Principle of Learning

- Tell me and I forgot.
- Show me and I remember.
- Involve me and I understand.

- 不聞不若聞之，
  聞之不若見之，
  見之不若知之，
  知之不若行之，
  學至乎行而止矣。
  ——荀子

- 古人學問無遺力，
  少壯工夫老始成。
  紙上得來終覺淺，
  絕知此事要躬行。
  ——陸游

Chapter 17

Working with Dialogs and Controls

Controls in a Dialog Box

File > Open > File

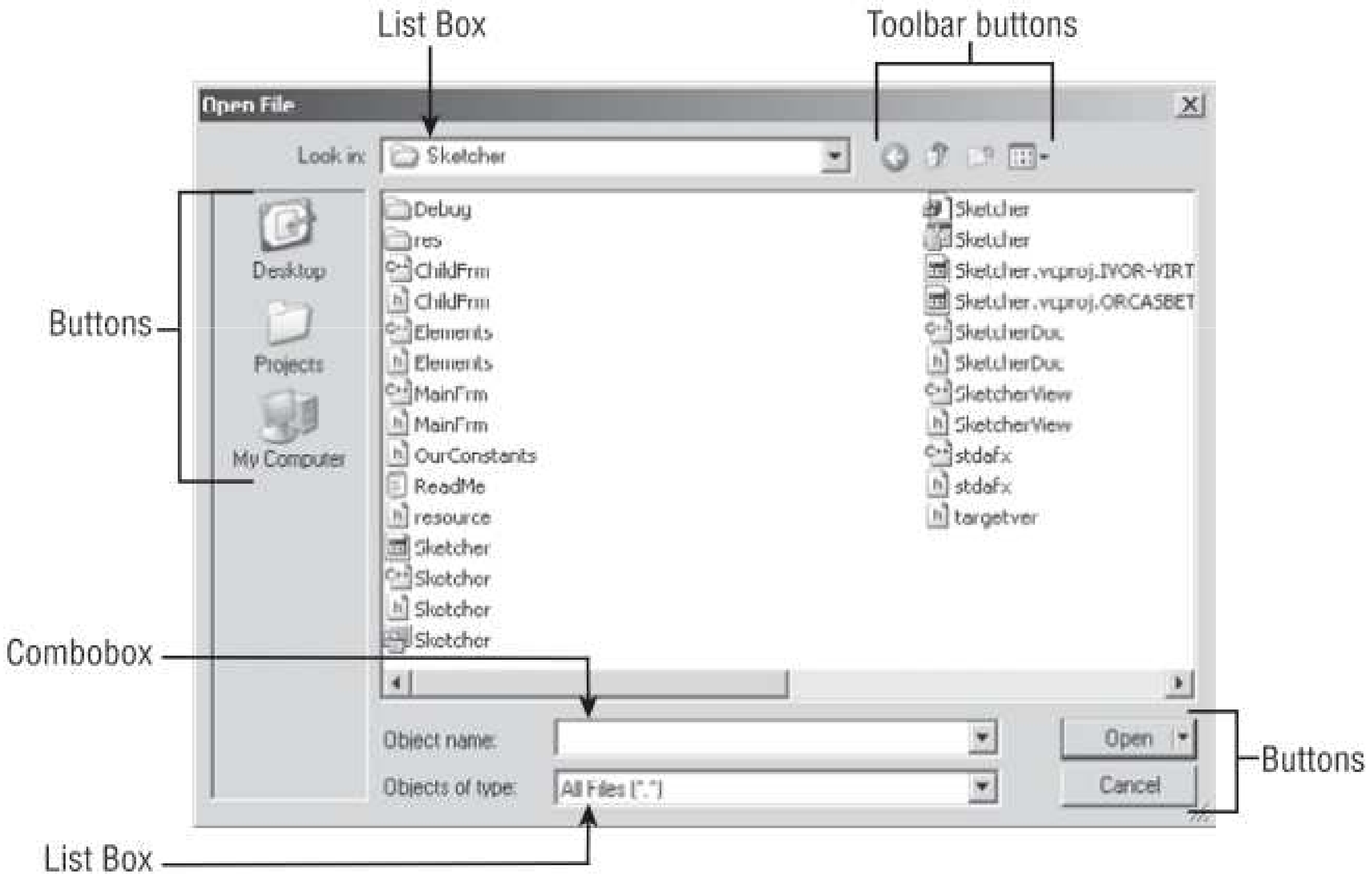
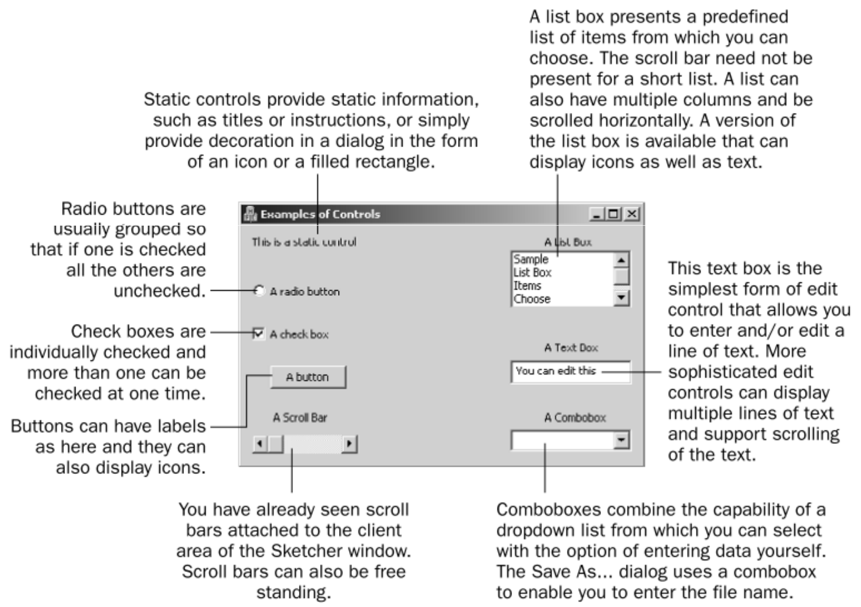


