4. 调用属性表的 DoModal 函数将属性表显示在屏幕上。

为了简化属性表的创建过程,大部分 MFC 编程人员在派生的属性表类中声明属性表页类实例。他们还编写属性表类的构造函数,由它调用 AddPage 将各页添加到属性表中。简单的属性表和它的页可以按照下面方式声明:

```
class CFirstPage : public CPropertyPage
{
public:
    CFirstPage () : CPropertyPage (IDD_FIRSTPAGE) {};
    // Declare CFirstPage's data members here.

protected:
    virtual void DoDataExchange (CDataExchange * );
};

class CSecondPage : public CPropertyPage
{
public:
    CSecondPage () : CPropertyPage (IDD_SECONDPAGE) {};
    // Declare CSecondPage's data members here.

protected:
    virtual void DoDataExchange (CDataExchange * );
};

class CMyPropertySheet : public CPropertySheet
{
public:
    CFirstPage m_firstPage;          // First page
    CSecondPage m_secondPage;        // Second page

    // Constructor adds the pages automatically.
    CMyPropertySheet (LPCTSTR pszCaption,
        CWnd * pParentWnd = NULL) :
        CPropertySheet (pszCaption, pParentWnd, 0)
    {
        AddPage (&m_firstPage);
        AddPage (&m_secondPage);
    }
};
```

在这个示例中,CFirstPage 代表属性表中的第一页,而 CSecondPage 则代表第二页。相关的对话框资源是 IDD_FIRSTPAGE 和 IDD_SECONDPAGE,它们都被页的类构造函数引用。有了这个基础,用两个简单语句就可以构造并显示出标题为 Properties 的模式属性表。

```
CMyPropertySheet ps (_T ("Properties"));
ps.DoModal ();
```

和 CDialog::DoModal 一样，如果用 OK 按钮清除属性表，则 CPropertySheet::DoModal 返回 IDOK，否则，返回 IDCANCEL。

由于属性表提供了 OK 和 Cancel 按钮，所以用作属性表页的对话框模板不应再包含这两种按钮。属性表还包含一个 Apply 按钮和可选择的 Help 按钮。属性表第一次出现时，Apply按钮被灰化，当属性表页通过参数 TRUE 调用它从 CPropertyPage 继承来的 SetModified 函数时，Apply 按钮恢复有效。一旦属性表中设置发生了变化，SetModified 就要被调用。例如：修改了编辑控件的正文，或单选按钮被单击。如果要捕获 Apply 按钮单击事件，派生的属性表类必须包含 ON_BN_CLICKED 处理程序。该按钮的 ID 为 ID_APPLY_NOW。该单击处理程序要通过参数 TRUE 调用 UpdateData，更新有效页的成员变量并将当前属性值传递给属性表的所有者。之后，该单击处理程序通过参数 FALSE 调用 SetModified(对每个属性表页只调用一次)，使 Apply 按钮灰化。

注意 Apply 按钮的 ON_BN_CLICKED 处理程序只为“有效属性表页”(即当前显示页)调用 UpdateData。这非常重要，因为属性表页直到被使用者激活时才真正建立起来。如果还没有单击属性表页标签，这时调用 UpdateData，MFC 会提示错误。当用户切换到另一页时，主结构为有效页调用 UpdateData。所以用户单击 Apply 按钮时，需要更新数据成员的页只是当前有效页。通过 CPropertySheet::GetActivePage 您可以得到指向有效页的指针。

利用 DDX 和 DDV 实现控件和属性表页数据成员间的数据传输，以及校验从控件中提取的数据不只很方便，而且还能把属性表处理中许多困难的工作交托给 MFC 去做。例如：属性表页第一次显示时，需要调用页的 OnInitDialog 函数。OnInitDialog 的默认实现会调用 UpdateData 将页中控件初始化。如果这时用户单击标签要激活另一页，则当前页的 OnKillActive 函数被调用，主结构调用 UpdateData 从控件中取出数据并核查。不久，刚被激活的页接收到 OnSetActive 通知，可能还有 OnInitDialog 通知。如果这时用户接着单击属性表的 OK 按钮，则当前页的 OnOK 处理程序被调用，主结构调用 UpdateData 从页中取出数据并校验。

主结构提供了几个关键虚函数的默认实现，而且由于这些虚函数确定了属性表的行为，所以属性表始终按固定的方式工作。通过覆盖页的 OnInitDialog、OnSetActive、OnKillActive、OnOK 和 OnCancel 函数并执行特有的处理过程，您可以自己定义属性表的操作，但是记着一定要调用基类中的等价函数，否则主结构不能工作。如果不用 DDX 和 DDV，则有必要对每个属性表页覆盖这五个函数，从而保证每页数据都得到适当的处理。DDX 和 DDV 通过把大部分工作交托给主结构简化了属性表的使用。

PropDemo 应用程序

图 8-11 所示的 PropDemo 应用程序和 DlgDemo1 和 DlgDemo2 相似，但是它使用属性表把配置信息展示给用户，而不是对话框。属性表中 Size 页包含的控件用来设置视图中椭圆的尺寸。Color 页包含的控件用来修改椭圆的颜色。因为该属性表是模式的，所以属性表显示时无法再激活主窗口。

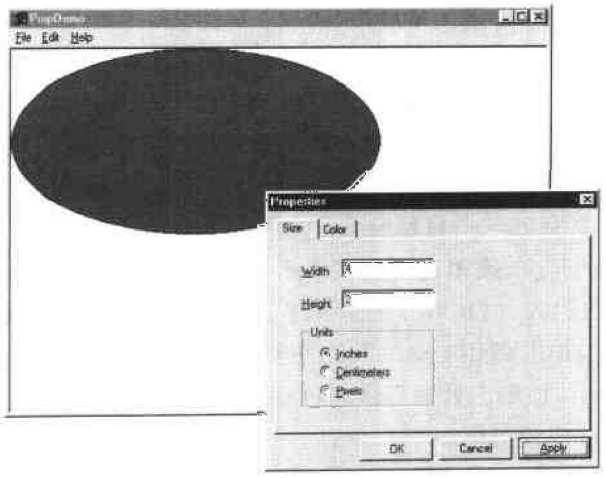

图 8-11 PropDemo 窗口和属性表

PropDemo 的部分资源代码请参见图 8-12。CMyPropertySheet 代表属性表自身,CSizePage 和 CColorPage 则分别代表属性表页。这三个类都是用 ClassWizard 派生来的。将 CSizePage 和 CColorPage 的实例 m_sizePage 和 m_colorPage 声明在 CMyPropertySheet 中,这样,属性表对象建立时,属性表页对象就自动被创建了。另外,还要把 m_sizePage 和 m_colorPage 声明为公用的,以便能在 CMyPropertySheet 外访问它们。

用户在 File 菜单中选择 Properties 命令时,CChildView::OnFileProperties 创建属性表。在栈上构造 CMyPropertySheet 对象后,OnFileProperties 将当前设置——宽、高、单位和颜色——复制到属性表页对象中相应的成员变量中:

```
CMyPropertySheet ps (_T ("Properties"));
ps.m_sizePage.m_nWidth = m_nWidth;
ps.m_sizePage.m_nHeight = m_nHeight;
ps.m_sizePage.m_nUnits = m_nUnits;
ps.m_colorPage.m_nColor = m_nColor;
```

然后 OnFileProperties 通过调用 DoModal 显示属性表。如果用 OK 键清除属性表,DoModal 则把属性表页的新设置信息复制出来,并调用 Invalidate 根据新设置重画视图:

```
if (ps.DoModal () == IDOK) {
    m_nWidth = ps.m_sizePage.m_nWidth;
    m_nHeight = ps.m_sizePage.m_nHeight;
    m_nUnits = ps.m_sizePage.m_nUnits;
    m_nColor = ps.m_colorPage.m_nColor;
    Invalidate ();
}
```

CSizePage 和 CColorPage 均把来自单选按钮的 ON_BN_CLICKED 通知和来自编辑控件的 EN_CHANGE 通知映射成同一个类成员函数 OnChange。OnChange 只包含一个语句：调用 SetModified 使属性表的 Apply 按钮有效。因此，如果 Apply 还处于灰化状态，在属性表页中单击任何按钮，或改变编辑控件正文，都会自动使 Apply 按钮有效。

单击 Apply 按钮时，CMyPropertySheet 的 OnApply 函数开始起作用。首先它对有效属性表页调用 UpdateData，把页控件中的用户输入复制到页的数据成员中。然后它再用各页数据成员中的属性设置将 ELLPROP 结构初始化，并向主窗口发送一个消息，其中包含该结构的地址信息。主窗口又将消息传递给视图。作为响应，视图把属性值复制到自己的数据成员中，并调用 Invalidate 实现重画动作。SendMessage 返回后，OnApply 通过调用各属性表页的 SetModified 函数，使 Apply 按钮无效。

MainFrm.h

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_MAINFRM_H__9CE2B4A8_9067_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_MAINFRM_H__9CE2B4A8_9067_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "ChildView.h"

class CMainFrame : public CFrameWnd
{

public:
    CMainFrame();
protected:
    DECLARE_DYNAMIC(CMainFrame)

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMainFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra,
        AFX_CMDHANDLERINFO* pHandlerInfo);
```

```
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
    CChildView    m_wndView;
// Generated message map functions
protected:
    //{{AFX_MSG(CMainFrame)
    afx_msg void OnSetFocus(CWnd *pOldWnd);
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    //}}AFX_MSG
    afx_msg LRESULT OnApply(WPARAM wParam, LPARAM lParam);
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.

#endif
// !defined(AFX_MAINFRM_H__9CE2B4A8_9067_11D2_8E53_006008A82731__INCLUDED_)
```

MainFrm.cpp

```
// MainFrm.cpp : implementation of the CMainFrame class
//

#include "stdafx.h"
#include "PropDemo.h"

#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNAMIC(CMainFrame, CFrameWnd)
```

```
BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
    ON_WM_SETFOCUS()
    ON_WM_CREATE()
    //}}AFX_MSG_MAP
    ON_MESSAGE(WM_USER_APPLY, OnApply)
END_MESSAGE_MAP()

// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
}

CMainFrame::~CMainFrame()
{
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    cs.dwExStyle &= ~WS_EX_CLIENTEDGE;
    cs.lpszClass = AfxRegisterWndClass(0);
    return TRUE;
}

// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}

#endif //_DEBUG

// CMainFrame message handlers
void CMainFrame::OnSetFocus(CWnd* pOldWnd)
{
    // forward focus to the view window
```

```
    m_wndView.SetFocus();
}
BOOL CMainFrame::OnCmdMsg(UINT nID, int nCode, void * pExtra,
    AFX_CMDHANDLERINFO * pHandlerInfo)
{
    // let the view have first crack at the command
    if (m_wndView.OnCmdMsg(nID, nCode, pExtra, pHandlerInfo))
        return TRUE;

    // otherwise, do default handling
    return CFrameWnd::OnCmdMsg(nID, nCode, pExtra, pHandlerInfo);
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    if (!m_wndView.Create(NULL, NULL, AFX_WS_DEFAULT_VIEW,
        CRect(0, 0, 0, 0), this, AFX_IDW_PANE_FIRST, NULL))
        return -1;

    return 0;
}

LRESULT CMainFrame::OnApply (WPARAM wParam, LPARAM lParam)
{
    m_wndView.SendMessage (WM_USER_APPLY, wParam, lParam);
    return 0;
}
```

ChildView.h

```
// ChildView.h : interface of the CChildView class
//
/////////////////////////////////////////////////////////////////////////////

#  if  ! defined ( AFX_CHILDVIEW_H__9CE2B4AA_9067_11D2_8E53_006008A82731__
INCLUDED_)
#define AFX_CHILDVIEW_H__9CE2B4AA_9067_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif //_MSC_VER > 1000
/////////////////////////////////////////////////////////////////////////////
// CChildView window

class CChildView : public CWnd
{
```

```
// Construction
public:
    CChildView();

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CChildView)
    protected:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CChildView();

    // Generated message map functions
protected:
    int m_nUnits;
    int m_nHeight;
    int m_nWidth;
    int m_nColor;
    //{{AFX_MSG(CChildView)
    afx_msg void OnPaint();
    afx_msg void OnFileProperties();
    //}}AFX_MSG
    afx_msg LRESULT OnApply (WPARAM wParam, LPARAM lParam);
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.

#endif
//!defined(AFX_CHILDVIEW_H__9CE2B4AA_9067_11D2_8E53_006008A82731__INCLUDED_)
```

ChildView.cpp

```
// ChildView.cpp : implementation of the CChildView class
//
```

```
#include "stdafx.h"
#include "PropDemo.h"
#include "ChildView.h"
#include "MyPropertySheet.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CChildView

CChildView::CChildView()
{
    m_nWidth = 4;
    m_nHeight = 2;
    m_nUnits = 0;
    m_nColor = 0;
}

CChildView::~CChildView()
{
}

BEGIN_MESSAGE_MAP(CChildView,CWnd)
    //{{AFX_MSG_MAP(CChildView)
    ON_WM_PAINT()
    ON_COMMAND(ID_FILE_PROPERTIES, OnFileProperties)
    //}}AFX_MSG_MAP
    ON_MESSAGE(WM_USER_APPLY, OnApply)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CChildView message handlers

BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)
{
    if (!CWnd::PreCreateWindow(cs))
        return FALSE;

    cs.dwExStyle|= WS_EX_CLIENTEDGE;
    cs.style &= ~WS_BORDER;
    cs.lpszClass = AfxRegisterWndClass(CS_HREDRAW|CS_VREDRAW|CS_DBLCLKS,
        ::LoadCursor(NULL, IDC_ARROW), HBRUSH(COLOR_WINDOW+1), NULL);

    return TRUE;
}
```

```
void CChildView::OnPaint()
{
    CPaintDC dc(this); // Device context for painting.

    CBrush brush (CColorPage::m_clrColors[m_nColor]);
    CBrush* pOldBrush = dc.SelectObject (&brush);

    switch (m_nUnits) {

    case 0: // Inches.
        dc.SetMapMode (MM_LOENGLISH);
        dc.Ellipse (0, 0, m_nWidth * 100, -m_nHeight * 100);
        break;

    case 1: // Centimeters.
        dc.SetMapMode (MM_LOMETRIC);
        dc.Ellipse (0, 0, m_nWidth * 100, -m_nHeight * 100);
        break;

    case 2: // Pixels.
        dc.SetMapMode (MM_TEXT);
        dc.Ellipse (0, 0, m_nWidth, m_nHeight);
        break;
    }
    dc.SelectObject (pOldBrush);
}

void CChildView::OnFileProperties()
{
    CMyPropertySheet ps (_T ("Properties"));
    ps.m_sizePage.m_nWidth = m_nWidth;
    ps.m_sizePage.m_nHeight = m_nHeight;
    ps.m_sizePage.m_nUnits = m_nUnits;
    ps.m_colorPage.m_nColor = m_nColor;

    if (ps.DoModal () == IDOK) {
        m_nWidth = ps.m_sizePage.m_nWidth;
        m_nHeight = ps.m_sizePage.m_nHeight;
        m_nUnits = ps.m_sizePage.m_nUnits;
        m_nColor = ps.m_colorPage.m_nColor;
        Invalidate ();
    }
}
LRESULT CChildView::OnApply (WPARAM wParam, LPARAM lParam)
{
    ELLPROP* pep = (ELLPROP*) lParam;
    m_nWidth = pep->nWidth;
    m_nHeight = pep->nHeight;
```

```
    m_nUnits = pep->nUnits;
    m_nColor = pep->nColor;
    Invalidate();
    return 0;
}
```

MyPropertySheet.h

```
#if !defined(AFX_MYPROPERTYSHEET_H__418271A3_90D4_11D2_8E53_006008A82731__
INCLUDED_)
#define AFX_MYPROPERTYSHEET_H__418271A3_90D4_11D2_8E53_006008A82731__INCLUDED_

#include "SizePage.h"    // Added by ClassView
#include "ColorPage.h"   // Added by ClassView
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// MyPropertySheet.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CMyPropertySheet

class CMyPropertySheet : public CPropertySheet
{
    DECLARE_DYNAMIC(CMyPropertySheet)

// Construction
public:
    CMyPropertySheet(UINT nIDCaption, CWnd* pParentWnd = NULL,
        UINT iSelectPage = 0);
    CMyPropertySheet(LPCTSTR pszCaption, CWnd* pParentWnd = NULL,
        UINT iSelectPage = 0);

// Attributes
public:
    CColorPage m_colorPage;
    CSizePage m_sizePage;

// Operations
public:
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMyPropertySheet)
    //}}AFX_VIRTUAL

// Implementation
public:
```

```
    virtual ~CMyPropertySheet();

    // Generated message map functions
protected:
    //{{AFX_MSG(CMyPropertySheet)
        // NOTE - the ClassWizard will add and remove
        // member functions here.
    //}}AFX_MSG
    afx_msg void OnApply ();
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.

#endif
// !defined(
//  AFX_MYPROPERTYSHEET_H__413271A3_90D4_11D2_8E53_006008A82731__INCLUDED_)
```

MyPropertySheet.cpp

```
// MyPropertySheet.cpp : implementation file
//

#include "stdafx.h"
#include "PropDemo.h"
#include "MyPropertySheet.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CMyPropertySheet

IMPLEMENT_DYNAMIC(CMyPropertySheet, CPropertySheet)

CMyPropertySheet::CMyPropertySheet(UINT nIDCaption, CWnd* pParentWnd,
    UINT iSelectPage) : CPropertySheet(nIDCaption, pParentWnd, iSelectPage)
{
    AddPage(&m_sizePage);
    AddPage(&m_colorPage);
}

CMyPropertySheet::CMyPropertySheet(LPCTSTR pszCaption, CWnd* pParentWnd,
```

```
    UINT iSelectPage) : CPropertySheet(pszCaption, pParentWnd, iSelectPage)
{
    AddPage (&m_sizePage);
    AddPage (&m_colorPage);
}

CMyPropertySheet::~CMyPropertySheet()
{
}

BEGIN_MESSAGE_MAP(CMyPropertySheet, CPropertySheet)
    //{{AFX_MSG_MAP(CMyPropertySheet)
        // NOTE - the ClassWizard will add and remove mapping macros here.
    //}}AFX_MSG_MAP
    ON_BN_CLICKED(ID_APPLY_NOW, OnApply)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMyPropertySheet message handlers

void CMyPropertySheet::OnApply ()
{
    GetActivePage()->UpdateData (TRUE);

    ELLPROP ep;
    ep.nWidth = m_sizePage.m_nWidth;
    ep.nHeight = m_sizePage.m_nHeight;
    ep.nUnits = m_sizePage.m_nUnits;
    ep.nColor = m_colorPage.m_nColor;

    GetParent()->SendMessage (WM_USER_APPLY, 0, (LPARAM) &ep);

    m_sizePage.SetModified (FALSE);
    m_colorPage.SetModified (FALSE);
}
```

SizePage.h

```
#if !defined(AFX_SIZEPAGE_H__418271A1_90D4_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_SIZEPAGE_H__418271A1_90D4_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// SizePage.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CSizePage dialog
```

```
class CSizePage : public CPropertyPage
{
    DECLARE_DYNCREATE(CSizePage)

// Construction
public:
    CSizePage();
    ~CSizePage();

// Dialog Data
    //{{AFX_DATA(CSizePage)
    enum { IDD = IDD_SIZE_PAGE };
    int        m_nWidth;
    int        m_nHeight;
    int        m_nUnits;
    //}}AFX_DATA

// Overrides
    // ClassWizard generate virtual function overrides
    //{{AFX_VIRTUAL(CSizePage)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CSizePage)
        // NOTE: the ClassWizard will add member functions here
    //}}AFX_MSG
    afx_msg void OnChange();
    DECLARE_MESSAGE_MAP()

};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.

