



软件开发技术丛书

Visual C++ MFC

编程实例

Visual C++ MFC Programming by Example

(美) John E. Swanke 著 前导工作室 译



机械工业出版社
China Machine Press



TP312C
S81d

437

软件开发技术丛书

Visual C++ MFC 编程实例

(美) John E. Swanke 著

前导工作室 译

本书附盘可从本馆主页 <http://lib.szu.edu.cn/>
上由“馆藏检索”该书详细信息后下载，
也可到视听部复制



A0875859



机械工业出版社
China Machine Press

本书通过85个实例全面讲述了应用MFC进行Visual C++编程的思想。每个实例均以编写一个应用程序要走的步骤编写。全书共分四部分进行介绍,第一部分是基础知识,第二部分讲述用户界面的实例,第三部分讲述MFC内部处理方面的实例,第四部分讲述打包实例。

全书基本上面向实例进行阐述,讲解透彻、易于掌握。本书既可作为初学者和大专院校师生的自学参考书,也可作为计算机软件开发人员的技术参考书。

John E. Swanke: Visual C++ MFC Programming by Example.

Original edition copyright © 1999 by Miller Freeman Inc. 600 Harrison Street, San Francisco, California 94107 USA.

All rights reserved.

本书中文版由美国 Miller Freeman 公司授权机械工业出版社独家出版。未经出版者书面许可,不得以任何方式复制或抄袭本书内容。

版权所有,侵权必究。

本书版权登记号: 图字: 01-1999-2867

图书在版编目(CIP)数据

Visual C++ MFC 编程实例 / (美)斯文克(Swanke, J.E.)著; 前导工作室译. - 北京: 机械工业出版社, 2000.1

(软件开发技术丛书)

书名原文: Visual C++ MFC Programming by Example

ISBN 7-111-07270-7

I.V... II. ①斯... ②前... III.C语言-程序设计 IV.TP312

中国版本图书馆CIP数据核字(1999)第67093号

机械工业出版社(北京市西城区百万庄大街22号 邮政编码 100037)

责任编辑: 吴 怡 李新阳

北京牛山世兴印刷厂印刷·新华书店北京发行所发行

2000年1月第1版·2000年3月第2次印刷

787mm×1092mm 1/16·22.25印张

印数: 6 001-10 000 册

定价: 48.00元(附光盘)

凡购本书, 如有倒页、脱页、缺页, 由本社发行部调换

前言

当学习一个新的应用程序时，研究一个相关的例子可以在较短的时间内获得较大的效果，而如果以随机地阅读一些陈旧的资料开始，或者更糟，阅读一些不连贯的CD或联机文档，可能要花上成倍的时间却收效甚微。

通过例子，可以更清晰地感受到一种语言或一个类的开发者是怎样引导用户使用他们的软件的。什么时候应该使用无模式对话框、一个对话条或者一个小型框架窗口呢？应该使用一个多文档界面(MDI)、单文档界面(SDI)或者只是一个普通的对话框应用程序？如果想改变初时的决定的话，又该怎么做呢？

本书提供了大量使用Microsoft的Visual C++和MFC的应用程序和特征的例子。与MFC所带的例子不一样，这些例子可以在当今最畅销的Windows 95/NT应用程序中找到。对于那些想要开发最畅销软件的人，本书具有很高的参考价值，每个例子都说明了它的工作机制，并提示用户可以从获得的精华。

本书内容

本书中的例子按编写一个应用程序时可能要走的步骤进行组织，从创建一个应用程序的工程开始，到最后修整界面。然而，每个应用程序又是独立的，可以以任意的顺序使用它们。

同类型的例子被合并到同一章中，全书包括以下四部分：

基础知识

尽管我尽力使本书面向例子，但仍感到前面了解一些基本知识有助于后面更容易地理解例子。当然，若不愿看这一部分的话也可以跳过去，但不要为此而感到不安。

用户界面实例

本部分的例子集中讨论应用程序的用户界面。因为MFC是以界面为中心的，因此本书的绝大多数例子包括在本部分。各例子以创建一个MFC应用程序时应该考虑的顺序进行安排，包括应用程序类型、菜单、工具栏和状态栏、视图、对话框和对话条、控件窗口和绘图。本部分省去了一些例子的应用，包括文本编辑器和向导。

内部处理实例

本部分的例子虽然仍然适用于MFC应用程序，但更能代表的是MFC的不可见部分，而不是用户界面。例子包括消息处理、文件、列表、映像、数组、剪切和粘贴，以及时间。

打包实例

最后一部分包括打包用户软件，不是打包成一个标准的可执行文件，而是打包成诸如动态链接库(DLL，包括MFC和非MFC)和资源库等形式。

关于CD

本书带有一张CD，包括了本书中每个可在Visual C++ 5.0上运行的例子的工程。如果想找书上某个特定例子的工程，只要在CD目录中找到该例子的名字即可。

除了特别提到的以外，CD上的大部分例子作为MDI应用程序创建，并且使用AppWizard的所有默认设置，工程名为“Wzd”。

关于实例向导

在CD上还有一个实例向导(SampleWizard)应用程序，它能帮助你直接把本书中的例子添加到用户的应用程序中。该应用程序引导你浏览例子的目录，如果选中了，将详细列出在工程中包括该例子所需的代码和提示。还可以用自己起的工程名替换例子的工程名(“Wzd”)。

实例向导位于CD的\SWD目录下，它使用CD上每个例子的\Wizard子目录，并包括了那个例子的所有细节。

只要运行SW.EXE，你就会发现所有的一切都会变得非常直观。

英文原书书号 ISDN: 0-87930-544-4

第一部分 基础知识

用Visual C++和MFC创建的应用程序大多会自动生成窗口，并且可以处理消息，进行绘图。Microsoft在这方面做了大量的工作，隐藏了内部工作，使我们能够更轻松地创建一个一般的应用程序。然而，当用户不能实现他们想要实现的功能时，适当地了解内部工作机制，对于消除编程上的困惑会有好处。更重要的是，知道怎样执行任务(诸如把窗口放置到什么地方，从什么地方获得一个消息和在任意地方绘图)，有助于分清用户的应用程序和由Visual C++和MFC自动提供的限于窗口、消息和绘图的应用程序。

