第 0 章　简介

如果不考虑大公司购买的大量商用软件包，那么游戏软件的销售高于其它任何软件。这一事实并不足为怪，一个好的计算机游戏就像一本好小说，把你带入一个神奇世界，而你周围的切似乎变得像另一个星系那么遥远。无论是轻松的娱乐游戏(如 Tetris)，一个使头脑麻木的测验游戏(如 Lemmings)，还是一个在神奇王国的妖魔鬼怪的冒险游戏(如 Ultima)。一旦你开始玩一个好的游戏，即使爆发核战争也不能把你从计算机前赶走，除非你已尽兴并打算关掉机器了。

也许在某天深夜玩完游戏后，你脑子里会闪过这样的念头，是不是编一个游戏程序会比玩游戏更带劲，至少你可以决定让计算机游戏干些什么事情，你可自己设计一座城堡，让里面住满令人毛骨悚然的怪物。当你的朋友和那些可怕的妖怪作战时，你将会因为是唯一知道所有答案的人而洋洋自得。

不管你是否相信，坐在计算机前玩自己编的计算机游戏是一件很刺激的事情。在玩游戏取乐的同时，你会越来越想自己编制一个程序，无论要花多少天，多少个星期，多少个月来琢磨程序，你在玩的时候仍会乐趣无穷。

0.1　程序设计的预备知识

本书并不是学习 C++程序设计的入门教材。要搞懂本书内容，应具备 C++的知识，并熟悉 Turbo C++系统。除此外，你至少还应了解一点关于面向对象的程序设计概念。假如你对 Turbo C++还有点生疏的话，文中的注释栏可帮你迅速回顾有关 Turbo C++的重要内容。假如以前你很少进行面向对象的程序设计，就只能采用速成进修的办法了，本书的附录 A 提供你所需要的关于这种新的程序设计方法的简洁介绍。

0.2　硬件和软件配置要求

要编译和运行随书所附磁盘上的程序，并从以后的课中得到更多信息，必须有以下软硬件配置：

- ☐ 带有 640K 基本内存的与 IBM 兼容的 80286 微机，并至少有 1 兆的扩展内存
- ☐ MS－DOS 3.31 版或更高版本的操作系统
- ☐ 一个硬盘驱动器
- ☐ 一个与 Microsoft 鼠标兼容的鼠标器
- ☐ VGA 显示器
- ☐ Turbo C++　3.1

通常，处理器的速度越快越好。快的处理器意味着迅速编译和快速程序运行，这对于游戏程序尤其重要，它往往迫使你最优配置。

0.3 本书概述

本书提供了大量的程序设计技术和技巧，帮助你设计所有的计算机游戏。以下是有关本书的概述。

第一章“游戏程序设计介绍”，阐述了设计计算机游戏所需的一些技巧。你将学习如何设计游戏，建立用户界面，处理图形和动画以及写简单的人工智能程序。

第二章“计算机游戏图形设计”，介绍了用一个绘图程序建立所需图形的技术，描绘逼真的三维图形，为金属和玻璃等实体涂色、加阴影和反射光，建立游戏模型等，而且多数技术几乎不需要艺术技巧。

第三章“事件驱动程序设计”，解释如何在 DOS 下设计和编写由事件驱动而不是仅仅通过用户键盘输入的程序。键盘和鼠标事件也包含在内。本章讨论事件处理和发送，并在用户的鼠标处理中提供一个基本类。

第四章“图形控制和窗口”，在第三章中的事件驱动程序设计中加入图形增强功能，用户可采用面向对象的程序设计技术开发一个窗口、对话框和按钮件的基本库。

第五章“Life 游戏”，介绍如何用在第四章中开发的窗口库建立一个富有特性的游戏程序。这一程序以 Conway 的生命游戏，即最早提出的人工生命模型程序为基础，为你提供了一个用一系列的单细胞计算机实现生命组合的机会。

第六章“加载和显示画面”提供了一个能处理 PCX 格式图形文件的 C++类。用户将学习 PCX 文件格式以及在处理这类图形文件的注意事项。本章还探讨了对于包含 16 种颜色的图像，特别是在处理由多个位平面图像时的一些特殊考虑。

第七章“创建游戏图像”，继续讨论在第五、六章中提出的计算机图形问题，引导用户建立自己的游戏程序图像。用户将学到如何从屏幕上“剪切”和在磁盘文件上保存图像。除此外，本章还提供了一个被称为图像机(The Image Machine)的图形实用程序，它将帮用户建立自己游戏的图形文件。

第八章“Dragonlord(龙的主人)”，介绍了一个商品化的地牢冒险游戏。当设计 Dragonlord 时，要采用前面几章学过的所有技术，并开发一些新技巧。在处理游戏图形时，采用了面向对象的窗口设计理论，采用了面向对象的程序设计以及其它一些技术以跟踪游戏者的进程。

第九章“纸牌游戏”，介绍了各种类型的计算机纸牌游戏程序设计技术。本章不仅阐明了一副纸牌的所有图像设计，而且介绍了在显示屏和内存中处理纸牌的 C++类。两个演示程序包含一个简单的黑桃 J 游戏给用户示范如何在自己的程序中使用 C++类。

第十章“扑克方阵”，采用前一章所建立的纸牌规则来开发基于扑克的纸牌游戏。扑克方阵中，在 5 乘 5 网阵的行和列上，你必须尽力去建立最有可能的几手牌。除了要搞清如何分析手中的牌以外，还应知道如何建立高分文件并显示出来。