#endif
//!defined(AFX_SIZEPAGE_H__418271A1_90D4_11D2_8F53_006008A82731__INCLUDED_)
```

SizePage.cpp

```
// SizePage.cpp : implementation file
//

#include "stdafx.h"
#include "PropDemo.h"
```

```
#include "SizePage.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSizePage property page

IMPLEMENT_DYNCREATE(CSizePage, CPropertyPage)

CSizePage::CSizePage() : CPropertyPage(CSizePage::IDD)
{
    //{{AFX_DATA_INIT(CSizePage)
    m_nWidth = 0;
    m_nHeight = 0;
    m_nUnits = -1;
    //}}AFX_DATA_INIT
}

CSizePage::~CSizePage()
{
}

void CSizePage::DoDataExchange(CDataExchange* pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CSizePage)
    DDX_Text(pDX, IDC_WIDTH, m_nWidth);
    DDV_MinMaxInt(pDX, m_nWidth, 1, 128);
    DDX_Text(pDX, IDC_HEIGHT, m_nHeight);
    DDV_MinMaxInt(pDX, m_nHeight, 1, 128);
    DDX_Radio(pDX, IDC_INCHES, m_nUnits);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CSizePage, CPropertyPage)
    //{{AFX_MSG_MAP(CSizePage)
        // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
    ON_EN_CHANGE(IDC_WIDTH, OnChange)
    ON_EN_CHANGE(IDC_HEIGHT, OnChange)
    ON_BN_CLICKED(IDC_INCHES, OnChange)
    ON_BN_CLICKED(IDC_CENTIMETERS, OnChange)
    ON_BN_CLICKED(IDC_PIXELS, OnChange)
END_MESSAGE_MAP()
```

```
/////////////////////////////////////////////////////////////////////////////
// CSizePage message handlers

void CSizePage::OnChange()
{
    SetModified(TRUE);
}
```

ColorPage.h

```
#if !defined(AFX_COLORPAGE_H__418271A2_90D4_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_COLORPAGE_H__418271A2_90D4_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// ColorPage.h : header file
//
/////////////////////////////////////////////////////////////////////////////
// CColorPage dialog

class CColorPage : public CPropertyPage
{
    DECLARE_DYNCREATE(CColorPage)

// Construction
public:
    CColorPage();
    ~CColorPage();
    static const COLORREF m_clrColors[3];

// Dialog Data
    //{{AFX_DATA(CColorPage)
    enum { IDD = IDD_COLOR_PAGE };
    int        m_nColor;
    //}}AFX_DATA

// Overrides
    // ClassWizard generate virtual function overrides
    //{{AFX_VIRTUAL(CColorPage)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CColorPage)
```

```
        // NOTE: the ClassWizard will add member functions here
    //{{AFX_MSG
    afx_msg void OnChange();
    DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.

#endif
//defined(AFX_COLORPAGE_H__418271A2_90D4_11D2_8E53_006008A82731__INCLUDED_)
```

ColorPage.cpp

```
// ColorPage.cpp : implementation file
//

#include "stdafx.h"
#include "PropDemo.h"
#include "ColorPage.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CColorPage property page

IMPLEMENT_DYNCREATE(CColorPage, CPropertyPage)

const COLORREF CColorPage::m_clrColors[3] = {
    RGB(255,   0,   0),    // Red
    RGB(  0, 255,   0),    // Green
    RGB(  0,   0, 255)     // Blue
};

CColorPage::CColorPage() : CPropertyPage(CColorPage::IDD)
{
    //{{AFX_DATA_INIT(CColorPage)
    m_nColor = -1;
    //}}AFX_DATA_INIT
}

CColorPage::~CColorPage()
{
```

```
}

void CColorPage::DoDataExchange(CDataExchange * pDX)
{
    CPropertyPage::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CColorPage)
    DDX_Radio(pDX, IDC_RED, m_nColor);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CColorPage, CPropertyPage)
    //{{AFX_MSG_MAP(CColorPage)
        // NOTE: the ClassWizard will add message map macros here
    //}}AFX_MSG_MAP
    ON_BN_CLICKED(IDC_RED, OnChange)
    ON_BN_CLICKED(IDC_GREEN, OnChange)
    ON_BN_CLICKED(IDC_BLUE, OnChange)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CColorPage message handlers

void CColorPage::OnChange()
{
    SetModified(TRUE);
}
```

图 8-12　PropDemo 应用程序

8.5　公用对话框

您会发现一些对话框经常出现在应用程序中。现在它们已经正当地成为操作系统的一部分了。在 Windows 3.1 之前，编程人员不得不自己编写 Open 和 Save As 对话框，以便在打开或保存文件之前从用户那儿得到文件名。因为设计和实现对话框都由编程人员来做，所以每个 Open 和 Save As 对话框都不一样，而且彼此差距很大。Windows 3.1 通过提供标准的对话框实现和其他 DLL 中常用的对话框（叫做"公用对话框库"）解决了这个长期以来困扰人们的难题。Windows 95 则增强了该库，它提供增强版的 Windows 3.1 公用对话框和用来输入页布局的新 Page Setup 对话框。Windows 98 和 Windows 2000 则进一步完善了公用对话框，使它们的功能更加强大了。

MFC 利用表 8-7 所列的类提供了公用对话框的 C++ 接口。

表 8-7 公用对话框类

类	对话框类型
CFileDialog	Open 和 Save As 对话框
CPrintDialog	Print 和 Print Setup 对话框
CPageSetupDialog	Page Setup 对话框
CFindReplaceDialog	Find 和 Replace 对话框
CColorDialog	Color 对话框
CFontDialog	Font 对话框

在 SDK 应用程序中,通过填充数据结构的域并调用一个 API 函数如::GetOpenFileName,就可以启动某个公用对话框。函数返回时,数据结构的某些域中包含用户输入的值。MFC 则简化了这个接口,它给大部分域提供了默认输入值,并提供成员函数从对话框中提取数据。在 MFC 应用程序中,打开文件前获取文件名通常很简单:

```
TCHAR szFilters[] =
   _T ("Text files (*.txt)|*.txt|All files (*.*)|*.*||");

CFileDialog dlg (TRUE, _T ("txt"), _T ("*.txt"),
    OFN_FILEMUSTEXIST|OFN_HIDEREADONLY, szFilters);

if (dlg.DoModal () == IDOK) {
    filename = dlg.GetPathName ();
    // Open the file and read it.
      .
      .
      .
}
```

传递给 CFileDialog 的构造函数的参数 TRUE 要求 MFC 显示一个 Open 对话框,而不是 Save As 对话框。参数 txt 和 *.txt 指定了文件的默认扩展名(如果用户不输入扩展名,该扩展名则自动添加在文件名后)和最初出现在对话框的 File Name 框中的正文。OFN 值是位标志,指定了对话框的属性。OFN_FILEMUSTEXIST 要求对话框检测用户输入的文件名,如果文件不存在,则拒绝文件名。而且 OFN_HIDEREADONLY 将默认方式下出现在对话框中的只读复选框隐藏起来。szFilters 指向一个字符串,该字符串指定用户的选择范围—某一文件类型。DoModal 返回时,可以用 CFileDialog::GetPathName 提取用户输入的文件名,以及完整的路径名。其他有用的 CFileDialog 函数包括 GetFileName,该函数提取文件名,但不提取路径名;还包括 GetFileTitle,它提取文件名,但不提取路径名和扩展名。

通常您会发现 MFC 的公用对话框类出奇地好用。部分原因是:您可以直接将公用对

话框类实例化，并避免派生自己的类。

8.5.1　修改公用对话框

您可以用多种方法修改 CFileDialog 的行为和其他公用对话框类。有一种简单的方法，它只要求改变传递给对话框构造函数的参数。例如：CFileDialog::CFileDialog 的第四个参数接受大约 24 个不同的位标志，这些位标志会影响对话框的外观和行为。这些标志的一个用处是用来创建 Open 对话框，该对话框有一个多重选择列表框，用户可以在其中选择多个文件，而不是仅仅一个。不能这样构造该对话框，

```
CFileDialog dlg (TRUE, _T ("txt"), _T ("*.txt"),
    OFN_FILEMUSTEXIST|OFN_HIDEREADONLY, szFilters);
```

而应该这样做：

```
CFileDialog dlg (TRUE, _T ("txt"),_T ("*.txt"),
    OFN_FILEMUSTEXIST|OFN_HIDEREADONLY|OFN_ALLOWMULTISELECT,
    szFilters);
```

DoModal 返回后，一组文件名被保存在由对话框对象的 m_ofn.lpstrFile 数据成员指称的缓冲区中。用 CFileDialog 的 GetStartPosition 和 GetNextPathName 函数可以轻松地从缓冲区获取文件名。

用 CFileDialog 构造对话框时，该类构造函数将用来定义对话框窗口标题、初始目录和其他参数的值填充在 OPENFILENAME 结构相应的域中。随后将结构的地址传递给::GetOpenFileName 或::GetSaveFileName。用来初始化结构的值中，一些来自于 CFileDialog 的构造函数的参数表，其他则是适用于大多数应用程序的默认值。另一种定义 Open 或 Save As 对话框的方法是：在构造对话框对象之后，但在调用 DoModal 之前修改 OPENFILENAME 结构的域值。通过公用数据成员 m_ofn 可以访问 OPENFILENAME 结构。

假定您想把多选项文件对话框的标题 Open 改成 Select File(s)，另外，还要求对话框关闭时选中的文件名过滤器在下次对话框打开时依旧处于选中状态，可以这样实现这些改变：

```
CFileDialog dlg (TRUE, _T ("txt"), NULL,
    OFN_FILEMUSTEXIST|OFN_ALLOWMULTISELECT,
    szFilters);

dlg.m_ofn.nFilterIndex = m_nFilterIndex;
static char szTitle[] = _T ("Select File(s)");
dlg.m_ofn.lpstrTitle = szTitle;
```

```
if (dlg.DoModal() == IDOK) {
    m_nFilterIndex = dlg.m_ofn.nFilterIndex;
    .
    .
    .
}
```

程序启动时,m_nFilterIndex 应设成 1。第一次创建对话框时,第一个选中的文件过滤器是默认值。当用户用 OK 按钮清除对话框时,当时选中的文件过滤器的索引值就从 OPENFILENAME 结构被复制出来并保存在 m_nFilterIndex 中。下次打开对话框时,该文件过滤器就会被自动选中。换句话说,对话框会记住用户选择的文件过滤器。如果要实现彻底封装,则可以让 m_nFilterIndex 成为对话框的一部分,而不是外部类的一个成员。首先从CFileDialog派生出自己的对话框类,然后将 m_nFilterIndex 声明为该类的静态成员变量,最后在首次构造 CMyFileDialog 对象之前把它初始化为 1。

通过从 CFileDialog 派生出自己的对话框类并覆盖几个关键的虚函数,您可以实现更多的修改。如果要自定义对话框的确认文件名方式,响应改变选中文件名方式,处理共享违规方式,除了 OnOK 和 OnCancel,还可以覆盖虚函数 OnFileNameOK、OnLBSelChangedNotify 和 OnShareViolation。如果要实现特殊任务,如增大对话框尺寸,添加或删除控件,您可以覆盖 OnInitDialog。(如果要覆盖 CFileDialog::OnInitDialog,一定要从自己编写的程序中调用基类版本。)例如:可以水平拉伸对话框,并创建一块预览域,显示当前选中文件的一小部分内容。通过覆盖 OnLBSelChangedNotify,可以在选项改变时更新预览窗口。

8.5.2 Phones 应用程序

Phones 是本章最后一个应用程序,它把本章和第 7 章介绍的许多概念都结合在一起。如图 8-13 所示,Phones 是一个简单的电话号码簿程序,它存储人名和电话号码。名字和电话号码的输入和编辑是在模式对话框中进行的。该对话框有一个标准编辑控件,用来输入和编辑名字;有一个数字编辑控件,用来输入和编辑电话号码;还有一个图标型按钮。输入应用程序的数据可以通过 File 菜单的 Open、Save 和 Save As 命令从磁盘读出和保存到磁盘上。Phones 用 CFileDialog 向用户询问文件名,用 CStdioFile 执行文件的输入和输出。它还用派生的列表框类 CPhonesListBox 作 CChildView 的基类,并用类中的消息反射使列表框响应它自己的鼠标双击通知。通过手工编辑 AppWizard 生成的 CChildView 类,我把原基类 CWnd 改换为 CPhonesListBox。相关的应用程序资源代码在图 8-14 中。

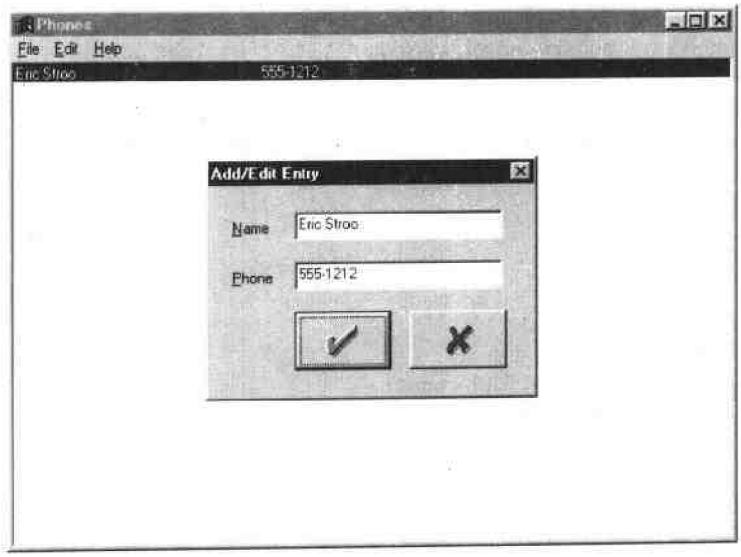

图 8-13　Phones 窗口和对话框

MainFrm.h

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_MAINFRM_H__7BE4B248_90ED_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_MAINFRM_H__7BE4B248_90ED_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#include "ChildView.h"

class CMainFrame : public CFrameWnd
{

public:
    CMainFrame();
protected:
    DECLARE_DYNAMIC(CMainFrame)
// Attributes
public:

// Operations
public:
```

```
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMainFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra,
        AFX_CMDHANDLERINFO* pHandlerInfo);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
    CChildView    m_wndView;

// Generated message map functions
protected:
    //{{AFX_MSG(CMainFrame)
    afx_msg void OnSetFocus(CWnd* pOldWnd);
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.

#endif
// !defined(AFX_MAINFRM_H__7BE4B248_90ED_11D2_8E53_006008A82731__INCLUDED_)
```

MainFrm.cpp

```
// MainFrm.cpp : implementation of the CMainFrame class
//
#include "stdafx.h"
#include "Phones.h"
#include "PhonesListBox.h"
#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
```

```
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNAMIC(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
    ON_WM_SETFOCUS()
    ON_WM_CREATE()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
}

CMainFrame::~CMainFrame()
{
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    cs.dwExStyle &= ~WS_EX_CLIENTEDGE;
    cs.lpszClass = AfxRegisterWndClass(0);
    return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics
#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
```

```
// CMainFrame message handlers
void CMainFrame::OnSetFocus(CWnd * pOldWnd)
{
    // forward focus to the view window
    m_wndView.SetFocus();
}

BOOL CMainFrame::OnCmdMsg(UINT nID, int nCode, void * pExtra,
    AFX_CMDHANDLERINFO * pHandlerInfo)
{
    // let the view have first crack at the command
    if (m_wndView.OnCmdMsg(nID, nCode, pExtra, pHandlerInfo))
        return TRUE;

    // otherwise, do default handling
    return CFrameWnd::OnCmdMsg(nID, nCode, pExtra, pHandlerInfo);
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    if (!m_wndView.Create(WS_CHILD|WS_VISIBLE|LBS_USETABSTOPS |
        LBS_SORT|LBS_NOTIFY|LBS_NOINTEGRALHEIGHT, CRect(0, 0, 0, 0),
        this, AFX_IDW_PANE_FIRST))
        return -1;

    return 0;
}
```

ChildView.h

```
// ChildView.h : interface of the CChildView class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_CHILDVIEW_H__7BE4B24A_90ED_11D2_8E53_006008A82731__
INCLUDED_)
#define AFX_CHILDVIEW_H__7BE4B24A_90ED_11D2_8E53_006008A82731__INCLUDED

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
// CChildView window

class CChildView : public CPhonesListBox
```

```
};
// Construction
public:
    CChildView();

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CChildView)
    protected:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CChildView();

    // Generated message map functions
protected:
    BOOL SaveFile(LPCTSTR pszFile);
    BOOL LoadFile(LPCTSTR pszFile);
    static const TCHAR m_szFilters[];
    CString m_strPathName;
    //{{AFX_MSG(CChildView)
    afx_msg void OnNewEntry();
    afx_msg void OnFileOpen();
    afx_msg void OnFileSave();
    afx_msg void OnFileSaveAs();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.

#endif
// !defined(
//    AFX_CHILDVIEW_H__7B84B24A_90ED_11D2_8E53_006008A82731__INCLUDED_)
```

ChildView.cpp

```
// ChildView.cpp : implementation of the CChildView class
//

#include "stdafx.h"
#include "Phones.h"
#include "PhonesListBox.h"
#include "ChildView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CChildView

const TCHAR CChildView::m_szFilters[] =
  _T ("Phone Files (*.phn)|*.phn|All Files (*.*)|*.*||");

CChildView::CChildView()
{
}
CChildView::~CChildView()
{
}

BEGIN_MESSAGE_MAP(CChildView, CPhonesListBox)
    //{{AFX_MSG_MAP(CChildView)
    ON_COMMAND(ID_FILE_NEW, OnNewEntry)
    ON_COMMAND(ID_FILE_OPEN, OnFileOpen)
    ON_COMMAND(ID_FILE_SAVE, OnFileSave)
    ON_COMMAND(ID_FILE_SAVE_AS, OnFileSaveAs)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CChildView message handlers

BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)
{
    if (!CPhonesListBox::PreCreateWindow(cs))
        return FALSE;

    cs.dwExStyle |= WS_EX_CLIENTEDGE;
    cs.style &= ~WS_BORDER;
```

```
    return TRUE;
}

void CChildView::OnNewEntry()
{
    NewEntry ();
}

void CChildView::OnFileOpen()
{
    CFileDialog dlg (TRUE,_T ("phn"),_T ("*.phn"),
        OFN_FILEMUSTEXIST|OFN_HIDEREADONLY, m_szFilters);

    if (dlg.DoModal () == IDOK) {
        if (LoadFile (dlg.GetPathName ())) {
            m_strPathName = dlg.GetPathName ();
            SetCurSel (0);
        }
    }
}

void CChildView::OnFileSave()
{
    if (!m_strPathName.IsEmpty ())
        SaveFile (m_strPathName);
    else // Need a file name first.
        OnFileSaveAs ();
}

void CChildView::OnFileSaveAs()
{
    CFileDialog dlg (FALSE,_T ("phn"), m_strPathName,
        OFN_OVERWRITEPROMPT|OFN_PATHMUSTEXIST|OFN_HIDEREADONLY,
        m_szFilters);

    if (dlg.DoModal () == IDOK)
        if (SaveFile (dlg.GetPathName ()))
            m_strPathName = dlg.GetPathName ();
}

BOOL CChildView::LoadFile(LPCTSTR pszFile)
{
    BOOL bResult = FALSE;

    try {
        CStdioFile file (pszFile, CFile::modeRead);
        ResetContent ();
        DWORD dwCount;
        file.Read (&dwCount, sizeof (dwCount));
```

```
        if (dwCount)
            for (int i = 0; i < (int) dwCount; i ++) {
                CString string;
                file.ReadString (string);
                AddString (string);
            }
        }
        bResult = TRUE;
    }
    catch (CFileException * e) {
        e -> ReportError ();
        e -> Delete ();
    }
    return bResult;
}

BOOL CChildView::SaveFile(LPCTSTR pszFile)
{
    BOOL bResult = FALSE;

    try {
        CStdioFile file (pszFile, CFile::modeWrite|CFile::modeCreate);
        DWORD dwCount = GetCount ();
        file.Write (&dwCount, sizeof (dwCount));
        if (dwCount) {
            for (int i = 0; i < (int) dwCount; i ++) {
                CString string;
                GetText (i, string);
                string += _T ("\n");
                file.WriteString (string);
            }
        }
        bResult = TRUE;
    }
    catch (CFileException * e) {
        e -> ReportError ();
        e -> Delete ();
    }
    return bResult;
}
```

PhonesListBox.h

```
#if !defined(AFX_PHONESLISTBOX_H__7BE4B250_90ED_11D2_8E53_006008A82731__
INCLUDED_)
#define AFX_PHONESLISTBOX_H__7BE4B250_90ED_11D2_8E53_006008A82731__INCLUDED_
```

```
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// PhonesListBox.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CPhonesListBox window

class CPhonesListBox : public CListBox
{
// Construction
public:
    CPhonesListBox();

// Attributes
public:

// Operations
public:
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CPhonesListBox)
    //}}AFX_VIRTUAL

// Implementation
public:
    void NewEntry();
    virtual ~CPhonesListBox();

    // Generated message map functions
protected:
    CFont m_font;
    //{{AFX_MSG(CPhonesListBox)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnEditItem();
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.

#endif
// !defined(
//    AFX_PHONESLISTBOX_H__7BE4B250_90ED_11D2_8E53_006008A82731__INCLUDED_)
```