Visual C++应用程序有四项主要基本知识：创建一个窗口、了解其他的MFC类、把消息发送到一个窗口和在一个窗口内绘图。当然还有其他一些基本知识，我们也将涉及时适当地进行讨论。不过本部分将讨论以下四项基本知识。

窗口

在第1章中，我们首先讨论在使用和不使用MFC的情况下创建一个窗口，以便清楚地了解MFC是如何工作的。MFC窗口既可以由属于MFC的C++类创建，也可以由一个早于并存在于MFC之外的非C++窗口类创建。我们将进一步观看窗口类并讨论那些由Windows操作系统提供的窗口类。最后，再看看MFC应用程序中都有什么。

类

在第2章中，我们将全面地讨论MFC提供的强大的功能。大多数MFC类是从三个MFC基类(CObject、CWnd和CCommandTarget)派生来的。并讨论构成一个应用程序的MFC类、支持窗口界面的MFC类、用来绘图的类、访问文件的类、维护数据和数据库的类和访问因特网(Internet)的MFC类。

消息处理

在第3章中，我们将讨论一个MFC应用程序是怎样通过消息与外界及应用程序进行通信的。还将讨论四种消息类型，并跟踪一个消息通过接收消息的类。最后将探讨该路径上的重定向消息。

绘图

在第4章中，我们将讨论在窗口中绘图的方法，包括绘图工具、绘图用的颜色、在屏幕或打印机上绘图、绘制位图和图标、绘制矩形和圆、绘制动画。

第1章 窗 口

在本章中，我们将讨论MFC用户界面的基本要素：窗口。在此基础上比较API窗口和MFC窗口的异同，描述如何创建一个窗口、销毁一个窗口和控制Windows操作系统与窗口的通信问题。

1.1 窗口和API环境

窗口是屏幕上的一个矩形区域，应用程序在该区域中显示数据并等待鼠标点击。Windows应用系统的用户界面可以包含许多窗口，每个窗口都有不同的特点，但都是互相联系的，如图1-1所示。

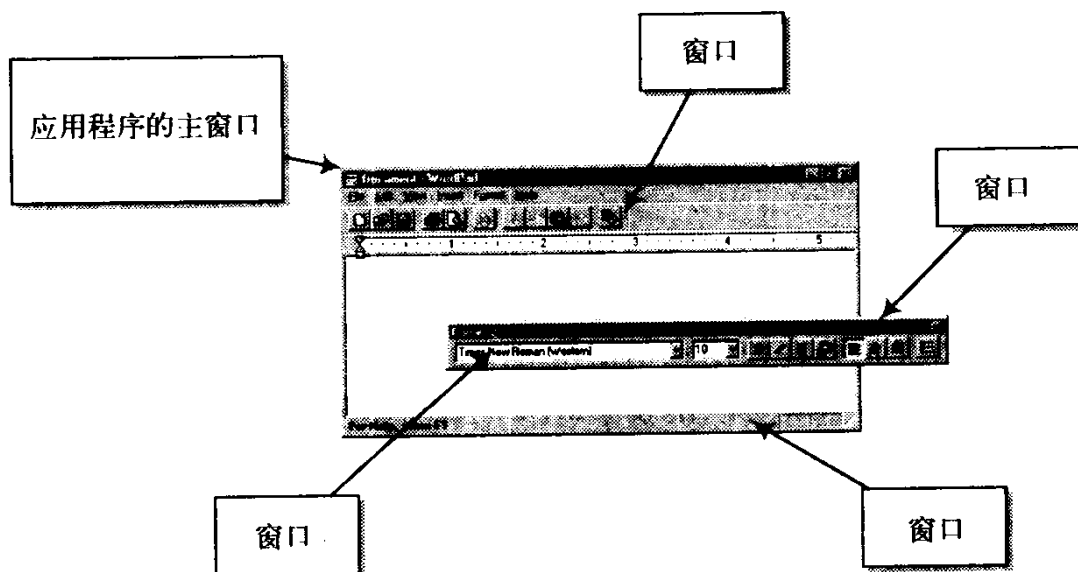


图1-1 Windows应用程序用户界面包括的窗口

1.1.1 三种类型窗口

在这么多窗口中，基本类型只有三种：重叠(Overlapped)窗口，弹出(Popup)窗口和子窗口(Child)。在它们之间并没有太多内在的差异，但是使用不同的窗口风格，它们的外观是不同的(见图1-2)。

重叠窗口 通常用于建立应用程序主窗口。事实上，有时也叫做“主”窗口或者“框架”窗口。

弹出窗口 通常以对话框和消息框的形式与用户对话。

子窗口 通常用在视图(View)中，如在文本编辑器中的文本显示，也用在控件中，如在对话框中的OK按钮。而对那些看起来像按钮或控件的子窗口，也称为“控件”窗口。

重叠窗口和弹出窗口的主要区别是弹出窗口出现时可以没有标题(也称为标题栏)。子窗口与重叠窗口或弹出窗口的主要区别是子窗口只能出现在另一个窗口中，并且子窗口的任何多余部分都被该窗口移去或剪切掉。另外，子窗口是唯一不能有菜单条的窗口。

参见图1-3中的Windows 应用程序，其中包括重叠窗口、弹出窗口和子窗口。

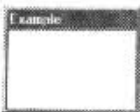
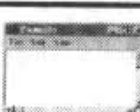
	重叠窗口	弹出窗口	子窗口
最基本的窗体		注意：在它的最基本的窗体中，系统不进行任何绘制。窗口都是客户区	注意：在它的最基本的窗体中，系统不进行任何绘制。窗口都是客户区
典型的外观			
共同的设置			

