C1. FaceIt & ViewIt Commands

As described in the "Startup" topics, communication between a program and FaceWare modules is accomplished via the global fRec record and the "FaceIt" dispatching procedure. The fRec record is described in the "fRec Record" topic, and commands supported by FaceIt and ViewIt that can be passed to the FaceIt dipatching procedure are described in the other topics under the "Commands" menu.

Command Format

The FaceIt dispatching procedure is found in the "FaceProcXY" file (or in some other form compatible with your programming environment). Calls to this procedure always have the form,

```
FaceIt(nil,[command],a,b,c,d);     Pascal
FaceIt(0,[command],a,b,c,d);       /* C, C++ */
call FaceIt(0,[command],a,b,c,d) !FORTRAN
```

where parameters a, b, c, and d are 4-byte integers, and where other record elements from the global fRec or other records may be used with particular commands.

Each FaceIt and ViewIt command name, its equivalent number, the parameters and record elements used by the command, and a complete description of the command are presented in the topics under the "Commands" menu. Command names can be used in place of the numbers in calls to the FaceIt procedure since they are declared as constants in the "FaceStorXY" file (or in some other way compatible with your programming environment).

Command Types

The commands described under the "Commands" menu are of 3 general types:

1. "Program Commands" are commands supported by the FaceIt module that deal with program-wide features (the main loop, the main menu bar, modeless window management, etc.).
2. "Window" and "Control Commands" are commands supported by the ViewIt module that deal directly with ViewIt windows and controls.
3. The "Utilities" commands are also supported by ViewIt, but provide a wide range of utility-type routines that augment the Macintosh toolbox. These utility routines are used internally by ViewIt and control drivers, but can also be called by your program for its own purposes.

Clobbered Variables

Most fRec variables are scratch variables that can be changed by any call to the FaceIt dispatching procedure or to the Control Manager (since the Control Manager then calls ViewIt via CDEF 1200). The scratch variables include all those prefixed by the letters "u", "w", or "c". This means that you cannot rely on the content of these fRec variables across calls to FaceIt or the Control Manager. The rule of thumb to follow is that if you will be using such an fRec variable in more than one FaceIt or Control Manager call, then save a copy of its contents in a local variable and use the copy. The major exception to this rule is that most utility-type routines preserve the content of the "w" and "c" variables since they do not deal directly with ViewIt windows.

Suppose, for example, that a control handle is needed for use in a single call to "SizeControl". In this case you can get away with using the cControl returned by GetCtl:

```
FaceIt(nil,GetCtl,1030,0,1,5);
SizeControl(fRec.cControl,100,100);
```

But if the control handle is also used in a subsequent call, then a copy of cControl should be used:

```
FaceIt(nil,GetCtl,1030,0,1,5);
theControl := fRec.cControl;
SizeControl(theControl,100,100);
FaceIt(nil,AddCtl,1004,0,ord(theControl),0);
```

since both the Control Manager and FaceIt calls can result in clobbering the current values in fRec.

The same precaution should be taken with commonly used variables such as uString, uMenuID, uMenuItem, wvHit, wcHit, and wiHit, although these are most often used as simple case selectors in case blocks in a way that does not require their contents to be saved. The following code, for example, is safe since it makes just one use of uMenuID between calls to DoLoop:

```
repeat
  FaceIt(nil,DoLoop,0,0,0,0);
```

```
  if (uMenuID = 1001) then
    ...
  else if (uMenuID = 1002) then
    ...
  else if (uMenuID = 1003) then
    ...
until false;
```

but the similar code that follows is asking for trouble since it assumes that the actions taken after each "if...then" do not affect the contents of uMenuID:

```
repeat
  FaceIt(nil,DoLoop,0,0,0,0);
  if (uMenuID = 1001) then
    ...
  if (uMenuID = 1002) then
    ...
  if (uMenuID = 1003) then
    ...
until false;
```

This code could also be fixed by simply storing the contents of uMenuID in a local variable and using that variable in the "if...then" blocks.