HZ BOOKS
华章科技

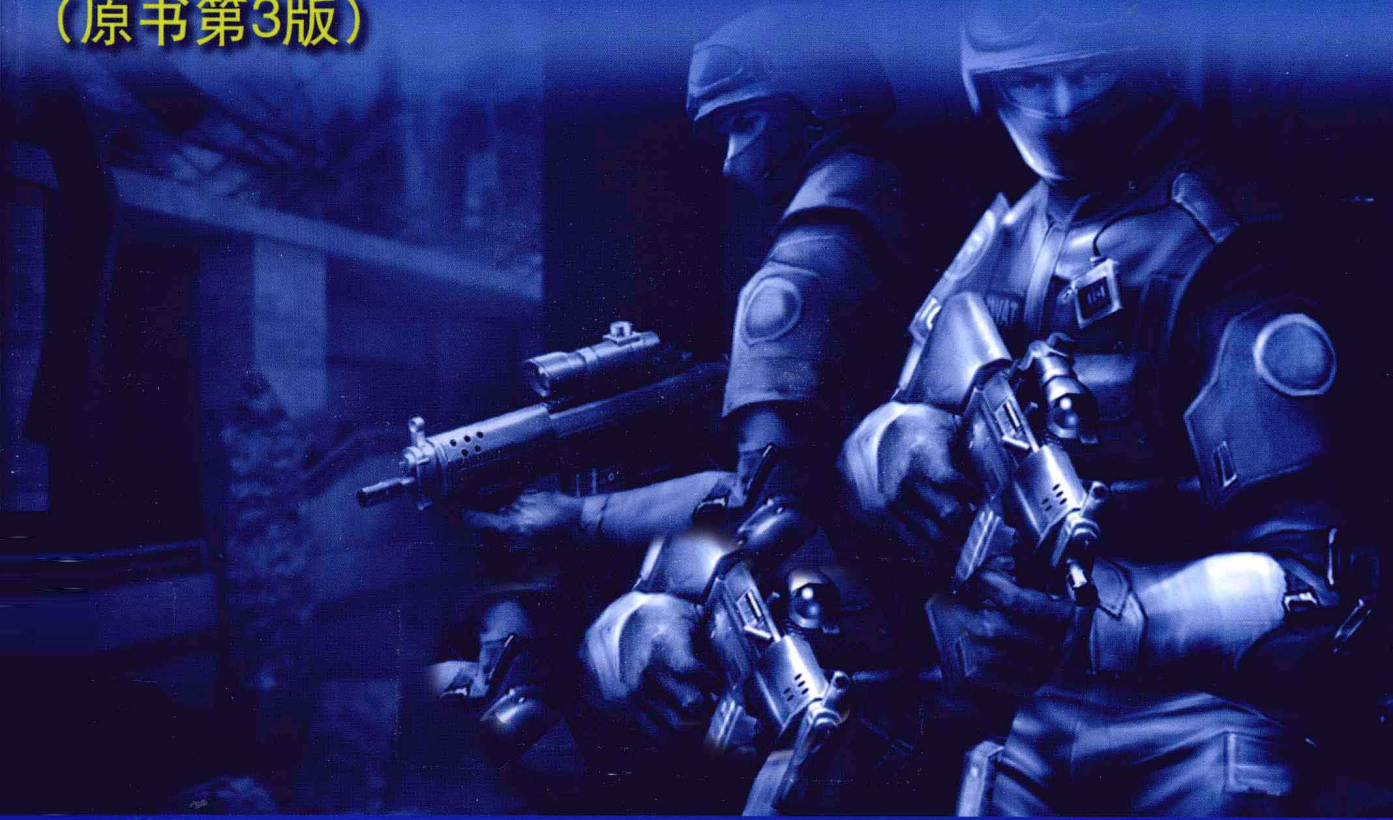
CENGAGE Learning

游戏开发技术系列丛书

BEGINNING GAME PROGRAMMING (Third Edition)

游戏编程入门

（原书第3版）



附赠光盘

（美）Jonathan S. Harbour 著
陈征 傅鑫 等译

机械工业出版社
China Machine Press

游戏开发技术系列丛书

BEGINNING GAME PROGRAMMING (Third Edition)

