# 目　　录

无忧书库 http://book.51soft.com

前言

第一章　Visual C++ 6.0概述 …… 1
1.1　Visual C++ 6.0的特性简介 …… 1
1.2　Visual C++ 6.0的软、硬件配置要求 …… 2
1.3　Visual C++ 6.0的安装 …… 3

第二章　Visual C++ 6.0开发环境 …… 8
2.1　Visual C++ 6.0主窗口 …… 9
2.2　Visual C++ 6.0菜单栏 …… 9
2.3　Visual C++ 6.0工具条 …… 26
2.4　项目与项目工作区 …… 28
2.5　资源与资源编辑器 …… 30
2.6　本章小结 …… 45

第三章　"Hello World!"应用程序 …… 46
3.1　应用程序的框架 …… 46
3.2　应用框架示例 …… 46
3.3　MFC库消息映射 …… 49
3.4　"Hello world"应用程序 …… 50
3.5　本章小结 …… 60

第四章　图形设备接口(GDI) …… 61
4.1　设备描述表 …… 61
4.2　GDI对象 …… 64
4.3　绘制基本图形 …… 69
4.4　GDI的示例程序 …… 78
4.5　本章小结 …… 83

第五章　鼠标、键盘及计时器 …… 84
5.1　鼠标 …… 84
5.2　键盘 …… 95

5.3 计时器 …… 104
5.4 本章小结 …… 108

**第六章 菜单** …… 109
6.1 概述 …… 109
6.2 普通下拉菜单 …… 114
6.3 上下文菜单 …… 118
6.4 改变菜单的显示状态 …… 122
6.5 动态地增、删和修改菜单项 …… 127
6.6 菜单间的切换 …… 136
6.7 本章小结 …… 138

**第七章 工具条** …… 139
7.1 概述 …… 139
7.2 制作普通的工具条 …… 145
7.3 制作高级工具条 …… 151
7.4 本章小结 …… 162

**第八章 状态条** …… 163
8.1 概述 …… 163
8.2 创建一个普通的状态条 …… 166
8.3 创建一个高级的状态条 …… 168
8.4 本章小结 …… 178

**第九章 对话框** …… 179
9.1 概述 …… 179
9.2 系统预制对话框的制作 …… 187
9.3 普通对话框的制作 …… 190
9.4 高级对话框的制作 …… 193
9.5 无模式对话框的制作 …… 203
9.6 属性对话框的制作 …… 210
9.7 对话框条 …… 221
9.8 基于对话框的应用程序 …… 225
9.9 本章小结 …… 228

**第十章 切分窗口与多视** …… 229
10.1 切分窗口 …… 229

10.2 多视选择 …… 229
10.3 动态和静态切分窗口 …… 230
10.4 单视类的 SDI 动态切分窗口示例 …… 231
10.5 双视类的 SDI 静态切分示例 …… 233
10.6 MDI 中的切分窗口示例 …… 237
10.7 本章小结 …… 241

**第十一章 文档与视的分离** …… 242
11.1 文档－视之间的相互作用函数 …… 242
11.2 最简单的文档、视应用 …… 245
11.3 CFormView 类 …… 246
11.4 文档与视的示例程序 …… 246
11.5 本章小结 …… 254

**第十二章 一个小型的绘图系统** …… 255
12.1 编制 Windows 程序的总体思路 …… 257
12.2 该应用程序的设计思路 …… 257
12.3 应用程序界面部分的制作 …… 291
12.4 应用程序功能的实现 …… 304
12.5 应用程序的源程序 …… 305
12.6 本章小结 …… 459

**附录 二维图形的几何变换** …… 460

**CD－ROM 的内容** …… 463

# 第一章　Visual C++ 6.0概述



- Visual C++ 6.0的特性简介
- Visual C++ 6.0的软、硬件配置要求
- Visual C++ 6.0的安装

随着计算机技术的飞速发展，软件编程概念已经发生了根本性的转变，应运而生的可视化编程技术获得了广泛应用。Visual C++ 6.0即是可视化编程的有力工具之一。该软件是Microsoft公司推出的Visual Studio的重要组成元素，是集创建编辑、编译、链接、运行调试为一体的软件集成开发环境。

## 1.1　Visual C++ 6.0的特性简介

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。

### 1.1.1　Visual C++ 6.0开发环境

Visual C++ 6.0开发环境Developer Studio是在Windows 98/NT环境下运行的一套集成工具，由文本编辑、资源编辑器、项目建立工具、优化编译器、增量连接器、源代码浏览器、集成调试器等组成。

使用Developer Studio，不仅可以创建由Visual C++ 6.0使用的源文件和其他文档，而且可以创建、查看和编写任何与ActiveX部件有关的文档(ActiveX文档)。

在Developer Studio中，可以在项目工作区中组织文件(file)、项目(project)和子项目。可以使用工作区窗口来查看和访问项目中的各种元素。项目工作区可以含有多个项目，每个项目要么是顶层项目，要么是其他项目的子项目。

关于Visual C++ 6.0的开发环境，我们将在第二章作详细的介绍。

### 1.1.2　向导(Wizard)

在Visual C++ 6.0中，可以使用向导、MFC类库和活动模板库(ATL)来开发Windows应用程序。

向导用于帮助用户生成各种不同类型应用程序的基本框架。例如，可以使用MFC AppWizard来生成完整的从开始文件出发的基于MFC类库的源文件和资源文件；可以使用

MFC ActiveX Control Wizard 生成创建 ActiveX 控制所需要的全部开始文件(如源文件、头文件、资源文件、模块定义文件、项目文件、对象描述语言文件等);可以使用 ISAPI Extension Wizard 生成创建 Internet 服务器(Server)或过滤器(Filter)所需要的全部文件;使用 ATL COM AppWizard 来创建 ATL 应用程序;使用 Custom AppWizard 来创建自定义的项目类型,并将其添加到创建项目时的可用项目类型列表中。

在创建应用程序的基本框架后,可以使用 ClassWizard 来创建新类,定义消息处理函数,覆盖虚拟函数,从对话框、表单视图或者记录视图的控件中获取数据并验证数据的合法性,在自动化对象中添加属性、事件和方法。此外,还可以使用 WizardBar 来定义消息处理函数、覆盖虚函数并浏览实现文件(.CPP)。

### 1.1.3 最快的集成数据库访问

Visual C++ 6.0 允许用户建立强有力的数据库应用程序:

·可以使用 ODBC 类和高性能的 32 位 ODBC 驱动程序来访问各种数据库管理系统。

·可以使用 DAO(数据访问对象)类通过编程语言来访问和操纵数据中的数据并管理数据库、数据库对象与结构。

### 1.1.4 强有力的 Internet 支持

Visual C++ 6.0 对 Internet 提供了强有力的支持:

- Win32 Internet API(WinInet)使 Internet 成为应用程序的一部分并简化了对 Internet 服务(FTP、HTTP 和 Gopher)的访问。
- ActiveX 控件可以用在 Internet 和桌面应用程序中。
- ActiveX 文档可以显示在整个 Web 浏览器(如 Internet Explorer)或 OLE 容器(如 Microsoft Office Binder)的整个客户窗口中。
- Asynchronous Monikers 使应用程序可以异步下载文件和控件属性。
- 可以使用 CHttpServer、CHttpFilter、CHttpServerContext、CHttpFilterContext 和 CHttpStream 类来创建动态链接库,以便把功能添加到 Internet 服务器和 Web 页中。

## 1.2 Visual C++ 6.0 的软、硬件配置要求

Visual C++ 6.0 运行所需的软、硬件配置应满足以下要求:

- Windows 95、Windows 98 或 Windows NT。
- IBM PC 及其兼容机,最好具有 80486 以上的微处理器。
- 16MB 以上内存,建议使用 32MB 内存。
- 安装该应用程序大约需要 300 多兆字节空间。
- 高密软盘驱动器。
- VGA 显示器。
- CD-ROM 驱动器(用于联机信息)。

## 1.3 Visual C++ 6.0的安装

在使用 Visual C++ 6.0 开发应用程序之前，必须先将 Visual C++ 6.0 和 Developer Studio 安装到计算机上。可以使用安装程序进行新的安装或更新已有的安装。如果是升级安装，安装程序在安装过程中仅仅复制新的或更新过的文件。此外，安装程序还可以根据配置情况对安装进行裁剪。

在 Windows 98 中安装 Visual C++ 6.0 企业版(Enterprise Edition)的过程如下：

(1) 打开计算机启动 Windows 98。

(2) 在 CD-ROM 驱动器中插入 Visual C++ 6.0 系统光盘。

(3) 单击“开始”按钮，从开始菜单选择“运行”命令，弹出“运行”对话框。在“打开”文本框键入以下命令(图 1.1)：

```
E:\Setup.exe
```

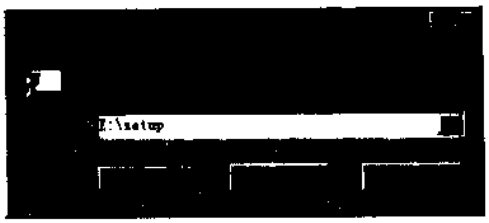


图 1.1 在“运行”对话框中键入安装程序的名称

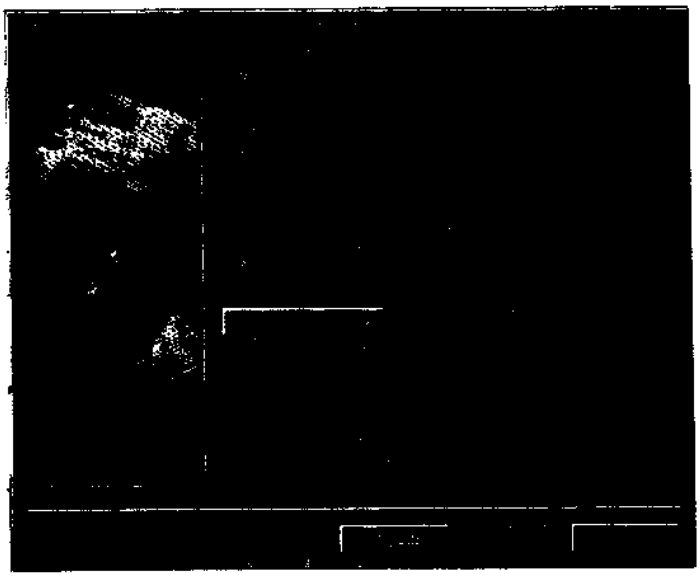


图 1.2 Visual Studio 6.0 Enterprise Edition 对话框

(4)弹出 Visual Studio 6.0 Enterprise Edition 对话框(图 1.2),单击 Next 按钮。

(5)弹出 End User License Agreement 对话框(图 1.3),选择 I accept the agreement 单选按钮,再单击 Next > 按钮。

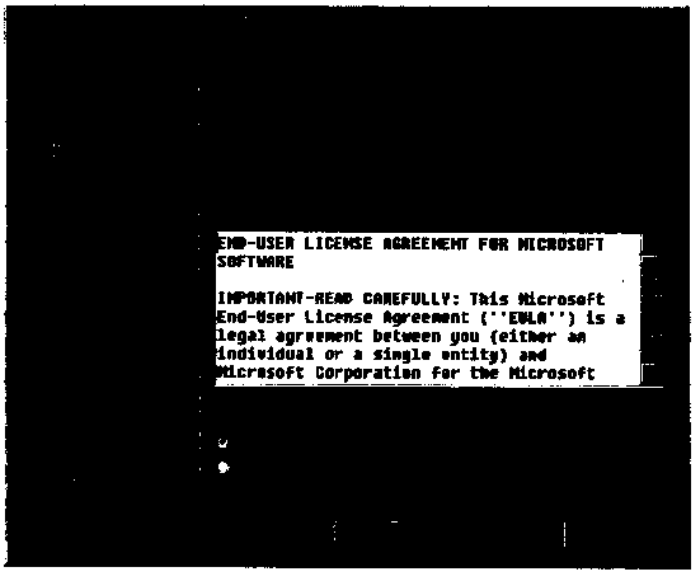



图 1.3 End User License Agreement 对话框

(6)弹出 Product Number and User ID 对话框(图 1.4),输入 ID 号、用户名称和公司名称,再单击 Next > 按钮。

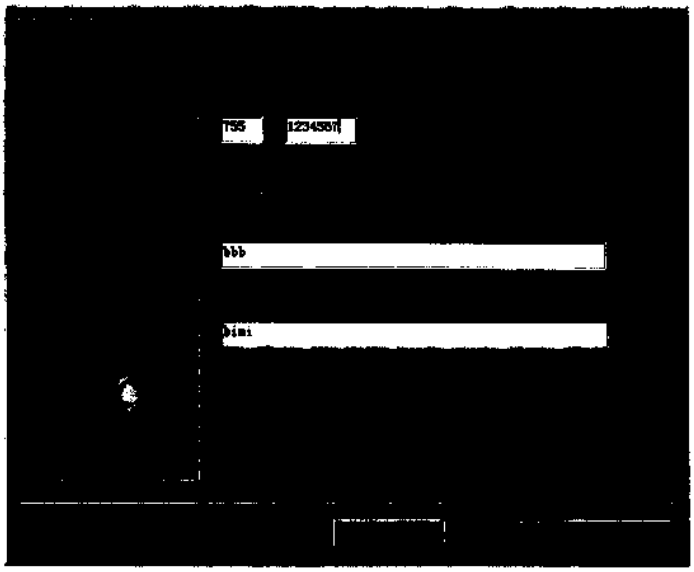



图 1.4 Product Number and User ID 对话框

(7)弹出 Visual Studio 6.0 Enterprise Edition 对话框(图 1.5),单击 Next > 按钮。

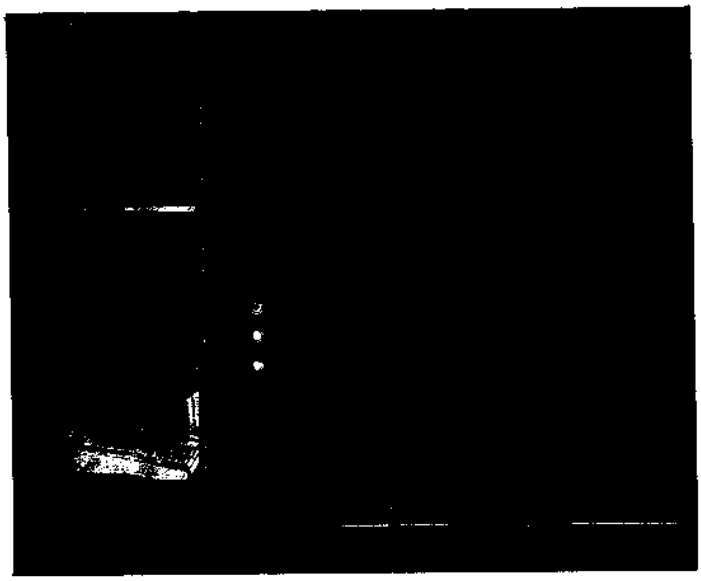


图 1.5　Visual Studio 6.0 Enterprise Edition 对话框

(8)弹出 Choose Common Install Folder 对话框(图 1.6),用户可以选择自己想要安装的文件夹。之后,单击 Next > 按钮。

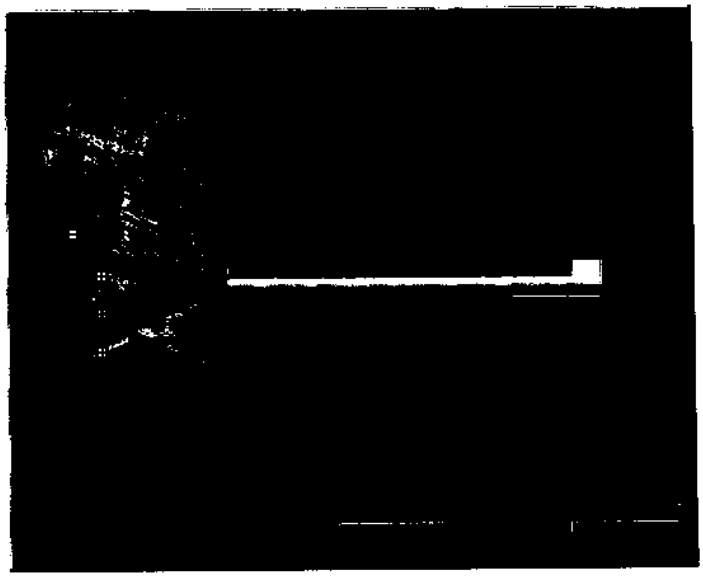


图 1.6　Choose Common Install Folder 对话框

(9)等待片刻后,弹出 Visual Studio 6.0 Enterprise Setup 对话框(图 1.7),选择 Continue 按钮。

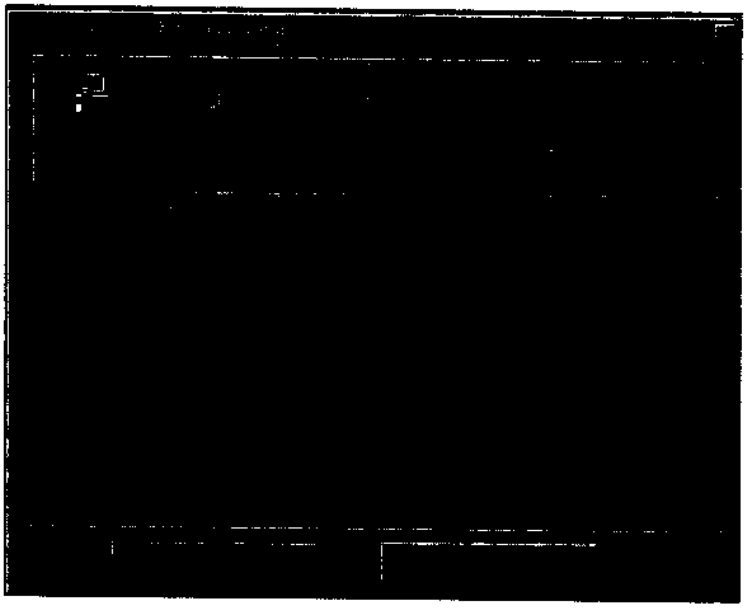


图 1.7 Visual Studio 6.0 Enterprise Setup 对话框

(10)弹出 Visual Studio 6.0 Enterprise Setup 对话框(图 1.8),单击 OK 按钮。

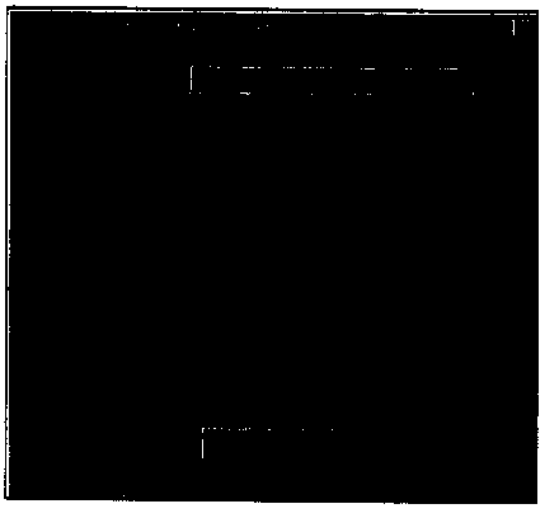


图 1.8 Visual Studio 6.0 Enterprise Setup 对话框

(11)等待一阵子后,弹出 Visual Studio 6.0 Enterprise - Custom 对话框(图 1.9),仅选择 Microsoft Visual C++6.0 和 Data Access 选项后,单击 Continue 按钮。由此便开始安装,屏幕上出现安装进度指示对话框。

(12)安装完组件后,便进入 Install MSDN 对话框(图 1.10)。如果用户想进一步安装 Visual C++ 6.0 所带的开发文档和例子,则单击 Next > 按钮进一步安装。否则,单击 Exit 按

钮退出安装。现在,我们选择 Exit 按钮,退出安装。

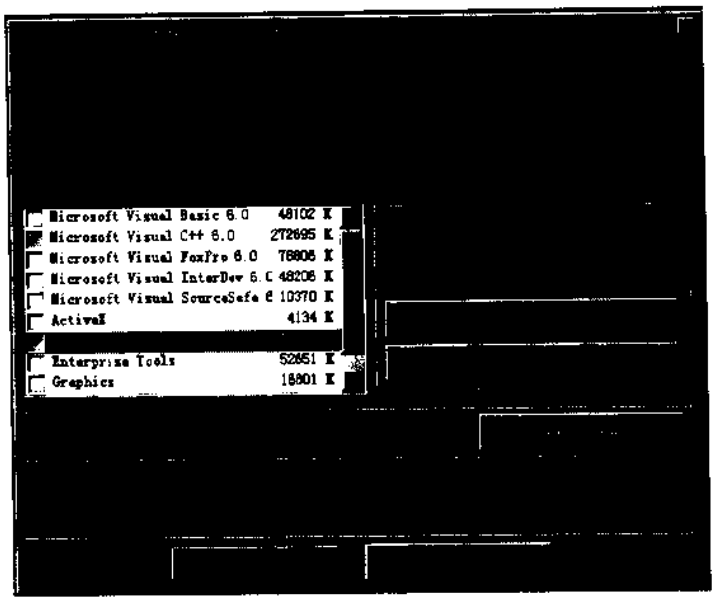



图 1.9　Visual Studio 6.0 Enterprise – Custom 对话框

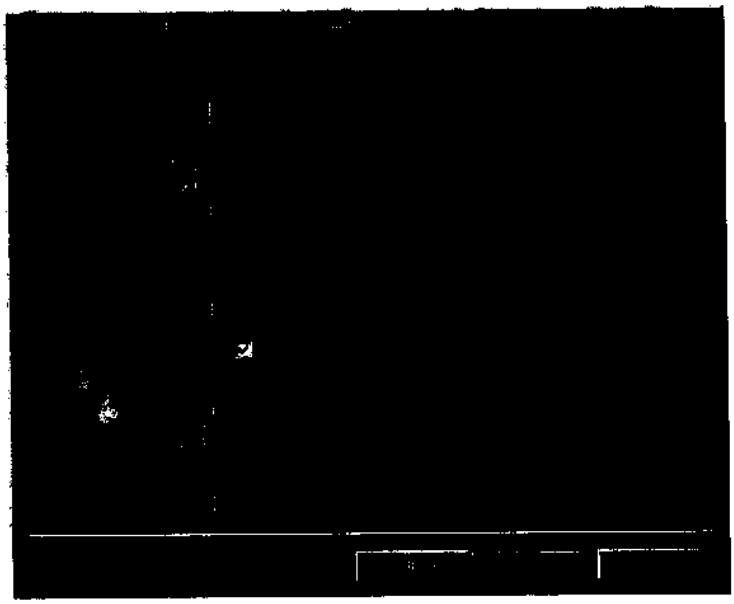


图 1.10　Install MSDN 对话框

至此，Visual C++ 6.0 即安装完成，整个过程大约需要 15～30 分钟。

# 第二章　Visual C++ 6.0 开发环境

- Visual C++ 6.0 主窗口
- Visual C++ 菜单栏
- Visual C++ 工具条
- 项目与项目工作区
- 资源与资源编辑器

Visual C++ 6.0 开发环境由一套综合的开发工具所组成，它提供了良好的可视化编程环境。在该环境下，可以对 C 和C++ 程序进行各种操作，包括建立、打开、浏览、编辑、保存、编译、链接和调试等，而这些操作都可以通过鼠标单击工具按钮来完成，非常方便有效。本章将详细介绍 Visual C++ 6.0 开发环境的众多功能。

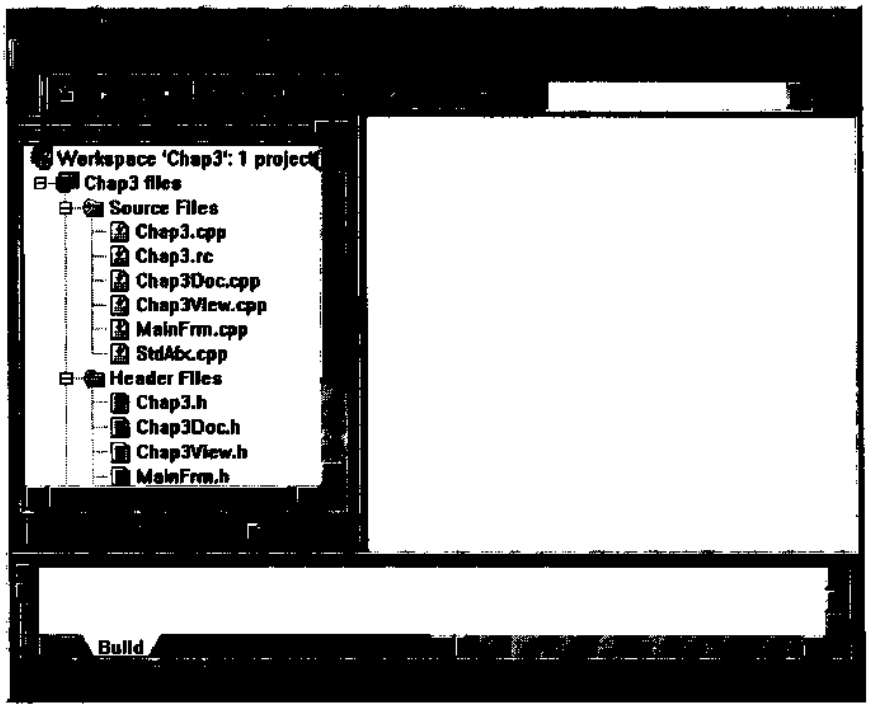



图 2.1　Visual C++ 6.0 开发环境

## 2.1 Visual C++ 6.0主窗口

首先，按第一章介绍的步骤将 Visual C++ 6.0 安装到 Windows 98 中。接着，单击“开始”按钮，从开始菜单启动 Visual C++ 6.0 进入开发环境 Developer Studio。打开一个项目后，Developer Studio 由标题栏、工具条、工作区窗口、源代码编辑窗口、输出窗口和状态栏组成(图 2.1)。

屏幕最上端是标题栏。标题栏用于显示应用程序名和所打开的文件名，标题栏的颜色用于表明对应窗口是否为激活的。标题栏左端为控制菜单框，是用于打开窗口控制菜单的图标。用鼠标单击该图标，将弹出窗口控制菜单。窗口控制菜单用于控制窗口的大小和位置，如还原、移动、关闭、最大化和最小化等。标题栏的右边有三个控制按钮，从左至右分别为最小化按钮、还原按钮和关闭按钮，这些按钮用于快速设置窗口大小。例如，使窗口填充整个屏幕、将窗口最小化为图标或关闭窗口。

标题栏的下面是菜单栏和工具条。工具条的下面是两个窗口，一个是工作区窗口，一个是源代码编辑窗口。工作区窗口的下面是输出窗口，用于显示项目建立过程中所产生的错误信息。屏幕最底端是状态栏，它给出当前操作或所选择命令的提示信息。

## 2.2 Visual C++ 6.0菜单栏

菜单栏由多个菜单组成。与 Windows 98 操作一致，选择菜单有两种方法，一种是用鼠标左键单击所选的菜单，另一种是键操作，即同时按下 Alt 键和所选菜单的热键字母(带下划线的字母，如File 中的 F)。选中某个菜单后，就会出现相应的下拉式子菜单。在下拉式子菜单中，有些菜单选项的右边对应着相应的加速键(如 Save 对应 Ctrl + S)，表示按加速键将直接执行菜单命令，这样可以避免进入多层菜单的麻烦。有些菜单选项后面带有三个圆点符(如 New...)，表示选择该项后将自动弹出一个对话框。有些菜单选项后面带有黑三角箭头(▶)，表示选择该项后将自动弹出级联菜单。若下拉式子菜单中的某些菜单选项显示为灰色，则表示这些选项在当前条件下不能选择。

此外，在窗口的不同位置单击鼠标右键将弹出上下文菜单，从上下文菜单可以执行与当前位置最为相关的要频繁执行的命令。

### 2.2.1 File菜单

在 File 菜单中包含有用于对文件进行操作的命令选项(图 2.2)。

File 菜单中各选项的含义，如表 2.1 所示。

表 2.1 File 菜单

| 菜单项 | 功能 |
| --- | --- |
| New | 可以创建新的文档、项目或者工作区 |
| Open | 打开已有的文件，如C++ 文件、宏文件、资源文件、项目文件等 |

（续表）

| 菜单项 | 功能 |
| --- | --- |
| Close | 关闭已打开的文件 |
| Open Workspace | 打开工作区文件 |
| Save Workspace | 保存打开的工作区 |
| Close Workspace | 关闭打开的工作区 |
| Save | 保存当前活动窗口或者选定窗口中的文件内容 |
| Save As | 将打开的文件用新的文件名加以保存 |
| Save All | 保存所有窗口内的文件内容 |
| Page Setup | 设置和格式化打印结果 |
| Print | 打印当前活动窗口中的内容 |
| Recent Files | 包含有最近打印的文件名 |
| Recent Workspaces | 包含有最近打开的工作区名 |
| Exit | 退出 Visual C++ 6.0开发环境 |

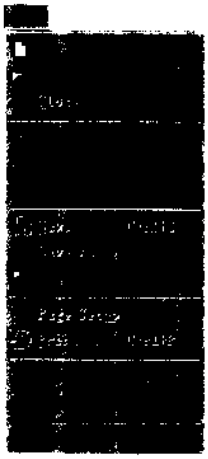


图 2.2 File菜单

下面，我们详细介绍 New 菜单项的用法。New 菜单项用于打开 New 对话框（图 2.3），它可以创建新的文档、项目或者工作区。

1.创建新的文件

要创建新的文件，从 Files 标签（图 2.4）单击要创建的文件类型，然后在 File 文本框中键入文件的名字。如果要添加新文件到已有的项目中，请选择 Add to project 复选框并选择项目名。

Visual C++ 6.0 可以创建的文件类型有：

·Active Server Page：创建活动服务器页。

·Binary File：创建二进制文件。

·Bitmap File：创建位图文件。

·C/C++ Header File：创建 C/C++ 头文件。

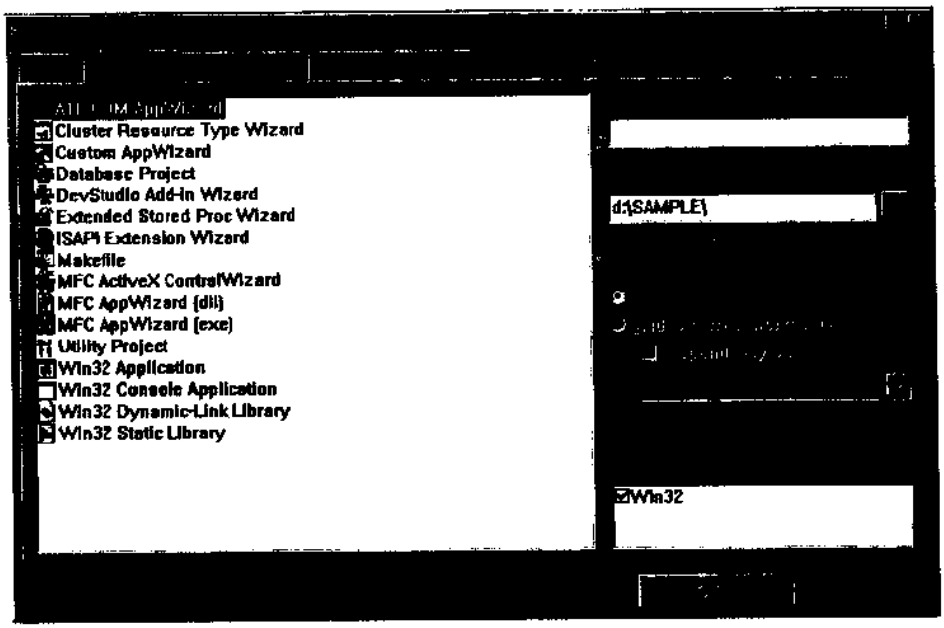



图 2.3　New 对话框

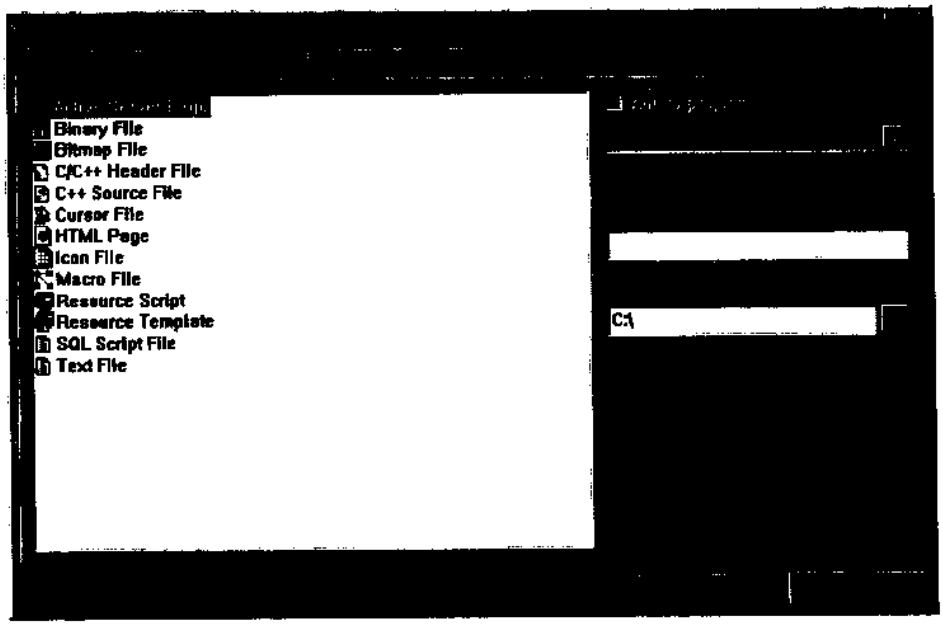



图 2.4　New 对话框的 Files 标签

·C++ Source File：创建C++源文件。

· Cursor File：创建光标文件。

·HTML Page：创建 HTML 文件。

·Icon File：创建图标文件。

·Macro File：创建宏文件。

·Resource Script：创建资源脚本文件。

·Resource Template：创建资源模板文件。

·SQL Script File：创建 SQL 脚本文件。

·Text File：创建文本文件。

2.创建新的项目

要创建新的项目，请从 Projects 标签(图 2.5)单击要创建的项目类型，然后在 Project name:文本框键入项目的名字。如果要添加新的项目到打开的工作区中，请选择 Add to current workspace 选项，否则将自动创建包含新项目的新工作区。如果要使新项目成为已有项目的子项目，请选中 Dependency of 复选框并指定项目名。

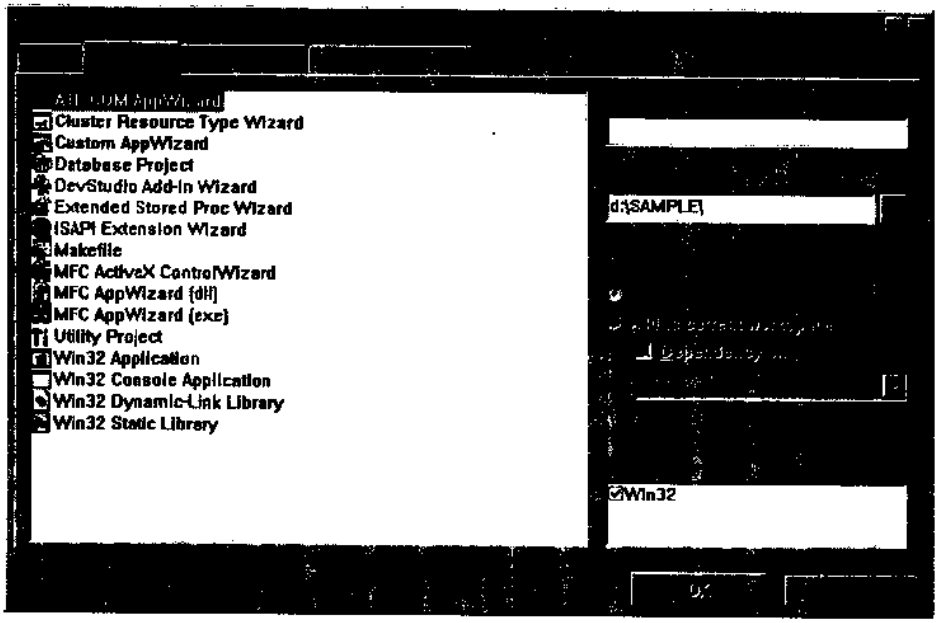



图 2.5 New 对话框的 Projects 标签

Visual C++ 6.0 可以创建的项目类型有：

·ATL COM AppWizard：创建 ATL 应用程序。

·Cluster Resource Type Wizard：创建资源组类型的程序向导。

·Custom AppWizard：创建自定义的 AppWizard。

·Database Project：直接创建数据库项目。

·DevStudio Add-in Wizard：创建自动化宏。

·Extended Stored Proc Wizard:创建扩展存贮过程的程序向导。

·ISAPI Extension Wizard：创建 Internet 服务器或过滤器。

·Makefile：创建 Make 文件。

·MFC ActiveX ControlWizard：创建 ActiveX 控件程序。

·MFC AppWizard(dll)：创建 MFC 动态链接库。

·MFC AppWizard(exe)：创建 MFC 可执行程序。

·Utility Project：创建工具项目。

·Win32 Application：创建 Win32 应用程序。

·Win32 Console Application：创建 Win32 控制台应用程序。

·Win32 Dynamic-Link Library：创建 Win32 动态链接库。

·Win32 Static Library：创建 Win32 静态库。

3.创建新的工作区

要创建新的工作区，从 Workspaces 标签选择一种工作区类型，然后在 Workspace name 文本框中键入工作区的名字。

4.创建新的文档

要创建新的文档，从 Other Documents 标签单击要创建的文档类型，然后在 File 文本框中键入文档的名字。如果要添加新的文档到已有的项目中，请选中 Add to project 复选框，然后选择项目名。

Visual C++ 6.0 可以创建的文档类型有 Excel 工作表、Excel 图表、PowerPoint 演示文稿和 Word 文档等。

选择 Page Setup 菜单项将弹出 Page Setup 对话框(图 2.6)。

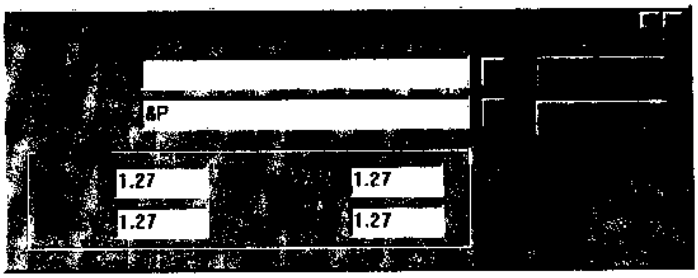



图 2.6　Page Setup 对话框

可以建立每个打印页的标题和脚注，并设置上、下、左、右页边距。表 2.2 列出了每个格式码对应的标题和脚注的类型。

**表 2.2　标题和脚注的格式码**

| 格式码 | 使用结果 | 格式码 | 使用结果 |
|---|---|---|---|
| &C | 正文居中 | &P | 加入页号 |
| &D | 加入系统日期 | &R | 右对齐正文 |
| &F | 使用文件的名字 | &T | 加入系统时间 |
| &L | 左对齐正文 | | |

### 2.2.2 Edit菜单

在Edit菜单中包含用于编辑或者搜索的命令选项(图2.7)。

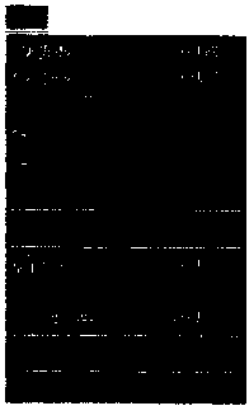


图2.7 Edit菜单

Edit菜单中各命令选项的含义,如表2.3所示。

表2.3 Edit菜单

| 菜单项 | 功能 |
|---|---|
| Undo | 取消最近一次的编辑修改操作 |
| Redo | 恢复被Undo命令取消的修改操作 |
| Cut | 剪切当前活动窗口中选定的内容到剪贴板中 |
| Copy | 复制当前活动窗口中选定的内容到剪贴板中 |
| Paste | 将剪贴板中的内容插入到当前光标所在的位置中 |
| Delete | 删除被选定的内容 |
| Select All | 选择当前活动窗口中的所有内容 |
| Find | 查找指定的字符串 |
| Find in Files | 在多个文件间搜索文本,搜索的对象可以是文本字符串,或是正则表达式 |
| Replace | 替换指定的文本串 |
| Go To | 可以指定将光标移到当前活动窗口的指定位置 |
| Breakpoints | 可以设置、删除和查看断点 |

下面,我们详细介绍Find和Breakpoints菜单项。

1.Find菜单项

选择Find菜单项将弹出Find对话框(图2.8)。

在Find对话框中,可以在Find What:文本框中输入欲查找的字符串,并设置查找的方向(Up或Down)。此外,还可以根据需要进行区分大小写字符串查找(Match Case)或进行整词查找(Match Whole Word Only),这时就应选择相应的复选框。对话框中的Regular



图 2.8　Find 对话框

Expression 选项特别有用，选择该项将按正则表达式来查找文件中匹配的文本。正则表达式是指用特殊的字符序列去匹配文件中的某个文本模式。表 2.4 列出了正则表达式的查找模式及其含义。

**表 2.4　正则表达式的查找模式**

| 查找模式 | 含义 | 示例 |
|---|---|---|
| * | 匹配任意多个字符 | Exam * 匹配 Exam，Example 和 Exam1 |
| . | 匹配单个字符 | Exam. 匹配 Exam2，Examp，不匹配 Example |
| ^ | 匹配以指定字符串开头的每一行 | ^Exam 匹配以 Exam 开头的各行 |
| + | 匹配以指定字符串结束的字符串 | + Exam 匹配 is_Exam 和 notExam |
| $ | 匹配以指定字符串结束的每一行 | Exam $匹配以 Exam 结束的各行 |
| [] | 匹配指定字符集中的字符 | Exam[1…9]匹配 Exam5，但不匹配 ExamS |
| \ | 匹配与指定字符一致的字符 | Exam[A…Z n \ 0…9]可匹配 ExamSn5，但不匹配 ExamAn9 |
| \|\| | 匹配花括号间指定字符串的任意序列 | Exam_ \|#\|\|*可匹配 Exam_，Exam_#1和 Exam_#1#1 |

2. Breakpoints 菜单项

选择 Breakpoints 菜单项将弹出 Breakpoints 对话框(图 2.9)，可以设置、删除和查看断点。

断点实际上是告诉调试器应该在何时何地中断程序的执行过程，以便检查程序代码、变量和寄存器值，必要的话可以修改、继续执行或中断执行。在 Visual C++ 6.0 中，断点分为位置断点、数据断点、消息断点和条件断点等类型。所有已设置的断点都出现在 Breakpoints 对话框底部的 Breakpoints 列表中。可以使用 Breakpoints 列表检查程序中的所有断点，也可以从列表中删除某一断点。

Breakpoints 对话框的 Location、Data 和 Messages 标签分别用于设置位置断点、数据断点和消息断点。条件断点的设置必须先设置位置断点，然后单击 Condition 按钮，从弹出的 Breakpoints Condition 对话框来指定中断程序执行的条件。

位置断点通常在源代码的指定行、函数的开始或指定的内存地址处设置。当程序执行到指定位置时，位置断点将中断程序的执行。数据断点是在某一变量或表达式上设置的。当变量或表达式的值改变时，数据断点将中断程序的执行。消息断点是在窗口函数 WndProc ( )上设置。当接收到指定的消息时，消息断点中断程序的执行。条件断点是一种

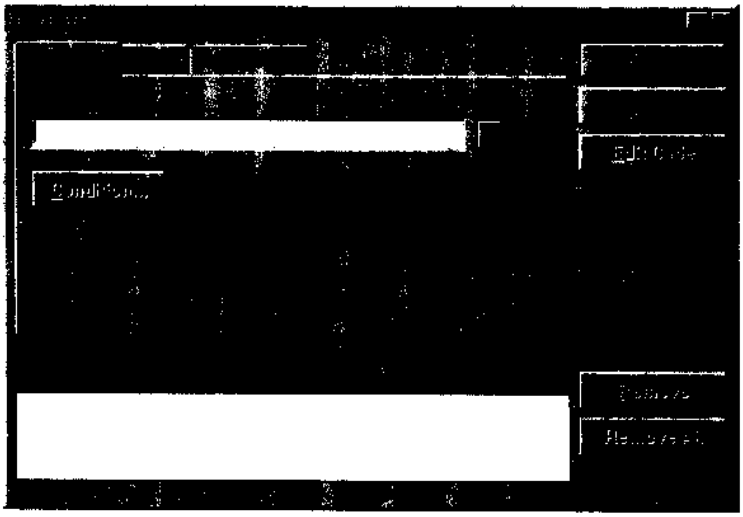


图 2.9 Breakpoints 对话框

位置断点，仅当指定的条件为真时中断程序的执行。

### 2.2.3 View 菜单

在 View 菜单中包含用于检查源代码和调试信息的命令选项(图 2.10)。



图 2.10 View 菜单

View 菜单中各命令选择的含义如表 2.5 所示。

**表 2.5 View 菜单**

| 菜单项 | 功能 |
|---|---|
| Class Wizard | 将启动 ClassWizard |
| Resource Symbols | 打开资源符号浏览器(图 2.11)，可以浏览和编辑资源符号 |
| Resource Includes | 可以修改资源符号文件名和预处理器指令 |

（续表）

| 菜单项 | 功能 |
| --- | --- |
| Full Screen | 按全屏幕方式显示活动窗口。单击 Toggle Full Screen 按钮或 Esc 键切换回原来的显示方式 |
| Workspace | 显示工作区窗口 |
| Output | 在输出窗口显示程序建立过程(如编译、链接等)的有关信息或错误信息，并显示调试运行时的输出结果 |
| Debug Windows | 弹出级联菜单，用于显示调试信息窗口 |
| Refresh | 刷新选定的内容 |
| Properties | 弹出属性对话框，可以设置或了解对象的属性 |

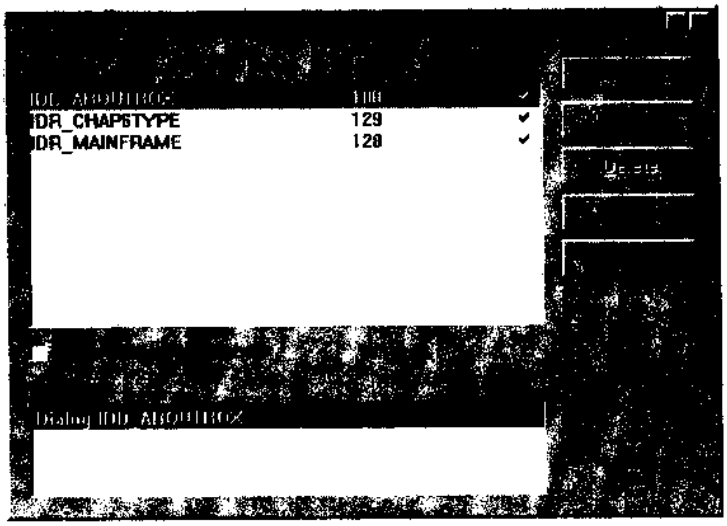



图 2.11　资源符号浏览器

下面，将详细介绍 ClassWizard 和 Debug Windows 菜单项。

1. ClassWizard 菜单项

选择 ClassWizard 菜单项将启动 ClassWizard，ClassWizard 是一个适用于 MFC 应用程序的专用工具。使用 ClassWizard，可以：

·创建新类。

·映射消息给与窗口、对话框、控件、菜单选项和加速键有关的处理函数。

·创建新的消息处理函数。

·删除消息处理消息。

·查看已经拥有处理函数的消息，并跳转到相应的处理代码中去。

·定义成员变量用于自动初始化、收集并验证输入到对话框或表单视图(Form View)中的数据。

·创建新类时，添加自动化方法和属性。

2. Debug Windows 菜单项

选择 Debug Windows 菜单项将弹出级联菜单，用于显示调试信息窗口。以下命令选项只有在调试运行状态下才可以使用。

·Watch：在 Watch 窗口显示变量或表达式的值。此外，还可以输入和编辑所要观察的表达式。

·Variables：显示当前语句和前一条语句中所使用的变量信息和函数返回值信息。这里的变量局部于当前函数或由 this 所指向的对象。

·Registers：选择该项将弹出 Registers 窗口，从中显示各通用寄存器及 CPU 状态寄存器的当前内容。而显示格式可以用 Tools 菜单中的 Options 对话框的 Debug 标签来设置。

·Memory：选择该项将弹出 Memory 窗口，从中显示内存的当前内容。而显示格式可以用 Tools 菜单中的 Options 对话框的 Debug 标签来设置。

·Call Stack：选择该项将弹出 Call Stack 窗口，从中显示所有已被调用但还未返回的函数。可以用 Tools 菜单中的 Options 对话框的 Debug 标签来设置有关的选项。

·Disassembly：选择该项将显示有关的反汇编代码及源代码以便用户直接进行反汇编调试或混合调试(即汇编调试和反汇编调试同时进行)。显示格式可以用 Tools 菜单中的 Options 对话框的 Debug 标签来设置。

### 2.2.4 Insert 菜单

使用 Insert 菜单中的命令选项(图 2.12)，可以创建新的类、创建新的资源、插入文件到文档中以及添加新的 ATL 对象到项目中，等等。



图 2.12 Insert 菜单

Insert 菜单中命令选项的含义如表 2.6 所示。

表 2.6 Insert 菜单

| 菜单项 | 功能 |
| --- | --- |
| New Class... | 创建新的类并添加到项目中 |
| New Form... | 创建定制格式视并添加到项目中 |
| Resource... | 创建新的资源或插入资源到资源文件中 |
| Resource Copy... | 创建选定资源的备份，即复制选定的资源 |

（续表）

| 菜单项 | 功能 |
| --- | --- |
| File As Text... | 弹出 Insert Resource 对话框，可以选择要插入到文档中的文件 |
| New ATL Object... | 启动 ATL Object Wizard，以便添加新的 ATL 对象到项目中 |

### 2.2.5 Project菜单

Project 菜单中的命令选项(图 2.13)用于管理项目和工作区。

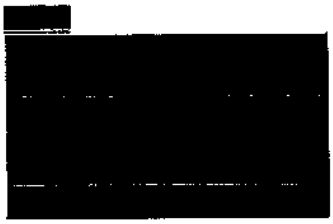


图 2.13 Project 菜单

Project 菜单中各命令选择的含义如表 2.7 所示。

表 2.7 Project 菜单

| 菜单项 | 功能 |
| --- | --- |
| Set Active Project | 选择指定的项目为工作区中的活动项目 |
| Add To Project | 用于添加文件、文件夹、数据链接以及可再用部件到项目中 |
| Dependencies | 弹出 Project Dependencies 对话框，可以编辑项目的依赖关系 |
| Settings | 弹出 Project Settings 对话框(图 2.14)，可以为项目配置指定不同的设置说明 |
| Export Makefile | 按外部 Make 文件格式导出可建立的项目 |
| Insert Project into Workspace | 可以插入已有的项目到工作区中 |

选择 Add To Project 菜单项将弹出级联菜单，可以把文件、文件夹、数据链接以及可再用部件添加到项目中。该级联菜单中几个选项的含义如下所示：

·New：弹出 New 对话框(图 2.15)，可以在工作区中创建新的文档。

·New Folder：弹出 New Folder 对话框，可以把新的文件夹插入到项目中。

·Files：弹出 Insert Files Into Project 对话框(图 2.16)，可以把已有的文件插入到项目中。

·Data Connection：添加数据链接到活动的项目中。

·Component and Controls：弹出 Component and Controls Gallery 对话框(图 2.17)，可以插入可再用部件或者已注册的 ActiveX 控件到项目中。插入时相当于插入相关的头文件(.H 文件)和实现文件(.CPP 文件)，并更新工作区窗口中的信息。

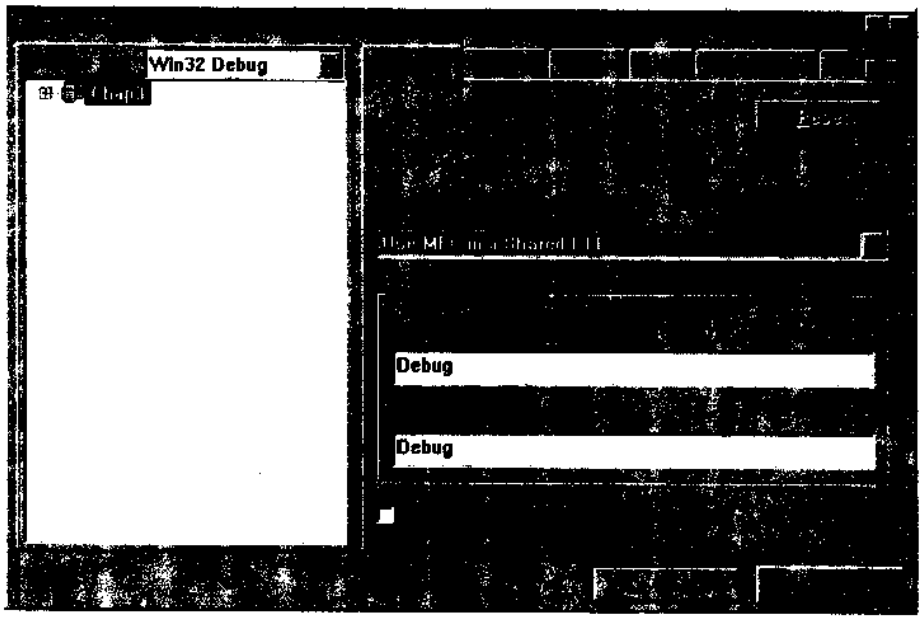



图 2.14 Project Settings 对话框

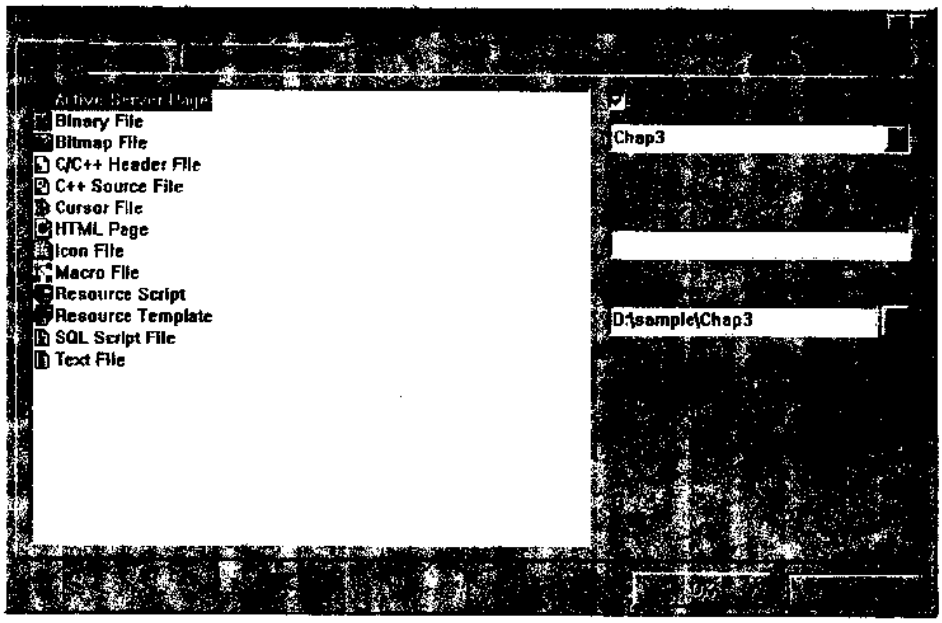



图 2.15 New 对话框

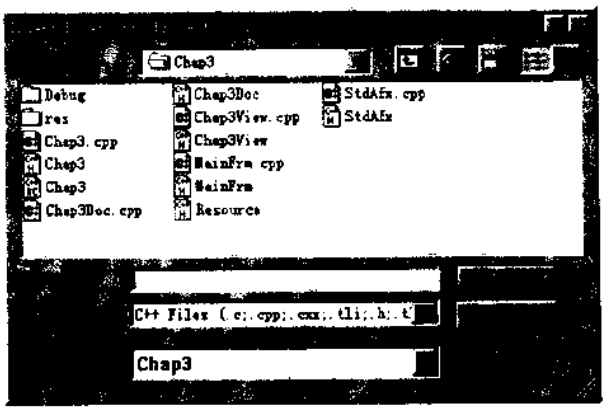



图 2.16　Insert Files Into Project 对话框

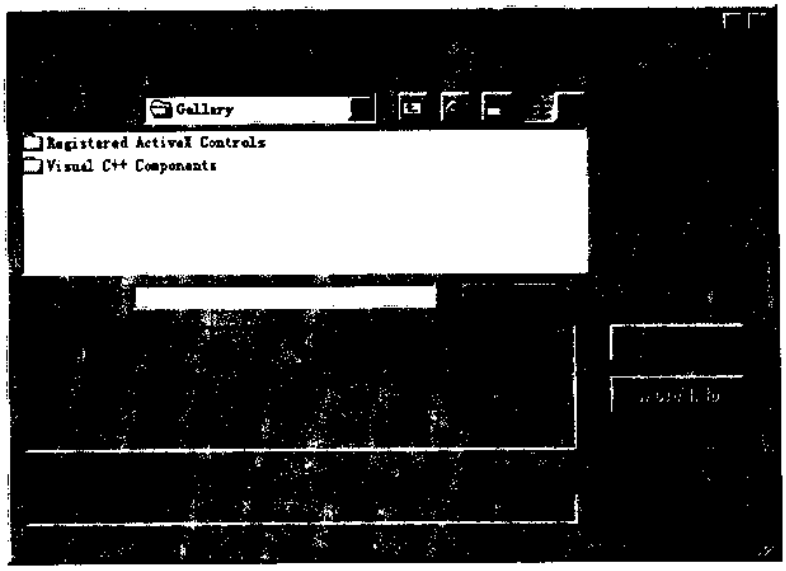



图 2.17　Component and Controls Gallery 对话框

### 2.2.6　Build 菜单

Build 菜单中的命令选项(图 2.18)用于编译、建立和执行应用程序。

Build 菜单中各命令选项的含义如表 2.8 所示。

**表 2.8　Build 菜单**

| 菜单项 | 功能 |
|---|---|
| Compile | 编译显示在源代码编辑窗口中的源文件，用于检查源文件中是否有语法错误 |
| Build | 查看项目中的所有文件，并对最近修改过的文件进行编译和链接。如果建立过程中检测出某些语法错误(如警告信息或错误信息)，将显示在输出窗口中 |

（续表）

| 菜单项 | 功能 |
| --- | --- |
| Rebuild All | 编译和链接项目中所有的文件 |
| Batch Build | 用于一次建立多个项目 |
| Clean | 删除项目的中间文件和输出文件 |
| Start Debug | 将启动调试器程序 |
| Debugger Remote Connection | 可以对远程调试链接设置进行编辑 |
| Execute | 用于运行程序 |
| Set Active Configuration | 用于选择活动项目的配置，如 Win32 Release 和 Win32 Debug |
| Configurations | 弹出 Configurations 对话框，可以编辑项目配置 |
| Profile | 检查程序运行行为的工具，可以找出代码中的高效部分，给出未执行代码区域的诊断信息 |

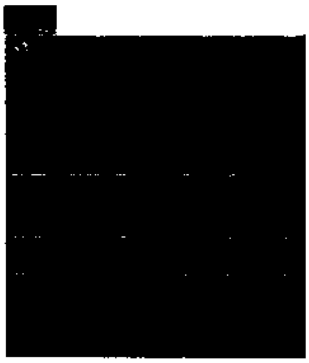


图 2.18 Build 菜单

选择 Start Debug 菜单项将弹出级联菜单，包含有启动调试器控制程序运行的子选项 Go、Step Into、Run To Cursor 和 Attach to Process。启动调试器后，Debug 菜单将代替 Build 菜单出现在菜单栏中。此外，Edit 菜单和 View 菜单中与调试有关的选项将可以使用。

### 2.2.7 Debug 菜单

启动调试器后，Debug 菜单(图 2.19)将取代 Build 菜单出现在菜单栏中。

Debug 菜单中各命令选择的含义如表 2.9 所示。

表 2.9 Debug 菜单

| 菜单项 | 功能 |
| --- | --- |
| Go | 在调试过程中从当前语句启动或继续运行程序，直至到达断点处停止 |
| Restart | 系统重新装载程序到内存并放弃所有变量的当前值(断点表达式和观察点表达式仍可用) |
| Stop Debugging | 中断当前的调试过程并返回正常的编辑状态 |

（续表）

| 菜单项 | 功能 |
| --- | --- |
| Break | 在当前位置暂停程序运行 |
| Apply Code Changes | 调试时，应用于代码的类型间的切换 |
| Step Into | 调试过程中单步执行程序。当执行到某一函数调用语句时，进入该函数内部，从头单步执行 |
| Step Over | 调试过程中单步执行程序。当执行到某一函数调用语句时，不进入该函数内部，而是直接执行该调用语句，接着再执行调用语句后面的语句 |
| Step Out | 当执行 Step Into 命令进入函数内部，可以使用 Step Out 选项使程序直接向下运行，直到从该函数内部返回，在该函数调用语句后面的语句处停止 |
| Run to Cursor | 调试运行程序时，使程序在运行到当前光标所在位置时停止 |
| Step Into Specific Function | 用于单步执行选定的函数 |
| Exceptions | 可以控制调试器如何处理系统异常和用户自定义异常 |
| Threads | 显示调试过程中可用的所有线程，可以挂起和恢复线程并设置焦点(Focus) |
| Modules... | 用于显示各模块的起始地址 |
| Show Next Statement | 表示正在执行的代码行 |
| QuickWatch | 可以查看及修改变量和表达式或将变量和表达式添加到 Watch 窗口中 |

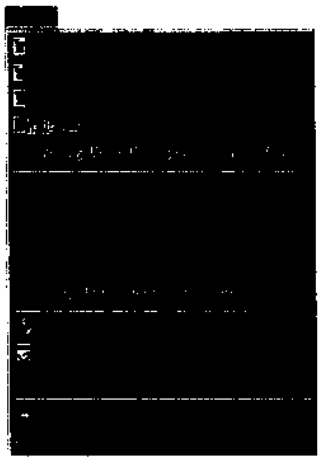


图 2.19　Debug 菜单

### 2.2.8　Tools 菜单

Tools 菜单中的命令选项(图 2.20)用于浏览程序符号、定制菜单与工具栏、激活常用的工具(如 Register Control 等)或者更改选项设置等。

Tools 菜单中各命令选择的含义如表 2.10 所示。

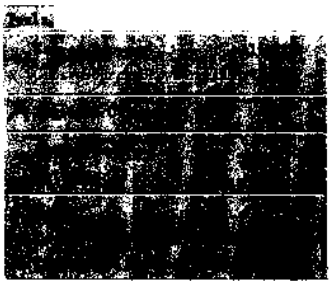


图 2.20 Tools 菜单

**表 2.10 Tools 菜单**

| 菜单项 | 功能 |
|---|---|
| Source Browser | 弹出浏览窗口 |
| Close Source Browser File | 关闭打开的浏览信息数据库 |
| Customize | 可以对命令、工具条、工具菜单和键盘加速键进行定制 |
| Options | 可以对 Visual C++ 6.0 的环境设置进行更改 |
| Macro | 创建和编辑宏文件 |
| 其他选项 | 启动相应的用户自定义工具(如 Register Control 等) |

选择 Source Browser 菜单项将弹出浏览窗口(图 2.21)。

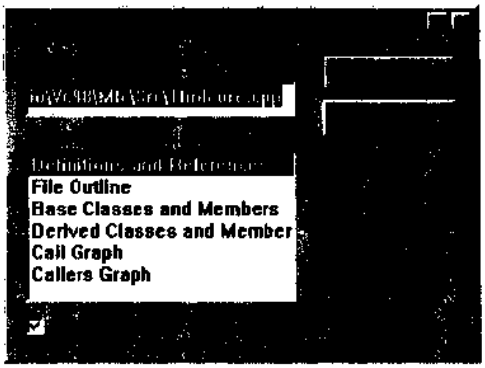



图 2.21 浏览窗口

使用浏览窗口主要可以查看以下信息:

·源文件中所有符号的信息。

·包含某个符号定义的源代码行。

·引用某个符号的所有源代码行。

·基类和派生类之间的关系。

·调用函数和被调用函数之间的关系。

选择 Options 菜单项将弹出 Options 对话框(图 2.22),可以对 Visual C++ 6.0 的环境进行设置,如调试器设置、窗口设置、目录设置、工作区设置、兼容性设置和格式设置等进行更改。

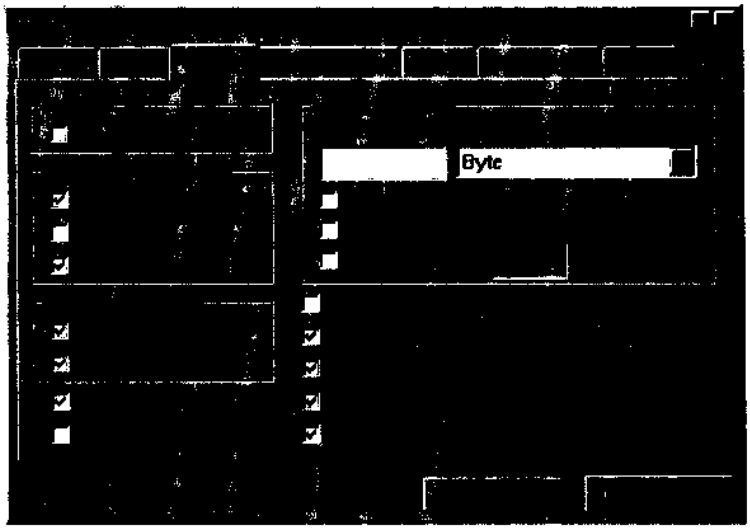



图 2.22　Options 对话框

### 2.2.9　Window 菜单

在 Window 菜单中包含用于控制窗口属性的命令选项(图 2.23)。

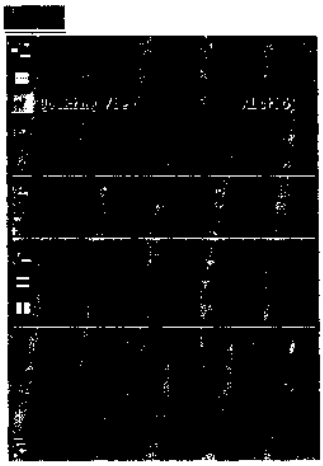


图 2.23　Window 菜单

Windows 菜单中各命令选项的含义如表 2.11 所示。

表 2.11 Window 菜单

| 菜单项 | 功能 |
|---|---|
| New Window | 打开新的窗口，从中显示当前文档信息 |
| Split | 将窗口拆分为多个窗格，便于同时查看同一文档的不同内容 |
| Docking View | 打开或者关闭窗口的可停靠(docking)特征 |
| Close | 关闭选定的活动窗口 |
| Close All | 关闭所有打开的窗口 |
| Next | 激活下一个窗口 |
| Previous | 激活上一个窗口 |
| Cascade | 将当前打开所有的窗口在屏幕上向下重叠排放 |
| Tile Horizontally | 使当前所有打开的窗口在屏幕上纵向平铺 |
| Tile Vertically | 使当前所有打开的窗口在屏幕上横向平铺 |
| 打开窗口的历史记录 | 列出最近打开的窗口的文件名 |
| Windows | 可以管理当前打开的窗口 |

### 2.2.10 Help 菜单

通过选择 Help 菜单的命令选项(图 2.24)可以了解 Visual C++ 6.0 的各种联机帮助信息。



图 2.24 Help 菜单

## 2.3 Visual C++ 6.0 工具条

工具条由某些操作按钮组成，分别对应着某些菜单选项或命令的功能。可以直接用鼠标单击这些按钮来完成指定的功能。工具条按钮大大简化了用户的操作过程，并使操作过程可视化。

Visual C++ 6.0 包含有十几种工具条。缺省状态时，屏幕工具条区域显示有两个工具条，即 Standard 工具条和 Build 工具条。

1.Standard 工具条

Standard 工具条主要由以下的工具按钮组成(图 2.25)：



图 2.25　Standard 工具条

·New Text File：创建新的文本文件。

·Open：打开已有的文档。

·Save：保存文档。

·Save All：保存所有打开的文件。

·Cut：剪切选定的内容到剪贴板中。

·Copy：复制选定的内容到剪贴板中。

·Paste：在当前插入点处插入剪贴板中的内容。

·Undo：取消最后的操作。

·Redo：重复先前取消的操作。

·Workspace：显示或者隐藏工作区窗口。

·Output：显示或者隐藏输出窗口。

·Window List：管理当前打开的窗口。

·Find in Files：在多个文件中搜索字符串。

·Find：激活查找工具。

·Search：搜索联机文档。

2.Build 工具条

Build 工具条主要由以下工具按钮组成(图 2.26)：

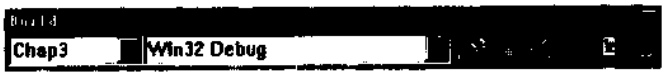



图 2.26　Build 工具条

·Select Active Project：选择活动项目。

·Select Active Configuration：选择活动配置。

·Compile：编译文件。

·Build：建立项目。

·Stop Build：停止项目的建立。

·Execute Program：执行程序。

·Go：启动或者继续执行程序。

·Insert/Remove Breakpoint：插入或者删除断点。

如果要在屏幕上显示或者隐藏工具条，请在屏幕工具条区域单击鼠标右键，从工具条上下文菜单选择或者清除相应的工具条。

## 2.4 项目与项目工作区

Developer Studio以项目工作区(project workspace)的方式来组织文件、项目和项目配置，通过项目工作区窗口可以查看和访问项目中的所有元素。

每个项目工作区由工作区目录中的项目工作区文件组成。项目工作区文件用于描述工作区及其内容，扩展名为.dsw。工作区目录是项目工作区的根目录，添加到项目工作区中的项目可以位于其他路径甚至不同的驱动器中。

首次创建项目工作区时，将创建一个项目工作区目录、一个项目工作区文件以及相关的文件(包括一个项目文件和一个工作区选项文件)。工作区选项文件用于存储项目工作区设置，扩展名为.opt。

建立项目后，可以添加任何其他目录的文件到项目中。添加文件到项目中并不改变文件的位置。项目仅仅是记录文件的名字和位置，并在工作区窗口显示图标以便指明在项目中该文件与其他文件的关系。

创建或者打开项目工作区时，Developer Studio将在项目工作区窗口中显示与项目有关的信息。图2.27是一个典型的项目工作区窗口。

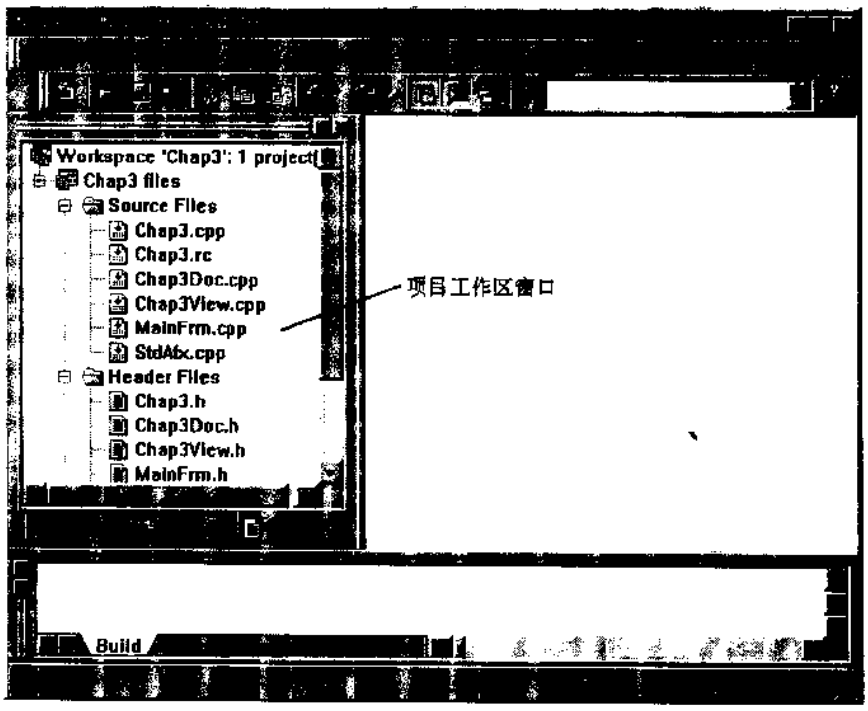



图2.27 项目工作区窗口

项目工作区窗口由三个面板构成，即ClassView、ResourceView和FileView。每个面板用于指定项目工作区中所有项目的不同视图。每个面板至少有一个顶层文件夹。顶层文件夹

由组成项目视图的元素组成。通过扩展文件夹可以显示视图的详细信息。视图中每个文件夹可以包含其他文件夹或各种元素(如子项目、文件、资源、类和标题等)。

1. ClassView 面板

ClassView 面板用于显示项目中定义的C++类(图 2.28)。扩展顶层文件夹可以显示类，扩展类可以显示类的成员。

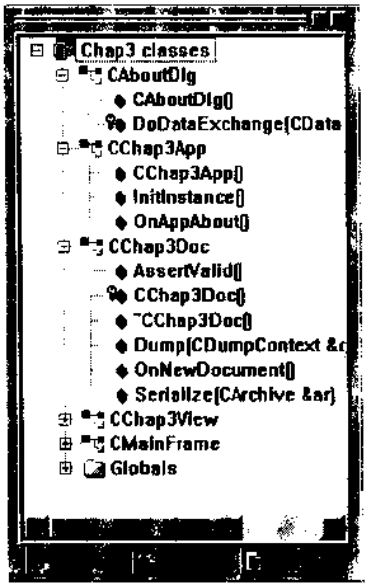



图 2.28　ClassView 面板

通过 ClassView 面板，可以定义新类、直接跳转到代码(如类定义、函数或者方法定义

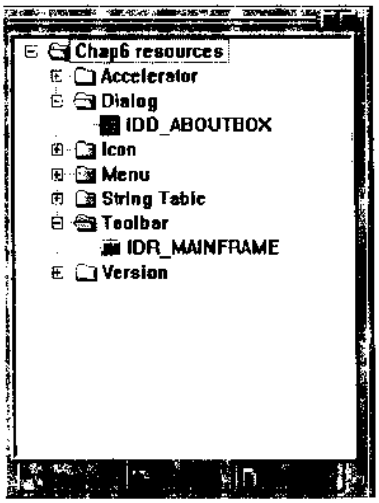



图 2.29　ResourceView 面板

等)、创建函数或者方法声明等。

2. ResourceView 面板

ResourceView 面板用于显示项目中包含的资源文件。扩展顶层文件夹可以显示资源类型(图 2.29),扩展资源类型可以显示其下的资源。

3. FileView 面板

FileView 面板显示项目之间的关系以及包含在项目工作区中的文件。扩展顶层文件夹可以显示项目中的文件(图 2.30)。

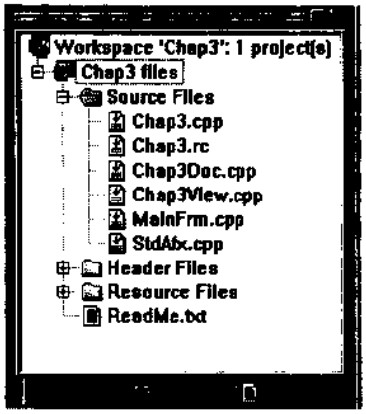



图 2.30 FileView 面板

在项目工作区窗口中,选择某一项并按 Alt + Enter 键将打开相应的属性对话框;双击某一项则以适当方式显示该项,例如,在源代码编辑窗口中显示源文件,在对话框编辑器中显示对话框;选择一项并单击鼠标右按钮,将弹出快捷菜单,从中可以执行频繁使用的命令。

## 2.5 资源与资源编辑器

Visual C++ 5.0 可以处理的资源有加速键(Accelerator)、位图(Bitmap)、光标(Cursor)、对话框(Dialog Box)、图标(Icon)、菜单(Menu)、串表(String Table)、工具条(Toolbar)和版本信息(Version Information)等。

### 2.5.1 资源编辑器

Developer Studio 提供有功能强大且易于使用的资源编辑器。它用于创建和修改应用程序的资源。使用资源编辑器,可以创建新的资源、修改已有的资源、拷贝已有的资源以及删除不再需要的资源。

创建或者打开资源时,系统将自动打开相应的编辑器。编辑器打开后,单击鼠标右键将弹出上下文菜单,其中列有与当前资源有关的命令。

1. 创建新的资源

从 Insert 菜单选择 Resource 命令，弹出 Insert Resource 对话框(图 2.31)。如果要创建新的资源，从 Resource Type 列表选择资源类型，然后单击 New 按钮。

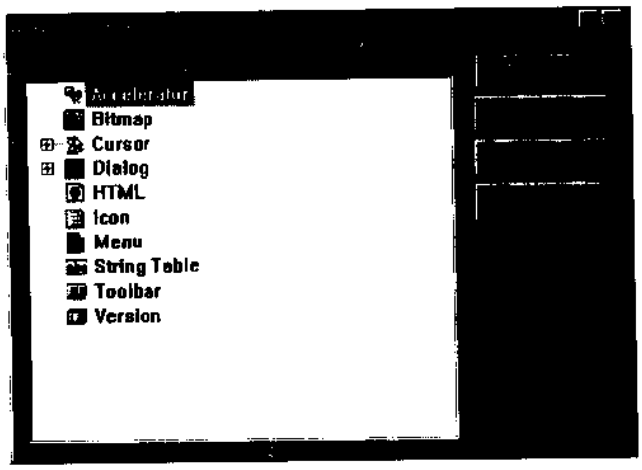



图 2.31　Insert Resource 对话框

新创建的资源将加入到当当前资源文件中。

此外，可以单击 Resource 工具条中的相应按钮(图 2.32)来创建新的资源。



图 2.32　Resource 工具条

Resource 工具条包含以下按钮：

·New Dialog：创建新的对话框资源。

·New Menu：创建新的菜单资源。

·New Cursor：创建新的光标资源。

·New Icon：创建新的图标资源。

·New Bitmap：创建新的位图资源。

·New Toolbar：创建新的工具条资源。

·New Accelerator：创建新的加速键表资源。

·New String Table：创建新的串表资源。

·New Version：创建或者打开版本信息资源。

·Resource Symbols：浏览和编辑资源文件中的资源符号。

2．查看和修改资源

可以使用项目工作区窗口的 ResourceView 面板(图 2.29)来查看资源。刚打开 ResourceView 面板时，系统自动压缩每个资源分类，可以单击"+"标记来扩展每一分类。

可以使用菜单命令来复制、移动、粘贴或者删除资源，也可以通过双击打开相应的编辑器来修改资源，还可以用资源属性对话框来修改资源的语言属性或条件属性。

3. 导入位图、光标或图标

可以将单独的位图、光标或图标文件导入到资源文件中，方法为：

(1) 在 ResourceView 面板中单击鼠标右键，从上下文菜单中选择 Import 命令，弹出 Import Resource 对话框(图 2.33)。

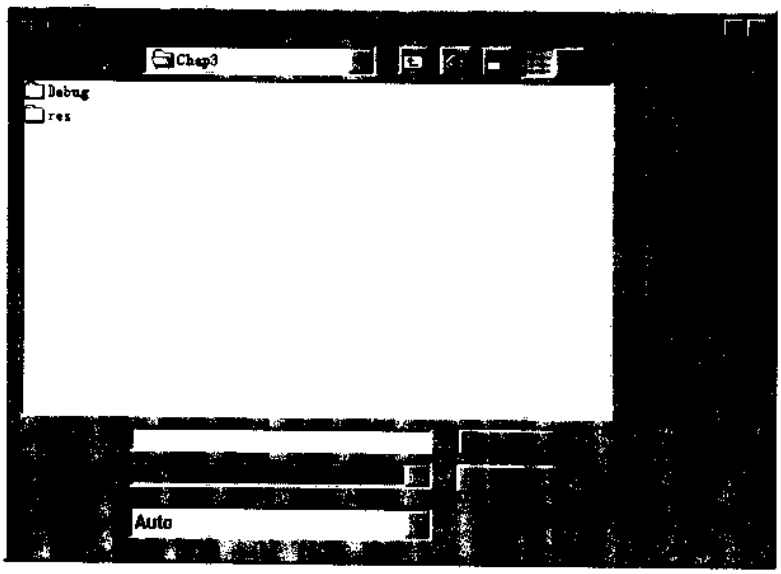



图 2.33 Import Resource 对话框

(2) 从对话框选择要导入的.BMP(位图)、.ICO(图标)或.CUR(光标)文件。

(3) 之后，单击 Import 按钮即可将文件添加到当前资源文件中。

此外，还可以使用上下文菜单的 Export 命令将位图、光标或图标从资源文件导出到单独的文件中。

### 2.5.2 资源符号

资源符号由映射到整数值上的文本串组成，用于在源代码或资源编辑器中引用资源或对象。在创建新的资源或对象时，系统自动为其提供缺省符号名(如 IDD_ABOUTBOX)和符号值。缺省时，符号名和符号值自动保存在系统生成的资源文件 resource.h 中。

可以使用资源属性对话框来改变资源的符号名或符号值，方法为：

(1)在 ResourceView 面板中选择要处理的资源。

(2) 从 View 菜单中选择 Properties 命令或按 Alt + Enter 键，弹出相应的资源属性对话框(图 2.34)。

(3) 在 ID 文本框输入新的符号名或符号值，或从已有的符号列表中选择一种符号。如果输入新的符号名，系统会自动为其赋值。

也可以在文本编辑器中直接修改 resource.h 文件来改变与某个资源或对象有关的符号。

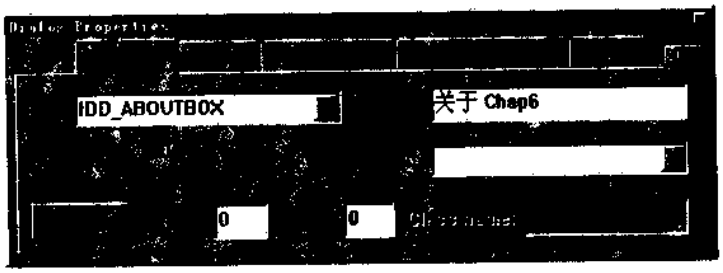



图 2.34 资源属性对话框

符号名通常带有描述性的前缀,以表示所代表的资源或对象类型。例如,加速键或菜单前缀为“IDR_”,对话框前缀为“IDD_”,光标前缀为“IDC_”,图标前缀为“IDI_”,位图前缀为“IDB_”,菜单项前缀为“IDM_”,命令前缀为“ID_”,控件前缀为“IDC_”,串表中的串前缀为“IDS_”,消息框中的串前缀为“IDP_”。

### 2.5.3 资源符号浏览器

随着应用程序的大小和复杂程度的增加,与其相关的资源符号也会不断增多。要手工跟踪分散在不同文件中的大量资源符号是相当困难的。资源符号浏览器(图 2.11)简化了资源符号的管理。使用资源符号浏览器可以:

·快速浏览已有资源符号的定义以便了解每个资源的符号值、已使用的资源符号列表及与每个符号相关的资源。

·创建新的资源符号。

·更改资源的符号名和符号值。

·删除不再使用的资源符号。

·快速切换到其个资源所对应的编辑器中。

### 2.5.4 对话框编辑器

对话框编辑器用于创建或者编辑对话框资源或对话框模板。使用对话框编辑器,可以做以下工作:

·添加、排列或编辑控件。

·改变控件的制表顺序(Tab Order)或助记键(Mnemonic Key)。

·调整对话框布局。

·添加或编辑 ActiveX 控件。

·创建用户自定义控件。

·导入 Visual Basic 表单到对话框资源中。

·测试对话框。

图 2.35 是打开某一对话框资源后的对话框编辑器。对话框编辑器打开时,将显示对话框工具条和控件工具条。

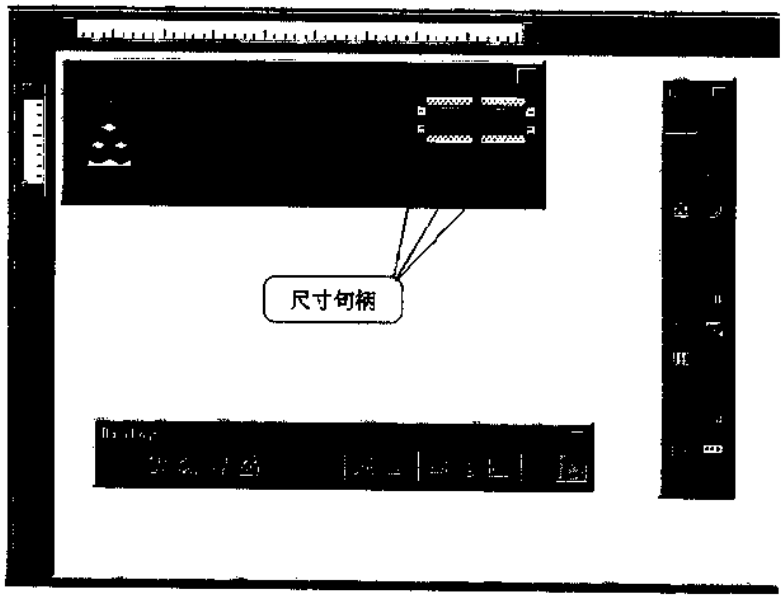



图 2.35 对话框编辑器

1. 添加并编辑控件

创建对话框的第一步就是把控件添加到对话框中。可以添加到对话框中的控件主要有图片(Picture)、静态文本(Static Text)、编辑框(Edit Box)、成组框(Group Box)、按钮(Button)、复选框(Check Box)、单选按钮(Radio Button)、组合框(Combo Box)、列表框(List Box)、水平滚动条(Horizontal Scroll Bar)和垂直滚动条(Vertical Scroll Bar)等。

添加控件最简单的方法就是将控件从控件工具条(图 2.36)拖到对话框编辑窗口的指定位置后释放鼠标。



图 2.36 控件工具条

添加控件后可以单击来选择要修改的控件，或者使用控件工具条的选择工具或按住 Shift 键再单击来选择多个控件。选择控件后，就可以对其进行移动、复制、删除或调整，也可以拖动尺寸句柄(图 2.35)来缩放控件。此外，还可以使用控件的属性窗口来修改控件属性。

2. 格式化对话框

对话框编辑器提供有专门的工具用于格式化对话框。这些工具都显示在对话框的工具条中(图 2.37)。对话框工具条包含以下工具按钮：

·Test：运行对话框以测试对话框的外观和行为。

·Align Left：将选定的控件按左对齐格式放置。

·Align Right：将选定的控件按右对齐格式放置。
·Align Top：将选定的控件按上对齐格式放置。
·Align Bottom：将选定的控件按下对齐格式放置。
·Central Vertical：将选定的控件按中心垂直对齐格式放置。
·Central Horizontal：将选定的控件按中心水平对齐格式放置。
·Space Across：使选定的控件两两水平间隔相同。
·Space Down：使选定的控件两两垂直间隔相同。
·Make Same Width：使选定的控件有相同的宽度。
·Make Same Height：使选定的控件有相同的高度。
·Make Same Size：使选定的控件有相同的宽度和高度。
·Toggle Grid：在显示或隐藏网格间切换。
·Toggle Guides：在显示或隐藏标尺间切换。



图 2.37 对话框工具条

此外，在进入对话框编辑状态时，屏幕菜单栏将增加 Layout 菜单，可以使用其中的命令选项来格式化对话框。

3．改变制表顺序和助记键

制表顺序是指在对话框中按 Tab 键将输入焦点从一个控件移到另一控件的顺序。通常都是从左到右，从上到下。每个控件的 Tap Stop 属性用于确定控件是否接收输入焦点。

如果要改变控件的制表顺序，请从 Layout 菜单选择 Tab Order 命令。此时每个控件的左上角标有一数字，用于指明该控件在当前制表顺序中的序号。依次单击各个控件，于是控件被单击的顺序即为新的制表顺序。设置后按回车键结束 Tab Order 方式。

Visual C++ 允许用户定义助记键，以便直接按助记键即可跳到相应的控件中。对于具有可见标题的控件(如按钮、复选框或单选按钮等)，定义助记键的方法为先选择该控件，然后从 View 菜单选择 Properties 命令，在属性窗口的 Caption 文本框中输入符号 & 再输入助记符。对于无可见标题的控件，可以先为其定义静态文本作为标题，然后在文本中欲作为助记符的字母前加上 & 号。

4．表单视图对话框

表单视图(Form View)是包含对话控件且与 CView 类兼容的窗口模板。有些应用程序的主要功能就是用于数据输入。这时应用程序的主视图除了包含用于输入数据的对话控件外，不再包含其他内容。

为了建立表单视图，首先要按通常的方法创建对话框，并在对话框属性窗口的 Style 标签中设置不同的风格属性(在 Style 下拉列表框中选择 Child、在 Border 下拉列表框中选择 None 并清除 Visiable 复选框)，在 General 标签中清除对话框的标题，然后用 MFC 类库中的 CFormView 类将表单视图嵌入到程序中。

可以用同样的方法来创建对话栏，不同之处在于要用MFC类库中的CDialogBar类来嵌入对话栏到程序中。

5. 测试运行对话框

Visual C++ 6.0可以在对话框编辑器中测试运行对话框，从而可以及时了解对话框的布局和行为。选择Layout菜单的Test命令或单击对话工具栏的Test按钮，即可进入测试运行状态。在测试运行状态，可以输入文本，从组合框或列表框进行选择，测试对话框的热键是否有效，等等。

### 2.5.5 菜单编辑器

菜单编辑器用于创建并编辑菜单资源。使用菜单编辑器，可以创建标准菜单和菜单选项，为菜单或菜单选项定义热键、加速键和状态栏提示。也可以创建上下文菜单，以便用鼠标右键来执行要频繁使用的命令。建立菜单或菜单选项后，可以用ClassWizard为菜单选项编写要执行的代码。图2.38为打开某一菜单资源后的菜单编辑器。

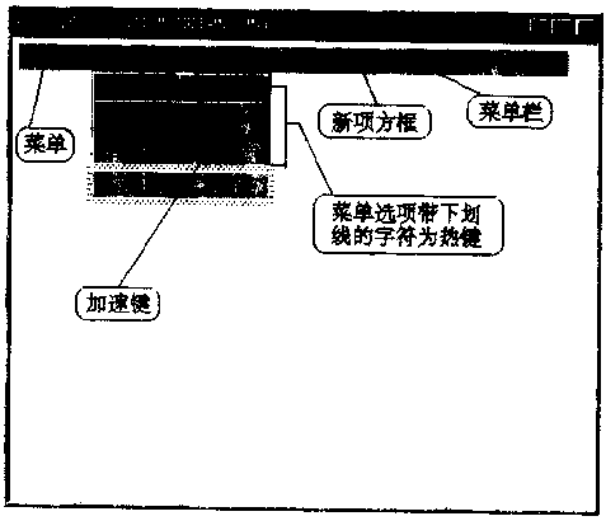



图2.38 菜单编辑器

1. 创建菜单和菜单选项

进入菜单编辑器后，就可以在菜单栏中创建菜单和菜单选项。

要创建菜单栏中的菜单，方法为：

(1) 在菜单栏中选择新项方框，或者按Tab键(向右移)、Shift + Tab键(向左移)或左右箭头键移到新项方框。如果要在某一菜单前插入新的菜单，则移到该菜单处按Ins键。

(2) 输入菜单名。开始输入名字时，系统弹出Menu Item Properties对话框(图2.39)。在对话框的Caption文本框输入菜单名。

如果要为菜单定义助记符，则在相应的字母前加符号&。注意，要确保同一菜单栏上的助记符不相冲突。如果要建立单项菜单而不带菜单选项，则应清除Pop-up复选框。

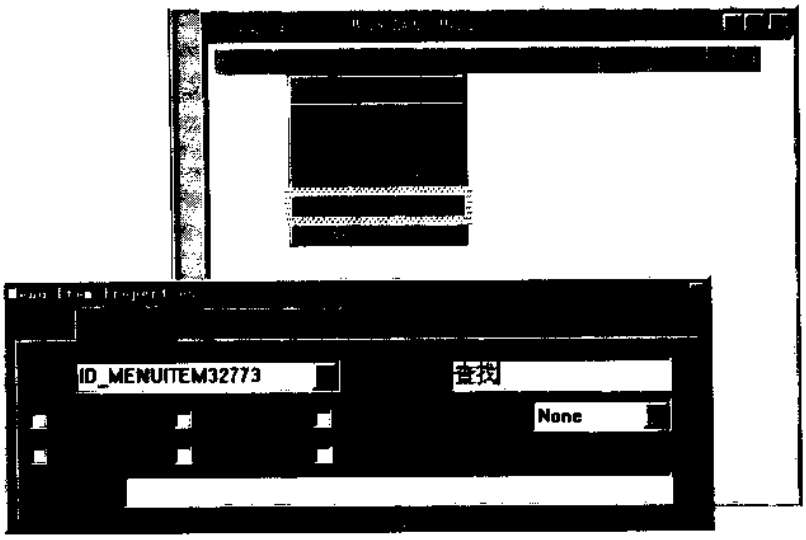



图 2.39　Menu Item Properties 对话框

创建菜单后就可以为其添加菜单选项，方法为：

(1) 选择菜单的新项方框，或者选中某个已有菜单选项再按 Ins 键，新项自动插在该项之前。

(2) 输入菜单选项的名字。开始输入名字时，系统弹出 Menu Item Properties 对话框。在对话框的 Caption 文本框输入菜单选项名。如果要为菜单定义助记符，则在相应的字母前加符号 &。

(3)在 ID 文本框，输入菜单选项的 ID 号或选取已有的 ID 号。如果不输入 ID 值，则系统根据选项名称自动生成一个 ID 值。

(4) 在菜单选项的属性对话框可为菜单选项指定风格。

有些菜单选项可能还含有下级子菜单(即级联菜单)。要创建级联菜单，方法为：

(1) 在菜单中欲显示级联菜单的位置按 Ins 键，输入菜单选项名称，开始输入时，系统弹出 Menu Item Properties 对话框。或者选取已有的菜单选项，然后按 Alt + Enter 键。

(2) 在菜单选项的属性对话框中选中 Pop - up 复选项，于是该项便被标以级联菜单符号(▶)，且在该项的右侧出现新项方框。

(3) 为级联菜单添加子菜单项(图 2.40)。

2. 定义加速键

有时候需要为菜单选项定义加速键，以便直接按加速键执行相应的命令。

(1) 选择要定义加速键的菜单选项，按 Alt + Enter 键，系统弹出 Menu Item Properties 对话框。

(2) 在 Caption 文本框中将加速键加到菜单标题的后面。如果在菜单标题后输入转义符 \ t，则所有加速键都按左对齐格式显示。

(3) 在加速键编辑器中建立相应的加速键表条目，并赋给与菜单选项相同的 ID 号。

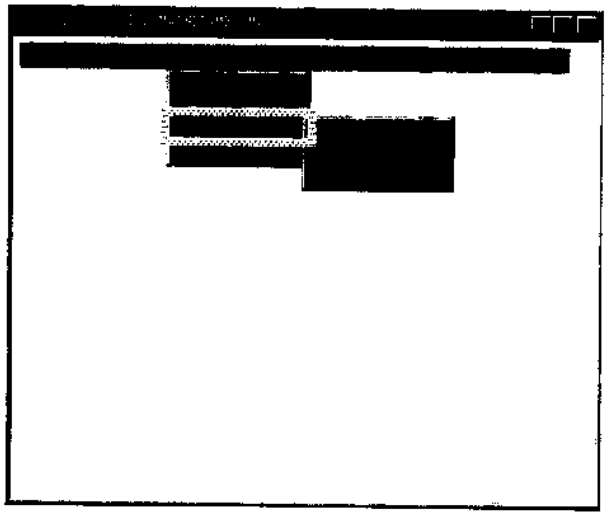


图 2.40 级联菜单

3. 定义状态栏提示

除了为菜单选项定义加速键，还可以为其定义状态栏提示。这样，只要选中该项，系统将在状态栏提示相应的描述性文本。

(1) 选择要定义状态栏提示的菜单选项，按 Alt + Enter 键，系统弹出 Menu Item Properties 对话框。

(2) 在 Prompt 文本框输入描述性文字。

图 2.41 是为菜单选项 Go 定义状态栏提示后的 Menu Item Properties 对话框。

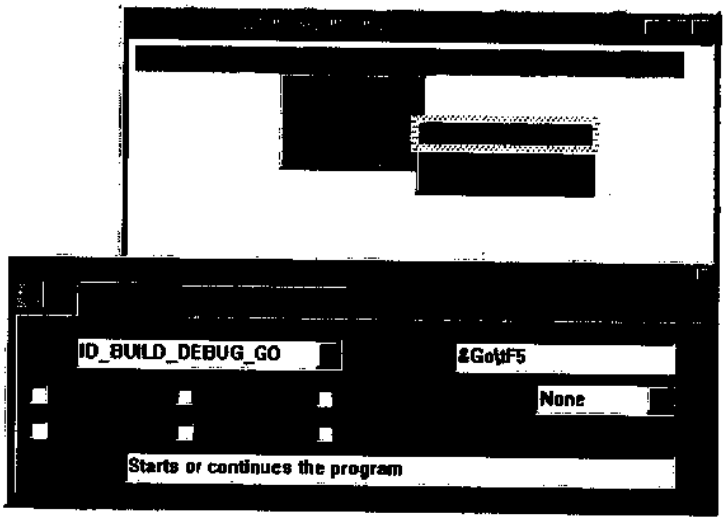



图 2.41 为菜单选项定义状态栏提示

4. 创建上下文菜单

我们知道，单击鼠标右键将弹出相应的上下文菜单。上下文菜单包含有与当前光标所指位置最为相关的命令。为了在应用程序中使用上下文菜单，首先要创建菜单本身，然后将其与应用程序代码链接。创建上下文菜单的步骤如下：

(1) 创建带空标题的菜单栏。

(2) 输入临时字符为菜单标题，以便在菜单栏创建菜单。

(3) 在菜单下创建上下文菜单的菜单选项。

(4) 再次使菜单栏为空，以便使上下文菜单显示在空的菜单栏下。

(5) 保存菜单资源。

(6) 在源文件中要添加与下列代码类似的代码：

```
CMenu menu;
//装载并验证菜单资源
VERIFY(menu.LoadMenu(IDR_MENU1));
CMenu * pPopup = menu.GetSubMenu(0);
ASSERT(pPopup! = NULL);

//显示菜单内容
pPopup - > TrackPopupMenu(TPM_LEFTALIGN|TPM_RIGHTBUTTON,
  x,y,AfxGetMainWnd( ));
```

在创建上下文菜单的菜单资源后，应用程序代码装载菜单资源并使用函数TrackPopupMenu( )来显示菜单内容。一旦用户在上下文菜单外单击鼠标，就应让上下文菜单消失。如果用户选择某个命令，则传递消息句柄给窗口。

建立上下文菜单后，可以在菜单编辑器中单击鼠标右键，从弹出的快捷菜单中选择View As Popup命令来查看或修改所建立的菜单。

### 2.5.6 加速键编辑器

加速键表是一种 Windows 资源，包含应用程序用到的所有加速键及相应的命令标识符。Visual C++ 6.0 允许应用程序包含多个加速键表。加速键通常是菜单或工具条上所用程序命令的键盘快捷键。定义加速键后，可以使用 ClassWizard 为加速键命令编写要执行的代码。

使用加速键编辑器，可以添加、删除、更改和浏览项目所用到的加速键，可以查看和更改与加速键表中每个条目有关的资源标识符(资源标识符用于在程序代码中引用加速键表中的每个条目)，还可以为某个菜单选项定义加速键。图 2.42 是打开某一加速键表资源后的加速键编辑器。

如果要在加速键表中添加新的加速键，按 Ins 键或选中表尾的新项方框输入加速键名，弹出 Accel Properties 对话框(图 2.43)。在 Key 文本框输入键名，在 ID 文本框输入加速键标识符。

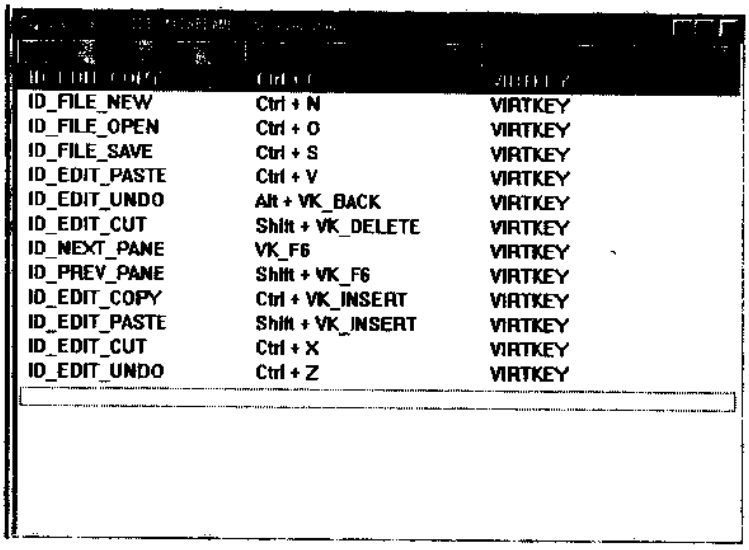



图 2.42 加速键编辑器

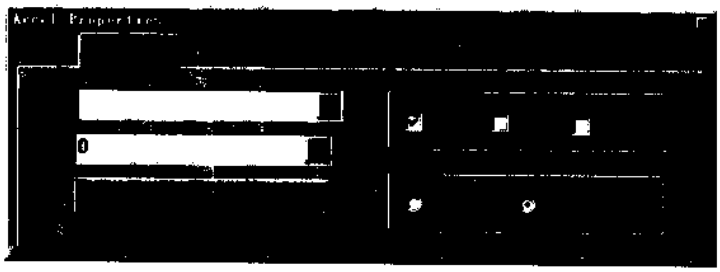


图 2.43 Accel Properties 对话框

Key 文本框中的合法输入为：

·0～255 间的某个整数，可以写成十进制、十六进制或八进制格式。Type 框的设置用于确定该数是作为 ASCII 码值还是虚拟键值。通常，一位数(0～9)被直接解释成相应的键，而不是 ASCII 码值。如果要输入 0～9 间的 ASCII 码值，则必须在数字前加两个 0(如 005)。

·单个键盘字符，大写的 A～Z 或数字 0～9 既可解释成 ASCII 码值，也可看成虚拟键值。其他字符都看成 ASCII 字符。

·单个 A～Z 间的字符(大写形式)前加符号^(如^D)，表示同时按下 Ctrl 和字母键所产生的组合键的 ASCII 值。

·任何合法的虚拟键标识符。可以单击 Key 文本框右侧的下箭头来选择标准的虚拟键标识符。




当输入 ASCII 值时，Modifiers 框中的修改符 Ctrl 和 Shift 是无效的，即不能通过符号^和控制键的组合来产生相应的虚拟键值。

此外，可以先单击 Next Key Typed 按钮，再按键盘上的相应键来定义加速键。

定义加速键后，如果要从加速键表中删除某一加速键，可以先指定要删除的加速键再按下 Del 键。还可以将加速键从一个资源文件移动或复制到另一资源文件中。

### 2.5.7　串编辑器

串表是一种 Windows 资源，它包含应用程序用到的所有串的 ID 号、值和标题。例如，状态栏提示可以放在串表中。每个应用程序只能有一个串表。在串表中，串以 16 个为一组构成段或块。某一串属于哪一段取决于该串的标识符值。例如，标识符值为 0～15 的串放在第一段，为 16～32 的串放在第二段，等等。要将串从某一段移到另一段，必须改变其标识符值。通常，每个串只在使用时才调入内存。

串编辑器用于编辑应用程序的串表资源。使用串编辑器，可以在串表中浏览串，向串表添加新的串，从串表删除某一串，将串从某一段移到另一段，将串从某一资源文件移到另一资源文件，修改串及其标识符，等等。

图 2.44 是打开某一应用程序的串表资源后的串编辑器。

| ID: MAIN FRAME | 5/8 | Chap6\n\nChap6\n\nChap6.Document\nChap6 |
|---|---|---|
| ID_BUILD_DEBUG_GO | 32777 | Starts or continues the program |
| AFX_IDS_APP_TITLE | 57344 | Chap6 |
| AFX_IDS_IDLEMESSAGE | 57345 | 就绪 |
| ID_FILE_NEW | 57600 | 建立新文档\n新建 |
| ID_FILE_OPEN | 57601 | 打开一个现有文档\n打开 |
| ID_FILE_CLOSE | 57602 | 关闭活动文档\n关闭 |
| ID_FILE_SAVE | 57603 | 保存活动文档\n保存 |
| ID_FILE_SAVE_AS | 57604 | 将活动文档以一个新文件名保存\n另存为 |
| ID_FILE_MRU_FILE1 | 57616 | 打开该文档 |
| ID_FILE_MRU_FILE2 | 57617 | 打开该文档 |
| ID_FILE_MRU_FILE3 | 57618 | 打开该文档 |
| ID_FILE_MRU_FILE4 | 57619 | 打开该文档 |
| ID_FILE_MRU_FILE5 | 57620 | 打开该文档 |
| ID_FILE_MRU_FILE6 | 57621 | 打开该文档 |
| ID_FILE_MRU_FILE7 | 57622 | 打开该文档 |
| ID_FILE_MRU_FILE8 | 57623 | 打开该文档 |
| ID_FILE_MRU_FILE9 | 57624 | 打开该文档 |
| ID_FILE_MRU_FILE10 | 57625 | 打开该文档 |
| ID_FILE_MRU_FILE11 | 57626 | 打开该文档 |

图 2.44　串编辑器

在串编辑器中，串表中的每个段用水平线分开。如果要在串表中添加新的串，则在要添加串的串段中，选择新项方框输入串标识符，或者选择某个串，再按 Ins 键，弹出 String Properties 对话框(图 2.45)。在 ID 文本框输入串标识符和值，在 Caption 文本框输入串标题。字符串的 Value 值由 Visual C++ 自动分配。

如果要从串表删除某个串，可以先选择该串再按 Del 键。如果要修改串及其标识符，则先指定欲修改的串，然后按 Alt + Enter 键，弹出 String Properties 对话框，从中修改串。

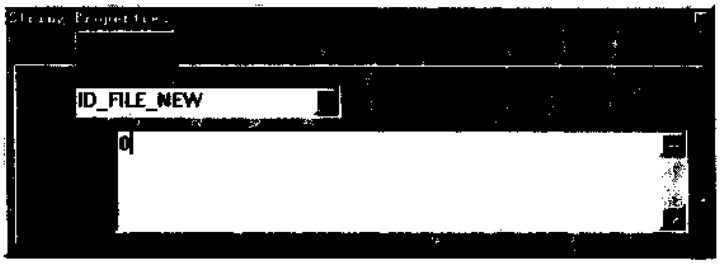



图 2.45 String Properties 对话框

### 2.5.9 图形编辑器

图形编辑器由一套功能强大的绘图工具组成，用于绘制位图、图标和光标。图 2.46 是正在编辑某一位图的图形编辑器窗口。

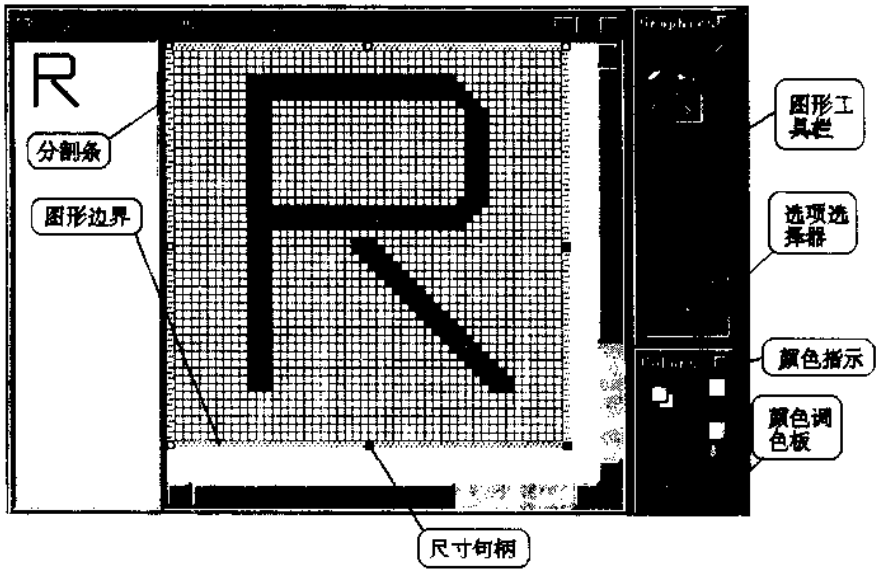



图 2.46 图形编辑器

图形编辑器窗口用两个视图来显示图形，视图间用分割条隔开。左边的视图以实际尺寸显示图形，右边的视图则是放大后的图形(缺省放大 6 倍)。

图形工具条由两部分组成。选项选择器用于设置绘图选项(如画刷宽度)；工具条由选择(Select)、区域选择(Select Region)、颜色拾取(Select Color)、橡皮擦(Erase)等 21 个工具组成，供用户绘制图形、输入文本、擦除并管理视图。

颜色调色板由两部分组成。颜色指示反映当前所用的前景色和背景色(对于图标和光标而言，则反映屏幕色和反转色)；调色板用于选择前景色或背景色。

进入图形编辑状态后，菜单栏将出现 Image 菜单。可以使用 Image 菜单的命令选项来

处理图像和管理颜色调色板。

1.设置位图属性

要改变位图属性，先打开要改变属性的位图，然后从 View 菜单选择 Properties 命令或按 Alt + Enter 键，弹出 Bitmap Properties 对话框(图 2.47)。

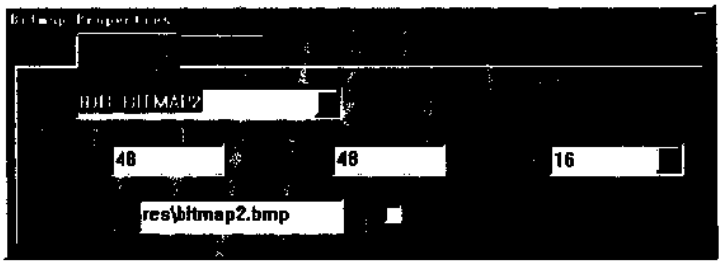



图 2.47　Bitmap Properties 对话框

Bitmap Properties 对话框含两个标签，即 General 和 Palette 标签。其中，General 标签用于设置位图资源的标识符、大小和颜色数等；Palette 标签用于改变位图中的所有颜色属性。

