

# Pull Down and Pop Menus



## ***Pull Down Menus***

**Pull down menus are used extensively in Windows applications. They have a number of advantages:**

* The menu names are visible textual items on the menu bar, so the user can see the main functions of the software at all times.
* Menu titles, or menu items (such as Open Project on the File menu) should be unambiguous if they are well chosen, whereas, it can be difficult to design satisfactory icons for abstract functions  is the standard icon for open.
* The menu bar takes up very little space on the screen.
* The menu options are generally available by means of access keys (holding down the Ctrl key then typing underlined letters of the menu title and menu item required, so Ctrl+O for �Open Project� on the �File� menu).
* Functions are grouped together to form a set of menu items that are related, this helps users to know where to look for facilities.
* An effort has been made to standardise the naming and ordering of menu items across applications (especially Microsoft), so that packages are easier to learn and use.

When building new applications in VB, existing menu conventions should be used unless there is a very good reason to depart from them.

Meus are created using the Menu Design Window found under the Window menu. This window can also be invoked by clicking on the  icon on the toolbar.



The top half of the Menu Design Window contains the menu control properties. The lower half is the menu control list box, where the menu/submenu structures for the current form are built up. The above Menu Design Window produces the menu shown below.



When an item is typed in the Caption text box, it also appears in the menu control list box. Selecting an item in the list box allows its properties to be edited (in a similar way to editing properties of controls in the properties window).

* An item that appears on the left margin on the list box is the menu title.
* Menu items are indented using the right arrow control. Move back to a previous level using the left arrow control.
* The up and down arrows are used to move among the menu entries.
* The insert button opens up space for an extra menu entry above the currently selected item in the list. The delete button removes the selected item.
* If submenus are required (further drop down menus off menu items) these are created by indenting twice (using the right arrow control). At run time, the presence of a submenu is indicated by this symbol to the right of the menu item it hangs off). Four levels of submenus are supported.
* The ampersand �&� is used to Precede the character that is to be underlined on the menu as the keyboard access character.
* If the Caption of a menu entry is set to a hyphen �-�, this causes a separator bar to appear in the menu. Use this device to divide menus into logical groups.
* The name of a menu entry is the name that will be used in the program code to refer to it.
* The shortcut list allows any menu entry to be assigned a shortcut (Ctrl+character) instead of, or as well, as an access key.
* If the property checked is set, this inserts a tick mark on the menu to indicate which of a set of options is currently in force.
* If the property Enabled is set to false, that item is gray (faded) on the menu at run time and does not respond to events.  
  At run time, mnuClose.Enabled = False would gray out the item Close on the first menu example.
* The property Visible is used to determine whether the menu item is visible or invisible at run time. If a menu item is invisible, the items below it move up to fill in the space. If a menu title is invisible, that menu does not appear on the menu and all the menu titles to the right of it are moved one place left.  
  At run time, **mnuSaveAs.Visible = False** would remove the item Save As... on the first menu example.

## ***Pop-up Menus***

A pop-up menu is one that is displayed on top of a form and is not attached to the menu bar.

**PopupMenu Method Example:**

The code below displays a pop-up menu at the cursor location when the user clicks the right mouse button over a form. To try this example, create a form with a menu named �mnuFile� (�mnuFile� must have at least one submenu). Copy the code into the Declarations section of the form then run the application.

**Private Sub Form\_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)  
 If Button = 2 Then PopupMenu mnuFile  
  
 End Sub**

In the above example, the pop-up menu appears at the X, Y position of the cursor, when the right mouse button is pressed (Button = 2). Use VB Help to explore the flags that can be set to specify the behaviour of a pop-up menu.

**Note:** All controls are displayed relative to the co-ordinates of the parent object, so setting X and Y both to 0 displays a pop-up menu at the top left corner of the form.

 [Menu](http://www.dcs.napier.ac.uk/hci/VB50/home.html)

*Tutorials using pull-down menus:*

** [Tutorial 5](http://www.dcs.napier.ac.uk/hci/VB50/HTML/tutorial_5.html) (Menus and List manipulation)

 [Tutorial 14](http://www.dcs.napier.ac.uk/hci/VB50/HTML/tutorial_14.html) (Multiple Document Interface)