Windows 程序设计（第3版）

张铮 孙宝山 周天立 编著

- 教你从最简单“Hello World”写到复杂的 Windows 下的防火墙程序
- 阐明用户模式下 Win32 程序的运行原理，介绍直接用 Win32 API 开发应用程序的相关技术
- 学习如何设计、实现框架和类。自制一个具体的 MFC，剖析支持 MFC 工作的关键技术的内部实现
- 详细演示 C++ 语言中虚函数、静态函数、继承和类模板等高级特性的具体应用
- 详细讲解 DLL 注入技术、远程进程技术、HOOK API 技术及内核模式程序设计
- 介绍使用 OpenGL、OpenAL 库的计算机 3D 图形和音频控制技术，帮助读者开发出更加丰富多彩的应用程序
- 书中包括 70 多个完整实例，为读者提供了丰富的参考

人民邮电出版社
POSTS & TELECOM PRESS

Windows 程序设计（第3版）

张铮 孙宝山 周天立 编著

人民邮电出版社
北京

图书在版编目（CIP）数据

Windows程序设计 / 张铮，孙宝山，周天立编著. -- 3版. -- 北京 : 人民邮电出版社，2015.4
ISBN 978-7-115-38162-0

Ⅰ. ①W… Ⅱ. ①张… ②孙… ③周… Ⅲ. ①Windows操作系统－程序设计 Ⅳ. ①TP316.7

中国版本图书馆CIP数据核字(2015)第030035号

内 容 提 要

Windows API 编程是最基本的编程方式，任何用户应用程序都必须运行在 API 函数之上。学习 Windows 程序设计最好先从学习 API 函数开始。同时 MFC 类库是最流行的编程工具之一，大部分商业软件使用了 MFC 框架程序。精通 MFC 是很多开发人员的目标。

本书试图为 Windows 程序设计初学者提供一条由入门到深入、由简单到复杂的编程设计之路，最终使他们有能力独立开发出像 Windows 防火墙一样复杂的应用程序。为此，本书首先介绍了 Win32 程序运行原理和最基本的 Win32 API 编程；然后通过模拟 MFC 中关键类、全局函数和宏定义的实现详细讲述了框架程序的设计方法和 MFC 的内部工作机制，并指出了这些机制是如何对用户程序造成影响的；继而完整讲述了开发内核驱动和 Windows 防火墙的过程；最后对计算机 3D 图形和音频控制技术进行了介绍。此外，书中各章均配以丰富的实例，它们从最简单的“Hello，World!”开始，再到多线程、用户界面、注册表和网络通信、3D 图形绘制等复杂的程序，内容涉及 Windows 编程设计的方方面面。

全书语言严谨流畅，针对初学者的特点，精心策划、由浅入深，是学习 Windows 编程由入门到深入的理想参考书。凡是具备 C++初步知识的读者都能读懂本书。本书可作为研究 Windows 程序设计的正式教程，也是一本供自学者从入门到深入学习 Windows 程序设计的很有帮助的参考教材。

◆ 编　　著　张　铮　孙宝山　周天立
责任编辑　张　涛
责任印制　张佳莹　焦志炜

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京中新伟业印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：30.5
字数：762 千字　　2015 年 4 月第 3 版
印数：14 501 – 18 000 册　　2015 年 4 月北京第 1 次印刷

定价：69.00 元（附光盘）

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315

前　言

许多人在刚开始接触 Windows 编程时，或从 VB 开始，或从 MFC 开始，这使得大家虽然写出了程序，但自己都不知道程序是如何运行的，从而造成写程序“容易”修改难、设计程序“容易”维护难的状况。本书是为 Windows 程序设计入门的初学者和想从根本上提高自己编程水平的爱好者编写的，试图为他们提供一条由入门到深入、由简单到复杂的编程设计之路。

API 函数是 Windows 系统提供给应用程序的编程接口，任何用户应用程序必须运行在 API 函数之上。直接使用 API 编程是了解操作系统运行细节的最佳方式，而且熟知 API 函数也是对程序开发者的一个最基本的要求。本书将以 API 函数作为起点介绍 Windows 编程，这样做的好处是使读者撇开 C++的特性专心熟悉 Win32 编程思路和消息驱动机制。

但是，在开发大型系统的时候，我们往往并不完全直接使用 API 函数，而是使用 MFC 类库框架程序。MFC 对 90%以上的 API 函数进行了面向对象化包装，完全体现了对象化程序设计的特点，是时下最流行的一个类库。

当读者熟悉最基本的 API 函数编程以后，就可以学习更高级的 MFC 编程了。虽然 MFC 仅仅是对 API 函数的简单封装，但由于读者对 C++语言的了解不够，不清楚框架程序的工作机制，即便是有经验的程序员在 MFC 复杂的结构面前也显得非常困惑。他们会“用”MFC，却不知道为什么这么“用”，在写的程序出错时这种现象带来的问题就很明显了，他们不会改。

这种只会“用”的知识层次不能够达到现实的要求，因为在面对一个大的项目的时候，代码往往需要手工添加和修改，而很少能够依靠 VC++的向导。为此，本书将从开发者的角度同读者一起来设计 MFC 中的类、函数和宏定义。通过对 MFC 类库的分析和了解，读者不仅能够更好地使用 MFC 类库，同时，对于自己设计和实现框架和类，无疑也有相当大的帮助。

