
CD-ROM
ENCLOSED
高级多媒体程序设计
[美] Steve Rimmer 著
杨士强 等译校
Advanced Multimedia Programming
■进行 MIDI 处理
■在 Windows 下制作实时动画
4
PHEI
McGraw-Hill
電

Advanced Multimedia Programming

高级多媒体程序设计

[美] Steve Rimmer 著
杨士强 等译校

电子工业出版社

内容简介

本书是利用C语言进行多媒体程序设计的一本很有实用价值的参考书。书中通过大量的实例程序，重点对语音和图形处理技术进行了详细的论述。语音处理部分涉及波形(wave)语音和MIDI音乐处理；图形处理部分包括位图、动画、屏幕保护以及video for windows等。书中程序采用C及C++语言编写，所有程序采用结构化程序设计方法，学过C及C++语言的读者可以很容易地理解与掌握有关技术。与本书一起发行的CD-ROM光盘中存储了所有源程序和执行程序的拷贝，读者可以参考、引用。

本书可供从事多媒体技术研究、开发的工程技术人员阅读，也可作为大、专学生和研究生的教学参考书使用。

Advanced Multimedia Programming
[美] Steve Rimmer 著
Windcrest®/McGraw-Hill 1995年出版

*

高级多媒体程序设计
杨士强 等译校
责任编辑 路 石

*

电子工业出版社出版(北京市万寿路)
电子工业出版社发行 各地新华书店经销
保定市印刷发行公司印刷厂印刷

*

开本：787×1092毫米 1/16 印张：22 字数：563千字
1995年10月第一版 1995年10月第一次印刷
印数：5000册 定价68.00元(含CD-ROM盘)
ISBN 7-5053-3347-X/TP.1282
著作权合同登记号图字：01-1995-161

译者前言

多媒体技术是集文字、图形、图象、语音、动画等数据为一体的综合信息处理技术。多媒体系统的推广普及必将深远地影响着人们的工作、学习、生活等各个领域，将进一步推动人类的文明与社会进步。

对于从事多媒体系统的研究开发人员来说，如何充分利用现有的资源与各种先进的开发工具，高效快速地开发出适合不同应用领域的多媒体系统，是当前非常迫切的问题。目前市面上虽然有很多涉及多媒体技术的专业书籍，但基本上都是进行原理性的叙述，不能为技术人员提供有实用价值的参考。本书正是为了满足这一方面的需求而翻译出版的。原书作者 Steve Rimmer 本身就是一位从事多媒体技术开发的专家，同时又是一位作家。他已出版了十几本有关计算机与图形学方面的专著，他的另一本多媒体专著《Multimedia Programming for Windows》已经译为中文并在国内发行，本书是它的续篇和姊妹篇。

书中利用大量的实例程序，重点对语音和图形处理技术进行了详细论述。主要涉及波形(wave)语音和 MIDI 音乐处理；位图、动画、屏幕保护以及 video for windows 等。书中所附的程序采用 C 及 C++语言编写，所有程序采用结构化程序设计方法，学过 C 及 C++语言的读者可以很容易地理解与掌握有关内容。读者学过有关多媒体技术的原理以后，可以进一步利用本书的实例程序进行实习，达到学用结合的目的。已经具有一定技术水平的开发人员，也可以利用书中的原理与程序，进一步提高理论水平与实际工作能力。总之，本书为读者掌握多媒体程序设计技术提供了一种切实可行的手段。同时，我们也希望通过本书的出版对正在蓬勃兴起的多媒体热潮提供有力的支持。

本书由杨士强负责组织翻译、校译。第三章、第七章及第四章的部分章节由杨小勤翻译，其余章节由杨士强、张国光、慕岩翻译，最后由杨士强进行审校整理。由于译者水平有限，内容有不妥之处，敬请批评指正。

译　者

1995 年 9 月于清华大学

目　录

第一章 序 论

本书是笔者所著的《Multimedia Programming for Windows》的续集和姊妹篇。那本书也是由 Windcrest/McGraw-Hill 出版社出版的，它主要介绍了 Windows 下的多媒体实用工具。本书将补充前者的内容，而且增加了一些 Windows 基本系统中没有包括的内容，提供给用户一个新的开发工具，包括动画处理、屏幕保护以及有关 Video for Windows 新资源的管理等等。

本书的内容是完全独立的。如果读者曾经阅读过《Multimedia Programming for Windows》，会有利于更好地理解本书的内容。关于 Windows 下实现多媒体技术的许多基本概念，例如 RIFF 文件(资源交换文件格式)、语音的采样等等，在那本书中已经进行过详细讨论。

从事"计算机科学"研究与从事程序设计是性质完全不同的两件工作，如同有些人在艺术上很有特长，而另一些人则擅长于科学研究一样。编写软件是一件技术性很强的工作，而且要有创造性和想象力。画家面对一幅空白的画布就如同看到了他所想象的图画，程序员面对空白的屏幕或窗口，就应该想象如何通过编写应用程序来实现自己要完成的工作。

