行了研究。第 6 章则介绍光照和材质。第 7 章和第 8 章介绍如何使用 OpenGL 来显示离散实体，我们首先讨论了像素和位图，接着又对纹理映射进行了详细的讨论。第 9 章介绍曲线和曲面。第 10 章介绍 OpenGL 着色语言(GLSL)，这种语言提供了一种 C 风格的编程方式，借助它我们可在应用程序中对可编程显卡进行编程。第 11 章涉及 OpenGL 的一些高级特性。

本书中既有完整的程序代码，也有代码片段。您在阅读本书时将发现，一旦您编写了少量 OpenGL 应用程序之后，许多代码都会在后续的程序中重复出现。所以，在介绍完前面几个例子后，我们将大部分重复代码都省略。读者可从我的个人主页 *www.cs.unm.edu/~angel* 下载所有示例的完整代码。

在本书的编写过程中，得到了很多人的大力支持。我要感谢现在或曾经在 Silicon Graphics 公司工作的许多人在学习 OpenGL 中给予我的热忱帮助。特别要感谢 Mark Kilgard 和 Mason Woo，当我在新墨西哥大学所开的图形学课程中最初采用 OpenGL 时，他们给予了我许多帮助。Mason、Kathleen Danielson、Dave Shreiner 以及 Vicki Shreiner 曾邀请我与他们在过去 9 年中，在 SIGGRAPH 一同讲授 OpenGL 的入门课程，正是通过这种形式的讲授迫使我对 OpenGL API 进行更深入的学习，并帮助我获得了如何教授 OpenGL 的一些心得。Mark、Nate Robins 以及 Brian Paul，他们通过构建 GLUT 库、Mesa 实现以及许多 OpenGL 示例程序，对我们这些使用和讲授 OpenGL 的人提供了非常大的帮助。本书第 1 版从编写到出版只经历了很短的时间，目的是满足图形学社区的一些需求。我非常感激那些对本书提出意见和印刷错误的读者。在编写本书第 2 版时，Mark Kilgard(NVIDIA)和 Dave Shreiner(SGI)对我的手稿进行了细致的审阅。他们给予了我太多的恩惠，因为他们对 OpenGL 开展的研究工作不知给予了我多少启示。现在我仍在非常快乐地享受着他们的恩惠。在我学习和研究 GLSL 时，Takeshi Hakamata 也给予了我莫大的帮助，我都欠他们一份人情。

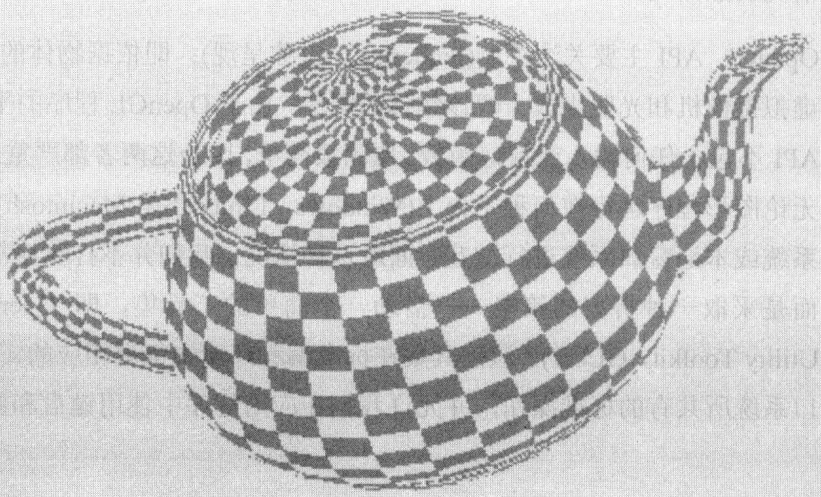
我陆续在 Addison-Wesley 出版公司出版了 7 本著作，我时常惊叹于该公司工作人员的能力和专业水准。来自 Addison-Wesley 的我的编辑 Matt Goldstein、Maite Suarez-Rivas 以及 Peter Gordon，他们都是我的好友，我们曾经在一起共进午餐。我的妻子和好搭档 Rose Mary Molnar，与我一同经历了另外一本书的出版周期，她理应获得更多的嘉奖。

