F2. Initializations

FaceIt-based programs always contain a few lines of code devoted to initialization followed by a "main event loop". The following program, for example, would automatically support access to Desk Accessories, interact properly with MultiFinder and System 7, and support several standard menu items (Pascal source):

```
 fRec.uName := 'Minimum.Rsrc';
 FaceIt(nil,DoInit,0,0,0,0);
 repeat
   FaceIt(nil,DoLoop,0,0,0,0);
 until false;
```

where the DoInit and DoLoop commands are defined in the "Commands" topic, and the "Minimum.Rsrc" file contains the clut, MENU, and STR# resources used by FaceIt, and the LoadIt module needed to call other FaceWare modules.

The simplicity of the above example program is one of the distinguishing characteristics of programming with FaceIt. Nearly all additional code that you add to a program will be program-specific, making a "code generator" unnecessary when programming with FaceIt.

The remainder of this topic discusses the initializations that are automatically done by FaceIt on DoInit, and the next topic discusses the main program event loop seen in all FaceIt-based programs.

Menu Initialization

Nearly all Mac programs make use of the main menu bar at the top of the main screen. The pull-down menus in this bar are always available unless they have been disabled by the program or a modal window has been opened. Menus in the main menu bar often contain hierarchical menu items attached to hierarchical menus, and windows often contain pop-up menus. FaceIt and ViewIt provide complete support for the initialization and management of all three of these common menu types.

• Auto-Loaded Menus

When DoInit is called, FaceIt looks for MENU resources in the program file or the program's temporary resource file (w/ file name = uName) and automatically initializes and installs these if they have been properly numbered. FaceIt searches for MENU resources with res IDs starting at 1001, 1002, ..., that have menu IDs of 101, 102, ... (menu ID = res ID - 900). MENUs continue to be loaded until 6 consecutive res IDs are encountered that do not have associated MENU resources (so you can leave gaps in your resource ID numbering for later additions).

MENU resources can be created and edited using Apple's ResEdit resource editor (included on the FaceWare Utilities disk if you purchased ViewIt). The "menu ID" of each MENU resource can be set by opening the "Edit Menu & MDEF ID" dialog in ResEdit. The following table reviews how menu ID numbers are used in FaceIt-based programs:

101-190	recommended for main program menus
191-195	reserved for temporary use by drivers
196-199	reserved for UtilIt's FSSC menus
236-255	reserved for use by ViewIt controls & DAs

where the range of menu IDs 101-190 would correspond to program MENU resources with res IDs 1001-1090.

What To Do: Most programmers begin programming with a copy of a demo resource file that already contains Apple, File, Edit, and Window menus (corresponding to resource IDs 1001-1004 and menu IDs 101-104). They then use ResEdit to modify these menus and/or to add new menus to support program-specific options. When adding a new menu, don't forget to set the menu ID to res ID - 900 or FaceIt will not auto-load the menu.

• Hierarchical Menus

Hierarchical menus can also be auto-loaded by FaceIt. The only difference between these menus and menu bar menus is that (1) the first character of their menu titles should be a "+" or "-" to inform FaceIt that the menu is not a menu bar menu (don't worry, this character is never displayed), and (2) there should be at least one menu item in another menu that is linked to this hierarchical menu. ResEdit can be used to establish the link by checking the "has Submenu" option when editing the hierarchical menu item and then entering the menu ID (not the res ID!) of the linked menu.

Hierarchical menus work best if they are numbered in the same, orderly manner as that described above for menu bar menus. For example, if MENU 1005 (menu ID 105) has 3 hierarchical menu items,