游戏编程入门

（原书第3版）

Jonathan S. Harbour 著
陈征 傅鑫 等译

机械工业出版社
China Machine Press

本书从基本的 Windows 编程开始，为游戏编程入门者介绍了使用 DirectX 在 Windows 下编写游戏所需的基础知识。读者将学习到把思想转化为现实所需的技术，比如 2D、3D 图形的绘制、背景卷动、处理游戏输入、音效、碰撞检测等。

本书语言简练，适合有志于进入游戏编程世界且有一定 C++ 编程基础的初学者阅读，也适合作为社会培训机构的培训教材。

Jonathan S. Harbour：Beginning Game Programming，Third Edition
EISBN: 978-1-435-45427-9

Cengage Learning Asia Pte. Ltd.
5 Shenton Way，# 01-01 UIC Building, Singapore 068808

本书版权登记号：图字：01-2010-5149

图书在版编目（CIP）数据

游戏编程入门（原书第 3 版）/（美）哈本（Harbour，J.S.）著；陈征等译.—北京：机械工业出版社，2011.1
（游戏开发技术系列丛书）
书名原文：Beginning Game Programming，Third Edition

ISBN 978-7-111-32860-5

Ⅰ. 游… Ⅱ. ①哈… ②陈… Ⅲ. 游戏－软件设计 Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字（2010）第 254220 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：秦 健
北京市荣盛彩色印刷有限公司印刷
2011 年 1 月第 1 版第 1 次印刷
186mm×240mm • 19 印张
标准书号：ISBN 978-7-111-32860-5
ISBN 978-7-89451-808-8（光盘）
定价：55.00 元（附光盘）

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；88376949；68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

译 者 序

游戏是计算机世界永恒的主题。玩游戏容易，写游戏就不那么简单了。很多初学者想自己编写游戏软件，却发现有大量鸿沟摆在面前，想一个一个越过却找不着门道。

本书从基本的 Windows 编程开始，为游戏编程入门者介绍了使用 DirectX 在 Windows 下编写游戏所需的基础知识。读者将学习到把思想转化为现实所需的技术，例如 2D 和 3D 图形的绘制、背景卷动及处理游戏输入、音效及碰撞检测等。

本书语言简练，适合有志于进入游戏编程世界的初学者阅读。无论读者是否具备 DirectX 的知识，甚至哪怕只是个中学生，只要有一定的 C++ 编程基础，就能从中获益。

参加本书翻译的人员有：陈征、傅鑫、孟庆麟、戴锋、许瑛琪、王开年、易小丽、陈婷、管学岗、王新彦、金惠敏等。

由于时间紧迫，加之译者水平有限，错误在所难免，恳请广大读者批评指正。

译者

2010 年 9 月

序

“我想做一个游戏设计师，我该怎么做才能得到这份工作？”这是我在面试或者与学生交谈时经常被问到的问题。在我和我的团队离开时，一位极具天赋的少年的父母曾经和我搭讪。我通常的回答是：“那么，你设计过什么？”这时，绝大多数人会给我一个冗长的解释，说他们有许多伟大的想法，但就是缺少让自己梦想成真的团队。我对此的反应是，跟他解释和我一起工作的人都有伟大的想法，但只有很少一部分是设计师。

我并不是有意要这么苛刻，但是，不会有任何成功的公司愿意给初出茅庐的人一个开发团队，给他超过 18 个月的时间和几百万美元的预算，却无需任何概念上的证明。这就是现实。Sid Meier（传奇游戏设计师，我很荣幸能和他一起在 Firaxis Games 工作）这样的人之所以能够鹤立鸡群是因为他不但能够采纳别人的想法还能将其转化为有趣的东西。当然，Sid 现在有了大团队来做他的项目，但他总是以相同的方式开始做项目，用他能找到或者自己制作的美工作品和声音作品来粗略制作出一组原型。正是这些粗糙的概念表达使得与创作过程不相关的人们可以立即看到一个想法的有趣之处，他从而也获得了预算和团队。每位崭露头角的设计师会记下笔记然后问道：“Sid 会怎么做？”

于是，一本像这样的图书就显得弥足珍贵。我认识 Jonathan 已有两年，当时我在游戏开发者大会的书店里看到了本书最初的版本。我的一个程序员朋友帮我从大量的相似书籍里把它挑了出来。他认为这是一本写得很好的书，而且认为对 DirectX 的强调将非常适用于我们在 Firaxis 所做的工作。另外一个同伴提及他曾经读过 Jonathan 关于 Game Boy Advance 编程的著作，而且印象深刻。我觉得他们给了我极好的建议，我在通读本书时获得了极大的享受。在阅读的时候，我注意到 Jonathan 是我们的游戏——Sid Meier 的《Civilization III》的粉丝。于是我联系了他，因为我从事过 Civ 系列大量的工作，并从此与他保持联系。