图1-2 窗口风格可以用来区分三种不同类型的窗口

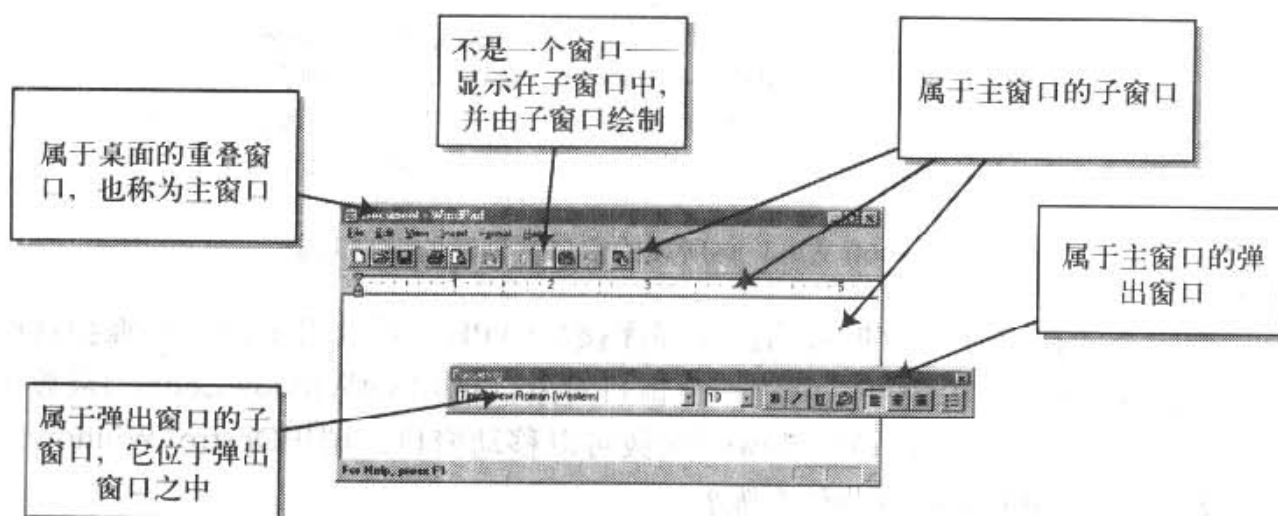


图1-3 由重叠窗口、弹出窗口和子窗口组成的窗口应用程序

1.1.2 客户区和非客户区

每个窗口都有由系统绘制的“非客户区”和由应用程序绘制的“客户区”。系统可以绘制图1-4显示的其中一个或者全部特征，当然也可以把所有的特征留给你去绘制。

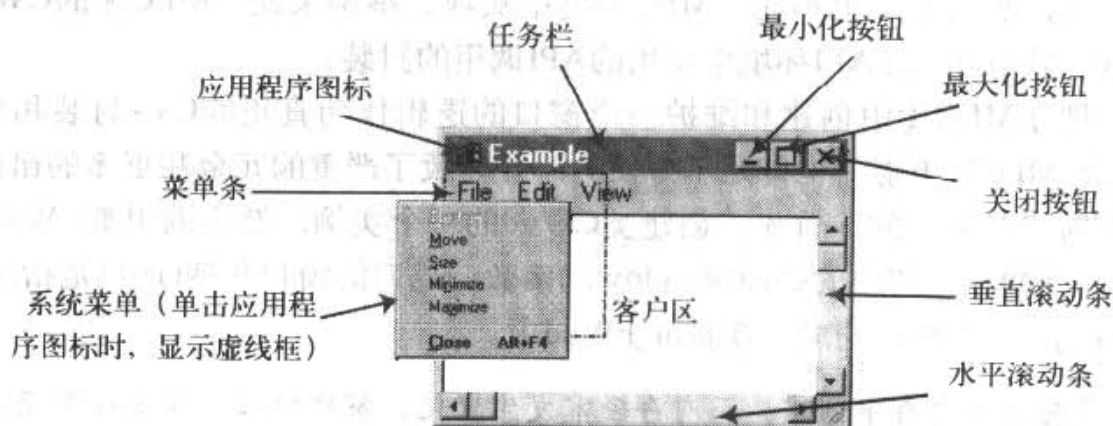


图1-4 窗口的非客户区可以由系统选择绘制

每个窗口代表内存中的一个窗口对象，并由该窗口对象告诉Windows操作系统将窗口绘制在何处，以及在对鼠标单击、键盘按下(假设该窗口拥有输入焦点)和时钟终止等事件作出响

应时应调用什么应用程序。窗口对象自身是窗口类的一个实例，它不是Visual C++的类，而是存在于Visual C++之外，并早于Visual C++的Microsoft Windows所属的类。然而，就像C++的类一样，窗口类为每个基于它创建的窗口定义了若干特征，如背景色和往何处发送消息等(见图1-5)。

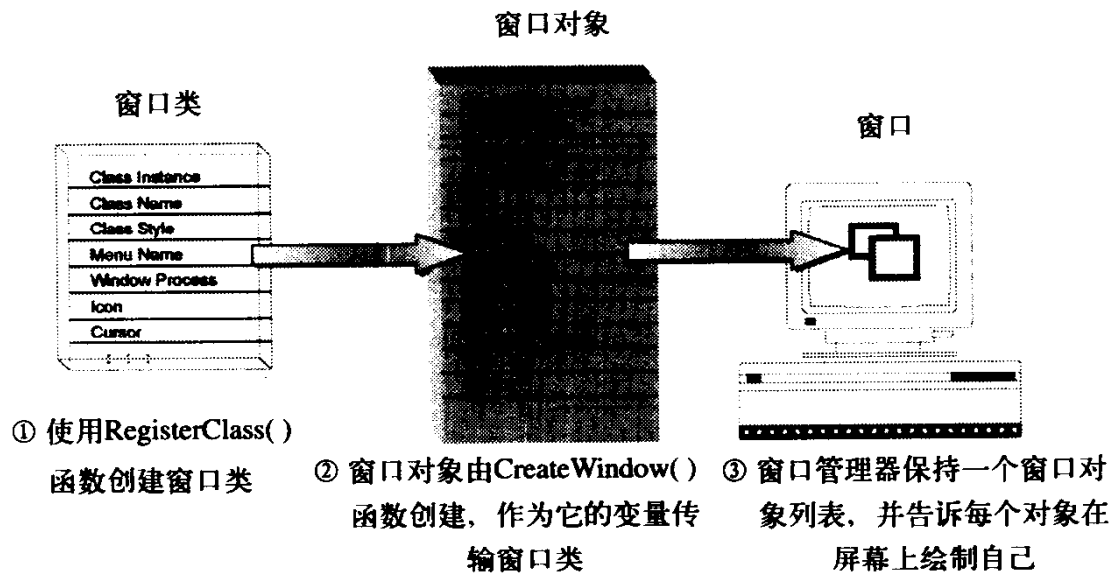


图1-5 Microsoft Windows所属的窗口类创建的窗口对象

Windows操作系统提供了一个扩展的应用程序接口(API)，可以用来创建和维护这些窗口对象。调用CreateWindow()函数可以创建一个窗口对象；调用SetWindowLong()函数可以改变由窗口类定义的特征；调用MoveWindow()函数可以移动窗口；调用DestroyWindow()可以退出窗口。如此说来，MFC又能做些什么呢？