目　录

第1章

绪　论

OpenGL 是一套应用程序编程接口(Application Programming Interface，API)。借助该 API，程序员可编写出对图形硬件具有访问能力的程序。对程序员来说，OpenGL 有两个重要优点。首先，OpenGL 非常接近底层硬件，使得用 OpenGL 编写的程序具有较高的运行效率；其次，OpenGL 易于掌握和使用。在本章中，我们将对 OpenGL 进行概述：它适于(或不适于)做哪些工作、它的组织结构以及我们如何对其进行表示。



1.1 OpenGL API

在现代计算机的诸多应用中，计算机图形学是一个很重要的方面。无论是在我们访问网页、玩互动游戏还是使用 CAD 软件包设计房屋，我们无一不在与计算机图形学打交道。随着硬件和软件的速度以及复杂性的日益增长，我们所使用的图形程序也“水涨船高”。这些应用程序的开发者创建应用程序时需要借助标准的软件接口。这样的接口使程序员可以不必为实现那些许多应用程序所共有的标准功能一次次重复编写代码，而且使程序员在编写应用程序时不必关心图形硬件的细节。这样，我们的开发效率越来越高，与此同时，程序的可移植性也得到了提升。程序员可以借助一套具有精心定义的接口的函数来与图形系统进行交互，我们将这套函数称为应用程序编程接口(Application Programming Interface，API)。

近年来涌现出许多图形 API。其中的一部分例如 GKS 和 PHIGS，已经上升到国际标准的水平，其他一部分 API 在特定的领域也得到了广泛应用。其中的大多数都惨遭夭折。另外一部分 API，如微软公司的 DirectX，只局限于特定的平台。OpenGL 来自于一个称为 GL(Graphics Library 的缩写)的接口，该接口最初是为 SGI 公司的硬件开发的。【1】

GL 的简单易用和功能强大得到了广泛的认可。它构成了 OpenGL 的基石，如今 OpenGL 已为多种图形硬件所支持。OpenGL 包含 200 多个可用于构建应用程序的函数。使用 OpenGL 编写的程序可被移植到任何支持该接口的计算机。在几乎所有的计算机和操作系统中都有 OpenGL 的相应实现。这些实现方式跨越了从纯软件实现到充分发挥目前最先进的硬件的加速功能的实现。一个典型的 OpenGL 应用程序可运行在具有任意实现方式的平台中，所要做的只是将针对该系统的 OpenGL 库重新进行编译。此外，OpenGL 还具有高度稳定性，这就保证了用 OpenGL 编写的程序具有很长的生命期。即便 API 随着硬件的发展而不断演化，情况仍然如此。

OpenGL API 主要关注绘制(rendering，也称呈现)；即依据物体的规格参数以及属性，借助虚拟摄像机和光照生成一幅该物体的图像。由于 OpenGL 程序的平台无关性，所以 OpenGL API 不包含任何输入或窗口函数。原因很简单，因为这两者都严重依赖于特定的平台。然而，无论图形程序运行在何种平台上(Windows，Linux 还是 Macintosh)，都不可避免地要和操作系统或本地窗口系统进行交互。面对这种情况，我们并不打算屈从于编写平台相关的代码，而是采取一种折衷的策略——借助一个简单的工具集，即 OpenGL 实用工具集(OpenGL Utility Toolkit，GLUT)。该工具集在标准编程环境中都有相应的实现，其 API 包含大多数窗口系统所共有的标准操作，并允许我们在应用程序中使用键盘和鼠标。

1.2 关于 OpenGL 的三种观点

虽然从某个层面上来看，OpenGL 只是一个具有访问硬件能力的函数库，如果能够从三种关联的方式来看待 OpenGL，还是很有意义的。理解这些观点将有助于我们更好地理解 OpenGL 和图形系统是如何发挥功能的，从而能更轻松地创建自己的应用程序。