PhonesListBox.cpp

```
// PhonesListBox.cpp : implementation file
//

#include "stdafx.h"
#include "Phones.h"
#include "PhonesListBox.h"
#include "PhoneEdit.h"
#include "EditDialog.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CPhonesListBox

CPhonesListBox::CPhonesListBox()
{
}

CPhonesListBox::~CPhonesListBox()
{
}

BEGIN_MESSAGE_MAP(CPhonesListBox, CListBox)
    //{{AFX_MSG_MAP(CPhonesListBox)
    ON_WM_CREATE()
    ON_CONTROL_REFLECT(LBN_DBLCLK, OnEditItem)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPhonesListBox message handlers

int CPhonesListBox::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CListBox::OnCreate(lpCreateStruct) == -1)
        return -1;

    m_font.CreatePointFont (80,_T ("MS Sans Serif"));
    SetFont (&m_font, FALSE);
    SetTabStops (128);
    return 0;
}
```

```
void CPhonesListBox::OnEditItem()
{
    CEditDialog dlg;

    CString strItem;
    int nIndex = GetCurSel ();
    GetText (nIndex, strItem);
    int nPos = strItem.Find (_T ('\t'));

    dlg.m_strName = strItem.Left (nPos);
    dlg.m_strPhone = strItem.Right (strItem.GetLength ()  nPos - 1);
    if (dlg.DoModal () == IDOK) {
        strItem = dlg.m_strName + _T ("\t") + dlg.m_strPhone;
        DeleteString (nIndex);
        AddString (strItem);
    }
    SetFocus ();
}

void CPhonesListBox::NewEntry()
{
    CEditDialog dlg;
    if (dlg.DoModal () == IDOK) {
        CString strItem = dlg.m_strName + _T ("\t") + dlg.m_strPhone;
        AddString (strItem);
    }
    SetFocus ();
}
```

EditDialog.h

```
#if !defined(AFX_EDITDIALOG_H__7BE4B252_90ED_11D2_8E53_006008A82731__
INCLUDED_)
#define AFX_EDITDIALOG_H__7BE4B252_90ED_11D2_8E53_006008A82731__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// EditDialog.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CEditDialog dialog

class CEditDialog : public CDialog
{
// Construction
public:
```

```
    CEditDialog(CWnd * pParent = NULL);   // standard constructor

// Dialog Data
    //{{AFX_DATA(CEditDialog)
    enum { IDD = IDD_EDITDLG };
    CButton       m_wndOK;
    CButton       m_wndCancel;
    CPhoneEdit    m_wndPhoneEdit;
    CString       m_strName;
    CString       m_strPhone;
    //}}AFX_DATA

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CEditDialog)
    protected:
    virtual void DoDataExchange(CDataExchange * pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
    //{{AFX_MSG(CEditDialog)
    virtual BOOL OnInitDialog();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.

#endif
// !defined(
//   AFX_EDITDIALOG_H__7BE4B252_90ED_11D2_8353_006008A82731__INCLUDED_)
```

EditDialog.cpp

```
// EditDialog.cpp : implementation file
//

#include "stdafx.h"
#include "Phones.h"
#include "PhoneEdit.h"
#include "EditDialog.h"

#ifdef _DEBUG
```

```
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
/////////////////////////////////////////////////////////////////////////////
// CEditDialog dialog

CEditDialog::CEditDialog(CWnd* pParent /* = NULL */)
    : CDialog(CEditDialog::IDD, pParent)
{
    //{{AFX_DATA_INIT(CEditDialog)
    m_strName = _T("");
    m_strPhone = _T("");
    //}}AFX_DATA_INIT
}

void CEditDialog::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CEditDialog)
    DDX_Control(pDX, IDOK, m_wndOK);
    DDX_Control(pDX, IDCANCEL, m_wndCancel);
    DDX_Control(pDX, IDC_PHONE, m_wndPhoneEdit);
    DDX_Text(pDX, IDC_NAME, m_strName);
    DDV_MaxChars(pDX, m_strName, 32);
    DDX_Text(pDX, IDC_PHONE, m_strPhone);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CEditDialog, CDialog)
    //{{AFX_MSG_MAP(CEditDialog)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CEditDialog message handlers

BOOL CEditDialog::OnInitDialog()
{
    CDialog::OnInitDialog();
    m_wndOK.SetIcon(AfxGetApp()->LoadIcon(IDI_OK));
    m_wndCancel.SetIcon(AfxGetApp()->LoadIcon(IDI_CANCEL));
    return TRUE;
}
```

PhoneEdit.h

```
#if !defined(AFX_PHONEEDIT_H__73E4B251_90ED_11D2_8E53_006008A82731__INCLUDED_)
#define AFX_PHONEEDIT_H__73E4B251_90ED_11D2_8E53_006008A82731__INCLUDED_
#if _MSC_VER > 1000
#pragma once
#endif //_MSC_VER > 1000
// PhoneEdit.h : header file
//

/////////////////////////////////////////////////////////////////////////////
// CPhoneEdit window

class CPhoneEdit : public CEdit
{
// Construction
public:
    CPhoneEdit();

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CPhoneEdit)
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CPhoneEdit();

    // Generated message map functions
protected:
    //{{AFX_MSG(CPhoneEdit)
    afx_msg void OnChar(UINT nChar, UINT nRepCnt, UINT nFlags);
    //}}AFX_MSG

    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
```

```
#endif
//!defined(AFX_PHONEEDIT_H__7BE4B251_90ED_11D2_8E53_006008A82731__INCLUDED_)
```

PhoneEdit.cpp

```
// PhoneEdit.cpp : implementation file
//

#include "stdafx.h"
#include "Phones.h"
#include "PhoneEdit.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CPhoneEdit

CPhoneEdit::CPhoneEdit()
{
}

CPhoneEdit::~CPhoneEdit()
{
}

BEGIN_MESSAGE_MAP(CPhoneEdit, CEdit)
    //{{AFX_MSG_MAP(CPhoneEdit)
    ON_WM_CHAR()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPhoneEdit message handlers

void CPhoneEdit::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    if (((nChar >= _T('0')) && (nChar <= _T('9'))) ||
        (nChar == VK_BACK)||(nChar == _T('('))||(nChar == _T(')')) ||
        (nChar == _T('-'))||(nChar == _T(' ')))

        CEdit::OnChar(nChar, nRepCnt, nFlags);
}
```

图 8-14　Phones 应用程序

Phones 资源代码中有一个最细微但很重要的成分,那就是 EditDialog.cpp 中这个看似简单的语句

```
DDX_Control (pDX, IDC_PHONE, m_wndPhoneEdit);
```

m_wndPhoneEdit 是 CEdit 的派生类 CPhoneEdit 的实例,它代表滤除非数字字符的编辑控件。但是 m_wndPhoneEdit 联系着 IDC_PHONE——一个由对话框模板创建的普通编辑控件。然而 IDC_PHONE 行为方式同 CPhoneEdit 相近,和 CEdit 不同,其原因就在于 DDX_Control 将控件归为子类,并把送往控件的消息导向 m_wndPhoneEdit 的消息映射表。其中的意味既简单又深远。如果您想让对话框控件像派生控件类的实例那样动作,只要用 DDX_Control 将该控件映射成一个类的实例即可。否则,在派生类中建立的任何特殊行为都会作废。

Phones 恰当地展示了 MFC 的 CFileDialog 类是如何使用的,以及文档是如何在磁盘上读、写的。但是它没有很好的数据保护功能。如果一组名字和电话号码中包含没有保存的变化,而这时又加载了另一组名字和电话号码,或干脆应用程序被关闭了,Phones 不会提示您保存变化。一个真正的应用程序则不能这样。Phones 还有其他缺点,比如:它不能将文件扩展名注册到操作系统中,双击鼠标不能打开被保存的文件。但是不要失望,在文档/视图体系结构中执行文档处理却是极其利落的,而且在下一章我们就开始编写文档/视图应用程序了。具体编写该应用程序时,MFC 会处理建立 Windows 应用程序所需的许多工作,如注册文件名扩展名,提示用户保存变化等。如果以前您从未编写过文档/视图应用程序,您会对主结构提供的支持程度感到惊喜的。

第Ⅱ部分

文档/视图体系结构

第 9 章　文档、视图和单文档界面

在 MFC 的早期，应用程序的体系结构与本书前 3 章中示例程序的样式几乎相同。在 MFC 1.0 版本中，应用程序具有两个主要的组件：代表应用程序自身的应用程序对象和代表应用程序窗口的窗口对象。应用程序对象的主要任务是创建窗口，反过来窗口再处理消息。除提供了一般用途的类如 CString 和 CTime 等来代表与 Microsoft Windows 无关的对象以外，MFC 几乎就是对 Windows API 的封装，它给窗口、对话框、设备描述表以及其他在 Windows 中以这种或那种形式存在的对象移植了面向对象的接口。

通过引入文档/视图体系结构，MFC 2.0 改变了 Windows 应用程序编制的方式。在文档/视图应用程序中，应用程序的数据由文档对象代表，而数据的视图由视图对象代表。文档和视图合作来处理用户的输入并绘制结果数据的文字或图形表示。MFC 的 CDocument 类是文档对象的基类，CView 类是视图对象的基类。应用程序的主窗口，其操作功能在 MFC 的 CFrameWnd 和 CMDIFrameWnd 类中实现，已经不再以消息处理为工作焦点了，而是主要用作了视图、工具栏以及其他用户界面对象的容器。

一种将文档与它的视图分开的程序设计模型具有许多好处，不仅仅是使软件组件的工作分工更明确，并形成高度模块化作业。利用 MFC 体系结构的更引人注目的原因是它简化了开发过程。主结构为您提供了处理常规杂务的程序代码，例如：在关闭一个文档之前要提示用户保存未保存的数据。这些提供的程序代码可以将一般的应用程序转换为 Active Document 服务器，可以将文档存入磁盘并读回，还可以简化打印编程等等。

MFC 支持两种类型的文档/视图应用程序。单文档界面（SDI）应用程序只支持一打开一个文档。多文档界面（MDI）应用程序允许同时打开两个以上文档，还支持给定文档的多个视图。WordPad 应用程序是一个 SDI 应用程序；Microsoft Word 是一个 MDI 应用程序。主结构隐藏了两种用户界面模型间的许多差别，以致编写 MDI 应用程序与编写 SDI 应用程序没有多少不同，但是现在程序开发者对于使用多文档界面并不积极，这是由于 SDI 模型改善了以文档为中心的用户界面。如果用户想同时编辑两个文档，Microsoft 更愿意每个文档都在分开的应用程序实例中被显示。所以在本章讨论文档/视图体系结构时明确地以单文档界面为重点。但是这里学到的任何内容也都适用于 MDI 应用程序，并且出于完整性的考虑，在第 11 章中我们将讨论多文档界面以及在 SDI 应用程序中支持多视图的方法。

9.1 文档/视图基础知识

在学习文档/视图体系结构之前,我们先从概念上把握一下所涉及的多种对象以及它们之间的关系。图 9-1 是 SDI 文档/视图应用程序的示意图。框架窗口是应用程序的顶层窗口,通常是 WS_OVERLAPPEDWINDOW 样式的窗口,带有可缩放边框、标题栏、系统菜单以及最小化、最大化和关闭按钮。视图是子窗口,大小与框架窗口相适应,在实际中作为框架窗口的客户区。应用程序的数据保存在文档对象中,数据的可视表示保存在视图中。对于 SDI 应用程序,框架窗口类是从 CFrameWnd 派生来的,文档类是从 CDocument 派生的,而视图类是从 CView 或相关类如 CScrollView 派生来的。

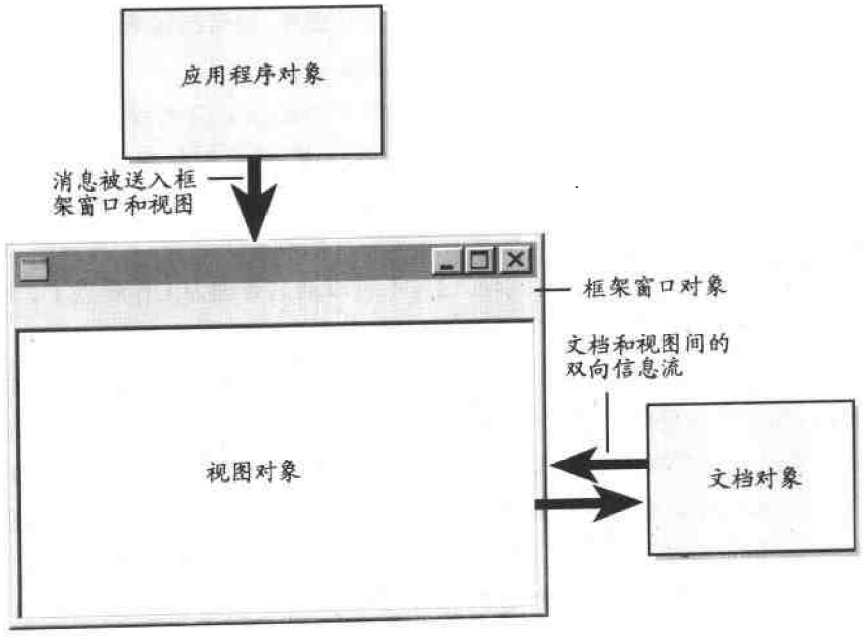

图 9-1 SDI 文档/视图体系结构

箭头代表数据流动的方向。应用程序对象提供消息循环给框架窗口和视图提取消息。视图对象将鼠标和键盘输入转换为处理保存在文档中的数据的命令,文档对象提供了视图所需要的用来输出的数据。对象各自之间还以其他方式互相影响,但是只有在深入学习了每个对象在程序操作中所起的作用并编写了自己的一到两个文档/视图应用程序之后,才能容易地掌握总体结构。

图 9-1 描述的体系结构隐含了应用程序的设计和操作的内容。在 MFC 1.0 样式的应用程序中,程序数据通常保存在框架窗口类内声明的成员变量中。框架窗口通过访问自己的成员变量并使用封装在 CDC 类中的 GDI 函数在自己的客户区绘制"视图"。文档/视图体系

结构通过将数据封装在独立的文档对象中并为程序的屏幕输出提供视图对象，从而增强了模块化程序设计的方法。文档/视图应用程序从来不会为框架窗口获取客户区设备描述表并在其中绘制输出，相反它绘制输出到视图中。看上去好像是在框架窗口中绘制，实际上所有输出都到了视图。如果愿意，可以给框架窗口绘制内容，但您是不会看到输出结果的，因为SDI框架窗口的客户区完全被视图遮盖了。

9.1.1　再看 InitInstance 函数

SDI文档/视图应用程序最有意思的一个方面是框架窗口、文档和视图对象的创建方式。如果看一下AppWizard生成的SDI应用程序中的InitInstance函数，会看到如下内容：

```
CSingleDocTemplate* pDocTemplate;
pDocTemplate = new CSingleDocTemplate(
  IDR_MAINFRAME,
  RUNTIME_CLASS(CMyDoc),
  RUNTIME_CLASS(CMainFrame),
  RUNTIME_CLASS(CMyView)
);
AddDocTemplate(pDocTemplate);
      .
      .
      .
CCommandLineInfo cmdInfo;
ParseCommandLine(cmdInfo);

if(! ProcessShellCommand(cmdInfo))
   return FALSE;

m_pMainWnd->ShowWindow(SW_SHOW);
m_pMainWnd->UpdateWindow();
```

这些程序代码与本书第一部分中示例程序中的启动程序代码完全不同。让我们仔细分析一下InitInstance函数，看看到底是什么使文档/视图应用程序启动并运行的。

首先，语句

```
CSingleDocTemplate* pDocTemplate;
pDocTemplate = new CSingleDocTemplate(
  IDR_MAINFRAME,
  RUNTIME_CLASS(CMyDoc),
  RUNTIME_CLASS(CMainFrame),
  RUNTIME_CLASS(CMyView)
);
```

从MFC的CSingleDocTemplate类创建了一个SDI文档模板。SDI文档模板是SDI文档/视图

应用程序的最主要的成分。它表示了用来管理应用程序数据的文档类、包含数据视图的框架窗口类,以及用来绘制可视数据表示的视图类。文档模板还保存了资源 ID,主结构用它来加载菜单、加速键以及其他形成应用程序用户界面的资源。AppWizard 在它生成的程序代码中使用了资源 ID IDR_MAINFRAME。类名字括号外的 RUNTIME_CLASS 宏对于所指定的类返回指向 CRuntimeClass 结构的指针,这就使得主结构可以在运行时创建这些类的对象了。这种动态创建机制是文档/视图体系结构的另一个重要的成分。在本章稍后我将说明它的工作原理。

在文档模板创建以后,语句

```
AddDocTemplate (pDocTemplate);
```

将它添加到由应用程序对象保存的文档模板列表中。用此方法注册的每个模板都定义了一个应用程序支持的文档类型。SDI 应用程序只注册一个文档类型,而 MDI 应用程序可以并且有时也注册多个类型。

语句

```
CCommandLineInfo cmdInfo;
ParseCommandLine (cmdInfo);
```

通过 CWinApp::ParseCommandLine 并用反映命令行输入参数的值来初始化 CCommandLineInfo 对象,其中通常包含文档文件名。语句

```
if (! ProcessShellCommand (cmdInfo)) return FALSE;
```

"处理"命令行参数。首先,ProcessShellCommand 调用 CWinApp::OnFileNew 来启动应用程序,如果文件名没有在命令行上输入就使用空文档,或者在指定了文档名称的情况下就使用 CWinApp::OpenDocumentFile 来加载一个文档。正是在程序执行的这个阶段,主结构用保存在文档模板中的信息来创建文档、框架窗口和视图对象。(可能您会奇怪为什么先创建的是文档接着才是框架窗口和视图。)如果初始化成功,则 ProcessShellCommand 返回 TRUE,否则返回 FALSE。如果初始化是成功的,语句

```
m_pMainWnd -> ShowWindow (SW_SHOW);
m_pMainWnd -> UpdateWindow ();
```

将在屏幕上显示应用程序的框架窗口(通过扩展显示视图)。

在应用程序被启动,文档、框架窗口和视图被创建之后,消息循环就开始工作了,应用程序也开始检索和处理消息。与 MFC 1.0 类型的应用程序不同,它的典型特点是将所有消息都映射给了框架窗口类的成员函数,文档/视图应用程序则是在应用程序、文档类、视图类、框架窗口类之间划分开了消息处理。主结构为使这些工作划分成为可能,在后台做了很多工作。在 Windows 中,只有窗口才可以接收消息,因此 MFC 使用了非常复杂的命令传递机制,就是将某种类型的消息按照预定的顺序从一个对象传递到另一个对象,直到处理消息的对象之一或消息被传递给

::DefWindowProc进行默认处理为止。在本章稍后讨论命令传递时就会很清楚为什么命令传递是MFC 强大的功能特性，如果没有它就会极大地抑制文档/视图体系结构的用途。

9.1.2　文档对象

在文档/视图体系结构中，数据被保存在文档对象中。文档对象是在主结构初始化从CDocument 派生出的类时创建的。术语“文档”有点容易让人误解，它与文字处理器和电子表格程序以及其他程序所处理的传统意义上的文档有些混淆。实际上，文档/视图中的“文档”含义更加广泛。文档几乎可以指任何东西，从扑克仿真游戏中的一付纸牌到在线连接再到远程数据资源；它是程序数据的抽象表示，在数据的保存和给用户提供数据之间划分了清晰的界限。通常，文档对象为其他对象，主要是视图，提供了公用成员函数，使用它可以访问文档的数据。所有数据的处理都由文档对象自己完成。

文档数据通常保存在派生文档类的成员变量中。Microsoft Visual C++ 中提供的 Scribble 教程将数据成员声明为公用类型，使得其他对象可以直接使用它们，而更严格的封装应该是将文档数据声明为私有类型，并提供可以用来访问它们的公用成员函数。例如，在文本编辑程序中，可能会将字符保存在 CByteArray 对象中并提供 AddChar 和 RemoveChar 函数，以便视图可以将它接收到的鼠标和键盘消息转换为命令来添加和删除字符。其他函数如 AddLine 和 DeleteLine，可以进一步丰富文档对象和与其联系的视图对象的接口功能。

CDocument 操作

在 MFC 文献中，术语“操作”被用来描述非虚拟类成员函数。派生文档类从 CDocument 中继承了几个重要的操作，其中一部分在表 9-1 中给出。

表 9-1　主要的 CDocument 操作

函数	说　明
GetFirstViewPosition	返回 POSITION，用来传递给 GetNextView，以开始列举与文档关联的视图
GetNextView	返回 CView 指针，指向与文档关联的视图列表中下一个视图
GetPathName	检索文档的文件名称和路径，例如 C:\Documents \Personal \MyFile.doc；如果文档还未命名则返回空字符串
GetTitle	检索文档的标题，例如 MyFile；如果文档还未命名则返回空字符串
IsModified	如果文档包含未保存的数据就返回非零值，否则为零
SetModifiedFlag	设置或清除文档中已修改的标志，该标志说明文档是否包含没有保存的数据
UpdateAllViews	通过调用每个视图的 OnUpdate 函数来更新与文档关联的所有视图

