普通高等教育“十二五”规划教材

（动漫游戏类）

# Game

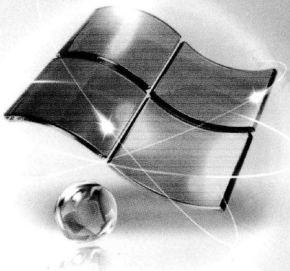
# 网络游戏Windows程序设计教程

房晓溪◎编著

中国水利水电出版社
www.waterpub.com.cn

普通高等教育“十二五”规划教材

（动漫游戏类）



# 网络游戏Windows程序设计教程

房晓溪◎编著

中国水利水电出版社
www.waterpub.com.cn

## 内 容 提 要

本书从易教与易学的实际目标出发，用丰富的范例对Windows网络游戏编程的知识作了生动、详细的讲解。全书共6章，内容包括Windows编程基础，MFC框架和消息，菜单、工具栏和状态栏，对话框程序设计，Windows游戏编程实践，计算机图形学基础。本书内容丰富，讲解精细，通俗易懂，边讲解边操作，大大降低了学习的难度，激发了学习的兴趣和动手的欲望。全书从始至终以讲解Windows网络游戏编程基础为重点，任务明确，步骤清晰，操作方便。每章均有学习要点与学习目标，方便读者抓住每章的重难点。

本书适用于全国高等院校计算机专业学生，游戏编程人员，各类网络游戏编程开发从业人员和爱好者的学习用书。

**图书在版编目（CIP）数据**

网络游戏Windows程序设计教程 / 房晓溪编著. -- 北京 : 中国水利水电出版社, 2011.10
普通高等教育“十二五”规划教材. 动漫游戏类
ISBN 978-7-5084-9052-6

Ⅰ. ①网… Ⅱ. ①房… Ⅲ. ①计算机网络－游戏－应用程序－程序设计 Ⅳ. ①G899

中国版本图书馆CIP数据核字(2011)第200876号

| | |
|---|---|
| 书　名 | 普通高等教育“十二五”规划教材（动漫游戏类）<br>网络游戏Windows程序设计教程 |
| 作　者 | 房晓溪　编著 |
| 出版发行 | 中国水利水电出版社<br>（北京市海淀区玉渊潭南路1号D座　100038）<br>网址：www.waterpub.com.cn<br>E-mail：sales@waterpub.com.cn<br>电话：（010）68367658（发行部） |
| 经　售 | 北京科水图书销售中心（零售）<br>电话：（010）88383994、63202643、68545874<br>全国各地新华书店和相关出版物销售网点 |
| 排　版 | 北京零视点图文设计有限公司 |
| 印　刷 | 北京市兴怀印刷厂 |
| 规　格 | 210mm×285mm　16开本　9.5印张　294千字 |
| 版　次 | 2011年10月第1版　2011年10月第1次印刷 |
| 印　数 | 0001—3000册 |
| 定　价 | 25.00元 |

凡购买我社图书，如有缺页、倒页、脱页的，本社发行部负责调换

# 丛书序

本系列教材是一套应用于游戏教学的专业教材，作者在游戏行业拥有多年的教学经验。本系列教材全面系统地对游戏的策划、程序、美术、运营等各个方面进行了详细的阐述、讲解和实训，并增加了案例分析，以使广大读者更深刻地体会到整个体系。

本系列教材包括以下20本：

- 《游戏概论教程》
- 《游戏策划教程》
- 《游戏架构教程》
- 《游戏运营教程》
- 《游戏美术基础教程》
- 《游戏像素图与界面制作教程》
- 《游戏色彩教程》
- 《游戏渲染教程》
- 《游戏角色建模教程》
- 《三维游戏设计与制作教程》
- 《Java手机基础教程》
- 《J2ME手机游戏项目实战教程》
- 《J2ME手机游戏开发教程》
- 《Symbian手机开发教程》
- 《网络游戏C++程序设计教程》
- 《网络游戏DX程序设计教程》
- 《网络游戏HLSL程序设计教程》
- 《网络游戏Windows程序设计教程》
- 《网络游戏引擎程序设计教程》
- 《游戏电子竞技教程》

本系列教材内容丰富，结构完整，通俗易懂，步骤明确，讲解详尽，是一套系统学习游戏的好书。本系列教材的编写目标是：努力追求“一读就懂，学了能用，一用就灵”的学习效果，可以作为全国各高等院校游戏专业课程的教材，也可作为广大游戏从业人员和管理者的学习用书。