Figure 17-1

Figure 17.2 in P.987



Static controls provide static information, such as titles or instructions, or simply provide decoration in a dialog in the form of an icon or a filled rectangle.

A list box presents a predefined list of items from which you can choose. The scroll bar need not be present for a short list. A list can also have multiple columns and be scrolled horizontally. A version of the list box is available that can display icons as well as text.

Radio buttons are usually grouped so that if one is checked all the others are unchecked.

Check boxes are individually checked and more than one can be checked at one time.

Buttons can have labels as here and they can also display icons.

This text box is the simplest form of edit control that allows you to enter and/or edit a line of text. More sophisticated edit controls can display multiple lines of text and support scrolling of the text.

You have already seen scroll bars attached to the client area of the Sketcher window. Scroll bars can also be free standing.

Comboboxes combine the capability of a dropdown list from which you can select with the option of entering data yourself. The Save As... dialog uses a combobox to enable you to enter the file name.

Figure 17-2

- Static Controls
- Button Controls
- Scrollbars
- List Boxes
- Edit Controls
- Combo boxes

Dialogs

- A dialog box is actually a window which pops up when you click a menu item.
- A dialog box is a window.
- Each controls in a dialog is also a specialized window.
  - They are derived from CWnd.

- To create and display a dialog box in an MFC program:
  1. Define the physical appearance in a resource file
  2. Use a dialog class (CDialog) object to manage the operation of the dialog and its controls.

Creating a Dialog Resource

- Resource View
  - Right-click the Dialog folder
  - Insert Dialog
- The dialog has OK and Cancel button controls already in place.
- Adding new controls is easy.
  - You can drag the control from the toolbox.
  - You can click the control and then click in the dialog.
  - You can double-click the control.

