Using ActiveX Controls

Microsoft ActiveX Controls
- Reusable software components that can be plugged into many different programs
- Allows you to design & use custom controls
- Like concept of hardware components
- Expansion of OLE technology
  - Enables combining documents created with different apps into a single doc
  - ActiveX allows it to work in a distributed environment (e.g., the internet)

COM Technology
- Microsoft’s Component Object Model
- Interface and interaction model
- Defines how to construct ActiveX objects & how interfaces are designed
- A COM “Interface”:
  - Like a function call into an ActiveX object
  - COM specifies how function must be built & called
  - And how to pass data & events to/from controls
  - Not specific to any language
- ActiveX controls can be used with many different tools (e.g., Access, FoxPro, VB)

Automation
- Key technology in ActiveX
- Enables an app embedded in another app to activate itself & control its part of the user interface
  - Does its thing and shuts itself down when user moves on
  - e.g., an Excel spreadsheet in a Word document

Servers and Containers
- Embedding an ActiveX object in another app
- Embedded object is implemented as an ActiveX “server”
- Containing object called a “container”
- A server can also be a container
  - (e.g., Internet Explorer)
- ActiveX control: a special case of ActiveX server
  - ActiveX controls cannot run on their own
  - Stored in an .ocx file
- In MFC any class derived from CWnd can be an ActiveX control container
  - COleControl is base class for ActiveX controls

Interaction between control & container
- Occur through three IDispatch Interfaces
  - Events
  - Properties
  - Methods
**ActiveX Control Events**
- Notification messages sent from the control to the container application
  - Usually as a result of user action
- Control sends event to container when something occurs inside control
  - e.g., mouse clicks, pressed buttons, expiring timers
- Triggering of events done in the IDispatch interface in the container
- Calls a handler function in the container
- Two types: Stock & Custom

**ActiveX Control Properties**
- Attributes of controls visible to and usually modifiable by container
  - Stock: e.g., background color, default font
  - Custom: related to functionality of control
- Provided by container but maintained by control
- Must also specify property aspects
  - name shown to container
  - internal variable used in code

**ActiveX Control Methods**
- Functions exposed by control and called by container
- Use Visual Studio Wizards to add methods to a control
  - Specify name, return type, & parameters

**Adding an ActiveX Control to a Dialog Box**
- Right click on dialog box
  - Click "Insert ActiveX control"
  - "Insert ActiveX Control" dialog box appears
- Scroll through ActiveX controls registered on system
  - Select the one you want
- Click "OK" and control will be added

**Configuring an ActiveX Control**
- Just as with other controls, use Class Wizard (Properties Box) to add message-handling functions and to associate with an MFC objects
- Add member variables just as though it were a standard control
- Most controls will have many properties exposed as variables and many methods (member functions)
An Example: Using the Microsoft Hierarchical Flex Grid Control

- **Grid Control**
  - Like a mini spreadsheet
  - Divided into rows and columns --> cells
  - Tracks active cells, size & contents of each cell
  - Data in a cell obtained through a member function call
  - You can:
    - Retrieve current row, cell, column information
    - Set attributes (font, size, contents) for current cell
    - Retrieve attributes of current cell

Preparing the App

- New MFC AppWizard (exe) application
  - Choose Dialog-based application type
  - In Advanced Features, make sure ActiveX Controls check box is selected
  - Name it GridCtrl

Adding the Microsoft ActiveX FlexGrid Control

- Right click on App’s dialog box
  - Click “Insert ActiveX Control”
  - “Insert ActiveX Control” dialog box appears
- Scroll through ActiveX controls registered on system
  - Select “Microsoft Flex Grid Control 6.0”
- Click “OK” and control will be added to app’s dialog box
- Expand size of control

- Click on grid control to bring up its properties box, change following properties:
  - ID: IDC_GRID
  - Rows: 5, Fixed Rows: 1
  - Cols: 5, Fixed Cols: 1
  - ScrollBars: 0-None
- Add an edit control
  - ID: IDC_EDIT
- Add a “Calculate” button
  - ID: IDC_CALC

Add and attach member control variables to edit and grid controls in the CGridCtrlDlg class:

<table>
<thead>
<tr>
<th>Resource ID</th>
<th>Category</th>
<th>Type</th>
<th>Variable name</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDC_EDIT</td>
<td>Control</td>
<td>CEdit</td>
<td>m_edit</td>
</tr>
<tr>
<td>IDC_GRID</td>
<td>Control</td>
<td>CMSFlexGrid</td>
<td>m_grid</td>
</tr>
</tbody>
</table>

Add protected member variables to CGridCtrlDlg class:

- BOOL m_bEditing
- int m_nRow
- int m_nCol
Add initialization code to CGridCtrlDlg::OnInitDialog
– See listing

Use Class Wizard to add a "Button Click" event handler for the Grid control
– Class: CGridCtrlDlg
– Lightning bolt (Events)
– Object ID: IDC_GRID
– Message: Click
– Handler Function: default ClickGrid()

Add code to ClickGrid()
– See listing

Recomputing the Totals

Add a click event handler to the “Calculate” button
– Object ID: IDC_CALC
– Class: CGridCtrlDlg
– Message: BN_CLICKED
– Function: default OnCalc()

Add code to OnBnClickedCalc()
– See listing

Build the Application