第十一章“Battle Bricks”，带你涉足了用 Turbo C++来设计运动游戏的问题。当你要开发一种被称为 Battle Bricks 的崩落型游戏时，你将学到如何使目标在显示屏上保持移动，以及如何在执行程序的同时建立声响效果。

第十二章“晶体”，提供了编程设计计算机游戏者的快速入门技术。这种策略游戏来源于古埃及的一种游戏，当你努力想从游戏板上捕抓到最大数量的晶体的同时，也给你提供了一个与

计算机挑战的机会。

0.4 本书程序的编译

本书的程序均是采用 Turbo C++ 3.1 编写的。这里假设你采用了缺省的设置和目录安装了 Turbo C++。如果已修改了其中的一些缺省设置或目录，而却没有把握预料这些改变会产生什么错误，就该重新安装 Turbo C++。

所有程序在磁盘上以章节顺序排列。用户会发现每一章的程序都在各自目录下面。第二章的程序在 CHAP02 目录下，第三章的程序在 CHAP03 目录下等等。要编译某章的程序，把整个这章目录下的内容拷贝到 Turbo C++ 目录中，然后启动 Turbo C++ 并装入要编译的程序的工程文件。例如，要编译第六章的程序，应首先把 CHAP06 目录下的文件拷贝到 Turbo C++ 目录下，然后把在新的 CHAP06 目录下的对应工程文件调入内存(用户必须首先解开压缩的程序文件，请参考本书后面的磁盘安装指南)。

0.5 使用程序须知

正如每一程序员所知，好的程序是有保护措施的。对于易于出错的每一步，您必须检查错误并给用户显示恰当的错误信息。遗憾的是，好的错误校验需要大量的附加程序代码。对于要做下一步的程序员来讲，这只是游戏的一部分，但对于撰写程序设计书的作者来讲，这些附加代码则有着特别的含义。

一本关于程序设计的书应尽可能地把它的主题阐述清楚。也就是说这些特色程序的源代码一定不能被大量的，而不直接服务于主题的细节搞得含糊不清。基于这一原因，本文并不是所有程序都有恰当的错误校验。用户的输入有时是无法检验的，假设目标的动态构造是成功的，而指针却并不总是检查有效性的。

简言之，如果你要在你自己的程序中使用本书中的任何代码，你应该加的任何的错误校验在本文中已被删掉了。千万不要在程序中作任何假设，假如在你使用代码的一些地方你并没有100%的把握肯定程序的运行状态，你就必须增加错误校验以保证你的用户使用时程序不致崩溃。正是由于本文作者在错误校验方面的粗枝大叶(为了一些正当的理由)，你应更加小心。

为保持源代码的清晰，本文所附程序通常避免使用一般的 C++ 简化算法，特别是嵌套的函数调用。

0.6 游戏该开始了

现在你已经对本文有了概括的了解，该到了用 Turbo C++ 设计游戏程序的时候了。马上你就会发现不仅游戏程序设计很有趣，而且这也是从计算机和 Turbo C++ 中获取更多知识的好方法。

克里根·沃那姆

一九九四年三月

第一章 游戏程序设计介绍

在狂热的青年时代，我是一个半职业摇滚乐队的吉它手。我永远不会忘记第一次走进录音棚与乐队一起录制一首歌的情景。在控制室里，有一块巨大的混音板(mixing board)，上面布满了比大白鲨牙齿还多的按钮和开关，右侧是一个插头安装板，从上面拉出了几十根电缆，每根都与一些重要设备相连。指示灯在闪烁，磁带转动着，一些有着特别名字的声音处理装置，像移相器、数字延迟器和多频补尝器在滴嗒作响。

当我看见所有那些复杂的机器，并意识到为享有在那儿的特权我要每小时付 60 美元(相当于今天每小时 150 美元)时，我几乎要走出录音棚。对我来说，在这个设备高级的录音棚转转，好像要花掉一生的积蓄似的，最后，眼睁睁地看着自己被剥削得身无分文，却连一个音也没录上。

幸运的是，有了在录音棚的经历，使我至少知道一般情况下录音棚是如何工作的，除此外正如生命中的每一件事情一样，录音棚实际上也不像表面看起来那样神秘。

计算机游戏也是如此，当你坐在计算机前玩最新的 arcade hit 或沉浸于最新的冒险游戏时，你可能会敬畏那些进入到你眼帘的发光像素的才能(你也应如此)。但是，正如在录音棚录一首歌一样，写游戏程序并不像你想象的那样困难。假如你已有一些编程经验，也具备了大量的编写计算机游戏程序的知识和技巧，只需要提炼那些着眼于游戏的技巧就可以了。本章中，你将学习到开发计算机游戏的一些技巧。

1.1 为什么要设计设游戏程序

也许你买这本书只是想借助计算机消遣一下，你在书店里正探寻着那些很严肃的编程手册，当这本书从书架上跳出来的时候，你想“嘿，这多么赏心悦目!”，但是当你拿着这本书走到收款台时，也许会有点受骗的感觉。总而言之，游戏总归不是严肃的计算，对吗？你应该学习如何写电子表格的程序，学习数据库、文字处理，对吗？