像这样的一本书，漂亮就漂亮在它将所有的借口都扫光了。它完美地介绍了游戏编程。它将携手你完成编写 C 代码并利用 DirectX 来实现游戏这一似乎复杂的过程。你在全然未知中已经得到了一个完全可用的框架来实现你的想法。你甚至可以创建自己的美工效果和声音的工具，让它们给游戏上妆。换言之，读者将得到所有的制作原型及证明自己并不仅仅是只有伟大想法的人所需的工具。相信我，经过这关键的一步，你就会成为那些想在这一行业中找到工作的人中的佼佼者。你将具备脱颖而出的能力，而目前有非常多的人都想在游戏开发中分一杯羹，这种能力至关重要。

那么，Sid 会怎么做？当他去年在制作 Sid Meier 的《Railroads!》原型的时候，他用 C 写了整个原型。他那时没有美工人员（他们那时都忙于其他项目），于是他学习了一种 3D 美工程序，自己做美工，然后把这些用到游戏中，他经常使用文本标签来保证游戏者知道游戏中涉及的东西。他使用来自前一个 Firaxis 游戏和 Internet 上的音频文件，四处点缀，增强游戏者的体验。他在很短的时间内创造出了一些东西，这些东西让发行商和其他人了解了这个游戏会有多么有趣。

欢迎前来冒险

欢迎来到游戏编程的冒险地带！笔者热衷游戏和游戏编程有许多年了，我和你一样对这一曾经神秘的主题充满激情。游戏，这里指的是 PC 游戏，曾经只出现在奇客的国度里——这是一个冒险家探索无限想象世界，然后自己努力创建出相似世界的地方。同时，在真实世界中，人们过着正常的生活。

那么为什么我们宁可当宅人呢？是因为我们觉得看屏幕上的像素更有趣吗？没错！

但有些人眼里的像素却是另外一些人的幻想世界或外太空冒险。最早的游戏只是屏幕上推来推去的一堆像素而已。但即使在过去玩那些原始的游戏时，我们的想像力还是经常会给我们带来更多意识不到的细节。

那么，你的激情是什么？或者说你最喜欢的游戏类型是什么？是经典的射击街机、幻想冒险、实时战略游戏，角色扮演游戏，还是与运动相关的游戏？我希望读者在阅读本书的同时能够在脑海中设计一个你自己的游戏，随着逐步深入到每一章，都应想象要如何创建这个游戏。本书并不想通过一堆带补丁的代码清单和下一步该如何操作的建议，给读者一个对游戏开发的“暖融融的”感觉。我希望读者读完最后一章时能获得一种完成的喜悦感。从某种程度上说这是一本完备的书籍，读者可从中学到制作自己早期的游戏项目的实用知识。在这里，读者学到的东西将足以用来编写一个完整的、质量足够好的游戏，并且能够满怀信心地与他人分享。

本书将教授读者如何用 C++ 语言编写 DirectX 代码。游戏编程是一个极富挑战的主题，不仅难以精通，而且难以入门。本书通过使用 C++ 和 DirectX 这两种行业工具来揭开游戏编程的神秘面纱。读者将学到使用 Windows 和 DirectX 来渲染 2D 和 3D 图形的方法。

我们将学习编写简单的 Windows 程序的方法。以此为基础，我们将学习 DirectX 的关键组成部分：Direct3D、DirectSound、DirectInput 和 D3DXSprite。本书将教会读者如何利用这些关键的 DirectX 组件以及如何编写通俗易懂的简单代码。在这个过程中，我们会将所有从每章收集来的新知识放到一个游戏库中，以便在将来的游戏项目中重用。在学习了编写简单游戏所需的所有知识之后，读者将看到创建一个横向卷轴射击游戏的方法！

