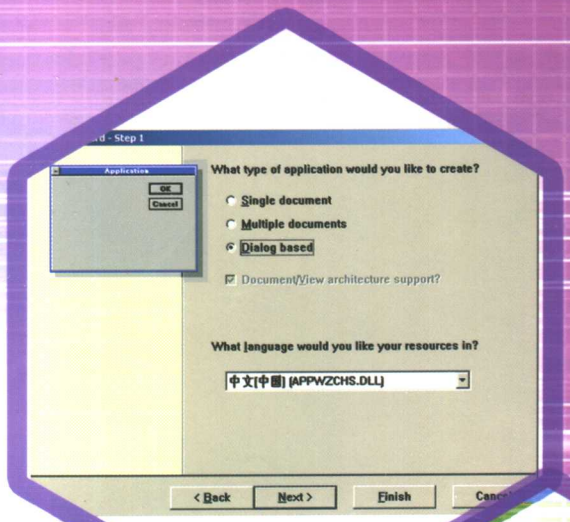
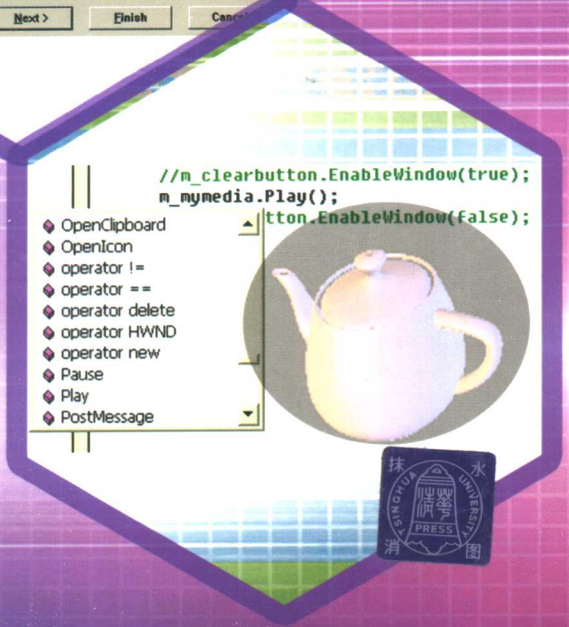
精品风范 倾力奉献

本书配光盘

Windows 多媒体编程基础

张 静 梁 澍 编著





清华大学出版社

Windows 多媒体编程基础

张 静 梁 澍 编著

清华大学出版社
北 京

内 容 简 介

在编写 Windows 多媒体应用程序时，Visual C++ 提供了最为高效、快捷的开发环境；所生成的多媒体程序在运行时具有最为优良的表现。本书介绍了使用 Visual C++ 进行多媒体程序开发的基础知识和实用技巧。

全书共分 10 章，内容包括 Visual C++ 编程基础、进入 Visual C++ 多媒体世界、多媒体文本处理、图形图像初探、深入图形图像编程、多媒体音频、多媒体动画和视频、OpenGL 图像处理简介、利用 DirectX 开发多媒体、综合应用。

配书 CD 光盘中包含了与各章内容密切相关的源代码工程，这些工程是作者致力于实战演练及深入挖掘的结果；衷心希望各位读者能够充分利用此光盘，在编程实践过程中找到快乐的感觉。

本书可供广大编程人员及多媒体开发人员阅读和参考。

图书在版编目(CIP)数据

Windows 多媒体编程基础/张静，梁澍编著.—北京：清华大学出版社，2005.8
ISBN 7-302-11354-8

I.W…　II.①张…②梁…　III.C 语言—程序设计　IV.TP312

中国版本图书馆 CIP 数据核字(2005)第 077543 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969
组稿编辑：张　瑜
文稿编辑：宋延青
封面设计：陈刘源
排版人员：朱　康
印 刷 者：北京市清华园胶印厂
装 订 者：三河市新茂装订有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：24.5　**字数**：578 千字
版　　次：2005 年 8 月第 1 版　2005 年 8 月第1 次印刷
书　　号：ISBN 7-302-11354-8/TP·7474
印　　数：1～4000
定　　价：39.00 元(含 1 张光盘)

前　言

近年来，多媒体技术的发展速度十分惊人。多媒体以它那震撼的音效、细致的画面、逼真的动画等引人入胜的特殊效果吸引了大批的计算机用户和应用软件开发人员。正因为如此，在软件开发中，多媒体的开发占了很大比重。同时，越来越多的计算机用户和从事开发设计工作的技术人员开始加入到多媒体应用程序开发的行列。

对于如何开发 Windows 下的多媒体应用程序，用户有两种选择：一是选择一种多媒体创建系统；二是通过常规的程序语言创建。使用多媒体创建系统虽然比较方便快捷，但功能不够强大，难于向多种应用整合。而程序语言功能强大，特别是 Visual C++, 更是各种编程语言中的佼佼者。任何在 Windows 下所能做的事，都可以用 Visual C++ 来完成。因而 Visual C++ 现在已成为程序员编程的首选工具。

本书从 Visual C++ 的特点、多媒体的基本概念入手，逐步深入地介绍 Visual C++ 的深层次多媒体应用。本书内容安排合理，并结合丰富的程序范例，使读者更容易掌握用 Visual C++ 来开发多媒体应用程序的方法和技巧。