本书后面讲述了 Windows 系统编程中当前最为热门的话题——DLL 注入技术、远程进程技术、HOOK API 技术等，并配有完整而具体的实例。

本书还讨论了 Windows 内核驱动程序设计和防火墙开发。这对于全面了解 Windows 操作系统的结构体系，学习独立开发应用软件是非常有帮助的。

最后，本书对当前流行的计算机 3D 图形绘制和音频控制进行了介绍，讲述了 OpenGL、OpenAL 开发库的相关用法，更加丰富了读者可开发的应用程序的内容。

——内容安排

本书试图从“Hello，World”这个简单的例子出发，通过 70 多个实例，由浅入深地讲述 Win32 API 程序设计、类库框架设计、MFC 程序设计、内核模式程序设计等，使读者在实践中熟练掌握 Windows 程序设计模式，并有能力写出特定功能的用户应用程序和简单的内核驱动程序。

在编程论坛上，笔者发现许多初学者搞不清楚 SDK、MSDN 和 Win32 API 等常用术语的意思。大多数初学者编写的代码没有固定的风格，更谈不上规范了。这都不是笔者想象中的初学者的样子，所以，本书的第 1 章要讨论这些问题。这部分内容写给从未接触过 Windows 程序的读者，使读者了解相关的知识。

SDK 编程是 Windows 下最基本的编程方式，它是依靠直接调用 API 函数来编写 Windows 应用程序的。本书的第 2、3、4 章详细讲述了最基本的 SDK 编程知识。

在第 5、6、7 章，笔者将和读者一起设计自己的“类库”，这个小“类库”是 MFC 的一个缩影，它将 MFC 中核心的东西，如运行期信息、线程/模块状态、消息映射、内存管理等都非常清晰地体现了出来。在封装 API 的时候，本书详细介绍了 C++语言中虚函数、静态函数、继承和类模板等高级特性的具体应用，叙述了框架程序管理应用程序的每一个细节，用各种精彩的实例讨论了如何进行对象化程序设计，如何使用 MFC 类库简化开发周期。整个过程就是深入了解对象化程序设计模式的过程，也是使读者彻底明白 MFC 内部工作机制的过程。

至此，读者对 Windows 的了解已经是比较系统了。无论是直接调用 API 还是使用微软类库 MFC，读者都应该可以写出界面规范的、标准的 Win32 应用程序，但是还有许多重要的 Windows 高级特性没有被使用。接下来的第 8、9、10、11 章中，通过对各种实例的剖析，详细讨论了 Windows 的高级特性，使读者在实践中提高自己的编程水平。

本书的最后两章对当前流行的计算机 3D 图形绘制和音频控制进行了介绍，详细讲述了 OpenGL、OpenAL 开发库的相关用法。

——对读者的假设

所有的书籍都假设一个基本的知识层次。

首先，读者应该熟知 C 编程语言。本书的 SDK 程序设计部分使用了 C 语言格式的例子演示 Win32 程序运行原理和 API 编程的细节。

其次，读者应该懂得 C++语言的基础知识，像简单的类的封装和对象的概念等。在这个基础上本书会详细介绍 C++的高级特性，用以设计和实现自己的框架和类。

再者，可视化编程的经验也是有用的。如果曾接触过 VB 或 MFC 等工具，将会更容易理解 Win32 程序的结构和 Windows 的消息驱动。

最后，本书不再假设你有任何 Windows 编程经验和其他程序设计语言的知识。

——致谢

感谢我的好友徐超提供并调试了许多实例代码；感谢陈香凝、任淑霞、王杉、闫丽霞、刘旭、张阳、李广鹏、郑琦、孙迪和李宏鹏等参与了部分章节的编写和修改；感谢张铮先生为本书的策划与编写提出的很多宝贵建议。最后，更要感谢我的母亲把我带到这个世界上，抚育我长大。也感谢我的姐姐，她总是诚恳地帮助我。

——关于附书代码和读者反馈

本书的 70 多个例子源代码全部可以在附书光盘中找到，代码全部使用 Visual C++6.0 和 Visual Studio 2010 编译通过。虽然本书中的所有的例子都已经在 Windows 98、Windows 2000、Windows XP、Win7 和 Win8 下测试通过，但由于许多工程比较复杂，也有存在 Bug 的可能，读者如果发现代码存在的错误或者发现书中的其他问题，请告知作者（book_better@sohu.com），以便在下一版中改进。