1.2.1 程序员的观点

一般而言，大多数图形应用程序主要包括如下三要素：

- 指定绘制对象
- 描述对象的属性
- 定义这些对象被观察的方式

现代图形系统支持两种不同的对象类型：几何类型和图像类型。几何对象通常存在于三维空间中，即包括如直线、点、多边形等简单对象，也包括如二次曲面或更一般的曲线和曲面的复杂实体。从图形学发展的历史看，计算机图形系统只与几何对象打交道，本书的绝大多数内容也主要关注这类对象。然而，随着硬件技术的迅速发展，图形系统目前也完全能够对图像进行处理。图像实际上就是像素的矩形数组，其中的每个元素可用来表示像素的颜色、灰度或其他属性。【2～3】

大多数 API 将对象的定义(例如是直线，还是多边形)与其显示方式进行了分离。这样，一条绿色的实线就与一条红色的虚线属于同一对象类型，仅仅是外观上存在差异。虽然这种分离貌似违背了面向对象的思想，但这样做与硬件的实际工作方式更为接近，而且使开发人员可将程序的逻辑流程与指定描述表面属性的诸多参数的冗长细节隔离开来。

计算机图形学遵循了现代物理和工程模型，因为图形学将对象的描述和行为与其显示方式进行了分离。为了生成一幅图像(image)或图片(picture)，我们必须同时对对象和形成图片的摄像机或视点方位进行描述。在 OpenGL 中，有一小部分函数专门用于对虚拟摄像机进行定位和对其进行描述。

虽然所有产生图形输出的程序必须包含以上三个要素，如果某个程序是交互式的，它还必须包含一些输入函数。最后，所有的程序必须具有一些针对本地操作系统和窗口环境的初始化和终止的函数。

以上观点给出了一种 OpenGL 函数的分类方法，在下一节中，我们将对 OpenGL 的函数进

行分类。但是，这并没有揭示 OpenGL 的实际工作原理。另外两种观点可为我们带来一些启发。

1.2.2 OpenGL 状态机

现在我们换一种角度，将 OpenGL 看作是一个具有输入和输出的状态机(state machine)，如图 1.1 所示。其中输入为几何对象的描述信息(例如线段和多边形)和离散对象(如位数组)，它们都是通过 OpenGL 函数调用来指定的。该状态机的输出为一个离散图元的集合或构成图像的像素集合。在输入和输出之间，是一台能够获取对象的描述信息、并将这些信息转化为一幅图像的机器。这台机器处理输入的方式取决于其内部状态。按照这种观点，OpenGL 具有两种类型的函数，即那些指定输入对象的函数以及那些修改机器状态的函数。状态修改函数包括那些指定颜色、观察条件、材质属性以及许多其他变量的函数。状态决定了输入的处理方式。【3～4】

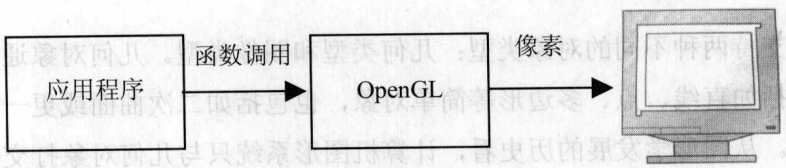


图 1.1 作为状态机的 OpenGL

1.2.3 OpenGL 绘制流水线

从实现角度，我们可以不同的方式来看待 OpenGL 程序的行为。OpenGL 的基础是一个所谓的流水线模型(pipeline model)。图 1.2 展示了一个简单的流水线。所有的实时系统和商业显卡都包含一个硬件流水线。几何图元，如多边形，可在程序中通过描述空间位置或顶点来指定其形状而由应用程序生成。这些顶点流经一系列模块，每个模块在图元经过时对其实施一种或多种操作。几何模块负责对流经的图元实施一种或多种变换，包括旋转、平移、缩放以及相对于 OpenGL 摄像机为图元进行定位。该模块也被用来决定某一对象对虚拟摄像机的可见性。位于绘制流水线末端的帧缓存(frame buffer)包含在显示设备上将要显示的像素信息。位于几何处理器(geometry processor)和帧缓存之间的是光栅化器(rasterizer)，它负责生成片段(fragment)，或者说那些可能位于几何对象中的像素以及一个片段处理器(fragment processor)，该处理器可对这些片段上色，并判别这些片段是否被其他对象所阻挡以及它们在图像中是否真的可见。