1.2 窗口和MFC环境

MFC窗口是C++和Windows API调用的综合。事实上，MFC窗口提供了许多(但不是全部)Windows API的C++封装，从而减轻了编写Windows应用程序时一些乏味的工作，并提供了一些新的服务。

MFC窗口不对窗口对象进行直接控制，而在API环境中却是可以的。如果不能在API环境中实现某项功能，那么肯定也不能在MFC环境中实现。举例来说，MFC库的CWnd可以创建一个窗口，但它只是过去在API环境中使用的API调用的封装。

微软已经把在MFC库中创建和维护一个窗口的逻辑作为真正的C++封装和控制。然而，这种方法虽使得MFC库万能，但却是不方便的，并导致了严重的冗余和更多的错误。

创建MFC窗口是复杂的，首先，创建类CWnd的一个实例，然后调用类CWnd的一个成员函数，该成员函数调用API中的CreatWindow()函数。返回的窗口句柄(这只是指向窗口对象的非直接指针)保存在类CWnd的成员变量m_hWnd中。

注意 因为窗口在内存中创建，而内存经常发生变化，窗口地址可能是经常变化的。因此，窗口句柄并非直接指向窗口对象，而是指向跟踪窗口对象地址的另一个指针。

销毁窗口同样也是复杂的，必须确保销毁了该窗口对象，以及封装该窗口对象的CWnd实例。虽然CWnd对象知道窗口对象，但是窗口对象并不知道CWnd对象(见图1-6)。

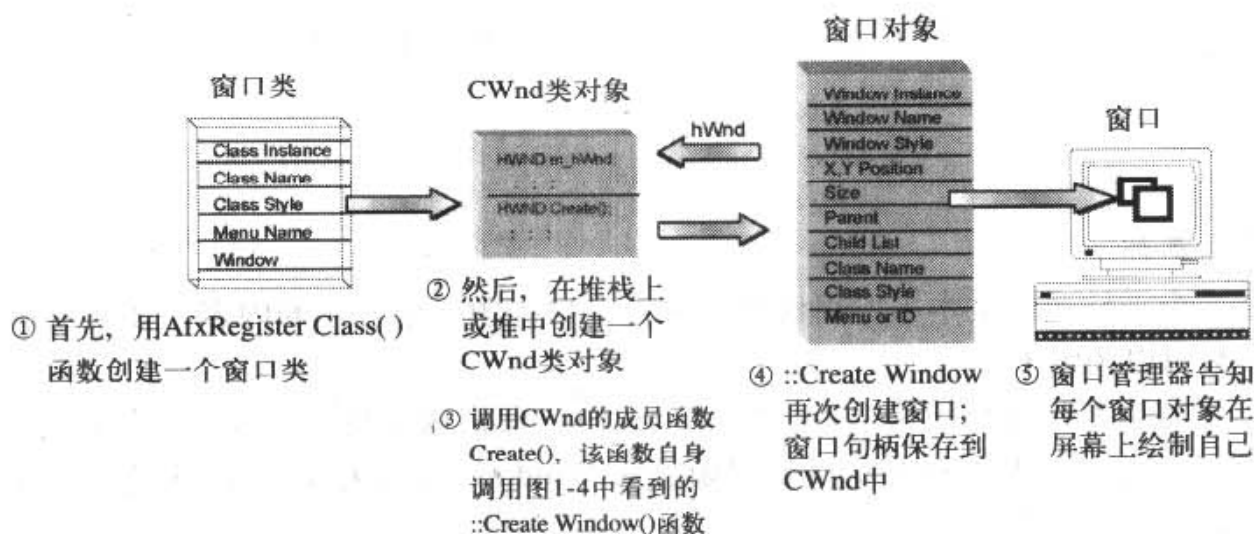


图1-6 应用两个对象创建MFC窗口

尽管窗口应用程序的用户界面可以由几十个、甚至几百个窗口构成, 但是大多数窗口还是由不到十个的窗口类创建的。即使在应用程序中有一千个窗口, 每个窗口也只能是三种基本类型之一: 重叠窗口、弹出窗口或子窗口。

1.3 怎样应用MFC创建一个窗口

可以用MFC的CWnd类创建一个窗口:

```
CWnd wnd;
BOOL b=wnd.CreateEx (ExStyle, ClassName, WindowName, Style, x, y, Width, Height, Parent, Menu, Param);
```

第一行代码创建了一个CWnd类的对象, 第二行通过调用Windows API函数CreateWindowEx() 创建了真正的窗口。

```
HWND hwnd=::CreateWindowEx (ExStyle, ClassName, WindowName, Style, x, y, Width, Height, Parent, Menu, Instance, Param);
```

因为CWnd类只是封装了用于创建窗口的Windows API函数(CreateWindowEx()), 因此, 从本质上讲, 创建窗口所必须的参数在API环境和MFC环境中是相同的:

- 参数Style和ExStyle决定窗口的“外观”和类型(重叠、弹出、子窗口)。
- 参数ClassName决定在创建窗口时使用的窗口类。
- 参数WindowName决定窗口标题内容(如果窗口标题有内容)。
- 参数x, y, Width和Height决定窗口的位置和大小。
- 参数Parent指向拥有该窗口的窗口指针(如果有这样的窗口)。
- 参数Menu指向内存中的一个对象, 作为它的菜单使用——除非创建一个子窗口, 如果是这样的话, 那么该参数是帮助父窗口识别子窗口的IDnumber。
- 参数Instance识别该窗口属于哪个应用程序, 以便发送到该窗口的消息能被发送到正确的应用程序的消息队列中。CWnd类填入Instance参数。
- 参数Param是在创建窗口时由窗口类使用的指针, 该指针是指向附加信息的非强制性结构的指针。