本书答疑和源程序下载支持网站为金羽图书网：http://www.book95.com。

编　者

目　录

第 1 章　Windows 程序设计基础 ····· 1

1.1　必须了解的东西 ····· 1

1.1.1　Windows 产品概述 ····· 1

1.1.2　开发工具 Visual C++ ····· 1

1.1.3　Windows 资料来源——MSDN ····· 2

1.1.4　Win32 API 简介 ····· 2

1.2　VC++的基本使用 ····· 2

1.2.1　应用程序的类型 ····· 3

1.2.2　第一个控制台应用程序 ····· 3

1.2.3　API 函数的调用方法 ····· 4

1.3　本书推荐的编程环境 ····· 5

1.4　代码的风格 ····· 5

1.4.1　变量的命名 ····· 5

1.4.2　代码的对齐方式 ····· 6

1.4.3　代码的注释 ····· 7

第 2 章　Win32 程序运行原理 ····· 8

2.1　CPU 的保护模式和 Windows 系统 ····· 8

2.1.1　Windows 的多任务实现 ····· 8

2.1.2　虚拟内存 ····· 8

2.1.3　内核模式和用户模式 ····· 9

2.2　内核对象 ····· 10

2.2.1　内核对象的引出 ····· 10

2.2.2　对象句柄 ····· 11

2.2.3　使用计数 ····· 11

2.3　进程的创建 ····· 11

2.3.1　进程（Process）和线程（Thread） ····· 11

2.3.2　应用程序的启动过程 ····· 12

2.3.3　CreateProcess 函数 ····· 13

2.3.4　创建进程的例子 ····· 15

2.4　进程控制 ····· 16

2.4.1　获取系统进程 ····· 16

2.4.2　终止当前进程 ····· 18

2.4.3　终止其他进程 ····· 19

2.4.4　保护进程 ····· 20

2.5　【实例】游戏内存修改器 ····· 20

2.5.1　实现原理 ····· 21

2.5.2　编写测试程序 ····· 22

2.5.3　搜索内存 ····· 22

2.5.4　写进程空间 ····· 25

2.5.5　提炼接口 ····· 26

第 3 章　Win32 程序的执行单元 ····· 27

3.1　多线程 ····· 27

3.1.1　线程的创建 ····· 27

3.1.2　线程内核对象 ····· 30

3.1.3　线程的终止 ····· 32

3.1.4　线程的优先级 ····· 33

3.1.5　C/C++运行期库 ····· 36

3.2　线程同步 ····· 37

3.2.1　临界区对象 ····· 37

3.2.2　互锁函数 ····· 40

3.2.3　事件内核对象 ····· 41

3.2.4　线程局部存储（TLS） ····· 43

3.3　设计自己的线程局部存储 ····· 46

3.3.1　CSimpleList 类 ····· 46

3.3.2　CNoTrackObject 类 ····· 51

3.3.3　CThreadSlotData 类 ····· 53

3.3.4　CThreadLocal 类模板 ····· 61

3.4　设计线程类——CWinThread ····· 64

3.5　【实例】多线程文件搜索器 ····· 73

3.5.1　搜索文件的基本知识 ····· 73

3.5.2　编程思路 ····· 75

第 4 章　Windows 图形界面 ····· 81

4.1　了解窗口 ····· 81

4.2 第一个窗口程序 ····· 82
4.2.1 创建 Win32 工程和 MessageBox 函数 ····· 82
4.2.2 Windows 的消息驱动 ····· 83
4.2.3 创建窗口 ····· 84
4.2.4 分析主程序代码 ····· 86
4.2.5 处理消息的代码 ····· 90
4.3 一个简单的打字程序 ····· 92
4.3.1 使用资源 ····· 92
4.3.2 菜单和图标 ····· 93
4.3.3 接收键盘输入 ····· 95
4.3.4 接收鼠标输入 ····· 97
4.3.5 设置文本颜色和背景色 ····· 98
4.4 GDI 基本图形 ····· 98
4.4.1 设备环境（Device Context） ····· 98
4.4.2 Windows 的颜色和像素点 ····· 100
4.4.3 绘制线条 ····· 101
4.4.4 绘制区域 ····· 104
4.4.5 坐标系统 ····· 105
4.5 【实例】小时钟 ····· 108
4.5.1 基础知识——定时器和系统时间 ····· 109
4.5.2 时钟程序 ····· 111
4.5.3 移动窗口 ····· 114
4.5.4 使用快捷菜单 ····· 115
第 5 章 框架管理基础 ····· 118
5.1 运行时类信息（CRuntimeClass 类） ····· 118
5.1.1 动态类型识别和动态创建 ····· 118
5.1.2 DECLARE_DYNAMIC 等宏的定义 ····· 123
5.2 调试支持 ····· 124
5.2.1 基本调试方法 ····· 124
5.2.2 调试输出 ····· 125
5.2.3 跟踪和断言 ····· 126
5.3 框架程序中的映射 ····· 127
5.3.1 映射的概念 ····· 127
5.3.2 内存分配方式 ····· 128
5.3.3 设计管理方式 ····· 130
5.3.4 句柄映射的实现 ····· 137
5.4 框架程序的状态信息 ····· 138
5.4.1 模块的概念 ····· 138
5.4.2 模块、线程的状态 ····· 139
5.5 框架程序的执行顺序 ····· 141
5.5.1 线程的生命周期 ····· 141
5.5.2 程序的初始化过程 ····· 143
5.5.3 框架程序应用举例 ····· 146
第 6 章 框架中的窗口 ····· 147
6.1 CWnd 类的引出 ····· 147
6.2 窗口句柄映射 ····· 148
6.2.1 向 CWnd 对象分发消息 ····· 148
6.2.2 消息的传递方式 ····· 151
6.3 创建窗口 ····· 153
6.3.1 窗口函数 ····· 153
6.3.2 注册窗口类 ····· 153
6.3.3 消息钩子 ····· 156
6.3.4 最终实现 ····· 160
6.3.5 创建窗口的例子 ····· 162
6.4 消息映射 ····· 164
6.4.1 消息映射表 ····· 164
6.4.2 DECLARE_MESSAGE_MAP 等宏的定义 ····· 166
6.5 消息处理 ····· 168
6.5.1 使用消息映射宏 ····· 168
6.5.2 消息的分发机制 ····· 171
6.5.3 消息映射应用举例 ····· 173
6.6 使用 Microsoft 基础类库 ····· 176
6.7 【实例】窗口查看器 ····· 178
6.7.1 窗口界面 ····· 178
6.7.2 获取目标窗口的信息 ····· 183
6.7.3 自制按钮 ····· 186
第 7 章 用户界面设计 ····· 190
7.1 对话框与子窗口控件基础 ····· 190
7.1.1 子窗口控件运行原理 ····· 190
7.1.2 对话框工作原理 ····· 193
7.2 使用对话框和控件与用户交互 ····· 194

7.2.1 以对话框为主界面的应用程序……194
7.2.2 常用子窗口控件……198
7.2.3 对话框与控件的颜色……199
7.3 通用控件……200
7.3.1 通用控件简介……200
7.3.2 使用通用控件……201
7.3.3 使用状态栏……205
7.3.4 使用列表视图……206
7.3.5 使用进度条……207
7.4 通用对话框……208
7.4.1 “打开”文件和“保存”文件对话框……209
7.4.2 浏览目录对话框……210
7.5 使用框架程序简化界面开发……212
7.5.1 在框架程序中使用对话框……212
7.5.2 CDialog 类……215
7.5.3 框架程序中的控件……217
7.5.4 使用向导……217
7.6 【实例】目录监视器……219
7.6.1 目录监视的基础知识……219
7.6.2 实例程序……220
7.6.3 使用 SkinMagic 美化界面……225