2. 创建图标和光标

图标和光标的编辑操作和位图基本上相同，但图标和光标具有与位图不同的属性。例如，对于不同的显示设备，每个图标或光标可以包含不同的图像。此外，光标具有热点(hotspot)——Windows 使用热点来跟踪光标位置。

在创建新的图标或光标时，图形编辑器首先创建 VGA 图像。图像开始以屏幕色来填充(图 2.48)。对于光标而言，热点被初始化为图像的左上角，坐标为(0, 0)。

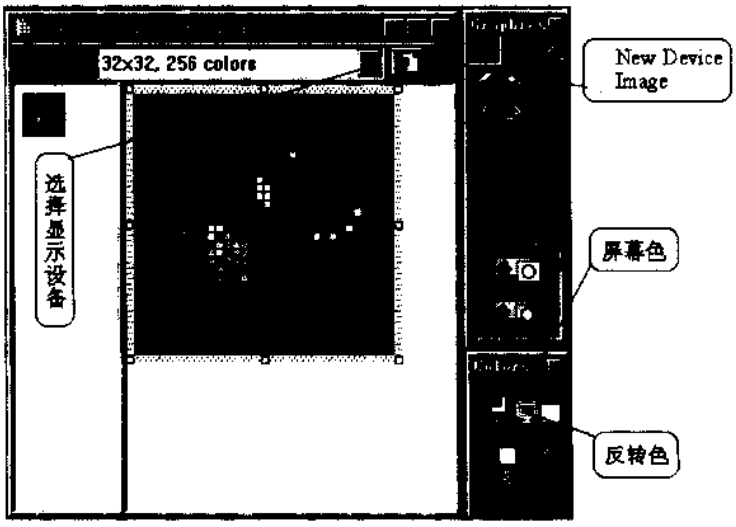



图 2.48　创建新图标时的图形编辑器

在创建新的图标或光标时，必须指定好目标显示设备。当打开图标或光标资源时，与当

前显示设备最为匹配的图形被自动打开。缺省时，图形编辑器支持如表 2.12 所示显示设备：

表 2.12 图形编辑器支持的显示设备

| 显示设备 | 颜色 | 宽度 | 高度 |
|---|---|---|---|
| Monochrome | 2 | 32 | 32 |
| Small | 16 | 16 | 16 |
| Normal | 16 | 32 | 32 |
| Large | 256 | 64 | 64 |

选择目标显示设备的方法为：单击图形编辑器窗口控制栏的 New Device Image 按钮，弹出 New Icon Image 对话框(图 2.49)。从对话框的 Target device 列表框选取所要的显示设备。

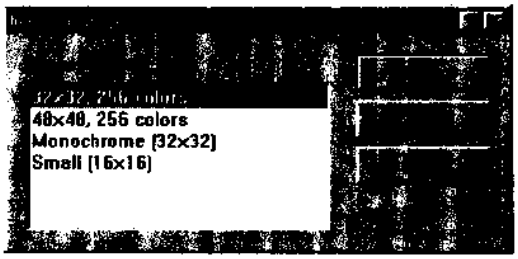



图 2.49 New Icon Image 对话框

如果要创建除标准显示设备外的其他定制显示设备，在 New Icon Image 对话框中单击 Custom 按钮，从弹出的 Custom Image 对话框(图 2.50)输入宽度、高度和颜色数。

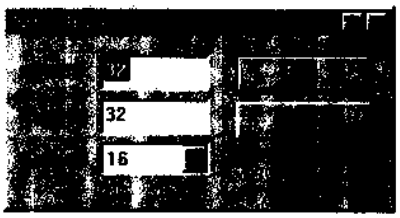



图 2.50 Custom Image 对话框

如果双击屏幕色或反转色指示器，系统将弹出 Custom Color Selector 对话框，从中可以对屏幕色或反转色进行定制。

在图形编辑器中创建光标时，控制栏将显示光标热点坐标和 Set Hotspot 按钮。缺省时热点坐标为(0,0)。如果要改变热点坐标，方法为：单击控制栏中的 Set Hotspot 按钮，然后单击要设为当前热点的位置即可。

### 2.5.10 工具条编辑器

工具条编辑器用于创建工具条资源并可以将已有位图转换为工具条资源。工具条编辑器以图形方式显示要处理的工具条和正被选择的工具按钮图像。图 2.51 是打开某一工具

条后的工具条编辑器。

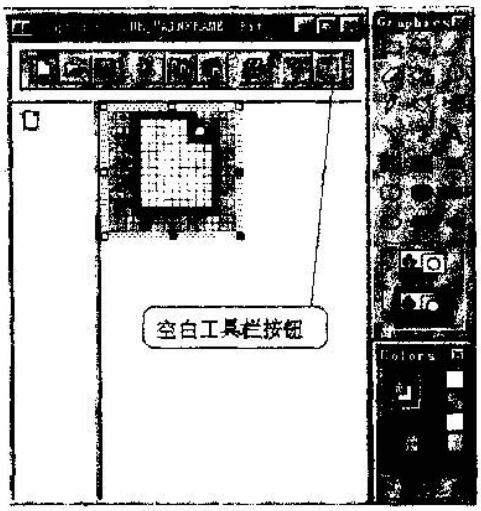



图 2.51 工具条编辑器

与图形编辑器类似，工具条编辑器用两个视图来显示被编辑的按钮图像。视图间用分割条分隔。被编辑按钮图像的上面是要处理的工具条，工具条中由模糊边界包围的是正被选择的按钮。

创建新的工具条有两种方法：一是直接创建，另一是将已有的位图转换为工具条。

如果要直接创建，请从 Insert 菜单选择 Resource 命令，在弹出的 Insert Resource 对话框中选择 Toolbar，然后单击 New 按钮。进入工具条编辑器后直接编辑。

如果要将已有的位图转换为工具条，方法为：

(1) 在图形编辑器中打开已有的位图资源。

(2) 从 Image 菜单中选择 Toolbar Editor 命令，弹出 New Toolbar Resource 对话框。在对话框中设置与位图匹配的图标图像的高度和宽度。然后，单击 OK 按钮进入工具条编辑器。

(3) 完成转换后，从 View 菜单选择 Properties 命令，从弹出的属性对话框设置工具条按钮的命令 ID。

创建工具条资源后，可以使用 ClassWizard 将工具条按钮与源代码连接。

## 2.6 本章小结

Visual Studio 为用户提供了一个集编辑、编译、连接、调试为一体的集成编程环境。本章对 Visual C++ 的编程环境——特别是资源编辑器，作了详细的讨论。通过学习本章，读者应当了解 Visual C++ 的大致功能，为以后学习编程打下基础。在以后各章中，我们会不加特别声明地应用 Visual C++ 集成环境所提供的功能，相信读者在使用过程中会为 Visual C++ 为我们提供的方便的编程工具而惊叹。

# 第三章 “Hello World!”应用程序

**本章要点**

- 应用程序的框架
- 应用程序的框架示例
- MFC 库消息映射
- “Hello World!”应用程序

从本章起，读者就要亲手编写 Windows 应用程序了。使用 Visual C++ 的 MFC 编写应用程序时，一般都要使用 Visual C++ 提供的模板程序。各种各样模板程序已为读者搭好了应用程序的框架，读者只用在此基础上进一步编写自己的应用程序。

## 3.1 应用程序的框架

应用程序的框架可以认为是，“为了生成一般的应用所必须的各种软组件的集成”。此时，或许读者还不太明白。如果读者真的想了解什么是应用框架，那么您就得把这本书读完，你很快就会在本章中看到一个应用框架例子。这将是学习应用框架的一个开端。

C++ 之所以是一种倍受欢迎的语言，原因之一就在于它可以通过类库来进行扩展。有一些类库是随编译器一起提供的，有一些是由其它软件公司销售的，还有一些则是由用户自己开发的。类库是一个可以在应用程序中使用的相互关联的C++ 类的集合。例如，矩阵类库可以用来完成一般的涉及矩阵的数学运算，通讯类库可以用来完成串联方式的数据交换。有些情况下，人们需要创建一些所提供类的对象，而在有些情况下，又可能需要派生出自己的类，所有这些都完全依赖于特定类库的设计。

应用框架是一种类库的超集。一般的类库只是一种可以用来嵌入到程序中的孤立的类的集合，但应用框架却定义了程序的结构。这听起来似乎反映了两者之间的差别，事实也的确如此。大多数 Windows 类库，包括 Microsoft 基本类库版本 2.5、Borland OWL 以及 Microsoft 基本类库版本 2.5，都被认为是一种应用框架。然而 Microsoft 基本类(MFC)库版本 2.5 却比其它类库提供了更多的重要特性。

## 3.2 应用框架示例

前面的内容仅是泛泛而谈，现在来看一些实际的代码。这些代码并非伪代码，而是真正的可以编译并运行的使用了 MFC 库的程序代码。这里给出的例子是一个经典的例子

"Hello,World!"。该例子是一个具有最短代码同时又可以运行的 MFC 库应用实例。最好将它同相应的 Windows SDK 应用程序比较一下！现在,读者还不必弄懂每一个程序行,不过,最好别怕麻烦,把它敲进去实际试试。等到了下一章,读者就可以使用真正的应用框架了。




按照惯例,MFC 库类名用大写字母"C"打头。

下面给出的是 MYAPP 应用的头文件和源代码文件。其中包含了两个类,CMyApp 和 CMyFrame,它们都是从 MFC 库基类中派生出来的。首先让我们来看看 MYAPP 应用程序的 MYAPP.H 头文件:

```
//应用程序类
class CMyApp:public CWinApp
{
public:
    virtual BOOL InitInstance();};

//框架窗口类
class CMyFrame:public CFrameWnd
{
public:
    CMyFrame();
Protected:
    afx_msg void OnLButtonDown(UINT nFlags,CPoint point);
    afx_msg void OnPaint();
    DECLARE_MESSAGE_MAP()
};
```

现在,我们再来看看 MYAPP 应用的 MYAPP.CPP 源文件:

```
#include<afxwin.h>//声明基类的文件库头文件
#include"myapp.h"

CMyApp NEAR theApp;//CMyApp 的对象

BOOL CMyApp::InitInstance()
{
    m_pMainWnd=new CMyFrame();
    m_pMainWnd->ShowWibndow(m_nCmdShow);
    m_pMainWnd->UpdateWindow();
    return TRUE;
}
BEGIN_MESSAGE_MAP(CMyFrame,CFrameWnd)
```

```
    ON_WM_LBUTTONDOWN()
    ON_WM_PAINT()
END_MESSAGE_MAP()

CMyFrame::CMyFrame()
{
    Create("AfxFrameOrVIEW","MYAPP Application");
}
void CMyFrame::OnLButtonDown(UINT nFlags,CPoint point)
{
    TRACE("Entering CMyFrame::OnLButtonDown - %1x,%d,%d\n",
        (long)nFlags,point.x,point.y);
}
void CMyFrame::OnPoint()
{
    CPaintDC dc(this);
    dc.TextOut(0,0,"Hello.World!");
}
```

下面简要介绍一下程序中的一些基本元素：

**WinMain 函数** 记住 Windows 总是要求每个应用都要有 WinMain 函数，读者之所以在此见不到该函数，是因为它被隐藏在应用框架内部了。

**CMyApp 类** 类 CMyApp 的对象就代表了一个应用程序。该程序定义了一个单独的全程 CMyApp 对象 theApp。CWinApp 基类决定了 theApp 的大部分行为。

**应用程序的入口** 当人们运行该应用程序时，Windows 会自动调用应用框架内部的 WinMain 函数，WinMain 函数会去查找该应用程序的全程构造对象。该对象是由 CWinApp 所派生出的类的对象。读者别忘了，在C++中全程对象在主程序被运行之前就已经被构造好了。

**CMyApp的 InitInstance 成员函数** 当 WinMain 发现了应用对象时，它会自动调用 InitInstance 成员函数，该函数会进一步调用相应的函数来完成主窗口的构造和显示工作。在派生出的应用类中，读者必须重新设计 InitInstance 函数，因为 CWinApp 基类根本无法知道具体应用需要什么样的主框架窗口。

**CWinApp 的 Run 成员函数** Run 函数被隐藏在基类中，它被用来负责传递应用的消息，从而维护着应用的运转。WinMain 的调用 InitInstance 之后紧接着调用 Run。

**CMyFrame 类** CMyFrame 类对象代表着应用的主框架窗口。当构造函数调用基类 CFrameWnd 的 Create 成员函数时，Windows 将创建具体的窗口结构，应用框架会将所创建的窗口结构连接到C++对象中。为了显示所创建的窗口，必须调用基类中的 ShowWindow 和 UpdateWindow 成员函数。

**CMyFrame 的 OnLButtonDown 函数** 在这里，读者可以提前看到类库的消息处理机制。我们已经将鼠标的左键被按下这一事件映射到了 CMyFrame 的一个成员函数，读者将会

在第四章中学习有关类库消息映射的细节。以后读者就会明白，当用户按下鼠标的左键时，OnLButtonDown 函数就会自动被调用。该函数将会激活类库 TRACE 宏，以便在调试窗口中显示一个消息。

**CMyFrame 的 OnPaint 函数** 应用框架在每次需要重新绘制窗口时，都要调用这个非常重要的且已经被映射过的 CMyFrame 类的成员函数。在程序刚被运行时，当用户改变了窗口大小时，或者当窗口的全部或部分被重新显现时，OnPaint 成员函数都会被调用。CPaintDC 和图形设备接口(GDI)有关，我们将在后面的章节中介绍。TextOut 函数用来显示"Hello, World!"。

**关闭应用程序** 用户可以通过关闭主框架窗口来终止应用程序。这一举动将会引起一系列事件的发生，首先 CMyFrame 对象将被删除，然后退出 Run，进而退出 WinMain，最后再删除 CMyApp 对象。

读者最好再仔细看一下该例子，不过这次应该着重从整体来观察。该应用的大部分功能都包含在类库的基类 CWinApp 和 CFrameWnd 中。在编写 MYAPP 的过程中，作者遵循了一些简单的结构规则，并且亲自编写了派生类中的关键函数。C++ 允许编程者"借"许多代码而用不着去复制，这就好比编程者和应用框架之间是非常不错的合作伙伴，应用框架为编程者提供了大致的结构，而编程者只需将应用具体化。

现在，读者应该开始明白为什么应用框架不仅仅是一种类库。应用框架不但定义了应用的结构，而且它所包含的东西还不仅仅是C++ 的基类，读者已经见过的 WinMain 函数，以及其它一些用来支持消息处理、诊断、DLL 等等诸如此类的元素都包含在应用框架中。

## 3.3 MFC 库消息映射

在讲到上面例子中的 OnLButtonDown 成员函数时，读者或许会认为该函数是虚函数的一个非常典型的代表。基类将为鼠标事件消息以及其它一些标准消息定义一些虚函数，如果需要的话，派生类可以对这些函数进行重新设计。一些 Windows 类库确实是这么做的。

MFC 库应用框架并没有采用虚函数来处理 Windows 消息，相反，它通过一些宏来将特定的消息映射到派生类中相应的成员函数。为什么类库应用框架没有采用虚函数呢？这主要是因为以下原因：类库中包含了 5 个窗口类，分别和 Windows 的五种窗口类型相对应。如果采用虚函数的办法来处理消息，那么每个窗口基类就要对 140 条消息分别定义一个虚函数。C++ 对每个虚函数都要求有一个 4 字节的传递结构，称作"vtable"，无论该虚函数是否在派生类中被重新定义，"vtable"都必不可少。这样，对于每个特定类型的窗口或控件，应用都需要一个 2.8KB 大小的表来支持虚消息控制函数。

那么对于菜单命令消息及按钮命令消息来说，又该如何来设计消息控制函数呢？总不能将它们也定义成窗口基类中的虚函数吧，因为不同的应用会有不同的菜单项和按钮。MFC 库的这种消息映射系统就避免了使用大的 vtable 表，并且能够处理各种各样应用的命令消息。这种体制也允许某些非窗口类(如文档类和应用类)来控制命令消息。这种消息映射体制同 Borland 作为 OWL 的一个组成部分所提供的"动态传递表"系统也有所不同，它不

需要C++作任何扩展。

MFC库消息控制函数要求人们提供函数原型、函数体以及在消息映射中的入口，而ClassWizard会帮助人们将消息控制函数引入人们所设计的类中，只要人们从列表框中选择一个Windows消息ID，ClassWizard就会自动产生具有正确参数及返回值的代码。

## 3.4 "Hello World"应用程序

为了生成应用程序框架，启动Visual C++，从File菜单中选择New。Visual C++弹出New对话框，切换到Projects标签，项目的类型选择MFC AppWizard(exe)，在Project name:编辑框中输入Chap 3，在Location:编辑框中选择想放置项目的位置，如图3.1所示。然后单击OK按钮。

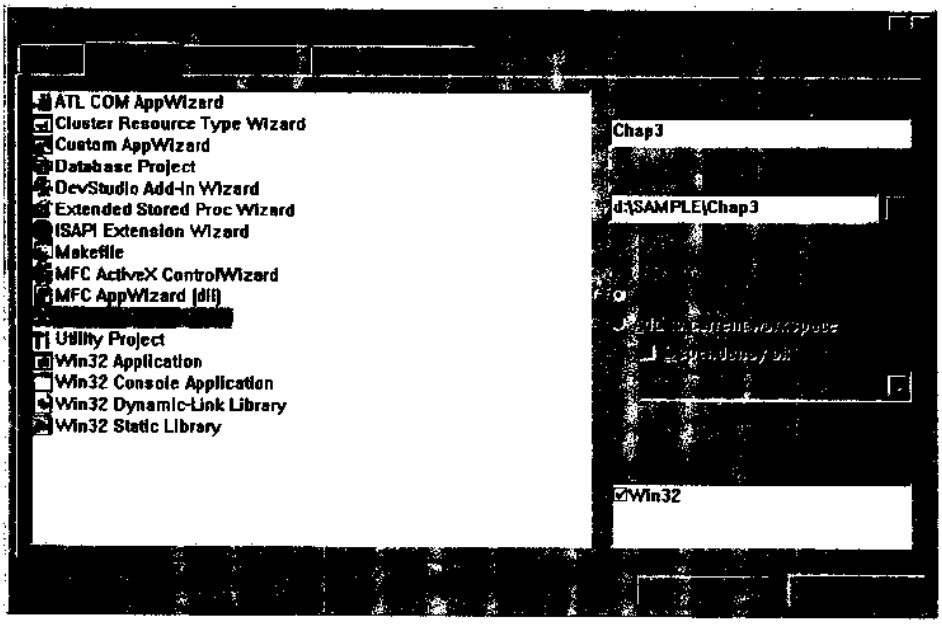



图3.1 Visual C++的New对话框

在单击OK按钮后，AppWizard开始询问关于应用程序特性的问题。这些问题包含六个"步骤"，每一步都集中于MFC框架的一个特定部分。正如在图3.2中看到的，每一步都有一个Next/Back按钮。这意味着如果走到了第5步，发现在第2步犯了一个错误，那么可以回到前面的步骤。这样做不会丢失已经输入的任何信息。图3.2显示的是第1步的画面。

第1步——应用程序的类型和语言的类型

在第1步里，系统提问建立什么类型的应用文件。读者可以选择Single文档(主窗口中只有一个窗口)，Multiple文档(如Microsoft Word或Sysedit，可以在主窗口内开多个子窗口)，或Dialog based(主窗口只不过是一个对话框)。Chap3使用单文档(Single document)。系统的缺省选项是多文档(Multiple document)。

读者也可以为资源选择语言。AppWizard 为应用程序生成了大量的基于文本的资源(菜单提示、对话框和工具栏帮助)。

在 What type of application would you like to create? 下选择 Single document 单选按钮,再单击 Next 按钮就到了第 2 步。

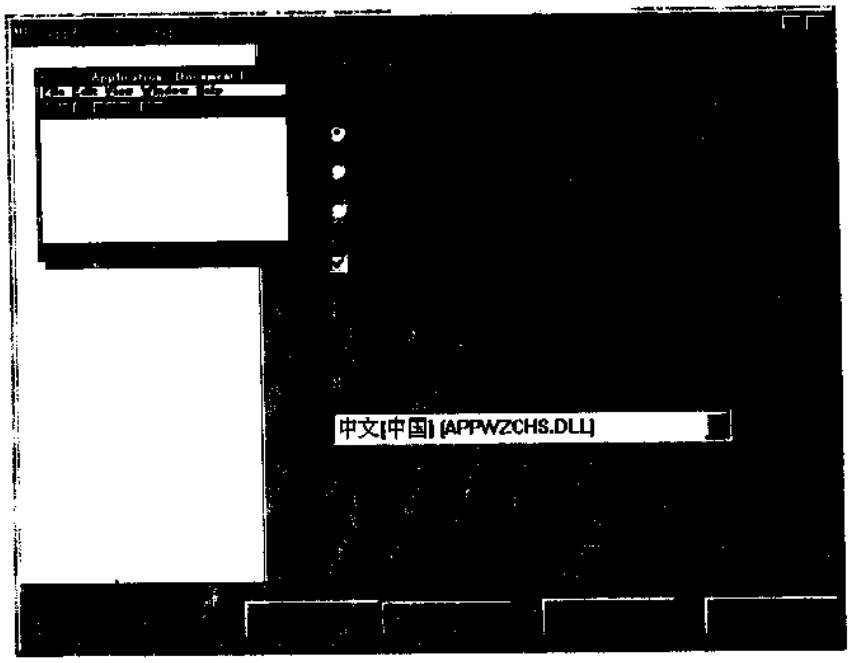



图 3.2 AppWizard 的第 1 步,从中选择应用程序的类型

第 2 步——数据库支持

第 2 步用于决定应用程序是否打算用 ODBC 类去存取一个数据库。图 3.3 显示了第 2 步的提问。

如上图所示,存在四种选择。缺省时表示不包括数据库支持。Chap3 使用这个选择,因为它不需要 ODBC 支持。如果想要 ODBC 支持,只需让 AppWizard 包含 ODBC 头文件即可,或者为应用程序指定一个数据库。如果选择了后者,数据库必须已经存在。在继续下面的内容之前,AppWizard 就要求用户选择一个数据库。

指定一个数据库会引起 AppWizard 生成由 CRecordSet 和 CRecordView 的派生类来支持数据库。这些选择生成一个基于表格的应用程序。这样的应用程序易于查看和更新数据库。文件支持的扩充部分为应用程序增加了文档序列化。这两个选项允许用 ODBC 维护数据库。缺少文件支持并不影响对数据库的读写。

由于该应用程序不使用数据库支持,单击 Next 按钮就可以到下一步。

第 3 步——OLE 支持

第 3 步必须有 OLE 支持,如图 3.4 所示对话框即为该步所特有的。

缺省情况下,代码中没有对 OLE 的支持,Chap3 不会是 OLE - aware,所以需要使用缺省值。

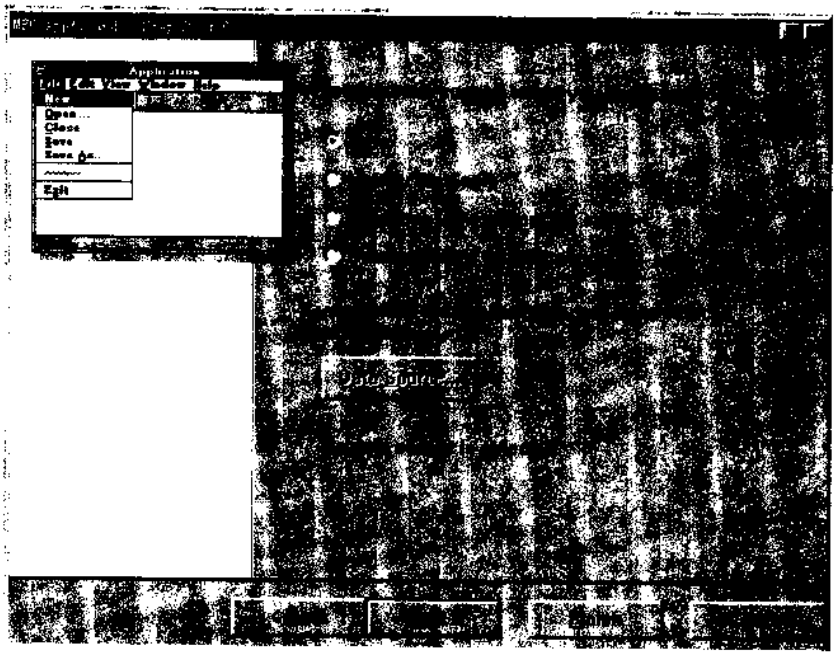


图 3.3 AppWizard 的第 2 步,从中选择是否要用 ODBC 支持

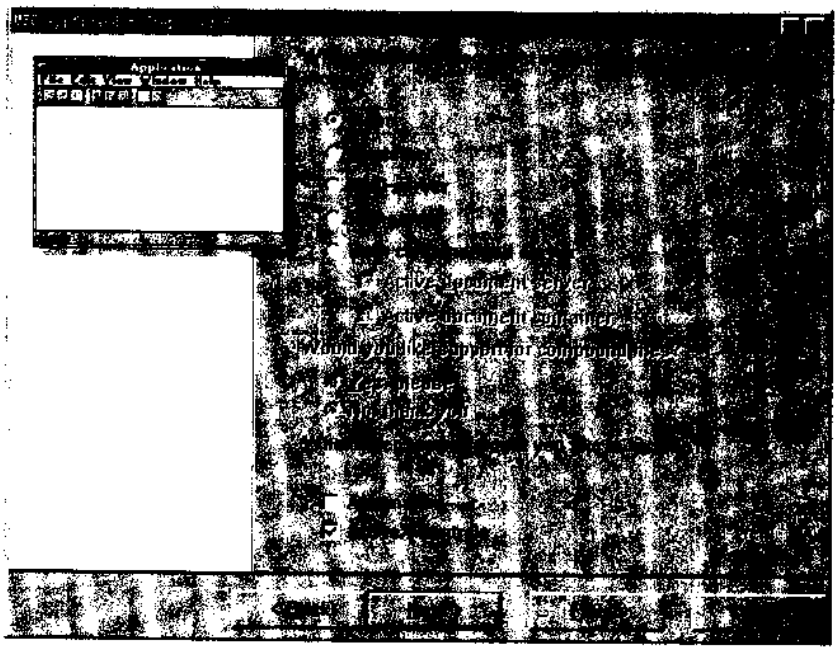


图 3.4 AppWizard 的第 3 步,从中选择是否要用 OLE 支持

应用程序可以是一个 OLE 容器或服务器,或二者兼具。如果一个应用程序就是一个

OLE 容器,它可以包含链接和嵌入对象。容器不能为其他的 OLE 程序提供支持,它只能维护嵌入对象。另一方面,服务器创建复合文档对象,使其嵌入 OLE 容器。如果选择了 Mini-server,应用程序就不能独立运行,而是被调用使之为其他程序建立 OLE 对象。完备服务器能够独立运行,并为其他应用程序建立 OLE 对象。从图 3.4 的列表显示中可以看到,一个应用程序可以同时作为容器和服务器。

如果想使应用程序面向 OLE 自动化服务,则选择"Yes, please"。否则,就不能被其他使用 OLE 自动服务的程序所访问。

不管在什么情况下,单击一下 Next 按钮就进入第 4 步。

第 4 步——应用程序的特性和高级选项

第 4 步是最复杂的一步,因为它有许多可能的选择。图 3.5 显示了第 4 步的画面。

在生成的应用程序中,读者想让 non - OLE 和 ODBC 具有 MFC 哪些特性? 第 4 步中的问题是与这些有联系的。在缺省的情况下,Docking toolbar(可停靠工具条),Initial status bar(初始状态栏),Printing and print preview(打印和打印预览)和 3D controls(使用三维控制项)等设置都处于打开状态。

MFC 应用程序通常有一个与主窗口相连接的工具条。AppWizard 提供了一组缺省位图,读者可以加入一个自己的按钮,如果可停靠工具条被选取,那么应用程序就得到它的支持,它既可以呆在窗口的顶部也可以像子窗口一样移动。在应用程序主窗口的上、下、左、右各个位置,可停靠工具条均可被定位。

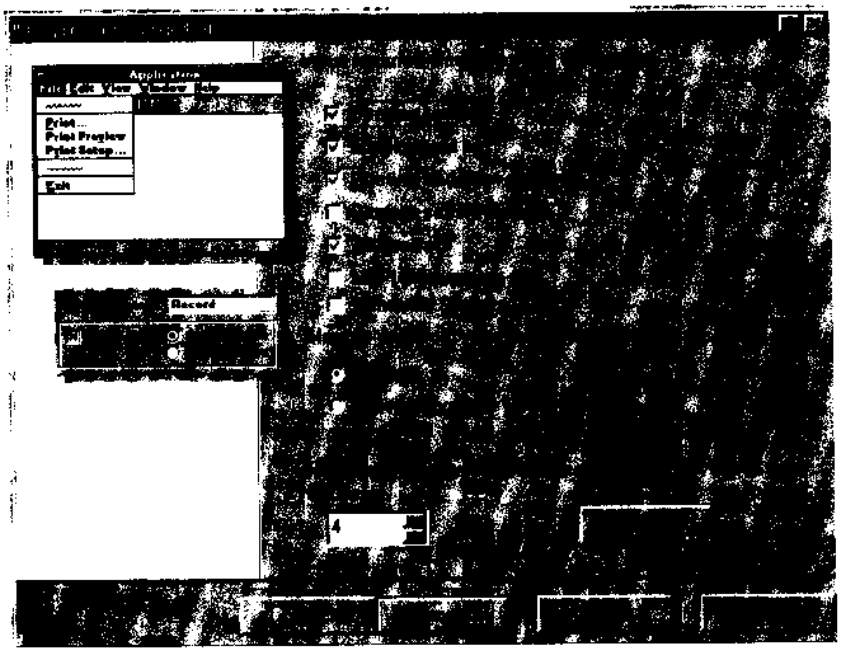



图 3.5 AppWizard 的第 4 步,从中选择应用程序的其他功能

在应用程序主窗口底部,Initial status bar 打开了一个状态栏。当这个状态栏在界面上移

动时,所创建的应用程序用它给用户提供帮助提示。例如,在出现一个串表时(用在第1步选择的语言编写的),就可以在代码中使用这个状态栏在应用程序运行时为用户提供信息。如果最初未用状态栏(在应用程序启动时),就不必选取这个按钮。

选择 Printing and print preview 为应用程序增加了支持标准打印和支持在屏幕上进行打印预览的程序代码。标准打印意味着生成一些类,这些类帮助代码支持打印而不需要编写打印程序。同时也提供了支持打印预览(每次1页或2页的打印预览,预览的缩放,等等)的类。然而,打印预览代码是非常类似的。为了使预览效果更好,对于大部分的应用程序来说都可能需要修改。打印预览的缺省代码没有提供页眉、页脚或特殊格式化的打印功能。

如果选择了上下文相关帮助,AppWizard 将生成一系列 RTF(Rich Text Format)文件,以及 Windows 帮助编译程序使用的支持文件。同时还为生成的应用程序菜单栏增加帮助菜单。RTF 文件知道应用程序的信息占位。为了用户能够使用,还需要对该 RTF 文件进行修改。虽然可以用任何一个编辑器来编辑 RTF 文件,但最好使用一个能理解 RTF 文件的程序(如 Microsoft Word)来修改帮助文件。在缺省状态下,不选择上下文相关帮助。Chap3 使用联机帮助,所以不选择这个选项。

最后,如果 3D controls 按钮被打开,在应用程序界面上将有一个"倾斜的"外观式样。

另外一个区域与最近使用文件的列表有关,MFC 用列表维护应用程序,缺省值是 4。MFC 为应用程序维护一个 INI 文件,这个应用程序包含它最近用到文件的列表清单。这个列表清单将显示在应用程序 File 菜单的底部。

Advanced 按钮产生另外一个对话框,这个对话框包含查看应用程序窗口的选择项。图 3.6 显示了第一个 Advanced Options 对话框。

这个对话框用于指定文档模板设置。换句话说,这个对话框让所生成的应用程序使用文件指定的设置。若应用程序要建立文件,就可以用这些选项标识文件。例如,如果决定应用程序建立的文件使用一个特殊的扩展名,那么就可以在 Filter Name 的描述字段内指定扩展名。当创建了用于文件打开和存储的公用对话框时,这个扩展名自动放入这些公共对话框的文件类型组合框内。因为 Chap3 不对文件进行写入,所以这些选项不发生变化。

该对话框顶部是几个选项标签。它们允许一个对话框有分离的"页",每一页带有完全不同的域、按钮和其他用户界面单元设置。单击标签可以激活相应的选项。

单击 Windows Styles 标签将出现关于窗口风格的对话框,如图 3.7 所示。

这个对话框为主窗口指定选项。主窗口是应用程序启动时,用户所看到的初级窗口,它包含建立的任何窗口。在这里读者可以把文本放入窗口中的标题栏中,并为应用程序指定可变边界、最小框、最大框和系统菜单。也可以指定应用程序主窗口是由最小化启动还是由最大化启动。

在该对话框的下面部分是 MDI child frame styles 选项,只有在建立 MDI 应用程序时才可以使用。它用于为 MDI 子窗口指定类似于 Main Frame 的选项。这些选项与主框架的是完全相同的,只是系统菜单没有复选框(因为 MDI 子窗口不能有系统菜单)。

Use split window 选项通过窗口顶部附近的一条线把一个窗口"分成"两部分。如果按住并拖动这条线,当前的子窗口将被分成两个独立的视窗。如果使用得当,它将增加一种查看

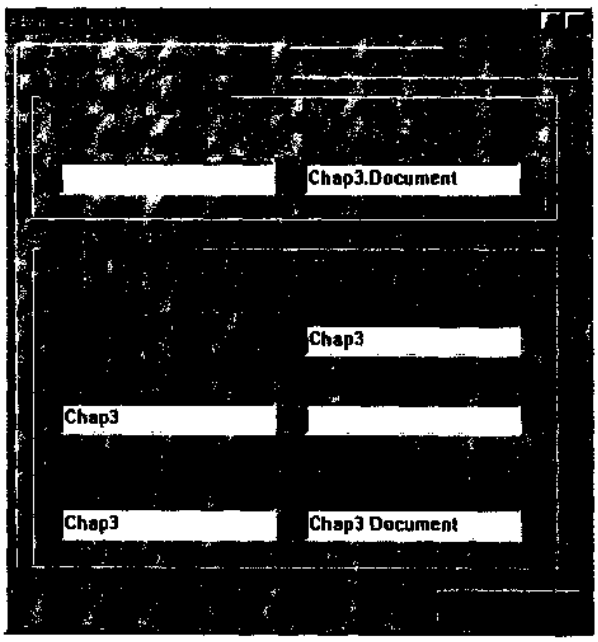



图 3.6 Advanced Options 对话框的 Document Template Strings 标签

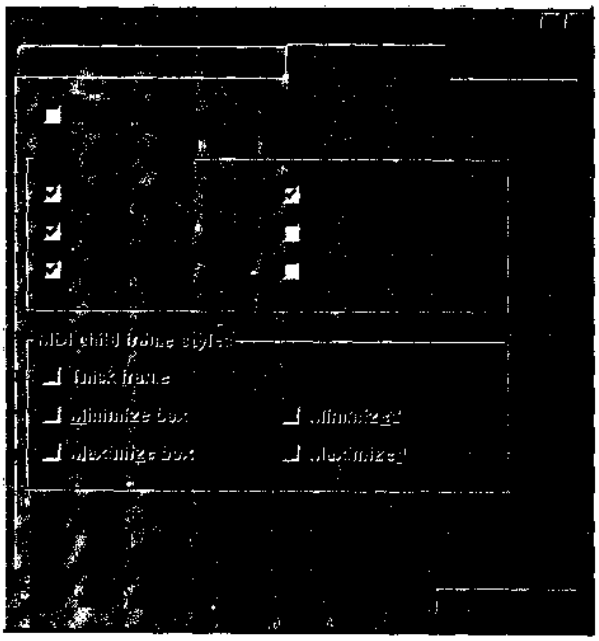


图 3.7 Advaced Options 对话框的 Windows Styles 标签

大文档的方法，用户可以用这种方法确定查看文档的哪一部分以及想看文档部分有多大。

Chap3 项目不需要改变任何缺省选项，所以先按下 Close 按钮再按下 Next 按钮就到了第

5步。

第5步——项目风格/源文件注释/MFC库类型

第5步处理源代码注释,项目文件类型以及MFC与应用程序联系的方式。图3.8显示了第5步的对话框。

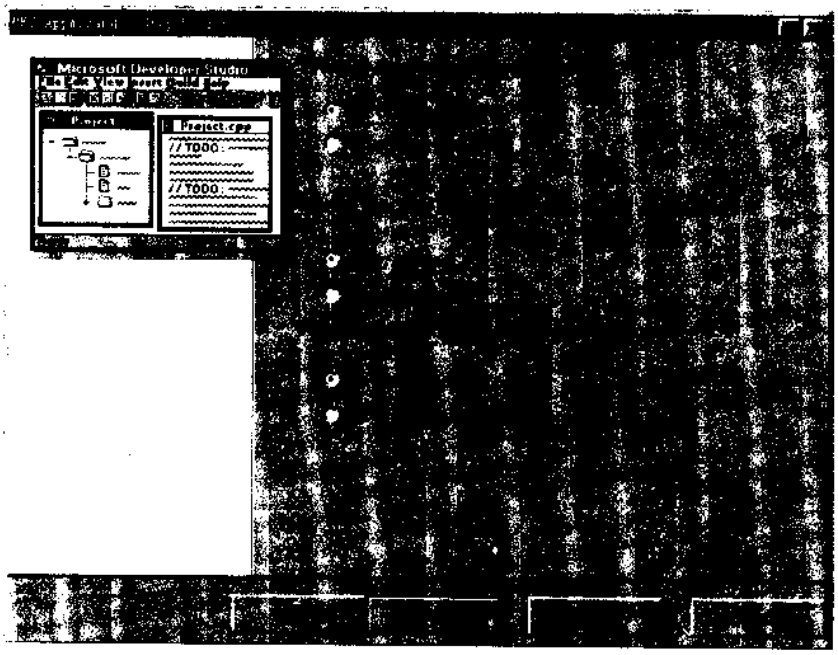



图3.8 AppWizard第5步的对话框

这些选项非常直观。“希望项目具有什么样的风格?”读者可以选择MFC标准的,或者选择窗口浏览器型的。在这里,我们用缺省选项。

接下来,“你希望产生源文件注释吗?”如果选择“Yes,please”,AppWizard就会在代码中放入注释告诉你在什么地方需要增加功能,通常是TODO注释形式。AppWizard也用少量的注释语句来提示用户程序是如何运行的。选择“No,thank you”,AppWizard将不产生任何注释,缺省状态是“Yes,Please”。

最后,需要决定是否要用MFC联入应用程序(借助于一个静态库),或者是否打算用MFC作为一个动态链接库。到底选择哪一个完全取决于编程者。链接DLL会使可执行文件变小。如果正在开发应用程序,这会节省硬盘的空间。缺省状态为链接DLL。链接一个静态库意味着少了一个文件分配给应用程序。

对于这个应用程序,不要去改变缺省状态中的任何选项。按下Next按钮就到了第6步,这是最后一步。

第6步——类名/文件名

在第6步中,基于对第1步~第5步问题的回答,AppWizard会把将要建立的新类的名称通知用户。图3.9显示了第6步的对话框。

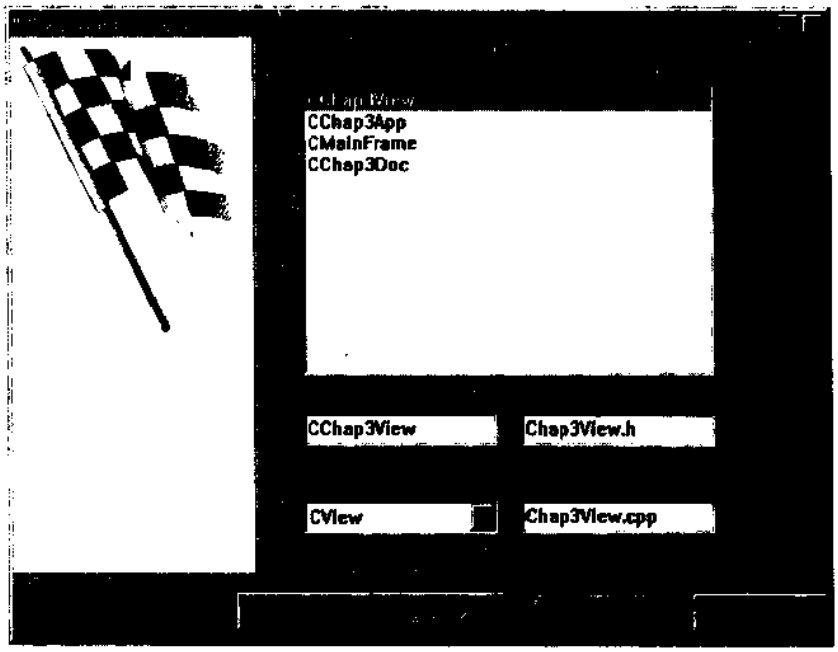



图 3.9 AppWizard 第 6 步,告知 MFC 产生的类名称

此时,AppWizard 将为 Chap3 应用程序建立四个新类。CChap3App 是应用程序类,由 CWinApp 派生出来转变成的。如前所述,所有的 MFC 应用程序必须有一个由应用程序自身定义的从 CWinApp 派生来的类。第二个类 CMainFrame,是一个拥有应用程序主窗口的类。CChap3Doc 和 CChap3View 是该应用程序的文档及视窗类。

如果不喜欢由 AppWizard 建立的名字,可以改变它们。如图 3.9 所示,CChap3View 在类列表框中被高亮显示。在这个列表框下面,AppWizard 在 Class name:编辑框中显示类名。若想把 CChap3View 变成其他的名字,只需输入新名字即可。注意其他的区域都变暗。有些名字 AppWizard 是不允许改变的。因为如果改变这些名字,已经做的工作会停止,或者会产生一个不可用的应用程序。此时,如果想要改变主程序的实际文件名,最好是回到前面的几步中去。在 Base class:编辑框中,应用程序对象是从 CView 继承来的,用户还可以选择其它的类。但是,对于 CChap3App 类,用户只能选择 CWinApp 为其基类。此外,别无选择。这是由于各类由其用途决定了彼此的继承关系。

总之,不要随便改变这些域,按下 Finish 按钮就完成了该应用程序生成工作。

在 AppWizard 生成应用程序之前将显示出最后一个对话框,即显示 New Project Information 对话框,如图 3.10 所示。

这个对话框总结了 AppWizard 的功能。到目前为止,AppWizard 实际上没创建任何东西。如果单击 Cancel 按钮将返回第 6 步。然后可以修改所犯的错误或取消全部 AppWizard 对话。若所有问题回答正确,单击 OK 按钮继续下去,AppWizard 将开始生成应用程序。

接下来,从 ClassView 中找到 CChap3View 的 OnDraw( )函数,如图 3.11 所示。双击

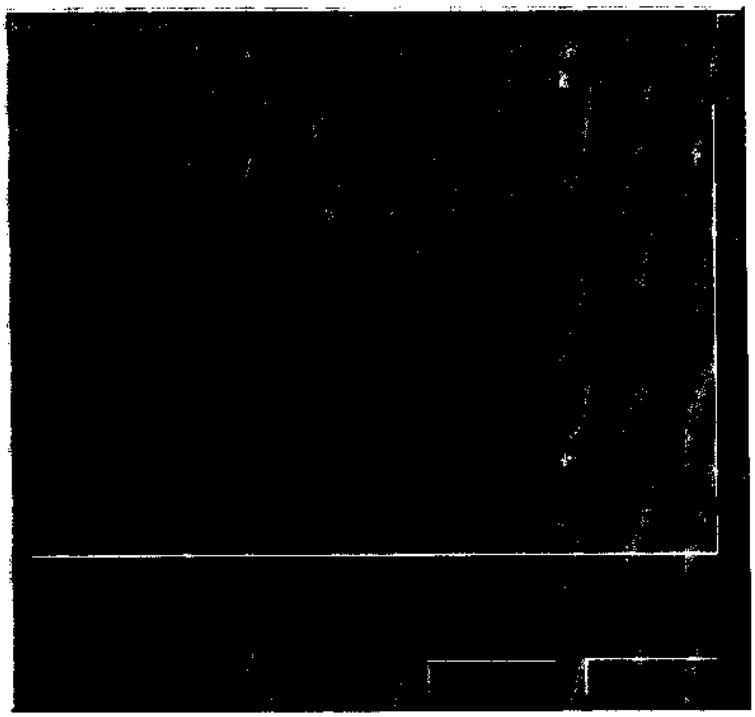


图 3.10 New Project Information 对话框,这是最后一个对话框

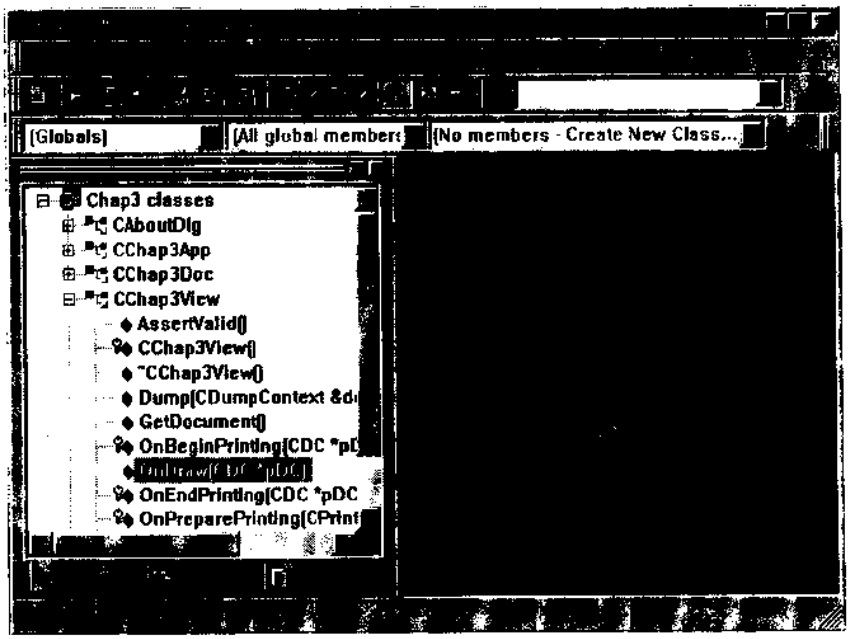



图 3.11 Workspace 的 classview 中的类、成员变量与函数

OnDraw( )函数,便会打开 Chap3View.cpp 文件,而且光标正置于 OnDraw( )函数中,如图 3.12 所示。在其中输入下列语句:

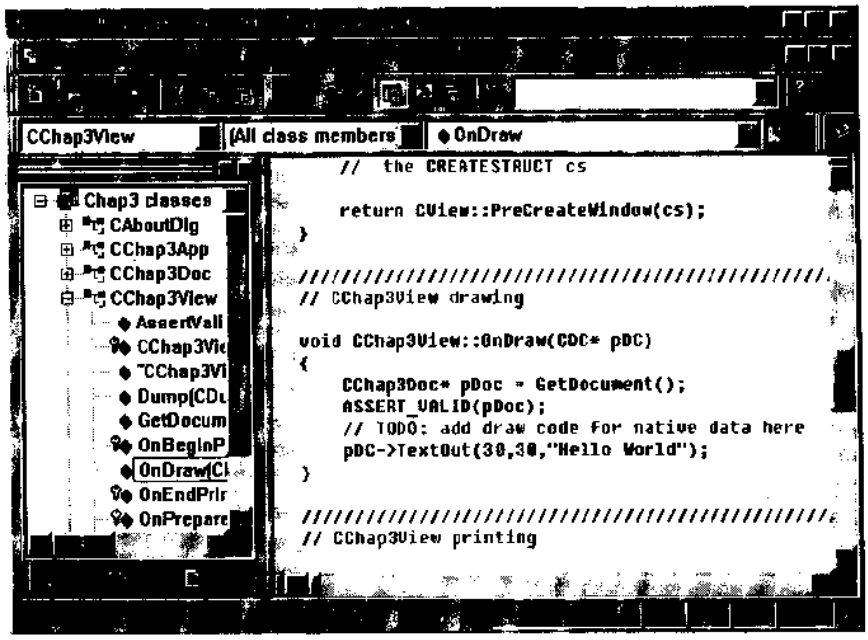



图 3.12 双击 CChap3View 的 OnDraw 函数后,OnDraw( )函数的编辑状态

```
pDC - - > TextOut(30,30,"Hello World!");
```

编译并运行该应用程序,将在应用程序客户区的距离左上角(30,30)的像素点处显示

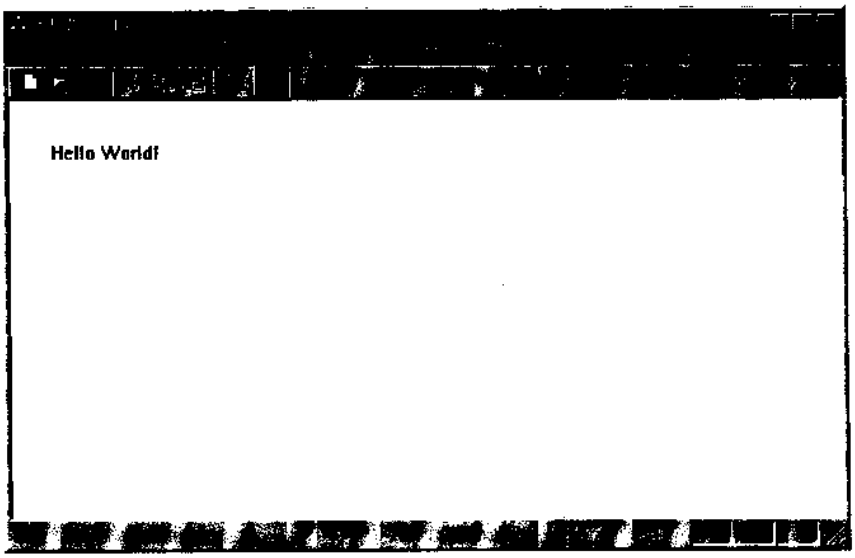



图 3.13 "Hello World!"应用程序的运行状态

"Hello World!",如图3.13所示。

至此,读者已完成了第一个Windows应用程序。读者仅在该程序中编写了一条语句,而其它大量的工作都是由AppWizard完成的。在以后的章节中,我们将进一步学习Visual C++是如何利用MFC完成复杂的应用程序的。

## 3.5 本章小结

本章讲述了Windows应用程序的框架,并引导读者完成了第一个Windows应用程序。通过学习本章,读者应当了解Windows应用程序的框架,掌握了如何利用AppWizard生成应用程序的基本框架。

# 第四章　图形设备接口(GDI)

**本章要点**

- 设备描述表的概念
- 创建和使用 GDI 对象
- 绘制点、直线、折线、曲线和封闭曲线

Windows 环境提供了功能强大的图形设备接口。当程序需要在屏幕或打印机上绘图时,就需要用到 GDI 函数。利用 GDI 函数可以方便地绘制点、线、矩形、多边形、椭圆、位图以及文本。本章将讲述在 Visual C++ 环境下如何使用 GDI。相信读者在学完本章后,就能够使用 Visual C++ 的基本绘图对象。

以前在 DOS 环境下,图形应用程序往往需要考虑各种特殊的硬件配置,比如需要考虑图形适配器的类型是 CGA、EGA 还是 VGA、SVGA。而 Windows 环境的图形设备接口(GDI)提供了图形系统的设备无关性,即开发图形应用程序时不用去考虑那些特殊的硬件配置,图形应用程序除了图形设备接口之外,还要求有与具体设备相关的设备驱动程序。有了这两个动态链接库,应用程序的开发就可以实现图形的设备无关性。图形设备接口动态链接库处理 Windows 应用程序的图形函数调用,并将其传递给相应的设备驱动程序。

另外,在 DOS 环境下,一个 DOS 应用程序独占整个显示屏幕、打印机和其它设备,而在 Windows 环境下,这些设备是系统中共享的资源,这也要求有一个统一的图形设备接口来支持各应用程序实现与图形有关的功能。

## 4.1　设备描述表

设备描述表是一种结构,它用来定义图形对象及其属性、并且影响 GDI 输出图形的方式。这些图形对象包括画线用的画笔(Pen)、填充图形所用的画刷(Brush)等。应用程序不能直接访问设备描述表,但可以通过函数调用检取设备描述表的句柄,从而间接地访问设备描述表。关于图形对象及其属性,将在后面陆续介绍。

Windows 中的设备描述表分为四种类型:显示器型、打印机型、内存型以及信息型。每种类型的设备描述表都有各自的特定用途,见表 4.1。

**表 4.1　设备描述表的类型和用途**

| 设备描述表 | 用途 |
| --- | --- |
| 显示器型 | 支持视频显示器上的绘图操作 |

（续表）

| 设备描述表 | 用途 |
| --- | --- |
| 打印机型 | 支持打印机和绘图仪上的绘图操作 |
| 内存型 | 支持位图上的绘制操作 |
| 信息型 | 支持设备数据的访问 |

显示器设备描述表是最常用的设备描述表。更进一步地，还可分为公共设备描述表和私有设备描述表。通过对函数 CWnd::GetDC()、CWnd::GetDCEx()或 CWnd::BeginPaint()的调用，应用程序就可以检取指定窗口的工作区的显示器设备描述表的句柄，可以在后续的 GDI 函数中使用显示器设备描述表在窗口工作区中画图。当完成绘图操作之后，应用程序应该调用 CWnd::ReleaseDC()或 CWnd::EndPaint()函数来释放设备描述表。

对于打印机设备描述表，应用程序是通过调用 CDC::CreateDC()函数来创建它，打印完毕后，应调用 CDC::DeleteDC()函数来删除它。

内存设备描述表是通过调用 CDC::CreateCompatibleDC()函数来创建的。它将创建一个与指定设备有兼容颜色格式的位图，内存设备描述表也常称为兼容设备描述表。

下面列出了与设备描述表有关的常用函数。

1. GetDC()函数

GetDC()函数用于获取指定窗口的工作区的显示器设备描述表，其原型说明如下：

```
CDC * CWnd::GetDC()
```

该函数不带任何输入参数。它用于检取一指针，它指向公用的或者私有的基于类的，为 CWnd 指定的类风格的设备描述表。对于公用设备描述表，GetDC()成员函数为每一次被检取的设备描述表指定缺省属性。对于私有设备场境，GetDC( )成员函数保持它先前所具有的属性不变。设备描述表可以用于后面的图形设备接口(GDI)函数中在客户区作图。

除非设备描述表属于窗口类，在完成作图之后，ReleaseDC()成员函数可以被调用来释放场境。因为在一个给定时刻，只有 5 个公共设备描述表是可用的，所以不释放设备描述表可能妨碍其他的应用程序访问设备描述表。当类在注册时，如果指定有 CS_CLASSDC、CS_OWNDC、CS_PARENTDC 等风格，则 GetDC( )成员函数将返回属于 CWnd 类的设备场境。

如果函数运行成功，则返回标识 CWnd 客户区的设备描述表，否则返回 NULL。返回的指针可能是临时性的，应该把它保存下来供以后使用。

如果需要在窗口的非工作区(如标题栏、菜单栏、滚动条等)进行画图操作，那么就应该使用 CWnd::GetWindowDC()函数来获取整个窗口的设备描述表，而窗口非客户区域中各元件(如标题栏、菜单栏、滚动条等)的几何尺寸可以通过调用 GetSystemMetrics()函数来检取。

2. ReleaseDC()函数

ReleaseDC()用于释放一个设备描述表，以便该设置描述表可以被其它的申请使用。ReleaseDC()成员函数的效果依赖于设备描述表的类型。关于每个对 GetWindowDC()成员函数和 GetDC()成员函数的调用，申请必须调用 ReleaseDC()成员函数。其原型说明如下：

```
int ReleaseDC(CDC * pDC);
```

其中 pDC 为待释放的设备描述表的指针

3．BeginPaint()函数

BeginPaint()函数为在指定窗口中画图作准备工作，并且把与画图有关的信息填写到 PAINTSTRUCT 结构中，其原型说明为：

```
CDC * CWnd::BeginPaint(LPPAINTSTRUCT lpPaint);
```

其中 lpPaint 参数是指向 PAINTSTRUCT 结构的指针。若函数调用成功，则返回指定窗口的显示器设备描述表的指针，否则返回 NULL。PAINTSTRUCT 结构包括了对画图操作有用的各种信息，它定义在 Winuser.h 头文件中：

```
typedef struct tagPAINTSTRUCT
{
    HDC         hdc;
    BOOL        fErase;
    RECT        rcPaint;
    BOOL        fRestore;
    BOOL        fIncUpdate;
    BYTE        rgbReserved[32];
}PAINTSTRUCT, * PPAINTSTRUCT, * NPPAINTSTRUCT, * LPPAINTSTRUCT;
```

在这个结构中，hdc 成员表示显示器设备描述表句柄，fErase 成员用于指定是否要擦除背景，而 rcPaint 则是指定需要画图的屏幕矩形结构。其余几个成员保留给 Windows 内部使用。

4．EndPaint()函数

EndPaint()函数用来结束指定窗口中的画图过程，其原型说明如下：

```
BOOL CWnd:: EndPaint(CONST PAINTSRUCT * lpPaint);
```

其参数的含义与 BeginPaint()函数中的相同，返回值总是 TRUE。




使用 EndPaint()函数，可以只更新窗口工作区中的部分区域(更新区域是由 InvalidateRect()函数或 InvalidateRgn()函数设定的，也可以在窗口改变尺寸、移动、滚动等影响工作区的操作之后，由系统设定更新区域。这样一来，就比重画整个屏幕要来得快。因此 BeginPaint()函数及 EndPaint()函数总是被用来响应 WM_PAINT 消息。

5．SelectObject()函数

SelectObject()函数用于把位图、画笔、画刷等对象选入指定的设备描述表中，用新对象替代同一类型的老对象，其原型说明如下：

```
CPen * CDC::SelectObject (CPen * pPen);
CBrush * CDC::SelectObject(CBrush * pBrush);
virtual CFont * CDC::SelectObject (CFont * pFont);
CBitmap * CDC::SelectObject(CBitmap * pBitmap);
int CDC::SelectObject(CRgn * pRgn);
```

该函数把一个对象选择进设备描述表。CDC 类专门为单种类型的 GDI 对象提供了五种版本,包括画笔、画刷、字体、位图和区域。新选择对象将替代同一类型的先前对象。例如:若 SelectObject()的 pObject 指向一个 CPen 对象,函数将以 pObjcet 指定的画笔替代当前画笔。

应用程序可以把位图选择进内存设备描述表,并且一次只能选择进一个内存设备描述表。位图的格式必须是单色的或是与设备描述表相兼容的;若不是这样,SelectObject()将返回 ERROR。

该函数返回当前被替代对象的指针。这个指针指向属于由 CGdiObject 派生出的某个类的对象,比如 CPen,这取决于函数所使用的版本。出现错误时返回 NULL。

使用区域参数的 SelectObject()函数的版本,具有与 SelectClipRgn()函数相同的功能。它的返回值可以是下列数值之一:

- COMPLEXREGION　　新的剪裁区域具有重叠的边界。
- ERROR　　设备描述表或区域无效。
- NULLREGION　　新的剪裁区域是空的。
- SIMPLEREGION　　新的剪裁区域没有重叠的边界。

6. DeleteObject()函数

DeleteObject()函数可以删除逻辑画笔、画刷、字体、位图、区域或调色板对象,并释放与该对象相关的所有系统资源,其原型说明如下:

```
BOOL CDC::DeleteObject(HGDIOBJ hObject);
```

其中参数 hObject 为逻辑画笔、画刷、字体、位图、区域或调色板的句柄。如果函数调用成功,则返回 TURE;要是对象句柄无效,或正在设备描述表中使用,那么返回值就是 FALSE。




当某个图形对象正在设备描述表中使用时,就不能删除该对象。

有关图形对象和图形方式的各种属性及操作,会在本章后面各节中陆续介绍。

## 4.2 GDI 对象

GDI 对象是 Windows 中图形设备接口的抽象绘图工具,它包括位图、画刷、画笔、调色板、字体、区域等。本节主要介绍画笔、画刷的基本概念及使用方法。

### 4.2.1　Pen(画笔)

画笔是 Windows 应用程序中用来绘制各种直线和曲线的一种图形工具。在绘图时,看上去似乎根本就没有使用画笔,照样画出了直线和曲线。其实,并不是没有使用画笔,而是因为在应用程序创建设备描述表时,就自动启用了一支默认的画笔,正是凭借这支无形的画笔,应用程序才有可能画出各种各样的直线和曲线。

画笔分为两种类型:修饰画笔和几何画笔。几何画笔用在对绘图有较高要求的场合。而通常画直线和曲线,使用修饰画笔就足够了。修饰画笔具有三种属性,分别为宽度、风格和颜色。本书后面如果不特别指明,那么所提到的画笔就是指修饰画笔。

#### 4.2.1.1　画笔的属性

画笔是一种逻辑笔,其属性如下:

· 宽度

修饰画笔的宽度用设备单位表示,而不是用逻辑单位表示,默认的画笔宽度是一个像素单位。

· 风格

画线时,有实线、虚线等多种画法,画笔的风格属性就是用来体现线条的这一外观特性,在 Windows 98 中可用的画笔风格有以下七种。

实线 ————————

划线 - - - - - - - - - - - -

点线 ..... .....

点划线 —·—·—·—·

双点划线 —··—··—··

空线

内框线 ————————

对于修饰画笔来说,内框线与实线几乎是一回事。

· 颜色

在画像素点时,画点函数可以直接定义点的颜色,而画各种直线、曲线的函数,要改变线的颜色,则必须通过设置画笔的颜色属性来实现。

#### 4.2.1.2　使用库存画笔

系统默认的画笔具有一个像素宽,风格为实线,颜色为黑色。系统中共存有三种画笔,称为库存画笔。应用程序可以使用 GetStockObject()函数来取出库存画笔。三种库存画笔分别是黑画笔、白画笔和空画笔,它们分别用常数 BLACK_PEN、WHITE_PEN 和 NULL_PEN 来表示。

#### 4.2.1.3 创建画笔

库存画笔虽然使用起来很简单,但毕竟种类太少了,而使用 CreatePen()函数,就可以创建出具有各种不同属性的画笔,CreatePen()函数的原型说明如下:

```
BOOL CPen::CreatePen(int nPenStyle,int nWidth,COLORREF crColor);
```

这个函数中的 nPenStyle、nWidth 和 crColor 这三个参数分别用来定义画笔的风格、宽度和颜色三种属性。画笔的七种风格分别由下表 4.2 中所列出的七个常量表示。

**表 4.2 画笔风格**

| 常量 | 含义 | 线型 |
|---|---|---|
| PS_SOLID | 实线 | |
| PS_DASH | 划线 | |
| PS_DOT | 点线 | |
| PS_DASHDOT | 点划线 | |
| PS_DASHDOTDOT | 双点划线 | |
| PS_NULL | 空线 | |
| PS_INSIDEFRAME | 内框线 | |

另外,还有一个 CreatePenIndirect()函数也是用来创建画笔对象,它的作用与 CreatePen()函数是完全一样的,只是画笔的三个属性不是直接出现在函数参数中,而是通过一个 LOGPEN 结构间接地给出。CreatePenIndirect()函数的原型说明如下:

```
BOOL CPen::CreatePenIndirect(LPLOGPEN lpLogPen)
```

其中参数 lpLogPen 是指向 LOGPEN 结构的指针,LOGPEN 结构中有三个成员,分别用于指定画笔的风格、宽度和颜色。LOGPEN 结构的具体形式定义如下:

```
typedef struct tagLOGPEN
{
    UINT      lopnStyle;
    POINT     lopnWidth;
    COLORREF  lopnColor;
}LOGPEN;
```

其中第一个和第三个成员分别表示画笔的风格和颜色,这与 CreatePen()函数中的第一个和第三个参数是完全一致的,而第二个成员 lopnWidth 虽然也是用于指定画笔的宽度,但是其类型却是 POINT 结构,而 POINT 结构中的 y 成员不起任何作用,只采用 x 成员来表示画笔宽度。

在实际应用中,既可以使用 CreatePenIndirect()函数来创建画笔,又可以使用 CreatePen()函数来创建画笔,其实这两个函数可以互换。有时用 CreatePen()函数来创建画笔显得更直接,而另外一些场合中,用 CreatePenIndirect()函数却显得更简练。

### 4.2.2 Brush(画刷)

画刷是 Windows 应用程序创建输出时使用的又一种绘图工具。封闭的图形(如矩形和椭圆等)可以使用当前选择的画刷填充。设备文本对象首次创建时便拥有缺省的画刷,这与画笔的情况类似。缺省画刷是将封闭图形的内部填充成全白色。为了改变当前画刷,既可以使用库存画刷,也可以创建定制的画刷,然后将其选入设备文本对象。

#### 4.2.2.1 画刷的属性

画刷具有风格与颜色两种属性。

·风格

画刷可以定制为实画刷(solid),阴影画刷(hatch)或图形画刷(pattern)。实画刷是用纯色填充图形的内部。阴影画刷用某种阴影模式填充图形内部。而图形画刷是将图形内部用位图一个接一个地填充。

·颜色

在初始化画刷时,通过相应的参数设置画刷的颜色属性,实现所需的颜色。颜色的参数的数据类型是 COLORREF,显示 RGB 值是一个 32 位的整数,包含红、绿、蓝三个颜色域,由 RGB(红,绿,蓝)的形式指定。每个域指定相应色彩的浓度,浓度值为 0 ~ 255,0 是最低强度,255 是最高强度。三种颜色的相对强度结合起来产生实际的颜色。

指定 RGB 值既可以是直接给出的数值,也可以是用宏 RGB()指定的。RGB()的定义为:

```
COLORREF RGB(BYTE bRed,BYTE bGreen,BYTE bBluc);
```

例如,RGB(255,0,0)为红色,RGB(0,0,0)为黑色。

#### 4.2.2.2 使用库存画刷

要选择库存画刷,只需调用 CDC::SelectStockObject()函数即可,其原型为:

```
virtual CGdiObject * CDC::SelectStockObject(int nIndex);
```

函数如果调用成功,则返回指向 CGdiObject 对象的指针(实际指向的对象是 CBrush 对象),否则返回 NULL。nIndex 是所要选入设备文本对象的库存画刷代码。对于画刷,其值如表 4.3 所示。

**表 4.3　库存画刷的宏代码**

| 宏代码 | 库存对象 | 宏代码 | 库存对象 |
|---|---|---|---|
| BLACK_BRUSH | 黑色画刷 | NULL_BRUSH | 空画刷(内部不填充) |
| DKGRAY_BRUSH | 深灰色画刷 | WHITE_BRUSH | 白色画刷(系统缺省画刷) |
| GRAY_BRUSH | 灰色画刷 | LTGRAY_BRUSH | 浅灰色画刷 |
| HOLLOW_BRUSH | 透明窗口画刷 | | |

例如,可以使用以下代码选择深灰色画刷:

```
pDC->SelectStockObject(DKGRAY_BRUSH);
```

#### 4.2.2.3 创建定制的画刷

在MFC类库中,CBrush类封装了GDI的画刷工具。在定义画刷对象时,要调用构造函数来创建缺省的画刷。使用画刷的主要步骤为:

(1)创建CBrush类对象。

(2)调用成员函数初始化画刷。

(3)将画刷对象选入当前设备文本对象,并将指针保存在原先的画刷对象中。

(4)调用绘图函数绘制图形。

(5)将原先的画刷对象选入设备文本对象,以恢复原来的状态。

画刷的初始化主要有以下几种:

(1)调用CBrush::CreateSolidBrush()来初始化实画刷,以便用纯色填充图形内部,函数原型为:

```
BOOL CBrush::CreateSolidBrush(COLORREF crColor);
```

其中参数crColor用于指定画刷的颜色。函数调用如果成功,则返回非零值,否则返回零。

(2)调用CBrush::CreateHatchBrush()来初始化阴影画刷。函数原型为:

```
BOOL CBrush::CreateHatchBrush(int nIndex,COLORREF crColor);
```

其中参数nIndex用于指定阴影模式,其值如表4.4所示。

**表4.4 阴影模式**

| 阴影模式 | 含义 |
|---|---|
| HS_BDIAGONAL | 反对角线 |
| HS_CROSS | 十字交叉线 |
| HS_DIAGCROSS | 斜十字交叉线 |
| HS_FDIAGONAL | 正对角线 |
| HS_HORIZONTAL | 水平线 |
| HS_VERTICAL | 垂直线 |

阴影线是出现在单色背景上等间隔的线。参数crColor用于指定画刷的前景色。

函数调用如果成功,则返回非零值,否则返回零。

(3)调用CBrush::CreatePatternBrush(CBitmap* pBitmap);

其中参数pBitmap为指向位图对象的指针。当使用该画刷填充图形时,图形内部将用位图一个接一个地填充。

函数调用如果成功,则返回非零值,否则返回零。

使用图形画刷要注意的是在删除图形画刷时,与画刷相联系的位图并未被删除。位图必须被单独地删除。

(4)使用 CBrush::CreateBrushIndirect()函数来创建画刷对象,但画刷的属性不是直接出现在函数参数中,而是以 LOGBRUSH 结构的形式间接地给出。

函数原型说明如下:

```
BOOL CBrush::CreateBrushIndirect(const LOGBRUSH * lpLogBrush);
```

其中的 lpLogBrush 参数是指向 LOGBRUSH 结构的指针。LOGBRUSH 结构中包含有关画刷的信息。LOGBRUSH 结构的具体形式定义如下:

```
typedef  struct  tagLOGBRUSH{//lb
    UINT        lbStyle;    //画刷风格
    COLORREF    lbColor;    //画刷颜色
    LONG        lbHatch;    //阴影线模式
}LOGBRUSH
```

函数调用如果成功,则返回非零值,否则返回零。

## 4.3　绘制基本图形

本节讲述了封装在 CDC 中的用于绘制一些基本图形的函数,并在本节末列举了一个如何使用这些基本绘图函数的示例程序 Chap4。

### 4.3.1　画点

画点是最基本的画图操作之一,可以通过调用 SetPixel()函数来实现的。SetPixel()函数可以在指定的坐标上设置指定的颜色,其函数原型说明如下:

```
COLORREF CDC:: SetPixel(int X,int Y,COLORREF crColor);
```

其中(X,Y)为点的坐标值(按逻辑单位),而 crColor 参数为点的颜色值。

如果函数调用成功,那么函数返回实际对像素点所设的 RGB 值。这个返回值未必等同于 crColor 所指定的颜色值,因为有时受设备限制,不能显示 crColor 所指定的颜色值,而只能取其近似值。如果函数调用失败,则返回值为 -1。

在示例程序 Chap4 中,有如下一段代码用于画点:

```
//绘制一组彩色点
pDC->TextOut(20,20,"Point");
pDC->SetPixel(100,20,RGB(255,0,0));
pDC->SetPixel(110,20,RGB(0,255,0));
pDC->SetPixel(120,20,RGB(0,0,255));
pDC->SetPixel(120,20,RGB(255,255,0));
pDC->SetPixel(140,20,RGB(255,0,255));
pDC->SetPixel(150,20,RGB(0,255,255));
pDC->SetPixel(160,20,RGB(0,0,0));
```

### 4.3.2 画直线

画线操作也是常用的绘图操作之一。LineTo()和 MoveTo()函数就是用来实现画线功能的两个函数,两个函数配合使用,就能完成任何绘制直线和折线的操作。

首先值得注意的是在画直线时,总存在一个称为当前位置的特殊位置。每次画直线都是以当前位置为起始点。画直线操作结束之后,直线的结束点位置又成为了当前位置。设置当前位置的理由是为了提高画线操作的效率。因为在大多数场合下,总是一条直线连着另一条直线,首尾相接,有了当前位置,会自动隐含地传递信息,就避免了每次画线时都要给出两点坐标。

LineTo()函数正是以当前位置所在点为直线始点,另指定直线终点,画出一段直线。其原型说明如下:

```
BOOL CDC:: LineTo(int nXEnd,int nYEnd);
```

直线终点位置由 nXEnd 和 nYEnd 参数指定。如果函数调用成功,那么(nXEnd,nYEnd)位置就成为当前位置,并返回 TRUE;否则,返回 FALSE。




LineTo()函数画出的直线是一条半开半闭的线段,直线始点包括在直线上,而直线终点却不包括在直线上。

如果当前要画的直线并不与上一条直线的终点相接,那么应该用 MoveTo()函数来调整当前位置。其函数原型说明如下:

```
BOOL CDC:: MoveTo(int X,int Y);
```

更新后当前位置的坐标由 X、Y 参数给出。

下面是 Chap4 中画直线部分的程序示例。其中创建并选择了不同的画笔来绘制直线。

```
//绘制一组彩色线
//设置颜色表
struct      tagColor
{
      int r,g,b;
} color[7]=
{
      {255,0,0},
      {0,255,0},
      {0,0,255},

      {255,255,0},
      {255,0,255},
      {0,255,255},
```

```
        {0,0,0},
};

//定义两个指向画笔的指针
CPen *    pNewPen;
CPen *    pOldPen;
for(i = 0;i < 7;i ++)
{
    pNewPen = new CPen; // 构造画笔,然后再初始化。
    if( pNewPen - > CreatePen( PS_SOLID,2,RGB(color[i].r,
                               color[i].g,color[i].b) ) )
      {
            // 把新创建的画笔选入设备描述表
            // 同时存储设备描述表中的原有画笔
            pOldPen = pDC - > SelectObject( pNewPen );

            // 用新画笔绘图
            pDC - > MoveTo(20,60 + i * 10);
            pDC - > LineTo(160,60 - i * 10);

            // 恢复设备描述表中的原有画笔
            pDC - > SelectObject( pOldPen );
      }
      else
      {
            //出错时,给出错误提示。
            AfxMessageBox("CreatePen Error!");
    }

      //删除新创建的画笔,以便再另外分配。
      delete pNewPen;
}
```

### 4.3.3　画折线和曲线

除了 LineTo()函数可用来画线之外,MFC 的 CDC 还提供了一系列用于画各种折线和曲线的函数,这里分别介绍如下。

#### 4.3.3.1　画折线

Polyline()函数用于画一条折线,而 PolyPolyline()函数则用来画多条折线,它们的原型说明分别如下:

```
BOOL CDC::Polyline(CONST POINT * lppt,int cPoints);
```

其中 lppt 参数是指向折线顶点数组的指针，而 cPoints 参数则指定折线顶点数组中的顶点数。

```
BOOL CDC:: PolyPolyline(CONST POINT * lppt,
                        CONST DWORD * lpdwPolyPoints,DWORD cCount);
```

其中 lppt 参数是指向保存顶点的数组指针，而各条折线的顶点数则保存在 lpdwPolyPoints 参数所指向的数组中，最后的 cCount 参数指定折线的数目。

这里举两个简单的例子来说明这两个函数的用法。

例如，绘制如图 4.1 中左图所示的一条折线，可调用 Polyline()函数实现，方法如下：

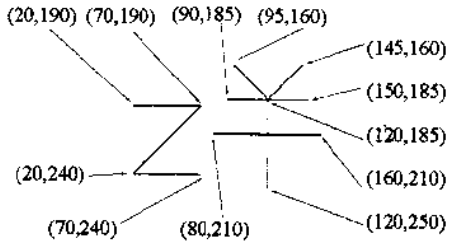



图 4.1 折线和多折线

首先定义折线的顶点数组：

```
POINT polyline[4] = {{20,190},{70,190},{20,240},{70,240}};
```

折线顶点为 4，所以把 Polyline()函数中的 cCount 参数设为 4，即用以下语句：

```
pDC - > PolyLine(polyline,4);
```

又如，用 PolyPolyline()函数画如图 4.1 中右图所示的多条折线，首先注意到其中有九个顶点，可分为四条折线，那么写下如下语句，就可画出这几条。

```
POINT polyPolyline[9] = {{95,160},{120,185},{120,250},{145,160},{120,185},
                         {90,185},{150,185},{80,210},{160,210}};
DWORD dwPolyPoints[4] = {3,2,2,2};
...
pDC - > PolyPolyline(polyPolylinePt,dwPolyPoins,4);
```

值得注意的是，因为一条折线至少要有两个顶点，因此 dwPolyPoints 数组中的数不应该小于 2。

与 Polyline()函数相应地还有一个 PolylineTo()函数，其参数是一致的。但是，PolylineTo()函数画折线总是把当前位置作为始点，并且在画完折线之后，会把折线终点所在位置设成新的当前位置。




调用 Polyline()函数和 PolyPolyline()函数不仅会采用当前位置，也会改变当前位置。

下面是示例程序 Chap4 中用于绘制折线的程序代码段：

```
//用系统的缺省笔绘制折线和曲线
POINT polyline[4] = {{20,190},{70,190},{20,240},{70,240}};
POINT polyPolyline[9] = {{95,160},{120,185},{120,250},{145,160},{120,185},
                         {90,185},{150,185},{80,210},{160,210}};
DWORD dwPolyPoints[4] = {3,2,2,2};

pDC->Polyline(polyline,4);
pDC->PolyPolyline(polyPolyline,dwPolyPoints,4);
```

#### 4.3.3.2　画弧线

通过调用 Arc()函数，应用程序就可以画一条椭圆弧线或者整个椭圆，而椭圆限定在一个矩形内，这个矩形称为外接矩形，它是看不见的。Arc()函数的原型说明如下：

```
BOOL CDC::Arc(int nLeftRect,int nTopRect,
              int nRightRect,int nBottomRect,
              int nXStartArc,int nYStartArc,
              int nXEndArc,int nYEndArc);
```

这里，(nLeftRect,nTopRect)是外接矩形的左上角坐标值，(nRightRect,nBottomRect)是外接矩形的右下角坐标。而椭圆中心与点(nXStartArc,nYStartArc)所构成的射线和椭圆的交点就成为椭圆弧线的起始点，椭圆中心与点(nXEndArc,nYEndArc)所构成的射线和椭圆的交点就成为弧线的终点。椭圆上弧线始点到终点的部分就形成弧线，这样通过这八个参数，就确定了一条椭圆弧，如图 4.2 所示。

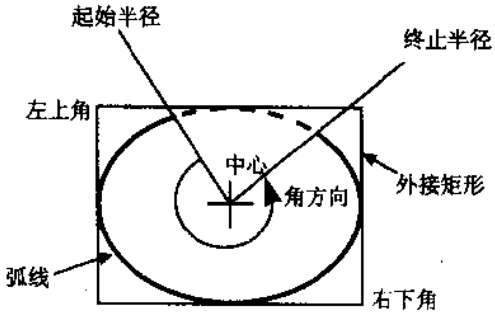



图 4.2　弧线

要唯一地确定一条椭圆弧线，除了以上八个参数之外，还有一个重要参数，那就是弧线始点到终点所取的方向。在 Windows98 中，这个方向总是逆时针的；在 Windows NT 中，默认绘制方向是逆时针方向，但可以通过调用 SetArcDirection()函数，将绘制方向改设为顺时针方向。

另一个画弧函数是 ArcTo()，它与 Arc()函数的关系就像 Polyline()函数与 PolylineTo()函数的关系一样，完全可以类推，这里就不赘述了。

在示例程序 Chap4 中用于绘制圆、圆弧和椭圆的代码段如下：

```
//绘制圆,圆弧和椭圆
for(i = 0;i < 8;i ++ )
{
    pDC - > Arc(260 - 5 * i,70 - 5 * i,260 + 5 * i,70 + 5 * i,260 + 5 * i,70,260 +
            5 * i,70);
}
for(i = 4;i < 8;i ++ )
{
      pDC - > Arc(260 - 10 * i,70 - 10 * i,260 + 10 * i,70 + 10 * i,
            (int)260 + 10 * i * cos(60 * 3.1415926/180),
            (int)70 + 10 * i * sin(60 * 3.1415926/180),
            (int)260 + 10 * i * cos(60 * 3.1415926/180),
            (int)70 - 10 * i * sin(60 * 3.1415926/180));

      pDC - > Arc(260 - 10 * i,70 - 10 * i,260 + 10 * i,70 + 10 * i,
            (int)260 - 10 * i * cos(60 * 3.1415926/180),
            (int)70 - 10 * i * sin(60 * 3.1415926/180),
            (int)260 - 10 * i * cos(60 * 3.1415926/180),
            (int)70 + 10 * i * sin(60 * 3.1415926/180));
}

CBrush *          pNewBrush;
CBrush *          pOldBrush;
for(i = 8;i > 1;i - - )
{
      pNewBrush = new CBrush; // 构造画刷,然后初始化
    if( pNewBrush - > CreateSolidBrush( RGB(color[8 - i].r,color[8 - i].g,
                                        color[8 - i].b) ) )
      {
            // 在设备描述表中选择新创建的画刷
            // 同时存储设备描述表中的原有画刷
            pOldBrush = pDC - > SelectObject( pNewBrush );

            // 用新画刷绘图
            pDC - > Ellipse(260 - 10 * i,200 - 5 * i,260 + 10 * i,200 + 5 * i);

            // 恢复设备描述表中的原有画刷
            pDC - > SelectObject( pOldBrush );
      }
      //删除新画刷,以便再分配另一个
      delete pNewBrush;
}
```

4.3.3.3　画贝齐尔曲线(Bezier Spline)

贝齐尔曲线是最常见的非规则曲线之一,它的形状便于控制,因此绘图时往往采用这种曲线。贝齐尔曲线属于三次曲线,只需要给定四点,就唯一确定了一条贝齐尔曲线,如图4.3所示。确定贝齐尔曲线的四个点是曲线的两端点和两控制点:P1和P4点指定了曲线的始点和终点,而P2、P3两控制点分别给出了曲线在两端点处的一阶导数(斜率)和二阶导数(曲率),从而确定了曲线的走向和形状。

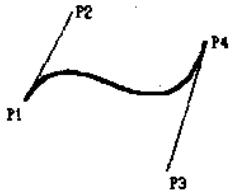



图4.3　贝齐尔曲线

调用PolyBezier()函数可以画出一条或多条贝齐尔曲线,其函数原型如下:

```
BOOL CDC:: PolyBezier(CONST POINT * lppt,DWORD cPoints)
```

其中lppt参数是曲线端点和控制点所组成的数组,cPoints参数表示lppt数组中的点数,lppt数组中的点应按照图4.3中所示的P1、P2、P3、P4的顺序排列。如果lppt数组用于画多条贝齐尔曲线,那么除了第一条曲线要用到四个点之外,后面的曲线只需用三个点。因为后面的曲线总是把前一条曲线的终点作为自己的始点。

在PolyBezier()函数中没有使用当前位置。如果要利用当前位置来画直线,就必须使用PolyBezierTo()函数。

在示例程序Chap4中,用于绘制贝齐尔曲线的程序代码段如下:

```
//绘制贝齐尔曲线(Bezier)
POINT polyBezier[4]={{20,310},{70,250},{110,290},{160,330}};
pDC->PolyBezier(polyBezier,4);
```

### 4.3.4　画封闭曲线

利用前面的画直线、折线、画弧的函数也可以画出椭圆形、矩形之类的封闭曲线,但Windows中还提供了一组画封闭曲线的函数,这使得能绘制矩形、多边形、椭圆形等封闭曲线更加简便。此外,这组画封闭曲线的函数不但可以利用画笔来画出轮廓线,同时还可利用画刷来填充这些封闭曲线所围起来的区域。下面就分别介绍这一组函数。

4.3.4.1　矩形

Rectangle()函数用来画矩形,其函数原型如下:

```
BOOL CDC:: Rectangle(int nLeftRect,int nTopRect,
                     int nRightRect,int nBottomRect);
```

其中参数 nLeftRect 和 nTopRect 给出了矩形左上角的坐标，而 nRightRect 和 nBottonRect 则给出矩形的右下角坐标。

4.3.4.2 椭圆形

Ellipse()函数的作用是画椭圆形，一个椭圆可由椭圆的中心、焦点、长轴、短轴等多组参数来确定。而在 Ellipse()函数中，椭圆是由其外接矩形来确定的。外接矩形的中心与椭圆中心重合，矩形的长与宽和椭圆的长短轴相等。Ellipse()函数的原型是：

```
BOOL CDC:: Ellipse(int nLeftRect,int nTopRect,
                   int nRightRect,int BottomRect);
```

其中的四个参数用来表示椭圆外接矩形的左上角和右下角坐标，如图 4.4 所示。

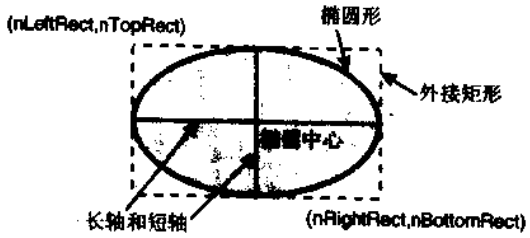



图 4.4 椭圆形

4.3.4.3 圆角矩形

圆角矩形把矩形和椭圆有机结合在一起，即把矩形的四个角椭圆化；如果把圆角矩形的四个圆角移凑到一起，就会拼成一个完整的椭圆。要画圆角矩形，就得使用 RoundRect()函数。RoundRect()函数的原型是：

```
BOOL CDC:: RoundRect(int nLeftRect,int nTopRect,
                     int nRightRect,int nBottomRect,int nWidth ,int nHeight);
```

其中参数 nLeftRect 和 nTopRect 表示矩形左上角坐标，nRightRect 和 nBottomRect 表示矩形右下角坐标，nWidth 是用来画圆角的椭圆的宽度，nHeight 则表示这个椭圆的高度，如图 4.5 所示。

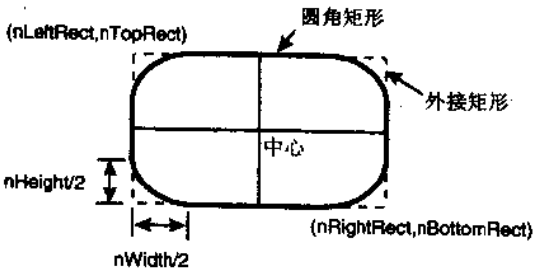



图 4.5 圆角矩形

4.3.4.4　弦形

一条椭圆弧及其对应的椭圆弦所围成的图形可称为弦形(如图 4.6 所示)。

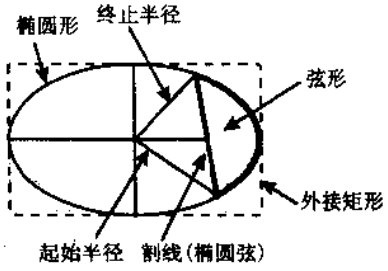



图 4.6　弦形

函数 Chord()就是用来画弦形,其参数的形式与含义和画弧的函数 Arc()的参数是完全一样的。

4.3.4.5　扇形

扇形是椭圆弧及其始终点对应的两条半径所围成的图形,其形状如图 4.7 所示。

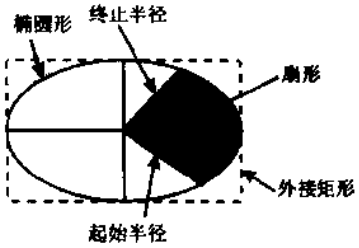



图 4.7　扇形

画扇形函数 Pie()的参数和 Arc()函数及 Chord()函数是一样的。

4.3.4.6　多边形

前面已经介绍过折线的画法,而多边形可以看作是由首尾相接的封闭折线所围成的图形,如图 4.8 所示。画多边形的函数 Polygon()原型如下:

```
BOOL CDC:: Polygon(CONST POINT * lpPoints,int nCount);
```

可以看出,Polygon()函数的参数形式与 Polyline()函数是相同的。但也略有区别。例如,要画一个三角形,使用 Polyline()函数,顶点数组中就得给出四个顶点(尽管始点和终点重复出现),而用 Polygon()函数就只须给出三个顶点。

4.3.4.7　多个多边形

与 PolyPolyline()可以画多条折线一样,使用 PolyPolygon()函数,一次可画出多个多边形,这两个函数的参数形式和含义也一样的。

在 Windows 所提供的应用程序编程接口中,用来画封闭图形的函数就是以上七个。



图4.8 多边形

在示例程序 Chap4 中,用于绘制封闭多边形的程序代码段如下:

```
//绘制矩形和圆角矩形
pDC->Rectangle(190,270,250,310);
pDC->RoundRect(260,270,330,310,30,20);

//绘制弦形和扇形
pDC->Chord(360-80,70-80,360+80,70+80,
      (int)360+80*cos(60*3.1415926/180),
      (int)70+80*sin(60*3.1415926/180),
      (int)360-80*cos(30*3.1415926/180),
      (int)70-80*sin(30*3.1415926/180));
pDC->Pie(480-80,70-80,480+80,70+80,
      (int)480+80*cos(60*3.1415926/180),
      (int)70+80*sin(60*3.1415926/180),
      (int)480+80*cos(30*3.1415926/180),
      (int)70-80*sin(30*3.1415926/180));

//绘制多边形
POINT polygon[3]={{400,160},{430,220},{350,210}};
POINT polyPolygon[6]={{450,120},{540,320},{480,280},
                      {400,300},{540,320},{480,280}};
int   polygonPoints[2]={3,3};
pDC->Polygon(polygon,3);
pDC->PolyPolygon(polyPolygon,polygonPoints,2);
}
```

## 4.4 GDI的示例程序

接下来,我们将动手编写 Chap4 示例程序来进一步了解 GDI。

### 试一试

1.应用 MFC AppWizard(exe) 创建一个新项目 Chap4,在第1步中选择 Single document。

创建的类：

```
Application: CChap4App in Chap4.h and Chap4.cpp
Frame: CMainFrame in MainFrm.h and MainFrm.cpp
Document: CChap4Doc in Chap4Doc.h and Chap4Doc.cpp
View: CChap4View in Chap4View.h and Chap4View.cpp
```

特征：

```
+ Initial toolbar in main frame
+ Initial status bar in main frame
+ Printing and Print Preview support in view
+ 3D Controls
+ Uses shared DLL implementation (MFC42.DLL)
```

2．在 CChap4View.cpp 的顶部添加：

```
#include "stdafx.h"
#include "Chap4.h"

#include "Chap4Doc.h"
#include "Chap4View.h"

//Add for cos(),sin()
#include <math.h>
```

3．编写 CChap4View::OnDraw 函数。

```
void CChap4View::OnDraw(CDC * pDC)
{
  int i;
  //由于在 OnDraw()函数中 pDC 是传入参数，
  //所以不用 CDC * pDC = GetDC()获取设备描述表指针。

  //绘制一组彩色点
  pDC->TextOut(20,20,"Point");
  pDC->SetPixel(100,20,RGB(255,0,0));
  pDC->SetPixel(110,20,RGB(0,255,0));
  pDC->SetPixel(120,20,RGB(0,0,255));
  pDC->SetPixel(130,20,RGB(255,255,0));
  pDC->SetPixel(140,20,RGB(255,0,255));
  pDC->SetPixel(150,20,RGB(0,255,255));
  pDC->SetPixel(160,20,RGB(0,0,0));
```

```
//绘制一组彩色线
//设置颜色表
struct      tagColor
{
      int r,g,b;
} color[7]=
{
      {255,0,0},
      {0,255,0},
      {0,0,255},

      {255,255,0},
      {255,0,255},
      {0,255,255},

      {0,0,0},
};

//定义两个指向画笔的指针
CPen *    pNewPen;
CPen *    pOldPen;
for(i=0;i<7;i++)
{
    pNewPen=new CPen; // 构造画笔,然后再初始化。
    if( pNewPen->CreatePen( PS_SOLID,2,RGB(color[i].r,
                              color[i].g,color[i].b) ) )
      {
          // 把新创建的画笔选入设备描述表
          // 同时存储设备描述表中的原有画笔
          pOldPen = pDC->SelectObject( pNewPen );

          // 用新画笔绘图
          pDC->MoveTo(20,60+i*10);
          pDC->LineTo(160,60+i*10);

          // 恢复设备描述表中的原有画笔
          pDC->SelectObject( pOldPen );
      }
      else
      {
          //出错时,给出错误提示。
          AfxMessageBox("CreatePen Error!");
      }
```

```
    //删除新创建的画笔,以便再另外分配。
    delete pNewPen;
}

//用系统的缺省笔绘制折线和曲线
POINT polyline[4] = {{20,190},{70,190},{20,240},{70,240}};
POINT polyPolyline[9] = {{95,160},{120,185},{120,250},{145,160},{120,185},
                         {90,185},{150,185},{80,210},{160,210}};
DWORD dwPolyPoints[4] = {3,2,2,2};

pDC->Polyline(polyline,4);
pDC->PolyPolyline(polyPolyline,dwPolyPoints,4);

//绘制贝齐尔曲线(Bezier)
POINT polyBezier[4] = {{20,310},{70,250},{110,290},{160,330}};
pDC->PolyBezier(polyBezier,4);


//绘制圆,圆弧和椭圆
for(i=0;i<8;i++)
{
    pDC->Arc(260-5*i,70-5*i,260+5*i,70+5*i,260+
        5*i,70,260+5*i,70);
}
for(i=4;i<8;i++)
{
    pDC->Arc(260-10*i,70-10*i,260+10*i,70+10*i,
        (int)260+10*i*cos(60*3.1415926/180),
        (int)70+10*i*sin(60*3.1415926/180),
        (int)260+10*i*cos(60*3.1415926/180),
        (int)70-10*i*sin(60*3.1415926/180));

    pDC->Arc(260-10*i,70-10*i,260+10*i,70+10*i,
        (int)260-10*i*cos(60*3.1415926/180),
        (int)70-10*i*sin(60*3.1415926/180),
        (int)260-10*i*cos(60*3.1415926/180),
        (int)70+10*i*sin(60*3.1415926/180));
}

CBrush*          pNewBrush;
CBrush*          pOldBrush;
for(i=8;i>1;i--)
```

```
    {
        pNewBrush = new CBrush; // 构造画刷,然后初始化
        if( pNewBrush -> CreateSolidBrush( RGB(color[8 - i].r,color[8 - i].g,
                                               color[8 - i].b) ) )
        {
            // 在设备描述表中选择新创建的画刷
            // 同时存储设备描述表中的原有画刷
            pOldBrush = pDC -> SelectObject( pNewBrush );

            // 用新画刷绘图
            pDC -> Ellipse(260 - 10 * i,200 - 5 * i,260 + 10 * i,200 + 5 * i);

            // 恢复设备描述表中的原有画刷
            pDC -> SelectObject( pOldBrush );
        }
        //删除新画刷,以便再分配另一个
        delete pNewBrush;
    }

    //绘制矩形和圆角矩形
    pDC -> Rectangle(190,270,250,310);
    pDC -> RoundRect(260,270,330,310,30,20);

    //绘制弦形和扇形
    pDC -> Chord(360 - 80,70 - 80,360 + 80,70 + 80,
          (int)360 + 80 * cos(60 * 3.1415926/180),
          (int)70 + 80 * sin(60 * 3.1415926/180),
          (int)360 + 80 * cos(30 * 3.1415926/180),
          (int)70 - 80 * sin(30 * 3.1415926/180));
    pDC -> Pie(480 - 80,70 - 80,480 + 80,70 + 80,
          (int)480 + 80 * cos(60 * 3.1415926/180),
          (int)70 + 80 * sin(60 * 3.1415926/180),
          (int)480 + 80 * cos(30 * 3.1415926/180),
          (int)70 - 80 * sin(30 * 3.1415926/180));

    //绘制多边形
    POINT polygon[3] = {{400,160},{430,220},{350,210}};
    POINT polyPolygon[6] = {{450,120},{540,320},{480,280},
                            {400,300},{540,320},{480,280}};
    int   polygonPoints[2] = {3,3};
    pDC -> Polygon(polygon,3);
    pDC -> PolyPolygon(polyPolygon,polygonPoints,2);
}
```

4. 创建和运行应用程序。

运行程序后,生成的程序界面如图 4.9 所示。

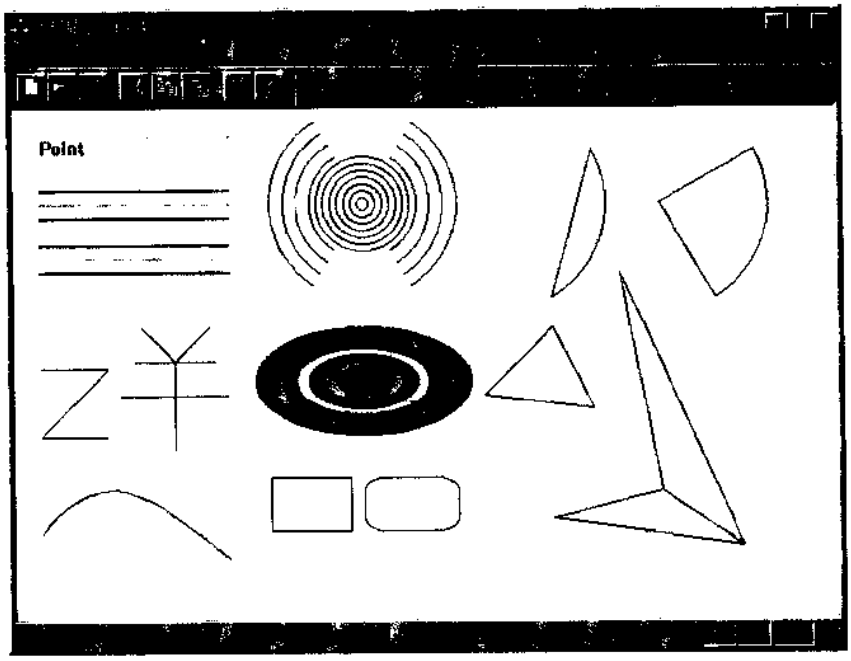



图 4.9　GDI 程序示例

## 4.5　本章小结

Windows 与 DOS 相比优势之一就是图形界面,本章介绍 Windows 作图系统——图形设备接口。通过本章学习,读者了解了设备描述表的基本概念,学会了创建和使用基本的 GDI 对象——画笔、画刷,掌握了绘制点、直线、折线、曲线以及封闭曲线的基本方法。

# 第五章　鼠标、键盘及计时器

**本章要点**

- 鼠标消息及其处理
- 改变鼠标的显示模式
- 键盘消息及其处理
- 设置计时器

当应用程序需要从用户处得到某些信息时，必然要用到输入设备。在 Windows 环境下，应用程序也是依赖于键盘、鼠标和计时器等输入设备进行输入的。由于 Windows 所特有的消息驱动体系结构，应用程序对输入的处理完全是一种基于消息的处理过程，而不涉及输入设备的具体硬件特性。因此，如何处理如键盘消息、鼠标消息和计时器消息是 Windows 程序实现交互的关键。

本章将从键盘、鼠标、计时器这三方面介绍一下 Windows 应用程序是如何处理各种输入消息的。

## 5.1　鼠　　标

鼠标是一种定位输入设备，通常有两个键(左键、右键)或三个键(左键、中键和右键)，鼠标的操作主要有以下三种基本方式：

- 单击：迅速按下并释放键。
- 双击：连续迅速地按下和放开键两次。
- 移动：移动鼠标的位置。

### 5.1.1　鼠标消息及其映射函数

对于一个典型的窗口，用于输出文档的区域我们称之为窗口的客户区，而非客户区是指标题栏、菜单栏、滚动条、边框等所占据的窗口区域。

Windows 为鼠标定义的消息中，有 10 个属于“客户区鼠标消息”，有 11 个是“非客户区鼠标消息”。对于非客户区消息，应用程序一般不作处理，而是交给 Windows 来处理，应用程序只需要处理客户区消息。

通常，Windows 只将键盘消息发送给具有输入焦点的窗口，但鼠标消息不受这种限制。只要鼠标通过某个窗口的客户区，该窗口的窗口函数便会收到 WM_MOUSEMOVE 消息。客户区中的鼠标消息如表 5.1 所示。在表 5.1 的第三栏中列出的是与这些消息对应的函数。

这些函数是 MFC 提供的，是定义在 CWnd 类中的虚函数，用户可以重载它们。

**表 5.1　客户区鼠标消息及其映射函数**

| 消息名称 | 鼠标操作 | 函数 |
|---|---|---|
| WM_MOUSEMOVE | 用户移动鼠标 | OnMouseMove(UINT nFlags, CPoint point); |
| WM_LBUTTONDOWN | 用户按下鼠标左键 | OnLButtonDown(UINT nFlags, CPoint point); |
| WM_LBUTTONUP | 用户放开鼠标左键 | OnLButtonUP(UINT nFlags, CPoint point); |
| WM_LBUTTONDBLCLK | 用户双击鼠标左键 | OnLButtonDblclk(UINT nFlags, CPoint point); |
| WM_RBUTTONDOWN | 用户按下鼠标右键 | OnRButtonDown(UINT nFlags, CPoint point); |
| WM_RBUTTONUP | 用户放开鼠标右键 | OnRButtonUp(UINT nFlags, CPoint point); |
| WM_RBUTTONDBLCLK | 用户双击鼠标右键 | OnRButtonDblClk(UINT nFlags, CPoint point); |
| WM_MBUTTONDOWN | 用户按下鼠标中键 | OnMButtonDown(UINT nFlags, CPoint point); |
| WM_MBUTTONUP | 用户放开鼠标中键 | OnMButtonUp(UINT nFlags, CPoint point); |
| WM_MBUTTONDBLCLK | 用户双击鼠标中键 | OnMButtonDblClk(UINT nFlags, CPoint point); |

以上 10 个消息处理函数，它们的 point 参数代表鼠标热点的坐标位置，point.x 为横坐标，point.y 为纵坐标。这里采用的是客户坐标系统。在这个坐标系统中，默认的坐标原点(0,0)位于客户区的左上角。

nFlags 参数中包含了鼠标键和键盘组合使用标志，用来描述鼠标键和键盘上 Shift 键和 Ctrl 键的组合状态。

- MK_LBUTTON：鼠标左键被按下，与 WM_LBUTTONDOWN 相同
- MK_RBUTTON：鼠标右键被按下，与 WM_RBUTTONDOWN 相同
- MK_MBUTTON：鼠标中键被按下，与 WM_MBUTTONDOWN 相同
- MK_SHIFT：键盘上的 Shift 键被按下
- MK_CONTROL：键盘上的 Ctrl 键被按下

若想知道某个键是否被按下，可用对应的位屏蔽值与 nFlags 参数作按位逻辑“与”运算，所得结果若为非零值，则表示键被按下，如：

```
if(nFlags & MK_LBUTTON) AfxMessageBox("LButton is pressed down!");
else AfxMessageBox("LButton up!");
```

这里需要说明的是，如何区分两次单击和一次双击，这取决于两次按下键之间的时间间隔，只有当时间间隔小于一定值时才被认为是一次双击，Windows 默认的时间间隔为 500 毫秒，应用程序可以调用::SetDoubleClickTime()函数来重新设定此值。




若要使窗口函数能接收到鼠标双击产生的消息，在注册窗口类时，必须指明该窗口具有 CS_DBLCLKS 风格，如：

```
wndclass.style = CS_HREDRAW|CS_VREDRAW|CS_DBLCLKS;
```

如果窗口风格不包含 CS_DBLCLKS,则即使进行了双击操作,该窗口也只能收到两条"WM_BUTTONDOWN"消息和两条"WM_BUTTONUP"消息。

### 5.1.2 鼠标消息的处理

处理鼠标消息的一般方法是重载以下消息映射函数来完成,如:

1.不检测 Shift 键和 Ctrl 键的状态:

```
OnLButtonDown(UINT nFlags,
              CPoint point)
{
};
```

2.检测 Shift 键和 Ctrl 键的状态:

```
OnLButtonDown(UINT nFlags,
              CPoint point)
{
    if((wParam&VK_CONTROL)&&(wParam&VK_SHIFT))
       /* Shift 键和 Ctrl 键都被按下 */
    break;
};
```

### 5.1.3 捕捉鼠标

通常情况下,只有当鼠标光标位于某一窗口的客户区或非客户区时,该窗口的窗口函数才能接收到鼠标消息。但是由于鼠标的随机性,难以保证光标始终不离开某一窗口。如果要使某个窗口能不间断地捕获鼠标消息,就必须对鼠标加以捕捉,从而使 Windows 发送的所有鼠标消息均定向到某一窗口,而不管鼠标光标位于何处。

调用 SetCapture()函数即可实现对鼠标的捕捉,例如:

```
SetCapture();
```

此语句的执行将导致 Windows 将向设置捕捉的窗口发送所有的鼠标消息。此时鼠标光标的坐标仍相对于客户区左上角,因此,此时鼠标位置值有可能是负值。

一旦某窗口捕捉了鼠标,系统的键盘功能暂时失效,其它窗口无法得到鼠标消息。因此,当该窗口不再需要捕获所有的鼠标消息时,应及时调用 ReleaseCapture()函数以释放鼠标,以便其它窗口可以正常地接收鼠标消息。

下面列出了与捕捉鼠标有关的两个函数。

1. SetCapture()函数

SetCapture()函数用于捕捉鼠标,其函数原型如下:

```
CWnd::SetCapture();
```

2. ReleaseCaputure()函数

Release SetCapture()函数用于释放鼠标，其原型说明如下：

```
CWnd::Release SetCapture( );
```

这两个函数都不带输入参数。

### 5.1.4　改变光标的类型

当鼠光标通过某个特定控件时，为了立即给用户提供反馈信息，需要改变光标的类型。例如，当鼠标移到客户区时需要显示成十字叉丝线；或者当鼠标在对话框中移到 Help 按钮上时，需要显示成问号型光标。

改变光标的形状，最简单的方法是响应 WM_SETCURSOR 消息。当光标在未被捕捉时从一个窗口移到另一个窗口，Windows 向窗口发送 WM_SETCURSOR 消息，MFC 应用程序通过 OnSetCursor()方法处理这个消息。

OnSetCursor()是一个虚函数，其原型如下：

```
protected afx_msg BOOL OnSetCursor(CWnd * pWind, UINT nHitTest, UINT message);
```

当鼠标输入未被捕获时，鼠标在 CWnd 对象内引起光标指针移动被调用。在处理前，无条件实施调用父窗口的 OnSetCursor。如果父窗口返回 TRUE 值，将停止进行的处理，调用父窗口赋予窗口设置子窗口的控制权。如果指针不是在客户区缺省操作，则将其设置成一向量，否则将其设置成已注册类指针。

如果 nHitTest 是 HERROR，且 message 是鼠标键按下消息，则将调用 Message_beep 成员函数。当 CWnd 进入菜单模式时，参数 message 是 0。

如果函数的返回值非零，则停止进一步处理，是零则继续处理。

鼠标的系统光标见表 5.2。

**表 5.2　系统光标**

| 光标形状 | 值 | 描述 |
|---|---|---|
|  | IDC_ARROW | 标准的箭头光标(选择) |
|  | IDC_CROSS | 十字型光标(通常用于精确定位) |
|  | IDC_SIZEALL | 四箭头光标(移动) |
|  | IDC_SIZEWE | 横向双箭头光标(水平调整) |
|  | IDC_SIZENS | 纵向双箭头光标(垂直调整) |
|  | IDC_SIZENWSE | 正斜向双箭头光标(正斜向调整) |
|  | IDC_SIZENESW | 反斜向双箭头光标(反斜向调整) |
|  | IDC_UPARROW | 垂直箭头光标 |

（续表）

| 光标形状 | 值 | 描述 |
| --- | --- | --- |
|  | IDC_IBEAM | I 型文本光标(选择文字) |
|  | IDC_WAIT | 漏斗型计时等待光标(等待) |
|  | IDC_APPSTARTING | 箭头加漏斗光标(后台运行) |
|  | IDC_NO | 圆加反斜线的光标(禁区，不可作为拖放目的) |

具体实现方法参见 5.1.6 节示例。

### 5.1.5 鼠标输入示例程序一

Chap5A 示例程序可以在窗口客户区内随手绘图。按下鼠标左键时落笔；移动鼠标便可绘图；放开左键便抬笔，停止绘图(见图 5.1)。

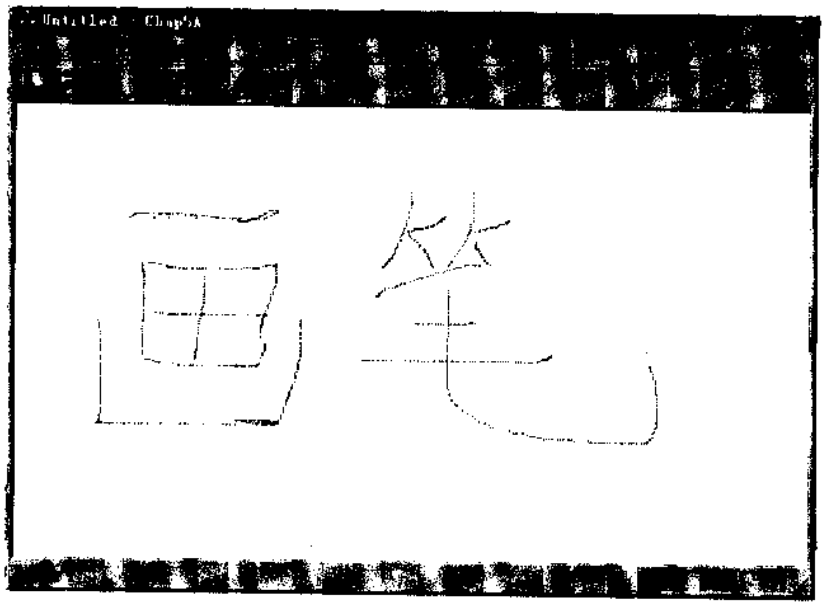


图 5.1 用鼠标随手绘图

**试一试**

1. 应用 MFC AppWizard(exe) 创建一个新项目 Chap5A，在第 1 步中选择 Single document。创建的类：

```
Application: CChap5AApp in Chap5A.h and Chap5A.cpp
Frame: CMainFrame in MainFrm.h and MainFrm.cpp
Document: CChap5ADoc in Chap5ADoc.h and Chap5ADoc.cpp
```