返回的hwnd参数是指向创建的窗口对象的指针, 但在未创建任何窗口时, 该参数返回值为NULL。窗口句柄自动地保存在CWnd类的m_hWnd成员变量中, 这在前面的图1-6中可以看到。

现在既然已了解了有关创建窗口的基本知识,那么让我们进一步来看看填写这些参数的规则。

规则

1. 窗口名称参数

该参数是一个零结尾的文本串,用该串指明在窗口标题栏中显示的内容。如果窗口没有标题栏,该参数可以为零(0)。

然而,某些通用控件也使用该参数。例如,按钮控件把该参数的内容放在按钮的表面。在创建窗口标题栏后,可以用类CWnd的成员函数SetWindowText()来改变窗口标题栏的名称。

2. 风格和扩展风格参数

这两个是32位的参数,用来指定创建什么类型的窗口。可以选择多种类型,如下面的例子所示:

,WS_CHILD|WS_VISIBLE,

窗口风格参数的作用包括:

- 用于创建三种基本窗口风格的风格,用WM_CHILD创建一个子窗口;用WM_POPUP创建一个弹出窗口。
- WM_OVERLAPPED创建一个重叠窗口。如果不为窗口指定任何一种风格,那么窗口风格默认为WM_OVERLAPPED。
- 用以增添窗口的非客户区特色的风格。例如,可以用WS_VSCROLL为窗口添加一个垂直滚动条,如图1-7所示其他非客户区窗口风格。

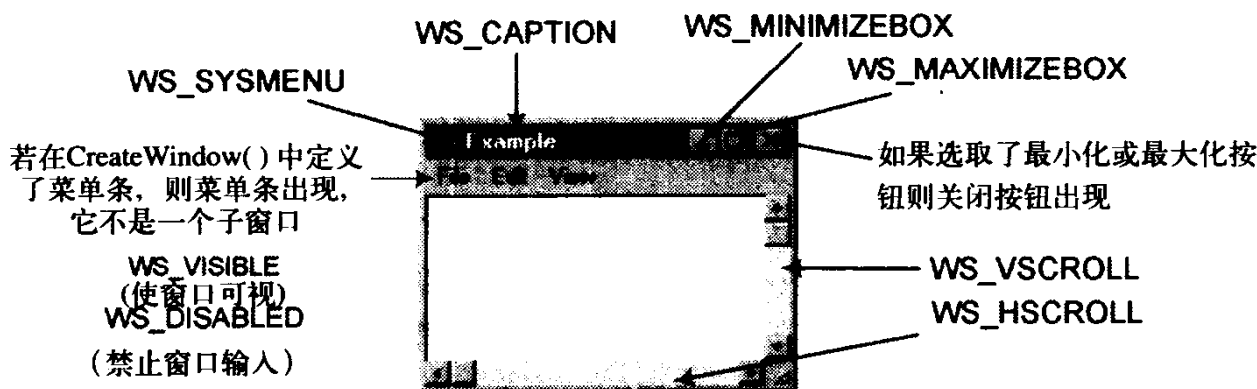


图1-7 非客户窗口风格用来增添窗口的特色

- 由每个通用控件定义的风格。例如,BS_FLAT按钮风格告诉按钮控件绘制一个二维按钮。
- 用来使窗口可视、可操作和/或初始最大化的风格。
- 用来标识控件组起始控件的风格,或者当用户敲击Tab键控制焦点在窗口中变化时,用来指示哪个控件有资格获得输入焦点的风格。
- 想了解更多有关窗口风格的例子,请参阅附录A。

在创建窗口以后,可以用CWnd的成员函数ModifyStyle()和ModifyStyleEx()改变窗口风格。某些风格可能要求重画窗口,这时,可以给ModifyStyle()函数添加第三个参数,自动激发类CWnd的成员函数SetWindowPos()做重画窗口工作。

```
CWnd wnd;
wnd.ModifyStyle(0,WS_BORDER,SWP_NOZORDER);
```

事实上,给ModifyStyle()或ModifyStyleEx()添加任何第三个参数,都需要添加下面的SetWindowPos()选项:SWP_NOZORDER、SWP_NOMOVE、SWP_NOACTIVATE和SWP_SHOW。

注意 有时候,甚至一个重画的窗口可能与新风格不相容,尤其对通用控件窗口来说更是这样。因为单个窗口决定重画窗口时使用什么风格。在这种情况下,唯一的办法是:在事先存储已有窗口的风格和其他参数后,销毁和重建该窗口。

3. X和Y位置参数

这是两个32位的参数,用于以像素为单位指定窗口的位置。创建重叠窗口和弹出窗口时,X和Y是相对于桌面窗口左上角的位置。而创建子窗口时,X和Y是相对于父窗口客户区的左上角位置。如果把X和Y参数都设置为CW_USEDEFAULT,那么系统将自动为窗口选定一个位置。系统层叠排列这些新窗口(见图1-8)。



图1-8 CW_USEDEFAULT允许系统自动设定窗口位置

然而,如果X和Y参数都设置为CW_USEDEFAULT,那么子窗口在创建的时候,总被创建在(0,0)的位置。

创建窗口以后,可以用类CWnd的成员函数MoveWindow()移动窗口。

4. 宽度和高度参数

这是两个32位的参数,用于以像素为单位指定窗口的大小。如果将参数Width和Height都设置为CW_USEDEFAULT,则系统将根据桌面窗口的大小,自动选定窗口的大小。然而,对于一个子窗口来说,系统将创建一个长和宽均为0的窗口。如果窗口的风格是WS_MINIMIZE或WS_MAXIMIZE,那么系统将忽视用户为Width和Height设置的任何值。