第 1 章介绍 Visual C++ 的界面，它是读者迈向用 Visual C++ 开发多媒体应用程序的第一步。通过几个小例子，给出用 Visual C++ 实现窗口编程的一般过程。

第 2 章从多媒体的定义、历史讲起，向读者讲述多媒体的一些基本知识，并简单介绍开发多媒体程序的重要软件 Visual C++，带领读者步入多媒体编程世界。

第 3 章介绍如何利用 Visual C++ 在 Windows 中显示文本，及如何产生不同的文本显示效果，并介绍一点关于文本控件的用法。

第 4 章和第 5 章介绍如何基于 MFC 基本类库开发 Windows 应用程序，并进行图形绘制。

第 6 章详细讲解多媒体音频。向读者介绍如何在 Windows 的应用程序中实现音响效果，同时还说明如何使用 Windows 的多媒体编程接口 MCI 及低级波形函数，来播放 Windows 的多媒体音频文件。

第 7 章则在第 6 章的基础上讲述如何达到良好的视频显示效果和进行视频播放。

第 8 章使读者了解 OpenGL 的基本概念，能在自己编写的多媒体软件中用 OpenGL 加入 3D 图形处理。

第 9 章讲解利用 DirectX 开发多媒体的基础知识。

第 10 章通过两个大型实例的讲解，引导读者深入学习。

本书的作者长期从事编程工作，在应用 Visual C++ 进行多媒体编程方面有一定的经验，这在书中相关内容里有所体现。作者在编写本书的过程中比较重视追踪程序设计的思路，形成一种特色。

本书在写作、加工、改稿、光盘内容筛选的过程中得到清华大学出版社张瑜先生的大力支持。在此表示诚挚的谢意。

衷心希望本书能对读者在 Windows 多媒体编程方面有所帮助。

编辑寄语

真正的多媒体是活跃在C++ 程序员指尖上的创造力，而不是普通人花上几天时间就可以大致掌握的一两种多媒体编辑软件的使用技巧。

C++ 程序员尽管已经掌握了一流的编程语言，但是距离能够以疾如闪电般的速度呈现静止画面、运动画面、三维场景、极品音效、人机互动等高品质程序要素尚有一步之遥。这就需要程序员以饱满的热情，深入钻研计算机多媒体技术。

目前书市中讲述 Windows 多媒体编程的书籍十分罕见。这种情况恰好反映了我国软件行业国际竞争力落后的局面。环顾世界，美国在计算机及其相关资源方面的投入产出十分巨大，其作为软件枭雄的霸主地位即使再过几十年可能也不会动摇。这就决定了各国都不能放弃 Windows 编程这样的持久商机。回顾往昔，当我们的学术界还在台湾软件业的影响下大力推崇 Pascal 编程，盛赞其语法规范性的时候，世界早已发生了翻天覆地的变化：C/C++ 语言异军突起，迅速以其编程的灵活性和代码的可继承性优势席卷软件行业，成为编写各种大型操作系统和各种应用程序时的主要程序开发语言。

我们使用 C/C++ 语言编写多媒体应用程序，能够从各个角度直击世界软件行业的最本质目标和最基础需求，研究编写各种应用软件时所采用的技术，甚至包括与之紧密相关的计算机硬件接口技术，然后才能顺理成章地融入全球软件市场和硬件市场。

至于 VB、Java、C#、J# 等高级语言，则并不适合于实现多媒体编程研究的目的。其根本原因是，这些语言在实现多媒体(及其他需要速度的应用)目标的时候，几乎完全又是在调用 C/C++ 语言编写的模块。包括 Pascal 程序莫不如此。由此可见，C/C++ 编程是必须掌握的技术。

本书采用 VC 6.0 作为多媒体开发工具。这在 .NET 鼓声正隆的时候仿佛觉得技术略显陈旧。其实这至少对多媒体开发而言是一种假象。须知微软的 Visual Studio 到了 6.0 版本的时候已经十分的成熟。 .NET 目标是试图汇集全世界编程人员的力量编写 Windows 应用程序，以与巨耗资源、在多媒体方面慢得像老牛一样的 Java 程序通过竞争达成平局。所以 .NET 之中的 VC 与 6.0 版本相比几乎没有任何提高，更多地却是要迎合 6.0 版本，以便妥善兼容原有的 C++ 代码。况且 .NET 代码在向其他平台移植时由于其中包含 .NET 数据类型而徒增难度。可见 VC 6.0 作为一种成熟的产品其生命力还将持续很久，全世界真正的程序员们还在普遍地使用它。

本书在 ActiveX 控件、MCI、OpenGL、DirectX 的例子中注重实用性研究和对编程应用本质的挖掘。虽然篇幅甚短，但内容比较独特，很适合略有 C/C++ 语言基础的人员甚至专业领域的编程人员进一步开启 Windows 编程思路。