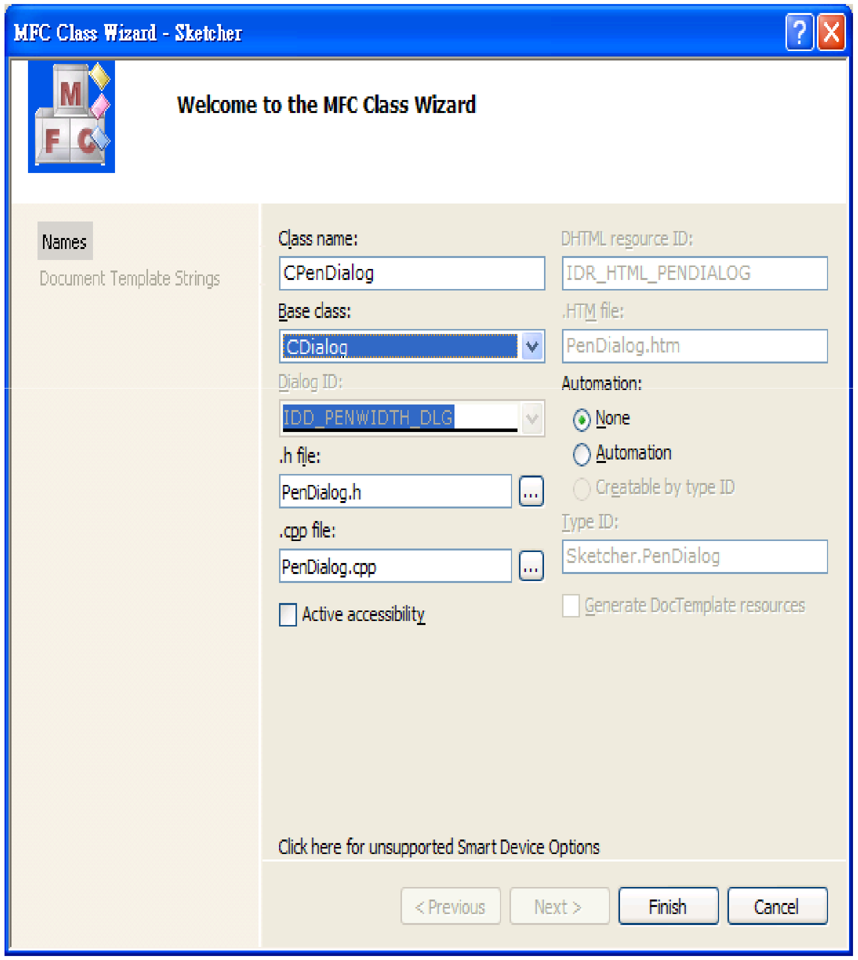
Properties of a Dialog

- Change the ID from `IDD_DIALOG1` to `IDD_PENWIDTH_DLG`
  - D stands for Dialog.
- Change the Caption property value to `Set Pen Width`.

- Ex17-1: Press Ctrl+T to display the dialog window.
  - You may click OK or Cancel buttons to close the dialog.

Adding a Dialog Class

- Right-click the dialog box
  - Select Add Class
  - Class name: CPenDialog
  - Base class: CDialog
- Figure 17-6 (P.991)


MFC Class Wizard - Sketcher
Welcome to the MFC Class Wizard
Names
Document Template Strings
< Previous
Next >
Finish
Cancel

Class name: CPenDialog

DHTML resource ID: IDR_HTML_PENDIALOG

Base class: CDialog

.HTM file: PenDialog.htm

Dialog ID: IDD_PENWIDTH_DLG

Automation: [✓ None] Automation Creatable by type ID

.h file: PenDialog.h

.cpp file: PenDialog.cpp

Type ID: Sketcher.PenDialog

Active accessibility

Generate DocTemplate resources

Click here for unsupported Smart Device Options

Displaying a Dialog (1)

- Modal
  - All other windows are suspended until the dialog box is closed.
  - Example: Class wizard
- Modeless
  - You can move the focus back and forth between the dialog box and other windows
  - Example: the Properties window

Displaying a Dialog (2)

- Rename the Color menu as Pen.
- `Insert Separator` (Figure 17-8 on P.993)
- Add a menu item “Width …” to display CPenDialog as a modal dialog.
  - An ellipsis (three period) indicates that it displays a dialog.
- Both the menu item and the toolbar button have their IDs as `ID_PENWIDTH`.