创建窗口以后,可以用类CWnd的成员函数MoveWindow()重新设置窗口的大小。

5. Z-Order

当几个窗口占据屏幕上同一区域时,Z-Order决定哪个窗口显示在哪个窗口之上。Z-Order(Z顺序)中的Z来源于坐标X-Y-Z轴的Z轴,其中Z轴垂直屏幕,并由屏幕指向外面。当窗口最初被创建或选中时,则窗口将出现在Z-Order的顶层。然而,该窗口永远不可能出现在一个最顶层窗口的上面,除非该窗口也是最顶层的窗口。“最顶层”窗口用WS_EX_TOPMOST窗口风格创建,并显示在所有非最顶层窗口的上面,而与该窗口是不是当前选中的窗口无关。

创建窗口后,可以用CWnd的成员函数SetWindowPos()改变窗口的Z顺序。

6. 父窗口或物主窗口参数

该参数是指向类CWnd对象的指针,根据创建的窗口类型标识是父窗口还是物主窗口

(Owner):

- 如果创建的是一个子窗口，那么用该参数来标识它的父窗口(该子窗口所放置的并为之所截断的窗口)。该值不能为NULL。子窗口只能出现在它的父窗口里面，当父窗口被销毁时它们也被销毁，并且当父窗口被隐藏或最小化时它们也被隐藏。
- 如果创建的是重叠窗口或弹出窗口，用该参数来标识物主窗口。如果该值为NULL，则桌面窗口成为物主窗口。从属窗口总是显示在它们的主窗口上面，并且随着物主窗口的销毁而被销毁；物主窗口最小化时，则从属窗口被隐藏；但当物主窗口被隐藏时，从属窗口并不被隐藏。

一个子窗口可能是另一个子窗口的父窗口，但绝不可能是一个物主窗口。如果试图使一个子窗口成为一个物主窗口，那么系统只能使那个子窗口的最顶层窗口作为物主窗口。图1-9是这些关系的概述。

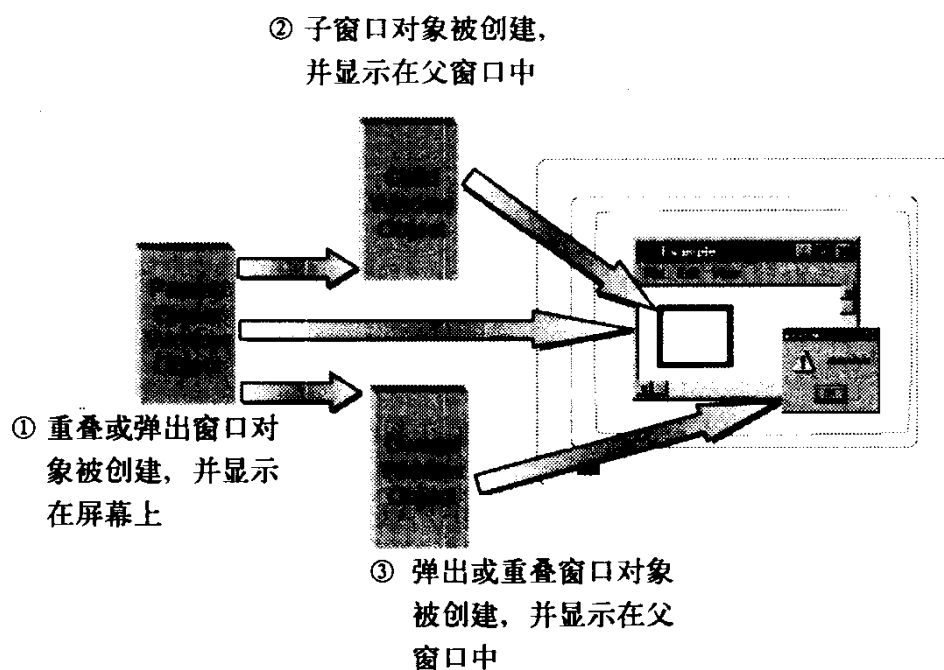


图1-9 物主、父、子窗口的层次关系

可以用CWnd的成员函数SetOwner()改变已有窗口的物主窗口，用CWnd的成员函数SetParent()改变父窗口。

7. 菜单或控件ID参数

该参数用来标识菜单(Menu)句柄或控件(Control)ID，这要依赖于创建的窗口是子窗口、重叠窗口，还是弹出窗口。

- 如果创建的是一个子窗口，用该参数标识控件ID，控件ID通常用来帮助父窗口识别子窗口。因为该参数寻求一个HMENU变量，因此，需要用HMENU类型重载变量定义控件ID。例如，如果子窗口的控件ID是102，应该用如下方法定义它：

..., (HMENU) 102, ...

- 如果创建的是一个重叠窗口或弹出窗口，用该参数定义窗口的菜单。若该值为NULL，则菜单默认为在该窗口的窗口类中定义的任何菜单；如果窗口类也没有指定的菜单，则该窗口在创建时没有菜单。可以用下面的方法为该参数加载一个菜单：

HMENU hMenu=::LoadMenu(AfxGetInstanceHandle(), MAKEINTRESOURCE (xx));

这里的xx是应用程序资源里面的菜单ID(Menu ID)。

可以用CWnd的SetDlgCtrlID()改变已有子窗口的ID。用类CWnd的成员函数SetMenu()改变一个重叠窗口或弹出窗口的菜单。

8. 实例

前面已提到, CWnd类将填入该参数。CWnd通过调用AfxGetInstanceHandle()获得该实例(Instance)。一个应用程序的实例从本质上标识了哪个程序在内存里。AfxGetInstanceHandle()是由MFC库提供的静态函数。

9. 参数

该32位参数(Parameter)是可选的。它通常是指向一个结构或者类对象的指针, 而该结构或者类对象是创建某种类型的窗口时需要提供的。例如, 当用MDICLIENT窗口类创建窗口时, 需要该参数提供一个CLIENTCREATESTRUCT结构的指针。

10. 类名参数

类名(Class Name)参数是一个零结尾字符串, 当创建一个窗口时, 用来标识使用那个窗口类。关于窗口类和窗口处理, 将在本章后面的内容中详细讨论。该参数不能为NULL, 在创建一个非常一般的MFC窗口时, 使用AfxRegisterWndClass (0)填入该参数。

