# FaceIt Demo Source Code Notes

The following compiler-independent notes correspond to the "C#" comments in the "fDemoXY" source code.   Read these notes while examining a printed copy of the source.

C1. "fDemoXY" provides example code that corresponds to the topics discussed in the FaceIt Guide which is found in the "ViewIt On-Line Help" window.   Also see the "Startup" topics in the ViewIt help window for a description of basic resource set up and initialization tasks performed at the start of the program.

C2. The FaceStorXY file defines FaceIt and ViewIt command constants and the fRec shared record used by all FaceWare modules.   The FaceProcXY file defines the "FaceIt" dispatching procedure and other supporting procedures.   All communication with FaceWare modules is made through this FaceIt dispatching procedure using the command and variable names defined in "Stor" files such as FaceStorXY.

C3. A good way to deal with program-specific resources during development is to keep them in a separate res file (a "temporary resource file").   FaceWare facilitates use of such a file by allowing you to pass the name of this file in the fRec variable uName when calling DoInit.   In this program the name being passed is "fDemo.Rsrc" which the FaceIt dispatching procedure will attempt to open if it cannot find the necessary resources within the program file.   Note that no harm is done by leaving this assignment in your program source after combining the temporary resource file with the program file.

C4. Before making use of any FaceWare module, DoInit must be called.   DoInit is described in "Initializations" in the startup topics, and in the "Program Commands" topic under "Commands" in the ViewIt help window.   In this program we are asking that the cursor be changed to a deck of cards whenever control is returned to the main program (add 1 to a), and that control be returned to the program whenever the active window is changed (add 2 to a).   This explains why a = 3 = 1 + 2 is passed.

C5. The "Editor" and "Clipboard" modeless windows are opened by calling NewWnd with a = FWND ID (1010 and 1020, respectively), and b = 1 to indicate that these windows are to be modeless (vs. b = 0 = modal).   The "Editor" window contains a TextCt text-editing control, and the "Clipboard" window contains a ScrapCt clipboard display control.   See the "Windows" topic in the ViewIt Guide for further info about opening ViewIt windows.

C6. This "Beeps & Notes" modeless window is opened by calling NewWnd with a = 1030 = FWND ID, and b = 1 to indicate that the window is to be modeless.   This window, the floating palette, and the "fDemo" main menu are used to illustrate several different ways of presenting a user with the three "Beep" options.

C7. The "Beeps" window is opened by calling NewWnd with a = 1040 = FWND ID, and b = 2 to indicate that the window is to be a modeless floating window.   This window is also used to support the "Hier. Palette" menu item in the "fDemo" menu.   See "Floating Windows" and "Menu Windows" in the "Windows" topic of the ViewIt guide for more info about these special window types.

C8. The main program event loop is an infinite loop that responds to program-specific events not handled by other modules.   The first statement simply passes control to FaceIt by calling DoLoop.   FaceIt then keeps control until an event occurs that belongs to the main program.   See "The Main Loop" in the FaceIt Guide for further info.

C9. As described in "The Main Loop" topic, the main program receives messages from FaceIt and ViewIt in the form of menu or pseudo-menu events.   In this program a case (or if...then...else) block based on uMenuID and uMenuItem is used to process these events.

C10. The first item in the Apple menu (uMenuItem = 1, uMenuID = 101) causes the program to open an alert with the ShoStr utility command.   ShoStr and other string display commands make it easy to quickly display such messages in any Font, Size, Style, and Color.

C11. The "Save Positions" menu item in the File menu is a program item that causes the program to save window positions.   The current positions of the ViewIt windows based on FWND 1010 (Editor), 1020 (Clipboard), and 1040 (Beeps) are saved by calling GetWVC to update each FWND in memory, followed by SavWnd to save the FWND resource to disk.

C12. This program illustrates many different ways of presenting the same "beep" options to a user.   The first way shown is use of a menu or hierarchical menu.   uMenuIDs 105 and 106 correspond to such menus:
   105 = menu ID of main menu entitled "fDemo"
   106 = menu ID of menu attached to "Hierarchical" item
which both contain "Beep Once", "Beep Twice", and "Beep Thrice" as their first three menu items.
Another way of presenting such options is via buttons in a window, such as the "Beep Once", "Beep Twice", and "Beep Thrice" buttons at the top of this window, or the apples in the floating or popped-up palette:
   1030 = FWND ID of "Beeps & Notes" window
   1040 = FWND ID of "Beeps" floating window
where the buttons in both windows have been assigned ID numbers 1, 2, and 3, respectively, so that these values are returned in uMenuItem.
   Since all of the ways of choosing the beep options return the same values of uMenuItem for the number of beeps, the same code is used to do the beeping for both the menu selections and window hits.

