Why can’t you use the space bar to select check box and radio button elements from a menu?

devblogs.microsoft.com/oldnewthing/20101227-00

December 27, 2010

Raymond Chen

Nektar wants to know why you can’t use the space bar to select check box and radio button elements from a menu.

The short answer is “Because it’s a menu, not a dialog box.”

The check mark and radio button are just visual adornments provided by the menu manager as a courtesy; they do not affect the behavior of the menu itself. Notice, for example, that there is no way to specify “radio button groups” in a menu, so the menu manager wouldn’t know which items needed to be deselected when you select a radio button menu item. (I guess it could infer them from separators, but then you would have people saying “I want my radio button group to exclude item number 4, but I don’t want to put a separator in between; that looks ugly.”)

And then how would a program reject an auto-selected check box or radio button? E.g., the user pushes the space bar to turn on *Show the Widget Bar* and an error occurs trying to show the Widget Bar. If the program displays an error dialog, that would dismiss the menu. So maybe the program would just silently re-uncheck the box, which leaves the user puzzled as to why the space bar “doesn’t work” for turning on the Widget Bar. Or worse, what if hiding the Widget Bar causes the menu to reconfigure itself? (Maybe there are some menu items that are visible only when the Widget Bar is selected; if the user hides the Widget Bar, those menu items need to be deleted.) Windows doesn’t have a precedent for menus reconfiguring themselves *while they’re being displayed*. What if one of the items that gets deleted when you hide the Widget Bar is the menu item that contains the Widget Bar checkbox? (“I turned off the Widget Bar, and my menu disappeared!”)

That said, there is no technical reason these design issues couldn’t be overcome. You could have a style like `MF_GROUP` that behaves like `WS_GROUP` to establish the scope of menu radio buttons; you could have some way to mark a menu item as “this is an unselected check box” or “this is an unselected radio button”; you could come up with a way for a program to reject a user’s attempt to change the check box status; you could design a way for menus to be dynamically reconfigured while they are open; you could even design a way for menus to respond in some vaguely reasonable way when the item the user just selected gets dynamically deleted! But all of these features take effort, and they detract from the simple design of a menu as “Here’s a list of things you can do. Pick one. Once you pick one, the menu dismisses.” Every feature starts out with minus 100 points and needs to beat out the other 200 items on the list.

Raymond Chen

Follow