1.4 怎样使用MFC销毁一个窗口

如前面所提到的一样, 删除一个MFC窗口可能有些烦琐, 必须按下面的顺序删除两个对象:

```
pWnd->DestroyWindow( ); //destroys Window Object
delete pWnd;             //destroys Cwnd Object
```

也可以只删除CWnd对象, 因为DestroyWindow()是从CWnd的析构函数中调用的, 但不提倡使用这种方法。销毁一个窗口而不先调用DestroyWindow()函数, 将使某些析构消息不能被任何从CWnd派生的类处理。你几乎不需自己销毁一个窗口, 用户或系统通常会做这项工作。如果需要在销毁窗口的同时销毁CWnd对象, 则应该在CWnd的派生类中添加下面的重载函数:

```
CYourWnd::PostNcDestroy( )
{
    delete this;
}
```

PostNcDestroy()是销毁窗口前调用的最后一个成员函数。但是, 几乎不需对该函数进行重载, 因为CWnd和派生类一般是嵌在另一个类中或建立在堆栈中的。

1.4.1 捆绑到一个已有的窗口

如果一个窗口是用Windows API在应用程序建立之前或外面创建的, 并且需要把它封装到CWnd类中, 可以用下面的语句实现:

```
CWnd wnd;
wnd.Attach (hwnd);
```

这里的hwnd是已有窗口的句柄。Attach()只是把CWnd的成员变量m_hWnd赋给hwnd。

也可以使用CWnd::FromHandle (hwnd), 它查看是否有CWnd对象已经封装了该窗口句柄并返回该CWnd对象的指针。如果不存在这样的CWnd对象, 将创建一个临时对象。如果是临

时对象，则不要使用该指针，因为在应用程序空闲的时候，该指针指向的对象将被删除。

1.4.2 窗口类

前面已提及，一个窗口类不是一个C++类，而是早于并存在于C++之外的窗口专有的类。窗口类的作用就像一个模板，可以由此创建其他窗口，并可共享某些特征，包括下面所示的特征：

类风格 包括能给予窗口的一些微小的特征。

类图标 如果窗口有图标，用它来指定在窗口的左上角处所画的图标。

类菜单 如果窗口有菜单，用它来指定窗口中显示的菜单。

类光标 当光标通过窗口的客户区时，用它来指定显示哪种鼠标形状。

背景色 定义用什么颜色来擦除窗口的背景色。窗口的客户区将显示该颜色。

窗口进程 定义处理任何发送到该窗口的消息时应该调用的函数。定义窗口进程可能是窗口类唯一最重要的前提。

1.4.3 窗口进程

窗口与环境的交互是通过发送和接收消息来实现的。如果系统要求窗口自己进行绘制，系统给它发送一个WM_PAINT消息；如果系统要求窗口销毁自己，则发送一个WM_DESTROY消息。这些消息都由窗口的窗口进程处理，该窗口进程的地址在窗口类中定义。对于发送到由相同的窗口类创建的窗口的消息，系统采用完全相同的窗口进程进行处理。相同的窗口进程是怎样跟踪所有的窗口的呢？它是怎么知道窗口A画在(10,34)，而窗口B画在(56,21)呢？所有这些工作的完成只需使用窗口的窗口对象。

例如，所有用按钮窗口类创建的窗口，都使用相同的按钮窗口进程。如果一个WM_PAINT消息发送到其中任何一个窗口，则按钮窗口进程根据每个窗口的窗口对象指定的大小、位置和风格画出确切的按钮(见图1-10)。