这些函数中，SetModifiedFlag 和 UpdateAllViews 是最常用的两个。每次修改了文档数据之后都要调用 SetModifiedFlag。此函数在文档对象内设置一个标志告诉 MFC 文档包含未保存的数据，并允许 MFC 在关闭文档之前提示用户该文档包含未保存的更改。您可以使用

IsModified 自己来确定文档是否被改过。UpdateAllViews 命令所有与文档关联的视图更新它们自己。实际上,是 UpdateAllViews 调用每个视图的 OnUpdate 函数,其默认操作是使视图无效实现重绘。在支持文档具有多个视图的应用程序中,每当文档数据改变后调用 UpdateAllViews 可以维持不同视图的同步。即使是单视图应用程序也可以调用 UpdateAllViews 来刷新基于当前保存在文档中数据的视图。

通过使用 GetFirstViewPosition 和 GetNextView 处理视图列表,文档对象可以列举它的视图,而且还可以单独与每个视图通信。下列从 MFC 源代码文件 Doccore.cpp 中节选的一段程序说明了 UpdateAllViews 使用 GetFirstViewPosition 和 GetNextView 调用每个视图的 OnUpdate 函数的过程。

```
POSITION pos = GetFirstViewPosition();
while (pos != NULL)
{
  CView* pView = GetNextView(pos);
     .
     .
     .
  pView->OnUpdate(pSender, lHint, pHint);
}
```

由于 OnUpdate 是 CView 的保护型成员函数,您可能会不理解这样的程序怎么可以编译通过。答案是 CDocument 在 Afxwin.h 中被声明为 CView 的友元。您可以在自己的程序中自由地调用 GetFirstViewPosition 和 GetNextView,但是如果也将文档声明为视图的友元,则只能从文档类中调用 OnUpdate。

CDocument 可覆盖函数

CDocument 也包含几个虚拟函数,或称"可覆盖函数",被覆盖来自定义文档的功能。其中的一些函数几乎总是在派生文档类中被覆盖。表 9-2 中给出了 4 个最常用的可覆盖函数。

表 9-2　主要的 CDocument 可覆盖函数

函数	说　明
OnNewDocument	在新文档被创建时由主结构调用。覆盖它是为了每次创建新文档时都对文档对象应用专门的初始化
OnOpenDocument	在从磁盘上装载文档时由主结构调用。覆盖它是为了每次装载新文档时都对文档对象应用专门的初始化
DeleteContents	主结构调用它来删除文档的内容。覆盖它是为了在文档关闭之前释放分配给文档的内存和其他资源
Serialize	主结构调用它在文档和磁盘之间串行化输出或输入。覆盖它是为了提供针对文档的串行化程序以便文档可以被装载和保存

在SDI应用程序中，MFC只实例化文档对象一次（在应用程序启动时），却在文档文件被打开和关闭时反复使用该对象。由于文档对象只创建一次，因此文档类构造函数也只执行一次初始化。但是如果在新文档创建时或已存在的文档从磁盘上装载时，想要对派生文档类包含的成员变量进行重新初始化，那怎么办呢？

这就是要使用OnNewDocument和OnOpenDocument的原因。每次新文档创建时MFC都调用文档的OnNewDocument函数，通常发生在用户从"文件"菜单选择了"新建"命令时。当文档从磁盘上装载时，即用户从"文件"菜单选择了"打开"命令时，MFC将调用OnOpenDocument。您可以在SDI文档类的构造函数中进行一次初始化。但是如果想随时在文档被创建或打开时进行某种初始化工作，就必须覆盖OnNewDocument或OnOpenDocument。

MFC提供了OnNewDocument和OnOpenDocument的默认实现，减轻了创建新文档和打开已有文档的工作负担。如果要覆盖OnNewDocument和OnOpenDocument，应该调用基类中相同的函数，如下：

```
BOOL CMyDoc::OnNewDocument ()
{
   if (!CDocument::OnNewDocument ())
      return FALSE;
   // Insert application-specific initialization code here.
   return TRUE;
}

BOOL CMyDoc::OnOpenDocument (LPCTSTR lpszPathName)
{
   if (!CDocument::OnOpenDocument (lpszPathName))
      return FALSE;
   // Insert application-specific initialization code here.
   return TRUE;
}
```

一般来说，MFC应用程序更经常覆盖OnNewDocument而不是OnOpenDocument，为什么？因为OnOpenDocument间接调用了文档中的Serialize函数，它使用从文档文件中检索到的值初始化文档的永久数据成员。只有非永久数据成员（不用Serialize初始化的）才需要在OnOpenDocument中被初始化。与它相比，OnNewDocument不执行文档数据成员的默认初始化。如果给文档类添加了数据成员并希望这些数据成员在新文档创建时再次被初始化，那么就需要覆盖OnNewDocument。

在新文档被创建或打开之前，主结构调用文档对象的虚拟DeleteContents函数来删除文档中已存在的数据。因此，SDI应用程序可以覆盖CDocument::DeleteContents并利用它释放分配给文档的任意资源，还可以执行其他必要的清理工作为重新使用文档对象作准备。MDI应用程序通常也是这样，但是MDI文档对象与SDI文档对象的不同之处在于，它们是在

用户打开和关闭文档时各自创建和销毁的。

在文档被打开或关闭时,主结构将调用文档对象的 Serialize 函数来串行化文档的数据。可以编写实现 Serialize 函数以便它能够将文档的数据流入或流出;而让主结构完成其他所有工作,包括打开文件来读取或写入,以及提供 CArchive 对象使您与难对付的实际的磁盘 I/O隔绝。派生文档类的 Serialize 函数的典型结构如下:

```
void CMyDoc::Serialize (CArchive& ar)
{
   if (ar.IsStoring ()) {
      // Write the document's persistent data to the archive.
   }
   else { // Loading, not storing
     // Read the document's persistent data from the archive.
   }
}
```

在注释的地方,可以加入一些程序,使用第 6 章介绍的串行化机制将文档的永久数据流出到档案或从档案中流入。对于一个简单的文档类,它的数据包括保存在 CString 成员变量 m_strName 和 m_strPhone 中的两个字符串,可以按下列方式编写 Serialize:

```
void CMyDoc::Serialize (CArchive& ar)
{
    if (ar.IsStoring ()) {
        ar << m_strName << m_strPhone;
    }
    else { // Loading, not storing.
        ar >> m_strName >> m_strPhone;
    }
}
```

如果文档数据是由原始数据类型和可串行化类如 CString 组成的,那么编写 Serialize 函数就极其容易,因为所有的输入输出都可以用 << 和 >> 运算符执行。对于结构和其他不可串行化的数据类型,可以使用 CArchive 函数 Read 和 Write。CArchive 甚至还包含 ReadString 和 WriteString 函数用来串行化原始字符串。如果所有这些方法都不成功,您可以调用 CArchive::GetFile 来获取 CFile 指针,这样就可以与同档案联系的文件直接交互了。在第 13 章的 HexDump 程序中会看到这种技术的应用。

其他不经常使用但很有用的 CDocument 覆盖函数包括:OnCloseDocument,在文档关闭时调用;OnSaveDocument,在文档保存时调用;SaveModified,在包含未保存数据的文档被关闭之前调用,用来询问用户是否保存已做的改动;以及 ReportSaveLoadException,串行化过程中有错误发生时调用。还有别的函数,但其中大部分都属于高级可覆盖函数,可能很少有机会用到它们。

9.1.3　视图对象

鉴于文档对象的唯一任务是管理应用程序的数据，因此视图对象就有两个用途：提供文档的可视化表示，以及将用户输入（特别是鼠标和键盘消息）转换为操作文档数据的命令。这样，文档和视图就被紧紧地联系在了一起，信息在它们之间双向传递。

MFC 的 CView 类定义了视图的基本属性。MFC 还包含一族从 CView 派生来的视图类，用来给视图添加功能。例如：CScrollView，给 CView 添加滚动功能。CScrollView 和其他 CView 派生类将在第 10 章讨论。

GetDocument 函数

一个文档可以具有与它联系的多个视图，而一个视图只能属于一个文档。主结构在视图的 m_pDocument 数据成员中保存了指向相关文档对象的指针，并将该指针提供给视图的 GetDocument 成员函数使用。就如同文档对象可以调用 GetFirstViewPosition 和 GetNextView 标识出它的视图一样，视图也可以使用 GetDocument 标识出自己的文档。

在 AppWizard 为视图类生成源代码时，它覆盖了基类的 GetDocument 函数，使该函数可以强制转换 m_pDocument 指向相应的文档类型并返回结果。这种覆盖就确保了对文档对象的类型安全访问，并消除了每次调用 GetDocument 时进行外部强制类型转换的必要。

CView 可覆盖函数

如同 CDocument 类一样，CView 也包含几个虚拟成员函数，在派生类中覆盖它们可以自定义视图的功能。在表 9-3 中给出了主要的可覆盖函数。其中最重要的是纯虚拟函数 OnDraw，每次在视图接收到 WM_PAINT 消息时调用它。在文档/视图应用程序中，WM_PAINT 消息由 OnPaint 处理程序处理，它使用 CPaintDC 对象来完成绘制工作。在文档/视图应用程序中，是主结构产生 WM_PAINT 消息，创建 CPaintDC 对象，并用指向 CPaintDC 对象的指针来调用视图的 OnDraw 函数。下列实现 OnDraw 的程序从文档对象中检索 CString 并在视图中心显示它：

```
void CMyView::OnDraw(CDC* pDC)
{
    CMyDoc* pDoc = GetDocument();
    CString string = pDoc->GetString();
    CRect rect;
    GetClientRect(&rect);
    pDC->DrawText(string, rect,
        DT_SINGLELINE|DT_CENTER|DT_VCENTER);
}
```

注意,OnDraw 使用的是提供的设备描述表指针而不是初始化自己的设备描述表。

表 9-3 主要的 CView 可覆盖函数

函数	说 明
OnDraw	被调用来绘制输出文档的数据。覆盖是为了绘制文档视图
OnInitialUpdate	视图第一次附加到文档时被调用。覆盖是为了每次在文档被创建或装载时都初始化视图对象
OnUpdate	在文档的数据已经修改或视图需要更新时调用。覆盖是为了实现有效更新功能,只重画视图中需要重画的部分而不重画整个视图

视图不必构造自己的设备描述表,这只是一个小小的便利。主结构使用 OnDraw 的真正原因是:可用相同的程序实现向窗口输出、打印和打印预览。在 WM_PAINT 消息到达后,主结构给视图传递一个指向屏幕设备描述表的指针以便输出到窗口中。在文档被打印时,主结构会调用相同的 OnDraw 函数并给它传递一个指向打印机设备描述表的指针。由于 GDI 是设备独立的图形系统,如果用户使用了两种不同的设备描述表,那么相同的程序可以在不同的两个设备上产生相同(或几乎相同)的输出。MFC 利用了这个特点使得打印工作(在 Windows 中通常很琐碎)省事了许多。事实上,在文档/视图应用程序中实现打印要比在其他传统的应用程序中容易许多。在第 13 章会学到完整的 MFC 打印体系结构。

其他两个在派生视图类中要经常覆盖的 CView 函数是 OnInitialUpdate 和 OnUpdate。在 SDI 应用程序中,视图和文档一样,只构造一次然后重复使用。在 SDI 中,每当文档被打开或创建时都要调用视图的 OnInitialUpdate 函数。OnInitialUpdate 的默认实现要调用 OnUpdate,而 OnUpdate 的默认实现接下来将使视图客户区无效并执行重绘。使用 OnInitialUpdate 来初始化视图类的数据成员,并在单文档基础上执行其他与视图相关的初始化。例如:在 CScrollView 派生类中,通常 OnInitialUpdate 要调用视图的 SetScrollSizes 函数来初始化滚动参数。在覆盖后的版本中调用 OnInitialUpdate 的基类中的版本很重要,否则新文档被打开或创建时视图不会被更新。

在文档数据被修改或其他对象(通常可能是文档对象或视图之一)调用 UpdateAllViews 时将调用 OnUpdate。根本没必要覆盖 OnUpdate 中默认状态下调用 Invalidate 的部分。但在实际中,却要经常覆盖 OnUpdate 来优化操作,只重绘视图中需要更新的部分而不是重绘整个视图。这种所谓的智能更新在多视图应用程序中特别有益,因为它消除了难看的视图闪烁。在第 11 章示例程序 Sketch 中您就会明白我的意思了。

在多视图应用程序中,任何时刻只有一个视图是活动视图,其他都是非活动视图。通常活动视图具有输入焦点。通过覆盖 CView::OnActivateView,视图可以确定何时激活何时无效。如果视图将被激活,OnActivateView 的第一个参数就是非零值,如果将要无效则为零。第 2 和第 3 个参数是 CView 指针,分别标识即将被激活或无效的视图。如果指针相同,说明应用程序的框架窗口没有在活动视图中进行改动就被激活了。视图对象有时利用 OnActivateView 函数

的这个功能来实现调色板。使用 CFrameWnd::GetActiveView 和 CFrameWnd::SetActiveView，框架窗口可以获取和设置活动视图。

9.1.4 框架窗口对象

到目前为止，我们已经看到了应用程序对象、文档对象和视图对象在文档/视图应用程序中所起的作用。现在该研究框架窗口对象了，它在屏幕上定义了应用程序的实际工作空间，同时也担当了视图的容器。SDI 应用程序只有一个框架窗口 CFrameWnd，它被用作应用程序的顶层窗口并用来包含视图。在下一章您会发现 MDI 应用程序使用两种不同类型的窗口，其中 CMDIFrameWnd 用作顶层窗口，而 CMDIChildWnd 窗口在顶层窗口中浮动用来包含应用程序文档的视图。

在文档/视图应用程序的操作中，框架窗口具有很重要的作用，同时也常常被误解。刚起步的 MFC 程序员通常将框架窗口作为一个简单的窗口。事实上，框架窗口是一种智能对象，它在文档/视图应用程序中管理着许多幕后工作。例如：MFC 的 CFrameWnd 类提供了 OnClose 和 OnQueryEndSession 处理程序，确保用户有机会在应用程序关闭或 Windows 关闭之前保存未保存的数据。在框架窗口被缩放时 CFrameWnd 还处理最重要的缩放视图的任务，它了解如何使用工具栏、状态栏以及其他用户界面对象。它还包含一些成员函数用来操纵工具栏和状态栏，标识活动文档和视图等等。

或许理解 CFrameWnd 类作用的最好方法可能就是把它与更普通的 CWnd 类进行比较。CWnd 类基本上是对一般窗口的 C++ 类型的包装。CFrameWnd 从 CWnd 派生而来并添加了许多作为框架窗口需要的功能，从而在文档/视图应用程序运行中具有了突出的作用。

9.1.5 动态对象创建

如果在程序运行的过程中主结构要创建文档、视图以及框架窗口对象，那么构造这些对象的类必须支持所谓的动态创建特性。MFC 的 DECLARE_DYNCREATE 和 IMPLEMENT_DYNCREATE 宏使得编写可动态创建的类非常容易。下面列出了需要做的工作：

1. 从 CObject 派生对象。
2. 在类声明中调用 DECLARE_DYNCREATE。DECLARE_DYNCREATE 只接收一个参数，即可动态创建类的名字。
3. 在类声明的外部调用 IMPLEMENT_DYNCREATE。IMPLEMENT_DYNCREATE 接收两个参数：可动态创建类的名字和其基类的名字。

用类似下列语句可以使用这些宏来创建类的实例：

```
RUNTIME_CLASS(CMyClass)->CreateObject();
```

使用此语句基本上与使用 new 运算符创建 CMyClass 对象没有什么不同，但它却回避了 C++ 语言中不让如下语句执行的缺点：

```
CString strClassName = _T("MyClass");
CMyClass* ptr = new strClassName;
```

当然,由于编译器不知道 strClassName 是个变量名而不是名义上的类名,所以它会试图去从名为 strClassName 的类中构造一个对象。MFC 的动态对象创建机制其实是应用程序用来注册类的一种方式,在这种方式下主结构能够创建这些类的对象。

在编写可动态创建的类时到底内部是怎样处理的呢? DECLARE_DYNCREATE 宏在类声明中添加了 3 个成员:一个静态 CRuntimeClass 数据成员、一个名为 GetRuntimeClass 的虚拟函数以及一个静态函数 CreateObject。当您编写了

```
DECLARE_DYNCREATE (CMyClass)
```

时,C++ 预处理程序输出:

```
public:
    static const AFX_DATA CRuntimeClass classCMyClass;
    virtual CRuntimeClass* GetRuntimeClass() const;
    static CObject* PASCAL CreateObject();
```

IMPLEMENT_DYNCREATE 通过类名和类实例大小等这样的信息来初始化 CRuntimeClass 结构。它还提供了 GetRuntimeClass 和 CreateObject 函数。如果 IMPLEMENT_DYNCREATE 按以下方式调用:

```
IMPLEMENT_DYNCREATE (CMyClass, CBaseClass)
```

CreateObject 就按以下方式实现:

```
CObject* PASCAL CMyClass::CreateObject()
    { return new CMyClass; }
```

在早期的 MFC 版本中,CreateObject 的实现方式不同,它用保存在类的 CRuntimeClass 结构中的尺寸信息来分配内存,并以手动方式在该内存空间中初始化对象。今天,CreateObject 的实现更符合 C++ 语言了,因为如果动态创建的类重载了 new 运算符,CreateObject 就会使用重载后的运算符。

9.1.6 有关 SDI 文档模板的其他内容

在本章的开始部分,已经看到了 SDI 文档模板对象从 CSingleDocTemplate 类被创建的过程。模板的构造函数被传递了 4 个参数:一个等于 IDR_MAINFRAME 的整数值以及 3 个 RUNTIME_CLASS 指针。现在我们该了解 3 个 RUNTIME_CLASS 宏的用途了,不过要先仔细研究一下所传递的第一个整数参数,它实际上是个多用途的资源 ID,用来标识下列四种资源:

- 应用程序图标
- 应用程序的菜单
- 伴随菜单的加速键列表
- 文档字符串(document string),首先指定应用程序为文档生成的默认文件扩展名以及未命名文档的默认名字。

在SDI文档/视图应用程序中,主结构创建顶层窗口是这样的:先用保存在文档模板中的运行时创建类的信息来创建一个框架窗口对象,然后再调用该对象的LoadFrame函数。LoadFrame接收的参数之一是资源ID,用来标识上面列出的4种资源。不必奇怪,主结构传递给LoadFrame的资源ID与为文档模板提供的完全相同。LoadFrame创建一个框架窗口并一次装载相关的菜单、加速键和图标,但如果想操作顺利进行,就必须给所有的资源分配相同的ID。这就是在AppWizard为文档/视图应用程序生成的RC文件中对多种不同的资源使用相同的ID的原因。

文档字符串是一个字符串资源,它由7个用"\n"字符分隔的子字符串组成。每个子字符串描述一种框架窗口或文档类型的特性。按从左到右的顺序,对于SDI应用程序,子字符串具有如下含义:

- 出现在框架窗口标题栏中的标题。通常是应用程序的名称,例如:"Microsoft Draw"。
- 分配给新文档的标题。如果这个子字符串被忽略,默认值为"Untitled"。
- 一个描述文档类型的名字,在注册了两个以上文档类型的MDI应用程序中,当用户选中File菜单上的New命令后,它将和其他文档类型一起出现在对话框中。在SDI应用程序中不使用此子字符串。
- 一个描述文档类型的名字,带有包含文件扩展名的通配符文件说明,例如:"Drawing Files (*.drw)"。在Open和Save As对话框中使用此子字符串。
- 对于某类型文档的默认文件扩展名,例如:".drw"。
- 不带空格的名字用来标识注册时的文档类型,例如:"Draw.Document"。如果应用程序调用CWinApp::RegisterShellFileTypes来注册此文档类型,此子字符串就成了以文档的文件扩展名命名的HKEY_CLASSES_ROOT子键的默认值。
- 一个描述文档类型的名字,例如:"Microsoft Draw Document"。与文档字符串中的前一个子字符串不同,此子字符串可以包含空格。如果应用程序使用CWinApp::RegisterShellFileTypes来注册文档类型,此子字符串就是命令解释器在属性页中显示的便于人们阅读的名字。

没必要提供所有7个子字符串;可以忽略个别子字符串,在"\n"分隔符后紧跟另一个"\n"即可,而且还可以将全部子字符串设置为一串空值。如果是用AppWizard创建应用程序,AppWizard会用Advanced Options对话框中键入的信息为您自动生成文档字符串,在AppWizard的Step4对话框中单击Advanced按钮时会出现该对话框。对于典型的SDI文档,字符