View: CChap5AView in Chap5AView.h and Chap5AView.cpp

特征：

+ Initial toolbar in main frame
+ Initial status bar in main frame
+ Printing and Print Preview support in view
+ 3D Controls
+ Uses shared DLL implementation (MFC42.DLL)

2.在 CChap5AView 中增加变量。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| BOOL | m_bDraw | protected |

3.利用 ClassWizard 在 CChap5AView 中增加消息处理函数。

| 消息 | 成员函数名 |
|---|---|
| WM_LBUTTONDOWN | OnLButtonDown |
| WM_LBUTTONUP | OnLButtonUp |
| WM_MOUSEMOVE | OnMouseMove |

4. 编写 CChap5AView::CChap5AView 构造函数。

```
CChap5AView::CChap5AView()
{
    m_bDraw = FALSE;
}
```

5.编写 CChap5AView::OnLButtonDown 函数。

```
void CChap5AView::OnLButtonDown(UINT nFlags, CPoint point)
{
    //捕捉鼠标
    SetCapture();
    //设置为绘图状态
    m_bDraw = TRUE;
}
```

6. 编写 CChap5AView::OnLButtonUp 函数。

```
void CChap5AView::OnLButtonUp(UINT nFlags, CPoint point)
{
    //捕捉鼠标
    ReleaseCapture();
    //设置为非绘图状态
    m_bDraw = FALSE;
```

```
}
```

7.编写 CChap5AView::OnLButtonMove 函数。

```
void CChap5AView::OnMouseMove(UINT nFlags, CPoint point)
{
    if(m_bDraw)
    {
        //获取设备描述表
        CDC * pDC = GetDC();
        pDC -> SetPixel(point,RGB(0,0,0));
    }
}
```

8.创建和运行应用程序。

按下鼠标左键,拖动鼠标进行徒手绘图。

### 5.1.6 鼠标输入示例程序二

Chap5B 示例程序将演示,如何使光标变形来提示用户,以便程序有更好的响应能力,提高交互性。

此例用到了用户自定义光标以及对话框,属于高级话题。读者可以在学完基本对话框程序的编制后再学习此节内容。此程序的执行情况如图 5.2、图 5.3 所示。

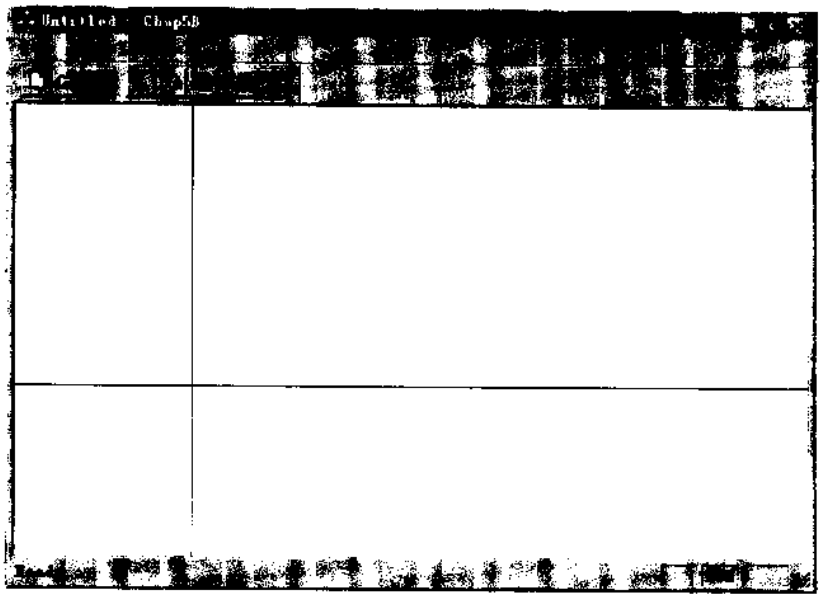



图 5.2 在客户区中鼠标变成十字叉丝线

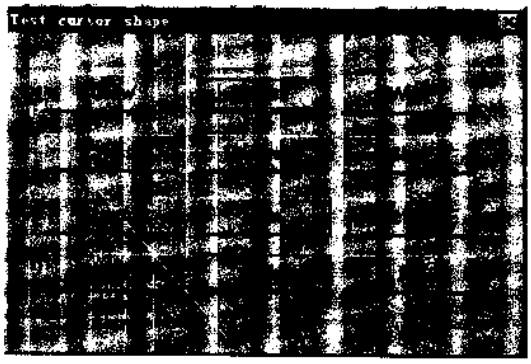



图 5.3　鼠标在对话框的不同按钮上变化形状

## 试一试

1. 应用 MFC AppWizard(exe) 创建一个新项目 Chap5B,在第 1 步中选择 Single document。创建的类:

```
Application: CChap5BApp in Chap5B.h and Chap5B.cpp
Frame: CMainFrame in MainFrm.h and MainFrm.cpp
Document: CChap5BDoc in Chap5BDoc.h and Chap5BDoc.cpp
View: CChap5BView in Chap5BView.h and Chap5BView.cpp
```

特征:

```
+ Initial toolbar in main frame
+ Initial status bar in main frame
+ Printing and Print Preview support in view
+ 3D Controls
+ Uses shared DLL implementation (MFC42.DLL)
```

2. 在 CChap5AView 中增加变量。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| CPoint | m_prePoint | protected |

3. 利用 ClassWizard 在 CChap5AView 中增加消息处理函数。

| 消息 | 成员函数名 |
|---|---|
| WM_SETCURSOR | OnSetCursor |
| WM_LBUTTONUP | OnLButtonUp |
| WM_MOUSEMOVE | OnMouseMove |

4. 编写 CChap5BView::CChap5BView 构造函数。

```
CChap5BView::CChap5BView()
{
    m_prePoint.x = m_prePoint.y = -1;
}
```

5.编写 CChap5BView::OnDraw 函数。

```
void CChap5BView::OnDraw(CDC* pDC)
{
    CRect rect;
    GetClientRect(rect);

    //绘制先前的光标线,
    //以避免在应用程序间切换时出现不应有的线
    pDC->MoveTo(rect.left,m_prePoint.y);
    pDC->LineTo(rect.right,m_prePoint.y);
    pDC->MoveTo(m_prePoint.x,rect.top);
    pDC->LineTo(m_prePoint.x,rect.bottom);
}
```

6.编写 CChap5BView::OnSetCursor 函数。

```
BOOL CChap5BView::OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message)
{
    SetCursor(AfxGetApp()->LoadCursor(IDC_CURSOR1));
    return TRUE;
}
```

7.编写 CChap5BView::OnMouseMove 函数。

```
void CChap5BView::OnMouseMove(UINT nFlags, CPoint point)
{
    CDC*  pDC=GetDC();

    //获取当前客户区的大小
    CRect rect;
    GetClientRect(rect);

    //设置 R2_NOTXORPEN 的绘图方式
    int nDrawMode=pDC->SetROP2(R2_NOTXORPEN);
    //覆盖先前的光标线
    pDC->MoveTo(rect.left,m_prePoint.y);
    pDC->LineTo(rect.right,m_prePoint.y);
    pDC->MoveTo(m_prePoint.x,rect.top);
    pDC->LineTo(m_prePoint.x,rect.bottom);
```

```
        //绘制新的光标线
        pDC->MoveTo(rect.left,point.y);
        pDC->LineTo(rect.right,point.y);
        pDC->MoveTo(point.x,rect.top);
        pDC->LineTo(point.x,rect.bottom);
        //设置先前的绘图方式
        pDC->SetROP2(nDrawMode);

        //记录当前点,以作为下次进入 MouseMove 的前一个点
        m_prePoint = point;
    }
```

8.编写 CChap5BView:: OnLButtonDown 函数。

```
void CChap5BView::OnLButtonDown(UINT nFlags, CPoint point)
{
    CCursorDlg  dlg;
    dlg.DoModal();
}
```

在该文件的顶端添加 # include "cursordlg.h"。

9.创建一个空白光标。

在 ResourceView 中用 Insert Resource Cursor 菜单给应用程序添加一个新光标。不对该光标图案进行任何编辑操作,直接将其存储。这就创建了一个空白光标。选中它后,在屏幕上什么也看见。这正有利于我们使用十字叉丝线光标。

10.创建一个对话框。

在 Resource View 中用 Insert Resource Dialog 菜单给应用程序添加一个对话框。使用资源编辑器给对话框增添 12 个按钮(见图 5.4)。控件的 ID 值和属性见下表。

| 控件 | ID 值 | 属性 |
|---|---|---|
| Arrow 按钮 | IDC_BTN_ARROW | 默认值 |
| IBeam 按钮 | IDC_BTN_IBEAM | 默认值 |
| Wait 按钮 | IDC_BTN_WAIT | 默认值 |
| Cross 按钮 | IDC_BTN_CROSS | 默认值 |
| UpArrow 按钮 | IDC_BTN_UPARROW | 默认值 |
| Size 按钮 | IDC_BTN_SIZE | 默认值 |
| SizeAll 按钮 | IDC_BTN_SIZEALL | 默认值 |
| Icon 按钮 | IDC_BTN_ICON | 默认值 |
| SizeNWSE 按钮 | IDC_BTN_SIZENWSE | 默认值 |
| SizeWE 按钮 | IDC_BTN_SIZEWE | 默认值 |
| Help 按钮 | IDC_BTN_HELP | 默认值 |
| No 按钮 | IDC_BTN_NO | 默认值 |

把对话框的 ID 值改为 IDD_TEST_SHAPE,Caption 改为 Test Cursor Shape。

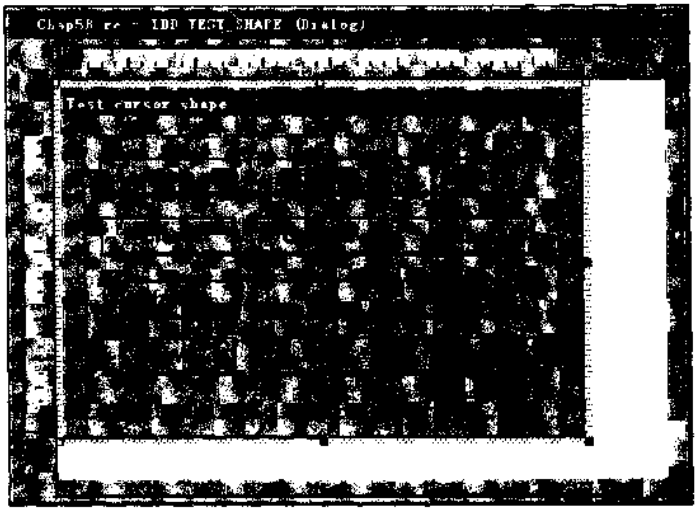



图 5.4 IDD_TEST_SHAPE 对话框

11. 创建 CCursorDlg 类。

利用 ClassWizard 根据下列信息建立一个新类。

类名:CCursorDlg

基类:CDialog

文件:CursorDlg

ID 号:IDD_TEST_SHAPE

12. 利用 ClassWizard 在 CCursorDlg 中增加消息处理函数。

| 消息 | 成员函数名 |
|---|---|
| WM_SETCURSOR | OnSetCursor |

13. 编写 CCursorDlg::OnSetCursor 函数。

```
BOOL CCursorDlg::OnSetCursor(CWnd * pWnd, UINT nHitTest, UINT message)
{
    const char * nID;

    switch(pWnd->GetDlgCtrlID())
    {
    case IDC_BTN_ARROW:     nID = IDC_ARROW; break;
    case IDC_BTN_IBEAM:     nID = IDC_IBEAM; break;
    case IDC_BTN_WAIT:      nID = IDC_WAIT; break;
    case IDC_BTN_CROSS:     nID = IDC_CROSS; break;
    case IDC_BTN_UPARROW:   nID = IDC_UPARROW; break;
```

```
        case IDC_BTN_SIZE:          nID = IDC_SIZE; break;
        case IDC_BTN_SIZEALL:       nID = IDC_SIZEALL; break;
        case IDC_BTN_ICON:          nID = IDC_ICON; break;
        case IDC_BTN_SIZENWSE:      nID = IDC_SIZENWSE; break;
        case IDC_BTN_SIZEWE:        nID = IDC_SIZEWE; break;
        case IDC_BTN_HELP:          nID = IDC_HELP; break;
        case IDC_BTN_NO:            nID = IDC_NO; break;
        default:
            return CDialog::OnSetCursor(pWnd, nHitTest, message);
        }

        SetCursor(AfxGetApp()->LoadStandardCursor(nID));
        return TRUE;
    }
```

14. 创建和运行应用程序

移动鼠标观察光标的变化。按下鼠标左键，弹出对话框，在对话框的各按钮上移动鼠标，观察光标的变化。

## 5.2 键　　盘

键盘作为输入设备，是 Windows 应用程序的一个重要的输入手段。当用户按下或释放一个键时，键盘驱动程序 KEYBOARD.DRV 中的键盘中断处理程序对所击键进行编码并调用 Windows 的用户模块 USER.EXE 中的有关程序来生成键盘消息，并最终发送到应用程序的消息队列中等待处理。而处理这些消息则是由应用程序的窗口过程来具体完成的。

键盘上每一个有意义键都对应着一个唯一的标识值，称为扫描码。当用户在按下或释放一个键时，都会产生扫描码。虽然扫描码可以用来作为键的标识，但它是依赖于具体设备的。因此，在应用程序中，使用的往往是与具体设备无关的虚拟码。虚拟码是由 Windows 系统定义的与设备无关的键标识值。

设备驱动程序截取键的扫描码，然后把它翻译为对应的虚拟码。这样，由于键盘输入，就产生了一条消息，它含有扫描码、虚拟码以及其它与击键有关的信息。设备驱动程序把这条信息放到系统消息队列中。Windows 从系统消息队列中取出这条消息，再把它发送到相应的线程消息队列中去，最后，由窗口过程从线程消息循环中取出键盘消息，进行必要的后续处理。

### 5.2.1 几个基本概念

在学习键盘消息及其处理之前，有几个基本概念需要了解。

·输入焦点

键盘必须由所有在 Windows 下运行的应用程序所共享。由于某些应用程序有好几个窗口，而键盘必须由同一应用程序的这些窗口共享。当在键盘上按下一个键时，只有一个窗口过程能接收到该键盘消息。接收到这个键盘消息的窗口称为有“输入焦点”的窗口，具有输入焦点的窗口可能是活动窗口，也可能是活动窗口的子窗口。当某一窗口为活动窗口时，Windows 会加亮显示其标题条(如果有的话)。如果窗口有边框，则其边框也被加亮显示。如果活动窗口为一图标，则此时没有一个窗口拥有输入焦点，但 Windows 仍将发送键盘消息给图标，只不过此时的消息格式有所不同。窗口函数通过捕获 WM_SETFOCUS 和 WM_KILLFOCUS 消息以确定当前窗口是否具有输入焦点。WM_SETFOCUS 表明窗口正在接收输入焦点，而 WM_KILLFOCUS 表示窗口正失去输入焦点。

·系统键与非系统键

“系统键”经常由输入键与 Alt 键组合产生，这种击键操作一般由 Windows 系统内部直接处理，而应用程序一般不必处理。如果应用程序处理了这种击键消息，则在处理完后，还应调用 DefWindowProc()函数，以便不影响 Windows 对它们的处理。对于系统键的击键过程，Windows 产生了两条系统击键消息：WM_SYSKEYDOWN 消息和 WM_SYSKEYUP 消息。

“非系统键”是指不与 Alt 键组合时被按下的键，一般产生 WM_KEYDOWN 和 WM_KEYUP 消息，应用程序窗口可以使用，也可以放弃这些消息，Windows 本身并不处理它们，参见表 5.3。

**表 5.3 击键消息**

| 键的种类 | 键按下时消息 | 键释放时的消息 |
|---|---|---|
| 系统键 | WM_SYSKEYDOWN | WM_SYSKEYUP |
| 非系统键 | WM_KEYDOWN | WM_KEYUP |

·虚拟键代码

虚拟键代码是一种与设备无关的键盘编码，它的值存放在键盘消息的 wParam 参数中，用以标识哪个键被按下或释放。最常用的虚拟键代码已被定义在 Windows.h 中，例如：VK_LEFT 表示“←”键，VK_F1 表示功能键 F1，VK_SHIFT 表示 Shift 键等。表 5.4 按照数值顺序列出所有的虚拟键代码。

**表 5.4 虚拟键代码**

| 符号常量名称 | 值(十六进制) | 等价的键盘键或鼠标键 |
|---|---|---|
| VK_LBUTTON | 01 | 鼠标左键 |
| VK_RBUTTON | 02 | 鼠标右键 |
| VK_CANCEL | 03 | 控制中断键 |
| VK_MBUTTON | 04 | 鼠标中键(对于三键鼠标) |
| - - - | 05~07 | 未定义 |
| VK_BACK | 08 | 退格键(BACKSPACE) |

（续表）

| 符号常量名称 | 值(十六进制) | 等价的键盘键或鼠标键 |
|---|---|---|
| VK_TAB | 09 | 制表键(TAB) |
| - - - | 0A ~ 0B | 未定义 |
| VK_CLEAR | 0C | CLEAR 键 |
| VK_RETURN | 0D | 回车键(ENTER) |
| - - - | 0E ~ 0F | 未定义 |
| VK_SHIFT | 10 | SHIFT 键 |
| VK_CONTROL | 11 | CTRL 键 |
| VK_MENU | 12 | ALT 键 |
| VK_PAUSE | 13 | PAUSE 键 |
| VK_CAPITAL | 14 | CAPSLOCK 键 |
| - - - | 15 ~ 19 | 为汉字系统保留 |
| - - - | 1A | 未定义 |
| VK_ESCAPE | 1B | ESC 键 |
| - - - | 1C ~ 1F | 为汉字系统保留 |
| VK_SPACE | 20 | SPACEBAR 空格键 |
| VK_PRIOR | 21 | PAGE UP 键 |
| VK_NEXT | 22 | PAGE DOWN 键 |
| VK_END | 23 | END 键 |
| VK_HOME | 24 | HOME 键 |
| VK_LEFT | 25 | LEFT ARROW 键(←) |
| VK_UP | 26 | UP ARROW 键(↑) |
| VK_RIGHT | 27 | RIGHT ARROW 键(→) |
| VK_DOWN | 28 | DOWN ARROW 键(↓) |
| VK_SELECT | 29 | SELECT 键 |
| - - - | 2A | 原始设备制造商(OEM)规定 |
| VK_EXECUTE | 2B | EXECUTE 键 |
| VK_SNAPSHOT | 2C | PRINT SCREEN 键 |
| VK_INSERT | 2D | INS 键 |
| VK_DELETE | 2E | DEL 键 |
| VK_HELP | 2F | HELP 键 |
| VK_0 - VK_9 | 30 ~ 39 | 0 键 ~ 9 键 |
| - - - | 3A ~ 40 | 未定义 |
| VK_A - VK_Z | 41 ~ 5A | A 键 ~ Z 键 |
| - - - | 5B ~ 5F | 未定义 |

（续表）

| 符号常量名称 | 值(十六进制) | 等价的键盘键或鼠标键 |
|---|---|---|
| VK_NUMPAD0 – 9 | 60 ~ 69 | 数字小键盘 0 键 ~ 9 键 |
| VK_MULTIPLY | 6A | 乘号键 |
| VK_ADD | 6B | 加号键 |
| VK_SEPARATOR | 6C | 分隔符键 |
| VK_SUBTRACT | 6D | 减号键 |
| VK_DECIMAL | 6E | 小数点键 |
| VK_DIVIDE | 6F | 除号键 |
| VK_F1 – VK_F24 | 70H ~ 87H | F1 键 ~ F24 键 |
| – – – | 88 ~ 8F | 未定义 |
| VK_NUMLOCK | 90 | NUM LOCK 键 |
| VK_SCROLL | 91 | SCROLL LOCK 键 |
| – – – | 92 ~ B9 | 未定义 |
| – – – | BA ~ C0 | OEM 规定 |
| – – – | C1 ~ DA | 未定义 |
| – – – | DB ~ E4 | OEM 规定 |
| – – – | E5 | 未定义 |
| – – – | E6 | OEM 规定 |
| – – – | E7 ~ E8 | 未定义 |
| – – – | E9 ~ F5 | OEM 规定 |
| – – – | F6 ~ FE | 未定义 |

### 5.2.2 键盘消息及其映射函数

当用户按下一个键或组合键时，Windows 将 WM_KEYDOWN 或 WM_SYSKEYDOWN 消息放入具有输入焦点的应用程序窗口的消息队列中；当键被释放时将产生 WM_KEYUP 或 WM_SYSKEYUP 消息并放入相应的消息队列中，MFC 提供了响应这些键盘消息的虚函数，参见表 5.5。

**表 5.5 键盘消息及其映射函数**

| 消息名称 | 键盘操作 | 函数 |
|---|---|---|
| WM_KEYDOWN | 非系统键被按下 | OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) |
| WM_KEYUP | 非系统键被释放 | OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags) |
| WM_SYSKEYDOWN | 系统键被按下 | OnSysKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) |
| WM_SYSKEYUP | 系统键被释放 | OnSysKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags) |

在函数中 nChar 记录了键的 OEM 扫描码，nRepCnt 记录了击键重复次数，nFlags 记录了

前一次击键的状态、Shift 键的状态等信息标志，表 5.6 列出 nFlags 的具体位所对应的击键状态。

**表 5.6　击键消息标志 nFlags**

| 位 | 描述 |
|---|---|
| 0～7 | 给出 OEM 键盘扫描码 |
| 8 | 指出是否为增强键盘上的扩展键。如果是扩展键，该值为 1，否则为 0 |
| 9～10 | 保留未用 |
| 11～12 | Windows 内部使用 |
| 13 | 给出关联码，表明是否使用了 ALT 键。如果按下了 ALT 键，该值为 1，否则为 0 |
| 14 | 给出前一次击键状态 |
| 15 | 给出转换状态。为 1，表示按键；为 0，表示释放键 |

一般情况下，大多数应用程序只使用很少的键盘消息，WM_SYSKEYDOWN 和 WM_SYSKEYUP 消息一般由系统内部处理，应用程序可不必处理，而只处理 WM_KEYDOWN 消息或 WM_KEYUP 消息。例如：如果我们只关心“↑”、“↓”、“←”、“→”四个键是否被按下并作相应的处理，则可参考下面一段程序。

```
OnKeyDown(UINT nChar,UINT nRepCnt,UINT nFlags)
{
    switch(nChar){
    case VK_LEFT:/*左箭头*/
        …
        break;
    case VK_RIGHT:/*右箭头*/
        …
        break;
    case VK_UP:/*上箭头*/
        …
        break;
    case VK_DOWN;/*下箭头*/
        …
        break;
    default:break;
}
```

### 5.2.3　字符消息及其映射函数

当击键过程的组合能产生可见的字符时，Windows 在发送键盘消息的同时还发送字符消息，而有些键只产生键盘消息而不产生字符消息，如功能键、Shift 键、Insert 键、Delete 键等。

在 Windows 应用程序的内部封装着下列消息循环：

```
While(GetMessage(&msg,0,NULL,NULL))
```

```
{
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}
```

当GetMessage()函数获取的消息是WM_KEYDOWN或WM_SYSKEYDOWN时,若击键产生了可见字符,TranslateMessage()函数把键盘输入消息转换成相应的字符消息WM_CHAR和WM_SYSCHAR,并将这条字符消息放入应用程序的消息队列中,它将成为在键盘消息之后,GetMessage()从应用程序消息队列中获取的下一条消息。

在Windows中共有四条字符消息,这些消息及其映射函数见表5.7。

**表5.7 字符消息及映射函数**

| 消息名称 | 映射函数 |
|---|---|
| WM_CHAR | OnChar(UINT nChar, UINT nRepCnt, UINT nFlags) |
| WM_DEADCHAR | OnDeadChar(UINT nChar, UINT nRepCnt, UINT nFlags) |
| WM_SYSCHAR | OnSysChar(UINT nChar, UINT nRepCnt, UINT nFlags) |
| WM_SYSDEADCHAR | OnSysDeadChar(UINT nChar, UINT nRepCnt, UINT nFlags) |




除了OnSysChar()的nFlags参数外,别的参数与键盘消息映射函数的参数完全相同。

WM_DEADCHAR和WM_SYSDEADCHAR称为"死字符消息",死字符的含义是指本身不能显示但能修改其它字符显示的字符,应用程序一般不对它们进行处理。

WM_CHAR和WM_DEADCHAR是由WM_KEYDOWN消息引出的,而WM_SYSCHAR和WM_SYSDEADCHAR是由WM_SYSKEYDOWN消息引出的。

由于用户击键方式和击键次序的不同,窗口函数所收到的消息的数目和次序也将不同,表5.8列出了一些击键与消息的对应关系。

**表5.8 击键与消息的关系**

| 击键方法 | 消息次序 | nChar参数 |
|---|---|---|
| 按下Z键然后释放 | WM_KEYDOWN | 虚拟键Z |
| | WM_CHAR | Z的ASCII码 |
| | WM_KEYUP | 虚拟键Z |
| 先按Shift键,再按Z键,然后释放Z键,最后释放Shift键 | WM_KEYDOWN | 虚拟键VK_SHIFT |
| | WM_KEYDOWN | 虚拟键Z |
| | | Z的ASCII码 |
| | WM_CHAR | 虚拟键Z |
| | WM_KEYUP | 虚拟键VK_SHIFT |

（续表）

| 击键方法 | 消息次序 | nChar 参数 |
| --- | --- | --- |
| 按下 Z 键持续一段时间后释放 | WM_KEYDOWN | 虚拟键 Z |
| | WM_CHAR | Z 的 ASCII 码 |
| | WM_KEYDOWN | 虚拟键 Z |
| | WM_CHAR | Z 的 ASCII 码 |
| | … | … |
| | WM_KEYDOWN | 虚拟键 Z |
| | WM_CHAR | Z 的 ASCII 码 |
| | WM_KEYDOWN | 虚拟键 Z |
| | WM_CHAR | Z 的 ASCII 码 |
| | WM_KEYUP | 虚拟键 Z |

### 5.2.4　键盘输入示例程序

Chap5C 示例程序演示了如何处理键盘消息。该程序允许用户在鼠标指定位置处输入字符，并允许用上下左右箭头键移动光标，指定输入位置。

**试一试**

1. 应用 MFC AppWizard(exe) 创建一个新项目 Chap5C，在第 1 步中选择 Single document。创建的类：

```
Application: CChap5CApp in Chap5C.h and Chap5C.cpp
Frame: CMainFrame in MainFrm.h and MainFrm.cpp
Document: CChap5CDoc in Chap5CDoc.h and Chap5CDoc.cpp
View: CChap5CView in Chap5CView.h and Chap5CView.cpp
```

特征：

```
+ Initial toolbar in main frame
+ Initial status bar in main frame
+ Printing and Print Preview support in view
+ 3D Controls
+ Uses shared DLL implementation (MFC42.DLL)
```

2. 在 CChap5CView 中增加变量和函数。

| 变量类型 | 变量名称 | 访问权限 |
| --- | --- | --- |
| CPoint | m_Position | protected |

| 函数类型 | 函数名称 | 访问权限 |
| --- | --- | --- |
| void | MoveCursor(CPoint point) | protected |

3. 利用 ClassWizard 在 CChap5CView 中增加消息。

| 处理函数消息 | 成员函数 |
| --- | --- |
| WM_LBUTTONUP | OnLButtonUp |
| WM_CHAR | OnChar |
| WM_KEYDOWN | OnKeyDown |

4.编写 CChap5CView::CChap5CView 构造函数。

```
CChap5CView::CChap5CView()
{
    m_Position.x = 40;
    m_Position.y = 40;
}
```

5. 编写 CChap5CView::OnDraw 函数。

```
void CChap5CView::OnDraw(CDC * pDC)
{
    //设置 R2_NOTXORPEN 的绘图方式
    int nDrawMode = pDC->SetROP2(R2_NOTXORPEN);
    //绘制输入光标
    pDC->MoveTo(m_Position.x,m_Position.y);
    pDC->LineTo(m_Position.x,m_Position.y + 16);
    //恢复先前的绘图方式
    pDC->SetROP2(nDrawMode);
}
```

6. 编写 CChap5CView::OnLButtonUp 函数。

```
void CChap5CView::OnLButtonUp(UINT nFlags, CPoint point)
{
    //设置当前文本输入位置
    MoveCursor(point);
}
```

7. 编写 CChap5CView::OnChar 函数。

```
void CChap5CView::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    char  tbuf[20];
    sprintf(tbuf,"%c",nChar);

    CDC *   pDC = GetDC();
    pDC->TextOut(m_Position.x + 1,m_Position.y,tbuf);
    CSize size = pDC->GetTextExtent(tbuf); //获取输出字符串的大小范围
    CPoint      point = m_Position + CPoint(size.cx + 1,0);
    MoveCursor(point);
```

```
}
```

8. 编写 CChap5CView::OnKeyDown 函数。

```
void CChap5CView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    CPoint      point;

    switch(nChar)
    {
    case VK_LEFT://光标向左移
        point = m_Position + CPoint( - 20 * nRepCnt,0);
        MoveCursor(point);
        break;
    case VK_RIGHT://光标向右移
        point = m_Position + CPoint(20 * nRepCnt,0);
        MoveCursor(point);
        break;
    case VK_UP://光标向上移
        point = m_Position + CPoint(0, - 20 * nRepCnt);
        MoveCursor(point);
        break;
    case VK_DOWN://光标向下移
        point = m_Position + CPoint(0,20 * nRepCnt);
        MoveCursor(point);
        break;
    default:
        break;
    }
}
```

9. 编写 CChap5CView::MoveCursor 函数。

```
void CChap5CView::MoveCursor(CPoint point)
{
    CDC * pDC = GetDC();
    //设置 R2_NOTXORPEN 的绘图方式
    int nDrawMode = pDC - > SetROP2(R2_NOTXORPEN);
    //覆盖先前的光标线
    pDC - > MoveTo(m_Position.x,m_Position.y);
    pDC - > LineTo(m_Position.x,m_Position.y + 16);

    //绘制新的光标线
    m_Position = point;
    pDC - > MoveTo(m_Position.x,m_Position.y);
```

```
        pDC->LineTo(m_Position.x,m_Position.y+16);
        //恢复先前的绘图方式
        pDC->SetROP2(nDrawMode);
    }
```

10.创建和运行应用程序。

在程序运行后,用户可以试着输入一些字符,光标会跟随用户输入字符相应地往后移。按下上下左右箭头键,可以改变光标的位置。在客户区中单击鼠标左键,也可以把光标移到鼠标的当前位置。

## 5.3 计 时 器

在 Windows 系统中,除了鼠标和键盘外,还有一种常用的输入设备就是系统时钟。应用程序通过计时器来访问系统时钟。计时器是系统内部程序,应用程序可以随时申请系统的计时器并给申请到的计时器赋以初值(称为超时值)。每隔 1 毫秒计数值减 1,计数值减为零时(即发生超时事件)会触发一个中断,Windows 系统会发出一条消息 WM_TIMER 给应用程序,由应用程序作相应的处理。

应用程序通过调用 SetTimer()函数来完成对新的计时器的申请和设置,这个函数的原型说明如下:

```
CWnd::SetTimer(UINT nIDEvent,UINT nEllapse, Void(CALLBACK EXPORT * lpfnTimer)
```

nIDEvent 是该计时器的标识号(标识符),当应用程序需要多个计时器时可多次调用 SetTimer 函数,但每个计时器的标识号都是唯一的,各不相同。nEllapse 参数为 UINT 类型,此值以毫秒(1/1000 秒)为单位,它就是要设定的计时初值。lpfnTimer 参数是一个回调函数的指针,该函数由应用程序定义,用来处理计时器超时事件。

lpfnTimer 一般可以不用,参数值设为 NULL,这时当发生超时事件时,系统向应用程序消息队列发送 WM_TIMER 消息。对应窗口过程收到一条 WM_TIMER 消息时,其映射函数为 OnTimer(UINT nIDEvent)。其中 nIDEvent 为计时器标识号,应与 SetTimer 中指定的一致。

应用程序也可以定义一个处理计时器超时事件的回调函数,把它赋值给 lpfnTimer 参数。这时,WM_TIMER 消息不由窗口过程处理,而改为由回调函数处理。下面是这个回调函数的原型说明:

```
VOID CALLBACK TimerProc(
    HWND hwnd,      //消息不由接收计时器处理的窗口句柄
    UINT uMsg,      //WM_TIME 消息
    UINT idEvent,   //计时器标识号
    DWORD dwTime    //当前系统时间(采用世界坐标时间格式)
);
```

当不再需要使用计时器时,调用 KillTimer()函数停止传送 WM_TIMER 消息,其函数原

型如下：

```
CWnd::KillTimer(UINT nIDEvent);
```

当计时器用尽后，再调用 SetTimer()函数将返回 NULL，表明无计时器可用，正常调用时返回值为非零值，是具体的某个计时器的标识号。

如果用户在程序中想要改变某个计时器的计时初值，必须先放弃该计时器的使用权，然后再次调用 SetTimer()函数，重新获得计时器并设定新的初值，例如：

```
...
SetTimer(IDTimer,1000,lpTimerFunc);
...
KillTimer(IDTimer);
SetTimer(IDTimer,2000,lpTimerFunc);
...
```

需要注意的一点是，由于 WM_TIMER 消息被放到正常的消息队列中，与其它消息一起排队。因此，即使应用程序对某一计时器设定了 500 毫秒的定时间隔，也不能保证应用程序每隔 500 毫秒就能收到一条 WM_TIMER 消息。这是由于当时应用程序本身或别的应用程序中的某个处理过程持续的时间过长时，Windows 得不到控制权，这期间应用程序将接收不到 WM_TIMER 消息。这就是说，应用程序所得到的计时器超时值是不精确的，其精度受到系统时钟速率和消息队列频度的限制。

下面，我们以一个具体示例程序来说明计时器的使用方法。在这个示例程序中用到了两个计时器，一个计时器用来控制光标闪烁显示，另一个计时器则产生每秒递增一次的计数。

## 试一试

1.打开 Chap5C 项目。

2.在 CChap5CView 中增加成员变量。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| BOOL | m_bCursorOn | protected |
| int | m_nCount | protected |

3.利用 ClassWizard 在 CChap5CView 中增加消息处理函数，以及 Create 和 DestroyWindow 虚函数。

| 消息 | 成员函数名 |
|---|---|
| WM_TIMER | OnTimer |

4．改写 CChap5CView::CChap5CView 构造函数。

```
CChap5CView::CChap5CView()
```

```
{
    m_Position.x = 40;
    m_Position.y = 40;
    m_nCount = 0;
    m_bCursorOn = TRUE;
}
```

5. 改写 CChap5CView::MoveCursor 函数。

```
void CChap5CView::MoveCursor(CPoint point)
{
    CDC * pDC = GetDC();
    //设置 R2_NOTXORPEN 的绘图方式
    int nDrawMode = pDC->SetROP2(R2_NOTXORPEN);
    if(m_bCursorOn)
    {
        //覆盖先前的光标线
        pDC->MoveTo(m_Position.x,m_Position.y);
        pDC->LineTo(m_Position.x,m_Position.y + 16);
    }
    else
        m_bCursorOn = TRUE;

    //绘制新的光标线
    m_Position = point;
    pDC->MoveTo(m_Position.x,m_Position.y);
    pDC->LineTo(m_Position.x,m_Position.y + 16);
    //恢复先前的绘图方式
    pDC->SetROP2(nDrawMode);
}
```

6. 编写 CChap5CView::OnTimer 函数。

```
void CChap5CView::OnTimer(UINT nIDEvent)
{
    CDC * pDC = GetDC();
    switch(nIDEvent)
    {
    case 1:
        {
        m_bCursorOn = ! m_bCursorOn;
        //设置 R2_NOTXORPEN 的绘图方式
        int nDrawMode = pDC->SetROP2(R2_NOTXORPEN);
        //覆盖先前的光标线
        pDC->MoveTo(m_Position.x,m_Position.y);
```

```
            pDC->LineTo(m_Position.x,m_Position.y+16);
            pDC->SetROP2(nDrawMode);
            break;
            }
        case 2:
            char tbuf[20];
            m_nCount++;
            sprintf(tbuf,"%d  ",m_nCount);
            pDC->TextOut(20,20,tbuf);
            break;
        default:
            break;
        }
    }
```

7. 编写 CChap5CView::Create 函数。

```
BOOL CChap5CView::Create(LPCTSTR lpszClassName, LPCTSTR lpszWindowName,
                         DWORD dwStyle, const RECT& rect,
                         CWnd* pParentWnd, UINT nID,
                         CCreateContext* pContext)
{
    int ret = CWnd::Create(lpszClassName, lpszWindowName,
                           dwStyle, rect, pParentWnd,
                           nID, pContext);
```

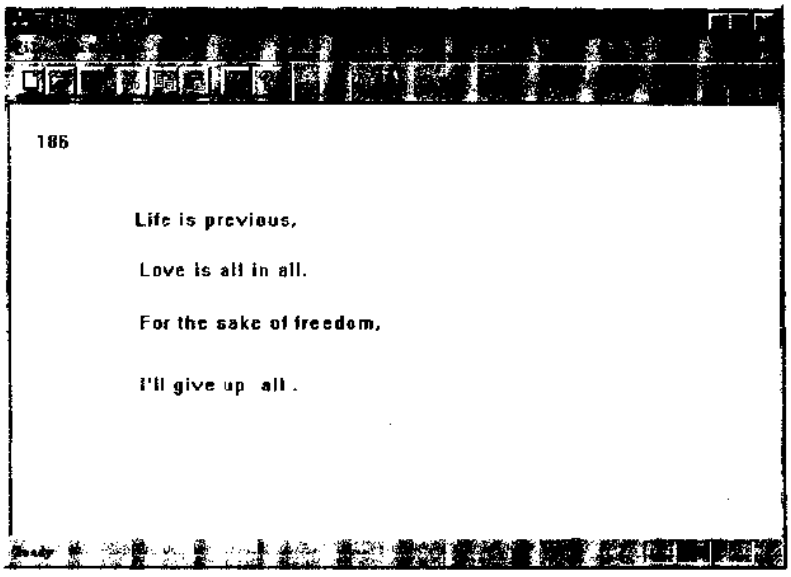



图 5.5　计时器、键盘示例程序

```
    ASSERT(SetTimer(1,500,NULL)! =0);
    ASSERT(SetTimer(2,1000,NULL)! =0);
    return ret;
}
```

8.编写 CChap5CView::DestroyWindow 函数。

```
BOOL CChap5CView::DestroyWindow()
{
    KillTimer(1);
    KillTimer(2);
    return CView::DestroyWindow();
}
```

9.创建和运行应用程序。

观察客户区左上角的计数器值的变化,注意光标是否在闪烁,如图 5.5 所示。

## 5.4 本章小结

应用程序最基本的单元是输入和输出。本章讲述了与鼠标输入、键盘输入以及计时器输入相关的知识。通过学习本章,读者了解了何时能够产生鼠标消息、键盘消息和计时器消息,知道了这些消息产生的先后顺序以及如何处理这些消息。通过本章实例,读者应当掌握了处理这些消息的方法。

# 第六章 菜 单

**本章要点**

- 菜单的组成以及命令实现机制
- 修改、编辑普通菜单
- 创建和调用上下文菜单
- 修改主菜单的显示状态
- 动态地增、删和修改菜单
- 在多个菜单间进行切换

菜单是 Windows 应用程序中最常见的元素。大量命令的调用都是通过选择菜单实现的。在使用 AppWizard 生成 SDI 或 MDI 应用程序框架时，AppWizard 将自动生成标准的应用框架菜单资源和菜单自理函数代码。为了满足实际需要，我们有时需要修改缺省菜单。本章将引导读者学习对菜单的编制，以便读者能得心应手地编写出满足设计需求的菜单。

## 6.1 概 述

本节先讲述组成菜单的基本元素——资源和 CMenu 类；接着，详细讲解菜单消息映射、消息传递顺序以及 UI 消息机制；最后，将介绍上下文菜单。通过学习本节，读者能够基本理解菜单。

### 6.1.1 菜单和资源

我们可以在 Visual Studio 中增加和删除菜单和菜单项，定义菜单标识符 ID，对菜单进行合理布局，设置菜单初始状态（正常、灰色、加检查标记）以及设置菜单提示等。

对于 MDI 应用程序，AppWizard 生成两个菜单资源：IDR_MAINFRAME 和 IDR_PRJNAMETYPE（PRJNAME 为应用程序的项目名称）。当没有打开 MDI 子窗口时显示 IDR_MAINFRAME 菜单，打开 MDI 子窗口之后显示 IDR_PRJNAMETYPE 菜单。在打开 MDI 子窗口以后，可以利用 Windows 菜单对 MDI 子窗口进行管理。对于 SDI 应用程序，AppWizard 只生成一个菜单资源：IDR_MAINFRAME。如果应用程序为数据库或 OLE 应用程序，那么 AppWizard 还生成其它相应的菜单资源。一般可以给每个菜单项赋予一个助记符。助记符就是菜单项文本中一个带下划线的字母。给菜单项增加助记符，可以增强菜单的标准键盘接口。创建菜单项助记符键的方法为：打开菜单对应的属性对话框，在对话框 Caption 编辑框中文本串的一个字符（助记符键）前加一个“&”字符（& 为转义字符）。例如：&New，应表示

为 &New,即字符 N 为 New 菜单项的助记符键。如果菜单项本来含有字符 &,则应输入 &&。每一个菜单项都有一个提示字符串,当选中一个菜单项时,相应的提示字符串就显示在状态条中。这些提示字符串保存在应用程序的字符串资源中,通过 ID 与相应的菜单项相联系。从 AppStudio 看,这些提示字符串是菜单项定义的一部分。在菜单属性对话框的 Prompt 编辑框中定义提示字符串,如图 6.1 所示。如果没有定义提示字符串,则在应用程序 Debug 版本执行时,将在 Debug 输出窗口中给出类似如下的警告信息:

```
Warning:no message line prompt for ID 0x8003
```

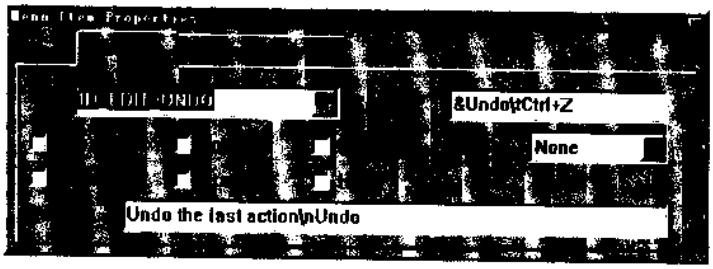



图 6.1 在 Menu Property 对话框的 Prompt 编辑框中增加提示字符串

除助记符外,菜单项有时还有一个对应的加速键表,作为菜单项的快捷键。

键盘加速键与助记符有两点不同:一是只有在菜单为活动菜单时助记符才有效,而加速键不是;二是助记符总是与一个菜单项相联,而键盘加速键除了与菜单项相联外,还与 CMenu 类相联。因为在主窗口创建时,菜单资源被隐含装载。只需利用 AppStudio 修改菜单资源,然后使用 ClassWizard 加入菜单处理函数。但是,如果需要创建上下文菜单,或者在运行时对菜单进行动态修改,就必须用到 CMenu 类。

可以通过下列三种方法获得指向 CMenu 对象的指针:

利用 CWnd::GetMenu()函数检取指向指定菜单的指针。

利用 CWnd::GetSystemMenu()函数检取指向系统菜单的指针。

利用 CMenu::LoadMenu()函数从应用程序的可执行文件装载菜单资源,并将它连接到 CMenu 对象上。

其中前两种方法用于检取指向现成菜单的指针,第三种方法用于获得指向新装载的菜单的指针,此时可以调用 CWnd::SetMenu()函数将该菜单设置为指定菜单。利用 CMenu 类的 m_hMenu 成员变量可获得该 CMenu 对象所连接的 Windows 菜单的 HMENU 句柄。

在获得菜单指针之后,就可以利用 CMenu 成员函数在运行时刻动态修改菜单(AppStudio 只能静态修改菜单资源)。CMenu 类的成员函数 AppendMenu()、InsertMenu()、ModifyMenu()和 RemoveMenu()分别用于添加、插入、修改和删除指定菜单项。标识菜单项的方法有两种:一是用 ID,二是用位置索引。

如果菜单由弹出式菜单构成,则通过成员函数 GetSubMenu()检取该菜单的一个弹出式菜单的 CMenu 对象。

成员函数 TrackPopupMenu()在指定位置显示一个浮动的弹出式菜单,并跟踪该菜单上

的选择。浮动的弹出式菜单可以出现在屏幕上的任何地方。

### 6.1.3 菜单消息映射和命令处理

不知道读者有没有注意到，当使用 ClassWizard 准备对一个菜单进行消息映射时，发现视、文档、框架和应用类都可以对该菜单 ID 进行映射。那么该菜单处理函数究竟应该在哪个类映射呢？在不同的类映射又有什么区别呢？处理的原则是：依据菜单的作用不同而进行不同映射。如果该菜单用于文档的显示编辑，则最好在视图类中映射。如果该菜单用于文档的打开和存储则最好在文档类中映射。对于通用菜单，可以在框架类中映射。About 对话框的显示菜单常在应用类中映射。

由于类的成员函数只能调用该类的其它成员函数和基类函数，因而在不同类中映射的结果是菜单处理函数能够使用的函数集合不同。例如在应用类映射的菜单处理函数不能使用 CWnd 类的 MessageBox()消息框，而只能调用 Windows SDK 消息框::MessageBox(0L,...)或调用全局消息框 AfxMessageBox()。

此外，通过 CWinThread 成员变量 m_pMainWnd 可获取应用程序主框架窗口的指针。在其它类映射的菜单处理函数只能通过全局函数 AfxGetApp 或全局变量 theApp 获取应用程序指针。

下面讨论命令是如何传递和寻找处理函数的。Windows 将消息发送给主框架窗口（所有消息通常都是发送给主框架窗口的），但如果菜单命令消息是送给其它对象的，此时命令传递顺序如表 6.1 所示。

**表 6.1 标准的命令传递顺序**

| 接收到命令的对象的类型 | 传递顺序 |
|---|---|
| MDI 框架窗口(CMDIFrameWnd) | 1.活动窗口 CMDIChildWnd<br>2.本框架窗口<br>3.应用程序(CWinApp 对象) |
| 子窗口(CFrameWnd, CMDIChildWnd) | 1.活动视<br>2.本子窗口<br>3.应用程序 |
| 视 | 1.本视<br>2.与该视相联的文档 |
| 文档 | 1.本文档<br>2.与该文档相联的文档模板 |
| 对话框 | 1.本对话框<br>2.拥有该对话框的窗口<br>3.应用程序 |

可以看出，对于 SDI 应用程序命令传递顺序为：视→文档→SDI 主框架窗口→应用程序；对于 MDI 应用程序传递顺序为：视图→文档→MDI 子框架窗口→MDI 主框架窗口应用程序。因而借助于命令传递系统，可以在程序中的任何位置映射菜单命令等消息，但最好能依据菜单功能合理映射处理函数。

使用 ClassWizard 映射菜单命令后，ClassWizard 自动向程序插入菜单命令消息映射入口、消息处理函数原型说明和空的处理函数体。例如：假设 Test 应用程序有一菜单项 Edit | Cut 在视类中映射，则向视类的实现文件中增加如下代码：

```
BEGIN_MESSAGE_MAP(CTestView,CView)    //消息映射入口
ON_COMMAND(ID_EDIT_CUT,OnEditCut)
END_MESSAGE_MAP()
void CTestView::OnEditCut()
{
    //命令自理函数
    //..程序代码
}
```

在视类头文件中增加如下代码：

```
afx_msg void OnEditCut();    //函数说明
```

### 6.1.4 更新命令用户接口(UI)消息

通常菜单项只有一个状态，但在许多情况下菜单项还可能有多个状态。例如：菜单项在某一时刻允许使用(正常)，而另一时刻不可使用(菜单变灰)；菜单可以加注检查标记，以表明被选中或未被选中。MFC 为自动更新用户接口提拱了更新命令用户接口消息。

更新命令 UI 消息只适用于弹出式菜单的菜单项，而对顶层菜单项则不适用，例如：不能利用更新命令 UI 消息来禁止 Edit 菜单项。在显示弹出式菜单时，框架按表 6.1 所示传递顺序发送弹出式菜单的所有菜单项的更新命令 UI 消息。如果菜单项有一个更新处理函数，则它被调用进行更新；如果没有，则框架检查命令处理函数是否存在。若不存在，则使菜单变灰。因此为了使一个菜单项有效，则必须有一个菜单处理函数或更新处理函数。


注意

如果将 CFrameWnd 类的成员变量 m_bAutoMenuEnable 设置为 FALSE，则使用户接口对象的默认无效状态变为有效。

与菜单自理函数一样，当使用 ClassWizard 加菜单更新函数(如 OnUpdateEditCut 函数)之后，ClassWizsrd 自动向应用程序添加代码。例如对 Edit | Cut 菜单项进行更新处理，在视类实现文件添加消息映射和空的函数体：

```
BEGIN_MESSAGE_MAP(CTestView,CView)    //消息映射入口
ON_UPDATE_COMMAND_UI(ID_EDIT,OnUpdateEditCut)
END_MESSAGE_MAP()
void CTestView:: OnUpdateEditCut(CCmdUI * pCmdUI)
{
    //更新命令处理函数
    //...程序代码
```

```
    pCmdUI->Enable(m_bHaveData)
}
```

向视类头文件中添加如下代码：

```
afx_msg void OnUpdateEditCut(CCmdUI * pCmdUI);    //函数说明
```

从上面看到，更新自理函数的参数为一个指向 CCmdUI 对象的指针，处理函数可以利用该指针对菜单项进行状态设置。因为控制类中也广泛用到更新命令 UI 消息。下面简单介绍 CCmdUI 类的功能及其成员函数。

CCmdUI 类只在 ON_UPDATE_COMMAND_UI 处理函数内使用，实现更新用户接口对象，如菜单、控件、控制条等。菜单项可以用控制条按钮或其它用户接口对象代替，但不能使用改变 ON_UPDATE_COMMAND_UI 处理函数内的代码。

表 6.2 小结了 CCmdUI 的成员函数对各种用户接口对象的作用。

**表 6.2 CCmdUI 成员函数小结**

| 用户界面项 | Enable | SetCheck | SetRadio | SetTest |
|---|---|---|---|---|
| 菜单项 | 允许(1)或禁止(0) | 用(x)，检取(1)或未检取(0) | 用点(.)，检取(1)或未检取(0) | 设置项文字 |
| 工具条按钮 | 允许或禁止 | 选择(1)未选择(0)或不确定(2) | 与 SetCheck 同 | (不可用) |
| 状态条框 | 使文本可见或不可见 | 设置弹出边框或正常边框 | 与 SetCheck 同 | 设置框文字 |
| 对话条中的正常按钮 | 允许或禁止 | 检取或检取框 | 与 SetCheck 同 | 设置按钮文字 |
| 对话条中的正常控件 | 允许或禁止 | (不可用) | (不可用) | 设置窗口文字 |

### 6.1.5 扩展命令和范围命令处理

菜单项除了 ON_COMMAND 和 ON_UPDATE_COMMAND_UI 消息映射宏之外，还有一种扩展类型的消息映射宏 ON_CONNAND_EX。该消息可以实现用同一个函数对多个相关命令消息进行处理，在处理函数内对命令 ID 的不同值进行不同处理。消息映射语法格式为：

```
ON_OCMMAND_EX(id1,memberFnx)
ON_COMMAND_EX(id2,memberFnx)
```

其中 id1、id2 为相关命令 ID，memberFnx 为命令处理函数。

MFC 还支持把一个范围内的消息映射成单个消息处理函数，如把一个范围内的命令 ID 映射成一个命令处理函数和一个命令更新处理函数。消息映射语法格式为：

```
ON_COMMAND_RANGE(id1,id2,memberFnx)
ON_UPDATE_COMMAND_UI_RANGE(id1,id2,UpdatememberFnx)
```

其中 id1 和 id2 为连续范围内命令 ID 的开始和结束值，memberFnx 为命令处理函数，UpdatememberFnx 为命令更新处理函数。

如果把相关命令 ID 值连续化，则该范围命令可取代扩展命令。这两种命令处理函数都

只有唯一一个 UINT 类型参数 nID。注意 ClassWizard 既不支持扩展命令消息，也不支持范围命令消息，程序员必须自己编写消息映射。因为这些消息映射不由 ClassWizard 管理，因而要放在//{{AFX_MSG_MAP 和//}}AFX_MSG_MAP 括号对之外。

### 6.1.6 上下文菜单

Windows 98 的用户接口的一个重要特征是鼠标右键功能。当用户单击鼠标右键时，系统弹出一个浮动菜单(称为上下文菜单)。

通过该菜单，用户可以方便快捷地进行一些操作。用户可以利用键盘加速键 Shift + F10 启动上下文菜单。现在市场上还有一种 Windows 95 兼容键盘，为 104 键。其中包括一个应用键，按该键也可以自动弹出上下文菜单。应用程序在响应弹出鼠标右键消息时，调用 LoadMenu()函数装载菜单资源(该菜单可以使用 AppStudio 创建)，然后调用 GetSubMenu()成员函数获取指向弹出式菜单的 CMenu 指针，最后调用 TrackPopupMenu()成员函数显示该浮动弹出式菜单，以及跟踪用户选择。

TrackPopupMenu()函数原型为：

```
BOOL TrackPopupMenu(UINT nFlags,int x,int y,CWnd* pWnd,LPCRECT lpRect = 0);
```

其中参数 nFlags 指定屏幕位置标志和鼠标键标志，通常为 TRM_LEFTALIGN(上下文菜单左边对齐 x 坐标)和 TPM_RIGHTBUTTON(跟踪右键)；x 为上下文菜单的水平屏幕坐标；y 指定上下文菜单的垂直屏幕坐标；pWnd 标识拥有此上下文菜单的窗口；lpRect 指向一个 RECT 结构或 CRect 对象，缺省值 0 表示当用户有了该菜单后单击时撤销该菜单。

## 6.2 普通下拉菜单

多数情况下，我们使用的都是普通下拉菜单。这种菜单为主窗口的缺省菜单。当窗口被创建时，缺省的菜单资源就会自动载入。Visual C++ 提供了一个非常方便的菜单资源编辑工具。借助该工具，可以定义相应菜单项中的所有字符串。每个菜单项都有一个 ID 值与之对应，如 ID_FILE_NEW。通过对该 ID 值进行映射在选择该项后 Windows 会自动调用与该 ID 值对应的映射函数。

在大多数菜单项中，都有一个字符的下面有下划线，如_File 菜单的 F 下面。在按下 Alt 键的同时按下 F 菜单项的待显示文字中添加 & 地址符号即可。

通过编写 Chap6 示例程序，读者将理解如何将缺省菜单改为读者自己的菜单，以及如何将菜单项与函数处理过程联上。

### 试一试

1.应用 MFC AppWizard(exe) 创建一个新项目 Chap6，选择 Finish 按钮，全部使用缺省选项。创建的类：

```
Application: CChap6App in Chap6.h and Chap6.cpp
```

```
Frame: CMainFrame in MainFrm.h and MainFrm.cpp
MDIChildFrame: CChildFrame in ChildFrm.h and ChildFrm.cpp
Document: CChap6Doc in Chap6Doc.h and Chap6Doc.cpp
View: CChap6View in Chap6View.h and Chap6View.cpp
```

特征：

```
+ Initial toolbar in main frame
+ Initial status bar in main frame
+ Printing and Print Preview support in view
+ 3D Controls
+ Uses shared DLL implementation (MFC42.DLL)
```

2.在 Workspace 窗口中选择 ResourceView 标签页。打开 Chap6 resource 文件夹，再打开 Menu 文件夹，如图 6.2 所示。双击 IDR_CHAP6TYPE，弹出菜单，如图 6.3 所示。

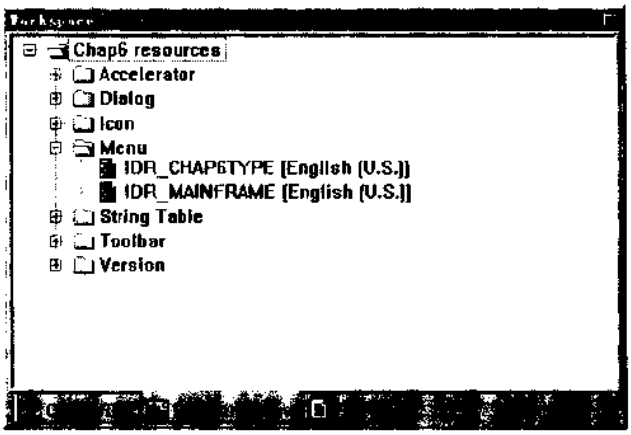



图 6.2 从 Workspace 窗口的 ResourceView 标签中编辑资源(菜单)

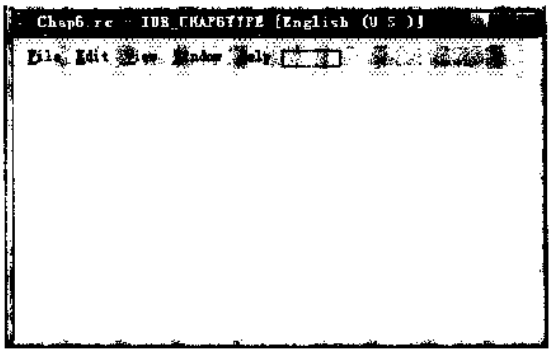


图 6.3 缺省的 IDR_CHAP6TYPE 菜单

3.用鼠标将菜单上最左边的菜单项框拖至 Windows 项的左边，再用键盘输入 &Draw。在键入 & 后，会弹出一个属性对话框，如图 6.4 所示。




图 6.4 属性对话框

接着,输入其它菜单项,生成如图 6.5 所示菜单。

| 菜单 | 菜单项 | 提示 Prompt | 命令 ID |
|---|---|---|---|
| Draw | &Line | Draw a Line | ID_DRAW_LINE |
| Draw | &Circle | Draw a Circle | ID_DRAW_CIRCLE |
| Draw | &Rectangle | Draw a Rectangle | ID_DRAW_RECTANGLE |

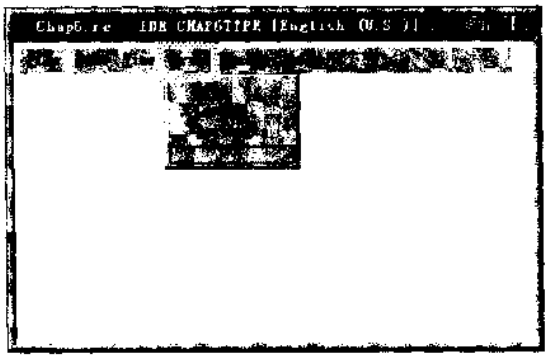


图 6.5 生成定制菜单

4.利用 ClassWizard 在 CChap6View 中增加消息处理函数。

| 对象 ID | 消息 | 成员函数 |
|---|---|---|
| ID_DRAW_LINE | COMMAND | OnDrawLine |
| ID_DRAW_CIRCLE | COMMAND | OnDrawCircle |
| ID_DRAW_RECTANGLE | COMMAND | OnDrawRectangle |

5. 编写 CChap6View 的成员函数。

```
void CChap6View::OnDrawCircle()
{
    AfxMessageBox("Chocse Draw\ \Circle item!");
}
```

```
void CChap6View::OnDrawLine()
{
    AfxMessageBox("Choose Draw\ \ Line item!");
}
```

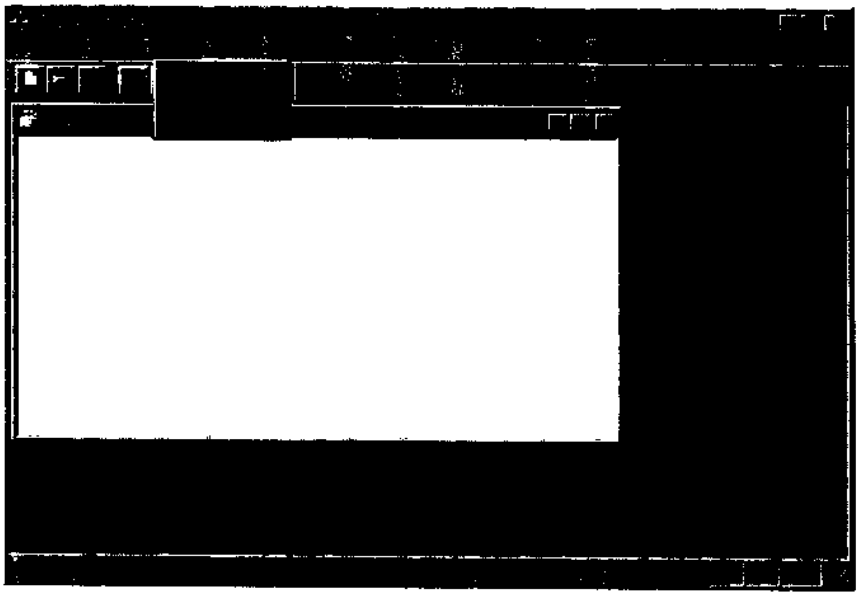


图 6.6 修改应用程序的普通菜单

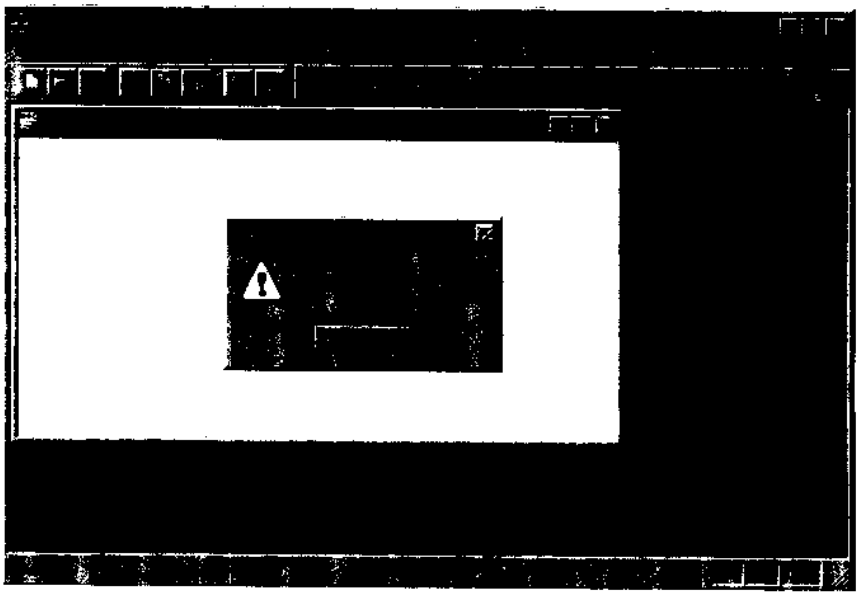


图 6.7 选择 Draw \ Line 后,应用程序的提示

```
void CChap6View::OnDrawRectangle()
{
    AfxMessageBox("Choose Draw\ \ Rectangle item!");
}
```

6.创建和运行应用程序。

从新生成的应用程序菜单中,分别选择 Draw \ Line、Draw \ Circle、Draw \ Rectangle 菜单项,系统会给出消息对话框提示用户选择了某菜单项。图 6.6 为新生成的应用程序的菜单。图 6.7 为选择 Draw \ Line 后,应用程序给用户的提示。

## 6.3 上下文菜单

在 Windows 系统中,上下文菜单就是单击鼠标右键时在鼠标光标处弹出的上下文菜单。为了操作方便,有时需要在单击右键时弹出上下文菜单。

我们可以从 Shared IDE Gallery 中添加 Pop-up Menu。

接下来,我们将在 Chap6 的 CMainFrame 和 CChap6View 中添加两个上下文菜单。

### 试一试

1.打开 Chap6 项目。

2.选择 Project \ Add To Project \ Components and Controls...菜单项,弹出 Components and Controls Gallery 对话框,如图 6.8 所示。双击 Vislal C++ Components 文件夹,显示出许多组

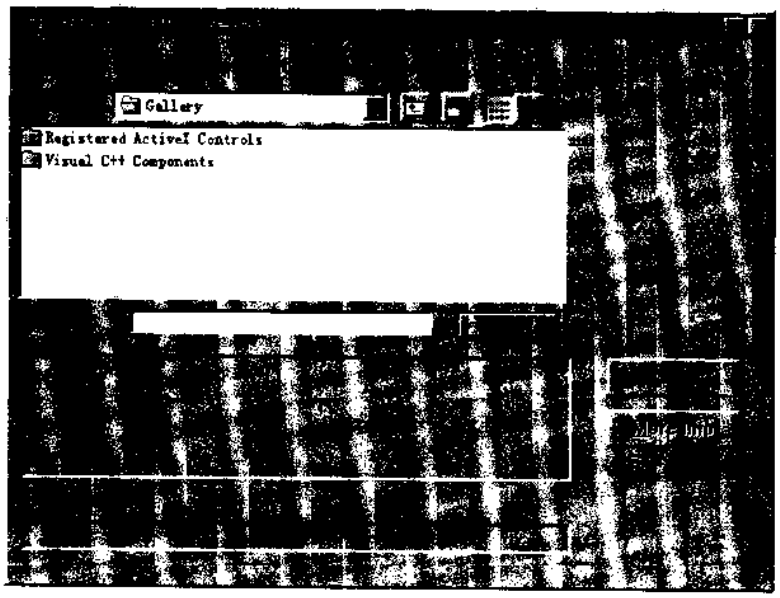



图 6.8 Components and Controls Gallery 对话框

件。再双击 Pop－up Menu 项，弹出 Microsoft Developer Studios 对话框，如图 6.9 所示，单击 OK 按钮。在新弹出的 Pop－up Menu 对话框(图 6.10)中，选择 CMainFrame，选择单击 OK 按钮，退回到 Components and Cnotrols Gallery 对话框。

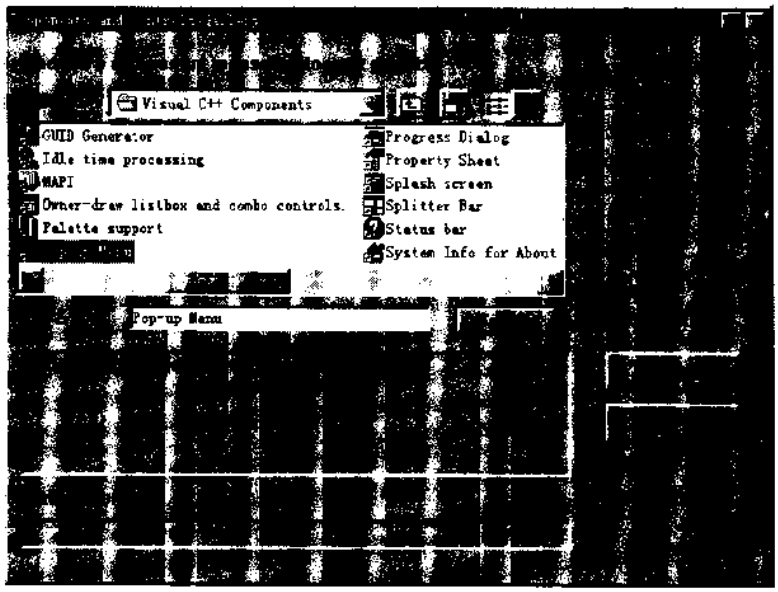



图 6.9 Microsoft Developer Studios 对话框




图 6.10 Pop－up Menu 对话框

3.在 Components and Contols Gallery 对话框中双击 Pop－up Menu 项，重复步骤 2 中的过程。注意，在 Pop－up Menu 对话框中，选择 CChap6View，单击 OK 按钮。再在 Components and Controls Gallery 对话框中单击 Close 按钮。

Pop－up Menu 组件给 CChap6View 添加了 OnContextMenu 函数。Chap6View.cpp 中所添加的具体内容如下：

```
BEGIN_MESSAGE_MAP(CChap6View, CView)
    ON_WM_CONTEXTMENU()
END_MESSAGE_MAP()
void CChap6View::OnContextMenu(CWnd * , CPoint point)
{
```

```
// CG: This block was added by the Pop-up Menu component
{
    if (point.x == -1 && point.y == -1){
        //keystroke invocation
        CRect rect;
        GetClientRect(rect);
        ClientToScreen(rect);

        point = rect.TopLeft();
        point.Offset(5, 5);
    }

    CMenu menu;
    VERIFY(menu.LoadMenu(CG_IDR_POPUP_CHAP6_VIEW));
    //获得 CG_IDR_POPUP_CHAP6_VIEW 菜单的索引值为 0 的子菜单
    CMenu* pPopup = menu.GetSubMenu(0);
    ASSERT(pPopup != NULL);
    CWnd* pWndPopupOwner = this;
    //获得主框架窗口指针
    while (pWndPopupOwner->GetStyle() & WS_CHILD)
        pWndPopupOwner = pWndPopupOwner->GetParent();
    //在 point 点处显示上下文菜单
    pPopup->TrackPopupMenu(TPM_LEFTALIGN | TPM_RIGHTBUTTON,
                           point.x, point.y, pWndPopupOwner);
}
}
```

CMainFrame 中所添加的内容，与 CChap6View 中的类似。接下来，对 CMainFrame 中的 OnContextMenu 进行修改。

4. 改写 CMainFrame::OnContextMenu 函数。

```
void CMainFrame::OnContextMenu(CWnd*, CPoint point)
{

    // CG: This block was added by the Pop-up Menu component
    {
        if (point.x == -1 && point.y == -1){
            //keystroke invocation
            CRect rect;
            GetClientRect(rect);
            ClientToScreen(rect);

            point = rect.TopLeft();
            point.Offset(5, 5);
```

```
        }

        CMenu menu;
        VERIFY(menu.LoadMenu(IDR_CHAP6TYPE));
        //获得 IDR_CHAP6TYPE 菜单的索引值为 4 的子菜单
        CMenu* pPopup = menu.GetSubMenu(4);
        ASSERT(pPopup != NULL);
        CWnd* pWndPopupOwner = this;
        //获得主框架窗口的指针
        while (pWndPopupOwner->GetStyle() & WS_CHILD)
            pWndPopupOwner = pWndPopupOwner->GetParent();
        //在 point 点处显示上下文菜单
        pPopup->TrackPopupMenu(TPM_LEFTALIGN | TPM_RIGHTBUTTON,
                               point.x, point.y, pWndPopupOwner);
    }
}
```

5. 将 ResourceView 中的 CG_IDR_POPUP_MAIN_FRAME 菜单删除。

6. 创建和运行应用程序。

分别在视窗口和框架窗口底板上单击鼠标右键，注意弹出的上下文菜单是否如图 6.11 和图 6.12 所示。

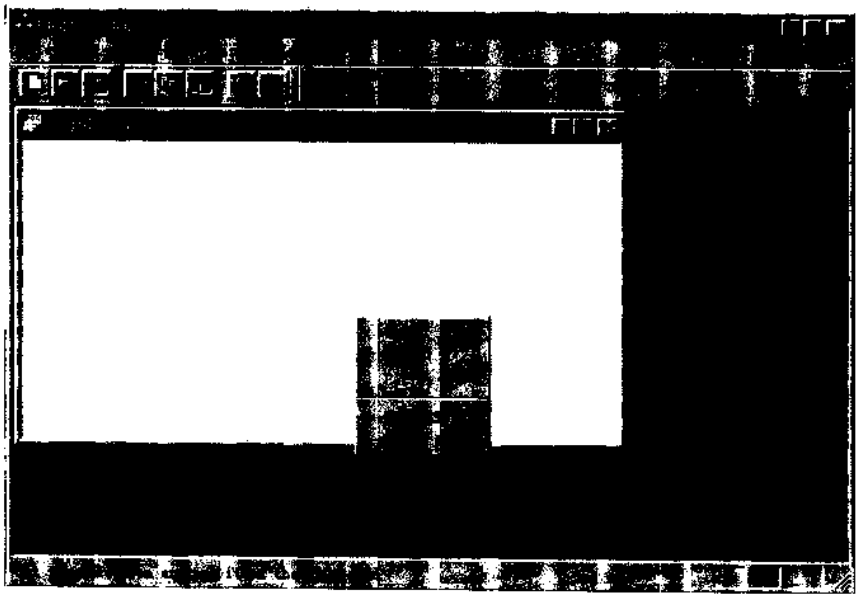


图 6.11 视窗口中的上下文菜单

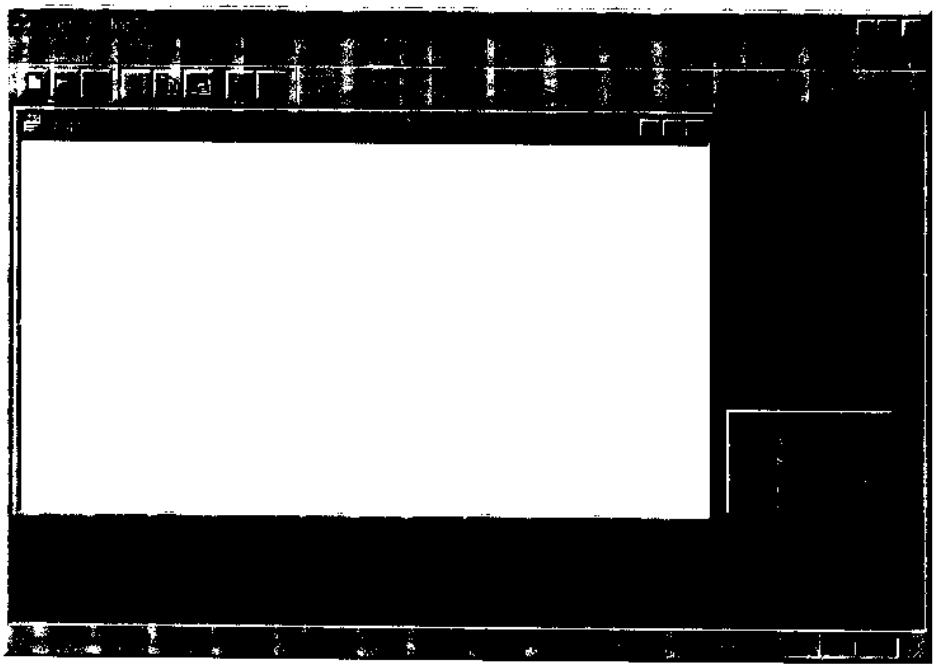


图 6.12 主框架窗口中的上下文菜单

## 6.4 改变菜单的显示状态

有时候我们需要根据应用程序的内部状态来对菜单项作相应的改变。例如，假设在应用程序的 Edit 菜单中有一个 Paste 选项。如果没有什么东西可粘贴时，Paste 就应当是无效状态。又如，为了标记住系统正处于何种状态，应对菜单项进行标记；为了醒目提示用户，应当将菜单项置为高亮度等。

为了使应用程序的状态与菜单项保持同步，我们可以采用 MFC 提供的一种方法。每当弹出式菜单第一次显示时，都会发送特殊的更新命令 UI 消息，该消息通常传递给与菜单项相联系的相同的对象。通过编辑与菜单项相联系的 UPDATE_COMMAND_UI 消息所映射的函数即可完成对菜单项状态的设置。此外，我们还可以用 CMenu::CheckMenuItem() 和 EnableMenuItem() 进行处理。

### 6.4.1 标记菜单项

为了对菜单项进行标记或去除标记，可按下述方法进行处理。

方法 1：利用 UPDATE_COMMAND_UI 消息映射函数。

```
BEGIN_MESSAGE_MAP(CChap6View, CView)
    ON_UPDATE_COMMAND_UI(ID_DRAW_LINE, OnUpdateDrawLine)
```

```
END_MESSAGE_MAP()
void CChap6View::OnUpdateDrawLine(CCmdUI * pCmdUI)
{
    pCmdUI->SetCheck(m_bDrawLine);
}
```

其中,ID_DRAW_LINE 为菜单项 Draw \ Line 的标识值,m_bDrawLine 记录了 Draw \ Line 菜单项是否被选中,True 表示选中,False 表示未选中。

方法 2:利用 CheckMenuItem()函数。

CheckMenuItem()的原型如下:

```
UINT CMenu::CheckMenuItem(UINT nIDCheckItem,UINT nCheck);
```

该函数在弹出式菜单中加上或去掉一个选中标志,nIDCheckItem 确定需改动的菜单项,nIDCheckItem 数能像确定一个菜单项一样确定一个弹出式菜单项。在选中弹出项时,它不需要更多的步骤,但顶层菜单不能被选中。一个弹出式菜单项必须用位置来选择,因为没有确定的菜单 ID 号与之相联。

该函数返回菜单项的先前状态。菜单项存在时,返回 MF_CHECKED 或 MF_UNCHECKED;菜单项不存在时,返回 -1。

### 6.4.2 使菜单项有效、无效

为了使菜单项有效或无效,可按下述方法进行处理。

方法 1:利用 UPDATE_COMMAND_UI 消息映射函数。

```
BEGIN_MESSAGE_MAP(CChap6View, CView)
    ON_UPDATE_COMMAND_UI(ID_EDIT_PASTE, OnUpdateEditPaste)
END_MESSAGE_MAP()
void CChap6View::OnUpdateEditPaste(CCmdUI * pCmdUI)
{
    pCmdUI->Enable(m_bEditPaste);
}
```

其中,ID_EDIT_PASTE 为菜单项 Edit \ Paste 的标识值。m_bEdit \ Paste 记录了 Edit \ Paste 菜单项是否被选取过,True 表示曾选择过,False 表示未曾选择。

方法 2:利用 CMenu::EnableMenuItem()函数。

CMenu::EnableMenuItem()的原型如下:

```
UINT CMenu::EnableMenuItem(UINT nIDEnableItem,UINT nEnable);
```

该函数可以设置菜单项,使菜单项可用、禁止、变灰。nIDEnableItem 确定要改动的菜单项。该函数返回菜单的预置状态(MF_DISABLED,MF_ENABLE 或 MF_GRAYED),或非法时为 -1。

### 6.4.3 高亮度显示菜单

可以用 CWnd::HiliteMenuItem()函数将某菜单项设为高亮度显示,或去掉高亮度显示。CWnd::HiliteMenuItem()函数的原型如下:

```
BOOL CWnd::HiliteMenuItem(CMenu * pMenu, UINT nIDHiliteItem,UINT nHilite);
```

其中 pMenu 为在一顶层菜单项上高亮度显示或除去高亮度显示的菜单。nIDHiliteItem 为待改变显示状态的菜单项的 ID 值。nHilite 为显示标志,取值为 MF_HILITE 和 MF_UNHILITE。而且这两个值不能在 ModifyMenu()成员函数中使用。

通过该函数的返回值可以确定菜单项是否高亮度显示。如果返回值为非 0 值,则菜单项处于高亮度显示状态,否则,处于一般状态。

### 6.4.4 改变菜单状态示例程序

现在,在 Chap6 应用程序中对菜单的状态进行处理,以便在选择 Draw \ Line、Draw \ Circle、Draw \ Rectangle 菜单项时,能作出相应的标记。根据是否选择了 Edit \ Copy 或 Edit \ Cut 选项,决定 Edit \ Paste 的状态有效或无效。

**试一试**

1.打开 Chap6 项目。

2.在 CChap6View 中增加变量。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| BOOL | m_bDrawLine | protected |
| BOOL | m_bDrawCircle | protected |
| BOOL | m_bDrawRectangle | protected |
| BOOL | m_bEditPaste | protected |

3.利用 ClassWizard 在 CChap6View 中增加消息处理函数。

| 对象 ID | 消息 | 成员函数 |
|---|---|---|
| ID_DRAW_LINE | UPDATE_COMMAND_UI | OnUpdateDrawline() |
| ID_DRAW_CIRCLE | UPDATE_COMMAND_UI | OnUpdateCircle() |
| ID_DRAW_RECTANGLE | UPDATE_COMMAND_UI | OnUpdateRectangle() |
| ID_EDIT_COPY | COMMAND | OnEditCopy() |
| ID_EDIT_CUT | COMMAND | OnEditCut() |
| ID_EDIT_PASTE | COMMAND | OnEditPaste() |
| ID_EDIT_PASTE | UPDATE_COMMAND_UI | OnUpdateEditPaste() |

4.编写 CChap6View 中的函数。

```
CChap6View::CChap6View()
{
    m_bDrawLine = FALSE;
    m_bDrawCircle = FALSE;
    m_bDrawRectangle = FALSE;
    m_bEditPaste = FALSE;
}

void CChap6View::OnDrawCircle()
{
    m_bDrawLine = FALSE;
    m_bDrawCircle = TRUE;
    m_bDrawRectangle = FALSE;
    AfxMessageBox("Choose Draw\ \Circle item!");
}

void CChap6View::OnDrawLine()
{
    m_bDrawLine = TRUE;
    m_bDrawCircle = FALSE;
    m_bDrawRectangle = FALSE;
    AfxMessageBox("Choose Draw\ \Line item!");
}

void CChap6View::OnDrawRectangle()
{
    m_bDrawLine = FALSE;
    m_bDrawCircle = FALSE;
    m_bDrawRectangle = TRUE;
    AfxMessageBox("Choose Draw\ \Rectangle item!");
}

void CChap6View::OnUpdateDrawCircle(CCmdUI * pCmdUI)
{
    pCmdUI->SetCheck(m_bDrawCircle);
}

void CChap6View::OnUpdateDrawLine(CCmdUI * pCmdUI)
{
    pCmdUI->SetCheck(m_bDrawLine);
}

void CChap6View::OnUpdateDrawRectangle(CCmdUI * pCmdUI)
```

```
{
    pCmdUI->SetCheck(m_bDrawRectangle);
}

void CChap6View::OnEditPaste()
{
    AfxMessageBox("There is something to paste!");
}

void CChap6View::OnEditCopy()
{
    m_bEditPaste = TRUE;
}

void CChap6View::OnEditCut()
{
    m_bEditPaste = TRUE;
}

void CChap6View::OnUpdateEditPaste(CCmdUI* pCmdUI)
{
    pCmdUI->Enable(m_bEditPaste);
}
```

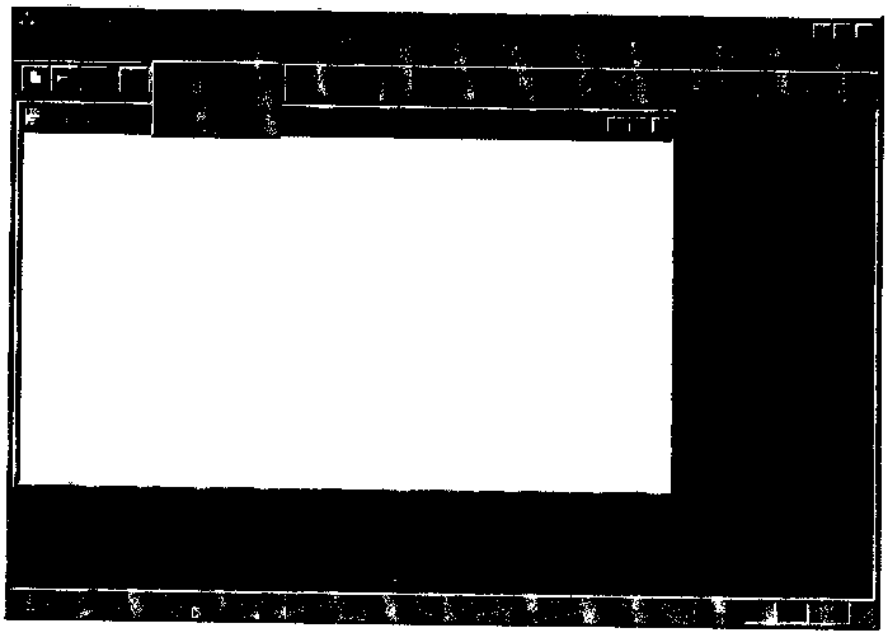


图 6.13 标志菜单项

5.创建和运行应用程序。

在该程序运行时,注意在选择 Draw \ Line、Draw \ Circle、Draw \ Rectangle 菜单项时,菜单项前面所做的标记。在选择 Draw \ Circle 菜单项以后,再打开 Draw 菜单,其状态应当如图 6.13 所示。在未选择 Edit \ Copy 或 Edit \ Cut 选项前,Edit \ Paste 应处于无效状态,如图 6.14 所示。

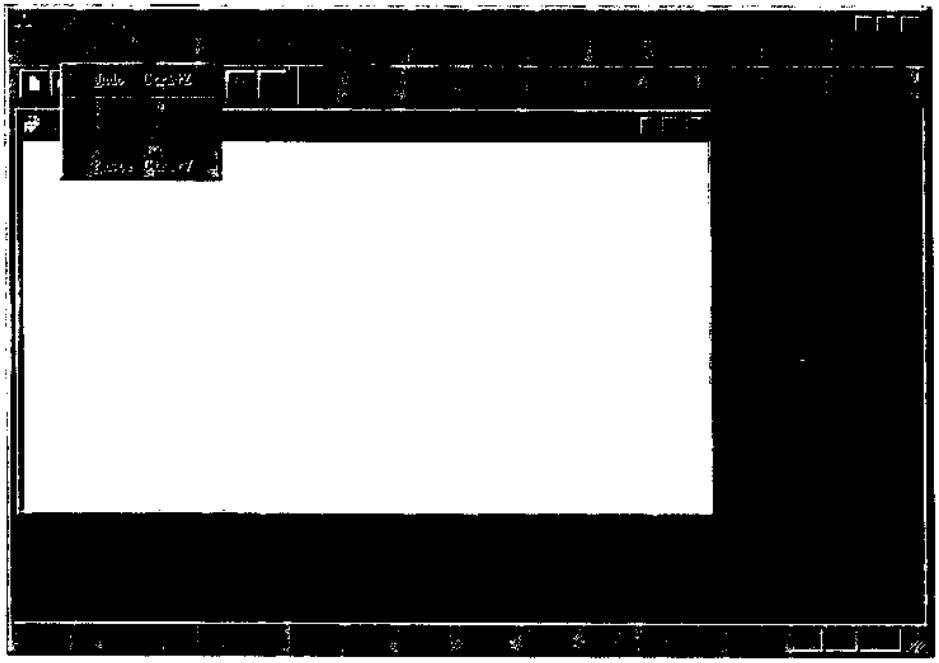


图 6.14 使用菜单项无效

## 6.5 动态地增、删和修改菜单项

在 Windows 应用程序中,时常需要根据应用程序的当前状态,动态地增、删和修改菜单项。MFC 的 CMenu 类提供了 AppendMenu()、InsertMenu()、RemoveMenu()、DeleteMenu()以及 ModifyMenu()函数用于动态地增、删和修改菜单项。这些函数的功能及用法如下:

1. AppendMenu()函数

AppendMenu()函数用于在菜单末尾添加一个新项。其函数原型如下:

```
BOOL CMenu::AppendMenu(UINT nFlags,UINT nIDNewItem = 0,LPCSTR lpszNewItem =
NULL);
BOOL CMenu::AppendMenu(UINT nFlags,UINT nIDNewItem,const CBitmap * pBmp);
```

应用程序可以通过设置 nFlags 的值来确定菜单项状态。当 nIDNewItem 确定了一个弹出式菜单时,它就被确认为是被扩展菜单的一部分。当菜单被破坏时,其扩展部分也将被破

环。一个扩展菜单同样可以从一个 CMenu 对象中获取以避免冲突。值得注意的是,值 MF_STRING 和 MF_OWNER_DRAW 对于位图版本的 AppendMenu()是无效的。

·MF_CHECKED 作为一个 MF_UNCHECKED 的开关,放置一个缺省的选中标志到选项后面。当应用程序提供选中标志位图时(参见 SetMenuItemBitmaps()成员函数),"选中标志打开"的点位图将被显示。

·MF_UNCHECKED 作为 MF_CHECKED 的开关,去掉选项后的选中标志。当应用程序提供了选中标志的位图(参见 SetMenuItemBitmaps()成员函数)时,"选中标志关闭"点位图将被显示。

·MF_DISABLED 使菜单项禁止,使它不能被选择,但不把它变灰。

·MF_ENABLED 使菜单项能被选择,并把它从变灰状态恢复。

·MF_GRAYED 使菜单项禁选并使之变灰。

·MF_MENUBARBREAK 在静态菜单中把选项放在新的行中或把弹出式菜单放于新栏中。新的弹出式菜单栏将与老的用竖线分开。

·MF_MENUBREAK 将一菜单项放于静态菜单的新一行中或弹出菜单的新栏中;新旧菜单不用分割线分开。

·MF_OWNERDRAW 确认选项是否为自绘制风格。当菜单为头一次显示时,拥有菜单的窗口返回一条 WM_MEASUREITEM 消息,它返回菜单项的高度和宽度,WM_DRAWITEM 消息是在拥有窗口改变菜单选择的外观时发送的,它对顶层菜单选项无效。

· MF_POPUP 确认一个选项是否有弹出式菜单与之相关。ID 参数可以确认一个与此选项相关的弹出菜单的句柄,它可用来为一个顶层菜单或弹出菜单的一个选项添加一个相关的弹出式菜单。

·MF_SEPARATOR 绘制一条垂直分界线,仅用于弹出式菜单,此线可以变灰、禁止或点亮,其他参数将忽略。

·MF_STRING 确定菜单选项是一个字串。

下列选择组间各项是互斥的,不能一起使用:

·MF_DISABLED,MF_ENABLED 和 MF_GRAYED

·MF_STRING,MF_OWNERDRAW,MF_SEPARATOR 和点位图版本

·MF_MENUBARBREAK 和 MF_MENUBREAK

·MF_CHECKED 和 MF_UNCHECKED

无论何时,只要窗口中的菜单被改变(无论窗口显示与否),应用程序都会调用 CWnd::DrawMenuBar。

该函数在调用成功时返回非 0,否则返回 0

2. InsertMenu()函数

InsertMenu()函数用于在菜单中插入菜单项,其原型如下:

```
BOOL CMenu::InsertMenu(UINT nPosition, UINT nFlags,
```

```
                                UINT nIDNewItem = 0,LPCSTR lpszNewItem = NULL);
BOOL CMenu::InsertMenu(UINT nPosition, UINT nFlags,
                                UINT nIDNewItem,const CBitmap * pBmp);
```

该函数用于向一个菜单中 nPosition 指定的位置插入一个菜单项,并把其它项向后移,其应用程序可以用对 nFlags 赋值的办法设置此菜单项的状态。只要窗口中的菜单发生变化(无论窗口是否显示),都要调用 CWnd::DrawMenuBar()。nIDNewItem 用于确定一个弹出菜单。在它插入菜单时,它会变为菜单的一部分。若菜单被破坏,它也随之被破坏。插入菜单项应从一个 CMenu 对象中获取,以免发生重复。

如果活动多文档窗口(MDI)的子窗口变为最大,则应用程序调用此函数在子窗口菜单中插入一个弹出式菜单,并设置 MF_BYPOSITION 标志。菜单将插入到所预期位置的左边。因为活动 MDI 子窗口的控制菜单将被插入到 MDI 框架窗口菜单条的第一个位置。为使位置正确,应用程序必须对位置变量加 1。一个应用程序可以用 WM_MDIGETACTIVE 消息确认当前活动子窗口是否最大。

该函数在成功时返回非 0,否则为 0。

3. RemoveMenu()函数

RemoveMenu()函数的函数原型为:

```
BOOL CMenu::RemoveMenu(UINT nPosition,UINT NFlags);
```

该函数用于删除一个弹出式菜单的菜单选项,但它不破坏一个弹出菜单的句柄,菜单仍可以被使用。在调用此函数之前,应用程序可以调用 GetSubMenu()成员函数获取弹出的 CMenu 对象以便重新使用。只要位于窗口内的菜单被改变(无论窗口显示与否),应用程序就必须调用 DrawMenuBar()函数。

该函数在成功时返回非 0,否则为 0。

4. DeleteMenu()函数

DeleteMenu()函数的函数原型如下:

```
BOOL CMenu::DeleteMenu(UINT nPosition,UINT nFlags);
```

该函数用于从菜单中删除一项。若菜单项有一隶属弹出菜单,DeleteMenu()破坏弹出菜单,释放由此弹出菜单占用的内存。当一个窗口中有菜单变化(无论窗口是否显示),应用程序会调用 DrawMenuBar()函数。

该函数成功时返回非 0,否则为 0。

5. ModifyMenu()函数

ModifyMenu()函数用于改变由 nPosition 确定的已存在的菜单项。其函数原型如下:

```
BOOL CMenu::ModifyMenu(UINT nPosition, UINT nFlags,
                                UINT nIDNewItem = 0,LPCSTR lpszNewItem = NULL);
BOOL CMenu::ModifyMenu(UINT nPosition, UINT nFlags,
                                UINT nIDNewItem,const CBitmap * pBmp);
```

函数改变由 nPosition 确定的已存在菜单项。应用程序可以用 nFlags 值来确定选项的新

状态。若此菜单改变了与此菜单相联的弹出菜单,它将破坏老的弹出式菜单并释放由此菜单占用的内存。当 nIDNewItem 确定的弹出式菜单变成插入的菜单的一部分时,若菜单被破坏,则插入菜单也将被破坏。插入菜单应从 CMenu 对象中获得,以避免冲突。

只要驻留于窗口的菜单被改变(无论窗口是否显示),应用程序都将调用 DrawMenuBar()函数。

当该函数被用来改变一个已存在选项状态的属性时,它比使用 CheckMenuItem()和 EnableMenuItem()成员函数快得多。

该函数调用成功时,返回非0,否则为0。

接下来,我们对 Chap6 项目中的菜单在程序运行时进行动态地修改。修改后的程序将在主菜单中加入 ObjSnap 菜单,其下方有 Endpoint、Midpoint、Center、Quardrant 以及 More 菜单项。在选择 More 选项以后,More 选项变为 Less,同时在菜单中添加 Intersection、Tangent 菜单项。单击 Less 又恢复原状。另一方面,在视中单击右键时,弹出的菜单中会添加 Mark All 和 Clear All 选项,并使 Clear All 处于无效状态。

## 试一试

1. 打开 Chap6 项目。

2. 在 ResourceView 标签的 String Table 中添加下列值。

| ID 值 | Caption 值 |
|---|---|
| ID_OBJSNAP_ENDPOINT | Snap to Endpoint |
| ID_OBJSNAP_MIDPOINT | Snap to Midpoint |
| ID_OBJSNAP_CENTER | Snap to Center |
| ID_OBJSNAP_QUARDRANT | Snap to Quardrant |
| ID_OBJSNAP_CHANGE | Snap to Shorter |
| ID_OBJSNAP_INTERSECTION | Snap to Intersection |
| ID_OBJSNAP_TANGENT | Snap to Tangent |
| ID_MARK_ALL | Mark all |
| ID_CLEAR_ALL | Clear all |

3. 在 ResourceView 中修改菜单 IDR_CHAP6TYPE(如图 6.15 所示)。

| 菜单 | 菜单项 | 提示 | 命令 ID |
|---|---|---|---|
| Objsnap | BeingDeleted | 无 | ID_IDIT_BEINGDELETEB |

4. 修改 CChap6App::InitInstance 函数。

```
BOOL CChap6App::InitInstance()
{
    AfxEnableControlContainer();
    ...
```

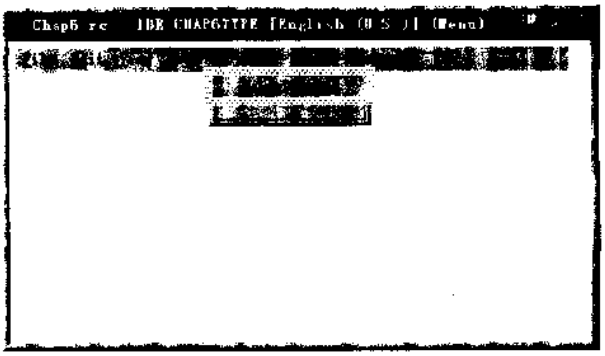


图 6.15 修改 IDR_CHAP6 TYPE 菜单

```
// Dispatch commands specified on the command line
if (! ProcessShellCommand(cmdInfo))
      return FALSE;

//获得框架菜单的指针
CMenu * pMenu = m_pMainWnd - > GetMenu();
ASSERT(pMenu);
//获得框架菜单的第五项菜单的指针
CMenu * pSubMenu = pMenu - > GetSubMenu(4);
ASSERT(pSubMenu);
//删除 BeingDelted 菜单项
pSubMenu - > DeleteMenu(0,MF_BYPOSITION);
//添加菜单项
pSubMenu - > AppendMenu(MF_STRING,ID_OBJSNAP_ENDPOINT,"&Endpoint");
pSubMenu - > AppendMenu(MF_STRING,ID_OBJSNAP_MIDPOINT,"&Midpoint");
pSubMenu - > AppendMenu(MF_STRING,ID_OBJSNAP_CENTER,"&Center");
pSubMenu - > AppendMenu(MF_STRING,ID_OBJSNAP_QUARDRANT,"&Quardrant");
pSubMenu - > AppendMenu(MF_STRING,ID_OBJSNAP_CHANGE,"&More");
//高亮度显示菜单项
m_pMainWnd - > HiliteMenuItem(pSubMenu,ID_OBJSNAP_CHANGE,MF_HILITE);


// The main window has been initialized, so show and update it.
pMainFrame - > ShowWindow(m_nCmdShow);
pMainFrame - > UpdateWindow();

return TRUE;
}
```

5. 在 CChap5AView 中增加变量。

| 变量类型 | 变量名称 | 访问权限 |
| --- | --- | --- |
| BOOL | m_bMarkAll | protected |

6.在 CChap6View 中手工增加消息处理函数。

在 Chap6View.h 中添加：

```
class CChap6View : public CView
{
      ...
protected:
      ...
      afx_msg void OnContextMenu(CWnd*, CPoint point);
      afx_msg void OnObjsnapChange();
      afx_msg void OnMarkAll();
      afx_msg void OnClearAll();
      //{{AFX_MSG(CChap6View)
      ....
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};
```

在 Chap6View.cpp 中添加：

```
BEGIN_MESSAGE_MAP(CChap6View, CView)
      ON_COMMAND(ID_OBJSNAP_CHANGE, OnObjsnapChange)
      ON_COMMAND(ID_MARK_ALL, OnMarkAll)
      ON_COMMAND(ID_CLEAR_ALL, OnClearAll)
      //{{AFX_MSG_MAP(CChap6View)
      ON_COMMAND(ID_DRAW_LINE, OnDrawLine)
      .....
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()

void CChap6View::OnObjsnapChange()
{
}

void CChap6View::CnMarkAll()
{
}

void CChap6View::OnClearAll()
{
}
```

7.修改 CChap6View::CChap6View 构造函数。

```
CChap6View::CChap6View()
```

```
{
    m_bDrawLine = FALSE;
    m_bDrawCircle = FALSE;
    m_bDrawRectangle = FALSE;
    m_bEditPaste = FALSE;
    m_bMarkAll = FALSE;
}
```

8.编写 CChap6View::OnObjsnapChange 函数。

```
void CChap6View::OnObjsnapChange()
{
    static BOOL bLong = FALSE;
    //获得框架菜单的指针
    CMenu * pMenu = AfxGetMainWnd() -> GetMenu();
    ASSERT(pMenu);
    //获得框架菜单的第五项菜单的指针
    CMenu * pSubMenu = pMenu -> GetSubMenu(4);
    ASSERT(pSubMenu);
    if(! bLong)
    {
        //添加菜单项
        pSubMenu -> AppendMenu(MF_STRING,ID_OBJSNAP_INTERSECTION,
                               "&Intersection");
        pSubMenu -> AppendMenu(MF_STRING,ID_OBJSNAP_TANGENT,"&Tangent");
        pSubMenu -> ModifyMenu(ID_OBJSNAP_CHANGE,MF_BYCOMMAND,
                               ID_OBJSNAP_CHANGE,"&Less");
    }
    else
    {
        //删除菜单项
        pSubMenu -> DeleteMenu(ID_OBJSNAP_INTERSECTION,MF_BYCOMMAND);
        pSubMenu -> DeleteMenu(ID_OBJSNAP_TANGENT,MF_BYCOMMAND);
        pSubMenu -> ModifyMenu(ID_OBJSNAP_CHANGE,MF_BYCOMMAND,
                               ID_OBJSNAP_CHANGE,"&More");
    }
    bLong = ! bLong;
}
```

9.编写 CChap6View::OnMarkAll 函数。

```
void CChap6View::OnMarkAll()
{
    m_bMarkAll = TRUE;
    AfxMessageBox("Mark all items!");
}
```

10. 编写 CChap6View::OnClearAll 函数。

```
void CChap6View::OnClearAll()
{
    m_bMarkAll = FALSE;
    AfxMessageBox("Clear all items!");
}
```

11. 修改 CChap6View:: OnContextMenu 函数。

```
void CChap6View::OnContextMenu(CWnd*, CPoint point)
{

    // CG: This block was added by the Pop-up Menu component
    {
        if (point.x == -1 && point.y == -1){
            //keystroke invocation
            CRect rect;
            GetClientRect(rect);
            ClientToScreen(rect);

            point = rect.TopLeft();
            point.Offset(5, 5);
        }

        CMenu menu;
        VERIFY(menu.LoadMenu(CG_IDR_POPUP_CHAP6_VIEW));

        CMenu* pPopup = menu.GetSubMenu(0);
        ASSERT(pPopup != NULL);

        pPopup->AppendMenu(MF_SEPARATOR);
        pPopup->AppendMenu(MF_STRING,ID_MARK_ALL,"&Mark All");
        pPopup->AppendMenu(MF_STRING,ID_CLEAR_ALL,"C&lear All");

        if(m_bMarkAll == FALSE)
            pPopup->EnableMenuItem(ID_CLEAR_ALL,MF_DISABLED);
        else
            pPopup->EnableMenuItem(ID_MARK_ALL,MF_DISABLED);


        CWnd* pWndPopupOwner = this;

        while (pWndPopupOwner->GetStyle() & WS_CHILD)
            pWndPopupOwner = pWndPopupOwner->GetParent();

        pPopup->TrackPopupMenu(TPM_LEFTALIGN | TPM_RIGHTBUTTON,
```

```
                                        point.x, point.y,
                    pWndPopupOwner);
        }
}
```

12. 修改 CChap6View::OnContextMenu 函数。

```
void CMainFrame::OnContextMenu(CWnd*, CPoint point)
{
    {
        ......
        CMenu menu;
        VERIFY(menu.LoadMenu(IDR_CHAP6TYPE));

        //将 4 改为 5。因为加入 ObjSnap 菜单后,Windows 项变为第六项。
        CMenu* pPopup = menu.GetSubMenu(5);
        ASSERT(pPopup != NULL);
        CWnd* pWndPopupOwner = this;
```

13. 创建和运行应用程序。

从菜单中选择 ObjSnap 菜单下的 More 或 Less 菜单项,注意该菜单的动态伸长与缩短,该菜单项的显示内容在 More 与 Less 之间互换(见图 6.16 和图 6.17)。在视窗口中单击右键,注意在上下文菜单中动态添加的菜单项,以及动态地设置上下文菜单中菜单项的状态。

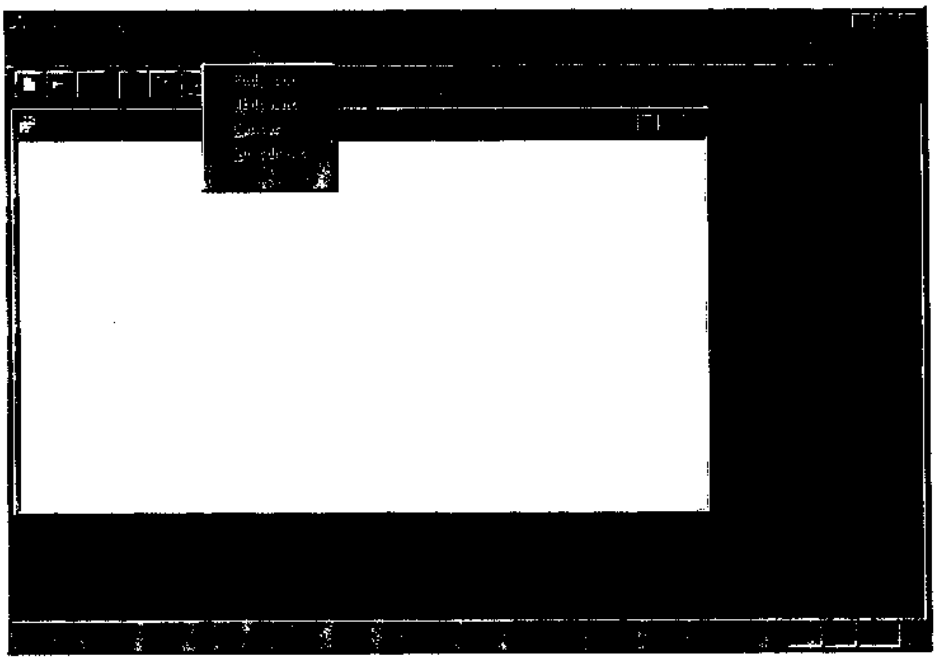


图 6.16 选项较少的菜单

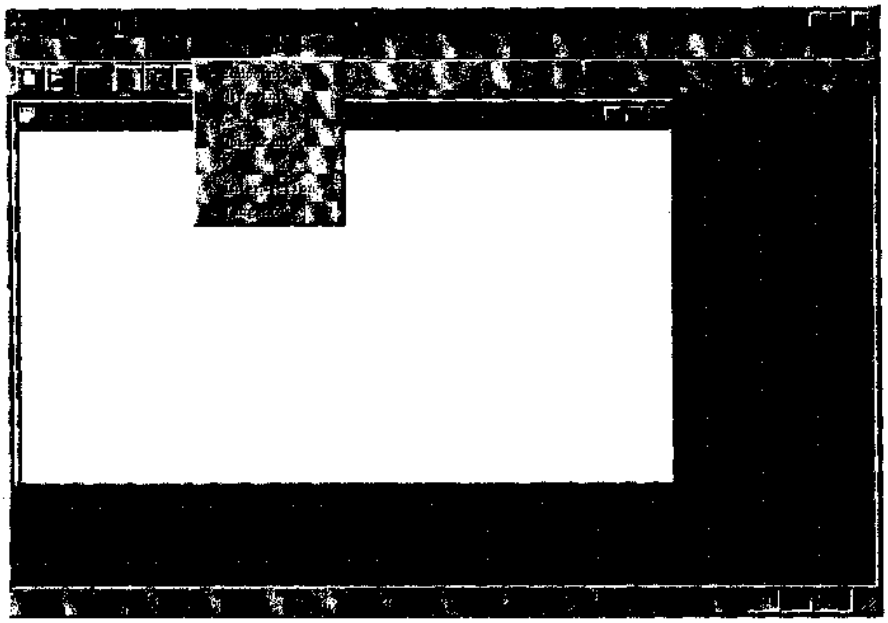


图 6.17 选项较多的菜单

## 6.6 菜单间的切换

有些应用程序的功能相当复杂,各功能模块独立性很强。此时,可以为该应用程序创建多个菜单,在运行不同的功能模块时装载相应的菜单。

接下来,我们对 Chap6 项目进行修改,使其可以在多个菜单间进行切换。用户可以通过按下 F1、F2、F3 键改变菜单。

### 试一试

1. 打开 Chap6 项目。

2. 在 ResourceView 中添加两个菜单。读者可在其中输入自己感兴趣的菜单项。为了使操作简单,我们仅在 IDR_MENUI 中添加了 Menu1 菜单及下两项 Item1 和 Item2;在 IDR_MENU2 中添加了 Menu 菜单,以及其下的两菜单项 Item1 和 Item2。

3. 在 CChap6App 中增加变量。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| CMenu * | m_pMenuOrg | public |
| CMenu * | m_pMenu1 | public |
| CMenu * | m_pMenu2 | public |

4.修改 CChap6App::InitInstance 函数。

```
BOOL CChap6App::InitInstance()
{
    AfxEnableControlContainer();
    ......
    //高亮度显示菜单项
    m_pMainWnd->HiliteMenuItem(pSubMenu,ID_OBJSNAP_CHANGE,MF_HILITE);

    //为菜单分配空间
    m_pMenuOrg = new CMenu;
    m_pMenu1 = new CMenu;
    m_pMenu2 = new CMenu;
    //装载菜单
    ASSERT(m_pMenuOrg->LoadMenu(IDR_CHAP5TYPE));
    ASSERT(m_pMenu1->LoadMenu(IDR_MENU1));
    ASSERT(m_pMenu2->LoadMenu(IDR_MENU2));
    //记录缺省的菜单项
    m_pMenuOrg = pMenu;
    //或者 m_pMenuOrg = m_pMainWnd->GetMenu();

    //为菜单分配空间
    m_pMenu1 = new CMenu;
    m_pMenu2 = new CMenu;
    //装载菜单
    ASSERT(m_pMenu1->LoadMenu(IDR_MENU1));
    ASSERT(m_pMenu2->LoadMenu(IDR_MENU2));

    // The main window has been initialized, so show and update it.
    pMainFrame->ShowWindow(m_nCmdShow);
    pMainFrame->UpdateWindow();

    return TRUE;
}
```

5.利用 ClassWizard 在 CChap6View 中增加消息处理函数。

| 消息 | 成员函数名 |
|---|---|
| WM_KEYUP | OnKeyUp |

6. 编写 CChap6View::OnKeyUp 函数。

```
void CChap6View::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
{
```

```
    CWnd*   pMainWnd = AfxGetMainWnd();
    extern CChap6App theApp;
    //根据用户所按下的按钮显示不同的菜单
    switch(nChar)
    {
    case VK_F1:
        pMainWnd->SetMenu(theApp.m_pOrgMenu);
        break;
    case VK_F2:
        pMainWnd->SetMenu(theApp.m_pMenu1);
        break;
    case VK_F3:
        pMainWnd->SetMenu(theApp.m_pMenu2);
    default:
        break;
    }
}
```

7.创建和运行应用程序。

按下F1、F2、F3键,观察主框架上的菜单在三个菜单之间的切换。

## 6.7 本章小结

MFC提供了一个功能强大的CMenu类,它用于创建、修改和管理菜单。通过学习本章,读者应当了解从菜单中调用命令的基本原理,学会了普通菜单、上下文菜单的创建过程,掌握了动态地增、删、修改菜单、改变菜单的显示状态以及如何在应用程序的多个菜单间进行切换。