听我讲个小故事。

那是在家用计算机还未发展的年代(确切说是 1981 年)，我拥有了自己的第一台计算机，是 Atari 400，那时，Atari 这种功能较强的微机同样以其游戏运行能力而闻名。而 1981 年，是电视游戏黄金时期的开始，Atari 则应称王了。

只可惜，几年的繁荣过后，电视游戏销量急剧下降，并拖跨了许多工业公司，计算机又变得严肃起来。虽然 Atari 也打算重振旗鼓，而由于它作为游戏计算机制造商的地位，始终不会作为严肃计算机计设者和生产者被认可。这是件不幸的事情，因为 Atari ST (Atari 400/800 的后继型)与 Macintosh 和 Amiga 一样，功能是超前的，这些产品比当时流行的“严肃”的 IBM 系列产品要先进得多。

问题在于 Atari 的那些该死的赌博游戏。谁想用一台游戏计算机来管理大量电子表格文件，平衡银行帐目，以及跟踪投资过程？它只是一个哑巴，难道不是吗？

事实上，能够运行奇妙游戏的计算机几乎能做任何事情，好的计算机游戏迫使你购买最高档的计算机，这包括它能迅速处理数据，生成图形和动画，建立逼真的音响效果。只有最新的计算机才能赶上当今功能极强的电子游戏，象 Microsoft 的 Flight Simulator 5.0。事实上，几乎很少有商务应用程序要比复杂的计算机游戏应用程序对计算机功能需求更高的情况。

同样，一位能设计商品化计算机游戏的程序员，完全可以设计其它的软件，尤其当今那些体现图像和声音应用的软件。可能你看这本书是为了借助于计算机消遣，但是在你的计划实施前，你将学到关于软件设计和程序编制的很有价值的内容——这些内容可用于许多不同类软件的设计。

因此，之所以要编程设计计算机游戏，很大程度上是因为它很趣，但也应知道游戏的编程经验将为你遇到的每一程序设计提供帮助。哎呀，既学习了，又娱乐了，多么完美的结合啊！

1.2 游戏程序设计要素

正如你发现的那样，好的计算机游戏会最大限度地发挥计算机本身的作用，事实上，好的计算机游戏在很多方面必须出色。要想设计人们都愿玩的游戏，首先你必须获得程序设计相关领域的经验，这些领域代表着游戏程序设计的要素。

- ☐ 游戏设计
- ☐ 图形设计
- ☐ 声音生成
- ☐ 控制与接口
- ☐ 图像处理
- ☐ 动画
- ☐ 算法
- ☐ 人工智能
- ☐ 游戏测试

这些游戏程序设计要素覆盖面很广，比如要学习里面的图形设计，就需要了解计算机是如何处理图像的，此外，游戏的设计牵扯到表中的其它要素。因此，你必须了解图像、声音、控制和算法是如何融合在一起的，才能设计游戏，并使其最终成为产品。

下面，你将学习更多的关于游戏要素的知识。

1.2.1 游戏设计

不管你的游戏是典型的凶杀案还是复杂的实战演习，这里面游戏者的唯一目的就是攻击屏幕上的任何目标，都是需要机智的头脑和巧妙的移动，你的游戏必须吸引人是第一位的。假如游戏很枯燥，不管图画设计多漂亮，音响效果多逼真，或者说游戏规则设计得多好，都是无济于事的。一份枯燥游戏设计只有在抽屉里积落灰尘的份（见到过软盘被当作杯垫使用吗？那些磁盘里装的一定是糟透的计算机游戏）。

很多因素决定了游戏的趣味性，很重要的一点，当然是游戏的主导思想。游戏的主导思想常常源于现实世界的人和事。比如，国际象棋——一直是很流行的棋盘游戏——属真正的实战演习游戏。而垄断者又是一种金融业的模拟游戏，游戏里游戏者总是试图击败他的对手。

计算机游戏与现实中的原型没有什么两样，对游戏者来讲，它们都有一些符合逻辑的目的，而且几乎毫无例处地被放入一个稍微可信的"世界"里，这个世界可能会象屏幕上的迷宫那样简单，也可能会象拥有大陆、国家和城市的地球那样复杂。在那个易疯狂上瘾的计算机游戏 Tetris 中，游戏者的世界只是屏幕上的一条窄道，他必须在上面堆放不同形状的物体，而在惊人的最后冒险中，游戏者的世界充满了森林、沼泽、城市、魔鬼以及其它东西，构成一部完全荒诞的闹剧。

不论你为游戏设想了一个什么样的计算机世界，这个世界有一定的规律可循，而且游戏者能掌握，是很必要的事情。对一个极有趣的游戏来讲，游戏者必须能领会出越过你放在他路途中的不同障碍方法。当一名选手玩游戏失败了，可能就是由于他还没有掌握这些规则中妙处，而不会是因为一个晴天霹雳把他击成了碎片。

当然，要建立一个符合逻辑的，公平而有效的游戏世界，你必须利用你作为一名程序员的所有技能。前面列出的所有编程方面的其它领域也要在这儿显身手。图画、音响、接口设备、计算机算法和许多其它因素都会决定一个游戏最终是否有吸引力，或者又是一个平庸的作品，它的磁盘又将被作为下一个家庭野餐时的杯垫了。

