February, 2000

Advisor Answers

Disabling OptionGroups

Visual FoxPro 6.0

Q: When an OptionGroup's Enabled property is set to .F., the buttons look no different than when it's .T. Is there a way to give a user a visual indication that the group has been disabled?

–Barbara Peisch, San Diego, CA

A: You're absolutely right. When you disable an OptionGroup, the display of the buttons in that group doesn't change. However, when you disable an individual button, it does dim. So, the solution is to pass the disable message on to the buttons in the group. It's easy to do so, using the SetAll method of the option group, like this:

```
This.SetAll("Enabled",.F.)
```

The corresponding command to re-enable the group is:

```
This.SetAll("Enabled",.T.)
```

In VFP 6, the best place to do this is in an Assign method for the Enabled property of the OptionGroup. That way, you can be sure it happens every time the Enabled property changes.

But what if some of the buttons in the group are disabled before the group as a whole is disabled? Issuing the two SetAll commands leaves them Enabled afterward. If you want to keep track of the prior state of the buttons and restore them, you have to do a little more work.

I've combined all this into a custom OptionGroup class called opgEnable. It uses a custom array property, aButtonStatus, and a logical property, lStatusSaved, to keep track of the individual buttons. The Assign method uses the SetAll call and uses two helper methods (SaveStatus and RestoreStatus) to handle the tracking. Here's the code:

```
* Enabled_Assign
LPARAMETERS vNewVal
* Allow you to enable and disable the group as a whole.

* Make sure buttons get saved initially
IF NOT This.lStatusSaved
   This.SaveStatus()
ENDIF

This.SetAll("Enabled", m.vNewVal)

IF m.vNewVal
   This.RestoreStatus()
ENDIF
```

```
THIS.Enabled = m.vNewVal

RETURN
```

The Assign method first saves the status of the individual buttons, if it's not already saved. SaveStatus is called to go through all the buttons and store their current status (enabled or disabled) to the array This.aButtonStatus. Then, whether the group is being enabled or disabled, SetAll is used to change Enabled for all the buttons. Then, if the group is being enabled, RestoreStatus loops through all the buttons, restoring their prior status from This.aButtonStatus. Finally, the group's Enabled property is set to the new value.

The lStatusSaved property allows us to make sure we don't overwrite the stored button status, if for some reason the group's Enabled property is set to .F. twice (or more) in a row. (Consider what would happen if the SaveStatus method were called twice. By checking This.lStatusSaved beforehand, we ensure that it's executed only the first time the Assign method is called for a given disabling of the group.)

Here's the code for SaveStatus and RestoreStatus:

```
* SaveStatus
* Save the current state of the buttons
LOCAL nButtonNumber

IF This.ButtonCount > 0
   DIMENSION This.aButtonStatus[ This.ButtonCount ]
   FOR nButtonNumber = 1 TO This.ButtonCount
      This.aButtonStatus[ nButtonNumber ] = ;
          This.Buttons[ nButtonNumber ].Enabled
   ENDFOR
   This.lStatusSaved = .T.
ENDIF

RETURN

*RestoreStatus
* Restore buttons' prior status
IF This.ButtonCount > 0
   FOR nButtonNumber = 1 TO This.ButtonCount
      This.Buttons[ nButtonNumber ].Enabled = ;
         This.aButtonStatus[ nButtonNumber ]
   ENDFOR

   * Once status has been restored, reset lStatusSaved
   * so we save it again the next time.
   This.lStatusSaved = .F.
ENDIF
```

Figure 1 shows a form containing an option group with two disabled buttons. Figure 2 shows the same form when you disable the entire group. When you enable the group again, it restores the configuration shown in figure 1. The form also contains a set of checkboxes mapped to the individual option buttons. Check and uncheck them to change which buttons are enabled initially, then disable and enable the group again and you can see that the group really does keep track of which buttons are currently enabled.

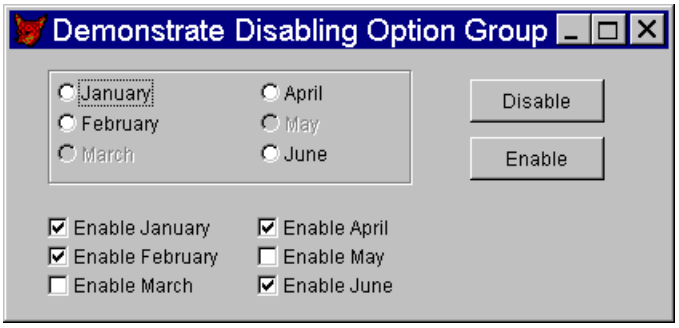

Figure 1. Partially disabled option group – When an option group has some buttons disabled, disabling and then re-enabling the whole group needs to respect the original state of those buttons.

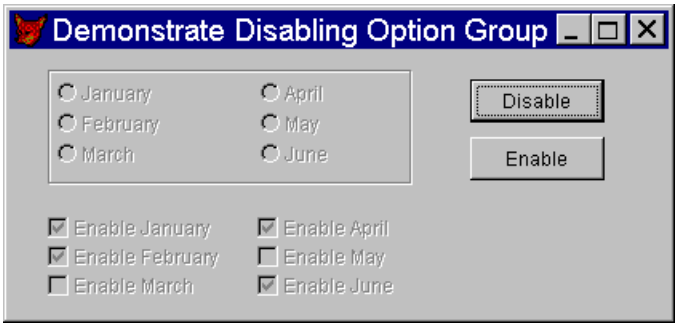

Figure 2. Disabled option group – To disable an entire option group, use an Assign method for the group's Enabled property.

This month's Professional Resource CD contains OptionGroup.VCX, a class library containing opgEnable, as well as the example form shown in figures 1 and 2.

–Tamar