then the associated hierarchical MENUs might be numbered 1006, 1007, 1008 (menu IDs 106, 107, 108), with the next main menu bar MENU being numbered 1009 (menu ID 109).
ResEdit Bug: The "Open Submenu" option in ResEdit only works if the submenu's resource ID is equal to its menu ID, which will not be the case for menus auto-loaded by FaceIt.
• Pop-Up Menus
Pop-up menus are similar to hierarchical menus but are usually not linked to hierarchical menu items. Rather, they must be "popped up" by calling "PopUpMenuSelect" (or the UtilIt command PopMen). In most cases, however, pop-up menus used by FaceIt-based programs are those supported by menu controls in ViewIt windows, and these are both initialized and managed by control drivers without the need for the main program to call PopUpMenuSelect.
In cases where your program must initialize and manage the pop-up menu, FaceIt can be used to auto-load the menu by assigning it a res ID and menu ID in accordance with the scheme presented above, and by setting the first character in its title to "+" or "-" (as with hierarchical menus). If such a program-managed pop-up contains standard items, then FaceIt's DoMenu command should be used to process items selected from the menu.

Window Initialization

The presence of STR# 1000 in the program or program's temporary resource file causes FaceIt to attempt to open modeless windows defined by "auto-initialization" strings in this string list. These auto-initialization strings have the format "[baseID],[versID],[resID],..." where "baseID" refers to the resource ID number of the window-driving FaceWare module, "versID" is the module's version ID, and "resID" is the resource ID of a resource or set of resources that define the appearance and content of the window.
The auto-initialization string to open the modeless ViewIt window corresponding to FWND 1000, for example, would simply be "1200,20,1000", which is equivalent to calling NewWnd (a ViewIt version 2.0 command) to open a new modeless window using FWND 1000.
What To Do: If you have started with a program resource file containing auto-initialization strings that are opening windows that you do not want, use ResEdit to delete these strings from STR# 1000. (Click on the row of asterisks above the string in the STR# resource and choose "Clear" from ResEdit's "Edit" menu.) If you would like to have other modeless windows auto-initialized by FaceIt, add new strings to STR# 1000.

Palette Initialization

The scheme used by Apple to designate colors on a Mac that supports Color QuickDraw allows you to choose from any one of 16 million colors. Unfortunately, you are not always able to display this many colors, and which of these can be displayed is determined by the "depth" of your screen and the current color "palette". The screen depths supported by Color QuickDraw are 1, 2, 4, 8, 16, and 32 bits per pixel, corresponding to 2, 4, 16, 256, 32768, and 16 million displayable colors, respectively.
When DoInit is called, and Color QuickDraw is supported, FaceIt looks for "clut" resource 1000 and uses this color table to reset the program-wide color palette. This color palette determines which colors will be displayed by the program from the 16 million possible red, green, and blue combinations. If the screen depth supports fewer colors than are in the palette, then only the first colors from the palette will be available.
The default clut 1000, for example, contains 16 colors: white, black, light gray, dark gray, plus 12 other common colors (in that order). This produces a reasonable set of colors at all screen depths:
1 bit/pixel --> white & black displayed
2 bits/pixel --> white, black, light gray, dark gray
4 bits/pixel --> all 16 colors from clut 1000
> 4 bits/pixel --> all 16 colors plus others
You may, on the other hand, have a need to work with 256 shades of gray. In this case a clut resource consisting of 256 shades of gray (including white and black) could be used in place of the default clut 1000, and this would force the display of 256 shades of gray on 8-bit deep monitors.
"clut" resources can be created and edited with ResEdit. A UtilIt command, SetPal2, is also available for switching the current program-wide palette from within programs. Experienced programmers can also obtain a handle to the current program-wide palette by calling "GetPalette" with an argument of -1 (i.e., "GetPalette(WindowPtr(-1))"). This handle can then be used with other Color QuickDraw calls to

directly manipulate the contents of the palette.

What To Do: The only programmers who need worry about clut 1000 are those whose programs need to define a large fraction of the displayable colors (such as 256 shades of gray on an 8-bit deep monitor).

TECHNICAL NOTE: Although each window in a program can have its own color palette, we recommend using the single program-wide palette for the following reasons: (1) it minimizes the amount of window updating that occurs when different windows in the same program are brought to the front, (2) it solves the problem of floating windows which would otherwise have to be assigned the palette of the active window, and (3) it greatly simplifies the manipulation and switching of color palettes (you only have one to deal with).