1.2.2 图形设计

许多计算机游戏的包装上之所以附有生动的说明和令人敬畏的屏幕画而是有原因的。不管人们要作出理智的购物决定有多难，每个人都会为精美的包装而心动。虽然你的聪明的一面会提醒你别太在意表面的难以置信的魔力，而你的冲动的一面会注意到表面的魔力是令人兴奋的暗示。当然，事实往往与包装相差很小，购者当心就是了。

这里并不是想告诉你应该把你的游戏包装比实际的漂亮，关键是，一个游戏产品看起来如何通常同运行起来漂亮一样重要。你想使游戏屏整齐而不杂乱，设计符合逻辑等等，但首要的是看起来生动。你的显示屏应对着每一远处的观看者叫喊"来玩吧"(如果它们叫喊"你母亲做的西红柿沙粒糟透了!"你最好换台新计算机)。

同一些事情一样，图形设计是一种职业技能，它需要多年的钻研和实践才能掌握。有幸的是，要设计吸引人的游戏画面，你不必成为图形设计的能手，因为你能通过不同的屏幕设计来对照哪种更有吸引力以及哪种设计在你的游戏世界里更适合。用你最欣赏的图形作出不同布置，并比较。尝试和错误对实现改进设计不仅是一种强有力的技术，而且也是一种很效的学习工具。实践得越多，你就会得到更多有关显示屏图形的真觉。

1.2.3 声音的生成

实际上，我们生活的世界是噪杂的。我们时时刻刻都同时被上百种声音包围着。如果想使你的游戏世界更逼真，同样要提供音响效果。这并不是说你必须重建真实世界的全方位音响，而就今天的计算机来讲，这也是不可能的。不管怎样，谁想在玩 Commander Dweeb 和 Nasty Neptunium Ninjas 兴致正高的时候听到狗叫或者是电话铃响呢?

虽然你不想在你的用户的耳朵里塞满不必要的噪杂声，但是却应提供恰如其份的音响暗示，当游戏者选择了屏幕上的按钮，她应听到按钮的咔哒声，同样，当游戏者玩回家的游戏时，她应听到击球声和人群的叫喊声(那比真正的击球声和人群的叫喊声美妙多了)。

星球上(或者，我应大胆地说，是在宇宙中)，没有不能用更好的声音效果改进的计算机游

戏，幸运的是，采用象 Sound Blaster 那样功能很强的声音卡，使今天的很多游戏具备了令人难以置信的数字化声音效果。遗憾的是，对于游戏程序设计的新手处理一块声音卡是一个难题，这不仅需要了解市场上许多声音卡的特点，而且还得知晓汇编语言程序设计(发抖了!)。

本书中的游戏是通过 PC 机的扬声器来建立音响效果的，虽然这样的音响效果不如数字化声音逼真，但也有两方面的优点：它适用于每种 IBM 的兼容机并且易于编程。一旦你掌握了这种音响效果的诀窍，就可以拿起一本语音程序设计书，它将帮助你进行声音卡程序设计。

音乐，虽然与语音效果相比重要性次之，但也应在计算机游戏中加入一些。音乐出现的地方就是游戏的开头，通常与片头一起出现。当游戏者升了一级，或者他完成了一些其它重要任务时，你也应想到采用音乐。

在计算机游戏中加入音乐，需要你具备一些作曲知识，不好的音乐比没有还要糟。假如你没有音乐方面的训练，可能你有这方面的朋友，那你们就一起为你的计算机游戏杰作作曲吧。如果你幸运的话，她甚至不会要求与你分享专利权。

1.2.4 控制与接口

游戏中所发生的一切事情都是在计算机里的，与事实世界中传统的广义游戏不同，游戏者不必拿着牌来回移动。要游戏者能够控制游戏，程序员必须提供一些游戏者能够操纵计算机内部数据的接口。在计算机游戏里，菜单和屏幕上的按钮会提醒用户选择操作和命令。除此外，键盘和鼠标如同游戏者的手，操纵或移动着屏幕上的目标。

一个好的游戏接口，游戏的操作是尽可能简单的，游戏者需要控制游戏的命令应该是符合逻辑且有效的。而且，你的游戏越逼真，它的控制游戏者学起来就越简单。例如，在国际象棋游戏中，你应让游戏者用鼠标指针来移动一个棋子，而不是通迫他用键盘键人他想要移动的棋子在棋盘上的位置。

1.2.5 图像处理

每种计算机游戏都必须处理不同类型的图像，这些图像可能是全屏幕的背景图形，代表游戏命令或者游戏部件的图标，或者是用于建图或其它复杂游戏屏的图块。当设计游戏时，必须决定需要什么样的图像。在游戏运行时，应该在程序里画出背景屏，还是用一个画图程序来生成屏幕画面，你只需在程序中调用就可以了？假如想节省内存，也许你该从小图块开始逐一建立游戏画面。

当设计游戏图形时，必须考虑这些问题，希望采用高质量的图形使得你的游戏设计尽可能地职业化(也就是说，你该找一位艺术家)，但是还要考虑这些图形所占的内存空间以及把图形从磁盘上调到计算机内存中需要的时间。大多数游戏者讨厌等待文件的加载，而且把很多数据保留在内存中，但这又会使游戏与内存小的计算机不兼容。

另一重要的问题就是建立游戏图形所需要的时间，你不可能花十年的时间绘制方方面面都详尽的图形，需要采用剪切(象铺砌，就是用小的图形拼成的游戏屏)来加速图形的设计过程。换句话讲，虽然在现实生活中每棵树是不同的，但在游戏里的树通常是没能任何区别的。