# 第七章 工 具 条

**本章要点**

- 工具条的基本组成元素
- 工具条的创建和消息处理过程
- 工具条的泊放
- 工具条的提示处理
- 制作普通工具条
- 制作高级工具条

迄今为止，我们的应用程序都包含了工具条。我们接受了 AppWizard 的缺省 Initial Toolbar 选项。这一章，我们将学习如何创建自己的工具条。

## 7.1 概 述

本节先讲述组成工具条的基本元素——资源和 CToolBar 类。接着，讲解了工具条的创建和处理过程。最后，讲述了工具条的泊放与提示。

学完本节，读者将基本理解工具条。

### 7.1.1 工具条位图

读者可能认为工具条中的每一个按钮都有自己的位图，但实际上整个工具条只有一个位图。如图 7.1 所示，该图为应用程序缺省工具条的位图。



图 7.1 缺省工具条的位图

工具条中的每一个按钮都有如表 7.1 所示的几种状态。

| 状态 | 含义 |
|---|---|
| 0 | 正常状态，未被按下 |
| TBBS_PRESSED | 当前被用鼠标选中 |
| TBBS_CHECKED | 处于检查状态 |
| TBBS_DISABLED | 无法使用 |

（续表）

| 状态 | 含义 |
| --- | --- |
| TBBS_INDETERMINATE | 处于允许状态，但既不是up也不是down |
| TBBS_CHECKED \| TBBS_DESABLED | 处于检查状态，但无法使用 |

按钮有两种行为方式，一种是开关式按钮，只有被鼠标选中时才处于按下状态；另一种是检查框按钮通过鼠标按键来使其在up和down两种状态之间转换。标准应用框架工具条中的所有按钮都是开关式按钮。

### 7.1.2 CToolBar类

MFC用CToolBar类表示工具条，其功能包括工具条的创建、按钮风格和尺寸的设置、工具条属性的检取等功能。CToolBar类定义了缺省构造函数、Create()、LoadBitmap()、SetButtons()、SetButtonInfo()等成员函数。对这几个函数说明如下。

1. Create()函数

Create()函数用于创建一个Windows工具条，并将其连接到CToolBar对象上，同时设置工具条的缺省高度。该函数原型为：

```
BOOL CToolBar::Create(CWnd* pParentWnd,DWORD dwstyle=WS_CHILD|
    WS_VISIBLE|CBRS_TOP,UINT nID=AFX_IDW_TOOLBAR);
```

其中参数pParentWnd为指向父窗口CWnd对象的指针；dwstyle指定工具条的风格，除了标准的Windows风格外，还可以是CBRS_TOP（工具条位于框架窗口的顶部）、CBRS_BOTTOM（工具条位于框架窗口的底部）、CBRS_NOALIGN（当父窗口大小改变时，工具条不被重定位）和CBRS_TOOLTIPS（显示工具提示）；nID指示该工具条的子窗口ID。

如果函数调用成功，则返回一个非零值，否则返回零。

2. LoadBitmap()函数

成员函数LoadBitmap()用于从位图资源中调入工具条位图。该函数的原型如下：

```
BOOL CToolBar::LoadBitmap(LPSCTR lpszResourceName);
BOOL CToolBar::LoadBitmap(UINT nIDResource);
```

该函数调用由lpszResourceName或nIDResource所指定的位图。位图中应包含工具条中每一个按钮的图像。如果图像不是标准尺寸（16点宽15点高），调用SetSize设置按钮和它的图像的尺寸。

如果函数操作成功，则返回非0值，否则返回0。

3. SetButtons函数

成员函数SetButtons()用于将各个按钮与位图中的相应图像相对应。该函数的原型如下：

```
BOOL CToolBar::SetButtons(const UINT FAR * lpIDArray, int nIDCount);
```

该函数把工具条中的每一个按钮的命令ID设置为lpIDArray数组中的对应元素值。

如果数组中的元素为 ID_SEPARATOR，则在工具条中的对应位置建立一个分隔区。这个函数同时还把按钮和分隔区的风格分别设置为 TBBS_BUTTON 和 TBBS_SEPARATOR，并指定每一个按钮的图像索引。图像索引指定按钮的图像在工具条资源位图中的位置。

读者不能在位图的分隔区中作图像，因为这个函数不为分隔区分配图像索引。如果在用户的应用程序中，工具条的位置 0、1、3 是按钮，那么位置 2 为分隔区。位置中的 0、1、2 图像分别指派给工具条中位置 0、1、3 处的按钮。

如果 lpIDArray 是 NULL，那么这个函数将给出 nIDcount 指定的项分配内存。使用 SetButtonInfo 设置每项的属性。

如果函数操作成功，则返回非 0 值。否则返回 0。

此外，成员函数 SetButtonInfo()可以设置按钮的 ID、风格以及图像索引编号。在 Windows 中，共有三种按钮风格：TBBS_BUTTON(标准按钮，缺省值)、TBBS_SEPARATOR(分隔符)和 TBBS_CHECKBOX(自动复选按钮)。

上面曾提过，工具条中所有按钮都具有相同的大小，缺省是 24×22 个像素。按钮上的图像也具有相同的大小，缺省为 16×15 个像素。如果不用缺省设置，则必须调用成员函数 SetSizes()来改变大小。另外，还可使用成员函数 SetHeight()来改变工具条的高度。

从 Visual C++ 5.0 起 CToolBar 类还增加了一个成员函数 LoadToolBar()，它直接装载工具条资源，相当于完成 LoadBitmap()和 SetButtons()的功能，这使得工具条的创建更为简便。该函数原型为：BOOL LoadToolBar(UINT nIDResource)，其中 nIDResource 为待装载的工具条资源 ID。

### 7.1.3 工具条的创建和处理

创建一个工具条通常可以分为四个步骤：

(1)构造一个 CToolBar 对象。

(2)调用 Create 函数创建工具条，并将该工具条与 CToolBar 对象相连接。

(3)调用 LoadBitmap()函数调入存有工具条按钮图像的位图。

(4)调用 SetButtons()设置按钮风格并将各个按钮与位图中的相应图像相联系。工具条的所有按钮图像都保存在同一个位图中，该位图为每个按钮存放一幅图像，并且图像必须一幅接一幅地存放。

按钮通过命令 ID 数组与它们所产生的命令相对应，命令 ID 在数组中的位置与按钮图像在位图中的位置是一致的。在 AppWizard 生成缺省的工具条时，在主框架窗口类的实现文件 MAINFRM.CPP 中已经生成“按钮”数组。该数组与下面给出的数组类似。

```
static UINT BASED_CODE buttons[] =
{
    // same order as in the bitmap 'toolbar.bmp'
    ID_FILE_NEW,
    ID_FILE_OPEN,
    ID_FILE_SAVE,
```

```
    ID_SEPARATOR,
    ID_EDIT_CUT,
    ID_EDIT_COPY,
    ID_EDIT_PASTE,
    ID_SEPARATOR,
    ID_FILE_PRINT,
    ID_APP_ABOUT,
    ID_CONTEXT_HELP,
};
```

其中 ID_SEPARATOR 常量标识分隔符，以把按钮分成组。在确定按钮位置时，会忽略分隔符。该“按钮”数组对应的工具条如图 7.1 所示。

在 MAINFRM.CPP 的 OnCreate()函数中创建工具条，代码如下：

```
if(! m_wndToolBar.Create(this) |
! m_wndToolBar.LoadBitmap(IDR_MAINFRAME)||
! m_wndToolBar.SetButtons(buttons,sizeof(buttons/sizeof(UINT))
)
    TRACE("Failed to create toolbar\n");
    return -1;  //创建失败
|
```

程序运行时，通过调用函数 SetButtons()和 SetButtonInfo()，可以动态地修改按钮的形状、位置和风格。

工具条中的按钮通常都有相应的菜单项命令，因而只需将“按钮”数组中的 ID 置为相应的菜单项命令的 ID，就可以通过工具条按钮或相应的菜单项命令执行同一段代码。这样应用程序可以对任何命令消息作出响应，不论它是来自菜单项还是工具条按钮，都调用同一处理函数进行处理。读者现在看到了，对于同一条命令，可以有多个用户接口，例如菜单项、键盘加速键、工具条按钮以及后面要介绍的另一种控制条——对话框条等。对于所有等价的用户接口对象的处理代码可以封装在同一个地方。按钮的消息种类，只有 ON_COMMAND 和 ON_UPDATE_COMMAND_UI 两种，这与菜单相同，而消息映射格式也和菜单相同。如果按钮 ID 有相应的菜单项命令，则该按钮的消息映射和菜单项命令相同，只需用 ClassWizard 加入消息映射即可。如果没有相应的菜单项命令，则无法用 ClassWizard 加入消息映射，必须手工加入消息映射。

例如，在 7.2 节中有一个绘图工具条，它是用来触发绘图命令的。在资源编辑器中该工具条的布局如图 7.3 所示。然后在文件 MAINFRM.CPP 的 OnCreate()函数中进行创建，代码如下：

```
if(! m_wndToolBar.Create(this)||
! m_wndToolBar.LoadToolBar(IDR_DRAW))
{
    TRACE0("Failed to create toolbar\n");
```

```
    return -1;    //创建失败
}
```

有关详细示例可参考7.2节中的有关创建普通工具条一节的内容。

### 7.1.4 工具条的泊位和漂浮

在Visual C++6.0中,工具条具有泊位和漂浮功能。工具条可以"泊位"到父窗口的任意边上,也可以在自己的小框架窗口内"漂浮"。

读者可以通过以下三步来设置工具条的泊位功能:

· 使可泊位在框架窗口中有效。

· 使泊位对工具条有效。

· 泊位工具条到框架窗口。

1.使可泊位在框架窗口中有效

要在框架窗口中泊位工具条,首先使框架窗口允许泊位。通过调用CFrameWnd类的成员函数EnableDocking()来决定框架窗口中工具条的泊位位置。缺省情况下,工具条是按照顶部、底部、左边和右边的顺序来泊位的。该函数的原型为:

```
void EnableDocking(DWORD dwDockStyle);
```

其中参数dwDockStyle指定框架窗口的哪一边可用作工具条的泊位点。该参数可以是下列值的位或组合:CBRS_ALIGN_TOP(允许泊位在客户区的顶部)、CBRS_ALIGN_BOTTOM(底部)、CBRS_ALIGN_LEFT(左边)、CBRS_ALIGN_RIGHT(右边)、CBRS_ALIGN_ANY(任何一边)和CBRS_FLOAT_MULTI(允许一个小框架窗口中有多个浮动工具条)。

2.使泊位对工具条有效

为了使工具条成为可泊位的工具条,还必须使泊位对工具条有效。对每个想要泊位的工具条都要调用CControlBar类的成员函数EnableDocking()来指定工具条可以泊位到父窗口的哪一边上。该函数原型为:

```
void EnableDocking(DWORD dwStyle);
```

其中参数dwStyle指定工具条可以泊位的父窗口的边,它可以是下列值的组合:CBRS_ALIGN_TOP、CBRS_ALIGN_BOTTOM、CBRS_ALIGN_LEFT、CBRS_ALIGN_RIGHT和CBRS_ALIGN_ANY。

该函数指的边必须与框架窗口中允许泊位的某边相吻合,否则工具条不能泊位到框架窗口上,而只能漂浮。

3.泊位和漂浮工具条

完成前两步之后,就可以调用CFrameWnd类的成员函数DockingControlBar()使一个工具条泊位在框架窗口上。该函数的原型为:

```
void DockControlBar(CControlBAr * pBar, UINT nDockBarID = 0, LPCRECT lpRect = NULL);
```

其中参数pBar指向泊位的工具条;nDockBarID指定框架窗口的哪一边用于泊位,该参

数可以是零(任何一边),或是以下值的位或组合:CBRS_ALIGN_TOP、CBRS_ALIGN_BOTTOM、CBRS_ALIGN_BOTTOM、CBRS_ALIGN_LEFT 和 CBRS_ALIGN_RIGHT;lpRect 指定工具条将泊位在目的框架窗口的非客户区中的位置(用屏幕坐标表示)。

如果用户在拖动工具条经过不能用于泊位的位置时释放了鼠标左键便由框架调用 FloatControlBar 函数。

下面是 AppWizard 生成的可泊位工具条的代码:

```
m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
EnableDocking(CBRS_ALIGN_ANY);
DockControlBar(&m_wndToolBar);
```

### 7.1.5 工具提示

工具提示是在把鼠标放在一个工具条的按钮上一小段时间后,出现的含有关于按钮用途的一段简短描述的小型弹出式窗口。为了使工具条有工具提示功能,必须满足以下两个条件:

- 工具条具有工具提示风格。
- 字符串资源中含有工具提示文字。

1.工具条具有工具提示风格

通过在工具条的 Create()函数中指定 CBRS_TOOLTIPS 风格,就可以使工具条具有工具提示功能。在工具条创建函数 Create()的 dwStyle 参数中指定该风格,将 MAINFRM.CPP 文件的 OnCreate()函数中的工具条创建代码改为:

```
if(!m_wndToolBar.Create(this,WS_CHILD|WS_VISIBLE|CBRS_TOP|CBRS_TOOLTIPS)||
    !m_wndToolBar.LoadBitmap(IDR_MAINFRAME)||
    !m_wndToolBar.SetButtons(buttons,sizeof(buttons)/sizeof(UINT)))
{
    TRACE0("Failed to create toolbar\n");
    return -1;          //创建失败
}
```

如果在传递给 CToolBar::Create()的风格参数中还包括 CBRS_FLYBY,便可以在鼠标通过工具条又不实际激活按钮时显示工具提示。除了可以在 Create()函数中设置工具条风格外,还可以使用 CControlBar::SetBarStyle()改变工具条风格。可以按下面代码激活应用程序中的工具提示:

```
m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle()|CBRS_TOOLIPS|CBRS_FLYBY);
```

2.增加工具提示文字

如果工具条按钮与某菜单项相对应,则在创建菜单资源时,可以在菜单属性对话框的提示字符串编辑框(Prompt)中输入工具提示文字。例如给对应于 File|New 菜单项的按钮增加

工具提示 New,如图 7.2 所示。提示字符串可用新行字符('\n')分隔,前面部分字符串为菜单提示,显示的是状态条第一个提示器框内显示的内容,后面部分字符串就是工具提示文字。无论按钮命令是否对应于某个菜单项命令,都可以在字符串资源编辑器中为包含工具条按钮命令的命令行提示的字符串资源添加提示文字。提示文字同样用新行字符'\n'分隔。字符串资源共享工具条按钮的 ID。具体操作步骤为:

1.打开资源文件.RC 中的字符串表。

2.在 ID 栏中寻找对应于工具条按钮 ID 的字符串表,假设某个按钮 ID 为 ID_FILE_NEW,则定位到该 ID,如图 7.2 所示。

3.双击对应的字符串栏,弹出属性对话框,就可以在其中修改按钮对应的字符串文字。只需在该字符串后面添加'\n+工具提示文字',就完成了添加工作。例如将该按钮的字符串"Create a new document"改为"Create a new document\nNew"。运行时,如果将鼠标放在按钮上,则会弹出包含"New"的工具提示小窗口,如图 7.2 所示。

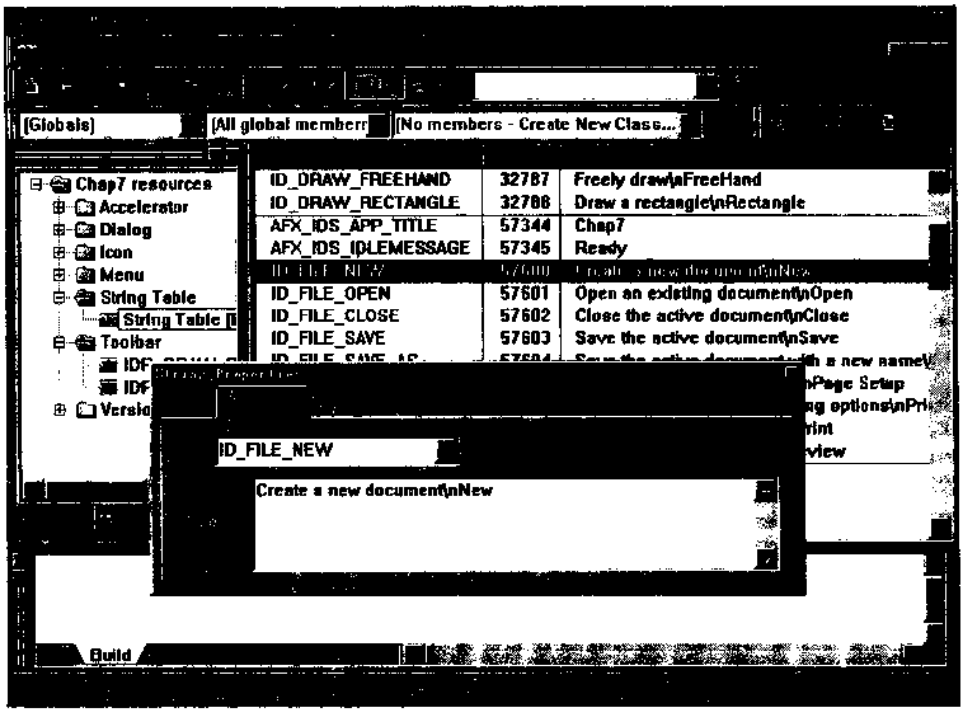



图 7.2 在字符串表中添加工具提示文字

## 7.2 制作普通的工具条

读者可以对缺省的初始工具条进行修改,以创建自己的工具条,也可以重新创建自己的工具条。接下来,我们将创建 Chap7 项目。在该应用程序中创建 IDR_DRAW 工具条。

## 试一试

1.应用 MFC AppWizard(exe) 创建一个新项目 Chap7,选择 Finish 按钮,全部使用缺省选项。创建的类:

```
Application: CChap7App in Chap7.h and Chap7.cpp
Frame: CMainFrame in MainFrm.h and MainFrm.cpp
MDIChildFrame: CChildFrame in ChildFrm.h and ChildFrm.cpp
Document: CChap7Doc in Chap7Doc.h and Chap7Doc.cpp
View: CChap7View in Chap7View.h and Chap7View.cpp
```

特征:

```
+ Initial toolbar in main frame
+ Initial status bar in main frame
+ Printing and Print Preview support in view
+ 3D Controls
+ Uses shared DLL implementation (MFC42.DLL)
```

2.在 Resource View 中添加绘制工具条 IDR_DRAW_BAR 的位图,如图 7.3 所示。

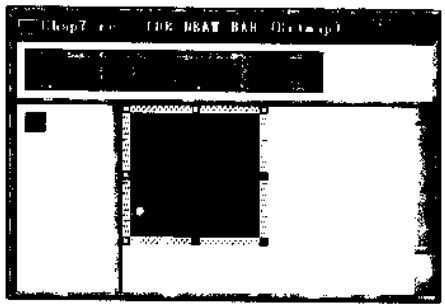


图 7.3 IDR_DRAW_BAR 工具的位图

3.在 ResourceView 中修改菜单 IDR_CHAP7TYPE,如图 7.4 所示。

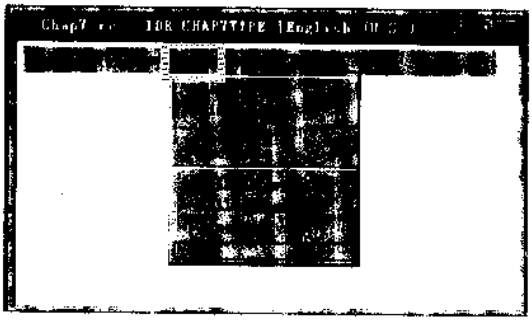


图 7.4 IDR_CHAPTTYPE 菜单

| 菜单 | 菜单项 | 提示 Prompt | 命令 ID |
|---|---|---|---|
| Draw | &Line | Draw a line \ nLine | ID_DRAW_LINE |
| Draw | &Circle | Draw a circle \ nCircle | ID_DRAW_CIRCLE |
| Draw | &Arc | Draw an arc \ nArc | ID_DRAW_ARC |
| Draw | &Rectangle | Draw a rectangle \ nRectangle | ID_DRAW_RECTANGLE |
| Draw | Separator | | |
| Draw | Filled C&ircle | Draw a filled Circle \ nFilled circle | ID_DRAW_FILLED_CIRCLE |
| Draw | R&ectangle | Draw a filled Rectangle \ nFilled Rectangle | ID_DRAW_FILLED_RECTANGLE |
| Draw | &Free Hand | Freely draw \ nFreeHand | ID_DRAW_FREEHAND |

4.打开 resource .h,将其中的 # define ID_BUTTONXXXXX XXXXX 删除。此处可能删除的项如下:

```
# define ID_BUTTON32771          32771
# define ID_BUTTON32772          32772
# define ID_BUTTON32773          32773
# define ID_BUTTON32774          32774
# define ID_BUTTON32775          32775
# define ID_BUTTON32780          32780
# define ID_BUTTON32781          32781
```

在读者程序中的数字可能与这里给出的不同。

5.以文本方式打开 Chap7.rc,将其中的 Toolbar 部分修改如下:(即将 ID_BUTTONXXXXX 改掉)。

```
//////////////////////////////////////////////////////////////////////////////
//
// Toolbar
//

IDR_DRAW_BAR TOOLBAR DISCARDABLE  16, 15
BEGIN
    BUTTON      ID_DRAW_LINE
    BUTTON      ID_DRAW_CIRCLE
    BUTTON      ID_DRAW_ARC
    BUTTON      ID_DRAW_RECTANGLE
    SEPARATOR
    BUTTON      ID_DRAW_FILLED_CIRCLE
    BUTTON      ID_DRAW_FILLED_RECTANGLE
    SEPARATOR
    BUTTON      ID_DRAW_FREEHAND
END
```

6.在 CMainFrame 中增加变量。

| 变量类型 | 变量名称 | 访问权限 |
| --- | --- | --- |
| CToolBar | m_wndDrawBar | protected |

7.改写 CMainFrame::OnCreate 函数。

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CMDIFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    if (! m_wndToolBar.Create(this) ||
        ! m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
    {
        TRACE0("Failed to create toolbar\n");
        return -1;      // fail to create
    }

    if(! m_wndDrawBar.Create(this,WS_CHILD|WS_VISIBLE|CBRS_SIZE_DYNAMIC|
        CBRS_TOP | CBRS_TOOLTIPS | CBRS_FLYBY ,IDR_DRAW_BAR) ||
        ! m_wndDrawBar.LoadBitmap(IDR_DRAW_BAR)||
        ! m_wndDrawBar.SetButtons(DrawButtons,
                                  sizeof(DrawButtons)/sizeof(UINT)))
    {
        TRACE0("Failed to create toolbar\n");
        return -1;      // fail to create
    }

    if (! m_wndStatusBar.Create(this) ||
        ! m_wndStatusBar.SetIndicators(indicators,
          sizeof(indicators)/sizeof(UINT)))
    {
        TRACE0("Failed to create status bar\n");
        return -1;      // fail to create
    }

    // TODO: Remove this if you don't want tool tips or a resizeable toolbar
    m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle() |
        CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC);

    // TODO: Delete these three lines if you don't want the toolbar to
    //  be dockable
    m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
```

```
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndToolBar);

    //使工具条可停靠
    m_wndDrawBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndDrawBar);
    //或者将上面三行换为下面一行
    //m_wndDrawBar.EnableDocking(CBRS_ALIGN_ANY);

    //设置窗口标题
    m_wndDrawBar.SetWindowText(_T("Draw"));
    return 0;
}
```

8.利用 ClassWizard 在 CChap7View 中增加消息处理函数。

| 对象 ID | 消息 | 成员函数 |
|---|---|---|
| ID_DRAW_LINE | COMMAND | OnDrawLine |
| ID_DRAW_CIRCLE | COMMAND | OnDrawCircle |
| ID_DRAW_ARC | COMMAND | OnDrawARC |
| ID_DRAW_RECTANGE | COMMAND | OnDrawRectangle |
| ID_DRAW_FILLED_CIRCLE | COMMAND | OnDrawFilledCircele |
| ID_DRAW_FILLED_RECTANGLE | COMMAND | OnDrawFilledRectangle |
| ID_DRAW_FREEHAND | COMMAND | OnDrawFreehand |

9.编写 CChap7View 中的函数。

这些函数用相应的消息对话框提示用户选择了某个命令。

```
void CChap7View::OnDrawArc()
{
    AfxMessageBox("Draw an arc!");
}

void CChap7View::OnDrawCircle()
{
    AfxMessageBox("Draw a circle!");
}

void CChap7View::OnDrawFilledCircle()
{
    AfxMessageBox("Draw a filled circle!");
}
```

```
void CChap7View::OnDrawFilledRectangle()
{
    AfxMessageBox("Draw a filled rectangle!");
}

void CChap7View::OnDrawFreehand()
{
    AfxMessageBox("Freely draw!");
}

void CChap7View::OnDrawLine()
{
    AfxMessageBox("Draw a line!");
}

void CChap7View::OnDrawRectangle()
{
    AfxMessageBox("Draw a rectangle!");
}
```

10.创建和运行应用程序。

将鼠标放在Draw工具条的某个按钮上,注意工具条按钮上所给出的提示(图7.5)。接着,用鼠标将工具条拖到别处停靠,并试着从工具条中调用命令。

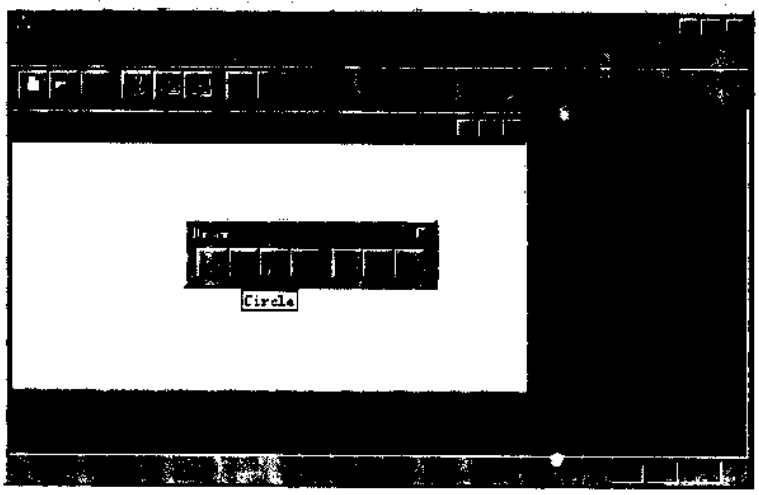



图7.5 普通工具条示例

## 7.3 制作高级工具条

为了操作方便,有时需要在工具条中添加一些除位图按钮以外的控件,譬如下拉列表框。

接下来,我们将对 Chap7 进行修改,以便在其中添加 Property 工具条,在该工具条中有两个列表框,它们分别表示颜色和线型,另外还有两个按钮,如图 7.6 所示。




图 7.6 Property 工具条

### 试一试

1. 打开项目 Chap7。

2. 在字符串表中添加字符串。

| ID 值 | Caption 值 |
|---|---|
| ID_SETDLE | Set System's Status \ nSet System |
| ID_REDRAW | Redraw the screen \ nRedraw |
| IDW_LINETYPE | Choose a linetype \ nLinetype |
| IDW_COLOR | Choose a color \ nColor |

3. 从 ResourceView 中添加工具条 IDR_PROPERTY_BAR。如图 7.7 所示。

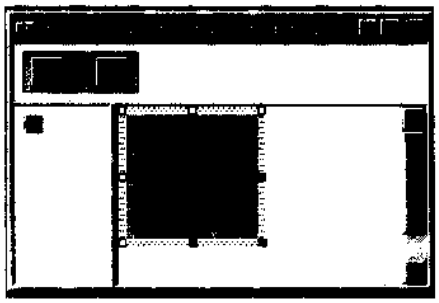


图 7.7 IDR_PROPERTY_BAR 工具条的位图

4. 把 Chap7.rc 只以 Text 方式打开,将 ID_BUTTONXXXXX 修改掉,如下所示:

```
IDR_PROPERTY_BAR TOOLBAR DISCARDABLE 16, 15
BEGIN
    BUTTON      ID_SETDLG
    BUTTON      ID_REDRAW
```

```
END
```

接着,将 resource.h 中的 ID_BUTTONXXXXX 删掉。

5.建立一个新类 CPropertyBar。

利用下列信息建立一个新类。

类名:CPropertyBar

基类:CButton

文件:PropertyBar.CPP

新类建立好后,将 PropertyBar.h 及 PropertyBar.CPP 中的 CButton 全部替换为 CToolBar。

6.在 CpropertyBar 中添加成员变量。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| CColorCmb | m_wndColorCmb | public |
| CLinetypeCmb | m_wndLinetypeCmb | public |

7.添加一个新类 CColorCmb。

利用下列信息建立一个新类。

类名:CColor Cmb

基类:CComboBox

文件:PropertyBar.CPP

8.添加一个新类 CLinetypeCmb。

利用下列信息建立一个新类。

类名:CLinetypeCmb

基类:CComboBox

文件:PropertyBar.CPP

9.在 PropertyBar.h 中,将与 CLinetype、CColortype 类相关的定义移到 CProperty 的定义之前。

10.利用 ClassWizard 在 CLineTypeCmb 中增加虚函数。

```
DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct)
MeasureItem(LPMEASUREITEMSTRUCT lpMeasureItemStruct)
DeleteItem(LPDELETEITEMSTRUCT lpDeleteItemStruct)
```

11.编写 CLineTypeCmb::DrawItem 函数。

```
void CLineTypeCmb::DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct)
{
    CDC * pDC = CDC::FromHandle(lpDrawItemStruct->hDC);
    CLineTypeData * pData = (CLineTypeData * )
                            (lpDrawItemStruct->itemData);
    ASSERT(pData);
    CRect rc(lpDrawItemStruct->rcItem);
```

```
    if(lpDrawItemStruct->itemID==LB_ERR)
        return;

    if(lpDrawItemStruct->itemAction&(ODA_DRAWENTIRE|ODA_SELECT))
        pData->DrawItem(pDC,rc,lpDrawItemStruct->
                        itemState&ODS_SELECTED);
}
```

12. 编写 CLineTypeCmb::MeasureItem 函数。

```
void CLineTypeCmb::MeasureItem(LPMEASUREITEMSTRUCT lpMeasureItemStruct)
{
    lpMeasureItemStruct->itemHeight=18;
}
```

13. 编写 CLineTypeCmb::DeleteItem 函数。

```
void CLineTypeCmb::DeleteItem(LPDELETEITEMSTRUCT lpDeleteItemStruct)
{
    CLineTypeData* pData=(CLineTypeData*)
                        (lpDeleteItemStruct->itemData);
    ASSERT(pData);
    delete pData;
}
```

14. 在 CLineTypeCmb 中增加成员函数。

| 函数类型 | 函数名称 | 访问权限 |
|---|---|---|
| int | AddItem(int nLineType, char* pszName) | public |

15. 编写 CLineTypeCmb::AddItem 函数。

```
int CLineTypeCmb::AddItem(int nLineType , char* pszName)
{
    CLineTypeData* pData=new CLineTypeData(nLineType,pszName);

    int nRet=AddString((LPCSTR)pData);
    if(nRet==LB_ERR)
        delete pData;

    return nRet;
}
```

16. 利用 ClassWizard 在 CLineTypeCmb 中增加消息处理函数。

| 消息名称 | 成员函数 |
| --- | --- |
| CBN_SELCHANGE | OnSelchange |

接着,编写 CLineTypeCmb::OnSelchange 函数。

```
void CLineTypeCmb::OnSelchange()
{
    int nIndex = GetCurSel();
    if(nIndex! = LB_ERR)
    {
        CLineTypeData * pData = (CLineTypeData * ) GetItemDataPtr(nIndex);
        if(pData! = (CLineTypeData * ) - 1)
        {
            char tbuf[40];
            sprintf(tbuf,"%s is selected!",pData - > m_strName);
            AfxMessageBox(tbuf);
        }
    }
}
```

17.利用 ClassWizard 在 CColorCmb 中增加虚函数。

```
DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct)
MeasureItem(LPMEASUREITEMSTRUCT lpMeasureItemStruct)
DeleteItem(LPDELETEITEMSTRUCT lpDeleteItemStruct)
```

18.编写 CColorCmb::DrawItem 函数。

```
void CColorCmb::DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct)
{
    CDC * pDC = CDC::FromHandle(lpDrawItemStruct - > hDC);
    CColorData * pData = (CColorData * ) (lpDrawItemStruct - > itemData);
    ASSERT(pData);
    CRect rc(lpDrawItemStruct - > rcItem);

    if(lpDrawItemStruct - > itemID = = LB_ERR)
        return;

    if(lpDrawItemStruct - > itemAction&(ODA_DRAWENTIRE|ODA_SELECT))
        pData - > DrawItem(pDC,rc,lpDrawItemStruct - >
                           itemState&ODS_SELECTED);
}
```

19.编写 CColorCmb::MeasureItem 函数。

```
void CColorCmb::MeasureItem(LPMEASUREITEMSTRUCT lpMeasureItemStruct)
```

```
    lpMeasureItemStruct->itemHeight = 18;
}
```

20.编写 CColorCmb::DeleteItem 函数。

```
void CColorCmb::DeleteItem(LPDELETEITEMSTRUCT lpDeleteItemStruct)
{
    CColorData* pData = (CColorData*)(lpDeleteItemStruct->itemData);
    ASSERT(pData);
    delete pData;
}
```

21.在 CColorCmb 中增加成员函数。

| 函数类型 | 函数名称 | 访问权限 |
|---|---|---|
| int | AddItem(COLORREF rgbColor, char* pszName) | public |

22.编写 CColorCmb::AddItem 函数。

```
int CColorCmb::AddItem(COLORREF rgbColor , char* pszName)
{
    CColorData* pData = new CColorData(rgbColor,pszName);

    int nRet = AddString((LPCSTR)pData);
    if(nRet == LB_ERR)
        delete pData;

    return nRet;
}
```

23.利用 ClassWizard 在 CColorCmb 中增加消息处理函数。

| 消息名称 | 函数 |
|---|---|
| CBN_SELCHANGE | OnSelchange |

接着,编写 CColorCmb::OnSelchange 函数。

```
void CColorCmb::OnSelcharge()
{
    int nIndex = GetCurSel();
    if(nIndex != LB_ERR)
    {
        CColorData* pData = (CColorData*) GetItemDataPtr(nIndex);
        if(pData != (CColorData*)-1)
        {
```

```
            char tbuf[40];
            sprintf(tbuf,"%s is selected!",pData->m_strName);
            AfxMessageBox(tbuf);
        }
    }
}
```

24.在 PropertyBar.h 文件中 CLineTypeCmb 类的定义之前,添加 CLineTypeData 类。具体内容如下:

```
class CLineTypeData
{
public:
    int         m_nLineType;
    CString m_strName;

public:
    CLineTypeData(int nLineType, char * pLineTypeName)
        :m_nLineType(nLineType) ,m_strName(pLineTypeName)
    {};
    void DrawItem(CDC * pDC,const CRect& rc,BOOL bSelected);
};
```

25.在 PropertyBar.CPP 文件中加入关于 CLineTypeDate 类的定义,具体内容如下:

```
void CLineTypeData::DrawItem(CDC * pDC,const CRect& rc,BOOL bSelected)
{
    CRect rcDraw(rc);

    COLORREF crBackground, crText;
    if(bSelected)
    {
        crBackground = GetSysColor(COLOR_HIGHLIGHT);
        crText = GetSysColor(COLOR_HIGHLIGHTTEXT);
    }
    else
    {
        crBackground = GetSysColor(COLOR_WINDOW);
        crText = GetSysColor(COLOR_WINDOWTEXT);
    }

    pDC->SetBkColor(crBackground);
    ExtTextOut(pDC->GetSafeHdc(),0,0,ETO_OPAQUE,rcDraw,NULL,0,NULL);

    CPen * pNewPen = new CPen;
```

```
    ASSERT( pNewPen->CreatePen( m_nLineType, 1, RGB(0,0,0) ) );
      CPen* pOldPen = pDC->SelectObject( pNewPen );
      pDC->MoveTo(rcDraw.left,(rcDraw.top+rcDraw.bottom)/2);
      pDC->LineTo(rcDraw.left+80,(rcDraw.top+rcDraw.bottom)/2);
      pDC->SelectObject( pOldPen );
      delete pNewPen;
      rcDraw.left+=80+5;

      pDC->SetBkColor(crBackground);
      pDC->SetTextColor(crText);
      pDC->DrawText(m_strName,m_strName.GetLength(),rcDraw,
            DT_SINGLELINE|DT_VCENTER|DT_LEFT);
}
```

26.在 PropertyBar.h 文件中,CColorCmb 类的定义前面添加类 CColorData。具体内容如下:

```
class CColorData{
public:
      COLORREF    m_rgbColor;
      CString          m_strName;

public:
      CColorData(COLORREF rgbColor, char* pColorName)
            :m_rgbColor(rgbColor) ,m_strName(pColorName)
      {};
      void DrawItem(CDC* pDC,const CRect& rc,BOOL bSelected);
};
```

27.在 PropertyBar.CPP 文件中加入关于 CColorData 类的定义,具体内容如下:

```
void CColorData::DrawItem(CDC* pDC,const CRect& rc,BOOL bSelected)
{
      CRect rcDraw(rc);

      COLORREF crBackground, crText;
      if(bSelected)
      {
            crBackground = GetSysColor(COLOR_HIGHLIGHT);
            crText = GetSysColor(COLOR_HIGHLIGHTTEXT);
      }
      else
      {
            crBackground = GetSysColor(COLOR_WINDOW);
            crText = GetSysColor(COLOR_WINDOWTEXT);
```

```
    }

    pDC->SetBkColor(crBackground);
    ExtTextOut(pDC->GetSafeHdc(),0,0,ETO_OPAQUE,rcDraw,NULL,0,NULL);

    CBrush*     pNewBrush=new CBrush;
    ASSERT(pNewBrush->CreateSolidBrush(m_rgbColor));
    CBrush*     pOldBrush = pDC->SelectObject(pNewBrush);
    pDC->Rectangle(rcDraw.left+2,rcDraw.top+2,rcDraw.left+16,
                   rcDraw.top+16);
    pDC->SelectObject(pOldBrush);
    delete pNewBrush;
    rcDraw.left+=16+5;

    pDC->SetBkColor(crBackground);
    pDC->SetTextColor(crText);
    pDC->DrawText(m_strName,m_strName.GetLength(),rcDraw,
        DT_SINGLELINE|DT_VCENTER|DT_LEFT);
}
```

28.在 CMainFrame 中添加变量。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| CPropertyBar | m_wndPropertyBar | protected |

并在 MainFrm.h 的顶端添加：

```
#include "PropertyBar.h"
```

29.在 CMainFrame 中添加成员函数。

| 函数类型 | 函数名称 | 访问权限 |
|---|---|---|
| BOOL | CreatePropertyBar() | protected |

30.编写 CMainFrame::CreatePropertyBar 函数。

```
BOOL CMainFrame::CreatePropertyBar()
{
    const int nDropHeight = 100;

    if(!m_wndPropertyBar.Create(this, WS_CHILD|WS_VISIBLE|CBRS_TOP|
       CBRS_SIZE_DYNAMIC|CBRS_TOOLTIPS|CBRS_FLYBY, IDR_PROPERTY_BAR)||
       !m_wndPropertyBar.LoadBitmap(IDR_PROPERTY_BAR)||
       !m_wndPropertyBar.SetButtons(PropertyButtons,
                          sizeof(PropertyButtons)/sizeof(UINT)))
```

```
}
    TRACE0("Failed to create propertybar \n");
    return FALSE;       // fail to create
}

//把第一个按钮与线型下拉列表框的间隔设置为 12 个像数宽
m_wndPropertyBar.SetButtonInfo(2, ID_SEPARATOR, TBBS_SEPARATOR, 12);
//设置线型下拉列表框的宽度
m_wndPropertyBar.SetButtonInfo(3, IDW_LINETYPE, TBBS_SEPARATOR, 200);
//把线型下拉列表框与颜色下拉列表框的间距设置为 12
m_wndPropertyBar.SetButtonInfo(4, ID_SEPARATOR, TBBS_SEPARATOR, 12);
//设置颜色下拉列表框的宽度
m_wndPropertyBar.SetButtonInfo(5, IDW_COLOR, TBBS_SEPARATOR, 100);

CRect rect;
m_wndPropertyBar.GetItemRect(3, &rect);
rect.bottom = rect.top + nDropHeight;
//创建线型下拉列表框
if (! m_wndPropertyBar.m_wndLineTypeCmb.Create(
          CBS_DROPDOWNLIST|WS_VISIBLE|WS_TABSTOP|
          WS_VSCROLL|CBS_OWNERDRAWVARIABLE,
          rect, &m_wndPropertyBar, IDW_LINETYPE))
{
    TRACE0("Failed to create combo - box \n");
    return FALSE;
}
//填充下拉列表项
m_wndPropertyBar.m_wndLineTypeCmb.AddItem(PS_SOLID,"Solid");
m_wndPropertyBar.m_wndLineTypeCmb.AddItem(PS_DASH,"Dash");
m_wndPropertyBar.m_wndLineTypeCmb.AddItem(PS_DOT,"Dot");
m_wndPropertyBar.m_wndLineTypeCmb.AddItem(PS_DASHDOT,"DashDot");
m_wndPropertyBar.m_wndLineTypeCmb.AddItem(PS_DASHDOTDOT,"DashDotDot");
m_wndPropertyBar.m_wndLineTypeCmb.AddItem(PS_NULL,"Null");
m_wndPropertyBar.m_wndLineTypeCmb.AddItem(PS_INSIDEFRAME,
                                          "InsideFrame");
//预先设定实线线型
m_wndPropertyBar.m_wndLineTypeCmb.SetCurSel(0);

//创建颜色下拉列表框
m_wndPropertyBar.GetItemRect(5, &rect);
rect.bottom = rect.top + nDropHeight;
if (! m_wndPropertyBar.m_wndColorCmb.Create(
          CBS_DROPDOWNLIST|WS_VISIBLE|WS_TABSTOP|
```

```
                WS_VSCROLL|CBS_OWNERDRAWVARIABLE,
                rect, &m_wndPropertyBar, IDW_COLOR))
    {
        TRACE0("Failed to create combo-box\n");
        return FALSE;
    }
    //填充列表项
    m_wndPropertyBar.m_wndColorCmb.AddItem(RGB(255,0,0),"Red");
    m_wndPropertyBar.m_wndColorCmb.AddItem(RGB(0,255,0),"Green");
    m_wndPropertyBar.m_wndColorCmb.AddItem(RGB(0,0,255),"Blue");
    m_wndPropertyBar.m_wndColorCmb.AddItem(RGB(255,255,0),"Yellow");
    m_wndPropertyBar.m_wndColorCmb.AddItem(RGB(0,255,255),"LightBlue");
    m_wndPropertyBar.m_wndColorCmb.AddItem(RGB(255,0,255),"Cyan");
    m_wndPropertyBar.m_wndColorCmb.AddItem(RGB(0,0,0),"Black");
    //预先设定黑色
    m_wndPropertyBar.m_wndColorCmb.SetCurSel(6);

    return TRUE;
}
```

31. 在 MainFrm.CPP 文件中，DrawButtons 定义的后面添加 PropertyButtons 的定义。

```
static UINT PropertyButtons[] =
{
    ID_PROPERTY_QUERY,
    ID_LINETYPE,
    ID_SEPARATOR,
    ID_SEPARATOR,//颜色组合框
    ID_SEPARATOR,
    ID_SEPARATOR,//线型组合框
};
```

32. 修改 CMainFrame::OnCreate 函数。

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CMDIFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    if (! m_wndToolBar.Create(this) ||
        ! m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
    {
        TRACE0("Failed to create toolbar\n");
        return -1;      // fail to create
    }
```

```
if (! m_wndDrawBar.Create(this, WS_CHILD | WS_VISIBLE |
      CBRS_SIZE_DYNAMIC |     CBRS_TOP | CBRS_TOOLTIPS | CBRS_FLYBY ,
      IDR_DRAW_BAR) ||
      ! m_wndDrawBar.LoadBitmap(IDR_DRAW_BAR)||
      ! m_wndDrawBar.SetButtons(DrawButtons,
                                sizeof(DrawButtons)/sizeof(UINT)))
{
      TRACE0("Failed to create toolbar\n");
      return -1;      // fail to create
}

if (! CreatePropertyBar())
      return -1;

if (! m_wndStatusBar.Create(this) ||
      ! m_wndStatusBar.SetIndicators(indicators,
        sizeof(indicators)/sizeof(UINT)))
{
      TRACE0("Failed to create status bar\n");
      return -1;      // fail to create
}

// TODO: Remove this if you don't want tool tips or a resizeable toolbar
m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle() |
      CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC);

// TODO: Delete these three lines if you don't want the toolbar to
//  be dockable
m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
EnableDocking(CBRS_ALIGN_ANY);
DockControlBar(&m_wndToolBar);

//使工具条可泊放
m_wndDrawBar.EnableDocking(CBRS_ALIGN_ANY);
EnableDocking(CBRS_ALIGN_ANY);
DockControlBar(&m_wndDrawBar);

m_wndPropertyBar.EnableDocking(CBRS_ALIGN_TOP|CBRS_ALIGN_BOTTOM);
EnableDocking(CBRS_ALIGN_TOP|CBRS_ALIGN_BOTTOM);
DockControlBar(&m_wndPropertyBar);
```

```
    //设置窗口标题
    m_wndToolBar.SetWindowText(_T("Standard"));
    m_wndDrawBar.SetWindowText(_T("Draw"));
    m_wndPropertyBar.SetWindowText(_T("Property"));

    return 0;
}
```

33. 创建和运行应用程序。

从 Property 工具条的颜色和线型下拉列表框中分别选择颜色和线型(图 7.8)。注意,此时除了下拉列表框中有所响应以外,系统还弹出一个消息对话框,该对话框会提示用户他所作的选择。

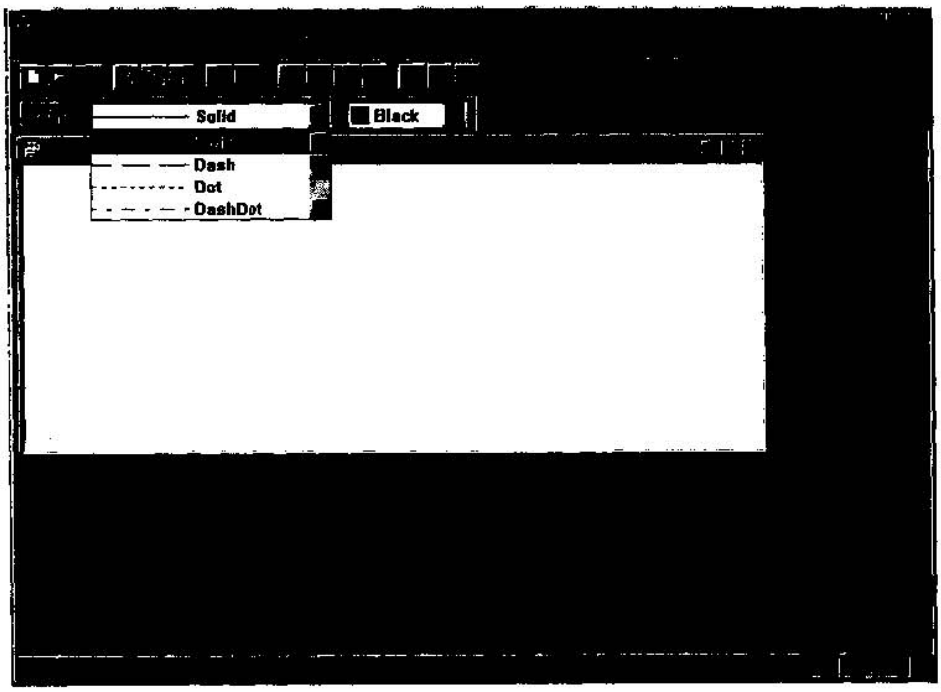



图 7.8 选择线型时的 Property 工具条

## 7.4 本章小结

MFC 提供了一个功能强大的 CToolBar 类。该类用于创建、修改和管理工具条。通常学习本章,读者应当了解工具条的基本组成、创建以及从工具条中调用命令的实现方法,学习了普通工具条和高级工具条的制作过程和处理方式。

# 第八章 状 态 条



- CToolBar 类的介绍
- 状态条的创建和处理方法
- 创建普通的状态条
- 创建高级状态条

迄今为止,读者所创建的所有示例程序都含有状态条。状态条就是含有一行文本输出框或"指示器"的控制条。这些输出一般可分为两类:消息行和状态指示器。消息行用于菜单项命令或工具命令的提示字符串以及其它指示帮助信息,状态指示器用于显示 SCROLL LOCK、NUM LOCK、CAPS LOCK 等开关键的状态以及其它状态。例如绘图软件的状态常显示当前鼠标在客户区中的位置的坐标值。通常状态条中第一个指示器用作消息行,长度是"可伸缩的",即该指示器占据了其它指示器框未用到的状态条长度,其它窗框是右对齐的。状态条一般位于框架窗口的底部。如果在 MFC AppWizard Step 4 of 6 中选择了 Initial Status Bar 选项,则 AppWizard 自动生成缺省的状态条。

## 8.1 概 述

本节较详细地介绍了 CToolBar 类,以及状态条的创建和处理过程。通过学习本节,读者将基本了解状态条的创建和功能实现过程。

### 8.1.1 CStatusBar 类

MFC 的 CStatusBar 类用来实现状态条的功能。CStatusBar 类为 CControlBar 类的派生类,它定义了一个缺省构造函数和一个 Create()成员函数。CStatusBar::Create()函数用于创建一个 Windows 状态条(子窗口),并将其连接到 CStatusBar 对象上,同时设置初始字体和状态条的高度。该函数的原型为:

```
BOOL Create(CWnd* pParentWnd, DWORD dwStyle = WS_CHILD| WS_VISIBLE|
        CBRS_BOTTOM,UINT nID = AFX_IDW_STATUS_BAR);
```

其中参数 pParentWnd 为指向父窗口 CWnd 对象的指针;dwStyle 指定状态条的风格,它除了标准的 Windows 风格外,还可以是 CBRS_TOP(状态条位于框架窗口的顶部)、CBRS_BOTTOM(状态条位于框架窗口的底部)和 CBRS_NOALIGN(当父窗口大小改变时,状态条不

被重定位);nID指定该状态条的子窗口ID。如果函数调用成功,则返回一个非零值,否则返回零。CStatusBar类还提供一系列成员函数来管理状态条的指示器,主要有:

1.SetIndicators函数

函数SetIndicators用于设置指示器的ID,并装入由每个ID指定的字符串资源,然后将指示器的文字设置为该字符串。该函数的原型如下:

```
BOOL CStatusBar:: SetIndicators(const UINT FAR * lpIDArray, int nIDCount);
```

该函数按照数组lpIDArray中的对应元素的值设定标识符ID的值,加载由每个ID所指定字符串资源,并把字符串设置为标识符的文字。如果该函数操作成功,则返回一非0值,否则返回0。

2. SetPaneInfo函数

函数SetPaneInfo()用于设置给定索引值的指示器ID、风格和宽度。该函数的原型如下:

```
void CStatusBar:: SetPaneInfo(int nIndex, UINT nID, UINT nStyle, int cxWidth);
```

该函数给标识符栏设定一个新的ID、风格和宽度。可供选择的标识符风格如下:

· SBPS_NOBORDERS 标题栏无边界。

· SBPS_POPOUT 反转边界,以使文字突出。

· SBPS_DISABCED 不写文字

· SBPS_STRETCH 扩展栏题栏,填上不用的空间,在每一个状态框中只有一个Pane可以使用这种风格。

· SBPS_NORMAL 一般方式。

3.SetPaneText()函数

函数SetPaneText()用于设置给定索引值的指示器的文字。该函数的原型如下:

```
BOOL CStatusBar::SetPaneText(int nIndex, LPCSTR lpszNewText,
                             BOOL bUpdate = TRUE);
```

用lpszNewText所指向的字符串来设置状态栏的文字。

如果函数成功,则返回一非0值,否则返回0。

### 8.1.2 状态条的创建和处理

创建一个状态条可以按照以下三个步骤执行:

(1)构造一个CStatusBar对象。

(2)调用Create()函数创建状态条窗口,并将该窗口连接到该CStatusBar对象上。

(3)调用SetIndicators()函数设置指示器ID。

字符串ID通常存储在一个数组中,函数SetIndicators()就是将这一组ID与对应的指示器联系起来。数组0表示第一个(最左边)指示器,依次类推。在创建状态条之后,就可以使用这些字符串ID或索引值访问任意一个指示器。

AppWizard在生成缺省的状态条时,在主框架窗口类的实现文件MAINFRM.CPP中创建

状态条数组。该数组与下列的类似：

```
static UINT BASED_CODE indicators[] =
{
ID_SEPARATOR,  //status line indicator
ID_INDICATOR_CAPS,
ID_INDICATOR_NUM,
ID_INDICATOR_SCRL,
};
```

其中 ID_SEPARATOR 用于标识消息行，其它三个常量为字符串资源 ID，用于标识状态，如按键状态。如果要增加指示器，则只需给数组中增加相应的 ID。通过增加 ID_SEPARATOR 常量还可以增加分隔符。第一个指示器具有伸缩风格，它将占用剩余空间。在 MAINFRM.CPP 的 OnCreate()函数中创建状态条，代码如下：

```
if (! m_wndStatusBar.Create(this)||
! m_wndStatusBar.SetIndicators(indicators,sizeof(indicators)/sizeof(UINT))
{
TRACE0("Failed to create status bar \n");
return -1;  //创建失败
}
```

下面介绍如何在程序运行时控制消息行的信息显示以及如何在状态指示器中指示特殊键的状态。有三种方法可以更新状态条指示器中的文字：

·调用 SetWindowText()只更新第一个指示器(索引值为)中的文字。

·在状态条的 ON_UPDATE_COMMAND_UI 处理函数中调用 SetText()函数。

·调用 SetPaneText()函数更新任何指示器中的文字。

例如在状态行中显示“Welcome to use the drawing system”，则可以通过下面两行代码完成。

```
CStatusBar * pStatus =(CStatusBar * )
AfxGetApp()->m_pMainWnd->GetDescendantWindow(AFX_IDW_STATUS_BAR);
pStatus->SetPaneText(0,"Welcome to use the drawing system");
```

首先通过调用 AfxGetApp()函数获取指向应用程序的指针，然后通过该指针获取该应用程序的数据成员 m_pMainWnd。CWnd 类的成员函数 GetDescendantWindow()用于获取 ID 为 AFX_IDW_STATUS_BAR 的标准状态条的指针，然后调用 CStatusBar 的成员函数 SetPaneText()对指示器值进行更改。SetPaneText()函数第一个参数为指示器的索引值，第二个参数为指示器将要显示的字符串。

如果想给菜单项添加菜单提示，而菜单提示在状态条的第一个指示器框内显示，只需在菜单编辑器的菜单项属性对话框中的提示编辑框内添加菜单提示文字。系统的自动管理在状态条上的菜单提示文字的显示。

通过调用函数 SetPaneInfo()可以改变指示器的风格和宽度等。例如，为了使第一个指

示器也像其它指示器一样具有立体边框,则只需在 CMainFrame 类的 OnCreate()函数中加入下面一行代码即可(加在状态条的 Create()调用之后)。

```
m_wndStatusBar.SetPaneInfo(0,0,SBPS_STRETCH,0);
```

状态指示器是否显示对应的字符串资源由该指示器 ID 的 ON_UPDATE_COMMAND_UI 更新命令消息处理函数 OnUpdateXXX 决定。和工具条按钮一样,OnUpdateXXX 函数向其以数 pCmdUI 发送 Enable 消息。状态条的更新处理函数在系统空闲阶段被调用,因而状态条在程序生命期中反复被更新。例如状态指示器指示系统键 Alt 的状态:

```
ON_UPDATE_COMMAND_UI(ID_INDICATOR_ALT,OnUpdatekeyAlt)
void CMainFrame::OnUpdatekeyAlt(CCmdUI * pCmdUI)
{
pCmdUI->Enable(::GetKeyState(VK_MENU)<0)
}
```

## 8.2 创建一个普通的状态条

本节的示例旨在说明,如何在状态条中显示系统提示信息。读者将在本节所创建的示例程序的状态条中显示鼠标的当前位置。

### 试一试

1.应用 MFC AppWizard(exe) 创建一个新项目 Chap8。选择 Finish 按钮,全部使用缺省选项。

创建的类:

```
Application: CChap8App in Chap8.h and Chap8.cpp
Frame: CMainFrame in MainFrm.h and MainFrm.cpp
MDIChildFrame: CChildFrame in ChildFrm.h and ChildFrm.cpp
Document: CChap8Doc in Chap8Doc.h and Chap8Doc.cpp
View: CChap8View in Chap8View.h and Chap8View.cpp
```

特征:

```
+ Initial toolbar in main frame
+ Initial status bar in main frame
+ Printing and Print Preview support in view
+ 3D Controls
+ Uses shared DLL implementation (MFC42.DLL)
```

2.在 MainFrm.CPP 文件中修改 indicators 的定义。

```
static UINT indicators[] =
{
```

```
    ID_SEPARATOR,          // status line indicator
    ID_SEPARATOR,
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCRL,
};
```

3. 修改 CMainFrame::OnCreate 函数。

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    .....
    if (! m_wndStatusBar.Create(this) ||
        ! m_wndStatusBar.SetIndicators(indicators,
          sizeof(indicators)/sizeof(UINT)))
    {
        TRACE0("Failed to create status bar\n");
        return -1;      // fail to create
    }

    //设置状态条的第一个窗格的宽度
    m_wndStatusBar.SetPaneInfo(1,ID_SEPARATOR,0,65);
    ......
}
```

4. 利用 ClassWizard 在 CChap8View 中增加消息处理函数。

| 消息 | 函数 |
| --- | --- |
| WM_MOUSEMOVE | OnMouseMove( ) |

5. 编写 CChap8View::OnMouseMove 函数。

```
void CChap8View::OnMouseMove(UINT nFlags, CPoint point)
{
    //获得状态条的指针
    CStatusBar* pStatus = (CStatusBar*)
      AfxGetApp()->m_pMainWnd->GetDescendantWindow(ID_VIEW_STATUS_BAR);

    if(pStatus)
    {
        char tbuf[40];
        sprintf(tbuf,"(%3d,%3d)",point.x,point.y);
        //在状态条的第二个窗格中输出当前鼠标的位置
        pStatus->SetPaneText(1,tbuf);
    }
```

6.创建和运行应用程序。

移动鼠标时,注意状态条中显示的鼠标位置的变化(图8.1)。

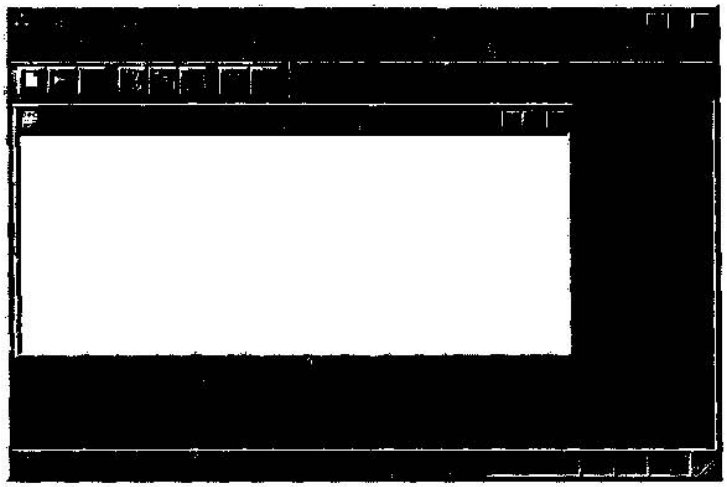


图8.1 在状态条中显示鼠标的当前位置

## 8.3 创建一个高级的状态条

有时候为了操作方便,需要通过状态条来改变应用系统的状态,或者在弹出的菜单时,使第一个窗格最大化,不显示其它窗格的信息。此时,就应当从CStatusBar类派生出读者自己的CMyStatusBar类。

接下来,我们分三步对Chap8项目进行修改,使其可以从状态栏中打开或关闭Snap、Grid、Ortho状态;在弹出菜单时,最大化显示状态栏的第一个窗格;以及在打开或关闭文件时在状态栏中显示进度标记。

第一步:从状态条中改变系统的状态。

### 试一试

1.打开Chap8项目。

2.建立一个新类CMystatusBar。

根据下列信息,利用ClassWizard建立一个新类。

类名:CMystatusBar

基类:CButton

文件:MyStatusBar.cpp

新类建好后,将MyStatusBar.h以及MyStatusBar.cpp文件中的CButton全部替换为

CStatusBar。

3.在 CMyStatusBar 中增加变量。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| BOOL | m_bSnap | protected |
| BOOL | m_bGrid | protected |
| BOOL | m_bOrtho | protected |

4.编写 CMyStatusBar::CMyStatusBar 构造函数。

```
CMyStatusBar::CMyStatusBar()
{
    m_bSnap = TRUE;
    m_bGrid = TRUE;
    m_bOrtho = TRUE;
}
```

5.在 CChap8View 中手工加入消息处理函数。

在 Chap8View.h 中加入:

```
protected::
    afx_msg void OnLButtonDblClk(UINT nFlags, CPoint point)
```

在 Chap8View.cpp 中加入:

```
BEGIN_MESSAGE_MAP(CMyStatusBar, CStatusBar)
    ON_WM_LBUTTONDBLCLK()
    //{{AFX_MSG_MAP(CMyStatusBar)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
////////////////////////////////////////////////////////////////////////////
// CMyStatusBar message handlers
void CMyStatusBar::OnLButtonDblClk(UINT nFlags, CPoint point)
{

}
```

6.编写 CMyStatusBar::OnLButtonDblClk 函数。

```
void CMyStatusBar::OnLButtonDblClk(UINT nFlags, CPoint point)
{
    int         nIsRedraw = 0;
    CRect rect;
    COLORREF crBackground,crText;
    //获得 Snap 矩形框
    GetItemRect(ID_SNAP,rect);
```

```
//判断双击点是否在 Snap 矩形框内
if(rect.PtInRect(point))
{
      m_bSnap = ! m_bSnap;
      //设置文本颜色
      if(m_bSnap)
            crText = RGB(0,0,0);//黑色
      else
            crText = RGB(128,128,128);//灰色
      //标记 Snap 为要重绘更新的矩形框
      nIsRedraw = ID_SNAP;
}

if(nIsRedraw = = FALSE)
{
      GetItemRect(ID_GRID,rect);
      if(rect.PtInRect(point))
      {
            m_bGrid = ! m_bGrid;
            //设置文本颜色
            if(m_bGrid)
                  crText = RGB(0,0,0);//黑色
            else
                  crText = RGB(128,128,128);//灰色
            //标记 Grid 为要重画的矩形框
            nIsRedraw = ID_GRID;
      }
}

if(nIsRedraw = = FALSE)
{
      GetItemRect(ID_ORTHO,rect);
      if(rect.PtInRect(point))
      {
            m_bOrtho = ! m_bOrtho;
            //设置文本颜色
            if(m_bOrtho)
                  crText = RGB(0,0,0);//黑色
            else
                  crText = RGB(128,128,128);//灰色
            //标记 Ortho 为要重绘的矩形框
            nIsRedraw = ID_ORTHO;
      }
```

```
    {

    //检查是否需要重画
    if(nIsRedraw == FALSE)    return;

    CDC *  pDC = GetDC();
    //设置文本和背景颜色
    crBackground = pDC -> GetPixel(rect.left + 1,rect.top + 1);
    pDC -> SetBkColor(crBackground);
    pDC -> SetTextColor(crText);

    rect.left += 3;
    rect.top += 2;
    rect.right -= 3;
    rect.bottom -= 2;
    switch(nIsRedraw)
    {
    case ID_SNAP:
      pDC -> ExtTextOut(rect.left + 3,rect.top,ETO_OPAQUE,rect,"SNAP",4,NULL);
      break;
    case ID_GRID:
      pDC -> ExtTextOut(rect.left + 3,rect.top,ETO_OPAQUE,rect,"GRID",4,NULL);
      break;
    case ID_ORTHO:
      pDC -> ExtTextOut(rect.left + 3,rect.top,ETO_OPAQUE,rect,"ORTHO",5,NULL);
      break;
    default:
      break;
    }
}
```

7. 在 MainFrm. cpp 中将先前对 indicators 的定义删除，并在 MyStatusBar. cpp 的 CMyStatusBar::OnLButtonDblClk 函数定义之前添加：

```
#define ID_STATUS      0
#define ID_SNAP        1
#define ID_GRID        2
#define ID_ORTHO       3

static UINT indicators[] =
{
    ID_SEPARATOR,    //ID_STATUS
    ID_SEPARATOR,    //ID_SNAP
    ID_SEPARATOR,    //ID_GRID
```

```
    ID_SEPARATOR,    //ID_ORTHO
};
```

8.在 CMyStatusBar 中添加成员函数。

| 函数类型 | 函数名称 | 访问权限 |
| --- | --- | --- |
| int | Init(CWnd * pMainWnd) | public |

9.编写 CMyStatusBar::Init 函数。

```
int CMyStatusBar::Init(CWnd * pMainWnd)
{
    if (! Create(pMainWnd) ||
        ! SetIndicators(indicators,
          sizeof(indicators)/sizeof(UINT)))
    {
        TRACE0("Failed to create status bar \n");
        return -1;      // fail to create
    }
    //设置状态条窗格的宽度
    SetPaneInfo(0,ID_SEPARATOR,ID_STATUS,130);
    SetPaneInfo(1,ID_SEPARATOR,ID_SNAP,42);
    SetPaneInfo(2,ID_SEPARATOR,ID_GRID,40);
    SetPaneInfo(3,ID_SEPARATOR,ID_ORTHO,52);

    return 1;
}
```

10.在 MainFrm.h 文件中将对 m_wndstatusBar 的定义改为：

```
protected:
    CMyStatusBar  m_wndStatusBar;
```

并在该文件顶端添加：

```
#include "MyStatusBar.h"
```

11.编写 CMainFrame::OnCreate 函数。

将 CMainFrame::OnCreate 函数中的下列语句：

```
    if (! Create(pMainWnd) ||
        ! SetIndicators(indicators,
          sizeof(indicators)/sizeof(UINT)))
    {
        TRACE0("Failed to create status bar \n");
        return -1;      // fail to create
    }
```

修改为：

```
if(! m_wndStatusBar.Init(this))
    return -1;
```

12. 修改 CChap8View::OnMouseMove 函数。

```
void CChap8View::OnMouseMove(UINT nFlags, CPoint point)
{
    //获得状态条的指针
    CStatusBar* pStatus=(CStatusBar*)
      AfxGetApp()->m_pMainWnd->GetDescendantWindow(ID_VIEW_STATUS_
BAR);

    if(pStatus)
    {
        char tbuf[40];
        sprintf(tbuf,"(%3d,%3d)",point.x,point.y);
        //在状态条的第一个窗格中输出当前鼠标的位置
        pStatus->SetPaneText(0,tbuf);
    }
}
```

13.创建和运行应用程序。

双击状态条的第二、三、四个窗格，注意其中文字的变化(图 8.2)。

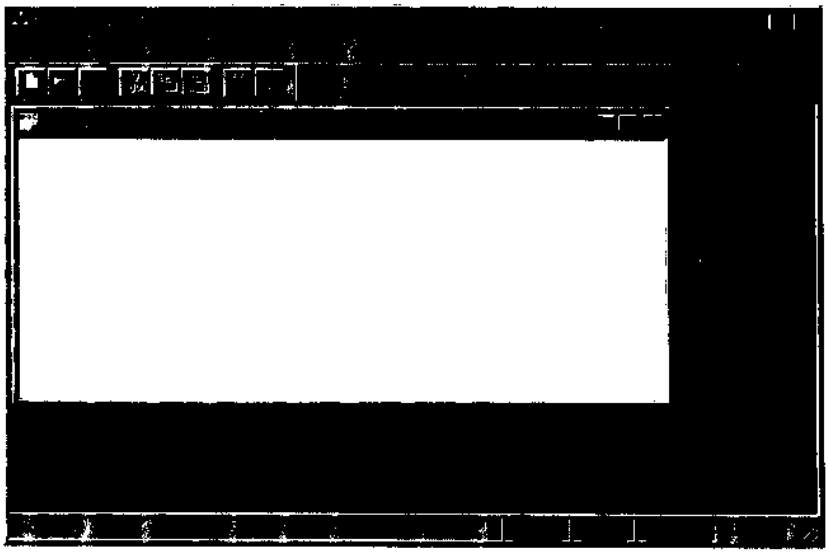


图 8.2 在状态条中用双击改变条该的状态

第二步:从菜单中选择菜单项时,状态条的提示窗格最大化。

在这里,我们将修改程序,以便用户在从菜单中选择菜单项时,状态条的提示窗格最大化,占满整个状态条。

## 试一试

1.打开 Chap8 项目。

2.在 CMainFrame 中增加变量。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| int | m_cxPaneWidth | protected |
| BOOL | m_bMenuSelected | protected |
| UINT | m_nPaneStyle | protected |
| UINT | m_nPaneID | protected |

3.利用 ClassWizard 在 CMainFrame 中增加消息处理函数。

| 消息 | 函数 |
|---|---|
| WM_INITMENUPOPUP | OnInitMenuPopup |
| WM_MENUSELCT | OnInit Menuselect |

4.编写 CMainFrame::CMainFrame 构造函数。

```
CMainFrame::CMainFrame()
{
    m_cxPaneWidth = -1;
    m_bMenuSelected = FALSE;
}
```

5.编写 CMainFrame::OnInitMenuPopup 函数。

```
void CMainFrame::OnInitMenuPopup(CMenu * pPopupMenu, UINT nIndex,
                                 BOOL bSysMenu)
{
    CMDIFrameWnd::OnInitMenuPopup(pPopupMenu, nIndex, bSysMenu);

    if(m_cxPaneWidth == -1&&m_bMenuSelected)
    {
        m_wndStatusBar.GetPaneInfo(0,m_nPaneID,m_nPaneStyle,
                                   m_cxPaneWidth);
        //将状态条的第一个窗格的宽度设置得足够大,以便覆盖整个状态条
        m_wndStatusBar.SetPaneInfo(0,m_nPaneID,SBPS_NOBORDERS|
                                   SBPS_STRETCH,10000);
    }
```

```
}
```

6. 编写 CMainFrame:: OnMenuSelect 函数。

```
void CMainFrame:: OnMenuSelect( UINT nItemID, UINT nFlags, HMENU hSysMenu )
{
    CFrameWnd:: OnMenuSelect(nItemID,nFlags,hSysMenu);

    if(nFlags = = 0xFFFF&&hSysMenu = = 0&&m_cxPaneWidth! = -1)
    {
        m_bMenuSelected = FALSE;
        m_wndStatusBar.SetPaneInfo(0,m_nPaneID,m_nPaneStyle,
                                   m_cxPaneWidth);
        m_cxPaneWidth = -1;
    }
    else
    {
        m_bMenuSelected = TRUE;
    }
}
```

7. 创建和运行应用程序。

打开菜单,观察状态条中第一个窗格的变化。图 8.3 是从 File 菜单中选择 Open 菜单项时,状态条的第一格最大化的状态。

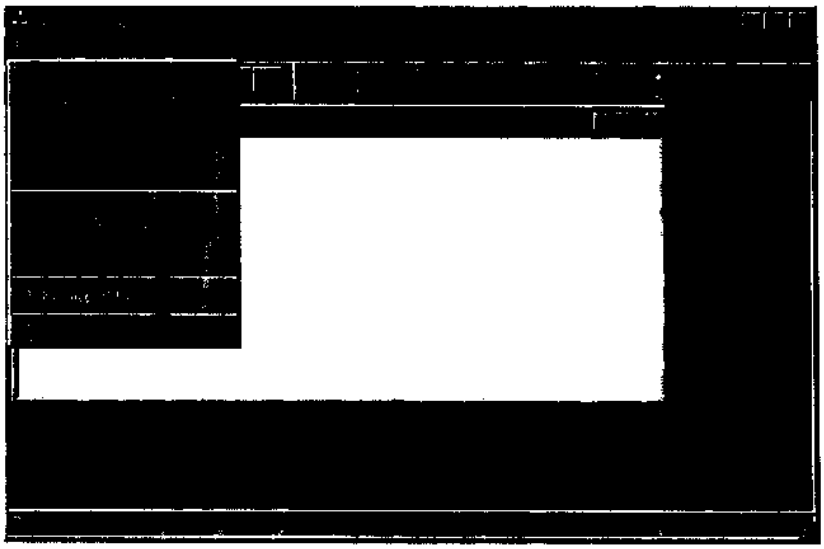


图 8.3 选择菜单项时,状态条的第一个窗格最大化

第三步:在打开、关闭文件时显示进度指示器。

有时候，在进行一些耗时较长的操作时，我们需要用进度指示器向用户提示该操作的进展情况，以免用户认为系统死机，没有响应。接下来，我们将在 Chap8 项目加入进度指示器。

## 试一试

1. 打开 Chap8 项目。

2. 在 CMyStatusBar 中增加变量。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| CProgressCtrl | m_progress | public |

3. 利用 ClassWizard 在 CMyStatusBar 中增加消息处理函数

| 消息 | 函数 |
|---|---|
| WM_TIMER | OnTimer |
| WM_CREATE | OnCreate |

4. 编写 CMyStatusBar::OnTimer 函数。

```
void CMyStatusBar::OnTimer(UINT nIDEvent)
{
    static int nCount = 0;
    m_progress.StepIt();

    if( ++nCount == 10)
    {
        nCount = 0;
        m_progress.SetPos(0);

        KillTimer(1);
        m_progress.MoveWindow(CRect(0,0,0,0));
    }
}
```

5. 编写 CMyStatusBar::OnCreate 函数。

```
int CMyStatusBar::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CStatusBar::OnCreate(lpCreateStruct) == -1)
        return -1;
    //创建进度条，并设置进度条的初始状态。
    m_progress.Create(WS_CHILD|WS_VISIBLE,CRect(0,0,0,0),this,100);
    m_progress.SetRange(0,10);
    m_progress.SetPos(0);
    m_progress.SetStep(1);
```

```
    return 0;
}
```

6.利用 ClassWizard 在 CChap8Doc 中增加消息处理函数。

| 对象 ID | 消息 | 成员函数 |
|---|---|---|
| ID_FILE_OPEN | COMMAND | OnFileOpen |
| ID_FILE_CLOSE | COMMAND | OnFileClose |

7. 编写 CChap8Doc::OnFileOpen 函数。

```
void CChap8Doc::OnFileOpen()
{
    //获得状态条指针
    CMyStatusBar* pStatusBar=(CMyStatusBar *)
        AfxGetMainWnd()->GetDescendantWindow(AFX_IDW_STATUS_BAR);

    pStatusBar->SetPaneText(0,"Open a file");
    CRect rect;
    pStatusBar->GetItemRect(1,rect);
    rect.right=rect.left+180;
    //在状态条中显示进度条。
    pStatusBar->m_progress.MoveWindow(rect);
    //设置计时器,显示过程进度。
    pStatusBar->SetTimer(1,500,NULL);
}
```

8. 编写 CChap8Doc::OnFileClose 函数。

```
void CChap8Doc::OnFileClose()
{
    CMyStatusBar* pStatusBar=(CMyStatusBar *)
        AfxGetMainWnd()->GetDescendantWindow(AFX_IDW_STATUS_BAR);

    pStatusBar->SetPaneText(0,"Close a file");
    CRect rect;
    pStatusBar->GetItemRect(1,rect);
    rect.right=rect.left+180;
    pStatusBar->m_progress.MoveWindow(rect);

    pStatusBar->SetTimer(1,500,NULL);
}
```

9.在 Chap8Doc.cpp 的顶端添加:

```
#include "Chap8Doc.h"
#include "MyStatusBar.h"
```

10. 创建和运行应用程序。

选择 File \ Open、File \ Close 菜单项，注意状态条中的进度指示器(图 8.4)。

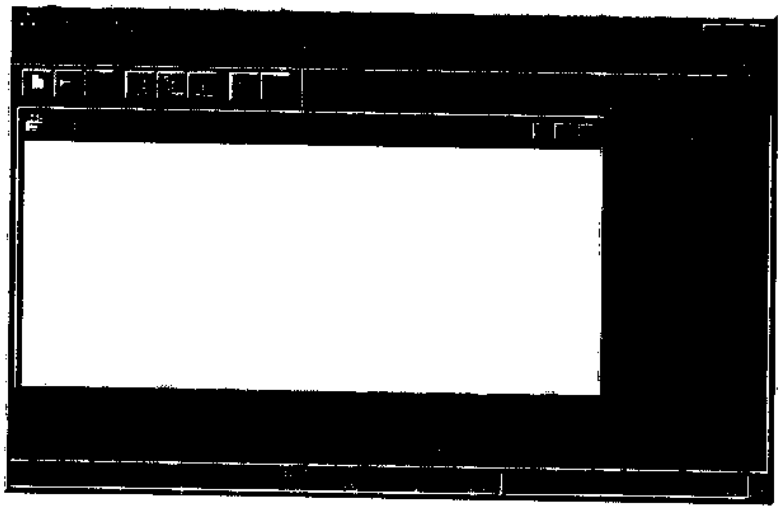


图 8.4 状态条中的进度指示器

## 8.4 本章小结

MFC 提供了一个功能强大的 CStatusBar 类。该类用于创建和管理状态条。通过学习本章，读者应当了解 CToolBar 的功能及其用法，学会状态条的创建和处理方法。通过实践本章所给的普通状态条和高级状态条的编制过程，读者应当掌握了状态条的编制。

# 第九章 对 话 框

**本章要点**

- 对话框和 CDialog 类
- 对话框的创建、显示以及数据交换机制
- 系统的预制对话框
- 无模式对话框的制作方法
- 模式对话框的制作方法
- 可停靠对话框控件
- 属性对话框控件
- 基于对话框的应用程序

对话框是 Windows 系统中进行数据交互的重要手段。程序通过对话框获取用户输入，用户通过消息框等对话框获得程序执行情况。有时对话框还可以直接作为应用程序的主界面，比如基于对话框的应用程序。本章将讲述对话框、对话框条以及基于对话框的应用程序。严格地讲，对话框条、基于对话框的应用程序不属于对话框，但由于它们用到了对话框资源，故而在本章一起介绍。

学习 Windows 编程，首先就要掌握各种控件的用法和对话框的使用技巧。下面就从对话框的概述开始，逐步介绍各种对话框的创建方法。

## 9.1 概 述

本节将讲述对话框的分类、对话框的创建和显示、CDialog 类以及 DDX/DDV 实现基理。通过学习本节，读者将初步了解对话框，为编写对话框进行交互打下基础。

### 9.1.1 对话框的种类

对话框分为模式对话框和无模式对话框两大类。

模式对话框就是在用户关闭它之后才能返回到父窗口的对话框。在未关闭对话框以前，程序不能继续执行。消息框便是最简单的模式对话框。模式对话框经常用于交换一些重要信息，强制用户完成所要求的动作。模式对话框不能具有 WS_CHILD 风格，因为禁止父窗口的同时也禁止了属于父窗口的所有子窗口。而无模式对话框不禁止父窗口，即无论无模式对话框是否关闭，应用程序都可以继续执行。

MFC 提供了 CDialog 对话框类，它同时支持这两种对话框。

为了使不同的应用程序具有类似的用户接口，Windows 提供了一组标准对话框。与之对应的，MFC 提供了相应的类来支持标准对话框。这些标准的对话框类主要有：CFileDialog 类——用于文件选择（打开或保存文件）对话框、CFontDialog 类——用于字体选择对话框、CColorDialog 类——用于颜色选择对话框、CPrintDialog 类——用于打印机设置对话框、CFindReplaceDialog 类——用于字符串查找与替换对话框。另外，MFC 还提供了一些用于 OLE 的标准对话框类。

### 9.1.2 对话框的创建和显示

创建对话框时，应当首先使用 AppStudio 对话框编辑器创建对话框模板资源。该模板指定对话框的有关属性，如对话框的 ID、大小、位置、风格以及各种控件的类型和位置等。其次要创建一个对话框类，用于管理该对话框，并提供程序接口。

创建对话框的一般过程可具体分为四个步骤：

(1)使用对话框编辑器设计对话框的外观布局，创建对话框模板资源。

(2)使用 ClassWizard 创建新的对话框类。

(3)使用 ClassWizard 进行消息映射，即将对话框资源的控件与对话框类中的消息处理函数联系起来。

(4)使用 ClassWizard 向对话框类中加入与控件相联的数据成员，并为控件指定对话数据交换和对话数据验证(DDX/DDV)。

除了第 2 步只执行一次外，其它三步经常需要反复执行，以达到设计要求。

下面具体介绍如何创建对话框类以及如何处理各种消息。

创建对话框对象分为两步：首先构造对话框对象，然后再创建对话框窗口。模式对话框和无模式对话框的创建和显示的过程是不同的，具体创建过程如下：。

1.模式对话框的创建和显示

创建和显示模式对话框很简单，只需要构造一个 CDialog 对象，然后调用 DoModal()成员函数就可以显示模式对话框。DoModal()函数管理与对话框的交互，直到用户单击 OK 或 Cancel 按钮为止。当 DoModal()函数返回时，对话框窗口自动关闭。在 DoModal()返回 IDOK 之后，但在对话框对象销毁之前，任何时候都可以检取用户输入的任何数据。

2.无模式对话框的创建和显示

创建和管理无模式对话框比模式对话框复杂得多，因为与模式对话框有关的操作都由系统自动管理，而无模式对话框的创建、显示和管理都要由程序员完成。创建无模式对话框也分为两步：首先构造一个 CDialog 对象，然后调用 CDialog::Create()成员函数来创建对话框窗口。最后使用 CWnd::DestroyWindow()函数来关闭无模式对话框。

如果想要在父窗口创建时显示对话框，则必须在对话框模板中使用 WS_VISIBLE 风格。在程序运行时可以通过调用 ShowWindow()函数来控制对话框的显示与否。

在检取用户输入数据之前，首先要调用 CWnd::UpdateData(TRUE)函数，然后，再使用对话框类的成员变量检取数据。

### 9.1.3 CDialog 对话框类

CDialog 类是所有对话框类的基类。它定义了一个构造函数和一个 Create()成员函数来创建对话框。构造函数根据对话框模板 ID(通常有一个 IDD 前缀,如 IDD_DIALOG1)来访问对话框资源。该函数经常用于构造一个基于资源的模式对话框。而 Create()函数则使用对话框模板,由资源创建无模式对话框。

CDialog 还定义了许多管理对话框的成员函数。其中最重要的有以下四个:

(1)函数 DoModal()用于激活模式对话框,完成后返回对话框结果。它在对话框活动期间与用户的所有窗口交互。在此对话框关闭之前,用户不能与应用程序的其它窗口交互。DoModal()函数的原型说明如下:

```
virtual int CDialog::DoModal( );
```

该函数不带输入参数。

当用户用鼠标单击对话框中的一个按钮,比如 OK 或 Cancel 时,消息处理函数如 OnOk()或 OnCancel(),将被调用以关闭对话框。缺省情况下,OnOk()成员函数将使对话框数据有效并进行修改,然后使用 IDOK 结构关闭对话框;而在缺省情况下,OnCancel()成员函数不能使数据有效也不进行修改,使用 IDCANCEL 结构来关闭对话框。程序员可以重载消息处理函数以改变性能。

函数返回一个 int 型数值,它指定了传递给 CDialog::EndDialog()成员函数的 nResult 参数的数值,用于关闭对话框。当函数无法建立对话框时返回 -1;发生错误时返回 IDABORT。

(2)函数 OnInitDialog()是一个虚函数,用于响应 WM_INITDIALOG 消息。在 Create()或 DoModal()调用期间,系统会发送 WM_INITDIALOG 消息给对话框,在对话框显示之前执行该函数。该函数经常被重载,以完成与应用程序有关的特殊的初始化操作。

OnInitDialog()函数的原型说明如下:

```
virtual BOOL CDialog::OnInitDialog( );
```

该函数不带输入参数。

在初始化对话框时,若程序员需要执行特殊的处理,则可以重载该成员函数。在重载版本中,首先调用基类函数 OnInitDialog(),但不管它的返回值。重载的成员函数通常返回 TRUE。

Windows 通过适用于所有 Microsoft Foundation Class Library 对话框标准的全局对话框过程来调用 OnInitDialog()函数,而不是通过消息映射。因此程序员不需要为该成员函数提供消息映射项。

该函数指定了应用程序是否把输入焦点设置到对话框的某个控件。若返回值为非零,Windows 则把输入焦点设置到对话框的第一个控件中。只有在应用程序显式地输入焦点设置到对话框的某个控件时才返回零。

(3)函数 OnOK()也是虚函数。在用户单击 OK 按钮(ID 为 IDOK)时调用该函数。如果

对话框含有对话数据交换和验证,则该函数缺省地实现验证对话框数据,并更新相应变量。对于无模式对话框,如果应用程序要实现 OK 按钮功能,则必须重载该函数,并在其中调用 DestroyWindow()函数来删除对话框。

OnOk()函数的原型说明如下:

```
protected virtual void CDialog::OnOk( );
```

该函数不带任何输入参数。

重载该成员函数以完成 OK 按钮的功能。若对话框中包括了自动的数据验证和交换,那么在缺省情况下执行本成员函数将验证对话框数据并修改应用程序中的适当变量。

若程序员在无模式对话框中完成了 OK 按钮,则必须重载 OnOK()成员函数并从中调用 DestroyWindows()。千万不要调用基类中的该成员函数,这是因为基类中的该函数将调用 EndDialog(),它将致使对话框不可见但并不废除它。

(4)函数 OnCancel()在用户单击 Cancel 按钮或按 Esc 键时被调用。对于模式对话框,单击 Cancel 按钮将自动终止模式对话框,DoModal()返回 IDCANCEL。对于无模式对话框,如果应用程序要实现 Cancel 按钮功能,则必须重载该函数,并在其中调用 DestroyWindows()函数以删除对话框。

OnCancel()函数的原型说明如下:

```
protected virtual void CDialog::OnCancel( )
```

该函数不带任何输入参数。重载本函数用于完成 Cancel 按钮的功能。缺省情况下,将通过调用 EndDialog(),并使 DoModal()返回 IDCANCEL 来终止模式对话框。

若程序员需要在无模式对话框中实现 Cancel 按钮,则可以重载 OnCancel()成员函数并从中调用 DestroyWindow()。不要调用基类成员函数,这是因为基类成员函数将调用 EndDialog(),它将导致对话框不可见但并不废除它。

后面三个虚函数通常被重载,重载目的如表 9.1 所示。

**表 9.1 CDialog 类中通常被重载的成员函数**

| 成员函数 | 所响应的消息 | 重载目的 |
|---|---|---|
| OnInitDialog() | WM_INITDIALOG | 初始化对话框的控件 |
| OnOK() | IDOK 按钮的 BN_CLICKED | 用户单击 OK 按钮时响应 |
| OnCancel() | IDCANCEL 按钮的 BN_CLICKED | 用户单击 Cancel 按钮时响应 |

除了 CDialog 类的成员函数管理对话框之外,CWnd 和 CWinApp 类也提供一些成员函数用于管理对话框。常用的函数如下所示:

CWnd::UpdateData()函数用于初始化对话框中的数据或检索有效的对话框数据。该函数只有一个参数,当参数为 FALSE 时用于初始化对话框;当参数为 TRUE(缺省值)时用于检索数据。应用程序通常调用该函数以取得对话框的各控件值。

CWnd::DoDataExchange()函数用于交换和验证对话框数据。该函数不能被直接调用。UpdateData()函数调用该函数以初始化对话框控件或检索数据。该函数可由 ClassWizard 来

维护。

DoDataExchange()函数的原型说明如下：

```
protected virtual void CWnd::DoDataExchange(CDataExchange * pDX)
```

框架调用此函数交换对话框数据，并使之有效。不要直接调用这个函数，它由UpdateData()成员函数调用。UpdateData()成员函数初始化对话框的控件并从对话框中检索数据。当用户从CDialog基类派生一个特定应用程序的对话框类时，如果希望使用框架的自动数据交换和检验，就应当重载此成员函数。ClassWizard将写一个此成员函数的重载类型，以便用户程序能够保持所需的对话数据交换(DDX)和检验(DDV)全局函数调用的数据映射。

为了自动地生成此成员函数的重载形式，首先用AppStudio建立一对话框资源，然后派生一个应用程序特定的对话框类。再引入ClassWizard，使用它链接变量、数据，以及新的对话框中各种控制的有效作用范围。之后，ClassWizard再写出重载的DoDataExchange()成员函数，它包含有数据映射。下面是一个由Class Wizard生成的DDX/DDV代码块的例子。

```
void CPenWidthsDlg::DoDataExchange(CDataExchange * pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CPenWidthsDlg)
    DDX_Text(pDX, IDC_THIN_PEN_WIDTH, m_nThinWidth);
    DDV_MinMaxInt(pDX, m_nThinWidth, 1, 20);
    DDX_Text(pDX, IDC_THICK_PEN_WIDTH, m_nThickWidth);
    DDV_MinMaxInt(pDX, m_nThickWidth, 1, 20);
    //}} AFX_DATA_MAP
}
```

ClassWizard将维护在定界符//{{和//}}内的代码，用户不要修改它们。DoDataExchange()重载成员函数必须位于源文件的宏语句之前。

CWinApp::SetDialogBkColor()函数用于设置对话框的背景颜色。所设置的颜色适用于所有的对话框和消息框。该函数通常在应用类的InitInstance()覆盖函数中被调用。

SetDialogBKColor()函数的原型说明如下：

```
protected void CWinApp::SetDialogBkColor (COLORREF drCtlBk = RGB (191, 192,
                192), COLORREF clrltlText = RGB(0,0,0));
```

在InitInstance()成员函数内调用此成员函数设置对话框和消息框的背景及文字的颜色。

### 9.1.4 对话数据交换/对话数据验证

对话数据交换(DDX)用于初始化对话框的控件和获取用户输入的数据。对话数据验证(DDV)用于对对话框中的数据输入进行验证。为了使对话框的数据成员能够使用DDX/DDV，则必须使用ClassWizard创建成员变量，设置数据类型，并指定验证规则。ClassWizard

通过一套特殊的语法格式来自动维护 DDX/DDV。

假设某对话框包含两个列表框控件 IDC_FILELIST、IDC_DIRLIST 和一个编辑控件 IDC_FILENAME,实现类为 CDirDlg。利用 DDX,可以把对话框类中定义的成员变量和对话框中的控件联系起来。每个控件都对应有一个控件(control)变量和一个值(value)变量。打开 MFC ClassWizard 对话框,选择 Member Variables 标签,再选择要设置 DDX 的控件,接着按 Add Variable...按钮,加入变量。我们加入列表控件的控件变量 m_FileList 和 m_DirList,编辑控件的控件变量 m_EditFilename 和值变量 m_strFilename。

在变量类型为 CString 和 Numeric(int,UINT,long,DWORD,float,double)时,ClassWizard 支持 DDV。类型为 CString 的变量可以设置变量的最大长度,类型为 Numeric 的变量可以设置变量的最大值和最小值。DDV 自动验证数据是否满足这些限制条件。我们给编辑控件的值变量 m_strFilename 加入限制条件:最大字符串长度为 256 字节。

如果使用 DDX 机制,则一般在 OnInitDialog()处理函数或对话框构造函数中设置成员变量的初始值。当一个对话框显示之后,所有初始值被传递到控件中。加入成员变量后,单击 OK 按钮关闭 ClassWizard 对话框,那么 ClassWizard 自动地在构造函数中加入控件的初始化值。例如,ClassWizard 自动地将变量 m_strFilename 初始化为""。

```
CDirDlg::CDirDlg(CWnd * pParent/ * = NULL * /):CDialog(CDirDlg::IDD,pParent)
{
  //{{AFX_DATA_INIT(CDirDlg)
  m_strFilename = T("");
  //}}AFX_DATA_INIT
}
```

有关数据的 DDX/DDV 在对话框类的 DoDataExchange()函数中实现。MFC 为不同类型的交换提供不同的 DDX 函数,为要求 DDV 的变量提供 DDV 函数。例如,ClassWizard 自动为以上变量加入以下 DDX/DDV 函数:

```
void CDirDlg::DoDataExchange(CDataExchange * pDX)
{
  CDialog::DoDataExchange(pDX);
  //{{AFX_DATA_MAP(CDirDlg)
  DDX_Control(pDX,IDC_FILENAME,m_EditFilename);
  DDX_Control(pDX,IDC_FILELIST,m_FileList);
  DDX_Control(pDX,IDC_DIRLIST,m_DirList);
  DDX_Text(pDX,IDC_FILENAME,m_strFilename);
  DDV_MaxChars(pDX,m_strFilename,256);
  //}}AFX_DATA_MAP
}
```

其中//{{AFX_DATA_MAP 和//}}AFX_DATA_MAP 定界符之间的 DDX_和 DDV_行就是一个"数据映射"。特定控件的 DDV 函数调用必须紧接在同一控件的 DDX 函数后面。

### 9.1.5 Windows 标准控件

Windows 系统提供了许多标准控件窗口，包括静态控件、按钮、编辑控件、列表框和组合框等。与此相对应，MFC 提供了一组控件类来实现各种控件的功能。表 9.2 列出了这些类及其对应的标准控件。

表 9.2 标准控件窗口类

| 类 | Windows 控件 |
|---|---|
| CStatic | 静态控件 |
| CButton | 按钮控件：命令按钮、复选框、单选按钮或成组框控件 |
| CEdit | 编辑控件 |
| CListBox | 列表框控件 |
| CComboBox | 组合框控件 |
| CScrollBar | 滚动条控件 |

每个控件类都封装了一个 Windows 控件。使用控件类成员函数可以获取和设置控件的值或状态，并响应由控件发送到其父窗口的各种标准消息。通常在对话框中创建控件对象。另外，也可以在其它窗口中创建控件对象。

除了标准的 Windows 控件外，MFC 还提供了控制条类和切分窗口类等附加控件类。其中控制条类包括工具条类(对应类为 CToolBar 类)、状态条类(对应类为 CStatusBar 类)和对话条类(对应类为 CDialogBar 类)。MFC 用 CSplitterWnd 类实现切分窗口。

#### 9.1.5.1 静态控件

静态控件是包含文本或图形的子窗口。Windows 提供了四种类型的静态控件：矩形、边框、文本和图标。静态控件通常用于标记其它控件，或者是生成框和线，以及用来把一组控件同其它控件相分隔。静态控件既不能接收用户输入，也不能被选择。

CStatic 类实现了静态控件。静态控件通常显示固定的文字，使用基类 CWnd 的成员函数 GetWindowText()和 SetWindowText()来获取和设置与静态控件相关的正文。

#### 9.1.5.2 编辑控件

编辑控件是一个矩形子窗口，用户可以在其中输入并编辑文字。编辑控件可以有多种特征，如多行编辑、滚动等。

MFC 提供 CEdit 类来实现编辑控件功能。和静态控件一样，它也使用基类 CWnd 的成员函数 GetWindowText()和 SetWindowText()来获取和设置 CEdit 对象的内容。最简单的方法就是利用 ClassWizard 增加 CEdit 类型的成员变量，然后利用 CWnd 的成员函数 UpdateData()更新变量值。

编辑控件可以向父窗口发送八种通知消息，消息的前缀名为 EN_，常用的有 EN_CHANGE(编辑控件中文本被改变)和 EN_KILLFOCUS(编辑控件失去输入焦点)。程序员可以使用 ClassWizard 向父窗口类中加入相应的消息映射。这里的消息映射项的格式为：

ON_通知消息(id,memberFxn)

其中 id 指定发送通知消息的控制的子窗口 ID,memberFxn 是用来处理消息的函数名称。消息处理函数的原型为:

afx_msg void memberFxn()

9.1.5.3 按钮控件

Windows 95 提供了五种按钮:命令按钮、复选框、单选按钮、成组框和自绘制按钮。MFC 提供类 CButton 来实现按钮控件功能。应用程序既可以从对话框模板创建按钮控件,也可以手工编写代码直接创建。从对话框模板创建时,只需向对话框类加入 CButton 类型的成员变量即可。对于手工创建,则首先要调用 CButton 构造对象,然后再调用成员函数 Create()来创建按钮控件,并将它连接到 CButton 对象上。

当用户选择按钮时,会产生两种消息:BN_CLICKED(单击按钮)和 BN_DOUBLECLICKED(双击按钮)。程序员可以通过使用 ClassWizard 向父窗口类(通常是一个 CDialog 派生类)加入消息映射,其格式与编辑控件的消息映射相同。

1. 命令按钮

几乎所有的对话框中都包含命令按钮。应用程序通常使用命令按钮启动某一操作。

2. 复选框

应用程序通常在成组框中显示复选框,用户可以在一组相关但又独立的选项中进行选择,在同一时间可以选择任意多个。它通常有两种状态:选中(框中有个 X)和未选中(没有 X)。另外,如果为三态复选框,则还有禁止状态(灰框)。

3. 单选按钮

应用程序通常在成组框中显示单选按钮,用户可以在一组相关但又对立的选项中进行选择,在同一时间只能选择其中一个。类似于复选框,它通常也有两种状态:选中(按钮上有个黑点)和未选中(没有黑点)。

4. 成组框

成组框是一个矩形框,其中是一组控件,如复选框或单选按钮。它用于把具有公共用途的控件组织在一起。成组框不能被选中,应用程序也不能向它发送消息。

5. 自绘制按钮

最典型的自绘制按钮就是位图按钮,它是用位图图像代替文字标签的按钮控件。位图按钮作用与命令按钮相同。MFC 提供 CButton 派生类 CBitmapButton 来实现位图按钮功能,通过几个独立的位图来分别代表按钮的松开(up)、按下(down)、获得输入状态(focused)和禁止(disabled)状态。

9.1.5.4 列表框控件

列表框显示一个项列表,如文件名列表。用户可以对列表框进行浏览和选择。在单选列表框中,用户只能选择一个列表项。在多选列表框中,用户可选择一组列表项。如果列表框不能一次显示完列表项,那么系统会自动提供一个滚动条来实现滚动浏览。

MFC 提供类 CListBox 来实现列表框功能。

成员函数 AddString()用于把字符串插入到列表框中。如果在列表框创建时未指定 LBS_SORT 风格，则将字符串添加到列表的末尾。否则，在字符串插入列表时会自动排序。

成员函数 InsertString()也用于将字符串插入列表框中。与 AddString 不同的是，它可以将字符串插入到列表框中的指定位置。

成员函数 Dir()可以将当前目录中的文件名添加到列表框中。利用该成员函数就可实现对文件、目录和驱动器的列表功能。

对于单选列表框，使用成员函数 GetCurSel()返回当前所选择串的索引(以零为基准)。对于多选列表框，使用成员函数 GetSelItems()返回所选串的索引。

列表框可以向父窗口发送六种通知消息，消息前缀名为 LBN_，包括 LBN_DBLCLK(用户双击列表框中的一个字符串)、LBN_ERRSPACE(列表框不能分配足够的内存)、LBN_KILLFOCUS(列表框失去输入焦点)、LBN_SELCANCLE(取消当前列表框选择)、LBN_SELCHANGE(列表框中选择将改变)和 LBN_SETFOCUS(列表框将接收输入焦点)。程序员通过使用 ClassWizard 向父窗口类中加入相应的消息映射，其格式与编辑控件消息映射相同。

在以后各节中，读者将学习各类对话框的具体编写实现过程。

## 9.2 系统预制对话框的制作

系统预制对话框是 MFC 提供的 Windows 系统的标准对话框。这些对话框在 Windows 编程中广泛使用。与预制对话框相对应的类有：CColorDialog、CFileDialog、CFindReplaceDialog、CFontDialog 及 CPrintDialog 等。本节将以示例程序 Chap9 向读者展示文件对话框和颜色对话框的调用方法。

### 试一试

1.应用 MFC AppWizard(exe) 创建一个新项目 Chap9。选择 Finish 按钮，全部使用缺省选项。

创建的类：

```
Application: CChap9App in Chap9.h and Chap9.cpp
Frame: CMainFrame in MainFrm.h and MainFrm.cpp
MDIChildFrame: CChildFrame in ChildFrm.h and ChildFrm.cpp
Document: CChap9Doc in Chap9Doc.h and Chap9Doc.cpp
View: CChap9View in Chap9View.h and Chap9View.cpp
```

特征：

```
+ Initial toolbar in main frame
+ Initial status bar in main frame
+ Printing and Print Preview support in view
+ 3D Controls
```

+ Uses shared DLL implementation (MFC42.DLL)

2.从 ResourceView 中编辑 IDR_CHAP9TYPE 菜单,如图 9.1 所示。

| 菜单 | 菜单项 | 提示 Prompt | 命令 ID |
|---|---|---|---|
| & Property | & Color | Test Color dialog | ID_PROPERTY_COLOR |

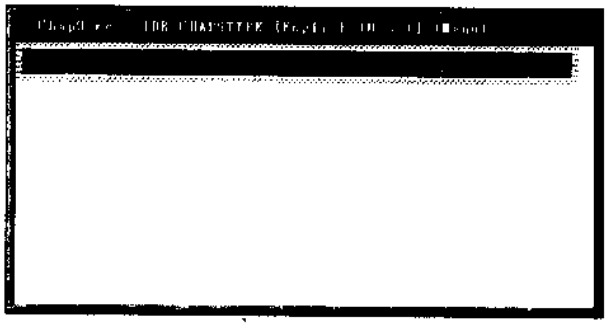


图 9.1 IDR_CHAP9TYPE 菜单

3. 利用 ClassWizard 在 CChap9Doc 中增加消息处理函数。

| 对象 | 消息 | 函数 |
|---|---|---|
| ID_FILE_OPEN | COMMAND | OnFileOpen |
| ID_FILE_SAVE_AS | COMMAND | OnFileSaveAs |
| ID_PROPERTY_COLOR | COMMAND | OnPropertyColor |

4.编写 CChap9Doc::OnFileOpen 函数。

```
void CChap9Doc::OnFileOpen()
{
    //用 TRUE 参数定义打开文件对话框。
    CFileDialog dlg(TRUE);

    //进行人机交互对话
    int ret = dlg.DoModal();

    //选择 OK 后,显示用户所选择的打开文件的名称和路径。
    if(ret == IDOK)
    {
        CString pathName = dlg.GetPathName();
        CString fileName = dlg.GetFileName();
        char  tbuf[120];
        sprintf(tbuf,"The %s file in %s is opened!",
            fileName,pathName);
        AfxMessageBox(tbuf);
```

```
    }
}
```

5. 编写 CChap9Doc::OnFileSaveAs 函数。

```
void CChap9Doc::OnFileSaveAs()
{
    //用 FALSE 参数定义存储文件对话框。
    CFileDialog dlg(FALSE);

    //进行人机交互对话
    int ret = dlg.DoModal();

    //选择 OK 后,显示用户存储文件的名称及其所在路径。
    if(ret == IDOK)
    {
        CString pathName = dlg.GetPathName();
        CString fileName = dlg.GetFileName();
        char  tbuf[120];
        sprintf(tbuf,"The %s file in %s is saved!",
            fileName,pathName);
        AfxMessageBox(tbuf);
    }
}
```

6. 编写 CChap9Doc::OnPropertyColor 函数。

```
void CChap9Doc::OnPropertyColor()
{
    //定义颜色对话框
    CColorDialog    dlg;

    //进行人机交互
    int ret = dlg.DoModal();

    //选择 OK 后,显示用户所选择的颜色值。
    if(ret == IDOK)
    {
        COLORREF    rgb = dlg.GetColor();
        char        tbuf[120];
        sprintf(tbuf,"The (%d,%d,%d) color is selected!",
            GetRValue(rgb),GetGValue(rgb),GetBValue(rgb));
        AfxMessageBox(tbuf);
    }

}
```

7. 创建和运行应用程序。

分别从菜单中选择 File \ Open、File \ Save As 和 Property \ Color 菜单项，显示打开文件、存储文件和颜色对话框。

## 9.3 普通对话框的制作

接下来，我们将制作一个除法运算对话框。用户可以在该对话框中输入被除数和除数。输入完毕后单击 OK 按钮将计算出商和余数。如果用户把整除的复选框选中，在单击 OK 按钮后，应用程序将检查是否能整除。若能整除，则返回商，否则回到原对话框中，并将焦点置于被除数编辑框；若除数为 0，则将焦点置于除数输入框，并给出相应的提示。

### 试一试

1. 打开 chap9 项目。

2. 从 ResourceView 中添加如图 9.2 所示对话框。

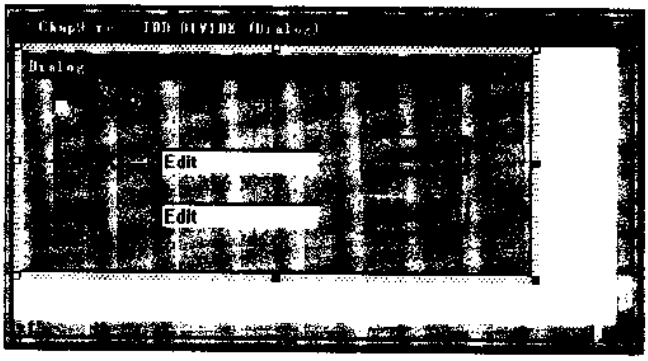



图 9.2 IDD_DIVIDE 对话框

对话框的 ID 值为 IDD_DIVISION。其中的控件 ID 和属性见下表：

| 控件 | ID | 属性 |
|---|---|---|
| 复选框 | IDC_CHECK | Caption = "整除" |
| 静态文本框 | IDC_STATIC | Caption = "被除数" |
| 静态文本框 | IDC_STATIC | Caption = "除数" |
| 被除数编辑框 | IDC_EDIT_DIVIDENT | 默认设置 |
| 除数编辑框 | IDC_EDIT_DIVISOR | 默认设置 |
| OK 按钮 | IDOK | Caption = "OK" |
| Cancel 按钮 | IDCANCEL | Caption = "Cancel" |

3. 建立一个新类 CDivisionDlg。根据下列信息利用 Classwizard 建立一个新类。

类名：CDivisionDlg

基类:CDialog

文件:DivisionDlg.cpp

ID 值:IDD_DIVISION

4.利用 ClassWizard 在 CDivisionDlg 中增加消息处理函数。

| 对象 | 消息 | 函数 |
|---|---|---|
| IDOK | BN_CLICKED | OnOK |

切换到 ClassWizard 的 Member Variables 标签中,添加下列成员变量。

| 对象 | 类型 | 成员 |
|---|---|---|
| IDC_CHECK | BOOL | m_bCheck |
| IDC_EDIT_DIVIDENT | int | m_nDividend |
| IDC_EDIT_DIVISOR | int | m_nDivisor |

5.编写 CDivisionDlg::OnOK 函数。

```
void CDivisionDlg::OnOK()
{
    //提取对话框中的数据
    UpdateData(TRUE);
    //除数为 0 时,给出警告,并将焦点设置到除数编辑框。
    if(m_nDivisor==0)
    {
        AfxMessageBox("除数不应当是 0!");
        CWnd* pWnd=GetDlgItem(IDC_EDIT_DIVISOR);
        pWnd->SetFocus();
        return;
    }

    //需要整除时
    if(m_bCheck==TRUE)
    {
        int n=m_nDividend/m_nDivisor;
        if(m_nDividend-n*m_nDivisor!=0)
        {
            AfxMessageBox("不能整除!");
            CWnd* pWnd=GetDlgItem(IDC_EDIT_DIVIDENT);
            pWnd->SetFocus();
            return;
        }
    }
```

```
    CDialog::OnOK();
}
```

6.从 ResourceView 中修改 IDR_CHAP9TYPE 菜单,如图 9.3 所示。

| 菜单 | 菜单项 | 提示 Prompt | 命令 ID |
|---|---|---|---|
| &Property | Test&1 | Test the first dialog | ID_PROPERTY_TEST1 |
| &Property | Test&2 | Test the second dialog | ID_PROPERTY_TEST2 |

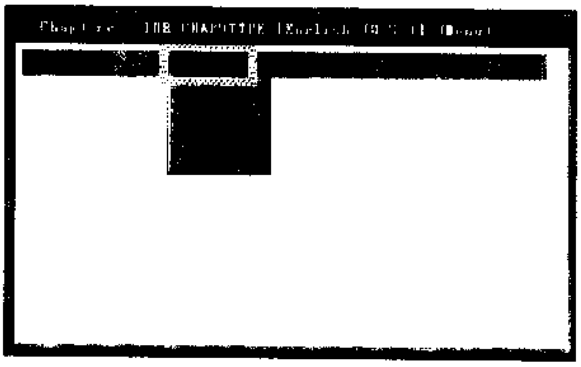


图 9.3 IDR_CHAP9TYPE 菜单

7.利用 ClassWizard 在 CChap9Doc 中增加消息处理函数。

| 对象 | 消息 | 函数 |
|---|---|---|
| ID_PROPERTY_TEST1 | COMMAND | OnPropertyTest1() |
| ID_PROPERTY_TEST2 | COMMAND | OnPropertyTest2() |

8. 编写 CChap9Doc::OnPropertyTest1 函数。

```
void CChap9Doc::OnPropertyTest1()
{
    CDivisionDlg    dlg;
    int ret = dlg.DoModal();
    if(ret == IDOK)
    {
        int     n = dlg.m_nDividend/dlg.m_nDivisor;
        int     r = dlg.m_nDividend - n * dlg.m_nDivisor;
        char  tbuf[160];
        sprintf(tbuf,"%d除以%d,\n商 = %d,\n余数 = %d",
            dlg.m_nDividend,dlg.m_nDivisor,n,r);
        AfxMessageBox(tbuf);
    }
}
```

在 Chap9Doc.cpp 的顶端添加：