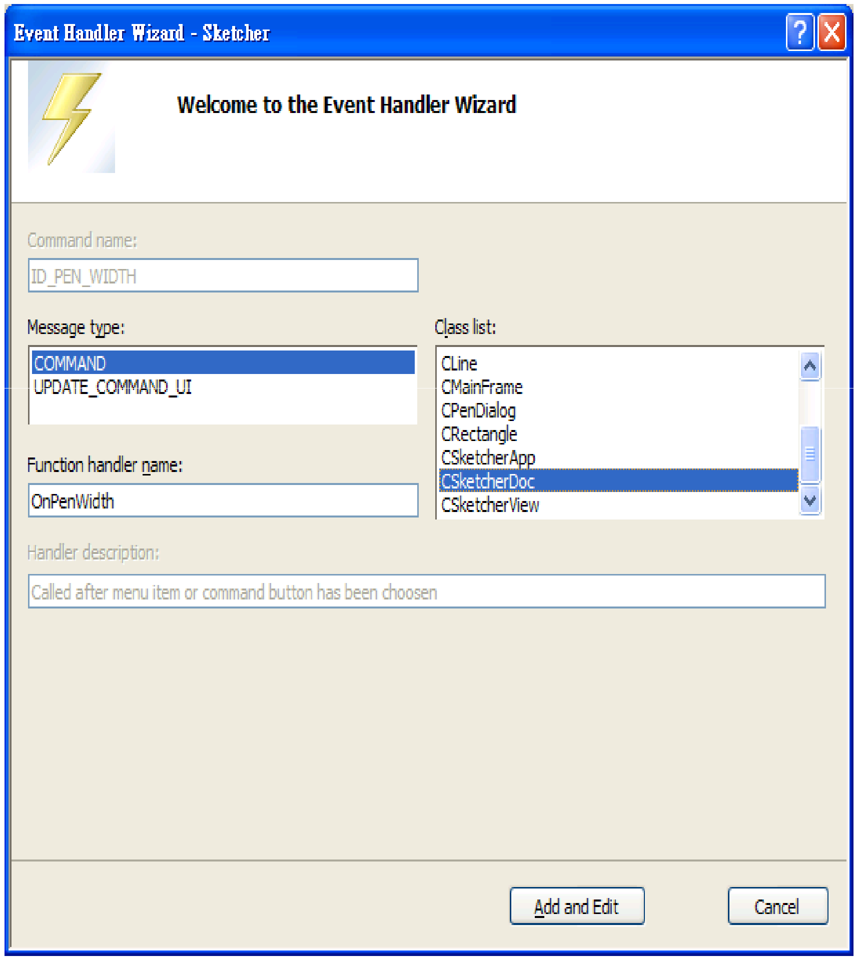
Code to Display the Dialog

- Right-click the Width menu item
- Add Event Handler to the CSketcherDoc class.

```
void
    CSketcherDoc::OnPenWidth()
{
    CPenDialog aDlg;
    // Create a local dialog object

    // Dispaly the dialog as modal
    aDlg.DoModal();
}
```

- #include PenDialog.h at the beginning of SketcherDoc.cpp


Event Handler Wizard - Sketcher
Welcome to the Event Handler Wizard
Add and Edit
Cancel

Command name: ID_PEN_WIDTH

Message type: COMMAND (selected), UPDATE_COMMAND_UI

Class list: CLine, CMainFrame, CPenDialog, CRectangle, CSketcherApp, CSketcherDoc (selected), CSketcherView