配书光盘中包含了各章中所引述的例子，都是可以实际编译运行的程序。

本书的主要作者在编写分工上做到了较好的结合，在对编辑反馈信息的处理方面细心、周到、精益求精。正是由于他们的不懈努力，本书方能获得完善并顺利出版。编辑在加工过程中对作者们的才干十分敬仰，并学到了一些东西。预祝各位程序员朋友也能够从书中获益。同时祝愿今后此行业中能够涌现出更多的 C++ Windows 多媒体作者。

光盘使用说明

尊敬的读者：

首先感谢您对本书的热心支持。《Windows 多媒体编程基础》是面向 C++ 学习者的一本编程参考书籍。光盘中提供了书中涉及的各个例子的工程源代码。使用时可将光盘内容拷贝到电脑硬盘上，然后从 Visual C++ 6.0 的菜单栏中选择【文件】|【打开工作区】命令，将弹出【打开工作区】对话框，从中定位到相应章节的工程目录，可以看到 *.dsw 格式的工程文件；打开该文件即可调入该工程所涉及的全部文件。直接单击 ! 按钮运行即可；也可以对程序略作修改，然后重新编译运行(另外，直接双击程序子目录下面的.dsw 文件，即可以打开 Visual C++，在里面可以浏览该程序的源代码并运行程序。或者从部分工程中进入程序子目录下面的 Debug 目录，里面有以程序名命名的*.exe 文件，双击即可运行程序)。

下面以列表形式对光盘中各章实例的具体内容进行总结(总计 52 个工程)。

章	目 录 名	工程内容
1	wm_char	一个最简单的 Win32 应用程序，响应击键动作
	Hello	演示如何建立基于对话框的 MFC 应用程序
	Frame	演示单文档 MFC 工程中各主要文件的作用
2	LoadPic	演示位图加载和缩放显示功能
	PageV	演示 CHtmlView 类的简单使用，用户可以在应用程序中打造自己的浏览器(功能非常强大，看你怎么用)
	ST	演示 CStatic 控件的用法。证明静态不静(可以允满动感)
	MyButtonTest	演示普通按钮的动态创建
	MyBitmapButton	演示位图按钮的动态创建
3	Aaa	演示基本 GDI 绘图、文本输出(此章内容琐碎故有合并)
	StaticText	操纵简单的静态文本控件和文本框控件(同上)
	CFontEnum	用程序来枚举系统中的全部字体
	CFontInfo	如何获得当前字体的基本信息
	RichTextbox	用“Microsoft Rich Textbox Control，version 6.0” ActiveX 控件制作文本编辑器
	MyWord	用 CRichEditView 做基类的文本编辑器(可插图)
4	Version1	演示 ActiveX 控件包装。与书中的例子是一致的。由于这种例子很少，所以书中尽量作出了较为详细的描述
	Version2	比 Version1 简单一点的东西，另外一个普通应用程序的 ActiveX 控件包装。目的是证明包装方法行得通
	Version3	更复杂一点的，其中有 COM 自动化代码，其实还是 Version1 的翻版而已

续表

章	目 录 名	工程内容
5	SimpleBrowse	第 5 章实际只有一个 SimpleBrowse 工程(简单的图片浏览器)。其他目录都是编译时所需要的库和包含文件等(BIN 是调试输出)。该工程较好地演示了一个工程中各要素之间的关系。至少可帮助理解.h/.lib/.dll 文件各自扮演的角色
6	CDPlay	用 mciSendCommand 制作简单的 Wav 声音播放器
	MIDIPlay	照上一个例子略有改进，可以播放*.mid 和*.wav 文件
	Play	主要展示 CCDAudio 包装类的使用
	LLS	在 MyWave1 包装类中演示了 Windows 低级音频编程
7	BitmapCartoon	动态显示时间的时钟，用 WM_TIMER 消息控制位图更换
	MYMMC	使用 Multimedia MCI 控件制作的 AVI 播放器
	Videoplay(aviply)	视频播放程序。用 aviVideoFileOpen 核心函数。注意该函数中运用了 mciSendCommand 和 mciSendString 混合编程
	MMDlg_version1	用 myPlayAvi 核心函数制作播放器。纯 mciSendString 方式
	MMDlg_version2	上面 aviply 播放器的 MFC 版本(注意上述三个例子都是在借助于 window 命令在应用程序主窗口中播放)
	W2	演示使用 Windows Media Player 控件(Windows 自带的 ActiveX)制作电影播放器，兼容大多数影片格式
8	MYGL	演示单文档 MFC 工程的 OpenGL 初始化操作(画点和线)
	Win32GL	演示 Win32 应用程序的 OpenGL 初始化操作(画点和线)
	REDUCED_V1	典型的 AUX 窗口程序，可以产生不错的动画
	REDUCED_V2	把 AUX 的复杂操作剥离掉了，然后对“变换”进行深入剖析，得出一些特殊结论，总之还很好用，读者应试一下
	Triangle	8.4.1 节“用 OpenGL 绘制三角形”的结果工程。其中包含极为简单的 OpenGL 环境搭建。是个 Console 工程
	Triangle_version_teapot	8.4.2 节“用 OpenGL 绘制茶壶”的结果工程。在上一个例子架构的基础上，加入灯光等，渲染出一个漂亮的旋转茶壶
	MFCVERSION	上面的茶壶在单文档应用程序中做是个什么样子？
	DLGVERSION	把旋转的茶壶放置在 MFC 程序的对话框中了。使用了一个小技巧，成了控件的模样！MFC 与 OpenGL 的完美结合
	DLGVERSION_2	删除了 Timer，使用 CWinApp 类的 OnIdle 函数建立动画
9	MyDirectX	极为简单的 DirectX 环境设置演示。将全部 DirectDraw 代码灌注在 button1 的处理函数中。无须使用任何 SDK
	MyDirectX1	对上面的小动物稍加改进，用 WM_TIMER 消息控制，使之活动起来。FULLSCREEN 方式
	MyDirectX1_Windowed	把上一个例子改造成非全屏的窗口程序。很有效的尝试

