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
  - Enabled combining docs 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
  - 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
- Embedded object is implemented as an ActiveX server
- Containing object called a container
- A server can also be a container
  - (e.g., Internet Explorer)
- An ActiveX control is a special case of an ActiveX server
- 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
IDispatch Interfaces

ActiveX Control Events
- Notification messages sent from the control to the container application
- 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
- 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
- Use Class Wizard Automation tab to specify control properties
- Must also specify property aspects
  - name shown to container
  - internal variable used in code

ActiveX Control Methods
- Function exposed by control and called by container
- Use Class Wizard Automation tab to add methods to a control
  - Specify name, return type, & parameters

Components and Controls Gallery
- Visual Studio's store of reusable components
- Most are ActiveX controls
- Adding your own classes to the Gallery:
  - Open project containing the class
  - Open Class View in project workspace
  - Right click on class name
  - Select "Add to Gallery"
- Displaying all available controls on computer:
  - Main menu: "Project | Add to Project | "Components and Controls" -- > Component Gallery dialog box
  - Select Registered ActiveX Controls from list box

Adding an ActiveX Control to Dialog Box Editor
- So you can use it like any of the other standard controls
  - Select desired ActiveX control icon
  - Click "Insert" button
  - Click "OK" on resulting message box
  - Click "OK" to resulting list box containing the classes that will be added to your project
  - Click "Close" to get rid of Gallery dialog box
- Now control will appear in Dialog box editor tool box and can be added by dragging and dropping
Configuring ActiveX Control w/ Class Wizard

- Just as with other controls, use Class Wizard to add message-handling functions and associate it with an MFC object
- Add member variables just as though it were a standard control
- Most controls will have many properties exposed as variables

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

The GridCtrl App

- Add member variables just as though it were a standard control
- Most controls will have many properties exposed as variables

Preparing the App

- New MFC AppWizard (exe) application
  - Choose Dialog-based
  - Makes sure ActiveX Controls check box is selected
- Add the grid control to the app (see above)
  - Then open ResourceView and double-click on IDD_GRIDCTRL_DLG
  - Drag and drop the the new grid control from the tool bar to the dialog-app

- Use Class Wizard to attach member 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 "Click" handler for the Grid control
- Class: CGridCtrlDlg
- Tab: Message Maps
- Object ID: IDC_GRID
- Message: Click
- Handler Function: default OnClickGrid()

Add code to OnClickGrid() -- See listing

Recomputing the totals

Use Class Wizard to a message handler to the "Calculate" button
- Object ID: IDC_CALC
- Class: CGridCtrlDlg
- Message: BN_CLICKED
- Function: default OnCalc()

Add code to OnCalc() -- See listing

Build the Application