C13. In addition to the beeping buttons in this window, the window also contains an enabled picture control that returns a message when hit:   uMenuID = 1030 = FWND ID, and uMenuItem = wiHit = 4 = control ID (set in ViewIt's Title or Control dialog).   In response to the hit message, the program uses the PopMen utility command to pop up the hierarchical menu (MENU 1007) that was previously auto-installed by FaceIt on DoInit.   It pops up the menu directly above the picture in the window by specifying the proper local coordinates when calling PopMen.   Since the menu popped up is a ViewIt menu window, selection of an item in this menu generates a hit message which is the same as that returned by clicking in the floating window:   uMenuID = 1040, uMenuItem = wiHit = 1, 2, or 3.
   A simpler way of popping up such a ViewIt menu window is to link it to a menu control (see the "Quick Help" menu in the ViewIt help window). In this case, however, the contents of the menu window are not displayed until the menu is popped up, which may or may not be what you desire. The combination of menu controls, menu windows, and use of the PopMen command provide a wide range of options to choose from for displaying palette-like menus.

C14. Item #8 (uMenuItem = 8) from the "fDemo" menu (uMenuID = 105) is used to illustrate topics discussed in "Background Processing" in the FaceIt Guide:   support for background processing and switching, and the monitoring of events without calling Get/WaitNextEvent.

C15. The utility command ShoAlt is used to open ALRT 1010 centered horizontally and vertically on the main screen (b = 0, c = 1, d = 1).   (A modal ViewIt window could also be used as an alert.)

C16. If "Cancel" was not chosen (uResult = mode > 1), then the program enters an endless loop that beeps once every three seconds.

C17. If demonstrating background processing/switching (mode = 2), then GetNextEvent is called each time thru the loop to check for any pending events that must be handled by FaceIt, and other events of interest.   If an auto-key event is found (fEvent.what = 5), then the loop is exited, otherwise the event is passed on to FaceIt to handle via DoEvnt.

C18. If demonstrating the monitoring of events without calling Get/WaitNextEvent (mode = 3), then the toolbox call "GetKeys" is used to check the status of the keyboard.   If, in this example, the bits corresponding to the Option and Command keys are both set, then the loop is exited.   When using "BitTst" to test bits, the following bit offsets apply:   48 = Command, 61 = Option, 62 = Caps Lock, 63 = Shift, and 40 = Period key.

C19. The next case is another example of responding to a pseudo-menu event. The event being processed is one that we asked to be informed of when calling DoInit:   a change in the identity of the active window.   This event returns with uMenuID = 1100 (the baseID of FaceIt module), and uMenuItem = 2 (to distinguish it from other pseudo menu events posted by FaceIt).   Knowing when the active window changes is useful when you have program menu items that are "context sensitive".   In this example we simply change the text of the last item in the "fDemo" menu to track the type of window that is active.   The active window type is easily identified by the baseID of the window-driving module (stored in fActiveID), and the ID of the FWND used to open the window (stored in fActiveResID).

C20. The last case is another example of responding to a pseudo-menu event.   The event being processed is posted when the user attempts to open or print files "from the Finder".   This event returns with,
   uMenuID = 1100 (the baseID of FaceIt)
   uMenuItem = 512
   uResult = 1 (open) or 2 (print)
   uString = file type
   uName = file name
   uI2 = working directory reference number
and the current directory set to that containing the file needing to be opened or printed.   To keep the example simple, this code only opens TEXT files from the Finder into a single, existing window with a TextCt text-editing control.   The command GetCtl is used to get the control's control handle which is then used when executing TextCt's OpnCTxt command to open the file.   Also note how the file name returned in uName is preserved across the GetCtl command since calls to the FaceIt procedure do not preserve "u" variables.   See "Finder Resources" in the FaceIt Guide for more info.