## F3. The Main Loop

All FaceIt-based programs contain a main event loop that consists of a simple loop around a call to DoLoop:

```
repeat
  FaceIt(nil,DoLoop,0,0,0,0);
  [respond here to messages]
until false;
```

DoLoop handles "low level" events returned by the toolbox call "WaitNextEvent". FaceIt preprocesses these low-level events and only returns control to the program (returns from DoLoop) when an event occurs that must be handled by the program. Control is returned with a message in the form of a menu or pseudo-menu event (described below).

This approach dramatically reduces the complexity of main event loops in FaceIt-based programs without robbing the program of its control over the main loop. This differs from competing "program generators" that generate large amounts of repetitious, complex code, and from high-level programming environments that completely hide the main loop from programmers.

## Message Types

The "messages" returned by FaceIt can be classified on the basis of the value of the fRec variable uMenuID, where uMenuID can be either,

- zero (for unprocessed System 7 Apple Events)
- a menu ID (for menu selections)
- an FWND ID (for hits in ViewIt windows)
- a module's baseID (for module-specific messages)

Menu IDs will typically be in the range of 101 to 190 for program menus, FWND IDs in the range of 1000 to 1099 for program windows, and baseIDs in the range of 1100 to 7499 for FaceWare modules.

Thus a typical FaceIt-based main event loop contains a case (or if...else) block based on the value of uMenuID:

```
repeat
  FaceIt(nil,DoLoop,0,0,0,0);
  case uMenuID of
    104:
      [respond to menu selection (menu ID 104)]
    1001:
      [respond to window hit (FWND 1001)]
    1100:
      [respond to module message (FCMD 1100)]
    otherwise
  end;
until false;
```

The message is then further differentiated by the value of uMenuItem and other fRec variables, where uMenuItem is typically a menu item number, window item number, or module-specific message number:

```
repeat
  FaceIt(nil,DoLoop,0,0,0,0);
  case uMenuID of
    104:
      if (uMenuItem = 2) then
        [second item in menu selected]
    1001:
      if (uMenuItem = 4) then
        [fourth item in window hit]
    1100:
      if (uMenuItem = 2) then
        [active window changed - a FaceIt message]
    otherwise
  end;
until false;
```

Thus we often refer to the messages returned by FaceIt as "menu" or "pseudo-menu" events since they are always differentiated on the basis of uMenuID and uMenuItem.

Other fRec variables such as uString, uResult, wiHit, wvHit, wcHit, wClick, wEvent, and fEvent are often used to return additional info with messages. Which fRec variables are used with each message will be documented with the message. Menu events, for example, are described in the "Menu Handling" topic, window events in "Windows" topic in ViewIt Guide, and FaceIt-specific messages in the description of the DoInit command in "Commands" topic.

What To Do: If incorporating support for FaceIt into an existing program, add the few lines of initialization code required and a simple DoLoop-based loop like that shown above and in the demo programs. Then, as you add program menu items and windows, add new cases to the loop to handle any messages returned by these.

## Module Messages

FaceWare modules often support special, module-specific messages. As mentioned above, these messages are denoted by the fact that uMenuID returns with the baseID of the module (1100 for FaceIt, 1200 for ViewIt, etc.). FaceIt itself supports several optional messages that can be turned on or off according to the value of parameter a when calling DoInit (see "Commands" topic).

The most commonly used FaceIt message, for example, is the one that returns control to the program when the active window identity changes (i.e., when a new window becomes the active window). The message is enabled by adding 2 to a when calling DoInit, and results in returning uMenuID = 1100 and uMenuItem = 2 after an active window change:

```
 ...
 FaceIt(nil,DoInit,2,0,0,0);
 ...
 repeat
   FaceIt(nil,DoLoop,0,0,0,0);
   ...
   else if (uMenuID = 1100) then
     if (uMenuItem = 2) then
       [active window changed]
   ...
 until false;
```

Which window has become active can then be determined by examing the fActive... variables in fRec. This is useful in cases where the state of program menu or window items must be adjusted to correspond to the active window.

## Apple Events

With the introduction of System 7, Apple added a new type of event, the "Apple Event", which can be used to pass info between and within programs. To receive Apple events, the program must be running under System ≥7 and have a SIZE resource with its "High level event aware" option set.

FaceIt provides support for the four "required" Apple events (see "Finder Resources"). All other Apple events are returned to the program from DoLoop with uMenuID = 0 and the event in fEvent (fEvent.what = 23). You can then respond to these events in a program-specific manner (see Inside Mac V6 for details).

## Foreign Windows

When using FaceIt, any non-FaceWare windows that cause update events to occur (have non-empty update regions) are automatically hidden by FaceIt. This is done to prevent such windows from "flooding" the event loop with update events that are never processed properly. If you need to mix your own windows with FaceWare windows, then either use the FaceSt module in place of FaceIt (see the "Hybrid Programs" topic), or use ViewIt windows with controls that are driven by main program code via override procedures.