续表

章	目 录 名	工程内容
9	SoundOnly	向读者演示 DirectSound 的独立性。是一个基于对话框的 MFC 工程，完全不涉及其他 DirectX 部件。注意 DirectSound 具有混音能力，可以连续打开并播放多个*.wav 文件
	MyDirectX2	在这个工程中，我把 DirectDaw 与 DirectSouns 混杂在一起了，并且增加了两个测试声音的按钮，用于演示对音量和播放速度的控制。不是很完善，你可以试着改进一下
	Additional	DirectDraw + VFW 编程演示
10	坦克大战	看一个大型项目是如何组织的。3 个可编译子项目位于 src 目录中，对应了 3 个工程
	音频编辑	比较大型的实用软件源代码

目　录

第 1 章　Visual C++ 编程基础 1

1.1 Win32 基础 1
1.1.1 Windows 基础 1
1.1.2 窗口过程、事件和消息 2
1.1.3 数据类型 3
1.1.4 WinMain()函数 3
1.1.5 一个最简单的 Win32 程序 4
1.2 Visual C++ 开发环境 8
1.2.1 Visual C++ 6.0 开发环境介绍 8
1.2.2 Visual C++ 6.0 的帮助系统——MSDN 环境 10
1.2.3 建立一个工程 10
1.3 用 Visual C++ 开发 Windows 应用程序 16
1.3.1 MFC 简介 16
1.3.2 MFC 消息处理机制 16
1.3.3 一个框架性的 MFC 单文档应用程序 18

第 2 章　进入 Visual C++ 多媒体世界 38

2.1 多媒体基础 38
2.1.1 多媒体的定义 38
2.1.2 多媒体的发展 38
2.1.3 多媒体的应用 39
2.1.4 多媒体的前景 41
2.2 Windows 的多媒体组成 42
2.2.1 文本 42
2.2.2 静态图形 43
2.2.3 动画 46
2.2.4 音频 47
2.2.5 视频 47
2.2.6 超文本 47
2.3 多媒体程序设计 50
2.3.1 如何用 Visual C++ 开发多媒体 50
2.3.2 Windows API 50
2.3.3 使用控件 51
2.4 ActiveX 技术简介 55
2.4.1 ActiveX 控件的基本知识 55
2.4.2 在 Visual C++ 中使用 ActiveX 控件 56

第 3 章　多媒体文本处理 60

3.1 设备上下文与文本输出 60
3.1.1 什么是设备上下文 60
3.1.2 CDC 类及其派生类 61
3.1.3 文本显示函数 69
3.2 使用字体 73
3.2.1 字体描述 73
3.2.2 创建各种各样的字体 77
3.2.3 字体的选择 80
3.3 文本控件的使用 83
3.3.1 使用静态文本控件 84
3.3.2 使用编辑框控件 85
3.3.3 使用 RichEdit 控件 86
3.3.4 使用 ActiveX 控件 86
3.4 实例：文本编辑器 88

第 4 章　图形图像初探 92

4.1 Windows 绘图基础 92
4.1.1 Windows 坐标系统 92
4.1.2 Windows 中的颜色 92
4.1.3 画笔和画刷 93
4.2 基本图形的绘制 93
4.2.1 基本绘图函数 93
4.2.2 使用画笔 95
4.2.3 使用画刷 97
4.3 Windows 中的位图 99

4.3.1 位图结构 99
4.3.2 CBitmap 类 101
4.4 对位图的操作 101
4.5 图标 102
4.5.1 图标结构 103
4.5.2 图标的操作函数 103
4.6 实例：简单的绘图程序 103
4.6.1 创建 Demo1 画笔应用程序 104
4.6.2 转化成控件 106
4.6.3 在其他应用程序中使用控件 113

第 5 章 深入图形图像编程 114

5.1 图形图像显示 114
5.1.1 Windows 和调色板 114
5.1.2 使用颜色的三种方法 116
5.1.3 调色板的创建和设置 117
5.1.4 DDB 和 DIB 的使用 118
5.2 图像操作技巧 120
5.3 常用图像格式 122
5.3.1 JPEG 图像格式 122
5.3.2 JPEG 图像操作函数 126
5.4 实例：图像浏览器 129