从何处开始

我的游戏开发哲学是让普通程序员有的放矢。本书将从一开始就进入正题，而不是讲解标准 C++ 库中的每个函数调用。所以，如果读者对 C++ 还不熟悉的话，那么现在就开始学习它吧。可以肯定的是，有大量和本书所用的语言一样强大（甚至有过之）的伟大产品可以为我们所用。例如 Blitz Basic（见 Maneesh Sethi 所著的《Game Programming for Teens》一书）和 DarkBASIC（见 Jonathan Harbour 和 Joshua Smith 所著的《DarkBASIC Pro Game Programming》第 2 版），这是两种能向用户提供完整工具包的游戏开发工具，包括编译器、编辑器、游戏库 / 引擎以及生成无需任何类型的运行库和独立的 Windows/DirectX 游戏的能力。如果读者是个 C++ 语言的新手，或者

而他自己一个人做了这一切，就如他在车库里工作的那些“旧时光”一样。

那么，你会怎么做？如果你想在业内获得一份游戏设计师的工作，甚至只是想制作一个酷酷的游戏来教你的女儿学数学，你就应该购买这本书。投入进来，完成练习，开始开发你自己的游戏库——Sid 还在用一些还是 Commodore 64 时代的代码。让想像力自由飞翔，然后找到将想法转换为人们可以实际享受的东西。

无论做什么，只要去做就可以。这是设计师的学习和成长之路，也是实现游戏设计师梦想的金钥匙。如果 Sid 还不是 Sid，也没有那么多可以运用自如的工具，那么他可能已经开始在设计这些工具了。

Barry E. Caudill
执行制作人
Firaxis Games
2K Games
Take 2 Interactive

目　录

第三部分 游戏项目

第四部分 附 录

第一部分 >>>

Windows 和 DirectX 游戏编程引言

本书第一部分介绍 Windows API（应用程序编程接口），这是开始 DirectX 编程之前必须掌握的基础知识。前两章通过讲解编写简单 Windows 程序的方法、Windows 消息系统的工作原理以及创建不停息的消息循环（正是它给 Windows 程序带来了高性能）的方法，让读者对 Windows 的工作原理有一个大概的了解。第 3 章介绍 DirectX，读者在这里将学习创建 Direct3D 渲染设备以及设置渲染系统的方法。

- 第 1 章　Windows 初步
- 第 2 章　侦听 Windows 消息
- 第 3 章　初始化 Direct3D

第 1 章　Windows 初步

让人趋之若鹜、不掌握不痛快的计算机编程技术中，游戏编程显然是最复杂的编程形式之一。游戏不仅是巨大的技术成就，更是一种艺术工作。许多在技术上令人惊奇的游戏无人问津，而在技术上不是那么精湛的游戏却大行其道，给制作者带来滚滚财源。尽管我们的最终目标是要成为一名游戏程序员，但作为爱好，这会是你所有的爱好中最让人享受的一种，它所带来的感受可谓有苦有甜。而我希望你已经做好了进行这一冒险的准备！本章提供的是开始编写 Windows 游戏所需的重要信息，它是后面两章的前导，在那里会给出 Windows 程序机制的概要介绍。

本章将展示一个简单的 Windows 程序。这些信息对后续三章的学习很重要，因为需要依靠这些知识将读者带入 DirectX 的世界。如果对这些介绍性的知识掌握不牢，那么以后还需要随时回来复习，因为随后的章节将依赖于我们对 Windows 工作原理的基本理解。如果已经有编写 Windows 程序的经验，则将对学习本书非常有帮助，不过，我并不做这种假定，而是在此给出 Windows 程序的基础知识，这些正是开始编写 DirectX 代码所需的。

实际上，只要投入其中，就不难发现 Windows 编程其实颇为有趣！虽然有些代码看起来可能像外星文字，但很快你就会非常熟悉它们。如果本章的内容让你觉得不知所措，那么不要太过担心，因为后续章节中还会有重复，这样会让你牢记要点所在。本章的目标是展示编写一个简单的 Windows 程序、创建项目、键入代码及编译、运行这个程序的方法。

本章将学到：

- 如何正确看待游戏编程。
- 如何按需选择最好的编译器。
- 如何创建 Win32 应用程序项目。
- 如何编写简单的 Windows 程序。

1.1　Windows 编程概述

如果你是 Windows 编程新手，那么将体验到很棒的经历！因为对于编写游戏，Windows 是一个重要的操作系统（不过以前可不总是如此）。首先，Windows 下有许多很棒的编译器和语言。其次，它是世界上最流行的操作系统，任何针对 Windows 写的游戏都有流行起来的潜力。而 Windows 的第三个伟大之处在于有令人惊异的 DirectX SDK 为我们效劳。DirectX 不仅是如今最广泛使用的游戏编程库，它还很容易上手。不过不要误解我的意思——Direct X 易于学习，但想精通它却是另一回事。本书将教授读者如何使用它，或者说如何运用它来创建我们自己的游戏。如果要精通它，则单单这本书所能提供的还远远不够。DirectX 非常值得花时间去学习，尤其是如果你想跟上游戏开发的最新研究成果的话（因为如今大多数关于游戏开发的文章和书籍的着重点都是 DirectX）。