```
#include "Chap9Doc.h"
#include "DivisionDlg.h"
```

9. 创建和运行应用程序。

选择 Property \ Test1 菜单项，弹出对话框，在其中输入被除数和除数进行测试。

## 9.4　高级对话框的制作

本节将介绍一个较复杂的对话框。在该对话框中有 6 个自绘制按钮。通过 More > > 或 < < Less 按钮可将其打开、关闭，如图 9.4、图 9.5 所示。选择 Pick < < 按钮，对话框将消隐一段时间，隔一会儿再出现。通过学习该对话框，读者将学会如何伸展收缩一个对话框，如何使对话框暂时消失，以及如何定制自绘制按钮。

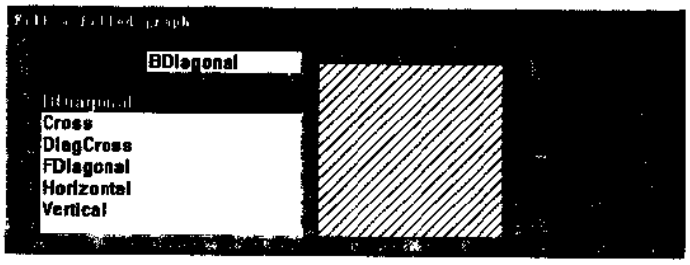



图 9.4　未展开的填充对话框

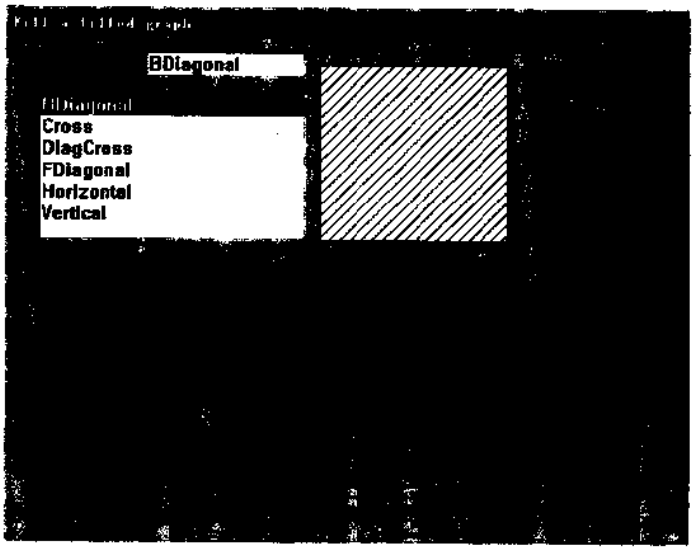



图 9.5　已展开的填充对话框

## 试一试

1.打开 Chap9 项目。

2.从 Resource View 中添加如图 9.6 所示的 IDD_DIALOG_FILL 对话框。

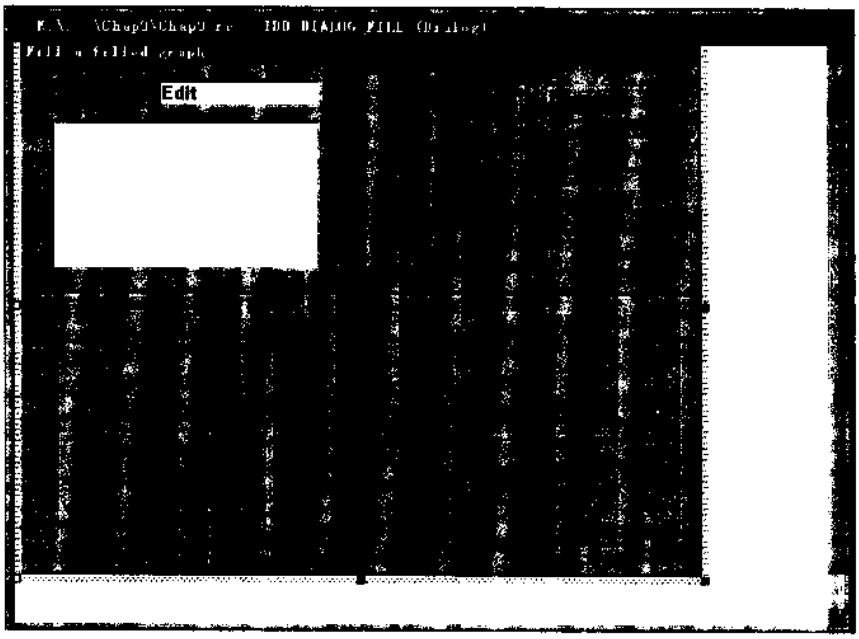



图 9.6 IDD_DIALOG_FILL 对话框

该对话框的 Caption 值为 Fill a filled graph。

| 控件 | 标识符 | 属性 |
|---|---|---|
| Fill Pattern 静态文本 | IDC_STATIC | "Fill Pattern" |
| Fill Pattern 框编辑框 | IDC_EDIT_PATTERN | 默认设置 |
| 列表框 | IDC_EDIT_PATTERN | 默认设置 |
| 中上方的图片框 | IDC_SELECTION | 默认设置 |
| Pick 按钮 | IDC_BUTTON_PICK | &Pick < < |
| Fill 按钮 | IDOK | &Fill |
| Cancel 按钮 | IDCANCEL | &Cancel |
| More 按钮 | IDC_BUTTON_MORE | &More |
| 分界符 | IDC_SLIDER | 去掉 Visible 前的勾号 |

图 9.6 中下面的六个按钮从左至右,从上至下为六个 Button 按钮。其 ID 值依次为 IDC_PATTERN1 到 IDC_PATTERN6,Style 为 Owner draw。

3.建立一个新类 CFillDlg。根据下列信息利用 ClassWizard 建立一个新类。

类名:CFillDlg

基类:CDialog

文件:FillDlg.cpp

ID 值:IDD_DIALOG_FILL

4.利用 ClassWizard 在 CFillDlg 中增加消息处理函数。

| 对象 | 消息 | 函数 |
|---|---|---|
| IDC_BUTTON_MORE | BN_CLICKED | OnButtonMore |
| IDC_BUTTON_PICK | BN_CLICKED | OnButtonPick |
| IDC_PATTERN1 | BN_CLICKED | OnPattern1 |
| IDC_PATTERN2 | BN_CLICKED | OnPattern2 |
| IDC_PATTERN3 | BN_CLICKED | OnPattern3 |
| IDC_PATTERN4 | BN_CLICKED | OnPattern4 |
| IDC_PATTERN5 | BN_CLICKED | OnPattern5 |
| IDC_PATTERN6 | BN_CLICKED | OnPattern6 |
| IDC_LIST_PATTERN | LBN_SEL CHANGE | OnSelchangelistPattern |
| CFillDlg | WM_INITDIALOG | OnInitDialog |
| CFillDlg | WM_PAINT | OnPaint |

切换到 ClassWizard 的 Member Variables 标签中,添加下列成员变量。

| 对象 | 类型 | 成员 |
|---|---|---|
| IDC_LIST_PATTERN | CListBox | m_PatternList |

5. 在 CFillDlg 中增加变量和函数。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| CPatternBtn | m_PatternBtn[6] | protected |
| int | m_nCurPattern | protected |

| 函数类型 | 函数名称 | 访问权限 |
|---|---|---|
| void | Update(int nIndex) | protected |
| void | ExpandDialog(int nResourceID, BOOL bExpand) | protected |
| void | EnableVisibleChildren() | protected |

6. 编写 CFillDlg::DoDataExchange()函数。

```
void CFillDlg::DoDataExchange(CDataExchange * pDX)
{
    CDialog::DoDataExchange(pDX);
```

```
    DDX_Control(pDX, IDC_PATTERN1, m_PatternBtn[0]);
    DDX_Control(pDX, IDC_PATTERN2, m_PatternBtn[1]);
    DDX_Control(pDX, IDC_PATTERN3, m_PatternBtn[2]);
    DDX_Control(pDX, IDC_PATTERN4, m_PatternBtn[3]);
    DDX_Control(pDX, IDC_PATTERN5, m_PatternBtn[4]);
    DDX_Control(pDX, IDC_PATTERN6, m_PatternBtn[5]);

    //{{AFX_DATA_MAP(CFillDlg)
    DDX_Control(pDX, IDC_LIST_PATTERN, m_PatternList);
    //}}AFX_DATA_MAP
}
```

7. 编写 CFillDlg::OnButtonMore()函数。

```
void CFillDlg::OnButtonMore()
{
    static BOOL bExpand = TRUE;

    ExpandDialog(IDC_SLIDER,bExpand);
    bExpand = ! bExpand;
}
```

8. 编写 CFillDlg::OnButtonPick()函数。

```
void CFillDlg::OnButtonPick()
{
    EndDialog(IDC_BUTTON_PICK);
}
```

9. 编写 CFillDlg::ExpandDialog 函数。

```
void CFillDlg::ExpandDialog(int nResourceID, BOOL bExpand)
{
    static CRect    rcLarge;
    static CRect    rcSmall;
    CString             sExpand;

    if(rcLarge.IsRectNull())
    {
        CRect rcLandmark;

        //获得对话框的分界子窗口 IDC_SLIDER 的指针。
        CWnd * pWndLandmark = GetDlgItem(nResourceID);
        ASSERT(pWndLandmark);

        //获得整个对话框(伸展开的对话框)的大小。
```

```
        GetWindowRect(rcLarge);

        //获得对话框的分界子窗口 IDC_SLIDER 的大小。
        pWndLandmark->GetWindowRect(rcLandmark);

        //设置收缩后的对话框的大小。
        rcSmall = rcLarge;
        rcSmall.bottom= rcLandmark.top;
    }

    if(bExpand)
    {
        //伸展时,用整个对话框的大小来设置对话框窗口大小。
       SetWindowPos(NULL,0,0,rcLarge.Width(),rcLarge.Height(),
            SWP_NOMOVE|SWP_NOZORDER);

        sExpand="<< &Less";
        EnableVisibleChildren();
    }
    else
    {
        //收缩时,用收缩后的对话框的大小来设置对话框窗口大小。
        SetWindowPos(NULL,0,0,rcSmall.Width(),rcSmall.Height(),
             SWP_NOMOVE|SWP_NOZORDER);

        sExpand="&More>>";
        EnableVisibleChildren();
    }

    //设置对话框的伸展收缩按钮的显示文本。
    SetDlgItemText(IDC_BUTTON_MORE,sExpand);
}
```

10. 编写 CFillDlg::EnableVisibleChildren()函数。

```
void CFillDlg::EnableVisibleChildren()
{
    //获得对话框中的子控件。

    CWnd* pWndCtrl=GetWindow(GW_CHILD);
    CRect rcTest;
    CRect rcControl;
    CRect rcShow;
```

```
    //获取当前对话框的大小。
    GetWindowRect(rcShow);

    while(pWndCtrl! =NULL)
    {
        //获得子控件窗口的大小。
        pWndCtrl->GetWindowRect(rcControl);

        //判断子控件是否在对话框显示窗口中,以决定该控件是否有效。
        if(rcTest.IntersectRect(rcShow,rcControl))
            pWndCtrl->EnableWindow(TRUE);
        else
            pWndCtrl->EnableWindow(FALSE);

        //获得对话框的下一个子控件。
        pWndCtrl=pWndCtrl->GetWindow(GW_HWNDNEXT);
    }
}
```

11. 编写 CFillDlg::OnInitDialog()函数。

```
BOOL CFillDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // TODO: Add extra initialization here
    //设置未扩展对话框
    ExpandDialog(IDC_SLIDER,FALSE);

    //设置列表框选项
    m_PatternList.AddString("BDiagonal");
    m_PatternList.AddString("Cross");
    m_PatternList.AddString("DiagCross");
    m_PatternList.AddString("FDiagonal");
    m_PatternList.AddString("Horizontal");
    m_PatternList.AddString("Vertical");

    //设置图像按钮序号
    for(int i=0;i<6;i++)
    {
        m_PatternBtn[i].m_nIndex=i;
    }

    //预先设置对话框项
```

```
    m_nCurPattern = HS_BDIAGONAL;
    Update(0);

    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}
```

12. 编写 CFillDlg::Update()函数。

```
void CFillDlg::Update(int nIndex)
{
    //获得该选项的文本
    char tbuf[40];
    m_PatternList.GetText(nIndex,tbuf);
    SetDlgItemText(IDC_EDIT_PATTERN,tbuf);

    //设置列表框项
    m_PatternList.SetCurSel(nIndex);

    //在选择填充方式框中绘图
    CWnd* pWnd = GetDlgItem(IDC_SELECTION);
    CRect rect;
    pWnd->GetClientRect(rect);

    int       nBrushPattern[6] =
              {
                  HS_BDIAGONAL,
                  HS_CROSS,
                  HS_DIAGCROSS,
                  HS_FDIAGONAL,
                  HS_HORIZONTAL,
                  HS_VERTICAL,
              };

    CDC*   pDC = pWnd->GetDC();
    CBrush* pNewBrush = new CBrush;
    ASSERT(nIndex<6);
    pNewBrush->CreateHatchBrush(nBrushPattern[nIndex],RGB(0,0,0));
    CBrush* pOldBrush = pDC->SelectObject(pNewBrush);
    pDC->Rectangle(rect);
    pDC->SelectObject(pOldBrush);
    delete      pNewBrush;

    m_nCurPattern = nBrushPattern[nIndex];
```

```
}
```

13. 编写 CFillDlg::OnPaint()函数。

```
void CFillDlg::OnPaint()
{
    CPaintDC dc(this); // device context for painting

    //在选择填充方式框绘图
    CWnd * pWnd = GetDlgItem(IDC_SELECTION);
    CRect rect;
    pWnd->GetClientRect(rect);


    CDC *   pDC = pWnd->GetDC();
    CBrush * pNewBrush = new CBrush;
    pNewBrush->CreateHatchBrush(m_nCurPattern,RGB(0,0,0));
    CBrush * pOldBrush = pDC->SelectObject(pNewBrush);
    pDC->Rectangle(rect);
    pDC->SelectObject(pOldBrush);
    delete      pNewBrush;
}
```

14. 编写 CFillDlg::OnSelchangeListPattern()函数。

```
void CFillDlg::OnSelchangeListPattern()
{
    //获得当前选项
    int index = m_PatternList.GetCurSel();
    ASSERT(index! = LB_ERR);

    Update(index);
}
```

15. 编写 CFillDlg::OnPattern1()至 CFillDlg::OnPattern6()函数。

```
void CFillDlg::OnPattern1()
{
    Update(0);
}

void CFillDlg::OnPattern2()
{
    Update(1);
}
```

```
void CFillDlg::OnPattern3()
{
    Update(2);
}

void CFillDlg::OnPattern4()
{
    Update(3);
}

void CFillDlg::OnPattern5()
{
    Update(4);
}

void CFillDlg::OnPattern6()
{
    Update(5);
}
```

16. 建立一个新类 CPatternBtn。根据下列信息，利用 ClassWizard 建立一个新类：

类名：CPatterBtn

基类：CButton

文件：FillDlg.cpp

接着，在 FillDlg.h 中将关于 CPattenDlg 类的定义移到 CfillDlg 类定义的前面。

该类是一个自绘制按钮类。

17. 在 CPatternBtn 中增加变量。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| int | m_nIndex | public |

18. 利用 ClassWizard 在 CPatternBtn 中覆盖虚函数。

```
public:
    virtual void DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct);
```

19. 编写 CPatternBtn::DrawItem()函数。

```
void CPatternBtn::DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct)
{
    CDC * pDC = CDC::FromHandle(lpDrawItemStruct->hDC);
    CRect rect(lpDrawItemStruct->rcItem);

    int        nBrushPattern[6] =
```

```
        {
            HS_BDIAGONAL,
            HS_CROSS,
            HS_DIAGCROSS,
            HS_FDIAGONAL,
            HS_HORIZONTAL,
            HS_VERTICAL,
        };

    //创建填充刷子
    CBrush*          pNewBrush = new CBrush;

    pNewBrush->CreateHatchBrush(nBrushPattern[m_nIndex],RGB(0,0,0));
    CBrush* pOldBrush = pDC->SelectObject(pNewBrush);

    //用填充刷子图案填充矩型
    pDC->Rectangle(rect);
    pDC->SelectObject(pOldBrush);
    delete      pNewBrush;
}
```

20. 编写 CChap9Doc::OnPropertyTest2()函数。

```
void CChap9Doc::OnPropertyTest2()
{
    CFillDlg    dlg;
    int             ret = IDC_BUTTON_PICK;

    //选择 Pick 按钮时,对话框消失,延迟一断时间后,又显示出来。
    while(ret == IDC_BUTTON_PICK)
    {
        ret = dlg.DoModal();
        if(ret == IDC_BUTTON_PICK)
        {
            //延时
            for(int i = 0;i < 10000;i++)
                for(int j = 0;j < 10000;j++);
        }
    }
}
```

在 FillDlg.cpp 的顶端添加:

```
#include "Chap9Doc.h"
#include "DivisionDlg.h"
```

```
#include "FillDlg.h"
```

21. 创建和运行应用程序。

选择 Property \ Test2 菜单项，弹出对话框，对其进行测试。

## 9.5 无模式对话框的制作

接下来，我们将在 Chap9 项目中添加一个无模式对话框。通过该对话框，我们将修改子窗口的底色。该对话框的运行情况如图 9.7 所示。

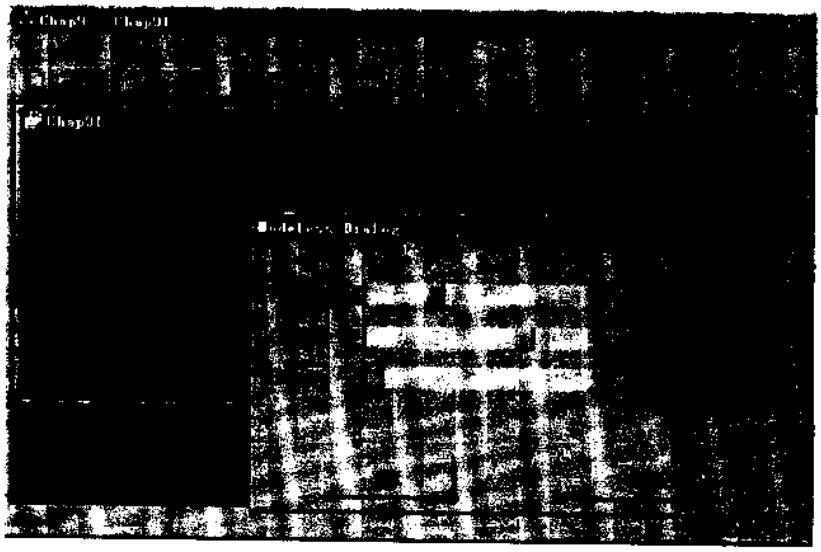


图 9.7 通过无模式对话框修改窗口的底色

### 试一试

1. 打开 Chap9 项目

2. 从 ResourceView 中添加如图 9.8 所示的 IDD_DIALOG_MODELESS 对话框。

该对话框的 Caption 值为 Modeless Dialog。

| 控件 | 标识符 | 属性 |
|---|---|---|
| Red 静态文本 | IDC_STATIC | "Red" |
| Green 静态文本 | IDC_STATIC | "Green" |
| Blue 静态文本 | IDC_STATIL | "Blue" |
| Red 滚动条 | IDC_SCOLLBAR_RED | 默认设置 |
| Green 滚动条 | IDC_SCOLLBAR_GREEN | 默认设置 |

（续表）

| 控件 | 标识符 | 属性 |
|---|---|---|
| Blue 滚动条 | IDC_SCOLLBAR_BLUE | 默认设置 |
| 图片框 | IDC_PICTURE_COLOR | 默认设置 |
| OK 按钮 | IDOK | “OK” |
| Cancel 按钮 | IDCANCEL | “Cancel” |

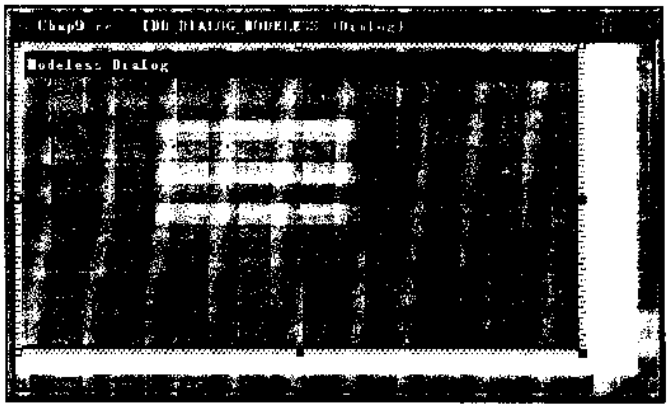



图 9.8 IDD_DIALOG_MODELESS 对话框

3. 建立一个新类 CModelessDlg。根据下列信息，利用 ClassWizard 建立一个新类：

类名：CModelessDlg

基类：CDialog

文件：ModelessDlg

ID 值：IDD_DIALOG_MODELESS

4. 利用 ClassWizard 在 CModelessDlg 中增加消息处理函数。

| 对象 | 消息 | 函数 |
|---|---|---|
| IDOK | BN_CLICKED | OnOK() |
| IDCANCEL | BN_CLICKED | OnCancel() |
| CModelessDlg | ON_WM_HSCROLL | OnHScroll() |
| &CModelessDlg | ON_WM_INITDIALOG | OnInitDialog() |

切换到 ClassWizard 的 Member Variables 标签中，添加下列成员变量：

| 对象 | 类型 | 成员 |
|---|---|---|
| IDC_SCROLLBAR_BLUE | int | m_nBlue |
| IDC_SCROLLBAR_BLUE | CScrollBar | m_nScrollBlue |
| IDC_SCROLLBAR_GREEN | int | m_nGreen |

（续表）

| 对象 | 类型 | 成员 |
| --- | --- | --- |
| IDC_SCROLLBAR_GREEN | CScrollBar | m_ScrollGreen |
| IDC_SCROLLBAR_RED | int | m_nRed |
| IDC_SCROLLBAR_RED | CScrollBar | m_ScrollRed |

5. 在 CModelessDlg 中增加变量和函数。

| 变量类型 | 变量名称 | 访问权限 |
| --- | --- | --- |
| CWnd * | m_pParentWnd | public |

| 函数类型 | 函数名称 | 访问权限 |
| --- | --- | --- |
| CWnd * | Create() | public |

6. 编写 CModelessDlg::CModelessDlg()函数。

```
CModelessDlg::CModelessDlg(CWnd* pParent /* =NULL*/)
//    : CDialog(CModelessDlg::IDD, pParent)
{
      m_pParentWnd=pParent;
      //{{AFX_DATA_INIT(CModelessDlg)
      m_nBlue = 0;
      m_nGreen = 0;
      m_nRed = 0;
      //}}AFX_DATA_INIT
}
```

注意将 CDialog(CModelessDlg::IDD, pParent)注释掉。

7. 编写 CModelessDlg::OnHScroll()函数。

```
void CModelessDlg::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
{
      int nCurPos,nStep=5;

      nCurPos=pScrollBar->GetScrollPos();
      switch(nSBCode)
      {
      //拖动滑块
      case SB_THUMBPOSITION:
            pScrollBar->SetScrollPos(nPos);
            break;

      //按下左边的箭头
      case SB_LINEUP:
```

```
        if((nCurPos - nStep) > 0)
        {
            nCurPos - = nStep;
        }
        else
        {
            nCurPos = 0;
        }
        pScrollBar - > SetScrollPos(nCurPos);
        break;

    //按下右边的箭头
    case SB_LINEDOWN:
        if((nCurPos + nStep) < 255)
        {
            nCurPos + = nStep;
        }
        else
        {
            nCurPos = 255;
        }
        pScrollBar - > SetScrollPos(nCurPos);
        break;
    default:
        break;
    }

    //根据选中的滚动条给相应的变量赋值
    if(pScrollBar = = &m_scrollRed)
        m_nRed = nCurPos;
    else if(pScrollBar = = &m_scrollGreen)
        m_nGreen = nCurPos;
    else
        m_nBlue = nCurPos;

    //绘制图片框
    CWnd * pWnd = GetDlgItem(IDC_PICTURE_COLOR);
    CRect rect;
    pWnd - > GetClientRect(rect);

    CDC *  pDC = pWnd - > GetDC();
    CBrush * pNewBrush = new CBrush;
    pNewBrush - > CreateSolidBrush(RGB(m_nRed,m_nGreen,m_nBlue));
```

```
    CBrush * pOldBrush = pDC - > SelectObject(pNewBrush);
    pDC - > Rectangle(rect);
    pDC - > SelectObject(pOldBrush);
    delete      pNewBrush;

    //给父窗口发送绘制消息
    if(m_pParentWnd! = NULL)
    {
      m_pParentWnd - > SendMessage(WM_REDRAW,RGB(m_nRed,m_nGreen,m_nBlue));
    }
}
```

8. 编写 CModelessDlg::OnInitDialog()函数。

```
BOOL CModelessDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    //设置红色滚动条的范围与当前值
    m_scrollRed.SetScrollRange(0,255);
    m_scrollRed.SetScrollPos(m_nRed);

    //设置绿色滚动条的范围与当前值
    m_scrollGreen.SetScrollRange(0,255);
    m_scrollGreen.SetScrollPos(m nRed);

    //设置蓝色滚动条的范围与当前值
    m_scrollBlue.SetScrollRange(0,255);
    m_scrollRed.SetScrollPos(m_nRed);

    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}
```

9. 编写 CModelessDlg::Create()函数。

```
BOOL CModelessDlg::Create()
{
    return CDialog::Create(CModelessDlg::IDD);
}
```

10. 编写 CModelessDlg::OnCancel()函数。

```
void CModelessDlg::OnCancel()
{
    //不要调用 CDialog::OnCancel();
```

```
        if(m_pParentWnd! = NULL)
        {
            DestroyWindow();
        }
    }
```

11. 编写 CModelessDlg::OnOK()函数。

```
void CModelessDlg::OnOK()
{
    //不要调用 CDialog::OnOK();
    if(m_pParentWnd! = NULL)
    {
        DestroyWindow();
    }
}
```

12. 从 ResourceView 修改 IDR_CHAP9TYPE 菜单，如图 9.9 所示。

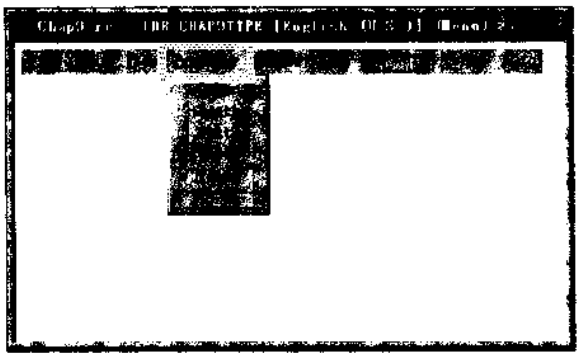


图 9.9 IDR_CHAP9TYPE 菜单

| 菜单 | 菜单项 | 提示 Prompt | 命令 ID |
|---|---|---|---|
| &Property | Test&3 | Test the third dialog | ID_PROPERTY_Test3 |
| &Property | Test&4 | Test the fourth dialog | ID_PROPERTY_Test4 |

13. 利用 ClassWizard 在 CChap9View 中增加消息处理函数。

| 对象 | 消息 | 函数 |
|---|---|---|
| ID_PROPERTY_TEST3 | COMMAND | OnPropertyTest3 |
| ID_PROPERTY_TEST4 | COMMAND | OnPropertyTest4 |

14. 添加用户自定义消息处理函数。

在 resource.h 文件的顶端加入用户自定义消息：

```
//{{NO_DEPENDENCIES}}
```

```
// Microsoft Developer Studio generated include file.
// Used by Chap9.rc
//
#define WM_REDRAW                       WM_USER + 5

#define IDD_ABOUTBOX                    100
```




WM_USER 是 Windows 为用户定义的第一个消息。由于前几个消息已由应用程序缺省地应用，所以应偏移几个，从后面开始使用。

在 CChap9View.h 中添加 OnRedraw()函数。

```
protected:
    void OnRedraw(UINT wParam,LONG lParam);
```

在 CChap9View.cpp 中添加对 OnRedraw()函数的消息映射宏。

```
BEGIN_MESSAGE_MAP(CChap9View, CView)
    ON_MESSAGE(WM_REDRAW, OnRedraw)
    //}}AFX_MSG_MAP(CChap9View)
    ON_COMMAND(ID_PROPERTY_TEST3, OnPropertyTest3)
    ON_COMMAND(ID_PROPERTY_TEST4, OnPropertyTest4)
    //}}AFX_MSG_MAP
    ......
END_MESSAGE_MAP()
```

接下来，编写 OnRedraw()函数。

```
void CChap9View::OnRedraw(UINT wParam,LONG lParam)
{
    CDC *   pDC = GetDC();
    CRect *       rect;
    GetClientRect(rect);

    CBrush * pNewBrush = new CBrush;
    pNewBrush -> CreateSolidBrush(wParam);
    CBrush * pOldBrush = pDC -> SelectObject(pNewBrush);
    pDC -> Rectangle(rect);
    pDC  > SelectObject(pOldBrush);
    delete      pNewBrush;

}
```

15. 在 CChap9View 中增加变量。

| 变量类型 | 变量名称 | 访问权限 |
| --- | --- | --- |
| CModelessDlg * | m_pDlg | public |

16. 编写 CChap9View 的构造函数和析构函数。

```
CChap9View::CChap9View()
{
    m_pDlg = new CModelessDlg((CWnd *) this);
}

CChap9View::~CChap9View()
{
    delete m_pDlg;
}
```

17. 编写 CChap9View::OnPropertyTest3()函数。

```
void CChap9View::OnPropertyTest3()
{
    if(m_pDlg->GetSafeHwnd() == 0)
    {
        //显示对话框
        m_pDlg->Create();
    }
}
```

18. 添加头文件。

在 Chap9View.cpp,Chap9.cpp 的顶端添加下列文件:

```
#include "Chap9Doc.h"
#include "ModelessDlg.h"
#include "Chap9View.h"
```

19. 创建和运行应用程序。

选择 Property \ test3 菜单,弹出无模式对话框。拖动红、绿、蓝三根滚动条,注意对话框中图片框以及子框口底色的变化。

## 9.6 属性对话框的制作

为了利用有限的屏幕空间,进行数量更大的数据交换,可以引入属性对话框。属性对话框就是将多个属性页(普通对话框)合成一个属性单的对话框。例如,Visual C++6.0 中的 MFC ClassWizard 对话框即是属性对话框,如图 9.10 所示。

接下来,我们将在 Chap 9 项目中添加状态对话框(CStatusDlg)。

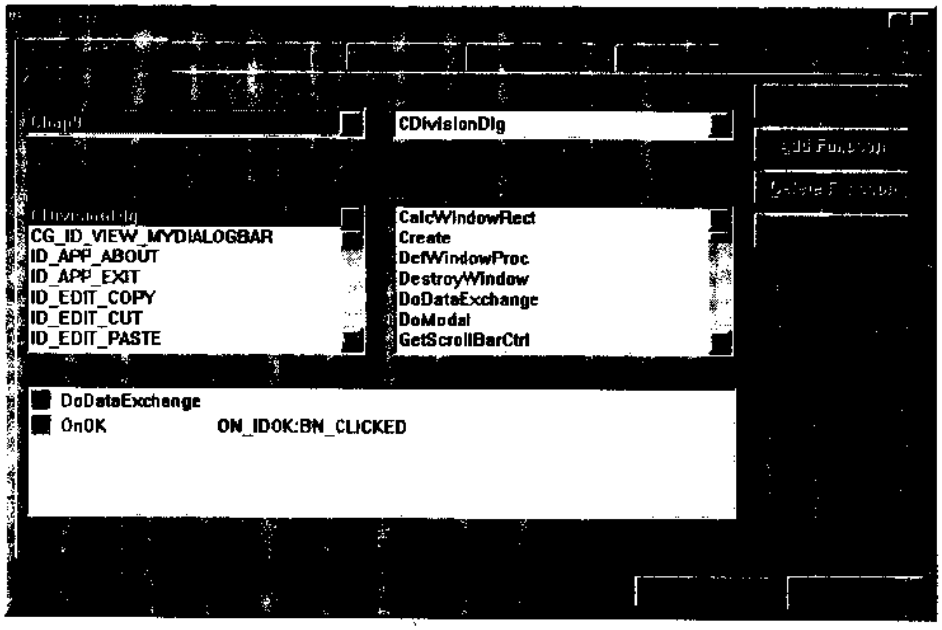



图 9.10 MFC ClassWizard 属性对话框

## 试一试

1.打开 chap9 项目。

2.选择 Project \ Add To Project \ Components and Controls...菜单项,弹出 Components and controls Gallery 对话框。双击 Developer Studio Components 文件夹,再双击 Property Sheet 组件,弹出 Microsoft Developer Studio 对话框,选择确定。接着弹出 Property Sheet Wizard 对话框,如图 9.11 所示选择"下一步 >"按钮。弹出如图 9.12 所示对话框,再选择"下一步 >"按钮。此时出现如图 9.13 所示对话框,从下拉列表框中选择 CChap9View,再选择"下一步 >"按钮。出现如图 9.14 所示对话框,将数值改为 3,选择"下一步 >"按钮,弹出如图 9.15 所示对话框。选择 Change 按钮,弹出如图 9.16 所示 Change Class 对话框。修改类名 Class 为 CStatusDlg,头文件 Header File 为 StatusDlg.h,实现文件 Implementation File 改为 StatusDlg.cpp。

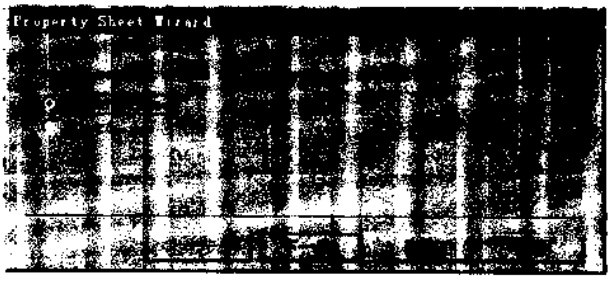



图 9.11 Property Sheet Wizard 对话框

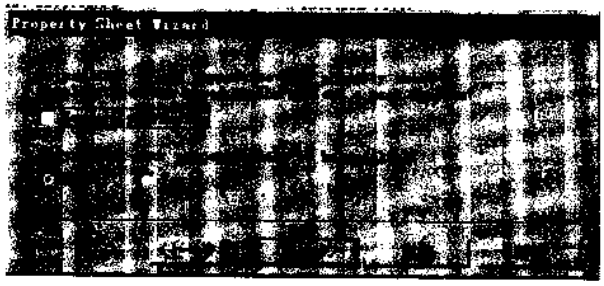



图 9.12 Property Sheet Wizard 对话框

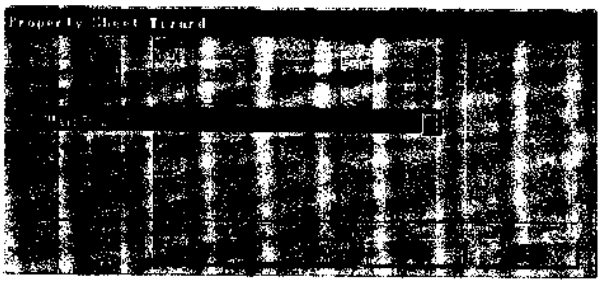



图 9.13 Property Sheet Wizard 对话框

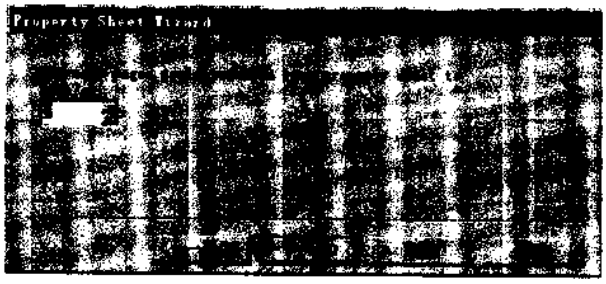



图 9.14 Property Sheet Wizard 对话框

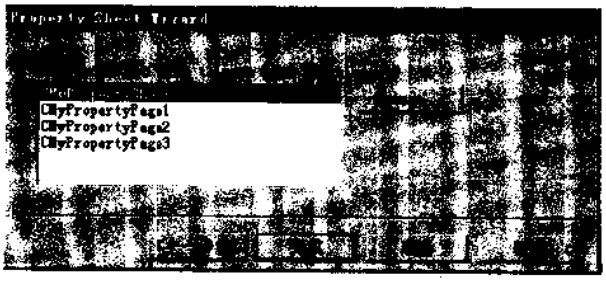



图 9.15 Property Sheet Wizard 对话框

单击 OK 按钮，退回到 Property Sheet Wizard 对话框中。接着修改 CMyPropertyPage1、CMyPropertyPage2 及 CMyProperty3 的类名以及所在文件，具体内容如下：

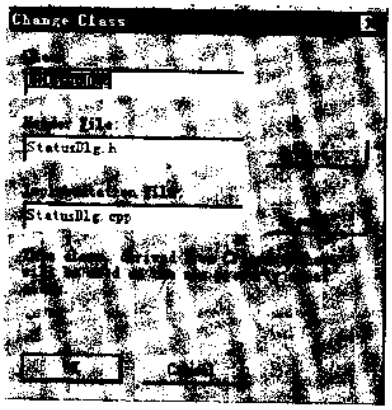



图 9.16 Change Class 对话框

类名分别为:COrdinatePage、CSetPage、COSnapPage

头文件为:StatusPage.h

实现文件为:StatusPage.cpp

修改后单击"完成"按钮。回到 Components and Controls Gallery 对话框,再单击 Close 按钮。

Property Sheet 组件为我们生成的代码在 StatusDlg.h、StatusDlg.cpp、StatusPage.h 以及 StatusPage.cpp 文件中。同时,它还生成了三个缺省的对话框资源:IDD_PROPPAGE1、IDD_PROPPAGE2 和 IDD_PROPPAGE3。而且该组件还在视类中添加了一些代码。

3.从 ResourceView 中编辑 IDD_PROPPAGE1 对话框资源,如图 9.17 所示。将对话框的 ID 值改为 IDD_ORDINATEPAGE,Caption 改为 Ordinate。其中的控件见下表:

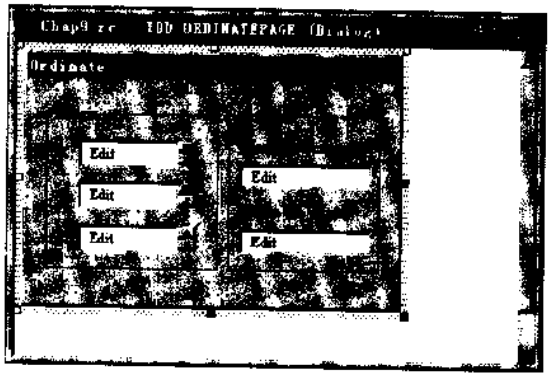



图 9.17 IDD_ORDINATEPAGE 对话框

| 控件 | 标识符 | 属性 |
|---|---|---|
| Origin 组合框 | IDC_STATIC | "Origin" |
| X:静态文本 | IDC_STATIC | "X:" |

（续表）

| 控件 | 标识符 | 属性 |
|---|---|---|
| Y:静态文本 | IDC_STATIC | "Y:" |
| Z:静态文本 | IDC_STATIC | "Z:" |
| X编辑框 | IDC_EDIT_X | 默认设置 |
| Y编辑框 | IDC_EDIT_Y | 默认设置 |
| Z编辑框 | IDC_EDIT_Z | 默认设置 |
| X微调按钮 | IDC_SPIN_X | 默认设置 |
| Y微调按钮 | IDC_SPIN_Y | 默认设置 |
| Z微调按钮 | IDC_SPIN_Z | 默认设置 |
| Scale组合框 | IDC_STATIC | "Scale" |
| World To Screen静态文本 | IDC_STATIC | "World To Screen:" |
| Screen To World静态文本 | IDC_STATIC | "Screen To World:" |
| World To Screen编辑框 | IDC_EDIT_WTOS | 默认设置 |
| Screen To World编辑框 | IDC_EDIT_STOW | 默认设置 |

4.从ResourceView中编辑IDD_PROPPAGE2对话框资源，如图9.18所示。将对话框的ID值改为IDD_SETPAGE，Caption改为Set。该对话框中的控件见下表：

| 控件 | 标识符 | 属性 |
|---|---|---|
| Grid组合框 | IDC_STATIC | "Grid" |
| On单选按钮 | IDC_GRID_ON | Group = checkde |
| Off单选按钮 | IDC_GRID_OFF | 默认设置 |
| Size静态文本 | IDC_STATIC | "Size" |
| Size编辑框 | IDC_EDIT_GRID | 默认设置 |
| Snap组合框 | IDC_STATIC | "Snap" |
| On单选按钮 | IDC_SNAP_ON | Group = Checked |
| Off单选按钮 | IDC_SNAP_OFF | 默认设置 |
| Size静态文本 | IDC_STATIC | "Size" |
| Size编辑框 | IDC_EDIT_SNAP | 默认设置 |
| Ortho组合框 | IDC_STATIC | 默认设置 |
| On单选按钮 | IDC_ORTHO_ON | Group = checkde |
| Off单选按钮 | IDC_ORTHO_OFF | 默认设置 |

5.从ResourceView中编辑IDD_PROPPAGE3对话框资源，如图9.19所示。将该对话框资源的ID值改为IDD_OSNAPPAGE，Caption改为OSnap。该对话框中的控件见下表：

| 控件 | 标识符 | 属性 |
|---|---|---|
| Type 组字框 | IDC_STATIC | "Type" |
| Endpoint 复选按钮 | IDC_ENDPOINT | 默认设置 |
| Midpoint 复选按钮 | IDC_MIDPOINT | 默认设置 |
| Center 复选按钮 | IDC_CENTER | 默认设置 |
| Quardant 复选按钮 | IDC_QUARDANT | 默认设置 |
| Size 组字框 | IDC_STATIC | "Size" |
| 图片框 | IDC_PICTURE_OSNAP | 默认设置 |
| 滚动条 | IDC_OSNAP_SIZE | 默认设置 |

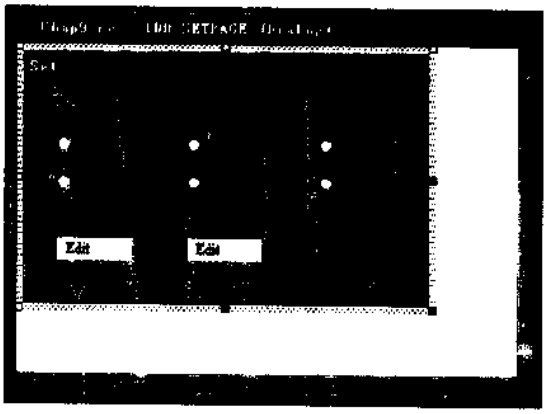



图 9.18 IDD_SETPAGE 对话框

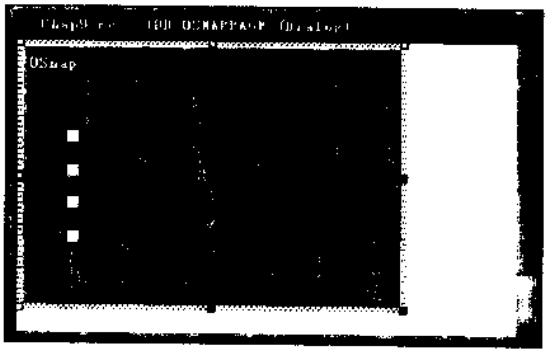



图 9.19 IDD_OSNAPPAGE 对话框资源

6.修改 StatusPage.h 文件。将 CordinatePage、CosnapPage 以及 CSetPage 中的 enum { IDD = IDD_PROPPAGEX}分别改为：

```
enum { IDD = IDD_ORDINATEPAGE };
enum { IDD = IDD_SETPAGE };
enum { IDD = IDD_OSNAPPAGE };
```

7.利用 ClassWizard 在 COrdinatePage 中增加消息处理函数。

| 对象 | 消息 | 函数 |
|---|---|---|
| IDC_SPIN_X | UDN_DELTAPOS | OnDeltaposSpinX |
| IDC_SPIN_Y | UDN_DELTAPOS | OnDeltaposSpinY |
| IDC_SPIN_Z | UDN_DELTAPOS | OnDeltaposSpinZ |
| COrdinatePage | WM_INITDIALOG | OnInitDialog |
| IDC_EDIT_STOW | EN_KILLFOCUS | OnKillfocusEditStow |
| IDC_EDIT_WTOS | EN_KILLFOCUS | OnKillfocusEsitWtos |

切换到 ClassWizard 的 Member Variables 标签中,添加下列成员变量:

| 对象 | 类型 | 成员 |
|---|---|---|
| IDC_EDIT_STOW | double | m_dStoW |
| IDC_EDIT_WTOS | double | m_dWtos |
| IDC_EDIT_X | double | m_dOriginX |
| IDC_EDIT_Y | double | m_dOriginY |
| IDC_EDIT_Z | double | m_dOriginZ |

8.利用 ClassWizard 在 CSetPage 中增加变量。

| 对象 | 类型 | 成员 |
|---|---|---|
| IDC_EDIT_GRID | double | m_dGrid |
| IDC_EDIT_SNAP | double | m_dSnap |
| IDC_GRID_ON | int | m_nGrid |
| IDC_ORTHO_ON | int | m_nOrtho |
| IDC_OSNAP_ON | int | m_nOSnap |

9.利用 ClassWizard 在 COSnapPage 中增加消息处理函数。

| 对象 | 消息 | 函数 |
|---|---|---|
| COSnapPage | WM_HSCROLL | OnHScroll |
| COSnapPage | WM_INITDIALOG | OnInitDialog |

切换到 ClassWizard 的 Member Variables 标签中,添加下列成员变量:

| 对象 | 类型 | 成员 |
|---|---|---|
| IDC_CENTER | BOOL | m_bCenter |
| IDC_ENDPOINT | BOOL | m_bEndpoint |
| IDC_MIDPOINT | BOOL | m_bMidpoint |
| IDC_QUARDANT | BOOl | m_bQuardant |

10. 编写 COrdinatePage::COrdinatePage()函数。

```
COrdinatePage::COrdinatePage() : CPropertyPage(COrdinatePage::IDD)
{
    //{{AFX_DATA_INIT(COrdinatePage)
    m_dStoW = 1.0;
    m_dWtoS = 1.0;
    m_dOriginX = 0.0;
    m_dOriginY = 0.0;
    m_dOriginZ = 0.0;
    //}}AFX_DATA_INIT
}
```

11. 编写 CSetPage::CSetPage()函数。

```
CSetPage::CSetPage() : CPropertyPage(CSetPage::IDD)
{
    //{{AFX_DATA_INIT(CSetPage)
    m_dGrid = 0.5;
    m_dSnap = 0.5;
    m_nGrid = 1;
    m_nOrtho = 1;
    m_nSnap = 1;
    //}}AFX_DATA_INIT
}
```

12. 编写 COSnapPage::COSnapPage()函数。

```
COSnapPage::COSnapPage() : CPropertyPage(COSnapPage::IDD)
{
    m_nOSnapSize=4;
    //{{AFX_DATA_INIT(COSnapPage)
    m_bCenter = FALSE;
    m_bEndpoint = FALSE;
    m_bMidpoint = FALSE;
    m_bQuardant = FALSE;
    //}}AFX_DATA_INIT
}
```

在 COSnapPage 中添加 m_nOSnapSize 变量。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| int | m_nOSnapSize | public |

13. 编写 COrdinatePage::OnInitDialog()函数。

```
BOOL COrdinatePage::OnInitDialog()
```

```
{
    CPropertyPage::OnInitDialog();

    CSpinButtonCtrl *  pSpin;
    pSpin = (CSpinButtonCtrl *)GetDlgItem(IDC_SPIN_X);
    pSpin->SetRange(-10000,10000);
    pSpin = (CSpinButtonCtrl *)GetDlgItem(IDC_SPIN_Y);
    pSpin->SetRange(-10000,10000);
    pSpin = (CSpinButtonCtrl *)GetDlgItem(IDC_SPIN_Z);
    pSpin->SetRange(-10000,10000);

    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}
```

14. 编写 COrdinatePage::OnDeltaposSpinX()函数。

```
void COrdinatePage::OnDeltaposSpinX(NMHDR* pNMHDR, LRESULT* pResult)
{
    NM_UPDOWN* pNMUpDown = (NM_UPDOWN*)pNMHDR;

    m_dOriginX += pNMUpDown->iDelta;
    char tbuf[80];
    sprintf(tbuf,"%f",m_dOriginX);
    SetDlgItemText(IDC_EDIT_X,tbuf);

    *pResult = 0;
}
```

15. 编写 COrdinatePage::OnDeltaposSpinY()函数。

```
void COrdinatePage::OnDeltaposSpinY(NMHDR* pNMHDR, LRESULT* pResult)
{
    NM_UPDOWN* pNMUpDown = (NM_UPDOWN*)pNMHDR;

    m_dOriginY += pNMUpDown->iDelta;
    char tbuf[80];
    sprintf(tbuf,"%f",m_dOriginY);
    SetDlgItemText(IDC_EDIT_Y,tbuf);

    *pResult = 0;
}
```

16. 编写 COrdinatePage::OnDeltaposSpinZ()函数。

```
void COrdinatePage::OnDeltaposSpinZ(NMHDR* pNMHDR, LRESULT* pResult)
```

```
{
    NM_UPDOWN* pNMUpDown = (NM_UPDOWN*)pNMHDR;

    m_dOriginZ += pNMUpDown->iDelta;
    char tbuf[80];
    sprintf(tbuf,"%f",m_dOriginZ);
    SetDlgItemText(IDC_EDIT_Z,tbuf);

    *pResult = 0;
}
```

17. 编写 COrdinatePage::OnKillfocusEditStow()函数。

```
void COrdinatePage::OnKillfocusEditStow()
{
    UpdateData();
    if(fabs(m_dStoW)<1e-16)
    {
        AfxMessageBox("The scale of screen to world should not be a zero!");
        GetDlgItem(IDC_EDIT_STOW)->SetFocus();
    }
    else
    {
        m_dWtoS=1.0/m_dStoW;
        char tbuf[80];
        sprintf(tbuf,"%f",m_dWtoS);
        SetDlgItemText(IDC_EDIT_WTOS,tbuf);
    }
}
```

在 StatusPage.cpp 的顶端添加下列头文件：

```
#include <math.h>
```

18. 编写 COrdinatePage::OnKillfocusEditWtos()函数。

```
void COrdinatePage::OnKillfocusEditWtos()
{
    UpdateData();
    if(fabs(m_dWtoS)<1e-16)
    {
        AfxMessageBox("The scale of world to screen should not be a zero!");
        GetDlgItem(IDC_EDIT_WTOS)->SetFocus();
    }
    else
    {
```

```
        m_dStoW = 1.0/m_dWtoS;
        char tbuf[80];
        sprintf(tbuf,"%f",m_dStoW);
        SetDlgItemText(IDC_EDIT_STOW,tbuf);
    }
}
```

19. 编写 COSnapPage::OnInitDialog()函数。

```
BOOL COSnapPage::OnInitDialog()
{
    CPropertyPage::OnInitDialog();

    //设置红色滚动条的范围与当前值
    CSliderCtrl * pSlider = (CSliderCtrl *) GetDlgItem(IDC_OSNAP_SIZE);
    pSlider->SetRange(2,20);
    pSlider->SetPos(m_nOSnapSize);

    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}
```

20. 编写 COSnapPage::OnHScroll()函数。

```
void (UINT nSBCode, UINT nPos, CScrollBar * pScrollBar)
{
    CSliderCtrl * pSlider = (CSliderCtrl *) GetDlgItem(IDC_OSNAP_SIZE);
    m_nOSnapSize = pSlider->GetPos();

    //绘制图片框
    CWnd * pWnd = GetDlgItem(IDC_PICTURE_OSNAP);
    CRect rect;
    pWnd->GetClientRect(rect);
    int nCenterX = (rect.left + rect.right)/2;
    int nCenterY = (rect.top + rect.bottom)/2;

    CDC *  pDC = pWnd->GetDC();

    //刷新图片框
    pDC->Rectangle(rect);

    //绘制十字叉丝线
    pDC->MoveTo(rect.left + 5,nCenterY);
    pDC->LineTo(rect.right - 5,nCenterY);
    pDC->MoveTo(nCenterX,rect.top + 5);
```

```
    pDC->LineTo(nCenterX,rect.bottom-5);

    //绘制十字框
    pDC->Rectangle(nCenterX-m_nOSnapSize,
                   nCenterY-m_nOSnapSize,
                   nCenterX+m_nOSnapSize,
                   nCenterY+m_nOSnapSize);
}
```

21. 编写 CChap9View::OnPropertyTest4()函数。

```
void CChap9View::OnPropertyTest4()
{
    CStatusDlg  dlg;
    dlg.DoModal();
}
```




读者应当删除 CChap9View::OnProperty()函数。该函数由属性对话框组件添加，它应当与某个消息项相对应。由于我们已用 CChap9View::OnPropertyTest4 函数来替代该函数了，所以应当把它删掉。但不删除时，对应用程序无任何影响。

22. 创建和运行应用程序。

在不同的属性页间切换，从中输入各种值，注意这种属性对话框与普通对话框的共同点以及差别。选择 Property \ Test4 菜单项，弹出如图 9.20 所示对话框。

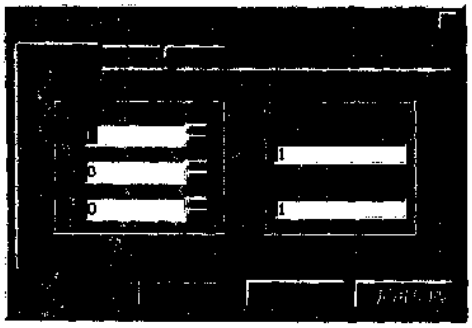


图 9.20 Property Sheet 对话框

## 9.7 对话框条

严格地讲，对话框条不属于对话框，但它的外观及其所用资源与对话框的极其相似。事实上，对话框条的支持类 CDialogBar 是继承 CControlBar 类而来的，它与菜单条和状态栏的基类相同，在使用时也相似。本节将引导读者在 Chap9 项目中添加一个对话框条。利用该对

话框条中的按钮调用 Property 菜单中的菜单项。

## 试一试

1.打开 Chap9 项目。

2.选择 Add To Project \ Components and Controls...菜单项,弹出 Components and Controls Gallery 对话框。双击 Developer Studio Components 文件夹,再双击 Dialog bar 组件。弹出 Microsoft Developer Studio 对话框,单击 OK 按钮。出现 Dialog Bar 对话框,如图 9.21 所示。在 Default docking edge 组合框中单击 Left 单选按钮,再单击 OK 按钮。接着,弹出 Dialog Bar Component 对话框,单击 OK 按钮。回到 Components and Controls Gallery 对话框,再单击 Close 按钮。CMyDialogBar 已加入到 Chap9 项目中。Dialog Bar 组件为我们添加的代码如下:




图 9.21 Dialog Bar 对话框

在 MainFrm.h 中对 m_wndMyDialogBar 的定义:

```
protected:
    CDialogBar m_wndMyDialogBar;
```

在 CMainFrame::OnCreate 函数中添加的代码为:

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    .....
    // TODO: Add a menu item that will toggle the visibility of the
    // dialog bar named "My Dialog Bar":
    //  1. In ResourceView, open the menu resource that is used by
    //       the CMainFrame class
    //  2. Select the View submenu
    //  3. Double-click on the blank item at the bottom of the submenu
    //  4. Assign the new item an ID: CG_ID_VIEW_MYDIALOGBAR
    //  5. Assign the item a Caption: My Dialog Bar

    // TODO: Change the value of CG_ID_VIEW_MYDIALOGBAR to an appropriate value:
    //  1. Open the file resource.h
```

```
    // CG: The following block was inserted by the 'Dialog Bar' component
    {
      // Initialize dialog bar m_wndMyDialogBar
      if (! m_wndMyDialogBar.Create(this, CG_IDD_MYDIALOGBAR,
        CBRS_LEFT | CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_HIDE_INPLACE,
        CG_ID_VIEW_MYDIALOGBAR))
      {
          TRACE0("Failed to create dialog bar m_wndMyDialogBar\n");
          return -1;        // fail to create
      }

      m_wndMyDialogBar.EnableDocking(CBRS_ALIGN_LEFT | CBRS_ALIGN_RIGHT);
      EnableDocking(CBRS_ALIGN_ANY);
      DockControlBar(&m_wndMyDialogBar);

    }

    return 0;
}
```

3.从 ResourceView 中编辑 IDR_CHAP9TYPE 菜单,如图 9.22 所示。

| 菜单 | 菜单项 | 提示 Prompt | ID 值 |
|---|---|---|---|
| &View | &MyDialogBar | Toggle the Visibility of the dialog bar named"My Dialog Bar" | CG_ID_VIEWMYDIALOGBAR |

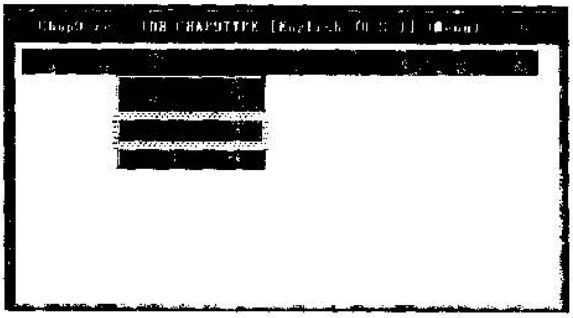


图 9.22 IDR_CHAP9TYPE 菜单

4.从 ResourceView 中编辑 CG_IDD_MYDIALOGBAR 对话框资源,如图 9.23 所示。对话框中的控件及属性如下表:

| 控件 | 标识符 | 属性 |
|---|---|---|
| Invoke Dialog 静态文本 | IDC_STATIC | "Invoke Dialog" |
| Color 按钮 | ID_PROPERTY_COLOR | "Color" |
| Test1 按钮 | ID_PROPERTY_TEST1 | "TEST1" |

（续表）

| 控件 | 标识符 | 属性 |
| --- | --- | --- |
| Test2 按钮 | ID_PROPERTY_TEST2 | "TEST2" |
| Test3 按钮 | ID_PROPERTY_TEST3 | "TEST3" |
| Test4 按钮 | ID_PROPERTY_TEST4 | "TEST4" |

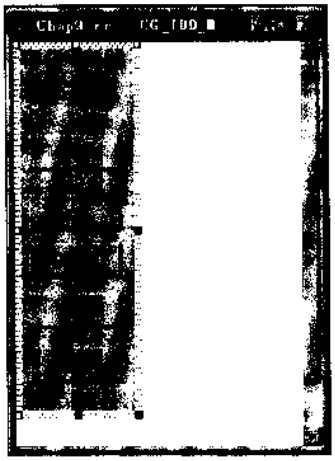


图 9.23 CG_IDD_MYDIALOGBAR 对话框资源

5. 创建和运行应用程序。

单击 Dialog Bar 中的按钮，调用对话框。拖动 Dialog Bar，观察该窗口的停靠情况（图 9.24）。

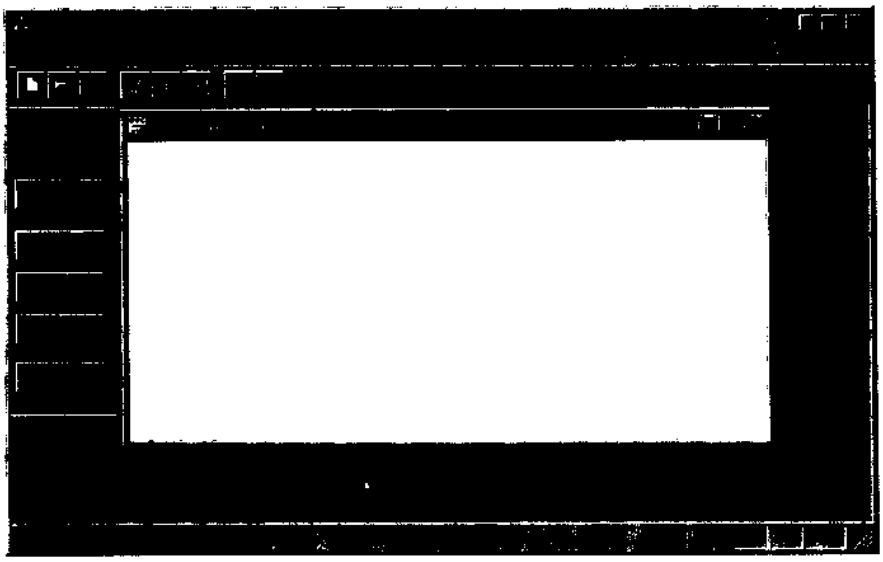


图 9.24 具有 Dialog Bar 的应用程序

## 9.8 基于对话框的应用程序

当应用程序的外观大小固定时,我们便可以使用基于对话框的应用程序。顾名思义,基于对话框的应用程序即是在对话框中运行的程序,它以一个对话框为应用程序的框体。如图 9.25 所示的 CD Player 即是基于对话框的应用程序。

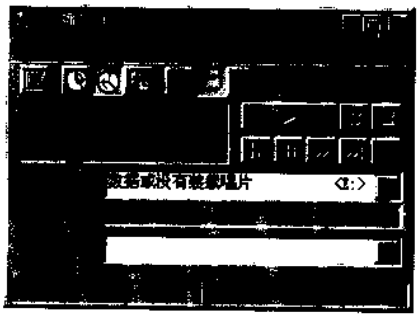


图 9.25 CD Player 应用程序

本节将引导读者创建一个基于对话框的四则运算器,如图 9.26 所示。

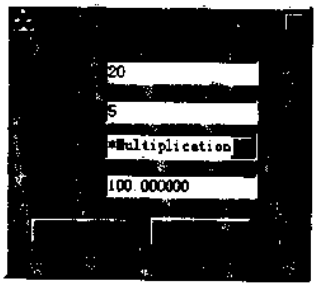



图 9.26 基于对话框的四则运算器

### 试一试

1.应用 MFC AppWizard(exe) 创建一个新项目 Chap9Dlg。在第 1 步中选择 Dialog based。创建的类:

```
Application: CChap9DlgApp in Chap9Dlg.h and Chap9Dlg.cpp
Dialog: CChap9DlgDlg in Chap9DlgDlg.h and Chap9DlgDlg.cpp
```

特征:

```
+ About box on system menu
+ 3D Controls
+ Uses shared DLL implementation (MFC42.DLL)
```

2.从 ResourceView 中编辑 IDD_CHAP9DLG_DIAIOG 对话框,如图 9.27 所示。

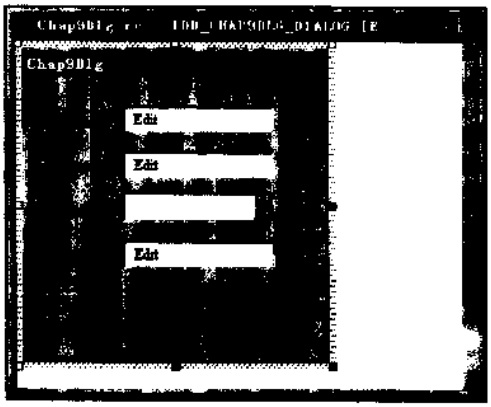



图 9.27 IDD_CHAP9DLG_DIALOG 对话框

该对话框中的控件及其属性如下表:

| 控件 | 标识符 | 属性 |
|---|---|---|
| Operand1 静态文本 | IDC_STATIC | "Operand1:" |
| Operand2 静态文本 | IDC_STATIC | "Operand2:" |
| Operator:静态文本 | IDC_STATIC | "Operator:" |
| Result:静态文本 | IDC_STAIC | "Result:" |
| Operand1 编辑框 | IDC_EDIT_OPER1 | 默认设置 |
| Operand2 编辑框 | IDC_EDIT_OPER2 | 默认设置 |
| Operator 下拉列表框 | IDC_COMBO_OPERATE | Data = " + Addition"<br>Ctrl + Return " - Substract"<br>Ctrl + Return " * Multiplicator"<br>Ctrl + Return "/Division" |
| Result 编辑框 | IDC_EDIT_RESULT | 默认设置 |
| Compute 按钮 | IDC_COMPUTE | "Compute" |
| Exit 按钮 | IDOK | "Exit" |

3.利用 ClassWizard 在 CChap9DlgDlg 中增加消息处理函数。

| 对象 | 消息 | 函数 |
|---|---|---|
| IDC_COMPUTE | BN_CLICKED | OnCompute |

切换到 Member Variables 标签中,在 CChap9Dlgdlg 类中添加下列成员变量:

| 对象 | 类型 | 成员 |
|---|---|---|
| IDC_EDIT_OPER1 | double | m_dOperator1 |
| IDC_EDIT_OPER2 | double | m_dOperator2 |

4.编写 CChap9DlgDlg::OnInitDialog()函数。

```
BOOL CChap9DlgDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    .....
    // TODO: Add extra initialization here
    CComboBox*  pComboBox=(CComboBox *)GetDlgItem(IDC_COMBO_OPERATOR);
    pComboBox->SetCurSel(0);

    return TRUE;  // return TRUE  unless you set the focus to a control
}
```

5. 编写 CChap9DlgDlg::OnCompute()函数。

```
void CChap9DlgDlg::OnCompute()
{
    //获得数据
    UpdateData();
    CComboBox*  pComboBox=(CComboBox *)GetDlgItem(IDC_COMBO_OPERATOR);
    int             nIndex=pComboBox->GetCurSel();
    char        tbuf[80];
    int             ret=pComboBox->GetLBText(nIndex, tbuf);
    ASSERT(ret!=CB_ERR);

    //计算
    double          dResult;
    switch(tbuf[0])
    {
    case '+':
        dResult=m_dOperator1+m_dOperator2;
        break;
    case '-':
        dResult=m_dOperator1-m_dOperator2;
        break;
    case '*':
        dResult=m_dOperator1*m_dOperator2;
        break;
    case '/':
        if(fabs(m_dOperator2)>1e-16)
        {
            dResult=m_dOperator1/m_dOperator2;
        }
        else
```

```
            }
                AfxMessageBox("Divisor should not be zero!");
            }
            break;
        default:
            break;
        }

        //输出到 IDC_EDIT_RESULT 编辑框中
        sprintf(tbuf,"%f",dResult);
        SetDlgItemText(IDC_EDIT_RESULT,tbuf);
    }
```

并在 Chap9DlgDlg.cpp 的顶端添加下列头文件：

```
#include <math.h>
```

6.创建和运行应用程序。

在对话框中输入两个操作数，选择运算符号，接着单击 Compute 按钮。计算结果将出现在 Result 编辑框中(图 9.26)。

## 9.9 本章小结

本章以丰富的实例介绍了对话框、对话框条以及基于对话框的应用程序。通过学习本章，读者将了解到对话框的分类、CDialog 类，理解对话框的数据交换(DDX)和数据验证(DDV)机制，掌握系统预制对话框的调用方法，学会模式对话框、无模式对话框、属性对话框、可停靠对话框控件以及基于对话框的应用程序的制作方法。

# 第十章　切分窗口与多视

本章要点

- 切分窗口与多视选择
- 动态切分窗口与静态切分窗口
- 单视类的 SDI 动态切分窗口示例
- 多视类的 SDI 静态切分窗口示例
- MDI 中的切分窗口示例

在迄今为止我们所遇到的例子中，与文档相连的视只有一个。如果读者使用过 Word for Windows 的话，就会知道对文档的各个部分同时开两个窗口是非常方便的。这两个窗口可以都是通常的视，或者一个窗口包含的是页布局视，而另一个则包含的是外形视。应用框架有若干种方式来表示多视，这些方式包括切分窗口和 MDI 子窗口。在本章中，我们将对这两种表示方式进行学习，并且会发现在这两种表示方式中，创建同一视类的多个视对象都是比较容易的（通常的视），而要在同一应用程序中使用两个或更多的视类（比如上面提到的外形视和页布局视），则相对来说要困难一些。

## 10.1　切分窗口

切分窗口看上去像是一种特殊的框架窗口，在各个窗片中包含了若干个视。应用程序在创建的时候就可对窗口进行划分，用户也可以通过选择菜单命令或拖动切分窗口滚动条中的切分框对窗口进行划分。当窗口被切分之后，用户可通过鼠标移动窗棱或边框来调整窗片的相对尺寸。切分窗口既适用于 SDI 应用程序，也适用于 MDI 应用程序。

切分窗口是通过 CSplitterWnd 类来表示的。对 Windows 来说，CSplitterWnd 对象是一个真正的窗口，它完全占据了框架窗口（CFrameWnd 或 CMDIChildWnd）的用户区域，而视窗口则占据了切分窗口的窗片区域。切分窗口并不参与命令传递机制，（窗片中）活动着的视窗口从逻辑上来看被直接连接到了它的框架窗口中。

## 10.2　多视选择

当把多视表示方式与应用模式相结合时，就可以有许多排列组合方式。下面列出的就是其中一些可能的组合方式：

SDI 应用，带切分窗口，只有一个视类——本章中的第一个例子就属于这种情况。其中

每一个切分窗片都可以被滚动到文档的不同部分。程序员可以决定所包含的水平和竖直窗片的最大数目，而用户则可以在运行中对窗口进行切分。

SDI应用，带切分窗口，具有多个视类——本章的第二个例子说明的正是这种情形。程序员可决定窗片的数目以及视的次序，而用户可在运行中改变窗片的尺寸。

MDI应用，不带切分窗口，只有一个视类——我们所见到的标准MDI应用程序都属于这种情况，用户可以通过选择New Windows菜单项为已经打开的文档再创建一个新的子窗口。

MDI应用，不带切分窗口，具有多个视类——只需对标准MDI应用程序做很小的改动就可以允许使用多视。

MDI应用，带切分子窗口——本章的第三个应用程序即属于这种情况。

## 10.3 动态和静态切分窗口

动态切分窗口允许用户在任何时候都可对窗口进行切分，既可以通过选择菜单项来对窗口进行切分，也可以通过拖动滚动条中的切分框对窗口进行切分。动态切分窗口中的窗片通常使用的是同一个视类。当切分窗口被创建时，左上窗片通常被初始化成一个特殊的视。在动态切分窗口中，滚动条在视中是被共用的。例如，在只进行了一次水平切分的窗口中，底滚动条同时控制着上、下两个视。

对于静态切分窗口，当窗口第一次被创建时，窗片就已经被切分好了，用户可以移动窗棱，但此时已不可能对窗口进行合并或再划分。静态切分窗口允许使用多个视类，并且可在创建时对这些视类进行配置。在静态切分窗口中，每个窗片都有自己的滚动条。




读者或许已经注意到了ClassWizard Add Class对话框中的“切分”类类型选项，如果选择了该选项，那么就得到了一个子窗口类，它是由CMDIChildWnd派生出来的，并且具有动态切分的性能。该类只能被用在MDI应用中，如果需要一个SDI切分窗口，则必须遵照本章例子中所采取的一些步骤。

CSplitterWnd::Create()函数用于创建动态的切分窗口，其原型说明如下：

```
BOOL Create (CWnd * pParentWnd, int nMaxRows, int nMaxCols,
             SIZE sizeMin, CCreatContext * pContext, DWORD dwStyle = WS_CHILD|
             WS_VISIBCLE | WS_HSCROLL | WS_VSCROLL |SPLS_DYNAMIC_SPLIT,
             UINT nID = AFX_IDW_PANE_FIRST);
```

建立动态的切分窗口，首先调用构造函数，然后调用Create成员函数。通过下列的步骤，可以将CSplitterWnd嵌入到父类CFrameWnd或CMDIChildWnd对象中。

(1) 将CSplitterWnd成员变量嵌入父类框架。

(2) 替换父类框架的OnCreateClient()成员函数。

(3) 从重载的OnCreateClient()内调用CSplitterWnd的构造函数：Create()成员函数。

当用户在一父类框架中建立一分割窗口时，把父类框架的pContext参数传递给分割窗

口。否则，这个参数将为 NULL。

动态切分窗口最初的行高度和列宽度是由参数 sizeMin 设定的。这些下限值，将用于确定区域的大小。当不能完整地显示所需的内容，可以通过 SetRowInfo()和 SetColumnInfo()成员函数重新设定。

函数调用成功时返回非 0 值，否则返回 0。

```
CSplitterWnd::CreateStatic
    BOOL CreateStatic(CWnd * pParentWnd, int nRows, int nCols,
                      DWORD dwStyle = WS_CHILD|WS_VISIBLE,
                      UINT nID = AFX_IDW_PANE_FIRST);
```

建立一静态分割窗口，首先调用构造函数，然后调用 CreateStatic 成员函数。通过以下步骤，CSplitterWnd 被嵌入到父类 CFrameWnd 或 CMDIChildWnd 对象中：

(1) 把 CSplitterWnd 成员变量嵌入父类框架中。

(2) 重载父类框架的 OnCreateClient 成员函数。

(3) 从重载的 OnCreateClient 内调用 CSplitterWnd 构造函数及 CreateStatic 成员函数。

静态切分窗口中的区域数目是固定的，而且各区域经常属于不同的类。

当用户建立一静态分割窗口时，应当同时建立窗口内的所有区域。CreateView()成员函数通常用于这个目的，但是用户还是必须建立其他的不可显示的类。

静态分割窗口的初始化行高，及列宽为 0。这些限定值用于确定它是否足够大，以便完整地显示所需内容。它们可以通过调用成员函数 SetRowInfo()和 SetColumnInfo()加以调整。

为了使一静态分割窗口拥有滚动条，可以把 WS_HSCROLL 和 WS_VSCROLL 风格加到 dwStyle 上。

函数如果操作成功，则返回一非 0 值，否则返回 0。

## 10.4　单视类的 SDI 动态切分窗口示例

在本例中，用户可以动态地将视窗口切分为两个或四个窗片。制作时，将用到 Callery SplitterWnd 组件。它可以为用户添加 CSplitterWnd 类。

### 试一试

1.应用 MFC AppWizard(exe) 创建一个新项目 Chap10A。在第 1 步中选择 Single document。

创建的类：

```
Application: CChap10AApp in Chap10A.h and Chap10A.cpp
Frame: CMainFrame in MainFrm.h and MainFrm.cpp
Document: CChap10ADoc in Chap10ADoc.h and Chap10ADoc.cpp
View: CChap10AView in Chap10AView.h and Chap10AView.cpp
```

特征：

```
+ Initial toolbar in main frame
+ Initial status bar in main frame
+ Printing and Print Preview support in view
+ 3D Controls
+ Uses shared DLL implementation (MFC42.DLL)
```

2.调用 Project \ Add To Project \ Components and Controls...菜单，弹出 Components and Controls Gallery 对话框。双击 Developer Studio Components 文件夹，再双击 Split Bars，弹出 Microsoft Developer Studio 消息框，单击 OK 按钮。接着，弹出 Split Bar 对话框，在 Type of split bar 下面单击 Both 单选按钮，再单击 OK 按钮。Split Bars 组件为我们添加的代码为：

在 CMainFrame 中添加了对象 m_WndSplitter 和虚函数：OnCreateClient：

```
protected:
    CSplitterWnd m_wndSplitter;
    virtual BOOL OnCreateClient(LPCREATESTRUCT lpcs,
                                CCreateContext* pContext);
```

在 CMainFrame::OnCreateClient 中添加了下列代码：

```
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT lpcs,
                                CCreateContext* pContext)
{
    // CG: The following block was added by the Split Bars component.
    {
        if (! m_wndSplitter.Create(this,
                2, 2,        // TODO: adjust the number of rows, columns
                CSize(10, 10), // TODO: adjust the minimum pane size
                pContext))
        {
            TRACE0("Failed to create split bar ");
            return FALSE;    // failed to create
        }

        return TRUE;
    }
}
```

3.从 ResourceView 中修改 IDR_MAINFRAME 菜单，如图 10.1 所示。

| 菜单 | 菜单项 | 提示 Prompt | 命令 ID |
|---|---|---|---|
| &Window | S&plit | Split a window | ID_WINDOW_SPLIT |

4.编写 void CChap10AView::OnDraw 函数。

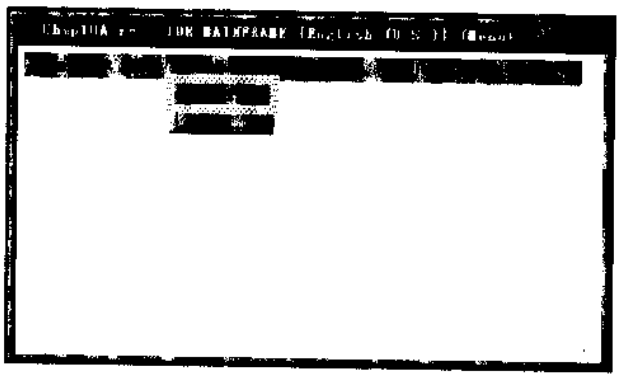


图 10.1　IDR_MAINFRAME 菜单

```
void CChap10AView::OnDraw(CDC * pDC)
{
    pDC->TextOut(10,30,"Choose Window\ \ Split to split a window!");
}
```

5.创建和运行应用程序。

选择 Window \ Split 菜单项剖分窗口(图 10.2)。

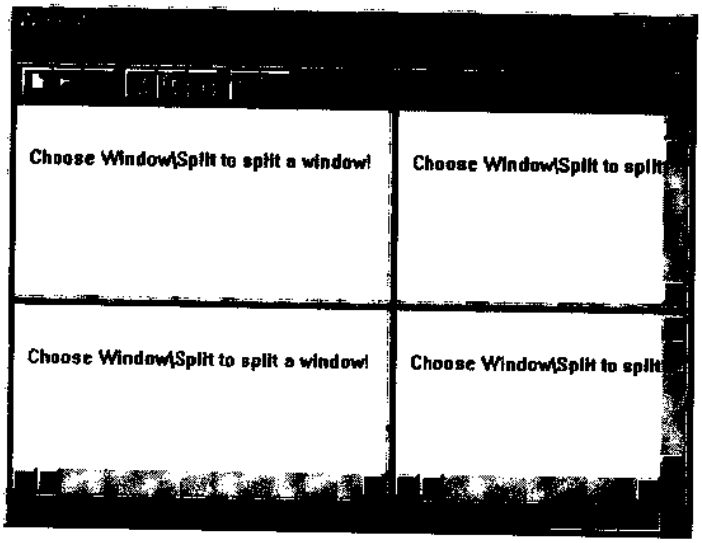



图 10.2　单视单文档动态切分窗口

## 10.5　双视类的 SDI 静态切分示例

Chap10B 项目将生成一个静态切分窗口。两个视中显示的状态各不相同,一个是文本,

另一个是文本的 16 进制 ASCII 代码。

## 试一试

1.应用 MFC AppWizard(exe) 创建一个新项目 Chap10B。在第 1 步中选择 Single document。

在第 6 步中修改视类的名称。从列表框中选择 CChap10BView。接着,将 Class name 改为 CTextView,Header file 改为 TextView.h,Implementation file 改为 TextView.cpp,如图 10.3 所示。

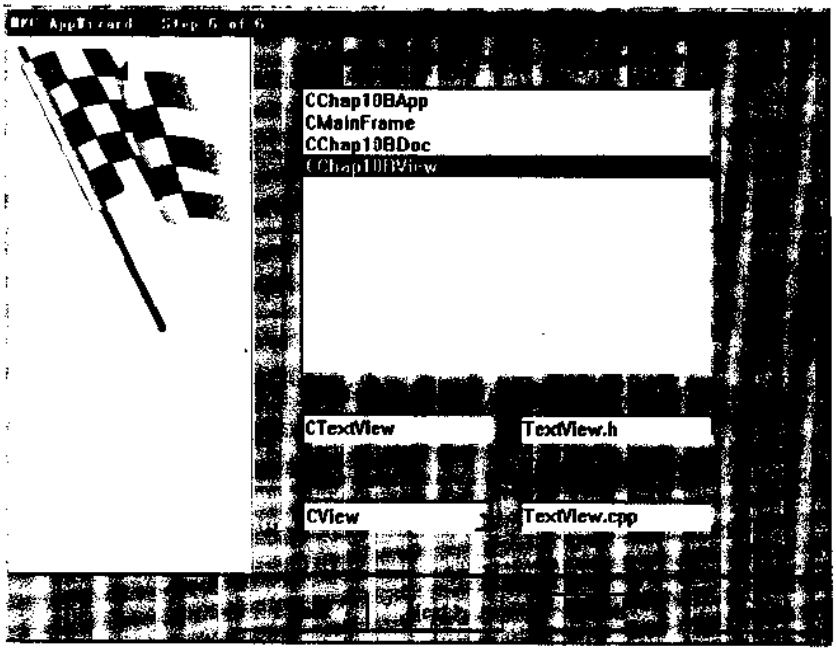



图 10.3 MFC AppWizard - step6 of 6 对话框

创建的类:

```
Application: CChap10BApp in Chap10B.h and Chap10B.cpp
Frame: CMainFrame in MainFrm.h and MainFrm.cpp
Document: CChap10BDoc in Chap10BDoc.h and Chap10BDoc.cpp
View: CTextView in TextView.h and TextView.cpp
```

特征:

```
+ Initial toolbar in main frame
+ Initial status bar in main frame
+ Printing and Print Preview support in view
+ 3D Controls
+ Uses shared DLL implementation (MFC42.DLL)
```

2. 调用 Project \ Add To Project \ Components and Controls... 菜单，弹出 Components and Controls Gallery 对话框。双击 Developer Studio Components 文件夹，再双击 Split Bars，弹出 Microsoft Developer Studio 消息框，单击 OK 按钮。接着弹出 Split Bar 对话框，在 Type of split bar 下面选择 Both 单选钮，再单击 OK 按钮。

3. 修改 CMainFrame::OnCreateClient 函数。

```
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT lpcs,
                                CCreateContext * pContext)
{
    BOOL ret = m_wndSplitter.CreateStatic(this,2,1);
    ret | = m_wndSplitter.CreateView(0,0,RUNTIME_CLASS(CTextView),
                                     CSize(100,100),pContext);
    ret | = m_wndSplitter.CreateView(1,0,RUNTIME_CLASS(CHexView),
                                     CSize(100,100),pContext);
    return ret;
}
```

4. 在 MainFrm.cpp 的顶部添加：

```
#include "Chap10Bdoc.h"
#include "TextView.h"
#include "HexView.h"
```

5. 建立一个新类 CHexView。利用下列信息使用 ClassWizard 建立一个新类：

类名：CHexView

基类：CView

文件：HexView.cpp

6. 在 CChap10BView 中增加函数。

```
// Attributes
public:
    CChap10BDoc * GetDocument();
        { return (CChap10BDoc * )m_pDocument; }
```

7. 编写 CHexView::OnDraw 函数。

```
void CHexView::OnDraw(CDC * pDC)
{
    CChap10BDoc * pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    int l = strlen(pDoc - > m_string);
    for(int i = 0;i < l;i + + )
    {
        char tbuf[10];
```

```
            //获取字符的 ASCII 码,并以十六进制形式写入 tbuf 中。
            int n = pDoc->m_string[i] & 0x00ff;
            sprintf(tbuf,"%2x",n);
            if(i<20)
                pDC->TextOut(10+20*i,10,tbuf);
            else
                pDC->TextOut(10+20*(i-20),40,tbuf);
        }
    }
```

8. 在 CHewView.cpp 的顶端添加:

```
#include "Chap10BDoc.h"
#include "HexView.h"
```

9. 编写 CTextView::OnDraw 函数。

```
void CTextView::OnDraw(CDC* pDC)
{
    CChap10BDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    pDC->TextOut(10,10,pDoc->m_string);
}
```

10. 在 CChap10BDoc 中增加变量。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| char | m_string[50] | public |

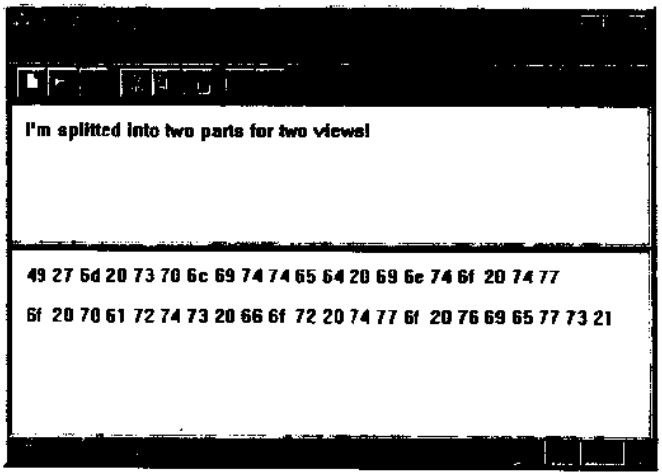



图 10.4 具有双视类的 SDI 应用程序

11.编写 CChap10BDoc::CChap10BDoc 构造函数。

```
CChap10BDoc::CChap10BDoc()
{
    //初始化待显示的字符串。
    strcpy(m_string,"I'm splitted into two parts for two views!");
}
```

12.创建和运行应用程序。

运行程序后将生成如图 10.4 所示的应用程序界面。

## 10.6　MDI 中的切分窗口示例

或许读者已经考虑到了,如果某两个窗口中的视类极其相似,如 CAD 系统中的三视图,各视图仅相差一个视点时,为每个窗口创建一个视类将非常烦琐。此时,若仅定义一个视类,并为各窗口视对象用一个标志值来区分视,从而显示不同的内容。

本例将在 MDI 应用中用模板添加动态切分窗口和静态切分窗口,并将视对象的标志值显示在窗口中。

### 试一试

1.应用 MFC AppWizard(exe) 创建一个新项目 Chap10C。在第 6 步中修改子框架类的名称。从列表框中选择 CChildFrame。接着,将 Class name 改为 CDynSpltFrame,Header file 改为 DynSpltFrm.h,Implementation file 改为 DynSpltFrm.cpp,如图 10.5 所示。

创建的类:

```
Application: CChap10CApp in Chap10C.h and Chap10C.cpp
Frame: CMainFrame in MainFrm.h and MainFrm.cpp
MDIChildFrame: CDynSpltFrame in DynSpltFrm.h and DynSpltFrm.cpp
Document: CChap10CDoc in Chap10CDoc.h and Chap10CDoc.cpp
View: CChap10CView in Chap10CView.h and Chap10CView.cpp
```

特征:

```
+ Initial toolbar in main frame
+ Initial status bar in main frame
+ Printing and Print Preview support in view
+ 3D Controls
+ Uses shared DLL implementation (MFC42.DLL)
```

2. 调用 Project \ Add To Project \ Components and Controls... 菜单,弹出 Components and Controls Gallery 对话框。双击 Developer Studio Components 文件夹,再双击 Split Bars,弹出

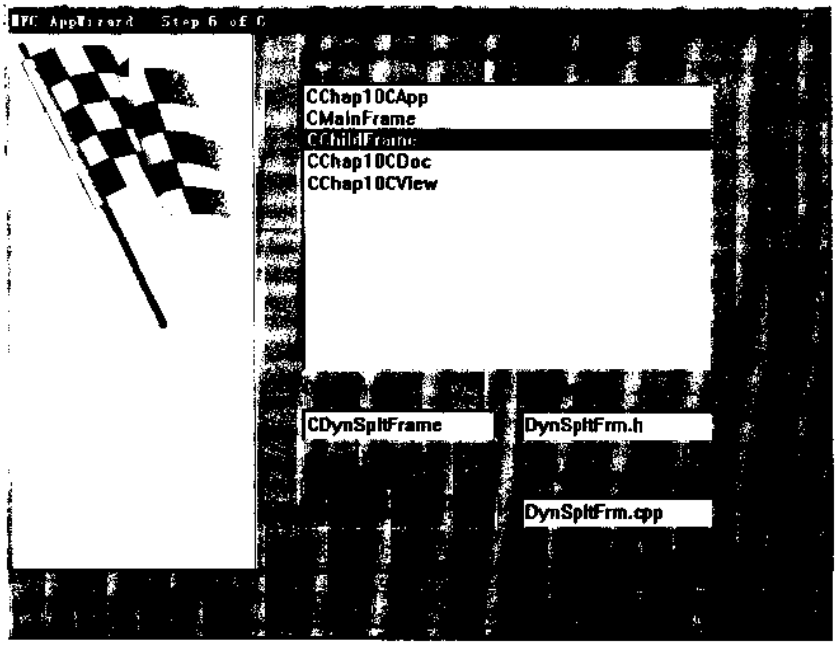



图 10.5 MFC AppWizard_Step 6 of 6 对话框

Microsoft Developer Studio 消息框，单击 OK 按钮。接着弹出 Split Bar 对话框，在 Type of split bar 下面选择 Both 单选按钮，再单击 OK 按钮。

3. 从 ResourceView 中修改 IDR_CHP10TYPE 菜单，如图 10.6 所示。

| 菜单 | 菜单项 | 提示 Prompt | 命令 ID |
|---|---|---|---|
| &Window | s&plit | Split the active window into panes \ Split | ID_WINDOW_SPLIT |

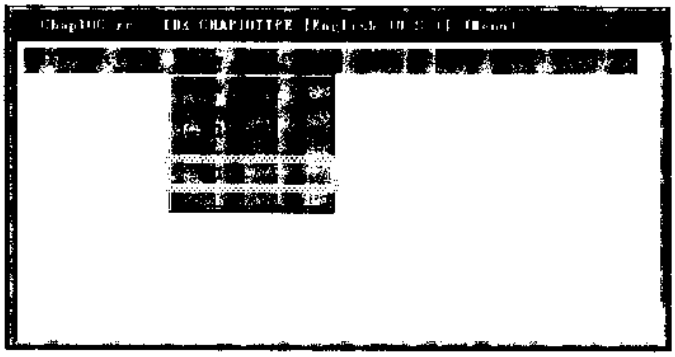


图 10.6 IDR_CHP10TYPE 菜单

4. 建立一个新类 CStaticSplitFrm。利用下列信息使用 ClassWizard 建立一个新类：

类名：CStaticSpltFrm

基类：CMDIWnd

文件：StaticSpltFrm.cpp

5.从菜单中调用 Project \ Add To Project \ Components and Controls...菜单项，弹出 Components and Controls Gallery 对话框。双击 Developer Studio Components 文件夹，再双击 Split Bars，弹出 Microsoft Developer Studio 消息框，单击 OK 按钮。接着弹出 Split Bar 对话框，在 Type of split bar 下面单击 CStaticSpltFrm 单选按钮，再单击 OK 按钮。

6.编写 CStaticSpltFrm::OnCreateClient 函数。

```
BOOL CStaticSpltFrm::OnCreateClient(LPCREATESTRUCT lpcs,
                                    CCreateContext * pContext)
{
     BOOL ret = m_wndSplitter.CreateStatic(this,2,1);
     ret |= m_wndSplitter.CreateView(0,0,RUNTIME_CLASS(CChap10CView),
                                     CSize(100,100),pContext);
     ret |= m_wndSplitter.CreateView(1,0,RUNTIME_CLASS(CChap10CView),
                                     CSize(100,100),pContext);
     return ret;
}
```

在 StaticSpltFrm.cpp 的顶端添加：

```
#include "StaticSpltFrm.h"
#include "Chap10CDoc.h"
#include "Chap10CView.h"
```

7. 编写 CChap10CApp::InitInstance 函数。

```
BOOL CChap10CApp::InitInstance()
{
     ......
     CMultiDocTemplate * pDocTemplate;
     pDocTemplate = new CMultiDocTemplate(
          IDR_CHAP10TYPE,
          RUNTIME_CLASS(CChap10CDoc),
          RUNTIME_CLASS(CDynSplitFrame), // custom MDI child frame
          RUNTIME_CLASS(CChap10CView));
     AddDocTemplate(pDocTemplate);

     //定义并增加新的文档视框架模板
     pDocTemplate = new CMultiDocTemplate(
          IDR_CHAP10TYPE2,
          RUNTIME_CLASS(CChap10CDoc),
          RUNTIME_CLASS(CStaticSpltFrm), // custom MDI child frame
          RUNTIME_CLASS(CChap10CView));
     AddDocTemplate(pDocTemplate);
```

```
    // create main MDI Frame window
    CMainFrame * pMainFrame = new CMainFrame;
    if (! pMainFrame - > LoadFrame(IDR_MAINFRAME))
        return FALSE;
    m_pMainWnd = pMainFrame;
    ......
}
```

在 Chap10C.cpp 的顶端添加：

```
#include "DynSpltFrm.h"
#include "StaticSpltFrm.h"
#include "Chap10CDoc.h"
#include "Chap10CView.h"
```

8.在 CChap10CView 中增加变量。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| int | m_nViewNo | protected |

从 ResourceView 中修改 IDR_CHAP10TYPE 字符串，同时添加 IDR_CHAP10TYPE2 字符串。

| ID 值 | Caption 值 |
|---|---|
| IDR_CHAP10TYPE | \ nChapio \ nDynamic Split Window \ n \ n \ nChap10C Document \ nChap10C Document |
| IDR_CHAP10TYPE2 | \ nChapio \ nStatic Split Window \ n \ n \ nChap10C Document \ nChap10C Document |

9.编写 CChap10CView::CChap10CView 构造函数。

```
static int g_nRecordNo = 0;
CChap10CView::CChap10CView()
{
    //初始化视对象时,记录视对象编号。
    m_nViewNo = g_nRecordNo;
    g_nRecordNo + + ;
}
```

10. 编写 CChap10CView::OnDraw 函数。

```
void CChap10CView::OnDraw(CDC * pDC)
{
    CChap10CDoc * pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    char tbuf[80];
    //显示视对象编号,以表现不同的视窗口对应不同的视对象。
    sprintf(tbuf,"This is No. %d View!",m_nViewNo);
    pDC - > TextOut(10,10,tbuf);
```

11. 创建和运行应用程序。

启动程序时会出现如图 10.7 所示对话框，单击“确定”按钮。用 Window \ Split 菜单项切分子窗口或者用 Windows \ New Window 菜单项添加新窗口，注意子窗口中视对象编号的变化(图 10.8)。利用此编号或类似变量便可区分视对象。

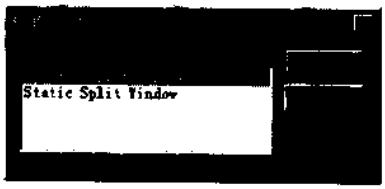



图 10.7　新建对话框

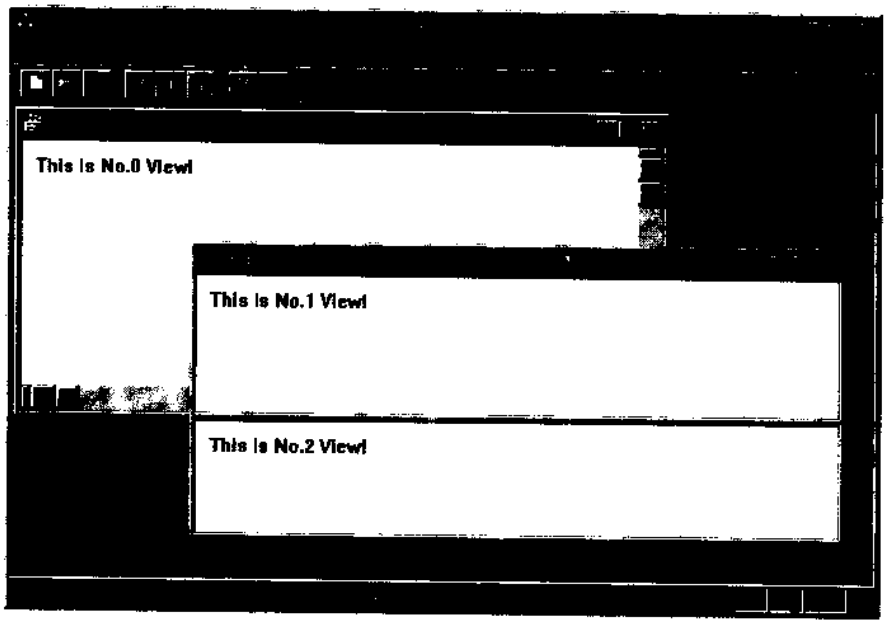



图 10.8　多文档界面下的窗口剖分与多视

## 10.7　本章小结

本章介绍了切分窗口与视的关系。通过学习本章，读者学习到了静态和动态的切分窗口，掌握了在 SDI 和 MDI 中运用单视类或多视类对窗口进行动态切分或静态切分的技术。

# 第十一章　文档与视的分离

**本章要点**

- 文档与视的相互关系及其通信方式
- 简单的文档－视应用
- CFormView 类
- 文档、视程序示例

下面我们来介绍文档和视之间的相互关系。当命令消息向文档对象和视对象发送时，实际上，我们先前已经看到了它们之间的这种相互作用关系。本章将介绍文档是如何保存应用程序的数据的，而视又是如何向用户展示这些数据的，并且还将介绍在应用程序的执行过程中，文档和视又是如何相互对话的。

## 11.1　文档－视之间的相互作用函数

读者已经了解到，文档对象是用来保存数据的，而视对象则是用来显示数据、并允许对数据进行编辑的。SDI 应用程序的文档类是由 CDocument 类派生出来的，它可以包含一个或多个视类，而这些视类最终都是由 CView 类派生的。在文档、视以及应用框架的其余部分之间存在着非常复杂的相互作用过程。为了了解这一过程，我们首先学习有关文档类和视类的四个非常重要的成员函数。其中有两个为虚函数，需要在派生类中对它们进行覆盖；而其余两个不是虚函数，需要在派生类中对它们进行调用。下面就介绍这四个函数。

### 11.1.1　CView 类的 GetDocument()函数

视对象只与一个文档对象相联系，它的 GetDocument()函数允许应用程序在视中得到与之相联系的文档。如果用户在编辑控件中输入了新的数据，那么视对象接收到了一条消息。此时，视必须通知文档对象对其内部数据进行相应的更新。GetDocument()函数返回的是指向文档的指针，利用它我们就可以对文档类成员函数及公共数据成员进行访问。

GetDocument()函数的原型说明如下：

```
CDocument * CView::GetDocument( ) const;
```

调用这个函数取得视窗口文件的指针。这将使用户程序能够调用文件成员函数，返回一个指向与视窗口相联的 CDocument 对象的指针。如果没有文件与视窗口相连，则返回 NULL。

当 AppWizard 产生 CView 的派生类时，它同时也生成一个保证文档类型的 GetDoument() 函数，它返回的是指向派生文档类（注意不是 CDocument 类）的指针。该函数是一个内联函数，它类似于如下的形式：

```
CMyDoc *   GetDocument()
{
    return(CMyDoc *)m_pDocument;
}
```

当编译器在视类代码中遇到对 GetDocument() 函数的调用时，它调用的是派生视类（不是 CView 类）中的该函数。因此，我们不必将它所返回的指针强制转换为指向派生文档类的指针。

CView 类中的 GetDocument() 函数所返回的是指向 CDocument 对象的指针。

### 11.1.2　CDocument 类的 UpdateAllViews() 函数

如果文档数据发生了改变，那么必须通知所有的视，以便它们能够更新所显示的数据。如果在派生文档类的成员函数中调用 UpdateAllView() 函数，那么它的第一个参数 pSender 应为 NULL。如果在派生视类的成员函数中调用该函数，则应该以如下方式将 pSender 参数设置为当前视：

```
GetDocument( ) - > UpdateAllView(this);
```

其中非空参数就使得应用框架不再通知当前的视，因为假定当前视已经自己进行了更新。该函数还有一些提示参数，可以利用它们给视提供一些特殊的与应用程序有关的消息，以便视能够决定究竟哪些部分应该更新，这是对该函数的更高级的用法。

UpdateAllViews() 函数的原型说明如下：

```
void CDocument::UpdateAllViews(CView * pSender, LPARAM lHint = 0L
            CObject * pHint = NULL);
```

文件被修改后调用本函数。在调用 SetModifiedFlag 成员函数后，应该调用本函数。UpdateAllViews() 将通知链接在文件上的所有视窗口（除 pSender 参数指定的视窗口之外），文件已被修改。通常，在用户通过视窗口改变文件后，程序员从视窗口类中调用本函数。

本函数将为除了发送视窗口之外的所有文件的视窗口调用 OnUpdate 成员函数，所传递的参数为 pHint 和 lHint。使用这些参数向视窗口传递有关对文件进行修改的消息。程序员可以使用 lHint 对消息编码，或定义一个 CObject 派生类对象来存储有关修改的消息，然后使用 pHint 来传递类对象。在 CView 派生类中可以重载 OnUpdate 成员函数，以优化对视窗口基于传递消息的显式修改。

### 11.1.3　CView 类的 OnUpdate() 函数

CView 类的 OnUpdate( ) 函数是一个虚函数。在应用程序调用了 CDocument 类的 UpdateAllViews() 函数时，应用框架就会相应地调用该函数。我们也可以直接在派生视类中

调用它。通常派生视类的 OnUpdate()函数要访问文档,读取文档的数据,然后再更新视的数据成员或控件,以便反映出文档的变化。此外,可以利用 OnUpdate()函数使视的某部分无效,以触发视的 OnDraw()函数,利用文档数据重新绘制窗口。OnUpdate()函数类似于下面的形式:

```
void CMyView::OnUpdate(CView* pSender,LPARAM lHint ,CObject * pHint)
{
    CMyDocument * pMyDoc = GetDocument();
    CString lastName = pMyDoc - > GetLastName();
    m_pNameStatic - > SetWindowText(lastName);//m_pNameStatic is
                                    //a CMyView data member
}
```

其中的提示信息是由 UpdateAllViews()函数直接传递过来的。缺省的 OnUpdate()函数使得整个窗口矩形无效。在重载该函数时,可以利用提示信息来定义一个较小的无效矩形。

当 CDocument 类的 UpdateAllViews()函数被调用时,如果 pSender 参数指向了某个特定的视对象,那么除了被指定的视之外,文档的其它所有视的 OnUpdate()函数都会被调用。

OnUpdate()函数的原型说明如下:

```
protected virtual void CView::OnUpdate(CView * pSender, LPARAM(lHint, CObject
* pHint));
```

在视窗口的文件被修正以后,框架调用此函数。这个函数是被 CDocument::UpdateAllView 调用的,它使得视窗口更新其显示内容以反映文件的变化。OnInitialUpdate 函数的缺省值调用也将产生 OnUpdate 函数调用。缺省方式引用将使视窗口的整个客户区无效。当视对象接收 WM_PAINT 消息时,便给它作上标记。如果用户仅仅想更新视窗口中与文件中被修改部分相对应的区域,可以重载这个函数。为了完成这一功能,用户程序必须使用暗示参数,传递有关修正文件的消息。

可以使用 lHint 定义特殊的暗示值。典型的暗示值是 bitmask 的枚举类型的。文件传递这些值中的一个。可以使用 pHint 从 CObject 类派生出的 Hint 类,以便文件把一个指针传递到暗示对象。当重载 OnUpdate()函数时,应当使用 CObject::IsKindOf 成员函数决定对象在运行时的类型。

通常,不能通过 OnUpdate 成员函数直接进行任何绘制工作,而应该确定对待更新矩形区域的设备坐标描述,然后再把该矩形传递给 CWnd::InvalidateRect()。当下一次接收到 WM_PAINT 消息时,就开始重新形成视窗口的客户区显示。

如果 lHint 是 0,pHint 是 Null,那么文件传送了一个类更新通知。如果视窗口接收到类更新通知,或者它不能解码出暗示参数,则整个客户区将处于无效状态。

### 11.1.4 CView 的 OnInitialUpdate()函数

当应用程序被启动时,或者当用户选择了 New 菜单项命令时,或者用户选择了 Open 命令时,CView 的 OnInitialUpdate()虚函数都会被自动调用。CView::OnInitialUpdate()函数除了

调用 OnUpdate()函数之外，不做其它任何事情。如果需要在派生视类中重载 OnInitialUpdate()函数，则一定要在其中调用基类的 OnInitialUpdate()函数，或者调用派生类的 OnUpdate()函数。可以利用派生类的 OnInitialUpdate()函数初始化视对象。当应用程序启动时，应用程序框架在调用了 OnCreate()函数之后（如果映射了 OnCreate()函数的话），会立刻调用 OnInitialUpdate()函数。OnCreate()函数只能被调用一次，而 OnInitialUpdate()函数可以被调用多次。

OnInitialUpdate()函数的原型说明如下：

```
protected virtual void CView::OnInitialUpdate();
```

在视窗口第一次与文件相连以后，在它被最初显示以前，框架调用此成员函数。这个函数的缺省方式引用将以不带任何提示消息（也就是说 lHint 参数为 0，pHint 参数为 NULL）的方式调用 OnUpdate 成员函数。重载此函数以执行所需要文件消息的初始化。例如：如果用户的应用程序有一个固定长度的文件，则就可以根据文件的长度，使用这个函数设置视窗口的滚动最大范围。如果用户的应用程序支持变化长度的文件，那么在每一次改变文件长度时，都会调用 OnUpdate 成员函数对视窗口的滚动范围作相应的更新。

## 11.2　最简单的文档、视应用

如果我们不打算让文档具有多视，只打算利用框架对文件的支持，这时就可以不使用 UpdateAllViews()和 OnUpdate()函数，而只需按照以下的步骤来创建该应用程序。

1.在 AppWizard 产生的派生文档类的头文件中定义 public 型的文档的数据成员，以存储应用程序中的数据。也可以将派生视类说明成文档类的 friend。

2.在派生视类中，重载 OnInitialUpdate()虚成员函数。当文档数据被初始化或者被从磁盘中读出时，应用框架就会自动调用 OnInitialUpdate()函数。OnInitialUpdate()函数对视进行更新，以便反映出当前的文档数据。

3.在派生视类中，用窗口消息控制函数和命令消息控制函数直接更新文档数据成员，利用 GetDocument()函数访问文档对象。

下面列出的是在这一简单的文档－视架构中事件发生的顺序：

应用启动CMyDocument 对象被创建

　　CMyView 对象被创建

　　视窗口被创建

　　CMyView::OnCreate()被调用(如果被映射了的话)

　　CMyView::OnInitialUpdate()被调用

　　视对象被初始化

　　视窗口被显示

用户编辑数据 CMyView 函数对 CMyDocument 数据成员进行更新

用户退出应用 CMyView 对象被删除

CMyDoument()对象被删除

## 11.3 CFormView 类

CFormView 类是一个非常有用的视类,具有许多无模式对话框的特点。像 CDialog 派生类一样,CFormView 的派生类也与相应的对话框资源相联系。在对话框资源中定义了格式视的框架特性,并且列举出了它所包含的所有的控件。CFormView 类也支持 DDX 和 DDV,它们和我们在第九章对话框中所见到的函数完全相同。

警告:当用 App Studio 为格式视创建对话框时,必须在 Dialog Properties 对话框中指定如下的项:

```
Style = Child
Border = None
Visible = unchecked
```

CFormView 对象可以直接接受来自于它的控件的通知消息,也可以接受来自于应用框架的命令消息。应用框架的这一命令处理能力使得 CFormView 同 CDialog 有明显的区别,同时也使得在应用框架的主菜单或工具条中对视进行控制变得较为容易。

CFormView 是从 CScrollView 派生出来的,而不是从 CDialog 派生来的。因此,我们不能假设它可支持 CDialog 的所有成员函数。CFormView 类并不包含 OnInitDialog()虚函数、OnOk()虚函数以及 OnCancel()虚函数,它也不调用 UpdateData()和 DDX 函数。我们必须在适当的时候,通常是在对控件通知消息或命令消息进行响应时,自己调用这些函数。尽管 CFormView 类并不是从 CDialog 类派生而来的,但它却是围绕着 Windows 的对话框创建的。因此,我们可以使用许多 CDialog 类的成员函数,如 GotoDlgCtrl()和 NextDlgCtrl()。只须将 CFormView 指针强制转换为 CDialog 指针即可。

下面的语句是从 CFormView 派生类的某个成员函数中提取出来的,它将输入定位到某个指定的控件。GetDlgItem()是一个 CWnd 的成员函数,它同样也被 CFormView 的派生类所继承。

```
((CDialog*)this)->GotoDlgCtrl(GetDlgItem(IDC_NAME));
```

也可以在 AppWizard 的 Classes 对话框中将 CFormView 指定为视类的基类。当我们选择 CFormView 时,AppWizard 会自动产生一个具有相应风格属性集的空对话框。下一步就利用 ClassWizard 来加进控件通知消息控制函数、命令消息控制函数和更新命令 UI 控制函数。当然,还可以定义一些数据成员和限制条件。

说明:如果想利用 ClassWizard 在 CFormView 派生类中加入菜单命令消息控制函数,则应该在选择了适当的菜单后,但必须在 AppStudio 中运行 ClassWizard。

## 11.4 文档与视的示例程序

本示例将对 Chap10C 进行修改。修改后的应用程序可以在动态窗口的任意一个视中输

入文字。当从菜单中选择 UpdateAllViews 选项时，相关的所有视都要被更新。此外，还将静态切分的一个视改为格式视，以便从格式视中输入文字。

本示例程序分两步完成，第一步我们先生成动态切分的视文档应用程序。

## 试一试

1. 打开 Chap10C 项目。

2. 从 ResourceView 中编辑 IDR_CHAP10TYPE 菜单，如图 11.1 所示。

| 菜单 | 菜单项 | 提示 Prompt | ID 值 |
|---|---|---|---|
| &View | &UpdateAllViews | Update all views | ID_VIEW_UPDATEALLVIEWS |

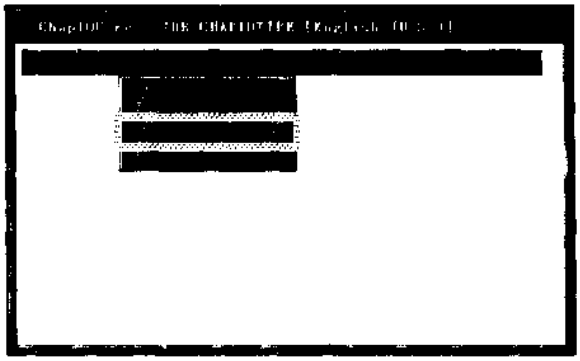


图 11.1　IDR_CHAP10TYPE 菜单

3. 在 CChap10CDoc 中增加变量。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| char | m_string[200] | public |
| int | m_nCurPos | public |
| CPoint | m_Position | public |

4. 利用 ClassWizard 在 CChap10CDoc 中增加消息处理函数

| 消息 | 函数 |
|---|---|
| ID_VIEW_UPDATEALLVIEWS | OnViewUpdateAllViews |

5. 编写 CChap10CDoc::CChap10CDoc 函数。

```
CChap10CDoc::CChap10CDoc()
{
    m_nCurPos = 0;
    m_string[0] = '\0';
    m_Position.x = 10;
```

```
    m_Position.y = 40;
}
```

6. 编写 CChap10CDoc::OnViewUpdateAllViews 函数。

```
void CChap10CDoc::OnViewUpdateAllViews()
{
    //设置字符串结束标志
    m_string[m_nCurPos] = '\0';
    UpdateAllViews(NULL);
}
```

7. 在 CChap10CView 中增加变量和函数。

| 变量类型 | 变量名称 | 访问权限 |
|---|---|---|
| CPoint | m_Position | protected |

| 函数类型 | 函数名称 | 访问权限 |
|---|---|---|
| void | MoveCursor(CPoint point) | protected |