第 8 章 Windows 文件操作和内存映射文件……227

8.1 文件操作……227
8.1.1 创建和读写文件……227
8.1.2 获取文件信息……231
8.1.3 常用文件操作……232
8.1.4 检查 PE 文件有效性的例子……234
8.1.5 MFC 的支持（CFile 类）……235
8.2 驱动器和目录……237
8.2.1 驱动器操作……238
8.2.2 目录操作……240
8.3 使用注册表……240
8.3.1 注册表的结构……240
8.3.2 管理注册表……241
8.3.3 注册表 API 应用举例（设置开机自动启动）……243
8.3.4 ATL 库的支持（CRegKey 类）……244
8.4 内存映射文件……244
8.4.1 内存映射文件相关函数……245
8.4.2 使用内存映射文件读 BMP 文件的例子……246
8.4.3 进程间共享内存……251
8.4.4 封装共享内存类 CShareMemory……253
8.5 一个文件切割系统的实现……254
8.5.1 通信机制……254
8.5.2 分割合并机制……255
8.5.3 接口函数……259
8.5.4 最终实现……259
8.6 【实例】文件切割器开发实例……263

第 9 章 动态链接库和钩子……271

9.1 动态链接库……271
9.1.1 动态链接库的概念……271
9.1.2 创建动态链接库工程……271
9.1.3 动态链接库中的函数……273
9.1.4 使用导出函数……274
9.2 Windows 钩子……276
9.2.1 钩子的概念……276
9.2.2 钩子的安装与卸载……277
9.2.3 键盘钩子实例……278
9.3 挂钩 API 技术（HOOK API）……282
9.3.1 实现原理……283
9.3.2 使用钩子注入 DLL……283
9.3.3 HOOK 过程……284
9.3.4 封装 CAPIHook 类……288
9.3.5 HOOK 实例——进程保护器……294
9.4 其他常用的侦测方法……297
9.4.1 使用注册表注入 DLL……298
9.4.2 使用远程线程注入 DLL……298
9.4.3 通过覆盖代码挂钩 API……303

9.5 【实例】用户模式下侦测 Win32 API 的例子 ……306
第 10 章 TCP/IP 和网络通信 ……311
10.1 网络基础知识 ……311
10.1.1 以太网（Ethernet） ……311
10.1.2 以太网接口堆栈 ……312
10.1.3 服务器/客户机模型 ……313
10.2 Winsock 接口 ……313
10.2.1 套接字（Socket）的概念和类型 ……314
10.2.2 Winsock 的寻址方式和字节顺序 ……314
10.2.3 Winsock 编程流程 ……316
10.2.4 典型过程图 ……318
10.2.5 服务器和客户方程序举例 ……319
10.2.6 UDP 协议编程 ……322
10.3 网络程序实际应用 ……323
10.3.1 设置 I/O 模式 ……323
10.3.2 TCP 服务器实例 ……324
10.3.3 TCP 客户端实例 ……331
10.4 截拦网络数据 ……334
10.4.1 DLL 工程框架 ……335
10.4.2 数据交换机制 ……335
10.4.3 数据的过滤 ……337
10.5 【实例】IP 封包截获工具 IPPack 源代码分析 ……338
10.5.1 主窗口界面 ……339
10.5.2 注入 DLL ……341
10.5.3 处理封包 ……345
第 11 章 内核模式程序设计与 Windows 防火墙开发 ……348
11.1 Windows 操作系统的体系结构 ……348
11.1.1 Windows 2000/XP 组件结构图 ……348
11.1.2 环境子系统和子系统 DLL ……349
11.1.3 系统核心（core） ……350
11.1.4 设备驱动程序 ……352
11.2 服务 ……353
11.2.1 服务控制管理器（Service Control Manager） ……353
11.2.2 服务控制程序（Service Control Program） ……354
11.2.3 封装 CDriver 类 ……356
11.3 开发内核驱动的准备工作 ……360
11.3.1 驱动程序开发工具箱（Driver Development Kit，DDK） ……360
11.3.2 编译和连接内核模式驱动的方法 ……360
11.3.3 创建第一个驱动程序 ……361
11.4 内核模式程序设计基础知识 ……362
11.4.1 UNICODE 字符串 ……362
11.4.2 设备对象 ……362
11.4.3 驱动程序的基本组成 ……363
11.4.4 I/O 请求包（I/O request packet，IRP）和 I/O 堆栈 ……364
11.4.5 完整驱动程序 ……366
11.5 内核模式与用户模式交互 ……369
11.5.1 扩展派遣接口 ……369
11.5.2 IOCTL 应用举例 ……370
11.6 IP 过滤钩子驱动 ……374
11.6.1 创建过滤钩子（Filter-hook）驱动 ……374
11.6.2 IP 过滤钩子驱动工程框架 ……376
11.6.3 过滤列表 ……378
11.6.4 编写过滤函数 ……379
11.6.5 注册钩子回调函数 ……381
11.6.6 处理 IOCTL 设备控制代码 ……383
11.7 【实例】防火墙开发实例 ……384
11.7.1 文档视图 ……384
11.7.2 文档对象 ……387
11.7.3 视图对象 ……388
11.7.4 主窗口对象 ……390
第 12 章 3D 图形绘制及 OpenGL ……393
12.1 计算机 3D 图形渲染技术概述 ……393
12.1.1 计算机 3D 图形技术历史 ……393