1.2.6 动画

一旦你已经学会了设计和处理计算机图像，就该学下一步：动画了。动画是使目标活起来，

并能在计算机屏幕上运动的过程。利用一系列的图像，可以使小鸡在路上摇摆，岩石从峭壁上滚落或者宇宙飞船从发射台上疾驰。

由于动画要花费计算机的大量能量，它要求你对每一需要动的目标均设计两幅或更多幅的图形，这会使你对这种编程技术失去偏好。然而，一些巧妙的动画次序对静止的显示屏能增加许多动画图形。

例如，当游戏者移动游戏部件时，并不是使得这个部件简单地从当前位置消失并又重新出现，你应使它看起来渐渐消失，而后又逐渐恢复，或者说，如果一游戏部件代表了一个人或者一种动物，你应使他漫步到新位置。

这种简单的动画效果能使得你的游戏玩起来更有趣，更吸引人。虽然动画对程序员来说是一项大工程，但却是值得的。

1.2.7 算法

虽然算法这个术语听起来象极讨厌的术语，但却真的是一个简单词。一个算法就是解决一个问题的步骤。在生活中，每天都用算法，你为早餐做煎饼时，必须遵从一个步骤；当你开车上班时，必须遵从另外的法则；这些规则系统帮助你完成生活中简单(有时并不简单)的事。

计算机的算法帮你解决计算问题，换句话讲，要设计计算机游戏，你需要解决如何让计算机做事—— 那些你以前没在计算机上做过的事情。例如，在扑克游戏中你如何决定谁的手气最好，或者如何建立一位聪明的计算机游戏者？你必须设计一个算法，一旦你知道如何用计算机来解决一个问题，你就能写下这些特殊的代码，而不管你采用什么程序设计语言。纵览全书，你可以看到许多解决游戏问题的算法。

1.2.8 人工智能

人工智能例程是那些使计算机显得聪明一些的算法(说到聪明，我并不是说那种能计算游戏者的得分或处理游戏者的输入数据的能力，而是指计算机能够作为一名对抗者的能力)。如果你想设计对抗计算机游戏者的游戏，必须建立能使计算机与人对抗的算法。算法的复杂性决定于游戏的复杂性以及你想使计算机表演的好坏程度。

例如，要建立一个计算机棋手的好的算法是不容易写的，因为赢棋需要大量的计谋，而另一方面，一个计算机棋手的算法也是容易写的，例如，每轮你只要简单地使计算机选择随机的一步棋就可以了，虽然由这种算法生成的计算机游戏者是很容易被击败的。你该明白了，要写的算法决定了游戏的难易程度。

1.2.9 游戏测试

读完本书后，你学到了关于设计和编写计算机游戏的一切，你该开始完成你的杰作了(我希望如此)。然而，写完游戏程序，你的任务远没有完成，因为最后你还必须进行全面地测试，以保证游戏运行良好。

测试一个游戏的最好方法就是把它交给几个可信的朋友，观察他们玩的情况，并记录下那些没按预定设计走的路径。记住，不仅要观察程序的错误(那些使程序做了意想不到的事情或者使计算机出故障的东西)，而且要观察接口的错误，这些错误会导致程序无法运行。

在你的朋友玩完游戏后，就问问游戏里哪些是他们喜欢的，哪些是他们不喜欢的，并找出

那些他们认为该改进的地方。你不必同意他们说的每一点，但应保持谦虚，认真听取他们的建议并记下来以便以后查阅。千万不要有对立情绪，为帮助你做得更好，你的朋友不会对你的设计批评太多。记住，没有完美无缺的程序设计，程序总是有改进的余地。测试完成后，再落实那些有价值的建义。测试游戏的唯一途径就是找几个人不断地玩它。当然，在把游戏交给几个朋友前，你应该把游戏玩得烂熟，以致于宁可逐页地翻看电话号码薄也不愿再看一眼屏幕了。

1.3　小结

设计一个计算机游戏程序，要求把你的拿手的程序设计技巧应用进来。要完成一个成功的游戏，你必须首先设计它，也就是说你必须深入思考，不断地实践，最终完成游戏的图形设计和接口设计。当你设计游戏时，就应该考虑什么样的图像和声音能使游戏活起来。好的动画效果和智能算法也将帮你的游戏成为下一个畅销货。

第二章，你将开始深入学习本章中所预习的论题，尤其要学习如何设计和建立有效的计算机图形。采用一些基本的技术和工具，你可能会惊讶：为游戏或为一些其它的应用程序建立足够的计算图形简直再简单不过了。

第二章 计算机游戏图形设计

通常计算机游戏中最重要的因素，除了其可游戏性外，就是它的图形了。游戏的画面越逼真，就会有越多的人对游戏感兴趣。其实，画面再重要不过了，许多劣质的游戏(对不起，没有名字)变得孤独起来，主要就是由于他们的视觉效果造成的。

遗憾的是，大多数程序员想具备能用几罐颜料画出黑猩猩的艺术天赋。为使设计的游戏有艺术感染力，很多程序员求助于雇佣画家。许多有才华的程序员能设计刺激性游戏的，但当发现自己艺术修养有限的时候，就放弃了自己的游戏程序设计思想。