希望本系列教材能为广大读者带来方便，欢迎广大读者提出宝贵的意见，以便我们在以后的版本中不断改进。联系E-mail：fangxiaoxi2002cn@yahoo.com.cn。

作者

2011年1月

# 前　言

Windows操作系统本身和其上的软件技术发展是相当迅速的，Windows不仅是当今主流的操作系统，还是网络游戏主要的开发平台。不论是网络游戏客户端开发还是服务器端开发，都有很多成功的网络游戏作品运行于Windows操作系统上。虽然对于服务器端的开发，还有Unix/Linux以及其他种类的操作系统可以选择，但是相关技术知识相当繁杂，因此，本书详细介绍Windows平台下的程序设计和所涉及到的Windows编程基础知识，这对于开发者针对DirectX程序设计和WinSock程序设计的理解和掌握是相当重要的。本书的作者多年来一直从事动漫游戏开发和高等教育工作，有丰富的实践工作经验。作者将自己在教学和实际开发过程中的一些经验与体会进行整理和总结，最终完成了这本书的编写，希望与广大读者分享。

本书共有6章内容：第1章介绍了Windows编程的基本术语、WindowsGDI进行图形图像绘制的方法、使用消息机制获取鼠标和键盘输入。第2章介绍了MFC框架的基本原理、使用MFC的消息机制、建立用户交互过程。第3章介绍了在MFC中使用资源编辑器编辑菜单和工具栏、实现菜单和工具栏消息响应函数。第4章介绍了建立对话框程序的过程、对话框程序运行的流程数据交换机制的实现、MFC常用控件的创建和访问方法、通用对话框的使用。第5章介绍了使用SDK实现小游戏的方法、使用MFC实现小游戏的方法。第6章介绍了计算机图形学基础知识、向量的基本运算规则及其含义、矩阵运算基本规则图形、几何变换的原理和矩阵在变换中的应用。书中各章节都附有习题，这些内容不仅仅是为了便于学生复习思考，更主要的是作为课堂教学的一种延续。

本书由房晓溪编著，宋忠良、方兴海、卢娜、王俊、严克勇、王瑶、纪赫男参加了本书的部分编写工作。中国水利水电出版社的编辑为本书的编写提出了很多指导性的意见并进行了精心的编辑加工，在此表示衷心感谢。

由于时间仓促，笔者水平有限，在本书编写过程中，难免有不足和错误的地方，恳请读者提出批评和指正。

作　者

2011年3月

# 目　录

# 第1章
# Windows编程基础

**学习目标：**

※ Windows编程基本概念（掌握）
※ Windows GDI函数的使用（掌握）
※ 用消息机制处理用户交互过程（理解）

**背景知识：**

※ C++基础

**本章要点：**

※ Windows GDI函数
※ Windows消息机制

# 引言

Windows是当今主流的操作系统，也是网络游戏主要的开发平台。不论是网络游戏客户端开发还是服务器端开发，都有很多成功的网络游戏作品运行于Windows操作系统上。虽然对于服务器端开发来讲，还有UNIX/Linux以及其他种类的操作系统可以选择，但是相关技术知识相当繁杂，因此，本章将重点介绍Windows平台下的程序设计，其中涉及的若干Windows编程基础概念，对于开发者针对DirectX程序设计和WinSock程序设计的理解和掌握是相当重要的。

Windows操作系统本身和其上的软件技术发展是相当迅速的，从早期的Windows 3.1直到目前的Windows XP的各个版本，其功能愈加强大和完善。网络游戏客户端设计所关注的DirectX技术也在快速发展，当开发者对DirectX 7.0版本到DirectX 9.0版本，甚至即将推出的DirectX 10.0版本间的若干细微变化还应接不暇时，意在整合以Windows、Xbox、Windows CE为代表的家用电脑、家用游戏机、移动设备这三种游戏平台的微软新一代“通用软件开发平台”XNA也已经在如火如荼的宣传之中了，它的含义是DirectX&Xbox、NextGeneration、Architecture。