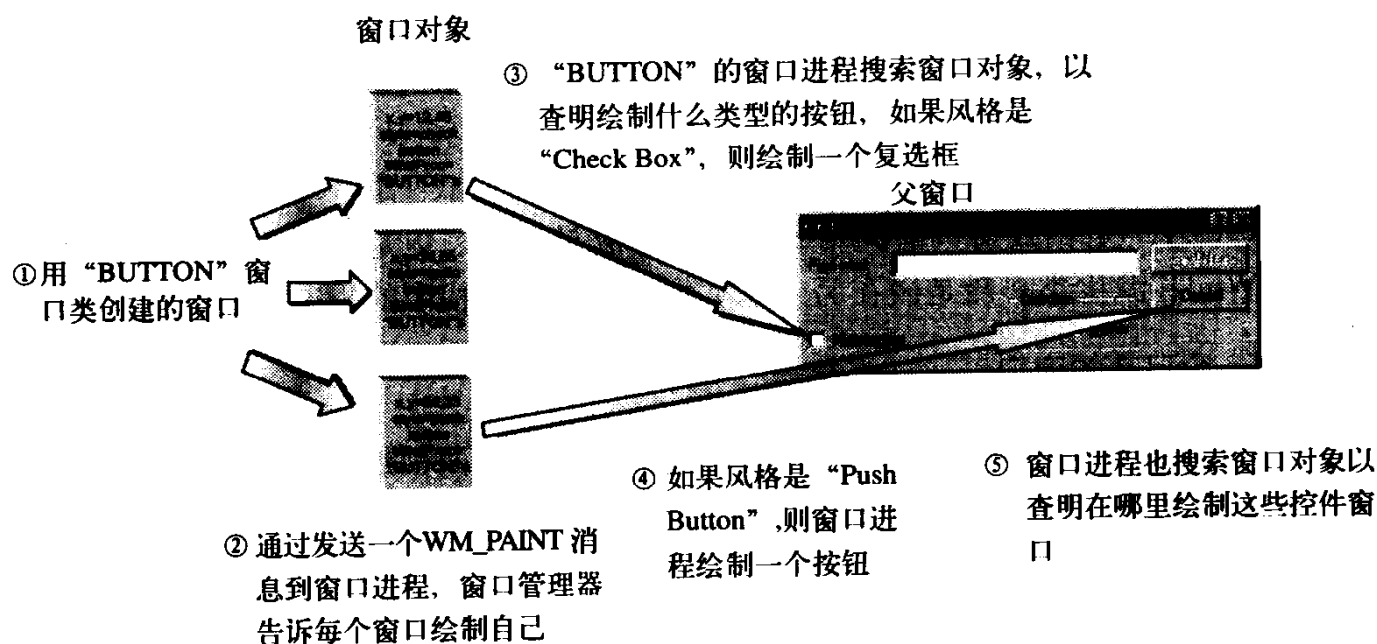


图1-10 按钮窗口进程使用窗口对象指定对哪个窗口进行操纵

1.5 怎样使用MFC创建一个窗口类

当用户创建一个窗口类时，实际上只是在三个操作系统列表之一中注册一个WNDCLASS结构。系统为每个窗口类类型维护一个列表：

系统全局类(System Global Class) 在系统启动时注册，且必须注册，对所有应用程序都有效。

应用程序全局类(Application Global Class) 由应用程序注册，只对应用程序及应用程序的线程有效。

应用程序局部类(Application Local Class) 由应用程序注册，并只对注册它们的应用程序或DLL有效。

当系统搜索一个窗口类时，从应用程序局部类开始，然后搜索应用程序全局类，最后搜索系统全局类。

要创建一个窗口类，可以先创建WNDCLASS结构的实例，然后用MFC类库的AfxRegisterClass()注册它。也可以用MFC类库的AfxRegisterWndClass()来创建一个基于调用参数的WNDCLASS对象。

1.5.1 使用AfxRegisterWndClass()函数注册一个窗口类

AfxRegisterWndClass()函数在使用上是非常自动化的，一些通常需要你提供的参数都能自己填入，甚至连新的窗口类的名字也能自动产生。

```
LPCTSTR lpszClassName=AfxRegisterWndClass(UINT nClassStyle, HCURSOR hCursor=0,HBRUSH hbrBackground=0, HICON hIcon=0);
```

这些参数的使用规则如下：

1. 类名

根据传给该函数的参数，为新的窗口类产生名字。如果传输的参数完全相同，那么创建的窗口类也将完全相同。如果需要创建新的窗口类，可以用AfxRegisterClass()。

2. 风格

窗口类风格由下面选项列表中一系列标记的或(OR)运算提供：

类 风 格	描 述
CS_OWNDC	为该窗口类创建的每个窗口分配唯一的设备环境。有关设备环境的更详尽资料见第4章
CS_PARENTDC	从系统高速缓存中检索设备环境，然后设置该设备环境的剪裁区，以组合到父窗口中，以便子窗口能画在父窗口上
CS_CLASSDC	分配一个设备环境给所有由该窗口类创建的窗口使用
CS_SAVEBITS	由该类创建的任何窗口的视频存储区将被保存，以便窗口移动或关闭时，不需要重画基础窗口——这对快速机器来说作用不大
CS_BYTEALIGNCLIENT	当计算机的显示卡和CPU速度较慢时，这两种风格有用。添加这些风格后，当窗口在字节边缘时，显示卡更容易移动窗口
CS_BYTEALIGNWINDOW	
CS_GLOBALCLASS	如果设置该风格，则该类是应用程序的全局类，否则它是一个应用程序局部类
CS_VREDRAW	如果设置了垂直风格，并且窗口的垂直大小发生了变化，则整个窗口被重画。
CS_HREDRAW	水平风格也这样
CS_NOCLOSE	禁用系统菜单中的关闭命令
CS_DBLCLKS	如果未设置该参数，并且双击由该窗口类创建的窗口，则传送给应用程序的将不是双击事件，而是两个相继完成的单击事件

3. 图标(Icon)

该参数是显示在窗口左上角的图标的句柄，但只适用于使用WS_SYSMENU的窗口风格。应用程序主窗口的图标也显示在任务栏上。如果将该参数设置为NULL，并且设置了WS_SYSMENU风格，则系统将提供一个默认的图标。在MFC环境中，绝大部分图标已被处理过，用CWnd的SetIcon()可改变已有的图标。

4. 光标(Cursor)

该参数是鼠标移经应用程序窗口的客户区时，将要显示的鼠标光标句柄。如果将该参数设置为NULL，则显示的是箭头光标。可以用下面的语句装载一个光标：

```
HICON hIcon=AfxGetApp()->LoadCursor (xx);
```

这里的xx是应用程序资源中光标的名字或ID。

这里指定的光标意味着是该窗口的默认光标。如果想动态地改变光标形状，则应该处理该窗口的WM_SETCURSOR消息，并用SetCursor()来改变光标形状(见第8章例33)。

5. 背景色

当系统创建窗口时，先在显示窗口的地方画一个矩形区域，以擦除该区域的背景色。该参数指定填充该矩形域时所用画刷的句柄(参见第4章有关画刷的详细内容)。为窗口类创建的画刷对象在该类退出注册时被自动释放。

在指定背景色时，也可以不创建画刷对象，而指定下面所列系统颜色之一：

```
COLOR_ACTIVEBORDER
COLOR_ACTIVECAPTION
COLOR_APPWORKSPACE
COLOR_BACKGROUND
COLOR_BTNFACE
COLOR_BTNSHADOW
COLOR_BTNTEXT
COLOR_CAPTIONTEXT
COLOR_GRAYTEXT
COLOR_HIGHLIGHT
COLOR_HIGHLIGHTTEXT
COLOR_INACTIVEBORDER
COLOR_INACTIVECAPTION
COLOR_MENU
COLOR_MENUTEXT
COLOR_SCROLLBAR
COLOR_WINDOW
COLOR_WINDOWFRAME
COLOR_WINDOWTEXT
```

然而，必须分配颜色到一个画刷句柄类型，并加1。
(HBRUSH) (COLOR_WINDOW+1)

设置该参数为NULL，则在画一个窗口之前，系统不对窗口进行擦除。在非客户区的绘制还是同平常一样，但客户区将保持窗口被画前原来屏幕所显示的样子。若该参数设为NULL，应确认窗口是画全部客户区，还是处理WM_ERASEBKGND消息以擦除背景颜色。

1.5.2 使用AfxRegisterClass() 函数创建一个窗口类

如果想完全控制一个新窗口类的创建，如指定窗口类的名字，则应该用下面的语句：