在开始编写 DirectX 代码之前，需要学习如何编写简单的 Windows 应用程序，并且学习

Windows 处理消息的方法。那么，就让我们从头开始吧！什么是 Windows？

Windows 是一个多任务、多线程的操作系统。这句话的意思是，Windows 可以同时运行多个程序，而这些程序中的每一个又都可以有许多运行中的线程。不难想象，这样的操作系统架构能够和诸如 Intel Core2 和 i7 这样的多核处理器良好共事。

建议　作为参考，本书中的程序（和截图）是在一台具备 Intel Core2Quad Q6600 处理器、2 GB DDR2 内存、Nvidia 8800GT 512 MB DDR3 视频卡的 PC 上开发的。在编写本书时，这是一台性能处于中上游水平的 PC。

1.1.1　认识 Windows

没有几个操作系统能像 Windows 这样一个版本一个版本地逐级延伸。目前使用中的许多 Windows 版本（本书编写时主要是 Windows Vista 和 Windows XP）之间差别不大，为某个版本编写的程序几乎无需修改就可以在另一个版本上运行。例如，在 1998 年使用 Microsoft Visual C++ 6.0 在 Windows NT 4.0 或 Windows 98 下编译的一个程序仍旧可运行于最新版本的 Windows XP 和 Vista 之上。你的游戏库中或许还有几个 20 世纪 90 年代后期出品的支持 DirectX 早期版本的游戏（例如 DirectX 8.0）。如果这样的游戏仍旧可在新 PC 上运行，你也无需惊奇。这无疑是非常好的事情，因为这是一个我们可以依赖多年的、能够运行我们的代码的平台。而这也是 Windows PC 游戏开发人员不甚满意而控制台游戏（console game）开发人员较为满意的地方之一，因为我们可以相信控制台系统不会改变，而 PC 变化太快，在某些情况下，有些游戏在一两年之后就产生技术问题了。

好，我们证实了 Windows 程序的长寿（在软件工业中也称为“保质期”）。那么，Windows 到底能做什么呢？

建议　在本书中，凡是提及“Windows”的地方，所指都包括了与当前主题——PC 和游戏编程相关的最新 Windows 版本。也就是说，应该包括所有以前的、当前的和将来的兼容 Windows 版本。从实用的角度来说，实际上这仅限于 32 位程序。从这里开始，凡是提及“Windows”的地方，我们均可认为其包含了所有诸如这样的版本：Windows XP、Windows 2003、Windows Vista 和 Windows 7。例如，本书的前两个版本分别使用 Windows 2000 和 Windows XP 开发的，但它的源代码在最近 5 年内只需很小甚至不需要更改。

根据所编写的程序类型不同，Windows 编程可简可繁。如果你有编写应用程序的经验和开发背景，那么就会对图形用户界面（GUI）编程的复杂性有很好的理解。只需几个菜单和窗体，就足以把你淹没在数十个（就算到不了数百个）控件中难以自拔。作为多任务的操作系统，Windows 非常优秀，因为它是消息驱动的。面向对象编程的拥护者争辩说 Windows 是个面向对象的操作系统。事实上它不是。在工作方式上，当今最新的 Windows 版本几乎和 Windows 的早期版本（例如老的 Windows 286、Windows 3.0 等）一样——驱动操作系统的是**消息**，不是对象。操作系统就如同人类的神经系统，只是没那么错综复杂。如果将人类的神经系统以抽象的方式来简化，那么就会看到在人体中，刺激通过神经元从感官器官传递到大脑，而后从大

脑传递到肌肉。

建议　虽然 64 位计算是将来的潮流，但对于程序员来说，它不会像从 16 位到 32 位的转换那样成为一个大问题，因为处理器、操作系统和开发工具都同时转换了，所以这种转换将几乎不可察觉（现实就是如此）。