12.1.2 Direct3D 技术 ……394
12.2 OpenGL 简介 ……395
12.2.1 OpenGL 技术 ……395
12.2.2 OpenGL 技术的特点 ……395
12.2.3 OpenGL 辅助库简介 ……396
12.3 OpenGL 中的基本概念 ……397
12.3.1 OpenGL 中的函数库简介 ……397
12.3.2 OpenGL 函数命名规则 ……400
12.3.3 OpenGL 中的数据类型 ……400
12.3.4 OpenGL 中的投影变换 ……401
12.3.5 OpenGL 中的色彩模式 ……404
12.3.6 OpenGL 中的纹理贴图 ……406
12.3.7 OpenGL 中的光照 ……409
12.4 OpenGL 开发库的获取 ……412
12.4.1 头文件（.h）的配置 ……412
12.4.2 动态链接库文件（.dll）的配置 ……413
12.4.3 静态链接库文件（.lib）的配置 ……413
12.5 【实例】基于 Win32 的 OpenGL 框架 ……414
12.5.1 建立 Win32 应用程序 ……414
12.5.2 头文件设置及全局变量 ……416
12.5.3 改变 OpenGL 场景尺寸 ……417
12.5.4 OpenGL 的初始化 ……417
12.5.5 OpenGL 的绘制 ……418
12.5.6 关闭 OpenGL ……418
12.5.7 创建 OpenGL 窗口 ……420
12.5.8 处理窗口的消息及键盘事件处理 ……424
12.5.9 WinMain 函数 ……425
12.5.10 本节小结 ……428
12.5.11 本节源码 ……428
12.6 采用 MFC 框架开发 OpenGL 应用程序的优缺点 ……434
12.7 【实例】基于 MFC 中单文档结构的 OpenGL 框架搭建 ……435
12.7.1 创建工程 ……435
12.8.2 添加头文件 ……435
12.7.3 添加库文件 ……436
12.7.4 本节小结 ……441
12.7.5 本节源码 ……441
12.8 【实例】基本平面图形的绘制 ……445
12.8.1 绘图所用的函数介绍 ……445
12.8.2 绘图代码详解 ……445
12.9 【实例】彩色图形的绘制 ……447
12.9.1 绘图所用的函数介绍 ……447
12.9.2 绘图代码详解 ……447
12.10 【实例】图形的平移、旋转与缩放 ……448
12.10.1 绘图所用的函数介绍 ……448
12.10.2 绘图代码详解 ……449
12.11 【实例】立方体的绘制 ……451
12.11.1 绘图所用的函数介绍 ……451
12.11.2 绘图代码详解 ……451
12.12 【实例】使用 GLU 库快速绘制常用几何体 ……452
12.12.1 绘图所用的函数介绍 ……452
12.12.2 绘图代码详解 ……453

第 13 章　音频控制技术及 OpenAL ……455

13.1 计算机音频控制技术概述 ……455
13.1.1 计算机音频控制技术历史 ……455
13.1.2 DirectX Audio 技术 ……456
13.1.3 OpenAL 技术 ……456
13.2 OpenAL 简介 ……456
13.2.1 什么是 OpenAL ……456
13.2.2 OpenAL 的历史 ……457
13.2.3 OpenAL 技术的特点 ……457
13.2.4 OpenAL 支持的音频格式 ……457
13.3 OpenAL 中的基本概念 ……457
13.3.1 OpenAL 中的命名规则 ……457
13.3.2 OpenAL 中的数据类型 ……458
13.3.3 OpenAL 中的缓存 Buffer、声源 Source、听众 Listener ……458
13.3.4 OpenAL 中的设备 Device、环境 Context ……458

13.4 VC++中的 OpenAL 程序设计应用实例……459
13.4.1 OpenAL 开发库的获取……459
13.4.2 【实例】OpenAL SDK Samples 说明……463
13.4.3 【实例】使用 OpenAL 进行设备枚举……465
13.4.4 【实例】使用 OpenAL 播放 WAV 文件……470
附录 MFC 结构体系图……474
参考文献……475

第 1 章　Windows 程序设计基础

本章介绍开发 Win32 程序前的准备工作，包括了解 Windows 产品，熟悉开发工具 VC++，知道如何直接从 Microsoft 获取帮助，如何写风格固定的规范的代码等。

1.1　必须了解的东西

1.1.1　Windows 产品概述

Windows 的操作系统有：

- Windows 95、Windows 98、Windows Me、Windows 2000、Windows 2003
- Windows XP Professional
- Windows XP Home
- Windows XP Media Center Edition
- Windows XP Tablet PC Edition

它们都是 32 位的操作系统，即 CPU 能同时处理的数据的位数为 32 位。Win32 指的是针对 32 位处理器设计的 Windows 操作系统。

本书要讨论的是 32 位环境下 Windows 应用程序设计。Microsoft 为每一个平台都提供了相同的应用程序编程接口（Application Programming Interface，即 API），这意味着如果学会了为一个系统平台编写应用程序，那么也就知道了如何为其他平台编写程序了。

本书主要讲解如何使用 Windows API 函数写 Windows 应用程序，它适用于所有的系统平台。事实上，系统间的差异是存在的，不同系统提供的函数可能是以不同的方式运行，这在书中会尽量指出。

现在，主流的操作系统是 Windows XP，“XP”代表的英文单词是“experience”，象征着此系列的操作系统能够给用户带来新的体验。本书中的例子都是在 Windows XP 系统下完成的。同样，编者也假定您使用的操作系统是 Windows XP 或 Windows 2000，或者是更高的版本。

1.1.2　开发工具 Visual C++

Visual C++是 Windows 环境下最优秀的 C++编译器之一，它是 Microsoft 公司开发的 Visual Studio 系列产品的一部分。Visual C++ .NET 2003 是目前此系列产品的最新版本，但是由于 Visual C++ 6.0 小巧易用，对计算机的软硬件环境要求比较低，而且能够胜任几乎全部的 Windows 应用程序的开发工作，所以，大部分软件开发公司主要的开发平台还是 Visual C++ 6.0。本书的例子代码是用 Visual C++ 6.0 编写并编译的，不过它们也可以在.NET 下编译通过。