假如对这点早有耳闻，那么这儿有好消息告诉你：通常用于计算机游戏的图形不是很难画的，学习画简单计算机图形极类似于学习如何把蛋糕粉制成蛋糕，一旦你知道了这是怎么一回事，那就很简单了。很多计算机图形设计只是一种技术，而不是一种技巧。当然，一点关于计算机图形的课程不可能把你变成画家，假如你的游戏需要大量详尽的画面，像人、怪物和建筑，可能仍需找一位画家。但是读下去，你会惊讶地发现，仅学完此课关于图形的课程，你就能学会很多东西。

2.1 三维图形制作入门

游戏常常以其"逼真的三维图形"而自豪，但事实上，计算机屏上的每样东西都是平面的，即二维的。这些图像看起来是三维的，只不过是错觉罢了。本书所称的三维图形是简单的二维图像，但却如因照片那样有景深的视错觉。书中游戏的许多图形，最常用的术语就是"135°视图"或者"伪等角视图"图形。也就是说对一些物体，视角是能看见立方体的三面即前面、侧面和顶面。但在你学习135°视图图形前，将先接触到计算机三维绘图技术的速成课程，并将在第四章"图形控制和窗口"中应用这些技术。

采用图形来产生视深感觉的方法很简单，事实上，只需要几条线。例如，最常用而有效的三维绘图技术是基于所有光线来源于物体上方的思想，毕竟在地球上，太阳光不会从底部照过来，而人工的光源也往往放在视平面或高于视平面处。因此，人们很自然地就把亮的物体表面与顶面联系起来，而把阴影面与底面联系起来。

图2.1的绘图中仅用了三种颜色：白、灰和黑，表明了这一准则。灰是自然的背景色，既不是强光处也不是阴影处。第一幅图里，画家在背景上画了一个翻转的"L"来代表矩形的两条边，中间那副图，画家用白线完成了这个矩形，因此，矩形看起来象凸出来的，使人们产生了三维错觉。最后一幅图，画家只是把基底的"L"改成白线，而用黑色来完成矩形，因此，看起来矩形象是凹进去的。

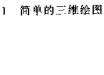

图2.1 简单的三维绘图技术

把这些要素堆积起来(画一个要素要受另一个的限制),就可以建立多层次和不同视深的视觉了,如图 2.2 所示,还可以采用这一技术画出选中后由凸起变为凹陷的按钮。

图 2.2　三维物体的堆积

这种采光方法很有用,它基于以下几点原因,第一,很简单;第二,它利用了一般人的感官和直觉;第三,设计上也很简单,以致于通过编程很容易完成,只要在屏幕上标出几条线,而不是象在画图程序中绘画那样,然后把这幅图加载到你的程序里就可以了。程序 2.1 是一个简短的程序,表明如何用 Turbo C++来画这样的三维图形。

程序 2.1　3DDEMO.CPP——一个简单的三维绘图程序

```
//////////////////////////////////////////////////////////////////
// 3DDEMO.CPP: Simple 3-D graphics drawing demonstration.
//             Make sure that Borland's EGAVGA.BGI
//             graphics driver is in the same directory
//             as the program.
//////////////////////////////////////////////////////////////////
#include <stdlib.h>
#include <conio.h>
#include <conio.h>
#include <iostream.h>
#include <graphics.h>

// Function prototypes.
void StartGraphics(void);
void DrawGraphics(void);

//////////////////////////////////////////////////////////////////
// main()
//////////////////////////////////////////////////////////////////
int main(void)
{
  // Initialize VGA graphics.
  StartGraphics();

  // Draw the 3-D graphics.
  DrawGraphics();

  // Wait for a keypress.
  getch();
  return 1;
}
```

```
//////////////////////////////////////////////////////////////
// DrawGraphics()
//////////////////////////////////////////////////////////////
void DrawGraphics(void)
{
  // Set the fill style and color.
  setfillstyle(SOLID—FILL, LIGHTGRAY);

  // Draw a gray background rectangle.
  bar(100, 100, 300, 240);

  // Draw the top, protruding 3—D rectangle.
  setcolor(WHITE);
  moveto(120, 160);
  lineto(120, 120);
  lineto(279, 120);
  setcolor(BLACK);
  moveto(120, 160);
  lineto(279, 160);
  lineto(279, 120);

  // Draw the bottom, indented 3—D rectangle.
  moveto(120, 220);
  lineto(120, 180);
  lineto(279, 180);
  setcolor(WHITE);
  moveto(120, 220);
  lineto(279, 220);
  lineto(279, 180);
}

//////////////////////////////////////////////////////////////
// StartGraphics()
//
// This function initializes Borland's graphics driver
// for the high—resolution VGA screen.
//////////////////////////////////////////////////////////////
void StartGraphics(void)
{
  int gdriver = VGA, gmode = VGAHI, errorcode;

  initgraph(&gdriver, &gmode, "");
  if ( (errorcode = graphresult()) != grOk)
```

```
    {
      cout << "Graphics error: " << errorcode << '\n';
      getch();
      abort();
    }
}
```

运行此程序时，屏幕上将显示图 2.3，灰色背景上的两个矩形均由四条线构成，而上面矩形是凸出的，下面的是凹陷的。

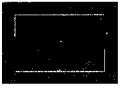

图 2.3　3DDEMO 的屏幕显示

注意：在 Turbo C++程序中使用图形前，你必须首先加载和初始化适当的 Borland 图形驱动程序。要做这一步，必须调用 initgraph()，如在程序 2.1 中的 StartGraphics()函数中所示。要使用 Borland 图形库，在程序头必须有#include <graphics.h>行，以便得在连接操作中选择图形库。在 Option 菜单上选取 Linker 命令，然后选取 Libraries(库)，在所显示的对话框中，打开图形库校检框，图形库就能选取了。