技术的发展如此迅猛，作为游戏开发者来说，意味着需要投入大量的时间和精力去跟踪学习新的游戏开发技术，这样虽然能够保证游戏的高效开发及技术先进性，但是新技术的学习曲线也是不容忽视的。开发者应该熟知Windows平台的底层开发和高级开发包之间的关系和区别，如果对Windows编程基础还掌握得不是很充分很透彻，那么即使开发包、开发工具再先进，也不能保证开发者可以顺利进行游戏开发。Windows基础方式的编程早在Visual C++ 1.5版本的时候就已经很明显了，那就是基于Windows API的SDK方式的编程，一种C风格的API编程。随后出现的微软MFC库、FCL库可以理解为是对Windows API的重新封装，一种C++风格的面向对象的编程。从技术学习的角度来看，如果希望掌握Windows编程的本质，值得深入研究的是Windows底层的API，也就是SDK方式的Windows编程，而且DirectX的绝大多数版本及相关技术资料都是采用这种方式来进行开发的。因此，虽然本书中的案例开发采用了Visual C++.NET 2003的开发环境，但是本章在介绍Windows编程基础的时候，将重点介绍Windows的SDK方式的开发、消息机制、相关概念及重要的API函数。

API是Windows SDK方式开发的基础，Windows平台上运行的各类软件都可以用Windows API来实现。具体的开发方式非常类似于采用C语言的标准函数库的面向过程的开发，为程序实现一个入口函数main，进而实现其他函数，在main函数中指定各类函数的调用关系及逻辑。Windows API同样也是一个非常庞大的函数库，里面汇聚了Windows平台上的各种系统调用函数，可以实现Windows窗口绘制、网络数据包收发、键盘鼠标的输入获取等众多游戏设计所关注的功能。既然API函数如此庞大，重点又是Windows SDK方式的开发，那么对它的学习肯定也不是一蹴而就的。开发者通常要配备Windows API大全、Windows消息大全这样的参考手册，或者结合微软的MSDN来开发，原因很简单，几乎没有人能对如此大量的Windows API函数和这些函数的参数类型、参数个数、函数返回值、调用方式等熟记于心！本章只拣选对游戏开发比较重要的Windows API来介绍，由于开发目的及实现方式的不同会导致所采用的API也各不相同，所以这里的介绍可能不够全面，只能靠开发者不断地熟悉及实践来扩大知识积累，这一点尤为重要。

当然，在掌握了大量的Windows API函数之后，游戏开发者也可以考虑在开发环境上进行变通，比如不采用Visual C++.NET 2003来开发，而采用诸如Borland C++或者Delphi这样的开发环境来开发都是可行的，关键取决于游戏开发者的技术熟练程度。

## 1.1 Windows 编程约定

### 1.1.1 Windows编程常见缩写

API：Application Programing Interface，应用程序接口。

SDK：Software Development Kit，软件开发包。
MFC：Microsoft Foundation Class，微软基础类库。
MSDN：Microsoft Developer Network，微软开发者网络。
GDI：Graphics Device Interface，图像设备接口。
MDI：Multiple Documents Interface，多重文档界面。
SDI：Simple Document Interface，单文档界面。

### 1.1.2 Windows编程常见数据类型

Windows编程常见数据类型如表1-1所示。

表1-1 Windows编程常见数据类型

| 类型定义 | 含义 |
| --- | --- |
| BOOL | 布尔型（逻辑型）变量（应为TRUE或FALSE） |
| BOOLEAN | 布尔型（逻辑型）变量（应为TRUE或FALSE） |
| BYTE | 字节（8位） |
| CCHAR | Windows字符 |
| CHAR | Windows字符 |
| TCHAR | 取决于预处理器的符号UNICODE是否定义 |
| COLORREF | RGB（红、绿、蓝）颜色值（32位） |
| CONST | 在执行时其值保持不变的变量 |
| DLGPROC | 指向应用程序定义的对话框过程回调过程的指针 |
| DWORD | 双字（32位） |
| DWORDLONG | 双双字（64位） |
| FARPROC | 指向应用程序定义的指针 |
| FLOAT | 浮点型变量 |
| GLOBALHANDLE | 全局内存块句柄 |
| HACCEL | 加速键表句柄 |
| HANDLE | 对象句柄 |
| HBITMAP | 位图句柄 |
| HBRUSH | 画刷句柄 |
| HDC | 设备环境句柄 |
| HFILE | 文件句柄 |
| HFONT | 字体句柄 |
| HGDIOBJ | GDI（图形设备接口）对象句柄 |
| HGLOBAL | 全局内存块句柄 |
| HHOOK | 钩子句柄 |
| HICON | 图标句柄 |