建议您也使用 Visual C++ 6.0 来学习 Windows 程序设计，等有了一些软件开发经验之后再去研究.NET 提供的新功能。为了能够在 Visual C++ 6.0 中使用操作系统的新特性，您只需更新一下 SDK 工具即可。

SDK 是 Software Development Kit 的缩写，意思是软件开发工具箱。Microsoft 的 Platform SDK 为开发者提供了开发 Windows 应用程序必要的文档、头文件和例子代码。

VC++ 6.0 自带的 SDK 工具太老了（98 年），对许多新的特性都不支持。在写这本书时 Microsoft 公司刚刚发布了适用于操作系统 Microsoft Windows XP Service Pack 2 的最新的 SDK，用户可以很方便地从 http://www.microsoft.com/msdownload/platformsdk/sdkupdate/站点下载。这个 SDK 既能够被 Visual C++ 6.0 使用，也能够被 Visual C++ .NET 使用。它可以保证用户拥有适用于 Windows XP Service Pack 2 发行版的最新的文档、例子和 SDK 构造环境（包括头文件、运行期库和工具）。

此 SDK 中的 SDK 文档提供了为各种版本的 Windows 系统开发应用程序所需的应用程序编程接口的帮助信息，它还包含关于 Windows Server 2003 的最新信息。

当然，没有必要非得更新 VC++ 6.0 自带的 SDK，本书会在需要的地方明确地指出。

1.1.3 Windows 资料来源——MSDN

MSDN 是微软程序员开发网络（Microsoft Developer Network），是为帮助开发人员使用 Microsoft 的产品和技术写应用程序的一系列在线或者离线的服务。

在使用 VC++ 6.0/.NET 编写程序时，如果想动态地获取帮助，那么就应该安装 MSDN。MSDN 中包含了编程信息、技术论文、文档、工具、程序代码以及新产品的 Beta 测试包。而且，MSDN 也包含相应版本的 SDK 工具。想成为高手必须学会自己查阅 MSDN（或 SDK 文档）来解决问题。

1.1.4 Win32 API 简介

API 是 Application Programming Interface 的简写，意思是应用程序编程接口。可以把它想象成一个程序库，提供各式各样与 Windows 系统服务有关的函数。例如 CreateFile 是用来创建文件的 API 函数；C 的标准库函数 create 也提供了创建文件的函数，但是它是靠调用 CreateFile 函数完成创建文件功能的。事实上，在 Windows 下运行的程序最终都是通过调用 API 函数来完成工作的，因此，可以把 Win32 API 看成是最底层的服务。

通常所说的 SDK 编程就是直接调用 API 函数进行编程。但是 API 函数数量众多，详细了解每一个函数的用法是不可能的，也是完全没有必要的。用户只需知道哪些功能由哪些 API 函数提供就行了，等使用它们时再去查阅帮助文件。

Win32 API 是指编制 32 位应用程序时用的一组函数、结构、宏定义。在 Win32 的环境下，任何语言都是建立在 Win32 API 基础上的，只不过 Visual FoxPro、Visual Basic 等软件对 API 封装得很深。以后本书讨论的应用程序全是通过直接调用 API 函数来实现的（介绍内核驱动的章节除外）。

1.2 VC++的基本使用

本节讲述编写控制台应用程序的方法和如何在程序中调用 API 函数。这些知识非常简单，目的是让从没有接触过 VC++ 6.0 的读者能够轻松入门。

1.2.1　应用程序的类型

Windows 支持两种类型的应用程序：一种是基于图形用户界面（Graphical User Interface，GUI）的窗口应用程序，这是大家常见的 Windows 应用程序；另一种是基于控制台用户界面（Console User Interface，CUI）的应用程序，即“MS-DOS”界面的应用程序。不要以为使用控制台环境的程序就不是 Windows 程序，它可以使用所有的 Win32 API，甚至可以创建窗口进行绘图。所以，这两种应用程序类型间的界限是非常模糊的。

控制台应用程序不需要创建自己的窗口，其输入输出方式也很简单。从这里开始讲述有利于初学者抛开复杂的 Windows 界面管理和消息循环，而去专心研究 API 函数的细节，了解常用的内核对象。下一小节具体介绍如何用 VC++ 6.0 创建控制台应用程序。

1.2.2　第一个控制台应用程序

本节不会全面介绍 VC++ 6.0 的使用方法，而是在后继章节中陆续地介绍。等看完本书后，相信您对集成编译器的使用就很清楚了。下面是使用 VC++ 6.0 创建控制台应用程序的整个过程。

（1）运行 VC++ 6.0，选择菜单命令“File/New...”，在打开的 New 对话框中打开 Projects 选项卡，选项卡左侧的列表框中有多种工程类型，单击 Win32 Console Application（控制台程序）选项，然后在右侧的“Project Name”中输入工程名 01FirstApp，在“Location”中输入存放工程文件的路径“E:\MYWORK\BOOK_CODE”，如图 1.1 左图所示。

（2）单击 New 对话框的 OK 按钮，出现如图 1.1 右图所示的对话框。在这个对话框里，VC 要求你选择一个控制台应用程序的类型，它们分别是空的工程（An empty project）、简单的程序（A simple application）、能够打印出“Hello, World!”字符串的程序（A “Hello, World!” application）、支持 MFC 的应用程序（An application that supports MFC）。在这里选择第三种。