多媒体作为一个应用领域，它的最大特点是没有一个明确的界限，它所涉及的研究开发领域是完全开放的。所谓"多种"媒体，可以包括人们所想象得到的、能够作为信息载体的任何事物，包括声音、图形、动画、音乐等等，以及你认为可以作为信息传播载体的任何东西。如果一个人能够从事与"多媒体"有关的工作，那么他就可以干他所喜欢的工作，而且可以获得丰厚的收入。

但是，本书所涉及的 Windows 下的多媒体技术局限在有明确定义的范围之内。Windows 的基本系统并没有真正的多媒体处理功能，然而它提供了一些策略，在系统内部保留了多媒体扩充的接口。Windows 多媒体扩充系统与 Windows 基本系统集成在一起，提供了一个系统调用库，以便处理类似于播放声音和音乐、显示视频图象、有选择性地播放光盘上的音频数据等功能。

如果仅仅用 Windows 的多媒体扩充系统来代表"多媒体技术"这个整体的研究领域，是不切合实际的。然而，它的确是学习和掌握多媒体技术的一个很好的入门工具。

1.1 声音及其来源

Windows 下的多媒体实际上是一些独立的资源的集成，这些资源是可以完成下列一些操作的系统调用：

★ 记录和播放 WAV(波形)类型的采样声音文件。

★ 记录和顺序播放 MIDI(Musical Instrument Digital Interface)类型的音乐文件。

★ 记录和播放 AVI(Audio Video Interleave)类型的数字视频文件。

★ 管理不经常使用的输入设备，如游戏棒。

★ 选择和播放光盘的音频数据。

最后一个功能将不在本书中介绍，因为在《Multimedia Programming for Windows》中已经进行过详细的叙述。

Windows 多媒体扩充系统可以处理下列几种复杂的文件类型：采样的 WAV 声音文件，顺序的 MIDI 音乐文件，Video for Windows 的 AVI 视频文件。这些类型的文件可以被 Windows 多媒体扩充系统的各级系统调用所支持。Windows 的媒体控制接口(MCI)象黑盒子一样处理这些文件，只要将多媒体文件的路径告诉 MCI，向 MCI 发送播放命令，系统就能自动完成所有操作。而操作人员完全可以不必关心系统内部的处理过程。

Windows 也允许用户在底层完成对这些媒体数据的操作，可以在这个环境中完成媒体的播放操作，并且了解每一步操作过程的细节。但是如果希望在打开一个应用程序时让计算机产生一点声音，却是非常不方便的。如果希望继续增加 Windows 多媒体扩充系统所提供的功能，就必须在底层开始开发工作。

下面这个播放 MIDI 音乐文件的实例可以帮助理解为什么有时需要涉及底层开发工作。对于播放 MIDI 音乐文件的最简单方法是，将这个文件传送给 MCI 接口(这些将在第六章详细介绍)。在 MCI 控制下播放 MIDI 文件的源程序代码仅需 20 行左右。

播放 MIDI 音乐文件的一种比较复杂的方法是，将每个音乐文件分解成离散的小节(称作它的 MIDI 消息)，然后将这些消息直接送到计算机的 MIDI 接口。如果用软件控制这一段操作，大约需要用几百条源代码才能完成，但是所完成的功能与通过 MCI 接口用二十几行代码实现的功能相同。

然而，如果想在应用系统中增加一些 MIDI 音乐硬件的话，MCI 接口可以检测这些硬件，但是不能控制它们，因为 MCI 接口只能播放以 MIDI 文件形式存储在磁盘上的音乐。以资源对象与指令数据形式存储的 MIDI 音乐文件对于一般的用户来说是很难理解的，为了克服这些问题，用户需要深入理解 Windows 中 MIDI 接口的底层工作原理。

图 1-1 所显示的是 PLAYSONG 应用程序的主窗口，这是本书要详细讨论的一个实例程序。图中的 PLAYSONG 应用程序通过选择三个传统 Celtic 小提琴音调来播放音乐而不用依赖于 MIDI 文件，它显示了 Windows 下 MIDI 最底层的工作过程。然而 PLAYSONG 的主窗口只是一种演示工具，它不能通过 MCI 接口调用。图 1-1 中应用了三个对话框，让用户来选择被播放音乐的声音。这些可以帮助你理解 MIDI 的工作过程以及 MIDI 音符信息的实际结构。

《Multimedia Programming for Windows》主要介绍 Windows 所提供的有关多媒体元素的高级接口，但是书中很少涉及到有关底层调用的内容。

在 Windows 下开发软件时经常会遇到这样一个问题：几乎所有的工作都需要一些实用工具，而这些实用工具通常是 Windows 下所没有的，或者是没有提供方便的使用方法。一个真正的好软件是很难编写的，就连 Windows 多媒体扩充系统的作者也这样认为。

