Creating and using a Custom ActiveX Control

AXCrtl and AXCont: Example ActiveX control and Container
- AXCrtl displays a picture
- User clicks on picture (event), it switches to another picture & beeps
- Properties allow AXCont container app to set control's background color
- Container can call an About() method in control that gives info about the control

Creating the ActiveX Control
- File | New | “Projects” tab
- “MFC ActiveX Control” Template
- Name it AXCtrl
- “Control Names”: Take Defaults
- “Finish”

Creating the Bitmaps
- Project | Add Resource | Bitmap | New
- Draw bitmap (about 150 X 150 pixels)
- Keep defaults
  - ID: IDB_BITMAP1
  - Filename: DAY.BMP
- Repeat with second bitmap
  - ID: IDB_BITMAP2
  - Filename: NIGHT.BMP

Loading the Bitmaps
- Add public variables to CAXCtrlCtrl class
  - CBitmap* m_CurrentBitmap
  - CBitmap m_BitmapNight
  - CBitmap m_BitmapDay
- Add code to constructor:
  m_BitmapNight.LoadBitmap(IDB_BITMAP2);
  m_BitmapDay.LoadBitmap(IDB_BITMAP1);
  m_CurrentBitmap=&m_BitmapDay;
Adding a Click Message Handler

- In CAXCtrlCtrl Properties Box
  - “Message Maps”
  - WM_LBUTTONDOWN in “Messages” List
  - Add following code to OnLButtonUp():
    ```c
    if (m_CurrentBitmap == &m_BitmapNight)
    m_CurrentBitmap=&m_BitmapDay;
    else
    m_CurrentBitmap=&m_BitmapNight;
    InvalidateControl();  // force call to OnDraw()
    ```

Enabling BackColor Property

- Expand CAXctrlLib node in Class View
- Right click on _DAXCtrl node
  - Add | Add Property
  - Brings up “Add Property Wizard” Dialog Box
  - Select BackColor stock property from “Property Name” list
  - “Stock” option should be selected
  - “Finish” button

MFC stores value of BackColor property & initializes it to background color of any container the control is in
- If property is changed, control is invalidated, forcing OnDraw() to redraw it

Defining Properties

- BackColor Stock Property
  - A predefined property
  - Lets container app change control’s background color

Coding for OnDraw()

```c
void CAXCtrlCtrl::OnDraw(CDC* pdc, const CRect& rcBounds,
const CRect& rcInvalid)
{
// TODO: Replace the following code with your own drawing code.
BITMAP BM;
CDC MemDC;
CBrush Brush (TranslateColor(GetBackColor()));  // get color from control &
// translate to COLORREF
MemDC.CreateCompatibleDC(NULL);
MemDC.SelectObject(m_CurrentBitmap);
m_CurrentBitmap->GetObject(sizeof(BM), &BM);
pdc->BitBlt((rcBounds.right - BM.bmWidth)/2, 
(rcBounds.bottom - BM.bmHeight)/2, BM.bmWidth, BM.bmHeight, &MemDC, 0, 0, SRCCOPY);
}
```

Property Pages

- For developers so they can work with the new control
- Provides users of control with a way to set its properties
  - Select Resource View & expand dialog folder
  - Click on control to open its property box
  - Change the value of the property
  - (Illustrate with AXCont4)

- In “Test Container” app that comes with Visual Studio, you can display the properties pages of a control
  - Each property page is displayed as a tab of the Control Properties Dialog Box
  - Each page contains a property of the control
  - Properties can be modified

- Container app can assign initial values to the control’s properties
- A new ActiveX control has a single property page
  - Defined in IDD_PROPPAGE_AXCTRL dialog resource