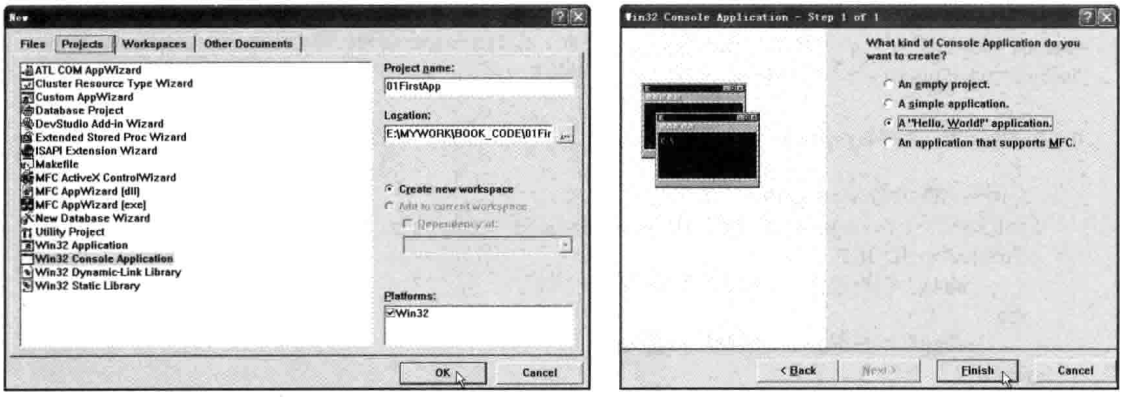

图 1.1　创建 Win32 Console Application 工程

（3）单击 Finish 按钮，弹出一个消息框，直接单击 OK 按钮即可建立一个简单的工程框架，其中含有 VC 自动生成的程序入口函数 main，如图 1.2 所示。

在第二步时，也可以选择其他的控制台工程类型，其结果大同小异，只是 VC 自动生产的代码不同而已。例如，可以选中第一个选项“An empty project”，即建立一个空的工程，然后自己向工程中添加文件（通过菜单命令“File/New...”Files 选项卡）并定义入口函数 main。

现在，向 01FirstApp 工程中添加你自己的文件或代码就行了。程序编写完毕可以按<Ctrl>+F7 键编译，按 F7 键编译连接，按<Ctrl>+F5 组合键运行程序。

如果要将存在的文件添加到工程中，使用菜单命令“Project/Add To Project/Files...”即可。

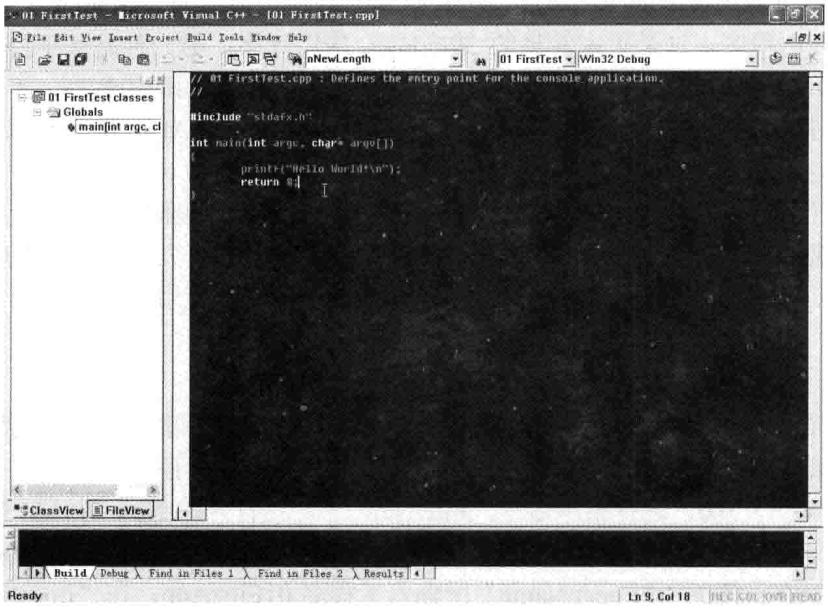

图 1.2　最终生成的工程

1.2.3　API 函数的调用方法

在 VC++ 6.0 下使用 API 函数是非常方便的，只要在文件的开头包含上相应的头文件，然后在程序中直接调用它们就可以了。下面是一个调用 API 函数的例子，修改 01FirstApp 工程中 main 函数的实现代码如下。

```
#include "stdafx.h"        // 这是 VC 自动添加的头文件。为了减少文件间的依赖性，本书建议不使用它
#include <windows.h>   // 包含 MessageBox 函数声明的头文件

int main(int argc, char* argv[])
{
    // 调用 API 函数 MessageBox
    int nSelect = ::MessageBox(NULL, "Hello, Windows XP", "Greetings", MB_OKCANCEL);
    if(nSelect == IDOK)
        printf(" 用户选择了“确定”按钮 \n");
    else
        printf(" 用户选择了“取消”按钮 \n");
    return 0;
}
```

运行程序，除了显示一个控制台外还会弹出一个小对话框，如图 1.3 所示。

图 1.3　MessageBox 函数的调用结果

MessageBox 是众多的 API 函数中的一个，它声明在 windows.h 文件中，用于显示一个指定风格的对话框。在自己的程序中调用 API 函数的方法非常简单，具体步骤如下。

（1）包含要调用函数的声明文件。

（2）连接到指定的库文件（即 lib 文件）。VC 默认已经连接了常用的 lib 文件，所以一般情况下，这一步对我们是透明的。如果需要显式设置的话（如在网络编程时需要添加 WS2_32.lib

库），可以在文件的开头使用“#pragma comment(lib, "mylib.lib")”命令。其中 mylib.lib 是目标库文件。

（3）在 API 函数前加“::”符号，表示这是一个全局的函数，以与 C++类的成员函数相区分。

如果想获得某个 API 函数详细信息，只需将光标移向此函数，按 F1 键即可。也可以打开 MSDN 文档（或 SDK 文档），直接将函数名输入到索引栏来查找函数的用法。