第 6 章 多媒体音频 136

6.1 数字音频基础 136
6.1.1 模拟音频和数字音频 136
6.1.2 数字音频的采样和量化 137
6.1.3 数字音频的文件格式 137
6.1.4 数字音频的应用 138
6.2 多媒体控制接口 MCI 138
6.2.1 MCI 简介 138
6.2.2 MCI 命令系统 139
6.2.3 MCI 的使用 142
6.3 MIDI 音乐合成技术 144
6.4 多媒体文件 I/O 与低级波形音频函数 145
6.5 实例 1：MIDI/WAV 播放器 150
6.6 实例 2：CD 播放器 154

第 7 章 多媒体动画和视频 167

7.1 使用 GDI 绘制动画 167
7.1.1 GDI 基础 167
7.1.2 定时器 168
7.1.3 位图动画 169
7.1.4 图标光标动画 171
7.2 使用 MCIWnd 控件播放视频 171
7.2.1 MCIWnd 控件 171
7.2.2 播放动画示例 173
7.3 其他视频控件 174
7.3.1 CAnimateCtrl 控件 174
7.3.2 Multimedia MCI 控件 175
7.4 用 MCI 函数播放视频 179
7.4.1 MCI 概述 180
7.4.2 MCI 命令接口的使用 180
7.4.3 MCI 字符串接口的使用 181
7.4.4 接口的选择 183
7.4.5 处理 MCI 通知 183
7.4.6 如何用 MCI 播放 AVI 文件 185
7.5 实例 1：视频播放器 1 190
7.6 实例 2：视频播放器 2 193
7.7 实例 3：视频播放器 3 194

第 8 章 OpenGL 图像处理简介 198

8.1 OpenGL 概述 198
8.1.1 OpenGL 简介 198
8.1.2 OpenGL 的基本组成 199
8.1.3 OpenGL 的主要功能 200
8.2 OpenGL 的基本操作 201
8.2.1 各种变换命令 201
8.2.2 使用颜色 202
8.2.3 光照 204
8.2.4 材质 207
8.2.5 位图和图像 208
8.2.6 纹理 211
8.2.7 几何要素与操作 214

8.2.8 帧缓存和动画.....216
8.2.9 显示列表.....217
8.3 在 Visual C++中使用 OpenGL.....219
8.3.1 MFC 单文档 OpenGL 应用程序.....219
8.3.2 Win32 OpenGL 应用程序.....222
8.3.3 GLUT 窗口程序.....226
8.3.4 AUX 窗口程序.....228
8.3.5 精简 AUX 窗口程序.....232
8.4 综合实例：利用 OpenGL 制作三维场景.....234
8.4.1 用 OpenGL 绘制三角形.....235
8.4.2 用 OpenGL 绘制茶壶.....239
8.4.3 将代码移植到 MFC 应用程序中.....243
第 9 章 利用 DirectX 开发多媒体.....244
9.1 DirectX 概述.....244
9.1.1 什么是 DirectX.....244
9.1.2 DirectX 的基本结构.....246
9.1.3 DirectX 的接口.....247
9.1.4 在 Visual C++中使用 DirectX.....247
9.2 DirectDraw.....248
9.2.1 DirectDraw 对象.....249
9.2.2 使用 DirectDraw 编程.....250
9.3 DirectSound.....255
9.3.1 DirectSound 对象.....255
9.3.2 DirectSound 对象初始化.....256
9.3.3 对 DirectSound 操作.....257
9.3.4 使用 DirectSound 编程.....259
9.4 借用 DirectDraw 表面播放低级视频.....261
第 10 章 综合应用.....264
10.1 游戏——坦克大战.....264
10.1.1 片头动画.....265
10.1.2 游戏源代码剖析.....265
10.1.3 地图编辑器.....319
10.2 音频编辑.....320
10.2.1 建立工程.....320
10.2.2 源代码中涉及的四个结构.....323
10.2.3 构建三个类.....323
10.2.4 将新增代码加入工程.....372
10.2.5 程序运行演示.....373

第1章 Visual C++ 编程基础

Visual C++ 6.0 是 Microsoft 公司在多年使用不断改进的基础上推出的，它用于 Win32 平台应用程序(application)、服务(service)和控件(control)的开发。

在这一章中，读者将学会如何同 Windows 打交道，以及如何建立 Windows 程序框架。然后读者将初步了解 MFC 的作用和用途，以及 MFC 的基本原理，并对 MFC 为什么是一项如此有用的技术有一个感性的认识；最后，我们将通过创建一个框架性的 MFC 单文档应用程序，来加深对 MFC 的理解。

1.1 Win32 基础

在真正进入 MFC 并了解它究竟是什么之前，建立起关于 MFC 的目标操作环境是很重要的。之所以设计 MFC，就是为了构造在 Win32 平台上运行的应用程序，这些平台主要包括 Windows 95、Windows 98、Windows NT、Windows 2000 和 Windows XP，以及 Microsoft 公司推出的新一代 64 位操作系统 Windows 2003。虽然这几个操作系统在某些方面互不相同，但它们都使用了 Win32 平台的通用技术。

Win32 的重大意义在于它建立了 Windows 应用程序能够获得的一些特性和性能。它的设计完全是为了获得用户所期望的简明的图形特性和性能，而不涉及屏幕后面的具体事物。而对于一个程序开发者来说，之所以必须了解 Win32 是因为微软在构建 MFC 时，完全基于 Win32 API，且 Win32 API 是开发 Win32 应用程序的标准编程接口。