8. 利用 ClassWizard 在 CChap10CView 中增加虚函数 OnUpdate,以及消息处理函数。

| 消息 | 函数 |
|---|---|
| WM_LBUTTONUP | OnLButtonUp |
| WM_CHAR | OnChar |

9. 编写 CChap10CView::CChap10CView 函数。

```
CChap10CView::CChap10CView()
{
    m_nViewNo = g_nRecordNo;
    g_nRecordNo++;

    m_Position.x = 40;
    m_Position.y = 40;
}
```

10. 编写 CChap10CView::OnDraw 函数。

```
void CChap10CView::OnDraw(CDC* pDC)
{
    CChap10CDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    //用白刷子清屏
    CBrush* pOldBrush = (CBrush*)pDC->SelectStockObject(WHITE_BRUSH);
```

```
    CRect rect;
    GetClientRect(rect);
    pDC->Rectangle(rect);
    pDC->SelectObject( pOldBrush );

    //设置 R2_NOTXORPEN 的绘图方式
    int nDrawMode = pDC->SetROP2(R2_NOTXORPEN);
    //绘制输入光标
    pDC->MoveTo(m_Position.x,m_Position.y);
    pDC->LineTo(m_Position.x,m_Position.y+16);
    //恢复先前的绘图方式
    pDC->SetROP2(nDrawMode);

    char tbuf[80];
    sprintf(tbuf,"This is No.%d View!",m_nViewNo);
    pDC->TextOut(10,10,tbuf);

    //输出文档中记录的文本
    pDC->TextOut(pDoc->m_Position.x,
                 pDoc->m_Position.y,
                 pDoc->m_string);
}
```

11. 编写 CChap10CView::OnLButtonUp 函数。

```
void CChap10CView::OnLButtonUp(UINT nFlags, CPoint point)
{
    //设置重新输入文本
    GetDocument()->m_nCurPos = 0;
    //设置重新输入文本的位置
    GetDocument()->m_Position.x = point.x+1;
    GetDocument()->m_Position.y = point.y;
    //设置当前文本输入位置
    MoveCursor(point);
}
```

12. 编写 CChap10CView::OnChar 函数。

```
void CChap10CView::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    char  tbuf[20];
    sprintf(tbuf,"%c",nChar);

    CDC *  pDC = GetDC();
    pDC->TextOut(m_Position.x+1,m_Position.y,tbuf);
```

```
        //获取输出字符串的大小范围
        CSize size = pDC->GetTextExtent(tbuf);
        CPoint      point = m_Position + CPoint(size.cx+1,0);
        MoveCursor(point);

        //获取文档指针
        CChap10CDoc * pDoc = GetDocument();
        //记录字符
        pDoc->m_string[pDoc->m_nCurPos] = nChar;
        //字符位置标记加1
        if(pDoc->m_nCurPos < 200 - 1)
        {
            pDoc->m_nCurPos++;
        }
        else
        {
            AfxMessageBox("This string is too long! \n"
                          "It should be less than 200 characters!");
        }
    }
```

13.编写 CChap10CView::OnUpdate 函数。

```
void CChap10CView::OnUpdate(CView * pSender, LPARAM lHint, CObject * pHint)
{
    CPoint      point;
    point.x = GetDocument()->m_Position.x - 1;
    point.y = GetDocument()->m_Position.y;
    MoveCursor(point);
    CDC *   pDC = GetDC();
    OnDraw(pDC);
}
```

14.编写 CChap10CView::MoveCursor 函数。

```
void CChap10CView::MoveCursor(CPoint point)
{
    CDC * pDC = GetDC();
    //设置R2_NOTXORPEN的绘图方式
    int nDrawMode = pDC->SetROP2(R2_NOTXORPEN);
    //覆盖先前的光标线
    pDC->MoveTo(m_Position.x,m_Position.y);
    pDC->LineTo(m_Position.x,m_Position.y+16);

    //绘制新的光标线
```

```
        m_Position = point;
        pDC - > MoveTo(m_Position.x,m_Position.y);
        pDC - > LineTo(m_Position.x,m_Position.y + 16);
        //恢复先前的绘图方式
        pDC - > SetROP2(nDrawMode);
    }
```

15.创建和运行应用程序。

关于动态切分的视文档程序已经完成。读者可以编译并运行它。将窗口动态切分后，输入字符串，接着选择 View \ UpdateAllViews 菜单项，注意各视的更新情况。

接下来，我们将把静态切分窗口中的一个视改为格式视 CFormView。用户可以在该视中输入文本。注意：在该窗口的普通视中也可以输入文本，而且在选择 View \ UpdateAllViews 后，格式视中的文本也会刷新。

## 试一试

1.打开 Chap10C 项目。

2.从 ResourceView 中插入一个对话框资源，如图 11.2 所示。

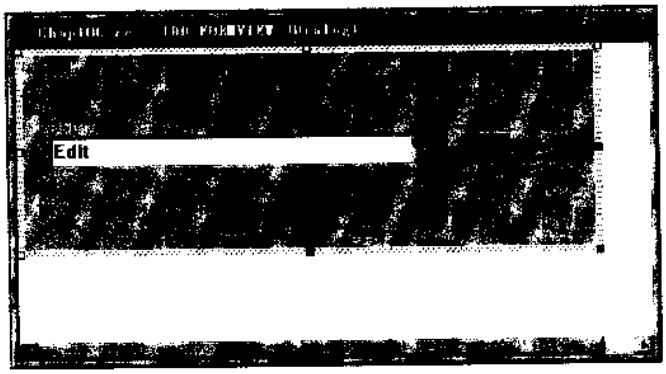



图 11.2　IDD_FORMVIEW 对话框

在 DialogProperties 对话框中将对话框的 ID 值改为 IDD_FORMVIEW。同时在 Style 中设置下列项：

```
Style = Child
Border = None
```

| 控件 | ID 值 | 属性 |
|---|---|---|
| Static 控件 | IDC_STATIC | Caption = Enter text |
| Edit 控件 | IDC_EDIT_TEXT | 默认设置 |
| Enter 按钮 | IDC_BTN_ENTER | Caption = &Enter |

3.建立一个新类 CEnterFormView。根据下列信息,利用 ClassWizard 建立一个新类:

类名:CEnterFormView

基类:EnterFormView.cpp

文件:EnterFormView.cpp

ID 值: IDD_FORMVIEW

4.利用 ClassWizard 在 CEnterFormView 中增加虚函数 OnUpdate,以及消息处理函数。

| 对象 | 消息 | 函数 |
|---|---|---|
| IDC_BTN_ENTER | BN_CLICKED | OnBtnEnter |

5. 在 CEnterFormView 中增加函数。

```
class CEnterFormView : public CFormView
{
protected:
    CEnterFormView();   // protected constructor used by dynamic creation
    DECLARE_DYNCREATE(CEnterFormView)

// Form Data
public:
    //{{AFX_DATA(CEnterFormView)
    enum { IDD = IDD_FORMVIEW };
        // NOTE: the ClassWizard will add data members here
    //}}AFX_DATA

// Attributes
public:
    CChap10CDoc* GetDocument()
    {
        return (CChap10CDoc *) m_pDocument;
    }
    .......
}
```

6.编写 CEnterFormView::OnBtnEnter 函数。

```
void CEnterFormView::OnBtnEnter()
{
    CChap10CDoc *       pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    //从 Edit 框中获取字符串
    GetDlgItemText(IDC_EDIT_TEXT,pDoc->m_string,200);
    pDoc->m_nCurPos = strlen(pDoc->m_string);
```

```
    //更新所有的与该文档相关的视
    pDoc->UpdateAllViews(NULL);
}
```

7. 编写 CEnterFormView::OnUpdate 函数。

```
void CEnterFormView::OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint)
{
    CChap10CDoc*      pDoc = GetDocument();
    //设置 Edit 框中的字符串
    SetDlgItemText(IDC_EDIT_TEXT,pDoc->m_string);
}
```

8. 在 EnterFormView.cpp 中添加头文件。

```
#include "stdafx.h"
#include "Chap10C.h"
#include "Chap10CDoc.h"
#include "Chap10CView.h"
#include "EnterFormView.h"
```

9. 修改 CStaticSpltFrm::OnCreateClient 函数。

```
BOOL CStaticSpltFrm::OnCreateClient(LPCREATESTRUCT lpcs,
                                    CCreateContext* pContext)
{
    BOOL ret = m_wndSplitter.CreateStatic(this,2,1);
    ret |= m_wndSplitter.CreateView(0,0,RUNTIME_CLASS(CChap10CView),
                                    CSize(100,100),pContext);
    ret |= m_wndSplitter.CreateView(1,0,RUNTIME_CLASS(CEnterFormView),
                                    CSize(100,100),pContext);
    return ret;
}
```

10. 在 StaticSpltFrm.cpp 中添加头文件。

```
#include "Chap10CDoc.h"
#include "Chap10CView.h"
#include "EnterFormView.h"
```

11. 创建和运行应用程序。

从格式视的编辑框中输入文本，再单击 Enter 按钮，注意格式视上方的视窗口中文本的变化。或者从格式视上方的窗口中输入字符串，再选择 View \ UpdateAllViews 菜单项，注意格式视的编辑框中文本的变化，系统运行情况参见图 11.3。

为了更好地理解文档对象，读者可以仿照在视窗口中显示视对象标识号的方式显示文档对象的标识号。具体做法可参考 10.6 节中的示例。

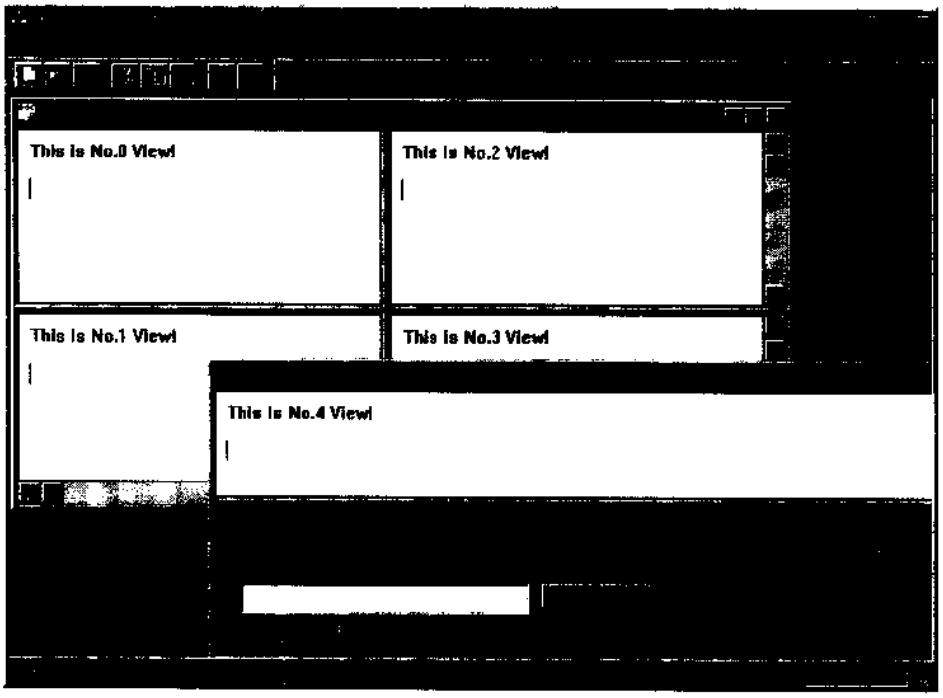



图 11.3 文档 – 视结构示例

## 11.5 本章小结

本章讲述了文档与视之间的相互关系。通过学习本章,读者将进一步了解 MFC 中的文档与视,从而实现规范化编程。

# 第十二章　一个小型的绘图系统

**本章要点**

- 编制 Windows 程序的总体思想
- 应用程序的设计思路
- 应用程序界面部分的制作
- 应用程序功能的实现
- 应用程序的源程序

本章我们将综合前面所学的知识，制作一个小型的绘图系统。之所以称其小是因为系统的功能都很简单，每一类仅举几个算法最简单的功能。例如，在对象捕捉中仅捕捉端点、中点、圆心以及圆的象限点，而并不像 AutoCAD，还可以捕捉切点、交点等。但就对象捕捉功能而言，捕捉本章所列举的四种点以后，再捕捉别的点，实现方法相似，无非是 switch 语句中

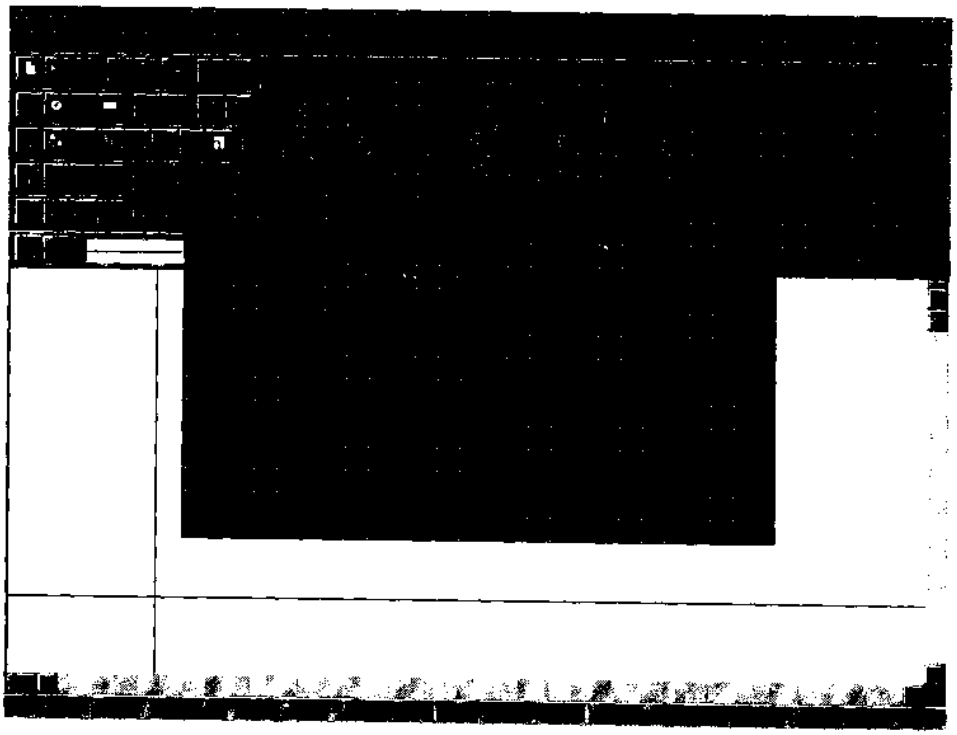


图 12.1　启动时的 A Little CAD 应用程序

多添 case 分支。但需要指出的是,求切点或交点的算法,比求本章所列举的所捕捉的四种点的算法要难。之所以称这个应用程序为一系统,是因为其功能的类别几乎覆盖了绘图时所需要功能的所有类别。例如,绘制基本图形功能,绘制图形时,可能需要用对象捕捉、捕捉网格或正交等辅助功能来帮助输入点。绘制好的图形,有时或许要进行删除、旋转、镜像、平移等变换。在观察图形时,还用到一些诸如 Zoom In、Zoom Out 之类的命令。此外,还有当前的颜色、线型等属性设置。如果读者不太熟悉一般的绘图软件,如 AutoCAD,则可能对上面所提及的一些术语不太明白。没有关系,本章随后将在系统分析中详述上面所提到的绘图功能。

读者或许知道,CAD 即指计算机辅助设计(Computer Aided Design),而非计算机辅助绘图(Computer Aided Drawing)。既然本章的应用程序是仿 AutoCAD 而做的,可以完成一些简单的图形,难道因此便称 CAD? 如果在绘图的过程中有所创新,则可以称绘图为设计,这便无可厚非了。但从设计角度出发,用此应用程序,功能未免又太简单有限了一点。所以,我们不知应该为该应用程序取何名字,姑且叫作 A little CAD 系统吧! 希望有兴趣的读者,在学会了该示例的基础上,接着往下做,丰富其功能。

图 12.1 为该应用程序启动时的情景,图 12.2 为该应用程序运行时的界面。

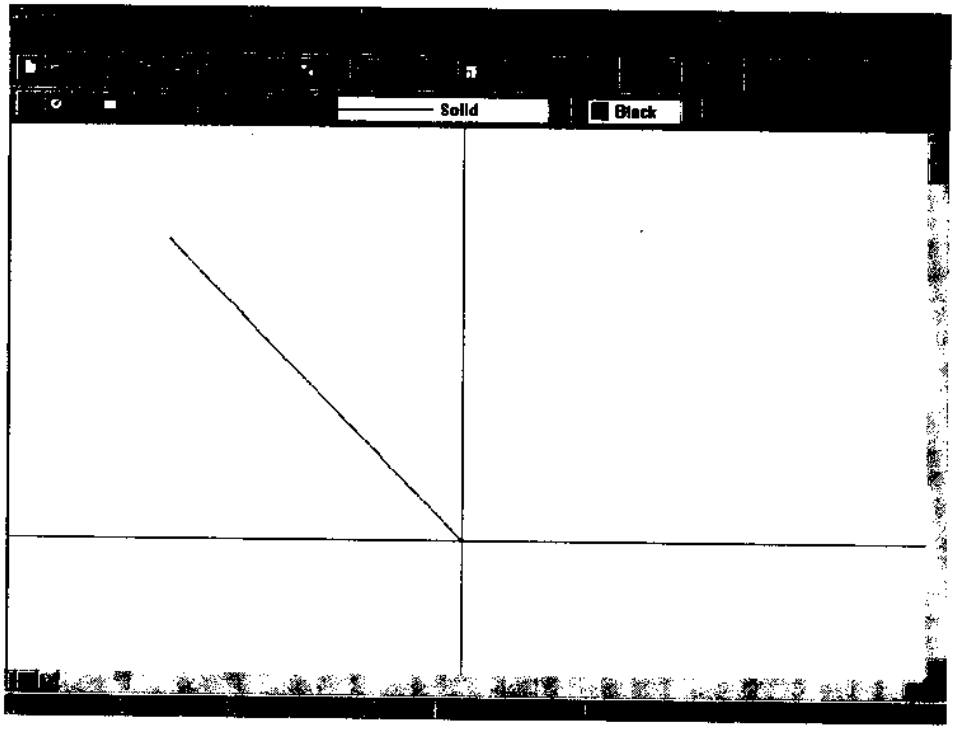



图 12.2 运行时 A Little CAD 应用程序

# 12.1　编制 Windows 程序的总体思路

Windows 程序之所以不同于 DOS 程序，是因为它是基于消息传递、事件触发来驱动运行的。所谓事件触发是指用户通过使用外部输入设备，一般是鼠标和键盘往 Windows 系统中发送消息事件。此外，系统的内部时钟，在设置了时钟中断输入后，也可以产生事件触发。所谓消息传递，就是 Windows 运行的一种机制，是把事件触发所产生的消息排成一个队列——消息对象，然后将消息往触发该事件消息的对象中发送，从而驱动程序运行。

由于 Windows 程序是基于消息传递、事件触发来驱动运行的，所以在编写 Windows 程序时，一般是从 OnLButtonDown()、OnLButtonUp()、OnMouseMove()以及 OnKeyDown()等鼠标、键盘消息对应的函数去调用相应的功能。例如，在绘图系统中有许多绘图功能，为了绘制一条直线，系统内部必须有一个量来标识我们要画直线了。而对于绘制一条直线而言，无非是需要两个端点。在输入第二个端点后，将两端点一连接即生成。于是在系统内部还需要另一变量来标识我们是在画直线的第一个端点还是第二个端点。这个变量即指明当前任务所处的当前步骤。一般而言，有了这两个变量，我们便可以在鼠标消息对应的函数中根据这两个量来决定我们在绘制哪一种实体，正在绘图中的哪一步(是画第一个端点，还是第二个端点)。

DOS 程序则与此不同，它是从头到尾地执行某一项功能。例如，在绘图系统中选择了画直线功能后，它会提示用户输入第一个端点。在用户输入后，它又紧接着要用户输入第二个端点，周而复始。它的当前任务以及当前任务的当前步骤，均由程序在所执行代码段的位置来决定。

综上所述，Windows 程序以消息驱动，从消息映射函数中根据当前任务以及当前任务的当前步骤来判定该执行哪段程序代码。而 DOS 程序则在一个函数内部通过一步步提示它所需要的交互输入量来完成操作。

# 12.2　该应用程序的设计思路

## 12.2.1　功能分析

一般的图形系统都具备绘制基本图元、修改图形、对象捕捉、图形观察，以及系统状态设置方面的功能。

例举该应用程序的目的在于向读者讲明如何编写 Windows 程序，所以在该应用程序中所列举的功能均本着从简从略出发，各类功能具体如下。

### 12.2.1.1　绘制基本图元

图元即是图形系统的基本元素，一般是指未经过任何编辑处理，便能用系统功能直接绘制生成的图形整体。

本系统能绘制的基本图元如下：直线、圆、圆弧、矩形柜、填充圆、填充矩形、点和文本。

12.2.1.2 修改图形

图形便是图元的集合。在图形中可以有单个图元、多个图元以及编辑处理后的图元。图形在绘制过程中,有时需要对图形进行修改。譬如,需要将某个绘制好的小零件进行平移,以便放入到装配图中;或者将画错的实体删除掉等。

该系统中修改图形的功能有:删除(擦除)、拷贝、剪切、粘贴、镜像、平移、旋转、比例缩放。但拷贝、剪切、粘贴功能尚未完成。有兴趣的读者可以自己试着做做。在该系统中,为这些功能的扩展留下了接口。

12.2.1.3 对象捕捉

在绘制图形时,有时需要从某一直线的端点出发绘制另一条直线,或者需要以某圆的圆心为圆心,绘制同心圆。这种在屏幕上捕获已绘制图形的特殊点,以便精确定位输入点绘图的方法被称为对象捕捉。

在该系统中能捕捉的特殊点如下:端点、中点、圆心和象限点。所谓端点,在这里是指直线的端点,矩形的四个角点(填充或未填充矩形的)以及圆弧的起始点和终止点。中点是指直线、矩形边框线上的中点。圆心是指圆、圆弧、填充圆的圆心。象限点是指圆的圆周上与时钟零点、三点、六点、九点方向的交点。最后需要指出的是,对于点和文本对象,可以用端点和中点捕捉法捕捉点和文本对象的左上角点。

12.2.1.4 图形观察

图形系统为绘图所提供的最方便的功能是可以随意地观察图形。在小小的计算机屏幕上,借助于图形系统,我们可以把千丈高楼置于其间,观察外形,评判整体效果;亦可以把手表零件或芯片中的电路放大千倍万倍,进行设计。凡在计算机中保存的图形,我们都可以放大、缩小、平移、旋转,以便从各个方面观察它。

该应用程序所提供的图形观察功能有:左移、右移、上移、下移、用窗口缩放图形、缩小图形、放大图形、收缩以装入整个图形。所谓上下左右移动观察是指在触发相应命令后,整个图形上下左右移动。用窗口缩放图形是指用鼠标点取一矩形框,以便将矩形框内的所有对象放大至客户区显示。缩小、放大图形是指图形按系统内部设置比例整体缩小、放大。收缩以装入整个图形是指按整幅图的最左、最右、最上、最下点,即最大和最小的X、Y值,将整个图形收缩至屏幕中。

12.2.1.5 系统状态设置

以上功能均与系统的当前状态有关。当前的颜色、线型设置便决定了在随后绘制的对象,如直线、圆等所具备的颜色和线型。

当前所能捕捉对象的开关,这与对象捕捉相似,但又有所不同。在对象捕捉中仅能选取一种对象,要么是端点,要么是中点,而且只能选择一次。此处则不同,在该功能打开时,它可以无限次地选择对象捕捉中所提供对象的任务组合,即既可以捕捉中点,又可以捕捉端点,还可以捕捉圆心等。

捕捉功能开关是指在打开该功能时,所能输入点的坐标是预设最小值的整数倍。例如,若捕捉预设的最小值是0.5,则在打开该功能时,输入的值是0.5、-0.5等,决不会出现0.

4999、0.5001、-0.4999、-0.5001 等情况。这对控制输入"整点"大有裨益。

网格开关功能是指有时候为了绘图、目测定位方便,便可以打开网格。所谓网格是指在屏幕上出现的一系列阵列点,利用这些点便可以方便地目测定位。

此外,还有正交输入开关。在正交输入打开时,用户仅能绘制水平线或竖直线。最后,还有重绘功能,以及系统状态设置功能。

除了上面所提到的几类功能外,还有一个特殊的功能。用户可以用动态切分窗口将用户区分成两块或四块,分别从不同的视图中观察图形。希望读者认真思考一下此处的四个视窗口是如何工作的,视对象是如何区分的。

### 12.2.2　系统中的基本对象

其实,在上一节"功能分析"中已经提到了系统中的基本对象。请读者回头看看绘制基本图元中所列的图元:直线、圆、圆弧、矩形框、填充圆、填充矩形、点以及文本。

为了准确无误地表示这些图元,各元素中所需存储记录的信息如下:

直线——起点的 x、y 坐标值,终点的 x、y 坐标值。

圆——圆心的 x、y 坐标值、半径值。

圆弧——圆心的 x、y 坐标值,半径值,起点的 x、y 坐标值,终点的 x、y 坐标值。

填充圆——填充圆心的 x、y 坐标值、半径值,所填充模式的类型值。

矩形框——矩形框的左上角点以及右下角点的 x、y 坐标值。

填充矩形——矩形框的左上角点以及右下角点的 x、y 坐标值,所填充模式的类型值。

点——点的 x、y 坐标值。

文本——文本左上角插入点的 x、y 坐标值以及文本字符串。

以上对象除了点和文本以外,都还需要线型值,以表征线型。所有对象均需要颜色值。

### 12.2.3　实体对象在系统中的存储方法

众所周知,对数据的存储分为两种。一种是在磁盘上永久性地静态存储,另一种是在系统运行过程中将数据在内存中动态地存储。在这一节中,我们要讨论的是如何把各种不同对象的数据动态地存储在内存中。

在系统中动态地存储数据的最好方法是用链表,利用 new 和 delete 关键字可以方便地分配和删除链节。在链表中,各链节元素应当相同,这样才便于用链节中的指针来指向。然而,由上节分析可知,我们的系统中各图元对象所需的表征几何特性的数据各不相同。如何将它们统一起来用链表管理呢?下面列出了三种方法来解决此问题:

1. 每种对象建立一个链表。以上八种对象共需要八个链表。这种方法可行,而且笔者也实现过,只是过于笨拙。想一想是管理的链表越多好呢,还是越少好。

2. 可以建一个类,在该类中包括各种类型的图元所需的几何特征数据。在类中再设置一个实体对象类型标识符。通过它来判定这是哪种元素,用哪种方法处理。这方法也可行,但浪费了存储空间,而且在调用或编写类的成员函数时容易出现混乱。

3. 第三种方法充分利用了C++ 基类与派生类可以互相转换的特点,从而使存储合理,

操作简单。具体作法如下：系统中的八种实体对象，都共同继承 CEntity 类。继承 CEntity 的派生类中含有各种实体各自的几何信息。在 CEntity 中放入各种实体都需要的实体类型值、线型值、颜色值。注意，点和文本对象不需要线型值，但为了编程统一的需要，也令其用到了，只是没有什么效果罢了。CEntity 类中的表征实体类型的值，用来判断实体的类型，以便进行对象类型的转换。为了使各链节连接起来，在 CEntity 类中还需要指向前一链节、后一链节的指针，以便建立双向链表。在系统中各实体对象相连的示意图如图 12.3 所示。

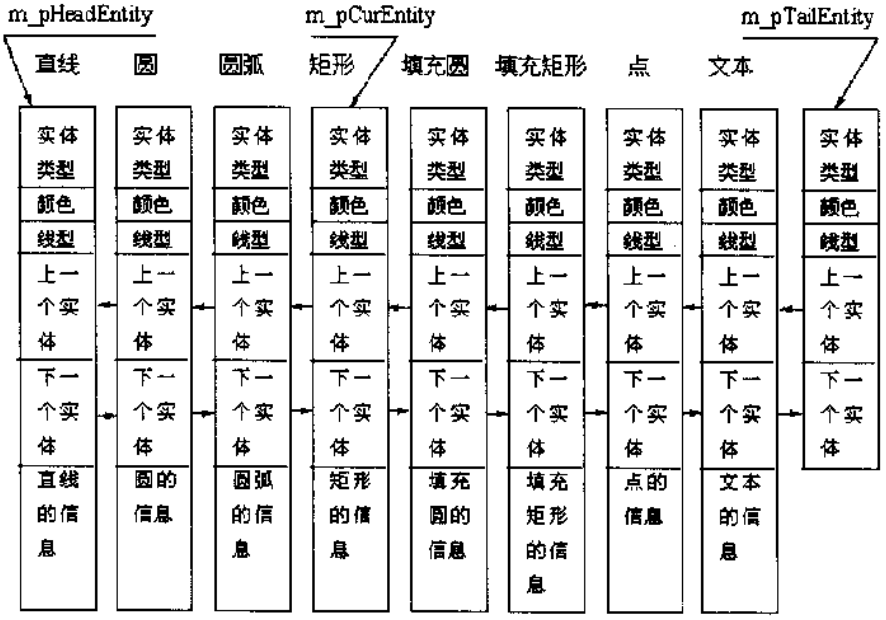



图 12.3 各实体对象相连的示意图




1. 链表是通过 CEntity 基类对象建立起来的。链表的头、尾、当前指针以及各链节类的上一个、下一个指针均是指向 CEntity 的指针。对于不同对象的操作，是通过类的转换，即根据基类 CEntity 中的实体类型值将 CEntity 对象转化为其他五种类型的对象，再进行操作。

2. 为了减少实体对象类的个数，实质上我们也用了方法 2 的思想，即将圆、圆弧、填充圆放在一个类中，在该类中添全这三种对象所需的几何数据；将矩形框和填充矩形合在一个类中。

3. 或许读者会纳闷，为何在该链表中链尾是一个 CEntity 基类对象。这将在下一节“系统中文件、数据的组织策略”中介绍。

4. 最后，需要指出的是上述链表只是一个示意图。实际运行中的与之相似，只是链节可能更多，各链节的内容、顺序不同罢了。

接下来，将各种基本实体对象定义的代码列出。注意，其中只列出了各类实体的数据，未列出其中的成员函数。

```
class CEntity
{
public:
      ......

      UINT        m_nType;
      COLORREF    m_nColor;
      UINT        m_nLineType;

      CEntity *   m_pNext;
      CEntity *   m_pPre;
};

class CLine: public CEntity
{
public:
      ......

      double      m_dSx,m_dSy;
      double      m_dEx,m_dEy;
};

class CCircle  : public CEntity
{
public:
      ......

      double m_dCenterX,m_dCenterY;
      double m_dR;
      double m_dSx,m_dSy;
      double m_dEx,m_dEy;
      UINT   m_nFillPattern;
};

class CRectangle:public CEntity
{
public:
      ......

      double      m_dLeft,m_dTop;
      double      m_dRight,m_dBottom;
      UINT        m_nFillPattern;
};
```

```
class CPnt : public CEntity
{
public:
      ......

      double m_dX,m_dY;
};

class CText : public CEntity
{
public:
      ......

      double      m_dX,m_dY;
      CString     m_strText;
};
```

### 12.2.4 系统中文件、数据的组织策略

现在,读者已经知道了,在该应用程序中实体对象的数据结构,也知道了这些数据以链表的形式进行组织存储。可是该链表究竟该如何建立?在重绘图形、拾取图形以及对象捕捉时又该怎样去对链表进行遍历搜索呢?与各种实体对象相关的代码又该如何组织在文件中呢?这些问题,我们将在本节得到解答。

在"视与文档"那一章中,我们已知道:视是用来操作对象的,凡是能从屏幕上看到的对对象的操作都可以,而且也都应当放在视类中完成。文档是用来统筹数据的。一方面,支持着视类,以便使数据在视中得到正确的显示。另一方面,在视类对对象进行了操作时,文档类能及时地修改数据,促使视更新显示。因此,我们把该系统中的实体对象链表均放在文档类中,视类通过 GetDocument()函数获取文档中的数据,以便操纵这些数据。

好了,现在的问题是如何在文档中组织数据了。

从上一节的分析,我们知道,该系统中为了存储各种类型的实体,共定义了六个类。其中 CEntity 为基类,另外五个类:CLine、CCircle、CRectangle、CPnt、CText 为 CEntity 的派生类。读者可以用 ClassWizard 增加这些类。这些类也便各自属于实现文件(.cpp)和头文件(.h)。这里共有六个.cpp 文件,六个.h 文件。如下所示:

```
entity.h          entity.cpp
line.h            line.cpp
circle.h          circle.cpp
rectangle.h       rectangle.cpp
point.h           point.cpp
text.h            text.cpp
```

为了在文档类中建立实体链表，我们用到了两个变量：m_pEntityHead 和 m_pEntityTail。这两个变量均是指向 CEntity 的指针。为了使对链表的操作简单，我们在程序一启动时便建立了链表。建立链表的程序在 CCadDocument 类的构造函数中。这也就是为何在上一节的链表图中尾部有一个 CEntity 对象。读者或许要问事先建立一个链表，究竟有何优点？优点也不多，只是每次在创建对象时不必判断，这是否是该链表的第一个链节，即是否要建立链表。在从头到尾遍历时也不必用 m_pNext 是否为 NULL 来判断是否搜索到头了。这些可能并不是什么难点、关键处，但事先建好链表后，对链表的操作便只剩下了添加或删除链节，而不必判断链表是否已不存在，或者已经存在。这使得对链表的操作变得简单，思维过程也简单了。每次在添加链节时，先用 new 分配空间，赋好初值后，将该对象的指针强制转化成 CEntity 类型的对象再添加到链表头部。在删除中间链节时（永远也不删除链尾处链节，除非退出系统），将该链节的前后两链节的指针值改变即可；在删除链头时，只用删去链头，将 m_pEntityHead 指向下一个链节即可。在删除链节时，我想大家能够领悟到为何要用双向链表了吧！在从头至尾遍历时，只用判断当前指针是否为 m_pEntityTail 便可知是否到了链尾。

在文档类中还有一个指向链节的成员变量 m_pEntityCurrent。读者或许会问，链表已有了 m_pEntityHead、m_pEntityTail 来指向头和尾，还要 m_pEntityCurrent 干什么呢？其实，在从头至尾遍历链表进行对象捕捉时便会用到它。当几个特征点隔得较近时，系统并不知道哪个点是用户想捕捉的。它只能先显示出一个可能是用户想要的点。用户不满意后，移动鼠标，系统又从当前位置继续往下搜索，找下一个可能的点，直到用户满意为止。试想，如果不用 m_pEntityCurrent 记住所遍历的当前位置，下次再重新遍历时，恐怕很难，甚至永远也捕捉不到用户想要的点。因为有些点实在隔得太近，在尚未遍历到用户想要捕捉的点所在对象之前便已捕捉到了其它对象而停止搜索。

在文档类中还有五个对象：m_Line、m_Circle、m_Rectangle、m_Pnt、m_Text。在用户绘制（创建）对象时，用这些成员变量保存创建过程中的数据。直到图形绘好后，才创建新链节，将存储在这些变量中的数据转存到新创建的对象中，加入链表。用这五个成员变量可以防止在中途取消绘图操作时内存丢失。这是内存管理策略之一。否则，先创建对象，尚不知所创建的对象所需数据是否已经全部输入，是将该对象加入链表呢，还是删去了，亦或是忘了处理，造成内存丢失。内存丢失是编程的一大忌。

此外，在文档类中还有一个指向 CEntity 类的指针数组 m_pEntitySelected[10]以及 m_nEntitySelCnt。m_pEntitySelected[10]用于标记已选中的实体，即将用于删除、平移、镜像、旋转等操作的实体，可以称为选择集。m_nEntitySelCnt 记录已选择实体的数目。每次在进行删除、平移、镜像、旋转等修改操作之前，先将 m_nEntitySelCnt 设置为 0，即代表选择集为空。操作结束后也将该值设置为 0。

在文档中关于实体操作变量的定义如下：

```
// Attributes
public:
    double      m_dChgA[3][3];          //变换矩阵
    CEntity *   m_pEntityHead;          //实体链表头指针
```

```
CEntity*    m_pEntityTail;              //实体链表尾指针
CEntity*    m_pCurEntity;               //实体链表中的当前实体指针
CEntity*    m_pEntitySelected[10];      //实体选择集指针
int         m_nEntitySelCnt;            //实体选择集中的实体数目

//预定义的几个实体
CLine       m_Line;
CCircle     m_Circle;
CRectangle  m_Rectangle;
CPnt        m_Pnt;
CText       m_Text;
```

### 12.2.5 系统中的全局变量

为了编程实现的方便,有时候我们需要将一些经常在各个不同的模块中用到的变量,设置为全局变量。这样,我们就不必在调用函数时频繁地传递这些参数。

在本应用程序中,设置的全局变量共有四类:一类是与系统当前任务以及执行当前任务正处在的步骤相关的,另一类是与鼠标操作相关的,第三类是与系统的状态相关的,最后一类是与视类相关的。这些变量都在 Cad.h 文件中声明,在 Cad.cpp 文件中定义。接下来,分别介绍这几类变量的用途。

#### 12.2.5.1 与系统的任务以及任务执行所处步骤相关的变量

在分析 Windows 程序与 DOS 程序的区别时,我们曾提到 Windows 程序是基于消息传递、事物触发而驱动的。因而我们需要用两个变量来表征系统的当前状态。其中一个表示任务,另一个表示执行该任务所处的步骤。

但是,在系统中单有这一组变量来表示任务和步骤是不够的。譬如,在画直线时,用户想去捕捉某个点来作为端点,或者缩放一个窗口以便接着画线。此时,我们不能打断前一个画直线任务,但我们又必须启动另一个任务来执行诸如对象捕捉或窗口缩放之类的辅助操作。于是,我们必须还有另一个变量来标记住辅助任务的状态。辅助任务与 AutoCAD 的透明命令有些相似。熟悉 AutoCAD 的读者或许知道,AutoCAD 允许用户在绘图时调用透明命令来辅助绘图,如对象捕捉、窗口缩放、打开或关闭网格、捕捉、正交等。为此,我们需要两组任务:一组由 g_nCurTask 和 g_nStep 记录,用于执行绘图、修改等与图形数据的增加、修改、删除有关的操作;另一组由 g_nOpqTask 和 g_nOpqStep 记录,主要用于对象捕捉、窗口缩放等辅助绘图任务。




由于辅助(透明)任务总是用于辅助绘图的,因而对它的调用总在主任务之前,所以在 OnMouseMove()、OnLButtonDown()以及 OnRButtonDown()函数中总是先调用与辅助任务相关的函数,后调用与主任务相关的函数。切记,该顺序不可调换。至于原因,将在与对象捕捉实现方法相关的章节中讲明。

在 Cad.h 文件中与任务及步骤相关的变量声明如下:

```
//记录主任务和辅助任务
extern UINT       g_nCurTask;
extern UINT       g_nStep;
extern UINT       g_nOpqTask;
extern UINT       g_nOpqStep;
```

在 Cad.cpp 文件中与任务及步聚相关的变量定义如下：

```
//记录主任务和辅助任务
UINT  g_nCurTask = 0;
UINT  g_nStep = 0;
UINT  g_nOpqTask = 0;
UINT  g_nOpqStep = 0;
```

最后，仍需指出的是，有些任务(功能)是一步即可完成的，我们称其为单步任务，如开关网格功能，开关对象捕捉功能，定比例缩放图形以及上下左右移动图形观察。除了单步任务以外，余下的都是多步任务。这种任务在最终实现以前，需要用户输入多个数据。一般而言，对于单步任务，在消息映射函数中将其执行了即可，而不必改变任务号以及步骤值来执行。对于多步任务则必须通过前面所述的用任务号与相应步骤值来标记状态，以执行相应的功能。

12.2.5.2　与鼠标操作相关的全局变量

在绘图系统中用得最多的输入应当是鼠标输入。如何记住鼠标的当前位置(X、Y 坐标值)以及按键的当前状态(左键是按下，还是弹起)对我们编程接收输入信息至关重要。

对于鼠标按键的当前状态，我们只处理左、右键的按下状态，状态相对简单，用得不太多，故而只用作函数的参数传递即可。况且在不同按键状态时，我们还可以通过调用不同的函数来区分，因而不将按键的状态设置为全局变量。有时候在用鼠标进行时，需要判断键盘的 Ctrl、Shift、Alt 等键是否按下，以决定执行何种操作。这时，就需要用一个变量来记录键盘的当前状态。在该系统中，预设了该状态值 g_mFlags(但在已完成的应用程序中未用到该值。)至于鼠标所在位置，绘图时用得实在太多，所以将与鼠标位置有关的量全部定义成全局变量。这些变量在 Cad.h 中的声明为：

```
//记录当前鼠标状态
extern UINT       g_nFlags;
extern double     g_dCurX;
extern double     g_dCurY;
extern double     g_dPreX;
extern double     g_dPreY;
extern double     g_dOrgX;
extern double     g_dOrgY;
```

这些变量在 Cad.cpp 文件中的定义为：

```
//记录当前鼠标状态
```

```
UINT        g_nFlags = 0;
double      g_dCurX = 0;
double      g_dCurY = 0;
double      g_dPreX = 0;
double      g_dPreY = 0;
double      g_dOrgX = 0;
double      g_dOrgY = 0;
```




这些全局变量的前缀均为g_,以此区分变量的生命期和作用域。由前缀d知道这些变量均为double类型的。读者或许会问,既然屏幕上的鼠标位置点坐标均是整型数,为什么要用double型的变量来存储呢?其实,这正是因为屏幕上的点在进一步处理之前,先转化成世界坐标了。读者或许还不明白,这将在"改变观察视图和输入预处理"一节中详细解释。

其中,g_dCurX、g_dCurY记录了鼠标的当前位置,g_dPreX、p_dPreY记录了鼠标在上一次发送WM_MOUSEMOVE消息时所处的位置,g_dOrgX、g_dOrgY记录了鼠标在上一次按下左键时所处的位置。

现在,以绘制直线为例讲解这些变量的用途。在选择了绘制直线命令后,g_nCurTask被设置为ID_DRAW_LINE,g_nStep被设置为0。此时,鼠标在客户区中移动,仅有g_dCurX、g_dCurY、g_dPreX、g_dPreY变化。g_dCurX、g_dCurY记录了鼠标的当前位置,而g_dPreX、g_dPreY记录鼠标在前一刻的位置。用户在选定直线的一端点后,按下左键,系统会把g_dCurX、g_dCurY的值赋给g_dOrgX、g_dOrgY,直线的一个端点选定。再移动鼠标,去选择另一个端点。此时,用户将看到有橡皮筋线画出。这正是利用了这六个变量间的特殊关系。在画橡皮筋线时,先将绘图方式设置为RP2_NOT,再将OrgP(由g_dOrgX、g_dOrgY表示的点)与PreP(由g_dPreX、g_dPreY表示的点)相连,这便擦除了先前的线。接着,再将OrgP与CurP(由g_dCurX、g_dCurY表示的点)相连,这便绘出了新的线。鼠标每一移动一下,发一个WM_MOUSEMOVE消息,便执行一次上述操作。这样便形成了跟着鼠标移动的橡皮筋线。选定直线的另一端点后,按下左键系统会将OrgP与PreP相连,去掉橡皮筋线。接着,再以当前颜色和线型连接OrgP、CurP,绘制直线。注意,在选定第二个端点,按下左键时,OrgP点已变成了当前点,即将g_dCurX、g_dCurY的值,分别赋给了g_dOrgX,g_dOrgY。

其实,记录鼠标的当前位置、前一个位置,以及上一次按下左键的位置不仅仅在绘制直线中有用,甚至可以这么说,由于绘制直线时,是用m_Line保存绘制直线的几何信息的(参见12.2.3节),所以我们可以直接用m_Line中的起点来取代g_OrgX、g_OrgY的值所代表点。但是,在平移,旋转,镜像,甚至窗口缩放中,我们不必设置专门的变量来记录基准点或选中缩放窗口的第一个角点。这时,g_dOrgX、g_dOrgY变量便尤为重要。读者可以在编程时再细细体味这六个变量的作用,它们是如何相互赋值的。

#### 12.2.5.3 与系统状态相关的全局变量

所谓系统状态是指系统当前的Snap、Osnap、Grid(在状态条上)开关是否打开,Snap、Grid

的间距有多大，系统的当前颜色为何，线型为何。由于这些信息在系统中广泛使用，所以都被设置为全局变量。这些变量在 Cad.h 中的声明如下：

```
//记录当前颜色和线性
extern UINT       g_nCurColor;
extern UINT       g_nCurLineType;

//记录 GRID,ORTHO,SNAP 状态
extern BOOL       g_bGrid;
extern BOOL       g_bOrtho;
extern BOOL       g_bSnap;

//记录 Grid 大小和 Snap 大小
extern double     g_dGrid;
extern double     g_dSnap;

//记录对象捕捉类型和屏幕像素点距离
extern int        g_nOSnapType;
extern int        g_nOSnapSize;
```

这些变量在 Cad.cpp 中的定义为：

```
//记录当前颜色和线性
UINT  g_nCurColor = RGB(0,0,0);
UINT  g_nCurLineType = PS_SOLID;

//记录 GRID,ORTHO,SNAP 状态
BOOL  g_bGrid = FALSE;
BOOL  g_bOrtho = FALSE;
BOOL  g_bSnap = FALSE;

//记录 Grid 大小和 Snap 大小
double      g_dGrid = 0.5;
double      g_dSnap = 0.5;

//记录对象捕捉类型和屏幕像素点距离
int         g_nOSnapType = 0;
int         g_nOSnapSize = 20;
```

其中 g_bGrid、g_bOSnap、g_bSnap 分别代表 Grid、OSnap、Snap 是呈打开状态，还是关闭状态。g_dGrid 表示网格的间距大小，g_dSnap 为点捕捉的精度值 g_nOSnapType 代表欲捕捉的类型，按位表示。此外，g_nLinetype、g_nColor 分别表示当前的线型和颜色。

12.2.5.4　与视类相关的全局变量

由于该应用程序是支持动态切分窗口的，因而在客户区分成几个窗片后，各窗片均对应

一个视对象。由于各个视对象所反映的图形数据是相同的,因为它们有共同的文档对象支持,所以各个视对象在显示图形时的主要差别便在于:这些图形在本窗片中显示时,世界坐标与屏幕坐标的映射关系不同。这在坐标变换中的具体体现便是世界坐标向屏幕坐标变换时的比例不同,两个系统的坐标原点的相对位置关系不同。因此,世界坐标向屏幕坐标变换的比例,原点的相对位置关系均为各视对象所独有。表征这种关系的变量以及实现坐标转换的函数便放在视类中定义。关于世界坐标与屏幕坐标究竟是如何映射、如何转变的,将在"改变观察视图与输入预处理"中详细介绍。

讲了这么多,读者或许更疑惑了。究竟什么是与视类相关的全局变量?其实说来也简单,就是一个指向视对象的指针。它在 Cad.h 中的声明如下:

```
extern class CCadView*   g_pCurView;
```

它在 Cad.cpp 中的定义如下:

```
CCadView*    g_pCurView=NULL;
```

前面所讲的无非是让读者明白,由于应用程序支持切分窗口,因而可能存在多个视对象。而各个视对象的差别仅在于世界坐标与屏幕坐标间的变换,因而表示变换差别的是在各个视对象中。为此系统在重绘图形时,需要用到从世界坐标往屏幕坐标的变换,即各个视对象中的坐标变换函数,所以设计了这个指向视对象的全局变量。

笔者也曾想过同时传递视指针 pView 以及指向设备环境表的指针 CDC。然而,这两者实在太相似了。况且,可以由 pView 通过 pView->GetDC()获取设备环境表指针。这无疑将对设备环境表的管理造成混乱。因为在 Windows 系统中仅有五个设备环境表可用。如果总在函数调用时,总由 pView->GetDC()获得 CDC 来绘图,则很容易超出五个设备环境表的限制,从而造成错误。为此,这里必须声明,我们纵然可以由 g_pCurView->GetDC()获得设备环境表绘图,但我们决不能这么做。我们使用 g_pCurView 的目的仅在于实现坐标变换。

### 12.2.6 视文档结构对图形对象的操作方法

在第十一章中,我们已知道了视与文档的相互关系。在本章的 12.2.4 节中我们也提到了视、文档、基本实体对象间的相互关系,但究竟应当如何组织数据呢?这一节将对这一问题再作进一步的探讨。

在本应用程序中,视、文档、基本实体的层次关系如图 12.4 所示。

#### 12.2.6.1 第一层:用户与系统交互层

由上可知,该应用程序就结构而言分为四层。其中最顶层为用户与系统交互层。与之相关的操作都封装在 CCadView 中。其中最关键的、用于对实体数据操作的函数如下:

1.移动鼠标时,所调用的函数

```
void CCadView::OnMouseMove(UINT nFlags, CPoint point)
{
```

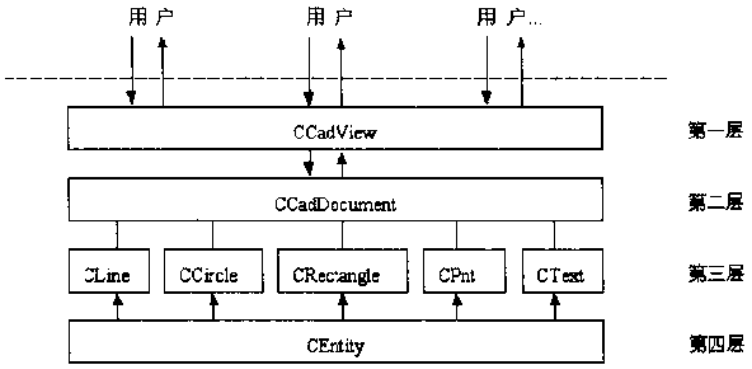



图 12.4　视文档、基本实体层次关系图

```
g_nFlags = nFlags;
ScreenToWorld(g_dCurX,g_dCurY,point);
g_pCurView = this;
CCadDoc * pDoc = GetDocument();
ASSERT_VALID(pDoc);
CDC * pDC = GetDC();

DrawCursor(pDC,point);
//记录当前点,以作为下次进入 MouseMove 的前一个点
m_prePoint = point;

if((g_nOpqTask > = ID_OSNAP_ENDPOINT&&g_nOpqTask < = ID_OSNAP_QUARDANT)
      ||g_nOSnapType! = 0)
{
      pDoc - > OSnap(pDC,FALSE);
}
if(g_nOpqTask > = ID_ZOOM_WINDOW&&g_nOpqTask < = ID_ZOOM_ALL)
      ZoomWindow(pDC,FALSE);

if(g_nCurTask > = ID_DRAW_LINE&&g_nCurTask < = ID_DRAW_TEXT)
      pDoc - > Create(pDC,FALSE);

if((g_nCurTask > = ID_EDIT_COPY&&g_nCurTask < = ID_EDIT_PASTE)
  ||(g_nCurTask > = ID_MODIFY_ERASE&&g_nCurTask < = ID_MODIFY_SCALE))
{
      pDoc - > ModifyMove(pDC);
}

//获得状态条的指针
CMyStatusBar * pStatus = (CMyStatusBar * )
```

```
    AfxGetApp()->m_pMainWnd->GetDescendantWindow(ID_VIEW_STATUS_BAR);
    if(pStatus)
    {
        char tbuf[40];
        sprintf(tbuf,"(%8.3f,%8.3f)",g_dCurX,g_dCurY);
        //在状态条的第二个窗格中输出当前鼠标的位置
        pStatus->SetPaneText(1,tbuf);
    }

    g_dPreX=g_dCurX;
    g_dPreY=g_dCurY;
}
```

2. 鼠标左键按下时,所调用的函数

```
void CCadView::OnLButtonDown(UINT nFlags, CPoint point)
{
    g_nFlags=nFlags;
    ScreenToWorld(g_dCurX,g_dCurY,point);
    g_pCurView=this;

    CCadDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    CDC* pDC=GetDC();

    if((g_nOpqTask>=ID_OSNAP_ENDPOINT&&g_nOpqTask<=ID_OSNAP_QUARDANT)
        ||g_nOSnapType!=0)
    {
        pDoc->OSnap(pDC,TRUE);
    }
    if(g_nOpqTask==ID_ZOOM_WINDOW)
        ZoomWindow(pDC,TRUE);

    if(g_nCurTask>=ID_DRAW_LINE&&g_nCurTask<=ID_DRAW_TEXT)
        pDoc->Create(pDC,TRUE);

    if((g_nCurTask>=ID_EDIT_COPY&&g_nCurTask<=ID_EDIT_PASTE)
      ||(g_nCurTask>=ID_MODIFY_ERASE&&g_nCurTask<=ID_MODIFY_SCALE))
    {
        pDoc->ModifyLButton(pDC);
    }

    g_dOrgX=g_dCurX;
    g_dOrgY=g_dCurY;
```

```
}
```

3.鼠标右键按下时,所调用的函数

```
void CCadView::OnRButtonDown(UINT nFlags, CPoint point)
{
    CCadDoc * pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    CDC * pDC = GetDC();

    if((g_nCurTask >= ID_EDIT_COPY&&g_nCurTask <= ID_EDIT_PASTE)
      ||(g_nCurTask >= ID_MODIFY_ERASE&&g_nCurTask <= ID_MODIFY_SCALE))
    {
        pDoc->ModifyRButton(pDC);
    }
}
```

4.按下键盘上某键时,所调用的函数

```
void CCadView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    //获取当前客户区的大小
    CRect rect;
    GetClientRect(rect);
    //获得偏移变量
    double      delta_x = (rect.right - rect.left) * m_dScreenToWorld/6;
    double      delta_y = (rect.bottom - rect.top) * m_dScreenToWorld/6;

    switch(nChar)
    {
    case VK_LEFT://图形向左移
        m_dOrgX += delta_x;
        break;
    case VK_RIGHT://图形向右移
        m_dOrgX -= delta_x;
        break;
    case VK_UP://图形向上移
        m_dOrgY -= delta_y;
        break;
    case VK_DOWN://图形向下移
        m_dOrgY += delta_y;
        break;
    case VK_NEXT://图形放大
        m_dWorldToScreen *= 0.618;
        if(m_dWorldToScreen < 1e-10) m_dWorldToScreen = 1.0;
```

```
            m_dScreenToWorld = 1.0/m_dWorldToScreen;
            break;
        case VK_PRIOR://图形缩小
            m_dScreenToWorld *= 0.618;
            if(m_dScreenToWorld < 1e-10) m_dScreenToWorld = 1.0;
            m_dWorldToScreen = 1.0/m_dScreenToWorld;
            break;
        default:
            break;
        }

        CDC * pDC = GetDC();
        pDC->Rectangle(rect);
        OnDraw(pDC);
    }
```

这四个函数的共性是通过 pDoc 直接操纵 CCadDocument 类中的函数。

此外，还有在选择绘图或修改、缩放、对象捕捉等命令时，设置系统当前任务以及步骤的消息映射函数。它们的消息映射宏及其函数如下：

```
ON_COMMAND_RANGE(ID_DRAW_LINE,ID_DRAW_TEXT,OnDrawCommand)
ON_COMMAND_RANGE(ID_MODIFY_ERASE,ID_MODIFY_SCALE,OnModifyCommand)
ON_COMMAND_RANGE(ID_OSNAP_ENDPOINT,ID_OSNAP_QUARDANT,OnOSnapCommand)
ON_COMMAND_RANGE(ID_ZOOM_WINDOW,ID_ZOOM_ALL,OnZoomCommand)
ON_COMMAND_RANGE(ID_EDIT_COPY,ID_EDIT_PASTE,OnNotFinishCommand)
```

1.用于处理绘图命令的函数

```
void CCadView::OnDrawCommand(DWORD wParam,LPARAM lParam)
{
    g_nCurTask = wParam;
    g_nStep = 0;

    char * prompt[] = {"Enter the first point",
                       "Enter the center",
                       "Enter the center",
                       "Enter the first point",
                       "Enter the center",
                       "Enter the first point",
                       "Enter the point",
                       "Enter the insert point",
                       };

    CStatusBar * pStatus = (CStatusBar * )
```

```
    AfxGetApp()->m_pMainWnd->GetDescendantWindow(ID_VIEW_STATUS_BAR);
    ASSERT(pStatus);

    pStatus->SetPaneText(0,prompt[wParam-ID_DRAW_LINE]);

}
```

2.用于处理修改命令的函数

```
void CCadView::OnModifyCommand(DWORD wParam,LPARAM lParam)
{
    g_nCurTask = wParam;
    g_nStep = 0;
    CCadDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    pDoc->m_nEntitySelCnt = 0;
    char* prompt[] = {"Pick entities to erase",
                      "Pick entities to mirror",
                      "Pick entities to move",
                      "Pick entities to rotate",
                      "Pick entities to scale",
                      };

    CStatusBar* pStatus = (CStatusBar*)
      AfxGetApp()->m_pMainWnd->GetDescendantWindow(ID_VIEW_STATUS_BAR);
    ASSERT(pStatus);

    pStatus->SetPaneText(0,prompt[wParam-ID_MODIFY_ERASE]);
}
```

3.用于处理对象捕捉命令的函数

```
void CCadView::OnOSnapCommand(DWORD wParam,LPARAM lParam)
{
    g_nOpqTask = wParam;
    g_bOSnapped = FALSE;
    g_nOpqStep = 0;

    char* prompt[] = {"Snap end point",
                      "Snap middle point",
                      "Snap center",
                      "Snap quardant point",
                      };

    CStatusBar* pStatus = (CStatusBar*)
```

```
        AfxGetApp()->m_pMainWnd->GetDescendantWindow(ID_VIEW_STATUS_BAR);
    ASSERT(pStatus);

    pStatus->SetPaneText(0,prompt[wParam-ID_OSNAP_ENDPOINT]);
}
```

4.用于处理缩入命令的函数

```
void CCadView::OnZoomCommand(DWORD wParam,LPARAM lParam)
{
    g_nOpqStep=0;

    if(wParam==ID_ZOOM_WINDOW)
    {
        g_nOpqTask=wParam;
        g_nCurTask=0;

        CStatusBar* pStatus=(CStatusBar*)
          AfxGetApp()->m_pMainWnd->
                GetDescendantWindow(ID_VIEW_STATUS_BAR);
        ASSERT(pStatus);

        pStatus->SetPaneText(0,"Enter the first conner point");
    }
    else if(wParam==ID_ZOOM_IN)
    {
        SendMessage(WM_KEYDOWN,VK_NEXT);
        g_nOpqTask=0;
    }
    else if(wParam==ID_ZOOM_OUT)
    {
        SendMessage(WM_KEYDOWN,VK_PRIOR);
        g_nOpqTask=0;
    }
    else
    {
        m_dOrgX=(g_dMaxX+g_dMinX)/2.0;
        m_dOrgY=(g_dMaxY+g_dMinY)/2.0;
        //获取当前客户区的大小
        CRect rect;
        GetClientRect(rect);
        double dStoW_X=fabs((g_dMaxX-g_dMinX)/(rect.right-rect.left));
        double dStoW_Y=fabs((g_dMaxY-g_dMinY)/(rect.bottom-rect.top));
        if(dStoW_X>dStoW_Y)
```

```
            m_dScreenToWorld = dStoW_X;
        else
            m_dScreenToWorld = dStoW_Y;
        if(fabs(m_dScreenToWorld) < 1e - 10)
            m_dScreenToWorld = 1.0;
        m_dWorldToScreen = 1/m_dScreenToWorld;

        CDC * pDC = GetDC();
        pDC - > Rectangle(rect);
        OnDraw(pDC);
    }
}
```

5.用于提示功能尚未完成的函数

```
void CCadView::OnNotFinishCommand(DWORD wParam,LPARAM lParam)
{
    AfxMessageBox("Sorry! This function is not finished!");
}
```

12.2.6.2　第二层:实体对象间数据操纵实现层

该应用程序的第二层是实体对象间数据操纵实现层。这里的操作基本上都是对实体对象链表的操纵。这可以参考 12.2.4 节。这个部分也是该系统实现其功能的关键所在。

1.在绘制对象时,增加链节

在选择了绘图命令后,g_nCurTask、g_nStep 由视中的命令映射函数设置了初值。移动鼠标或按鼠标消息映射函数调用文档的 Create()函数。文档类中的 Create()函数的定义如下:

```
//创建新实体
void CCadDoc::Create(CDC * pDC,BOOL bSure)
{
    BOOL        bIsCreated = FALSE;
    CEntity *   pNewEntity = NULL;
    double      x[16],y[16];
    int         cnt = 0;

    switch(g_nCurTask)
    {
    case ID_DRAW_LINE:
        bIsCreated = m_Line.Create(pDC,bSure);
        if(bIsCreated = = TRUE)
        {
            CLine * pNewLine = new CLine(m_Line.m_dSx,m_Line.m_dSy,
```

```
                                        m_Line.m_dEx,m_Line.m_dEy);
            pNewEntity=(CEntity*) pNewLine;
            pNewLine->Draw(pDC);

            x[cnt]=m_Line.m_dSx;
            y[cnt]=m_Line.m_dSy;
            cnt++;
            x[cnt]=m_Line.m_dEx;
            y[cnt]=m_Line.m_dEy;
            cnt++;
        }
        break;
    case ID_DRAW_CIRCLE:
    case ID_DRAW_ARC:
    case ID_DRAW_FILLED_CIRCLE:
        bIsCreated=m_Circle.Create(pDC,bSure);
        if(bIsCreated==TRUE)
        {
            CCircle* pNewCircle;
            if(g_nCurTask==ID_DRAW_CIRCLE)
            {
              pNewCircle=new CCircle(m_Circle.m_dCenterX,
                        m_Circle.m_dCenterY,m_Circle.m_dR);
            }
            else if(g_nCurTask==ID_DRAW_ARC)
            {
              pNewCircle=new CCircle(m_Circle.m_dCenterX,
                        m_Circle.m_dCenterY,m_Circle.m_dR,
                        m_Circle.m_dSx,m_Circle.m_dSy,
                        m_Circle.m_dEx,m_Circle.m_dEy);
            }
            else
            {
              pNewCircle=new CCircle(m_Circle.m_dCenterX,
                        m_Circle.m_dCenterY,m_Circle.m_dR,
                        m_Circle.m_nFillPattern);
            }
            pNewEntity=(CEntity*) pNewCircle;
            pNewCircle->Draw(pDC);

            x[cnt]=m_Circle.m_dCenterX+m_Circle.m_dR;
            y[cnt]=m_Circle.m_dCenterY+m_Circle.m_dR;
            cnt++;
```

```
            x[cnt] = m_Circle.m_dCenterX - m_Circle.m_dR;
            y[cnt] = m_Circle.m_dCenterY - m_Circle.m_dR;
            cnt ++ ;
        }
        break;
    case ID_DRAW_RECTANGLE:
    case ID_DRAW_FILLED_RECTANGLE:
        bIsCreated = m_Rectangle.Create(pDC,bSure);
        if(bIsCreated = = TRUE)
        {
            CRectangle * pNewRectangle;
            if(g_nCurTask = = ID_DRAW_RECTANGLE)
            {
              pNewRectangle = new CRectangle(m_Rectangle.m_dLeft,
                        m_Rectangle.m_dTop,m_Rectangle.m_dRight,
                        m_Rectangle.m_dBottom);
            }
            else
            {
              pNewRectangle = new CRectangle(m_Rectangle.m_dLeft,
                        m_Rectangle.m_dTop,m_Rectangle.m_dRight,
                        m_Rectangle.m_dBottom,m_Rectangle.m_nFillPattern);
            }
            pNewEntity = (CEntity * ) pNewRectangle;
            pNewRectangle - > Draw(pDC);

            x[cnt] = m_Rectangle.m_dLeft;
            y[cnt] = m_Rectangle.m_dTop;
            cnt ++ ;
            x[cnt] = m_Rectangle.m_dRight;
            y[cnt] = m_Rectangle.m_dBottom;
            cnt ++ ;
        }
        break;
    case ID_DRAW_POINT:
        bIsCreated = m_Pnt.Create(pDC,bSure);
        if(bIsCreated = = TRUE)
        {
            CPnt * pNewPnt = new CPnt(m_Pnt.m_dX,m_Pnt.m_dY);
            pNewEntity = (CEntity * ) pNewPnt;
            pNewPnt - > Draw(pDC);

            x[cnt] = m_Pnt.m_dX;
```

```
                y[cnt] = m_Pnt.m_dY;
                cnt++;
            }
            break;
        case ID_DRAW_TEXT:
            bIsCreated = m_Text.Create(pDC,bSure);
            if(bIsCreated == TRUE)
            {
                CText * pNewText = new CText(m_Text.m_dX,m_Text.m_dY,
                                             m_Text.m_strText);
                pNewEntity = (CEntity *) pNewText;
                pNewText -> Draw(pDC);

                x[cnt] = m_Text.m_dX;
                y[cnt] = m_Text.m_dY;
                cnt++;
            }
            break;
        default:
            break;
        }

        if(bIsCreated == TRUE&&pNewEntity != NULL)
        {
            pNewEntity -> m_pNext = m_pEntityHead;
            m_pEntityHead -> m_pPre = pNewEntity;
            m_pEntityHead = pNewEntity;

            //获取当前图形所在空间大小
            for(int i = 0;i < cnt;i++)
            {
                if(x[i] > g_dMaxX) g_dMaxX = x[i];
                if(x[i] < g_dMinX) g_dMinX = x[i];
                if(y[i] > g_dMaxY) g_dMaxY = y[i];
                if(y[i] < g_dMinY) g_dMinY = y[i];
            }
        }
    }
```

系统将根据 g_nCurTask 去调用各个对象的 Create()函数。各种实体对象的 Create()函数将在后面讲述“各类实体对象内部操纵实现层”中讨论。此时，读者只需认为各实体对象的 Create()可以一步步记录下创建该对象所需的数据信息即可。在实体创建好后，实体对象 Create()函数便返回 TRUE 值，其它时候均为 FALSE。在 Create()函数返回 TRUE 值时，便

分配新实体的存储空间，将该实体对象所需信息传入。例如：对于直线，便将存储在 m_Line 中的端点信息传给 pNewLine 所指向的对象。接下来，便把指向新实体的指针类型强制转化成 CEntity 类型的，存储在 pNewEntity 中，再将 pNewEntity 放到实体对象链表的头部即可。注意，在生成新实体后，调用了 Draw()函数。例如，对于直线即为 pNewLine->Draw()函数。这是为了用当前颜色和线型绘制出图形。

2.在重绘图形对象时，对实体对象链表进行遍历

在 CCadDocument 中有一个 Draw()函数。它用于重绘系统中所有的图形。这个函数在视类的 OnDraw()中调用。其实，Draw()函数的实现方法便是对实体对象链表的遍历过程。在 CCadDocument 中对 Draw()的定义如下：

```
//绘制系统中的所有实体
void CCadDoc::Draw(CDC * pDC)
{
    //遍历实体链表绘图
    CEntity* pEntity = m_pEntityHead;
    while(pEntity! =m_pEntityTail)
    {
        switch(pEntity->m_nType)
        {
        case ID_DRAW_LINE:
            ((CLine*) pEntity)->Draw(pDC);
            break;
        case ID_DRAW_CIRCLE:
        case ID_DRAW_ARC:
        case ID_DRAW_FILLED_CIRCLE:
            ((CCircle*) pEntity)->Draw(pDC);
            break;
        case ID_DRAW_RECTANGLE:
        case ID_DRAW_FILLED_RECTANGLE:
            ((CRectangle*) pEntity)->Draw(pDC);
            break;
        case ID_DRAW_POINT:
            ((CPnt*) pEntity)->Draw(pDC);
            break;
        case ID_DRAW_TEXT:
            ((CText*) pEntity)->Draw(pDC);
            break;
        default:
            break;
        }
        pEntity = pEntity->m_pNext;
    }
```

在函数中用 While 循环语句对链表进行遍历。最初的对象类型为 CEntity,之后根据 m_nType 判定实体类型,进行强制类型转换。再调用各类实体自己的 Draw()函数。当 pEntity 等于 m_pEntityTail 时便退出循环,不对链表尾部链节做任何处理(原因可参见 12.2.4 节。)

3.对象的删除、镜像、平移、旋转、比例缩放的实现方法

删除、镜像、平移、旋转、比例缩放几种操作都是对选择集的操作。还记得 12.2.4 一节中提到的 m_pEntitySelected[10]、m_nEntitySelCnt 变量吗!这两个变量即是用于选择集操作的。

在选定删除、镜像、平移、旋转、比例缩放操作后,系统会等待用户拾取进行这些操作的对象,即形成选择集。注意,所选择对象不能超过 10 个,否则系统会给出警告提示。在选择完对象后,按下鼠标右键。系统便会提示用户输入基准点,该基准点就存储在 g_dOrgX、g_dOrgY 变量中。这一点,我们曾在"系统的全局变量"一节中讲 g_dOrgX、g_dOrgY 的用途时提过。接下来,移动鼠标,便可以看到所选择对象在按要求变化。变化位置确定后,按下鼠标左键,系统便将选择集中的对象进行了相应的变化。需要指出的是,在删除对象时,按下鼠标右键的同时,操作便结束了。该操作不存在选择基准点与确定点的操作。

凑巧的是镜像、平移、旋转、比例缩放均需要用两个点来确定所需做的变化。对于镜像而言,这两个点决定了镜像的对称直线。对平移而言,这两个点之间的位移决定了平移的 X、Y 方向变化量。对于旋转而言,这两点连线与 X 轴正向的夹角为旋转角度。唯有比例综合缩放稍有不同。它以 g_dOrgX、g_dOrgY 记录的点作为比例缩放的位似变换中心,以确定点与基准点间的 X、Y 方向的距离与相对屏幕尺寸的某一固定值的商,取较小者作为缩放的比例因子。

讲了这么多,现在已经知道该对哪些对象进行操作,也知道该进行什么样的操作了。例如,要平移多远,要旋转多少度,要比例缩放多少倍。可是究竟应怎样去计算实现呢?这就涉及到了坐标变换。这里所说的坐标变换与前面提到的世界坐标与屏幕坐标之间的变换不同。这里所说的坐标变换是指点或对象、长度在世界坐标系内部的变换。这如同观察点不变,让空间中的实体对象相对地球变化(若以地球作为观察的参考系)。而屏幕坐标与世界坐标之间的变换却尤如实体相对地球不动,观察者绕着圈,从不同的角度、不同的距离位置去观察该实物。显示在屏幕上的图形,正尤如我们从不同位置对一个相对地球固定的物体拍摄的相片一样。从本质而言,这里的坐标变换指要改变实体数据值的变化,而从世界坐标到屏幕坐标的变化不改变实体的数据值。

究竟该怎样进行坐标变换。这已涉及到图形学中坐标变换的基本理论,需要相当的数字知识。考虑到读者在这个差异可能较大,再加之篇幅限制,故只在附录中列出这里用到的坐标变换的基本矩阵。感兴趣的读者可以查阅相关书籍体会坐标变化。一般的读者仅使用后面给出的结论即可。

知道该怎样变化后,我们只要以选择集中实体为标准,将其经过上述坐标变换。坐标变换将新数据放在为之分配的新对象中。在选定第二个点后,修改操作结束。系统将重新绘制新生成对象。读者或许会问,新对象是何时分配的呢?新对象是在选择好选择集,按下鼠

标右键时利用 AddEntity()函数加入的。

文档类中与删除、镜像、平移、旋转、比例缩放相关的函数如下。

1.用于删除链表中实体对象的函数

```
void CCadDoc::DeleteEntity(CDC * pDC)
{
    CEntity * pEntity = NULL;
    CEntity * pEntity2 = NULL;

    //删除实体链表中选择集记录的实体链节
    for(int i = 0;i < m_nEntitySelCnt;i ++ )
    {
        pEntity = m_pEntitySelected[i];

        switch(pEntity - > m_nType)
        {
        case ID_DRAW_LINE:
            ((CLine * )pEntity) - > Draw(pDC,RGB(255,255,255),PS_SOLID);
            break;
        case ID_DRAW_CIRCLE:
        case ID_DRAW_ARC:
        case ID_DRAW_FILLED_CIRCLE:
            ((CCircle * )pEntity) - > Draw(pDC,RGB(255,255,255),
                PS_SOLID);
            break;
        case ID_DRAW_RECTANGLE:
        case ID_DRAW_FILLED_RECTANGLE:
            ((CRectangle * )pEntity) - > Draw(pDC,RGB(255,255,255),
                                        PS_SCLID);
            break;
        case ID_DRAW_POINT:
            ((CPnt * )pEntity) - > Draw(pDC,RGB(255,255,255),PS_SOLID);
            break;
        case ID_DRAW_TEXT:
            ((CText * )pEntity) - > Draw(pDC,RGB(255,255,255),PS_SOLID);
            break;
        default:
            break;
        }

        if(pEntity! = m_pEntityHead)
        {
            pEntity2 = pEntity - > m_pNext;
```

```
                pEntity2->m_pPre = pEntity->m_pPre;
                pEntity2 = pEntity->m_pPre;
                pEntity2->m_pNext = pEntity->m_pNext;
            }
            else
            {
                m_pEntityHead = m_pEntityHead->m_pNext;
            }

            delete pEntity;
        }

        m_pCurEntity = m_pEntityHead;
        m_nEntitySelCnt = 0;
    }
```

2.在实体平移、旋转、缩放时,用于设置坐标变换矩阵的函数

```
void CCadDoc::ModifyMove(CDC * pDC)
{
    if(g_nStep == 0) return;

    switch(g_nCurTask)
    {
    case ID_MODIFY_MIRROR:
        if(g_nStep == 2)
        {
            InitChgA();
            double a = g_dCurX - g_dOrgX;
            double b = g_dCurY - g_dOrgY;
            double c = sqrt(a * a + b * b);
            double sin_theta = b/c;
            double cos_theta = a/c;

            m_dChgA[0][0] = cos_theta * cos_theta - sin_theta * sin_theta;
            m_dChgA[0][1] = 2 * sin_theta * cos_theta;
            m_dChgA[1][0] = m_dChgA[0][1];;
            m_dChgA[1][1] = - m_dChgA[0][0];
            m_dChgA[2][0] = g_dOrgX * m_dChgA[1][1] -
                g_dOrgY * m_dChgA[0][1] + g_dOrgX;
            m_dChgA[2][1] = - g_dOrgX * m_dChgA[0][1] +
                g_dOrgY * m_dChgA[0][0] + g_dOrgY;
            ChangeEntity(pDC);
        }
```

```
        break;
    case ID_MODIFY_MOVE:
        if(g_nStep==2)
        {
            InitChgA();
            m_dChgA[2][0]=g_dCurX-g_dOrgX;
            m_dChgA[2][1]=g_dCurY-g_dOrgY;

            ChangeEntity(pDC);
        }
        break;
    case ID_MODIFY_ROTATE:
        if(g_nStep==2)
        {
            InitChgA();
            double a=g_dCurX-g_dOrgX;
            double b=g_dCurY-g_dOrgY;
            double c=sqrt(a*a+b*b);
            double sin_theta=b/c;
            double cos_theta=a/c;

            m_dChgA[0][0]=cos_theta;
            m_dChgA[0][1]=sin_theta;
            m_dChgA[1][0]=-sin_theta;
            m_dChgA[1][1]=cos_theta;
            m_dChgA[2][0]=(1.0-cos_theta)*g_dOrgX+
                          sin_theta*g_dOrgY;
            m_dChgA[2][1]=(1.0-cos_theta)*g_dOrgY-
                          sin_theta*g_dOrgX;
            ChangeEntity(pDC);
        }
        break;
    case ID_MODIFY_SCALE:
        if(g_nStep==2)
        {
            double scaleX=fabs(g_dCurX-g_dOrgX)/(20*g_pCurView->
                m_dScreenToWorld);
            double scaleY=fabs(g_dCurY-g_dOrgY)/(20*g_pCurView->
                m_dScreenToWorld);

            if(scaleX>scaleY) scaleY=scaleX;
            else  scaleX=scaleY;
```

```
                InitChgA();
                m_dChgA[0][0] = scaleX;
                m_dChgA[1][1] = scaleY;
                m_dChgA[2][0] = (1.0 - scaleX) * g_dOrgX;
                m_dChgA[2][1] = (1.0 - scaleY) * g_dOrgY;

                ChangeEntity(pDC);
            }
            break;
        default:
            break;
        }
    }
```

#### 12.2.6.3 其他层

第三层,各类实体对象内部操纵实现层将在下一节中讲述。第四层,各类实体对象共同的基类已经在12.2.3节中讲过。它仅是一个用于建立链表的基类。

### 12.2.7 各类实体对象内部基本操作的实现

在CLine、CCircle、CRectangle、CPnt、CText五个类中,除了成员变量略有不同外,其中的成员函数的名称、个数以及用途完全相同。为了介绍方便,我们仅以CLine为例介绍这几个类的实现方法。

#### 12.2.7.1 Create()函数

该函数用于创建新的直线,其定义如下:

```
BOOL CLine::Create(CDC * pDC,BOOL bSure)
{
    BOOL  ret = FALSE;

    if(bSure = = TRUE)
    {
        CStatusBar * pStatus = (CStatusBar * )
            AfxGetApp() - > m_pMainWnd - >
                GetDescendantWindow(ID_VIEW_STATUS_BAR);
        ASSERT(pStatus);

        switch(g_nStep)
        {
        case 0:
            m_dSx = g_dCurX;
            m_dSy = g_dCurY;
            g_nStep ++ ;
```

```
                pStatus->SetPaneText(0,"Enter the end point");
                break;
            case 1:
                m_dEx = g_dCurX;
                m_dEy = g_dCurY;
                g_nStep = 0;
                pStatus->SetPaneText(0,"Enter the first point");
                ret = TRUE;
                break;
            default:
                break;
            }
        }
        else
        {
            if(g_nStep! =1) return FALSE;
            //设置 R2_NOT 的绘图方式
            int nDrawMode = pDC->SetROP2(R2_NOT);

            CPoint      orgPoint,curPoint,prePoint;
            g_pCurView->WorldToScreen(orgPoint,g_dOrgX,g_dOrgY);
            g_pCurView->WorldToScreen(curPoint,g_dCurX,g_dCurY);
            g_pCurView->WorldToScreen(prePoint,g_dPreX,g_dPreY);
            //覆盖先前的线
            pDC->MoveTo(orgPoint);
            pDC->LineTo(prePoint);
            //绘制新的线
            pDC->MoveTo(orgPoint);
            pDC->LineTo(curPoint);
            //设置先前的绘图方式
            pDC->SetROP2(nDrawMode);
        }
        return ret;
    }
```

其中 pDC 为设备环境表指针，bSure 的值为鼠标的状态。单移动鼠标时，bSure 为 FALSE；在按下鼠标键时，bSure 的值为 TRUE。读者可参看上一节中与鼠标消息相关的函数 OnMouseMove()、OnLButtonDown()以及 OnRButtonDown()，注意在调用文档类的 Create()函数时，bSure 的输入参数值。

在该函数中根据 bSure 的值分为两大部分。一部分是 bSure 为 FALSE 时，根据是否已选定了直线的起点，决定是否绘制橡皮筋线。另一部分是 bSure 为 TRUE 时，一步步记录绘制直线所需信息。例如，第 0 步时记录起点，步骤加 1。在第 1 步时又记录终点，同时将步骤置

为0,以便绘制下一条直线。同时,还将 ret 值设置为 TRUE,以通知文档类中的 Create()函数,已生成了一条直线,可以记录它,并将它加入实体对象链表中了。接下来,便等待绘制下一条直线,直至取消该任务为止,即修改掉 g_nCurTask 的值,令其不再为 ID_DRAW_LINE。

12.2.7.2 Draw()函数

Draw()函数用于对象自我绘制。对直线而言,调用它便可将自己一条直线绘出。Draw 函数的定义如下:

```
void CLine::Draw(CDC * pDC,UINT nColor,UINT nLineType)
{
    CPen * pNewPen = new CPen;
    if(nColor = = 0&&nLineType = = 0)
        pNewPen - > CreatePen(m_nLineType,1,m_nColor);
    else
        pNewPen - > CreatePen(nLineType,1,nColor);

    CPen * pOldPen = pDC - > SelectObject( pNewPen );

    CPoint      point;
    g_pCurView - > WorldToScreen(point,m_dSx,m_dSy);
    pDC - > MoveTo(point);
    g_pCurView - > WorldToScreen(point,m_dEx,m_dEy);
    pDC - > LineTo(point);

    pDC - > SelectObject( pOldPen );
}
```

在 Draw 函数中有两个缺省输入变量 nLineType 和 nColor。这两个变量是为了在选中该对象时,用红色的虚线绘制该对象以提示用户。在缺省情况下,它们均取实体对象自身的颜色和线型。

12.2.7.3 Pick()函数

Pick()函数用于判定该实体对象是否被选中。该函数的定义如下:

```
BOOL CLine::Pick()
{
    BOOL  ret = FALSE;
    double      x = g_dCurX - m_dSx;
    double      y = g_dCurY - m_dSy;

    double      a = m_dEx - m_dSx;
    double      b = m_dEy - m_dSy;
    double      c = sqrt(a * a + b * b);
```

```
    double      sin_theta = b/c;
    double      cos_theta = a/c;

    double      chg_x = x * cos_theta + y * sin_theta;
    double      chg_y = -x * sin_theta + y * cos_theta;

    double      sd = 5.0 * g_pCurView->m_dScreenToWorld;
    if(fabs(chg_y) < sd&&(chg_x >= 0&&chg_x <= c))
        ret = TRUE;

    return ret;
}
```

要判定当前鼠标点是否在直线上，或者在其附近，这有一定难度。在上面的程序段中，又用到了坐标变换的思想。有些读者可能不易于理解，这没有关系，知道大概是这么做的就行了。具体思路如下：如图 12.5(a)示，要判断点 P 是否在 SE 直线附近。这比较困难。但若以 S 点为原点，将 E 点旋转到 X 轴上变为如图 12.5(b)所示。此时，再判断 P 点是否在 SE 附近，这就很简单了。由图 12.5(a)变化至图 12.5(b)的实质就是：将 P 点绕 S 点旋转负 θ 度。S 点自然为(0,0)，而 E 点则变(d,0)其中 d 代表 SE 的距离。要在 b 中判断 P(px,py)点是否在 SE 附近，只用判断 px 的绝对值是否小于规定的域值，py 是否在 0 与 d 之间即可。

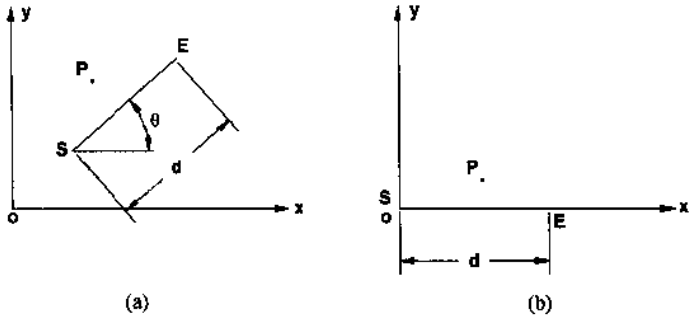



图 12.5　利用坐标旋转判断点与直线的位置关系

#### 12.2.7.4　OSnap()函数

OSnap()函数用于对象捕捉。在何时何处调用它，这将在“改变观察视图以及输入预处理”中讲解。该函数的定义如下：

```
BOOL CLine::OSnap()
{
    double      d;
    double      sd = g_nOSnapSize * g_pCurView->m_dScreenToWorld;
    double      x[3],y[3];
    int         cnt = 0;
    BOOL        ret = FALSE;
```

```
switch(g_nCpqTask)
{
case ID_OSNAP_ENDPOINT:
      x[cnt] = m_dSx;
      y[cnt] = m_dSy;
      cnt ++ ;
      x[cnt] = m_dEx;
      y[cnt] = m_dEy;
      cnt ++ ;
      break;
case ID_OSNAP_MIDPOINT:
      x[cnt] = (m_dSx + m_dEx)/2.0;
      y[cnt] = (m_dSy + m_dEy)/2.0;
      cnt ++ ;
      break;
default:
      if((g_nOSnapType&1) = = 1)
      {
            x[cnt] = m_dSx;
            y[cnt] = m_dSy;
            cnt ++ ;
            x[cnt] = m_dEx;
            y[cnt] = m_dEy;
            cnt ++ ;
      }
      if((g_nOSnapType&2) = = 2)
      {
            x[cnt] = (m_dSx + m_dEx)/2.0;
            y[cnt] = (m_dSy + m_dEy)/2.0;
            cnt ++ ;
      }
      break;
}

for(int i = 0;i < cnt;i ++ )
{
      d = GetTwoPntDis2(g_dCurX,g_dCurY,x[i],y[i]);
      if(d < sd * sd)
      {
            g_dCurX = x[i];
            g_dCurY = y[i];
            ret = TRUE;
```

```
                    break;
                }
            }

            return ret;
        }
```

设计该函数的巧妙之处，或许大家都注意到了，在函数中有两个数组 x[3]，y[3]以及 cnt。这一组变量用于在 switch 语句块中记录待检查点。将这些点统计到一起后，再用随后的循环语句判断该对象中哪个点被选中了。试想想，若不用 x、y 数组以及 cnt 计数，每个点都调用一次 GetTwoPntDis2()，再判断所得距离平方是否小于规定值，这代码将有多大的冗余。在选中对象捕捉点后，程序会将 x、y 数组中的对应值赋给 g_dCurX、g_dCurY，这便是对象捕捉能实现的秘密所在。在捕捉到点时，函数返回 TRUE，否则为 FALSE。

此外，还需要说明的是 switch 语句块中的 default 分支。别的分支都是根据辅助（透明）任务 g_nOpqTask 的当前欲捕捉点类型进行取点的。唯有 default 分支是 g_bOSnap 为 TRUE 时，根据 g_nOSnapType 所记录的捕捉点类型选取待检查点。g_nOSnapType 是按位表示的，即从右数第 1 位是否为 1 来代表是否捕捉端点。从右数第 2 位是否为 1 来代表是否捕捉中点。从右数第 3 位是否为 1，代表是否捕捉圆心。从右数第 4 位是否为 1，代表是否捕捉象限点。

在另外 4 个类中也有相同的函数。它们的功能以及用法与此相同。由于 CCircle 和 CRectargle 类中分别表示三种、两种类型的实体对象，所以它们在 Create()、Draw()、Pick()、OSnap()中与另外三个类（CLine、CPnt、CText）略有不同。它们还需要根据 m_nType 或者 g_nCurTask 来进一步区分实体对象。由于这些相对简单，故不再赘述。

### 12.2.8　改变观察视图和输入预处理

这一节主要讲解两部分内容。即观察视图是如何缩放、上下左右平移的以及鼠标输入点是如何进行预处理，以便实现 Grid、OSnap 以及 Ortho 功能的。

#### 12.2.8.1　改变观察视图

其实观察视图变换的实质是世界坐标与屏幕坐标的映射关系。关于这一点，在 12.2.6 节中讲述实体对象的镜像、平移、旋转和比例缩放时曾提到过。而且我们也知道了这与在世界坐标系中对实体对象进行坐标变换是不同的。在 12.2.5 节中讲与视类相关的全局变量时，也曾提过世界坐标是通过一个比例因子和一个世界坐标原点与屏幕坐标原点的相互关系，按一种方法进行映射变换的。接下来，我们将具体讲述实现方法。

假设取一张地图作为我们的二维实体对象世界。为了观察该地图，我们可以从中任取一块，在放大若干倍后进行观察。我们所选取的这个观察块即相当于我们的屏幕视图。至于这一块选在何处，这就是屏幕坐标原点与世界坐标原点的对应关系。在本系统中以视图中心为屏幕系统的原点，它正与世界坐标中由视类的 m_dOrgX、m_dOrgY 变量所表示的点相对应。这个放大倍数即是世界坐标中的长度大小与屏幕坐标中的长度大小的对应关系，

即世界中的1米长度相当于屏幕坐标中的多少个像素点长度。这个比例关系由视类的 m_dWorldToClient、m_dClientToWorld 变量表示。

关于世界坐标与屏幕坐标间的变化，由视类的 WorldToScreen()和 ScreenToWorld()函数来实现，它们的定义如下所示：

1.用于从屏幕坐标到世界坐标的函数

```
void CCadView::ScreenToWorld(double& x, double& y, CPoint& point)
{
    //获取当前客户区的大小
    CRect rect;
    GetClientRect(rect);
    int         nSOrgX=(rect.left+rect.right)/2;
    int         nSOrgY=(rect.top+rect.bottom)/2;

    x=(point.x-nSOrgX)*m_dScreenToWorld+m_dOrgX;
    y=(nSOrgY-point.y)*m_dScreenToWorld+m_dOrgY;

    if(g_bSnap==TRUE)
    {
        if(g_dSnap<1e-10) g_dSnap=0.5;
        long n;
        n=x/g_dSnap;
        x=n*g_dSnap;
        n=y/g_dSnap;
        y=n*g_dSnap;
        WorldToScreen(point,x,y);
    }
    if(g_bOrtho==TRUE&&g_nStep!=0)
    {
        if(fabs(g_dCurX-g_dOrgX)>fabs(g_dCurY-g_dCrgY))
            y=g_dOrgY;
        else
            x=g_dOrgX;
    }
}
```

2.用于从世界坐标到屏幕坐标的函数

```
void CCadView::WorldToScreen(CPoint& point, double x, double y)
{
    //获取当前客户区的大小
    CRect rect;
    GetClientRect(rect);
    int         nSOrgX=(rect.left+rect.right)/2;
```

```
    int         nSOrgY = (rect.top + rect.bottom)/2;

    point.x = (int) (x - m_dOrgX) * m_dWorldToScreen + nSOrgX;
    point.y = (int) nSOrgY - (y - m_dOrgY) * m_dWorldToScreen;
}
```

在调用各类实体对象的 Draw()函数绘制图形时，Draw()函数会自动调用 WorldToScreen()函数来将世界坐标转化为屏幕坐标，再绘图。这也是为何要设置指向视类的全局变量 g_pCurView 的原因所在。参见 12.2.5 节系统中的全局变量。

因此，缩放图形只用改变 m_dScreenToWorld 和 m_dWorldToScreen 变量，上下左右平移视图只用改变 m_dOriginX 和 m_dOriginY，再调用一次 Redraw()函数将图形从头至尾绘制一遍即可。参见 12.2.6 一节中的 OnKeyDown 函数以及 OnZoomCommand 函数便明白了。

#### 12.2.8.2 输入预处理

接下来，我们将学习本节的第二部分——输入预处理。前面在 12.2.5 节“系统的全局变量”中曾提到过，为何记录鼠标位置的六个变量全是 double 型的值。这是因为在 OnMouseMove()、OnLButtonDown()中我们用视类的 ScreenToWorld()函数将整数型的屏幕坐标转换成了双精度型的世界坐标。在绘图记录实体对象的几何信息时，我们应当保存的是它的世界坐标信息，而非屏幕坐标信息。由上面讲述的观察视图可以随意变换，我们知道保存屏幕坐标是不可靠的。

既然有了从屏幕坐标往世界坐标的变换，我们便可根据系统的当前状态 m_bSnap、m_bGrid 来灵活地处理输入的 g_dCurX、g_dCurY 值。在 m_bSnap 为 TRUE 时，我们必须令 g_dCurX、g_dCurY 为 g_dSnapSize 的整数倍。此时，也可能会修改到输入的 point 值。因而 point 以引用方式输入。当 m_bGrid 为 TRUE 时，正交模式打开，只能给出水平线和垂直线，此时应根据 X、Y 方向哪个坐标值变化大，来将 g_dOrgX 或 g_OrgY 赋给 g_dCurX 或 g_dCurY，从而实现正交化绘图。当然，正交化绘图不只用于绘直线时。如水平或垂直平移实体对象时也可能用到。参见本节前面的 ScreenToWorld()函数，系统是如何实现输入预处理的。

此外，对于输入预处理还有对象捕捉。它是用辅助(透明)任务实现的。实现方法也是在选中点对象后，将 g_dCurX、g_dCurY 变量设置为选择点的坐标。

总之，灵活地处理 g_dCurX、g_dCurY 变量的值对于实现一些辅助功能大有益处。

至此，我们已了解了此应用程序的设计思路，也学习了其中的一些技巧。接下来，我们该动手制作自己的应用程序了。我将在下面的章节中引导用户实现上述功能。希望读者能在制作时体会到其中的乐趣。至于有些片断程序读不明白，可以暂时搁下，把握住 12.2 节所述的总体思路才是主要的。

## 12.3 应用程序界面部分的制作

这一节将引导读者制作该应用程序的界面。本书的重点之一是引导读者制作灵活多样

的界面,以便使应用程序用起来更加方便、高效,实现界面与功能的完美结合。在前面章节中,我们已经学习了界面的制作。在这一节,我们将把前面的知识综合起来,在原有的基础上进一步提高。

说明:由于该应用程序功能众多,程序量庞大,故在此只说明编制该应用程序的步骤。读者可以参照本书配套光盘所提供的程序,完成该应用程序。

### 12.3.1 创建应用程序

利用 AppWizard 创建单文档的 MFC AppWizard(exe)应用程序。在第 4 步中选择 Advanced 按钮,弹出 Advanced Options 对话框。在 File extension:中输入 dat。切换到 Window Styles 标签中,在其中选择 Usesplit Window 复选框。在 Main frame styles 中,选择 Maximized 复选框。

在生成 Cad 项目后,编译并运行该程序,注意看窗口是否在启动时就呈最大状态,View 菜单下的 Split 菜单项以及在选择 File \ Open 菜单项时,出现的 Open 对话框的文件类型(*.dat)。

### 12.3.2 改变鼠标的形状

参考 Chap5B 项目,编写改变鼠标光标形状的程序段。为了使程序较有条理,在 CCadView 中添加 DrawCurssor()方法和 DrawSys()方法。将 Chap5BView::MouseMove()中与绘制鼠标光标相关的代码拷贝至 DrawCursor 中,并修改 CCadView::MouseMove()函数。

将 Chap5BView::OnDraw 中的代码拷贝至 CCadView::Drawsys()中,并修改 CCadView::OnDraw()函数。

m_prePoint 变量的定义以及赋初始值参见 Chap5B 项目。

### 12.3.3 编写系统的菜单

从 ResourceView 中编写 IDR_MAINFRAME 菜单,如图 12.6 所示。

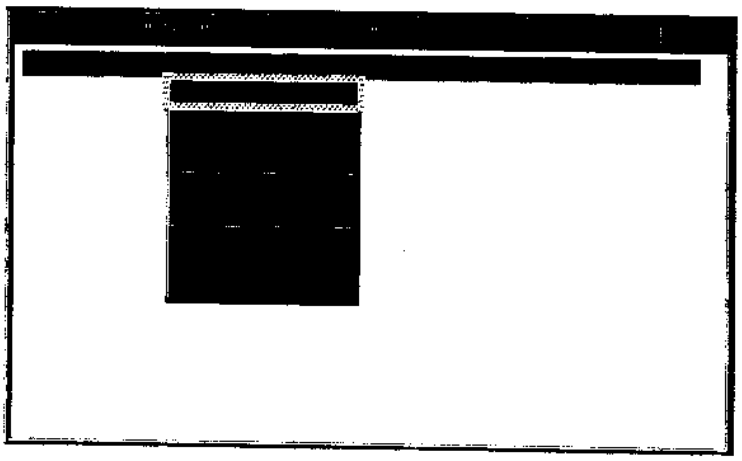


图 12.6 IDR_MAINFRAME 菜单

| 菜单 | 菜单项 | 提示 Prompt | 命令 ID |
|---|---|---|---|
| &Draw | &Line | Draw a line \ nLine | ID_DRAW_LINE |
| &Draw | &Circle | Draw a circle \ nCircle | ID_DRAW_CIRCLE |
| &Draw | &Arc | Draw an arc \ nArc | ID_DRAW_ARC |
| &Draw | &Rectangle | Draw a rectangle \ nRectangle | ID_DRAW_RECTANGLE |
| &Draw | SEPARATOR | | |
| &Draw | Filled C&ircle | Draw a filled circle \ nFilled Circle | ID_DRAW_FILLED_CIRCLE |
| &Draw | Filled R&ectangle | Draw a filled rectangle \ nFilled Rectangle | ID_DRAW_FILLED_RECTANGLE |
| &Draw | SEPARATOR | | |
| &Draw | &Point | Draw a point \ nPoint | ID_DRAW_POINT |
| &Draw | &Text | Write text \ nText | ID_DRAW_TEXT |
| &Modify | &Erase | Erase objects \ nErase | ID_MODIFY_ERASE |
| &Modify | Mi&rror | Mirror objects \ nMirror | ID_MODIFY_MIRROR |
| &Modify | Mo&ve | Move objects \ nMove | ID_MODIFY_MOVE |
| &Modify | &Rotate | Rotate objects \ nRotate | ID_MODIFY_ROTATE |
| &Modify | &Scale | Scale a object \ nScale | ID_MODIFY_SCALE |
| &OSnap | &point | Snap an point \ npoint | ID_OSNAP_POINT |
| &OSnap | &Midpoint | Snap a midpoint \ nMidpoint | ID_OSNAP_MIDPOINT |
| &OSnap | &Center | Snap a center \ nCenter | ID_OSNAP_CENTER |
| &OSnap | &Quardant | Snap a quardant \ nQuardant | ID_OSNAP_QUARDANT |
| &Zoom | &Window | Zoom by a window \ nWindow | ID_ZOOM_WINDOW |
| &Zoom | Zoom &In | Zoom in \ nZoom In | ID_ZOOM_IN |
| &Zoom | Zoom &Out | Zoom out \ nZoom Out | ID_ZOOM_OUT |
| &Zoom | Zoom &All | Zoom all \ nZoom All | ID_ZOOM_ALL |

### 12.3.4　应用程序的工具条的制作

1.从 ResourceView 中添加如图 12.7 所示工具条 IDR_DRAW。

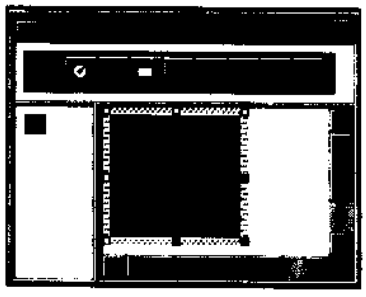


图 12.7　IDR_DRAW 工具条

2.从 ResourceView 中添加如图 12.8 所示工具条 IDR_MODIFY。



图 12.8 IDR_MODIFY 工具条

3.从 ResourceView 中添加如图 12.9 所示工具条 IDR_ZOOM。

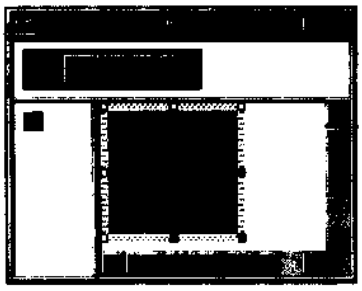


图 12.9 IDR_ZOOM 工具条

4.从 ResourceView 中添加如图 12.10 所示工具条 IDR_OSNAP。



图 12.10 IDR_OSNAP 工具条

5.从 ReourceView 中添加如图 12.11 所示工具条 IDR_PROPERTY。

6.从 ReourceView 中添加下列字符串。

| 对象 ID 值 | Caption |
|---|---|
| IDD_SETDLG | Set system's status \ nSet System |
| IDD_REDRAW | Redraw the screen \ nRedraw |

7.从 resource.h 中删除对 ID_BUTTONXXXX 常量的定义。

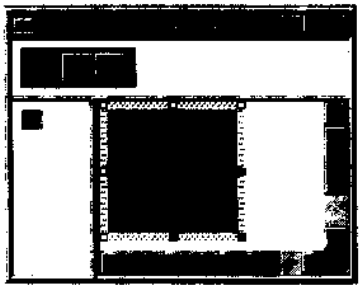


图 12.11　IDR_PROPERTY 工具条

8.以 Text 方式打开 Cad.rc,修改对工具条的定义,如下所示。

```
//////////////////////////////////////////////////////////////////
//
// Toolbar
//

IDR_DRAW_BAR TOOLBAR DISCARDABLE  16, 15
BEGIN
    BUTTON      ID_DRAW_LINE
    BUTTON      ID_DRAW_CIRCLE
    BUTTON      ID_DRAW_ARC
    BUTTON      ID_DRAW_RECTANGLE
    BUTTON      ID_DRAW_FILLED_CIRCLE
    BUTTON      ID_DRAW_FILLED_RECTANGLE
    BUTTON      ID_DRAW_POINT
    BUTTON      ID_DRAW_TEXT
END

IDR_OSNAP_BAR TOOLBAR DISCARDABLE  16, 15
BEGIN
    BUTTON      ID_OSNAP_ENDPOINT
    BUTTON      ID_OSNAP_MIDPOINT
    BUTTON      ID_OSNAP_CENTER
    BUTTON      ID_OSNAP_QUARDANT
END

IDR_ZOOM_BAR TOOLBAR DISCARDABLE  16, 15
BEGIN
    BUTTON      ID_ZOOM_WINDOW
    BUTTON      ID_ZOOM_IN
    BUTTON      ID_ZOOM_OUT
```

```
    BUTTON      ID_ZOOM_ALL
END

IDR_MODIFY_BAR TOOLBAR DISCARDABLE  16, 15
BEGIN
    BUTTON      ID_MODIFY_ERASE
    BUTTON      ID_EDIT_COPY
    BUTTON      ID_EDIT_CUT
    BUTTON      ID_EDIT_PASTE
    SEPARATOR
    BUTTON      ID_MODIFY_MIRROR
    BUTTON      ID_MODIFY_MOVE
    BUTTON      ID_MODIFY_ROTATE
    BUTTON      ID_MODIFY_SCALE
END

IDR_PROPERTY_BAR TOOLBAR DISCARDABLE  16, 15
BEGIN
    BUTTON      ID_PROPERTY
    BUTTON      ID_COLOR
END
```




第7、8步也可免去不做。未做7、8步时，不能用LoadToolBar()函数加载菜单。

9.将Chap7项目中的Property.cpp、Property.h文件拷贝至此项目所在文件夹中，并打开Property.cpp，将包括的头文件chap7.h改为cad.h。编译该文件，将其加入到cad项目中。

10.在MainFrm.cpp中添加与工具条相关的代码，如下所示：

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    if (! m_wndToolBar.Create(this) ||
        ! m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
    {
        TRACE0("Failed to create toolbar\n");
        return -1;      // fail to create
    }
    m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle() |
        CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC);
```

```
if (! m_wndDrawBar.Create(this, WS_CHILD | WS_VISIBLE |
      CBRS_SIZE_DYNAMIC |     CBRS_TOP | CBRS_TOOLTIPS | CBRS_FLYBY ,
      IDR_DRAW_BAR) ||
      ! m_wndDrawBar.LoadBitmap(IDR_DRAW_BAR)||
      ! m_wndDrawBar.SetButtons(DrawButtons,
           sizeof(DrawButtons)/sizeof(UINT)))
{
      TRACE0("Failed to create toolbar\n");
      return -1;      // fail to create
}

if (! m_wndModifyBar.Create(this, WS_CHILD | WS_VISIBLE |
      CBRS_SIZE_DYNAMIC |     CBRS_TOP | CBRS_TOOLTIPS | CBRS_FLYBY ,
      IDR_MODIFY_BAR) ||
      ! m_wndModifyBar.LoadBitmap(IDR_MODIFY_BAR)||
      ! m_wndModifyBar.SetButtons(ModifyButtons,
           sizeof(ModifyButtons)/sizeof(UINT)))
{
      TRACE0("Failed to create toolbar\n");
      return -1;      // fail to create
}

if (! m_wndOSnapBar.Create(this, WS_CHILD | WS_VISIBLE |
      CBRS_SIZE_DYNAMIC |     CBRS_TOP | CBRS_TOOLTIPS | CBRS_FLYBY ,
      IDR_CSNAP_BAR) ||
      ! m_wndOSnapBar.LoadBitmap(IDR_OSNAP_BAR)||
      ! m_wndOSnapBar.SetButtons(OSnapButtons,
           sizeof(OSnapButtons)/sizeof(UINT)))
{
      TRACE0("Failed to create toolbar\n");
      return -1;      // fail to create
}

if (! m_wndZoomBar.Create(this, WS_CHILD | WS_VISIBLE |
      CBRS_SIZE_DYNAMIC |     CBRS_TOP | CBRS_TOOLTIPS | CBRS_FLYBY ,
      IDR_ZOOM_BAR) ||
      ! m_wndZoomBar.LoadBitmap(IDR_ZOOM_BAR)||
      ! m_wndZoomBar.SetButtons(ZoomButtons,
           sizeof(ZoomButtons)/sizeof(UINT)))
{
      TRACE0("Failed to create toolbar\n");
      return -1;      // fail to create
}
```

```
    if (! CreatePropertyBar())
        return -1;

    if(! m_wndStatusBar.Init(this))
        return -1;

    m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndToolBar);
    m_wndToolBar.SetWindowText("Standard");

    m_wndDrawBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndDrawBar);
    m_wndDrawBar.SetWindowText("Draw");

    m_wndModifyBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndModifyBar);
    m_wndModifyBar.SetWindowText("Modify");

    m_wndOSnapBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndOSnapBar);
    m_wndOSnapBar.SetWindowText("Object Snap");

    m_wndZoomBar.EnableDocking(CBRS_ALIGN_ANY);
    EnableDocking(CBRS_ALIGN_ANY);
    DockControlBar(&m_wndZoomBar);
    m_wndZoomBar.SetWindowText("Zoom");

    m_wndPropertyBar.EnableDocking(CBRS_ALIGN_TOP|CBRS_ALIGN_BOTTOM);
    EnableDocking(CBRS_ALIGN_TOP|CBRS_ALIGN_BOTTOM);
    DockControlBar(&m_wndPropertyBar);
    m_wndPropertyBar.SetWindowText("Property");

    // CG: The following line was added by the Splash Screen component.
CSplashWnd::ShowSplashScreen(this);
    return 0;
}
```

函数也可以从 Chap7 项目的 MainFrm.cpp 中拷贝，再修改。

### 12.3.5　应用程序的状态栏的制作

1.从 Chap8 项目中将 MyStatusBar.cpp 和 MYStatusBar.h 文件拷贝至该项目所在目录中。打开 MyStatusBar.cpp 文件,将包含的头文件 Chap8.h 改为 cad.h。编译该文件,将其加入到项目中。

2.在 MystatusBar.cpp 文件的 CMystatusBar::CMystatusBar 构造函数中将为 m_bSnap、m_bOSnap、m_bGrid 赋值的语句删除,同时在 MyStatusBar.h 中将对这三个变量的定义删除。再回到 MyStatusBar.cpp 文件中,将上面三个变量分别替换为 g_bSnap、g_bOSnap 和 g_bGrid。

3.修改 CMainFrame::OnCreate()函数,在其中添加关于状态栏的定义。

### 12.3.6　应用程序的对话框的制作

该应用程序用到三个对话框,一个为文本输入对话框,另外两个为第九章中所讲的高级对话框和属性对话框。

文本输入对话框相对简单,读者可以参照第九章中"普通对话框的制作"一节来完成。至于第九章中的两个对话框,读者可以参考"高级对话框的制作"和"属性对话框的制作"这两节再做一遍,也可以按下列步骤完成。

1.从 Chap9 项目的 resource.h 文件中将下列内容拷贝至 cad 项目的 resource.h 文件中,同时将_APS_NEXT_CONTROL_VALUE 后面的数值改为 1054。

```
#define IDC_EDIT_PATTERN                1003
#define IDC_LIST_PATTERN                1004
#define IDC_BUTTON_PICK                 1006
#define IDC_SLIDER                      1007
#define IDC_BUTTON_MORE                 1008
#define IDC_SELECTION                   1013
#define IDC_PATTERN3                    1021
#define IDC_PATTERN2                    1022
#define IDC_SCROLLBAR_RED               1022
#define IDC_PATTERN1                    1023
#define IDC_SCROLLBAR_GREEN             1023
#define IDC_PATTERN6                    1024
#define IDC_SCROLLBAR_BLUE              1024
#define IDC_PATTERN5                    1025
#define IDC_PATTERN4                    1026
#define IDC_STATIC_RED                  1026
#define IDC_STATIC_GREEN                1027
#define IDC_STATIC_BLUE                 1028
#define IDC_PICTURE_COLOR               1032
#define IDC_EDIT_X                      1033
#define IDC_SPIN_X                      1034
#define IDC_EDIT_Y                      1035
#define IDC_SPIN_Y                      1036
```

```
#define IDC_EDIT_Z                      1037
#define IDC_SPIN_Z                      1038
#define IDC_EDIT_WTOS                   1039
#define IDC_EDIT_STOW                   1040
#define IDC_GRID_ON                     1040
#define IDC_GRID_OFF                    1041
#define IDC_SNAP_ON                     1042
#define IDC_SNAP_OFF                    1043
#define IDC_ORTHO_ON                    1044
#define IDC_ORTHO_OFF                   1045
#define IDC_EDIT_GRID                   1046
#define IDC_ENDPOINT                    1047
#define IDC_EDIT_SNAP                   1048
#define IDC_MIDPOINT                    1048
#define IDC_CENTER                      1049
#define IDC_QUARDANT                    1050
#define IDC_OSNAP_SIZE                  1051
#define IDC_PICTURE_OSNAP               1052
#define IDC_EDIT_TEXT                   1053
```

2.从 Chap9 项目的 Chap9.rc 中将下面的关于对话框定义的内容拷贝至 cad.rc 中。注意摆放的位置。

```
/////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_DIALOG_FILL DIALOG DISCARDABLE  0, 0, 250, 185
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Fill a filled graph"
FONT 10, "System"
BEGIN
    DEFPUSHBUTTON   "&Fill",IDOK,195,25,40,13
    PUSHBUTTON      "&Cancel",IDCANCEL,195,44,40,13
    LTEXT           "Fill Pattern:",IDC_STATIC,10,5,40,10
    EDITTEXT        IDC_EDIT_PATTERN,50,5,60,10,ES_AUTOHSCROLL
    LISTBOX         IDC_LIST_PATTERN,10,20,100,55,LBS_SORT |
                    LBS_NOINTEGRALHEIGHT | WS_VSCROLL | WS_TABSTOP
    PUSHBUTTON      "&Pick< <",IDC_BUTTON_PICK,195,7,40,13
    CONTROL         "Slider1",IDC_SLIDER,"msctls_trackbar32",TBS_BOTH |
                    TBS_NOTICKS | NOT WS_VISIBLE,0,81,250,6
    PUSHBUTTON      "&More",IDC_BUTTON_MORE,195,62,40,13
    CONTROL         "",IDC_SELECTION,"Static",SS_BLACKFRAME,115,10,70,65
```