2.2　如何把二维方块变为三维立方体

正如你在程序 2.1 中所看到的那样，用 Borland 图形函数，如 moveto()和 lineto()来制作简单的三维图形是件很容易的事情。然而，要把图形作得更细腻，就必须使用绘画程序了。用绘画程序所画的图形我们称为位图式图形或位图。位图不能采用调用函数来画(至少，不很容易)，而必须把一幅完整的图传送到屏幕上。你将在第七章"创建游戏图像"中学习这种传送。本章只讲授一些简单而有效的位图绘制技术(在你了解如何发动汽车前想在赛车比赛中获胜是毫无意义的)。

把平面图形变成看起来象三维图形的技术是很多的。最复杂的那种既需要大量的艺术技巧，又需要一些非常特别的图形工具(如三维造型和透视软件包)。这种高级方法不是本书讨论的内容。然而，还有一些简单而有效的技术，甚至连一些设计能手在用最基本的图形编辑器和绘画程序制作漂亮图形时都没有采用。

建立三维视觉的一个基本技巧就是加简单的阴影。阴影是形状上与主物体相同的黑色轮廓，位于物体的后面，通过阴影与主物体相离建立视深错觉，如图 2.4 所示，你可以通过在实心黑方块上画一个实心白方块来查看效果。

图 2.4　建立阴影效果

许多计算机绘画程序的阴影效果是这样生成的：拷贝所绘制的图形，把复制件涂上比原件更暗的颜色使复制件看起来象阴影，然后把原件放到涂过色的阴影上，如图 2.4 所示，而原物体看起来要比阴影部分离你更近一些。

如图 2.5 所示，如果为模仿感觉的变化，把阴影缩小一点的话，你会得到更逼真的效果。物体离你越远，就显得越小，同样物体投影的阴影越小，阴影就显得离你越远。

图 2.5 在阴影上加大视深

虽然前述技术是一种在物体和其背景间建立三维效果的一种好途径，但却不是使物体本身具备三维效果的好方法。采用一种阴影的变化技术来绘制三维物体，这种变化同素描类似，就是在相互遮盖的两个方块的角处加上连线。

如图 2.6 所示，首先画家画了两个斜向略微遮盖的正方形，然后，画出两个正方形的对应角顶点连线，构成一个立方体框架，它给出我们的三维的感觉，但并不像实体。为使立方体看起来像实心的，必须选出立方体前面的部分，然后删掉一些透过前面、顶面和可见侧面的线(图中的第三个立方体表示了画家想画出一个实心立方体要去掉的线)。转眼间，一个实实在在的立方体出现了。

图 2.6 制作一个三维立方体

正方形彼此遮盖越少，物体显得越长(如果起初的正方形一点都不遮盖的话，你所画的就不象一个正方形体了，而更象铁轨的枕木)，这一绘图技术不仅限于正方形，对其它一些图形像三角形，九边形等等，你可以如法炮制。除圆和随圆外，这一技术几乎适合于所有的几何图形。由于圆和椭圆没有可供连线的角，因此这类物体的三维制作技术略有不同(参见图 2.7)。

图 2.7 三维椭圆柱和圆柱

告诫：假如你不想使你的三维图形看起来象来自于超空间的话，就要保证两个源图对正，也就是说，一个图相对于另一图不能有相对转动，否则最终就会形成如图 2.8 所示，稀奇古怪的图形。

图 2.8 不恰当的三维立方体画法

现在你懂得了一些一般法则，可以制作一些较复杂的计算机图形了，包括前面提到的 135°图形。在后面的例子里，你将继续使用这一技术，即把同一物体的两幅画面连接起来生成

一个新的三维物体图形。而这回，你是把这一技术用于计算机绘图程序设计，而不是素描了。

2.3　偏置粘贴以达到三维视图效果

Dragonlord 是本文中的一种游戏，它需要一幅由许多独特的房间模型组成的地牢图，并用135°视图表示出来。这种房间的制作是偏置粘贴技术的一个很多好的例证。

注意：这里的偏置粘贴技术，要求使用图形编辑器或绘图程序，或你画拷贝完整的图形以及粘贴复制图而不破坏源图。而且，也要求你能制作"透明"的物体视图。这种物体有透明的背景，即不需删除放于背景处的图像。

要偏置粘贴，首先启动绘图程序，画一正方形框(并非是一个填充的正方形)，然后，在一条边上开个口代表门。现在，用绘图程序里的剪切函数来复制方块，然后粘贴到屏幕上。选一个与第一方块不同的颜色填充到复制图上。如图 2.9 所示，当你要制作一个三维房间时，就可以采用这两个方块作为源图。

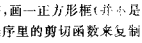

图 2.9　采用偏置粘贴来制作一个房间

采用偏置粘贴技术制作房间，首先要复制你重新上过色的方框，把它移到屏幕的空白处，并粘贴，然后再粘贴另一方框，放到比第一幅图高出一个像素的左侧，如图 2.10 所示。这样重复做几次，而每次粘贴都偏置同样的量，最后复制你画的第一个框并粘贴到刚制作好的那堆图上，偏置量也如其它一样，这样，你就得到了如图 2.9 所示的三维房间图。