串资源语句如下：

```
STRINGTABLE
BEGIN
    IDR_MAINFRAME "Microsoft Draw\n\n\nDraw Files( * .drw)\n.drw\n
        Draw.Document\nMicrosoft Draw Document"
END
```

STRINGTABLE 生成一个字符串列表资源(包含一个以上文本字符串的资源，每一个都有唯一的资源 ID 标识)，这就像 DIALOG 生成对话框资源以及 MENU 生成菜单资源一样。当应用程序用一个空文档打开时，其框架窗口的标题就是“Untitled-Microsoft Draw”。此应用程序保存的文档默认文件扩展名为“.drw”，而且“Draw Files (* .drw)”将成为 Open 和 Save As 对话框中列出的文件类型选项之一。

任何时候如果有需要，都可以使用 MFC 的 CDocTemplate::GetDocString 函数从文档字符串中检索各个子字符串。例如语句

```
CString strDefExt;
pDocTemplate->GetDocString(strDefExt, CDocTemplate::filterExt);
```

将文档的默认文件扩展名复制到名为 strDefExt 的 CString 变量中。

9.1.7 用操作系统命令解释器注册文档类型

在 Windows 中，可以双击文档图标或是右键单击它并在上下文菜单中选择 Open 命令将该文档连同创建它的应用程序一起打开。另外，还可以从上下文菜单中选择 Print 或将文档图标拖放到打印机图标上来打印文档。

要实现这些操作，应用程序必须用操作系统命令解释器来注册其文档类型，这就涉及到在注册表的 HKEY_CLASSES_ROOT 部分添加一系列输入项，用来标识每个文档的文件扩展名以及用来打开和打印该类文件的命令。一些应用程序有两种方式进行此类注册：一种是提供了 REG 文件，用户可以把它合并到注册表；另一种是使用::RegCreateKey、::RegSetValue 和其他 Win32 注册函数通过编程将必要的输入项添加到注册表中。但是 MFC 应用程序却可以通过简单的函数调用来注册它支持的所有文档类型。调用 CWinApp::RegisterShellFileTypes 并在调用 AddDocTemplate 之后传递一个 TRUE 参数可以在应用程序、其创建的文档以及操作系统命令解释器之间铸就关键的链接。在 AppWizard 创建文档/视图应用程序时，它会在应用程序类的 InitInstance 函数中自动包含对 RegisterShellFileTypes 的调用。

相关的 CWinApp 函数 EnableShellOpen 给 MDI 文档/视图应用程序添加了漂亮的特性。如果 MDI 应用程序使用 RegisterShellFileTypes 和 EnableShellOpen 注册了其文档类型，在应用程序运行时如果用户双击了文档图标，命令解释器并不会自动启动第二个应用程序的实例；首先它会使用动态数据交换(DDE)给已存在的实例发送“打开”命令以及文档的文件名称。MFC 的 CDocManager 类中内置的 DDE 处理程序通过调用 OnOpenDocument 打开文档来作出

响应。这样，文档就出现在顶层 MDI 框架内的新窗口中了，正像用应用程序的 File 菜单中 Open 命令打开的一样。相似的 DDE 命令能够使运行中的应用程序实例满足通过操作系统命令解释器提出的打印请求。

9.1.8 命令传送

文档/视图体系结构中最引人注目的特性是应用程序几乎可以在任何地方处理命令消息。“命令消息”是 MFC 中的术语，指的是 WM_COMMAND 消息，当选择了菜单项、键盘加速键被按下或工具栏按钮被单击时产生此消息。框架窗口实际上是大多数命令消息的接收者，但命令消息可以在视图类、文档类、甚至应用程序类中被处理，只要在该类的消息映射表中对想要处理的消息添加输入项即可。命令传送使得您可以将命令处理程序放在最合适的地方，避免把它们都堆在框架窗口类中。对于菜单项、工具栏按钮以及其他用户界面对象的更新命令也遵循命令传送机制，因此也可以把 ON_UPDATE_COMMAND_UI 处理程序放在非框架窗口中。

使命令传送工作的机制根植于 MFC 深层内部。当框架窗口接收到 WM_COMMAND 消息时，会调用所有 CCmdTarget 派生类特有的虚拟 OnCmdMsg 函数来开始传送过程。CFrameWnd 中 OnCmdMsg 的实现如下：

```
BOOL CFrameWnd::OnCmdMsg(...)
{
    // Pump through current view FIRST.
    CView* pView = GetActiveView();
    if (pView != NULL && pView->OnCmdMsg(...))
        return TRUE;

    // Then pump through frame.
    if (CWnd::OnCmdMsg(...))
        return TRUE;

    // Last but not least, pump through application.
    CWinApp* pApp = AfxGetApp();
    if (pApp != NULL && pApp->OnCmdMsg(...))
        return TRUE;

    return FALSE;
}
```

CFrameWnd::OnCmdMsg 首先调用视图的 OnCmdMsg 函数将消息传送给活动视图。如果 pView->OnCmdMsg 返回 0，说明视图没有处理消息（也就是说视图消息映射表中没有包含这个特殊消息的输入项），框架窗口就会亲自调用 CWnd::OnCmdMsg 来处理消息。如果这样也不行，框架窗口就让应用程序对象去处理。最终如果没有对象可以处理此消息，CFrameWnd::OnCmdMsg 将返回 FALSE，主结构将消息传送给::DefWindowProc 进行默认

处理。

这就说明了框架窗口获得的命令消息被传送给活动视图和应用程序对象的过程,但是和文档对象有什么关系呢? 在 CFrameWnd::OnCmdMsg 调用活动视图的 OnCmdMsg 函数时,视图首先自己来处理消息。如果对该消息没有处理程序,视图就会调用文档的 OnCmdMsg 函数。如果文档也不能处理消息,它就会向上将消息传送给文档模板。图 9-2 说明了递交给 SDI 框架窗口的命令消息所经历的路径。活动视图首先处理消息,接着是与视图关联的文档、文档模板、框架窗口,最后是应用程序对象。如果路径上的某个对象可以处理消息,那么消息传送就会停止,如果没有对象的消息映射表中包含此消息的输入项,它就会被一路传送给::DefWindowProc。对于递交给 MDI 框架窗口的命令消息,传送过程几乎完全相同,只是主结构要确保所有相关对象,包括包含活动 MDI 视图的子窗口框架,都有机会处理消息。

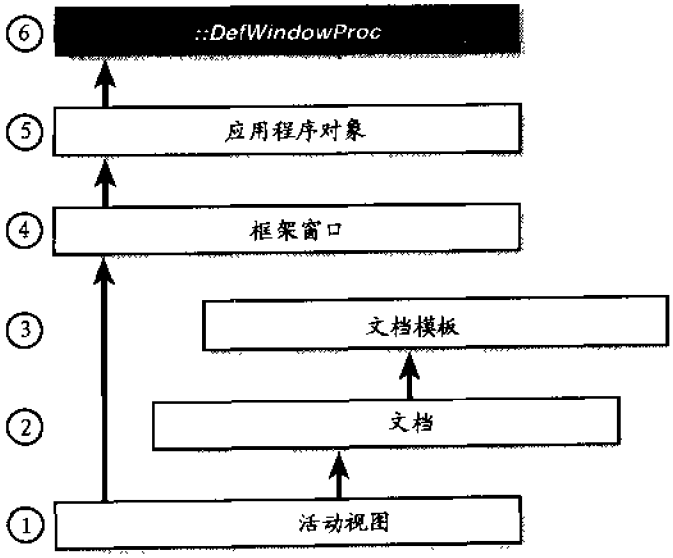

图 9-2　递送给 SDI 框架窗口的命令消息的传送示意图

看一下典型的文档/视图应用程序处理从菜单、加速键、工具栏按钮来的命令就会明了命令传送的意义了。习惯上,File 菜单中的 New、Open 和 Exit 命令由应用程序对象处理,其中 CWinApp 为它们提供了 OnFileNew、OnFileOpen 和 OnAppExit 命令处理程序。File 菜单中的 Save 和 Save As 命令通常由文档对象处理,它提供了名为 Document::OnFileSave 和 CDocument::OnFileSaveAs 的默认命令处理程序。显示和隐藏工具栏和状态栏的命令由框架窗口使用 CFrameWnd 成员函数来处理,其他大多数命令则由文档或视图处理。

考虑在哪里放置消息处理程序时,要记住重要的一点,只有命令消息和用户界面更新才遵循传送机制。标准 Windows 消息如 WM_CHAR、WM_LBUTTONDOWN、WM_CREATE 以及 WM_SIZE 必须在接收这些消息的窗口中处理。鼠标和键盘消息通常传给视图,大多数其他

消息则传给框架窗口。文档对象和应用程序对象从不接收非命令消息。

9.1.9 预定义的命令 ID 和命令处理程序

在编写文档/视图应用程序时，通常不会为所有菜单命令亲自编写处理程序。CWinApp、CDocument、CFrameWnd 和其他 MFC 类为常用菜单命令（如 File 中的 Open 和 Save 命令）提供了默认处理程序。另外，主结构还提供了多种标准菜单项命令 ID，例如 ID_FILE_OPEN 和 ID_FILE_SAVE，其中许多已经预先添加到使用它们的类的消息映射表中了。

表 9-4 中列出了最常用的预定义命令 ID 和它们相关的命令处理程序。“预添加”列表明处理程序是被自动调用（是）还是只有在相应的输入项添加到类的消息映射表中才被调用（否）。将相应的 ID 赋给菜单项就可以预添加处理程序；没有被预添加的处理程序只有用消息映射输入项将菜单项 ID 链接给函数时才能执行。例如：在 CWinApp 的 OnFileNew 和 OnFileOpen 函数中可以看到 File 菜单中 New 和 Open 命令的默认实现，但是除非提供了 ON_COMMAND 消息映射输入项，否则两个函数都不会连结到应用程序中。而另一方面，CWinApp::OnAppExit 完全可以独立工作，不需要消息映射表输入项。您需要做的工作只是将 ID 值 ID_APP_EXIT 赋给 File 菜单中的 Exit 菜单项，OnAppExit 就会神奇地在用户从 File 菜单选择了 Exit 之后结束应用程序。为什么会这样？因为 CWinApp 的消息映射表中包含 ON_COMMAND (ID_APP_EXIT, OnAppExit)输入项，而且消息映射表和其他类成员一样是通过继承传递的。

表 9-4 预定义的命令 ID 和命令处理程序

命令 ID	菜单项	默认处理程序	预添加？
File 菜单			
ID_FILE_NEW	New	CWinApp::OnFileNew	否
ID_FILE_OPEN	Open	CWinApp::OnFileOpen	否
ID_FILE_SAVE	Save	CDocument::OnFileSave	是
ID_FILE_SAVE_AS	Save As	CDocument::OnFileSaveAs	是
ID_FILE_PAGE_SETUP	Page Setup	无	N/A
ID_FILE_PRINT_SETUP	Print Setup	CWinApp::OnFilePrintSetup	否
ID_FILE_PRINT	Print	CView::OnFilePrint	否
ID_FILE_PRINT_PREVIEW	Print Preview	CView::OnFilePrintPreview	否
ID_FILE_SEND_MAIL	Send Mail	CDocument::OnFileSendMail	否
ID_FILE_MRU_FILE1- ID_FILE_MRU_FILE16	N/A	CWinApp::OnOpenRecentFile	是
ID_APP_EXIT	Exit	CWinApp::OnAppExit	是

续表

命令 ID	菜单项	默认处理程序	预添加?
Edit 菜单			
ID_EDIT_CLEAR	Clear	无	N/A
ID_EDIT_CLEAR_ALL	Clear All	无	N/A
ID_EDIT_CUT	Cut	无	N/A
ID_EDIT_COPY	Copy	无	N/A
ID_EDIT_PASTE	Paste	无	N/A
ID_EDIT_PASTE_LINK	Paste Link	无	N/A
ID_EDIT_PASTE_SPECIAL	Paste Special	无	N/A
ID_EDIT_FIND	Find	无	N/A
ID_EDIT_REPLACE	Replace	无	N/A
ID_EDIT_UNDO	Undo	无	N/A
ID_EDIT_REDO	Redo	无	N/A
ID_EDIT_REPEAT	Repeat	无	N/A
ID_EDIT_SELECT_ALL	Select All	无	N/A
View 菜单			
ID_VIEW_TOOLBAR	Toolbar	CFrameWnd::OnBarCheck	是
ID_VIEW_STATUS_BAR	Status Bar	CFrameWnd::OnBarCheck	是
Window 菜单(仅用于 MDI 应用程序)			
ID_WINDOW_NEW	New Window	CMDIFrameWnd::OnWindowNew	是
ID_WINDOW_ARRANGE	Arrange All	CMDIFrameWnd::OnMDIWindowCmd	是
ID_WINDOW_CASCADE	Cascade	CMDIFrameWnd::OnMDIWindowCmd	是
ID_WINDOW_TILE_HORZ	Tile Horizontal	CMDIFrameWnd::OnMDIWindowCmd	是
ID_WINDOW_TILE_VERT	Tile Vertical	CMDIFrameWnd::OnMDIWindowCmd	是
Help 菜单			
ID_APP_ABOUT	About AppName	无	N/A

MFC 还为一些命令提供了更新处理程序,包括:

- CFrameWnd::OnUpdateControlBarMenu 针对 ID_VIEW_TOOLBAR 和 ID_VIEW_STATUS_BAR 命令。
- CMDIFrameWnd::OnUpdateMDIWindowCmd 针对 ID_WINDOW_ARRANGE、ID_WINDOW_CASCADE、ID_WINDOW_TILE_HORZ、ID_WINDOW_TILE_VERT 和 ID_WINDOW_NEW 命令。
- CDocument::OnUpdateFileSendMail 针对 ID_FILE_SEND_MAIL 命令。

MFC 的 CEditView 和 CRichEditView 类对于 Edit 菜单中的一些选项提供了命令处理程

序，但其他类型的视图必须提供自己的处理程序。

您没必要非得使用主结构提供的预定义命令 ID 或命令处理程序。您完全可以自己创建和设计自定义的命令 ID，并提供消息映射表输入项来关联自己的命令 ID 和默认的命令处理程序。甚至还可以用自己的处理程序替换掉默认的命令处理程序。简言之，您可以任意或多或少地使用主结构提供的支持。但是如果依靠主结构越多，您需要亲自编写的程序代码就越少。

9.2　第一个文档/视图应用程序

既然对文档/视图体系结构和一些实现的细节内容已经有了一定的了解和感受，就让我们编写一个文档/视图应用程序吧。如果本章第一部分所讨论的概念有些抽象的话，看看实现文档/视图应用程序的程序代码会帮助我们把概念具体化。

9.2.1　SdiSquares 应用程序

图 9-3 所示的程序是一个 SDI 文档/视图应用程序，它显示一个具有 4 行 4 列的正方形网格。开始时每个正方形都是白色。但是可以用鼠标左键单击正方形来改变其颜色。默认状态下，单击会将颜色改为红色。可以从 Color 菜单中选择供选的颜色来生成一个多色网格，最多能包含 6 种不同的颜色。

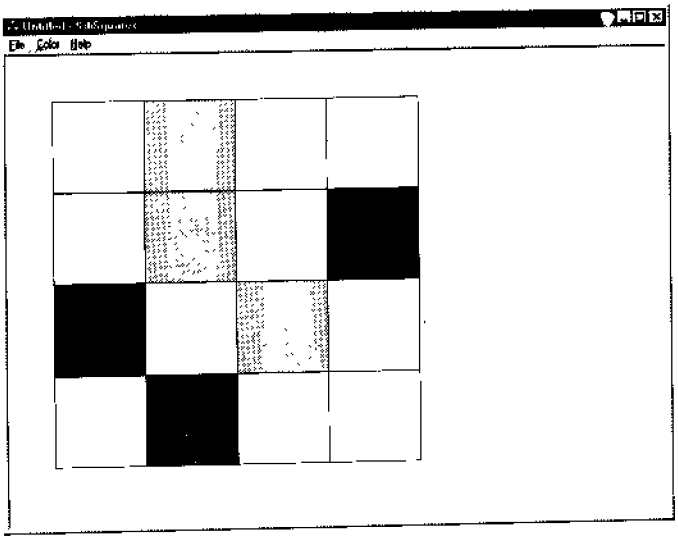

图 9-3　SdiSquares 的窗口

虽然 SdiSquares 仅仅是最基本的文档/视图应用程序,但它说明了文档/视图体系结构的所有原则。在应用程序的操作中 4 个基本类起了关键的作用:

- 应用程序类 CSquaresApp,从 CWinApp 派生而来。
- 框架窗口类 CMainFrame,从 CFrameWnd 派生而来。
- 文档类 CSquaresDoc,从 CDocument 派生而来。
- 视图类 CSquaresView,从 CView 派生而来。

这些类的源程序代码在图 9-4 中给出。

在 SdiSquares 中,组成"文档"的是一个保存 COLORREF 值的二维数组,定义了每个方格的颜色,以及另外一个 COLORREF 值,定义了"当前颜色"(在方格被单击时赋给方格的颜色)。方格的颜色保存在名为 m_clrGrid 的保护型 CSquaresDoc 成员变量中,它是一个保存 COLORREF 值的 4 × 4 数组。当前颜色保存在名为 m_clrCurrentColor 的独立的 CSquaresDoc 成员变量中。在文档的 OnNewDocument 函数中,m_clrGrid 的所有 16 个元素被初始化为白色,m_clrCurrentColor 被初始化为红色。这些变量在 OnNewDocument 中而不是在 CSquaresDoc 的构造函数中被初始化就确保了它们在新文档创建时可以被重新设置。如果在文档的构造函数中对它们进行初始化,那它们只能初始化一次(在应用程序启动时),并且会在新文档创建时保留当前值,这是因为 SDI 应用程序仅构造文档对象一次并在文档被创建和销毁时重复使用它。

为把文档数据提供给视图,CSquaresDoc 应用了 3 个公用成员函数。GetCurrentColor 返回当前颜色(m_clrCurrentColor 的值)。GetSquare 返回给定行列位置(m_clrGrid[i][j])处方格的颜色。SetSquare 将颜色赋给指定行列位置的方格。在赋给方格颜色之后,SetSquare 将调用文档的 SetModifiedFlag 把文档标记为已修改,并调用 UpdateAllViews 重绘视图来显示更新后的网格:

```
m_clrGrid[i][j] = color;
SetModifiedFlag (TRUE);
UpdateAllViews (NULL);
```

GetCurrentColor、GetSquare 以及 SetSquare 作为文档和视图之间的桥梁。由于文档的数据成员是保护型,所以视图不能直接访问它们,但是它可以调用文档的访问函数,因为它们被声明为公用型。

视图的 OnDraw 函数在屏幕上绘制网格。着色方格通过嵌套 for 循环来绘制,以迭代方式一次一行 -列地绘制网格。每次循环迭代,视图都要检索相应方格的颜色,通过保存 GetDocument 返回值的 pDoc 指针来调用文档的 GetSquare 函数就可以实现检索:

```
CSquaresDoc * pDoc = GetDocument();
ASSERT_VALID(pDoc);
        .
        .
        .
```

```
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
      COLORREF color = pDoc->GetSquare(i, j);
      CBrush brush(color);
      int x1 = (j * 100) + 50;
      int y1 = (i * -100) - 50;
      int x2 = x1 + 100;
      int y2 = y1 - 100;
      CRect rect(x1, y1, x2, y2);
      pDC->FillRect(rect, &brush);
    }
}
```

AppWizard 插入了对 GetDocument 的调用和 ASSERT_VALID；我把其他所有语句都添加在了 OnDraw 中。在计算中 OnDraw 使用的是负的 y 值，因为它在 MM_LOENGLISH 映射模式下绘制，该模式下客户区 y 坐标为负。

视图包含 WM_LBUTTONDOWN 处理程序，它将单击处坐标从设备坐标转换为 MM_LOENGLISH 坐标，然后检查看（如果有的话）是哪个方格被击中了。如果单击发生在方格中，CSquaresView::OnLButtonDown 将调用文档的 GetCurrentColor 函数来检索当前颜色：

```
CSquaresDoc* pDoc = GetDocument();
COLORREF clrCurrentColor = pDoc->GetCurrentColor();
```

然后再调用文档的 SetSquare 函数将颜色赋给被击中的方格：

```
pDoc->SetSquare(i, j, clrCurrentColor);
```

在这里可以清楚地看到添加给 CSquaresDoc 的公用成员函数将文档和视图联系起来了，也可以明白 GetDocument 之所以重要的原因。由于视图根本不去管文档的数据实际的保存方式，因此可以在一点儿也不影响视图类的情况下修改文档的内部储存机制。