```
    CONTROL         "",IDC_PATTERN3,"Button",BS_OWNERDRAW | WS_TABSTOP,165,
                    90,65,40
    CONTROL         "",IDC_PATTERN2,"Button",BS_OWNERDRAW | WS_TABSTOP,90,
                    90,65,40
    CONTROL         "",IDC_PATTERN1,"Button",BS_OWNERDRAW | WS_TABSTOP,13,90,
                    65,40
    CONTROL         "",IDC_PATTERN6,"Button",BS_OWNERDRAW | WS_TABSTOP,165,
                    135,65,40
    CONTROL         "",IDC_PATTERN5,"Button",BS_OWNERDRAW | WS_TABSTOP,90,
                    135,65,40
    CONTROL         "",IDC_PATTERN4,"Button",BS_OWNERDRAW | WS_TABSTOP,13,
                    135,65,40
END

IDD_ORDINATEPAGE DIALOG DISCARDABLE  0, 0, 212, 95
STYLE WS_CHILD | WS_DISABLED | WS_CAPTION
CAPTION "Ordinate"
FONT 8, "MS Sans Serif"
BEGIN
    GROUPBOX        "Origin",IDC_STATIC,15,10,85,70
    LTEXT           "X:",IDC_STATIC,21,27,10,10
    EDITTEXT        IDC_EDIT_X,31,25,50,12,ES_AUTOHSCROLL
    CONTROL         "Spin1",IDC_SPIN_X,"msctls_updown32",UDS_ARROWKEYS,81,
                    25,10,10
    LTEXT           "Y:",IDC_STATIC,21,45,10,10
    EDITTEXT        IDC_EDIT_Y,31,43,50,12,ES_AUTOHSCROLL
    CONTROL         "Spin1",IDC_SPIN_Y,"msctls_updown32",UDS_ARROWKEYS,81,
                    43,10,10
    LTEXT           "Z:",IDC_STATIC,21,63,10,10
    EDITTEXT        IDC_EDIT_Z,31,61,50,12,ES_AUTOHSCROLL | WS_DISABLED
    CONTROL         "Spin1",IDC_SPIN_Z,"msctls_updown32",UDS_ARROWKEYS |
                    WS_DISABLED,81,61,10,10
    GROUPBOX        "Scale",IDC_STATIC,120,10,75,70
    LTEXT           "World To Screen:",IDC_STATIC,125,23,65,10
    EDITTEXT        IDC_EDIT_WTOS,125,35,65,10,ES_AUTOHSCROLL
    LTEXT           "Screen To World:",IDC_STATIC,125,50,65,10
    EDITTEXT        IDC_EDIT_STOW,125,62,65,10,ES_AUTOHSCROLL
END

IDD_SETPAGE DIALOG DISCARDABLE  0, 0, 197, 93
STYLE WS_CHILD | WS_DISABLED | WS_CAPTION
CAPTION "Set"
FONT 8, "MS Sans Serif"
```

```
BEGIN
    GROUPBOX    "Grid",IDC_STATIC,8,10,50,75
    CONTROL     "On",IDC_GRID_ON,"Button",BS_AUTORADIOBUTTON | WS_GROUP,
                13,22,25,10
    CONTROL     "Off",IDC_GRID_OFF,"Button",BS_AUTORADIOBUTTON,13,38,25,
                10
    GROUPBOX    "Snap",IDC_STATIC,73,10,50,75
    CONTROL     "On",IDC_SNAP_ON,"Button",BS_AUTORADIOBUTTON | WS_GROUP,
                78,22,25,10
    CONTROL     "Off",IDC_SNAP_OFF,"Button",BS_AUTORADIOBUTTON,78,38,25,
                10
    GROUPBOX    "Ortho",IDC_STATIC,138,10,50,45
    CONTROL     "On",IDC_ORTHO_ON,"Button",BS_AUTORADIOBUTTON | WS_GROUP,
                143,22,25,10
    CONTROL     "Off",IDC_ORTHO_OFF,"Button",BS_AUTORADIOBUTTON,143,38,
                25,10
    LTEXT       "Size:",IDC_STATIC,13,51,25,10
    EDITTEXT    IDC_EDIT_GRID,13,65,40,12,ES_AUTOHSCROLL
    LTEXT       "Size:",IDC_STATIC,78,50,25,10
    EDITTEXT    IDC_EDIT_SNAP,78,65,40,12,ES_AUTOHSCROLL
END

IDD_OSNAPPAGE DIALOG DISCARDABLE  0, 0, 183, 94
STYLE WS_CHILD | WS_DISABLED | WS_CAPTION
CAPTION "OSnap"
FONT 8, "MS Sans Serif"
BEGIN
    GROUPBOX    "Type",IDC_STATIC,10,10,75,70
    CONTROL     "Endpoint",IDC_ENDPOINT,"Button",BS_AUTOCHECKBOX |
                WS_TABSTOP,18,20,55,10
    CONTROL     "Middpoint",IDC_MIDPOINT,"Button",BS_AUTOCHECKBOX
                WS_TABSTOP,18,34,55,10
    CONTROL     "Center",IDC_CENTER,"Button",BS_AUTOCHECKBOX |
                WS_TABSTOP,18,48,55,10
    CONTROL     "Quardant",IDC_QUARDANT,"Button",BS_AUTOCHECKBOX |
                WS_TABSTOP,18,62,55,10
    CONTROL     "Slider1",IDC_OSNAP_SIZE,"msctls_trackbar32",TBS_BOTH |
                TBS_NOTICKS | WS_TABSTOP,105,65,70,10
    GROUPBOX    "Size",IDC_STATIC,100,10,80,70
    CONTROL     "",IDC_PICTURE_OSNAP,"Static",SS_BLACKFRAME,110,20,60,40
END

IDD_ENTER_TEXT DIALOG DISCARDABLE  0, 0, 159, 80
```

```
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Enter text"
FONT 10, "System"
BEGIN
    DEFPUSHBUTTON "OK",IDOK,19,54,50,14
    PUSHBUTTON  "Cancel",IDCANCEL,89,54,50,14
    LTEXT       "Enter text:",IDC_STATIC,10,10,35,10
    EDITTEXT    IDC_EDIT_TEXT,10,26,140,15,ES_AUTOHSCROLL
END
```

3.从 Chap9 项目所在文件夹中将 FillDlg.h 和 FillDlg.cpp 拷贝到该项目中。将 FillDlg.cpp 文件包含的 Chap9.h 头文件改为 cad.h。编译该文件,将其加入 cad 项目中。

4. 从 Chap9 项目所在文件夹中将 SetSheet.cpp、SetSheet.h、SetPage.cpp 和 SetPage.h 文件拷贝至 cad 项目所在目录中。将 SetSheet.cpp 和 SetPage.cpp 文件包含的 Chap9.h 头文件改为 cad.h。分别编译这两个文件,将其加入到 cad 项目中。

### 12.3.7 应用程序封面的制作

一般的商业程序都有封面。在此,我们也为我们的应用程序添加一个封面。

从菜单中选择 Project \ Add To Project \ Components and Controls 菜单项,弹出 Components and Controls Gallery。双击 Develper Studio Componerts 文件夹,选择 Splash Screen 组件,再单击 Close 按钮。Visual C++ 会往项目中添加封面。在 Cad 项目中添加了 CSplandWnd 类。

从 ResourceView 中选择 IDB_SPLASH 位图,编辑它,以设计读者自己的封面。笔者设计的封面,如图 12.12 所示。



图 12.12 应用程序的封面

修改 Csplash::OnCreate 函数,增加封面存在的延时(增至 7.5 秒),如下所示:

```
int CSplashWnd::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    // Center the window.
    CenterWindow();

    // Set a timer to destroy the splash screen.
    SetTimer(1, 7500, NULL);

    return 0;
}
```

关于该应用程序界面的制作到此为止。接下来,我们要编写该应用程序功能的实现代码。

## 12.4 应用程序功能的实现

在 12.2 节中,我们了解了该应用程序实现其功能的思路。接下来,将向读者讲述这些功能是如何一步步实现的。

在编写该程序时,作者是在定义好了各类实体的基本数据以后,按照系统的功能,从四个层次(参见 12.2.6 中系统的操作结构图)同时进行编程实现的。具体过程简介如下:

1.用 ClassWizard 在 cad 项目中添加 CLine、CCircle、CRectangle、CPnt 和 CText 类。修改类的继承关系,并在其中添加数据成员。

2.在视类中添加消息映射创建以及消息映射函数,并编写这些函数。

3.在视内中完成世界坐标系与屏幕坐标系的映射函数,并在 OnMouseMove 函数中通过往状态栏中显示鼠标当前位置检查世界坐标与屏幕坐标变换是否正确。此时,也将转换后的鼠标位置坐标值赋给了 g_dCurX、g_dCurY、g_dPreX、g_dPreY、g_dOrgX 以及 g_dOrgY。




此时尚未考虑到 Snap、Grid 功能,故 ScreenToClient 中没有最终存在的两个 if 语句块。

4.完成文档文件中最初建立链表的函数。

5.完成绘制图元功能。

6.完成对象捕捉功能。

7.完成修改功能。

8.完成缩放功能。

9.完成 OSnap、Snap、Grid 功能,将系统设置对话框与系统相连。

完成该程序的过程基本如上所述。在下一节中,将给出该应用程序的所有源文件。读

者可以参照它完成整个程序。

## 12.5　应用程序的源程序

作为对全书的总结,本章的 Cad 项目的源程序如下。

1. 资源文件 Cad.rc 和 Resource.h

Cad.rc 和 Resource.h 文件中含有与应用程序的资源相关的定义和说明。资源文件由资源编辑器进行编辑。读者也可以以文本方式对资源文件进行添加、删除以及修改处理。

**程序清单 12.1　资源头文件 Resource.h**

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by Cad.rc
//
#define IDD_ABOUTBOX                    100
#define IDB_SPLASH                      102
#define IDS_PROPSHT_CAPTION             103
#define IDR_MAINFRAME                   128
#define IDR_CADTYPE                     129
#define IDC_CURSOR1                     130
#define IDR_MODIFY_BAR                  131
#define IDR_OSNAP_BAR                   134
#define IDR_ZOOM_BAR                    136
#define IDR_DRAW_BAR                    140
#define IDR_PROPERTY_BAR                142
#define IDD_DIALOG_FILL                 145
#define IDD_CRDINATEPAGE                146
#define IDD_SETPAGE                     147
#define IDD_OSNAPPAGE                   148
#define IDD_ENTER_TEXT                  149
#define IDC_EDIT_PATTERN                1003
#define IDC_LIST_PATTERN                1004
#define IDC_BUTTON_PICK                 1006
#define IDC_SLIDER                      1007
#define IDC_BUTTON_MORE                 1008
#define IDC_SELECTION                   1013
#define IDC_PATTERN3                    1021
#define IDC_PATTERN2                    1022
#define IDC_SCROLLBAR_RED               1022
#define IDC_PATTERN1                    1023
#define IDC_SCROLLBAR_GREEN             1023
#define IDC_PATTERN6                    1024
```

```
# define IDC_SCROLLBAR_BLUE          1024
# define IDC_PATTERN5                1025
# define IDC_PATTERN4                1026
# define IDC_STATIC_RED              1026
# define IDC_STATIC_GREEN            1027
# define IDC_STATIC_BLUE             1028
# define IDC_PICTURE_COLOR           1032
# define IDC_EDIT_X                  1033
# define IDC_SPIN_X                  1034
# define IDC_EDIT_Y                  1035
# define IDC_SPIN_Y                  1036
# define IDC_EDIT_Z                  1037
# define IDC_SPIN_Z                  1038
# define IDC_EDIT_WTOS               1039
# define IDC_EDIT_STOW               1040
# define IDC_GRID_ON                 1040
# define IDC_GRID_OFF                1041
# define IDC_SNAP_ON                 1042
# define IDC_SNAP_OFF                1043
# define IDC_ORTHO_ON                1044
# define IDC_ORTHO_OFF               1045
# define IDC_EDIT_GRID               1046
# define IDC_ENDPOINT                1047
# define IDC_EDIT_SNAP               1048
# define IDC_MIDPOINT                1048
# define IDC_CENTER                  1049
# define IDC_QUARDANT                1050
# define IDC_OSNAP_SIZE              1051
# define IDC_PICTURE_OSNAP           1052
# define IDC_EDIT_TEXT               1053
# define ID_DRAW_LINE                32771
# define ID_DRAW_CIRCLE              32772
# define ID_DRAW_ARC                 32773
# define ID_DRAW_RECTANGLE           32774
# define ID_DRAW_FILLED_CIRCLE       32775
# define ID_DRAW_FILLED_RECTANGLE    32776
# define ID_DRAW_POINT               32777
# define ID_DRAW_TEXT                32778
# define ID_MODIFY_ERASE             32779
# define ID_MODIFY_MIRROR            32780
# define ID_MODIFY_MOVE              32781
# define ID_MODIFY_ROTATE            32782
# define ID_MODIFY_SCALE             32783
```

```
#define ID_OSNAP_ENDPOINT              32784
#define ID_OSNAP_MIDPOINT              32785
#define ID_OSNAP_CENTER                32786
#define ID_OSNAP_QUARDANT              32787
#define ID_ZOOM_WINDOW                 32788
#define ID_ZOOM_IN                     32789
#define ID_ZOOM_OUT                    32790
#define ID_ZOOM_ALL                    32791
#define ID_PROPERTY                    61446
#define ID_SETDLG                      61446
#define ID_COLOR                       61447
#define ID_REDRAW                      61447
#define IDW_COLOR                      61448
#define IDW_LINETYPE                   61449

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_3D_CONTROLS               1
#define _APS_NEXT_RESOURCE_VALUE       150
#define _APS_NEXT_COMMAND_VALUE        32792
#define _APS_NEXT_CONTROL_VALUE        1054
#define _APS_NEXT_SYMED_VALUE          104
#endif
#endif
```

**程序清单 12.2　资源文件 Cad.rc**

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

////////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

////////////////////////////////////////////////////////////////////////////////
```

```
// Chinese (P.R.C.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_CHS)
#ifdef _WIN32
LANGUAGE LANG_CHINESE, SUBLANG_CHINESE_SIMPLIFIED
#pragma code_page(936)
#endif //_WIN32

#ifdef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "#define _AFX_NO_OLE_RESOURCES\r\n"
    "#define _AFX_NO_TRACKER_RESOURCES\r\n"
    "#define _AFX_NO_PROPERTY_RESOURCES\r\n"
    "\r\n"
    "#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)\r\n"
    "#ifdef _WIN32\r\n"
    "LANGUAGE 9, 1\r\n"
    "#pragma code_page(1252)\r\n"
    "#endif\r\n"
    "#include ""res\\Cad.rc2""  // non-Microsoft Visual C++ edited resources
\r\n"
    "#include ""afxres.rc""          // Standard components\r\n"
    "#include ""afxprint.rc""       // printing/print preview resources\r\n"
    "#endif\0"
END
```

```
#endif    // APSTUDIO_INVOKED


//////////////////////////////////////////////////////////////////////////////
//
// Cursor
//

IDC_CURSOR1             CURSOR  DISCARDABLE     "res\\cursor1.cur"

//////////////////////////////////////////////////////////////////////////////
//
// Bitmap
//

IDR_MODIFY_BAR          BITMAP  DISCARDABLE     "res\\toolbar1.bmp"
IDR_OSNAP_BAR           BITMAP  DISCARDABLE     "res\\toolbar2.bmp"
IDR_ZOOM_BAR            BITMAP  DISCARDABLE     "res\\toolbar3.bmp"
IDR_DRAW_BAR            BITMAP  DISCARDABLE     "res\\toolbar4.bmp"
IDR_PROPERTY_BAR        BITMAP  DISCARDABLE     "res\\toolbar5.bmp"
IDB_SPLASH              BITMAP  DISCARDABLE     "Splsh16.bmp"

//////////////////////////////////////////////////////////////////////////////
//
// Toolbar
//

IDR_DRAW_BAR TOOLBAR DISCARDABLE  16, 15
BEGIN
    BUTTON      ID_DRAW_LINE
    BUTTON      ID_DRAW_CIRCLE
    BUTTON      ID_DRAW_ARC
    BUTTON      ID_DRAW_RECTANGLE
    BUTTON      ID_DRAW_FILLED_CIRCLE
    BUTTON      ID_DRAW_FILLED_RECTANGLE
    BUTTON      ID_DRAW_POINT
    BUTTON      ID_DRAW_TEXT
END

IDR_OSNAP_BAR TOOLBAR DISCARDABLE  16, 15
BEGIN
    BUTTON      ID_OSNAP_ENDPOINT
```

```
    BUTTON      ID_OSNAP_MIDPOINT
    BUTTON      ID_OSNAP_CENTER
    BUTTON      ID_OSNAP_QUARDANT
END

IDR_ZOOM_BAR TOOLBAR DISCARDABLE  16, 15
BEGIN
    BUTTON      ID_ZOOM_WINDOW
    BUTTON      ID_ZOOM_IN
    BUTTON      ID_ZOOM_OUT
    BUTTON      ID_ZOOM_ALL
END

IDR_MODIFY_BAR TOOLBAR DISCARDABLE  16, 15
BEGIN
    BUTTON      ID_MODIFY_ERASE
    BUTTON      ID_EDIT_COPY
    BUTTON      ID_EDIT_CUT
    BUTTON      ID_EDIT_PASTE
    SEPARATOR
    BUTTON      ID_MODIFY_MIRROR
    BUTTON      ID_MODIFY_MOVE
    BUTTON      ID_MODIFY_ROTATE
    BUTTON      ID_MODIFY_SCALE
END

IDR_PROPERTY_BAR TOOLBAR DISCARDABLE  16, 15
BEGIN
    BUTTON      ID_PROPERTY
    BUTTON      ID_COLOR
END


/////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_DIALOG_FILL DIALOG DISCARDABLE  0, 0, 250, 185
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Fill a filled graph"
FONT 10, "System"
BEGIN
```

```
    DEFPUSHBUTTON   "&Fill",IDOK,195,25,40,13
    PUSHBUTTON      "&Cancel",IDCANCEL,195,44,40,13
    LTEXT           "Fill Pattern:",IDC_STATIC,10,5,40,10
    EDITTEXT        IDC_EDIT_PATTERN,50,5,60,10,ES_AUTOHSCROLL
    LISTBOX         IDC_LIST_PATTERN,10,20,100,55,LBS_SORT |
                    LBS_NOINTEGRALHEIGHT | WS_VSCROLL | WS_TABSTOP
    PUSHBUTTON      "&Pick< <",IDC_BUTTON_PICK,195,7,40,13
    CONTROL         "Slider1",IDC_SLIDER,"msctls_trackbar32",TBS_BOTH |
                    TBS_NOTICKS | NOT WS_VISIBLE,0,81,250,6
    PUSHBUTTON      "&More",IDC_BUTTON_MORE,195,62,40,13
    CONTROL         "",IDC_SELECTION,"Static",SS_BLACKFRAME,115,10,70,65
    CONTROL         " ",IDC_PATTERN3,"Button",BS_OWNERDRAW | WS_TABSTOP,165,
                    90,65,40
    CONTROL         " ",IDC_PATTERN2,"Button",BS_OWNERDRAW | WS_TABSTOP,90,
                    90,65,40
    CONTROL         "",IDC_PATTERN1,"Button",BS_OWNERDRAW | WS_TABSTOP,13,90,
                    65,40
    CONTROL         " ",IDC_PATTERN6,"Button",BS_OWNERDRAW | WS_TABSTOP,165,
                    135,65,40
    CONTROL         " ",IDC_PATTERN5,"Button",BS_OWNERDRAW | WS_TABSTOP,90,
                    135,65,40
    CONTROL         " ",IDC_PATTERN4,"Button",BS_OWNERDRAW | WS_TABSTOP,13,
                    135,65,40
END

IDD_ORDINATEPAGE DIALOG DISCARDABLE  0, 0, 212, 95
STYLE WS_CHILD | WS_DISABLED | WS_CAPTION
CAPTION "Ordinate"
FONT 8, "MS Sans Serif"
BEGIN
    GROUPBOX        "Origin",IDC_STATIC,15,10,85,70
    LTEXT           "X:",IDC_STATIC,21,27,10,10
    EDITTEXT        IDC_EDIT_X,31,25,50,12,ES_AUTOHSCROLL
    CONTROL         "Spin1",IDC_SPIN_X,"msctls_updown32",UDS_ARROWKEYS,81,25,
                    10,10
    LTEXT           "Y:",IDC_STATIC,21,45,10,10
    EDITTEXT        IDC_EDIT_Y,31,43,50,12,ES_AUTOHSCROLL
    CONTROL         "Spin1",IDC_SPIN_Y,"msctls_updown32",UDS_ARROWKEYS,81,43,
                    10,10
    LTEXT           "Z:",IDC_STATIC,21,63,10,10
    EDITTEXT        IDC_EDIT_Z,31,61,50,12,ES_AUTOHSCROLL | WS_DISABLED
    CONTROL         "Spin1",IDC_SPIN_Z,"msctls_updown32",UDS_ARROWKEYS |
                    WS_DISABLED,81,61,10,10
```

```
    GROUPBOX        "Scale",IDC_STATIC,120,10,75,70
    LTEXT           "World To Screen:",IDC_STATIC,125,23,65,10
    EDITTEXT        IDC_EDIT_WTOS,125,35,65,10,ES_AUTOHSCROLL
    LTEXT           "Screen To World:",IDC_STATIC,125,50,65,10
    EDITTEXT        IDC_EDIT_STOW,125,62,65,10,ES_AUTOHSCROLL
END

IDD_SETPAGE DIALOG DISCARDABLE  0, 0, 197, 93
STYLE WS_CHILD | WS_DISABLED | WS_CAPTION
CAPTION "Set"
FONT 8, "MS Sans Serif"
BEGIN
    GROUPBOX        "Grid",IDC_STATIC,8,10,50,75
    CONTROL         "On",IDC_GRID_ON,"Button",BS_AUTORADIOBUTTON | WS_GROUP,
                    13,22,25,10
    CONTROL         "Off",IDC_GRID_OFF,"Button",BS_AUTORADIOBUTTON,13,38,25,
                    10
    GROUPBOX        "Snap",IDC_STATIC,73,10,50,75
    CONTROL         "On",IDC_SNAP_ON,"Button",BS_AUTORADIOBUTTON | WS_GROUP,
                    78,22,25,10
    CONTROL         "Off",IDC_SNAP_OFF,"Button",BS_AUTORADIOBUTTON,78,38,25,
                    10
    GROUPBOX        "Ortho",IDC_STATIC,138,10,50,45
    CONTROL         "On",IDC_ORTHO_ON,"Button",BS_AUTORADIOBUTTON | WS_GROUP,
                    143,22,25,10
    CONTROL         "Off",IDC_ORTHO_OFF,"Button",BS_AUTORADIOBUTTON,143,38,
                    25,10
    LTEXT           "Size:",IDC_STATIC,13,51,25,10
    EDITTEXT        IDC_EDIT_GRID,13,65,40,12,ES_AUTOHSCROLL
    LTEXT           "Size:",IDC_STATIC,78,50,25,10
    EDITTEXT        IDC_EDIT_SNAP,78,65,40,12,ES_AUTOHSCROLL
END

IDD_OSNAPPAGE DIALOG DISCARDABLE  0, 0, 183, 94
STYLE WS_CHILD | WS_DISABLED | WS_CAPTION
CAPTION "OSnap"
FONT 8, "MS Sans Serif"
BEGIN
    GROUPBOX        "Type",IDC_STATIC,10,10,75,70
    CONTROL         "Endpoint",IDC_ENDPOINT,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,18,20,55,10
    CONTROL         "Middpoint",IDC_MIDPOINT,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,18,34,55,10
```

```
    CONTROL         "Center",IDC_CENTER,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,18,48,55,10
    CONTROL         "Quardant",IDC_QUARDANT,"Button",BS_AUTOCHECKBOX |
                    WS_TABSTOP,18,62,55,10
    CONTROL         "Slider1",IDC_OSNAP_SIZE,"msctls_trackbar32",TBS_BOTH |
                    TBS_NOTICKS | WS_TABSTOP,105,65,70,10
    GROUPBOX        "Size",IDC_STATIC,100,10,80,70
    CONTROL         "",IDC_PICTURE_OSNAP,"Static",SS_BLACKFRAME,110,20,60,40
END

IDD_ENTER_TEXT DIALOG DISCARDABLE  0, 0, 159, 80
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Enter text"
FONT 10, "System"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,19,54,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,89,54,50,14
    LTEXT           "Enter text:",IDC_STATIC,10,10,35,10
    EDITTEXT        IDC_EDIT_TEXT,10,26,140,15,ES_AUTOHSCROLL
END


/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_ENTER_TEXT, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 152
        TOPMARGIN, 7
        BOTTOMMARGIN, 73
    END
END
#endif    // APSTUDIO_INVOKED

#endif    // Chinese (P.R.C.) resources
/////////////////////////////////////////////////////////////////////////////
```

```
//////////////////////////////////////////////////////////////////////////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

//////////////////////////////////////////////////////////////////////////////
//
// Bitmap
//

IDR_MAINFRAME           BITMAP  MOVEABLE PURE   "res\\Toolbar.bmp"

//////////////////////////////////////////////////////////////////////////////
//
// Toolbar
//

IDR_MAINFRAME TOOLBAR DISCARDABLE  16, 15
BEGIN
    BUTTON      ID_FILE_NEW
    BUTTON      ID_FILE_OPEN
    BUTTON      ID_FILE_SAVE
    SEPARATOR
    BUTTON      ID_EDIT_CUT
    BUTTON      ID_EDIT_COPY
    BUTTON      ID_EDIT_PASTE
    SEPARATOR
    BUTTON      ID_FILE_PRINT
    BUTTON      ID_APP_ABOUT
END


//////////////////////////////////////////////////////////////////////////////
//
// Dialog
//

IDD_ABOUTBOX DIALOG DISCARDABLE  0, 0, 217, 55
```

```
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About Cad"
FONT 8, "MS Sans Serif"
BEGIN
    ICON            IDR_MAINFRAME,IDC_STATIC,11,17,20,20
    LTEXT           "Cad Version 1.0",IDC_STATIC,40,10,119,8,SS_NOPREFIX
    LTEXT           "Copyright (C) 1996",IDC_STATIC,40,25,119,8
    DEFPUSHBUTTON   "OK",IDOK,178,7,32,14,WS_GROUP
END


/////////////////////////////////////////////////////////////////////////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_ABOUTBOX, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 210
        TOPMARGIN, 7
        BOTTOMMARGIN, 48
    END
END
#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDR_MAINFRAME           ICON    DISCARDABLE     "res\\Cad.ico"
IDR_CADTYPE             ICON    DISCARDABLE     "res\\CadDoc.ico"

/////////////////////////////////////////////////////////////////////////////
//
// Menu
```

```
//

IDR_MAINFRAME MENU PRELOAD DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&New \ tCtrl + N",          ID_FILE_NEW
        MENUITEM "&Open... \ tCtrl + O",      ID_FILE_OPEN
        MENUITEM "&Save \ tCtrl + S",         ID_FILE_SAVE
        MENUITEM "Save &As...",               ID_FILE_SAVE_AS
        MENUITEM SEPARATOR
        MENUITEM "&Print... \ tCtrl + P",     ID_FILE_PRINT
        MENUITEM "Print Pre&view",            ID_FILE_PRINT_PREVIEW
        MENUITEM "P&rint Setup...",           ID_FILE_PRINT_SETUP
        MENUITEM SEPARATOR
        MENUITEM "Recent File",               ID_FILE_MRU_FILE1, GRAYED
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                     ID_APP_EXIT
    END
    POPUP "&Edit"
    BEGIN
        MENUITEM "&Undo \ tCtrl + Z",         ID_EDIT_UNDO
        MENUITEM SEPARATOR
        MENUITEM "Cu&t \ tCtrl + X",          ID_EDIT_CUT
        MENUITEM "&Copy \ tCtrl + C",         ID_EDIT_COPY
        MENUITEM "&Paste \ tCtrl + V",        ID_EDIT_PASTE
    END
    POPUP "&View"
    BEGIN
        MENUITEM "&Toolbar",                  ID_VIEW_TOOLBAR
        MENUITEM "&Status Bar",               ID_VIEW_STATUS_BAR
        MENUITEM "S&plit",                    ID_WINDOW_SPLIT
    END
    POPUP "&Draw"
    BEGIN
        MENUITEM "&Line",                     ID_DRAW_LINE
        MENUITEM "&Circle",                   ID_DRAW_CIRCLE
        MENUITEM "&Arc",                      ID_DRAW_ARC
        MENUITEM "&Rectangle",                ID_DRAW_RECTANGLE
        MENUITEM SEPARATOR
        MENUITEM "Filled C&ircle",            ID_DRAW_FILLED_CIRCLE
        MENUITEM "Filled R&ectangle",         ID_DRAW_FILLED_RECTANGLE
        MENUITEM SEPARATOR
```

```
        MENUITEM "&Point",                      ID_DRAW_POINT
        MENUITEM "&Text",                       ID_DRAW_TEXT
    END
    POPUP "&Modify"
    BEGIN
        MENUITEM "&Erase",                      ID_MODIFY_ERASE
        MENUITEM "Mi&rror",                     ID_MODIFY_MIRROR
        MENUITEM "Mo&ve",                       ID_MODIFY_MOVE
        MENUITEM "&Rotate",                     ID_MODIFY_ROTATE
        MENUITEM "&Scale",                      ID_MODIFY_SCALE
    END
    POPUP "&OSnap"
    BEGIN
        MENUITEM "&Endpoint",                   ID_OSNAP_ENDPOINT
        MENUITEM "&Midpoint",                   ID_OSNAP_MIDPOINT
        MENUITEM "&Center",                     ID_OSNAP_CENTER
        MENUITEM "&Quardant",                   ID_OSNAP_QUARDANT
    END
    POPUP "&Zoom"
    BEGIN
        MENUITEM "&Window",                     ID_ZOOM_WINDOW
        MENUITEM "Zoom &In",                    ID_ZOOM_IN
        MENUITEM "Zoom &Out",                   ID_ZOOM_OUT
        MENUITEM "Zoom &All",                   ID_ZOOM_ALL
    END
    POPUP "&Help"
    BEGIN
        MENUITEM "&About Cad...",               ID_APP_ABOUT
    END
END


////////////////////////////////////////////////////////////////////////////
//
// Accelerator
//

IDR_MAINFRAME ACCELERATORS PRELOAD MOVEABLE PURE
BEGIN
    "N",            ID_FILE_NEW,            VIRTKEY, CONTROL
    "O",            ID_FILE_OPEN,           VIRTKEY, CONTROL
    "S",            ID_FILE_SAVE,           VIRTKEY, CONTROL
    "P",            ID_FILE_PRINT,          VIRTKEY, CONTROL
```

```
    "Z",            ID_EDIT_UNDO,          VIRTKEY, CONTROL
    "X",            ID_EDIT_CUT,           VIRTKEY, CONTROL
    "C",            ID_EDIT_COPY,          VIRTKEY, CONTROL
    "V",            ID_EDIT_PASTE,         VIRTKEY, CONTROL
    VK_BACK,        ID_EDIT_UNDO,          VIRTKEY, ALT
    VK_DELETE,      ID_EDIT_CUT,           VIRTKEY, SHIFT
    VK_INSERT,      ID_EDIT_COPY,          VIRTKEY, CONTROL
    VK_INSERT,      ID_EDIT_PASTE,         VIRTKEY, SHIFT
    VK_F6,          ID_NEXT_PANE,          VIRTKEY
    VK_F6,          ID_PREV_PANE,          VIRTKEY, SHIFT
END


#ifndef _MAC
////////////////////////////////////////////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
FILEVERSION 1,0,0,1
PRODUCTVERSION 1,0,0,1
FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
FILEFLAGS 0x1L
#else
FILEFLAGS 0x0L
#endif
FILEOS 0x4L
FILETYPE 0x1L
FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904B0"
        BEGIN
            VALUE "CompanyName", "\0"
            VALUE "FileDescription", "Cad MFC Application\0"
            VALUE "FileVersion", "1, 0, 0, 1\0"
            VALUE "InternalName", "Cad\0"
            VALUE "LegalCopyright", "Copyright (C) 1996\0"
            VALUE "LegalTrademarks", "\0"
            VALUE "OriginalFilename", "Cad.EXE\0"
```

```
            VALUE "ProductName", "Cad Application \0"
            VALUE "ProductVersion", "1, 0, 0, 1\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END

#endif    // ! MAC


/////////////////////////////////////////////////////////////////////////////
//
// String Table
//

STRINGTABLE PRELOAD DISCARDABLE
BEGIN
    IDR_MAINFRAME           "Cad \n \nCad \nCad Files (*.dat) \n.dat \nCad.
Document \nCad Document"
END

STRINGTABLE PRELOAD DISCARDABLE
BEGIN
    IDS_PROPSHT_CAPTION     "Property Sheet"
END

STRINGTABLE PRELOAD DISCARDABLE
BEGIN
    AFX_IDS_APP_TITLE       "Cad"
    AFX_IDS_IDLEMESSAGE     "Ready"
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_INDICATOR_EXT        "EXT"
    ID_INDICATOR_CAPS       "CAP"
    ID_INDICATOR_NUM        "NUM"
    ID_INDICATOR_SCRL       "SCRL"
    ID_INDICATOR_OVR        "OVR"
    ID_INDICATOR_REC        "REC"
```

```
    END

    STRINGTABLE DISCARDABLE
    BEGIN
        ID_FILE_NEW             "Create a new document \ nNew"
        ID_FILE_OPEN            "Open an existing document \ nOpen"
        ID_FILE_CLOSE           "Close the active document \ nClose"
        ID_FILE_SAVE            "Save the active document \ nSave"
        ID_FILE_SAVE_AS         "Save the active document with a new name \ nSave As"
        ID_FILE_PAGE_SETUP      "Change the printing options \ nPage Setup"
        ID_FILE_PRINT_SETUP     "Change the printer and printing options \ nPrint
Setup"
        ID_FILE_PRINT           "Print the active document \ nPrint"
        ID_FILE_PRINT_PREVIEW   "Display full pages \ nPrint Preview"
    END

    STRINGTABLE DISCARDABLE
    BEGIN
        ID_APP_ABOUT            "Display program information, version number and
copyright \ nAbout"
        ID_APP_EXIT             "Quit the application; prompts to save documents \
nExit"
    END

    STRINGTABLE DISCARDABLE
    BEGIN
        ID_FILE_MRU_FILE1       "Open this document"
        ID_FILE_MRU_FILE2       "Open this document"
        ID_FILE_MRU_FILE3       "Open this document"
        ID_FILE_MRU_FILE4       "Open this document"
        ID_FILE_MRU_FILE5       "Open this document"
        ID_FILE_MRU_FILE6       "Open this document"
        ID_FILE_MRU_FILE7       "Open this document"
        ID_FILE_MRU_FILE8       "Open this document"
        ID_FILE_MRU_FILE9       "Open this document"
        ID_FILE_MRU_FILE10      "Open this document"
        ID_FILE_MRU_FILE11      "Open this document"
        ID_FILE_MRU_FILE12      "Open this document"
        ID_FILE_MRU_FILE13      "Open this document"
        ID_FILE_MRU_FILE14      "Open this document"
        ID_FILE_MRU_FILE15      "Open this document"
        ID_FILE_MRU_FILE16      "Open this document"
    END
```

```
    STRINGTABLE DISCARDABLE
    BEGIN
        ID_NEXT_PANE            "Switch to the next window pane \ nNext Pane"
        ID_PREV_PANE            "Switch back to the previous window pane \ nPrevious
Pane"
    END

    STRINGTABLE DISCARDABLE
    BEGIN
        ID_WINDOW_SPLIT         "Split the active window into panes \ nSplit"
    END

    STRINGTABLE DISCARDABLE
    BEGIN
        ID_EDIT_CLEAR           "Erase the selection \ nErase"
        ID_EDIT_CLEAR_ALL       "Erase everything \ nErase All"
        ID_EDIT_COPY            "Copy the selection and put it on the Clipboard \
nCopy"
        ID_EDIT_CUT             "Cut the selection and put it on the Clipboard \
nCut"
        ID_EDIT_FIND            "Find the specified text \ nFind"
        ID_EDIT_PASTE           "Insert Clipboard contents \ nPaste"
        ID_EDIT_REPEAT          "Repeat the last action \ nRepeat"
        ID_EDIT_REPLACE         "Replace specific text with different text \
nReplace"
        ID_EDIT_SELECT_ALL      "Select the entire document \ nSelect All"
        ID_EDIT_UNDO            "Undo the last action \ nUndo"
        ID_EDIT_REDO            "Redo the previously undone action \ nRedo"
    END

    STRINGTABLE DISCARDABLE
    BEGIN
        ID_VIEW_TOOLBAR         "Show or hide the toolbar \ nToggle ToolBar"
        ID_VIEW_STATUS_BAR      "Show or hide the status bar \ nToggle StatusBar"
    END

    STRINGTABLE DISCARDABLE
    BEGIN
        AFX_IDS_SCSIZE          "Change the window size"
        AFX_IDS_SCMOVE          "Change the window position"
        AFX_IDS_SCMINIMIZE      "Reduce the window to an icon"
        AFX_IDS_SCMAXIMIZE      "Enlarge the window to full size"
```

```
    AFX_IDS_SCNEXTWINDOW    "Switch to the next document window"
    AFX_IDS_SCPREVWINDOW    "Switch to the previous document window"
    AFX_IDS_SCCLOSE         "Close the active window and prompts to save the
documents"
END

STRINGTABLE DISCARDABLE
BEGIN
    AFX_IDS_SCRESTORE       "Restore the window to normal size"
    AFX_IDS_SCTASKLIST      "Activate Task List"
END

STRINGTABLE DISCARDABLE
BEGIN
    AFX_IDS_PREVIEW_CLOSE   "Close print preview mode \ nCancel Preview"
    ID_SETDLG               "Set system's status \ nSet System"
    ID_REDRAW               "Redraw the screen \ nRedraw"
    IDW_LINETYPE            "Choose a linetype \ nLinetype"
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_DRAW_LINE            "Draw a line \ nLine"
    ID_DRAW_CIRCLE          "Draw a circle \ nCircle"
    ID_DRAW_ARC             "Draw an arc \ nArc"
    ID_DRAW_RECTANGLE       "Draw a rectangle \ nRectangle"
    ID_DRAW_FILLED_CIRCLE   "Draw a filled circle \ nFilled Circle"
    ID_DRAW_FILLED_RECTANGLE "Draw a filled rectangle \ nFilled Rectangle"
    ID_DRAW_POINT           "Draw a point \ nPoint"
    ID_DRAW_TEXT            "Write text \ nText"
    ID_MODIFY_ERASE         "Erase objects \ nErase"
    ID_MODIFY_MIRROR        "Mirror objects \ nMirror"
    ID_MODIFY_MOVE          "Move objects \ nMove"
    ID_MODIFY_ROTATE        "Rotate objects \ nRotate"
    ID_MODIFY_SCALE         "Scale a object \ nScale"
END

STRINGTABLE DISCARDABLE
BEGIN
    ID_OSNAP_ENDPOINT       "Snap an endpoint \ nEndpoint"
    ID_OSNAP_MIDPOINT       "Snap a midpoint \ nMidpoint"
    ID_OSNAP_CENTER         "Snap a center \ nCenter"
    ID_OSNAP_QUARDANT       "Snap a quardant \ nQuardant"
```

```
    ID_ZOOM_WINDOW          "Zoom by a window \ nWindow"
    ID_ZOOM_IN              "Zoom in \ nZoom In"
    ID_ZOOM_OUT             "Zoom out \ nZoom Out"
    ID_ZOOM_ALL             "Zoom all \ nZoom All"
END

#endif    // English (U.S.) resources
/////////////////////////////////////////////////////////////////////////////



#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
#define _AFX_NO_OLE_RESOURCES
#define _AFX_NO_TRACKER_RESOURCES
#define _AFX_NO_PROPERTY_RESOURCES

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE 9, 1
#pragma code_page(1252)
#endif
#include "res \ Cad.rc2"  // non - Microsoft Visual C++ edited resources
#include "afxres.rc"          // Standard components
#include "afxprint.rc"        // printing/print preview resources
#endif
/////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

2.应用程序的预编译文件 StdAfx.h 和 StdAfx.cpp

**程序清单 12.3　预编译头文件 StdAfx.h**

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#if !defined(AFX_STDAFX_H__12E57A09_ADD2_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
#define AFX_STDAFX_H__12E57A09_ADD2_11CF_BF57_B29AF85F4C3B__INCLUDED_
```

```
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#define VC_EXTRALEAN            // Exclude rarely-used stuff from Windows
headers

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#include <afxdisp.h>        // MFC OLE automation classes
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>                 // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT


//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif // !defined(AFX_STDAFX_H__12E57A09_ADD2_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
```

**程序清单 12.4 预编译实现文件 StdAfx.cpp**

```
// stdafx.cpp : source file that includes just the standard includes
//    Cad.pch will be the pre-compiled header
//    stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

3.应用程序的应用类所在的 Cad.h、Cad.cpp 文件

Cad.h 和 Cad.cpp 文件其中含有与应用程序初始化相关的代码。该应用程序的这两个文件与一般的单文档应用程序的这两个文件极其相似。在这两个文件中,我们几乎未做任何修改工作。

**程序清单 12.5 应用类头文件 Cad.h**

```
//

#if !defined(AFX_CAD_H__12E57A07_ADD2_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
#define AFX_CAD_H__12E57A07_ADD2_11CF_BF57_B29AF85F4C3B__INCLUDED

#if _MSC_VER >= 1000
#pragma once
```

```
#endif // _MSC_VER >= 1000

#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols

//记录主任务和辅助任务
extern UINT     g_nCurTask;
extern UINT     g_nStep;
extern UINT     g_nOpqTask;
extern UINT     g_nOpqStep;

//记录对象捕捉
extern BOOL     g_bOSnapped;
//记录当前颜色和线性
extern UINT     g_nCurColor;
extern UINT     g_nCurLineType;

//记录当前鼠标状态
extern UINT     g_nFlags;
extern double   g_dCurX;
extern double   g_dCurY;
extern double   g_dPreX;
extern double   g_dPreY;
extern double   g_dOrgX;
extern double   g_dOrgY;

//记录 GRID,ORTHO,SNAP 状态
extern BOOL     g_bGrid;
extern BOOL     g_bOrtho;
extern BOOL     g_bSnap;

//记录 Grid 大小和 Snap 大小
extern double   g_dGrid;
extern double   g_dSnap;

//记录对象捕捉类型和屏幕像素点距离
extern int      g_nOSnapType;
extern int      g_nOSnapSize;

//记录当前图形所在区域
```

```
extern double    g_dMaxX;
extern double    g_dMaxY;
extern double    g_dMinX;
extern double    g_dMinY;

//记录当前活动视指针
extern class CCadView *  g_pCurView;
/////////////////////////////////////////////////////////////////////////////
// CCadApp:
// See Cad.cpp for the implementation of this class
//

class CCadApp : public CWinApp
{
public:
     virtual BOOL PreTranslateMessage(MSG * pMsg);
     CCadApp();

// Overrides
     // ClassWizard generated virtual function overrides
     //{{AFX_VIRTUAL(CCadApp)
     public:
     virtual BOOL InitInstance();
     //}}AFX_VIRTUAL

// Implementation

     //{{AFX_MSG(CCadApp)
     afx_msg void OnAppAbout();
          // NOTE - the ClassWizard will add and remove member functions here.
          //    DO NOT EDIT what you see in these blocks of generated code !
     //}}AFX_MSG
     DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif // ! defined(AFX_CAD_H__12E57A07_ADD2_11CF_BF57_B29AF85F4C3B__
```

```
INCLUDED_)
```

**程序清单 12.6　应用类实现文件 Cad.cpp**

```
// Cad.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "Cad.h"

#include "MyStatusBar.h"
#include "PropertyBar.h"
#include "MainFrm.h"
#include "CadDoc.h"
#include "CadView.h"
#include "Splash.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//记录主任务和辅助任务
UINT  g_nCurTask = 0;
UINT  g_nStep = 0;
UINT  g_nOpqTask = 0;
UINT  g_nOpqStep = 0;

//记录对象捕捉
BOOL  g_bOSnapped = FALSE;
//记录当前颜色和线性
UINT  g_nCurColor = RGB(0,0,0);
UINT  g_nCurLineType = PS_SOLID;

//记录当前鼠标状态
UINT  g_nFlags = 0;
double      g_dCurX = 0;
double      g_dCurY = 0;
double      g_dPreX = 0;
double      g_dPreY = 0;
double      g_dOrgX = 0;
double      g_dOrgY = 0;
```

```
//记录 GRID,ORTHO,SNAP 状态
BOOL  g_bGrid = FALSE;
BOOL  g_bOrtho = FALSE;
BOOL  g_bSnap = FALSE;

//记录 Grid 大小和 Snap 大小
double      g_dGrid = 0.5;
double      g_dSnap = 0.5;

//记录对象捕捉类型和屏幕象素点距离
int         g_nOSnapType = 0;
int         g_nOSnapSize = 20;

//记录当前图形所在区域
double      g_dMaxX = 1;
double      g_dMaxY = 1;
double      g_dMinX = -1;
double      g_dMinY = -1;

//记录当前活动视指针
CCadView*    g_pCurView = NULL;
/////////////////////////////////////////////////////////////////////////////
// CCadApp

BEGIN_MESSAGE_MAP(CCadApp, CWinApp)
     //{{AFX_MSG_MAP(CCadApp)
     ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
          // NOTE - the ClassWizard will add and remove mapping macros here.
          //    DO NOT EDIT what you see in these blocks of generated code!
     //}}AFX_MSG_MAP
     // Standard file based document commands
     ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
     ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
     // Standard print setup command
     ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CCadApp construction

CCadApp::CCadApp()
{
     // TODO: add construction code here,
```

```
        // Place all significant initialization in InitInstance
    }

/////////////////////////////////////////////////////////////////////////////
// The one and only CCadApp object.

CCadApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CCadApp initialization

BOOL CCadApp::InitInstance()
{
        // CG: The following block was added by the Splash Screen component. \   {\
            AfxEnableControlContainer();

        // Standard initialization
        // If you are not using these features and wish to reduce the size
        //  of your final executable, you should remove from the following
        //  the specific initialization routines you do not need.

#ifdef _AFXDLL
        Enable3dControls();                 // Call this when using MFC in a shared
DLL
#else
        Enable3dControlsStatic();    // Call this when linking to MFC statically
#endif

        // Change the registry key under which our settings are stored.
        // You should modify this string to be something appropriate
        // such as the name of your company or organization.
        SetRegistryKey(_T("Local AppWizard-Generated Applications"));

        LoadStdProfileSettings();  // Load standard INI file options (including
MRU)

        // Register the application's document templates.  Document templates
        //  serve as the connection between documents, frame windows and views.

        CSingleDocTemplate* pDocTemplate;
        pDocTemplate = new CSingleDocTemplate(
                IDR_MAINFRAME,
                RUNTIME_CLASS(CCadDoc),
```

```
        RUNTIME_CLASS(CMainFrame),       // main SDI frame window
        RUNTIME_CLASS(CCadView));
    AddDocTemplate(pDocTemplate);

    // Enable DDE Execute open
    EnableShellOpen();
    RegisterShellFileTypes(TRUE);

    // Parse command line for standard shell commands, DDE, file open
    CCommandLineInfo cmdInfo;
    ParseCommandLine(cmdInfo);

    // Dispatch commands specified on the command line
    if (! ProcessShellCommand(cmdInfo))
        return FALSE;

    // The one and only window has been initialized, so show and update it.
    m_pMainWnd->ShowWindow(SW_SHOWMAXIMIZED);
    m_pMainWnd->UpdateWindow();

    // Enable drag/drop open
    m_pMainWnd->DragAcceptFiles();

    return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CAbcutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();

// Dialog Data
    //{{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAboutDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
```

```
        //}}AFX_VIRTUAL

    // Implementation
    protected:
        //{{AFX_MSG(CAboutDlg)
            // No message handlers
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
    };

    CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
    {
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
    }

    void CAboutDlg::DoDataExchange(CDataExchange* pDX)
    {
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAboutDlg)
        //}}AFX_DATA_MAP
    }

    BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
            // No message handlers
        //}}AFX_MSG_MAP
    END_MESSAGE_MAP()

    // App command to run the dialog
    void CCadApp::OnAppAbout()
    {
        CAboutDlg aboutDlg;
        aboutDlg.DoModal();
    }

    /////////////////////////////////////////////////////////////////////////////
    // CCadApp commands

    BOOL CCadApp::PreTranslateMessage(MSG* pMsg)
    {
        // CG: The following lines were added by the Splash Screen component.    if
(CSplashWnd::PreTranslateAppMessage(pMsg))            return TRUE;        return
```

```
CWinApp::PreTranslateMessage(pMsg);
    |
```

4.框架定义文件 MainFrm.h 和 MainFrm.cpp

MainFrm.h 和 MainFrm.cpp 文件含有与框架、工具条、状态条相关的代码。我们对 CMainFrm::OnCreate()函数作了较大的改动,在其中添加了生成 Draw、Modify、Property、OSnap 工具条以及创建状态条的代码。

**程序清单 12.7 框架定义头文件 MainFrm.h**

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_MAINFRM_H__12E57A0B_ADD2_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
#define AFX_MAINFRM_H__12E57A0B_ADD2_11CF_BF57_B29AF85F4C3B__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CMainFrame : public CFrameWnd
{
protected: // create from serialization only
      CMainFrame();
      DECLARE_DYNCREATE(CMainFrame)

// Attributes
protected:
      CSplitterWnd m_wndSplitter;
public:

// Operations
public:

// Overrides
      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CMainFrame)
      public:
        virtual BOOL OnCreateClient( LPCREATESTRUCT lpcs, CCreateContext *
pContext);
      virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
      //}}AFX_VIRTUAL
```

```
// Implementation
public:
        virtual ~CMainFrame();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // control bar embedded members
        CMyStatusBar      m_wndStatusBar;
        CToolBar          m_wndToolBar;
        CToolBar          m_wndDrawBar;
        CToolBar          m_wndModifyBar;
        CToolBar          m_wndOSnapBar;
        CToolBar          m_wndZoomBar;
        CPropertyBar      m_wndPropertyBar;

        int         m_cxPaneWidth;
        BOOL  m_bMenuSelected;
        UINT  m_nPaneStyle;
        UINT  m_nPaneID;

        CreatePropertyBar();
// Generated message map functions
protected:
        //{{AFX_MSG(CMainFrame)
        afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
         afx_msg void OnInitMenuPopup(CMenu * pPopupMenu, UINT nIndex, BOOL
bSysMenu);
        afx_msg void OnMenuSelect( UINT nItemID, UINT nFlags, HMENU hSysMenu );
        afx_msg void OnPaint();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif // !defined(AFX_MAINFRM_H__12E57A0B_ADD2_11CF_BF57_B29AF85F4C3B__
```

```
INCLUDED_)
```

**程序清单 12.8 框架定义实现文件 MainFrm.cpp**

```
// MainFrm.cpp : implementation of the CMainFrame class
//

#include "stdafx.h"
#include "Cad.h"

#include "MyStatusBar.h"
#include "PropertyBar.h"
#include "MainFrm.h"
#include "Splash.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
    ON_WM_CREATE()
    ON_WM_INITMENUPOPUP()
    ON_WM_MENUSELECT()
    ON_WM_PAINT()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT DrawButtons[] =
{
    ID_DRAW_LINE,
    ID_DRAW_CIRCLE,
    ID_DRAW_ARC,
    ID_DRAW_RECTANGLE,
        ID_SEPARATOR,

    ID_DRAW_FILLED_CIRCLE,
```

```
    ID_DRAW_FILLED_RECTANGLE,
        ID_SEPARATOR,

    ID_DRAW_POINT,
    ID_DRAW_TEXT,
};

static UINT ModifyButtons[] =
{
    ID_MODIFY_ERASE,
    ID_EDIT_COPY,
    ID_EDIT_CUT,
    ID_EDIT_PASTE,
    ID_MODIFY_MIRROR,
    ID_MODIFY_MOVE,
    ID_MODIFY_ROTATE,
    ID_MODIFY_SCALE,
};

static UINT OSnapButtons[] =
{
    ID_OSNAP_ENDPOINT,
    ID_OSNAP_MIDPOINT,
    ID_OSNAP_CENTER,
    ID_OSNAP_QUARDANT,
};

static UINT ZoomButtons[] =
{
    ID_ZOOM_WINDOW,
    ID_ZOOM_IN,
    ID_ZOOM_OUT,
    ID_ZOOM_ALL,
};

static UINT PropertyButtons[] =
{
    ID_SETDLG,
    ID_REDRAW,
    ID_SEPARATOR,
    ID_SEPARATOR,//颜色组合框
    ID_SEPARATOR,
    ID_SEPARATOR,//线型组合框
```

```
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
    m_cxPaneWidth = -1;
    m_bMenuSelected = FALSE;
}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    if (!m_wndToolBar.Create(this) ||
        !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
    {
        TRACE0("Failed to create toolbar\n");
        return -1;      // fail to create
    }
    m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle() |
        CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC);

    if (!m_wndDrawBar.Create(this, WS_CHILD | WS_VISIBLE |
        CBRS_SIZE_DYNAMIC |     CBRS_TOP | CBRS_TOOLTIPS | CBRS_FLYBY ,
        IDR_DRAW_BAR) ||
        !m_wndDrawBar.LoadBitmap(IDR_DRAW_BAR)||
        !m_wndDrawBar.SetButtons(DrawButtons, sizeof(DrawButtons)/sizeof
(UINT)))
    {
        TRACE0("Failed to create toolbar\n");
        return -1;      // fail to create
    }

    if (!m_wndModifyBar.Create(this, WS_CHILD | WS_VISIBLE |
        CBRS_SIZE_DYNAMIC |     CBRS_TOP | CBRS_TOOLTIPS | CBRS_FLYBY ,
        IDR_MODIFY_BAR) |
```

```
            ! m_wndModifyBar.LoadBitmap(IDR_MODIFY_BAR)||
            ! m_wndModifyBar.SetButtons(ModifyButtons, sizeof(ModifyButtons)/
sizeof(UINT)))
        {
            TRACE0("Failed to create toolbar\n");
            return -1;      // fail to create
        }

        if (! m_wndOSnapBar.Create(this, WS_CHILD | WS_VISIBLE |
            CBRS_SIZE_DYNAMIC |     CBRS_TOP ! CBRS_TOOLTIPS | CBRS_FLYBY ,
            IDR_OSNAP_BAR) ||
            ! m_wndOSnapBar.LoadBitmap(IDR_OSNAP_BAR)||
             ! m_wndOSnapBar.SetButtons(OSnapButtons, sizeof(OSnapButtons)/
sizeof(UINT)))
        {
            TRACE0("Failed to create toolbar\n");
            return -1;      // fail to create
        }

        if (! m_wndZoomBar.Create(this, WS_CHILD | WS_VISIBLE |
            CBRS_SIZE_DYNAMIC |     CBRS_TOP | CBRS_TOOLTIPS | CBRS_FLYBY ,
            IDR_ZOOM_BAR) ||
            ! m_wndZoomBar.LoadBitmap(IDR_ZOOM_BAR)||
            ! m_wndZoomBar.SetButtons(ZoomButtons, sizeof(ZoomButtons)/sizeof
(UINT)))
        {
            TRACE0("Failed to create toolbar\n");
            return -1;      // fail to create
        }

        if (! CreatePropertyBar())
            return -1;

        if(! m_wndStatusBar.Init(this))
            return -1;

        m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
        EnableDocking(CBRS_ALIGN_ANY);
        DockControlBar(&m_wndToolBar);
        m_wndToolBar.SetWindowText("Standard");

        m_wndDrawBar.EnableDocking(CBRS_ALIGN_ANY);
        EnableDocking(CBRS_ALIGN_ANY);
```

```
        DockControlBar(&m_wndDrawBar);
        m_wndDrawBar.SetWindowText("Draw");

        m_wndModifyBar.EnableDocking(CBRS_ALIGN_ANY);
        EnableDocking(CBRS_ALIGN_ANY);
        DockControlBar(&m_wndModifyBar);
        m_wndModifyBar.SetWindowText("Modify");

        m_wndOSnapBar.EnableDocking(CBRS_ALIGN_ANY);
        EnableDocking(CBRS_ALIGN_ANY);
        DockControlBar(&m_wndOSnapBar);
        m_wndOSnapBar.SetWindowText("Object Snap");

        m_wndZoomBar.EnableDocking(CBRS_ALIGN_ANY);
        EnableDocking(CBRS_ALIGN_ANY);
        DockControlBar(&m_wndZoomBar);
        m_wndZoomBar.SetWindowText("Zoom");

        m_wndPropertyBar.EnableDocking(CBRS_ALIGN_TOP|CBRS_ALIGN_BOTTOM);
        EnableDocking(CBRS_ALIGN_TOP|CBRS_ALIGN_BOTTOM);
        DockControlBar(&m_wndPropertyBar);
        m_wndPropertyBar.SetWindowText("Property");

        // CG: The following line was added by the Splash Screen component.
CSplashWnd::ShowSplashScreen(this);
        return 0;
    }

    BOOL CMainFrame::OnCreateClient( LPCREATESTRUCT /* lpcs */,
        CCreateContext* pContext)
    {
        return m_wndSplitter.Create( this,
            2, 2,               // TODO: adjust the number of rows, columns
            CSize( 10, 10 ),    // TODO: adjust the minimum pane size
            pContext );
    }

    BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
    {
        // TODO: Modify the Window class or styles here by modifying
        //  the CREATESTRUCT cs

        cs.style = WS_OVERLAPPED | WS_CAPTION | FWS_ADDTOTITLE
```

```
        | WS_THICKFRAME | WS_SYSMENU | WS_MINIMIZEBOX | WS_MAXIMIZEBOX |
WS_MAXIMIZE;

    return CFrameWnd::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers

BOOL CMainFrame::CreatePropertyBar()
{
    const int nDropHeight = 100;

    if (!m_wndPropertyBar.Create(this, WS_CHILD|WS_VISIBLE|CBRS_TOP|
CBRS_SIZE_DYNAMIC|
            CBRS_TOOLTIPS|CBRS_FLYBY, IDR_PROPERTY_BAR)||
        !m_wndPropertyBar.LoadBitmap(IDR_PROPERTY_BAR)||
            !m_wndPropertyBar.SetButtons(PropertyButtons, sizeof
(PropertyButtons)/sizeof(UINT)))
    {
        TRACE0("Failed to create propertybar\n");
        return FALSE;       // fail to create
    }

    //第一个按钮与线型下拉列表框的间隔设置为12个像素宽
    m_wndPropertyBar.SetButtonInfo(2, ID_SEPARATOR, TBBS_SEPARATOR, 12);
    //设置线型下拉列表框的宽度
```

```
        m_wndPropertyBar.SetButtonInfo(3, IDW_LINETYPE, TBBS_SEPARATOR, 200);
        m_wndPropertyBar.SetButtonInfo(4, ID_SEPARATOR, TBBS_SEPARATOR, 12);
        //设置颜色下拉列表框的宽度
        m_wndPropertyBar.SetButtonInfo(5, IDW_COLOR, TBBS_SEPARATOR, 100);

        CRect rect;
        m_wndPropertyBar.GetItemRect(3, &rect);
        rect.bottom = rect.top + nDropHeight;
        //创建线型下拉列表框
        if (! m_wndPropertyBar.m_wndLineTypeCmb.Create(
                    CBS_DROPDOWNLIST|WS_VISIBLE|WS_TABSTOP|
                    WS_VSCROLL|CBS_OWNERDRAWVARIABLE,
                    rect, &m_wndPropertyBar, IDW_LINETYPE))
        {
            TRACE0("Failed to create combo-box \n");
            return FALSE;
        }
        //填充下拉列表项
        m_wndPropertyBar.m_wndLineTypeCmb.AddItem(PS_SOLID,"Solid");
        m_wndPropertyBar.m_wndLineTypeCmb.AddItem(PS_DASH,"Dash");
        m_wndPropertyBar.m_wndLineTypeCmb.AddItem(PS_DOT,"Dot");
        m_wndPropertyBar.m_wndLineTypeCmb.AddItem(PS_DASHDOT,"DashDot");
        m_wndPropertyBar.m_wndLineTypeCmb.AddItem(PS_DASHDOTDOT,"DashDotDot");
        m_wndPropertyBar.m_wndLineTypeCmb.AddItem(PS_NULL,"Null");
        m_wndPropertyBar.m_wndLineTypeCmb.AddItem(PS_INSIDEFRAME,"
InsideFrame");
        //预先设定实线线型
        m_wndPropertyBar.m_wndLineTypeCmb.SetCurSel(0);

        //创建颜色下拉列表框
        m_wndPropertyBar.GetItemRect(5, &rect);
        rect.bottom = rect.top + nDropHeight;
        if (! m_wndPropertyBar.m_wndColorCmb.Create(
                    CBS_DROPDOWNLIST|WS_VISIBLE|WS_TABSTOP|
                    WS_VSCROLL|CBS_OWNERDRAWVARIABLE,
                    rect, &m_wndPropertyBar, IDW_COLOR))
        {
            TRACE0("Failed to create combo-box \n");
            return FALSE;
        }
        //填充列表项
        m_wndPropertyBar.m_wndColorCmb.AddItem(RGB(255,0,0),"Red");
        m_wndPropertyBar.m_wndColorCmb.AddItem(RGB(0,255,0),"Green");
```

```
        m_wndPropertyBar.m_wndColorCmb.AddItem(RGB(0,0,255),"Blue");
        m_wndPropertyBar.m_wndColorCmb.AddItem(RGB(255,255,0),"Yellow");
        m_wndPropertyBar.m_wndColorCmb.AddItem(RGB(0,255,255),"LightBlue");
        m_wndPropertyBar.m_wndColorCmb.AddItem(RGB(255,0,255),"Cyan");
        m_wndPropertyBar.m_wndColorCmb.AddItem(RGB(0,0,0),"Black");
        //预先设定黑色
        m_wndPropertyBar.m_wndColorCmb.SetCurSel(6);

        return TRUE;
    }

    void CMainFrame::OnInitMenuPopup(CMenu * pPopupMenu, UINT nIndex, BOOL
bSysMenu)
    {
        CFrameWnd::OnInitMenuPopup(pPopupMenu, nIndex, bSysMenu);

        if(m_cxPaneWidth==-1&&m_bMenuSelected)
        {
            m_wndStatusBar.GetPaneInfo(0, m_nPaneID, m_nPaneStyle, m_
cxPaneWidth);
            m_wndStatusBar.SetPaneInfo(0, m_nPaneID, SBPS_NOBORDERS|SBPS_
STRETCH,10000);
        }

    }

    void CMainFrame::OnMenuSelect( UINT nItemID, UINT nFlags, HMENU hSysMenu )
    {
        CFrameWnd::OnMenuSelect(nItemID,nFlags,hSysMenu);

        if(nFlags==0xFFFF&&hSysMenu==0&&m_cxPaneWidth!=-1)
        {
            m_bMenuSelected=FALSE;
            m_wndStatusBar.SetPaneInfo(0, m_nPaneID, m_nPaneStyle, m_
cxPaneWidth);
            m_cxPaneWidth=-1;
        }
        else
        {
            m_bMenuSelected=TRUE;
        }
    }
```

```
void CMainFrame::OnPaint()
{
    CPaintDC dc(this); // device context for painting

    // TODO: Add your message handler code here
    //绘制状态条的 SNAP,GRID,OTHRO 窗格
    CMyStatusBar* pStatus = (CMyStatusBar*)
        AfxGetApp()->m_pMainWnd->GetDescendantWindow(ID_VIEW_
STATUS_BAR);
    pStatus->Redraw();

    // Do not call CFrameWnd::OnPaint() for painting messages
}
```

5.属性工具条定义文件 PropertyBar.h 和 PropertyBar.cpp

PropertyBar.h 和 PropertyBar.cpp 文件含有与属性工具条定义相关的代码。在这两个文件中定义了两个自绘制下拉列表框类。这两个类分别管理 Property 工具条中的线型下拉列表框和颜色下拉列表框。此外,为表示这两个下拉列表框中的数据项还分别定义了两个类。这两个类中含有与各自属性相关的变量。例如,颜色下拉列表框中的数据项类中含有存储颜色值的变量。线型下拉列表框中的数据项类中含有存储线型类型值的变量。

**程序清单 12.9 属性工具条定义头文件 PropertyBar.h**

```
#if !defined(AFX_PROPERTYBAR_H__07598702_BD43_11CF_BF56_B29AF85F4C3B__
INCLUDED_)
#define AFX_PROPERTYBAR_H__07598702_BD43_11CF_BF56_B29AF85F4C3B__
INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// PropertyBar.h : header file
//

class CLineTypeData
{
public:
    int         m_nLineType;
    CString m_strName;

public:
    CLineTypeData(int nLineType, char* pLineTypeName)
        :m_nLineType(nLineType),m_strName(pLineTypeName)
```

```
    }};
    void DrawItem(CDC * pDC,const CRect& rc,BOOL bSelected);
};

/////////////////////////////////////////////////////////////////////////////
// CLineTypeCmb window

class CLineTypeCmb : public CComboBox
{
// Construction
public:
    CLineTypeCmb();

// Attributes
public:

// Operations
public:
    int AddItem(int   nLineType, char * pszName);
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CLineTypeCmb)
    public:
    virtual void DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct);
    virtual void MeasureItem(LPMEASUREITEMSTRUCT lpMeasureItemStruct);
    virtual void DeleteItem(LPDELETEITEMSTRUCT lpDeleteItemStruct);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CLineTypeCmb();

    // Generated message map functions
protected:
    //{{AFX_MSG(CLineTypeCmb)
    afx_msg void OnSelchange();
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

```
class CColorData
{
public:
    COLORREF    m_rgbColor;
    CString         m_strName;

public:
    CColorData(COLORREF rgbColor, char * pColorName)
        :m_rgbColor(rgbColor) ,m_strName(pColorName)
    {};
    void DrawItem(CDC * pDC,const CRect& rc,BOOL bSelected);
};

/////////////////////////////////////////////////////////////////////////////
// CColorCmb window

class CColorCmb : public CComboBox
{
// Construction
public:
    CColorCmb();

// Attributes
public:

// Operations
public:
    int AddItem(COLORREF rgbColor, char * pszName);

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CColorCmb)
    public:
    virtual void DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct);
    virtual void MeasureItem(LPMEASUREITEMSTRUCT lpMeasureItemStruct);
    virtual void DeleteItem(LPDELETEITEMSTRUCT lpDeleteItemStruct);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CColorCmb();

    // Generated message map functions
```

```
protected:
	//{{AFX_MSG(CColorCmb)
	afx_msg void OnSelchange();
	//}}AFX_MSG

	DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////////////
// CPropertyBar window

class CPropertyBar : public CToolBar
{
// Construction
public:
	CPropertyBar();

// Attributes
public:
	CLineTypeCmb	m_wndLineTypeCmb;
	CColorCmb	m_wndColorCmb;

// Operations
public:

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CPropertyBar)
	//}}AFX_VIRTUAL

// Implementation
public:
	virtual ~CPropertyBar();

	// Generated message map functions
protected:
	//{{AFX_MSG(CPropertyBar)
		// NOTE - the ClassWizard will add and remove member functions here.
	//}}AFX_MSG

	DECLARE_MESSAGE_MAP()
```

```
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif // !defined(AFX_PROPERTYBAR_H__075987C2_BD43_11CF_3F56_
B29AF85F4C3B__INCLUDED_)
```

**程序清单 12.10 属性工具条定义实现文件 PropertyBar.cpp**

```
// PropertyBar.cpp : implementation file
//

#include "stdafx.h"
#include "Cad.h"
#include "PropertyBar.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

void CLineTypeData::DrawItem(CDC * pDC,const CRect& rc,BOOL bSelected)
{
      CRect rcDraw(rc);

      COLORREF crBackground, crText;
      if(bSelected)
      {
            crBackground = GetSysColor(COLOR_HIGHLIGHT);
            crText = GetSysColor(COLOR_HIGHLIGHTTEXT);
      }
      else
      {
            crBackground = GetSysColor(COLOR_WINDOW);
            crText = GetSysColor(COLOR_WINDOWTEXT);
      }

      pDC->SetBkColor(crBackground);
      ExtTextOut(pDC->GetSafeHdc(),0,0,ETO_OPAQUE,rcDraw,NULL,0,NULL);
```

```
    CPen * pNewPen = new CPen;
    ASSERT( pNewPen - > CreatePen( m_nLineType, 1, RGB(0,0,0) ) );
    CPen * pOldPen = pDC - > SelectObject( pNewPen );
    pDC - > MoveTo(rcDraw.left,(rcDraw.top + rcDraw.bottom)/2);
    pDC - > LineTo(rcDraw.left + 80,(rcDraw.top + rcDraw.bottom)/2);
    pDC - > SelectObject( pOldPen );
    delete pNewPen;

    rcDraw.left + = 80 + 5;

    pDC - > SetBkColor(crBackground);
    pDC - > SetTextColor(crText);
    pDC - > DrawText(m_strName,m_strName.GetLength(),rcDraw,
        DT_SINGLELINE|DT_VCENTER|DT_LEFT);
}

/////////////////////////////////////////////////////////////////////////////
// CLineTypeCmb
CLineTypeCmb::CLineTypeCmb()
{
}

CLineTypeCmb::~CLineTypeCmb()
{
}


BEGIN_MESSAGE_MAP(CLineTypeCmb, CComboBox)
    //{{AFX_MSG_MAP(CLineTypeCmb)
    ON_CONTROL_REFLECT(CBN_SELCHANGE, OnSelchange)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CLineTypeCmb message handlers

int CLineTypeCmb::AddItem(int nLineType , char * pszName)
{
    CLineTypeData * pData = new CLineTypeData(nLineType,pszName);

    int nRet = AddString((LPCSTR)pData);
    if(nRet = = LB_ERR)
        delete pData;
```

```
        return nRet;
    }

    void CLineTypeCmb::DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct)
    {
        CDC * pDC = CDC::FromHandle(lpDrawItemStruct->hDC);
          CLineTypeData *  pData = (CLineTypeData *) (lpDrawItemStruct ->
itemData);
        ASSERT(pData);
        CRect rc(lpDrawItemStruct->rcItem);

        if(lpDrawItemStruct->itemID==LB_ERR)
            return;

        if(lpDrawItemStruct->itemAction&(ODA_DRAWENTIRE|ODA_SELECT))
            pData->DrawItem(pDC, rc, lpDrawItemStruct->itemState&ODS_
SELECTED);
    }

    void CLineTypeCmb::MeasureItem(LPMEASUREITEMSTRUCT lpMeasureItemStruct)
    {
        lpMeasureItemStruct->itemHeight=18;
    }

    void CLineTypeCmb::DeleteItem(LPDELETEITEMSTRUCT lpDeleteItemStruct)
    {
          CLineTypeData *  pData = (CLineTypeData *) (lpDeleteItemStruct ->
itemData);
        ASSERT(pData);
        delete pData;
    }

    void CLineTypeCmb::OnSelchange()
    {
        int nIndex=GetCurSel();
        if(nIndex!=LB_ERR)
        {
            CLineTypeData* pData=(CLineTypeData *) GetItemDataPtr(nIndex);
            if(pData!=(CLineTypeData *)-1)
            {
                g_nCurLineType=pData->m_nLineType;
            }
```

```
    }
}

void CColorData::DrawItem(CDC * pDC,const CRect& rc,BOOL bSelected)
{
    CRect rcDraw(rc);

    COLORREF crBackground, crText;
    if(bSelected)
    {
        crBackground = GetSysColor(COLOR_HIGHLIGHT);
        crText = GetSysColor(COLOR_HIGHLIGHTTEXT);
    }
    else
    {
        crBackground = GetSysColor(COLOR_WINDOW);
        crText = GetSysColor(COLOR_WINDOWTEXT);
    }

    pDC->SetBkColor(crBackground);
    ExtTextOut(pDC->GetSafeHdc(),0,0,ETO_OPAQUE,rcDraw,NULL,0,NULL);

    CBrush*     pNewBrush = new CBrush;
    ASSERT( pNewBrush->CreateSolidBrush( m_rgbColor) ) ;
    CBrush*     pOldBrush = pDC->SelectObject( pNewBrush );
    pDC->Rectangle(rcDraw.left + 2,rcDraw.top + 2,rcDraw.left + 16,rcDraw.
top + 16);
    pDC->SelectObject( pOldBrush );
    delete  pNewBrush;

    rcDraw.left += 16 + 5;

    pDC->SetBkColor(crBackground);
    pDC->SetTextColor(crText);
    pDC->DrawText(m_strName,m_strName.GetLength(),rcDraw,
        DT_SINGLELINE|DT_VCENTER|DT_LEFT);
}

////////////////////////////////////////////////////////////////////////////
// CColorCmb
CColorCmb::CColorCmb()
{
}
```

```
CColorCmb::~CColorCmb()
{
}


BEGIN_MESSAGE_MAP(CColorCmb, CComboBox)
    //{{AFX_MSG_MAP(CColorCmb)
    ON_CONTROL_REFLECT(CBN_SELCHANGE, OnSelchange)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CColorCmb message handlers
int CColorCmb::AddItem(COLORREF rgbColor , char* pszName)
{
    CColorData* pData = new CColorData(rgbColor,pszName);

    int nRet = AddString((LPCSTR)pData);
    if(nRet == LB_ERR)
        delete pData;

    return nRet;
}

void CColorCmb::DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct)
{
    CDC* pDC = CDC::FromHandle(lpDrawItemStruct->hDC);
    CColorData* pData = (CColorData*)(lpDrawItemStruct->itemData);
    ASSERT(pData);
    CRect rc(lpDrawItemStruct->rcItem);

    if(lpDrawItemStruct->itemID == LB_ERR)
        return;

    if(lpDrawItemStruct->itemAction&(ODA_DRAWENTIRE|ODA_SELECT))
        pData->DrawItem(pDC, rc, lpDrawItemStruct->itemState&ODS
SELECTED);
}

void CColorCmb::MeasureItem(LPMEASUREITEMSTRUCT lpMeasureItemStruct)
{
    lpMeasureItemStruct->itemHeight = 18;
```

```
}

void CColorCmb::DeleteItem(LPDELETEITEMSTRUCT lpDeleteItemStruct)
{
    CColorData * pData = (CColorData * )(lpDeleteItemStruct - > itemData);
    ASSERT(pData);
    delete pData;
}

void CColorCmb::OnSelchange()
{
    int nIndex = GetCurSel();
    if(nIndex! = LB_ERR)
    {
        CColorData * pData = (CColorData * ) GetItemDataPtr(nIndex);
        if(pData! = (CColorData * ) - 1)
        {
            g_nCurColor = pData - > m_rgbColor;
        }
    }
}
////////////////////////////////////////////////////////////////////////////
// CPropertyBar

CPropertyBar::CPropertyBar()
{
}

CPropertyBar::~CPropertyBar()
{
}


BEGIN_MESSAGE_MAP(CPropertyBar, CToolBar)
    //{{AFX_MSG_MAP(CPropertyBar)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
// CPropertyBar message handlers
```

6.状态条定义文件 MystatusBar.h 和 MystatusBar.cpp

MyStatusBar.h和MystatusBar.cpp文件中含有与状态条定义相关的代码。对于状态条中OSnap、Grid、Ortho窗格的绘制是由主框架中的CMainFrame::OnPaint()函数调用该类的Redraw()函数实现的。

**程序清单12.11 状态条定义头文件MyStatusBar.h**

```
#if !defined(AFX_MYSTAUTSBAR_H__628D7342_BE1D_11CF_BF56_B29AF85F4C3B__
INCLUDED_)
#define AFX_MYSTAUTSBAR_H__628D7342_BE1D_11CF_BF56_B29AF85F4C3B__
INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// MyStautsBar.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CMyStatusBar window

class CMyStatusBar : public CStatusBar
{
// Construction
public:
    CMyStatusBar();

// Attributes
public:
    CProgressCtrl m_progress;
// Operations
public:
    int        Init(CWnd* pMainWnd);
    void  Redraw();
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMyStatusBar)
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMyStatusBar();

    // Generated message map functions
protected:
```

```
        afx_msg void OnLButtonDblClk(UINT nFlags, CPoint point);
        //{{AFX_MSG(CMyStatusBar)
        afx_msg void OnTimer(UINT nIDEvent);
        afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
        //}}AFX_MSG

        DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif // !defined(AFX_MYSTAUTSBAR_H__628D7342_BE1D_11CF_BF56_
B29AF85F4C3B__INCLUDED_)
```

**程序清单 12.12　状态条定义实现文件 MyStatusBar.cpp**

```
// MyStautsBar.cpp : implementation file
//

#include "stdafx.h"
#include "Cad.h"
#include "MyStatusBar.h"

#include "CadDoc.h"
#include "CadView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMyStatusBar

CMyStatusBar::CMyStatusBar()
{
}

CMyStatusBar::~CMyStatusBar()
```

```
;
|


BEGIN_MESSAGE_MAP(CMyStatusBar, CStatusBar)
    ON_WM_LBUTTONDBLCLK()
    //{{AFX_MSG_MAP(CMyStatusBar)
    ON_WM_TIMER()
    ON_WM_CREATE()
//  ON_WM_PAINT()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMyStatusBar message handlers
#define ID_STATUS 0
#define ID_MOUSE  1
#define ID_SNAP       2
#define ID_GRID       3
#define ID_ORTHO  4

static UINT indicators[] =
{
    ID_SEPARATOR,     //ID_STATUS
    ID_SEPARATOR,     //ID_MOUSE
    ID_SEPARATOR,     //ID_SNAP
    ID_SEPARATOR,     //ID_GRID
    ID_SEPARATOR,     //ID_ORTHO
};              ,

void CMyStatusBar::OnLButtonDblClk(UINT nFlags, CPoint point)
{
    int       nIsRedraw = 0;
    CRect rect;
    COLORREF crBackground,crText;

    GetItemRect(ID_SNAP,rect);
    if(rect.PtInRect(point))
    {
        g_bSnap = ! g_bSnap;
        if(g_bSnap)
            crText = RGB(0,0,0);
        else
```

```
            crText = RGB(128,128,128);
        nIsRedraw = ID_SNAP;
    }

    if(nIsRedraw = = FALSE)
    {
        GetItemRect(ID_GRID,rect);
        if(rect.PtInRect(point))
        {
            g_bGrid = ! g_bGrid;
            if(g_bGrid)
                crText = RGB(0,0,0);
            else
                crText = RGB(128,128,128);
            nIsRedraw = ID_GRID;

            CDC * pDC = g_pCurView - >GetDC();
            g.pCurView - >DrawGrid(pDC);
        }
    }

    if(nIsRedraw = = FALSE)
    {
        GetItemRect(ID_ORTHO,rect);
        if(rect.PtInRect(point))
        {
            g_bOrtho = ! g_bOrtho;
            if(g_bOrtho)
                crText = RGB(0,0,0);
            else
                crText = RGB(128,128,128);
            nIsRedraw = ID_ORTHO;
        }
    }

    //检查是否需要重画
    if(nIsRedraw = = FALSE)    return;

    CDC *  pDC = GetDC();
    //设置文本和背景颜色
    crBackground = pDC - >GetPixel(rect.left + 1,rect.top + 1);
    pDC - >SetBkColor(crBackground);
    pDC - >SetTextColor(crText);
```

```
        rect.left += 3;
        rect.top += 2;
        rect.right -= 3;
        rect.bottom -= 2;
        switch(nIsRedraw)
        {
        case ID_SNAP:
            pDC->ExtTextOut(rect.left+3,rect.top,ETO_OPAQUE,rect,"SNAP",4,
NULL);
            break;
        case ID_GRID:
            pDC->ExtTextOut(rect.left+3,rect.top,ETO_OPAQUE,rect,"GRID",4,
NULL);
            break;
        case ID_ORTHO:
            pDC->ExtTextOut(rect.left+3,rect.top,ETO_OPAQUE,rect,"ORTHO",
5,NULL);
            break;
        default:
            break;
        }
    }

    int CMyStatusBar::Init(CWnd* pMainWnd)
    {
        if (!Create(pMainWnd) ||
            !SetIndicators(indicators,
              sizeof(indicators)/sizeof(UINT)))
        {
            TRACE0("Failed to create status bar\n");
            return -1;      // fail to create
        }
        //设置状态条窗格的宽度
        SetPaneInfo(0,ID_SEPARATOR,ID_STATUS,360);
        SetPaneInfo(1,ID_SEPARATOR,ID_MOUSE,120);
        SetPaneInfo(2,ID_SEPARATOR,ID_SNAP,42);
        SetPaneInfo(3,ID_SEPARATOR,ID_GRID,40);
        SetPaneInfo(4,ID_SEPARATOR,ID_ORTHO,52);

        SetTimer(1,500,NULL);
        return 1;
    }
```

```
void CMyStatusBar::OnTimer(UINT nIDEvent)
{
	static int nCount = 0;
	m_progress.StepIt();

	if( ++nCount = = 10)
	{
		nCount = 0;
		m_progress.SetPos(0);

		KillTimer(1);
		m_progress.MoveWindow(CRect(0,0,0,0));
	}
}

int CMyStatusBar::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CStatusBar::OnCreate(lpCreateStruct) = = -1)
		return -1;

	m_progress.Create(WS_CHILD|WS_VISIBLE,CRect(0,0,0,0),this,100);
	m_progress.SetRange(0,10);
	m_progress.SetPos(0);
	m_progress.SetStep(1);

	return 0;
}


void CMyStatusBar::Redraw()
{
	CPaintDC dc(this); // device context for painting

	// TODO: Add your message handler code here
	CRect rect;
	COLORREF crBackground,crText;

	//设置文本和背景颜色
	crBackground = GetSysColor(COLOR_MENU);
	dc.SetBkColor(crBackground);

	GetItemRect(ID_SNAP,rect);
```

```
        rect.left + = 3;
        rect.top + = 2;
        rect.right - = 3;
        rect.bottom - = 2;
        if(g_bSnap)
              crText = RGB(0,0,0);
        else
              crText = RGB(128,128,128);
        dc.SetTextColor(crText);
        dc.ExtTextOut(rect.left + 3,rect.top,ETO_OPAQUE,rect,"SNAP",4,NULL);

        GetItemRect(ID_GRID,rect);
        rect.left + = 3;
        rect.top + = 2;
        rect.right - = 3;
        rect.bottom - = 2;
        if(g_bGrid)
              crText = RGB(0,0,0);
        else
              crText = RGB(128,128,128);
        dc.SetTextColor(crText);
        dc.ExtTextOut(rect.left + 3,rect.top,ETO_OPAQUE,rect,"GRID",4,NULL);

        GetItemRect(ID_ORTHO,rect);
        rect.left + = 3;
        rect.top + = 2;
        rect.right - = 3;
        rect.bottom - = 2;
        if(g_bOrtho)
              crText = RGB(0,0,0);
        else
              crText = RGB(128,128,128);
        dc.SetTextColor(crText);
        dc.ExtTextOut(rect.left + 3,rect.top,ETO_OPAQUE,rect,"ORTHO",5,NULL);

        // Do not call CFrameWnd::OnPaint() for painting messages
}
```

7.封面定义文件 Splash.h 和 Splash.cpp

Splash.h 和 Splash.cpp 文件中含有与应用程序封面相关的定义。这两个文件是由 Gallery Components 的 Splash Screen 组件生成的。我们只修改了其中的封面显示延时参数。

**程序清单 12.13 封面定义头文件 Splash.h**

```
// CG: This file was added by the Splash Screen component.
```

```
#ifndef _SPLASH_SCRN_
#define _SPLASH_SCRN_

// Splash.h : header file
//

/////////////////////////////////////////////////////////////////////////////
//   Splash Screen class

class CSplashWnd : public CWnd
{
// Construction
protected:
      CSplashWnd();

// Attributes:
public:
      CBitmap m_bitmap;

// Operations
public:
      static void EnableSplashScreen(BOOL bEnable = TRUE);
      static void ShowSplashScreen(CWnd* pParentWnd = NULL);
      static BOOL PreTranslateAppMessage(MSG* pMsg);

// Overrides
      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CSplashWnd)
      //}}AFX_VIRTUAL

// Implementation
public:
      ~CSplashWnd();
      virtual void PostNcDestroy();

protected:
      BOOL Create(CWnd* pParentWnd = NULL);
      void HideSplashScreen();
      static BOOL c_bShowSplashWnd;
      static CSplashWnd* c_pSplashWnd;

// Generated message map functions
```

```
protected:
    //{{AFX_MSG(CSplashWnd)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnPaint();
    afx_msg void OnTimer(UINT nIDEvent);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#endif
```

**程序清单 12.14 封面定义实现文件 Splash.cpp**

```
// CG: This file was added by the Splash Screen component.
// Splash.cpp : implementation file
//

#include "stdafx.h"  // e.g. stdafx.h
#include "resource.h"  // e.g. resource.h

#include "Splash.h"  // e.g. splash.h

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
//   Splash Screen class

BOOL CSplashWnd::c_bShowSplashWnd;
CSplashWnd* CSplashWnd::c_pSplashWnd;
CSplashWnd::CSplashWnd()
{
}

CSplashWnd::~CSplashWnd()
{
    // Clear the static window pointer.
    ASSERT(c_pSplashWnd == this);
    c_pSplashWnd = NULL;
}
```

```
BEGIN_MESSAGE_MAP(CSplashWnd, CWnd)
    //{{AFX_MSG_MAP(CSplashWnd)
    ON_WM_CREATE()
    ON_WM_PAINT()
    ON_WM_TIMER()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

void CSplashWnd::EnableSplashScreen(BOOL bEnable /* = TRUE */)
{
    c_bShowSplashWnd = bEnable;
}

void CSplashWnd::ShowSplashScreen(CWnd* pParentWnd /* = NULL */)
{
    if (! c_bShowSplashWnd || c_pSplashWnd != NULL)
        return;

    // Allocate a new splash screen, and create the window.
    c_pSplashWnd = new CSplashWnd;
    if (! c_pSplashWnd->Create(pParentWnd))
        delete c_pSplashWnd;
    else
        c_pSplashWnd->UpdateWindow();
}

BOOL CSplashWnd::PreTranslateAppMessage(MSG* pMsg)
{
    if (c_pSplashWnd == NULL)
        return FALSE;

    // If we get a keyboard or mouse message, hide the splash screen.
    if (pMsg->message == WM_KEYDOWN ||
        pMsg->message == WM_SYSKEYDOWN ||
        pMsg->message == WM_LBUTTONDOWN ||
        pMsg->message == WM_RBUTTONDOWN ||
        pMsg->message == WM_MBUTTONDOWN ||
        pMsg->message == WM_NCLBUTTONDOWN ||
        pMsg->message == WM_NCRBUTTONDOWN ||
        pMsg->message == WM_NCMBUTTONDOWN)
    {
        c_pSplashWnd->HideSplashScreen();
        return TRUE;        // message handled here
```

```
    }

    return FALSE;     // message not handled
}

BOOL CSplashWnd::Create(CWnd* pParentWnd /* = NULL */)
{
    if (!m_bitmap.LoadBitmap(IDB_SPLASH))
        return FALSE;

    BITMAP bm;
    m_bitmap.GetBitmap(&bm);

    return CreateEx(0,
        AfxRegisterWndClass(0, AfxGetApp()->LoadStandardCursor(IDC_
ARROW)),
        NULL, WS_POPUP | WS_VISIBLE, 0, 0, bm.bmWidth, bm.bmHeight,
pParentWnd->GetSafeHwnd(), NULL);
}

void CSplashWnd::HideSplashScreen()
{
    // Destroy the window, and update the mainframe.
    DestroyWindow();
    AfxGetMainWnd()->UpdateWindow();
}

void CSplashWnd::PostNcDestroy()
{
    // Free the C++ class.
    delete this;
}

int CSplashWnd::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    // Center the window.
    CenterWindow();

    // Set a timer to destroy the splash screen.
    SetTimer(1, 7500, NULL);
```

```
    return 0;
}

void CSplashWnd::OnPaint()
{
    CPaintDC dc(this);

    CDC dcImage;
    if (! dcImage.CreateCompatibleDC(&dc))
        return;

    BITMAP bm;
    m_bitmap.GetBitmap(&bm);

    // Paint the image.
    CBitmap * pOldBitmap = dcImage.SelectObject(&m_bitmap);
    dc.BitBlt(0, 0, bm.bmWidth, bm.bmHeight, &dcImage, 0, 0, SRCCOPY);
    dcImage.SelectObject(pOldBitmap);
}

void CSplashWnd::OnTimer(UINT nIDEvent)
{
    // Destroy the splash screen window.
    HideSplashScreen();
}
```

8.视定义文件 CadView.h 和 CadView.cpp

CadView.h 和 CadView.cpp 文件中含有与视类相关的定义。这两个文件中含有实现该应用程序层次框架结构中第一层的所有功能的函数(参见 12.2 节)。也就是说,用户的任何输入都是由这两个文件中的函数进行接收处理的。我们在这两个文件中做了大量的工作。

**程序清单 12.15 视定义头文件 CadView.h**

```
// CadView.h : interface of the CCadView class
//
/////////////////////////////////////////////////////////////////////////////

#if ! defined(AFX_CADVIEW_H__12E57A0F_ADD2_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
#define AFX_CADVIEW_H__12E57A0F_ADD2_11CF_BF57_B29AF85F4C3B__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
```

```
#endif // _MSC_VER >= 1000

class CCadView : public CView
{
protected: // create from serialization only
    CCadView();
    DECLARE_DYNCREATE(CCadView)

// Attributes
public:
    CCadDoc* GetDocument();

    //记录世界坐标与屏幕坐标的关系
    double      m_dOrgX,m_dOrgY;
    double      m_dWorldToScreen,m_dScreenToWorld;

    //进行世界坐标与屏幕坐标变换
    void WorldToScreen(CPoint& point,double x,double y);
    void ScreenToWorld(double& x,double& y,CPoint& point);

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CCadView)
    public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    protected:
    virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
    virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
    //}}AFX_VIRTUAL

// Implementation
public:
    //绘制网格点
    void DrawGrid(CDC* pDC);
    virtual ~CCadView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
```

```
protected:

// Generated message map functions
protected:
      void ZoomWindow(CDC * pDC,BOOL bSure);
      void DrawSys(CDC * pDC);
      CPoint m_prePoint;
      void DrawCursor(CDC * pDC,CPoint point);
      //{{AFX_MSG(CCadView)
      afx_msg BOOL OnSetCursor(CWnd * pWnd, UINT nHitTest, UINT message);
      afx_msg void OnMouseMove(UINT nFlags, CPoint point);
      afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
      afx_msg void OnRedraw();
      afx_msg void OnSetDlg();
      afx_msg void OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags);
      afx_msg void OnRButtonDown(UINT nFlags, CPoint point);
      //}}AFX_MSG
      afx_msg void OnDrawCommand(DWORD wParam,LPARAM lParam);
      afx_msg void OnModifyCommand(DWORD wParam,LPARAM lParam);
      afx_msg void OnOSnapCommand(DWORD wParam,LPARAM lParam);
      afx_msg void OnZoomCommand(DWORD wParam,LPARAM lParam);
      afx_msg void OnNotFinishCommand(DWORD wParam,LPARAM lParam);
      DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in CadView.cpp
inline CCadDoc * CCadView::GetDocument()
   { return (CCadDoc * )m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif // ! defined(AFX_CADVIEW_H__12E57A0F_ADD2_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
```

**程序清单 12.16　视定义实现文件 CadView.cpp**

```
// CadView.cpp : implementation of the CCadView class
//
```

```
#include "stdafx.h"
#include "Cad.h"

#include "CadDoc.h"
#include "CadView.h"
#include "StatusDlg.h"
#include "MyStatusBar.h"

#include <math.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CCadView

IMPLEMENT_DYNCREATE(CCadView, CView)

BEGIN_MESSAGE_MAP(CCadView, CView)
    //{{AFX_MSG_MAP(CCadView)
    ON_WM_SETCURSOR()
    ON_WM_MOUSEMOVE()
    ON_WM_LBUTTONDOWN()
    ON_COMMAND(ID_REDRAW, OnRedraw)
    ON_COMMAND(ID_SETDLG, OnSetDlg)
    ON_WM_KEYDOWN()
    ON_WM_RBUTTONDOWN()
    //}}AFX_MSG_MAP
    // Standard printing commands
    ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)

    ON_COMMAND_RANGE(ID_DRAW_LINE,ID_DRAW_TEXT,OnDrawCommand)
    ON_COMMAND_RANGE(ID_MODIFY_ERASE,ID_MODIFY_SCALE,OnModifyCommand)
    ON_COMMAND_RANGE(ID_OSNAP_ENDPOINT,ID_OSNAP_QUARDANT,OnOSnapCommand)
    ON_COMMAND_RANGE(ID_ZOOM_WINDOW,ID_ZOOM_ALL,OnZoomCommand)
    ON_COMMAND_RANGE(ID_EDIT_COPY,ID_EDIT_PASTE,OnNotFinishCommand)

END_MESSAGE_MAP()
```

```
////////////////////////////////////////////////////////////////////////////
// CCadView construction/destruction

CCadView::CCadView()
{
    m_prePoint.x = m_prePoint.y = -1;
    //设置世界坐标与屏幕坐标的关系
    m_dOrgX = m_dOrgY = 0.0;
    m_dWorldToScreen = m_dScreenToWorld = 1.0;
}

CCadView::~CCadView()
{
}

////////////////////////////////////////////////////////////////////////////
// CCadView drawing

void CCadView::OnDraw(CDC* pDC)
{
    CCadDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    // TODO: add draw code for native data here
    //获取当前客户区的大小
    pDoc->Draw(pDC);
    DrawSys(pDC);
}

////////////////////////////////////////////////////////////////////////////
// CCadView printing

BOOL CCadView::OnPreparePrinting(CPrintInfo* pInfo)
{
    // default preparation
    return DoPreparePrinting(pInfo);
}

void CCadView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add extra initialization before printing
}
```

```
void CCadView::OnEndPrinting(CDC * /*pDC*/, CPrintInfo * /*pInfo*/)
{
    // TODO: add cleanup after printing
}

/////////////////////////////////////////////////////////////////////////////
// CCadView diagnostics

#ifdef _DEBUG
void CCadView::AssertValid() const
{
    CView::AssertValid();
}

void CCadView::Dump(CDumpContext& dc) const
{
    CView::Dump(dc);
}

CCadDoc * CCadView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CCadDoc)));
    return (CCadDoc *)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CCadView message handlers

BOOL CCadView::OnSetCursor(CWnd * pWnd, UINT nHitTest, UINT message)
{
    SetCursor(AfxGetApp()->LoadCursor(IDC_CURSOR1));
    return TRUE;
}

void CCadView::OnMouseMove(UINT nFlags, CPoint point)
{
    g_nFlags = nFlags;
    ScreenToWorld(g_dCurX,g_dCurY,point);
    g_pCurView = this;
    CCadDoc * pDoc = GetDocument();
    ASSERT_VALID(pDoc);
```

```
        CDC * pDC = GetDC();

        DrawCursor(pDC,point);
        //记录当前点,以作为下次进入 MouseMove 的前一个点
        m_prePoint = point;


         if ((g_ nOpqTask > = ID_ OSNAP_ ENDPOINT&&g_ nOpqTask < = ID_ OSNAP_
QUARDANT)
              ||g_nOSnapType! = 0)
        {
              pDoc - > OSnap(pDC,FALSE);
        }
        if(g_ nOpqTask > = ID_ ZOOM_ WINDOW&&g_ nOpqTask < = ID_ ZOOM_ ALL)
              ZoomWindow(pDC,FALSE);

        if(g_ nCurTask > = ID_ DRAW_ LINE&&g_ nCurTask < = ID_ DRAW_ TEXT)
              pDoc - > Create(pDC,FALSE);

        if((g_ nCurTask > = ID_ EDIT_ COPY&&g_ nCurTask < = ID_ EDIT_ PASTE)
               ||(g_ nCurTask > = ID_ MODIFY_ ERASE&&g_ nCurTask < = ID_ MODIFY_
SCALE))
        {
              pDoc - > ModifyMove(pDC);
        }


        //获得状态条的指针
        CMyStatusBar * pStatus = (CMyStatusBar * )
               AfxGetApp( ) - > m_ pMainWnd - > GetDescendantWindow( ID_ VIEW_
STATUS_ BAR);
        if(pStatus)
        {
              char tbuf[40];
              sprintf(tbuf,"( %8.3f, %8.3f)",g_ dCurX,g_ dCurY);
              //在状态条的第二个窗格中输出当前鼠标的位置
              pStatus - > SetPaneText(1,tbuf);
        }

        g_ dPreX = g_ dCurX;
        g_ dPreY = g_ dCurY;
    }
```

```
void CCadView::OnLButtonDown(UINT nFlags, CPoint point)
{
    g_nFlags = nFlags;
    ScreenToWorld(g_dCurX,g_dCurY,point);
    g_pCurView = this;

    CCadDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    CDC* pDC = GetDC();

     if((g_nOpqTask >= ID_OSNAP_ENDPOINT&&g_nOpqTask <= ID_OSNAP
QUARDANT)
        ||g_nOSnapType! = 0)
    {
        pDoc->OSnap(pDC,TRUE);
    }
    if(g_nOpqTask == ID_ZOOM_WINDOW)
        ZoomWindow(pDC,TRUE);

    if(g_nCurTask >= ID_DRAW_LINE&&g_nCurTask <= ID_DRAW_TEXT)
        pDoc->Create(pDC,TRUE);

    if((g_nCurTask >= ID_EDIT_COPY&&g_nCurTask <= ID_EDIT_PASTE)
        ||(g_nCurTask >= ID_MODIFY_ERASE&&g_nCurTask <= ID_MODIFY_
SCALE))
    {
        pDoc->ModifyLButton(pDC);
    }

    g_dOrgX = g_dCurX;
    g_dOrgY = g_dCurY;
}

void CCadView::OnRButtonDown(UINT nFlags, CPoint point)
{
    CCadDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    CDC* pDC = GetDC();

    if((g_nCurTask >= ID_EDIT_COPY&&g_nCurTask <= ID_EDIT_PASTE)
        ||(g_nCurTask >= ID_MODIFY_ERASE&&g_nCurTask <= ID_MODIFY_
SCALE))
    {
```

```
            pDoc->ModifyRButton(pDC);
        }
}

void CCadView::DrawCursor(CDC* pDC,CPoint point)
{
    //获取当前客户区的大小
    CRect rect;
    GetClientRect(rect);

    //设置R2_NOT的绘图方式
    int nDrawMode = pDC->SetROP2(R2_NOT);
    //覆盖先前的光标线
    pDC->MoveTo(rect.left,m_prePoint.y);
    pDC->LineTo(rect.right,m_prePoint.y);
    pDC->MoveTo(m_prePoint.x,rect.top);
    pDC->LineTo(m_prePoint.x,rect.bottom);

    //绘制新的光标线
    pDC->MoveTo(rect.left,point.y);
    pDC->LineTo(rect.right,point.y);
    pDC->MoveTo(point.x,rect.top);
    pDC->LineTo(point.x,rect.bottom);
    //设置先前的绘图方式
    pDC->SetROP2(nDrawMode);
}

void CCadView::DrawSys(CDC* pDC)
{
    CRect rect;
    GetClientRect(rect);

    //绘制先前的光标线,
    //以避免在应用程序间切换时出现不应有的线
    pDC->MoveTo(rect.left,m_prePoint.y);
    pDC->LineTo(rect.right,m_prePoint.y);
    pDC->MoveTo(m_prePoint.x,rect.top);
    pDC->LineTo(m_prePoint.x,rect.bottom);

    DrawGrid(pDC);
}

void CCadView::OnRedraw()
```

```
{
    //获取当前客户区的大小
    CRect rect;
    GetClientRect(rect);

    CDC * pDC = GetDC();
    pDC -> Rectangle(rect);
    OnDraw(pDC);
}

void CCadView::OnSetDlg()
{
    CStatusDlg  dlg;

    dlg.m_Page1.m_dOriginX = m_dOrgX;
    dlg.m_Page1.m_dOriginY = m_dOrgY;
    dlg.m_Page1.m_dStoW = m_dScreenToWorld;
    dlg.m_Page1.m_dWtoS = m_dWorldToScreen;

    dlg.m_Page2.m_dGrid = g_dGrid;
    dlg.m_Page2.m_dSnap = g_dSnap;
    dlg.m_Page2.m_nGrid = (g_bGrid_ = TRUE ? 0:1);
    dlg.m_Page2.m_nOrtho = (g_bOrtho = = TRUE ? 0:1);
    dlg.m_Page2.m_nSnap = (g_bSnap = = TRUE ? 0:1);

    dlg.m_Page3.m_bEndpoint = (TRUE,g_nOSnapType&1,FALSE);
    dlg.m_Page3.m_bMidpoint = (g_nOSnapType&2,TRUE,FALSE);
    dlg.m_Page3.m_bCenter = (TRUE,g_nOSnapType&4,FALSE);
    dlg.m_Page3.m_bQuardant = (TRUE,g_nOSnapType&8,FALSE);
    dlg.m_Page3.m_nOSnapSize = g_nOSnapSize/2;

    int ret = dlg.DoModal();

    if(ret = = IDOK)
    {
        m_dOrgX = dlg.m_Page1.m_dOriginX;
        m_dOrgY = dlg.m_Page1.m_dOriginY;
        m_dScreenToWorld = dlg.m_Page1.m_dStoW;
        if(fabs(m_dScreenToWorld) < 1e-10)
            m_dScreenToWorld = 1.0;
        m_dWorldToScreen = 1/m_dScreenToWorld;

        g_dGrid = dlg.m_Page2.m_dGrid;
```

```
                g_dSnap = dlg.m_Page2.m_dSnap;
                g_bGrid = (dlg.m_Page2.m_nGrid = = 0 ? TRUE:FALSE);
                g_bOrtho = (dlg.m_Page2.m_nOrtho = = 0 ? TRUE:FALSE);
                g_bSnap = (dlg.m_Page2.m_nSnap = = 0 ? TRUE:FALSE);

                  g_nOSnapType = dlg.m_Page3.m_bEndpoint + 2 * dlg.m_Page3.m_
bMidpoint +
                                 4 * dlg.m_Page3.m_bCenter + 8 * dlg.m_Page3.m_
bQuardant;
                g_nOSnapSize = dlg.m_Page3.m_nOSnapSize * 2;

                //绘制状态条的 SNAP,GRID,OTHRO 窗格
                CMyStatusBar * pStatus = (CMyStatusBar * )
                      AfxGetApp() - > m_pMainWnd - > GetDescendantWindow(ID_VIEW_
STATUS_BAR);
                pStatus - > Redraw();

                //绘制网格
                CDC * pDC = GetDC();
                DrawGrid(pDC);
          }
    }

    void CCadView::OnDrawCommand(DWORD wParam,LPARAM lParam)
    {
          g_nCurTask = wParam;
          g_nStep = 0;

          char * prompt[] = {"Enter the first point",
                             "Enter the center",
                             "Enter the center",
                             "Enter the first point",
                             "Enter the center",
                             "Enter the first point",
                             "Enter the point",
                             "Enter the insert point",
                             };


          CStatusBar * pStatus = (CStatusBar * )
                AfxGetApp() - > m_pMainWnd - > GetDescendantWindow(ID_VIEW_
STATUS_BAR);
          ASSERT(pStatus);
```

```
        pStatus->SetPaneText(0,prompt[wParam-ID_DRAW_LINE]);

    }

    void CCadView::OnModifyCommand(DWORD wParam,LPARAM lParam)
    {
        g_nCurTask=wParam;
        g_nStep=0;
        CCadDoc* pDoc=GetDocument();
        ASSERT_VALID(pDoc);
        pDoc->m_nEntitySelCnt=0;
        char* prompt[]={"Pick entities to erase",
                        "Pick entities to mirror",
                        "Pick entities to move",
                        "Pick entities to rotate",
                        "Pick entities to scale",
                        };

        CStatusBar* pStatus=(CStatusBar*)
            AfxGetApp()->m_pMainWnd->GetDescendantWindow(ID_VIEW_
STATUS_BAR);
        ASSERT(pStatus);

        pStatus->SetPaneText(0,prompt[wParam-ID_MODIFY_ERASE]);
    }

    void CCadView::OnOSnapCommand(DWORD wParam,LPARAM lParam)
    {
        g_nOpqTask=wParam;
        g_bOSnapped=FALSE;
        g_nOpqStep=0;

        char* prompt[]={"Snap end point",
                        "Snap middle point",
                        "Snap center",
                        "Snap quardant point",
                        };

        CStatusBar* pStatus=(CStatusBar*)
            AfxGetApp()->m_pMainWnd->GetDescendantWindow(ID_VIEW
```

```
STATUS_BAR);
        ASSERT(pStatus);

        pStatus->SetPaneText(0,prompt[wParam-ID_OSNAP_ENDPOINT]);
    }

    void CCadView::OnZoomCommand(DWORD wParam,LPARAM lParam)
    {
        g_nOpqStep=0;

        if(wParam==ID_ZOOM_WINDOW)
        {
            g_nOpqTask=wParam;
            g_nCurTask=0;

            CStatusBar* pStatus=(CStatusBar*)
                AfxGetApp()->m_pMainWnd->GetDescendantWindow(ID_VIEW_
STATUS_BAR);
            ASSERT(pStatus);

            pStatus->SetPaneText(0,"Enter the first conner point");
        }
        else if(wParam==ID_ZOOM_IN)
        {
            SendMessage(WM_KEYDOWN,VK_NEXT);
            g_nOpqTask=0;
        }
        else if(wParam==ID_ZOOM_OUT)
        {
            SendMessage(WM_KEYDOWN,VK_PRIOR);
            g_nOpqTask=0;
        }
        else
        {
            m_dOrgX=(g_dMaxX+g_dMinX)/2.0;
            m_dOrgY=(g_dMaxY+g_dMinY)/2.0;
            //获取当前客户区的大小
            CRect rect;
            GetClientRect(rect);
            double dStoW_X=fabs((g_dMaxX-g_dMinX)/(rect.right-rect.left));
            double dStoW_Y=fabs((g_dMaxY-g_dMinY)/(rect.bottom-rect.top));
            if(dStoW_X>dStoW_Y)
                m_dScreenToWorld=dStoW_X;
```

```
            else
                m_dScreenToWorld = dStoW_Y;
            if(fabs(m_dScreenToWorld) < 1e-10)
                m_dScreenToWorld = 1.0;
            m_dWorldToScreen = 1/m_dScreenToWorld;

            CDC * pDC = GetDC();
            pDC -> Rectangle(rect);
            OnDraw(pDC);
        }
    }

    void CCadView::OnNotFinishCommand(DWORD wParam,LPARAM lParam)
    {
        AfxMessageBox("Sorry! This function is not finished!");
    }

    void CCadView::ScreenToWorld(double& x, double& y, CPoint& point)
    {
        //获取当前客户区的大小
        CRect rect;
        GetClientRect(rect);
        int         nSOrgX = (rect.left + rect.right)/2;
        int         nSOrgY = (rect.top + rect.bottom)/2;

        x = (point.x - nSOrgX) * m_dScreenToWorld + m_dOrgX;
        y = (nSOrgY - point.y) * m_dScreenToWorld + m_dOrgY;

        if(g_bSnap == TRUE)
        {
            if(g_dSnap < 1e-10) g_dSnap = 0.5;
            long n;
            n = x/g_dSnap;
            x = n * g_dSnap;
            n = y/g_dSnap;
            y = n * g_dSnap;
            WorldToScreen(point,x,y);
        }
        if(g_bOrtho == TRUE&&g_nStep! = 0)
        {
            if(fabs(g_dCurX - g_dOrgX) > fabs(g_dCurY - g_dOrgY))
                y = g_dOrgY;
            else
```

```
                x = g_dOrgX;
        }
}

void CCadView::WorldToScreen(CPoint& point, double x, double y)
{
    //获取当前客户区的大小
    CRect rect;
    GetClientRect(rect);
    int         nSOrgX = (rect.left + rect.right)/2;
    int         nSOrgY = (rect.top + rect.bottom)/2;

    point.x = (int) (x - m_dOrgX) * m_dWorldToScreen + nSOrgX;
    point.y = (int) nSOrgY - (y - m_dOrgY) * m_dWorldToScreen;
}

void CCadView::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    //获取当前客户区的大小
    CRect rect;
    GetClientRect(rect);
    //获得偏移变量
    double      delta_x = (rect.right - rect.left) * m_dScreenToWorld/6;
    double      delta_y = (rect.bottom - rect.top) * m_dScreenToWorld/6;

    switch(nChar)
    {
    case VK_LEFT://图形向左移
        m_dOrgX += delta_x;
        break;
    case VK_RIGHT://图形向右移
        m_dOrgX -= delta_x;
        break;
    case VK_UP://图形向上移
        m_dOrgY -= delta_y;
        break;
    case VK_DOWN://图形向下移
        m_dOrgY += delta_y;
        break;
    case VK_NEXT://图形放大
        m_dWorldToScreen *= 0.618;
        if(m_dWorldToScreen < 1e-10) m_dWorldToScreen = 1.0;
        m_dScreenToWorld = 1.0/m_dWorldToScreen;
```

```
            break;
        case VK_PRIOR://图形缩小
            m_dScreenToWorld *= 0.618;
            if(m_dScreenToWorld < 1e-10) m_dScreenToWorld = 1.0;
            m_dWorldToScreen = 1.0/m_dScreenToWorld;
            break;
        default:
            break;
        }

        CDC * pDC = GetDC();
        pDC->Rectangle(rect);
        OrDraw(pDC);
    }

    void CCadView::ZoomWindow(CDC * pDC, BOOL bSure)
    {
        if(bSure == TRUE)
        {
            CStatusBar * pStatus = (CStatusBar *)
                AfxGetApp()->m_pMainWnd->GetDescendantWindow(ID_VIEW
STATUS_BAR);
            ASSERT(pStatus);

            switch(g_nOpqStep)
            {
            case 0:
                pStatus->SetPaneText(0,"Enter the other corner");
                g_nOpqStep++;
                break;
            case 1:
                {
                m_dOrgX = (g_dOrgX + g_dCurX)/2.0;
                m_dOrgY = (g_dOrgY + g_dCurY)/2.0;
                //获取当前客户区的大小
                CRect rect;
                GetClientRect(rect);

                double dStoW_X = fabs((g_dCurX - g_dOrgX)/(rect.right - rect.
left));
                double dStoW_Y = fabs((g_dCurY - g_dOrgY)/(rect.bottom - rect.
top));
                if(dStoW_X > dStoW_Y)
```

```
                m_dScreenToWorld = dStoW_X;
            else
                m_dScreenToWorld = dStoW_Y;
            if(fabs(m_dScreenToWorld) < 1e-10)
                m_dScreenToWorld = 1.0;
            m_dWorldToScreen = 1/m_dScreenToWorld;

            CDC * pDC = GetDC();
            pDC->Rectangle(rect);
            OnDraw(pDC);

            g_nOpqStep = 0;
            g_nOpqTask = 0;
            break;
            }
        default:
            break;
        }
    }
    else
    {
        if(g_nOpqStep! = 1) return ;
        //设置 R2_NOT 的绘图方式
        int nDrawMode = pDC->SetROP2(R2_NOT);

        CPoint      orgPoint,curPoint,prePoint;
        g_pCurView->WorldToScreen(orgPoint,g_dOrgX,g_dOrgY);
        g_pCurView->WorldToScreen(curPoint,g_dCurX,g_dCurY);
        g_pCurView->WorldToScreen(prePoint,g_dPreX,g_dPreY);
        //覆盖先前的线
        pDC->MoveTo(orgPoint);
        pDC->LineTo(orgPoint.x,prePoint.y);
        pDC->LineTo(prePoint);
        pDC->LineTo(prePoint.x,orgPoint.y);
        pDC->LineTo(orgPoint);
        //绘制新的线
        pDC->MoveTo(orgPoint);
        pDC->LineTo(orgPoint.x,curPoint.y);
        pDC->LineTo(curPoint);
        pDC->LineTo(curPoint.x,orgPoint.y);
        pDC->LineTo(orgPoint);
        //设置先前的绘图方式
        pDC->SetROP2(nDrawMode);
```

```
    }
    return ;
}

void CCadView::DrawGrid(CDC * pDC)
{
    //获取当前客户区的大小
    CRect rect;
    GetClientRect(rect);

    CPoint      orgPoint;
    WorldToScreen(orgPoint,0,0);
    int         nDistance = g_dGrid * m_dWorldToScreen;

    if(nDistance < 5) return;

    int n,x,y;
    n = (rect.left - orgPoint.x)/nDistance;
    x = orgPoint.x + n * nDistance;
    n = (rect.top - orgPoint.y)/nDistance;
    y = orgPoint.y + n * nDistance;

    UINT  color;
    if(g_bGrid = = TRJE) color - RGB(0,0,0);
    else  color = RGB(255,255,255);

    for(int j = y;j < rect.bottom;j = j + nDistance)
    {
        for(int i = x;i < rect.right;i = i + nDistance)
        {
            pDC - > SetPixel(i,j,color);
        }
    }
}
```