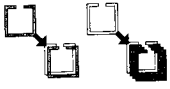

图 2.10　偏置粘贴技术

通过加一些更复杂的阴影，可使得你要设计的三维图形更逼真。一种好方法就是在你采用偏置粘贴技术前，在起初的形状表面加上不同颜色和灰度的阴影。图 2.11 就是一个这类阴影的例子，示例的光源在物体的右上方(如太阳符所标示)，你可通过光源所处的位置来决定物体每个面的亮度。为使效果更逼真，就把每个表面当作一面墙，假如它面向光线的话，就该亮一些，而显然其它面就在阴影里。

图 2.11　偏置粘贴前为物体加阴影

加完阴影后，可采用偏置粘贴技术制作房间模型了。首先把一些加过阴影的图形堆积起来，而每幅图都在水平方向和垂直方向偏置一个像素。然后，用一种颜色填充原先的图并"加"到所堆积图上。假如你的制作最初以图 2.11 所示图形开始，那么将以图 2.12 结束。如你所见，

改变偏置粘贴的方向也改变了物体外观。

图 2.12 采用偏置粘贴给物体加阴影

你可把偏置粘贴技术用于几乎任何形状的物体。如图 2.13 所示，你甚至可以采用这一技术来制作圆物体模型(虽然它们需要更细微差别的阴影)

图 2.13 用偏置粘贴给圆物体加阴影

2.4 特别的提示和技巧

除三维房间和物体外，毫无疑问，在游戏中你需要许多其它物体的图形。一些诸如砖头的物体是很容易制作的，而像电信端口或玻璃球，则需要更多的技巧了。为帮助你入门，本章的后面将论述基本的提示、技术和技巧，使你能够避开或者解决难以绘制的图的绘制问题。

2.4.1 可辨认物体的选择

一般地，在绘制游戏图形中最难的要属制作一个期望很像原型的物体模型了，分辨率和颜色的限制常常使计算机绘图复杂化，但有时仅搞清如何表示一个物体就足以让你到药柜里找阿司匹林。

例如，你的游戏设计可能需要一位戴墨镜的英雄，他还能看见某些其它的游戏物体，但是镜片只不过是玻璃片，很难表示得逼真。不管画得有多好，你能认出这是一副镜片的图吗？一个玻璃片能代表许多其它的物体，也就是说它的视觉意义是模糊的。

这种情况下，在游戏里你就要用可辨认的物体来代替模糊的物体。例如，你应该用眼镜来代替镜片。游戏者就不会把一副眼镜错看成其它物体了。

2.4.2 设计图标

制作计算机游戏图标比爬沾满油污的树还困难。虽然一幅刀剑图标代表着战斗，而一副剪刀图标代表着一些剪切功能，但是，如何表示不易看到的功能呢？诸如存储游戏或显示得分板。

遗憾的是，设计图标是不能讲授的技巧，它需要很多的想象，尝试和教训。对图标来讲，一

幅画面显然并不值一千字，但当你设计图标时，应极力用简单、清晰的图像清楚地代表与其相关的功能。假如不能保证这一点的话，还不如用文字标记来代替图标呢。计算机接口并不比蹩脚的图标设计更令人恼火。

2.4.3　绘制金属件

不管你是否相信，绘制金属表面是计算机绘图里最简单的事情。这里提供你可采用的一种特制的颜色，并给出两种或更多种深浅程度，对于金属平面，只要在物体上画出强光处的斜线就可以了。假如有更多的浓淡程度可供选用，你可以画得复杂一些，使明暗交替出现。图 2.14 表示了这种金属的绘图技术，你所采用的这种特殊颜色的浓淡层次越多，结果就会越逼真。

图 2.14　利用强光及其反射绘制金属表面。

可用类似的技术绘制金属曲面，但必须遵从一些稍有不同的规则。首先强光处必须出现在离光线最近的边缘处，而阴影处则出现离光线最远的边缘。图 2.15 就是一个如何通过改变灰度和设置恰当的强光位置来制作一个看起来象金属件的圆柱的例子。注意，圆柱体阴影部分从左到右由深到浅产生一种光反射的效果。

图 2.15　绘制金属曲面

由于反光许多金属显现出强烈的明暗对比，也就是说，明暗过渡越光滑，这些表面看起来越来越不象金属面，添加特别的强光点或称"热点"(仅热光最强的点，如图 2.15 所示)将使这些表面更象金属面。

2.4.4　绘制玻璃器件

由于玻璃是透明的，画玻璃就象画空气一样都没有必要。你必须画的是玻璃器皿里盛东西的状态。通常，一种最简单的玻璃器件制作就是画出它的一般形状，并象画金属件那样给它加上强光和阴影，然后画出玻璃皿中的东西，并恰当地加上强光和阴影。

图 2.16 展示了这一制作，采用带有白色的强光带和深灰色阴影的浅灰色来画瓶子，然后，用与瓶子不同灰度的颜色来画瓶中的液体，面在瓶的外廓线处用细长的本色线构画以表示玻璃的厚度，以达到透明的效果。

图 2.16　绘制玻璃器皿

因为玻璃也是反光的，因此同金属表面一样，可以通过在玻璃上加强光带来制作玻璃表