Adding a Background Color Property Page to AXCtrl App

- The Stock Color property page
- Used to set value of Control’s BackColor property when container app is designed
- In CAXCtrlCtrl class (Property pages section)
- Change 1 to 2 in macro:
  ```c
  BEGIN_PROPPAGEIDS(CAXCtrlCtrl, 2);
  ```
- Add second PROPPAGEID:
  ```c
  PROPPAGEID(CLSID_CColorPropPage)
  ```
- Macro will link Color property page with BackColor property
Defining Methods

- We'll use the predefined AboutBox method
- When a container calls it, the control displays an “About” dialog box
  - Defined in IDD_ABOUTBOX_AXCTRL dialog resource
- To add other Methods you would:
  - Select _DAXCtrl interface node in Class View
  - Right click and select Add | Add Method
  - Specify the Method Name & return type, parameters, etc.
  - Edit new method adding your code

Defining Events

- Once defined, control can call an associated Fire function
  - e.g., FireClick() for click action on control
- Calling the Fire function called “firing an event”
- Causes an event handling function in container to be called
- For stock events MFC provides Fire functions & calling code
- For custom events ClassWizard can generate Fire function.
  - We must write calling code when event is to be fired

Defining a Click stock event for AXCtrl App

- Right click on CAXCtrlCtrl class
  - “Add” | “Add Event”
    - “Add Event Wizard” opens up
  - Select Event Name “Click” & Event type “Stock”
  - “Finish
- FireClick is defined in COleControl base class
- MFC adds code to call it to fire the Click event whenever user clicks on the control
  - So no calls to FireClick() need to be added

Building / Registering the Control

- Build as usual
  - Generates the file AXCtrl.ocx
  - Also registers the control on the system being used
    - So it can be accessed by containers you write

Making the Control usable to other apps

- Should provide an installation program
  - To register the control on the user’s system
- See online help
- Done automatically when ActiveX control is built on the machine Visual Studio is running on
Testing the Control
- Use “Test Container” program that comes with VC++
  - “Tools” / “ActiveX Control Test Container”
  - Brings the Test Container
- “Edit” / “Insert New Control”
- Select “AXCtrl Control” & click “OK”
  - Brings the control into the Test Container (enlarge it)
- Now Properties can be tested
  - “Edit” / “Properties” -> Properties Page
  - Try changing Background color property
- Methods can also be invoked
  - “Control” / “Invoke Methods” / Invoke AboutBox(Method)
- Click Event can also be fired
  - Click on the control

Adding the ActiveX Control to the Project
- 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 AXCtrl
- Click “OK”
- Increase size of control

Designing the App’s Dialog Box
- Open IDD_AXCONT_DIALOG
- Delete static text “TODO” & “OK” button
- Change caption of Cancel button to “Close”
- Add an “About” button
  - IDC_ABOUT

Creating the ActiveX Control Container Application
- New MFC AppWizard (exe) application
  - Choose Dialog-based application type
- Advanced Features: leave “ActiveX Controls” option selected
- User Interface Features: Dialog title: “ActiveX Control Container Demo”

Customizing Initial Properties
- Right click on the ActiveX Control, select “Properties”
  - Note “BackColors” property page
- Open “Color” property page (click on down arrow)
  - Click on Red button to set background color to red
Attach ActiveX Control to a Wrapper Class Object

- Want code in dialog box class to be able to access functions in the control
  - To change properties & call its methods
- Add member variable to CAXContDlg class
  - (Right click on control, Add | Add member variable)
  - Name: m_AXCtrl, Category: Control (default), Variable Type: CAxctrlctrl1 (only choice)
- “Finish”

Adding Button Click Handler for “About” Button

- Class: CAXContDlg
- Properties Box | “Events”
  - Select IDC_ABOUT
    - BN_CLICKED, “Add Function” --> OnBnClickedAbout()
    - Edit Code:
      m_AXCtrl.AboutBox();

Adding a Click (on control) Event Handler

- Class: CAXContDlg
- Properties Box | “Events”
  - Select IDC_AXCTRLCTRL1
    - Add code:
      ::MessageBeep(MB_OK);

Build and run the application