在 MFC 产生以前，所有 Windows 程序都是以 Win32 API 用 C 语言进行开发的，这样在编写应用程序时非常麻烦，连一个简单的小程序都需要编写上百行代码。Windows API 编程的复杂性促进了 MFC 的产生。但是使用 MFC 编程仍然离不开 Win32 API，这是因为：①MFC 类库并未囊括程序开发所需要的全部功能，可以运用 Win32 API 自建扩展类库；②在建立特定程序的过程中，经常会发现直接调用 Win32 API 可以获得更高的运行效率；③因为 MFC 实质上是借助于 C++ 技术对 Win32 API 进行的包装，没有 Win32 API 的基础支持，也就根本不会有 MFC。

1.1.1 Windows 基础

在 Win32 中，一切都开始于对窗口的理解。窗口被简单地定义为一个能够接收信息和显示输出信息的矩形区域。Win32 定义了许多种类型的窗口，从应用程序界面到按钮再到滚动条都属于窗口的范畴。

一般意义上的窗口分为两个部分：客户区和非客户区。客户区是应用程序能够进行控制的区域。而窗口的其他区域则是非客户区，它的大部分是由 Win32 系统进行操作和控制的。如图 1.1 所示。

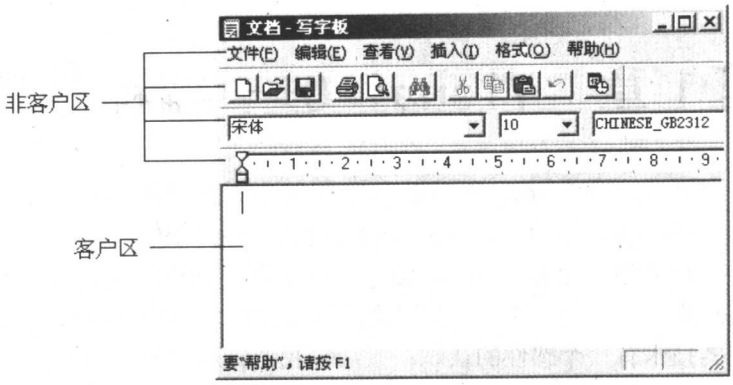


图 1.1 基本 Win32 应用程序窗口

Win32 API 包含了范围广泛的窗口操作函数。同时，它也定义了许多不同种类的窗口类(代码参见后文)。当然 Win32 中定义的窗口类绝非 C++ 之中的“类(class)”概念，而是一个窗口属性模板，实质上是基于结构(struct)的，程序使用它来创建窗口(Microsoft 在使用 C++ 构建 MFC 之前，一直在使用 C 语言构筑 Win32 程序，即“窗口类”的说法诞生于大量运用 C++ 之前)。窗口类还包含一个指向窗口过程的指针，它决定了窗口的功能。

1.1.2 窗口过程、事件和消息

Win32 窗口的核心是窗口过程，窗口过程是一个包含驱动窗口行为所有代码的特殊函数，是一个由 Windows 调用而非由程序调用的函数。所有的 Windows 程序必须包括一些特殊的函数，它们不由用户的程序调用，而是由 Windows 调用。这样的函数通常称为窗口过程或窗口函数。当 Windows 需要向程序中传递一条消息时，将调用窗口函数。Windows 正是通过此函数与用户的程序打交道。窗口函数在它的参数内接收消息。所有窗口函数的返回类型必须被说明为 LRESULT CALLBACK。LRESULT 是一种类型定义(typdef)，它是长整型数据的另一个名字。CALLBACK 表示按照约定函数将被 Windows 调用。在 Windows 中，被 Windows 调用的任何函数都被称作“回调函数”。

除了接收 Windows 发送的消息外，窗口函数还必须启动消息所指示的操作，它包含一个巨大的 switch 语句，该语句链接了程序将对每条消息的具体响应。用户的程序不必响应 Windows 所发出的全部消息。对于那些用户程序所不关心的消息，可让 Windows 提供默认处理。

Windows 有大量的消息。所有的消息都是 32 位整型值，而且，所有的消息都与消息所需要的任何传统信息链接在一起。每条消息由一个惟一的 32 位整型值来表示，且每条消息相当于某个事件。这些消息都有标准宏名，通常涉及消息时，我们都使用宏名，而不使用明确的整型值。常用的 Windows 消息宏如表 1.1 所示。

每条消息都伴随着两个其他值，可从这两个值获得与具体消息相关的消息。其中一个值是 WPARAM 类型，另一个是 LPARAM 类型。对于 Windows 而言，这两个类型都被转换成 32 位整型数。通常将其分别称做 wParam 和 lParam。通常，它们保存鼠标器光标的坐标值、按键值或者与系统相关的值。

实际处理消息的函数是程序的窗口函数，此函数被传递 4 个参数：消息所服务的窗口的句柄、消息本身和最后的两个参数 wParam 和 lParam。

表 1.1　常用的 Windows 消息宏

宏 名 称	含　义
WM_CHAR	键盘上有键被按下
WM_PAINT	窗口重绘
WM_MOVE	窗口位置有变化
WM_CLOSE	窗口关闭
WM_LBUTTONDOWN	鼠标左键按下
WM_LBUTTONUP	鼠标左键抬起
WM_COMMAND	菜单被点击
WM_SIZE	窗口尺寸发生变化