续表

| 类型定义 | 含义 |
| --- | --- |
| HINSTANCE | 实例句柄 |
| HLOCAL | 本地内存句柄 |
| HMENU | 菜单句柄 |
| HOOKPROC | 指向应用程序定义的钩子的指针 |
| HPALETTE | 调色板句柄 |
| HPEN | 画笔句柄 |
| HWND | 窗口句柄 |
| LOCALHAND | 本地内存句柄 |
| LONG | 32位无符号值 |
| LONGLONG | 64位无符号值 |
| LPARAM | 32位消息参数 |
| LPCSTR | 指向Windows常量字符串（以空字符结束）的指针 |
| LPSTR | 指向Windows字符串（以空字符结束）的指针 |
| LPVOID | 指向任意类型的指针 |
| LRESULT | 常规函数返回值 |
| MSG | Windows消息 |
| PROC | 指向回调函数的指针 |
| SHORT | 短整型数 |
| UCHAR | 无符号Windows字符 |
| UINT | 无符号整数 |
| ULONG | 无符号长整型数（32位） |
| USHORT | 无符号短整型数（16位） |
| VOID | 任意类型 |
| WINAPI | FAR PASCAL的等价声明方式 |
| WNDPROC | 指向在应用程序中定义的窗口过程的指针 |
| WORD | 无符号字（16位） |
| WPARAM | 32位消息参数，早期Win3.x中为16位 |

以上列表中包括字符型、整型、浮点型、布尔型、指针型以及Windows应用程序特有的句柄型，表示指针型的数据类型往往以P或LP作为前缀，而句柄型总是冠以H。当然上面的列表并非全部，还有诸如CALLBACK回调函数的声明方式、WNDCLASS窗口类的声明方式等繁多的类型定义或宏定义。

### 1.1.3 Windows编程命名规则

介绍了常见的Windows数据类型后，不能不提到著名的“匈牙利命名法”。

（1）属性部分。

| | |
|---|---|
| 全局变量： | g_ |
| 常量： | c_ |
| 类成员变量： | m_ |

（2）类型部分。

| | |
|---|---|
| 指针： | p |
| 句柄： | h |
| 布尔型： | b |
| 浮点型： | f |
| 无符号： | u |

（3）描述部分。

| | |
|---|---|
| 初始化： | Init |
| 临时变量： | Tmp |
| 目的对象： | Dst |
| 源对象： | Src |
| 窗口： | Wnd |

了解这种命名规则对理解常见的Windows程序非常有帮助，对于代码调试也用途很大，现在可以结合这种命名规则来理解下面的定义。

hWnd：h表示句柄，Wnd表示窗口。

g_bFlag：g表示全局变量，b表示布尔型，Flag表示状态标志。

结合以上介绍的Windows编程常见数据类型，对不同类型的句柄进行理解记忆，在后续的Windows程序开发中就会受益匪浅。下面结合各个项目来具体了解一下这些相关概念。

## 1.2 Windows编程基本概念

### 1.2.1 句柄

几乎Windows编程涉及的很多内容都和句柄有关联。句柄是用来标识项目的，这些项目包括：

- 窗口（window）。
- 资源（resource），包括图标（icon）、光标（cursor）、菜单（menu）、字符串（string）等。
- GDI对象（GDI object），包括位图（bitmap）、画刷（brush）、元文件（metafile）、调色板（palette）、画笔（pen）、区域（region）以及设备环境（device context）。
- 模块（module）。
- 实例（instance）。
- 文件（file）。
- 内存块（block of memory）。
- 控件（control）。
- 字体（font）。

注：device context通常译作设备上下文，这里为了理解方便称为“设备环境”。

1．窗口

“窗口”是Windows程序实现的基础，大多数Windows程序都是以窗口的方式运行的。网络游戏客户端编程也要用DirectX在窗口基础上实现程序，而网络游戏服务器端编程中，诸如监测工具、GM工具等程序实现也常以窗口方式运行。在Windows开发中，可以通过设置参数来决定窗口类型，这样可以实现多样的窗口外观，比如有菜单栏的窗口、有状态栏的窗口、有滚动条的窗口等。从技术角度上看，每个