我将针对 Color 菜单中命令的命令和更新处理程序放在了文档类中，这是因为 m_clrCurrentColor 是文档类的一个成员。命令处理程序只是将 RGB 颜色值赋给 m_clrCurrentColor。更新处理程序使用 CCmdUI::SetRadio 来设置当前颜色。我要是能使用 MFC 的 ON_COMMAND_RANGE 和 ON_UPDATE_COMMAND_UI_RANGE 宏为所有 6 个菜单项提供一个公用的命令处理程序和更新处理程序的话，就可以不必编写 6 个独立的命令处理程序和六个更新处理程序了。但是，由于 ClassWizard 没有提供任何输出 RANGE 宏的方法，所以如果需要，就不得不手工添加。

在用户将 SdiSquares 文档保存到磁盘或从磁盘读取时，MFC 会调用文档的 Serialize 函数。在文档被保存时，CSquaresDoc::Serialize 通过将 m_clrGrid 和 m_clrCurrentColor 串行化输出给档案来作出响应，在打开文档时从档案中串行化输入。下列程序可以完成此工作：

```
void CSquaresDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        for (int i = 0; i < 4; i ++)
            for (int j = 0; j < 4; j ++)
                ar << m_clrGrid[i][j];
        ar << m_clrCurrentColor;
    }
    else
    {
        for (int i = 0; i < 4; i ++)
            for (int j = 0; j < 4; j ++)
                ar >> m_clrGrid[i][j];
        ar >> m_clrCurrentColor;
    }
}
```

因为 COLORREF 是 DWORD 类型,而且 MFC 为 DWORD 重载了 << 和 >> 运算符,所以 m_clrGrid 和 m_clrCurrentColor 值可以根据语法简单地被串行化。AppWizard 生成什么也不做的 Serialize 函数,其中包含对 IsStoring 的调用。您要提供将文档中永久数据流入流出的程序。注意 MFC 处理了次要的工作,例如:给用户显示 Open 或 Save As 对话框,打开文件以供读写等等。这就是为什么在文档/视图应用程序中对保存和装载文档的处理,工作量比在传统的应用程序中少许多的原因。

在图 9-4 中您可能已经看出来了,我使用了 AppWizard 来生成 SdiSquares 的最初的源程序代码,并用了 ClassWizard 来编写消息处理程序、命令处理程序以及更新处理程序。我没有必要去碰 AppWizard 为应用程序类和框架窗口类生成的程序代码。我的大部分工作都与文档和视图类有关,这也正是文档/视图应用程序典型的开发过程。

SdiSquares.h

```
// SdiSquares.h : main header file for the SDISQUARES application
//

#if !defined(
  AFX_SDISQUARES_H__00156CE5_BB17_11D2_A2FD_0000861BAE71__INCLUDED_)
#define AFX_SDISQUARES_H__00156CE5_BB17_11D2_A2FD_0000861BAE71__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

/////////////////////////////////////////////////////////////////////////////
// CSquaresApp:
// See SdiSquares.cpp for the implementation of this class
//

class CSquaresApp : public CWinApp
{
public:
    CSquaresApp();

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSquaresApp)
    public:
    virtual BOOL InitInstance();
    //}}AFX_VIRTUAL

// Implementation
    //{{AFX_MSG(CSquaresApp)
    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 Visual C++ will insert additional declarations
// immediately before the previous line.

# endif
// !defined(
//  AFX_SDISQUARES_H__00156CE5_BB17_11D2_A2FD_0000861BAE71__INCLUDED_)
```

SdiSquares.cpp

```
// SdiSquares.cpp : Defines the class behaviors for the application.
//
# include "stdafx.h"
# include "SdiSquares.h"
```

```
#include "MainFrm.h"
#include "SquaresDoc.h"
#include "SquaresView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSquaresApp

BEGIN_MESSAGE_MAP(CSquaresApp, CWinApp)
    //{{AFX_MSG_MAP(CSquaresApp)
    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)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSquaresApp construction

CSquaresApp::CSquaresApp()
{
    // TODO: add construction code here,
    // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CSquaresApp object

CSquaresApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CSquaresApp initialization

BOOL CSquaresApp::InitInstance()
{
    // 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.
```

```
    // Change the registry key under which our settings are stored.
    // TODO: 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(CSquaresDoc),
        RUNTIME_CLASS(CMainFrame),       // main SDI frame window
        RUNTIME_CLASS(CSquaresView));
    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_SHOW);
    m_pMainWnd->UpdateWindow();

    // Enable drag/drop open
    m_pMainWnd->DragAcceptFiles();

    return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg 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 CSquaresApp::OnAppAbout()
{
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
}

/////////////////////////////////////////////////////////////////////////////
// CSquaresApp message handlers
```

MainFrm.h

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(
    AFX_MAINFRM_H__00156CE9_BB17_11D2_A2FD_0000861BAE71__INCLUDED_)
#define AFX_MAINFRM_H__00156CE9_BB17_11D2_A2FD_0000861BAE71__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
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMainFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL
// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

// Generated message map functions
protected:
    //{{AFX_MSG(CMainFrame)
      // 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 Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(AFX_MAINFRM_H__00156CE9_BB17_11D2_A2FD_0000861BAE71__INCLUDED_)
```

MainFrm.cpp

```
// MainFrm.cpp : implementation of the CMainFrame class
//

#include "stdafx.h"
#include "SdiSquares.h"

#include "MainFrm.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)
        // 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()

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
    // TODO: add member initialization code here

    }
```

```
CMainFrame::CMainFrame()
{
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
    if( !CFrameWnd::PreCreateWindow(cs) )
        return FALSE;
    // TODO: Modify the Window class or styles here by modifying
    //  the CREATESTRUCT cs

    return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}

#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers
```

SquaresDoc.h

```
// SquaresDoc.h : interface of the CSquaresDoc class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(
    AFX_SQUARESDOC_H__00156CEB_BB17_11D2_A2FD_0000861BAE71__H__INCLUDED_ )
#define AFX_SQUARESDOC_H__00156CEB_BB17_11D2_A2FD_0000861BAE71__H__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif //_MSC_VER > 1000

class CSquaresDoc : public CDocument
```

```
|
protected: // create from serialization only
    CSquaresDoc();
    DECLARE_DYNCREATE(CSquaresDoc)

// Attributes
public:

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSquaresDoc)
    public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    //}}AFX_VIRTUAL

// Implementation
public:
    void SetSquare (int i, int j, COLORREF color);
    COLORREF GetSquare (int i, int j);
    COLORREF GetCurrentColor();
    virtual CSquaresDoc();
#ifdef_DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    COLORREF m_clrCurrentColor;
    COLORREF m_clrGrid[4][4];
    //{{AFX_MSG(CSquaresDoc)
    afx_msg void OnColorRed();
    afx_msg void OnColorYellow();
    afx_msg void OnColorGreen();
    afx_msg void OnColorCyan();
    afx_msg void OnColorBlue();
    afx_msg void OnColorWhite();
    afx_msg void OnUpdateColorRed(CCmdUI * pCmdUI);
    afx_msg void OnUpdateColorYellow(CCmdUI * pCmdUI);
    afx_msg void OnUpdateColorGreen(CCmdUI * pCmdUI);
```

```
    afx_msg void OnUpdateColorCyan(CCmdUI * pCmdUI);
    afx_msg void OnUpdateColorBlue(CCmdUI * pCmdUI);
    afx_msg void OnUpdateColorWhite(CCmdUI * pCmdUI);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(
//     AFX_SQUARESDOC_H__00156CBB_BB17_11D2_A2FD_0000861BAE71__INCLUDED_)
```

SquaresDoc.cpp

```
// SquaresDoc.cpp : implementation of the CSquaresDoc class
//
#include "stdafx.h"
#include "Sdi Squares.h"

#include "SquaresDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CSquaresDoc

IMPLEMENT_DYNCREATE(CSquaresDoc, CDocument)

BEGIN_MESSAGE_MAP(CSquaresDoc, CDocument)
    //{{AFX_MSG_MAP(CSquaresDoc)
    ON_COMMAND(ID_COLOR_RED, OnColorRed)
    ON_COMMAND(ID_COLOR_YELLOW, OnColorYellow)
    ON_COMMAND(ID_COLOR_GREEN, OnColorGreen)
    ON_COMMAND(ID_COLOR_CYAN, OnColorCyan)
    ON_COMMAND(ID_COLOR_BLUE, OnColorBlue)
    ON_COMMAND(ID_COLOR_WHITE, OnColorWhite)
    ON_UPDATE_COMMAND_UI(ID_COLOR_RED, OnUpdateColorRed)
```

```
    ON_UPDATE_COMMAND_UI(ID_COLOR_YELLOW, OnUpdateColorYellow)
    ON_UPDATE_COMMAND_UI(ID_COLOR_GREEN, OnUpdateColorGreen)
    ON_UPDATE_COMMAND_UI(ID_COLOR_CYAN, OnUpdateColorCyan)
    ON_UPDATE_COMMAND_UI(ID_COLOR_BLUE, OnUpdateColorBlue)
    ON_UPDATE_COMMAND_UI(ID_COLOR_WHITE, OnUpdateColorWhite)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSquaresDoc construction/destruction

CSquaresDoc::CSquaresDoc()
{
    // TODO: add one-time construction code here

}

CSquaresDoc::~CSquaresDoc()
{
}

BOOL CSquaresDoc::OnNewDocument()
{
    if (!CDocument::OnNewDocument())
        return FALSE;

    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
            m_clrGrid[i][j] = RGB(255, 255, 255);

    m_clrCurrentColor = RGB(255, 0, 0);
    return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CSquaresDoc serialization

void CSquaresDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        for (int i = 0; i < 4; i++)
            for (int j = 0; j < 4; j++)
                ar << m_clrGrid[i][j];
        ar << m_clrCurrentColor;
    }
    else
    {
```

```
        for (int i = 0; i < 4; i ++)
            for (int j = 0; j < 4; j ++)
                ar >> m_clrGrid[i][j];
        ar >> m_clrCurrentColor;
    }
}

/////////////////////////////////////////////////////////////////////////////
// CSquaresDoc diagnostics

#ifdef _DEBUG
void CSquaresDoc::AssertValid() const
{
    CDocument::AssertValid();
}
void CSquaresDoc::Dump(CDumpContext& dc) const
{
    CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CSquaresDoc commands

COLORREF CSquaresDoc::GetCurrentColor()
{
    return m_clrCurrentColor;
}

COLORREF CSquaresDoc::GetSquare(int i, int j)
{
    ASSERT (i >= 0 && i <= 3 && j >= 0 && j <= 3);
    return m_clrGrid[i][j];
}

void CSquaresDoc::SetSquare(int i, int j, COLORREF color)
{
    ASSERT (i >= 0 && i <= 3 && j >= 0 && j <= 3);
    m_clrGrid[i][j] = color;
    SetModifiedFlag (TRUE);
    UpdateAllViews (NULL);
}

void CSquaresDoc::OnColorRed()
{
    m_clrCurrentColor = RGB (255, 0, 0);
}
```

```
void CSquaresDoc::OnColorYellow()
{
    m_clrCurrentColor = RGB (255, 255, 0);
}

void CSquaresDoc::OnColorGreen()
{
    m_clrCurrentColor = RGB (0, 255, 0);
}

void CSquaresDoc::OnColorCyan()
{
    m_clrCurrentColor = RGB (0, 255, 255);
}

void CSquaresDoc::OnColorBlue()
{
    m_clrCurrentColor = RGB (0, 0, 255);
}

void CSquaresDoc::OnColorWhite()
{
    m_clrCurrentColor = RGB (255, 255, 255);
}

void CSquaresDoc::OnUpdateColorRed(CCmdUI * pCmdUI)
{
    pCmdUI->SetRadio (m_clrCurrentColor == RGB (255, 0, 0));
}

void CSquaresDoc::OnUpdateColorYellow(CCmdUI * pCmdUI)
{
    pCmdUI->SetRadio (m_clrCurrentColor == RGB (255, 255, 0));
}

void CSquaresDoc::OnUpdateColorGreen(CCmdUI * pCmdUI)
{
    pCmdUI->SetRadio (m_clrCurrentColor == RGB (0, 255, 0));
}

void CSquaresDoc::OnUpdateColorCyan(CCmdUI * pCmdUI)
{
    pCmdUI->SetRadio (m_clrCurrentColor == RGB (0, 255, 255));
}

void CSquaresDoc::OnUpdateColorBlue(CCmdUI * pCmdUI)
{
```

```
    pCmdUI->SetRadio(m_clrCurrentColor == RGB(0, 0, 255));
}

void CSquaresDoc::OnUpdateColorWhite(CCmdUI* pCmdUI)
{
    pCmdUI->SetRadio(m_clrCurrentColor == RGB(255, 255, 255));
}
```

SquaresView.h

```
// SquaresView.h : interface of the CSquaresView class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(
    AFX_SQUARESVIEW_H__00156CED_BB17_11D2_A2FD_0000861BAE71__INCLUDED_)
# define AFX_SQUARESVIEW_H__00156CED_BB17_11D2_A2FD_0000861BAE71__IN-
CLUDED_

#if_MSC_VER > 1000
#pragma once
#endif // MSC_VER > 1000

class CSquaresView : public CView
{
protected: // create from serialization only
    CSquaresView();
    DECLARE_DYNCREATE(CSquaresView)

// Attributes
public:
    CSquaresDoc* GetDocument();

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSquaresView)
    public:
    virtual void OnDraw(CDC* pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    //}}AFX_VIRTUAL

// Implementation
public:
```

```
    virtual CSquaresView();
#ifdef_DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:
// Generated message map functions
protected:
    //{{AFX_MSG(CSquaresView)
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifndef_DEBUG  // debug version in SquaresView.cpp
inline CSquaresDoc* CSquaresView::GetDocument()
   { return (CSquaresDoc*)m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
// immediately before the previous line.

#endif
// !defined(
//         AFX_SQUARESVIEW_H__00156CED_BB17_11D2_A2FD_0000861BAE71__H__IN-
CLUDED_)
```

SquaresView.cpp

```
// SquaresView.cpp : implementation of the CSquaresView class
//

#include "stdafx.h"
#include "SdiSquares.h"

#include "SquaresDoc.h"
#include "SquaresView.h"

#ifdef_DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

```
/////////////////////////////////////////////////////////////////////////////
// CSquaresView
IMPLEMENT_DYNCREATE(CSquaresView, CView)

BEGIN_MESSAGE_MAP(CSquaresView, CView)
    //{{AFX_MSG_MAP(CSquaresView)
    ON_WM_LBUTTONDOWN()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CSquaresView construction/destruction

CSquaresView::CSquaresView()
{
    // TODO: add construction code here

}

CSquaresView::~CSquaresView()
{
}

BOOL CSquaresView::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
    //  the CREATESTRUCT cs

    return CView::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CSquaresView drawing

void CSquaresView::OnDraw(CDC* pDC)

    CSquaresDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    //
    // Set the mapping mode to MM_LOENGLISH.
    //
    pDC->SetMapMode(MM_LOENGLISH);

    //
    // Draw the 16 squares.
    //
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
```

```
            COLORREF color = pDoc->GetSquare(i, j);
            CBrush brush(color);
            int x1 = (j * 100) + 50;
            int y1 = (i * -100) - 50;
            int x2 = x1 + 100;
            int y2 = y1 - 100;
            CRect rect(x1, y1, x2, y2);
            pDC->FillRect(rect, &brush);
        }
    }

    //
    // Then the draw the grid lines surrounding them.
    //
    for (int x = 50; x <= 450; x += 100) {
        pDC->MoveTo(x, -50);
        pDC->LineTo(x, -450);
    }

    for (int y = -50; y >= -450; y -= 100) {
        pDC->MoveTo(50, y);
        pDC->LineTo(450, y);
    }
}

/////////////////////////////////////////////////////////////////////////////
// CSquaresView diagnostics

#ifdef _DEBUG
void CSquaresView::AssertValid() const
{
    CView::AssertValid();
}

void CSquaresView::Dump(CDumpContext& dc) const
{
    CView::Dump(dc);
}

CSquaresDoc* CSquaresView::GetDocument() // non-debug version is inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CSquaresDoc)));
    return (CSquaresDoc*)m_pDocument;
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
```

```
// CSquaresView message handlers

void CSquaresView::OnLButtonDown(UINT nFlags, CPoint point)
{
    CView::OnLButtonDown(nFlags, point);

    //
    // Convert to click coordinates to MM_LOENGLISH units.
    //
    CClientDC dc(this);
    dc.SetMapMode(MM_LOENGLISH);
    CPoint pos = point;
    dc.DPtoLP(&pos);

    //
    // If a square was clicked, set its color to the current color.
    //
    if (pos.x >= 50 && pos.x <= 450 && pos.y <= -50 && pos.y >= -450) {
        int i = (-pos.y - 50) / 100;
        int j = (pos.x - 50) / 100;
        CSquaresDoc* pDoc = GetDocument();
        COLORREF clrCurrentColor = pDoc->GetCurrentColor();
        pDoc->SetSquare(i, j, clrCurrentColor);
    }
}
```

图 9-4 SdiSpuares 应用程序

9.2.2 循序渐进地创建 SdiSquares

虽然理解 SdiSquares 的运行过程很重要，但掌握创建它的过程也很重要。当使用 AppWizard 和 ClassWizard 精心制作 MFC 应用程序时，向导会为您编写一部分代码，而您编写剩下的部分。但还有制作过程的问题。虽然我可以不去详细介绍创建 SdiSquares 的每个细节，但如果我对起码的整个创建过程都不介绍那就是一种失职。因此，下面讲述的就是循序渐进地生成 SdiSquares 的过程以及您自己创建应用程序时可以采用的方法。

1. 使用 AppWizard 创建一个新的工程名为 SdiSquares。在 AppWizard 的 Step 1 对话框中，选择 Single Document 作为应用程序类型并选中复选框 Document/View Architecture Support，如图 9-5 所示，在 Step 3 对话框中，取消对 ActiveX Controls 的选择。在 Step 4 中，取消对 Docking Toolbar、Initial Status Bar、Printing And Print Preview 以及 3D Controls 的选择。并且在 Step 4 对话框中单击 Advanced 按钮，并在 File Extension 编

辑框中输入字符 sqr (如图 9-6 所示)来定义 SdiSquares 文档的默认文件扩展名。在 Step 6 对话框中,手动编辑类名与图 9-4 中的类名匹配。在其他任何地方,都接受 AppWizard 的默认设置。

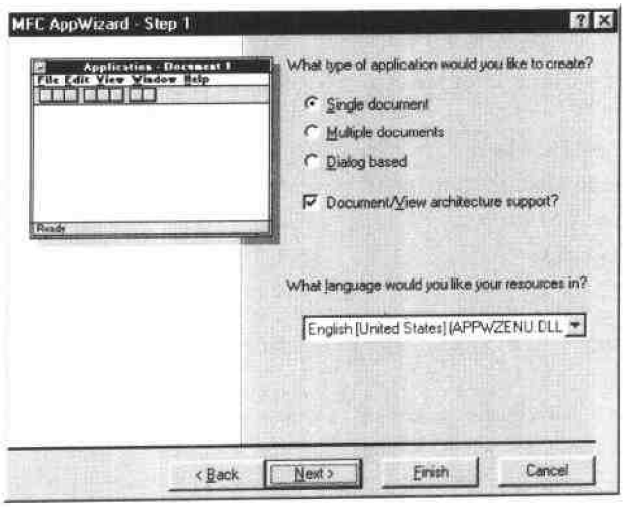

图 9-5 用 AppWizard 创建 SDI 文档/视图应用程序

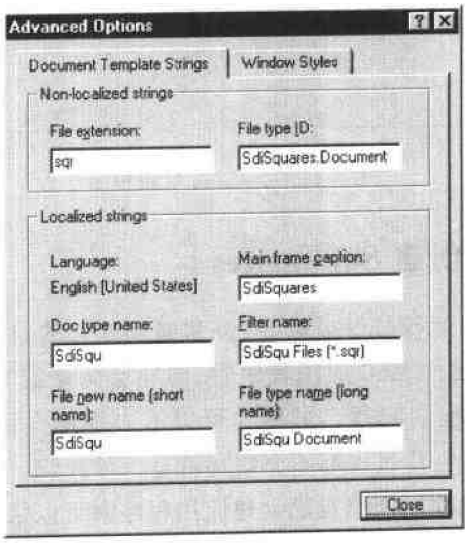

图 9-6 为 SdiSquares 文档指定默认文件扩展名

2. 给文档类添加成员变量 m_clrGrid 和 m_clrCurrentColor,并在 OnNewDocument 中添加代码来初始化它们。AppWizard 覆盖了 OnNewDocument,因此您需要做的只是添加初始化数据成员的语句。

