Creating and using a Custom ActiveX Control
(From Michael Young, “Mastering Visual C++”)

ActiveX Control (Review)
- A custom control that can be plugged into any container application
- Mechanisms for interaction between a control and its container:
  - Properties
    - attributes of control a container can read/change
  - Methods
    - Functions provided by control callable by container
  - Events
    - Occurrence in control to trigger container response

ActiveX Events, Properties, and Methods

AXCtrl and AXCont: Example ActiveX control and Container
- AXCtrl displays a picture
- User clicks on picture (event), it switches to another picture & beeps
- Properties allow AXCont container to change control’s background color and add/remove a frame around the picture
- Container can call an About() method in control that give info about the control

Creating the ActiveX Control
- File / New / Projects
- MFCActiveX Control Wizard
- Name it AXCtrl / “OK”
- “Finish” in Step 1 & 2 Dialog boxes
- “OK”
Creating the bitmaps

- Insert / Resource / New
- Draw bitmap (about 150 X 150 pixels)
- Keep default ID
  - IDB_BITMAP1
- Repeat with second bitmap
  - IDB_BITMAP2

Loading the Bitmaps

- Add variables to CAXCtrlCtrl class
  - CBitmap* m_CurrentBitmap
  - CBitmap m_BitmapNight
  - CBitmap m_BitmapDay
- Add code to constructor:
  m_BitmapNight.LoadBitmap(IDB_BITMAP1);
  m_BitmapDay.LoadBitmap(IDB_BITMAP2);
  m_CurrentBitmap=&m_BitmapDay;

The AXCtrl ActiveX Control

- ClassWizard / “Automation” tab
- “BackingProperty” tab
- BackColor Stock Property
  - A predefined property
  - Lets container change control’s background color
- A ShowFrame Custom Property
  - Lets container place or remove a frame around the picture in the control

Adding a Click Message Handler

- Run ClassWizard
  - “Message Maps” Tab / CAXCtrlCtrl Class
  - WM_LBUTTONDOWN in “Messages” List
- Add following code to OnLButtonUp():
  if (m_CurrentBitmap == &m_BitmapNight )
    m_CurrentBitmap=&m_BitmapDay;
  else
    m_CurrentBitmap=&m_BitmapNight;
  InvalidateControl(); //force call to OnDraw()

Enabling BackColor Property

- ClassWizard / “Automation” tab
- CAXCtrlCtrl Class / Click “Add Property”
  - Brings up Add Property Dialog Box
  - Select BackColor stock property from “External Name” list
  - “Stock” option should be selected
  - OK
- MFC stores value of BackColor property & initializes it to background color of any container control is in
- If property is changed, control is invalidated, forcing OnDraw() to redraw it
Defining ShowFrame Custom Property

- Again Click “Add Property”
  - External Name: ShowFrame
  - Type: BOOL
  - Variable name: m_showFrame
  - Default Notification fn: OnShowFrameChanged()
- If container changes value of property, MFC code assigns new value to m_showFrame & calls OnShowFrameChanged() fn -- we must add code to this

Coding for ShowFrame property

- Initialize ShowFrame property and m_showFrame variable:
- In CAXCtrlCtrl::DoPropExchange()
  PX_Bool(pPx, “ShowFrame”, m_showFrame, FALSE);
  - Initializes both to FALSE
- In CAXCtrlCtrl::OnShowFrameChanged()
  InvalidateControl();
  - Forces control’s OnDraw() to execute

Coding for OnDraw()

- Replace default ellipse-drawing code in OnDraw()

```cpp
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());
    pdc->FillRect(rcBounds, &Brush);
    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);

    if(m_showFrame)
    {
        CBrush *pOldBrush = (CBrush *)pdc->SelectStockObject(NULL_BRUSH);
        CPen Pen(PS_SOLID | PS_INSIDEFRAME, 20, RGB(0,0,0));
        CPen *pOldPen = pdc->SelectObject(&Pen);
        pdc->Rectangle(rcBounds);
        pdc->SelectObject(pOldPen);
        pdc->SelectObject(pOldBrush);
    }
}
```