有时将消息译成两个包含 wParam 和 lParam 参数的字。为了便于访问 wParam 和 lParam，Windows 定义了两个称做 LOWORD 和 HIWORD 的宏，它们分别返回一个长整型数据的低位和高位字。用法如下：

```
x  =  LOWORD(lParam);
y  =  HIWORD(lParam);
```

不久读者就会看到这些宏的用法。

如前所述，Windows 通过向程序中发送消息来与用户的程序进行通信。所有的 Windows 应用程序必须在 WinMain()函数内建立一个消息循环。此循环从应用程序的消息序列中读取任何未处理的消息，再将其送还 Windows，然后以该消息作为参数来调用程序的窗口函数。这是所有 Windows 程序在运行过程中所遵循的方法。

1.1.3　数据类型

请注意，Windows 程序并不广泛使用标准 C/C++ 数据类型，如 int 和 char 类型。Windows 所使用的数据类型已经在 windows.h 文件或其相关文件中被定义了。此文件被 Microsoft 公司所支持。而且必须包含在所有的 Windows 程序中。一些最常见的类型是 HANDLE、HWND、UINT、BYTE、WPRD、DWORD、LONG、BOOL、LPSTR 和 LPCSTR。HANDLE 是用作句柄的 32 位整型数，句柄有很多类型，但是它们的大小与 HANDLE 相同，简单地说，句柄是一个标识资源的值。

除上述的基本类型外，Windows 也定义了一些结构体，包括 Win32 框架程序所需要的两种结构体：MSG 和 WNDCLASSEX。

这些由 Microsoft 自定义的数据类型很显然是为了通过大写字母字体来强调程序中的类型成分，从而使 Windows 程序美观一致，对于提高程序的可读性产生了很大的帮助。

1.1.4　WinMain()函数

所有的 Windows 程序都从调用 WinMain()函数开始，即以其作为程序的入口点。

Windows 所使用的调用 WinMain()函数的调用约定是 WINAPI。WinMain()函数的返回值是 int。并且 WinMain()函数仅仅做一些创建和初始化的工作后就退出了。在 WinMain()函数创建了一个主窗口以后，由应用程序的其他部分执行窗口过程响应消息(事件)。这是过程驱动与事件驱动程序设计之间的主要区别：过程式的程序通过对操作系统的调用来接收输入，而事件驱动程序则运用回调函数来处理源于操作系统的消息输入。

这里顺便指出：MFC 把 WinMain()函数隐藏了起来(不等于没有)，因此在 MFC 应用程序中不需要考虑 WinMain()函数。这样就给程序编写带来了巨大的便利，免去了对 WinMain()函数里面大多数千篇一律的代码的书写和分析工作(窗口类参数的设置功能在 MFC 中通过框架类 CMainFrame 来完成)。

1.1.5 一个最简单的 Win32 程序

上面我们已经对 Windows 程序基础做了一个简单的介绍，现在来具体地研究一个简单的 Windows 程序。我们通过编写一个响应按键操作的程序来讲解 Windows 程序是如何接收并处理不同的消息的。

最常见的 Windows 消息是在按下一个键时产生的。此消息称作 WM_CHAR。理解下面一点是很重要的，即用户的应用程序从未接收到过直接来自键盘的按键操作。事实上，每次按一个键时，操作系统就发出 WM_CHAR 消息，并调用窗口函数，向该函数传递几个参数以反映按键操作的具体状况：HWND 类型的 hWnd 参数用于传递当前活动窗口的句柄，将 WM_CHAR 消息当作 UINT 类型的参数；而 wParam 参数中则包含所按键的 ASCII 码值；此外还有 lParam 参数，通过宏的解析，LOWORD(lParam)中包含由于持续按键所导致的重复按键次数，HIWORD(lParam)的各个位的含义如表 1.2 所示。

表 1.2 lParam 如何为键盘数据编码

位	意 义
15	如果键正被释放则置位；如果键正被按下则清零
14	如果在发送消息前按键则置位；否则清零
13	Alt 键被按下时置位；Alt 没被按下则清零
12	由 Windows 使用
11	由 Windows 使用
10	由 Windows 使用
9	由 Windows 使用
8	如果按下的键是有增强键盘所提供的扩充键则置位，否则清零
7～0	依赖于生产厂家的键盘代码(即扫描代码)

对于我们来说，此时惟一重要的值可能是 wParam，因为它保存着所按键的信息，当然对于 Windows 提供的其他有关系统状态的详细信息，用户也可以根据自己的喜好或实际需要选择使用。