1.3　本书推荐的编程环境

程序开发者长时间盯着屏幕，对自己眼睛的伤害比较大。在编写程序时可以通过改变编程工具的默认颜色来减少显示器对眼睛的伤害，具体做法如下。

（1）单击菜单“Tools/Options”，弹出 Options 窗口，在 Format 页中选取 Category 中的 All Windows，如图 1.4 所示。

（2）在 Colors 栏中对文本颜色、背景色、关键字的颜色等进行设置。

一般来说，按照表 1.1 对各项进行设置后眼睛会觉得舒服许多。

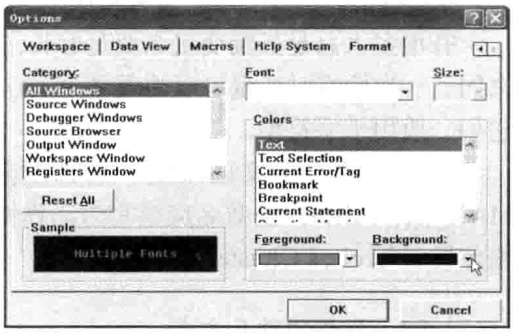

图 1.4　设置编辑器的颜色

表 1.1　　建议的颜色修改表

Colors	Forground	Background
Text（文本）	绿色	深蓝色
Text Selection（选定文本）	蓝色	灰色
BookMark（书签）	黑色	绿色
Breakpoint（断点）	白色	红色
Keyword（关键字）	白色	Automatic
Comment（注释）	灰色	Automatic
Number（数字）	绿色	Automatic

1.4　代码的风格

许多软件公司对员工编写的代码的风格都有规定，比如规定了哪些地方要使用缩排、跳格键的长度、变量命名方式、不同功能代码间空的行数等。这样的好处是可以统一规范不同程序工作者所编制的代码，便于交流和交叉修改等。所以，在本书的开始就将这一点明确地提出来，希望您今后编写有着规范风格的代码，并在编程实践中养成这个好习惯。

1.4.1　变量的命名

（1）变量名应简短且富于描述。变量名的选用应该易于记忆，即能够指出其用途。许多初

学者都不能恰当地给自定义的变量或函数命名，一些编程书籍上竟然通篇地用 IDC_LIST1、IDC_LIST2 这样毫无意义的变量名称，这就跟用 Exam1、Exam2……作为工程的名称一样叫人难以理解。难道就没有一个单词能够形容这些变量的作用和工程的功能吗？如果在数以万计的代码中还采用这种不负责任的命名方式的话，谁能够看懂这些代码？谁又愿意去看它们呢？

（2）变量的名字应该是非形式的、简单的、容易记忆的。变量的作用越大，它的名字要携带的信息就该越多，全局变量应该受到更多的注意。本书规定的变量命名规则为：[限定范围的前缀]+[数据类型前缀]+[有意义的单词]。这一规定的应用举例如下。

```
#define MAX_BUFFER 256          // 定义一个常量，一般常量名应全大写
char g_szTitle[MAX_BUFFER];     // g_前缀表示全局变量，sz 表示类型为字符串，title 是标题的意思
int m_nErrorCode;               // m_前缀表示类的成员变量，n 表示类型为长整型，error code 是错误代码的意思
BOOL bResult;                   // 变量默认即为局部变量，故无需任何限定范围的前缀，b 表示类型为布尔型
```

有很多人总是鼓励变量名要足够长，以携带信息。这是不对的，因为清晰都是随着简洁而来的。一次性临时变量可以被取名为 i、j、k、m 和 n，它们一般用于整型；也可以是 c、d、e，它们一般用于字符型。

（3）作为非明文的规定，局部变量应用小写字母（如 I、j），常量名应全大写（如 MAX_BUFFER），函数名应该写为动作性的（如 CreateDirectory），结构名（类名）应该带有整体性（如 class CRaster）。

1.4.2 代码的对齐方式

“{”、“}”表示一个块，是一个相对独立的语义单元。代码的行与行之间应该按块对齐，而各块之间又应当有适当的缩进，如下面代码所示。

```
void Alert(int i)
{
    while(i > 0)
    {
        // Beep 函数会使扬声器发出简单的声音
        // 要调用这个函数你应该包含上头文件“windows.h”
        Beep(1000, 1000);
        i--;
    }
}
```

用这种方法写出来的程序结构清晰、层次分明，可以使人瞬间毫不费力地读完，没有一点视觉上的障碍。相反，如果不注意对齐和缩进的话，写出来的程序就会显得没有层次，不便于阅读。

块与块间的缩进是靠<Tab>键来完成的，在 VC++中，<Tab>键的默认设置是 4 个字符宽，这使得代码的缩进程度不够明显。本书规定，<Tab>键要设置为 8。在 VC++ 6.0 中打开菜单“Tools/Options”，弹出 Options 对话框，切换到 Tabs 选项卡，将 Tab size 设为 8，如图 1.5 所示。

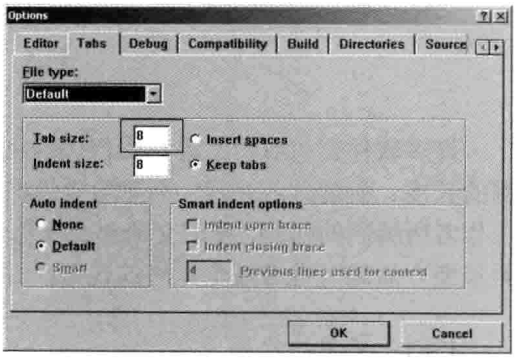

图 1.5 设置 Tab 缩进长度

合理使用空格可以使程序看起来更清爽，而不是一团乱麻。一般在分隔参数、赋值语句和表达式等需要清晰明了的地方使用空格，如下面代码所示。