本书所涉及的一些多媒体方面的内容，不是 Windows 多媒体扩充系统所完成的功能，特别是 Windows 多媒体实用工具没有增加任何有关处理静止图形的 Windows 资源。在 Windows 下显示图片的方法一直是非常模糊的，尽管已经有了多种看起来有些重复甚至互相矛盾的方法，但是在 Windows 有关的软件开发工具的文档中并没有给出如何从中选择合适的方法的指导。本书中讨论了一种显示图形的可行方法。

利用第四章和第五章所介绍的位图图形的有关技术，可以用来显示图象。因此读者一定不要错过阅读这些内容。这两章不仅叙述了如何简单地显示图形图象，并且介绍了图形文件的各种处理及显示方式。另外，还包括 Windows 下的动画技术，Windows 的游戏棒接口，以及如何创建 Windows 的屏幕保护程序。

Windows 的屏幕保护软件是当前商用软件市场上发展很快的。这是由于在屏幕上显示一

些生动的带有艺术色彩的运动图形是非常吸引人的。

本书的最后一章介绍微软公司的 Video for Windows 软件开发包(SDK)的 1.1 版本。尽管设计 Video for Windows 的最初动机只是程序员的一种异想天开的设想,但是这个新的软件开发包可以让用户相当容易地应用 Video for Windows 的功能来实现各种构思。

也许有人会说书中的实例程序并不好用或者说并不能拿过来直接使用,有这种疑问的原因是由于对多媒体技术缺少全面的理解。多媒体就象是一块空白的大画布,要靠自己的想象去创造最美的画面。可以从自己的多媒体软件中选择合适的操作函数,以便去实现人们的创意。应该把多媒体想象成是一个绘画过程,而不应简单地把它想象成是一幅画好的图画。

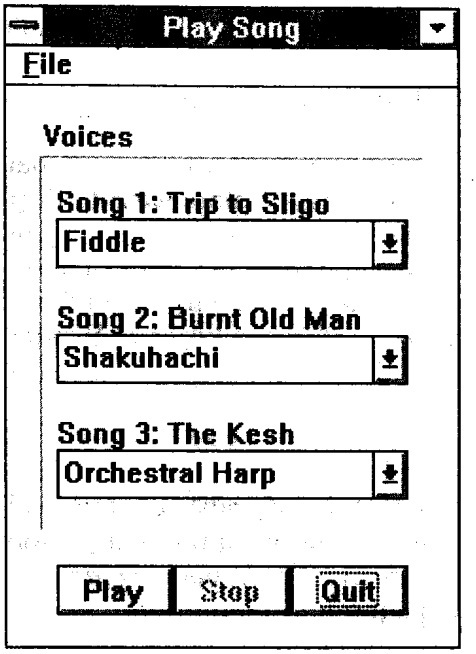

图 1-1 PLAYSONG 应用程序

1.2 硬件和软件环境

本书中的程序采用的语言是 Borland C++ for Windows。如果使用其它版本的 Windows 下的 C 语言编译器,基本上可以编译运行这些程序,但是需要注意下面几个问题。

书中所有实例程序的源代码、资源元素、DEF 文件、PRJ 文件等都已放在与书一起发行的光盘上。读者只要按照 README. TXT 文件的提示,将源代码安装到硬盘上,应该很容易编译这些实例程序。然后就可以开始编写自己的多媒体应用程序了。

在创建自己的程序时,需要按照下列条件设置编译选项:

★ 存储器模式应该设置成小模式或中模式。

★ 默认的字符类型应该是无符号数。

★ 寄存器变量应该被关闭。

★ 跳转优化应该被关闭。

本书中实例应用程序的工程文件(project file)已经设置成特殊的目录结构,如果在用户的硬盘上有不同的目录结构,一定要在每个工程文件中改变库文件和包含文件的目录结构,否则编译器会出现错误提示信息。

下面是用于这些应用程序的目录路径:

★ Borland C++ for Windows 的路径是:C:\BC

★ 书中的源程序代码被安装在 C:\BC\SMPW

★ 书中所有应用程序的包含文件路径为:C:\BC\INCLUDE

★ 本书最后一章 Video for Windows 的包含文件的路径为:C:\VFWDK\INC

★ 库函数路径为:C:\BC\LIB

书中所有应用程序的实例都需要调用微软公司的多媒体开发工具包(MDK)所提供的多

媒体扩充系统，MDK 软件的包装如图 1-2 所示。

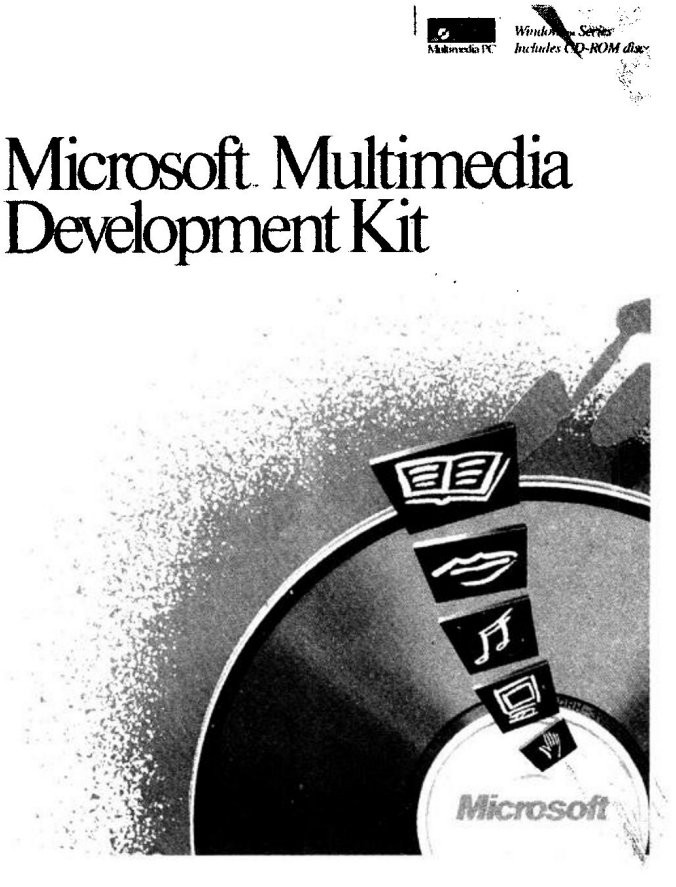

图 1-2 微软公司的多媒体开发工具包(MDK)

在编译书中的实例程序之前，应该首先在硬盘上安装 MDK。有些应用程序还需要通过 Windows 的“控制面板”来安装几个驱动程序才能正常运行。驱动程序的安装将在有关章节叙述。

第七章的 Video for Windows 软件需要调用 Video for Windows 软件开发包(简称 Video for Windows SDK)1.1 版。同时也需要在硬盘上安装 Video for Windows 1.1 版。

如果在编译过程中遇到错误或警告信息，一般来说这些错误信息是由编译器产生的。只有一个例外的情况：在最后一章 Video for Windows SDK 中产生的警告信息并没有实际意义，因此可以忽略。

此外，书中所涉及的一些应用程序都是专门针对 Borland C++设计的，如果应用不同的编译器，如微软的 C 语言，可以删除这些与 Borland 语言相关的特殊语句。特别是 Borland 中的 BWCC.DLL 客户控制库(custom control library)在本书中大部分的应用窗口和对话框中是用来创建点阴影(drop shadow)的，它可以在大部分应用程序的“About”对话框中建立消息对话框并且管理位图。

如果使用非 Borland 语言，应该进行下列修改：

★ 书中的每一段源程序都包括了 BWCC.H 头文件，当使用其它非 Borland 编译器时，必须删除 #include 语句。

★ 每一段应用程序都在它的 WinMain 函数中调用了 BWCCGetVersion，请删除这个调用。

★ 大部分应用程序具有 RC 文件，它们包含 BorBtn 类型的 Borland 客户按钮控制函数，请用传统 Windows 按钮来替换这些控制按钮。

★ 有几个应用程序用到了其它的 Borland 客户控制函数，例如用 BorCheck 来检查对话框，请用同样类型的 Windows 控制函数替换它们。

★ 所有应用程序的 RC 文件都包括 BorShade 客户控制函数。这是用于点阴影处理的，它们的作用更加吸引人，也更先进，但是没有其它的意义。请用传统 Windows 的矩形来替换它们或者删除这些操作。

★ 除去第四章中的 BMP-FX 应用程序之外，本书中的每个 RC 文件都有一个“About”对话框，它是一个资源代号为 ID 801 的 Borland 按钮。这是有关 Borland 按钮工具的一个非常特殊的应用，它显示一个位图。在非 Borland 语言的环境下可以删除这些按钮，第四章中讨论了在对话框中显示位图的替代方法。

最后要指出的是，本书中的每一个源程序都包括下列宏定义：

```
#define DoMessage(hwnd,string)BWCCMessageBox(hwnd,string,"Message",\
                                MB_OK | MB_ICONINFORMATION)
```

如果在非 Borland 语言的环境下编译书中的应用程序，应该作一点修改：

```
#define DoMessage(hwnd,string)MessageBox(hwnd,string,"Message",\
                                MB_OK | MB_ICONINFORMATION)
```

DoMessage 宏调用被用来显示简单的对话，它利用由 BWCC.DLL 所提供的 BWCCMessageBox 调用，去创建一个消息对话框，这个对话框与书中出现的其它对话框基本上是一致的。如果 BWCC.DLL 不能使用的话，可以使用传统 Windows 系统中的 MessageBox API 调用。