Function handler name: OnPenWidth

Handler description: Called after menu item or command button has been choosen

Class CPenDialog

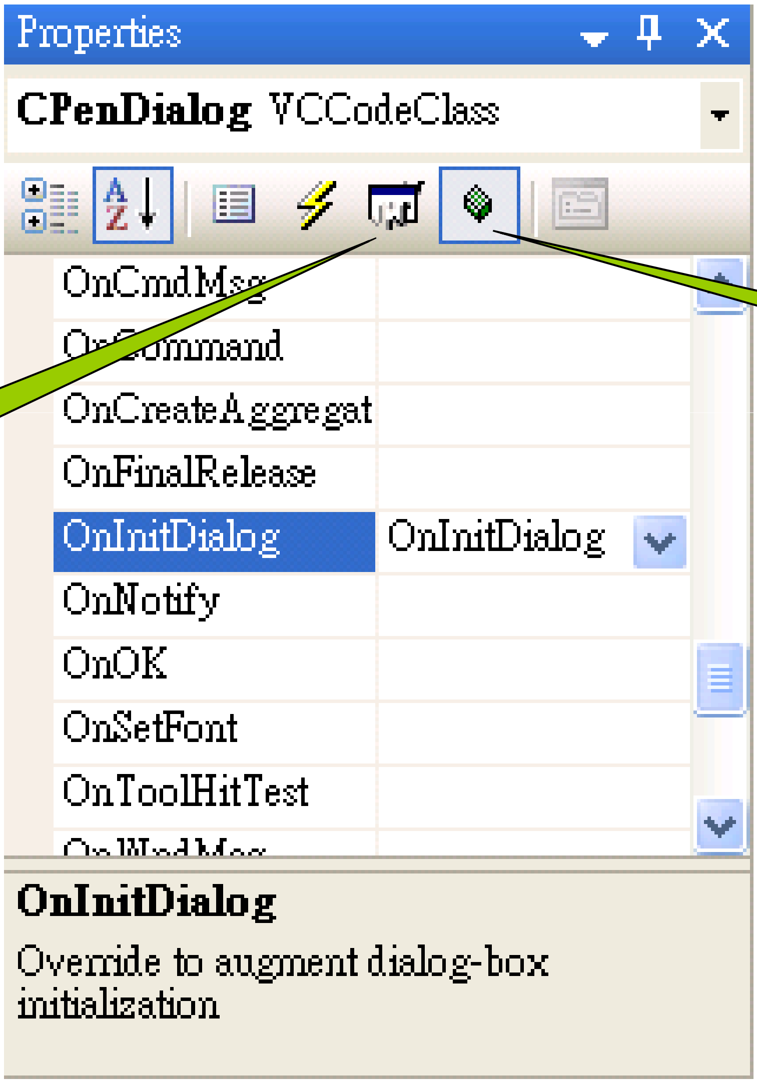
- Store the pen width in a data member, m_PenWidth

```
class CPenDialog : public CDialog
{
public:
   CPenDialog(CWnd* pParent = NULL);   // standard constructor
   virtual ~CPenDialog();

// Dialog Data
   enum { IDD = IDD_PENWIDTH_DLG };

public:
   // Record the pen width
   int m_PenWidth;
};
```

Overrides the `OnInitDialog()` function



Properties

CPenDialog VCCodeClass

OnCmdMsg	
OnCommand	
OnCreateAggregat	
OnFinalRelease	
OnInitDialog	OnInitDialog
OnNotify	
OnOK	
OnSetFont	
OnToolHitTest	
OnWndMsg	