1.1.2　Windows 消息机制

下面通过一个常见的场景来帮助我们对操作系统和人类神经系统进行比较。假设你的皮肤上的神经检测到了某些事件，例如温度的改变或者某些东西对你的触摸。如果使用右手的手指触摸左手臂，会发生什么？你会“感觉”到触摸。为什么呢？当你触摸你的手臂时，感觉到触摸的不是手臂而是大脑。“触摸”这一感觉并非由手臂本身感受到，其实是大脑定位到了事件，于是你识别出了触摸的来源。这几乎就如同对中枢神经系统中的神经元进行查询，看它们是否参与了“触摸事件”。大脑“看到”触摸消息传递链中的神经元，于是就可确定手臂上发生触摸的位置。现在，请触摸你的手臂，并且在手臂上来回移动手指，你感觉到发生什么了吗？这不是持续的“模拟”测量，因为在皮肤上有许多离散的触摸敏感神经元。运动的感觉实际上是以数字的方式传递到大脑的。你可能会反驳我的论断，认为压力的感觉是模拟的。在这里我们深入到了某种抽象概念中，但笔者认为压力的感觉是以离散增量传递到大脑的，它并不是以电容性的模拟信号来传递的。

这个概念和 Windows 编程有什么联系呢？触摸的感觉与 Windows 消息有非常类似的工作方式。外部的事件，例如鼠标单击，会导致小的电信号从鼠标传递到 USB 口，然后再进入系统总线，可以认为这是计算机的神经系统。操作系统（Windows）从系统总线检出这一信号并且生成一个消息传递给正在运行的应用程序（例如我们的游戏）。而后，程序就如有意识的心灵那样对“触摸的感觉”做出反应。计算机的潜意识（处理所有事件处理逻辑的操作系统）将这一事件“呈现”给程序，让其知晓。

建议　随着时间的推移，高级信息系统似乎趋向于模仿神经世界，未来我们最终构造出的终极版的超级计算机，可能会和人类大脑相像。

目前还有一个问题。人类可没有两个大脑。还记得笔者关于技术模仿生物大脑的评述吗？当今大多数处理器厂家都朝着在单个硅芯片上集成多个处理器内核的方向前进。多核系统在今天已是常态，绝不是特例（本书在 2004 年首次出版时还不是这样）。

DirectX 9、10 还是 11？

编写本书时 DirectX 10 已经存在两年时间了，而且 DirectX 11 已经处于早期的测试阶段。实际上，在 CD-ROM 中包含的 DirectX SDK 中，就有 Direct3D 11 的一个早期版本。这对于我们这些仍旧使用 DirectX 9（这是本书所关注的）编写游戏程序的人来说意味着什么呢？

对 DirectX 9 的支持仍旧如此广泛的原因是它支持 Windows XP，而 Windows XP 仍旧是最广泛使用的 Windows 版本（因为大多数个人和企业还没有升级到 Vista）。DirectX 10 及更新的版本还不能运行于 Windows XP，但 DirectX 9 的代码可运行于任何 Windows 的现代版本上——这就

是 DirectX 9 仍旧流行的原因。

就算 Windows 7 正式发布时会带 DirectX 11，但我们可继续编写 DirectX 9 代码，因为在可预见的将来，DirectX SDK 将继续提供所有编译并运行 DirectX 9、10 和 11 程序所需的头文件、库文件和 DLL 文件。

1.1.3 多任务

首先，Windows 是一个抢占式多任务操作系统，也就是说计算机可以同时运行多个程序。Windows 通过让每个程序运行很短的时间——以毫秒，或者秒的千分数计算的时间来实现这一特性。从一个程序非常快地跳转到另一个程序称为**时间分片**，Windows 通过为内存中的每个程序创建虚拟地址空间（一个小的“模拟的”计算机）来处理时间分片。每当 Windows 跳转到下一个程序时，会储存当前程序的状态，以便在轮到这个程序接受处理器时间时能够接着执行。这些状态包括处理器寄存器值和任何可能被下一个进程覆盖的数据。然后，当程序以时间分片方案重新得到处理时，这些值就会被恢复到处理器寄存器中，这样程序可以回到离开的位置继续执行。

建议　不要觉得这样做是对处理器周期的浪费，要知道就在这么几个毫秒中，处理器可运行好几百万条指令。现代处理器已经可达到每秒 10 亿次浮点运算的水平，在短短的“时间片”中很容易就能完成巨量的数学计算。

