11. Summary

"What steps should I take to start using ViewIt and FaceIt in a new or existing programming project?"

New Programs

First, create a new ".Rsrc" resource file to store program resources during development. A good place to get such a file is to copy and rename the "Minimum.Rsrc" file since this file contains LoadIt (required by FaceWare modules) plus other program resources (such as default MENUs).

Second, construct a small program that does little more than call DoInit and DoLoop. A good example of such a program is the "MinimumXY" demo which you can copy and rename. Also change the "Minimum.Rsrc" file name found within it to the name of your resource file.

Third, try compiling and running your new program. The "Recompiling Demos" topic in "Getting Started" describes some of the problems you might encounter.

Fourth, if you would like to add program-controlled main menu items, then use ResEdit or other resource editor to add such items to the MENU resources in your resource file. The "Menu Handling" topic in the FaceIt Guide describes the types of items that can be added to these menus, and the "Initializations" topic in the FaceIt Guide describes the rules to follow when adding new MENU resources.

Fifth, begin adding code below DoLoop in your main event loop that handles events and messages returned by DoLoop. If, for example, you added a program menu item "Run" as the 4th item in the MENU resource with menuID 105, then a case or if...then statement should be added to handle the corresponding menu event:

```
repeat
  FaceIt(nil,DoLoop,0,0,0,0);
  if (uMenuID = 105) and (uMenuItem = 4) then
    ...
```

Note that all code appearing in your main event loop is added to respond to events or messages that you have chosen to support. This differs greatly from most other interface builders that appear to make getting started easy, but then require you to wade through large amounts of code to figure out how to add program-specific functionality.

Once you have a simple program up and running, begin adding ViewIt modal or modeless windows along with corresponding code to handle events from these windows. Modeless ViewIt windows are often opened as part of the initialization code in a program:

```
FaceIt(nil,DoInit,0,0,0,0);
...
FaceIt(nil,NewWnd,1001,1,0,0);
...
repeat
  FaceIt(nil,DoLoop,0,0,0,0);
  ...
```

Each ViewIt window requires an FWND resource, but ViewIt creates one for you if it cannot find one with the resource ID passed to NewWnd (described in "Adding Windows" in the "Windows" topic of the ViewIt Guide). Thus you can write and execute code to open a ViewIt window before the FWND is created, and then enter edit mode and edit the window from within the running program.

Processing events from modeless windows is similar to processing menu events, but uMenuID returns the FWND ID and wcHit the number of the item hit (see "Windows" in ViewIt Guide for a complete description of window events):

```
repeat
  FaceIt(nil,DoLoop,0,0,0,0);
  if (uMenuID = 1001) and (wcHit = 6) then
    ...
```

which would respond to a hit in the sixth control of the window associated with FWND 1001.

Modal windows are opened, responded to, and closed within isolated sections of your program code. Most of the "vDemo" example program is devoted to illustrating the opening and managing of modal windows, so take the time to study this program in detail. The "Windows" topic in the ViewIt Guide describes the types of events and messages returned from modal windows.

ViewIt windows contain views and controls. These objects can be copied from other windows, or imported via the "+" import menu when in edit mode. The "Editing" topic in the ViewIt Guide discusses on-line editing of these objects, the "Views" topic discusses views, and "Controls" discusses control

features and how to learn more about the various custom controls supported by ViewIt control drivers.

Moving information to and from ViewIt windows usually involves transferring data between program variables and control values. ViewIt makes this task easy by supporting "data linking" which is discussed in the "Data Links" topic of the ViewIt Guide.

When finished with an application, you may want to create a stand-alone version that does not require presence of the FaceWare file. This issue and other resource management issues are discussed in "Resources" in the ViewIt Guide. When adding resources to a finished program, you'll also want to add several Finder-related resources that affect the program's interaction with the Finder. "Finder Resources" in the FaceIt Guide discusses adding such resources.

Existing Programs

To adapt an existing program, incorporate the required elements that have been discussed in previous topics:

1. Include the "FaceStorXY" and "FaceProcXY" files in your programming project (where "XY" denotes a compiler, as described in the "About Compilers" program). Examine one of our demo programs to determine the manner in which this is done for the compiler you are using. Some modules require additional include files.

2. Add the LoadIt module to your program or other resource file used by your program. If you will not be using FaceIt, then use MoveIt to move LoadIt (found on the Utilities disk and in demo ".Rsrc" files) into your own res file. If using FaceIt, then simply copy and rename one of our demo res files, or copy and rename the "Minimum.Rsrc" res file.

3. Add a call to FaceWare initialization command DoInit near the beginning of your main program code. If using FaceIt (parameter $c \geq 0$ in call to DoInit), then DoInit will auto-load main program menus and perform other tasks described in the FaceIt Guide. If not using FaceIt ($c < 0$ or FaceSt is being used), then DoInit's actions are "silent" and will not affect the appearance of your program.

4. If using FaceIt, add a "main event loop" to your program that begins with a call to DoLoop followed by an if...else or case block that processes the messages and events returned.