书中的某些应用程序要涉及到一些硬件配置。为了使书中的软件更有效地运行，建议尽量使用先进的机器，起码是 386 或 486 系统，“奔腾”机也不会觉得浪费。Video for Windows 需要一块快速的视频加速卡。最少需要 8MB 内存，才能突破 Windows 多媒体扩充系统的局限，当然最好是 16MB。另外，要保留尽量多的磁盘空间，并且需要安装 CD-ROM 驱动器，以便读取本书中的源代码、Windows 多媒体开发包(MDK)和 Video for Windows 的软件开发包(SDK)。

书中的许多应用程序需要一块 Windows 兼容的声音卡，在第二章中将会讲到，本书中的大部分源程序都是在一块非常普通的声卡上开发的，并且用很多其它硬件板进行了测试，即使是非常低档的声音卡，也没有发现问题。为保证系统的稳定，我建议不要购买低于 75 美元的板子。

专业期刊上经常有一些关于声卡的综述文章，它们经常推荐高质量的系统如 TurtleBeach 声音卡。其中不免有些脱离实际的空谈。如果你相当富有的话，可以购买这些卡，然而并没有太多必要。

如果想显示第五章介绍的“戴廉价太阳镜的沙滩球”图形的话，那么就需要在系统中按装一个游戏棒。“沙滩球”如图 1-3 所示。

游戏棒也象声卡一样，产品档次相差很多，从 Diet Pepsi 到 Dom Perignon 等各种名牌。与声卡不同的是，即使是最高档的游戏棒也不很昂贵。我使用 Gravis 游戏棒来控制沙滩球操作，一开始我先选择了一个便宜的，但是不能使沙滩球移动，然后多花了 10 美元选择了 Gravis 游

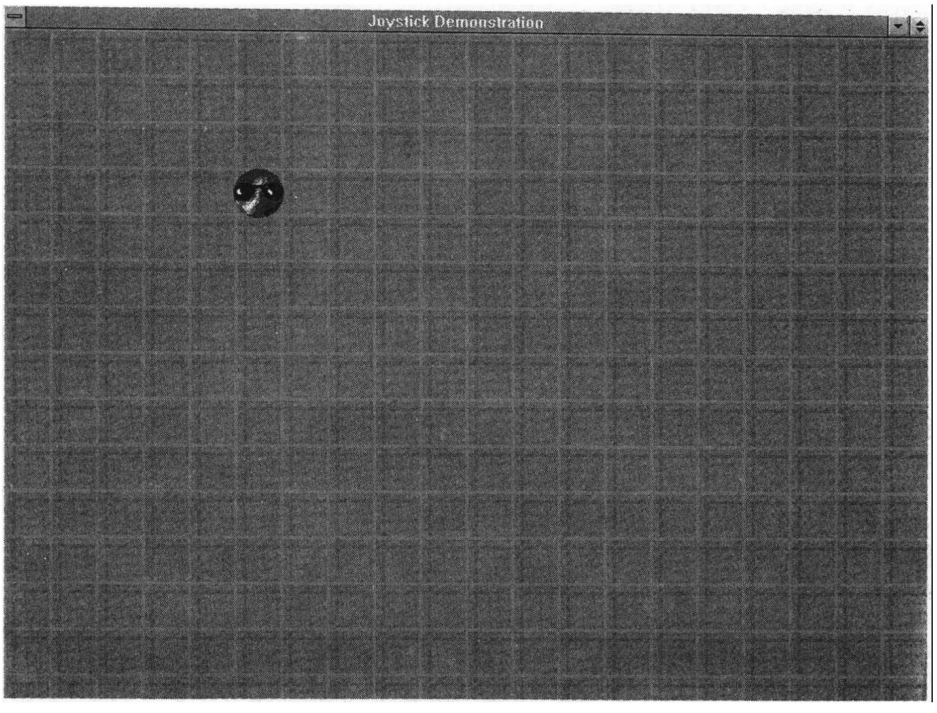

图 1-3　戴廉价太阳镜的沙滩球

戏棒，取得了理想的效果。

1.3　开始工作

插在本书中间的简单实例程序都是用 ANSI C 编写的。C++是传统 C 语言的扩充，C++的编译器内部隐含了 C 语言的编译器，因此可以用 C++编译器去编译 C 源程序，只要简单地认为 C++的扩充没有存在就可以了。

如果已经习惯了用 C++编程而不喜欢用 C，只要将书中的 C 源代码进行一点简单修改就变成了 C++代码。此外，象 Borland C++这样的语言，允许用户自由地混合 C 和 C++语句。可以直接应用书中提供的函数并且加入自己的 C++代码。