可以认为，Windows 有其自己的基于事件的中枢神经系统。当我们按下一个键，就会有消息从**按键**事件中创建并且在系统中散播，直到有程序检出这个消息并使用它。因为这里提及“散播”，所以在这里需要澄清一点。Windows 3.0、3.1 和 3.11 是**非抢占**式操作系统，从技术上说它们只是 16 位 MS-DOS 之上的非常先进的程序而已。Windows 的这些早期版本更像是 MS-DOS 的外壳，而不是真正的操作系统，所以无法真正地“拥有”整个计算机系统。我们可以编写一个可完全接管系统的 Windows 3.x 程序，无需为其他程序释放处理器周期。只要愿意，甚至可以锁住整个操作系统。早期的 Windows 程序为了获得“Windows Logo”认证（在那个时候是重要的营销问题），必须释放对计算机资源的控制。Windows 95 是第一个 32 位版的 Windows，而且它是这一操作系统家族的革命性进步，因为它是**抢占式**操作系统。

这里的意思是，操作系统有非常低级的内核用于管理计算机系统，没有任何程序可以接管这一系统，不像 Windows 3.x 那样。抢占式意味着操作系统可以抢占一个程序的运行，使其暂停，并且，操作系统可在以后让程序再次启动运行。当有许多程序和进程（每个都有一个或多个线程）请求处理器时间时，就称其为时间分片系统，这也是 Windows 的工作方式。不难想象，在使用这样的操作系统时，多处理器系统实在是很有优势。

对于游戏开发者而言，多核的 Intel 或 AMD 系统是很棒的配置。首先，SMP（symmetric multiprocessing，对称多处理）处理器通常有更多的内部高速缓冲存储器。处理能力是越大越好！在过去，玩游戏或者开发游戏时可能必须关闭大多数运行中的其他应用程序，但现代的多核系统可轻而易举地处理多应用程序同时运行的问题，在编写游戏时根本觉察不出系统运行有拖拖拉拉的现象。当然，巨量的内存也有帮助——游戏开发至少要 2GB RAM。既然已经开工，那么就购买你的系统上可处理的最快的内存芯片吧，因为对于计算机硬件而言，速度比容量更好！

图 1-1 显示了非抢占式多任务工作原理的概览。注意，每个程序都接受处理器控制，并且之后必须显式释放控制以便计算机系统正确工作。这样的程序还必须小心的是，不能占用太多处理器时间；实质上，非抢占式操作系统程序必须自发共享处理器。

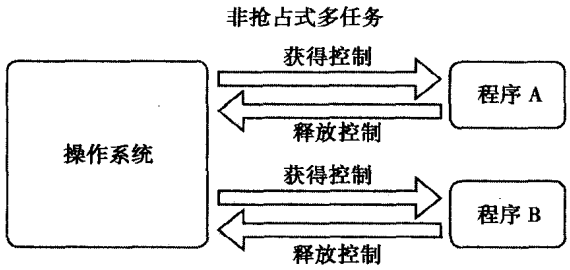


图 1-1　非抢占式多任务要求每个程序自发释放控制。操作系统对应用程序的控制非常有限

图 1-2 显示了抢占式多任务的工作原理。不难看出，图 1-2 与图 1-1 相似（这样易于比较），不过，现在操作系统控制所有的一切，无需等待程序“听话”并且共享处理器时间。在时间片分配的毫秒数满了之后操作系统只需挂起程序即可，在对系统中所有运行中的进程和线程循环一遍之后，再把更多处理器时间交给这个程序。