为处理 WM_CHAR 消息，必须将其添加到程序窗口函数内的 switch 语句中。例如，下面这个窗口函数所做的处理就是简单地将按键内容显示在屏幕上。

```
TCHAR str[255] = "";      // 保存输出的字符串
LRESULT CALLBACK WindowsFunc(HWND hWnd, UINT message,
                             WPARAM wParam, LPARAM lParam)
{
    HDC hdc;
    switch(message)
    {
        case WM_CHAR:    // 按键消息处理
            hdc = GetDC(hWnd);    // 获得设备上下文
            TextOut(hdc,1,1,"  ",3);      // 擦除原有字符
            sprintf(str,"%c",(char)wParam);    // 把字符转换为字符串
            TextOut(hdc,1,1,str,strlen(str));   //输出字符
            ReleaseDC(hWnd,hdc);   // 释放设备描述表
            break;
        case WM_DESTROY:   // 终止应用程序
            PostQuitMessage(0);
            break;
        default:
            return DefWindowProc(hWnd, message, wParam, lParam);
  }
  return 0;
}
```

让我们简略分析一下这些代码的作用。首先必须在用户的程序和屏幕之间建立一个链路，即设备描述表(简称 DC，又称设备上下文、设备环境)，通过调用 GetDC()来获得它，一旦获得一个设备描述表，就可以向屏幕上输出。在此进程的最后，通过使用 ReleaseDC()来将设备描述表释放。GetDC()和 ReleaseDC()都是 Win32 API 函数，它们的原型如下：

```
HDC  GetDC(HWND  hWnd);
int  ReleaseDC(HWND hWnd,HDC hdc);
```

GetDC()返回一个与窗口有关的设备描述表，且该窗口的句柄是由 hWnd 指定的。HDC 类型指定一个设备描述表的句柄。如果不能获得设备描述表，则返回 NULL。

如果设备描述表成功释放，则 ReleaseDC()返回 true，否则返回 false。hWnd 参数是窗口的句柄，为此窗口而释放设备描述表。hdc 参数是通过调用 GetDC()而获得的设备描述表的句柄。

实际输出字符的函数是 Win32 API 函数 TextOut()，其原型如下：

```
BOOL  TextOut(HDC  DC,int x,int y,LPCSTR  lpstr,int length);
```

TextOut()函数在由 x，y 所指定的窗口坐标处输出由 lpstr 所指定的字符串，字符串的长度由 length 指定。如果操作成功，则 TextOut()函数返回非零值，否则返回零。

在上面的窗口函数例子中，每次接收到 WM_CHAR 消息时，通过使用 sprintf()函数，将用户所键入的字符转换成长度为一个字符的字符串，之后通过使用 TextOut()函数在位置(1，1)处显示该字符串。在窗口中，客户区域左上角的位置是(0，0)。窗口坐标往往与窗口有关，而与屏幕无关。因此，当输入字符时，不管窗口实际位于屏幕何处，字符都显示在窗口的左上角。

在上述程序中调用第一个 TextOut()函数的原因是为了用空白输出来擦拭以前显示的

字符。因为 Windows 系统是基于图形的系统，其字符长短不同，所以一个字符覆盖另一个字符并不能完全擦除前面的字符。而用几个空格就可以把问题解决了。

下面是处理 WM_CHAR 消息的一个完整程序。读者只需在 VC 6.0 中创建一个 Win32 Application 工程，例如以 wm_char 为工程名称，然后在 Step 1 of 1 对话框中选中 An empty project 单选按钮，即可创建一个不包含任何文件的工程；然后选择 File | New 菜单命令，在 New 对话框的列表中选择 C++ Source File，并在 File 文本框中键入 wm_char 作为新文件名称，然后将下面的全部代码放入所生成的空文件中编译即可。如图 1.2 所示是此程序在运行时所创建的窗口。

```
#include <windows.h>
#include <string.h>
#include <stdio.h>

LRESULT CALLBACK WindowsFunc(HWND hWnd, UINT message,
                             WPARAM wParam, LPARAM lParam);
TCHAR szWinName[] = "MyWin";   // windows 类名
TCHAR str[255] = "";     // 保存输出的字符串
int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                    LPSTR     lpCmdLine,
                     int      nCmdShow)
{
    HWND hWnd;
    MSG msg;
    WNDCLASSEX wcl;
    //  定义 一个 windows 类
    wcl.cbSize = sizeof(WNDCLASSEX);
    wcl.hInstance = hInstance;
    wcl.lpfnWndProc = WindowsFunc;
    wcl.lpszClassName = szWinName;
    wcl.style = 0;    // 默认风格
    wcl.hIcon = LoadIcon(NULL,IDI_APPLICATION);  //  标准图标
    wcl.hIconSm = LoadIcon(NULL,IDI_WINLOGO);  //  最小化图标
    wcl.hCursor = LoadCursor(NULL,IDC_ARROW);  //  光标类型
    wcl.lpszMenuName = NULL;   // 没有菜单
    wcl.cbClsExtra = 0;
    wcl.cbWndExtra = 0;
    //  设置窗体的背景色为白色
    wcl.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
    // 注册这个窗体
    if(!RegisterClassEx(&wcl))      return 0;
    hWnd = CreateWindow(
        szWinName,   // 窗体名称
        " Processing WM_CHAR Messages", // 标题
        WS_OVERLAPPEDWINDOW, // 窗体风格
        CW_USEDEFAULT,  // X 坐标
        CW_USEDEFAULT,  // Y 坐标
        CW_USEDEFAULT,  // 宽度
        CW_USEDEFAULT,  // 高度
```