OnInitDialog

Override to augment dialog-box initialization

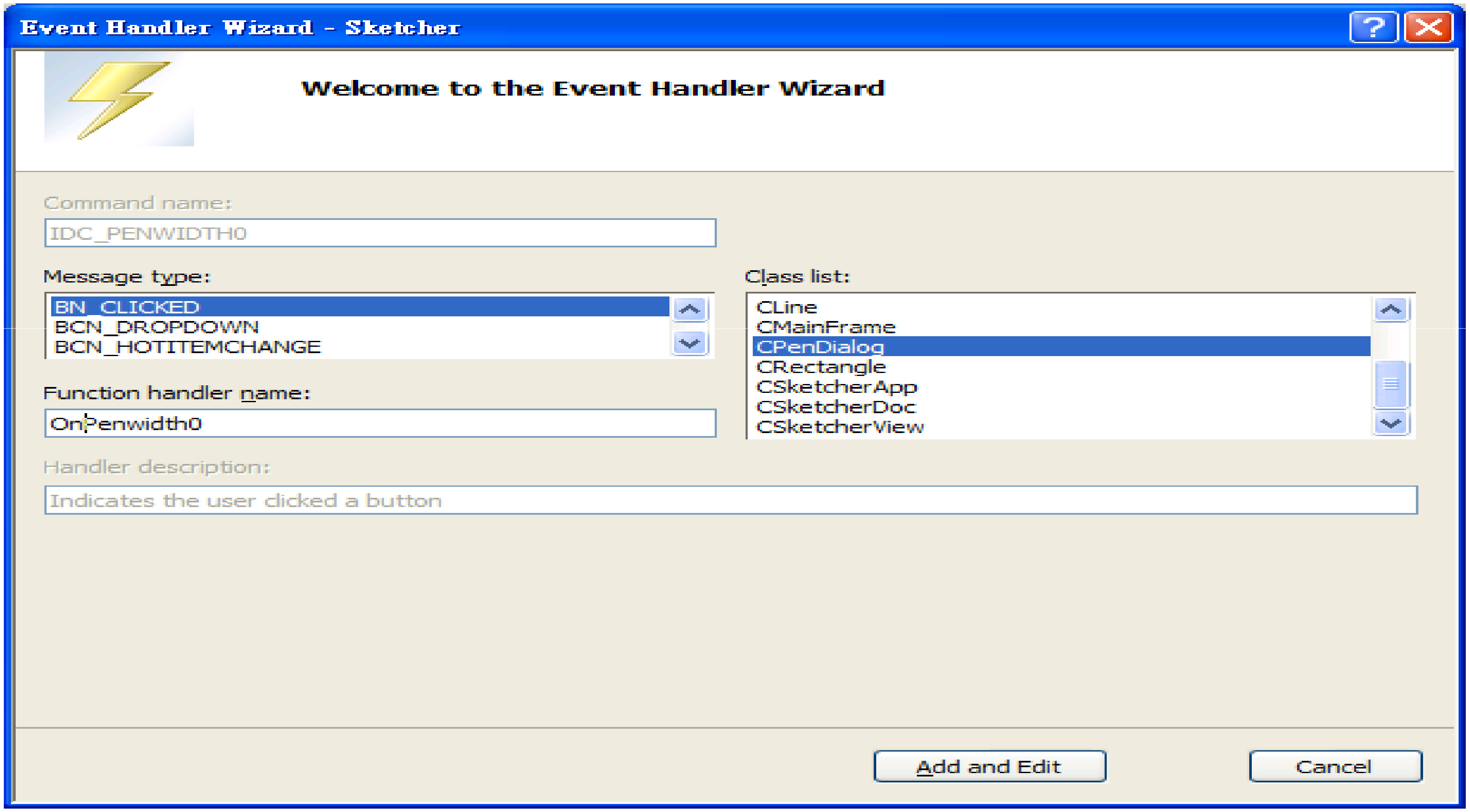
OnInitDialog()

Inherited indirectly from CWnd through CDialog.

```
BOOL CPenDialog::OnInitDialog()
{
	CDialog::OnInitDialog();

	switch (m_PenWidth)
	{
	case 1:
		CheckDlgButton(IDC_PENWIDTH1, 1);
		break;
	case 2:
		CheckDlgButton(IDC_PENWIDTH2, 1);
		break;
	case 3:
		CheckDlgButton(IDC_PENWIDTH3, 1);
		break;
	case 4:
		CheckDlgButton(IDC_PENWIDTH4, 1);
		break;
	case 5:
		CheckDlgButton(IDC_PENWIDTH5, 1);
		break;
	default:
		CheckDlgButton(IDC_PENWIDTH0, 1);
		break;
	}
	return TRUE;  // return TRUE unless you set the focus to a control
}
```