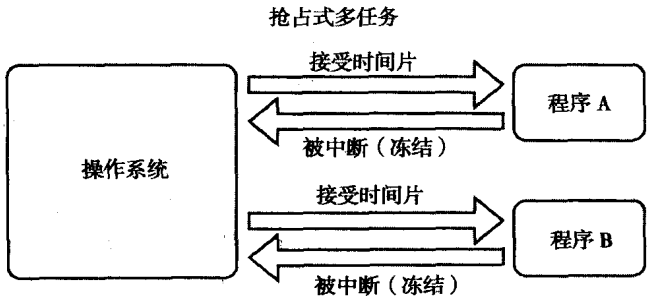


图 1-2　抢占式多任务操作系统对系统有完全的控制，它为每个运行中的进程和线程分配时间片

1.1.4　多线程

多线程是将程序分解成多个独立的、为了完成某个任务（或者完成独立的任务）而一起工作的部分。这与系统级别的多任务不是一回事。多线程有点像多 - 多任务，每个程序都有其自己的运行部分，而这些小程序片段对操作系统执行的时间分片系统毫无察觉。对于主 Windows 程序及其所有的线程，它们对系统有完全的控制，而对操作系统将时间片分配给每个线程或进程并无“知觉”。所以，多线程意味着每个程序能够将处理托付给其自己的迷你程序。例如，象棋程序可以创建一个进程以便在游戏者忙于考虑下一步棋时提前思考。“思考”线程可在等待游戏者的同时继续更新着法和对攻着法。虽然这可以在等待用户输入时使用能够思考的程序循环很容易地实现，但具备将这一过程交付给线程来执行的能力可给程序带来显著的好处。

举个例子来说，我们可以在一个 Windows 程序中创建两个线程并且让每个线程有其自己的循环。对于每个线程，其循环无尽运行并且极快地运行，没有中断。但是在系统级别上，会给每个线程授予一个处理器时间片。依据处理器的速度和操作系统的不同，线程每秒可能会被中断 50、100 甚至 1000 次，但线程对这样的中断毫无察觉（想象一下我们在夜里睡觉时会醒来许多次！与人类不同，计算机并不会注意到！）。图 1-3 说明了程序、进程和线程之间的关系。

建议　多线程编程是个迷人的主题，值得我们花时间学习！笔者在《Game Programming All In One》一书的第 3 版（ISBN 1-59863-289-2）中简单讲解了这一主题。讲解了能让多线程编程变成小菜一碟的 Posix Threads 库的使用方法。在读完本书之后，它会是很好的后续材料。笔者发现大多数初学者都能很快地学会 Allegro 游戏库。如果你准备好接受更大的挑战，那么笔者的针对这一主题的新书《Multi-Threaded Game Engine Design》（ISBN 1-4354-5417-0）值得一读。不过，如果对 C++ 和 DirectX 尚不熟练，要做好进行巨量编程训练的准备！

多线程对游戏编程非常有用。在一个游戏循环中涉及的许多任务都可以交付给可独立执行的不同线程实现，每个线程都与主程序通信。其中一个线程可用于自动处理屏幕更新。而后，程序必须做的就是确认所有的对象都在屏幕上，并且双缓存以特定的时间进行更新，而这个线程将按时执行工作——甚至可能使用内置的计时措施来保证无论使用什么处理器游戏都能以统一的速度运行。大多数流行的游戏引擎都是多线程的，也就是说它们天生就支持多处理器。这对于那些为了买多核系统而花费更多的游戏者而言，实在是物有所值。而独立的游戏服务器（经常提供流行的在线游戏以便游戏者运行他们自己的游戏）如果能够支持多处理器就更为理想，因为它需要大量的处理能力来处理有许多游戏者的大型游戏。双处理器游戏服务器更能够处理大量游戏者。

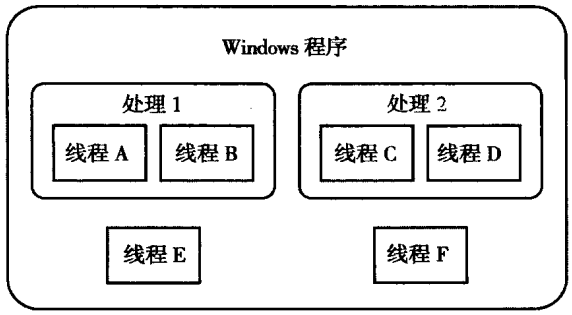


图 1-3　多线程程序可以有多线程处理也可以有独立的线程

建议　双缓存是内存中的一种位图映像，可以用它来为游戏绘制图像，然后这一映像会被复制到屏幕上，以非常平滑的渲染效果显示出来。

1.1.5　事件处理

到这里，有人可能会问："Windows 如何与这么多同时运行的程序保持联系呢？"首先，Windows 处理这一问题要求程序必须是事件驱动的。其次，Windows 使用全系统范围内的消息来通信。Windows 消息是操作系统发送给每个运行中的程序的小数据包，它有三个主要内容——窗口句柄、实例标识符和消息类型，用于告诉程序有事件发生。事件通常涉及用户输入，例如鼠标单击或按键，不过事件也可以来自网络库的通信端口或 TCP/IP 套接字（用在多人游戏中）。