BTNDMO.EXE

Concept

The basic concept of this demo revolves around, what I consider to be, two main requirements for a custom control: a container; and custom events, properties, and methods. Lets examine these two requirements more closely:

a. Container. Basically, the container's job is twofold; it is the area on screen in which the control is allowed to do all of its painting, and it also must be able to send/receive Windows messages.

b. Events, Properties, and Methods. The control must be able to map custom events and properties to the container, and have a mechanism to implement custom methods. This would allow for painting/updating of the control on the screen when required, reaction to the setting/getting of properties, etc.

Ultimately, the container should have complete control over its painting, and custom events, properties, and methods. With VB4, all of these requirements can be achieved through the use of the picturebox control and a class module.

Picturebox Control. The picturebox control already meets all of our requirements for a custom control except for one main ingredient: Flexibility in customizing the control to fit our needs (i.e. custom events and properties). The picturebox has its own dc, which allows the freedom/flexibility to paint our control exactly as needed. The painting can be done using both the built-in methods (Print, PaintPicture, Line, Circle, etc), and the api methods (DrawText, Stretchblt, LineTo, Ellipse, etc) when required. With respect to events and properties, the picturebox control already implements most (if not all) standard events and properties that may be required (Borderstyle, Backcolor, Forecolor, Font, GotFocus, LostFocus, Paint, etc). The missing ingredient, flexibility in customization, can be easily be achieved with the use of a class module.

Class Module. The class module allows us to customize the container (i.e. picturebox control) by encapsulating all the code requirements for our custom control within the class itself. Then, by mapping all the picturebox events to the class we can achieve the flexibility in customization required to meet our needs. By "wrapping" the picturebox control in a class module, all that is required to use the class in different projects is to add the class module to the project, add a picturebox control for each custom control required to the desired form (control/new class instance arrays work extremely well in this situation), map the events to the class, and declare a new instance of the class for each custom control. Once the code for the custom control class module is written, tested, and debugged, the process of adding the class/pictureboxes to new projects takes only a matter of a few minutes.

Mapping The Picturebox To The Class. The mapping of events to the class is simple and straight forward -- depending on exactly which events you decide to implement in your class, all you have to do is define an event in your class that the picturebox event will call. For example, all controls will need to draw themselves -- this is acheived by mapping the paint event of the picturebox to the appropriate event in the class. In this demo the Paint event is mapped to the Refresh method in the class:

```
'The new instance of the class would be declared in the
'declaration section of the form -- recommend to be
'declared as Public in order to reflect the same exposure
'given to normal controls.
Public clsCustomControl As New CCustomControl

Sub picCustomControl_Paint()
    'Call the class implementation of this event
    Call clsCustomControl.Refresh
End Sub
```

All other events implemented in the class would be mapped in the same manner.

Conclusion. Specifics on this demo are coded into the project files themselves as comments. The demo is very simple and is only designed to show the relationship between the picturebox and the class -- it was coded for comfort not speed <g> (i.e. paint events repaint the whole control instead of implementing a check for invalid rects, etc). This is only a basic example, but the logic/relationships shown in the demo can easily be improved upon and enhanced in future projects.

Have fun! :)

Doug Marquardt
CIS: 72253,3113