C++是很好的结构化语言，它们可以将多个程序员分头完成的工作组织成一个大的工程(project)。然而本书的应用程序并没有很充分地显示 C++语言的特点。

本书将帮助用户访问 Windows 多媒体扩充系统、多媒体开发包(MDK)和 Video for Windows 软件开发包(SDK)提供的实用工具。书中不会教读者如何用 C 语言编写程序，如果没有学过 C 语言程序设计就来学习这本书的话，就如同不会游泳的初学者背着铁锚跳进深水池。学习 C 语言是相当容易的，可以先将此书暂时放在旁边一两周，找一本有关 C 语言的入门教材来阅读。

应用 C 语言编写 Windows 环境下的应用程序比在 DOS 下困难得多，你最好应该有在 DOS 或者 Windows 环境下编程的经验。一个比较容易的办法是，通过修改现有的源程序来掌握程序设计的方法，这样比直接编写程序要简单得多。可以将书中的实例程序稍加修改，从中可以熟悉多媒体的基本操作，直到掌握有关技术。

多媒体是 Windows 应用环境中留下的没有充分开发的最好的一片丛林。丛林中的所有生物都是天然的，所有的工具也都是原始的，包括齿、爪、钩、扒以及野兽等。它们是通向丛林的开路先锋。如果它们不能进入迪斯尼世界的大门，也就没有什么可值得自豪的了。

这本书正是为你徒步进入 Windows 多媒体领域提供了必要的食粮。

第二章　波形文件

在 Windows 下最让人感兴趣而且经常使用的多媒体功能是播放声音文件的工具。波形文件格式将声音数据通过合适的声音卡重新生成为 CD 质量的音频。可以利用波形文件来使计算机发声，产生特技效果；当 Windows 下的系统事件发生时加上声音；当选中 File 菜单中的 Exit 命令时发出语音提示信息。

Windows 多媒体扩展系统的语音操作功能并不是很值得玄耀的。尽管如此，它仍然象 Excel、Word 等许多软件一样，组成了一个完整的软件包。我发现 Windows 就像动画片一样，不断地弹出非常动人的并且配有声音的画面。

《Multimedia Programming for Windows》一书中曾经详细地讨论过，Windows 下的波形文件本身的结构多种多样，而且用于播放波形文件的系统调用的结构也很复杂。这些都是很令人烦恼的事情。

本章包含几个 Windows 下波形文件的应用程序和波形文件的声音。如果对 Windows 多媒体扩展系统不是很熟悉的话，象本书其余章节那样，可以找出与《Multimedia Programming for Windows》中相应章节基本上一样的部分。数字化的声音并不是在任何情况下都有最好的效果，尤其是波形声音文件也会在人们不太注意的地方没有很精细加工。

使用本章讨论的实例应用程序，必须在系统中安装一块声音卡。可以通过 Windows 的扬声器驱动程序有限制地访问波形声音，这个驱动程序只能通过 PC 扬声器来产生质量低劣的数字化声音。扬声器驱动程序不能支持高质量的 MCI 声音接口。这是由于技术上的限制还是出于其它原因，还是不很清楚。

另外，我必须指出，高质量的声音卡仍然相当昂贵，但是也很少有超出预算能力的硬件。我的大多数计算机都配有声音卡的原因是，我买了不少数量的声音卡，当开发应用程序时，希望从中找出一块能与机器一起正常工作。可能是因为该机器所连接的外部设备太多，所以，即使是我试用过的相当好的声音卡在插入机器时也会与硬件有冲突。

这种情况看起来似乎没有希望了，直到有一天在邻近村庄的一个计算机商店里，注意到这里有声音卡的极品，它甚至可以适合于最不兼容的系统。我记得这种卡叫 Zoltrix，它是一种兼容的声霸卡。它虽然不具备最好的声音质量，但是包括两个扬声器和一些软件在内的整个系统价值不到 100 美元。它可以唱歌，而其它昂贵得多的"前辈"却只能发出类似咳嗽和抱怨的声音。在某些情况下，反而是最便宜的声音卡才能很好地工作。

2.1　声音文件资源

对大多数 Windows 用户来说，最熟悉的播放声音的方法是使用 WAV 文件。声音能从磁盘读出并播放，如果要在对话框中加入新的消息发声部分，那么这是一种方便的存储形式。但是如果想在软件中加入一段声音，WAV 文件就不是最理想的方法。在应用程序中过多地使用 WAV 文件，对系统资源的利用来说不是最优的。因为从磁盘上打开和装入离散的波形文件是相当耗时的。

如果将每一段声音作为一个分离的文件来存储的话，也为偶然删除和修改这些文件提供了机会。

除了 WAV 文件之外，声音也可以作为资源列表形式存储在应用程序的 EXE 文件中。这意味着所存储的声音可以通过 LoadResource 调用来装入、使用或放弃。装入一个资源项所需的时间比读入分离的 WAV 磁盘文件所需的时间要短得多。当然删除或修改这些资源对象也是可以做到的。与 Borland C++ for Windows 一起的 Resource Workshop 应用程序为你管理这些资源。