3. 给文档类添加成员函数 GetCurrentColor、GetSquare 以及 SetSquare。确保它们为公用成员函数，以便视图可以访问它们。
4. 修改 AppWizard 包含在文档类中的 Serialize 函数来串行化 m_clrGrid 和 m_clrCurrentColor。
5. 实现视图的 OnDraw 函数。AppWizard 生成一个什么也不做的 OnDraw 函数；您可以编写程序来执行应用程序指定的任务。
6. 给视图添加 WM_LBUTTONDOWN 处理程序（OnLButtonDown）。可以手工也可以使用 ClassWizard 来添加消息处理程序。我使用了 ClassWizard。
7. 打开 AppWizard 生成的应用程序的菜单进行编辑，删除 Edit 菜单并添加 Color 菜单。然后为新菜单项编写命令和更新处理程序。就消息处理程序而言，既可以手工也可以在 ClassWizard 的帮助下添加命令和更新处理程序。我又一次使用了 ClassWizard。

最后，在圆满地结束编程以前，您还可以编辑应用程序的图标。AppWizard 在创建工程时生成两个图标。IDR_MAINFRAME 是应用程序图标，出现在框架窗口的标题栏。IDR_SDISQUTYPE 是应用程序的“文档图标”，用来在操作系统命令解释器中代表 SdiSquares 文档文件。在 InitInstance 调用 RegisterShellFileTypes 时文档图标要向系统注册。

9.3　文档 + 视图 = 较少的工作量

在使用 SdiSquares 时，会注意到 MFC 提供了许多应用程序的基本功能。例如：文档可以被打开和保存，即使我们仅仅添加了 8 行代码来支持此操作。如果在修改了方格颜色之后，想打开另一个文档或退出应用程序，就会出现一个消息框询问您是否愿意保存先前的修改。在操作系统命令解释器中双击一个 SdiSquares 文档文件，SdiSquares 就会自动运行并打开该文档。由于使用文档和视图来创建应用程序，所以 MFC 提供了如此众多的功能特性。在本章稍后您将看到使用文档/视图体系结构的其他优点。

我曾经第一次看到由 AppWizard 生成的最小的 MFC 应用程序时，就被相当少的程序代码惊呆了。当时我没有意识到，是主结构通过简单明了的消息映射输入项提供了整块的应用程序，消息映射表输入项如下：

```
ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileNew)
```

而且程序其他部分（特别是 File 菜单中的 Save 和 Save As 命令）也是由主结构实现的，但是在消息映射表输入项中却看不到，因为消息映射是在基类中进行的。所有这一切，就好像有魔法在起作用，我明白如果想完全理解文档/视图机制的话就必须深入研究下去。

正如我很快发现的那样,在文档/视图体系结构中并不存在什么魔法,只是一些隐藏在预处理宏中的巧妙编程,以及成千上万行用来处理常规(和非常规)杂务的程序,这些杂务包括:在框架窗口缩放之后缩放一个视图、执行与命令解释器的 DDE 交互等。因为没有花时间对 AppWizard 生成的代码进行深入的研究,所以许多程序员都没有把握它的总体结构。SdiSquares 是一个最基本的文档/视图应用程序,由于没有不必要的附加程序代码,所以它易于理解。如果理解了 SdiSquares,那您就已经开始理解文档/视图体系结构了。

第 10 章　滚动视图、HTML 视图以及其他视图类型

MFC 的 CView 类定义了视图的基本功能，它只是 MFC 供您支配的几个视图类之一。相关的类如 CScrollView、CTreeView 和 CHtmlView，都是直接或间接从 CView 派生来的，它们提供了您或许会用到的附加功能，您可以把它们作为创建自己的视图类的基类。表 10-1 中列出了 MFC 6.0 和以后版本中有效的视图类。

在第 9 章中介绍了 CView，在 SdiSquares 中它被用作视图的基类。在本章中我们将看看其他 MFC 提供的视图类，并通过研究实际的示例程序来说明它们的用法。首先是 CScrollView，除了 CView 以外，它可能就是 MFC 程序员最常使用的视图类了。

表 10-1　MFC 视图类

类名	说　明
CView	所有视图类的基类
CCtrlView	CEditView、CRichEditView、CListView 和 CTreeView 类的基类，用来创建基于其他控件类型的视图类
CEditView	封装了多行编辑控件，并添加了打印、查找以及查找替换功能
CRichEditView	封装了多功能编辑控件
CListView	封装了列表视图控件
CTreeView	封装了树视图控件
CHtmlView	从 HTML 文件和其他 Microsoft Internet Explorer WebBrowser 控件支持的媒体创建视图
CScrollView	给 CView 添加了滚动功能，是 CFormView 的基类
CFormView	实现可滚动的从对话框模板创建的“窗体”视图
CRecordView	CFormView 的派生类，用来显示从 ODBC 数据库获得的记录
CDaoRecordView	CRecordView 的 DAO 版本
COleDBRecordView	CRecordView 的 OLE DB 版本

10.1　滚动视图

CScrollView 给 CView 添加了基本的滚动功能。它包含 WM_VSCROLL 和 WM_HSCROLL 消息的处理程序，让 MFC 担负响应滚动条消息所要做的涉及滚动窗口的大量工作。它还包含一些成员函数，可以用来执行一些基本任务，如滚动到指定位置和检索当前滚

动到的位置。CScrollView 完全靠自己处理滚动任务,所以除了实现 OnDraw 以外可以对它不做任何工作。通常,在 CScrollView 中实现 OnDraw 的方法与在 CView 中完全相同。除非希望有效使用它优化滚动操作,否则 OnDraw 基本不需要特殊的逻辑来支持滚动。

10.1.1 CScrollView 基础

使用 CScrollView 创建滚动视图本身很简单。这里给出 3 个基本步骤。术语"物理视图"指的是占据屏幕的视图窗口和空间,"逻辑视图"指的是可以使用滚动条看到的虚拟工作空间。

1. 从 CScrollView 派生应用程序的视图类。如果使用 AppWizard 创建工程,可以从 AppWizard的 Step 6 对话框提供的基类列表中选择 CScrollView,如图 10-1 所示。

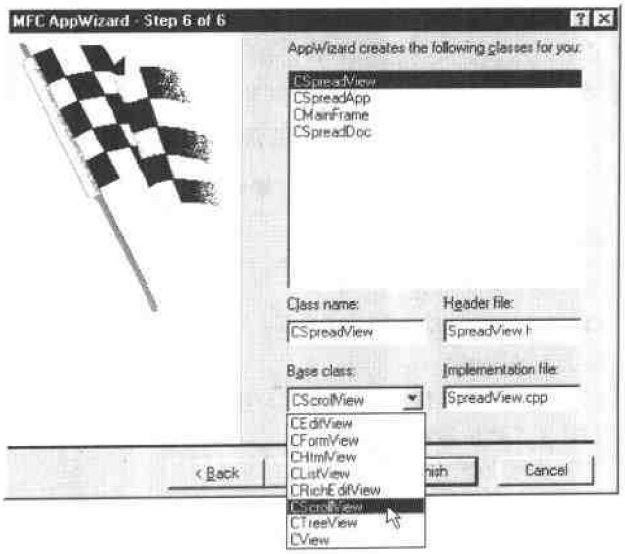

图 10-1 使用 AppWizard 吹建基于 CScrollView 的应用程序

2. 在视图类中覆盖 OnInitialUpdate,并调用 SetScrollSizes 来指定视图的逻辑尺寸。通过此方式可以告诉 MFC 可滚动视图占据的区域大小。如果使用 AppWizard 创建工程并在 Step 6 对话框中选择了 CScrollView,那么 AppWizard 就会为您覆盖 OnInitialUpdate并插入对 SetScrollSizes 的调用,将视图的逻辑宽度和高度设置为 100 像素。
3. 把视图当作常规 CView 来实现 OnDraw。

用这种方法创建的滚动视图可以自动滚动来响应滚动条事件。它会自动在 OnDraw 的输出中考虑滚动到的位置。如果物理视图尺寸超出逻辑视图尺寸,它就会隐藏滚动条,并且在滚动条可见时会自动调整滚动条滑块的大小来反映物理和逻辑视图的相对比例大小。

CScrollView::SetScrollSizes 接受 4 个参数，其中 2 个可选。这些参数按顺序是：

- 指定映射模式的整数(必要)
- 指定视图逻辑尺寸的 SIZE 结构或 CSize 对象(必要)
- SIZE 结构或 CSize 对象，用来指定“页尺寸”，即单击滚动轴时 MFC 对视图的滚动量(可选)
- SIZE 结构或 CSize 对象，用来指定行尺寸，即单击滚动条箭头时 MFC 对视图的滚动量(可选)

如果忽略了最后 2 个参数，那么 MFC 会使用适合实际情况的默认值来设置页尺寸和行尺寸。下列 OnInitialUpdate 函数将逻辑视图尺寸设置为 1 280 像素宽和 1 024 像素高：

```
void CMyView::OnInitialUpdate()
{
   CScrollView::OnInitialUpdate();
   SetScrollSizes(MM_TEXT, CSize(1280, 1024));
}
```

下列代码将视图尺寸设置为 8½ × 11 英寸大小：

```
void CMyView::OnInitialUpdate()
{
   CScrollView::OnInitialUpdate();
   SetScrollSizes(MM_LOENGLISH, CSize(850, 1100));
}
```

下列代码与上述代码功能相同，而且还设置了在响应 SB_PAGEUP/DOWN/LEFT/RIGHT 时视图滚动 2 英寸，在响应 SB_LINEUP/DOWN/LEFT/RIGHT 事件时滚动 1/4 英寸的功能：

```
void CMyView::OnInitialUpdate()
{
   CScrollView::OnInitialUpdate();
   SetScrollSizes(MM_LOENGLISH, CSize(850, 1100),
      CSize(200, 200), CSize(25, 25));
}
```

在 SetScrollSizes 的第 1 个参数中指定的映射模式确定了测量第 2、第 3 和第 4 个参数的单位。可以指定除了 MM_ISOTROPIC 和 MM_ANISOTROPIC 以外的任何模式。当调用 OnDraw 时，映射模式已经设置成了在 SetScrollSizes 调用中指定的模式。所以实现 OnDraw 时不必再亲自调用 SetMapMode 了。

难道这就是创建滚动视图所要做的一切工作吗？差不多。在使用 CScrollView 时应该记住两个原则：

- 如果在视图中 OnDraw 函数之外绘制输出，就要调用 CScrollView::OnPrepareDC 让 MFC 在输出中考虑映射模式和滚动位置的影响。

- 如果响应鼠标消息时执行命中测试,就要使用 CDC::DPtoLP 将单击处的坐标从设备坐标转换为逻辑坐标,从而在命中测试中考虑到映射模式和滚动位置的影响。

有关 CScrollView 工作方式的一些背景知识可以帮助我们理解这些原则的重要性,也可以使我们明白,为什么一个对滚动一无所知的普通 OnDraw 函数,在成为 CScrollView 的一员之后,就居然神奇般地可以根据当前滚动位置调整自己的输出了。

当滚动事件发生时,CScrollView 就用 OnVScroll 或 OnHScroll 消息处理程序捕获随后的消息,并调用::ScrollWindow 来水平或垂直滚动视图。稍后,视图的 OnPaint 函数将被调用,以绘制由::ScrollWindow 造成的失效窗口的一部分。以下是 CScrollView 从 CView 继承来的 OnPaint处理程序:

```
CPaintDC dc(this);
OnPrepareDC(&dc);
OnDraw(&dc);
```

在调用 OnDraw 之前,CView::OnPaint 会调用虚 OnPrepareDC 函数。CScrollView 覆盖 OnPrepareDC并在其中调用 CDC::SetMapMode 来设置映射模式,调用 CDC::SetViewportOrg 将视口原点转换为等于水平和垂直滚动位置的量。因此,在 OnDraw 重绘视图时滚动位置就被自动考虑进去了。要感谢 CScrollView::OnPrepareDC,它使得一个一般的从 CView 移植给 CScrollView 的函数可以自动适应滚动位置的修改。

现在让我们考虑一下,如果您自己在视图的 OnDraw 函数外部实例化了一个设备描述表类,并在 CScrollView 中执行绘制操作,这时会有什么情况发生呢?除非首先像 OnPaint 那样调用 OnPrepareDC 来准备设备描述表,否则 SetViewportOrg 不会被调用,并且绘制操作将相对于物理视图的左上角而不是逻辑视图的左上角被执行。如果使用两种不同的坐标系统绘制输出,文档的视图很快就混乱了。所以,当如下所示在 OnDraw 以外执行 CScrollView 窗口中的绘制操作时:

```
CClientDC dc (this);
// Draw something with dc.
```

要习惯性地首先将设备描述表传递给 OnPrepareDC,如下所示:

```
CClientDC dc (this);
OnPrepareDC (&dc);
// Draw something with dc.
```

出于同样原因,如果在 CScrollView 中有一个点的设备坐标,想使用 CDC::DPtoLP 将设备坐标转换为逻辑坐标来得到该点在逻辑视图中的相应位置,就要首先调用 OnPrepareDC 设置映射模式并将滚动位置因素考虑进去。下列 WM_LBUTTONDOWN 处理程序执行简单的命中测试,以确定单击位置处于逻辑视图的上半部还是下半部。

```
void CMyView::OnLButtonDown (UINT nFlags, CPoint point)
{
    CPoint pos = point;
    CClientDC dc (this);
    OnPrepareDC (&dc);
    dc.DPtoLP (&pos);

    CSize size = GetTotalSize ();
    if (::abs (pos.y) < (size.cy / 2)) {
        // Upper half was clicked.
    }
    else {
         // Lower half was clicked.
    }
}
```

传递给 OnLButtonDown 的 CPoint 对象和其他鼠标消息处理程序总是包含设备坐标，因此如果想知道逻辑视图空间中的相应点的坐标的话，坐标转换是必然的。

10.1.2　CScrollView 操作

CScrollView 包含一些成员函数，可以使用它们来操作滚动视图。例如，可以调用 GetScrollPosition 从 CScrollView 中检索当前水平或垂直滚动位置：

```
CPoint pos = GetScrollPosition ();
```

可以使用 ScrollToPosition 来滚动到给定位置：

```
ScrollToPosition (CPoint (100, 100));
```

还可以使用 GetTotalSize 测量视图的逻辑宽度和高度：

```
CSize size = GetTotalSize ();
int nWidth = size.cx;
int nHeight = size.cy;
```

CScrollView 中最有趣的一个成员函数是 SetScaleToFitSize。假设在您的应用程序中想实现 Zoom To Fit 命令，以将整个逻辑视图缩放在物理视图中。用 SetScaleToFitSize 就很容易实现：

```
SetScaleToFitSize (GetTotalSize ());
```

要恢复视图的默认可滚动形式，只要再一次调用 SetScrollSizes 就可以了。偶尔，您可能会在应用程序的整个生命周期中多次调用 SetScrollSizes 来动态地调整滚动参数。例如，如果随着文档中数据的增加，逻辑视图的尺寸也要增加，那么使用 SetScrollSizes 在每次文档增大时都增加视图的逻辑尺寸是非常合理的。

10.1.3 优化滚动操作

CScrollView 的设计思路是您编写的 OnDraw 程序不必以直接的方式考虑滚动位置的处理。所以,从 CView 借来的 OnDraw 函数通常在 CScrollView 中不需要修改就能使用。但是"能使用"和"能高效使用"是两回事。

CScrollView 对视图的 OnDraw 函数施加的压力要远比 CView 大,因为滚动会产生更多的 OnDraw 调用。通常由滚动条事件导致的 OnDraw 调用仅要求重绘几行像素点。如果 OnDraw 试图绘制整个视图,GDI 就会通过剪切无效矩形外的像素点来消除不必要的输出。但是剪切需要时间,结果是根据 OnDraw 试图绘制无效矩形外的内容而浪费的 CPU 周期的不同,滚动操作的效果可能很好,但也可能非常糟糕。

在获得一个滚动视图后,应该通过拖动滚动条滑块来检测它的操作效果。如果滚动效果能接受,那就成功了。但是如果不能接受(实际中更经常的是不能接受),就应该修改视图的OnDraw 函数,使它能标识出无效矩形,并尽可能地把绘制的像素限制在矩形范围内。

优化 OnDraw 的关键是 CDC 函数 GetClipBox。它在传递给 OnDraw 的设备描述表对象中被调用,GetClipBox 用无效矩形的逻辑坐标下的尺寸和位置来初始化 RECT 结构或 CRect 对象,如下所示:

```
CRect rect;
pDC -> GetClipBox (&rect);
```

用此方式初始化了的 CRect 可以区分需要重绘的视图部分。如何使用此信息则完全取决于应用程序。在下一小节的示例程序中,我们在可滚动视图中显示了一份电子表格,它将由 GetClipBox 返回的坐标转换为行列号,并使用此结果绘制那些落在(全部或部分)无效矩形内的单元。这只是一个例子,说明了如何使用 GetClipBox 通过消除不必要的输出而优化绘制操作。在以后的章节中还会看到其他例子。

10.1.4 ScrollDemo 应用程序

图 10-2 所示的 ScrollDemo 应用程序说明了许多以上各小节讨论的原则。ScrollDemo 显示了一张电子表格,它具有 26 列宽 99 行高。在电子表格中有一个"当前单元格",用浅蓝色加亮显示。用鼠标左键单击单元格可以使该单元格成为当前单元格并同时加亮显示。显示电子表格的可滚动视图是由 CScrollView 派生类 CScrollDemoView 定义的。在图 10-3 中给出了 CScrollDemoView 的源程序代码。

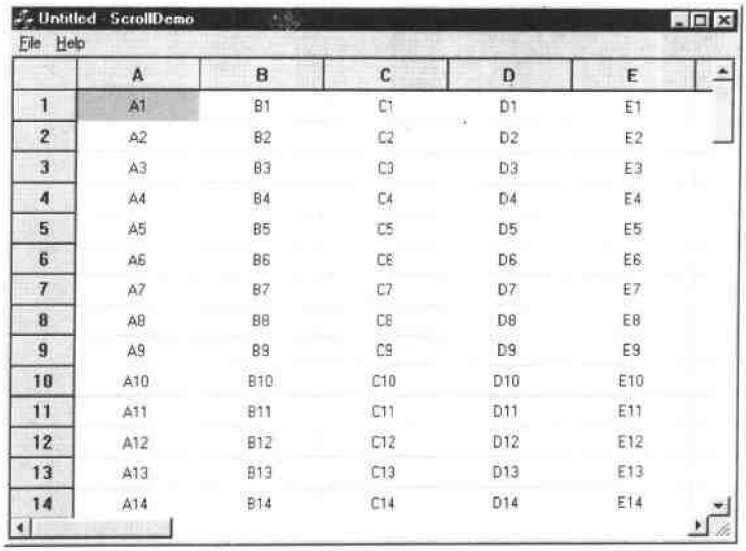

图 10-2　ScrollDemo 窗口

ScrollDemoView.h

```
// ScrollDemoView.h : interface of the CScrollDemoView class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_SCROLLDEMOVIEW_H__DCCF4E0D_9735_11D2_8E53_006008A82731
__INCLUDED_)
#define AFX_SCROLLDEMOVIEW_H__DCCF4E0D_9735_11D2_8E53_006008A82731__
INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif //_MSC_VER > 1000

class CScrollDemoView : public CScrollView
{
protected: // create from serialization only
    CScrollDemoView();
    DECLARE_DYNCREATE(CScrollDemoView)

// Attributes
public:
    CScrollDemoDoc* GetDocument();

// Operations
public:
```

```
// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CScrollDemoView)
    public:
    virtual void OnDraw(CDC * pDC);  // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    protected:
    virtual void OnInitialUpdate(); // called first time after construct
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CScrollDemoView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
protected:

// Generated message map functions
protected:
    BOOL m_bSmooth;
    void GetCellRect (int row, int col, LPRECT pRect);
    void DrawAddress (CDC * pDC, int row, int col);
    void DrawPointer (CDC * pDC, int row, int col, BOOL bHighlight);
    CFont m_font;
    int m_nCurrentCol;
    int m_nCurrentRow;
    int m_nRibbonWidth;
    int m_nCellHeight;
    int m_nCellWidth;
    //{{AFX_MSG(CScrollDemoView)
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifndef _DEBUG  // debug version in ScrollDemoView.cpp
inline CScrollDemoDoc * CScrollDemoView::GetDocument()
   { return (CScrollDemoDoc * )m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
// before the previous line.
```

```
#endif
// !defined(
//     AFX_SCROLLDEMOVIEW_H__DCCF4E0D_9735_11D2_8E53_006008A82731__
INCLUDED_)
```