窗口都具有窗口句柄（HWND类型的变量），在系统内可以通过窗口句柄定位具体的可见或不可见的窗口。每个窗口又对应着窗口类，同样的窗口类可以用来创建多个具有相同外观和相同特性的窗口，例如Windows的文件夹可以是相同外观，也可以同时具有相同的特性，可以使用滚动条浏览文件等。

2．资源

“资源”的概念比较广泛，Windows的视窗系统又实在是庞大而复杂。在Windows程序设计中，对话框、光标、图标、位图、菜单、快捷键、字符串、版本信息等诸多构成窗口的要素都是资源类型。在Visual C++ 6.0版本的界面中也提供了专门的Tab页来方便开发者对资源进行有效管理并能够将资源进行编译、存放在.rc格式的文件中。每个资源都对应唯一的ID值，应用程序通过资源ID对资源进行定位。浅显的理解是资源为在窗口程序开发中相对固定的窗口要素。实际在窗口程序开发中，不但要通过设计工具来制作、导入各种资源，还要对这些资源实现代码控制，比如对话框这种资源通常是要对用户的输入进行响应的，具体能完成什么功能就要靠开发者具体实现。

3．GDI对象

“GDI对象”在游戏客户端开发中应用比较少，因为游戏界面的众多元素通常都以图像来实现，以突出游戏的画面风格。这里需要强调的就是设备环境，通常在SDK方式的开发中，都要涉及HDC的数据类型，所有的与图形有关的调用将在HDC变量初始化后继续进行，所以简单地将HDC理解为它是图像显示设备的一种抽象，这样在进行绘图的时候就可以忽略计算机显卡的具体细节。

4．模块

“模块”在Windows编程中实际上就是对应着DLL编程。DLL（Dynamic Link Library，动态链接库）是在Windows系统中广泛采用的函数封装方式。对应于C语言中常见的静态函数库.lib在生成可执行文件时即参与链接的方式，.dll可以在可执行文件运行时根据需要将动态链接库加载到内存以继续运行。这样的动态加载方式的优点显而易见，那就是可以大大地减少可执行文件的大小，对于有大量函数可以共用的程序非常有优势。比如，Windows系统是很庞大的，可共用的函数众多。在系统层面上，早期Windows版本中采用user.dll、kernel.dll和gui.dll作为核心的动态链接库，在应用层面上，Windows提供网络链接的库为ws2_32.dll，也是采用动态链接库实现的。

5．实例

“实例”也是一种重要的句柄，Windows系统中创建的各个应用程序都对应着不同的实例句柄。Windows是一种多用户操作系统，同样的应用程序可以启动多次。每次开启同样的应用程序时，创建的实例是不一样的。这里需要注意的是实例句柄和窗口句柄的关系：如果应用程序可以开启窗口，那么一个应用程序可能开启多个窗口，也就是说同样的实例句柄可能关联着多个窗口句柄；如果相同的应用程序再次启动，同样开启了多个窗口，那么新启动的应用程序具有新的实例句柄以及和这个句柄对应的新的窗口句柄。

6．文件

“文件”句柄在Windows编程中实现数据永久存储时很重要。在Windows平台发展过程中的文件系统有FAT、FAT32、NTFS。如果开发者在访问一个文件的时候都要考虑到操作系统存储文件时的数据组织方式，这无疑给开发者带来了很大的负担。因此Windows API中封装了与文件操作相关的大量函数，而这些函数的使用，都和文件句柄有关系。因此文件句柄也是重要的句柄类型。

7．内存块

“内存块”句柄与Windows操作系统在内存管理上的实现方式关系密切。在早期的计算机中通常可使用的内存和硬盘是严格区分的，并且内存管理上通常都采用特殊的寻址方式，非常不利于开发者操作内存。而在现代操作系统中，通常都使用了“页”式内存管理，而可寻的地址范围则更大。在Windows系统中，应用程序可寻址的内存达到4GB，但实际上低端2GB内存是Windows系统使用的，所以通常可以认为可寻址可使用的内存为2GB。当然，PC中内存一般没有这么大，此时操作系统会利用硬盘空间“虚拟”

出更多的内存供应用程序使用，但是硬盘数据的存取速度和内存数据的存取速度是无法比拟的，所以如果程序要大量使用虚拟出来的内存，那么程序本身的响应速度和系统性能都会下降。为了操作内存就必须使用内存块句柄。