Add Event Handler to Radio Buttons



Event Handler Wizard - Sketcher

Welcome to the Event Handler Wizard

Command name:
IDC_PENWIDTH0

Message type:
BN_CLICKED
BCN_DROPDOWN
BCN_HOTITEMCHANGE

Class list:
CLine
CMainFrame
CPenDialog
CRectangle
CSketcherApp
CSketcherDoc
CSketcherView

Function handler name:
OnPenwidth0

Handler description:
Indicates the user clicked a button

PenDialog.cpp

```
void CPenDialog::OnPenwidth0()
{
    m_PenWidth = 0;
}

void CPenDialog::OnPenwidth1()
{
    m_PenWidth = 1;
}

void CPenDialog::OnPenwidth2()
{
    m_PenWidth = 2;
}
```

`OnPenWidth()` handler in CSketcherDoc

```
void CSketcherDoc::OnPenWidth()
{
    CPenDialog aDlg;  // Create a local dialog object

    // Set the pen wdith in the dialog to that stored in the
    document
    aDlg.m_PenWidth = m_PenWidth;

    // Dispaly the dialog as modal
    if (aDlg.DoModal() == IDOK)
        m_PenWidth = aDlg.m_PenWidth;
}
```

Adding Pen Widths to the Document

```
class CSketcherDoc : public CDocument
{
// Operations
public:
    unsigned int GetElementType()
    { return m_Element; }
    COLORREF GetElementColor()
    { return m_Color; }
    int GetPenWidth()
    { return m_PenWidth; }
protected:
    // Current element type
    unsigned int m_Element;
    COLORREF m_Color;          // Current drawing color
    int m_PenWidth;            // Current pen width
};
```

Constructor of CSketcherDoc

```
CSketcherDoc::CSketcherDoc()
: m_Element(LINE)
, m_Color(BLACK)
, m_PenWidth(0)              // 1 pixel pen
{
  // TODO: add one-time construction code
  here


}
```

Modify `GetBoundRect()` to deal with a pen width of zero

```
CRect CElement::GetBoundRect()
{
  CRect BoundingRect;
  BoundingRect = m_EnclosingRect;

  BoundingRect.InflateRect(m_Pen, m_Pen);
  return BoundingRect;
}
```

P.879

Modify `GetBoundRect()` to deal with a pen width of zero

```
CRect CElement::GetBoundRect()
{
  CRect BoundingRect;
  BoundingRect = m_EnclosingRect;

  int Offset = (m_Pen == 0) ? 1 : m_Pen;
  BoundingRect.InflateRect(Offset, Offset);
  return BoundingRect;
}
```

Constructor declaration of CLine

```
CLine(CPoint Start, CPoint End, COLORREF
aColor, int PenWidth);
```

CreateElement() in CSketcherView

```
CElement* CSketcherView::CreateElement(void)
{
    // Get a pointer to the document for this view
   CSketcherDoc* pDoc = GetDocument();

   // Now select the element using the type stored in the document
   switch(pDoc->GetElementType())
   {
     case LINE:
        return new CLine(m_FirstPoint, m_SecondPoint,
                         pDoc->GetElementColor(), pDoc->GetPenWidth());

     default:
        // Something's gone wrong
        AfxMessageBox(_T("Bad Element code"), MB_OK);
         AfxAbort();
        return NULL;
    }
}
```

Constructor Implementation of CLine

```
CLine::CLine(CPoint Start, CPoint End, COLORREF
   aColor, int PenWidth)
{
   m_StartPoint = Start;
   m_EndPoint = End;
   m_Color = aColor;
   m_Pen = PenWidth;

   m_EnclosingRect = CRect(Start, End);
   m_EnclosingRect.NormalizeRect();
}
```