不幸的是，Windows 多媒体扩展系统只能灵活地播放以分离的磁盘文件形式存储的波形文件，而不能同样对待以资源形式存储的波形文件。例如，MCI 接口就是坚持只能处理磁盘文件。可以通过底层的 WaveOut 调用来播放存储于内存中的声音文件，这些在《Multimedia Programming for Windows》中有详细介绍，并在本章后面还会有所接触，但是必须进行数据交换来管理它。WaveOut 调用希望处理原始的采样声音，而存储于资源中的波形文件完全是以 RIFF 块的形式表示的 WAV 文件。

sndPlaySound 函数是设计用来接受存储于缓冲区中的波形文件数据的，就和磁盘上的 WAV 文件一样。它有一些限制，我们马上要讨论到，而且其中有些限制是相当麻烦的。虽然这样，仍然不存在将应用程序资源表中的声音播放出来的简单方法。

sndPlaySound 函数接受两个参数，如下所示：

```
sndPlaySound(LPSTR pointer,WORD flags)
```

参数 pointer 指向一个被播放的磁盘 WAV 文件的文件名或者是指向一个缓冲区，这个缓冲区中包含有 WAV 文件的内容。后者同样支持通过 LoadResource 装入的波形文件的声音。

参数 flags 允许你决定这个函数将怎样处理它的第一个参数以及怎样播放指定的声音。下面是 sndPlaySound 能识别的标志值。注意，可以将多个标志值“或”在一起使用。

SND_SYNC	告诉 sndPlaySound 去播放指定的声音，当播放完声音后返回。
SND_A	SYNC　告诉 sndPlaySound 播放指定声音并立即返回。如果参数 pointer 不是磁盘文件的路经名，那么声音必须锁存在内存中直至结束。
SND_N	ODEFAULT　如果指定的声音不能取出或播放，告诉 sndPlaySound 不播放任何东西。如果没有该标志，当它不能播放所要求的声音时，sndPlaySound 将播放由 WIN. INI 文件所指定的默认声音。
SND_M	EMORY　告诉 sndPlaySound，它的 pointer 参数指向的是存储于内存中的声音，而不是一个文件名。
SND_LOOP	告诉 sndPlaySound 重复播放指定的声音，直到它停止或者断电。
SND_NOSTOP	如果有一个声音正被播放，告诉 sndPlaySound 立即返回。

如果 sndPlaySound 调用成功，返回“true”值，否则返回“false”。如果它的 pointer 参数是 NULL，它将立即停止播放当前的声音。可用 SND_NOSTOP 标志来测试当前是否有声音正在播放。

sndPlaySound 的主要限制在于，当声音结束后没有一种机制来通知应用程序。必须在循环中有一个短暂的等待，通过带有 SND_NOSTOP 标志的 sndPlaySound 调用来检测是否所有的声音都已播放结束，利用 sndPlaySound 调用将多个声音串连在一起是不可能的。采用这种处理声音系统的方法，在 Windows 下很浪费资源。如果采取这样的方法，Windows 将会使它永

远运行下去。

函数 sndPlaySound 主要用于当播放声音在什么时间结束都没有关系的情况下播放单个声音。除了它本身的声音信息外，可以播放内存中的声音，播放结束以后，sndPlaySound 将清除它所播放的声音，释放内部的存储器，并在声音一开始播放就自动地关闭 Windows 的声音驱动程序。