8．控件

“控件”句柄也是经常使用的一种句柄。现代操作系统往往在软件的可重用性上重点实现。相关的OLE技术、ActiveX技术代表了Windows平台上控件技术的发展。虽然在游戏程序开发当中很少采用已有的控件，但是对控件技术的掌握可以使得大量的常规程序开发工作变得简单，可以利用大量的控件资源快速地实现程序功能。

9．字体

“字体”句柄在设计界面信息交互时十分重要。不论是窗口应用开发还是游戏客户端图形图像设计，从用户使用的角度来讲，图形图像对信息传达的力度是远远不够的，面对计算机，用户获取的大量信息还是来自于文字信息。对于游戏客户端设计来说，使用字体句柄来指定所要选择的字体库、字形、字号。通过美观的字体来塑造表现力丰富的画面信息，实现玩家与游戏之间、玩家与玩家之间的信息交流。

句柄的使用通常通过调用特定的API函数来进行，例如：

```
//加载资源ID为IDI_WINLOGO的图标并返回HICON类型的句柄
HICON hIcon = LoadIcon(NULL,IDI_WINLOGO);
//加载Windows默认的箭头鼠标形状并返回HCURSOR类型的句柄
LoadCursor(NULL,IDC_ARROW);
//取得Windows预定义的黑色刷子用来绘制区域并返回HBRUSH 类型的句柄
HBRUSH hbrBackground = GetStockObject(BLACK_BRUSH);
```

## 1.2.2 Windows消息机制简介

1．消息

Windows是一个消息驱动的操作系统，因此理解Windows的消息机制非常重要。消息为应用程序和应用程序间、应用程序和操作系统间提供了信息传递的渠道。在早期的16位Windows系统中，操作系统实现方式是协同式多任务系统，整个系统只有一个消息队列，所有应用程序都要访问这个消息队列以便检查是否有自己所关心的消息。在后来的32位Windows 系统中，操作系统实现方式是抢占式多任务系统，系统中每个运行中的应用程序都有一个消息队列，系统不用等到应用程序完成消息处理就可以得到控制权。

消息是由消息ID（UINT）和两个消息参数（WPARAM与LPARAM）构成的。不论是用户和窗口的键盘鼠标交互还是窗口本身状态的改变，系统都会发送特定的消息到当前窗口，而每个窗口都有预先指定的消息处理函数（通常是WndProc），在消息处理函数中会识别系统所发送的消息，根据消息ID进行具体处理。

```
typedef struct tagMSG {                    //消息结构体
    HWND                hwnd;              //窗口句柄
    UINT                message;           //消息的类型
    WPARAM              wParam;            //消息的额外信息，含义由消息类型确定
    LPARAM              lParam;            //消息的额外信息，含义由消息类型确定
    DWORD               time;              //投递消息的时间
    POINT               pt;                //投递消息时光标的位置
} MSG;
```

比如，当键盘被按下时，WM_KEYDOWN消息发送到当前窗口，消息的WPARAM参数中包含按键的字符信息；当鼠标左键被按下时，WM_LBUTTONDOWN消息发送到当前窗口，消息的LPARAM参数的低16位和高16位分别封装了鼠标点击时的x坐标和y坐标；当用户关闭窗口时，WM_QUIT消息发送到当前窗口，可以识别这个消息做关闭处理；当用户点击窗口菜单项时，WM_COMMAND消息发送到当前窗口，可以通过消息参数继续识别所点击的菜单项来进行处理；当窗口需要重新绘制以显示图形时，WM_PAINT消息发送到当前窗口，可以在这个消息的处理过程中调用各种绘图API来显示图形，Windows SDK方式的开发实际上就是开发者根据实现目标选择需要处理的Windows消息，进而对这些消息进行处理，完成应用程序开发。常见的Windows消息参见下面的表1-2。