ScrollDemoView.cpp

```
// ScrollDemoView.cpp : implementation of the CScrollDemoView class
//

#include "stdafx.h"
#include "ScrollDemo.h"
#include "ScrollDemoDoc.h"
#include "ScrollDemoView.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CScrollDemoView

IMPLEMENT_DYNCREATE(CScrollDemoView, CScrollView)

BEGIN_MESSAGE_MAP(CScrollDemoView, CScrollView)
    //{{AFX_MSG_MAP(CScrollDemoView)
    ON_WM_LBUTTONDOWN()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CScrollDemoView construction/destruction

CScrollDemoView::CScrollDemoView()
{
    m_font.CreatePointFont (80, _T ("MS Sans Serif"));
}

CScrollDemoView::~CScrollDemoView()
{
}

BOOL CScrollDemoView::PreCreateWindow(CREATESTRUCT& cs)
{
    return CScrollView::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CScrollDemoView drawing
```

```
void CScrollDemoView::OnDraw(CDC * pDC)
{
    CScrollDemoDoc * pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    //
    // Draw the grid lines.
    //
    CSize size = GetTotalSize();

    CPen pen(PS_SOLID, 0, RGB(192, 192, 192));
    CPen * pOldPen = pDC->SelectObject(&pen);
    for (int i = 0; i < 99; i++) {
        int y = (i * m_nCellHeight) + m_nCellHeight;
        pDC->MoveTo(0, y);
        pDC->LineTo(size.cx, y);
    }

    for (int j = 0; j < 26; j++) {
        int x = (j * m_nCellWidth) + m_nRibbonWidth;
        pDC->MoveTo(x, 0);
        pDC->LineTo(x, size.cy);
    }

    pDC->SelectObject(pOldPen);

    //
    // Draw the bodies of the rows and column headers.
    //
    CBrush brush;
    brush.CreateStockObject(LTGRAY_BRUSH);

    CRect rcTop(0, 0, size.cx, m_nCellHeight);
    pDC->FillRect(rcTop, &brush);
    CRect rcLeft(0, 0, m_nRibbonWidth, size.cy);
    pDC->FillRect(rcLeft, &brush);

    pDC->MoveTo(0, m_nCellHeight);
    pDC->LineTo(size.cx, m_nCellHeight);
    pDC->MoveTo(m_nRibbonWidth, 0);
    pDC->LineTo(m_nRibbonWidth, size.cy);

    pDC->SetBkMode(TRANSPARENT);

    //
    // Add numbers and button outlines to the row headers.
    //
    for (i = 0; i < 99; i++) {
```

```
        int y = (i * m_nCellHeight) + m_nCellHeight;
        pDC->MoveTo(0, y);
        pDC->LineTo(m_nRibbonWidth, y);

        CString string;
        string.Format(_T("%d"), i + 1);
        CRect rect(0, y, m_nRibbonWidth, y + m_nCellHeight);
        pDC->DrawText(string, &rect, DT_SINGLELINE|
            DT_CENTER|DT_VCENTER);

        rect.top++;
        pDC->Draw3dRect(rect, RGB(255, 255, 255),
            RGB(128, 128, 128));
    }

    //
    // Add letters and button outlines to the column headers.
    //
    for (j = 0; j < 26; j++)
        int x = (j * m_nCellWidth) + m_nRibbonWidth;
        pDC->MoveTo(x, 0);
        pDC->LineTo(x, m_nCellHeight);

        CString string;
        string.Format(_T("%c"), j + 'A');

        CRect rect(x, 0, x + m_nCellWidth, m_nCellHeight);
        pDC->DrawText(string, &rect, DT_SINGLELINE|
            DT_CENTER|DT_VCENTER);

        rect.left++;
        pDC->Draw3dRect(rect, RGB(255, 255, 255),
            RGB(128, 128, 128));
    }

    //
    // Draw address labels into the individual cells.
    //
    CRect rect;
    pDC->GetClipBox(&rect);
    int nStartRow = max(0, (rect.top - m_nCellHeight) / m_nCellHeight);
    int nEndRow = min(98, (rect.bottom - 1) / m_nCellHeight);
    int nStartCol = max(0, (rect.left - m_nRibbonWidth) / m_nCellWidth);
    int nEndCol = min(25, ((rect.right + m_nCellWidth - 1) -
        m_nRibbonWidth) / m_nCellWidth);

    for (i = nStartRow; i <= nEndRow; i++)
        for (j = nStartCol; j <= nEndCol; j++)
            DrawAddress(pDC, i, j);
```

```
    //
    // Draw the cell pointer.
    //
    DrawPointer (pDC, m_nCurrentRow, m_nCurrentCol, TRUE);
}

void CScrollDemoView::OnInitialUpdate()
{
    CScrollView::OnInitialUpdate();

    m_nCurrentRow = 0;
    m_nCurrentCol = 0;
    m_bSmooth = FALSE;

    CClientDC dc (this);
    m_nCellWidth = dc.GetDeviceCaps (LOGPIXELSX);
    m_nCellHeight = dc.GetDeviceCaps (LOGPIXELSY) / 4;
    m_nRibbonWidth = m_nCellWidth / 2;

    int nWidth = (26 * m_nCellWidth) + m_nRibbonWidth;
    int nHeight = m_nCellHeight * 100;
    SetScrollSizes (MM_TEXT, CSize (nWidth, nHeight));
}

////////////////////////////////////////////////////////////////////////////
// CScrollDemoView diagnostics

#ifdef _DEBUG
void CScrollDemoView::AssertValid() const
{
    CScrollView::AssertValid();
}

void CScrollDemoView::Dump(CDumpContext& dc) const
{
    CScrollView::Dump(dc);
}

CScrollDemoDoc* CScrollDemoView::GetDocument() // non-debug version is
                    inline
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CScrollDemoDoc)));
    return (CScrollDemoDoc*)m_pDocument;
}
#endif //_DEBUG

////////////////////////////////////////////////////////////////////////////
// CScrollDemoView message handlers

void CScrollDemoView::OnLButtonDown(UINT nFlags, CPoint point)
```

```
{
    CScrollView::OnLButtonDown(nFlags, point);
    //
    // Convert the click point to logical coordinates.
    //
    CPoint pos = point;
    CClientDC dc(this);
    OnPrepareDC(&dc);
    dc.DPtoLP(&pos);

    //
    // If a cell was clicked, move the cell pointer.
    //
    CSize size = GetTotalSize();
    if (pos.x > m_nRibbonWidth && pos.x < size.cx &&
        pos.y > m_nCellHeight && pos.y < size.cy) {

        int row = (pos.y - m_nCellHeight) / m_nCellHeight;
        int col = (pos.x - m_nRibbonWidth) / m_nCellWidth;
        ASSERT(row >= 0 && row <= 98 && col >= 0 && col <= 25);

        DrawPointer(&dc, m_nCurrentRow, m_nCurrentCol, FALSE);
        m_nCurrentRow = row;
        m_nCurrentCol = col;
        DrawPointer(&dc, m_nCurrentRow, m_nCurrentCol, TRUE);
    }
}

void CScrollDemoView::DrawPointer(CDC* pDC, int row, int col,
    BOOL bHighlight)
{
    CRect rect;
    GetCellRect(row, col, &rect);
    CBrush brush(bHighlight ? RGB(0, 255, 255) :
        ::GetSysColor(COLOR_WINDOW));
    pDC->FillRect(rect, &brush);
    DrawAddress(pDC, row, col);
}

void CScrollDemoView::DrawAddress(CDC pDC, int row, int col)
{
    CRect rect;
    GetCellRect(row, col, &rect);

    CString string;
    string.Format(_T("%c%d"), col + _T('A'), row + 1);

    pDC->SetBkMode(TRANSPARENT);
```

```
    CFont * pOldFont = pDC -> SelectObject (&m_font);
    pDC -> DrawText (string, rect, DT_SINGLELINE|DT_CENTER|DT_VCENTER);
    pDC -> SelectObject (pOldFont);


void CScrollDemoView::GetCellRect(int row, int col, LPRECT pRect)

    pRect -> left = m_nRibbonWidth + (col * m_nCellWidth) + 1;
    pRect -> top = m_nCellHeight + (row * m_nCellHeight) + 1;
    pRect -> right = pRect -> left + m_nCellWidth - 1;
    pRect -> bottom = pRect -> top + m_nCellHeight - 1;
}
```

图 10-3 ScrollDemo 应用程序

由于 CScrollView 管理着滚动的大部分工作,所以在 ScrollDemo 中直接支持滚动操作的代码非常少。但是它使用了 GetClipBox 来优化 OnDraw 的操作。在每次调用时并没有试图去绘制所有 2 574 个电子表格单元格,OnDraw 将剪切框转换为起始和终止行列编号,并且只绘制落在这些范围内的单元。相关的代码在 OnDraw 的结尾处:

```
CRect rect;
pDC -> GetClipBox (&rect);
int nStartRow = max (0, (rect.top - m_nCellHeight) / m_nCellHeight);
int nEndRow = min (98, (rect.bottom - 1) / m_nCellHeight);
int nStartCol = max (0, (rect.left - m_nRibbonWidth) / m_nCellWidth);
int nEndCol = min (25, ((rect.right + m_nCellWidth - 1) -
    m_nRibbonWidth) / m_nCellWidth);

for (i = nStartRow; i <= nEndRow; i++)
    for (j = nStartCol; j <= nEndCol; j++)
        DrawAddress (pDC, i, j);
```

作为实验,可以修改 for 循环来绘制每个单元格:

```
for (i = 0; i < 99; i++)
    for (j = 0; j < 26; j++)
        DrawAddress (pDC, i, j);
```

然后滚动电子表格。很快您会发现在滚动视图中优化 OnDraw 为什么不仅仅是可选操作,而且是非常必要的。

可以对视图的 OnLButtonDown 函数进行另一个有趣的实验,该函数移动加亮显示单元格来响应鼠标单击。在使用传递给它的 CPoint 对象来确定单击处的行列编号之前,OnLButtonDown 用下列语句将 CPoint 的设备坐标转换为逻辑坐标:

```
CPoint pos = point;
CClientDC dc(this);
OnPrepareDC(&dc);
dc.DPtoLP(&pos);
```

要看看如果 OnLButtonDown 没有在 CScrollView 中考虑滚动位置的话会发生什么情况，可以删除对 DPtoLP 的调用，并在水平或垂直方向把电子表格滚动一小段距离，然后随便单击一下电子表格。

10.1.5　普通视图转换为滚动视图

如果您使用 AppWizard 生成了一个基于 CView 的应用程序，但后来又决定使用 CScrollView，那怎么办呢？不可以使用 MFC 向导将 CView 转换为 CScrollView，但是您可以依靠手工来实现转换。方法如下：

1. 查找视图的头文件和 CPP 文件，将所有出现的 CView 修改为 CScrollView，除了在函数参数列表中出现的 CView*。
2. 如果还没有覆盖的话就覆盖 OnInitialUpdate，并插入对 SetScrollSizes 的调用。

如果您进行了第 1 步而忘了第 2 步，很快在运行应用程序时就会知道，MFC 会对您执行断言提示。如果不知道视图的逻辑尺寸，MFC 就无法管理滚动视图。

10.2　HTML 视图

MFC 中功能最强大的一个新类是 CHtmlView，它将作为 Microsoft Internet Explorer 的核心和灵魂的 WebBrowser 控件转换为了一个完整独立的 MFC 视图。CHtmlView 可以显示 HTML 文档。您只要提供一个 URL，用来指定 Internet 上、内联网上或本地硬盘上的文档，CHtmlView 就会以 Internet Explorer 的方式显示该文档。以 WebBrowser 控件为基础，CHtmlView 继承了许多宝贵的财富，具有了丰富的附加功能，从通过简单的函数调用在历史列表中前进或后退的功能，到安排 Dynamic HTML（DHTML）文档的能力。CHtmlView 也是一个 Active Document 容器，这就是说用它可以显示由 Microsoft Word、Microsoft Excel 以及其他 Active Document 服务器创建的文档。它甚至可以显示硬盘上文件夹中的内容，像 Internet Explorer 那样。

由于 CHtmlView 包含大量的成员函数，它们给 WebBrowser 控件提供了 C++ 接口，所以它极其复杂。但是尽管如此，它却非常易于使用。只要使用几个成员函数，您就能创建在资源和功能方面与 Internet Explorer 媲美的应用程序。事实上，可以使用 CHtmlView 和其他 MFC 类（如 CToolBar）在几天内就开发出超过 Internet Explorer 的产品。Visual C++ 带有一个 MFC 示例程序 MFCIE 就说明了具体方法。如果您想尽快学点儿本事，可以在几分钟内生成一个基本的浏览器。要注意由于 CHtmlView 从 WebBrowser 控件中派生了大部分功能，又因为 WebBrowser 控件是 Internet Explorer 的一部分，所以使用 CHtmlView 的应用程序只能在安装了

Internet Explorer 4.0 或更高版本的系统上运行。

10.2.1 CHtmlView 操作

一个很好的开始学习 CHtmlView 的方法是先熟悉它的非虚拟成员函数,或称为操作。表 10-2 列出了大多数程序中最常用的一些操作。对于其他(还有许多)操作,可以参考 MFC 资料。

表 10-2 主要的 CHtmlView 操作

函数	说 明
GetBusy	表明是否在进行下载
GetLocationName	如果显示的是 HTML 页,则检索该页的标题;如果当前显示的是文件或文件夹,则检索文件或文件夹的名字
GetLocationURL	检索当前显示的资源的 URL,例如 http://www.microsoft.com/或 file://C:/HTML Files/Clock.htm
GoBack	转移到历史列表中的前一项
GoForward	转移到历史列表中的后一项
Navigate	显示指定 URL 处的资源
Refresh	重新加载当前显示的资源
Stop	停止加载资源

这些函数执行的操作对于熟悉 Internet Explorer 的人都是明白易懂的。例如,如果您正在编写一个浏览器,就可能会用这些单行命令处理程序在其中添加一些 Back、Forward、Refresh和 Stop 按钮:

```
// In CMyView's message map
ON_COMMAND (ID_BACK, OnBack)
ON_COMMAND (ID_FORWARD, OnForward)
ON_COMMAND (ID_REFRESH, OnRefresh)
ON_COMMAND (ID_STOP, OnStop)
      .
      .
      .
void CMyView::OnBack ()
{
   GoBack ();
}

void CMyView::OnForward ()
{
   GoForward ();
}
```

```
void CMyView::OnRefresh()
{
    Refresh();

void CMyView::OnStop()

    Stop();
}
```

WebBrowser 控件通过名为 IWebBrowser2 的 COM 接口对外展示了大量的功能。大多数非虚拟成员函数，包括这里给出的成员函数，基本上都是 C++ 对 IWebBrowser2 方法调用的封装。

当用户单击 HTML 文档中的超级链接时，CHtmlView 会自动跳转到相关联的 URL 上。可以使用 Navigate 函数通过编程来转移到其他 URL 处。语句

```
Navigate(_T("http://www.microsoft.com"));
```

将显示 Microsoft 的 Web 站点的主页。Navigate 也接受基于文件的 URL，例如语句

```
Navigate(_T("file://c:/my documents/budget.xls"));
```

将在 HTML 视图中显示 Excel 电子表格。之所以能如此，是因为 Excel 是一个 Active Document 服务器，但这必须要求 Excel 安装在宿主 PC 上。您也可以传递标识文件夹而不是文件的路径名：

```
Navigate(_T("file://c:/my documents"));
```

相关的 CHtmlView 函数 Navigate2 除了能执行 Navigate 的工作外，还能完成更多任务。由于它可以在传递路径名的地方接受指向 ITEMIDLIST 结构的指针，所以 Navigate2 可以用来访问处于命令解释器的名字空间中任何地方的对象。相反，Navigate 只能用于文件系统对象。

10.2.2　CHtmlView 可覆盖函数

CHtmlView 包含几个虚拟函数，可以在派生类中覆盖它们来获取有关 WebBrowser 控件和所显示资源状态的最新信息。表 10-3 中给出了几个这样的函数。

表 10-3　主要的 CHtmlView 可覆盖函数

函数	说　明
OnNavigateComplete2	在访问新的 URL 之后调用
OnBeforeNavigate2	在访问新的 URL 之前调用
OnProgressChange	用来更新下载状态
OnDownloadBegin	用来表明下载将开始

续表

函数	说 明
OnDownloadComplete	用来表明下载完成
OnTitleChange	在文档标题修改时调用
OnDocumentComplete	用来表明文档被成功下载

不幸的是,Visual C++ 文档对有关调用这些函数的原因或条件仅提供了一些概括性的信息。这就是程序副本更具有说明性的原因。下面给出了一个函数调用日志,这些调用发生在调用 CHtmlView::Navigate 向 home.microsoft.com 转移的过程中:

```
OnBeforeNavigate2 ("http://home.microsoft.com/")
OnDownloadBegin ()
OnProgressChange (100/10000)
OnProgressChange (150/10000)
OnProgressChange (150/10000)
OnProgressChange (200/10000)
OnProgressChange (250/10000)
OnProgressChange (300/10000)
OnProgressChange (350/10000)
OnProgressChange (400/10000)
OnProgressChange (450/10000)
OnProgressChange (500/10000)
OnProgressChange (550/10000)
OnDownloadComplete ()
OnDownloadBegin ()
OnProgressChange (600/10000)
OnProgressChange (650/10000)
OnProgressChange (700/10000)
OnProgressChange (750/10000)
OnProgressChange (800/10000)
OnProgressChange (850/10000)
OnProgressChange (900/10000)
OnProgressChange (950/10000)
OnProgressChange (1000/10000)
OnProgressChange (1050/10000)
OnProgressChange (1100/10000)
OnProgressChange (1150/10000)
OnProgressChange (1200/10000)
OnProgressChange (1250/10000)
OnProgressChange (131400/1000000)
OnTitleChange ("http://home.microsoft.com/")
OnNavigateComplete2 ("http://home.microsoft.com/")
OnTitleChange ("MSN.COM")
```

```
OnProgressChange (146500/1000000)
OnTitleChange ("MSN.COM")
OnProgressChange (158200/1000000)
OnProgressChange (286500/1000000)
OnProgressChange (452300/1000000)
OnTitleChange ("MSN.COM")
OnProgressChange (692800/1000000)
OnProgressChange (787000/1000000)
OnTitleChange ("MSN.COM")
    .
    .
    .
OnDownloadComplete ()
OnTitleChange ("MSN.COM")
OnDocumentComplete ("http://home.microsoft.com/")
OnProgressChange (0/0)
```

您可以清楚地看到调用 OnBeforeNavigate2 是为了表明 WebBrowser 控件的意图是转移到新的 URL 处，在连接建立之后才调用 OnNavigateComplete2，一旦网页全部下载完毕就调用 OnDocumentComplete。在它们之间调用 OnDownloadBegin 和 OnDownloadComplete 来标记单独各页的下载，调用 OnProgressChange 表示这些下载过程。OnProgressChange 接收两个参数：一个长整型用来指定已下载的字节数、一个长整型指定即将下载的字节数。用第 2 个参数除第一个参数再乘 100 得到的百分比可以用在进度栏或其他控件中。将第 1 个参数设置为 -1 或将两个参数都设置为 0 来调用 OnProgressChange 也可以表示下载完成。

与 Visual C++ 一同发售的 MFCIE 示例程序提供了一个如何使用这些函数的例子。它用 OnTitleChange 来更新在其标题栏中显示的文档标题，使用 OnBeforeNavigate2 来启动一个动画表示正在下载，还使用 OnDocumentComplete 停止下载并更新显示在地址栏中的 URL。本质上，它是使用 OnBeforeNavigate2 和 OnDocumentComplete 来标记文档下载的开始和结束，用 OnTitleChange 来显示从 HTML 中分析得到的标题。

10.2.3　在基于 CHtmlView 的应用程序中使用 DHTML

编写家用的专用浏览器是 CHtmlView 的一个很好的用途，但是 CHtmlView 还有其他多种功能。一些 MFC 开发者发现 CHtmlView 很有趣，因为用它可以编写瘦客户程序。“瘦客户程序”是一种应用程序，它派生了 HTML 程序、DHTML 程序或其他 Web 程序设计媒体中的全部或部分功能。对 DHTML 的详细讨论超出了本书的范围，但是一个示例程序可以说明 CHtmlView 和 DHTML 混合工作的方式。

假设您想编写一个 Windows 应用程序，它可以模拟数字时钟。一种方法是使用 Visual C++ 编写 MFC 时钟应用程序，另一种方法是创建基于 CHtmlView 的应用程序，运行 DHTML 脚本，脚本再运行时钟。后者的优点在于应用程序感觉上是由普通的 HTML 文件定义的。