图 2-1 说明了本书中一个应用程序的 About 对话框。它们看起来都很类似。但是图 2-1 不能说明当该对话框打开时，波形文件声音的播放过程。

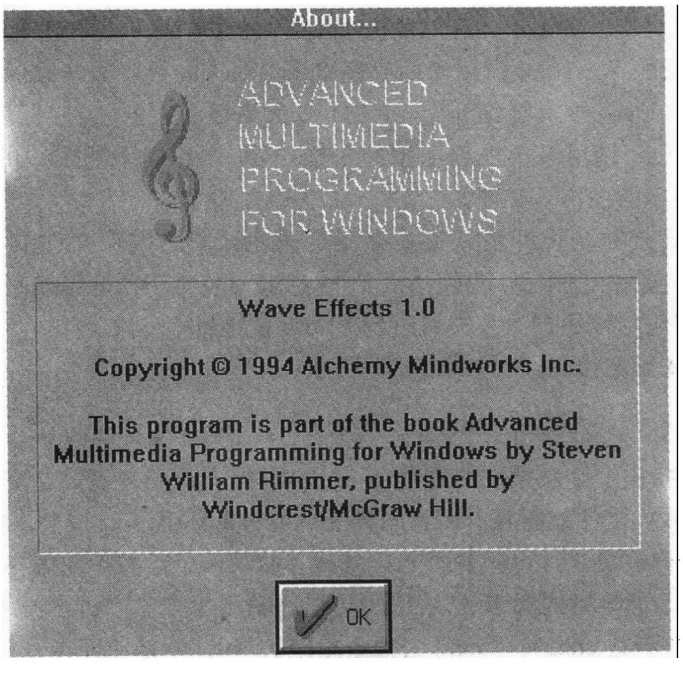

图 2-1　一个 About 对话框

象大多数灵巧的对话框一样，图 2-1 中的 About 对话框完成的大多数工作是由其消息处理程序来执行的。在这种情况下，应用程序的资源描述文件有一点协同作用。

下面是如何在资源描述文件中包含一个波形文件的示例：

```
AboutWave RCDATA ABOUT.WAV
```

它将创建一个名叫 AboutWave 而内容是 ABOUT.WAV 文件的资源列表对象。这个资源列表对象随后可以被函数 LoadResource 调用。

对于存放在资源表中适当的波形文件，应用程序能象下面这样将其装入内存并且播放：

```
HANDLE sound;
LPSTR psound;
HANDLE handle;
if((handle=FindResource(hlnst,"AboutWave",RT_RCDATA))! =NULL){
        if((sound=LoadResource(hlnst,handle))! =NULL){
            if((psound=LockResource(sound))! =MUll){
                    sndPlaySound(psound,SND_ASYNC ¦
                        SND_MEMORY ¦ SND_NOSTOP);
```

```
        }
    }
}
```

这部分代码开始通过 FindResource 在资源中定位一个 handle(句柄)。LoadResource 调用将波形文件装入到内存,如果它在前面的实例程序中已经装入内存,则仅向波形文件返回一个句柄。函数 LockResource 将装入的资源锁定并返回一个指向它的远程指针,该指针在 sndPlaySound 中可用作它的 pointer 参数。

请注意,作为 sndPlaySound 第二个参数的 flags,SND_ASYNC 标志告诉 sndPlaySound 异步地播放 AboutWave 声音,当声音开始播放后就将立即返回。SND_MEMORY 标志告诉 sndPlaySound,它的 pointer 参数指向的是内存中的声音,而不是一个文件名。SND_NOSTOP 标志说明不要中断正在播放的声音,这是在本应用程序中很少用到的方法。

如果计算机中的声音设备是 Windows 扬声器驱动程序,那么 SND_ASYNC 标志将被忽略,因为扬声器驱动程序不能播放异步声音。利用 sndPlaySound 通过 Windows 扬声器驱动程序播放声音,将导致播放所用的系统出现暂时的停止,直到播放完毕。

对于使用真正的声音卡的系统当然不会发生上述情况。

在能够异步播放声音的系统中,记住下面的事情仍然是非常重要的:当前述部分的代码执行完后,可能仍将有一段声音正处于播放之中。Windows 通常运行于保护模式,这就意味着如果应用程序试图访问不属于它的内存——即使该应用程序仅想从中读取数据——将导致一个保护模式错误。比如,在下面情况下就会导致上面所说的情况发生:如果 sndPlaySound 正播放内存中的一段声音,并且在声音结束以前该声音的缓冲区是没有锁定或被释放掉了。

在你调用 UnlockResource 为它的缓冲区解锁以前,你必须确保通过 sndPlaySound 所启动的内存中的声音已经播放完毕。最可靠的办法是,在调用 UnlockResource 以前先调用 sndPlaySound,并且将 pointer 参数置为 NULL。

下面是怎样处理由前述代码所创建的对象的方法:

```
sndPlaySound(NULL,SND_SYNC);
UnlockResource(Sound);
FreeResource(Sound);
```

这里给你提出了一些逻辑上的困难:取决于你如何正确地使用 sndPlaySound。如果这儿讨论的第二段代码紧跟着第一段,那么正在播放的声音会立即结束,一切归于平静。理想的是应该有一种等待机制,至少等到该声音完全结束。但正如前面讨论过的那样,sndPlaySound 并不提供这种功能。

在对话框中,sndPlaySound 被用来产生声音,在开始发声到关闭对话框之间的延迟将取决于用户点中 OK 按钮的时间。这是决定一段声音播放长短的极好的方法。在大多数情况下,声音会正常地结束,但是通过第二次调用 sndPlaySound 来结束播放是一个很好的方法,除非操作人员非常快地点中 OK 按钮。

需要注意,在一个完整的对话框的消息处理程序中,装入和播放声音的代码自然应该在 WM_INITDIALOG 情况下,而结束这个操作的代码则应在 WM_COMMAND 情况下。由于消息处理程序的局部变量仅在一个消息的生存周期有效——作为堆栈中的对象——所以将 sound 和 psound 定义为静态变量是很重要的,以免 Windows 发现它要解锁和释放的是无效的变量。