Property Pages

- In container app, work similar to “Properties” dialog box
- Comes up in “Properties” of Control
- Each tab contains a property of the control
- Can be used to assign initial values to the control’s properties
  - Defined in IDD_PROPPAGE_AXCTRL dialog resource
- We’ll modify it for the ShowFrame property
  - Replace default static control with a check box
  - ID=IDC_SHOWFRAME, caption=“Display a frame around the picture”
  - Now link new check box to ShowFrame property

Add a second property page

- The Stock Color property page
- Used to set value of Control’s BackColor property when container app is designed
- In CAXCtrlCtrl class, // Property page section
- Change 1 to 2 in macro:
  BEGIN_PROPPAGEIDS(CAXCtrlCtrl, 2);
- Add second PROPPAGEID:
  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:
  - Use ClassWizard / “Automation” tab
  - Select control class (CAXCtrlCtrl)
  - Click “Add Method” button
  - Enter name, return type, parameters
  - Method could be either stock or custom

Defining Events
- Once defined, control can call 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
- ClassWizard / “ActiveX Events” tab
  - “Add Event” -- opens “Add Event” dialog box
  - Select “Click” stock event in “External Name” list
  - Make sure “Stock” option is enabled
  - Click “OK”
- 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 others
- Should provide an installation program
  - To register the control on the user’s system
- See online help on:
  - “Distributing ActiveX Controls”
Testing the Control

- Use “Test Container” program that comes with VC++
  - “Tools” / “ActiveX Control Test Container”
    - Brings up 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 ShowFrame 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

- Generates a wrapper class: CAXCtrl
  - So program can interact with control
- Appends an “OCX” button for the control to the Controls toolbar of the dialog editor

Creating the ActiveX Control Container Application

- File / New / Projects
- MFC AppWizard (exe)
- Dialog Based option
- In step 2 deselect About box
- Leave ActiveX Controls option selected
- Title: ActiveX Control Container Demo

Steps

- “Project” / “Add to Project” / “Components and Controls”
  - Opens Components & Controls Dialog Box
- Select Gallery folder, double-click “Registered ActiveX Controls”
- Select AXCtrl (our new control)
- Click “Insert” & “OK”
- Click “Close”
- This will add the control to the Dialog box toolbar
  - “OCX” button

Designing the App’s Dialog Box

- Open IDD_AXCONT_DIALOG
- Delete static text “TODO” & “OK” button
- Change caption of Cancel button to Close
- Add “About” and “Frame” buttons
  - IDC_ABOUT, IDC_FRAME
- Drag our ActiveX Control (“OCX” button) from toolbar into dialog box
- Enlarge it so bitmap and large surrounding are visible
Customizing Initial Properties

- Right click on the ActiveX Control, select “Properties”
  - Note “Control” and “Colors” property pages
- Open “Color” property page (click on tab)
  - Click on Red button to set background color to red
- Open “Control” property page (click on tab)
  - Check “Display Frame…” check box
    - Will set ShowFrame property to TRUE
    - Overrides FALSE value we set in program
    - Produces a black frame around picture

Attach ActiveX Control to a Wrapper Class Object

- So code in dialog box class can access functions in the control
  - To change properties & call its method
- ClassWizard / “Member Variables” tab
- Dialog class: CAXContDlg
- Select IDC_AXCTRLCTRL1 (the control)
- “Add Variable” button
  - Name: m_AXCtrl, Category: Control (default), Type: CAXCtrl (only choice)
  - “OK”

Interaction Between Control & Container

Defining Message Handlers for “About” & “Frame” Buttons

- ClassWizard / “Message Maps” tab
- Class: CAXContDlg
- Select IDC_FRAME
  - BN_CLICKED, “Add Function” --> OnFrame
  - Edit Code:
    - m_AXCtrl.SetShowFrame (!m_AXCtrl.GetShowFrame());
- Select IDC_ABOUT
  - BN_CLICKED, “Add Function” --> OnAbout
  - Edit Code:
    - m_AXCtrl.AboutBox();

Adding a Click Event Handler

- ClassWizard / “Message Maps” tab
- Class: CAXContDlg
- Select IDC_AXCTRLCTRL1
- Select “Click” Message (only event fired)
- Select “Add Function” and add code:
  - ::MessageBeep(MB_OK);

Build and run the application