表1-2　常见Windows消息

| 消息名称 | 消息含义 |
| --- | --- |
| WM_CREATE | 应用程序创建一个窗口 |
| WM_DESTROY | 一个窗口被销毁 |
| WM_MOVE | 移动一个窗口 |
| WM_SIZE | 改变一个窗口的大小 |
| WM_ACTIVATE | 一个窗口被激活或失去激活状态 |
| WM_SETFOCUS | 获得焦点后 |
| WM_KILLFOCUS | 失去焦点 |
| WM_ENABLE | 改变Enable状态 |
| WM_SETREDRAW | 设置窗口是否能重画 |
| WM_SETTEXT | 应用程序发送此消息来设置一个窗口的文本 |
| WM_GETTEXT | 应用程序发送此消息来复制对应窗口的文本到缓冲区 |
| WM_GETTEXTLENGTH | 得到与一个窗口有关的文本的长度（不包含空字符） |
| WM_PAINT | 要求一个窗口重画自己 |
| WM_CLOSE | 当一个窗口或应用程序要关闭时发送一个信号 |
| WM_QUERYENDSESSION | 当用户选择结束对话框或程序自己调用ExitWindows函数 |
| WM_QUIT | 用来结束程序运行或当程序调用postquitmessage函数 |
| WM_QUERYOPEN | 当用户窗口恢复以前的大小位置时，把此消息发送给某个图标 |
| WM_SHOWWINDOW | 当隐藏或显示窗口时发送此消息给这个窗口 |
| WM_DRAWITEM | 当控件的可视外观改变时发送此消息给这些控件的拥有者 |
| WM_SETFONT | 当绘制文本时程序发送此消息得到控件要用的颜色 |
| WM_GETFONT | 应用程序发送此消息得到当前控件绘制文本的字体 |
| WM_SETHOTKEY | 应用程序发送此消息让一个窗口与一个热键相关联 |
| WM_GETHOTKEY | 应用程序发送此消息来判断热键与某个窗口是否有关联 |
| WM_QUERYDRAGICON | 此消息发送给最小化窗口 |
| WM_KEYDOWN | 按下一个键 |
| WM_KEYUP | 释放一个键 |
| WM_CHAR | 按下某键，并已发出WM_KEYDOWN、WM_KEYUP消息 |
| WM_SYSKEYDOWN | 当用户按住Alt键同时按下其他键 |
| WM_SYSKEYUP | 当用户释放一个键同时还按着Alt键 |
| WM_INITDIALOG | 在一个对话框程序被显示前发送此消息 |
| WM_COMMAND | 选择一条菜单命令项或当某个控件发送一条消息给它的父窗口 |
| WM_TIMER | 发生了定时器事件 |
| WM_MOUSEMOVE | 移动鼠标 |

续表

| 消息名称 | 消息含义 |
| --- | --- |
| WM_LBUTTONDOWN | 按下鼠标左键 |
| WM_LBUTTONUP | 释放鼠标左键 |
| WM_LBUTTONDBLCLK | 双击鼠标左键 |
| WM_RBUTTONDOWN | 按下鼠标右键 |
| WM_RBUTTONUP | 释放鼠标右键 |
| WM_RBUTTONDBLCLK | 双击鼠标右键 |
| WM_MBUTTONDOWN | 按下鼠标中键 |
| WM_MBUTTONUP | 释放鼠标中键 |
| WM_MBUTTONDBLCLK | 双击鼠标中键 |
| WM_MOUSEWHEEL | 当鼠标滚轮转动时发送此消息 |
| WM_CAPTURECHANGED | 当失去捕获的鼠标时发送此消息给窗口 |
| WM_MOVING | 移动窗口时发送此消息 |
| WM_CUT | 发送此消息给一个编辑框或ComboBox来删除当前选择的文本 |
| WM_COPY | 发送此消息给一个编辑框或ComboBox来复制当前选择的文本 |
| WM_PASTE | 发送此消息给EditControl或ComboBox从剪贴板中得到数据 |
| WM_CLEAR | 发送此消息给EditControl或ComboBox清除当前选择的内容 |
| WM_UNDO | 发送此消息给EditControl或ComboBox撤销最后一次操作 |
| WM_USER | 此消息能帮助应用程序自定义消息 |

Windows以宏的方式预定义了许多系统消息，开发者也可以自定义消息。

2．消息循环

Windows如何处理这些消息呢？这里要重点介绍一下Windows系统的消息循环机制。前面提到过，Windows系统中运行的每个应用程序都拥有自己的消息队列，每个应用程序都通过while循环不停地获取消息，当发现应用程序本身关心的消息时就通过swtich语句来分别进行处理，否则就让应用程序处于空闲状态。典型的消息循环语句如下：

```
while (GetMessage(&msg, NULL, 0, 0))
 {
  TranslateMessage(&msg);
  DispatchMessage(&msg);
 }
```

关于消息循环中API函数的含义将在后面结合实际代码来说明，这里先做个了解。

## 1.3 Windows SDK典型的程序结构

一个典型的Win32窗口应用程序结构从开发逻辑上说如图1-1所示。