Managing the UI state of accelerators and focus rectangles

devblogs.microsoft.com/oldnewthing/20050503-00

May 3, 2005

Raymond Chen

Starting with Windows 2000, keyboard indicators such as underlined accelerators and focus rectangles (collectively known as “keyboard cues”) are hidden by default, and are revealed only when you start using the keyboard. You can control this behavior from the Desktop Control Panel, under Appearance, Effects, “Hide underlined letters for keyboard navigation until I press the Alt key”.

Note that this setting actually controls both underlined letters and focus rectangles, even though the text describes only one of the effects. Underlines are hidden until you press the Alt key, and focus rectangles are hidden until you either press the Alt key or press the Tab key.

Here’s how it works.

There are three UI state mesages: `WM_CHANGEUISTATE`, `WM_QUERYUISTATE` and `WM_UPDATEUISTATE`. The third one is, in my opinion, a misnomer. It really should be called something like `WM_UISTATECHANGED` since it is a notification that something has happened, not a message that you send to cause something to happen.

When a dialog box or menu is displayed via a mouse click, keyboard cues are hidden; if the dialog box or menu was displayed via a keypress, then keyboard cues are visible. This decision is made by sending a `WM_CHANGEUISTATE` message to the root window with the `UIS_INITIALIZE` flag. This is done automatically by the dialog manager, but if you’re doing your own custom windows, you’ll have to send it yourself.

The `WM_CHANGEUISTATE` message bubbles up to the top-level window, which changes the window UI state accordingly, then broadcasts a `WM_UPDATEUISTATE` message to all its child windows to notify them that the state has changed. (Of course, if the `WM_CHANGEUISTATE` message has no effect—for example, hiding something that is already hidden—then the `WM_UPDATEUISTATE` message is optimized out since the entire operation is a no-op.)

When a window that draws keyboard cues receives a `WM_UPDATEUISTATE` message, it typically invalidates itself so that the cues can be redrawn/erased, depending on the new state.

At drawing time, a window that draws keyboard cues can use the `WM_QUERYUISTATE` message to determine which keyboard cues are visible and which are hidden, and draw its content accordingly. If focus rectangles are hidden, then the window should skip the call to the `DrawFocusRect` function. If keyboard underlines are hidden, then the window suppresses underlines in its text drawing. If the window uses the `DrawText` function, it can pass the `DT_HIDEPREFIX` flag to suppress the underlines. If you are responding to the `WM_DRAWITEM` message, then you should check for the `ODS_NOACCEL` and `ODS_NOFOCUSRECT` flags to determine whether you should draw an underline accelerator or a focus rectangle.

Finally, during execution you may discover that the user has used the keyboard to perform navigation within your control. For example, the listview control may have noticed that the user has used the arrow keys to change the selected item. When this happens, the control sends itself a `WM_CHANGEUISTATE` specifying which keyboard cues should be revealed. As noted above, the `WM_CHANGEUISTATE` message eventually causes all the windows in the window tree to receive a `WM_UPDATEUISTATE` message if their states need to change.

The `IsDialogMessage` function sends `WM_CHANGEUISTATE` messages as appropriate, so dialog boxes and anybody else who uses `IsDialogMessage` gets keyboard-cues tracking for free.

Raymond Chen

Follow