9. 文档定义文件 CadDoc.h 和 CadDoc.cpp

CadDoc.h 和 CadDoc.cpp 文件中含有与文档类相关的定义。这两个文件中含有实现该应用程序层次框架结构中第二层的所有功能的函数(参见 12.2 节)。换句话说,该系统中的所有数据的管理、组织皆是由这两个文件中的函数完成的。我们在这两个文件中定义了大量的用于数据链表管理的变量。这两个文件是 CadView.h、CadView.cpp 文件与另外六个定义实体对象类的头文件和实现文件进行通信的桥梁。

**程序清单 12.17 文档定义头文件 CadDoc.h**

```
// CadDcc.h : interface of the CCadDoc class
```

```
//
////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_CADDOC_H__12E57A0D_ADD2_11CF_BF57_B29AF85F4C3B__INCLUDED_)
#define AFX_CADDOC_H__12E57A0D_ADD2_11CF_BF57_B29AF85F4C3B__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#include "Entity.h"
#include "Line.h"
#include "Circle.h"
#include "Rectangle.h"
#include "Point.h"
#include "Text.h"

class CCadDoc : public CDocument
{
protected: // create from serialization only
    CCadDoc();
    DECLARE_DYNCREATE(CCadDoc)

// Attributes
public:
    double      m_dChgA[3][3];          //变换矩阵
    CEntity*    m_pEntityHead;          //实体链表头指针
    CEntity*    m_pEntityTail;          //实体链表尾指针
    CEntity*    m_pCurEntity;           //实体链表中的当前实体指针
    CEntity*    m_pEntitySelected[10];  //实体选择集指针
    int         m_nEntitySelCnt;        //实体选择集中的实体数目

    //预定义的几个实体
    CLine       m_Line;
    CCircle     m_Circle;
    CRectangle  m_Rectangle;
    CPnt        m_Pnt;
    CText       m_Text;
// Operations
public:

// Overrides
```

```
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CCadDoc)
        public:
        virtual BOOL OnNewDocument();
        virtual void Serialize(CArchive& ar);
        //}}AFX_VIRTUAL

    // Implementation
    public:
        //以下几个函数用于实体平移、旋转、缩放等坐标变换
        void InitChgA();
        void ChangePoint(double& x,double& y,double sx,double sy);
        void ChangeLine(CLine * pNewLine,CLine * pLine);
        void ChangeCircle(CCircle * pNewCircle,CCircle * pCircle);
           void ChangeRectangle ( CRectangle * pNewRectangle, CRectangle *
pRectangle);
        void ChangePnt(CPnt * pNewPnt,CPnt * pPnt);
        void ChangeText(CText * pNewText,CText * pText);
        void ChangeEntity(CDC * pDC);

        void PickEntity(CDC * pDC);    //拾取实体
        void AddEntity();              //在实体链表中添加实体
        void DeleteEntity(CDC * pDC);//在实体链表中删除实体

        //以下几个函数用于实体平移,旋转,缩放等的操作
        void ModifyLButton(CDC * pDC);
        void ModifyRButton(CDC * pDC);
        void ModifyMove(CDC * pDC);

        void OSnap(CDC * pDC,BOOL bSure);        //对象捕捉
        void Draw(CDC * pDC);                    //绘制系统中的所有实体
        void Create(CDC * pDC,BOOL bSure);       //创建新实体
        virtual ~CCadDoc();
    #ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& dc) const;
    #endif

    protected:

    // Generated message map functions
    protected:
        void DrawOSnapPnt(CDC * pDC,BOOL bOSnapped);    //绘制对象捕捉点
```

```
        //{{AFX_MSG(CCadDoc)
            // NOTE - the ClassWizard will add and remove member functions here.
            //    DO NOT EDIT what you see in these blocks of generated code !
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif // !defined(AFX_CADDOC_H__12E57A0D_ADD2_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
```

**程序清单 12.18　文档定义实现文件 CadDoc.cpp**

```
// CadDoc.cpp : implementation of the CCadDoc class
//

#include "stdafx.h"
#include "Cad.h"
#include "CadDoc.h"
#include "CadView.h"

#include <math.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CCadDoc

IMPLEMENT_DYNCREATE(CCadDoc, CDocument)

BEGIN_MESSAGE_MAP(CCadDoc, CDocument)
    //{{AFX_MSG_MAP(CCadDoc)
        // NOTE - the ClassWizard will add and remove mapping macros here.
        //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
```

```
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
// CCadDoc construction/destruction

CCadDoc::CCadDoc()
{
	//创建链表
	m_pEntityHead = new CEntity;
	m_pEntityTail = m_pEntityHead;
	m_pCurEntity = m_pEntityHead;

	//设置当前选择集中实体的数目
	m_nEntitySelCnt = 0;
}

CCadDoc::~CCadDoc()
{
	//释放实体链表所占内存
	CEntity *	pEntity = m_pEntityHead;
	while(m_pEntityHead! = m_pEntityTail)
	{
		pEntity = m_pEntityHead;
		m_pEntityHead = pEntity->m_pNext;
		delete pEntity;
	}
	delete m_pEntityTail;
}

BOOL CCadDoc::OnNewDocument()
{
	if (! CDocument::OnNewDocument())
		return FALSE;

	// TODO: add reinitialization code here
	// (SDI documents will reuse this document)

	return TRUE;
}

////////////////////////////////////////////////////////////////////////////
```

```
// CCadDoc serialization

void CCadDoc::Serialize(CArchive& ar)
{
	if (ar.IsStoring())
	{
		// TODO: add storing code here
	}
	else
	{
		// TODO: add loading code here
	}
}

/////////////////////////////////////////////////////////////////////////////
// CCadDoc diagnostics

#ifdef _DEBUG
void CCadDoc::AssertValid() const
{
	CDocument::AssertValid();
}

void CCadDoc::Dump(CDumpContext& dc) const
{
	CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CCadDoc commands
//创建新实体
void CCadDoc::Create(CDC * pDC,BOOL bSure)
{
	BOOL		bIsCreated = FALSE;
	CEntity *	pNewEntity = NULL;
	double		x[16],y[16];
	int		cnt = 0;

	switch(g_nCurTask)
	{
	case ID_DRAW_LINE:
		bIsCreated = m_Line.Create(pDC,bSure);
```

```
            if(bIsCreated = = TRUE)
            {
                CLine * pNewLine = new CLine(m_Line.m_dSx,m_Line.m_dSy,
                                                    m_Line.m_dEx,m_Line.m_
dEy);
                pNewEntity = (CEntity * ) pNewLine;
                pNewLine - > Draw(pDC);

                x[cnt] = m_Line.m_dSx;
                y[cnt] = m_Line.m_dSy;
                cnt ++ ;
                x[cnt] = m_Line.m_dEx;
                y[cnt] = m_Line.m_dEy;
                cnt ++ ;
            }
            break;
        case ID_DRAW_CIRCLE:
        case ID_DRAW_ARC:
        case ID_DRAW_FILLED_CIRCLE:
            bIsCreated = m_Circle.Create(pDC,bSure);
            if(bIsCreated = = TRUE)
            {
                CCircle * pNewCircle;
                if(g_nCurTask = = ID_DRAW_CIRCLE)
                {
                    pNewCircle = new CCircle(m_Circle.m_dCenterX,
                                        m_Circle.m_dCenterY,m_Circle.
m_dR);
                }
                else if(g_nCurTask = = ID_DRAW_ARC)
                {
                    pNewCircle = new CCircle(m_Circle.m_dCenterX,
                                    m_Circle.m_dCenterY,m_Circle.m_dR,
                                    m_Circle.m_dSx,m_Circle.m_dSy,
                                    m_Circle.m_dEx,m_Circle.m_dEy);
                }
                else
                {
                    pNewCircle = new CCircle(m_Circle.m_dCenterX,
                                        m_Circle.m_dCenterY,m_Circle.
m_dR,
                                    m_Circle.m_nFillPattern);
                }
```

```
                pNewEntity = (CEntity * ) pNewCircle;
                pNewCircle - > Draw(pDC);

                x[cnt] = m_Circle.m_dCenterX + m_Circle.m_dR;
                y[cnt] = m_Circle.m_dCenterY + m_Circle.m_dR;
                cnt ++ ;
                x[cnt] = m_Circle.m_dCenterX - m_Circle.m_dR;
                y[cnt] = m_Circle.m_dCenterY - m_Circle.m_dR;
                cnt ++ ;
            }
            break;
        case ID_DRAW_RECTANGLE:
        case ID_DRAW_FILLED_RECTANGLE:
            bIsCreated = m_Rectangle.Create(pDC,bSure);
            if(bIsCreated = = TRUE)
            {
                CRectangle * pNewRectangle;
                if(g_nCurTask = = ID_DRAW_RECTANGLE)
                {
                    pNewRectangle = new CRectangle(m_Rectangle.m_dLeft,
                                    m_Rectangle.m_dTop,m_Rectangle.m_
dRight,
                                    m_Rectangle.m_dBottom);
                }
                else
                {
                    pNewRectangle = new CRectangle(m_Rectangle.m_dLeft,
                                    m_Rectangle.m_dTop,m_Rectangle.m_
dRight,
                                    m_Rectangle.m_dBottom,m_Rectangle.
m_nFillPattern);
                }
                pNewEntity = (CEntity * ) pNewRectangle;
                pNewRectangle - > Draw(pDC);

                x[cnt] = m_Rectangle.m_dLeft;
                y[cnt] = m_Rectangle.m_dTop;
                cnt ++ ;
                x[cnt] = m_Rectangle.m_dRight;
                y[cnt] = m_Rectangle.m_dBottom;
                cnt ++ ;
            }
            break;
```

```
case ID_DRAW_POINT:
      bIsCreated = m_Pnt.Create(pDC,bSure);
      if(bIsCreated = = TRUE)
      {
            CPnt * pNewPnt = new CPnt(m_Pnt.m_dX,m_Pnt.m_dY);
            pNewEntity = (CEntity * ) pNewPnt;
            pNewPnt - > Draw(pDC);

            x[cnt] = m_Pnt.m_dX;
            y[cnt] = m_Pnt.m_dY;
            cnt ++ ;
      }
      break;
case ID_DRAW_TEXT:
      bIsCreated = m_Text.Create(pDC,bSure);
      if(bIsCreated = = TRUE)
      {
            CText * pNewText = new CText(m_Text.m_dX,m_Text.m_dY,
                                          m_Text.m_strText);
            pNewEntity = (CEntity * ) pNewText;
            pNewText - > Draw(pDC);

            x[cnt] = m_Text.m_dX;
            y[cnt] = m_Text.m_dY;
            cnt ++ ;
      }
      break;
default:
      break;
}

if(bIsCreated = = TRUE&&pNewEntity! = NULL)
{
      pNewEntity - > m_pNext = m_pEntityHead;
      m_pEntityHead - > m_pPre = pNewEntity;
      m_pEntityHead = pNewEntity;

      //获取当前图形所在空间大小
      for(int i = 0;i < cnt;i ++ )
      {
            if(x[i] > g_dMaxX) g_dMaxX = x[i];
            if(x[i] < g_dMinX) g_dMinX = x[i];
            if(y[i] > g_dMaxY) g_dMaxY = y[i];
```

```
                if(y[i] < g_dMinY) g_dMinY = y[i];
            }
        }
    }

    //绘制系统中的所有实体
    void CCadDoc::Draw(CDC * pDC)
    {
        //遍历实体链表绘图
        CEntity * pEntity = m_pEntityHead;
        while(pEntity! = m_pEntityTail)
        {
            switch(pEntity->m_nType)
            {
            case ID_DRAW_LINE:
                ((CLine * ) pEntity)->Draw(pDC);
                break;
            case ID_DRAW_CIRCLE:
            case ID_DRAW_ARC:
            case ID_DRAW_FILLED_CIRCLE:
                ((CCircle * ) pEntity)->Draw(pDC);
                break;
            case ID_DRAW_RECTANGLE:
            case ID_DRAW_FILLED_RECTANGLE:
                ((CRectangle * ) pEntity)->Draw(pDC);
                break;
            case ID_DRAW_POINT:
                ((CPnt * ) pEntity)->Draw(pDC);
                break;
            case ID_DRAW_TEXT:
                ((CText * ) pEntity)->Draw(pDC);
                break;
            default:
                break;
            }
            pEntity = pEntity->m_pNext;
        }
    }

    //对象捕捉
    void CCadDoc::OSnap(CDC * pDC, BOOL bSure)
    {
        //捕捉到对象
```

```
if(bSure = = TRUE&&g_bOSnapped = = TRUE)
{
    DrawOSnapPnt(pDC,FALSE);
    g_dCurX = g_dPreX;
    g_dCurY = g_dPreY;
    g_nOpqTask = 0;
    return ;
}

//遍历实体链表查看是否捕捉到对象
if(m_pCurEntity = = NULL||m_pCurEntity = = m_pEntityTail)
{
    m_pCurEntity = m_pEntityHead;
}
BOOL bOSnapped = FALSE;
while(m_pCurEntity! = m_pEntityTail&&bOSnapped = = FALSE)
{
    switch(m_pCurEntity - > m_nType)
    {
    case ID_DRAW_LINE:
        bOSnapped = ((CLine * ) m_pCurEntity) - > OSnap();
        break;
    case ID_DRAW_CIRCLE:
    case ID_DRAW_ARC:
    case ID_DRAW_FILLED_CIRCLE:
        bOSnapped = ((CCircle * ) m_pCurEntity) - > OSnap();
        break;
    case ID_DRAW_RECTANGLE:
    case ID_DRAW_FILLED_RECTANGLE:
        bOSnapped = ((CRectangle * ) m_pCurEntity) - > OSnap();
        break;
    case ID_DRAW_POINT:
        bOSnapped = ((CPnt * ) m_pCurEntity) - > OSnap();
        break;
    case ID_DRAW_TEXT:
        bOSnapped = ((CText * ) m_pCurEntity) - > OSnap();
        break;
    default:
        break;
    }
    m_pCurEntity = m_pCurEntity - > m_pNext;
}
//绘制捕捉点
```

```
        DrawOSnapPnt(pDC,bOSnapped);
    }

    //绘制对象捕捉点
    void CCadDoc::DrawOSnapPnt(CDC * pDC,BOOL bOSnapped)
    {
        static double dPreX,dPreY;

        //设置 R2_NOT 的绘图方式
        int nDrawMode = pDC->SetROP2(R2_NOT);

        //覆盖先前的捕捉标记
        if(g_bOSnapped==TRUE)
        {
            CPoint      point;
            g_pCurView->WorldToScreen(point,dPreX,dPreY);
            pDC->MoveTo(point.x-4,point.y-4);
            pDC->LineTo(point.x+4,point.y-4);
            pDC->LineTo(point.x+4,point.y+4);
            pDC->LineTo(point.x-4,point.y+4);
            pDC->LineTo(point.x-4,point.y-4);
        }

        //绘制新的捕捉标记
        if(bOSnapped==TRUE)
        {
            CPoint      point;
            g_pCurView->WorldToScreen(point,g_dCurX,g_dCurY);
            pDC->MoveTo(point.x-4,point.y-4);
            pDC->LineTo(point.x+4,point.y-4);
            pDC->LineTo(point.x+4,point.y+4);
            pDC->LineTo(point.x-4,point.y+4);
            pDC->LineTo(point.x-4,point.y-4);
            dPreX = g_dCurX;
            dPreY = g_dCurY;
        }
        //设置先前的绘图方式
        pDC->SetROP2(nDrawMode);

        g_bOSnapped = bOSnapped;
        return ;
    }
```

```
//拾取实体
void CCadDoc::PickEntity(CDC* pDC)
{
    BOOL  bPicked=FALSE;

    if(m_pCurEntity==NULL||m_pCurEntity==m_pEntityTail)
    {
        m_pCurEntity=m_pEntityHead;
    }

    //遍历实体链表查看是否拾取到实体
    while(m_pCurEntity!=m_pEntityTail)
    {
        switch(m_pCurEntity->m_nType)
        {
        case ID_DRAW_LINE:
            bPicked=((CLine*)m_pCurEntity)->Pick();
            if(bPicked==TRUE)
                ((CLine *)m_pCurEntity)->Draw(pDC,RGB(255,0,0),PS_
DASHDOT);
            break;
        case ID_DRAW_CIRCLE:
        case ID_DRAW_ARC:
        case ID_DRAW_FILLED_CIRCLE:
            bPicked=((CCircle*)m_pCurEntity)->Pick();
            if(bPicked==TRUE)
                ((CCircle *)m_pCurEntity)->Draw(pDC,RGB(255,0,0),
PS_DASHDOT);
            break;
        case ID_DRAW_RECTANGLE:
        case ID_DRAW_FILLED_RECTANGLE:
            bPicked=((CRectangle*)m_pCurEntity)->Pick();
            if(bPicked==TRUE)
                ((CRectangle *)m_pCurEntity)->Draw(pDC,RGB(255,0,
0),PS_DASHDOT);
            break;
        case ID_DRAW_POINT:
            bPicked=((CPnt*)m_pCurEntity)->Pick();
            if(bPicked==TRUE)
                ((CPnt *)m_pCurEntity)->Draw(pDC,RGB(255,0,0),PS_
DASHDOT);
            break;
        case ID_DRAW_TEXT:
```

```
                    bPicked = ((CText * ) m_pCurEntity) - > Pick();
                    if(bPicked = = TRUE)
                        ((CText * )m_pCurEntity) - > Draw(pDC,RGB(255,0,0),PS_
DASHDOT);
                    break;
                default:
                    break;
                }

                if(bPicked = = TRUE)
                {
                    BOOL  bSelected = FALSE;
                    for(int i = 0;i < m_nEntitySelCnt;i ++ )
                    {
                            if ( m_ pEntitySelected [ m_ nEntitySelCnt ] = - m_
pCurEntity)
                        {
                            bSelected = TRUE;
                            break;
                        }
                    }
                    if(bSelected = = TRUE)     break;

                    m_pEntitySelected[m_nEntitySelCnt] = m_pCurEntity;
                    m_nEntitySelCnt ++ ;
                    if(m_nEntitySelCnt > 10)
                    {
                        AfxMessageBox("You can only select ten entities at
most!");
                        g_nStep ++ ;
                    }
                    break;
                }
                m_pCurEntity = m_pCurEntity - > m_pNext;
            }
        }

    //在实体链表中添加实体
    void CCadDoc::AddEntity()
    {
        CEntity *     pNewEntity = NULL;

        //在链表头部添加新实体链节
```

```
for(int i = m_nEntitySelCnt - 1;i >= 0;i - - )
{
    switch(m_pEntitySelected[i] - >m_nType)
    {
    case ID_DRAW_LINE:
        {
        CLine * pNewLine = new CLine;
        * pNewLine = * ((CLine * )m_pEntitySelected[i]);
        pNewEntity = (CEntity * ) pNewLine;
        break;
        }
    case ID_DRAW_CIRCLE:
    case ID_DRAW_ARC:
    case ID_DRAW_FILLED_CIRCLE:
        {
        CCircle * pNewCircle = new CCircle;
        pNewEntity = (CEntity * ) pNewCircle;
        * pNewCircle = * ((CCircle * )m_pEntitySelected[i]);
        break;
        }
    case ID_DRAW_RECTANGLE:
    case ID_DRAW_FILLED_RECTANGLE:
        {
        CRectangle * pNewRectangle = new CRectangle;
        * pNewRectangle = * ((CRectangle * )m_pEntitySelected[i]);
        pNewEntity = (CEntity * ) pNewRectangle;
        break;
        }
    case ID_DRAW_POINT:
        {
        CPnt * pNewPnt = new CPnt;
        * pNewPnt = * ((CPnt * )m_pEntitySelected[i]);
        pNewEntity = (CEntity * ) pNewPnt;
        break;
        }
    case ID_DRAW_TEXT:
        {
        CText * pNewText = new CText;
        * pNewText = * ((CText * )m_pEntitySelected[i]);
        pNewEntity = (CEntity * ) pNewText;
        break;
        }
    default:
```

```
                break;
            }

            pNewEntity->m_pNext = m_pEntityHead;
            m_pEntityHead->m_pPre = pNewEntity;
            m_pEntityHead = pNewEntity;
        }
    }

    //在实体链表中删除实体
    void CCadDoc::DeleteEntity(CDC* pDC)
    {
        CEntity* pEntity = NULL;
        CEntity* pEntity2 = NULL;

        //删除实体链表中选择集记录的实体链节
        for(int i = 0;i < m_nEntitySelCnt;i++)
        {
            pEntity = m_pEntitySelected[i];

            switch(pEntity->m_nType)
            {
            case ID_DRAW_LINE:
                ((CLine *)pEntity)->Draw(pDC,RGB(255,255,255),PS_SOLID);
                break;
            case ID_DRAW_CIRCLE:
            case ID_DRAW_ARC:
            case ID_DRAW_FILLED_CIRCLE:
                ((CCircle *)pEntity)->Draw(pDC,RGB(255,255,255),PS
SOLID);
                break;
            case ID_DRAW_RECTANGLE:
            case ID_DRAW_FILLED_RECTANGLE:
                ((CRectangle *)pEntity)->Draw(pDC,RGB(255,255,255),PS
SOLID);
                break;
            case ID_DRAW_POINT:
                ((CPnt *)pEntity)->Draw(pDC,RGB(255,255,255),PS_SOLID);
                break;
            case ID_DRAW_TEXT:
                ((CText *)pEntity)->Draw(pDC,RGB(255,255,255),PS_SOLID);
                break;
            default:
```

```
            break;
        }

        if(pEntity! = m_pEntityHead)
        {
            pEntity2 = pEntity - > m_pNext;
            pEntity2 - > m_pPre = pEntity - > m_pPre;
            pEntity2 = pEntity - > m_pPre;
            pEntity2 - > m_pNext = pEntity - > m_pNext;
        }
        else
        {
            m_pEntityHead = m_pEntityHead - > m_pNext;
        }

        delete pEntity;
    }

    m_pCurEntity = m_pEntityHead;
    m_nEntitySelCnt = 0;
}

//修改对象时按下鼠标左键时的操作
void CCadDoc::ModifyLButton(CDC * pDC)
{
    switch(g_nStep)
    {
    case 0:
        PickEntity(pDC);
        break;
    case 1:
        g_nStep ++ ;
        break;
    case 2:
        {
        g_nStep = 0;
        g_nCurTask = 0;

        //设置 R2_NOT 的绘图方式
        int nDrawMode = pDC - > SetROP2(R2_NOT);
        CPoint      orgPoint,prePoint;
        g_pCurView - > WorldToScreen(orgPoint,g_dOrgX,g_dOrgY);
        g_pCurView - > WorldToScreen(prePoint,g_dPreX,g_dPreY);
```

```
            //覆盖先前的线
            pDC->MoveTo(orgPoint);
            pDC->LineTo(prePoint);
            //设置先前的绘图方式
            pDC->SetROP2(nDrawMode);

            DeleteEntity(pDC);
            Draw(pDC);
            break;
            }
        default:
            break;
        }
    }

//修改对象时按下鼠标左键时的操作
void CCadDoc::ModifyRButton(CDC* pDC)
{
    if(g_nStep==0)
    {
        switch(g_nCurTask)
        {
        case ID_MODIFY_ERASE:
            DeleteEntity(pDC);
            g_nStep=0;
            g_nCurTask=0;
            break;
        case ID_MODIFY_MIRROR:
        case ID_MODIFY_MOVE:
        case ID_MODIFY_ROTATE:
        case ID_MODIFY_SCALE:
            AddEntity();
            g_nStep++;
            break;
        default:
            break;
        }
    }
}

//实体平移、旋转、缩放时设置变换坐标矩阵
void CCadDoc::ModifyMove(CDC* pDC)
{
```

```
        if(g_nStep==0) return;

        switch(g_nCurTask)
        {
        case ID_MODIFY_MIRROR:
            if(g_nStep==2)
            {
                InitChgA();
                double a=g_dCurX-g_dOrgX;
                double b=g_dCurY-g_dOrgY;
                double c=sqrt(a*a+b*b);
                double sin_theta=b/c;
                double cos_theta=a/c;

                m_dChgA[0][0]=cos_theta*cos_theta-sin_theta*sin_theta;
                m_dChgA[0][1]=2*sin_theta*cos_theta;
                m_dChgA[1][0]=m_dChgA[0][1];;
                m_dChgA[1][1]=-m_dChgA[0][0];
                m_dChgA[2][0]=g_dOrgX*m_dChgA[1][1]-g_dOrgY*m_dChgA[0]
[1]+g_dOrgX;
                m_dChgA[2][1]=-g_dOrgX*m_dChgA[0][1]+g_dOrgY*m_dChgA
[0][0]+g_dOrgY;
                ChangeEntity(pDC);
            }
            break;
        case ID_MODIFY_MOVE:
            if(g_nStep==2)
            {
                InitChgA();
                m_dChgA[2][0]=g_dCurX-g_dOrgX;
                m_dChgA[2][1]=g_dCurY-g_dOrgY;

                ChangeEntity(pDC);
            }
            break;
        case ID_MODIFY_ROTATE:
            if(g_nStep==2)
            {
                InitChgA();
                double a=g_dCurX-g_dOrgX;
                double b=g_dCurY-g_dOrgY;
                double c=sqrt(a*a+b*b);
                double sin_theta=b/c;
```

```
                    double cos_theta = a/c;

                    m_dChgA[0][0] = cos_theta;
                    m_dChgA[0][1] = sin_theta;
                    m_dChgA[1][0] = - sin_theta;
                    m_dChgA[1][1] = cos_theta;
                    m_dChgA[2][0] = (1.0 - cos_theta) * g_dOrgX + sin_theta * g
dOrgY;
                    m_dChgA[2][1] = (1.0 - cos_theta) * g_dOrgY - sin_theta * g
dOrgX;
                    ChangeEntity(pDC);
               }
               break;
          case ID_MODIFY_SCALE:
               if(g_nStep == 2)
               {
                    double scaleX = fabs(g_dCurX - g_dOrgX)/(20 * g_pCurView -> m_
dScreenToWorld);
                    double scaleY = fabs(g_dCurY - g_dOrgY)/(20 * g_pCurView -> m_
dScreenToWorld);

                    if(scaleX > scaleY) scaleY = scaleX;
                    else  scaleX = scaleY;

                    InitChgA();
                    m_dChgA[0][0] = scaleX;
                    m_dChgA[1][1] = scaleY;
                    m_dChgA[2][0] = (1.0 - scaleX) * g_dOrgX;
                    m_dChgA[2][1] = (1.0 - scaleY) * g_dOrgY;

                    ChangeEntity(pDC);
               }
               break;
          default:
               break;
          }
     }

     //对选择集中的实体进行变换
     void CCadDoc::ChangeEntity(CDC * pDC)
     {
          CEntity * pEntity;
          //设置 R2_NOT 的绘图方式
```

```
int nDrawMode = pDC - > SetROP2(R2_NOT);

CPoint      orgPoint,curPoint,prePoint;
g_pCurView - > WorldToScreen(orgPoint,g_dOrgX,g_dOrgY);
g_pCurView - > WorldToScreen(curPoint,g_dCurX,g_dCurY);
g_pCurView - > WorldToScreen(prePoint,g_dPreX,g_dPreY);
//覆盖先前的线
pDC - > MoveTo(orgPoint);
pDC - > LineTo(prePoint);
//绘制新的线
pDC - > MoveTo(orgPoint);
pDC - > LineTo(curPoint);

pEntity = m_pEntityHead;
for(int i = 0;i < m_nEntitySelCnt;i ++ )
{
      switch(m_pEntitySelected[i] - > m_nType)
      {
      case ID_DRAW_LINE:
            {
            CLine * pLine = (CLine * )m_pEntitySelected[i];
            CLine * pNewLine;

            if(pEntity - > m_nType = = ID_DRAW_LINE)
            {
                  pNewLine = (CLine * )pEntity;
            }
            else
            {
                  AfxMessageBox("Error in MoveEntity()!");
                  break;
            }

            pNewLine - > Draw(pDC);
            ChangeLine(pNewLine,pLine);
            pNewLine - > Draw(pDC);
            break;
            }
      case ID_DRAW_CIRCLE:
      case ID_DRAW_ARC:
      case ID_DRAW_FILLED_CIRCLE:
            {
            CCircle * pCircle = (CCircle * )m_pEntitySelected[i];
```

```
                CCircle * pNewCircle;

                if(pEntity - >m_nType = = ID_DRAW_CIRCLE
                    ||pEntity - >m_nType = = ID_DRAW_ARC
                    ||pEntity - >m_nType = = ID_DRAW_FILLED_CIRCLE)
                {
                    pNewCircle = (CCircle * )pEntity;
                }
                else
                {
                    AfxMessageBox("Error in MoveEntity()!");
                    break;
                }

                pNewCircle - >Draw(pDC);
                ChangeCircle(pNewCircle,pCircle);
                pNewCircle - >Draw(pDC);
                break;
                }
            case ID_DRAW_RECTANGLE:
            case ID_DRAW_FILLED_RECTANGLE:
                {
                 CRectangle * pRectangle = (CRectangle * )m_pEntitySelected
[i];
                CRectangle * pNewRectangle;

                if(pEntity - >m_nType = = ID_DRAW_RECTANGLE
                    ||pEntity - >m_nType = = ID_DRAW_FILLED_RECTANGLE)
                {
                    pNewRectangle = (CRectangle * )pEntity;
                }
                else
                {
                    AfxMessageBox("Error in MoveEntity()!");
                    break;
                }

                pNewRectangle - >Draw(pDC);
                ChangeRectangle(pNewRectangle,pRectangle);
                pNewRectangle - >Draw(pDC);
                break;
                }
            case ID_DRAW_POINT:
```

```
        {
        CPnt * pPnt = (CPnt * )m_pEntitySelected[i];
        CPnt * pNewPnt;

        if(pEntity->m_nType == ID_DRAW_POINT)
        {
            pNewPnt = (CPnt * )pEntity;
        }
        else
        {
            AfxMessageBox("Error in MoveEntity()!");
            break;
        }

        pNewPnt->Draw(pDC);
        ChangePnt(pNewPnt,pPnt);
        pNewPnt->Draw(pDC);
        break;
        }
    case ID_DRAW_TEXT:
        {
        CText * pText = (CText * )m_pEntitySelected[i];
        CText * pNewText;

        if(pEntity->m_nType == ID_DRAW_TEXT)
        {
            pNewText = (CText * )pEntity;
        }
        else
        {
            AfxMessageBox("Error in MoveEntity()!");
            break;
        }

        pNewText->Draw(pDC);
        ChangeText(pNewText,pText);
        pNewText->Draw(pDC);
        break;
        }
    default:
        break;
    }
```

```
            pEntity = pEntity - >m_pNext;
        }
        //设置先前的绘图方式
        pDC - > SetROP2(nDrawMode);
}

//初始化坐标变换矩阵
void CCadDoc::InitChgA()
{
      for(int i = 0;i < 3;i ++ )
      {
            for(int j = 0;j < 3;j ++ )
            {
                  if(i = = j)
                        m_dChgA[i][j] = 1.0;
                  else
                        m_dChgA[i][j] = 0;
            }
      }
}

//对点进行坐标变换
void CCadDoc::ChangePoint(double& x,double& y,double sx,double sy)
{
      x = sx * m_dChgA[0][0] + sy * m_dChgA[1][0] + m_dChgA[2][0];
      y = sx * m_dChgA[0][1] + sy * m_dChgA[1][1] + m_dChgA[2][1];
}

//对线进行坐标变换
void CCadDoc::ChangeLine(CLine * pNewLine,CLine * pLine)
{
      ChangePoint(pNewLine - >m_dSx,pNewLine - >m_dSy,
                  pLine - >m_dSx,pLine - >m_dSy);
      ChangePoint(pNewLine - >m_dEx,pNewLine - >m_dEy,
                  pLine - >m_dEx,pLine - >m_dEy);
}

//对圆进行坐标变换
void CCadDoc::ChangeCircle(CCircle * pNewCircle,CCircle * pCircle)
{
      ChangePoint(pNewCircle - >m_dSx,pNewCircle - >m_dSy,
                  pCircle - >m_dSx, pCircle - >m_dSy);
      ChangePoint(pNewCircle - >m_dEx,pNewCircle - >m_dEy,
```

```
                    pCircle->m_dEx,   pCircle->m_dEy);
        ChangePoint(pNewCircle->m_dCenterX,pNewCircle->m_dCenterY,
                    pCircle->m_dCenterX,pCircle->m_dCenterY);
        double dx = pNewCircle->m_dSx - pNewCircle->m_dCenterX;
        double dy = pNewCircle->m_dSy - pNewCircle->m_dCenterY;
        pNewCircle->m_dR = sqrt(dx * dx + dy * dy);
    }

    //对矩形进行坐标变换
    void CCadDoc::ChangeRectangle(CRectangle * pNewRectangle, CRectangle *
pRectangle)
    {
        ChangePoint(pNewRectangle->m_dLeft,pNewRectangle->m_dTop,
                    pRectangle->m_dLeft,pRectangle->m_dTop);
        ChangePoint(pNewRectangle->m_dRight,pNewRectangle->m_dBottom,
                    pRectangle->m_dRight,pRectangle->m_dBottom);
    }

    //对点进行坐标变换
    void CCadDoc::ChangePnt(CPnt * pNewPnt,CPnt * pPnt)
    {
        ChangePoint(pNewPnt->m_dX,pNewPnt->m_dY,
                    pPnt->m_dX,pPnt->m_dY);
    }

    //对文字进行坐标变换
    void CCadDoc::ChangeText(CText * pNewText,CText * pText)
    {
        ChangePoint(pNewText->m_dX,pNewText->m_dY,
                    pText->m_dX,pText->m_dY);
    }
```

10. 实体基类文件 Entity.h 和 Entity.cpp

Entity.h 和 Entity.cpp 文件中含有与基本图元共有数据的定义。这两个文件中含有用于实现该应用程序层次框架结构中第四层的所有功能的函数(参见 12.2 节)。即,该文件所定义的 Centity 类为 Cline、CCircle、CRectangle、CPnt 和 CText 所继承,以便使基本图元中都含有记录实体颜色与线型的变量,都含有指向共同对象链节的指针,以形成链表。此外,该类中还有 CetTwoPntDis2()函数。该函数用于计算两个点之间的距离。

**程序清单 12.19 实体基类头文件 Entity.h**

```
// Entity.h: interface for the CEntity class.
//
//////////////////////////////////////////////////////////////////////
```

```
#if !defined(AFX_ENTITY_H__F2D97FC2_B2EC_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
#define AFX_ENTITY_H__F2D97FC2_B2EC_11CF_BF57_B29AF85F4C3B__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CEntity
{
public:
    CEntity();
    //计算两点间距离的平方值
    double GetTwoPntDis2(double x1,double y1,double x2,double y2);

    virtual ~CEntity();

    UINT        m_nType;
    COLORREF    m_nColor;
    UINT        m_rLineType;

    CEntity *    m_pNext;
    CEntity *    m_pPre;
};

#endif // !defined(AFX_ENTITY_H__F2D97FC2_B2EC_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
```

**程序清单 12.20　实体基类实现文件 Entity.cpp**

```
// Entity.cpp: implementation of the CEntity class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "Cad.h"
#include "Entity.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
```

```
//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CEntity::CEntity()
{
    m_nType = 0;
}

CEntity::~CEntity()
{

}

//计算两点间距离的平方值
double CEntity::GetTwoPntDis2(double x1, double y1, double x2, double y2)
{
    return (x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2);
}
```

11. 直线定义文件 Line.h 和 Line.cpp 文件

Line.h 和 Line.cpp 文件中含有关于直线的定义。这两个文件中含有用于实现直线创建、绘制拾取以及对象捕捉的函数。这两个文件定义的 CLine 类处于该应用程序层次框架结构中的第三层。

**程序清单 12.21 直线定义头文件 Line.h**

```
// Line.h: interface for the CLine class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_LINE_H__804303E3_B27F_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
#define AFX_LINE_H__804303E3_B27F_11CF_BF57_B29AF85F4C3B__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CLine: public CEntity
{
public:
    BOOL Pick();
    BOOL OSnap();
```

```
    BOOL Create(CDC* pDC,BOOL bSure);
    void Draw(CDC* pDC,UINT nColor = 0,UINT nLineType = 0);
    CLine();
    CLine(double dSx,double dSy,double dEx,double dEy);
    virtual ~CLine();

    double          m_dSx,m_dSy;
    double          m_dEx,m_dEy;
};

#endif // !defined(AFX_LINE_H__804303E3_B27F_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
```

**程序清单 12.22 直线定义实现文件 Line.cpp**

```
// Line.cpp: implementation of the CLine class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "Cad.h"

#include "Entity.h"
#include "Line.h"
#include "CadDoc.h"
#include "CadView.h"

#include <math.h>

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CLine::CLine()
{
    m_dSx = m_dSy = 0.0;
    m_dEx = m_dEy = 0.0;
    m_nType = ID_DRAW_LINE;
```

```
        m_nColor = RGB(0,0,0);
        m_nLineType = PS_SOLID;
    }

    CLine::CLine(double dSx,double dSy,double dEx,double dEy)
    {
        m_dSx = dSx;
        m_dSy = dSy;
        m_dEx = dEx;
        m_dEy = dEy;
        m_nType = ID_DRAW_LINE;
        m_nColor = g_nCurColor;
        m_nLineType = g_nCurLineType;
    }
    CLine::~CLine()
    {

    }

    BOOL CLine::Create(CDC* pDC,BOOL bSure)
    {
        BOOL  ret = FALSE;

        if(bSure == TRUE)
        {
            CStatusBar* pStatus = (CStatusBar*)
                AfxGetApp()->m_pMainWnd->GetDescendantWindow(ID_VIEW_
STATUS_BAR);
            ASSERT(pStatus);

            switch(g_nStep)
            {
            case 0:
                m_dSx = g_dCurX;
                m_dSy = g_dCurY;
                g_nStep++;
                pStatus->SetPaneText(0,"Enter the end point");
                break;
            case 1:
                m_dEx = g_dCurX;
                m_dEy = g_dCurY;
                g_nStep = 0;
                pStatus->SetPaneText(0,"Enter the first point");
```

```
                ret = TRUE;
                break;
            default:
                break;
            }
        }
        else
        {
            if(g_nStep! =1) return FALSE;
            //设置 R2_NOT 的绘图方式
            int nDrawMode = pDC - > SetROP2(R2_NOT);

            CPoint      orgPoint,curPoint,prePoint;
            g_pCurView - > WorldToScreen(orgPoint,g_dOrgX,g_dOrgY);
            g_pCurView - > WorldToScreen(curPoint,g_dCurX,g_dCurY);
            g_pCurView - > WorldToScreen(prePoint,g_dPreX,g_dPreY);
            //覆盖先前的线
            pDC - > MoveTo(orgPoint);
            pDC - > LineTo(prePoint);
            //绘制新的线
            pDC - > MoveTo(orgPoint);
            pDC - > LineTo(curPoint);
            //设置先前的绘图方式
            pDC - > SetROP2(nDrawMode);
        }
        return ret;
    }

    void CLine::Draw(CDC * pDC,UINT nColor,UINT nLineType)
    {
        CPen * pNewPen = new CPen;
        if(nColor = = 0&&nLineType = = 0)
            pNewPen - > CreatePen(m_nLineType,1,m_nColor);
        else
            pNewPen - > CreatePen(nLineType,1,nColor);

        CPen * pOldPen = pDC - > SelectObject( pNewPen );

        CPoint      point;
        g_pCurView - > WorldToScreen(point,m_dSx,m_dSy);
        pDC - > MoveTo(point);
        g_pCurView - > WorldToScreen(point,m_dEx,m_dEy);
        pDC - > LineTo(point);
```

```
    pDC->SelectObject( pOldPen );
}

BOOL CLine::OSnap()
{
    double      d;
    double      sd=g_nOSnapSize*g_pCurView->m_dScreenToWorld;
    double      x[3],y[3];
    int         cnt=0;
    BOOL  ret=FALSE;

    switch(g_nOpqTask)
    {
    case ID_OSNAP_ENDPOINT:
        x[cnt]=m_dSx;
        y[cnt]=m_dSy;
        cnt++;
        x[cnt]=m_dEx;
        y[cnt]=m_dEy;
        cnt++;
        break;
    case ID_OSNAP_MIDPOINT:
        x[cnt]=(m_dSx-m_dEx)/2.0;
        y[cnt]=(m_dSy-m_dEy)/2.0;
        cnt++;
        break;
    default:
        if((g_nOSnapType&1)==1)
        {
            x[cnt]=m_dSx;
            y[cnt]=m_dSy;
            cnt++;
            x[cnt]=m_dEx;
            y[cnt]=m_dEy;
            cnt++;
        }
        if((g_nOSnapType&2)==2)
        {
            x[cnt]=(m_dSx+m_dEx)/2.0;
            y[cnt]=(m_dSy+m_dEy)/2.0;
            cnt++;
        }
```

```
            break;
        }

        for(int i = 0;i < cnt;i ++ )
        {
            d = GetTwoPntDis2(g_dCurX,g_dCurY,x[i],y[i]);
            if(d < sd * sd)
            {
                g_dCurX = x[i];
                g_dCurY = y[i];
                ret = TRUE;
                break;
            }
        }

        return ret;
    }

    BOOL CLine::Pick()
    {
        BOOL  ret = FALSE;
        double      x = g_dCurX - m_dSx;
        double      y = g_dCurY - m_dSy;

        double      a = m_dEx - m_dSx;
        double      b = m_dEy - m_dSy;
        double      c = sqrt(a * a + b * b);

        double      sin_theta = b/c;
        double  cos_theta = a/c;

        double      chg_x = x * cos_theta + y * sin_theta;
        double      chg_y = -x * sin_theta + y * cos_theta;

        double      sd = 5.0 * g_pCurView -> m_dScreenToWorld;
        if(fabs(chg_y) < sd&&(chg_x > = 0&&chg_x < = c))
            ret = TRUE;

        return ret;
    }
```

12. 圆、圆弧、填充圆定义文件 Circle.h 和 Circle.cpp

Circle.h 和 Circle.cpp 文件中含有关于圆的定义。这两个文件中含有用于实现圆创建、

绘制拾取以及对象捕捉的函数。这两个文件定义的 CCircle 类处于该应用程序层次框架结构中的第三层。

**程序清单 12.23 圆、圆弧、填充圆定义头文件 Circle.h**

```
// Circle.h: interface for the CCircle class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_CIRCLE_H__F2D97FC3_B2EC_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
#define AFX_CIRCLE_H__F2D97FC3_B2EC_11CF_BF57_B29AF85F4C3B__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CCircle  : public CEntity
{
public:
        BOOL Pick();
        BOOL OSnap();
        BOOL Create(CDC* pDC,BOOL bSure);
        void Draw(CDC* pDC,UINT nColor=0,UINT nLineType=0);
        CCircle();
        CCircle(double x,double y,double r,double sx=0,double sy=0,double ex=
0,double ey=0);
        CCircle(double x,double y,double r,UINT  nFillPattern);
        virtual ~CCircle();

        double m_dCenterX,m_dCenterY;
        double m_dR;
        double m_dSx,m_dSy;
        double m_dEx,m_dEy;
        UINT  m_nFillPattern;
};

#endif // !defined(AFX_CIRCLE_H__F2D97FC3_B2EC_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
```

**程序清单 12.24 圆、圆弧、填充圆定义实现文件 Circle.cpp**

```
// Circle.cpp: implementation of the CCircle class.
//
//////////////////////////////////////////////////////////////////////
```

```
#include "stdafx.h"
#include "Cad.h"

#include "Entity.h"
#include "Circle.h"
#include "CadDoc.h"
#include "CadView.h"

#include "FillDlg.h"
#include <math.h>

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CCircle::CCircle()
{
        m_dCenterX = m_dCenterY = 0;
        m_dR = 0;
        m_dSx = m_dSy = 0;
        m_dEx = m_dEy = 0;

        m_nType = ID_DRAW_CIRCLE;
        m_nColor = RGB(0,0,0);
        m_nLineType = PS_SOLID;
}

CCircle::CCircle(double x,double y,double r,double sx,double sy,double ex,
double ey)
{
        m_dCenterX = x;
        m_dCenterY = y;
        m_dR = r;
        m_dSx = sx;
        m_dSy = sy;
        m_dEx = ex;
        m_dEy = ey;
```

```
    m_nType = g_nCurTask;
    m_nColor = g_nCurColor;
    m_nLineType = g_nCurLineType;
}

CCircle::CCircle(double x,double y,double r,UINT nFillPattern)
{
    m_dCenterX = x;
    m_dCenterY = y;
    m_dR = r;
    m_dSx = m_dSy = 0;
    m_dEx = m_dEy = 0;
    m_nFillPattern = nFillPattern;

    m_nType = g_nCurTask;
    m_nColor = g_nCurColor;
    m_nLineType = g_nCurLineType;
}

CCircle::~CCircle()
{

}

BOOL CCircle::Create(CDC * pDC,BOOL bSure)
{
    BOOL  ret = FALSE;

    if(bSure == TRUE)
    {
        CStatusBar * pStatus = (CStatusBar * )
            AfxGetApp() -> m_pMainWnd -> GetDescendantWindow(ID_VIEW_
STATUS_BAR);
        ASSERT(pStatus);

        switch(g_nStep)
        {
        case 0:
            m_dCenterX = g_dCurX;
            m_dCenterY = g_dCurY;
            g_nStep++;
            pStatus -> SetPaneText(0,"Enter the radius");
```

```
        break;
    case 1:
        m_dR = sqrt((g_dCurX - m_dCenterX) * (g_dCurX - m_dCenterX)
              + (g_dCurY - m_dCenterY) * (g_dCurY - m_dCenterY));
        if(g_nCurTask = = ID_DRAW_CIRCLE)
        {
            g_nStep = 0;
            pStatus - > SetPaneText(0,"Enter the center");
            ret = TRUE;
        }
        else if(g_nCurTask = = ID_DRAW_FILLED_CIRCLE)
        {
            CFillDlg    dlg;
            dlg.DoModal();
            m_nFillPattern = dlg.m_nCurPattern;

            g_nStep = 0;
            pStatus - > SetPaneText(0,"Enter the center");
            ret = TRUE;
        }
        else
        {
            g_nStep++;
            pStatus - > SetPaneText(0,"Enter the start point");
        }
        break;
    case 2:
        {
        double x = g_dCurX - m_dCenterX;
        double y = g_dCurY - m_dCenterY;
        double r = sqrt(x * x + y * y);
        m_dSx = m_dR * x/r + m_dCenterX;
        m_dSy = m_dR * y/r + m_dCenterY;
        pStatus - > SetPaneText(0,"Enter the end point");
        g_nStep++;
        break;
        }
    case 3:
        {
        double x = g_dCurX - m_dCenterX;
        double y = g_dCurY - m_dCenterY;
        double r = sqrt(x * x + y * y);
        m_dEx = m_dR * x/r + m_dCenterX;
```

```
                    m_dEy = m_dR * y/r + m_dCenterY;
                    pStatus -> SetPaneText(0,"Enter the center");
                    g_nStep = 0;
                    ret = TRUE;
                    break;
                    }
                default:
                    break;
                }
            }
            else
            {
                if(g_nStep! = 1) return FALSE;
                //设置 R2_NOT 的绘图方式
                int nDrawMode = pDC -> SetROP2(R2_NOT);

                CPoint      orgPoint,curPoint,prePoint;
                g_pCurView -> WorldToScreen(orgPoint,g_dOrgX,g_dOrgY);
                //覆盖先前的线
                double dR = sqrt((g_dPreX - m_dCenterX) * (g_dPreX - m_dCenterX)
                                + (g_dPreY - m_dCenterY) * (g_dPreY - m_
dCenterY));
                int         r = (int) g_pCurView -> m_dWorldToScreen * dR;
                pDC -> Arc(orgPoint.x - r,orgPoint.y - r,orgPoint.x + r,orgPoint.y +
r,0,0,0,0);

                //绘制新的线
                dR = sqrt((g_dCurX - m_dCenterX) * (g_dCurX - m_dCenterX)
                                + (g_dCurY - m_dCenterY) * (g_dCurY - m_
dCenterY));
                r = (int) dR * g_pCurView -> m_dWorldToScreen;
                pDC -> Arc(orgPoint.x - r,orgPoint.y - r,orgPoint.x + r,orgPoint.y +
r,0,0,0,0);

                //设置先前的绘图方式
                pDC -> SetROP2(nDrawMode);
            }
            return ret;
    }

    void CCircle::Draw(CDC * pDC,UINT nColor,UINT nLineType)
    {
        CPen * pNewPen = new CPen;
```

```
    if(nColor = = 0&&nLineType = = 0)
        pNewPen - > CreatePen(m_nLineType,1,m_nColor);
    else
        pNewPen - > CreatePen(nLineType,1,nColor);
    CPen * pOldPen = pDC - > SelectObject( pNewPen );

    CPoint      cPnt,sPnt,ePnt;
    g_pCurView - > WorldToScreen(cPnt,m_dCenterX,m_dCenterY);
    g_pCurView - > WorldToScreen(sPnt,m_dSx,m_dSy);
    g_pCurView - > WorldToScreen(ePnt,m_dEx,m_dEy);
    int   r = (int) m_dR * g_pCurView - > m_dWorldToScreen;

    if(m_nType = = ID_DRAW_CIRCLE||m_nType = = ID_DRAW_ARC)
    {
        pDC - > Arc(cPnt.x - r,cPnt.y - r,cPnt.x + r,cPnt.y + r,
                sPnt.x,sPnt.y,ePnt.x,ePnt.y);
    }
    else
    {
        CBrush * pNewBrush = new CBrush;
        if(nColor = = 0&&nLineType = = 0)
            pNewBrush - > CreateHatchBrush(m_nFillPattern,m_nColor);
        else
            pNewBrush - > CreateHatchBrush(m_nFillPattern,nColor);
        CBrush * pOldBrush = pDC - > SelectObject(pNewBrush);
        pDC - > Pie(cPnt.x - r,cPnt.y - r,cPnt.x + r,cPnt.y + r,
                sPnt.x,sPnt.y,ePnt.x,ePnt.y);
        pDC - > SelectObject(pOldBrush);
        delete      pNewBrush;
    }
    pDC - > SelectObject( pOldPen );
}

BOOL CCircle::OSnap()
{
    double      d;
    double      sd = g_nOSnapSize * g_pCurView - > m_dScreenToWorld;
    double      x[7],y[7];
    int         cnt = 0;
    BOOL  ret = FALSE;

    switch(g_nOpqTask)
    {
```

```
        case ID_OSNAP_ENDPOINT:
            if(m_nType == ID_DRAW_ARC)
            {
                x[cnt] = m_dSx;
                y[cnt] = m_dSy;
                cnt++;
                x[cnt] = m_dEx;
                y[cnt] = m_dEy;
                cnt++;
            }
            break;
        case ID_OSNAP_CENTER:
            x[cnt] = m_dCenterX;
            y[cnt] = m_dCenterY;
            cnt++;
            break;
        case ID_OSNAP_QUARDANT:
            if(m_nType == ID_DRAW_CIRCLE||m_nType == ID_DRAW_FILLED_
CIRCLE)
            {
                x[cnt] = m_dCenterX + m_dR;
                y[cnt] = m_dCenterY;
                cnt++;
                x[cnt] = m_dCenterX - m_dR;
                y[cnt] = m_dCenterY;
                cnt++;
                x[cnt] = m_dCenterX;
                y[cnt] = m_dCenterY + m_dR;
                cnt++;
                x[cnt] = m_dCenterX;
                y[cnt] = m_dCenterY - m_dR;
                cnt++;
            }
            break;
        default:
            if((g_nOSnapType&1) == 1&&m_nType == ID_DRAW_ARC)
            {
                x[cnt] = m_dSx;
                y[cnt] = m_dSy;
                cnt++;
                x[cnt] = m_dEx;
                y[cnt] = m_dEy;
                cnt++;
```

```
            }
            if((g_nOSnapType&4) = = 4)
            {
                x[cnt] = m_dCenterX;
                y[cnt] = m_dCenterY;
                cnt ++ ;
            }
            if((g_nOSnapType&8) = = 8&&(m_nType = = ID_DRAW_CIRCLE||m_nType =
= ID_DRAW_FILLED_CIRCLE))
            {
                x[cnt] = m_dCenterX + m_dR;
                y[cnt] = m_dCenterY;
                cnt ++ ;
                x[cnt] = m_dCenterX - m_dR;
                y[cnt] = m_dCenterY;
                cnt ++ ;
                x[cnt] = m_dCenterX;
                y[cnt] = m_dCenterY + m_dR;
                cnt ++ ;
                x[cnt] = m_dCenterX;
                y[cnt] = m_dCenterY - m_dR;
                cnt ++ ;
            }
            break;
        }

        for(int i = 0;i < cnt;i ++ )
        {
            d = GetTwoPntDis2(g_dCurX,g_dCurY,x[i],y[i]);
            if(d < sd * sd)
            {
                g_dCurX = x[i];
                g_dCurY = y[i];
                ret = TRUE;
                break;
            }
        }

        return ret;
    }

    BOOL CCircle::Pick()
    {
```

```
    double sd = 5.0 * g_pCurView -> m_dScreenToWorld;
    double d = GetTwoPntDis2(g_dCurX,g_dCurY,m_dCenterX,m_dCenterY);
    BOOL   ret = FALSE;

    if(fabs(sqrt(d) - m_dR) < sd)
    {
        ret = TRUE;
    }

    if(ret = - TRUE&&m_nType = = ID_DRAW_ARC)
    {
        ret = FALSE;
    }

    return ret;
}
```

13. 矩形和填充矩形定义文件 Rectangle.h 和 Rectangle.cpp

Rectangle.h 和 Rectangle.cpp 文件中含有关于矩形和填充矩形的定义。这两个文件中含有用于实现矩形和填充矩形创建、绘制拾取以及对象捕捉的函数。这两个文件定义的 CRectangle 类处于该应用程序层次框架结构中的第三层。

**程序清单 12.25 矩形和填充矩形定义头文件 Rectangle.h**

```
// Rectangle.h: interface for the CRectangle class.
//
//////////////////////////////////////////////////////////////////////

#if ! defined(AFX_RECTANGLE_H__29C59BE2_B347_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
#define AFX_RECTANGLE_H__29C59BE2_B347_11CF_BF57_B29AF85F4C3B__
INCLUDED_

#if _MSC_VER > = 1000
#pragma once
#endif // _MSC_VER > = 1000

class CRectangle:public CEntity
{
public:
    BOOL Pick();
    BOOL OSnap();
    BOOL Create(CDC * pDC,BOOL bSure);
    void Draw(CDC * pDC,UINT nColor = 0,UINT nLineType = 0);
```

```
        CRectangle();
        CRectangle(double dLeft,double dTop,double dRight,
                   double dBottom,UINT nFillPattern = 0);
        virtual ~CRectangle();

        double      m_dLeft,m_dTop;
        double      m_dRight,m_dBottom;
        UINT  m_nFillPattern;
    };

    #endif // ! defined(AFX_RECTANGLE_H__29C59BE2_B347_11CF_BF57_B29AF85F4C3B
__INCLUDED_)
```

**程序清单 12.26 矩形和填充矩形定义实现文件 Rectangle.cpp**

```
// Rectangle.cpp: implementation of the CRectangle class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "Cad.h"

#include "Entity.h"
#include "Rectangle.h"
#include "CadDoc.h"
#include "CadView.h"

#include "FillDlg.h"

#include <math.h>

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CRectangle::CRectangle()
{
      m_dLeft = m_dTop = 0;
```

```
        m_dRight = m_dBottom = 0;

        m_nType = ID_DRAW_RECTANGLE;
        m_nColor = RGB(0,0,0);
        m_nLineType = PS_SOLID;
    }

    CRectangle::CRectangle(double dLeft,double dTop,double dRight,
                    double dBottom,UINT nFillPattern)
    {
        m_dLeft = dLeft;
        m_dTop = dTop;
        m_dRight = dRight;
        m_dBottom = dBottom;
        m_nFillPattern = nFillPattern;

        m_rType = g_nCurTask;
        m_rColor = g_nCurColor;
        m_rLineType = g_nCurLineType;
    }

    CRectangle::~CRectangle()
    {

    }

    BOOL CRectangle::Create(CDC * pDC,BOOL bSure)
    {
        BOOL  ret = FALSE;

        if(bSure = = TRUE)
        {
            CStatusBar * pStatus = (CStatusBar * )
                AfxGetApp() - > m_pMainWnd - > GetDescendantWindow(ID_VIEW_
STATUS_BAR);
            ASSERT(pStatus);

            switch(g_nStep)
            {
            case 0:
                m_dLeft = g_dCurX;
                m_dTop = g_dCurY;
                g_nStep + + ;
```

```
            pStatus->SetPaneText(0,"Enter the second point");
            break;
        case 1:
            m_dRight = g_dCurX;
            m_dBottom = g_dCurY;
            if(g_nCurTask == ID_DRAW_FILLED_RECTANGLE)
            {
                CFillDlg    dlg;
                dlg.DoModal();
                m_nFillPattern = dlg.m_nCurPattern;
            }

            g_nStep = 0;
            pStatus->SetPaneText(0,"Enter the first point");
            ret = TRUE;
            break;
        default:
            break;
        }
    }
    else
    {
        if(g_nStep!=1) return FALSE;
        //设置 R2_NOT 的绘图方式
        int nDrawMode = pDC->SetROP2(R2_NOT);

        CPoint      orgPoint,curPoint,prePoint;
        g_pCurView->WorldToScreen(orgPoint,g_dOrgX,g_dOrgY);
        g_pCurView->WorldToScreen(curPoint,g_dCurX,g_dCurY);
        g_pCurView->WorldToScreen(prePoint,g_dPreX,g_dPreY);
        //覆盖先前的线
        pDC->MoveTo(orgPoint);
        pDC->LineTo(orgPoint.x,prePoint.y);
        pDC->LineTo(prePoint);
        pDC->LineTo(prePoint.x,orgPoint.y);
        pDC->LineTo(orgPoint);
        //绘制新的线
        pDC->MoveTo(orgPoint);
        pDC->LineTo(orgPoint.x,curPoint.y);
        pDC->LineTo(curPoint);
        pDC->LineTo(curPoint.x,orgPoint.y);
        pDC->LineTo(orgPoint);
        //设置先前的绘图方式
```

```
            pDC->SetROP2(nDrawMode);
        }
        return ret;
}

void CRectangle::Draw(CDC* pDC,UINT nColor,UINT nLineType)
{
        CPen* pNewPen = new CPen;
        if(nColor==0&&nLineType==0)
            pNewPen->CreatePen(m_nLineType,1,m_nColor);
        else
            pNewPen->CreatePen(nLineType,1,nColor);
        CPen* pOldPen = pDC->SelectObject( pNewPen );

        CPoint      pt1,pt2;
        g_pCurView->WorldToScreen(pt1,m_dLeft,m_dTop);
        g_pCurView->WorldToScreen(pt2,m_dRight,m_dBottom);
        if(m_nType==ID_DRAW_RECTANGLE)
        {
            pDC->MoveTo(pt1);
            pDC->LineTo(pt1.x,pt2.y);
            pDC->LineTo(pt2);
            pDC->LineTo(pt2.x,pt1.y);
            pDC->LineTo(pt1);
        }
        else
        {
            CBrush* pNewBrush = new CBrush;
            if(nColor==0&&nLineType==0)
                pNewBrush->CreateHatchBrush(m_nFillPattern,m_nColor);
            else
                pNewBrush->CreateHatchBrush(m_nFillPattern,nColor);
            CBrush* pOldBrush = pDC->SelectObject(pNewBrush);
            pDC->Rectangle(pt1.x,pt1.y,pt2.x,pt2.y);
            pDC->SelectObject(pOldBrush);
            delete      pNewBrush;
        }

        pDC->SelectObject( pOldPen );
}

BOOL CRectangle::OSnap()
{
```

```
double      d;
double      sd=g_nOSnapSize*g_pCurView->m_dScreenToWorld;
double      x[8],y[8];
int         cnt=0;
BOOL  ret=FALSE;

switch(g_nOpqTask)
{
case ID_OSNAP_ENDPOINT:
      x[cnt]=m_dLeft;
      y[cnt]=m_dTop;
      cnt++;
      x[cnt]=m_dRight;
      y[cnt]=m_dBottom;
      cnt++;
      x[cnt]=m_dLeft;
      y[cnt]=m_dBottom;
      cnt++;
      x[cnt]=m_dRight;
      y[cnt]=m_dTop;
      cnt++;
      break;
case ID_OSNAP_MIDPOINT:
      x[cnt]=(m_dLeft+m_dRight)/2.0;
      y[cnt]=m_dTop;
      cnt++;
      x[cnt]=(m_dLeft+m_dRight)/2.0;
      y[cnt]=m_dBottom;
      cnt++;
      x[cnt]=m_dLeft;
      y[cnt]=(m_dTop+m_dBottom)/2.0;
      cnt++;
      x[cnt]=m_dRight;
      y[cnt]=(m_dTop+m_dBottom)/2.0;
      cnt++;
      break;
default:
      if((g_nOSnapType&1)==1)
      {
            x[cnt]=m_dLeft;
            y[cnt]=m_dTop;
            cnt++;
            x[cnt]=m_dRight;
```

```
            y[cnt] = m_dBottom;
            cnt++;
            x[cnt] = m_dLeft;
            y[cnt] = m_dBottom;
            cnt++;
            x[cnt] = m_dRight;
            y[cnt] = m_dTop;
            cnt++;
        }
        if((g_nOSnapType&2) == 2)
        {
            x[cnt] = (m_dLeft + m_dRight)/2.0;
            y[cnt] = m_dTop;
            cnt++;
            x[cnt] = (m_dLeft + m_dRight)/2.0;
            y[cnt] = m_dBottom;
            cnt++;
            x[cnt] = m_dLeft;
            y[cnt] = (m_dTop + m_dBottom)/2.0;
            cnt++;
            x[cnt] = m_dRight;
            y[cnt] = (m_dTop + m_dBottom)/2.0;
            cnt++;
        }
        break;
    }

    for(int i = 0;i < cnt;i++)
    {
        d = GetTwoPntDis2(g_dCurX,g_dCurY,x[i],y[i]);
        if(d < sd * sd)
        {
            g_dCurX = x[i];
            g_dCurY = y[i];
            ret = TRUE;
            break;
        }
    }

    return ret;
}

BOOL CRectangle::Pick()
```

```
{
    BOOL  ret = FALSE;
    double      sd = 5.0 * g_pCurView->m_dScreenToWorld;

    if(fabs(g_dCurX - m_dLeft) < sd||fabs(g_dCurX - m_dRight) < sd)
    {
        if((g_dCurY >= m_dTop&&g_dCurY <= m_dBottom)
            ||(g_dCurY <= m_dTop&&g_dCurY >= m_dBottom))
        {
            ret = TRUE;
        }
    }

    if(fabs(g_dCurY - m_dTop) < sd||fabs(g_dCurY - m_dBottom) < sd)
    {
        if((g_dCurX >= m_dLeft&&g_dCurX <= m_dRight)
            ||(g_dCurX <= m_dLeft&&g_dCurX >= m_dRight))
        {
            ret = TRUE;
        }
    }

    return ret;
}
```

14. 点定义文件 Point.h 和 Point.cpp 文件

Point.h 和 Point.cpp 文件中含有关于点的定义。这两个文件中含有用于实现点创建、绘制拾取以及对象捕捉的函数。这两个文件定义的 CCPnt 类处于该应用程序层次框架结构中的第三层。

**程序清单 12.27　点定义头文件 Point.h**

```
// Point.h: interface for the CPnt class.
//
//////////////////////////////////////////////////////////////////////

#if ! defined(AFX_POINT_H__F0C36702_B35A_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
#define AFX_POINT_H__F0C36702_B35A_11CF_BF57_B29AF85F4C3B__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
```

```
class CPnt : public CEntity
{
public:
	BOOL Pick();
	BOOL OSnap();
	BOOL Create(CDC * pDC,BOOL bSure);
	void Draw(CDC * pDC,UINT nColor = 0,UINT nLineType = 0);
	CPnt();
	CPnt(double dX,double dY);
	virtual ~CPnt();

	double m_dX,m_dY;
};

#endif // ! defined(AFX_POINT_H__F0C36702_B35A_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
```

**程序清单 12.28 点定义实现文件 Point.cpp**

```
// Point.cpp: implementation of the CPnt class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "Cad.h"

#include "Entity.h"
#include "Point.h"
#include "CadDoc.h"
#include "CadView.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CPnt::CPnt()
{
	m_dX = m_dY = 0;
```

```
        m_nType = ID_DRAW_POINT;
        m_nColor = RGB(0,0,0);
        m_nLineType = PS_SOLID;
    }

    CPnt::CPnt(double dX,double dY)
    {
        m_dX = dX;
        m_dY = dY;

        m_nType = ID_DRAW_POINT;
        m_nColor = g_nCurColor;
        m_nLineType = g_nCurLineType;
    }

    CPnt::~CPnt()
    {
    }

    BOOL CPnt::Create(CDC* pDC,BOOL bSure)
    {
        BOOL  ret = FALSE;

        if(bSure == TRUE)
        {
            CStatusBar* pStatus = (CStatusBar*)
                AfxGetApp()->m_pMainWnd->GetDescendantWindow(ID_VIEW_
STATUS_BAR);
            ASSERT(pStatus);

            switch(g_nStep)
            {
            case 0:
                m_dX = g_dCurX;
                m_dY = g_dCurY;

                g_nStep = 0;
                pStatus->SetPaneText(0,"Enter the point");
                ret = TRUE;
                break;
            default:
                break;
            }
```

```
    }

    return ret;
}

void CPnt::Draw(CDC * pDC,UINT nColor,UINT nLineType)
{
    CPen * pNewPen = new CPen;
    if(nColor = = 0&&nLineType = = 0)
        pNewPen - > CreatePen(m_nLineType,1,m_nColor);
    else
        pNewPen - > CreatePen(nLineType,1,nColor);
    CPen * pOldPen = pDC - > SelectObject( pNewPen );

    CPoint      point;
    g_pCurView - > WorldToScreen(point,m_dX,m_dY);
    pDC - > MoveTo(point.x - 2,point.y - 2);
    pDC - > LineTo(point.x + 2,point.y + 2);
    pDC - > MoveTo(point.x + 2,point.y - 2);
    pDC - > LineTo(point.x - 2,point.y + 2);

    pDC - > SelectObject( pOldPen );
}

BOOL CPnt::OSnap()
{
    double d = 0;
    double sd = g_nOSnapSize * g_pCurView - > m_dScreenToWorld;
    BOOL   ret = FALSE;

    switch(g_nOpqTask)
    {
    case ID_OSNAP_ENDPOINT:
    case ID_OSNAP_MIDPOINT:
        d = GetTwoPntDis2(g_dCurX,g_dCurY,m_dX,m_dY);
        if(d < sd * sd)
        {
            g_dCurX = m_dX;
            g_dCurY = m_dY;
            ret = TRUE;
        }
        break;
    default:
```

```
            if((g_nOSnapType&3)! = 0)
            {
                d = GetTwoPntDis2(g_dCurX,g_dCurY,m_dX,m_dY);
                if(d < sd * sd)
                {
                    g_dCurX = m_dX;
                    g_dCurY = m_dY;
                    ret = TRUE;
                }
            }
            break;
        }

        return ret;
    }

    BOOL CPnt::Pick()
    {
        double sd = 5.0 * 5.0 * g_pCurView - > m_dScreenToWorld;
        double d = GetTwoPntDis2(g_dCurX,g_dCurY,m_dX,m_dY);
        BOOL   ret = FALSE;

        if(d < sd)
        {
            ret = TRUE;
        }

        return ret;
    }
```

15.文本定义文件 Text.h 和 Text.cpp 文件

Text.h 和 Text.cpp 文件中含有关于文件的定义。这两个文件中含有用于实现文本创建、绘制拾取以及对象捕捉的函数。这两个文件定义的 CText 类处于该应用程序层次框架结构中的第三层。

**程序清单 12.29　文本定义头文件 Text.h**

```
// Text.h: interface for the CText class.
//
//////////////////////////////////////////////////////////////////////

#if ! defined(AFX_TEXT_H__F0C36703_B35A_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
#define AFX_TEXT_H__F0C36703_B35A_11CF_BF57_B29AF85F4C3B__INCLUDED_
```

```
#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CText : public CEntity
{
public:
    BOOL Pick();
    BOOL OSnap();
    BOOL Create(CDC * pDC,BOOL bSure);
    void Draw(CDC * pDC,UINT nColor = 0,UINT nLineType = 0);
    CText();
    CText(double dX,double dY,CString strText);
    virtual ~CText();

    double          m_dX,m_dY;
    CString         m_strText;
};

#endif // ! defined(AFX_TEXT_H__F0C36703_B35A_11CF_BF57_B29AF85F4C3E__
INCLUDED_)
```

**程序清单 12.30 文本定义实现文件 Text.cpp**

```
// Text.cpp: implementation of the CText class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "Cad.h"

#include "Entity.h"
#include "Text.h"
#include "CadDoc.h"
#include "CadView.h"

#include "EnterText.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#define new DEBUG_NEW
#endif
```

```
//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CText::CText()
{
	m_dX = m_dY = 0;
	m_strText = "";

	m_nType = ID_DRAW_TEXT;
	m_nColor = RGB(0,0,0);
	m_nLineType = PS_SOLID;
}

CText::CText(double dX,double dY,CString strText)
{
	m_dX = dX;
	m_dY = dY;
	m_strText = strText;

	m_nType = ID_DRAW_TEXT;
	m_nColor = g_nCurColor;
	m_nLineType = g_nCurLineType;
}

CText::~CText()
{

}

BOOL CText::Create(CDC * pDC,BOOL bSure)
{
	BOOL  ret = FALSE;

	if(bSure = = TRUE)
	{
		CStatusBar * pStatus = (CStatusBar * )
			AfxGetApp() - > m_pMainWnd - > GetDescendantWindow(ID_VIEW_
STATUS_BAR);
		ASSERT(pStatus);

		switch(g_nStep)
		{
```

```
            case 0:
                {
                m_dX = g_dCurX;
                m_dY = g_dCurY;

                CEnterTextDlg dlg;
                if(dlg.DoModal() == IDOK)
                {
                    m_strText = dlg.m_strText;
                    ret = TRUE;
                }
                g_nStep = 0;
                pStatus -> SetPaneText(0,"Enter the point");
                break;
                }
            default:
                break;
            }
        }

        return ret;
}

void CText::Draw(CDC * pDC,UINT nColor,UINT nLineType)
{
        CPoint      point;
        g_pCurView -> WorldToScreen(point,m_dX,m_dY);
        if(nColor == 0)
            pDC -> SetTextColor(m_nColor);
        else
            pDC -> SetTextColor(nColor);

        pDC -> TextOut(point.x,point.y,m_strText);
}

BOOL CText::OSnap()
{
        double d = 0;
        double sd = g_nOSnapSize * g_pCurView -> m_dScreenToWorld;
        BOOL   ret = FALSE;

        switch(g_nOpcTask)
        {
```

```
        case ID_OSNAP_ENDPOINT:
        case ID_OSNAP_MIDPOINT:
            d = GetTwoPntDis2(g_dCurX,g_dCurY,m_dX,m_dY);
            if(d< sd * sd)
            {
                g_dCurX = m_dX;
                g_dCurY = m_dY;
                ret = TRUE;
            }
            break;
        default:
            if((g_nOSnapType&3)! = 0)
            {
                d = GetTwoPntDis2(g_dCurX,g_dCurY,m_dX,m_dY);
                if(d < sd * sd)
                {
                    g_dCurX = m_dX;
                    g_dCurY = m_dY;
                    ret = TRUE;
                }
            }
            break;
        }

        return ret;
    }

    BOOL CText::Pick()
    {
        double sd = 5.0 * g_pCurView - > m_dScreenToWorld;
        double d = GetTwoPntDis2(g_dCurX,g_dCurY,m_dX,m_dY);
        BOOL   ret = FALSE;

        if(d< sd * sd)
        {
            ret = TRUE;
        }

        return ret;
    }
```

16. 填充对话框定义文件 FillDlg.h 和 FillDlg.cpp

FillDlg.h 和 FillDlg.cpp 中含有与填充对话框相关的定义。这两个文件从 Chap9 项目所

在文件夹中拷贝而来,未作任何修改。

**程序清单 12.31 填充对话框定义头文件 FillDlg.h**

```
#if !defined(AFX_FILLDLG_H__1C90D9E3_C2C3_11CF_BF56_B29AF85F4C3B__INCLUDED_)
#define AFX_FILLDLG_H__1C90D9E3_C2C3_11CF_BF56_B29AF85F4C3B__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// FillDlg.h : header file
//

class CPatternBtn : public CButton
{
// Construction
public:
	CPatternBtn();

// Attributes
public:
	int		m_nIndex;
// Operations
public:

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CPatternBtn)
	public:
	virtual void DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct);
	//}}AFX_VIRTUAL

// Implementation
public:
	virtual ~CPatternBtn();

	// Generated message map functions
protected:
	//{{AFX_MSG(CPatternBtn)
		// NOTE - the ClassWizard will add and remove member functions here.
	//}}AFX_MSG

	DECLARE_MESSAGE_MAP()
```

```
};

/////////////////////////////////////////////////////////////////////////////
//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

#endif // !defined(AFX_FILLDLG_H__1C90D9E3_C2C3_11CF_BF56_B29AF85F4C3B_
__INCLUDED_)

/////////////////////////////////////////////////////////////////////////////
// CFillDlg dialog

class CFillDlg : public CDialog
{
// Construction
public:
    CFillDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
    //{{AFX_DATA(CFillDlg)
    enum { IDD = IDD_DIALOG_FILL };
    CListBox    m_PatternList;
    //}}AFX_DATA


// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CFillDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Attribute:
public:
    int m_nCurPattern;

// Implementation
protected:
    CPatternBtn m_PatternBtn[6];
    void Update(int nIndex);
    void ExpandDialog(int nResourceID,BOOL bExpand);
    void EnableVisibleChildren();
```

```
    // Generated message map functions
    //{{AFX_MSG(CFillDlg)
    afx_msg void OnButtonMore();
    afx_msg void OnButtonPick();
    virtual BOOL OnInitDialog();
    afx_msg void OnSelchangeListPattern();
    afx_msg void OnPattern1();
    afx_msg void OnPattern2();
    afx_msg void OnPattern3();
    afx_msg void OnPattern4();
    afx_msg void OnPattern5();
    afx_msg void OnPattern6();
    afx_msg void OnPaint();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
// CPatternBtn window
```

**程序清单 12.32 填充对话框定义实现文件 FillDlg.cpp**

```
// FillDlg.cpp : implementation file
//

#include "stdafx.h"
#include "Cad.h"
#include "FillDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CFillDlg dialog


CFillDlg::CFillDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CFillDlg::IDD, pParent)
{
    //{{AFX_DATA_INIT(CFillDlg)
```

```
    //}}AFX_DATA_INIT
}


void CFillDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);

    DDX_Control(pDX, IDC_PATTERN1, m_PatternBtn[0]);
    DDX_Control(pDX, IDC_PATTERN2, m_PatternBtn[1]);
    DDX_Control(pDX, IDC_PATTERN3, m_PatternBtn[2]);
    DDX_Control(pDX, IDC_PATTERN4, m_PatternBtn[3]);
    DDX_Control(pDX, IDC_PATTERN5, m_PatternBtn[4]);
    DDX_Control(pDX, IDC_PATTERN6, m_PatternBtn[5]);

    //{{AFX_DATA_MAP(CFillDlg)
    DDX_Control(pDX, IDC_LIST_PATTERN, m_PatternList);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CFillDlg, CDialog)
    //{{AFX_MSG_MAP(CFillDlg)
    ON_BN_CLICKED(IDC_BUTTON_MORE, OnButtonMore)
    ON_3N_CLICKED(IDC_BUTTON_PICK, OnButtonPick)
    ON_LBN_SELCHANGE(IDC_LIST_PATTERN, OnSelchangeListPattern)
    ON_BN_CLICKED(IDC_PATTERN1, OnPattern1)
    ON_BN_CLICKED(IDC_PATTERN2, OnPattern2)
    ON_BN_CLICKED(IDC_PATTERN3, OnPattern3)
    ON_BN_CLICKED(IDC_PATTERN4, OnPattern4)
    ON_BN_CLICKED(IDC_PATTERN5, OnPattern5)
    ON_BN_CLICKED(IDC_PATTERN6, OnPattern6)
    ON_WM_PAINT()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CFillDlg message handlers

void CFillDlg::OnButtonMore()
{
    static BOOL bExpand = TRUE;
```

```
    ExpandDialog(IDC_SLIDER,bExpand);
    bExpand = ! bExpand;
}

void CFillDlg::OnButtonPick()
{
    EndDialog(IDC_BUTTON_PICK);
}

void CFillDlg::ExpandDialog(int nResourceID, BOOL bExpand)
{
    static CRect    rcLarge;
    static CRect    rcSmall;
    CString             sExpand;

    if(rcLarge.IsRectNull())
    {
        CRect rcLandmark;
        CWnd * pWndLandmark = GetDlgItem(nResourceID);
        ASSERT(pWndLandmark);

        GetWindowRect(rcLarge);
        pWndLandmark - > GetWindowRect(rcLandmark);

        rcSmall = rcLarge;
        rcSmall.bottom = rcLandmark.top;
    }

    if(bExpand)
    {
        SetWindowPos(NULL,0,0,rcLarge.Width(),rcLarge.Height(),
            SWP_NOMOVE|SWP_NOZORDER);

        sExpand = "< < &Less";
        EnableVisibleChildren();
    }
    else
    {
        SetWindowPos(NULL,0,0,rcSmall.Width(),rcSmall.Height(),
            SWP_NOMOVE|SWP_NOZORDER);

        sExpand = "&More > >";
        EnableVisibleChildren();
```

```
    }

    SetDlgItemText(IDC_BUTTON_MORE,sExpand);
}

void CFillDlg::EnableVisibleChildren()
{
    CWnd * pWndCtrl = GetWindow(GW_CHILD);
    CRect rcTest;
    CRect rcControl;
    CRect rcShow;

    GetWindowRect(rcShow);

    while(pWndCtrl! = NULL)
    {
        pWndCtrl->GetWindowRect(rcControl);

        if(rcTest.IntersectRect(rcShow,rcControl))
            pWndCtrl->EnableWindow(TRUE);
        else
            pWndCtrl->EnableWindow(FALSE);

        pWndCtrl = pWndCtrl->GetWindow(GW_HWNDNEXT);
    }
}

BOOL CFillDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // TODO: Add extra initialization here
    //设置未扩展对话框
    ExpandDialog(IDC_SLIDER,FALSE);

    //设置列表框选项
    m_PatternList.AddString("BDiagonal");
    m_PatternList.AddString("Cross");
    m_PatternList.AddString("DiagCross");
    m_PatternList.AddString("FDiagonal");
    m_PatternList.AddString("Horizontal");
    m_PatternList.AddString("Vertical");
```

```
    //设置图像按钮序号
    for(int i = 0;i < 6;i ++ )
    {
        m_PatternBtn[i].m_nIndex = i;
    }

    //预先设置对话框项
    m_nCurPattern = HS_BDIAGONAL;
    Update(0);

    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}

void CFillDlg::Update(int nIndex)
{
    //获得该选项的文本
    char tbuf[40];
    m_PatternList.GetText(nIndex,tbuf);
    SetDlgItemText(IDC_EDIT_PATTERN,tbuf);

    //设置列表框项
    m_PatternList.SetCurSel(nIndex);

    //在选择填充方式框绘图
    CWnd * pWnd = GetDlgItem(IDC_SELECTION);
    CRect rect;
    pWnd -> GetClientRect(rect);

    int         nBrushPattern[6] =
                {
                    HS_BDIAGONAL,
                    HS_CROSS,
                    HS_DIAGCROSS,
                    HS_FDIAGONAL,
                    HS_HORIZONTAL,
                    HS_VERTICAL,
                };

    CDC *   pDC = pWnd -> GetDC();
    CBrush * pNewBrush = new CBrush;
    ASSERT(nIndex < 6);
    pNewBrush -> CreateHatchBrush(nBrushPattern[nIndex],RGB(0,0,0));
```

```
    CBrush * pOldBrush = pDC - > SelectObject(pNewBrush);
    pDC - > Rectangle(rect);
    pDC - > SelectObject(pOldBrush);
    delete      pNewBrush;

    m_nCurPattern = nBrushPattern[nIndex];
}

void CFillDlg::OnPaint()
{
    CPaintDC dc(this); // device context for painting

    //在选择填充方式框绘图
    CWnd * pWnd = GetDlgItem(IDC_SELECTION);
    CRect rect;
    pWnd - > GetClientRect(rect);

    CDC *   pDC = pWnd - > GetDC();
    CBrush * pNewBrush = new CBrush;
    pNewBrush - > CreateHatchBrush(m_nCurPattern,RGB(0,0,0));
    CBrush * pOldBrush = pDC - > SelectObject(pNewBrush);
    pDC - > Rectangle(rect);
    pDC - > SelectObject(pOldBrush);
    delete      pNewBrush;
}

void CFillDlg::OnSelchangeListPattern()
{
    //获得当前选项
    int index = m_PatternList.GetCurSel();
    ASSERT(index! = LB_ERR);

    Update(index);
}

void CFillDlg::OnPattern1()
{
    Update(0);
}

void CFillDlg::OnPattern2()
{
    Update(1);
```

```
}

void CFillDlg::OnPattern3()
{
    Update(2);
}

void CFillDlg::OnPattern4()
{
    Update(3);
}

void CFillDlg::OnPattern5()
{
    Update(4);
}

void CFillDlg::OnPattern6()
{
    Update(5);
}

/////////////////////////////////////////////////////////////////////////////
// CPatternBtn

CPatternBtn::CPatternBtn()
{
}

CPatternBtn::~CPatternBtn()
{
}


BEGIN_MESSAGE_MAP(CPatternBtn, CButton)
    //{{AFX_MSG_MAP(CPatternBtn)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPatternBtn message handlers
```

```
void CPatternBtn::DrawItem(LPDRAWITEMSTRUCT lpDrawItemStruct)
{
    CDC* pDC=CDC::FromHandle(lpDrawItemStruct->hDC);
    CRect rect(lpDrawItemStruct->rcItem);

    int        nBrushPattern[6]=
               {
                    HS_BDIAGONAL,
                    HS_CROSS,
                    HS_DIAGCROSS,
                    HS_FDIAGONAL,
                    HS_HORIZONTAL,
                    HS_VERTICAL,
               };

    CBrush*          pNewBrush=new CBrush;
    pNewBrush->CreateHatchBrush(nBrushPattern[m_nIndex],RGB(0,0,0));
    CBrush* pOldBrush=pDC->SelectObject(pNewBrush);
    pDC->Rectangle(rect);
    pDC->SelectObject(pOldBrush);
    delete     pNewBrush;
}
```

17.状态设置属性对话框定义文件 StatusDlg.h、StatusPage.h、StatusDlg.cpp 和 StatusPage.cpp

这四种文件中含有与形状设置对话框相关的含义。它们都从第九章的 Chap9 项目中拷贝而来。在这里对这几个文件做了适当修改,以便它能够与系统的状态设置变量相联系。

**程序清单 12.33　状态设置属性单头文件 StatusDlg.h**

```
// StatusDlg.h : header file
//
// This class defines custom modal property sheet
// CStatusDlg.

#ifndef __STATUSDLG_H__
#define __STATUSDLG_H__

#include "StatusPage.h"

/////////////////////////////////////////////////////////////////////////////
// CStatusDlg

class CStatusDlg : public CPropertySheet
```

```
{
    DECLARE_DYNAMIC(CStatusDlg)

// Construction
public:
    CStatusDlg(CWnd* pWndParent = NULL);

// Attributes
public:
    COrdinatePage m_Page1;
    CSetPage m_Page2;
    COSnapPage m_Page3;

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CStatusDlg)
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CStatusDlg();

// Generated message map functions
protected:
    //{{AFX_MSG(CStatusDlg)
        // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

#endif      // __STATUSDLG_H__
```

**程序清单 12.34 状态设置属性单实现文件 StatusDlg.cpp**

```
// StatusDlg1.cpp : implementation file
//

#include "stdafx.h"
#include "resource.h"
```

```
# include "StatusDlg.h"

# ifdef _DEBUG
# undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
# endif

/////////////////////////////////////////////////////////////////////////////
// CStatusDlg

IMPLEMENT_DYNAMIC(CStatusDlg, CPropertySheet)

CStatusDlg::CStatusDlg(CWnd* pWndParent)
     : CPropertySheet(IDS_PROPSHT_CAPTION, pWndParent)
{
     // Add all of the property pages here.  Note that
     // the order that they appear in here will be
     // the order they appear in on screen.  By default,
     // the first page of the set is the active one.
     // One way to make a different property page the
     // active one is to call SetActivePage().

     AddPage(&m_Page1);
     AddPage(&m_Page2);
     AddPage(&m_Page3);
}

CStatusDlg::~CStatusDlg()
{
}


BEGIN_MESSAGE_MAP(CStatusDlg, CPropertySheet)
     //{{AFX_MSG_MAP(CStatusDlg)
          // NOTE - the ClassWizard will add and remove mapping macros here.
     //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// CStatusDlg message handlers
```

**程序清单 12.35　状态设置属性页头文件 StatusPage.h**

```
// StatusPage.h : header file
```

```
//

#ifndef __STATUSPAGE_H__
#define __STATUSPAGE_H__

/////////////////////////////////////////////////////////////////////////////
// COrdinatePage dialog

class COrdinatePage : public CPropertyPage
{
    DECLARE_DYNCREATE(COrdinatePage)

// Construction
public:
    COrdinatePage();
    ~COrdinatePage();

// Dialog Data
    //{{AFX_DATA(COrdinatePage)
    enum { IDD = IDD_ORDINATEPAGE };
    double      m_dStoW;
    double      m_dWtoS;
    double      m_dOriginX;
    double      m_dOriginY;
    double      m_dOriginZ;
    //}}AFX_DATA


// Overrides
    // ClassWizard generate virtual function overrides
    //{{AFX_VIRTUAL(COrdinatePage)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(COrdinatePage)
    afx_msg void OnDeltaposSpinX(NMHDR* pNMHDR, LRESULT* pResult);
    virtual BOOL OnInitDialog();
    afx_msg void OnDeltaposSpinY(NMHDR* pNMHDR, LRESULT* pResult);
    afx_msg void OnDeltaposSpinZ(NMHDR* pNMHDR, LRESULT* pResult);
```

```
    afx_msg void OnKillfocusEditStow();
    afx_msg void OnKillfocusEditWtos();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()

};

/////////////////////////////////////////////////////////////////////////////
// CSetPage dialog

class CSetPage : public CPropertyPage
{
    DECLARE_DYNCREATE(CSetPage)

// Construction
public:
    CSetPage();
    ~CSetPage();

// Dialog Data
    //{{AFX_DATA(CSetPage)
    enum { IDD = IDD_SETPAGE };
    double      m_dGrid;
    double      m_dSnap;
    int         m_nGrid;
    int         m_nOrtho;
    int         m_nSnap;
    //}}AFX_DATA


// Overrides
    // ClassWizard generate virtual function overrides
    //{{AFX_VIRTUAL(CSetPage)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CSetPage)
        // NOTE: the ClassWizard will add member functions here
```

```
    //{{AFX_MSG
    DECLARE_MESSAGE_MAP()

};

/////////////////////////////////////////////////////////////////////////////
// COSnapPage dialog

class COSnapPage : public CPropertyPage
{
    DECLARE_DYNCREATE(COSnapPage)

// Construction
public:
    int m_nOSnapSize;
    COSnapPage();
    ~COSnapPage();

// Dialog Data
    //{{AFX_DATA(COSnapPage)
    enum { IDD = IDD_OSNAPPAGE };
    BOOL  m_bCenter;
    BOOL  m_bEndpoint;
    BOOL  m_bMidpoint;
    BOOL  m_bQuardant;
    //}}AFX_DATA


// Overrides
    // ClassWizard generate virtual function overrides
    //{{AFX_VIRTUAL(COSnapPage)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(COSnapPage)
    afx_msg void OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar);
    virtual BOOL OnInitDialog();
    //}}AFX_MSG
```

```
    DECLARE_MESSAGE_MAP()

};


#endif // __STATUSPAGE_H__
```

**程序清单 12.36　状态设置属性页实现文件 StatusPage.cpp**

```
// StatusPage1.cpp : implementation file
//

#include "stdafx.h"
#include "resource.h"
#include "StatusPage.h"

#include <math.h>

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

IMPLEMENT_DYNCREATE(COrdinatePage, CPropertyPage)
IMPLEMENT_DYNCREATE(CSetPage, CPropertyPage)
IMPLEMENT_DYNCREATE(COSnapPage, CPropertyPage)


/////////////////////////////////////////////////////////////////////////////
// COrdinatePage property page

COrdinatePage::COrdinatePage() : CPropertyPage(COrdinatePage::IDD)
{
    //{{AFX_DATA_INIT(COrdinatePage)
    m_dStoW = 1.0;
    m_dWtoS = 1.0;
    m_dOriginX = 0.0;
    m_dOriginY = 0.0;
    m_dOriginZ = 0.0;
    //}}AFX_DATA_INIT
}

COrdinatePage::~COrdinatePage()
```

```
}
}

void COrdinatePage::DoDataExchange(CDataExchange * pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(COrdinatePage)
    DDX_Text(pDX, IDC_EDIT_STOW, m_dStoW);
    DDX_Text(pDX, IDC_EDIT_WTOS, m_dWtoS);
    DDX_Text(pDX, IDC_EDIT_X, m_dOriginX);
    DDX_Text(pDX, IDC_EDIT_Y, m_dOriginY);
    DDX_Text(pDX, IDC_EDIT_Z, m_dOriginZ);
    //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(COrdinatePage, CPropertyPage)
    //{{AFX_MSG_MAP(COrdinatePage)
    ON_NOTIFY(UDN_DELTAPOS, IDC_SPIN_X, OnDeltaposSpinX)
    ON_NOTIFY(UDN_DELTAPOS, IDC_SPIN_Y, OnDeltaposSpinY)
    ON_NOTIFY(UDN_DELTAPOS, IDC_SPIN_Z, OnDeltaposSpinZ)
    ON_EN_KILLFOCUS(IDC_EDIT_STOW, OnKillfocusEditStow)
    ON_EN_KILLFOCUS(IDC_EDIT_WTOS, OnKillfocusEditWtos)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()


////////////////////////////////////////////////////////////////////////////
// CSetPage property page

CSetPage::CSetPage() : CPropertyPage(CSetPage::IDD)
{
    //{{AFX_DATA_INIT(CSetPage)
    m_dGrid = 0.5;
    m_dSnap = 0.5;
    m_nGrid = 1;
    m_nOrtho = 1;
    m_nSnap = 1;
    //}}AFX_DATA_INIT
}

CSetPage::~CSetPage()
{
```

```
}

void CSetPage::DoDataExchange(CDataExchange* pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CSetPage)
    DDX_Text(pDX, IDC_EDIT_GRID, m_dGrid);
    DDX_Text(pDX, IDC_EDIT_SNAP, m_dSnap);
    DDX_Radio(pDX, IDC_GRID_ON, m_nGrid);
    DDX_Radio(pDX, IDC_ORTHO_ON, m_nOrtho);
    DDX_Radio(pDX, IDC_SNAP_ON, m_nSnap);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CSetPage, CPropertyPage)
    //{{AFX_MSG_MAP(CSetPage)
        // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// COSnapPage property page

COSnapPage::COSnapPage() : CPropertyPage(COSnapPage::IDD)
{
    m_nOSnapSize = 4;
    //{{AFX_DATA_INIT(COSnapPage)
    m_bCenter = FALSE;
    m_bEndpoint = FALSE;
    m_bMidpoint = FALSE;
    m_bQuardant = FALSE;
    //}}AFX_DATA_INIT
}

COSnapPage::~COSnapPage()
{
}

void COSnapPage::DoDataExchange(CDataExchange* pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(COSnapPage)
    DDX_Check(pDX, IDC_CENTER, m_bCenter);
```

```
    DDX_Check(pDX, IDC_ENDPOINT, m_bEndpoint);
    DDX_Check(pDX, IDC_MIDPOINT, m_bMidpoint);
    DDX_Check(pDX, IDC_QUARDANT, m_bQuardant);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(COSnapPage, CPropertyPage)
    //{{AFX_MSG_MAP(COSnapPage)
    ON_WM_HSCROLL()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

BOOL COrdinatePage::OnInitDialog()
{
    CPropertyPage::OnInitDialog();

    CSpinButtonCtrl*  pSpin;
    pSpin=(CSpinButtonCtrl *)GetDlgItem(IDC_SPIN_X);
    pSpin->SetRange(-10000,10000);
    pSpin=(CSpinButtonCtrl *)GetDlgItem(IDC_SPIN_Y);
    pSpin->SetRange(-10000,10000);
    pSpin=(CSpinButtonCtrl *)GetDlgItem(IDC_SPIN_Z);
    pSpin->SetRange(-10000,10000);

    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}

void COrdinatePage::OnDeltaposSpinX(NMHDR* pNMHDR, LRESULT* pResult)
{
    NM_UPDOWN* pNMUpDown = (NM_UPDOWN*)pNMHDR;

    m_dOriginX+=pNMUpDown->iDelta;
    char tbuf[80];
    sprintf(tbuf,"%f",m_dOriginX);
    SetDlgItemText(IDC_EDIT_X,tbuf);

    *pResult = 0;
}

void COrdinatePage::OnDeltaposSpinY(NMHDR* pNMHDR, LRESULT* pResult)
{
    NM_UPDOWN* pNMUpDown = (NM_UPDOWN*)pNMHDR;
```

```
    m_dOriginY += pNMUpDown->iDelta;
    char tbuf[80];
    sprintf(tbuf,"%f",m_dOriginY);
    SetDlgItemText(IDC_EDIT_Y,tbuf);

    *pResult = 0;
}

void COrdinatePage::OnDeltaposSpinZ(NMHDR* pNMHDR, LRESULT* pResult)
{
    NM_UPDOWN* pNMUpDown = (NM_UPDOWN*)pNMHDR;

    m_dOriginZ += pNMUpDown->iDelta;
    char tbuf[80];
    sprintf(tbuf,"%f",m_dOriginZ);
    SetDlgItemText(IDC_EDIT_Z,tbuf);

    *pResult = 0;
}

void COrdinatePage::OnKillfocusEditStow()
{
    UpdateData();
    if(fabs(m_dStoW)<1e-16)
    {
        AfxMessageBox("The scale of screen to world should not be a zero!");
        GetDlgItem(IDC_EDIT_STOW)->SetFocus();
    }
    else
    {
        m_dWtoS=1.0/m_dStoW;
        char tbuf[80];
        sprintf(tbuf,"%f",m_dWtoS);
        SetDlgItemText(IDC_EDIT_WTOS,tbuf);
    }
}

void COrdinatePage::OnKillfocusEditWtos()
{
    UpdateData();
    if(fabs(m_dWtoS)<1e-16)
    {
```

```
            AfxMessageBox("The scale of world to screen should not be a zero!");
            GetDlgItem(IDC_EDIT_WTOS)->SetFocus();
        }
        else
        {
            m_dStoW = 1.0/m_dWtoS;
            char tbuf[80];
            sprintf(tbuf,"%f",m_dStoW);
            SetDlgItemText(IDC_EDIT_STOW,tbuf);
        }
    }

    BOOL COSnapPage::OnInitDialog()
    {
        CPropertyPage::OnInitDialog();

        //设置红色滚动条的范围与当前值
        CSliderCtrl* pSlider = (CSliderCtrl*) GetDlgItem(IDC_OSNAP_SIZE);
        pSlider->SetRange(2,20);
        pSlider->SetPos(m_nOSnapSize);

        return TRUE;  // return TRUE unless you set the focus to a control
                      // EXCEPTION: OCX Property Pages should return FALSE
    }

    void COSnapPage::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar* pScrollBar)
    {
        CSliderCtrl* pSlider = (CSliderCtrl*) GetDlgItem(IDC_OSNAP_SIZE);
        m_nOSnapSize = pSlider->GetPos();

        //绘制图片框
        CWnd* pWnd = GetDlgItem(IDC_PICTURE_OSNAP);
        CRect rect;
        pWnd->GetClientRect(rect);
        int nCenterX = (rect.left + rect.right)/2;
        int nCenterY = (rect.top + rect.bottom)/2;

        CDC*  pDC = pWnd->GetDC();

        //刷新图片框
        pDC->Rectangle(rect);

        //绘制十字叉丝线
```

```
        pDC->MoveTo(rect.left+5,nCenterY);
        pDC->LineTo(rect.right-5,nCenterY);
        pDC->MoveTo(nCenterX,rect.top+5);
        pDC->LineTo(nCenterX,rect.bottom-5);

        //绘制十字框
        pDC->Rectangle(nCenterX-m_nOSnapSize,
                       nCenterY  m_nOSnapSize,
                       nCenterX+m_nOSnapSize,
                       nCenterY+m_nOSnapSize);
    }
```

18.文本输入对话框定义文件 EnterText.h 和 EnterText.cpp

这两个文件定义了输入文本时要使用的对话框。

**程序清单 12.37　文本输入对话框定义头文件 EnterText.h**

```
#if ! defined(AFX_ENTERTEXT_H__F0C36704_B35A_11CF_BF57_B29AF85F4C3B__
INCLUDED_)
#define AFX_ENTERTEXT_H__F0C3670C4  B35A_11CF_BF57_B29AF85F4C3B__
INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000
// EnterText.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CEnterTextDlg dialog

class CEnterTextDlg : public CDialog
{
// Construction
public:
    CEnterTextDlg(CWnd* pParent = NULL);   // standard constructor

// Dialog Data
    //{{AFX_DATA(CEnterTextDlg)
    enum { IDD = IDD_ENTER_TEXT };
    CString     m_strText;
    //}}AFX_DATA

// Overrides
```

```
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CEnterTextDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        //}}AFX_VIRTUAL

    // Implementation
    protected:

        // Generated message map functions
        //{{AFX_MSG(CEnterTextDlg)
            // NOTE: the ClassWizard will add member functions here
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
    };

    //{{AFX_INSERT_LOCATION}}
    // Microsoft Developer Studio will insert additional declarations immediately
before the previous line.

    #endif // !defined(AFX_ENTERTEXT_H__F0C36704_B35A_11CF_BF57_B29AF85F4C3B
__INCLUDED_)
```

**程序清单 12.38 文本输入对话框定义实现文件 EnterText.cpp**

```
    // EnterText.cpp : implementation file
    //

    #include "stdafx.h"
    #include "Cad.h"
    #include "EnterText.h"

    #ifdef _DEBUG
    #define new DEBUG_NEW
    #undef THIS_FILE
    static char THIS_FILE[] = __FILE__;
    #endif

    /////////////////////////////////////////////////////////////////////////////
    // CEnterTextDlg dialog

    CEnterTextDlg::CEnterTextDlg(CWnd* pParent /* = NULL*/)
        : CDialog(CEnterTextDlg::IDD, pParent)
    {
```

```
    //{{AFX_DATA_INIT(CEnterTextDlg)
    m_strText = _T("");
    //}}AFX_DATA_INIT
}


void CEnterTextDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CEnterTextDlg)
    DDX_Text(pDX, IDC_EDIT_TEXT, m_strText);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CEnterTextDlg, CDialog)
    //{{AFX_MSG_MAP(CEnterTextDlg)
        // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CEnterTextDlg message handlers
```

## 12.6　本章小结

本章的简要介绍了 Windows 程序的编写思路。通过学习本章，读者可以对前面各章所讲的界面制作进行综合提高，以便进一步理解应用程序的界面是如何实现的。同时，读者也掌握了 Windows 编程的基本思路及其实现方法。

# 附录　二维图形的几何变换

说明：本附录只列出了第十二章中对实体进行坐标变换时所用到的公式。

坐标变换一般是指对图形的几何信息经过几何变换后产生新的图形。坐标变换既可以看作是坐标系不动而图形变动，变动后的图形在坐标系中的坐标值发生变化；也可以看作图形不动而坐标系变动，变动后，该图形在新的坐标系下具有新的坐标值。但这两种情况本质上是一样的。本节所讨论的几何变换属于后一种情况。对于线框图的变换，通常以点变换作为基础，把图形的一系列顶点作几何变换后，连接新的顶点系列即可产生新的图形。对于用参数方程描述的图形，可以通过参数方程作几何变换，实现对图形的变换。目前，我们讨论的基本上是图形拓扑关系不变的几何变换。若引进不同的几何变换算子，即可实现改变图形拓扑关系的几何变换，由此可以产生许多形状各异、更复杂的图形。利用了齐次坐标表示，我们可以方便地用变换矩阵实现对图形的变换。假设二维坐标变换前的一点坐标为[$x$ $y$ 1]，变换后为[$x^*$ $y^*$ 1]。

**1. 二维变换矩阵**

二维图形几何变换矩阵可用下式表示：

$$T_{2D}=\left[\begin{array}{cc:c} a & d & g \\ b & e & h \\ \hdashline c & f & i \end{array}\right]$$

从变换功能上可把 $T_{2D}$分为四个矩阵，其中$\begin{bmatrix} a & b \\ b & c \end{bmatrix}$是对图形进行缩放、旋转、对称、错切等变换；[$c$ $f$]是对图形进行平移变换；$\begin{bmatrix} g \\ h \end{bmatrix}$对图形作投影变换，$g$ 的作用是在 $x$ 轴的 $1/g$ 处产生一个灭点，$h$ 的作用是在 $y$ 轴的 $1/h$ 处产生一个灭点；[$i$]是对整体图形作伸缩变换。$T_{2D}$作为单位矩阵即定义二维空间中的直角坐标系，此时 $T_{2D}$可看作是三个行向量，其中[1 0 0]表示 $x$ 轴上的无穷远点，[0 1 0]表示 $y$ 轴上的无穷远点，[0 0 1]表示坐标原点。

**2. 平移变换**

$$[x^* \quad y^* \quad 1]=[x \quad y \quad 1]\cdot\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}=[x+T_x \quad y+T_y \quad 1]$$

**3. 比例变换**

$$[x^* \quad y^* \quad 1] = [x \quad y \quad 1] \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} = [S_x \cdot x \quad S_y \cdot y \quad 1]$$

(1)当 $S_x = S_y = 1$ 时,为恒等比例变换,即图形不变。

(2)当 $S_x = S_y > 1$ 时,图形沿两个坐标轴方向等比例放大。

(3)当 $S_x = S_y < 1$ 时,图形沿两个坐标轴方向等比例缩小。

(4)当 $S_x \neq S_y$ 时,图形沿两个坐标轴方向作非均匀的比例变换。

**4. 旋转变换**

$$[x^* \quad y^* \quad 1] = [x \quad y \quad 1] \cdot \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= [x \cdot \cos\theta - y \cdot \sin\theta \quad x \cdot \sin\theta + y \cdot \cos\theta \quad 1]$$

在 $xoy$ 平面上的二维图形绕原点顺时针旋转 $\theta$ 角,则变换矩阵为

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

比例、旋转变换是与参考点有关的。上面介绍的均是相对原点所作的比例、旋转变换。如果要相对某一个参考点$(x_f, y_f)$作比例、旋转变换,其变换的过程是先把坐标系原点平移至$(x_f, y_f)$。在新的坐标系下作比例或旋转变换后,再将坐标原点平移回去,其变换公式如下。

**5.相对$(x_f, y_f)$点的比例变换**

$$T_{sf} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_f & -y_f & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_f & y_f & 1 \end{bmatrix}$$

$$= \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ (1-s_x) \cdot x_f & (1-s_y) \cdot y_f & 1 \end{bmatrix}$$

**6.相对$(x_f, y_f)$点的旋转变换**

$$T_{rf} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_f & -y_f & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_f & y_f & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ (1-\cos\theta) \cdot x_f + y_f \cdot \sin\theta & (1-\cos\theta) \cdot y_f - x_f \cdot \sin\theta & 1 \end{bmatrix}$$

**7.相对过点($x_f$,$y_f$)与X轴正向夹角为θ的直线的对称变换**

$$\begin{bmatrix}1&0&0\\0&1&0\\-x_f&-y_f&1\end{bmatrix}\begin{bmatrix}\cos\theta&-\sin\theta&0\\\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix}\begin{bmatrix}1&0&0\\0&-1&0\\0&0&1\end{bmatrix}\begin{bmatrix}\cos\theta&\sin\theta&0\\-\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix}\begin{bmatrix}1&0&0\\0&1&0\\x_f&y_f&1\end{bmatrix}$$

$$=-\begin{bmatrix}1&0&0\\0&1&0\\-x_f&-y_f&1\end{bmatrix}\begin{bmatrix}\cos\theta&-\sin\theta&0\\\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix}\begin{bmatrix}\cos\theta&\sin\theta&0\\-\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix}\begin{bmatrix}1&0&0\\0&1&0\\x_f&y_f&1\end{bmatrix}$$

$$=\begin{bmatrix}\cos\theta&-\sin\theta&0\\\sin\theta&\cos\theta&0\\-\cos\theta\cdot x_f-y_f\cdot\sin\theta&x_f\cdot\sin\theta-\cos\theta\cdot y_f&1\end{bmatrix}\begin{bmatrix}\cos\theta&\sin\theta&0\\-\sin\theta&\cos\theta&0\\x_f&y_f&1\end{bmatrix}$$

$$=\begin{bmatrix}\cos\theta&-\sin\theta&0\\\sin\theta&\cos\theta&0\\-\cos\theta\cdot x_f-y_f\cdot\sin\theta&x_f\cdot\sin\theta-\cos\theta\cdot y_f&1\end{bmatrix}\begin{bmatrix}\cos\theta&\sin\theta&0\\\sin\theta&-\cos\theta&0\\x_f&y_f&1\end{bmatrix}$$

$$=\begin{bmatrix}\cos^2\theta-\sin^2\theta&2\sin\theta\cos\theta&0\\2\sin\theta\cos\theta&\sin^2\theta-\cos^2\theta&0\\x_f(\sin^2\theta-\cos^2\theta)-2y_f\sin\theta\cos\theta+x_f&y_f(\cos^2\theta-\sin^2\theta)-2x_f\sin\theta\cos\theta+y_f&1\end{bmatrix}$$

**8.几点说明**

(1)平移变换只改变图形的位置,不改变图形的大小和形状。

(2)旋转变换仍保持图形各部分间的线性关系和角度关系,变换后直线的长度不变。

(3)比例变换可改变图形的大小和形状。

(4)拓扑不变的几何变换不改变图形的连接关系和平行关系。

# 无忧书库书籍版权申明

无忧书库提供的所有书籍、资料版权归原作者和出版商所有。

本站提供下载的书籍仅供个人学习、参考之用，任何集体、个人不得用于商业用途。

请您在下载书籍 **24** 小时之后删除书籍，购买正版书籍！

本站书籍如有侵犯您的版权，请与我们联系，我们将尽快删除。联系信箱：**pcbook@51soft.com**。

无忧书库
**http://pcbook.51soft.com**
**2000** 年 **4** 月 **13** 日